Role and Task of the System Architect

Gerrit Muller
Buskerud University College
Frogs vei 41 P.O. Box 235, NO-3603 Kongsberg Norway
gaudisite@gmail.com

Abstract

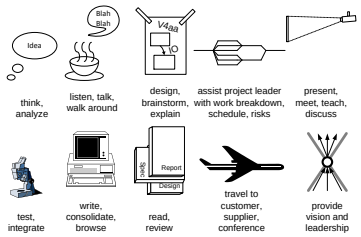
The role and the task of the system architect are described in this module.

version: 1.0 status: preliminary draft October 20, 2017

Contents

Chapter 1

The Role and Task of the System Architect



1.1 Introduction

Architects and organizations are often struggling with the role of the system architect (or software architect or any other kind of architect). This struggle is partially caused by the intangible nature of the responsibilities of the architect. At the other hand (good) architects are highly appreciated, even if their quantifiable output is low.

This article starts with specific deliverables, then discusses the more abstract responsibilities and, finally, discusses the day to day activities of an architect.

The role of the software architect is nicely discussed in [1].

1.2 Deliverables of the System Architect

We start at looking for the tangible output that is expected from architects. Project leaders and program managers do expect deliverables to be finished at appropriate milestones. Most Product Creation Processes define the deliverables of a System Architect to be artifacts such as documents or models. These artifacts are symbolized by the stack in Figure 1.1.

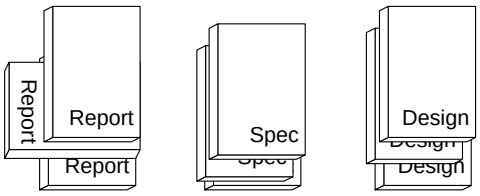


Figure 1.1: Deliverables of a system architect consists of artifacts forming a stack of paper when printed

Figure 1.2 shows the main deliverables of a System Architect more specific. Quite often the System Architect does not even produce all deliverables mentioned here, but the architect does take the responsibility for these deliverables by coordinating and integrating contributions of others. Note that some of these deliverables are part of the Policy and Planning Process.

Customer and Life-Cycle Needs *(what is needed)*

System Specification *(what will be realized)*

Design Specification *(how the system will be realized)*

Verification Specification *(how the system will be verified)*

Verification Report *(the result of the verification)*

Feasibility Report *(the results of a feasibility study)*

Roadmap

Figure 1.2: More specific list of deliverables of a System Architect

1.3 System Architect Responsibilities

The System Architect has a limited set of primary responsibilities, as visualized in figure 1.3. The primary responsibilities are:

Balance of system properties as well as internal design properties. The system should be balanced: for example, the cost of subsystems should correspond with its added value in terms of functionality and performance. Architecting is a continuous balancing act in many incomparable dimensions and quantities.

Consistency across many organizational and design boundaries; From needs to implementation details, from system level to detailed implementation.

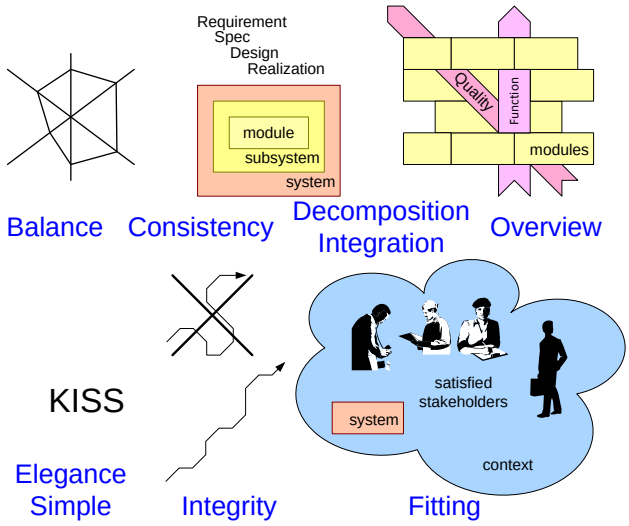


Figure 1.3: The primary responsibilities of the system architect are not tangible and easily measurable

Decomposition, Integration Decomposition is the standard answer in dealing with complex and big problems. Decomposing Systems in subsystems, subsystems in modules et cetera is a major responsibility of the architect. In most systems many decomposition dimensions are required: physical, logical, functional, and many more, see [3]. The complementary action of decomposition, however, is integration. The integral functioning and performance of the system is the ultimate goal of product creation, which emphasizes the importance of integration. In practice integration is much more difficult than decomposition, in fact the architect must decompose in such a way that integration is feasible.

