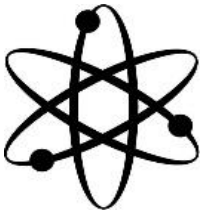

Atom Free IT
for true business agility

Good Architecture

v14 1-9-2023
robert.deckers@atomfreeit.com

Abstract

To establish, manage, and convey a good architecture, the architect chooses how much attention to pay to certain stakeholders, concerns, models and other architectural aspects. To make meaningful choices, the architect must first determine what a good architecture is in this specific situation. If the architect has a clear picture of when the architecture is good for a specific system and its environment, it becomes easier to achieve a good architecture.

A good architecture has the following characteristics:

- Correct: the architecture is based on adequate information about the environment, including the prioritized interests of stakeholders. Interests can be both goals and constraints. The architecture provides the right balance between the interests. The main question in determining correctness of an architecture is "Does the system fit in its environment?".
- Consistent: The architecture statements do not contradict and form a unified whole. This applies not only to the statements within the architecture itself, but also in relation to statements about the environment. In addition, the architecture must be realizable in the system and provide a solution for all important requirements. The main question regarding consistency is: "Is the system well-engineered?".
- Communicated: the stakeholders are aware of their relationship with the architecture. They should have the right expectations of the architecture and know what their own contribution is. Stakeholders must understand how their requirements are secured in the architecture. A communicated architecture begins with a communicable architecture description. The main question to determine if an architecture is communicated is "Does everyone know what they need to know?".

To check if an architecture is good and to give direction to the architectural activities, the architect can focus on answering a given set of questions related to correctness, consistency, and communication. For the business manager, there is a short list of questions available. The architect must be able to answer each question or should be able to explain why an answer cannot be given. The questions and the answers help to identify improvement areas for the architecture.

Contents

1 Introduction

During a discussion with software architects about education in software architecture the question arose what a good architect is. The answers of the architects stated mostly what skills an architect should have. A good architect should be able to analyze, abstract, structure, communicate, mediate, keep an overview, know the application domain, and understand how to use required technology. But, how do you know if an architect has these skills? "It should be observable in the architecture" was the reply. "A good architect realizes a good architecture." That makes sense. But what is a good architecture?

To establish, manage and advocate a good architecture, the architect makes choices regarding stakeholder concerns, models, views, and other architectural aspects. Several approaches (including (Bass, Clements, & Kazamn, 2003), (Rozanski & Woods, 2005), (Bosch, 2000), (Muller, 2004)) offer techniques to make those choices. However, those techniques are not the same and it is not always clear which technique is the most suitable in a specific situation. Does the use of a particular technique lead to a good architecture? What risks result from a distribution of attention to various architectural aspects? The describes techniques all tell a different story, because they implicitly have a different idea about what a good architecture is. So, to find the answers, one should first know what a good architecture is in a specific situation. If the architect has a clear view of when the architecture is good for a specific system and its environment, then it becomes easier to really achieve a good architecture.

This article offers concepts and techniques to understand what a good architecture is in a specific situation. Section 2 considers good architecture from the perspective of the definition of software architecture. Section 3 introduces the characteristics of a good architecture. Sections 4, 5, and 6 explain the notions of correct, consistent, and communicated in more detail. Section 7 presents a list of questions that help the architect to determine if an architecture is good. An architect can use these questions to determine which aspects and which activities should be given attention. Section 8 contains a simple questionnaire that helps the business manager to assess if an architecture is good. This article is mainly based on the corresponding chapter from DYA|Software (Deckers & Steeghs, 2010).

2 Defining good architecture

To understand any software architecture approach, a shared definition of software architecture is needed. There are many of those definitions. The one used in this document is based on the IEEE42010 standard (ISO/IEC, 2007) in combination with DYA (van Steenbergen, 2006). The first focusses on the concepts used to describe an architecture. The second positions architecture as a process instrument to support business goals and drive system development. The resulting definition is aimed at all types of software systems and realization processes:

> The software architecture of a system is a set of statements that give direction to the design, realization and evolution of the software in its environment.
>
> In which statements form a model of:
>
> - The structure of the system elements, their mutual relations and/or the relations between the elements and the environment.
> - The guidelines for the creation of the structure, the elements and the relations between them.

