Behavioral competencies of a successful software architect in ASML
Effect of behavior on performance

Wolberink, S.W.R.

Award date:
2022

Eindhoven University of Technology

Master Thesis final report

In partial fulfillment of the requirements for the degree of Master of Science in Innovation Management

Behavioral competencies of a successful software architect in ASML

Effect of behavior on performance

Eindhoven, Februari 2021

Author:

S.W.R. Wolberink

Supervisors:

M. Razavian (1st supervisor TU/e)

H. de Groot(supervisor ASML)

A. Serebrenik (2nd supervisor TU/e)

A. Kleingeld (3rd supervisor TU/e

Eindhoven University(Tu/e)

Department of Industrial Engineering & Innovation Sciences

Master Thesis Innovation Management

Keywords: Software architect, behavioral competencies, stakeholder perception, manages complexity, communicates effectively

Abstract: In modern times of meetings and alignments the software architect should not only focus on technical competencies, but also behavioral competencies. In this explorative case study we look at which behavioral competencies are important for a software architect and how they lead to success. Two clear top clusters of important behavioral competencies were found, with on top manages complexity and communicates effectively. In addition, there were indications on how often a combination of behavioral competencies lead to success and how trust plays a key role as pre-requisite.

Acknowledgments

I am profoundly grateful to my university and company supervisors Maryam, Harmke, Alexander and Ad. For me as a student who has never conducted a full scientific research the deep ocean I ended up in felt overwhelming. Without their support and knowledge my research would not have been possible. In my ambition I often heard the words ‘stop trying to solve the world’, . By lending their knowledge from their varying areas of expertise I felt myself growing in my role as researchers, but also as a person. I did not solve the world, but I am proud of what I created.

I would also like to specifically thank ASML and all the people whom I’ve met there. Not only did they give me a welcome feeling but without their interest and participation this research would not have been possible.

Finally, I would also like to give a special thanks to friends and family for supporting me during the research and writing stage.

Summary

This research is an explorative case study of the behavioral competencies of a software architect. Both in literature and in the industry, most topics have addressed the technical side of the software architect(Muccini et al., 2018) and less then 5% have incorporated elements of behavior(Lenberg et al., 2015). Existing literature has addressed behavior in software architects in various ways(Bass et al., 2008a; Bredemeyer, 2002; Bredemeyer & Malan, 2002; Clements et al., 2007; Erder & Pureur, 2017; Hoorn et al., 2011; Klein, 2016; Kruchten, 1999, 2008), however there is still a lack of understanding on which of these behaviors are the most important and how they are important. One of the ways of operationalizing behavior is via (behavioral) competencies, often used in consultancy(Bartram, 2005; Bredemeyer, 2002; Kornferry, n.d.). Few case studies have been conducted before to analyze the behavioral of software architect in the field(Martini et al., 2014; Premraj et al., 2011). This research aims to address these gaps by conducting an explorative case study with the following research question:

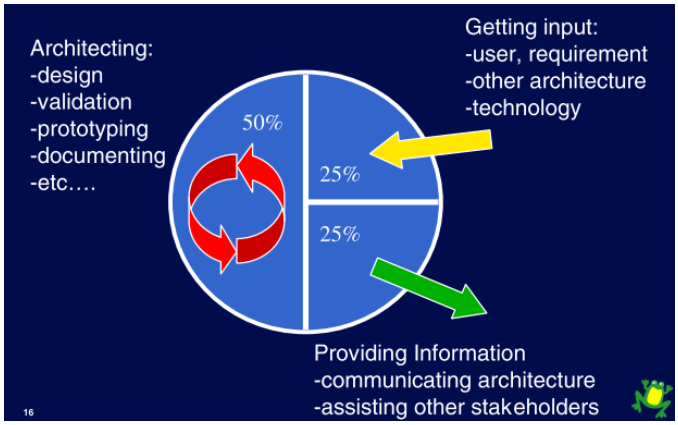

Figure a: What do architects really do?(Kruchten, 2008)

(RQ): ***What are the most important behavioral competencies of a successful software architect?***

The internally used behavioral competencies of the company are used since employees are already familiar with these and since internal frameworks are more transparent for the researcher. The internal framework defines 20 behavioral competencies(Kornferry, n.d.) which all have leadership aspects, the 20 behavioral competencies are given in table a. These behavioral competencies will be used to identify the most important behavioral competencies of a software architect. In order to understand which behavioral competencies are important, it first has to be clear what an architect does. In general, the architect is the technical leader responsible for the systems integrity(Bass et al., 2008; Britto et al., 2016; Eeles, 2006; Weinreich & Groher, 2016). In this role there seem three distinct hierarchal levels: system-level architects, product-level architects and team-level architects(Britto et al., 2016; Martini et al., 2014). Team-level architects are excluded from the scope. The activities of the architect, overview in figure a, can be described as activities focusing on getting input, architecting activities and activities focusing on providing information(Kruchten, 2008).

Methods

This research is an explorative case study with the goal of expanding the knowledge of the software architects behavioral competencies. In order to know which and how behavioral competencies are important for a successful software architect a multi-method approach was taken. First, a literature study was conducted to orient and create expectations on the which question. The literature study followed the snowballing procedure of Wohlin(2014) selecting the starting articles from the leading sources as recommended by Webster & Watson(2002). A questionnaire was held, following the principles of Blumberg et al.(2011) to gather quantitative data to analyze both the ranking and compare different groups in order to answer the which-question. Follow-up interviews were used, designed

following the steps of Turner(2010) to get more in-depth information in how these important behavioral competencies lead to success.

Questionnaire results

The questionnaire used an implied 'success' asking the 220 participants to rank all 20 behavioral competencies(Kornferry, n.d.) on a numeric scale of 1 to 7 on how important they were for successful software architects. Both the mean-based automatic clustering method scott-knott ESD, figure b, and the absolute ranking showed how the behavioral competencies manages complexity(BC4) and communicates effectively(BC3) were the top cluster of important behavioral competencies. Every single group or subgroup had these as their top 2, supporting hypotheses 1 and 2 which saw these two as 'most' important. The second cluster consisted of Decision quality(BC20) and Collaboration(BC12), which were either the 3rd or 4th most important behavioral competencies for both stakeholders(n=161) and architects(n=59). This would reject the hypotheses 3 stating decision quality being the most important, since it is not part of the top cluster. This does not mean it is not important, it is just not the top cluster. However, stakeholders and architects did switch them around, where architects valued collaboration as the 3rd ranked and stakeholders decision quality as 3rd ranked.

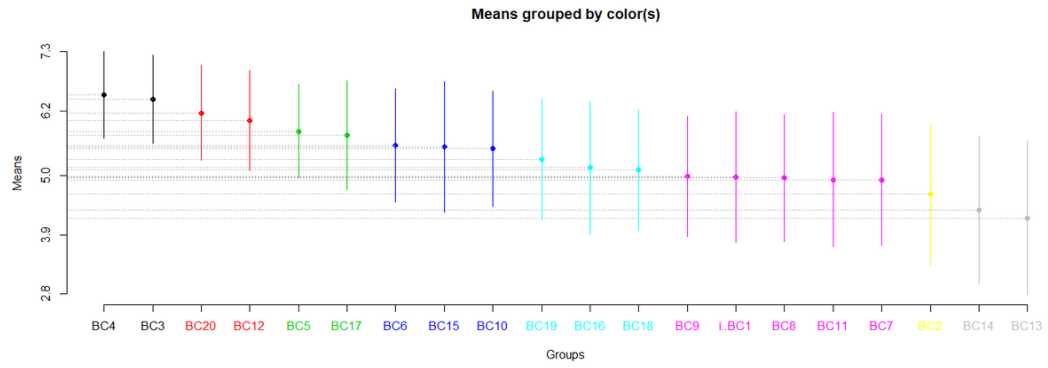

Figure b: Scott Knott ESD cluster results

The mann-whitney U-test found decision quality to be almost significantly lower(p-value= 0.065 for architects compared to stakeholders. However, in the exploration of the sub-groups this difference seemed to be caused by the product-level architects whom valued Decision quality almost significantly lower(p-value= 0.072) than system-level architects. Next to this there were significant differences in plans and aligns and balances stakeholders between architects and stakeholders. Since the difference between architects and stakeholders on Collaboration was not significant, hypotheses 5 was rejected. Since Decision quality was almost significant and is expected to become significant with increased sample size, this would indicate a difference. However, since it was not significant, the formal conclusion would be to deem the outcome of hypotheses 4 to be inconclusive.

Table a: list of behavioral competencies(Kornferry, n.d.)			
1 Ensures accountability	6 Cultivates innovation	11 Plans and aligns	16 Business insight
2 Develops talent	7 Drives engagement	12 Collaborates	17 Strategic mindset
3 Communicates effectively	8 Drives results	13 Build effective teams	18 Demonstrates self-awareness
4 Manages complexity	9 Values differences	14 Optimizes work processes	19 Self-development
5 Instills trust	10 Situational adaptability	15 Balances stakeholders	20 Decision quality

Interview results

The 11 interviews explored how the top ranked behavioral competencies led to success. First, it confirmed the top 4 behavioral competencies, them being present in all 11 interviews. Communicates effectively seemed to happen both in the process of gathering information, but also when convincing others. Manages complexity was seemed linked to the knowledge an architect has and dealing with all the different information inputs, some also addressed it being part of knowing what to communicate to others. Collaboration happened both in working together in order to assemble all information and get to the problem, but also working together to make the decision and execute it together. Decision quality was more often seen as the decision itself focused on both making an individual decision or guiding the decisions in order to create a joint decision. Interesting was how instills trust was present in 9 out of 11 interviews. Instills trust seemed to be required in order to have success when convincing others.

Conclusion

To answer the research question: What are the important behavioral competencies for a successful software architect? The most important behavioral competencies of a software architect are manages complexity and communicates effectively. However, there is a second cluster consisting of collaboration and decision quality which can also be considered important. Furthermore, instills trust seems to be a pre-condition for success.

The questionnaire contributes to existing literature by creating a sense of prioritization but also indications how different groups could perceive what is important differently. The interviews allowed for these important behavioral competencies to be mapped on the different activities of the software architect(Kruchten, 2008), as can be seen in figure c. However, maybe even more important it indicated how multiple behavioral competencies are used together in order to be a successful software architect.

Limitations and future research

This last insight shows how observing single behavioral competencies does not always create the full dynamics. With the knowledge of which behavioral competencies are important, future research should focus on combinations of behavioral competencies in order to understand their dynamics. In addition, more research is needed to explore the different dynamics of the behavioral competencies and how they lead to success. Furthermore, one of the major drawbacks is that success has not been defined, but rather was implied by the participants. The insights are still valuable with a high response(n=220), however, more research is needed to explore what success really means. In addition, the interviews gave insights but they could be biased based upon the case context, more case studies are required to verify the findings of this research.

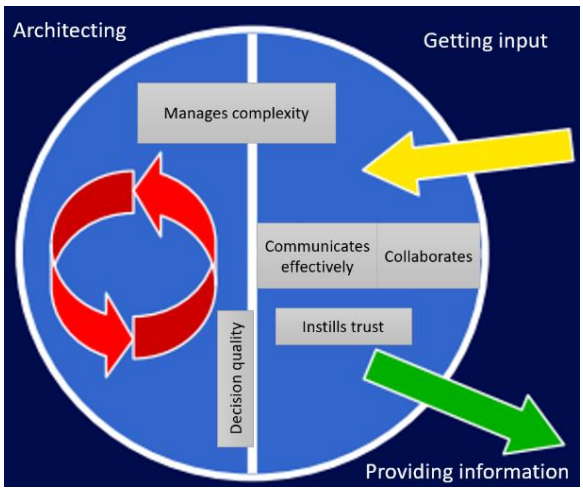

Figure c: Behavioral competencies mapped on the architects activities

Table of contents

List of Figures

List of Tables

Chapter 1: Introduction

From the software engineers to the highest architects, social interactions have become an unavoidable part of the job. Think about the amount of alignment and requirements management that have to be done to create software products. In all these interactions with people, the behavior of individuals becomes more and more important. One of the roles in which the behavior and interactions are even more important is the software architect. The software architect can be described as the technical leader responsible for maintaining the system integrity(Bass et al., 2008; Britto et al., 2016; Eeles, 2006) and is often seen as a decision-maker(Fowler, 2003). Half of the role of the software architect is related to gathering and giving other roles information(Kruchten, 2008). But which types of behavior should a software architect apply to be successful?

Research in technical fields, like the software engineering field, is slowly starting to realize the importance of these behavioral aspects of a job. However, less than 5% of the software engineering literature incorporates behavior as an element(Lenberg et al., 2015). In contrast, the roles within software engineering becoming more social. And the bigger the company, the more potential interaction is required of every role. Both in literature and in practice most topics that have been researched in the past have had a technical focus(Muccini et al., 2018). Behavior affects success in the software engineering domain(Lenberg et al., 2015), and thus for the role software architect. Companies often operationalize behavior by expressing it via (behavioral) competencies(Bartram, 2005; Bredemeyer, 2002; Kornferry, n.d.). Competencies are a combination of knowledge, skills and behavior, and can be trained. Technical competencies focus on the tasks, tools and knowledge required to do the 'hard' part of the job like using a certain tool or coding language. Behavioral competencies, also expressed as non-technical or 'soft', focus on the people and deductive part of the job. This research aims to contribute to the field of behavioral software engineering by exploring which behavioral competencies are important for a software architect and in what context.

1.1 ASML

ASML is a company with over 28,000 employees and 14 billion in net sales(ASML, 2020) making it a big player in the semiconductor industry. The European Union wants to secure Europe's relevance in the global semiconductor industry, this will enforce this giant European company's relevance on a global scale(ASML, 2022). In addition, the growth is not about to stop since there is a global chip shortage which even lowered car production by 100,000(BBC, 2021). The company creates highly complex machines which can 'print' layers on small plates creating a chip. These layers are expressed in nanometers, requiring precise and highly technological equipment. The machines themselves are about the size of a bus containing 100,000 parts and 2km cables requiring multiple cargo planes when shipping(Wired, 2021). Developing these machines is a lengthy and costly process where ASML invested 6 billion euros over 17 years in R&D(ASML, n.d.-c). However, this development is required to stay competitive in the market. In addition to developing, older products like the PAS and TWINSCAN systems are still supported.

These big complex machines are not only a bunch of hardware thrown together, but it also contains a lot of software. The success of software projects is crucial as it is used for user interfaces, measuring, automated responses, steering hardware, and various other purposes. Different components can

contain software steering the sub-element itself, but can also be a separate module that measures certain elements in the machine itself. Logically, it is a daunting task for software architects to maintain system integrity and control the dependencies of software as it is both crucial for success, but also dependent on hardware. Even though there is evidence on the importance of the role of behavior on success in software projects, we know little about how it exactly functions. This case study will focus on the software side of the machine.

1.1.1 Problem analysis

ASML is a large company creating complex products where many separate groups need to work together, the structure of the case company can be found in figure 1a. To create a new scanner the machine is split up into components, there are multiple different clusters referred to as SF's all addressing a component. Every component has different functionalities, the responsibility of function groups(FC's) Every FC has multiple sub parts. This structure creates a complex structure of decision-making and interactions. This structure follows a matrix function where in the SF's there is a long term and a short term focus. The long-term focus(5 year) is represented on the platform side whereas the short-term (2 year) focus is represented on the product side, overview of architects can be found in figure 1b.

Architects work together with multiple stakeholders of different types. Apart from working with (internal) customers, as a technical leader, they often work with engineers. They are also placed in the '3-in-a-box' which is a decision triangle with project and product management.

ASML already has an existing framework containing 12 behavioral competencies plus 8 leadership competencies making together 20 behavioral competencies. They have been selected out of the internationally known Korn-Ferry framework as being the most important competencies within the ASML context(ASML, n.d.-b, n.d.-a). The definitions of these competencies can be found in appendix A. Muccini et al.(2018) addressed the lack of knowledge on the behavior of software architects in both literature and practice. As ASML is growing a lot, it is important to know the impact of behavioral competencies and their importance at different levels as it will help to create effective succession plans and train people for the next level in their careers. It can also help to create better profiles when searching for new candidates externally.

Where the size and growth of ASML presents a challenge on their part, it creates a research opportunity. Having a large complex system with a lot of dependencies creates the need for multiple architects maintaining the technical elements and integrity And the size of the development ensures that there are more architects available to study than there are in an ordinary company.

1.1.2 Existing roles and scope

This research will focus on experienced architects as for starting architects there can be a discussion on their role, for example, is it more lead designer or is it also including other non-architect tasks. By starting at the FC level we will talk to both architects as well as stakeholders who have enough experience to explicate their views on what makes a software architect successful and who have seen enough developments to differentiate based on experience between successful and contra-paradigms of working. The first layer of software architect roles which are included are functional software

architects(FCA) and functional software test architects(FCT). The FCA is responsible for the mid- and longer-term reference architecture of the FC. Where its test counterpart is responsible for test-strategy, testability and quality. In this layer, there are also have train architects, who are architects responsible for the execution of development for the coming ~1.5 years when product definition is agreed upon. The role of FCA and train architect is sometimes combined and sometimes separated. The layer above consists of architects responsible for a larger overview like an entire product or an entire subfunction, namely: SF architect, platform architect, or product architect. An overview of all the architects and their segments is given in figure 1b.

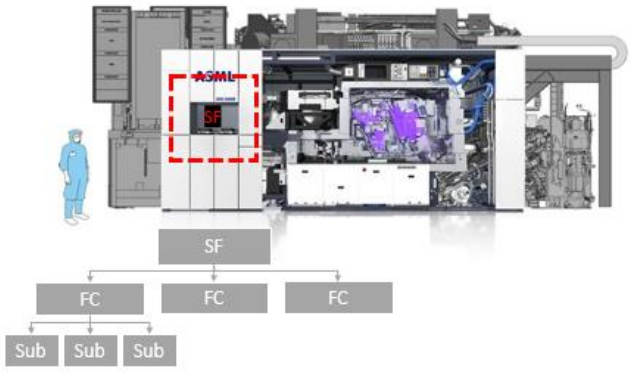

Figure 1a: Hierarchal structure of case company

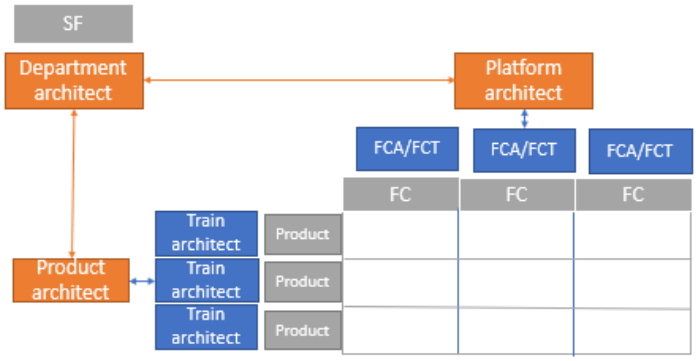

Figure 1b: Architect types in case company

The literature describes three layers of software architects(Britto et al., 2016). First, the system-level architects are responsible at a system-level where multiple products have to work together. They focus on high-level design and risks(Britto et al., 2016). In ASML's context, this would be the department architect, product architect and platform architect. As shown in figure 1b, their responsibilities are over multiple groups over the two scope axis, which is also supported by literature(Britto et al., 2016; Martini et al., 2014). Second, the product-level architects. Where they have more responsibilities regarding solving problems directly with teams(Britto et al., 2016), they are responsible for the practical decisions and execution involved with the decisions made on the system level. They are still responsible for risk management and design decisions, and work often with system-level architects. In ASML's structure, these would be seen as the FCA, FCT and train architect who both have the in-between responsibilities between architecture and a team while still not micro managing. The third hierarchal layer, team-level architects, was not included in this scope.

1.2 State of the art

The state of the art is a result of the literature review. This chapter will present the main status focusing on the gaps In literature, and the research question. Chapter 2 will address the methodology of the literature study and focus on forming hypotheses. The visualization of the research question can be found in figure 1c.

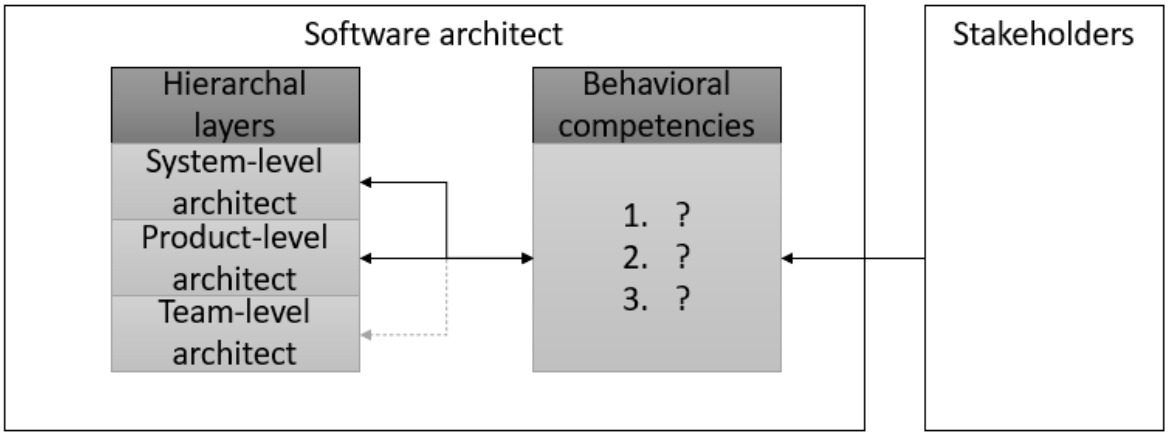

Figure 1c: Visualization of literature gaps

Existing literature in the entire software engineering domain has included behavior, but not often. The field of behavioral software engineering has been getting more attention but less than 5% of the research has focused on behavioral elements(Lenberg et al., 2015). Even though only a limited number of behavioral elements have been linked to success on a team, project, or organizational scale(Baltes & Diehl, 2019), there is a clear link on behavioral elements that influence being successful. These findings originate from the software engineering domain, and even though the software architect is part of this domain findings for this specific sub-category are even more difficult to find. This is not only a research gap but also a gap for practitioners where in the past 25 years both industry and literature have mainly focused on technical aspects of the software architecture(Muccini et al., 2018). Consequently, there is a lot of information on the technical elements of the software architect. If there is research about the architect and its expected skills, the focus originates from this technical focus studying the role and task descriptions(Bass et al., 2008; Kruchten, 2008) or from analyzing job offers(Ahmed et al., 2015). Filling the gap by turning towards behavioral competencies instead of only focusing on the technical tasks at hand will also benefit the technical solution space in the future. It is known that in architecture-related issues, 40% is not related to architecture itself but rather in elements like communication(Premraj et al., 2011). Additionally, research should not only take the perspective of the task description of a software architect but should also focus on the gap where practitioners show what they consider to be important. There are case studies like Britto et al. (2016) or studies using consultancy frameworks as its base(Kruchten, 2008; Razavian & Lago, 2015) which include behavior, but rarely do they explore how competencies lead to success and which are related to being successful from the perspective of the practitioner. In addition, there is a lack of case studies verifying findings from literature in regards to the software architect.

1.2.1 Behavioral competency

In literature, many different terms have been used to address the behavior of the software architect. In a certain sense, depending on the definition you use, people might address the same thing with a different name. To define competency, “A **competency** is the set of behaviour patterns that the incumbent needs to bring to a position in order to perform its tasks and functions with

competence”(Bass et al., 2008). Competence is being seen as: “Being competent, adequacy, possession of suitable or sufficient skills OR creating valuable results without using excessively costly behavior.” (Bass et al., 2008, p.5). Few use the term competency directly(Bass et al, 2008; Bredemeyer, 2002), however many talk about different aspects of an architect which would be included in this definition of competency like skills, knowledge or experience(Bass et al., 2008; Bredemeyer, 2002; Clements et al., 2007; Hoorn et al., 2011; Klein, 2016; Kruchten, 1999, 2008; Razavian & Lago, 2015). As mentioned before, much of the literature focuses on technical elements of the software architect like the technical competencies. However, this creates a larger gap in the knowledge surrounding behavioral competencies of a software architect, even though behavioral competencies are a widespread concept in industry and consultancy(Bredemeyer, 2002).

1.2.2 Operationalizing competencies

Measuring behavior or behavioral competencies has been a difficult subject to research, let alone in engineering fields like software engineering. This research made a clear scoping towards focusing on the less common behavioral competencies over the more explored technical competencies. Technical competencies focus on more concrete skills like knowing a specific methodology or tool in a specific domain. One widely used tool to measure ‘people’ is the Myers-Briggs Type Indicator(MBTI) which is a personality type of research. Good software architects would be introvert, intuition-based, thinking people who would either have a judging or perceiving personality(Kruchten, 1999). But, this tool mainly focuses on personality, rather than behavior competencies. In addition, MBTI has been criticized in the past and has rarely been used in software engineering research in the past making previous findings less relevant due to the change into agile working methods(Sach et al., 2010).

Behavioral competencies mainly find usage in consultancy frameworks like Bredemeyer's (2002) or ASML’s used Kornferry (n.d.) framework. Articles like Kruchten(1999, 2008) used the Bredemeyer framework as a base for his work. These frameworks are not transparent to people outside the organization but are known inside the organization creating the preference to use these in research. There is a strong preference to use internal frameworks in research, not only due to the availability of definitions and material, but also the better fit between organization and model. Therefore the Kornferry model will be chosen over one like Bredemeyers.

Alternatively, there are competency models outside of the software engineering domain one such model would be ‘The great eight competencies’(Bartram, 2005). A huge drawback of using generic competency models would be the lack of context and the mismatch between terms and definitions used within the software engineering domain. In addition, even though this model is scientifically validated, it is based on the SHL consultancy firm. This firm is a leader but it experiences the same untransparent issues as other models. The choice of using the internal framework is clear, but a comparison between Kornferry (n.d.)and Bartram (2005) could be beneficial due to the scientific relevance of the great eight competencies.

