# ArchCaMO - A Maturity Model for Software Architecture Description Based on ISO/IEC/IEEE 42010:2011

Ademir A. C. Júnior[1(✉)], Sanjay Misra[2], and Michel S. Soares[1]

[1] Federal University of Sergipe, Av. Marechal Rondon, s/n - Jardim Rosa Elze, São Cristóvão, Brazil
{ademiralcj,michel}@dcomp.ufs.br
[2] Convenant University, KM 10 Idiroko Rd, Ota, Nigeria
sanjay.misra@covenantuniversity.edu.ng

**Abstract.** Academia and Industry have acknowledged that having a fully described software architecture is a crucial asset for software development and maintenance. The description of a software architecture is read by many stakeholders when developing and maintaining complex software systems which are composed by multiple elements, including software, systems, hardware and processes. 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. Therefore, it is not unusual that software architects have to deal with difficulties in terms of how to describe the architecture. This paper brings two main purposes: first is the idea of establishing levels of a maturity model for software architectures, which can help in organizing, describing and communicating the software architecture for multiple stakeholders, and, second, a way to evaluate how mature is the software architecture.

**Keywords:** Software architecture · Maturity models · ISO/IEC/IEEE 42010

## 1 Introduction

Software architecture has been defined in multiple ways since the 1960's. From the first ideas of structuring and decomposing complex systems into more manageable modules [1–3], to modern definitions with focus on cooperating components, decision making [4] and knowledge management (AKM) [5].

Software architecture has been considered by many researchers and practitioners a fundamental asset in software development and maintenance [6–8]. For instance, the authors of a book on documenting software architecture [9] express

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11623, pp. 31–42, 2019.
https://doi.org/10.1007/978-3-030-24308-1_3

that without an appropriate software architecture for the problem to be solved by a software, the project will fail. Other authors also express the difficulties brought by not using correctly a software architecture to handle the complexity of software-intensive systems [10], 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 [11].

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 [8,12, 13]. Creating a reliable, 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 in such a way that it is understandable, and with information that is easy to retrieve.

A case of technical debt occurs when the architectural documentation is insufficient, incomplete, or outdated. One possible reason is due to the contest between agilists and more process-oriented personnel [11,14]. The assumption in this paper 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, are still open issues in industry. It is frequent that industry practitioners, and even researchers, do not even know how to start to describe the architecture [8], do not know how much architecture to use [15,16], have issues regarding agility in processes and what is named Big Design Up-Front (BDUF) architecture [17], or even which architectural elements are necessary to describe in the software architecture documents [18–20].

One possible solution for describing a software architecture is to rely on models and standards such as Kruchten's 4+1 Model [21] or ISO/IEC/IEEE 42010:2011 [22]. 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.

The proposal in this paper is to describe a model named ArchCaMo to access the level of architectural maturity using ISO/IEC/IEEE 42010:2011 as context. 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. In addition, 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 exactly what to expect in terms of activities and deliverables.

Although other maturity models for software architectures were proposed, as described in the next section, to the best of our knowledge, no other models were introduced with focus on ISO/IEC/IEEE 42010:2011.

## 2 Related Works

Many maturity models were proposed in past decades in order to access 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 and for a variety of domains.

SW-CMM [23], and later its evolution, Staged CMMI [24], propose evaluation in 5 levels, in which each level includes a number of 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 [25], authors contribute towards 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 [26] 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) [27]. 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, briefly introduced in the next section. Therefore, 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.

ISO/IEC/IEEE 42010:2011 [22] addresses the creation, analysis and sustainment of architectures of systems through the use of architecture descriptions. This standard proposes a conceptual model of architecture description, including concepts such as views, models, concerns, rationale, viewpoints, frameworks and architecture description languages. The standard provides a core ontology for description of software architectures.

Several terms which have some relation to software architecture are defined in the standard. According to the standard *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. An *Architecture* is the fundamental concept or properties of a

system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.

ISO/IEC/IEEE 42010:2011 does not specify any format for recording architecture descriptions, but it describes the basic context of an architecture description. This 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 their own 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, as 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 use the architecture description as a guide to operational and infrastructure support and configuration management. Managers use the architecture description as a support to software planning activities, such as establishing the schedule, budge, and the team.

A complete description on ISO/IEC/IEEE 42010:2011 can be find in the official standard document [22], and its practical use has been explored by many researchers in past years, for instance in health systems [28,29], industry [30,31], transportation systems [32], business [33], defense [34] and energy [35].