We use this definition as the starting point for determining what a good architecture is. The word "good" is simply added to the definition. This yields the following: A good software architecture of a system is a **good whole** of **good statements** that **give direction well** to the design, realization, and evolution of the software in its environment.

A good architecture is a good set of statements. This implies that it is clear which statements belong to the architecture (and which not), and that the statements in the architecture are interrelated and do not contradict each other. This applies to both statements in the architecture and to statements about the relation with the system environment. So, it concerns all architectural models, guidelines, and statements about the environment.

A good architecture is built from good statements. These statements are a description of the right problem, a right solution to the problem, or the relationship between problem and solution. The term problem stands for the goals and requirements that the system should contribute to and for which the system is a solution. Goals and requirements can be considered on many levels. For example, if the system is all the software for a business line in an organization, then they might be the business objectives of that business line. If the system is an application, then they might be the functional requirements of the application.

A good architecture gives direction well. For this purpose, the architecture statements must be used for activities that contribute to the realization of the system. In order to give direction to those activities, the statements should be understood by the people that have to apply them, like programmers, testers, and infrastructure designers. Architecture statements may also be defined in a formal language such that they can be used in tools like rule checkers, or code generators.

3 Characteristics of a good architecture

The concept of "good software architecture" compasses three sub characteristics. These characteristics make it possible to determine step by step whether an architecture is good or not. This is more practical than considering the total architecture spectrum at once. The division into separate characteristics also makes it possible to identify improvement areas for the architecture.

Software architecture plays a central role in the system realization process. Via the architecture, one determines how key requirements are met, and how the system will be realized with the available resources. This requires a good view on those requirements and resources, otherwise the system may not fit its environment. Also, the architecture must be free from contradictions and must be detailed enough, otherwise you cannot determine if the architecture complies with the requirements and if it can be applied predictably in the system. The architecture must also be aligned with the right stakeholders, otherwise the realization team may not be able to use the architecture, or the stakeholders may not be able to assess if the architecture is the right balance between the stakeholder interests. A good architecture should therefore have the following three characteristics:

- **Correct**: The architecture is based on adequate information about the environment and in particular the interests of stakeholders. Interests can be both goals and constraints, and they are prioritized. The architecture provides the right balance between the interests.
- **Consistent**: The architecture is coherent. The architecture statements do not contradict. This applies not only to the statements within the architecture itself, but also in relation to statements about the environment. The architecture must be implementable in the system. The architecture must provide a solution for the high priority requirements.
- **Communicated**: The stakeholders are aware of their relation with the architecture. They have the right expectations of the architecture and they know what their own contribution to the architecture is. Stakeholders must understand how their requirements are secured in the architecture. A communicated architecture begins with a communicable architecture description.

This notion of "good architecture" contains no assumptions about form and context of the architecture. There is no bias towards specific technologies, qualities, or architectural styles. Correctness, consistency and communication are properties that are applicable to all systems that must be realized by people, and wherein the architecture's role is to secure the most important system properties. A good architecture is different for each system, but should always be correct, consistent and communicated.

> Statements like "an architecture should be flexible, future proof, simple and complete" are often applicable. But these statements are not always valid. There are always examples of systems that these statements do not apply to or are not the most important criterion for success.

Different stakeholders use different representations with different formats to communicate and reason about their activities and decisions. All stakeholders have their own perception of their tasks and of the system. The architecture must be translated into the language of the stakeholders and must fit their statements and perceptions.