The framework connects the behavioral competencies with the big five personality traits: Agreeableness, extraversion, contentiousness, openness and Neuroticism. The 20 behavioral competencies of Bartram(2005) compared to the 20 behavioral competencies of Kornferry(n.d.) seem to have a lot of similarities. The main difference between the two models is the lack of tailoring on specific

architect elements where there is too much noise, however, what is there is more elaborate and focused on specific sub-elements. In general, all elements are present, but Bartram has a more generic description. This description splits up elements like communication and managing complexity into separate elements. Kornferry is more adapted to software architects where the pillars are more in line with company policy, this creates a more effective operationalization in this context. In addition, the definitions are simpler yet allow for more focus on the view of what a person should know like business insight and how it relates to what a business needs like drives results and drives engagement. One of the most important inclusions of Kornferry would be the inclusion of Decision quality itself as a behavioral competency. As will be shown in the next chapter, making decisions and the quality of them is a critical part of what an architect should do. With only a few differences and practical differences in how it is shown these frameworks generally express the same sentiment, however, the differences favor the usage of the existing framework of Kornferry.

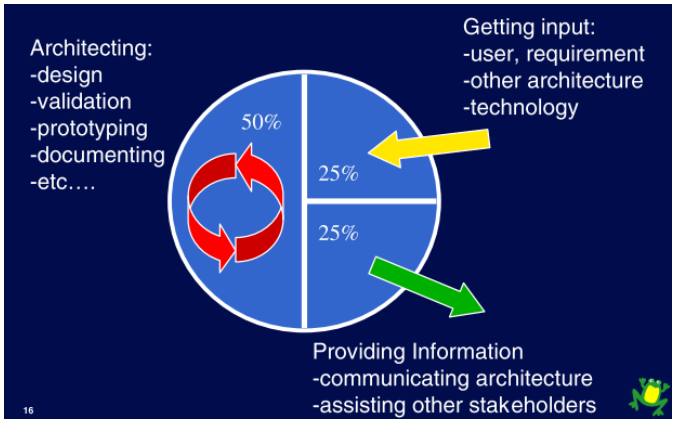

Figure 1d: What do architects really do? (Kruchten, 2008)

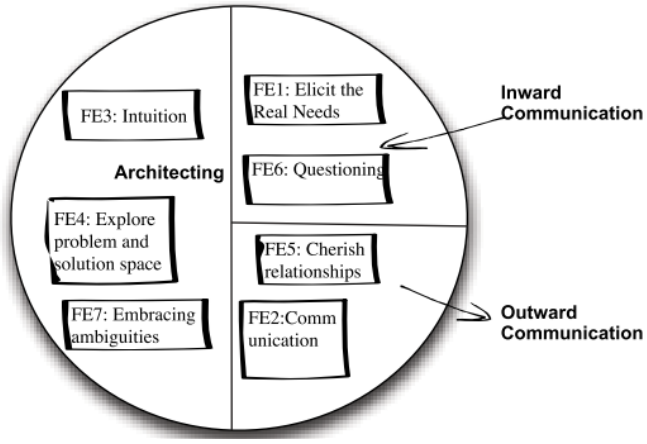

Figure 1e: Map of expertise on the architect activities(Razavian & Lago, 2015)

1.2.2 The architect

To further understand the gaps in existing literature, we have to know what a software architect is. There is an agreement of the agrees that the main responsibility of a software architect is the system's integrity(Bass et al., 2008; Britto et al., 2016; Eeles, 2006; Weinreich & Groher, 2016) however, there are also some arguments of an additional mentor(Britto et al., 2016) or leadership(Bass et al., 2008) role for the software architect. Even though it is clear what is expected from the architecting part of the software architect, what is expected in the role of technical leader or knowledge manager still sees some debate. This could be considered an element that requires attention in future research.

One of the clearest descriptions of the software architects' role and activities divides the role into three elements: architecting, inwards communication, and outwards communication(Kruchten, 2008). These three elements have an optimal balance, given in figure 1d, where the architect should get input 25% of the time. This focuses on getting and analyzing input like user requirements, knowledge about technology, and architectural knowledge. In addition, the architect should spend 50% of the time on architecting activities like designing, documenting, prototyping, and validating. The final part is providing information that focusses on the communication of the architecture and assisting stakeholders.

In addition, four antipatterns are identified which indicate the wrong behavior in software architects. These antipatterns could be useful in identifying a lack of certain competencies in practice. These are Goldplating, the architect in the ivory tower, the absent architect, and being the consultant. If it is doing too much architecting or only focusing only on one type of communication, all these patterns are seen as undesirable. Even though this does indicate what architects should do, it does not discuss what successful behavior is attached to it. There is a direct relation between the model of Kruchten(2008) and (feminine) expertise being mapped on it, figure 1e. For example, cherish relationships and communication were observed to be part of providing information. Even though expertise is related but not the same as behavioral competencies, it does show how the relationship between behavior and the core activities is present and relevant.

1.2.3. Different architect levels

Just like with ASML, the case study of Oce(Premraj et al., 2011) shows different levels of architects and different types of roles working with software architects, figure 1f. Where roles depend on context, the previous chapter defined the different layers and linked them to ASML's context. The two models describe the different roles as follows. (Britto et al., 2016) separates a system-level, product-level, and team-level architect based upon what they refer to as being a guardian or a mentor in combination with being a decision-maker or a problem solver with the team as was defined by (Fowler, 2003). Martini et al.(2014) created a framework where a chief, governance, and team architect were separated with responsibilities and focus for each role. This focus mainly included what level of design decisions are being taken, what type of risks management should be conducted, and what type of documenting should be generated. Chief architects focused on high-level decisions and architecture. Governance architect focused on inter-feature architecting and testability. The team-level architect does not have to be a dedicated role, but can be distributed responsibilities(Martini et al., 2014) and is the executor of the detailed design decisions made. Both include the architecture as the main responsibility including the flow of hierarchy of these decisions through the different levels. Britto's(2016) definitions are used from the context of an Ericson case study. The origin of these definitions is based on the role of the architect where system-level architects focus more on decision making, the team-level architect are more team-oriented and help solve problems on that level. This description fits the perception and tasks of ASML's roles as explained before.

Britto et al.(2016)	ASML	Martini et al.(2014)
System-level architect	Department-, Product-, platform architects	Chief architect
Product-level architect	FCA,FCT and Train architects	Governance architect
Team-level architect	<outside scope>	Team architect

Figure 1f: mapping of hierarchal layers

1.3 Research question

As shown in the previous section existing literature provides a solid platform on what to base this research on. One of the previously mentioned limits of existing literature is the origin of existing literature. Literature originates from either existing work breakdowns(Kruchten, 1999) or job vacancies(Ahmed et al., 2015), or when it does involve case studies it focuses on what architecting

activities are(Oliveira et al., 2019). Literature coming from consultancy frameworks like Bredemeyer(Bredemeyer, 2002) or the generic 'great eight' framework based on the work of a consultancy firm(Bartram, 2005)are focused on this point of view too. The case studies that did happen were limited to IT organizations(Hoorn et al., 2011), Oce(Premraj et al., 2011), and Ericson(Britto et al., 2016) from which only Ericson could even be compared to this cases situation. More field research should be conducted in companies.

As mentioned before who the software architect is and what he should do is described. There are sources taking into account behavioral competencies(Bass et al., 2008a; Bredemeyer, 2002; Bredemeyer & Malan, 2002; Clements et al., 2007; Erder & Pureur, 2017; Hoorn et al., 2011; Klein, 2016; Kruchten, 1999, 2008), even though less than 5% of the total architecting literature addresses behavior(Lenberg et al., 2015). However, these sources do not focus on how and why these competencies are important. In addition, even though there are a lot of similarities between the models like the need for intrapersonal skills(Bass et al., 2008; Bredemeyer, 2002; Clements et al., 2007) or communication(Bass et al., 2008; Bredemeyer, 2002; Ferrari et al., 2009; Kruchten, 2008; Razavian & Lago, 2015). But there are also many inconsistencies or differences, as addressed before. This is in combination the rarity of taking different stakeholders or differences in different hierarchal roles(Britto et al., 2016; Martini et al., 2014) into account. This might have more implications considering the future of research should consider the relation of competencies with multiple other perspectives like the organization(Bass et al., 2008) and teams(Kruchten, 1999; Tang et al., 2017). This all is even separated from the lack of knowledge on what it means to be successful as a software architect, within software projects the architect is structurally underrepresented(Agarwal & Rathod, 2006).

Based on the case company goal and following the literature the main research question is formed:

Research question(RQ): ***What are the most important behavioral competencies of a successful software architect?***

We partly also answer the question of why these are most important in a complex D&E environment.

- *SQ 1. Which behavioral competencies are important for a successful software architect?*
- *SQ 2. How do these important behavioral competencies make a software architect successful?*
- *SQ 3. Are there any differences between architects and stakeholders in which behavioral competencies are considered important for a successful software architect?*

1.4 Research methodology

As mentioned in the introduction the role of a software architect is difficult to research since there are few architects within a company, if they even have architects. The large size of the new product development in ASML allows this research to be effective when conducted in ASML. Existing research has shown to have some gaps which led to a wider research question. To answer the research question, this explorative study tries to get to know which behavioral competencies are important but also how they function in practice. To do this a mixed-method approach was taken to answer different elements of the research question itself. First, a literature study has been conducted to gather existing knowledge about the behavior and behavioral competencies of software architects. This literature study maps the

existing knowledge and will form a basis from which hypotheses can be formed. Since literature about behavioral competencies is very limited in the software engineering domain, this systematic literature study follows the research method of snowballing. Snowballing is especially useful in explorative research(Wohlin, 2014), like this study. Chapter two addresses the existing knowledge and hypotheses following the systematic literature study, trying to create a body of knowledge and create an expectation to answer the research question.

Chapter three will address a quantitative research element, in the form of a questionnaire, to get a basic understanding of which behavioral competencies are considered to be more important for software architects. It will help answer sub-question 1 and 3. A quantitative element helps the researcher to create a more objective answer to which behavioral questions are considered important. A questionnaire is an efficient data collection mechanism when the researcher knows exactly what is required and how variables of interest are measured(Sekaran & Bougie, 2016). In addition, it also helps to reach a wider audience. Even though in this research the quantitative element is based on human evaluations, the 'wisdom of the crowd' principle still creates a reliable way to answer the question "which" behavioral competencies are most important. The 20 behavioral competencies(Kornferry, n.d.), overview in appendix A, used within ASML form the base of this questionnaire. Participants will be familiar with these concepts and definitions. Where the data collected from a questionnaire could also help find differences and similarities between groups, it does not answer the 'how' and 'why' questions.

The how represented in sub-question 2, can be answered with a questionnaire next to also verifying the answers given in the third chapter on which behavioral competencies are important for a successful software architect(SQ1). The drawback of a questionnaire is the inability to find in-depth reasoning behind the answer. To answer this 'how', semi-structured interviews are used to elicit in-depth information to identify the critical factor(s) and fill the knowledge gap(Sekaran & Bougie, 2016). Alternatively focus groups can be used for exploratory studies aimed at generating generalizations to answer the 'why'-question(Sekaran & Bougie, 2016). However, due to restrictions given by the company regarding the time the option of focus groups is not used.

Every chapter in this report will elaborate more upon the specific methodology used, followed by the results of the chapter itself. It will follow the structure as described above and is visualized in figure 1g.

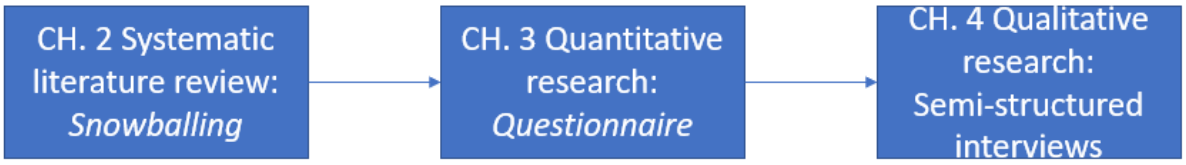

Figure 1g: research methodology overview

Chapter 2 Theory

This chapter will address the conducted literature review forming the existing body of knowledge. In this research setup, the literature study itself was part of a preliminary study conducted to build the rhetoric of this study. As a result of the search for knowledge to create an understanding of what makes a software architect successful, the primary goal of this thesis is to expand the body of knowledge on behavior competencies for software architects and use this knowledge to further the utility of behavioral competencies in practice. First, this chapter will address the methodology of the literature study and will then address the outcome.

2.1 Methodology

Based on the problem context for the case company, the goal of this systematic literature review is to explore the body of knowledge and identify what behavioral competencies are important for a software architect. However, the term behavioral competency is not widely used in the software engineering domain. Luckily, the actual goal was not only to gather specific behavioral competencies but to gather the different non-technical elements or behavioral elements of the software architects and their operationalization. The aim was to gather knowledge and find specific gaps in the literature in regards to the role and behavior of the software architect. This was a clear first step in answering the research question: which behavioral competencies are important for software architects. This led to the following literature review research question:

LR-RQ: What are the gaps in the literature regarding the role and competencies of a software architect?

To answer the literature research question four sub-questions were formed to guide the findings :

LR-*SQ1:* How is the role of the architect currently described in the literature?

LR- *SQ2: What are the existing competencies of software architects?*

LR- *SQ3: How are these competencies operationalized?*

LR- *SQ4: How do the SA competencies help create success?*

2.1.1 Snowballing

There are two common methods of literature research: systematic literature reviews and systematic mapping studies. Search queries, the often choice for literature reviews, require clear and well-defined strings, and snowballing, a systematic mapping form, requires a starting set of papers from which other sources and citing articles are reviewed. The snowballing method is especially suited for exploratory studies(Wohlin, 2014), like this one. In addition, forming good search terms for behavior in software architects proved to be difficult due to the small focus on behavioral research in the software engineering domain(Lenberg et al., 2015).

Wohlin(2014) describes a method for the snowballing methodology, as shown in figure 2a. The first step is to identify a start set of papers. From the existing literature we know that the body of behaviorally oriented articles in the software engineering domain is very limited(Lenberg et al., 2015), let alone trying to take into account the software architect specifically. This is confirmed by searching in search engine

Scopus for terms like ((“competency” OR “competencies”) AND “Software architect”) or (Competency AND software AND architect) yielding a respective three and eighteen results from which none would have been included based on the search criteria. Webster & Watson(2002) suggest selecting the set of papers from leading sources. Initially, this research concept originated from the idea of (Kruchten, 2008), (Razavian & Lago, 2015) and (Mendes et al., 2021). Kruchten is one of the leading authors concerning software architects and this research is conducted with inspiration from Razavian & Lago’s(2015) perspective of linking the central role of the software architect to different types of experiences a person can have. The article of Mendes et al.(2021) is a recent, but leading, publication focused on the relationship between decision-making styles and personality in the software engineering domain. Primarily, this was included because of the linking of personality into a more behavior-like topic in the software engineering domain.

The second step is the snowballing procedure itself, Scopus was used as the search engine for this research. First of all, snowballing is an iterative approach analyzing the source list and citing articles where each iteration is expected to experience a fall-off in relevant articles(Wohlin, 2014). In every iteration the source list of each new primary source is evaluated for potential relevant sources, this process is rereferred to as backwards-snowballing. In addition, forwards snowballing, the process of evaluating the citing articles, is used similarly to the backwards-snowballing. The initial considerations are based upon the title, reference place, and abstract where the content is judged on selection criteria(which will be elaborated upon in the next paragraph). The final decision is based upon the full paper and iterations will stop if no new papers are found.

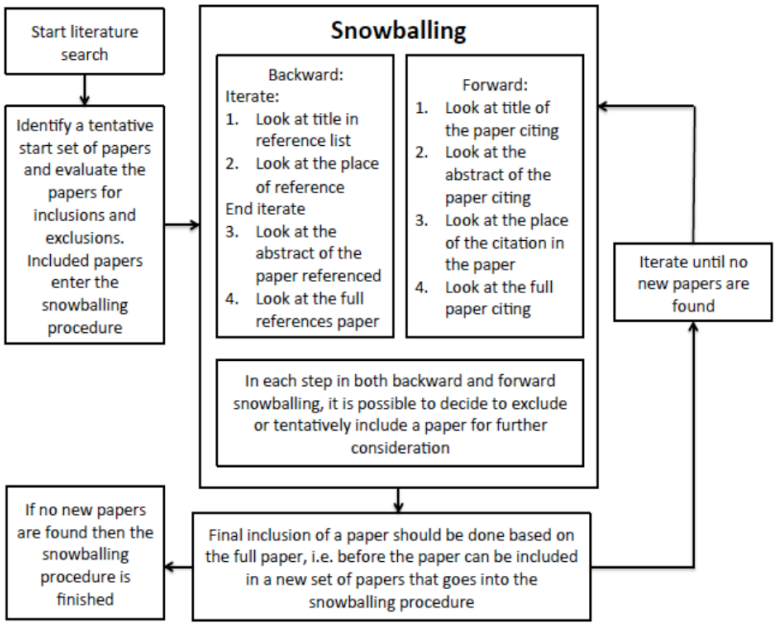

Figure 2a: snowballing procedure(Wohlin, 2014, p.4)

2.1.2 Selection criteria

In the iterative process, the scope started to differentiate between findings about the software architect and findings in the software engineering domain. Since the software architect is part of the software engineering domain, articles from this domain do not always (fully) apply to the software architect. In addition, there were some subjects, like decision making, which were not included in the scope. Snowballing requires a selection procedure, to make the selection process more rigorous selection

criteria are formed. However, there are primary articles focusing on the behavioral competencies of a software architect or direct elements of this. But there are also potentially useful articles addressing the software engineering domain or other aspects outside of the scope like decision making. To distinguish between these there direct literature is identified as primary, while other (relevant) literature is classified as supportive literature.

The selection criteria, overview in table 2a, are based on the research question, research goal and are a result of the iterative research within literature. For example, even though the software architect is placed within the software engineering domain the article of Mendes et al.(2021) and articles following Mendes were considered supportive. Where the specific focus on decision making could yield interesting findings it was not in line with the scope of this research, thus it was considered supportive. The first criteria, I1 and I2, are formed due to the basic demands of research trying. The research question tries to focus on software architects and the literature review tried to find the existing behavior and role of the software architect. These are represented in I3, I4, and I5. Decision-making was a difficult topic since some articles referred to decision-making as part of the role of the software architect, but others referred to it as a more technical element. In addition to this distinguishment, some articles addressed not the software architect but addressed the software engineering domain, focusing on the generic domain, software teams of software project success. Naturally, the software architect is part of the software engineering domain., however, if sources do not specifically distinguish the software architect it is uncertain if these also fully apply to the software architect. The findings of these studies could be valuable but the research scope specifically focuses on the software architect. To distinguish these sources addressing the software architect indirectly while still fulfilling the previous requirements are included as supportive sources. However, this requires exclusion criteria to balance what isn't included. The first two exclusion criteria, E1 and E2, are based on the practicality of research. The third exclusion criteria are based upon the previous inclusion of other sources from the software engineering domain and other domains which directly refer to the behavioral aspects found to be important for a software architect. One additional difference which has to be made is the exclusion of articles that address technical elements, the research question specifically explored the behavioral competencies and not the technical competencies. These sources would address the architecture of systems and/or sub-systems. The differentiation between primary and supportive articles can be found in appendix B.

Table 2a: Systematic literature study selection criteria

Inclusion criteria:	Exclusion criteria:
I1. Studies and sources published in academic journals and conferences but also book chapters and publications **I2.** Sources are written in English **I3.** Articles referring to the role, duties, or behavior of software architect **I4.** Articles referring to decisions made by and behavior of software architect when referring to software architecture **I5.** Articles referring to behavior in software engineering in which software architects are included as supportive sources(decision making, project management, etc.)	**E1.** Duplicates or later versions of the same article **E2.** Articles that were not available or traceable for full-text review **E3.** Articles outside the software engineering domain that do not address any specific **E4.** Articles addressing the architecture of systems or subsystems

2.1.3 Iterations and evaluations

The snowballing procedure finished in the third iteration where no new papers were found, the split between primary and secondary articles is represented in Appendix B. As mentioned before the results are evaluated in two ways. In the first check, there is no reason to doubt the findings of this literature study as it did not seem to contradict existing systematic literature reviews from (Lenberg et al., 2015) and (Baltes & Diehl, 2019). In general, many excluded articles did have the technical focus and the included articles followed the perspective of being written from a technical perspective. However, the role of the architect has seen discussion in literature and these articles do include behavior or elements of the behavior of a software architect. Using google scholar, three different search terms were used from which the first 50 results were screened. The search terms were a spread ranging from the most basic scope of the software architect to excluding technical sources focused on architecture and sources focusing on enterprise (architects). Just short of half of the primary sources show up in these results, confirming at least having touched a relevant part of literature regarding the software architect and its behavior.

Table 2b: Evaluation based on searching another search engine

Search term	Included sources
(software architect)	16
(“Software architect”) AND (Competencies OR competency OR skill)	13
(“Software architect”) AND (Competencies OR competency OR skill) NOT(Architecture OR Enterprise)	15

2.2 Academic literature

As mentioned before the final articles which were included as starting articles were the preliminary studies of Razavian & Lago(2015) and Kruchten(2008) supported by the software engineering study of Mendes(2021).

2.2.1 Software engineering domain

Research in the software engineering domain has been trying to introduce behavioral elements in their studies. The field of behavioral software engineering(Lenberg et al., 2015) and existing literature of software engineering only explored a limited number of these behavioral elements leading to success on a team, project, or organizational scale(Baltes & Diehl, 2018). Less than 5% of the research has been focusing on behavioral elements(Lenberg et al., 2015). Even though the software architect is part of the software engineering domain, logically their behavioral elements have also not been well researched. Both industry and existing literature mainly focused on technical aspects of software architecture(Muccini et al., 2018). There is literature on what a software architect should do(Hoorn et al., 2011; Kruchten, 2008; Kruchten, 1999; Clements et al, 2007; Bass et al., 2008; Bredemeyer, 2002), should know (Bredemeyer, 2002; Hoorn et al., 2011; Bass et al., 2008; Martini et al., 2014) and a start on who they are(Erder & Pureur, 2017; Kruchten, 1999; Klein, 2016;Bredemeyer, 2002; Bredemeyer & Malan, 2002). All literature agrees that the main responsibility of a software architect is the system's integrity (Britto et al., 2016; Eeles, 2006; Bass et al., 2008), however there are also some arguments of an additional mentor(Britto et al., 2016) or leadership Bass et al., 2008) role for the software architect.

The literature lacks explanation on different types of software architects, what if this could explain the differences between literature? The role of software architect could be split up using three hierarchal levels(Britto et al., 2016;Martini et al., 2014). Other studies often assume one type of software architect, a clear gap in the literature, which negates the different roles and expectations attached to different hierarchal levels. Britto et al.(2016) separates a system-level, product-level, and team-level architect based upon what they refer to as being a guardian or a mentor in combination with being a decision-maker or a problem solver as was defined by Fowler(2003). Martini et al.(2014) created a framework where a chief, governance, and team architect were separated with responsibilities and focus for each role. Both sources can be seen as complementary since they don't oppose each other's fundamentals. In addition to hierarchy, there could also be differences in the specific role and responsibilities of a software architect since the role of a software architect could be executed by different people (Eeles, 2006). Incorporating these different types of context and types of architects will fill multiple gaps in the literature.

With a wide range of 'flavors' of software architects' context, it becomes even more important in conducting research. Existing literature about competency, skills, or knowledge has often analyzed the software architect from the point of view of the role and task descriptions(Kruchten, 2008; Bass et al., 2008) or job offers(Ahmed et al., 2015). When looking at the behavior, skills, or competency existing research either is set in a case study (Britto et al., 2016) or based upon a consultancy framework(Kruchten, 2008; Razavian & Lago, 2015). All these previous options look at what is being done and not which competencies are actually leading to success or are desired for success and how they do this.

2.2.2 Behavior and the software architect

Kruchten(2008) identified the key components of the role of the software architect as architecting activities, inwards communication, and outwards communication in a 50/25/25 split. However, there are also some undesirable behaviors of software architects represented in four antipatterns. These antipatterns are identified as creating the perfect architecture for the wrong system, creating the perfect architecture but too hard to implement, architects in their ivory tower and the absent architect(Kruchten, 2008). Each of these groups show different behavior in one of the key components of the role of the software architect. However, there is a lack of understanding on what causes these antipatterns or how they can be avoided. There are (feminine) expertise's linked to the different components, the expertise's intuition, embracing ambiguities and exploring problem and solution space and are linked to the architecting component(Razavian & Lago, 2015). However, these types of research are few in number, in line with findings of behavioral software engineering research(Lenberg et al., 2015). To fill the gap in understanding the antipatterns of software architects and why they occur the behavioral elements have to be explored. These behavioral elements are separated from the technical elements, thus are sometimes referred to as non-technical.

This is even more important when considering the social element of the role of the software architect. The relation between software architecting and organization has been mentioned before (Bass et al., 2008). An interesting element is a lack of understanding in the existing literature about what is meant with success or being successful for software architects. While there is literature about what entails

success in software projects(Agarwal & Rathod,2006[s]), no research has been conducted on what this means specifically for a software architect. The whole social side of the 'technical leader' the software architect can include roles like mentoring(Britto et al.,2016; Bass et al.,2008) but also leadership and management(Bass et al., 2008). However, more often literature mentions the need for communication(Clements et al., 2007; Kruchten,2008, Razavian & Lago, 2015), collaboration(Sherman et al., 2016), and political maneuverability(Bredemeyer, 2002). Some see the role of software architect as a joint decision-maker (Rosa et al., 2020[s]; Tang et al., 2017). Razavian & Lago(2015) address the importance of diversity within software teams in which they separate feminine competencies relevant to the role of the software architect. Dyba et al.(2014) needs leaders/managers to shape and create an organizational context that can support clusters of competencies to be used by the organizations' projects. Software decisions are often made in group environments, it requires an understanding of how software professionals in groups invoke knowledge in their communication, reasoning, and decision making(Tang et al., 2017). A team, in this research, is defined as: *"a small number of people with complementary skills who are committed to a common purpose, performance goals and approach for which they hold themselves mutually accountable."*(Katzenbach & Smith, 1993)

Behavioral competencies contribute to for example joint decision making(Rosa et al., 2020[s]), communication(Kruchten, 1999;Kruchten, 2008; Oliveira et al., 2014). What happens more is the mention of competencies being important due to the fit with the task description of the software architect(Kruchten, 1999; Bass et al, 2008; Clements et al., 2007; Kruchten, 2008; Razavian & Lago(2015). Bass et al.(2008b) see a key role for the organization to stimulate and coordinate competencies. The mention of software architecting teams(Kruchten, 1999), the role of software architect being executed by multiple roles(Eeles, 2006), many of these sources describe a connection between competencies and an organization, a team, performance, or a project. But very few address how they do this and what they influence.