Overview of the entire system and its context helps to make sensible specification and design decisions. The architect should provide overview to all members of the product creation team. Most of these members have a very limited horizon. The architect should help them by providing proper context information to make local design decisions.

Elegance, Simplicity are properties of a "good" architecture. The dangerous aspect of this responsibility is the highly subjective nature of elegance and simplicity. The appreciation of simplicity and elegance should be assessed or acknowledged by others than the architect.

Integrity of the system specification and design over time. The focus of a development team is often wandering over time, sometimes it depends on the

hype of the week. The architect is responsible for maintaining a balanced and focused development over time. For instance, when cost price reduction is required then the architect should keep performance and reliability on the agenda.

Fitting in stakeholder needs and system context, during the entire life cycle, is one of the core responsibilities of the architect. The architect must connect depth knowledge with breadth knowledge.

We can condense the primary responsibility of the System Architect as: to ensure the good functioning of the System Architecting Process. In practice, this responsibility is often shared by a team of System Architects, with one chief architect taking the overall responsibility.

responsibility	primary owner
business plan, profit	business manager
schedule, resources	project leader
market, saleability	marketing manager
technology	technology manager
process, people	line manager
detailed designs	engineers

Figure 1.4: (Incomplete) list of secondary responsibilities of the system architect and the related primary owner

The list of primary responsibilities as discussed above is suffering from a lack of measurability and is rather intangible. Systems Architects also have secondary responsibilities, where these are primarily owned by other persons. Most other roles in product creation are much sharper defined, as shown in Figure 1.4. For instance the business manager is responsible for the business plan and the financial results. The project leader is responsible for the schedule and hence for completing the project in time and within budget. The marketing manager is responsible for addressing the relevant markets and hence for market share and salability of the product. The technology manager is responsible for the timely availability of technologies and related tools. The line manager is responsible for the availability of the right people, with skills and processes to do their job. Final example are the engineers who are responsible for the design of their component or module.

1.4 What does the System Architect do?

Figure 1.5 shows the variety of activities of the day to day work of a system architect. A large amount of time is spent in gathering, filtering, processing and discussing detailed data in an informal setting. These activities are complemented by more formal activities like meetings, visits, reviews et cetera.

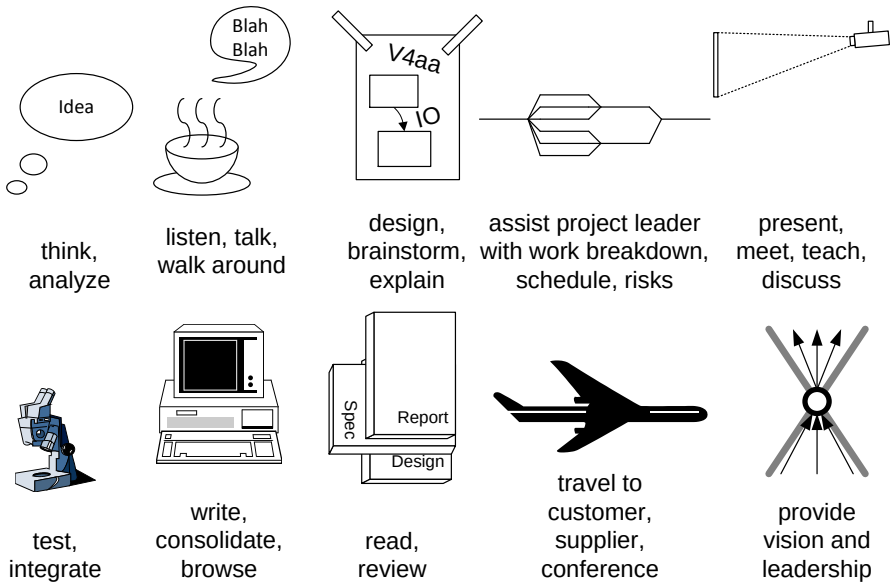


Figure 1.5: The System Architect performs a large amount of activities, where most of the activities are barely visible for the environment, while they are crucial for the functioning of architects

The system architect is rapidly switching between specific detailed views and abstract higher level views. The concurrent development of these views is a key characteristic of the way a system architect works.