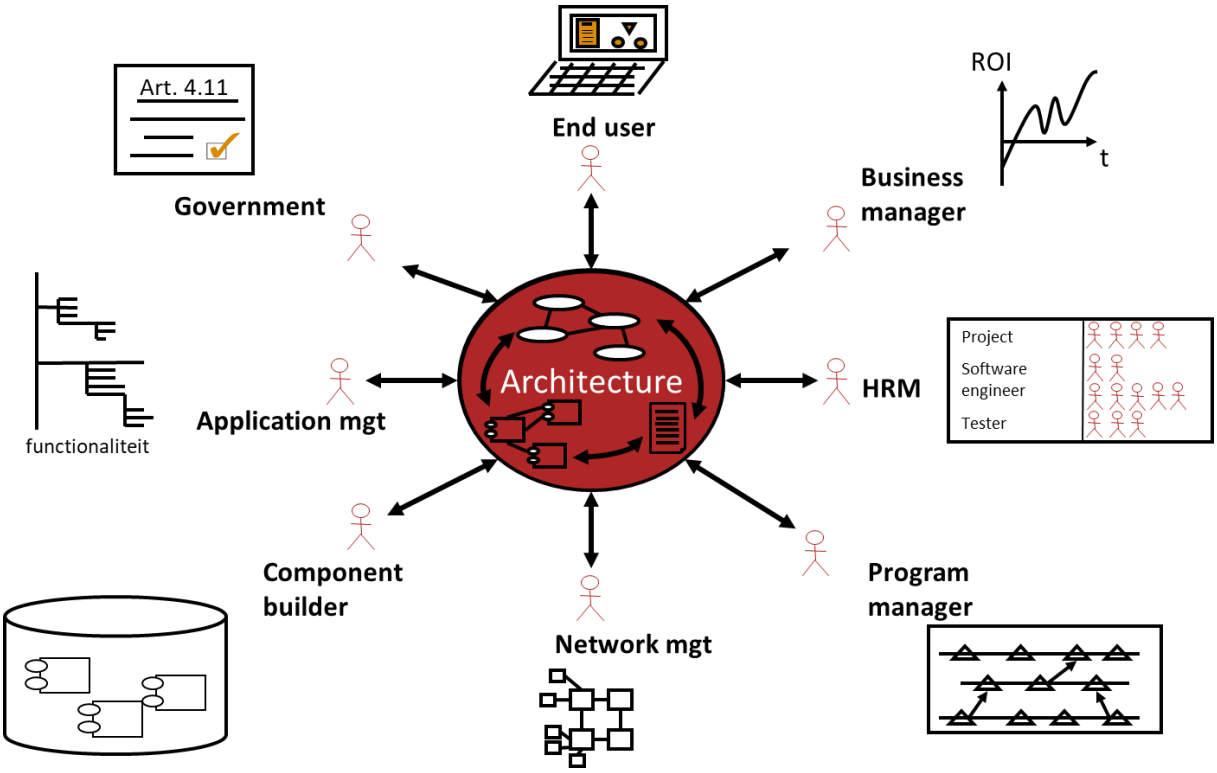

Figure 1 Good architecture: correct, consistent, and communicated

Figure 1 loosely depicts correctness, consistency, and communication of architecture. For example, the architecture must be in accordance with:

- The project plan of the program manager
- The network topology of the network administrator / network architect.
- The components that can be delivered by the component's builder (and manager)
- The financial planning of the business manager.
- The legislation from the government.
- The functional structure of application management.
- The user interface of the end user.
- The training and recruitment plans of the HR department.

The various architectural models and guidelines must also be mutually consistent. The following sections explain correctness, consistency, and communication of architecture in more detail.

4 Correct architecture

Correctness means that the architecture should be based on validated statements about the environment and in particular the stakeholder interests. It must form the right balance between stakeholder interests, and it must fit with other statements about the environment. The main question in determining correctness of an architecture is "**Does the system fit in its environment?**".

Correctness of the architecture is achieved by the following:

- **Adequate environment analysis**: there is adequate consideration of stakeholders, stakeholder interests, available resources and technology, and other information about the environment of the system. This information must be specific enough to form the starting point of the architecture creation.
- **Conscious balance of interests**: a guiding architecture is the result of a deliberate and traceable balance between different and possibly conflicting interests, and other environment statements. Furthermore, the interests are prioritized. The priority setting is a criterion for trade-offs between the system properties, like the system decomposition and used design patterns.
- **Validation of environment statements**: the stakeholders recognize and acknowledge their statements of interests, concerns, and goals. The validity of environment information is reviewed. Stakeholders understand that their interests are the subject of trade-off, which also must be validated.