2.2.3 Competency

One way existing way to represent behavioral elements of a role is to use the term competency(Bredemeyer, 2002; Bass et al., 2008). As cited by Bass et al.(2008): "A **competency** is the set of behaviour patterns that the incumbent needs to bring to a position in order to perform its tasks and functions with competence"(Woodruffe, 1993, p.29). This definition requires an additional definition where **competence** is seen as "Being competent, adequacy, possession of suitable or sufficient skills OR creating valuable results without using excessively costly behavior." (Bass et al., 2008, p.5). This will be the definition used in this report. Using this definition means skills would be included under competency since it is part of the competence definition. A software architect needs to have both domain expertise as well as software expertise(Kruchten, 1999; Oliveira et al, 2019; Bass et al., 2008), this would at least suggest the need for knowledge to have competence.

A software architect is seen as the technical leader or decision-maker(Fowler, 2003; Britto et al., 2016; Eeles, 2006) with the additional mentor or leadership role. Within these sources, there are various views of different competencies. Many sources include communication as competency in different forms(Kruchten, 1999; Oliveira et al., 2019;Clements et al., 2007; Kruchten, 2008; Bredemeyer, 2002; Razavian & Lago, 2015) and split communication into outward, inward communication(Bass et al., 2008;

Clements et al, 2007) or getting input & providing information(Kruchten, 2008; Razavian et al., 2015). Sometimes in general(Bass et al., 2008) or both(Clements et al., 2007) are separated.

Oliveira et al.(2019) claims that the roles, responsibilities, activities, and tasks performed by software and system architects are still largely unknown and diffuse in organizations. A bold statement with the existing literature and consultancy work focusing on these tasks and responsibilities. Hoorn et al.(2011) separate 5 different architecting activities: Communication, decision making, quality assessment, documentation, and knowledge acquisition. Which are explored with the supportive methods of decision management, search efficiency, community building, intelligent advice, and knowledge management
Leadership skills(Kruchten, 1999; Eeles, 2006; Clements et al, 2007; Bredemeyer, 2002) which is referred to as technical leadership(Kruchten, 1999; Eeles, 2006). For Bass et al.(2008) leadership skills is an element of work skills alongside Effectively managing workload(Bass et al., 2008; Clements et al, 2007), excelling in a corporate environment(Bass et al., 2008; Clements et al, 2007) and skills for handling information(Bass et al., 2008; Clements et al, 2007). Bredermeyers's (2002) framework would add team context(vision), decision making, team building, and motivating others to this dimension of leadership. Interpersonal skills can be split between 'within team' and 'with other people' (Bass et al., 2008; Clements et al, 2007). Razavian & Lago(2015) add intuition, exploration of problem and solution space, and embracing ambiguities as non-technical competencies. The exploration of problem and solution space and the embracing ambiguities are also present in the Bredemeyer(2002) framework and the skill and knowledge (Bass et al., 2008).

2.3 Important behavioral competencies

As mentioned before the body of literature provides for a baseline in knowledge. However, the literature review helps answer the research question by providing knowledge that can help create hypotheses. Hypotheses can be tested in the different methods. Each of the sub-chapters will form hypotheses focused on addressing a specific gap in the literature.

2.3.1 The most important competencies

This chapter will interpret the results from the literature study to form hypotheses. This sub-chapter will address the expectations focused on the gap in which behavioral competencies can be considered important.

2.3.1.1 Communications

From prior literature(Bass et al., 2008; Clements et al., 2007; Kruchten, 1999, 2008; Razavian & Lago, 2015) being able to communicate effectively with a diverse set of stakeholders has been identified as one of the most important behavioral competencies for SW architects. Kruchten (2008) argues the relevant balance of an optimal software architect to be 50 percent architecting, 25 percent inwards focus(getting input) and 25% outwards focus(providing information), for both of which effective communication is a key parameter. In the definition the inwards focus is defined as getting input from the outside world: listening to customers, users, product manager, and other stakeholders (developers, distributors, customer support, etc.)(Kruchten, 2008). Learning about technologies, other systems' architecture and architectural practices." Outwards focus is defined as "providing information or help to other stakeholders or organizations: communicating the architecture: project management, product

definition." Within these definitions listening to customers, users, product managers, and other stakeholders is emphasized in combination with being able to present the architecture to a multitude of stakeholders. This seems to fit the definition of the behavioral competency '***Communicates Effectively***': "*Developing and delivering multi-mode communications that convey a clear understanding of the unique needs of different audiences".* (Kornferry, n.d.). Clements et al.(2007) see external communication, internal communication, and general communication skills as sub-groups of communication skills that are identified as important for an architect. In addition to this interacting with stakeholders is seen as one of the key duties of a software architect(Bass et al., 2008)where the different stakeholders are seen as clients, developers, and others.

Communication is present in various forms, and it is uniformly seen as an important part of the software architect(Bass et al., 2008; Bredemeyer, 2002; Clements et al., 2007; Kruchten, 1999, 2008; Oliveira et al., 2019; Razavian & Lago, 2015). This creates the expectation that communicates effectively will be identified as one of the top 3 most important behavioral competencies for SW architects both by architects themselves as well as by their internal stakeholders.

Hypothesis 1 The behavioral competency "communicates effectively" is one of the top three behavioral competencies identified as instrumental to be a successful software architect by SW architects

2.3.1.2 Decisions and gathering knowledge

One of the key tasks of a software architect is maintaining the architectural integrity of the system(Bass et al., 2008; Clements et al., 2007; Fowler, 2003; Kruchten, 1999, 2008). In an ideal situation, the architect should be architecting(design, validation, prototyping, documenting, etc) 50% of the time. The software architect is considered the technical leader(Bass et al., 2008; Britto et al., 2016; Kruchten, 2008), however, being a technical leader is ultimately responsible for the technical elements. One of the views was the software architect being an important decision-maker(Fowler, 2003), others saw a clear role in joint decision making led by the software architect((Mendes et al., 2021). Minimum viable architecture, facilitate decision making, experience, and knowledge of using the relevant tools at the appropriate time, deal with ambiguous contexts can all be seen as part of the decision-making process of a software architect(Erder & Pureur, 2016). In many cases, decision-making seems one of the clear-cut behavioral competencies of a software architect.

With the core activity of a software architect being architecting itself in which decision making seems very important, the content and impact of these decisions should naturally be important. This fits the description "*Making good and timely decisions that keep the organization moving forward*", the definition of the behavioral competence '**Decision Quality**'.

Software projects get wicked problems, but wicked problems have no good or bad solutions, there is no single formula (van Vliet & Tang, 2016). In consequence, the requirements change too and become more difficult to satisfy(Spinellis, 2016). The one responsible for these requirements: the software architect. One of the key skills emphasized are decision making(Tang et al., 2017) and problem-solving (Bass et al., 2008; Clements et al., 2007). Much existing research on the software architects focuses on decision-making (Lenberg et al., 2015; Muccini et al., 2018; Tang et al., 2017). Weinreich & Groher

(2016) argues the team role of the architect which is not only a decision-maker but also a knowledge manager. There is some form of irreversibility within architecture that drives complexity (Fowler, 2003). This is a logical consequence when considering software architecture itself as the structure or structures of a system that is comprised of software components, the externally visible properties of those components, and the relations among them(L. Bass et al., 1997). It encompasses the set of significant decisions about the organization of a software system i.e. selection of structural elements and interfaces, behavior regarding the collaborations of those elements, and the composition of these structural and behavioral elements in a larger subsystem. In these complex structures, a natural need for expertise in exploring the problem and solution space or embracing ambiguities(Razavian & Lago, 2015) arises.

As explained previously, the need for the nature and role of being a software architect is highly intertwined with having, managing, and understanding knowledge and information. The intensity of information on these subjects indicates the importance of '**Manages complexity**', defined as "*Making sense of complex, high quantity, and sometimes contradictory information to effectively solve problems"*.

In addition to making sense of the complex information, Bass et al(2008) and Clements et al(2007) would also emphasize the handling of the unknown factors and unexpected developments as two additional important skills specifically for a software architect. These situations require problem-solving and analytical skills while being adaptable and flexible(Bass et al., 2008). Reacting to unknown and unexpected situations indicate the need for a certain degree of adaptability, Oliveira(2019) would even argue the continuous change of the role of software architect because of this. Creating the expectation of '**situational adaptability**', defined as "*Adapting approach and demeanor in real time to match the shifting demands of different situations.*", which is important for being a successful software architect.

Hypothesis 2: The behavioral competency "Manages complexity is one of the top three behavioral competencies identified as instrumental to be a successful software architect by SW architects

Hypothesis 3: The behavioral competency "Decision quality" is one of the top three behavioral competencies identified as instrumental to be a successful software architect by SW architects

2.3.2 Including hierarchy

One of the gaps was focused on only a few sources including hierarchal differences for software architects. One thing became clear there is a difference between hierarchal layers of software architects(Britto et al., 2016; Martini et al., 2014), but does this also result in different behavioral competencies being important? This is a relevant topic due to the industry interest in what is required in each layer.

First of all, the definition of Britto et al. (2016) is more similar to the definitions used by the case company. Seeing the system-level architect as someone who has to tie multiple products together to maintain the system(Britto et al., 2016) fits well with the roles of department architect, product architect and platform architect. As shown in chapter one these roles are focused more on the high-level design and decisions. The product-level architects would be considered the functional architect(FCA) and functional test architect(FCT) who are responsible for the execution of the different high-level

design decisions while still being involved in the decisions themselves. Sub-function architects would be considered as team-level architects, directly being involved with the teams.

This research scoped down to the system-level architect and the product level-architect, changing the question to are the important behavioral competencies different between system-level architects and product-level architects? The main difference between these is the level of design decisions taken(Britto et al., 2016; Martini et al., 2014) but the product-level architects could also be seen as the problem-solver solving problems together with the engineers(architectus oryzus)(Britto et al., 2016). Previously the mentor and leadership role of the software architect was addressed(Bass et al., 2008; Bredemeyer, 2002; Britto et al., 2016).

With the main responsibility of every different hierarchal architect being rooted in system integrity and maintainability(Britto et al., 2016) the aim of every layer would be the same. In addition, every layer is trying to enforce design(Britto et al., 2016; Martini et al., 2014). When looking at the core of the software architect there is the architecting, getting input, and providing information(Kruchten, 2008). To maintain system integrity, even though this happens with a different focus, all these steps are required from every software architect. With the system-level and product-level architects, both are required to gather information and provide information. Even though product-level architects might be placed closer to the engineering teams themselves, the system-level architects still have to gather information and provide information. Given, this would be from and to a different spectrum of stakeholders, the required skillset and the same behavior is expected from both layers of architects. In addition, both groups work together often and are depending on the other group in regards to how the system integrity is maintained(Britto et al., 2016; Martini et al., 2014). The similar core dynamics in combination with the close working together does create interest in the difference between how system-level architects and product-level architects rank the important behavioral competencies. With a potential difference and the difference having relevancy in practice since there are distinguishable layers(Britto et al., 2016; Martini et al., 2014). This study should explore the potential differences between the different hierarchal layers of architects, resulting in the following exploration:

Exploration question 1: Are there any differences in which behavioral competencies are seen as important between the system-level software architects and product-level software architects?

2.3.3 Stakeholders

One of the gaps in the literature is considering what is important only from the perspective of the software architect. This research specifically includes the stakeholders. Even though literature does not include stakeholders some concepts suggest potential differences in what is considered important for a software architect. Literature referred to the architect with its role as a decision-maker(Britto et al., 2016; Eeles, 2006; Fowler, 2003) but also as a joint decision-maker(Rosa, 2020). In addition, the antipatterns show the need for an architect to be involved and not rule from its ivory tower(Kruchten, 2008). If anti-patterns exist and are founded in how much they balance gathering input, architecting, and providing information the consequences of these antipatterns should also affect someone to be perceived as ‘negative’. From the previously mentioned

One of these antipatterns was the architect in the ivory tower who is mainly focused on the architecting itself(Kruchten, 2008). One of the other antipatterns is being just a consultant, which relates to the architect not architecting but mainly providing others with information(Kruchten, 2008). The final antipattern is the goldplating where the architect lacks the balance in regards to not giving enough information to others(Kruchten, 2008). All these things have one thing in common, stakeholders feel the consequences of an architect not behaving as expected. One behavioral competency which impacts stakeholders more than architects themselves would be 'decision quality'. If decisions are not made at the right moment or the wrong decisions are made the ones who have to deal with this are the ones making the software. This could be the reason for a different perception between stakeholders and architects about the behavior competency decision quality where stakeholders would find it more important than architects.

Hypotheses 4: It is expected that stakeholders will rank the behavioral competency 'decision quality' higher compared to software architects.

Architects depend on others when gathering input it being 25% of the activities of a software architect(Kruchten, 2008), if others do not provide input there is not much to architect. There is a reason why interpersonal skills both within the team(team player, balanced participation, and working effectively with superiors, colleagues, and customers) and with other people(being diplomatic and committed to others' success) are considered important for a software architect(Bass et al., 2008). The architect depends on others for at least 25% of their activities in the form of gathering input(Kruchten, 2008) and requires interpersonal skills to work well with their team and others. This is expected to create a higher sense of importance from architects towards behavioral competencies which make them dependent on others. The behavioral competency closest to this is 'collaboration' is most related to the architects depending on others where they have to work together. This would create the following hypotheses:

Hypotheses 5: It is expected that software architects will rank the behavioral competency 'collaboration' higher compared to stakeholders.

Within the stakeholder group, there are also various types of stakeholders. The Oce-canon group case showed the existing roles in Oce where above every team there was in a triangle with a project management and a system integrator(Premraj et al., 2011). In addition to this, every team had an architect with software developers under the architect. As previously shown in chapter 1, ASML has similar roles with architects being placed in the 3-in-a-box with a long-term and short-term stakeholder.

As with the architect, there being multiple types of stakeholders could give different perceptions on what is important in a software architects behavior. The software engineers under an architect might have different expectations since their dependency is different than that of a project manager. Just like hierarchy being present in software architects, hierarchy is in place with stakeholders. Due to their difference in positions, there could be different opinions. In line with the previous exploration, there is not enough evidence to support a hypothesis. There is enough indication to want to explore the different stakeholders. Next to the management roles and the engineers similar to the Oce case study(Premraj et al., 2011), ASML also has roles related to their agile structure. This would result in

three categories of stakeholders to be observed: management, agile-roles, and engineers. This leads to the following exploration:

Exploration question 2: Are there any differences in which behavioral competencies are seen as important between the different stakeholder sub-groups: management, agile-roles, and engineers?

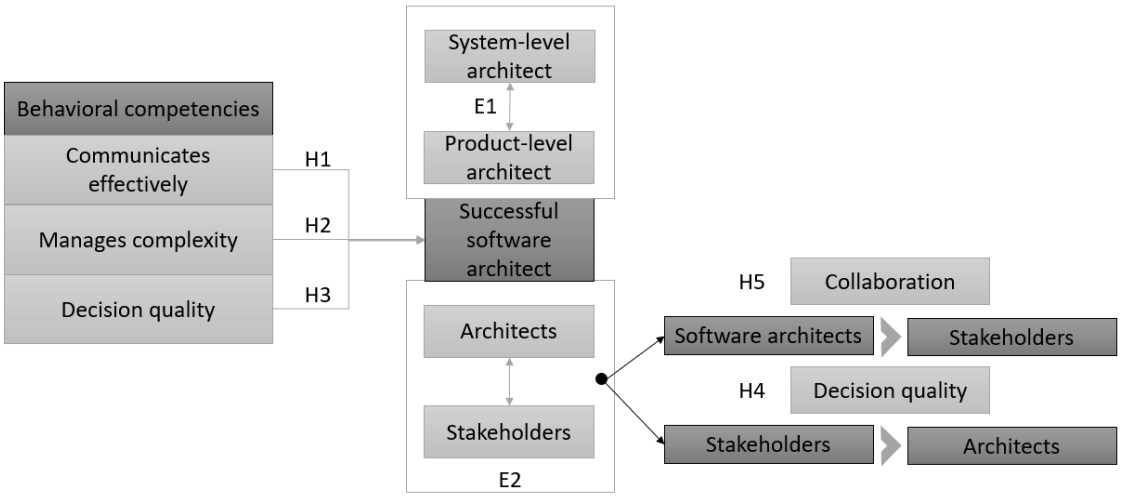

Figure 2b: Overview research model

2.4 summary findings literature review

In conclusion, this literature study tried to analyze the existing body of literature regarding the software architect. A complete overview of the included hypotheses and exploration topics can be found in figure 2b. It included the perceptions of the role of the software architect but also defined what is considered a (behavioral) competency. It expects to find communicates effectively, manages complexity and decision quality as an answer to the research question. Furthermore, no differences are expected between hierarchal layers, however, there are expected preferences where architects are expected to value collaboration over decision quality. Stakeholders would have this preference switched.

Chapter 3: Quantitative research: Questionnaire

This chapter will discuss the methodology and results of the questionnaire. The quantifiable data will be used to answer the research questions and test the hypotheses.

3.1 Methodology of the questionnaire

A questionnaire is an efficient data collection mechanism when the researcher knows exactly what is required and how variables of interest are measured(Sekaran & Bougie, 2016). The behavioral competencies which are used by ASML are known and people are expected to be familiar with them. In addition, a questionnaire will be able to reach a wider audience of various roles. Questionnaires also can produce more responses increasing the quantifiability of any results. Questionnaires will be used to answer which behavioral competencies are considered important for software architects and by whom. However, questionnaires are less suitable for research where it is unknown 'why' behavioral competencies are important and how successful is perceived. The questionnaire should provide the right balance between validity, reliability and practicality(Blumberg et al., 2011). The creation of the questionnaire will follow the principles of Blumberg et al. (2011).

3.1.1 Goal of the questionnaire

There are a large number of software architects in ASML, this allows for quantitative research. The goal of this questionnaire is to gather 'opinions' about which of the 20 behavioral competencies are important for software architects. With this quantity, it will allow for analyzing the outcome in regards to the set hypotheses. Hypotheses 1, 2, and 3 are straightforward interested in a ranking, the analysis should produce evidence if Communicates effectively, Manages complexity and decision quality are indeed the most important. Hypotheses 4, 5, and 6 will require the comparison of the behavioral competencies between multiple groups. Hypotheses four will require evidence if there is any difference in the ranking of important behavioral competencies between system-level architects and product-level architects. Hypotheses 5 and 6 will compare architects to stakeholders and should try and prove the expected difference in preference over which behavioral competencies are important. Any other findings will be addressed too.

3.1.2 Design of the questionnaire

The full questionnaire can be found in appendix C. Overall this research will follow the guidelines set by Blumberg et al. (2011) asking administrative questions first, then classification questions, and finally the target questions. The questions should be able to answer the hypotheses. Numerical scale type of questions provides both an absolute measure of importance and a ranking(relative measure) of the items rated(Blumberg et al., 2011, p.362), allowing for both being able to compare and rank values. It does require participants to answer to which of the previously set categories they belong to. The numerical scale questions will have equal intervals separating the numeric scale points. These types of questions will produce quasi-interval data. Numeric scales often have 5-point, 7-point or 10-point scales(Blumberg et al., 2011). With these questions, a clear middle point preferred ruling out the 10 point scale. However, the 5 point scale risk answers finding everything important more than a 7-point scale. Thus 7-point scale questions about the 20 different behavioral competencies will be used in the questionnaire. To mitigate the risk of people not being able to make a choice or not knowing the

difference between a score of 6 and an equal score in another category, an evaluative question is created. This forced-choice question will require the participant to select a top 3 of most important behavioral competencies.

The data-collection instrument should avoid poor selection of content and should not be confusing or ambiguous for participants(Blumberg et al., 2011), thus creating the requirement of practicality and comfort for participants. For this reason, the expected duration of the questionnaire should be under 10 minutes, as was required by the case company. In addition, a final question was added. This question functioned as a backup however would allow participants to freely describe any (additional) details regarding important behavioral competencies. One final step before the usage of these survey results is pre-testing(Blumberg et al., 2011) evaluation on clarity, time restriction, and any other problems. The questionnaire was tested with stakeholders from three different hierarchal layers. Based on feedback the introduction and questions were updated. There were no major updates outside of clarifications.

3.1.3 Participant selection

In the introduction, the different types of software architects were introduced. This research targets all of them, in this case department architects, products architects, and train architects working within a software department or with software products will be considered software architects. The identified different stakeholders working with software architects were Agile-roles(internally referred to as SAFe roles), R&D management, product management, customer support related roles, and developer or competence engineers. All different types of management are classified as management stakeholders. The SAFe roles are classified as the middle layer representing agile roles. The developer or competence engineers will hierarchically be seen as the lowest level stakeholder and will be referred to as engineer-level stakeholders. Repeating the design-choice that in these stakeholder categories the team-level software architects are included in the developer or competence engineer category.

Participants who fall either in the architect of stakeholder categories are approached as long as they are part of the departments located within the MX cluster. There could be a risk where different departments could have different cultures. However, since these departments have the same structure and sending it to multiple departments would create a larger sample size outweigh the risk of cultural differences. As mentioned before the amount of software architects within companies is considered low, making it hard to do quantitative research. In this case, there are more than 1000 software developers from which 161 responded, 59 architects responded. Within the questionnaire, there are two tracks, one for software architects and one for non-software architects. Types of software architects were obtained from internal information, types of non-software architects were formed following internal recommendations. Questions were reformed depending on the point of view, i.e. instead of asking ‘how long have you been working as software architect?’ non-software architects were asked ‘how long have you been working with software architects?’. The age brackets were adapted following internal usage. Participants under 18 and people not giving their informed consent cannot be included in this research and will be referred to the end of the questionnaire.

3.1.4 Measurements

This chapter will discuss how the information gathered can be transformed into the theoretical section(Blumberg et al., 2011), and its limitations. It will also explain how the statistical tools help measure.

First, success was not defined but rather implied by where it reflects the perception of the participant. The evidence here would follow the 'wisdom of the crowd' principle where if a majority of participants would grade something a certain way, this would be interpreted as the truth. This holds up due to the large sample size in this context(220 employees in the same context) in an explorative research setting were creating a base is the goal. All hypotheses can be observed visually via a ranking. Even though this does not tell anything about the statistical difference, it is valuable information that could form the argument in favor of arguing differences in rankings and between the ranking of different groups. This ranking is based upon the 1-7 Likert scale. IBM SPSS will be used as the main tool for analyzing the data, however, in one instance R-studio was used to cluster the variables of the dataset. Next to tools, the main focus will be on deduction and logic.

3.1.4.1 Method 'most' important claims

The first three hypotheses claim certain behavioral competencies to be the most important. There are two ways to show this. First, the behavioral competencies rank the highest. This method is simple yet gives the sense of priority, however, as mentioned before it does not say anything about the statistical differences. Even if statistical tests do not deem a certain competency significant, the relative rankings can still be used to argue interest in a variable. Since the focus of this research is based upon the important behavioral competencies the scope will be on the upper segment.

To statistically show the difference between hypotheses 1, 2, and 3 one assumption is made where 'most' important variables should be clustered or at least statistically not different from each other. Alternatively, if they would not be clustered together, only the top clustered would be seen as 'most' important. **This does not mean the others are not important but could indicate gradations in importance**. Classic methods like the t-test could do this, however, there would be a lot of tests required to analyze the same dataset. This drawback of the appropriate paired t-test is that it functions in a way where there is a 5% chance error acceptance in the test. If too many tests would be done this would result in a huge combined bias. In the software engineering domain, there is a tool that automatically clusters the results based on the mean, the Scott Knott ESD. The Scott-Knott Effect Size Difference(ESD) leverages a hierarchical clustering by transforming the treatment means into statistically different groups(Tantithamthavorn et al., 2018).

The results from the statistical test will show clusters, each individual expectation will be checked if they are in the top segment. If there are multiple clusters near the top, they will be discussed, and depending on the data, they might be considered important.

3.1.4.2 Comparing groups

The next topics are all regarding group comparisons. Once again, the first step is observing the relative rankings, if there are differences in these rankings they will be discussed provide indications. To know if these differences are statistically different they should be compared.

The groups are created of separate groups. Normality can be assumed based on (large) sample sizes(Field, 2013) and the number n=30 has been used before however behavioral research rarely shows normal distributions eluding the results not to be normally distributed and some subgroups do not fulfill this requirement of n>30. Using the Sharpino-wilk test as an advised test for normality(Field, 2013), none of the variables were indicated to have normality. Since this would violate the normality assumption of independent t-test an alternative non-parametric test has to be used. Homogeneity will be discussed in the subchapters. The Wilcoxon rank-sum and Mann-Whitney u test are both the non-parametric equivalents of the independent t-test(Field, 2013). The Mann-Whitney U-test is a median-based test, which interprets significant p-values as there is a difference between the two variables.

Using this test hypotheses 4 and 5 can be tested by showing if there is a difference between architects and stakeholders in Collaboration and Decision quality. Since there is a hypothesis, the one-sided p-values will be interpreted and presented. Since this is explorative research other variables are also observed, however without hypotheses the two-sided p-values are presented.

To answer the explorative question 1, the differences between the two types of architects are observed. This will be done via the relative rankings and a statistical test. This will once again make use of the Mann-Whitney u-test since the variables violate the normality assumption of the independent t-test. The second cluster of stakeholder sub-groups is a test that includes 3 groups: management-roles, agile-roles, and engineer-roles. These statistical differences will be tested with the ANOVA test. The ANOVA test is considered robust, which does not require normality but should be interpreted with caution if the normality assumptions are not met. Values that indicate differences are explored in the post-hoc analysis. In the sub-groups the means are different from each other, therefore the respective values are corrected with the mean of the subgroups.

For tests of homogeneity, the Levene's test for equality of means is used. This variable tests homogeneity, if this test gives a significant value, equal variances should not be assumed. Otherwise, the equal variances assumed category of the t-test can be observed.

3.1.4.3 Data preparation

A full overview of the data transformation can be found in appendix D. Entries with missing data will be removed, just like participants who did not give informed consent. Answers including 'other' in regards to their role will be reviewed and placed into one of the existing categories if applicable, all 'other' answers were placed into the existing categories. One entry was deleted due to an unusable answer in one of the categorization questions.