Abstractions only exist for concrete facts

System Architects which stay too long at "high" abstraction levels drift away from reality, by creating their own virtual reality.

Figure 1.6 shows the bottom up elicitation of higher level views. A system architect sees a tremendous amount of details, most of these details are skipped, a smaller amount is analyzed or discussed. A small subset of these discussed details is shared as an issue with a broader team of designers and architects. Finally, the system architect consolidates the outcome in a limited set of views. The order of magnitude numbers cover the activities in one year.

The opposite flow in 1.6 is the implementation of many of the responsibilities

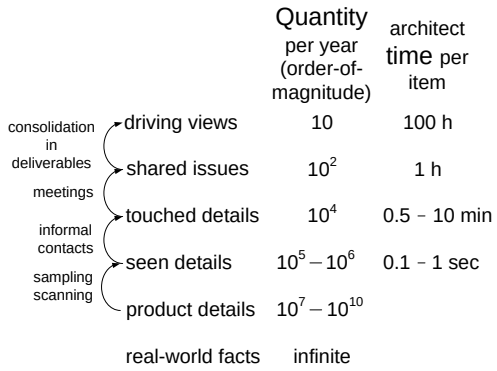


Figure 1.6: Bottom up elicitation of high level views

of the system architect. By providing overview, insight and fact-based direction a simple, elegant, balanced and consistent design will crystalize, where the integrity of designs goals and solutions are maintained during the project.

A lot of time spent by the architect serves the purpose of communication between many project members. The architect not only responsible for the system integration, but has also an integrating role in the project itself. The architect has to interact a lot with all the people mentioned in Figure 1.4, in order to fulfil the architect's responsibilities.

1.5 Task versus Role

The task of the system architect is to generate the agreed deliverables, see section 1.2 This measurable output is requested and tracked by the related managers: project leaders and the line managers. Many managers appreciate their architects only for this visible subset of their work.

The deliverables are only one of the means to fulfil the System Architect Responsibilities, as described in section 1.3. The system architect is doing a lot of nearly invisible work to achieve the system level goals, his primary responsibility. This work is described in section 1.4. Figure 1.7 shows this as a pyramid or iceberg: the top is clearly visible, the majority of the work is hidden in the bottom.

1.6 Acknowledgements

Nicolette Yovanof pointed out that the text belonging to Figure 2 and Table 2 was rather incomplete. She also mentioned that some more attention for the interaction with non-architects would be helpful. Chuck Kilmer provided feedback on "The Awakening of a System Architect", which resulted also in an update of

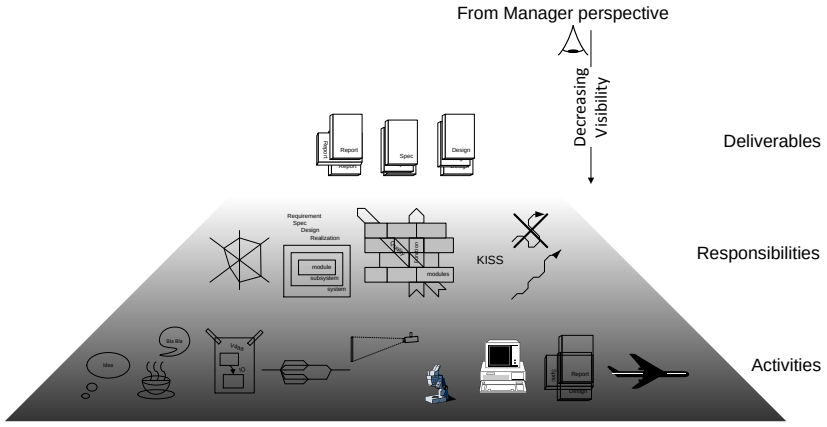
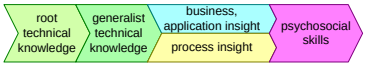


Figure 1.7: The visible outputs versus the (nearly) invisible work at the bottom

this paper. Byeong Ho Gong suggested a better coverage of the interfacing with customers/stakeholders. Pierre van de Laar provided textual improvements.