These topics will be further explained in the sections below.

4.1 Adequate environment analysis

It is essential that the architecture is based on accurate information about the system environment. This applies especially to the stakeholder interests. The architecture should include statements that address stakeholder interests and other environment information. In order to come up with the right set of interests, the consideration of the stakeholders and the system environment should be broad enough and detailed enough.

The changes and trends that happen in the environment of the system are also taken into consideration. Think of (future) developments in the application domain of the system, technology that is used by the system, or changes in the development organization or maintenance organization. The scope of the environment analysis should match the intended lifespan of the system.

The environment analysis provides an overview of the requirements, assumptions, and starting points for the architecture. The validity and priority of all these aspects must be clear. This is a prerequisite for the definition and the testing of the architecture.

4.2 Consciously balanced interests

Stakeholder interests are often conflicting and have different priorities. Therefore, the architect must find a balance between those interests. It cannot be known completely what the best balance is before the system is operational. What the best balance is can only be fully analyzed when the system reaches end of life. However, the architect can create, monitor and communicate the balance consciously. To balance the interests, the stakeholders along with their interests are identified. Furthermore, the importance of each stakeholder for the success of the system is analyzed, and the interests of each stakeholder are prioritized.

> When using web services, the encryption of each individual message goes at the expense of response time. Whether the response time is more important than securing information, must be assessed in the context of the system.

If an organization realizes several similar systems, then a reference architecture can be defined for these systems. A specific system architecture has the reference architecture as a starting point. It may be the case that the interests at the system level have a different priority than in the reference architecture. The system architecture then describes the deviations from the reference architecture. When the same deviations occur repeatedly for more systems, then the reference architecture probably must be reviewed and possibly refactored.
Migration software that transfers data from an old to a new database, usually doesn't have to be maintainable. Architecture guidelines for the maintainability of software, are thus less important for migration software. Instead, guidelines to minimize the development cost can be more dominant for the realization of migration software.

4.3 Validation of environment statements

To demonstrate the correctness, stakeholders should validate the architecture and the information on which it is based. They should be able to test the architecture against their requirements and goals. Therefore, the architecture is represented in different ways for different stakeholders. A good representation uses concepts and notations that the targeted stakeholder understands.

The priorities of the interests are also validated. The main stakeholders are involved in setting priorities. In some cases, a simple prioritization of stakeholder interests is not sufficient to make the right choices in architecture. In that case, it makes sense to let the stakeholders work together to create a shared view on the prioritization. Eventually, the owner of the system is the main responsible for the prioritization.

5 Consistent architecture

The architecture must be consistent. The architecture guidelines and other architecture statements may not contradict each other. Consistency is a requirement, because the architecture must ensure the most important system properties before the system is realized. If the architecture is inconsistent, then desired system properties cannot be guaranteed or even predicted. The main question regarding consistency is: "**Is the system well-engineered?**"

Consistency does not only apply to the statements inside the architecture definition, but also to the statements about the context of the architecture. For example, the source code must be consistent with the prescribed design patterns. Moreover, the structure and terminology of requirements must fit the system partition in the architecture. The system elements must be linkable to the system requirements. The system partition in the architecture should also correspond with the work packages in the project schedule.

Consistency is achieved by the following:

- **Verification on contradiction**: all the architecture statements and relevant environment statements form a whole. This whole should be analyzed for contradictions.
- **Demonstrable realizable**: It should be clear that it is possible to achieve the system with the architecture and the available resources. The development team should be able to apply the architecture in their activities.
- **Consciously managed**: it must be clear what does and does not belong to the architecture description and how the architecture is changed in the context of the realization process.

These topics will be explained in more detail in the following sections.

5.1 Verification on contradictions

The architecture must be described in such a way that its consistency can be verified. This applies not only to the statements in the architecture, but also to the information about the system environment. To check consistency, it is important that the statements are unambiguously interpretable. This requires that the same terminology in used in all architecture statements[1].