Additional columns were created with 'corrected' values which had corrected values based upon the mean of their group. This created corrected columns both based upon being a software architect or a stakeholder, and on the hierarchal layer the participant was added, the values are the original values corrected by the mean of all answers by that specific category. As an example, if architects averagely answered with 5 overall behavioral competencies then every value of every architect will be corrected with 5 in a separate setup.

Finally, the different types of roles for both software architects and stakeholders were grouped following their function. Additionally, columns were added with the different classifications to prepare

to answer the explorative questions and hypotheses 4 and 5. The overview of the different groups can be found in table 3a. Following Britto et al.(2016) the mid-level software architect, in this case, the FCA and FCTs, are referred to as a product-level architect. The high-level software architects, in this case, everything above FCA and FCT i.e. product, platform, department software architects, will be referred to as system-level architects.

3.1.5 Privacy and data storage

To ensure participants' privacy, all personal data will be fully anonymized. In addition, participants will not be asked to leave their personal information for the follow-up qualitative research. It will ask the participant to reach out via e-mail. The data will be stored safely and will not be shared with any third parties.

Table 3a: different types of groups and coding number

Architects	***Stakeholders***
1 (system-level architects)	*3 (management)*
2 (Product-level architect)	*4 (Agile projects)*
-	*5 (Engineers)*

3.2 Outcome questionnaire

First, this chapter will discuss some demographics after which the findings of the data analysis will be presented. The questionnaire itself was open from 15/11/2021 until 07/12/2021 and gained 226 complete responses from which 5 without informed consent and 1 additional entry was excluded based upon unusable categorical answers.

3.2.1 Demographics

The demographics asked for a function, gender, and experience in both role and ASML. As seen in figure 3a you can see the overwhelming majority of participants being male. This seems however similar but a bit worse than the normal distribution of about 10% females versus 90% males. In addition, there are fewer 'younger' people in this questionnaire which is not surprising.

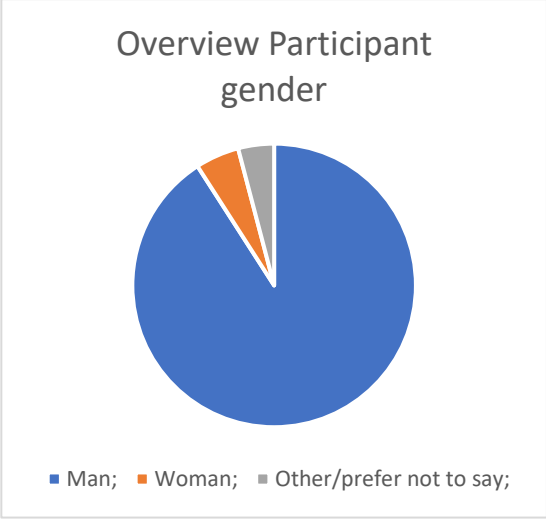

Figure 3a Gender demographics

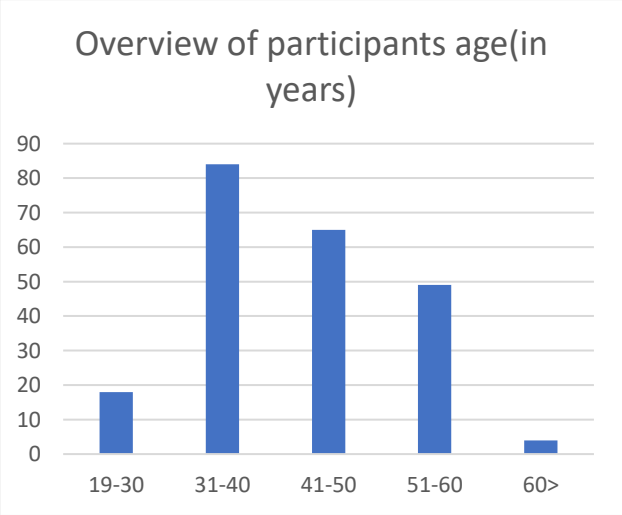

Figure 3b Age categories demographics

An additional context is that the different stakeholders are not uniformly distributed. This is a logical consequence of occurring roles however it might be relevant in later stages. There are a multitude more FCA, FCT roles than other architects. In addition, there are a lot more develop or competence engineers (note that competence engineer here is an engineer focused on a specific competence). This is unsurprising since there simply are more functional clusters or engineering jobs as for departmental jobs.

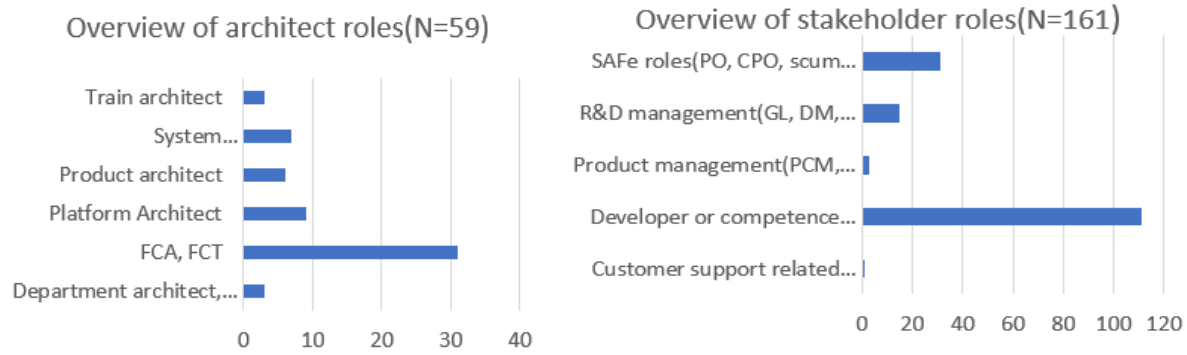

Figure 3c Participant roles architects and participants stakeholders

3.2.2. 'Most' important behavioral competencies

This chapter will try to analyze the data to test hypotheses 1, 2, and 3. Since there were three 'most' important behavioral competencies, it is tested if these are really at the top. First, the ranking will be observed, after which the different variables are tested. To be considered the most important, each of these variables should be present at the top of the ranking. The top-ranking, as shown in the most left column of table 3d, shows the number one and two spot to being manages complexity and communicates effectively. After this, an absolute gap seems to be present until the number three, decision quality. After, the number four collaboration seems closer to the number three. And the top 5 is closed by instills trust, however, visually instills trust seems to be closer to the number 6 than the number 4. In the ranking, it would seem like indeed the three expected behavioral competencies are present in the top 3. However, there does seem to be a larger set between the number 1 and 2, and the other categories.

Table 3b: Descriptive statistics behavioral competencies

	Complete dataset		
	N	M	SD
1. Ensures Accountability	220	4.97	1.21
2. Develops Talent	220	4.65	1.33
3. Communicates Effectively	220	6.41	0.82
4. Manages Complexity	220	6.5	0.80
5. Instills Trust	220	5.82	0.87
6. Cultivates Innovation	220	5.55	1.04
7. Drives Engagement	220	4.92	1.22
8. Drives Results	220	4.95	1.18
9. Values Differences	220	4.99	1.11
10. Situational Adaptability	220	5.5	1.07
11. Plans and Aligns	220	4.92	1.24
12. Collaborates	220	6.02	0.92
13. Builds Effective Teams	220	4.21	1.43
14. Optimizes Work Processes	220	4.36	1.37
15. Balances Stakeholders	220	5.53	1.21
16. Business Insight	220	5.15	1.23
17. Strategic Mindset	220	5.75	1.01
18. Demonstrates Self-awareness	220	5.1	1.11
19. Self-development	220	5.3	1.12
20. Decision Quality	220	6.16	0.88

The second part of the argumentation for this would be the statistical comparison between the different variables, using the Scott Knott clustering method. The outcome, shown in figure 3d, shows the automatic clusters generated. In this, the number 1 manages complexity, and the number 2, communicates effectively are clustered. Following the set assumption which sees 'most' important as the top-ranked, this would seem to support hypotheses 1 and 2 but would reject decision quality as the most important behavioral competency. However, this does not mean decision quality is unimportant. The second bracket of decision quality and collaboration can still be viewed as important as they are a cluster that is placed above average, but not the most important. Also note how the following cluster would be instills trust and strategic mindset, making it the third cluster in importance.

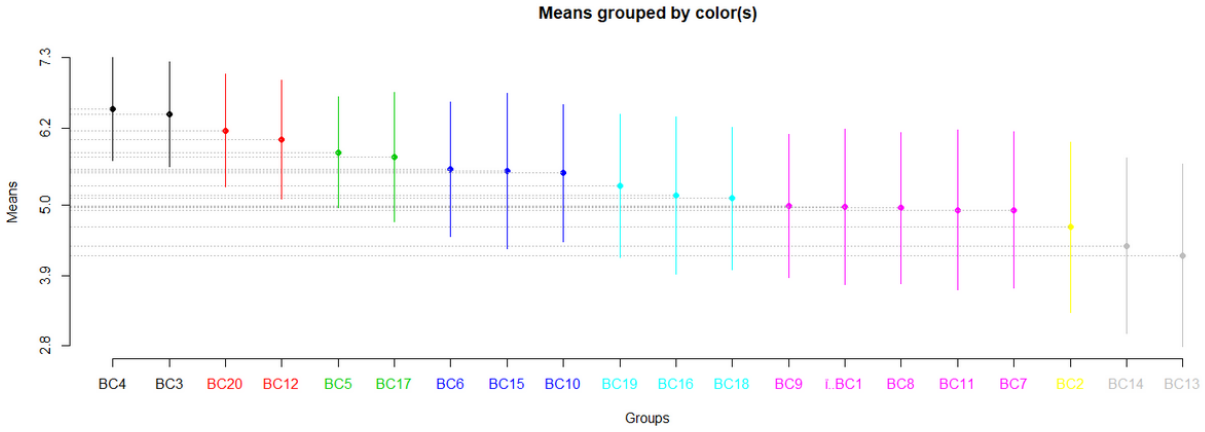

Figure 3d: Scott Knott ESD cluster results

As the number 3 and 4, this would suggest a secondary category of the importance of collaboration and decision quality, making them not the 'most' important but also be considered very important for software architects by all participants. This difference is also shown in the forced-choice question ranking, table 3c, where manages complexity and communicates effectively are in the top-3 a respective 150 and 140 times out of the 220. After this, there seems to be a drop-off after which the rest of the competencies are mentioned more often. Note that the interpretation here is the number of mentions in the top-3.

Table 3c: forced-choice top 3 mentions

	Behavioral competency
1	Manages complexity(150)
2	Communicates effectively(140)
3	Decision quality(59)
4	Strategic mindset(46)
5	Collaborates (42)
6	Instills trust(36)

3.2.2 Stakeholders and architects

Hypotheses 4 and 5 expected a difference between stakeholders and architects in how their respective ranking between collaboration and decision quality would be expressed. To see if there are any differences between stakeholders and architects first the relative rankings are compared to each other. Then the two groups are compared with the Mann-Whitney U-test on each of the 20 behavioral competencies to test hypotheses 4 and 5, but also explore if these would be the only differences. The group of 220 participants contained 59 software architects and 161 stakeholders. Based on the descriptive statistics the top of both groups is shown in table 3d. Where the complete dataset shows the top three as 'manages complexity', 'communicates effectively' and 'decision quality', on average

software architects seem to value collaboration more than decision quality switching them around in the number 3 and 4 spot. The number 1 and 2 seem to be the same for both groups, just like the number 5 instills trust. Communicates effectively and manages complexity being number 1 strengthens the support of, even for different groups, these are the most important.

Table 3d: Overview of the 6 most important behavioral competencies in general, for stakeholders and for architects.

	Total	**Stakeholders**	**Software architects**
1	Manages complexity (6.50)	Manages complexity(6.47)	Manages complexity(6.58)
2	Communicates effectively(6.41)	Communicates effectively(6.39)	Communicates effectively (6.47)
3	Decision quality (6.16)	Decision quality (6.23)	Collaboration(6.20)
4	Collaboration(6.02)	Collaboration(5.95)	Decision quality(5.97)
5	Instills trust(5.82)	Instills trust(5.79)	Instills trust(5.90)
6	Strategic mindset(5.75)	Strategic mindset(5.75)	Balances stakeholders(5.85)

The second part of the analysis is the independent t-test between the two groups. Based on the Levene's test all variables full fill the homogeneity assumption. Both groups are violated the normality assumption. Because of the hypotheses, the one-sided p-value will be observed for collaboration and decision quality.

Table 3e: descriptive statistics and u-test stakeholders and architects

	Stakeholders				**Architects**					
	N	**M**	**SD**	**Md**	**N**	**M**	**SD**	**Md**	**U**	**P**
1. Ensures Accountability	161	4.98	1.24	**5**	59	4.95	1.12	**5**	4671.5	.845
2. Develops Talent	161	4.6	1.37	**5**	59	4.81	1.21	**5**	4277	.242
3. Communicates Effectively	161	6.39	0.87	**7**	59	6.47	0.68	**7**	4628.5	.744
4. Manages Complexity	161	6.47	0.82	**7**	59	6.58	0.72	**7**	4421.5	.361
5. Instills Trust	161	5.79	0.88	**6**	59	5.9	0.85	**6**	4368	.329
6. Cultivates Innovation	161	5.6	1.06	**6**	59	5.44	0.99	**6**	4307	.266
7. Drives Engagement	161	4.93	1.25	**5**	59	4.9	1.13	**5**	4635.5	.777
8. Drives Results	161	4.93	1.21	**5**	59	5.02	1.11	**5**	4626.5	.759
9. Values Differences	161	5.05	1.08	**5**	59	4.81	1.18	**5**	4254	.217
10. Situational Adaptability	161	5.5	1.08	**6**	59	5.49	1.04	**6**	4691	.884
11. Plans and Aligns	161	5.06	1.21	**5**	59	4.54	1.26	**5**	3647.5	.006
12. Collaborates	161	5.95	0.97	**6**	59	6.2	0.76	**6**	4145	.121
13. Builds Effective Teams	161	4.16	1.45	**4**	59	4.36	1.35	**4**	4332.5	.307
14. Optimizes Work Processes	161	4.28	1.39	**4**	59	4.59	1.3	**5**	4108.5	.116
15. Balances Stakeholders	161	5.42	1.27	**6**	59	5.83	1	**6**	3924.5	.039
16. Business Insight	161	5.22	1.2	**5**	59	4.95	1.29	**5**	4179.5	.157
17. Strategic Mindset	161	5.77	0.99	**6**	59	5.69	1.06	**6**	4536.5	.594
18. Demonstrates Self-awareness	161	5.09	1.13	**5**	59	5.15	1.06	**5**	4689	.879
19. Self-development	161	5.33	1.1	**5**	59	5.22	1.18	**5**	4558.5	.632
20. Decision Quality	161	6.23	0.84	**6**	59	5.97	0.96	**6**	4032.5	.065

In the u-test, table 3e, a few differences come up. The Mann-Whitney U test revealed architects to score decision quality almost significantly lower(Md= 6, n=61) compared to the stakeholders(Md=6 , n=159), U=4032.5 , z=-1.842, p=0.065). Even though not fully significant, in this real-world setting it is expected to become significant when more samples are taken. Considering this full-support would not be correct, however, due to the expectancy of it becoming significant with more samples of architects it would also not be correct to reject this hypothesis. By reviewing the sub-groups in the later stage, the evidence might help support hypothesis 4. The difference which is not shown as different is collaboration, this would show evidence of rejecting hypotheses 5.

Table 3F: Ranking difference between system-level architects and product-level architects

	Software architects(combined)	**System-level architects**	**Product-level architects**
1	Manages complexity	Manages complexity(1.27)	Manages complexity(1.21)
2	Communicates effectively	Communicates effectively(1.26)	Communicates effectively(1.03)
3	Collaboration	Decision quality (.908)	Collaboration(.942)
4	Decision quality	Collaboration(.748)	Instills trust(.589)
5	Instills trust	Balances stakeholders(.548)	Balances stakeholders(.442)
6	Balances stakeholders	Instills trust(.508)	Decision quality(.412)

The other differences were not expected. The test revealed architects to score plans and aligns significantly lower(Md= 5, n=61) compared to the stakeholders(Md=5 , n=159), U=3647.5, z=-2.733, p=0.006). And revealed architects to score balances stakeholders significantly lower(Md= 6, n=61) compared to the stakeholders(Md=6 , n=159), U=3924, z=-2.059, p=0.039).

3.2.3 System-level architects and product-level architects

This chapter aims to answer the first explorative question: are there any differences in which behavioral competencies are seen as important between the system-level software architects and product-level software architects?

The first step was the observation of the ranking of the different sub-groups of architects based on the descriptive statistics of table 3g. The top of this ranking can be viewed in table 3f. With more than a full point above their average, both types of architects Have manages complexity and communicates effectively on their number 1 and 2 spot. Note here how the system-level architects score the absolute value of communicates effectively nearly on level with manages complexity where product-level architects score communicates effectively closer to collaboration. The most notable other differences are the drop-off of decision quality from the 3rd to the 6th spot with product-level architects, with product-level architects scoring decision quality almost half a point lower on average than system-level architects. Additionally, in absolute values it seems like after dropping below scoring thins above around .75 higher on average, there is a visual observation of a new bracket being in the below .6 above average category.

The second part of this explorative comparison was using the Mann-Whitney U-test, results can be found in table 3g. No homogeneity assumptions are violated. The Mann-Whitney U test revealed product-level architects to score decision quality almost significantly lower(Md= 6, n=34) compared to the system-level architects(Md=6 , n=25), U=314.5 , z=-1.801, p=0.072). This is the only notable finding. To answer, the explorative question. There might be an indication from both observations and from

statistics where product-level architects and system-level architects perceive the importance of decision quality differently. With these small sample sizes, it is expected for this value to become significant when the sample size is increased.

Table 3g: descriptive statistics and u-test architect sub-groups

	System-level architects				Product-level architects					
	N	**M**	**SD**	**Md**	**N**	**M**	**SD**	**Md**	**U**	**P**
1. Ensures Accountability	25	4.80	1.22	**5**	34	5.06	1.04	5	383	.499
2. Develops Talent	25	4.76	1.13	**5**	34	4.85	1.28	5	406	.760
3. Communicates Effectively	25	6.60	0.65	**7**	34	6.38	0.70	6	345	.163
4. Manages Complexity	25	6.60	0.58	**7**	34	6.56	0.82	7	413.5	.832
5. Instills Trust	25	5.84	1.03	**6**	34	5.94	0.69	6	419.5	.926
6. Cultivates Innovation	25	5.32	0.90	**6**	34	5.53	1.05	5.5	386.5	.527
7. Drives Engagement	25	4.96	1.24	**5**	34	4.85	1.05	5	392	.597
8. Drives Results	25	4.92	1.08	**5**	34	5.09	1.14	5	401	.702
9. Values Differences	25	4.96	0.89	**5**	34	4.71	1.36	5	371.5	.394
10. Situational Adaptability	25	5.44	0.96	**6**	34	5.53	1.11	5.5	409.5	.804
11. Plans and Aligns	25	4.48	1.08	**5**	34	4.59	1.40	5	399.5	.687
12. Collaborates	25	6.08	0.76	**6**	34	6.29	0.76	6	353.5	.235
13. Builds Effective Teams	25	4.20	1.41	**4**	34	4.47	1.31	5	382.5	.501
14. Optimizes Work Processes	25	4.64	1.32	**5**	34	4.56	1.31	5	408	.788
15. Balances Stakeholders	25	5.88	1.09	**6**	34	5.79	0.95	6	389	.563
16. Business Insight	25	4.92	1.55	**5**	34	4.97	1.09	5	415.5	.880
17. Strategic Mindset	25	5.64	0.91	**6**	34	5.74	1.16	6	388.5	.557
18. Demonstrates Self-awareness	25	5.00	1.26	**5**	34	5.26	0.90	5	392.5	.601
19. Self-development	25	5.36	1.25	**6**	34	5.12	1.12	5	350	.227
20. Decision Quality	25	6.24	0.78	**6**	34	5.76	1.05	6	314.5	.072

3.2.4 Comparing different stakeholder sub-groups

	Table 3h: stakeholder sub-group competency rankings (			
	Stakeholders	**Management**	**Agile-roles**	**Competence engineers**
1	*Manages complexity*	Manages complexity(1.01)	Manages complexity(1.16)	Manages complexity(1.14)
2	*Communicates effectively*	Communicates effectively(1.01)	Communicates effectively(.849)	Communicates effectively(1.11)
3	*Decision quality*	Collaboration(.955)	Decision quality(.974)	Decision quality(.904)
4	*Collaboration*	Decision quality (.677)	Collaboration(.599)	Collaboration(.561)

5	*Instills trust*	Strategic mindset(.510)	Instills trust(.599)	Instills trust(.453)
6	*Strategic mindet*	Situational adaptability(.510)	Strategic mindset(.505)	Strategic mindset(.399)

The last chapter compared the sub-groups of the software architect, it observed the differences between the two different classifications of architects. This answered the first explorative question of the hypotheses. In chapter 3.2.2 a difference between stakeholders and architects was observed. This chapter will address the second explorative question of the hypotheses: Are there any differences in which behavioral competencies are seen as important between the different stakeholder sub-groups: management, agile-roles, and engineers?

The first method of comparing these groups was via observing their ranking. To do this the ranking followed the corrected descriptive statistics, which can be found in table 3h. The top of the ranking is shown in table 3I, with the corrected averages between brackets. Note that findings in this chapter should be interpreted with caution, the management group only consisted of 18 participants versus the 32 participants of the agile-project roles and the 111 competence engineers. The correction of all the groups was 5.601 for the management stakeholders represented as 3, 5.276 for the agile-stakeholders represented with a 4, and with 5.3125 for the engineer.

Table 3I: ANOVA of different stakeholder subgroups

Measure	Management		Agile-role		Engineer		Sum of Squares	df	Mean Square	F	Sig.
	M	*SD*	*M*	*SD*	*M*	*SD*					
1. Ensures Accountability	-0.05	1.15	-0.21	1.27	-0.45	1.23	3.318	2	1.659	1.095	0.337
2. Develops Talent	-0.82	0.81	-0.93	1.36	-0.67	1.44	1.808	2	0.904	0.483	0.618
3. Communicates Effectively	1.01	0.50	0.85	1.16	1.12	0.80	1.862	2	0.931	1.257	0.287
4. Manages Complexity	1.01	0.61	1.16	1.11	1.14	0.76	0.297	2	0.149	0.218	0.804
5. Instills Trust	0.18	0.73	0.60	1.07	0.45	0.84	2.055	2	1.028	1.325	0.269
6. Cultivates Innovation	0.29	0.83	0.22	1.14	0.26	1.07	0.057	2	0.029	0.025	0.975
7. Drives Engagement	-0.43	1.10	-0.53	1.16	-0.38	1.30	0.572	2	0.286	0.182	0.834
8. Drives Results	-0.05	0.98	-0.43	0.99	-0.46	1.27	2.647	2	1.323	0.932	0.396
9. Values Differences	-0.66	0.94	-0.15	1.26	-0.27	1.06	3.092	2	1.546	1.305	0.274
10. Situational Adaptability	0.51	0.90	0.29	1.13	0.07	1.06	3.702	2	1.851	1.645	0.196
11. Plans and Aligns	-0.88	1.18	-0.28	1.39	-0.18	1.16	7.62	2	3.81	2.611	0.077
12. Collaborates	0.95	0.51	0.60	1.07	0.56	0.96	2.402	2	1.201	1.335	0.266
13. Builds Effective Teams	-1.49	1.23	-1.49	1.24	-1.03	1.53	7.288	2	3.644	1.736	0.18

14. Optimizes Work Processes	-1.60	1.03	-1.15	1.34	-0.94	1.46	7.049	2	3.524	1.812	0.167
15. Balances Stakeholders	0.40	0.69	0.01	1.35	0.06	1.30	2.066	2	1.033	0.652	0.522
16. Business Insight	0.29	0.90	0.10	1.04	-0.25	1.25	6.389	2	3.195	2.317	0.102
17. Strategic Mindset	0.51	1.02	0.51	1.07	0.40	0.96	0.401	2	0.2	0.206	0.814
18. Demonstrates Self-awareness	0.07	0.69	-0.12	0.99	-0.34	1.20	3.226	2	1.613	1.293	0.277
19. Self-development	0.07	0.91	-0.03	1.05	-0.02	1.14	0.114	2	0.057	0.047	0.954
20. Decision Quality	0.68	0.67	0.97	0.95	0.90	0.84	1.064	2	0.532	0.748	0.475

The ranking results in the following observation: the number 1 and 2 spot being taken once again by manages complexity and communicates effectively. However, where the agile roles and competence engineers rank the same competencies in their top 5, the management stakeholders rank collaboration higher just like the product-level architects. Additionally, instills trust is not in their top 5.

The F-test(ANOVA) is used to analyze the statistical differences. The homogeneity requirement is violated for BC15 'balances stakeholders' following the Levene's test, the Brown-Forsythe F will be shown instead of the Welch test. The homogeneity assumption is not met for the management group, even though the ANOVA is seen as a robust test this means results for the management group should be interpreted with caution. Also, note that the distribution of subgroups is as expected since there are fewer management roles than the other stakeholders in ASML. Due to the explorative nature of this research findings are still considered valuable. Table 3F shows the results of the ANOVA.

The results in the ANOVA do not produce any statistical evidence to support differences between the different stakeholders. However there was an almost statistically different result for plans and aligns, $F(2, 217) = 1.335$, $p = 0.077$ and for business insight $F(2, 217) = 2.317$, $p = 0.102$. Since there were no hypotheses the post-hoc test will be analyzed for these two values, table 3J. In this table the difference in plans and aligns is caused by a significant difference, $p=0.006$, between the management (1) and engineers(3) layers where the management layer scored plans and aligns a lot lower compared to the almost average scoring of the engineers of this behavioral competency. The observation of the posthoc test of business insight gave no additional results. When regarding the effect sizes, the decision quality variable has a medium effect size, the point estimate is .526. This is then followed by communicates effectively with an effect size of .352. Indicating the utility of this variable in the real world when comparing the different stakeholder subgroups.