## 3 ArchCaMo - Architectural Capability Model

Instead of establishing in a 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 own processes for each given practice to be executed at each level.

### 3.1 How the Levels Are Proposed

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

First, almost all the levels are linked to a topic of the standard. Therefore, we did not change the standard, but we have organized it in such a way that the architecture team can document the software/system's architecture in a sequential manner.

The second reference for ArchCaMo's levels is the SMS. We found initially 128 papers published between 2007 and 2018. After the selection phase, 19 studies expressed that they have used ISO/IEC/IEEE 42010:2011 on their architecture description, thus we identified what aspects/points of the standard were most or least used in these studies. As a result, according to this classification and our knowledge/experience about the standard, we proposed that the most used features of ISO/IEC/IEEE 42010:2011 would be considered in the initial levels and so on.

### 3.2 ArchCaMo Levels

The ArchCaMo model is defined in 5 levels. The initial level, Level 1, refers to all companies/organizations that do not have a formalized way to define the software architecture, so they may document some aspects that they need or think that is necessary for the project. This level is classified as unstable from the architectural point of view.

**Levels 2 and 3.** These levels are made up of most relevant aspects that are necessary to describe a software architecture. 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 1.** 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 1 describes the Minimum Architecture, which is the name of Level 2. This table is based on Context of architecture description and Conceptual model of an architecture description of ISO/IEC/IEEE 42010:2011.

Table 2 describes the Defined Architecture, and it is also based on Conceptual model of an architecture description, and Conceptual model of AD elements and correspondences of ISO/IEC/IEEE 42010:2011.

**Table 2.** 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 |

**Levels 4 and 5.** The 2 upper levels represented in Tables 3 (Consistent Architecture) and 4 (Quantified and Improved Architecture) 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 3 describes the Consistent Architecture, based on Conceptual model of an architecture framework, and Conceptual model of an architecture description language of ISO/IEC/IEEE 42010:2011.

**Table 3.** 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 |

Table 4 describes the Consistent Architecture, based on Conceptual model of an architecture framework, and Conceptual model of an architecture description language of ISO/IEC/IEEE 42010:2011.

**Table 4.** 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[a] |

[a] 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") [22]

## 4 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, we suppose that not all organizations have a defined process for defining, 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.

### 4.1 Level 2 - Minimum Architecture

- **P2.1** - Environment is identified. There are many environments, not only the place where the team is coding, but also, for instance, 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.
- **P2.2** - System-of-Interest is identified. System in which life cycle is under consideration [36]. 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 [22].
- **P2.4** - Stakeholders are identified. Each stakeholder is defined and ranked for their relative importance for that specific software product. In consonance with [22], 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 42010:2011 documentation, one can find some concerns that should be identified when applicable.
- **P2.6** - Viewpoints are defined. Through interpretation from architectures views it is established the viewpoints which will frame specific established 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 modelling conventions established by the architecture team, appropriate to the concerns to be addressed. Models are used to answer questions about the system-of-interest.

### 4.2 Level 3 - Defined Architecture

- **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 Level 4 - Improved Architecture

- **P4.1** - Known inconsistencies are recorded. An architecture description should record the inconsistencies, and include an analysis of consistency of its architecture models and its views [22].
- **P4.2** - Use of Architecture Frameworks. If the project is using an Architecture Framework, it should include conditions of applicability [22].
- **P4.3** - Use of Architecture Description Languages. The architecture description language should support all the aspects of the system-of-interest [22].

### 4.4 Level 5 - Quantified and Evaluated Architecture

- **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 to get information of the 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 Application of ArchCaMo in a Research Paper

This section describes an application of ArchCaMo in a software architecture proposed in an academic research paper. The paper [37] is a work in progress towards the specification of a conceptual architecture of a smart system, for supporting the management of disruptions in the manufacturing domain. Moreover, the work describes system architecture based on a number of interrelated viewpoints according to ISO/IEC/IEEE 42010:2011.