Chapter 2

The Awakening of a System Architect



2.1 Introduction

System architects are very rare commodity. This chapter describes the observed general growth pattern of system architects. We hope that by analysis of the the characteristics of existing system architects will facilitate the training of new system architects. Reference [4] contains a good description of a system architect.

2.2 The Development of a System Architect

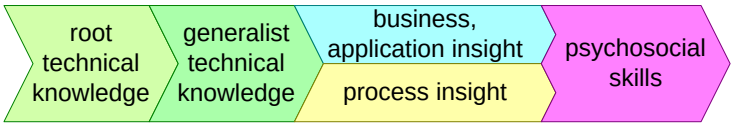


Figure 2.1: Typical Development of a System Architect

System architects need a wide range of knowledge, skills and experience to be effective. Figure 2.1 shows a typical development of a system architect.

The system architect is rooted in technology. A thorough understanding of a single technological subject is an essential underpinning. The next step is a broadening of the technical scope. Section 2.3 describes the path from a mono-disciplinary specialist to a multi-disciplinary system architect with broad technological knowledge.

When the awakening system architect has reached technological breadth, then it will become obvious that most encountered problems have a root cause outside of technology. The system architect starts to develop along two main parallel streams:

The business side: the market, customers, value, competition, logistics, service aspects

The process side: who is doing what and why, necessitated by the amount of involved stakeholders

During this phase the system architect will broaden in these two dimensions. The system architect will view these dimensions from a technological perspective. Again when a sufficient level of understanding is attained an awareness starts to grow that people behave much less rationally than technical designs. The growing awareness of the psychological and the sociological aspects is the next phase of growth.

2.3 Generalist versus Specialist

Most developers of complex high tech products are specialists. They need an in-depth understanding of the applicable technology to effectively guide the product development. The decomposition of the development work is most often optimized to create a work breakdown enabling these specialists to do their work with as much autonomy as possible.

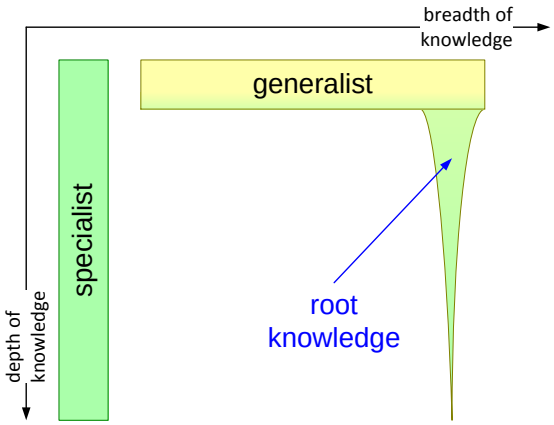


Figure 2.2: Generalist versus Specialist; depth versus breadth

Figure 2.2 is a visualization of the difference between a specialist and a generalist. Most generalists are constrained in the depth of their knowledge by normal human limitations, such as the amount of available time and the finite capacity of

the human mind. The figure also shows that a generalist has somewhere roots in detailed technical knowledge. These roots are important for the generalist self, since it provides an anchor and a frame of reference. It is also vital in the communication with other specialists, because it gives the generalist credibility.

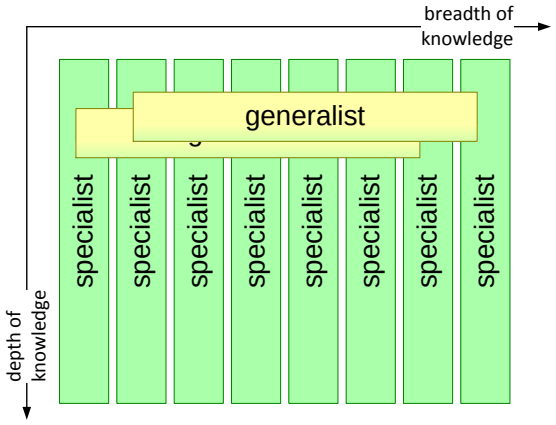


Figure 2.3: Generalists and Specialists are both needed in complex products, they have complementary expertise

Figure 2.3 shows that both generalists and specialists are needed. Specialists are needed for their in depth knowledge, while the generalists are needed for their general integrating ability. Normally there are much more specialists required than generalists.