Table 3J: post-hoc of sub-groups stakeholders

Dependent variable						95% confidence interval	
	(i)	(j)	MD	SE	P	LB	UB
11_COR_hierarchy	1	2	-.3356790	.2405742	.345	-.903412	.232054
		3	-.6858444	.2204990	.006	-1.206202	-.165487
	2	1	.3356790	.2405742	.345	-.232054	.903412
		3	-.3501654	.1908074	.161	-.800453	.100123

	3	1	.6858444*	.2204990	.006	.165487	1.206202
		2	.3501654	.1908074	.161	-.100123	.800453
16_COR_hierarchy	1	2	.0297192	.2393161	.992	-.535045	.594483
		3	.1304137	.2193459	.823	-.387223	.648050
	2	1	-.0297192	.2393161	.992	-.594483	.535045
		3	.1006945	.1898095	.856	-.347239	.548628
	3	1	-.1304137	.2193459	.823	-.648050	.387223
		2	-.1006945	.1898095	.856	-.548628	.347239

3.3 Summary results questionnaire

This chapter answers the research question, it gives a list of behavioral competencies namely manages complexity, communicates effectively and decision quality, collaboration and instills trust as the respective top 5 important behavioral competencies. This supports hypotheses 1, 2, and 3 which claim manages complexity, communicates effectively and decision quality as most important behavioral competencies. However, the clustering showed how decision quality was not in the top cluster, unlike manages complexity and communicates effectively. This would be the evidence with which decision quality is not the most important behavioral competency, rejecting hypotheses 3. In addition, the number three decision quality is perceived as less important as collaboration switching to the number four spot for software architects. This is mainly due to the opinion of what is important by product-level architects who see decision quality as less important for them compared to system-level architects. This would provide support to reject hypotheses 4 which expected no difference between these levels of architects. Stakeholders perceived the same five competencies as the most important however on average valued decision quality moreover collaboration with both being on the number three and four spots just like with architects. The number five for both groups is considered to be ‘instills trust’.

Chapter 4: Qualitative research interview

The questionnaire produced evidence on answering which behavioral competencies are most important. However, it does not answer sub-question 4 which wants to know how the important behavioral competencies actually interact in practice. This chapter will follow up on the interviews by exploring how the behavioral competencies lead to success. This is expected to create a context that will expand knowledge on how the behavioral competencies function in practice. In addition, it also validates the results found in the questionnaire. This chapter will focus on the important behavioral competencies, as is the focus of the research question.

4.1 Interview setup

Interviews often provide in-depth information about the interviewee's experiences and viewpoints of a discussed topic (Turner, 2010). The flexibility of a semi-structured interview allows for the elaboration of information necessary for the participants of the interview(Gill et al., 2008). The goal of this interview is to explore how (important) behavioral competencies lead to success. Since Interviews can adapt the questions based on the interactions(Turner, 2010), interviews are the right method.

The interviews will be set up following the principles of Turner(2010) which emphasizes the importance of preparation, selecting candidates, pilot testing, constructing effective research questions, implementing the interview, and interpreting data. The design will have to adhere to the time restriction given by the case company of a maximum of 1-hour interviews with people including introductions and closing statements. In addition, due to the global pandemic, the interviews will take place digitally. Questions will ask about descriptive examples of different situations in which a software architect was successful. Follow-up questions focus on how specific behavioral competencies led to success, these competencies are selected based upon the outcomes of chapter 3.

4.1.1 Interview research question

To structure the interviews a specific research question is created for the interview section. To know how behavioral competencies lead to success it has to be analyzed which are present in success situations. This leads to the following research question for this chapter:

Int-RQ: How do behavioral competencies contribute to success?

Chapter 3 gave some expectations where behavioral competencies like communicates effectively, manages complexity, collaboration, and decision quality are expected to be present. To know how the different behavioral competencies are used in practice, it should be known which are present forming the following sub-questions: (Int-SQ1) What behavioral competencies contribute to success? In both these questions, the term 'success' is implied by the participants, however, interviews could explore what this 'success' would be. Note here is that the main focus will be on the behavioral competencies however this also provides the opportunity to learn more about what success means for software architects This forms the explorative sub-question: (Int-SQ2) What are the performance outcomes which indicate success?

4.1.2 Participant selection

The second preparation step is the selection of candidates. Participants for the interviews were selected based on their role, experience and their connections. The experience is relevant since more experienced people, more often than not, can articulate examples better and have a better understanding of what creates success in their role. Furthermore, since the requirement for the questionnaire was people who are or work with software architects, this requirements sticks. This causes at minimum for the interviewee to value a potential outcome. In the questionnaire, different roles were seen as either stakeholder or architect and in a certain bracket of hierarchy. The interviews will target all the different groups to explore as many perspectives as possible creating a minimum of 5 interviews. However, the majority of the weight of the interviews is expected to be on the higher end due to the previously mentioned experience expectation. Since the questionnaire was sent out in the wider MX in which MCI is placed this will be taken into account by including another department to diversify the explorative potential of the interview.

For the selection of participants the method was as follows, at the end of the questionnaire a message was created that asked people interested in a follow-up to contact the researcher. Out of this group, a few people from the same department (MCI) volunteered and approached. Based on their group leads advice, experienced people were approached. The second set of interviews followed the same line. One experienced architect volunteered, and via his group lead every other role-slots was filled based on the experience they had from the group leads perspective. Note that from the first group people were included with more experience and more management expertise while in the second group more wider spread of stakeholders was taken into account. Once again this is due to the early scope to focus on a higher hierarchal level.

Table 4a: Role categories of interview participants

	Role category	**<Poule A>**	**<Poule 2>**
SA	System-level	ASML_03; ASML_10	ASML_08
	Product-level	ASML_09; ASML_06	ASML_05
stakeholder	Management	ASML_11 ; ASML_04	ASML_01
	Agile-role		ASML_02
	Engineer		ASML_07

4.1.3 Testing

As a test for the interview set up, a group session with a whiteboard was held within an architect meeting to review if the questions are clear and if the definitions are viewed similarly. After this test, the questions focused more on asking specific examples, rather than the definition as the examples show types of behavior. No other changes were made to the interview setup.

4.2 Data and coding setup

The collected recordings will be transcribed. The data will be stored safely with both the company and the researchers and will not be shared with third parties. The processed data will not show any names or specific technical elements to protect both the participants' and the companies' interests. Participants' names will be transformed as ASML_XX, where XX is the number of the interview held. In addition to the recordings also the raw interview notes are stored following the same naming structure as the interviews.

4.2.1 Coding

Coding is an iterative process. The interviews are coded in word using excerpts marked by a comment. To answer the first interview sub-question deductive coding will be used where the starting set and definitions of the 20 behavioral competencies(Kornferry, n.d.), overview given in table 4b. Excerpts of the 20 behavioral competencies are marked in yellow and a short summary will be given in the comments to protect the interest of the company. The identified behavioral competencies will be added to the excerpts with BC## where ## represents the number of the behavioral competencies. Using the same color for markings was a result of the iterative process where examples often contained multiple behavioral competencies. A few examples given are examples of bad behavior. Often these interactions do show what is valued, and what isn't in an example. In the coding, these negative examples are marked with a '-' next to the related behavioral competency(BC).

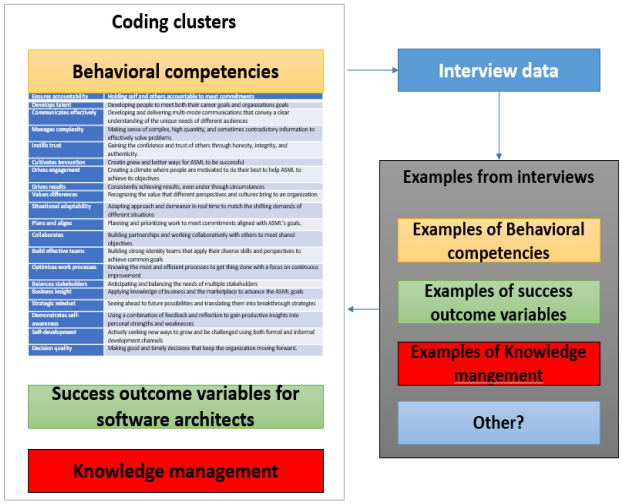

Figure 4a: Iterative coding cycle with final overview

Every interview will contain multiple examples, therefor the same behavioral competency could be present multiple times in the interviews. To answer which are present, the amount of times something is mentioned in each interview will be structured.

To explore the context of success, by searching for answers to the second sub-questions of what 'success' would entail, anything related to the output of success for software architects was marked in green. This will specifically search for performance outcomes that influenced the perception of success in a project or situation. Any other findings will be addressed as such, other. If any patterns emerge they will get their own color. In this iterative process, a new other category emerged: knowledge management.

Table 4b: overview behavioral competencies in interviews

Behavioral competency	Code number	Occurrence ratio
Communicates effectively	3	11/11
Manages complexity	4	11/11
Collaboration	12	11/11
Decision quality	20	11/11
Instills trust	5	9/11
Balances stakeholders	15	6/11
Strategic mindset	17	6/11
Business insights	16	3/11
Drives results	8	2/11
Plans and aligns	11	2/11

4.3 Interview results

This chapter will discuss the outcome of the interview. Note how this is part of an exploratory study, the findings aim to further the understanding of the important behavioral competencies of a software architect in support of which behavioral competencies are important. It will first discuss the behavioral competencies, after these the success outcome variables and knowledge management findings will be presented. The last two are more exploratory in nature.

4.3.1 What behavioral competencies contribute to the success

To answer the first question an overview of the present behavioral competencies in the interview is made, table 4b. The behavioral competencies communicates effectively(BC3), manages complexity(BC4), collaboration(BC12), and decision quality(BC20) are present in multiple examples in every interview. This is in line with the findings of chapter 3. Instills trust(BC5) is present often but is more prevalent in the examples of stakeholders. Finally, there is a third segment of behavioral competencies which appear sporadically, these results will not focus on these behavioral competencies. The full overview of excerpts and related behavioral competency can be found in appendix E.

4.3.2 How the top-ranked behavioral competencies lead to success

As mentioned in the methodology, in the iterative process many examples contained multiple behavioral competencies in the same situation. Take the example from ASML_09 which described *a specific deduction of the complexity where by using rationale the architect realized there were actually different intentions of the project, being able to simplify the old requirements into a single clear new requirement. To verify this simplification the architect turned to his project management counterpart and convinced him and others with various means via meetings.* The architect had to take into account all different types of information and perceptions of different stakeholders to come to the realization of how this could be simplified, which shows the managing complexity behavioral competency(BC4). To start this change the architect had to first convince and then work together with stakeholders to reach their shared goal of finishing the project as well as they can, which exemplifies the usage of network and collaborating with it(BC12). This convincing was done via various meetings and stances where the architect tried to communicate in the right format depending on the interest of the stakeholder(BC3).

This example illustrates what is present in many different examples of success: rarely just a single behavioral competency is present in success for software architects. There are similar examples showing this trend like when ASML_03 where by reviewing documents and talking to people the architect(BC3) identified how things were not in line with the roadmap(BC4) which allowed the architect to intervene by working with the engineers and steering them towards a better direction(BC12). This was done by using the right communication methods in order to convince the engineers(BC3) and all was done in order to create a better decision that is in line with the roadmap(BC20-). This shows how the answer to which behavioral competency is important in success situations is expected to be more than one behavioral competency.

In both situations, a combination of behavioral competencies was present in a single example. All interviews show a similar trend where in most cases it is not just one behavioral competency in the example of success. With this in mind, the next paragraph will exemplify how specific behavioral competencies lead to success.

4.3.2.1 How communicates effectively and collaboration lead to success

Four behavioral competencies were present in all interviews in a variety of ways. Two of these were communicates effectively and collaboration, this subchapter will explore how these are represented in the interviews. The definition of communicates effectively seems to focus more on the delivery of a clear understanding of the unique needs of different audiences. This would be more focused on the

giving information part of Kruchten(2008)'s circle. Interviews do confirm this perception of communicates effectively but also expand on it. On one side communicating effectively is part of the information gathering process like having formal and informal one-on-one talks, doing whiteboard sessions, and having formal meetings(ASML_08) to gather the input. The informal talks are more chats that can happen at any place at the coffee machine and don't necessarily have to be about work. Meetings are structured moments with clear goals for the meeting itself and often the moment to converse with stakeholders. A whiteboard session is a more brainstorm-like meeting focused on discussion and free-thinking. On the other side, communicates effectively helps to convey the message like for example the interaction where the architect needed different methods to convince others of the decisions(ASML_04).

In both the conveying the message and gathering information sides, the architect should be able to use a variety of methods to get the right information and to give the appropriate information, while being able to clearly express their intentions. Even though these perspectives are different, literature already indicated there are multiple different ways of splitting communication. For an architect, however, this would mean the original definition would be challenged and widened or split up since both perspectives seem to have a complementary relationship in the role of the software architect.

Architects should be able to express themselves to get people on board with ideas but also to gather their information. Good communication prevents miscommunications but in consequence also bugs and errors(ASML_09). This indicates the relation between the communicative side with the technical side of the architects' job.

This social and side of the architect is also present in the collaboration competency. Collaboration follows the same split, where it is important in both gathering input but also giving information. On the gathering input side, ASML_09 realized how a different department was needed and started to involve them and work with them to improve the organization as a whole. This is an iterative process where both sides need to take into account the pro's and con's of the other. In one example a stakeholder(ASML_01) saw how the architect saw an opportunity in getting rid of a weekly report which was also done daily. The architect used the meetings to work together with people and used meetings, one on one conversations in order to let people accept this change. However, the architect wanted to get rid of the larger element, all the details were created in a collaborative effort to simplify the process with the stakeholders.

However, different from communicates effectively, collaboration is more a single term, collaboration is seen as a process and architects strive to be part of this team process which allows both for gathering the information, but also making sure the solution is supported. ASML_03 mentioned always striving for getting more opinions, involving more relevant people, being a team player but also mentioned having to get involved when he observes people making mistakes. The interactive process of both involving people but also reacting to the same people is the dynamic of collaboration in a software architect. Alternatively, collaboration could also be seen as a part of the (joint) decision-making process. This will be discussed in the next sub-chapter.

4.3.2.2 How manages complexity and decision quality lead to success

The previous chapter mentioned how collaboration could be seen as part of the joint decision-making process. This paragraph address this finding and the dynamics of manages complexity and decision quality.

First, decision quality is loosely defined as making the right decision at the right moment, something an architect is expected to do. However, in the interviews decision quality and the decision itself were often as an intermediary moment. Making the right decision was often preceded and succeeded by other important behavioral competencies. An example from ASML_03, after having reviewed a project's documents the architect identified a mismatch between what was being done and the roadmap(Manages complexity). By finding this in time, the architect was able to intervene and steer the team towards another solution(collaboration and communicates effectively). This steering led to the decision itself being better since it now is in line with the roadmap(Decision quality). Had it been unnoticed it could have become a problem later on. Intervening was an individual decision, but also the direction to which was steered was a made technical decision. However, to make this decision work the architect had to work together with the team(collaboration) to get the new change actually executed. This exemplifies how decision quality is related to managing complexity as making decisions without having managed the information input is like going in blind. In addition, without the collaboration the original direction would probably not have changed and the new direction would not have been executed as it is meant to be. This is also where the previously mentioned joint decision-making comes in. Often the architect gathers information, makes an overview of the possible choices, and then jointly make a decision with the team(ASML_10). This does not mean the architect does not already have a preference of where he wants to go(ASML_11). It does mean that making the decisions together with the team indicates more success. ASML_05 indicated how making decisions that have a lot of resistance, will cause a drop in motivational level of the executing team(s). After which he mentioned being "*more of a moderator of the decision process than actually making the decision"(ASML_05)*. Whereas an architect often there are multiple different possible decisions. Which decision will be chosen does not matter as long as it can be viewed as good and does not interfere with the system's integrity, only if the options are less good the architect will steer the others away from that option.

The findings indicate how there are two distinct concepts surrounding behavioral competency decision quality. First, decision quality seems to require other behavioral competencies like communicates effectively, manages complexity, and collaborates to get to the decision, but also to make the decisions execution by the teams. If you explain your reasoning well, people will get on board and they will make the decision in your line of thinking(ASML_10). This brings us to the concept of joint decision-making. The architect can be viewed more as a moderator of the decision process(ASML_05) which works together with the teams to make the decision jointly together(ASML_10). This could be a consequence of the architect being dependent on the correct execution of the concepts by the executing parties. All of these examples show how decision quality as a behavioral competency does have its orientation in the technical side where it could be perceived as part of the architecting itself.

This being part of the architecting itself could also be considered for the behavioral competency manages complexity. Managing vast amounts of information comes in many forms, but based on

chapter 3 is also a critical skill for the architect. The argument could be made where manages complexity is part of the architecting itself because all information is translated into technical solutions. In one of the examples, ASML_09 used his technical knowledge and ability to manage the different information to see how a set of requirements were not actually the requirements. By talking to people the architect was able to go back to the history and motives behind the creation of these requirements. In a certain way, managing the different information streams and coming to a realization based on this exemplifies how manages complexity can also be viewed as a cognitive ability of the architect. Without the deductive skills with all different information and knowledge, this realization would not have been.

A different way is for the architect to manage the complexity of a system in smaller pieces to understand and enable others to understand it(ASML_03). Alternatively, an architect can split up larger improvements into smaller chunks to make them more feasible or make it easier to convince others to commit resources to the change(ASML_01). Splitting up could also be important because otherwise having too many details, would increase the complexity and harm the project because they cannot be managed(ASML_04).

Even though different, the examples above show why manages complexity is so important for an architect. Without it the architect would not understand the systems, without it the architect would not know which would be the best options or what the interests and dependencies are. Managing complexity represents the transfer of raw knowledge into useful knowledge in the context of the technical leader, the software architect. It is related to all other behavioral competencies and can be seen as the inexplicable way in which architects are successful.

4.3.2.3 Instills trust as a pre-requisite

Another interesting finding was the role of instills trust. Instills trust is seen as gathering the confidence and trust of others. In the interviews, trust was regularly present in the form of being a pre-requisite. On many occasions, stakeholders mentioned how they trusted an architect. In one of the examples stakeholder ASML_07 explained how architects have to make the right calls and decisions. However, they need the people involved and mandate to get the trust to do this. This trust was created based on experience and track record(ASML_07). In addition, ASML_07's trust was also influenced when he saw an architect being seen makes an impact on the project. This indicates how trust is a boundary for success. However, most of the time the architects themselves did not directly address trust themselves. They do indicate behavior that tries to build this trust like ASML_05 who mentioned: "I'll talk to them, I make them feel welcome, lower my own boundaries for them so they can talk to me or say there's not this high ranking architect somewhere that's people don't directly talk to you are approachable, so I spent more time there.". One architect did specifically mention how you have to build trust, otherwise, they will not believe you and it will make convincing others a lot more difficult(ASML_03). As a side note, this architect was one of the more experienced architects.

The dynamic of trust seems to be that of a pre-requisite. It builds over time and seems affected by experience and your track record. However, based on the information from ASML_03 and the stakeholders, it seems like it is more of a problem when there is no trust, making it hard to convince others. In success situations, other architects almost naturally build trust like ASML_05 but their motive seems more to be approachable rather than use it consciously for convincing others.

4.3.3 What performance outcomes indicate success

One of the 'other' findings was the exploration of the term 'success' and what type of performance outcomes would indicate success. The interview provided information on what it means to be successful as a software architect. A few of the interviews mentioned factors of the classical triangle cost, quality, and time. However, the interesting part is how architects focus more on the long-term technical view. Architect ASML_03 explained how sometimes stakeholders want to finish a project quickly, which is the time of the classical triangle. However, architects seek the best (technical) solution, trying to avoid technical debt. In this way, ASML_03 split the difference between the long-term perspective on the success of architects and the short-term perspective on the success of stakeholders. This example is enforced by ASML_05 who explains how a faster machine being a more complex machine with fewer options to expand on it, would not necessarily be chosen over a less complex and more expandable machine. This, however, contrasts with the perception of architect ASML_10 who mentioned happy customers, meeting expectations, having a happy crew, and doing it in a short timeframe as the reasons why a certain project was successful from his perspective. One explanation could come from ASML_04 who mentioned how an architect needs quite a lot of awareness and knowledge to realize what success for a project means in a specific architectural context.

The central takeaway here would be how outside of the classical triangle cost, time, and quality, other performance outcomes indicating success can also be present. From the interviews, the content of stakeholders or the own team could also be seen as a performance outcome. Additionally, and maybe even more important for the software architect specifically, would be the future perspectives of the project. Architects seem to focus on the long-term technical roadmap which also takes into account the maintainability and expansion opportunities of the project. In general, architects try to avoid unnecessary dependencies and technical debt.

4.3.4 The role of knowledge in the software architect

In the literature review, one of the sources addressed the perspective of the architect being a knowledge manager rather than a decision-maker(Weinreich & Groher, 2016). The interviews showed how knowledge plays an important role in the success of a software architect. Previously, manages complexity already partly relied on knowledge. Instills trust, was sometimes based on knowledge or the perceived previous knowledge in the form of a track record. Knowledge almost seems like a tool present in decision situations but also in own cognitive processes, like with manages complexity. When gathering information, knowledge is generated. When giving information, knowledge is transferred. And in between knowledge is churned. In the interviews, various connections between knowledge and behavioral competencies. Previously, the architect its role and responsibility had been the main focus, however, the term popped up often enough to get noticed to indeed indicate how sometimes the architect almost seems a knowledge manager. Steering and arguing, using knowledge as the tool to get things done. ASML_05 referred to how communicates effectively boils down to transferring information. The importance of (domain) knowledge in the architect is exemplified by having very few people who actually start as an architect(ASML_04) suggesting previous experience is needed. On the other side, having too much knowledge can also be harmful as ASML_09 expressed an example of avoiding getting more knowledge because otherwise, he would not be able to manage this stream of information. This

latter example indicates a connection between knowledge and behavioral competency manages complexity.

This additional finding should not come as a surprise considering half of the description of the architect being focused on gathering input and giving output(Kruchten, 2008). This came forth in the interview with ASML_10 who mentioned how all the pre-alignments were held to gather all the information before the definitive meeting itself. This example was previously used in support of the behavioral competencies collaboration and manages complexity. Also, remember how ASML_03 showed how knowledge was operationalized by managing the various information streams to avoid bad decisions which addressed how reviewing documents allows for the verification of the ideas being in line with the roadmap. This exemplified how knowledge plays an important role in identifying opportunities and problems. The architect has to know about specific company elements, software engineering concepts, and its own architectural elements.

The results do not give any conclusive evidence, however, they do indicate how knowledge is important for software architects. In addition, knowledge seems to have relations with the behavioral competencies manages complexity, collaboration, and communicates effectively.

4.4 summary interview chapter

To know what behavioral competencies are important, it is also required to know how these behavioral competencies lead to success. This chapter confirms the findings of the four important behavioral competencies found in chapter 3, all of these are present in all the interviews. The interviews do not provide any further information about the clusters from chapter 3. Communicates effectively happens both when gathering information and when convincing others. Manages complexity has close ties to managing all knowledge of the architect. However, It mainly focuses on managing the input of information and managing what information to give to others. Collaborates and decision quality are linked via joint decision making where making the decision together is related to the execution of the decision but is guided by the architect. Collaborates is also important for an architect to get information from others to get a more complete overview. The other side of decision quality focuses on the individual decisions an architect makes like wanting to make a change or deciding to intervene with others. Knowledge seems to play a role in a lot of these behavioral competencies, however, not enough information was available to make a decisive conclusion about how exactly.

Chapter 5 Conclusion

Since there were multiple different elements in this thesis let's summarize. This research tried to answer the question what are the most important behavioral competencies of a software architect. This chapter will summarize the findings of both the questionnaire and the semi-structured interviews together.

5.1 Top cluster

As expected, the most important behavioral competencies are manages complexity and communicates effectively. The questionnaire showed how they were placed number 1 and number 2 without exception, even in the different sub-groups there was this commonality forming a part of the answer to exploration 1 and 2. In addition, these formed a cluster, making them both the top cluster of behavioral competencies. This provides support for hypotheses 1 and 2.

In addition, as shown in figure 5a, the interviews provided insights into how communicates effectively and manages complexity both seemed to have multiple purposes within the different architecting activities. Architects needed to communicate effectively to both gather information and to convince others. Architects cannot know everything, therefore they rely on the input of others. However, they also rely on others to follow and execute the different architectural decisions. In both cases, the architect should use a multimode of communication methods and tools to convey their message depending on the interlocutor. To be able to know what to ask but also know what to say to convince others, the architects do need knowledge about their conversation partners and their interests.

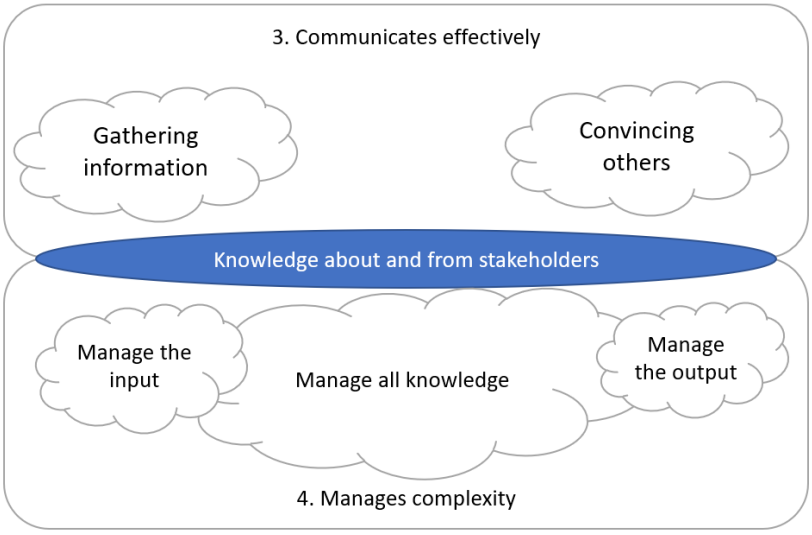

Figure 5a: The interview findings regarding Manages complexity and communicates effectively

This knowledge of conversation partners is also present in the managing complexity behavioral competency. In all facets of the job, an architect deals with information. This would also create the main argument on how managing complexity could be considered as part of the architecting of the job. In almost all situations the architect needs to manage all information available from different stakeholders, but also from the domain and the architect's personal knowledge. Managing the complexity would be

required for the incoming information to know what to ask. But also in the outcoming information where it is required to know what to say to convince others.

5.2 The other two important behavioral competencies

The answer to the most important behavioral competencies is answered above. However, this does not mean the other behavioral competencies are not important. The second cluster of important behavioral competencies was composed of collaboration and decision quality.