**Table 5.** ArchCaMO application level 2 in an example

| KAI | Description | Reference in the standard | Identification in the paper [37] |
|---|---|---|---|
| 2.1 | Environment is identified | Item 4.2 | Manufacturing Domain |
| 2.2 | System-of-Interest is identified | Item 5.2 | Smart system for supporting the management of disruptions |
| 2.3 | Supplementary information is identified | Item 5.2 | Context of the EU-funded H2020 DISRUPT |
| 2.4 | Stakeholders are identified | Item 5.3 | System users (domain experts, operational managers and decision makers), system administrators, and technology providers (including the suppliers, the developers/integrators, the testers and the maintainers of the system) |
| 2.5 | Concerns for each Stakeholder are identified | Item 5.3 | The purpose of the system is to support the disruption management lifecycle; The system will be added onto an existing factory ecosystem; The system will be based on a distributed architecture, consisting of loose coupling of (potentially existing) functional components, to enable flexibility in the system components and their functionality. This should reduce the risk for a vendor lock-in |
| 2.6 | Viewpoints are defined | Item 5.4 | Describes the system main components, their functionality and interfaces; Describes the environment into which the system will be deployed, including dependencies it has on the environment, and the mapping of system components to the environment |
| 2.7 | Multiple Views are defined | Item 5.5 | Logical view, Informational View and Physical View |
| 2.8 | Models are defined | Item 5.6 | UML (Class, Component and Deployment) |

In this example, displayed in Table 5, we read the research paper looking for any characteristic of the architecture description that follows the maturity model. After all analysis, we found that the description of the system architecture fits in Level 2, the Minimum Architecture. This conclusion may be because this is a short paper, or even it is a work in progress.

The authors did not find evidences for any one of the processes of Level 3, Level 4 or Level 5. Even if one of the KAI of any upper Level is fulfilled, the architecture would still be classified at Level 2.

This paper was chosen among 19 relevant papers found in a Systematic Mapping Study from another research. Even though this is a short paper, this paper was the one with more features following the standard ISO/IEC/IEEE 42010:2011.

## 6 Conclusion

In this paper, a maturity model for Software Architecture based on the standard ISO/IEC/IEEE 42010:2011, named ArchCaMO, is presented. First, notions about software models and software architecture models are briefly introduced. Then, basic notions of the standard ISO/IEC/IEEE 42010:2011 are presented.

The 5 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 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 the levels description, how the levels work through the process is described. To simplify these processes, the authors illustrated with an example, a software architecture published in [37] taken from the literature, showing each aspect of ISO/IEC/IEEE 42010:2011 that they have adopted.

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.

As for future works, our proposal is to evaluate other software architectures in the literature and in companies according to ArchCaMo.

**Acknowledgment.** This study was financed in part by the Fundação de Apoio a Pesquisa e Inovação Tecnológica do Estado de Sergipe.

## References

1. Dijkstra, E.W.: The structure of THE-multiprogramming system. Commun. ACM **26**(1), 49–52 (1968)
2. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM **15**(12), 1053–1058 (1972)
3. Parnas, D.L.: A technique for software module specification with examples. Commun. ACM **15**(5), 330–336 (1972)
4. Tofan, D., Galster, M., Avgeriou, P., Schuitema, W.: Past and future of software architectural decisions - a systematic mapping study. Inf. Softw. Technol. **56**(8), 850–872 (2014)

5. Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A.: 10 years of software architecture knowledge management: practice and future. J. Syst. Softw. **116**, 191–205 (2016)
6. Garlan, D.: Software architecture: a roadmap. In: Proceedings of the Conference on The Future of Software Engineering, ICSE 2000, pp. 91–101 (2000)
7. Kruchten, P., Obbink, H., Stafford, J.: The past, present, and future for software architecture. IEEE Softw. **23**(2), 22–30 (2006)
8. Garlan, D.: Software architecture: a travelogue. Proc. Future Softw. Eng. FOSE **2014**, 29–39 (2014)
9. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 3rd edn. Addison-Wesley Professional (2012)
10. Oquendo, F.: Software architecture challenges and emerging research in software-intensive systems-of-systems. In: Tekinerdogan, B., Zdun, U., Babar, A. (eds.) ECSA 2016. LNCS, vol. 9839, pp. 3–21. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48992-6_1
11. Booch, G.: The economics of architecture-first. IEEE Softw. **24**(5), 18–20 (2007)
12. Buchgeher, G., Weinreich, R., Kriechbaum, T.: Making the case for centralized software architecture management. In: Winkler, D., Biffl, S., Bergsmann, J. (eds.) SWQD 2016. LNBIP, vol. 238, pp. 109–121. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27033-3_8
13. Galster, M., Tamburri, D.A., Kazman, R.: Towards understanding the social and organizational dimensions of software architecting. SIGSOFT Softw. Eng. Notes **42**(3), 24–25 (2017)
14. Yang, C., Liang, P., Avgeriou, P.: A systematic mapping study on the combination of software architecture and agile development. J. Syst. Softw. **111**, 157–184 (2016)
15. 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, PLoP 2015, pp. 9:1–9:18 (2015)
16. 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, ICSE 2015, vol. 1, pp. 347–357 (2015)
17. Abrahamsson, P., Babar, M.A., Kruchten, P.: Agility and architecture: can they coexist? IEEE Softw. **27**, 16–22 (2010)
18. Ding, W., Liang, P., Tang, A., Vliet, H.V., Shahin, M.: How do open source communities document software architecture: an exploratory survey. In: 2014 19th International Conference on Engineering of Complex Computer Systems, pp. 136–145 (2014)
19. Graaf, K.A., Liang, P., Tang, A., Van Vliet, H.: How organisation of architecture documentation affects architectural knowledge retrieval. Sci. Comput. Program. **121**, 75–99 (2016)
20. Díaz-Pace, J.A., Villavicencio, C., Schiaffino, S., Nicoletti, M., Vázquez, H.: Producing just enough documentation: an optimization approach applied to the software architecture domain. J. Data Seman. **5**(1), 37–53 (2016)
21. Kruchten, P.B.: The 4+1 view model of architecture. IEEE Softw. **12**(6), 42–50 (1995)
22. 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), pp. 1–46 (2011)
23. Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B.: Capability maturity model for software (version 1.1). Technical report CMU/SEI-93-TR-024 ESC-TR-93-177, Software Engineering Institute, Pittsburgh, PA (1993)

24. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI for Development: Guidelines for Process Integration and Product Improvement, 3rd edn. Addison-Wesley Professional (2011)
25. Ahmed, F., Capretz, L.F.: An architecture process maturity model of software product line engineering. Innov. Syst. Softw. Eng. **7**(3), 191–207 (2011)
26. Ostadzadeh, S.S., Shams, F.: Towards a software architecture maturity model for improving ultra-large-scale systems interoperability. CoRR, abs/1401.5752 (2014)
27. Meyer, M., Helfert, M., O'Brien, C.: An analysis of enterprise architecture maturity frameworks. In: Grabis, J., Kirikova, M. (eds.) BIR 2011. LNBIP, vol. 90, pp. 167–177. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24511-4_13
28. França, J.M.S., de Lima, J.S., Soares, M.S.: Development of an electronic health record application using a multiple view service oriented architecture. In: Proceedings of the 19th International Conference on Enterprise Information Systems, ICEIS 2017, Porto, Portugal, 26–29 April 2017, vol. 2, pp. 308–315 (2017)
29. Crichton, R., Moodley, D., Pillay, A., Gakuba, R., Seebregts, C.J.: An architecture and reference implementation of an open health information mediator: enabling interoperability in the rwandan health information exchange. In: Weber, J., Perseil, I. (eds.) FHIES 2012. LNCS, vol. 7789, pp. 87–104. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39088-3_6
30. Musil, J., Musil, A., Weyns, D., Biffl, S.: An Architecture framework for collective intelligence systems. In: 2015 12th Working IEEE/IFIP Conference on Software Architecture, pp. 21–30 (2015)
31. Van Heesch, U., Avgeriou, P., Hilliard, R.: A documentation framework for architecture decisions. J. Syst. Softw. **85**(4), 795–820 (2012)
32. Karkhanis, P., Brand, M.G., Rajkarnikar, S.: Defining the C-ITS reference architecture. In: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), vol. 00, pp. 148–151 (2018)
33. Vidoni, M., Vecchietti, A.: Towards a reference architecture for advanced planning systems. In: Hammoudi, S., Maciaszek, L., Missikoff, M.M., Camp, O., Cordeiro, J. (eds.) Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS), vol. 1, pp. 433–440 (2016)
34. Williams, J.L., Stracener, J.T.: First steps in the development of a Program Organizational Architectural Framework (POAF). Syst. Eng. **16**(1), 45–70 (2013)
35. Effenberger, F., Hilbert, A.: Towards an energy information system architecture description for industrial manufacturers: decomposition & allocation view. Energy **112**, 599–605 (2016)
36. ISO/IEC/IEEE: Systems and software engineering-System life cycle processes (2008)
37. Kavakli, E., Buenabad-Chvez, J., Tountopoulos, V., Loucopoulos, P., Sakellariou, R.: WiP: an architecture for disruption management in smart manufacturing. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 279–281 (2018)