There are more functions in the Product Creation Process that benefit from a generalist profile. For instance the functions of project-leader or tester both require a broad area of know how.

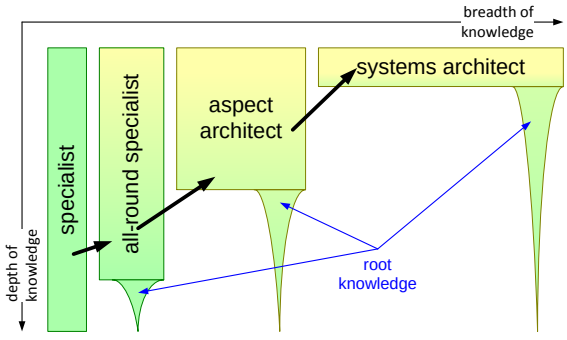


Figure 2.4: Growth in technical breadth, intermediate functions from specialist to system architect

Architects require a generalist profile, since one of their primary functions is to generate the top-level specification and design of the system. The step from a specialist to a generalist is of course not a binary transition. Figure 2.4 shows a more gradual spectrum from specialist to system architect. The arrows show that intermediate functions exist in larger product developments, forming natural stepping stones for the awakening architect.

Examples of aspect architects are:

subsystem architects subsystems are the main organizational decomposition. In hardware intensive systems subsystems tend to be physical, e.g. loader or generator. Typical number of subsystems is between 5 and 15.

SW, mechanics or electronics architects or discipline oriented architects. The architects ensure consistency across physical subsystems

function architects take responsibility for one system function, ensuring the soundness of that function.

quality architects take responsibility for one quality, e.g. safety, reliability, security.

For instance a software architect needs a significant in-depth knowledge of software engineering and technologies, in order to design the software architecture of the entire system. On the other hand a subsystem architect requires multi-disciplinary knowledge. The limited scope of one subsystem reduces the required breadth for the subsystem architect to a hopefully realistic level.

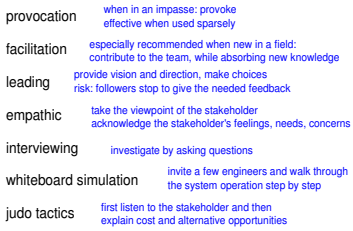
Many products are becoming so complex that a single architect is not capable of covering the entire breadth of the required detailed knowledge areas. In those cases a team of architects is required, where the architects are complementing each other in knowledge and skills. It is recommended that those architects have complementary roots as well; as this will improve the credibility of the team of architects.

2.4 Acknowledgements

Chuck Kilmer suggested a new title and offered many textual improvements.

Chapter 3

Architecting Interaction Styles



3.1 Introduction

A system architect has to use different interaction styles in different circumstances. In some circumstances a *leading* style is appropriate, while in other circumstances a *facilitating* style is more effective. Figure 3.1 shows the styles that are discussed in this chapter.

3.2 Provocation

A provocative style can be used by the architect when the discussion is in an impasse. The provocation can be based on taking an extreme viewpoint of one of the stakeholders and confronting the other stakeholders with the consequences. Such a provocation forces the involved stakeholders to formulate their needs more sharp, including the consequences of following the recommendation.

A provocative style should be applied scarcely. Once team members get used to this style then the style becomes ineffective. Most people do not like to be provoked continuously, so they stop to respond after a few provocations.

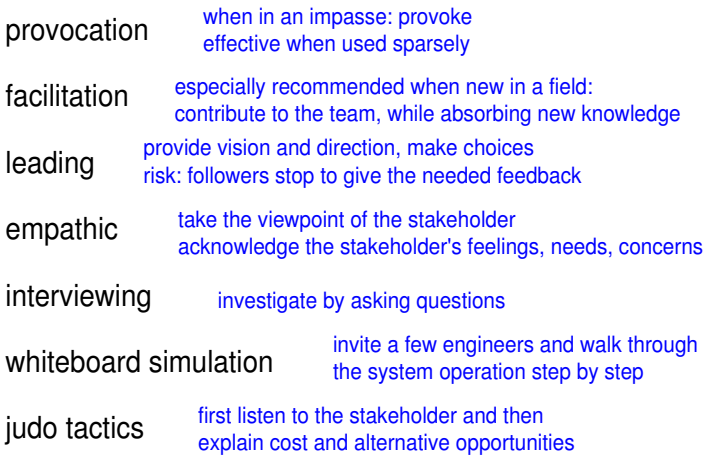


Figure 3.1: Interaction styles for architects

3.3 Facilitation

The facilitation style is a style where the architect serves the team by facilitating meetings and workshops. Facilitating a meeting means:

- preparing the meeting or workshop together with the owner of the meeting: determining the goal, participants, place, agenda, means.
- facilitating the meeting itself: timekeeping, managing the flips, writing action point and conclusions.
- finalizing the meeting: writing a report and presentation of the results, chasing follow-up actions.

The facilitation style is especially useful for architects entering a new domain. The architect provides visible value for the team, while as a spin off the architect learns a lot about the new domain.

3.4 Leading

A leading style is a style where the architect is highly visible. The architect provides vision and direction to the team. The leading architect can be recognized by looking at the followers: if they really follow the architect then the architect is effective as leader.

The risk of this style is that the team starts to trust the architect decisions too much. Most of the team members have much more know how about the design issues than the architect. The architect will often make decisions based on limited

know how that should be corrected by the specialists with more know how. The leading style sometimes inhibits the specialists to oppose the architect. The leading architect must be aware of this effect. Sometimes even invitations to oppose and provocations do not help to loosen up the followers.

3.5 Empathic

The empathic style is based on taking the viewpoint of the stakeholder under discussion. This goes much further than the objective rational view. The feelings and emotions of this stakeholder must be taken into account as well. The understanding of the state of mind is communicated back to the stakeholder. The result of this way of interacting is that the architect gets a much better insight in the stakeholder, while at the same time the stakeholder has the feeling to be taken seriously.

3.6 Interviewing

Architects pose lots of questions, questions are one of the most important instruments of the architect. The interviewing style makes excessive use of questions. The architect uses a priori knowledge to formulate open questions. These open questions must lead to an understanding of the stakeholder concerns.

The difficult part of this style is to use a priori know how in a limited and constructive way. The danger of a priori know how is that it limits observation and that suggestive questions are formulated instead of open questions.

3.7 White-board simulation

The white-board simulation style is used in meetings where a few specialists are present. The architect guides the specialists through a use case, where every specialist explains the system behavior from the specialist viewpoint. For example, the use case can be to push a *next channel* button on the user interface. In this example the user interface signal will trigger an avalanche of events in the system, going through many layers and propagating to many subsystems.

This guided simulation often reveals a lot of unknown system behavior, strange dependencies, inefficient sequences and many more engineering surprises. The normal reactions of the participants is that after a few steps they want to redesign the system. The architect should suppress this urge, by parking improvements at the side. The main purpose of this style is to build a shared understanding of the current design.

3.8 Judo tactics

The basis of judo tactics is that the architect starts to listen to the stakeholder, especially when the architect feels an urge to contradict the stakeholder. After listening to the stakeholder, and acknowledging the validity of the needs, the architect explains the costs and trade-offs. In many cases the stakeholders have a healthy feeling for value and cost and look for a reasonable balance. Quite often the result is a decision that the architect wanted to make right at the beginning. However, this style works only if the architect really listens, and is willing to take a different direction if needed. It might be that the architect discovers that the value for the stakeholder is much larger than originally assumed!

In many cases ill communication and bad listening skills block reasonable decisions. The judo style, where the architect starts to listen, avoids this trap.

Bibliography

[1] Dana Bredemeyer and Ruth Malan. Role of the software architect. http://www.bredemeyer.com/pdf_files/role.pdf, 1999.

[2] Gerrit Muller. The system architecture homepage. http://www.gaudisite.nl/index.html, 1999.

[3] Gerrit Muller. Architectural reasoning explained. http://www.gaudisite.nl/ArchitecturalReasoningBook.pdf, 2002.

[4] Eberhardt Rechtin and Mark W. Maier. *The Art of Systems Architecting*. CRC Press, Boca Raton, Florida, 1997.

History

Version: 1.0, date: March 25, 2004 changed by: Gerrit Muller

- Added Architecting Styles
- created reader