In chapter 3, Decision quality took the 3rd spot, which seemed to support hypothesis 3. However, Decision quality was not clustered with communicates effectively and manages complexity. It was clustered with the 4th ranked collaboration. This immediately rejects hypothesis 3, decision quality as 'most' important, since 'most' was defined as being clustered together at the top. IMPORTANT this does not mean it is not important, it is just not the 'most' important. Considering the subgroups, there was an almost significant difference between architects and stakeholders considering decision quality. Since it is not significant it would suggest rejecting stakeholders ranking decision quality higher, hypotheses 4. However, in absolute ranking, there is a difference where collaboration and decision quality are switched around. Between the architect-levels, this seemed to be mainly from the product-level architects, with the difference in Decision quality ranking with system-level architects being almost significant too. In chapter 4, decision quality happened in various situations and is related to a lot of the other important behavioral competencies. Regarding the decision quality itself, there were two types of decisions: individual decisions and guiding the decisions in a joint decision making. This could explain why there are differences in how decision quality is viewed. In the end, hypothesis 4 cannot be supported as the difference is not(yet) significant. However, with there being visual evidence, the expectancy of becoming significant with increased architect sample size and different types of decisions, fully rejecting this hypothesis would seem irresponsible in an explorative setting like this. Evidence surrounding hypothesis 4 is inconclusive.

The other behavioral competency of the 2nd cluster is collaboration. In the overall absolute ranking, it takes the 4th spot but overall the architects would place it on the 3rd spot. In general, this would seem to support hypothesis 5, collaboration being valued more by stakeholders than architects. Interestingly though is where product-level architects value collaboration on the third spot, system-level architects value decision quality as 3rd with collaboration being 4th. This is the same ranking as the agile roles and engineers stakeholder subgroups. In addition, the management stakeholder subgroup values collaboration as the 3rd rank, over the 4th of decision quality.

However, this difference between architects and stakeholders is not significant and is not expected to become significant with increased sample sizes. In addition, the values with which collaboration is scored is similar between both groups. This, in combination with the difference being not significant, rejects hypothesis 5. Collaboration is still part of the 2nd cluster of important behavioral competencies being either the number 3 or number 4. Just like communicates effectively the dynamics of collaboration are both in the gathering input and giving input side. Working together happens both to get the information input and create the overview via meetings, informal chats, and other methods. But when a plan or decision is made, collaboration is also required to create a successful implementation.

The execution of decisions is also seen as a collaborative process where the architect is open to changes. Guiding people towards the change and making the decision together is an important part of creating support for the execution itself and was often deemed as an important part of success. This joint decision-making seems a combination of collaborative and decision-making competencies for a software architect.

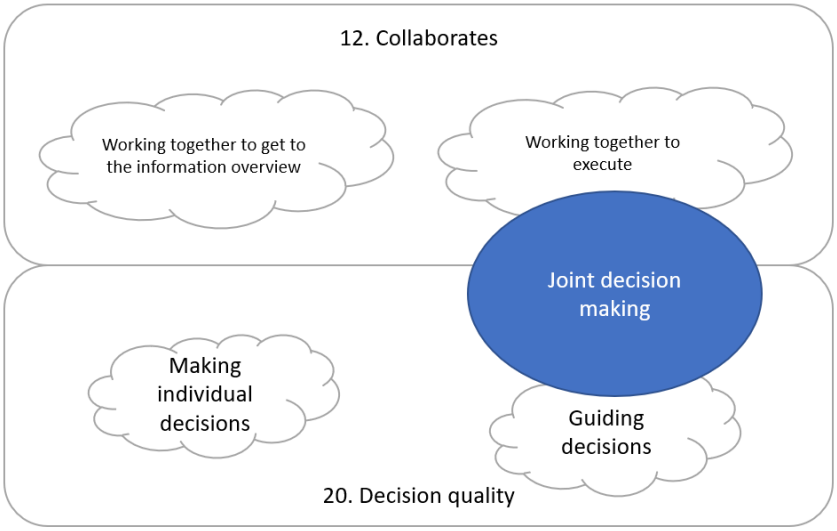

Figure 5b: The interview findings of collaborates and decision quality

5.3 Never just one behavioral competency

The previous two paragraphs exemplify the finding of chapter 4. In success situations, there are nearly always multiple behavioral competencies present in the behavior of a software architect. This is a logical outcome when considering how all the different architectural activities are needed to create a successful situation. As exemplified by the interaction between manages complexity and communicates effectively, if there is no communication, the information input will be very limited. However, if the complexity is not already managed to some extent, it will be very difficult to structure the information input in an effective way to get the right input.

When considering the role each behavioral competency plays, the important behavioral competency Decision Quality exemplifies how a behavioral competency could play a central role in forming a central point from which to work to of from. Another found dynamic was how the 5th ranked behavioral competency Instills Trust could be a pre-requisite. In the interviews trust, based upon both knowledge but also personal relation and track record, seemed to form the base on how stakeholders will view the words of the architects. Just like it is impossible to know all the details for the architect, it is impossible to know all the different forces and choices as a stakeholder. Stakeholders need to trust the architect to accept this larger unknown. In the behavior of architects, it becomes clear how they try to be approachable to generate this trust to do their job. The consequence of the loss of trust could be the failure of a project. However, in the context of success, trust seems to function more like a pre-requisite whereas in success situations the architects have trust.

5.4 Additional findings of knowledge management and performance outcomes indicating success

One of the additional observation of chapter 4 indicated how knowledge management seemed to have an important role in the successful behavior of a software architect. One other additional question posed in chapter 4 was how behavioral competencies contribute to success. To answer this question one sub-question asked what performance outcomes would indicate success. The interviews provided the following potential performance outcomes: classical triangle elements namely cost, quality and time. These were always to consider, however, there were also interviews in which architects themselves focused less on time and cost, but more on their future potential.

Chapter 6 Discussion

Chapter 5 combined the different findings from chapter 3 and chapter 4. This chapter will discuss how the findings relate to existing bodies of literature. The quick and dirty answer to the research question: what are the most important behavioral competencies of a successful software architect. The most important behavioral competencies of a software architect are manages complexity and communicates effectively. However, there is a second cluster consisting of collaboration and decision quality which can also be considered important. Furthermore, instills trust seems to be a pre-condition for success.

6.1 Theoretical implications

This research contributes by expanding the knowledge about the behavior of software architects within the behavioral software engineering domain(Lenberg et al.,2015; Baltes & Diehl, 2018). In general, the findings have been in line with the previous work of Kruchten(2008) and Razavian & Lago(2015). One important contribution of this research provides insights into practitioners' perspectives on which behavioral competencies are important but also gives insights in the context of these behavioral competencies of successful software architects. This allows for the mapping of these competencies on the architecting activities. In addition, the performance outcomes and knowledge management perspectives will be discussed shortly.

6.1.1 Mapping behavioral competencies on the architecting activities

The full overview of the mapped behavioral competencies can be found in figure 6a.

6.1.1.1 Internal and external activities

The role of communication was been widely regarded as important. As it was one of the key duties of the architect(Bass et al., 2008; Clements et al., 2007; Kruchten, 1999, 2008; Razavian & Lago, 2015). Where the internal framework of the case company mainly defines communication as externally focused(Kornferry, n.d.), other sources have split communication into outward, inward communication(Bass et al., 2008; Clements et al, 2007) or getting input & providing information(Kruchten, 2008; Razavian et al., 2015). This research supports how communicates effectively is one of the most important behavioral competencies. However, as seen from the interviews, communicates effectively is a required competency to get information and also for executing and convincing others of the architecture. This would place the used behavioral competency communicates effectively in both the internal and external focus of the architecting activities map(Kruchten, 2008). This would contrast the placement of communication in only the outward communication(Razavian & Lago, 2015).

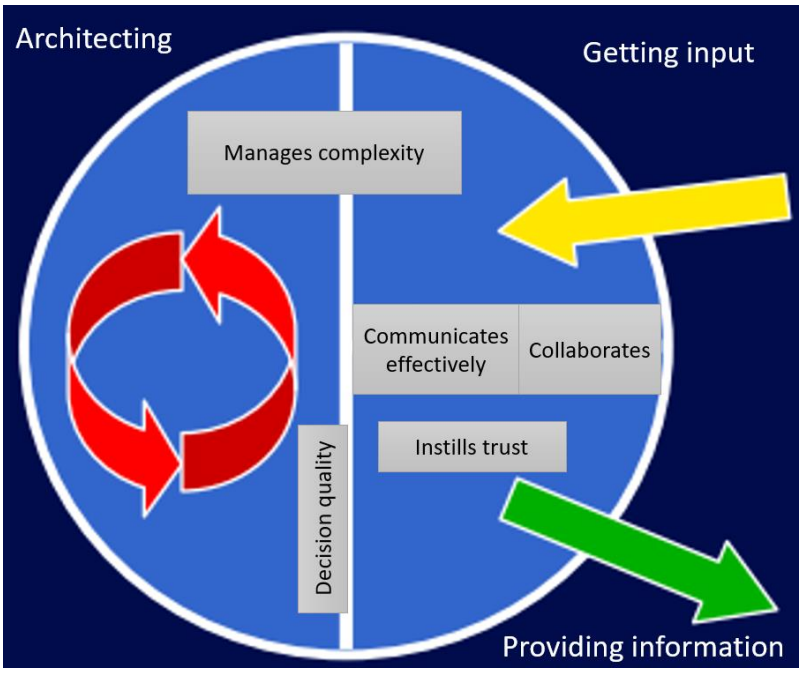

Figure 6a: Top 5 behavioral competencies mapped on architecting activities (Kruchten, 2008)

A similar argumentation would be used for collaboration. Where collaboration was initially not expected to be the most important behavioral competency, it was in the 2nd cluster of competencies. This cluster was still considered to contain important behavioral competencies. In addition, in contrast to what was expected, collaboration seemed to be important for both architects and stakeholders. This could be explained with the support of the interviews where collaboration included examples of the preparation and iterations of proposals and decisions, but also took a role in convincing and actually executing proposals and enforcing decisions. In the activities map, this would provide a similar position as communicates effectively in both internal and external focus sections. This importance of collaboration for architects could have been expected since they depend on others for their information and execution. In addition, the role of being a team player, and working effectively with superiors, colleagues, and supporters had been deemed an important skill in literature(Bass et al., 2008). The differentiating between working together to get the information overview and working together to make the idea happen did not get a lot of attention. These dynamics require further investigation in the future.

6.1.1.2 Architecting activities

Chapter 4 showed how manages complexity and decision quality were both considered to be technical dynamics. Manages complexity was one of the behavioral competencies which found close ties with other behavioral competencies. Manages complexity had a lot to do with information, as it was defined as managing large sums of, sometimes contradictory, information. This managing information input and output was expected from literature(Kruchten, 2008). However, architects should also have knowledge in the computer science domain, about existing technologies and platforms, and about the organization's context and management(Bass et al., 2008a). These types of knowledge are not only

technical in nature but also focus on key players and their intentions within the company(Bredemeyer, 2002). Managing vast amounts of knowledge is the backbone of the architect. This also shows why this behavioral competency is so important. Without managing the complexity, an architect will have no clue what is going on and how things will affect the system. Interviews mentioned how architects do not only guide other technically, but also have to convince other by giving alternatives and showing their consequences. If the architect did not manage all the information, both conversation and documentation, the architect will not be able to have the overview, let alone see the consequences. Where the gathering of the information requires different behavioral competencies, the management of complexity is directly related to the architecting activities cognitive process. Therefor, this research would argue manages complexity to be mainly part of the architecting activities, but requiring the information input, directly emphasizing why this relation between gathering input and architecting is so important.

Just like manages complexity, decision quality has a lot of dynamics with other behavioral competencies. Mainly to get to the decision quality other behavioral competencies are required, and in the execution of this decision it is also the other behavioral competencies that present themselves. Chapter 4 showed how decision quality could be related to both individual and group decision-making. Literature also considers both sides where decisions about the architecture can be seen as part of the individual software architects decision-making process(Erder & Pureur, 2016) and the role of the architect as a joint decision-making facilitator(Rosa et al., 2020; Tang et al., 2017). As the technical leader(Bass et al., 2013; Britto et al., 2016) who is busy with architecting activities half of the time(Kruchten, 2008) it would seem logical to have to make individual decisions as an architect. However, a few interviews emphasized how an architect can not know everything. Joint decision making was also addressed to have a direct relation with collaborating, addressing the collaboration in the execution and formation of a decision. In a sense, this is more linked to solving a problem on how to make a certain change rather than what exact steps have to be taken. The software architect is known to be a problem solver(Bass et al., 2008b; Clements et al., 2007; Fowler, 2003).

But what if there is both individual and joint decision-making? In a lot of the interviews, it was implied that an architect already found a gap. And when the architect decided to act on it a joint decision process was created to fill in the details. If joint decision-making is the active part where the architect has to listen and tweak the original decision to make a change, the decision to attempt a change of direction would be an individual decision. This individual decision would be highly related to managing large sums of information. The architect would fulfill the technical leader role in the joint decision process, guiding it towards acceptable decisions from a system or product level perspective. This would create the concept of decision quality actually originating more in the technical side of the software architect and representing the cognitive and deductive ability of the architect. Thus decision quality would mainly be part of the architecting activities but be actively producing decisions that are operationalized in the information output. Note how this also allows for the decision to be tweaked based upon the collaborative efforts afterwards, but the activation of the decision has already happened.

6.1.1.3 pre-condition trust

Chapter 4 argued that trust is a pre-condition and therefore instills trust also being part of creating the setting of that pre-condition. By having trust, stakeholders trust the architect's decisions and knowledge. This was found to have a basis in previous projects outcomes and argumentation of the architects. Existing literature had indicated the architect needed to be trusted advisors(Bredemeyer & Malan, 2002; Eeles, 2006; Klein, 2016), however, this has a direct link to the consultant antipattern(Kruchten, 2008). This research would argue this trust to go further than suggested by the literature. Trust is mainly required to successfully execute decisions, going further than just being an advisor. Both literature and these findings would place trust in the outwards communication part of the architecting activities, based upon this execution and convincing element. However, there are also cases where trust could contribute to the gathering of information due to it making the architect approachable.

6.1.2 Differences between system-level and product-level architects

One of the other findings was how there were indications of decision quality being different between system-level and product-level architects. This finding fit existing frameworks where higher-level architects are busy with higher-level decision-making (Martini et al., 2014) and in a certain sense, the product-level architects are enforcing and executing those high-level decisions. This could explain why decision quality would be viewed differently between the two groups. In addition, following the architect classifications of Fowler(2003), system-level architects were more pure decision-makers whereas product-level architects were both decision-makers and problem-solvers(Britto et al., 2016). With exploration 1, no difference was expected in behavioral competencies but rather in execution, this now seems to be challenged. However, product-level architect work closer to the work floor requiring them to work closer with these people(Britto et al., 2016; Martini et al., 2014). In addition, they suggest a more individual decision-making style compared to what has been found regarding joint decision-making. More research is required to see if the different hierarchal layers are different, and this research provides a good starting point: decision quality.

6.1.3 performance outcomes indicating success

The additional findings regarding the performance outcome consisted of cost, quality and time(triangle), stakeholder satisfaction and future plans, maintainability, and reducing dependencies(future perspectives). Even though this was not the main aim of this research, the role of knowledge for an architect has been discussed before. An architect should be knowledgeable in their domain (Berenbach, 2008; Kruchten, 1999) and over 100 knowledge areas for practicing architects have been identified (Clements et al., 2007). It is even discussed how storing and expanding architectural knowledge are non-technical activities of a software architect(Farenhorst & Van Vliet, 2009). This follows the point of view that the architect should be less of a lonesome decision-maker but focus more on their nature of being a knowledge sharer(Hoorn et al., 2011). This literature did not address what performance outcomes would indicate success. When broadening the view to generic forms of success like project success the findings do seem explicable. In the generic perception of project management success, there is indeed the triangle of quality, cost, and time(Atkinson, 1999; Baccarini, 1999; Collins & Baccarini, 2004; Westhuizen & Fitzgerald, 2005). The findings of satisfaction of stakeholders would be part of project management success in the different subdimensions of software project success(Sudhakar, 2012). The

findings of this research would argue for this future perspectives dimension to be considered a very important performance outcome that indicates the success of a software architect. The role of the software architect is being responsible for system integrity, within this integrity, scalability can be considered part of the responsibility(Bondi, 2009). This research does not give any conclusive evidence nor was it the intention of it, it does indicate how the future perspectives are an important performance outcome for architects specifically. With the amount of literature about (software) project success, it is very viable to do a study on what project success entails for software architects specifically.

6.1.4 Role of knowledge management

The interviews in chapter 4 do find indications of the dynamics of knowledge management in the role of software architects. There were indications on how knowledge management is important for software architects and how there seems to be a link between knowledge and manages complexity, collaboration, and communicates effectively. The importance of knowledge in the role of the software architects has been discussed(Babar et al., 2009; Capilla et al., 2016; Bass et al., 2013; Clements et al., 2007) and some even go as far as mentioning the software architect being a knowledge manager over a decision-maker(Weinreich & Groher, 2016). Existing literature has considered how a combination of collaboration, with subpart joint decision making and communication, and knowledge management are beneficial for successful design of software architectural solutions(Sherman et al., 2016), potentially explaining why these connections are there. This research does not give any definitive answers, it does indicate how there is a possible relation between behavioral competencies and knowledge management.

6.2 Managerial implications

One of the most important aspects of research and specifically research in innovation management is the application for practice. The industry played an important role in the origin of this research, what did they gain from the answer to the question of which behavioral competencies are important for a software architect?

First of all, it shows the order in chaos by showing that the top 2 and maybe even the top 5 behavioral competencies are consistent. The list can be used to express to architects what is expected and guides the direction for new architects of really building the network, use your technical knowledge to support and guide stakeholders, and value the informal and formal communications. The base of work is trust which consists of the technical contributions but also your way of making these contributions. Especially in larger organizations, like the case company, these secondary elements can be the difference between success and failure.

In addition, where stakeholders are in agreement over the importance of communicates effectively and manages complexity, the priority in seeing collaboration and decision quality is different. The perception of decision quality potentially lies within the product-level architects who see decision quality as less important whereas system-level architects rate it more similar to collaboration. This information could be valuable when considering moving architects up the ladder. This research did not produce a conclusive explanation, it is advised to work on the network and gather the perspectives of stakeholders(which they do value as it is part of collaboration). This, however, should be explicitly linked

to the decision quality where decision quality seemed to be relevant since it directly impacts the stakeholders.

Literature takes the perspective of individual decision-making or joint decision-making when talking about decision-making, but this research would suggest using both. As an individual, it is necessary for the architect to make technical decisions in line with the roadmap. In addition, the architect should be able to decide on how to split something into smaller more conceivable chunks. However, architects should know how to incorporate alternatives or other ideas when a good reason presents itself from stakeholders. To get to know the reasons it is required to make this decision together with stakeholders, the joint decision making. Furthermore, architects should provide clear and sound reasoning for stakeholders to relay their idea. If the reasoning is not sound, or there are major objections the execution of a decision could become faulty. To create the best decision successful architects often persuade stakeholders by involving them in the decision process and guiding them towards acceptable technical decisions.

Managers can anticipate there being both individual and joint decision-making for software architects. Especially in the joint decision-making process it is highly recommended to guide teams into a position that allows them to trust architects. Experienced architects can rely on their track record or by showing their knowledge. Managers should specifically support starting architects to build a network, but also create some form of success in smaller projects before moving to larger projects facilitating this 'track record'.

In general, stakeholders should realize the value of architects. They are there to avoid larger problems in the future but their work process is slow and methodical. Structurally an architect should be able to gather the right information, which is in the best interest of everyone. If there really is something the architect should be about, a climate should be created in which this is normal to do, stakeholders in a sense bear the responsibility to provide the architect with useful information. This is really labor-intensive and something which is not present in the findings but is important is the pressure on architects. Good architects have to talk to a lot of people for alignments, while still fulfilling their role to actually do the architecting work, which is half of their job following the findings of Kruchten(2008). This research does not provide concrete evidence on how to structure this. What this research can advise is to create opportunities for the architect to be able to do the aligning, collaborating, and network building, which came forward as important behavioral parts of the software architect.

6.3 Limitations and future research

First of all, earlier this chapter a finding was presented that indicated how multiple behavioral competencies are used in situations in which the software architect was successful. This is a clear opportunity for future research to search for the clustering or packages of behavioral competencies. Next to this, a large limitation was the implication of the term success. By not defining success there is some ambiguity in what the participants saw as success. The interviews did provide some exploration of performance outcomes which seemed in line with general literature about project success, as seen in paragraph 6.1.3. But more research is required to find what is seen as a success for a software architect specifically, and what it exactly affects.

One of the issues with case studies is the difficulty in distinguishing what is generalizable and what is contextual. The demographics seemed to be in line with previous findings(Razavian & Lago, 2015), and in general, the literature confirmed the findings. There are still open discussions like how communicates effectively could be split into either one, two, or three segments. However, in any case, these segments would be important too. What is difficult is to distinguish how much these results are influenced by the culture of ASML. Even though these findings are relevant and useful, more case studies are required to verify if these findings are in different contexts. Tied into this is how the differences between different groups regarding Decision quality were very close, but not significant. This is expected to become significant with an increased sample size of architects, however, this might be a problem since the existing response(n=220) was already considered high. In addition, a lot of the total amount of architects responded(n=59) which is almost a third of the 161 responses of stakeholders. The differences in group sizes affect the absolute rankings of groups where 1 subgroup could influence the whole overarching group. This might have happened in the explorative part with the stakeholder subgroups where the management subgroup only had 18 responses but these responses indicated something slightly different from the other stakeholder subgroups. More data would be required to distinguish more subgroups and create large enough sample sizes to compare these subgroups. Future research could not only help verify these findings but could also contribute more data to create explore more groups and subgroups. In addition to this, the sub-groups of stakeholders were not based on literature, but rather on deduction based on their role. Future research could focus on mapping the different types of stakeholders of a software architect, however, this would deviate from this line of research. Another group-related limitation was the exclusion of team-level architects from the scope. Even though this was the right choice, excluding it was still a limitation.

Additionally, this research did not include which architects the different stakeholders worked the most with and vice versa. Even though this would only widen the scope and not directly affect these findings, it could show more potential differences between the architect subgroups.

Next, even though the choice of the definitions is warranted it should be considered how these came to be. Where consultancy firms base their models on experience and literature, they are not transparent. Using something scientifically proven like Bartram(2005) would give a lot of drawbacks, but would give more scientifically proven terms. As discussed before, there are enough similarities and the behavioral competencies used by Kornferry(n.d.) are a lot less generic but still, it could be seen as a limitation.

Finally, conducting semi-structured interviews always has subjective elements, even though as a researcher everything is done to make these as objective as possible. Even though the eleven interviews showed certain similarities, the results still only give direction and potential context instead of hard evidence. Future research should focus on exploring how the important behavioral competencies function in different contexts.

6.4 Final summarization

There is a simple answer to the research question, what are the important behavioral competencies of a successful software architect. Manages complexity and communicates effectively are the clear top cluster followed by collaboration and decision quality. However, the simple answer is just a first step.

With the help of the interviews, the start of a next step was taken in uncovering how these behavioral competencies show themselves. The interviews allowed for the mapping of the found important behavioral competencies on the different architecting activities(Kruchten, 2008). Both communicates effectively and collaboration were needed in information gathering and information providing architect activities. Manages complexity and decision quality both were required to do the architecting activities, but had different forms depending on if it was part of the problem finding or solution creation. With all examples of successful software architects, multiple behavioral competencies seemed to be important. This research contributes to the existing understanding of the behavioral competencies of software architects, but so much more is there to be found. It can be said how this only seems the tip of the iceberg in understanding what and how behavioral competencies lead to success for a software architect.

References

Agarwal, N., & Rathod, U. (2006). Defining "success" for software projects: An exploratory revelation. *International Journal of Project Management*, *24*(4), 358–370. https://doi.org/10.1016/j.ijproman.2005.11.009

Ahmed, F., Campbell, P., Beg, A., & Fernando Capretz, L. (2015). *Soft Skills Requirements in Software Architecture's Job: An Exploratory Study*.

ASML. (n.d.-a). *Internal document 20 leadership competencies*. https://my.asml.com/sectors/es/hro/peopledevelopment/myGrowth/Pages/20 competencies/Our 20 Leadership Competencies.aspx?FilterField1=Location&FilterValue1=The Netherlands&FilterField2=Company&FilterValue2=ASML

ASML. (n.d.-b). *Internal document behavioral competencies*. https://my.asml.com/sectors/es/hro/peopledevelopment/myGrowth/Pages/Competencies/Competencies.aspx?FilterField1=Location&FilterValue1=The Netherlands&FilterField2=Company&FilterValue2=ASML

ASML. (n.d.-c). *Products: EUV lithography systems*. https://www.asml.com/en/products/euv-lithography-systems

ASML. (2020). *2020 Anual Report*. https://www.asml.com/en/investors/annual-report/2020

ASML. (2022). *European Chips Act- ASML position paper*. https://www.asml.com/en/news/press-releases/2022/asml-position-paper-on-eu-chips-act

Atkinson, R. (1999). Project management: Cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, *17*(6), 337–342. https://doi.org/10.1016/S0263-7863(98)00069-6

Babar, M. A., Dingsøyr, T., Lago, P., & Van Vliet, H. (2009). Software architecture knowledge management: Theory and practice. In *Software Architecture Knowledge Management: Theory and Practice*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02374-3

Baccarini, D. (1999). The Logical Framework Method for Defining Project Success. *Project Management Journal*, *30*(4), 25–32. https://doi.org/10.1177/875697289903000405

Baltes, S., & Diehl, S. (2019). Towards a theory of software development expertise. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI)*, *P-292*, 83–84. https://doi.org/10.18420/se2019-22

Bartram, D. (2005). The great eight competencies: A criterion-centric approach to validation. *Journal of Applied Psychology*, *90*(6), 1185–1203. https://doi.org/10.1037/0021-9010.90.6.1185

Bass, Clements, P. C., Kazman, R., & Klein, M. H. (2008). Models for evaluating and improving architecture competence. *Software Engineering Institute*, *March*, 87. http://repository.cmu.edu/sei/308

Bass, L., Clements, P., & Kazman, R. (1997). *Software architecture in practice*. Addison-Wesley Professional.

Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice* (3rd ed.). Addison-Wesley Professional. https://books.google.nl/books?hl=en&lr=&id=ZY6UZTjBnGQC&oi=fnd&pg=PA1&dq=+software+architecture+in+practice&ots=gspt_rO1kg&sig=TEzScUKzLDsCnYBPj-vcxafn-Wk#v=onepage&q=software architecture in practice&f=false

Bass, Len, Clements, P., Kazman, R., & Klein, M. (2008). Evaluating the software architecture competence of organizations. *7th IEEE/IFIP Working Conference on Software Architecture, WICSA 2008*, 249–252.

https://doi.org/10.1109/WICSA.2008.12

BBC. (2021). *Car productino hit by "pingdemic" and global chip shortage*. https://www.bbc.com/news/business-58002724

Berenbach, B. (2008). The other skills of the software architect. *Proceedings - International Conference on Software Engineering*, 7–11. https://doi.org/10.1145/1373307.1373310

Blumberg, B., Cooper, D. r., & Schindler, P. s. (2011). *Business research methods*. McGraw-Hill Education.

Bondi, A. B. (2009). The software architect as the guardian of system performance and scalability. *Proceedings of the 2009 ICSE Workshop on Leadership and Management in Software Architecture, LMSA 2009*, 28–31. https://doi.org/10.1109/LMSA.2009.5074861

Bredemeyer. (2002). *Technology Consulting Strategy Leadership Organizational Politics*. 2002.

Bredemeyer, D., & Malan, R. (2002). *The Role of the Architect*. http://www.bredemeyer.com

Britto, R., Šmite, D., & Damm, L. O. (2016). Software Architects in Large-Scale Distributed Projects: An Ericsson Case Study. *IEEE Software*, *33*(6), 48–55. https://doi.org/10.1109/MS.2016.146

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., & Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, *116*, 191–205. https://doi.org/10.1016/j.jss.2015.08.054

Clements, P., Kazman, R., Klein, M., Devesh, D., Reddy, S., & Verma, P. (2007). The duties, skills, and knowledge of software architects. *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*, 20–23. https://doi.org/10.1109/WICSA.2007.41

Collins, A., & Baccarini, D. (2004). Project success - A survey. *Journal of Construction Research*, *5*(2), 211–231. https://doi.org/10.1142/S1609945104000152

Eeles, P. (2006). Characteristics of a Software Architect. *The Rational Edge, IBM Resource*, 1–7. http://research.cs.queensu.ca/home/ahmed/home/teaching/CISC322/F08/files/CharacteristicsOfASoftware Architect.pdf

Erder, M., & Pureur, P. (2016). What's the Architect's Role in an Agile, Cloud-Centric World? *IEEE Software*, *33*(5), 30–33. https://doi.org/10.1109/MS.2016.119

Erder, M., & Pureur, P. (2017). What Type of People Are Software Architects? *IEEE Software*, *34*(4), 20–22. https://doi.org/10.1109/MS.2017.103

Farenhorst, R., & Van Vliet, H. (2009). Understanding How to Support Architects in Sharing Knowledge. In *International Conference on Software Engineering (31st : 2009 : Vancouver, B.C.)* (pp. 17–24). IEEE.

Farshidi, S., Jansen, S., & van der Werf, J. M. (2020). Capturing software architecture knowledge for pattern-driven design. *Journal of Systems and Software*, *169*, 110714. https://doi.org/10.1016/j.jss.2020.110714

Ferrari, R., Madhavji, N. H., & Wilding, M. (2009). The impact of non-technical factors on software architecture. *Proceedings of the 2009 ICSE Workshop on Leadership and Management in Software Architecture, LMSA 2009*, 32–36. https://doi.org/10.1109/LMSA.2009.5074862

Field, A. (2013). *Discovering statistics using IBM SPSS statistics* (4th editio). SAGE Publications Inc.

Fowler, M. (2003). Who needs an architect? *IEEE Software*, *20*(5), 11–13. https://doi.org/10.1109/MS.2003.1231144

Gill, P., Stewart, K., Treasure, E., & Chadwick, B. (2008). *Methods of data collection in qualitative research: interviews and focus groups*. *https://do*.

Hoorn, J. F., Farenhorst, R., Lago, P., & Van Vliet, H. (2011). The lonesome architect. *Journal of Systems and Software*, *84*(9), 1424–1435. https://doi.org/10.1016/j.jss.2010.11.909

Klein, J. (2016). What makes an architect successful? *IEEE Software*, *33*(1), 20–22. https://doi.org/10.1109/MS.2016.9

Kornferry. (n.d.). *Interally used framework: selection of 20 behavioral competencies as found in ASML(n.d.-b;n.d.-c) out of the 39 competencies framework of Kornferry*. https://neprisstore.blob.core.windows.net/sessiondocs/doc_10cd2119-d549-47ac-8784-215eb85cdc8f.pdf

Kruchten, P. (1999). The software architect. *Working Conference on Software Architecture*, 565–583. https://doi.org/10.1145/1230819.1241667

Kruchten, P. (2008). What do software architects really do? *Journal of Systems and Software*, *81*(12), 2413–2416. https://doi.org/10.1016/j.jss.2008.08.025

Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*, *107*, 15–37. https://doi.org/10.1016/j.jss.2015.04.084

Manteuffel, C., Avgeriou, P., & Hamberg, R. (2018). An exploratory case study on reusing architecture decisions in software-intensive system projects. *Journal of Systems and Software*, *144*(May), 60–83. https://doi.org/10.1016/j.jss.2018.05.064

Martini, A., Pareto, L., & Bosch, J. (2014). Role of architects in agile organizations. *In Continuous Software Engineering (Pp. 39-50). Springer, Cham.*, *9783319112*, 39–50. https://doi.org/10.1007/978-3-319-11283-1

Mendes, F., Mendes, E., Salleh, N., & Oivo, M. (2021). Insights on the relationship between decision-making style and personality in software engineering. *Information and Software Technology*, *136*(September 2020), 106586. https://doi.org/10.1016/j.infsof.2021.106586

Muccini, H., Lago, P., Vaidyanathan, K., Osborne, F., & Poort, E. (2018). *The History of Software Architecture - In the Eye of the Practitioner*. *March 2019*. http://arxiv.org/abs/1806.04055

Oliveira, M. R., Vieira, F. J. R., Misra, S., & Soares, M. S. (2019). A Survey on the Skills, Activities and Role of the Software Architect in Brazil. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11623 LNCS*. Springer International Publishing. https://doi.org/10.1007/978-3-030-24308-1_4

Premraj, R., Nauta, G., Tang, A., & Van Vliet, H. (2011). *The boomeranged software architect*. https://doi.org/10.1109/WICSA.2011.19

Razavian, M., & Lago, P. (2015). Feminine expertise in architecting teams. *IEEE Software*, *33*(4), 64–71. https://doi.org/10.1109/MS.2015.84

Razavian, M., Paech, B., & Tang, A. (2019). Empirical research for software architecture decision making: An analysis. *Journal of Systems and Software*, *149*, 360–381. https://doi.org/10.1016/j.jss.2018.12.003

Sach, R., Petre, M., & Sharp, H. (2010). The use of MBTI in software engineering. *Open Research Online*. https://doi.org/10.5860/choice.51-2973

Sekaran, U., & Bougie, R. (2016). Research Methods For Business : A Skill-Building Approach. In *Sekaran dan Bougie*.

Sherman, S., Hadar, I., Levy, M., & Unkelos-Shpigel, N. (2016). Enhancing software architecture via a knowledge management and collaboration tool. In *In Knowledge, Information and Creativity Support Systems (pp. 537-545). Springer, Cham.* (Vol. 416). https://doi.org/10.1007/978-3-319-27478-2_44

Spinellis, D. (2016). The Changing Role of the Software Architect. *IEEE Software*, *33*(6), 4–6. https://doi.org/10.1109/MS.2016.133

Sudhakar, G. P. (2012). A model of critical success factors for software projects. *Journal of Enterprise Information Management*, *25*(6), 537–558. https://doi.org/10.1108/17410391211272829

Tang, A., Razavian, M., Paech, B., & Hesse, T. M. (2017). Human Aspects in Software Architecture Decision Making: A Literature Review. *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, 107–116. https://doi.org/10.1109/ICSA.2017.15

Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2018). The Impact of Automated Parameter Optimization for Defect Prediction Models. *IEEE Transactions on Software Engineering.* https://doi.org/doi:10.1109/TSE.2018.2794977

Turner, D. W. (2010). *Qualitative interview design: A practical guide for novice investigators. Qualitative Report*. *15(3)*, 754–760. http://www.nova.edu/ssss/QR/QR15-3/qid.pdf

Van Der Ven, J. S., & Bosch, J. (2016). Busting Software Architecture Beliefs: A Survey on Success Factors in Architecture Decision Making. *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, 42–49. https://doi.org/10.1109/SEAA.2016.35

van Vliet, H., & Tang, A. (2016). Decision making in software architecture. *Journal of Systems and Software*, *117*, 638–644. https://doi.org/10.1016/j.jss.2016.01.017

Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), xiii–xxiii. https://doi.org/10.1.1.104.6570

Weinreich, R., & Groher, I. (2016). The Architect's Role in Practice: From Decision Maker to Knowledge Manager? *IEEE Software*, *33*(6), 63–69. https://doi.org/10.1109/MS.2016.143

Westhuizen, D. Van Der, & Fitzgerald, E. P. (2005). Defining and measuring project success. *European Conference on IS Management, Leadership and Governance*, 1–17. http://eprints.usq.edu.au/346/1/DependentVariableArticleV8.pdf

Wired. (2021). *The $150 Million Machine Keeping Moore's Law Alive*. https://www.wired.com/story/asml-extreme-ultraviolet-lithography-chips-moores-law/

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/2601248.2601268

Woods, E., & Bashroush, R. (2017). A model for prioritization of software architecture effort. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10475 LNCS*, 183–190. https://doi.org/10.1007/978-3-319-65831-5_13

Appendix

Appendix A: behavioral competencies (Kornferry, n.d.)

Ensures accountability	Holding self and others accountable to meet commitments
Develops talent	Developing people to meet both their career goals and organizations goals
Communicates effectively	Developing and delivering multi-mode communications that convey a clear understanding of the unique needs of different audiences
Manages complexity	Making sense of complex, high quantity, and sometimes contradictory information to effectively solve problems
Instills trust	Gaining the confidence and trust of others through honesty, integrity, and authenticity.
Cultivates innovation	Creating new and better ways for ASML to be successful
Drives engagement	Creating a climate where people are motivated to do their best to help ASML to achieve its objectives
Drives results	Consistently achieving results, even under though circumstances
Values differences	Recognizing the value that different perspectives and cultures bring to an organization
Situational adaptability	Adapting approach and demeanor in real time to match the shifting demands of different situations
Plans and aligns	Planning and prioritizing work to meet commitments aligned with ASML's goals.
Collaborates	Building partnerships and working collaboratively with others to meet shared objectives
Build effective teams	Building strong-identity teams that apply their diverse skills and perspectives to achieve common goals
Optimizes work processes	Knowing the most and efficient processes to get thing done with a focus on continuous improvement
Balances stakeholders	Anticipating and balancing the needs of multiple stakeholders
Business insight	Applying knowledge of business and the marketplace to advance the ASML goals
Strategic mindset	Seeing ahead to future possibilities and translating them into breakthrough strategies
Demonstrates self-awareness	Using a combination of feedback and reflection to gain productive insights into personal strengths and weaknesses
Self-development	Actively seeking new ways to grow and be challenged using both formal and informal development channels
Decision quality	Making good and timely decisions that keep the organization moving forward.

Appendix B: Literature overview

Key articles:

C1. (Razavian & Lago, 2015)
C2. (Kruchten, 2008)
C3. (Mendes et al., 2021)

Primary:

P1. (Kruchten, 1999)
P2. (Fowler, 2003)
P3. (Bredemeyer, 2002)
P4. (Premraj et al., 2011)
P5. (Hoorn et al., 2011)
P6. (Martini et al., 2014)
P7. (Sherman et al., 2016)
P8. (Britto et al., 2016)
P9. (Van Der Ven & Bosch, 2016)
P10. (Woods & Bashroush, 2017)
P11. (Oliveira et al., 2019)
P12. (Bredemeyer & Malan, 2002)
P13. (Bass et al., 2008a)
P14. (Bass et al., 2008b)
P15. (Clements et al., 2007)
P16. (Clerc et al., 2007)
P17. (Eeles, 2006)
P18. (Erder & Pureur, 2017)
P19. (Farenhorst & van Vliet, 2009)
P20. (Bass et al., 2013) V3 Or the older version 2 (Bass et al., 2003)
P21. (Agarwal & Rathod, 2006)*
P22. McBride(2007)
P23. Bondi(2009)*
P24. Correa(2013)*
P25. Booch(2011)*
P26. (Babar et al., 2009)
P27. (Rekhav & Muccini, 2014)
P28. (Klein, 2016)
P29. (Galster et al., 2017)
P30. (Hohpe et al., 2016)
P31. (Erder & Pureur, 2016)
P32. (Weinreich & Groher, 2016)
P33. (Downey & Babar, 2008)
P34. (Ahmed et al., 2015)

Supportive:

S1. (Manteuffel et al., 2018)
S2. (Razavian et al., 2019)
S3. (Farshidi et al., 2020)
S4. (Hofstede et al., 2010)
S5. (Damian et al., 2007)
S6. (Curtis et al., 1988)
S7. (Madison, 2010)
S8. (Liang et al., 2009)
S9. (van der Raadt et al., 2008)
S10. (Drury et al., 2012)
S11. (McAvoy & Butler, 2009)
S12. (Martini et al., 2013)
S13. (Wateridge, 1997)
S14. (poort et al, 2009)
S15. (Chow & Cao, 2008)
S16. (Bang et al., 2013)
S17. (Lenberg et al., 2015)
S18. (Groher & Weinreich, 2015)
S19. (Sonnentag, 1998)
S20. (Manteuffel et al., 2018)
S21. (Dingsøyr & van Vliet, 2009)
S22. (Jansen & Bosch, 2005)
S23. (Galster & Weyns, 2016)
S24. (Tyree & Akerman, 2005)
S25. (van Vliet & Tang, 2016)
S26. (Buchgeher et al., 2016)
S27. (Acuna & Juristo, 2004)
S28. (Takey & Carvalho, 2014)
S29. (Capilla et al., 2016)
S30. (Cockburn & Highsmith, 2001)
S31. (de Rezende et al., 2021)
S32. (James et al., 2017)
S33. (Hofstede et al., 2005)
S34. (Li et al., 2011)
S35. (Rekhav & Muccini, 2014)
S36. (Bi et al., 2021)
S37. (Bass & Berenbach, 2008)
S38. (Berenbach, 2008)
S39. (Dybe et al., 2014)
S40. (van Vliet & Tang, 2016)
S41. (Cunha et al., 2016)
S42. (Tang et al., 2010)
S43. (Razavian et al, 2016)
S44. (Rose et al., 2007)
S45. (Spinellis, 2016)
S46. (Kruchten, 2013)

Appendix C: Full questionnaire

Important behavioral competencies of a successful software architect

Survey for an innovation management master thesis in a collaboration between ASML and TU/e

* Required

Introduction

Thank you for participating in this survey for my master thesis!

This is survey is part of the master thesis project for innovation management study. The goal is to identify the most important behavioral competencies which make a software architect successful. The target group is every software architect and everyone working with the software architect. The study will contain a few demographic questions and the main questions about which competencies are most important for being a successful software architect. It is expected to finish this survey within 10 minutes.

As part of the scientific research conduct I will need your informed consent for voluntary participation, please read this information before answering the question below with 'I do' or 'I do not'.

---------informed consent information----------

Aim and benefit of the study

This study is part of a project conducted by Sietse Wolberink which in turn is part of a broader interdisciplinary study lead by Maryam Razavian, assistant professor at the Information systems group at Eindhoven University of Technology. With the current study we aim to obtain insight into what makes a software architect successful. The findings of this study hopefully contribute to a better understanding of how and which behavioral competencies affect the success of software architects.

Procedure and anonymity

You received an invitation because you currently have at least part-time employment and are not self-employed ("ZZP") at ASML. For this study, we ask you to complete a questionnaire, which takes about 10 minutes. This questionnaire is completely anonymous. This means that we will collect no data from which your identity could become known. You will not be asked to provide your name, email address or any other information that could identify who you are. The data will be accessible to the researcher(s) and will be published in aggregated form with the exception of the final open question which could be cited. Data shared with the company will be fully anonymized.

Participation and risks

Your participation is completely voluntary. You can refuse to participate without giving any reasons and you can stop your participation at any time during the study. The study does not involve any risks, detrimental side effects, or cause discomfort.

Code of ethics

This research adheres to the Code of Ethics of Eindhoven University.

Further information

If you would like more information about this study, the study design, or the results, you can contact Sietse Wolberink(sietse.wolberink@asml.com (mailto:sietse.wolberink@asml.com)) or Maryam Razavian(m.razavian@tue.nl).

I give my informed consent for voluntary participation: *

- ◯ I do
- ◯ I do not

Demographic questions

These questions will ask about some basic information about your role and experience. Time indication: 2-3 minutes

Gender *

- ☐ Woman
- ☐ Man
- ☐ Non-binary
- ☐ Prefer not to say
- ☐ Prefer to self-describe

If chosen 'Prefer to self-describe' in previous question(2) please disclose:
If you answered this question with a different answer, please leave this question empty

Age (years) *

- ◯ <18
- ◯ 19-30
- ◯ 31-40
- ◯ 41-50
- ◯ 51-60
- ◯ >60

What is your current role in ASML? *

- ◯ (Software) Architect
- ◯ Other

If you are a software architect yourself, which type of software architect? *

- ◯ Department architect, Cluster architect
- ◯ FCA, FCT
- ◯ Platform Architect
- ◯ Product architect
- ◯ System engineer/architect(Focus on software)
- ◯ Train architect
- ◯ Other

f you are not a software architect, what is your role within ASML(if multiple apply, pick the one you feel most related to)? *

- ◯ Architect(not software related)
- ◯ Customer support related role
- ◯ Developer or competence engineer
- ◯ Factory related role
- ◯ Product management(PCM, PL, etc.)
- ◯ R&D management(GL, DM, etc.)
- ◯ SAFe roles(PO, CPO, scum master, RTE, etc.)
- ◯ Other

If you are not a software architect yourself, which type of architect do you work with the most? *

- ◯ Department architect, Cluster Architect
- ◯ FCA, FCT
- ◯ Platform architect
- ◯ Product architect
- ◯ System engineer/architect(focus on software)
- ◯ Train architect
- ◯ Other

How long have you been working with software architects(years)? *

The value must be a number

How long have you been working as a software architect(years)? *

The value must be a number

How much experience do you have in ASML(in years)? *

The value must be a number

Survey questions

In this section two types of questions are asked in regards to the competencies of a software architect. One questions asks you to evaluate how important a competency is for being a successful software architect on a scale of 1(completely unimportant) to 7(crucial). Some competencies might not be applicable to the software architect, you can give them a low score. The other question asks you to pick a top 3 out of these competencies.

The final open question is an optional question.

Tip: If you are in doubt on what score to give, pick your first gut feeling.

There are only 20 rating questions and 3 ranking questions, they are expected to take around 5 minutes. The optional open question is the final question.

How important is each of these competencies for a successful software architect? *

	1 Insignificant	2 Very unimportant	3 Unimportant	4 Neither important nor unimportant	5 Important	6 Very important	7Crucial
1. Ensures Accountability *Holding self and others accountable to meet commitments*	○	○	○	○	○	○	○
2. Develops Talent *Developing people to meet both their career goals and the organization's goals*	○	○	○	○	○	○	○
3. Communicates Effectively *Developing and delivering multi-mode communications that convey a clear understanding of the unique needs of different audiences*	○	○	○	○	○	○	○
4. Manages Complexity *Making sense of complex, high quantity, and sometimes contradictory information to effectively solve problems*	○	○	○	○	○	○	○
5. Instills Trust *Gaining the confidence and trust of others through honesty, integrity, and authenticity.*	○	○	○	○	○	○	○

	1 Insignificant	2 Very unimportant	3 Unimportant	4 Neither important nor unimportant	5 Important	6 Very important	7Crucial
6. Cultivates Innovation *Creating new and better ways for ASML to be successful.*	○	○	○	○	○	○	○
7. Drives Engagement *Creating a climate where people are motivated to do their best to help ASML achieve its objectives.*	○	○	○	○	○	○	○
8. Drives Results *Consistently achieving results, even under tough circumstances*	○	○	○	○	○	○	○
9. Values Differences *Recognizing the value that different perspectives and cultures bring to an organization.*	○	○	○	○	○	○	○
10. Situational Adaptability *Adapting approach and demeanor in real time to match the shifting demands of different situations.*	○	○	○	○	○	○	○
11. Plans and Aligns *Planning and prioritizing work to meet commitments aligned with ASML's goals.*	○	○	○	○	○	○	○
12. Collaborates *Building partnerships and working collaboratively with others to meet shared objectives*	○	○	○	○	○	○	○

	1 Insignificant	2 Very unimportant	3 Unimportant	4 Neither important nor unimportant	5 Important	6 Very important	7Crucial
13. Builds Effective Teams *Building strong-identity teams that apply their diverse skills and perspectives to achieve common goals.*	○	○	○	○	○	○	○
14. Optimizes Work Processes *Knowing the most effective and efficient processes to get things done, with a focus on continuous improvement*	○	○	○	○	○	○	○
15. Balances Stakeholders *Anticipating and balancing the needs of multiple stakeholders.*	○	○	○	○	○	○	○
16. Business Insight *Applying knowledge of business and the marketplace to advance the ASML's goals.*	○	○	○	○	○	○	○
17. Strategic Mindset *Seeing ahead to future possibilities and translating them into breakthrough strategies*	○	○	○	○	○	○	○
18. Demonstrates Self-awareness *Using a combination of feedback and reflection to gain productive insight into personal strengths and weaknesses.*	○	○	○	○	○	○	○

	1 Insignificant	2 Very unimportant	3 Unimportant	4 Neither important nor unimportant	5 Important	6 Very important	7Crucial
19. Self-development *Actively seeking new ways to grow and be challenged using both formal and informal development channels.*	○	○	○	○	○	○	○
20. Decision Quality *Making good and timely decisions that keep the organization moving forward*	○	○	○	○	○	○	○

Ranking the 3 most important competencies for being a successful software architect from the previous question: what is the **number 1** most important competency for a successful software architect? *

- ○ 1. Ensures accountability
- ○ 2. Develops talent
- ○ 3. Communicates effectively
- ○ 4. Manages complexity
- ○ 5. Instills trust
- ○ 6. Cultivates innovation
- ○ 7. Drives Engagement
- ○ 8. Drives results
- ○ 9. Values differences
- ○ 10. Situational adaptability
- ○ 11. Plans and aligns
- ○ 12. Collaborates
- ○ 13. Builds effective teams
- ○ 14. Optimizes work processes

- ◯ 15. Balances stakeholders
- ◯ 16. Business Insight
- ◯ 17. Strategic mindset
- ◯ 18. Demonstrates self-awareness
- ◯ 19. Self-development
- ◯ 20. Decision quality

Ranking the 3 most important competencies for being a successful software architect from the previous question: what is the **number 2** competency? *

- ◯ 1. Ensures accountability
- ◯ 2. Develops talent
- ◯ 3. Communicates effectively
- ◯ 4. Manages complexity
- ◯ 5. Instills trust
- ◯ 6. Cultivates innovation
- ◯ 7. Drives Engagement
- ◯ 8. Drives results
- ◯ 9. Values differences
- ◯ 10. Situational adaptability
- ◯ 11. Plans and aligns
- ◯ 12. Collaborates
- ◯ 13. Builds effective teams
- ◯ 14. Optimizes work processes
- ◯ 15. Balances stakeholders
- ◯ 16. Business Insight
- ◯ 17. Strategic mindset

- ◯ 18. Demonstrates self-awareness
- ◯ 19. Self-development
- ◯ 20. Decision quality

Ranking the 3 most important competencies for being a successful software architect from the previous question: what is the **number 3** competency? *

- ◯ 1. Ensures accountability
- ◯ 2. Develops talent
- ◯ 3. Communicates effectively
- ◯ 4. Manages complexity
- ◯ 5. Instills trust
- ◯ 6. Cultivates innovation
- ◯ 7. Drives Engagement
- ◯ 8. Drives results
- ◯ 9. Values differences
- ◯ 10. Situational adaptability
- ◯ 11. Plans and aligns
- ◯ 12. Collaborates
- ◯ 13. Builds effective teams
- ◯ 14. Optimizes work processes
- ◯ 15. Balances stakeholders
- ◯ 16. Business Insight
- ◯ 17. Strategic mindset
- ◯ 18. Demonstrates self-awareness
- ◯ 19. Self-development
- ◯ 20. Decision quality

How would you describe what makes a software architect successful in 2-3 sentences? (Optional)

End of the survey

Thank you for participating in this survey and helping me with my thesis, don't forget to click on submit! If you have any questions feel free to contact me (sietse.wolberink@asml.com (mailto:sietse.wolberink@asml.com)).

In a follow up on this survey, interviews will be held. I(Sietse) need participants for this, if you are interested in participating in this follow up please send a message with "Participation interview" in the title to
sietse.wolberink@asml.com (mailto:sietse.wolberink@asml.com) .