Verifying the consistency is easier if the architecture is described in a formal architecture language (e.g., ACME, Rapide, Wright, C2 ADL, SADL, Archimate). In practice, the use of formal languages is limited to only a few architectural aspects. Many people find the concepts and notations of formal languages too difficult. Using a formal language especially pays off when there is an automated relation between the software and its architecture. For example, via automatic checks or code generation. This increases the consistency between the architecture and the software system.

As an alternative to a formal architectural language, a domain model can be used as the basis for the architecture description. The domain model defines the concepts to describe the architecture. The use of an explicit domain model ensures that the architecture description can be checked for consistency.

A consistent architecture can be assessed on aspects such as:

- How requirements are implemented in the design.

[1]It is theoretically possible that a translation is defined between all terms in and around the architecture. But, having to continuously translate statements is impractical and very confusing.

- How an upgrade from a purchased software component affects the project schedule.
- How a specific software design delivers the right quality.
- How fit a given software design is for iterative development.
- How work packages are defined from the specification of functional requirements.

5.2 Demonstrable realizable

It should be possible to realize a system that conforms to the architecture. This means that the architecture must not only be consistent in format, structure, and terminology (syntax), but also in terms of meaning and intention (semantics). It is easy to specify a system with great functionality and ultimate quality, but it is almost never easy to build it[2]. If there are important constraints regarding the realization time and costs, then a known realization method should be available. If ~~a~~ such a method is not present, then the feasibility of the architecture cannot be predicted. An architecture that cannot be realized within the given restrictions is simply unusable.

The system's feasibility is not guaranteed when those who must realize the system must follow contradicting guidelines. In this case, the architect must intervene by eliminating the contradictions. For example, by defining a guideline that helps to make a choice between two conflicting solution directions.

Because of user friendliness the directive "the user must provide data only once" is given, and because of security requirements the directive "the user must be authorized separately for each function" is given. The team that must build the system might not know how to match these directives. Therefore, the architect added the directive "single sign-on per user session" to help the builders.

The feasibility is also unclear when a desired system quality is not supported by any architecture statement. Achieving that quality is thus implicitly left to the realization team, without direction and control from the architecture. As a result, the architecture definition does not cover the appropriate system quality. Instead the quality depends on the skills and knowledge of the team. This can be acceptable if the team has a proven track record.

In the initial phase of an architecture, there may be conflicting directives. The architect should be aware of them. He must determine when and how conflicts are resolved. This happens for example if a reference architecture is not mature yet and projects still have to figure out what works and what not. This way, projects solve the contradictions and gradually make the architecture more consistent.

5.3 Consciously managed

An architecture should go as deep and wide as needed. To maintain a consistent architecture, it must be clear which statements do and which statements do not belong to the architecture. In addition, it is also important to know how the architecture statements are defined, and how they may be changed. To consciously manage an architecture does not necessarily mean that everything is described in detail and managed with a rigorous process. There are no specific requirements for managing architecture, other than those for the management of other development information (see for example (Hamer & Lepoeter, 1996)).

The architect also monitors the consistency of the software architecture in relation to other architectures and to the system environment. In the initial phase of an

[2]It is easy to specify a perpetuum mobile and a teleportation system, but it is very difficult to build them.

architecture, the monitoring of consistency is often easy. But, if the architecture is several years old, it is important to check if the original assumptions and original stakeholder requirements are still valid.

In the initial phase of a new system that must be on the market very soon, guidelines to keep the development duration short are important. Later, when gradually new system functions are added, system adaptability is important. The architecture must deal with this in a timely manner. Otherwise, the system will be hard to maintain.

6 Communicated architecture

A good architecture is communicated to those involved in the system realization activities. An architecture is a means to guide the process of system realization. Therefore, the architecture must be communicable. Stakeholders of the operational system must validate the architecture against their goals. Stakeholders that realize the system must focus on understandability and feasibility of the architecture. The pursuit of communicability affects the way the architecture is specified. In essence, communicated architecture is about the question "**Does everyone know what they need to know?**"

A communicated architecture is achieved by the following properties:

- **Translatable into action**: The involved people can take the proper actions based on what is communicated to them about the architecture.
- **Embedded in the organization**: There are enough stakeholders involved in the communication and they should receive information of sufficient depth to ensure the recognition and implementation of the architecture.
- **Consciously performed**: The communication between the stakeholders and the architecture is coordinated. The communication format and content are planned, and the architect is aware of who knows what and how the information is conveyed.

These topics will be further explained in the following sections.

6.1 Translatable into actions by stakeholders

Stakeholders have different roles with different responsibilities. Every stakeholder involved in the architecture should be able to translate the architecture into the right actions. This applies in particular to the development team. They should base their activities on the architecture.

Besides the development team, also other stakeholders must be able to handle the architecture:

- Project managers should base the project plan on it.
- Infrastructure Architects must understand what the required network capacity is.
- Testers should know what the units of testing are.
- Requirements engineers need to know for which quality attributes they should gather and specify requirements.

In order to take appropriate action, the architecture must be understood. This means that the architecture is presented in a language and via a medium that suits the targeted stakeholders. For example, an architecture should be presented to a business manager in terms of return on investment and financial risks. But in practice, architects often present technical models of the architecture to business management, such as components and interfaces, or the used programming techniques. Technical models do not help managers to make good decisions. So, these models should be left out in the communication to business management. Different stakeholders need different views. What a proper view is, is determined by the interest, role, and knowledge of the targeted stakeholder.

6.2 Embedded in the organization

To embed the architecture in the organization, the communication about the architecture should sufficiently cover the stakeholders and the proper aspects. The communication must not only explain the architecture itself, but also clarify the limitations and possibilities of the architecture. The communication towards the realization team is focused on securing the architecture in the software. This includes explaining how to

implement and apply the architecture, as well as checking if the architecture is applied properly. The communication towards the client is focused on verifying the architecture against the main client goals and requirements.

The architecture must have enough support within the organization to motivate the stakeholders to take the right actions. Therefore, communication is not only about handling the architecture itself, but also about gaining commitment and creating a shared goal. Rules and controls make the architect a policeman that nobody likes listening to. The architectural vision, the projected positive effects, and successes should also be communicated.

The management of the realization activities is not only based on the architecture. The followed development process and the used development tools may also address specific system qualities. These qualities should be in accordance with the architecture. For example, if time to market is more important than system reliability, then the architecture and the development process should both address this. As such, some architectural guidelines can be secured through the development suite (process, people, tools) and the architect can spend less time on guarding those guidelines.

6.3 Consciously executed

Because the architecture gives direction to the realization process of the system, the architecture must be carried out deliberately. The communication must be planned. The architect determines what is communicated to whom, how and when.

DYA (van Steenbergen, 2006) provides several techniques to communicate the architecture as the architecture vision, communication matrix, project start-architecture and the building permit. These help to embed the architecture communication in the development process of the organization.

7 Checklist for good architecture

The checklist below is not intended to grade an architecture. Its intention is to identify on which the architecture could be improved. The questions are organized according to the structure of Good Architecture. Their answers are either explicit or refer to a place where the real answer can be found.

Many questions start with "how". Their answers should express a way of working, and might include processes, roles, artefacts, methods, guidelines, or decision support, or refer to a place where this is described. Or, the answer is specific for each instance. For example, "How do the stakeholders recognize and acknowledge the stated concerns?" can be answered by "They sign of the concerns section in the architecture document.", or by "The users voted for them in the last users meeting, and the project manager defined his concerns in the project definition document.".