Appendix D: Full overview of data transformation

Data transformation steps:

1. Remove the rows without informed consent (id= 15, 79, 80, 152, 199)
2. Remove columns start time, completion time, email and name since they are either empty, anonymized or unneeded.
3. Other answers of software architects
    a. 47 SW test architect is actually already defined as the FCT so this will become an FCA, FCT. The other rows are emptied since a small mistake occurred in the initial phase when chosen Other with software architect. Answers to questions with If you are not a software architect, are removed. Answer from how long have you been working with software interchanged with answer how long have you been working as a software architect.
    b. 57 SW sub-function architect is actually pre-defined as a stakeholder by design in this scope. Current role changed to other and content of if you are a software architect removed. Due to a mistake in the forms until reaction 160 this actually produced the right questions afterwards for a stakeholder.
    c. 91 is a competence architect however in the original dataset specifically calls itself a software architect. This competence architect will be formed into the FCA, FCT category since it is the closest to the description while still considering itself a software architect. Competence architect within departments is most of the time technically oriented and the role is working the most with a department/cluster architect suggests at least a middle layer of architect. Therefor this choice is acceptable over removing over the dataset. '' if you are not a software architect'….' Questions removed content. Answer from how long have you been working with software interchanged with answer how long have you been working as a software architect.
    d. 134 identified itself as test architect which is the FCA, FCT category, category changed
    e. 147 ID mentioned being a feature architect which is a SW architect for specific applications. Since the person itself specifically mentions being a software architect this will not be changed. A feature architect is most similar to the FCA, FCT
    f. **155 identified** as a sub-function architect which was specifically taken as developer/competence engineer here. Current role is changed into Other, and non-software architect type into developer or competence engineer which was specifically predefined for the role sub-function architect. For a subfunction architect it can be assumed that it works the most with the FCA, FCT which is both expected from description, hierarchy and dataset. Transforming this into this category is acceptable. Working as and working with columns swapped.
    g. 177 identified itself as a software architect category other: CIDT architect which is a CPD integrated development toolkit(software). Which is specific for calibration performance and diagnostics. This is most similar to a product architect since CIDT is a specific(internal) product. Alternatively this would be a platform architect however, this suits less due to the nature of CIDT.

h. 198 identifies as a feature architect. Similar to 147 this will be transformed into FCA, FCT
i. 200 identifies as sub-fucntion architect. Following the same reason and transformation as 155 it will be transformed.
j. **********207 identifies as a product test architect. Even though product is specified. As a test architect this will be transformed into the FCA, FCT category where the FCT entails test architects.
k. 221 identified as sub-function architect and follows the same reasoning and transformations as 155

4. Non- software architect transformations:
    a. 45 identified as intergrator. This is a unique true other role. However, it answered roles of these architect unknown, everyone gets this title with the who do you work the most with. Therefor, this is unworkable as an answer or category and thus removed from the dataset.
    b. 59 answered he was a software architect but is currently a grouplead. Following the nature of this questionnaire this is changed to the R&D management category
    c. 60 answered lead engineer, this is a developer or competence engineer role. However this answer was given a few times. Future research should consider it to be a separate category like sub-function categories.
    d. 66 answered lead engineer, same process applies as with 60.
    e. 101 answered software designer, part of the developer category
    f. 108 answered lead designer, same applies as for the lead engineer. Transformed in Develo9per or competence engineer.
    g. *****110 mentioned being test infra lead, to developer or competence engineer. Same as 108
    h. 135 is a functional intergrator which even though is a different role, would be considered to be an engineer.
    i. 165 lead engineer of embedded software same argument as 60. Now developer or competence engineer
    j. 175 softwaer engineer, engineer is a developer.
    k. **********185 industrial engineer. Industrial engineer is a senior mediator role responsibe for certain products and should be considered either as product manager or R&D manager. In this case R&D due to the nature of software development as core focus would be more appropriate.
    l. 227 software design engineer is developer or competence engineer same argument as 60.

5. The working the most category has the following transcribtions:
    a. 30 RTE is a SAFe role Assuming this one will work the most with a train architect since this is the ‘SAFe’ architect
    b. 44 identified as working the most with a comptentene architect as seen before this role is most comparable or part of the FCA, FCT category (seen with 3c.)
    c. 73 mentioned managing archietcts and other developers, suggesting a more senior role managing architects and other developers which could be considered one of R&D roles,

which it also is. This suggests this person does work the most with architets and transforming this one into an FCA, FCT following the input description.

   d. 77 mentioned multiple roles instead of the most:' Train Architect, FCA, FA, Product Architects'. It starts with train architect therefor this is assumed to be the most important.
   e. 138, 192 and 227 mentioned sub-function architect here. Since this one is not taken into account it will be changed to the first next layer, FCA, FCT in line with the scope of this research.

6. Duplicate 'What is your current role in ASML?' and transform into binary where 0 = non-software architect or stakeholder and 1=software architect
7. Duplicate if you are a software architect twice
   a. 1 based on the role where: Department architect =1, FCA, FCT=2; Platform architect =3; product architect=4; system architect = 5; train architect = 6
   b. Another based on SA_hierarchy where 1= highest hierarchy of Department, platform, product and system architect. ; 2 = FCA, FCT and train architects
8. Duplicate non software architect roles twice where
   a. Based upon role with Customer support related role =1, Developer or competence engineer=2; Product management =3; R&D management =4; SAFE roles = 5 note here that the non-software architects and factory related roles do not occur and thus are not used.
   b. Hierarchy where Developer or competence engineer and customer support related role is considered the lowest layer =5 ;; the SAFe roles and customer support related role are mid level = 4;; product management and R&D are considered high level hierarchal roles= 3
      i. Note that future research should create more distinguishments in developer or competence engineer due to the large amount of answers and potentially including also mid layer roles
9. Working the most with line duplicated and coded following the same deviation as 7a: Department architect =1, FCA, FCT=2; Platform architect =3; product architect=4; system architect = 5; train architect = 6
   a. Another was created following the hierarchy: Another based on SA_hierarchy where 1= highest hierarchy of Department, platform, product and system architect. ; 2 = FCA, FCT and train architects

All [1-20] competency scale parts have the language answer attached. Removed it excess text.

Appendix E: Architect and Stakeholder interview coding output

	Software architects	
	Excerpt summary	Behavioral competencies
03	Making the right decision not for now but for the longer term(BC20)	20
	Stakeholders push for faster and have their own priorities however tha rchitects should make the right thing independent of time (BC4 & BC20)	4 20
	Seeing the opportunity to create disable options in a user interface and make it configurable creates more work but it is a long term option. (BC4 & BC17 & BC20)	4 17 20
	If you already have the idea(BC4) then it is easier to convince people(BC3) of the smaller pieces which they want implemented(BC20)	4 & 3 20
	Making a strong business case helps convince budget and money changes for the long term view(BC16)	16
	It helps to have a good relation with people (BC5/BC12)	5 & 12
	You have to build trust, otherwise they will not believe you and it will make convincing others a lot more difficult(BC5)	5
	Explaining your roadmap to different stakeholders and other architects to create awareness(BC3) this is a very good way to align with stakeholders(BC12)	12
	Administrating possible changes based on the questions and struggles of people(BC4) which are than transelated in the update(BC17 but not breakthrough)	4 17
	Reviewing documents and seeing how things are not in line with the roadmap(BC4) allows the architect to intervene and steer towards another solution(BC3 & BC12) resulting in a different and hopefully better decision(BC20)	4 3 & 12 20
	Sometimes convincing is not easy, disagreements can come up in the dm process(BC20)	20
	Always has been a teamplayer which want to involve others, hear opinions(BC12)	12
	In the documentation creation process meetings, design meetings and brainstorms are used(BC3) in order to make the right decisions(BC20)	3 20
	Going step by step via different type of communication methods(BC3) In order to shape and shape the design together by agreeing with intermediary pauses(BC20)	3 20
	Trusting others is also important since it gives resonsabilities(BC5)	5
	Really understand how decisions fit in the roadmap but also see how they affect user friendliness and customers and balancing the needs(BC15)	15
	Using meetings to get the discussions going with stakeholders and pre-align as muc as possible.(BC3 BC4 BC20)	3 & 4 & 20
	Doing small iterations and have more interactions with the team(BC12)	12
	In order to avoid misunderstandings have pre-discussions with drawings, reviews or documents to really check if everything was done like agreed(BC3)	3

	Architects need to anticipate things which are expected to be known, not to be known. Use simple words and translate it into the big overview(BC4 BC3)	3 4
	Explaining your reasoning well, it will really get people on board and they will make decisions(BC20) which are more in line of your own way of thinking. Use your rationale. (BC4 BC12)	20 4 12
	A key part of being an architect is having the complex system and chopping it up in smaller pieces and managing all the dependencies(BC4)	4
	A decision has multiple alternatives and the decision itself is related to the surroundings.(BC20) Having that long-term scope but also having other departments requires a lot of working together(BC12) and seeing what other solutions would be.	20 12
05	Making new people feel welcome so they are approachable (BC5) + network building(BC12)	5 12
	Not being able to review everything yourself(BC4)	4
	Have the trust so people come to you which is a result of building the network and collaborating(BC5->BC12)	5 12
	architect required for guidance and removing doubt(BC4 &BC3)	3 & 4
	Architects depend on people for information and their specific needs(Communicates effectively) but also pre-requisite to gather information(Pre-requisite Manages complexity)	3 & 4
	Example how architects dont make the decision(20). They work together towards the central goal(BC12), they have to develop the knowledge of the different needs by communication well(Implied BC3) Calling Decision making more the moderation of the decision process(Which implies BC4 but also BC20)	20 12 3 4
	More options mean more criteria that have to be reviewed (BC4)	4
	Getting to know the perspectives of the stakeholders (BC3)	3
	Making sense of all the different situations(BC4) and than using the right method to steer towards a direction(BC3)	4 3
	Also being able to take company perspective(BC16) which implied strategy making based on this(BC17)	16 17
	Showing how company, engieneer, marketing and non-technical peoples views all have to be taken into account(BC4)	4
	Much considerations of either redesigning or fixing existing elements(BC4) after which you need to convince everyone by using network in order to get all drawbacks out(BC12)	4 12
	Requiring a lot of talking and understanding, explaining stakeholder(BC3)	3
	Using the right arguments (BC3) but it implies BC4	3 & 4
	Changing the methods of communicating the ideas depending on who you are talking to(BC3)	3
	Build network via coffee chats etc with a lot of people(BC12) you get to know when to contact who (BC4)	12 4
	Taking into account more opinions creates more requirements (BC4)	4
	A lot of input & questions(BC4) which requires a lot of communication(BC3) and working together(BC12)	4 3 & 12
	Make sense of all the different things going on and different perspectives(BC4)	4

	A lot of tradeoffs in the process of maintaining old machines (BC4) which are required to make the right decision(BC20)	4 20
	Example of wrong assumptions which affect the future decisions but they cannot be changed anymore since the hardware is now fixed(BC20) but in order to realize this and how to fix it requires a lot of information and making sense of it(BC4)	20 4
08	Architecting realizing something doesn't make sense(BC4) after which the others are steered by sending them to check certain things(BC12) which is done in a format suiting the specific stakeholders(BC3)	4 12 3
	Choosing the right moment for a constructive discussion(BC3)	3
	The improved relation creating the consideration of the architect being part of the team rather than a checkbox(BC5)	5
	If decisions are made, (external) customers in the field may think what happened here. Gathering information from them requires using the connections and working together(BC12) but in order to convince them of the tradeoffs you need to communicate well with the right intentions(BC3) after which they can make the decision(BC20)	12 3 20
	Decisions have impact not only on the software bu also on its structure. That impact is not always desirable since it would require more effort and complications(BC20)	20
	There was a lack of alignment, mixed information and interpretations(BC4) after realizing this a couple of meetings wer used and a hackaton to align everyone(BC3)	4 3
	Wrong information being conveyed(BC3-) making the design itself coming out wrong(BC20)	3 20
	Using different methods to communicate a message in different forms depending on the audinece(BC3)	3
	Next to formal meetings just talking to people and doing whiteboard sessions to create an understanding(BC3)	3
	Collaboratino(BC12) being a tool to work together on talking through the different elements to make the right design(BC20)	12 20
	Working together with project management(BC12) is better for the relations with business as it aligns concepts and ideas and makes you realise what is important for everyone(BC3 & BC16)	12 3 16
	Finding a link (BC4) led to proposing changes with the stakeholders and working with them in order to do thisbut it was trimmed down as result of working together with the stakeholders. (BC12) This also implied having the right communicating methods for the convincing(BC3)	4 12 3
09	A different department is required for a project. After discussing with them, a they were pointed to a direction in line with the roadmap(BC4 was needed in prep, BC12) It is difficult to convince others but people are often open to the benefits of others too(BC15)	4 12 15
	Making a good story in order to transelate the technical environment to facilitate discussion based upon the architecture and explain why alternatives are preferred(BC3)	3
	Preparet he story in based upon the common interest and take everything outside of their own scope(BC3)	3

	Architects should take the technical perspective and not concern too much about projects, deadlines and pressure in order to create the best solution(BC4 as pre-requisite to making the right decision BC20)	4 20
	Giving in and being pragmatic by for example phasing a change in order to help other reach their commitments(BC15)	15
	Someone requested an extension on a module which the architect wanted to phase out. The architect realized the implications on his long term vision with this new information(BC4) which led to a discussion about the alternatives working together to still meet the required functionalities(BC12)	4 12
	Gathering information by asking people from project in which the architect has previously been working(BC12 & BC3)	12 3
	Specific deduction of the complexity where by using rationale the architect realized there were actually different intentions of the project, being able to simplify the old requirements into a single clear new requirement(BC4) This was then directly taken to his project counterpart who agreed(BC12) and was able to convince others by showing how this would simplify their problems(BC3) This could all be viewed in context of preparing the decision to change this feature(BC20)	4 12 3 20
	Making poor decisions will give consequences in future projects(BC20-). You can always come back to decisions if you know they are wrong	20
	Out of all the options and information the architects saw an opportunity to simplify(BC4) leading to the decision to push for this change(BC20)	20 4
	After futher discussion and working together towards this simplification(BC12) the decision was made together with a lot of people so the decision itself was good(BC20)	12 20
	Previous example where the other department was open to arguments of the other side in order to improve as collective(BC12)	12
	Being approachable and honest is needed and this will bring you far with stakeholders(BC5)	5
	Having discussions is really important, even if this gives long days having the chats is valuable, these have priority(BC3)	3
	Communicates effectively prevents miscommunications, bugs and errors(BC3)	3
	To be more successful an architect should mainly focus on conveying a clear understanding of others needs. This should be done for different audiences.(BC3)	3
	With all different kind of stakeholders and solutions communication is important in convincing others(BC12) In addition aligning all these types of information forces concessions(BC4)	12 4
	Trying to assemble information but getting too much can harm lnog term because of not coping with the complexity(BC4) -> KM	4
10	Working together with an end customer to install and analyze on location(BC12)	12
	Working together with the end customer working towards improvement(BC12) overcoming language barriers with pieces of papers, translation machines and handsigns(BC3)	12 3

	Depending what you see as decision quality(no bugs i.e.) having the proper balance between different elements for example doing things in a short timeframe(BC20)	20
	Making the decision together with an overview of possible choices (BC20)	20
	Identifying a certain opportunity and identify gaps from the start of a single product(BC4)	4
	Coming to an shared agreement with stakeholders which includes taking into account other aspects(BC12 & BC15)	12 15
	Doing a late session as a tool in order to push for decisions and convergence(BC3)	3
	Pre-align with every stakeholder in order to be able to have all the knowledge in the meeting itself. (BC3 where smart communication is needed, but also BC4 in managing all these inputs, and then BC12 since this is part of the collaborative process)	3 4 12
	Zooming out in order to identify the context & stakeholders incl their roles and the bigger picture(BC4)	4
	Pre-alignment had been useful(BC12), but now under time pressure action was taken(BC20)	12 20
	Seeing the skill to summarize(BC3) in order to set direction (BC20)	20
	Gathering all the inputs is required to make a balanced proposal(BC20) the architect has to understand the people and context but also the knowledge of the impact on the software(BC4). This is all gathered by working together with people and getting their view(BC12)	20 4 12
	Transelate all the different terms into simple language depending on the audience(BC3)	3
	Underestimating new technical changes and the availability of replacing a tool might cause the misunderstanding of consequences of having a completely new design. This is caused by bad communication(BC3)	3
	As an architect you should know where you are and where you want to go, think of intermediary steps and then making it explicit(BC17 & BC4)	17 4
	Using the architectural knowledge in order to work together with people via workshops(BC3) and making a joint breakdown and structure together(BC12 & BC20)	3 12 20
	Use workshops to align and understand stakeholders(BC3)	3

	Stakeholders	
	Excerpt summary	**Behavioral competencies numbers**
01	Design sessions are needed to facilitate the discussions between stakeholders. (BC12) In addition architects should split up larger improvements into smaller chunks(BC4)	12 4
	After identifying how a data transformation could be done easier(BC4) others had to be convinced for the investment in it by realizing their benefits(BC3)	4 3

	A testing protocol including reporting was made daily and weekly. The architect took in all the information and saw the opportunity(BC4) avoiding the double cost of doing it multiple times.	4
	Continuation last example. The architects cant just instantly remove this, a lot of people are familiar with the weekly version. Architects should take away any unrest(BC5) by having a lot of small meetings next to the large meetings. Many have to be informed(BC3) but after working together to get rid of the weekly report, people will accept it(BC12)	5 3 12
	Using different methods like powerpoint or test runs to convince people changing the message depending on their specific needs(BC3). In additon, having the decision team execute the teams to build trust(BC5)	3 5
	The stakeholder found the felt supported and comfortable by the architect, he knew the technical side was under control(BC5) suggesting the trust in the technical decision ability of the architect(BC20)	5 20
	An architect made an instructural video(BC3) but it made it look like making test cases was easy. Different people did not understand the nuance with this where not all test case making is the same, some even being manual. This created discontent with engineers but also created misunderstandings between how different stakeholders viewed this(BC15-) and bad decision timing(BC20-)	15 20
	Not pre-aligning cause the problem of misunderstandings in the whole community, the architect should have included others(BC12-)	12
	If one architect gets bypassed this might make the project miss an approval closer to the date of signoff(BC12-)	12
	If multiple projects are present in the same team, you want to have more detailed knowledge about the overview, whats happening in the teams and the feasibility overview(BC4)	4
	Making a timepath for projects, do feasibility, realization and then functionality implementation(BC11) in order to make the right decision after alignment and prototyping(BC20)	11 20
	Gathering the insights into the different shortcuts, acceptable boundaries, repair future, consequences and the complexity of the whole system(BC4)	4
	By working with the different engineers and seeing what they have to redo over and over again(BC12) the architect can look at what is going on and what to anticipate on(BC4 BC20)	12 4 20
04	Not being able to manage the complexity (BC4-) Consequence being replaced. Also implies the drop of trust(BC5-)	4 5
	Creating trust based on seniority, knowledge and skills(BC5)	5
	Incorporating views of others and set new direction with it(BC3 & BC4 in preparation but example of BC17 This requires the network(BC12)	3 & 4 12
	Using the right communication methods to achieve the path forward(BC3) by invertorazing and making sense of a lot of different perspectives(BC4)	3 4
	Long term planning and making the right decisions (BC11 & BC20)	11 & 12
	Using whiteboard sessions, face-toface meetings(BC3) to collectively build a proposal(BC12)	3
	Large designs can get you lost in detail quite quickly, architects need to manage this(BC4)	3 12

	Reviewing all proposals and validating(BC4) needed to help teams get unstuck by working together(BC12)	4 12
	Architects missing coding details leading to multiple reworks after the first decision(BC20-)	20
	Working on too much detail, not knowing about some changes(BC4-)	4
	Relying on the autonomously managing their work and execution of the architectS(Implicitly BC5)	5
	Digging into the problem even down to code level, but finding the solution(BC4)	4
	Creating an image of not sitting on sidelines(BC5)	5
	Understand context and spend time gathering knowledge and come up with proposals(BC17)	17
	Being able to explain to a wide range of audiences what is happening and changing tune when needed(BC3)	3
	Architect should be an effective communicator and be aware of the different expectations of the different audiences(BC3)	3
	If you cannot convey your message, it does not matter which stakeholder you are talking to(BC3-)	3
	Communicating and conveying message is critical for clearly explaining thing to the people doing the work(BC3 but requirement for BC12)	3 12
07	Working with multiple clusters, with time pressure in a complicated project the architect has to get a design which actually works now and in the future(BC4)	4
	Big project can go all over the place with different voices and opinions(BC4) being the driving force for the direction of the cluster under though circumstances (BC8)	4 8
	Architects have to show to make that they make the right calls(BC20) but in order to do this they need the trust of teams and people surrounding them in order to get the mandate to make bigger decisions. This support is build through experience(BC5)	20 5
	Architect starts involving different stakeholders and spar and communicate with them(BC 3 & BC4) but this also generated trust by seeing him do this and seeing the impact on the projects of this architects(BC5)	3 4 5
	There is a bigger scope and people get lost in it. So many stakeholders, ideas, procedures and opinions have to be managed(BC4) but if you transform this in to the right arguments it shows how you know what you are doing(BC3)	4 3
	Asking about the teams problems and thining of a solution together(BC12) which in combination with the delivery demands of the project requires the management of stakeholders(BC15)	12 15
	Having to balance ideas in long term and short term goals requires a lot of convincing otherwise of the benefits(BC3)	3
	The software archtiect should be involved when key decisions are being made(BC20)	20
	Architects should adhere to their policies but that also means sometimes products don’t happen, there still needs to be budget and they have to work with instead of work around others(BC12)	12

	The consequencies of changing elemetns could have a lot of impact on multiple other groups, this had to be realized(BC4)	4
	Communicating thoughts and preferences clearly will help stakeholders understand the tradeoff. They might not like it, but they will understand(BC3)	3
	Giving all the different documents to everyone with all the options(BC4 +KM) Communicating this via regular meetings in which stakeholders can give all their problems(BC3/BC12)	4 3 12
	Clear communication and tradeofs with alternatives fitted to the audience(BC3)	3
	With a lot of stakeholders, a good decision could please everyone(BC20) but it also needs to be taken into account how stable this design would be(BC4)	20 4
	An architect working with the team to help them out with his technical knowledge to make the product work(BC12)	12
	Architct basing not doing something on extensibility(BC20)	20
11	Know how to communicate to a larger group(BC3)	3
	Combining a good story wit reality to connect people together(BC3) and recognize that the existing way of working was not maintainable(BC4)	3 4
	Finding commonalities in the bigger picture(BC4)	4
	Finding the commonality so it benefits all releases or products because the work is done only once (BC4) but also find the benefit for others(BC3)	4 3
	Technical; Creating better ways for ASML to be successful(BC10);; really making a plan together to get to the goals(BC12)	10 12
	Having informal chats with all types of stakeholder top down and convince them step by step using a variety of methods both formal and informal(BC3 BC12)	3 12
	Taking stakeholders along in order to make them realize that there is a problem and think of the soluition together Joint decision making about solutions(BC20)	20
	Bad example: architect being direcotrive and mandatory tone. (BC3-/BC12-)	3 12
	Example of being willing to listen to others. Making the decision of going there together(BC12/BC20)	12 20
	Architect does not know everything(BC4)	4
	Organizain gsessions to gather input and explain what problems they are actually facing(BC3). After doing this individually, a meeting with all stakeholders was held and in a few hour the direction and decisions were clarified(BC12/BC20)	3 12 & 20
	Sometimes there are choices which have to keep being aligned(BC12), creating an very long discussion. The architect should then make the choice and convince others of this choice(BC20).	12 20
	Listen, get input, explain why decision was made(BC3) and make people stick to a decision(BC20)	3 20
	Combining sticking to the decision(BC20) but also listening and explaining it in a simple manner to others(BC3)	3
	Good decisions(BC20) are a result from setting the right direction and showing all alternatives(BC4) then communicating the decision table in a	20 4

	simple and easy to understand manner depending on the person and the persons habbits(BC3)	3
	Decision quality includes timing, quality, technical debt, team satisfaction, documentation. You name it. All of the the the non product timing quality (BC20)	20
	Having knowledge of where the product is in a life time cycle(BC4) but also realizing that due to it being end of life it should be phased out(BC20) then convincing the others of the same perspective(BC3) and work together to to best product(BC12)	4 20 3 12
	Having knowledge of where the product is in a life time cycle(BC4) but also realizing that due to it being end of life it should be phased out(BC20) then convincing the others of the same perspective(BC3) and work together to to best product(BC12)	4 20 3 12
	Convincing others in different ways depending on their need and goals(BC3)	3
	A worried stakeholder was taken into a meeting(BC12) using a combination of listening and explaining (BC3) which made the stakeholder trust the architect and at least felt his worries were heared(BC5)	12 3 5
	Knowing when to make what decision by acknowledging there are risks (BC20)	20
	Architect not being able to split something up(BC4-)	4
	Example of not working together with consequence of project getting cancelled and chaos in team(Bc12-)	12
	By being very open and creating a clear structure and planning(BC12) while aligning the different puzzle pieces and managing its complexity(BC4) the architect managed to get a project back on the rail. When 10 people start without a direction things will go everywhere. The architect supports them in the technical element with this(BC12)	12 4 12
	By using meetings and informal catchups(BC3) the architect manages to get information, know about plans and setback(BC4, BC12) this created the position where issues are shared and the architect is approachable(BC5) one engineer even saw the architect as a father figure.	3 4 12 5
	Example where zooming in and out when needed helps further the project(BC3 BC4) by showing other the bigger picture which requires the simplification of all larger parts	3 4
	Required to manage complexity(Bc4) to do this and than use different communication methods to find the right information(BC3).	4 3
	an architect can be open and transparent also in what he actually contributes which builds trust with the people they work with(BC5) But also helpful by working towards a common goal together in various ways(BC12)	5 12
	It is required to manage a lot of information without needing the detail(BC4)	4

	Other findings	
	Excerpt summary	Category
01	Using the reduction of maintenance effort as a argument to convince others (Success variable Maintanability)	Success metric
03	Stakeholders view short term perspective where architects view the long term success	SM
	Some stakeholders just want to finish projects	SM
	Success for project management is quick where architects want the best solutions.	SM
04	Succes factor: met major milestones	Success metric
	Success factor reduce number of dependencies	Success metric
	Few people start as architect, they grow into the role	KM
	Success factor: Within scope ,Within budget	Success metric
	Realizing what can be viewed as success needs quite a lot of awareness and knowledge Success factors: Minimum requirements and maintainability	Success metric
05	Outcomes for software(success outputs) Testability, Maintainability, Usefullness, Cost, Complexity, time	SM
	Everything is about transferring information	KM
07	Giving all the different documents to everyone with all the options(BC4 +KM)	KM
	Success factors: architect thought of future extendabilities	SM
09	Archtiect happy if others take into account & get convinced by the future maintainability(success factor)	SM
	Avoid getting overloaded with knowledge input KM	KM
10	Successful when expectations are met, award received, customers happy	SM
	Combining knowledge of the difficulties in software changes with increases in throughput = KM	KM
	Success outcomes: Happy customer, short timeframe, happy crew	SM
11	It is required to manage a lot of information without needing the detail	KM