1. **Correctness**
    a. **Adequate environment analysis:**
        i. What is the definition/scope of the system? Make demarcations in:
            1. Application domain (functionality and usage)
            2. Technology and technical design
            3. Development and management (process and organization)
            4. Lifetime (system lifecycle)
        ii. Who are the stakeholders and what are the main concerns of each stakeholder? Focus on stakeholders and concerns that are system independent.
        iii. What things in the environment are considered and analyzed (over time)? What things are explicitly ignored?
        Think of competitors, market trends, frameworks, existing architectures, technological developments, cooperation with partners, neighboring systems, and related projects. Per considered environment element:
            - What are the relevant information sources? Think of input documents, market information, legislation.
            - How is the environment information used as input for architecture? Different information is probably used differently. Think of requirements management, proof of concept, technology scans.
            - To which view(point)s is the environment element related and how?
    b. **Consciously balanced interests:**
        i. What is the prioritization of the different stakeholder concerns?
        ii. What trade-off (design and realization) decisions between concerns are made in the architecture? Per trade-off:
            - Which concerns do they affect?
            - How can you trace the trade-off to the concerns?
            - On which environment information (assumptions, trends, related systems, related projects, …), besides the concerns, is the trade-off based?
    c. **Validation of environment statements:**
        i. How do the stakeholders recognize and acknowledge the stated concerns? This can be different per stakeholder and concern.
        ii. How are other assumptions about the environment checked for validity?
        iii. How is the prioritization of concerns created, defined, and managed? The role of the system owner is the most important.
        iv. How do you know if the stakeholders understand the trade-offs?

2. **Consistency**
    a. **Verification on contradictions:**
        i. Which terminology is used in which views (and models)? Which languages? Is there an explicit dictionary, domain model, or domain specific language?
        ii. How is it ensured that the terminology is shared and understood?
        iii. What are the consistency (correspondence) rules across models and views, and how are they guarded (enforced and tested)?
        iv. How is the architecture analyzed for contradictions? What is the process, and which tools are involved?
        v. How is clear that all the architecture (design) statements are related to environment statements, like to a stakeholder concern, assumption about the environment, a related system, or some constraint?
    b. **Demonstrably realizable:**
        i. What are the most important design/implementation issues/aspects that the architecture should provide a solution direction for?
        ii. How is demonstrated that the system can be realized with the defined architecture? How do you know it is possible to implement the architecture with the development suite?
        iii. How is ensured that the architecture will be applied during implementation? How do you know the development/maintenance team will apply the architecture?
        iv. How is demonstrated that the system is really built with the defined architecture? How do you know that the architecture is applied in the system?
    c. **Consciously managed:**
        i. What are the most important decisions in the architecture?
        ii. What documentation belongs to the architecture, and what does not belong to the architecture? What is the relation between the architecture documentation and the related documents?
        iii. Which models (kinds), view(points), describe the architecture?
        iv. How is the architecture managed? Which processes, roles, and tasks are involved?
3. **Communication**
    a. **Translatable into stakeholder actions**:
        i. What role does the architecture play in the activities of each stakeholder? How do they use it or how does the architecting process use the stakeholder's activities/artefacts? Which views are involved in each activity?
        ii. How is known if the stakeholders have enough information of sufficient depth about the architecture (to take the right actions)?
        iii. How is determined if the architecture and architecting process is translated into actions in the right way?
    b. **Embedded in the organization:**
        i. How is known if the most important and influential stakeholders are involved in the communication?
        ii. Who recognizes and acknowledges the architecture, and who does not?
        iii. What is communicated about the architectural vision, the projected positive effects, and successes?
        iv. How is the architecture embedded in the development suite (process, tools, and people)?
    c. **Consciously performed:**
        i. How is determined who (stakeholders) should know what?
        ii. How is the format of each communication determined?
        iii. How is the communication coordinated?
        iv. How is known what is communicated to whom?

8 Managing the architect

The milestone "architecture defined" often occurs in the definition of a development process. But, how does the QA manager, business manager, or project leader determine if this is really the case? It is not easy to do a simple test because the architecture addresses the qualitative aspects of the system and architecture documentation and agreements may differ per situation.

1.1 Checking the architecture

During the realization of a system, the management may carry out several checks on the architecture. Besides using the checklist of the previous chapter, possible checking methods with different cost and different depth exist:

- An independent third party conducts an architecture assessment. This can be based on an architecture assessment method such as ATAM (Kazman, Klein, & Clements, 2000) or SAAM (Abowd, Bass, Kazman, & Webb, 2007). None of these existing methods is as complete as the consideration of good architecture in this document. Though, both ATAM and SAAM follow a predefined review process, which helps to organize and plan the assessment.
- Stakeholders are asked whether they see their interests represented in architecture. If their interests are not well represented, then they should understand why this is the case. Different points are important for different stakeholders:
    - The representatives of the system owner must approve the prioritization and balancing of interests.
    - The developers and designers understand how to implement the architecture.
    - The program and project managers can base a plan on architecture.
- The architect is asked what the status of the architecture is. The following paragraph helps to make this practical.

1.2 10 questions from the manager to the architect

The manager may ask the architect about the quality and progress of the architecture. The list below contains ten basic questions that managers with little knowledge and experience in architecture, can ask the architect. The purpose of the questions is to get a hint on the progress of the architecture task, and how well the architect is in control of that task. Of course, this short list does bring a complete and thorough view of the status of the architecture. When a more detailed picture is required, it is better to use a dedicated architecture assessment method, or the extensive questionnaire given in Section 7.

The basic questions to the architect are:

1. What is the scope of the architecture? Specifically regarding:
    a. Application domain (functionality and usage)
    b. Technology and technical design
    c. Development and management (process and organization)
    d. Lifetime (system lifecycle)
2. Who are the main stakeholders and what is the main concern of each stakeholder?
3. What are the five[3] main concerns that the architecture must address?

[3]Five is a manageable number. Three is often too little to catch the essence and seven is too much to handle.

4. What are the five most difficult design issues that are solved in the architecture? Which stakeholder concerns are influenced by the solutions?
5. What are the five most important trade-offs in the architecture, and how are these defined, disseminated and guarded?
6. How is the consistency of the five main concerns and their solution demonstrated?
7. What is the main role of each stakeholder in relation to the architecture process? (Throughout the architecture's life.)
8. Which architecture views are there to reason, to explain, to convey, to describe, etc.? To which stakeholders is each view addressed?
9. How are the stakeholders informed about how the architecture addresses their concerns (or not)? How do the stakeholders know what is expected from them and what actions to take?
10. What is so great about the architecture that I must not forget? What should I really know about the architecture that I have not heard in the answers to the previous questions?

The architect should be able to answer all the above questions or should be able to explain why an answer cannot be given. Optionally, the answers can be discussed with various stakeholders, so the business manager can test their validity. The answers should provide facts about the architecture and should not be subjective. The manager has the above questions to determine whether the architecture is good or not. In addition, the questions also help the architect to assess if the architecting task is on the right track.

9 References

Abowd, G., Bass, L., Kazman, R., & Webb, M. (2007). *SAAM: A Method for Analyzing the Properties of Software Architectures.* Software Engineering Institute.

Bass, L., Clements, P., & Kazamn, R. (2003). *Software Architecture in Practice, Second Edition.* Addison-Wesley.

Bosch, J. (2000). *Design & Use of Software Architectures, Adopting and evolving a product-line approach.* Addison-Wesley.

Deckers, R., & Steeghs, R. (2010). *DYA Software, Architectuuraanpak voor bedrijfskritische applicaties.* Kleine Uijl.

Hamer, P. v., & Lepoeter, K. (1996). Managing design data: the five dimensions of CAD frameworks, configuration management, and product data management. *Proceedings of the IEEE. Volume 84, Issue 1*, pp. 42-56. IEEE.

ISO/IEC. (2007). *Systems and software engineering -- Recommended practice for architectural descriptions of software intensive systems, ISO/IEC 42010:2007.* ISO.

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation.* CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Muller, G. J. (2004). *CAFCR: A Multi-view Method for Embedded Systems Architecting. Balancing Genericity and Specificity.*

Rozanski, N., & Woods, E. (2005). *Software Systems Architecture, Working With Stakeholders Using Viewpoints and Perspectives.* Addison-Wesley.

van Steenbergen, M. v. (2006). *Building an Enterprise Architecture Practice.* Springer.