An exploratory case study on reusing architecture decisions in software-intensive system projects

Manteuffel, Christian; Avgeriou, Paris; Hamberg, Roelof

Published in:
Journal of Systems and Software

DOI:
10.1016/j.jss.2018.05.064

Publication date:
2018

The Journal of Systems & Software 144 (2018) 60–83

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

An exploratory case study on reusing architecture decisions in software-intensive system projects

Christian Manteuffel[*,a], Paris Avgeriou[a], Roelof Hamberg[b]

[a] University of Groningen, Nijenborgh 9 9747 AG Groningen, The Netherlands
[b] Océ-Technologies B.V. P.O. Box 101, 5900 MA Venlo Sint Urbanusweg 43, 5914 CA Venlo The Netherlands

ABSTRACT

Reusing architecture decisions from previous projects promises to support architects when taking decisions. However, little is known about the state of art of decision-reuse and the benefits and challenges associated with reusing decisions. Therefore, we study how software architects reuse architecture decisions, the stakeholders and their concerns related to decision-reuse, and how architects perceive the ideal future state of decision-reuse. We conducted a qualitative explorative case study in the software-intensive systems industry. The study has shown that architects frequently reuse decisions but are confined to decisions they already know or have heard about. The results also suggest that architects reuse decisions in an ad-hoc manner. Moreover this study presents a conceptual model of decision-reuse and lists stakeholder concerns with regards to decision-reuse. The results of this study indicate that improving the documentation and discoverability of decisions holds a large potential to increase reuse of decisions and that decision documentation is not only important for system understanding or in the context of architecture reviews but also to support architects in upcoming projects.

1. Introduction

Taking architecture decisions is widely regarded as a difficult activity that often involves considerable effort spanning multiple days or weeks (Manteuffel et al., 2014; Tofan et al., 2013). Taking decisions can become especially aggravated in large-scale software-intensive system projects, characterized by a multi-disciplinary and distributed development environment, involving numerous stakeholders with conflicting interests, and a competitive business environment. In such projects there are multiple factors that increase the complexity and intricacy of taking decisions, for example, aiming at a short time-to-market in combination with the complexity of the to-be-developed systems; requiring the architect to design for different modes of operation; achieving compliance with controlled elements from other disciplines; and handling robustness of variations in operating conditions.

Reusing architecture decisions from previous projects promises to support the architect when taking decisions by reducing the effort associated with architecture analysis, synthesis and evaluation (Hofmeister et al., 2007) and by increasing the quality of decisions (Zimmermann et al., 2007). A systematic reuse of decisions allows the architect to identify key architectural issues, common domain-related stakeholders, recurring concerns, and decision-forces as well as proven or failed architecture solutions, patterns and technologies. More importantly a decision can also be reused in the context of a network of related reusable decisions: through decision relationships it is possible to identify corresponding architectural issues including conflicting or compatible architecture solutions. These reusable decision networks embody architectural knowledge relevant for a particular domain (van Heesch et al., 2012a) and provide more added value than the sum of individual decisions.

The idea of reusing architecture decisions and in general architecture knowledge (AK) is not new and has been explored in many ways (Tang et al., 2010). Traditionally, the focus of researchers and practitioners was on reusing system-generic AK, like patterns, tactics or technologies. System-generic AK particularly qualifies for reuse due to its system agnostic documentation, which makes it applicable to a wide-variety of systems. In contrast, system-specific AK entities, like requirements or decisions, are only reusable in the context of a particular system, a family of systems, or a system domain. However, system-generic AK is often not available or too high-level for niche or specialized systems. In these cases, system-specific AK assists architects "on the basis of reusable peer knowledge applied successfully in similar situations (Zimmermann, 2011)". Hence, there has been an increased interest in recent years in exploiting system-specific AK as reusable assets, in particular, architecture decisions (Zimmermann et al., 2007; Zimmermann, 2011; Soliman and Riebisch, 2014; Anvaari and

* Corresponding author.
E-mail addresses: c.manteuffel@rug.nl (C. Manteuffel), paris@cs.rug.nl (P. Avgeriou), roelof.hamberg@oce.com (R. Hamberg).

https://doi.org/10.1016/j.jss.2018.05.064
Received 20 October 2016; Received in revised form 2 May 2018; Accepted 28 May 2018
Available online 29 May 2018
0164-1212/

Zimmermann, 2014).

However, in industrial practice, architecture decisions are currently not being systematically reused and there are no approaches used in industry that facilitate systematic reuse; to be more specific, decisions are often not even explicitly captured (Tyree and Akerman, 2005; Capilla et al., 2015) which can be considered as a prerequisite for systematic reuse. Moreover, researchers only paid little attention to explore the benefits and challenges of systematically reusing decisions in order to provide evidence to practitioners. Therefore, we conducted a qualitative explorative case study in the software-intensive system industry, in which we study how software architects currently reuse architecture decisions and how they perceive the ideal future state of architecture decision-reuse.

This study allows us to identify how and to what extent architects currently reuse architecture decisions in their daily practice, stakeholders relevant to decision-reuse and their concerns, as well as incentives that motivate or inhibitors that discourage decision-reuse in industry. The results of this study contribute to a better understanding of the status quo of decision-reuse in the software-intensive system industry and present an important step towards a systematic decision-reuse approach which meets the needs of software architects.

The study has been conducted according to the guidelines for case study research in software engineering proposed by Runeson et al. (2012) and has been organized according to their template for conducting and reporting case studies. This study first gives a brief overview of related work on decision-reuse. It will then go on to outline the case study design including research questions, object of study, case selection, data collection and analysis. Section 4 presents the findings of the case study, discussed per research question. The fifth section is concerned with limitations of the study, followed by a concluding summary and recommendations for further research in Section 6.

2. Background and related work

The idea of reusable architectural knowledge that supports the architect in solving recurring design problems is not new. A recent study by Tang et al. (2010) found that several AK management tools facilitate reuse of system-generic AK, such as architecture patterns or architectural tactics, which present well-proven reusable solutions for recurring design problems. However, system-generic AK is often tailored to a particular domain like automotive systems or service-oriented architecture. This limitation reduces their applicability especially for companies that operate in a niche where architectural problems frequently require custom solutions involving inter-disciplinary problem solving (e.g. HW/SW Co-design). In these contexts, reusing system-specific architecture decisions from previous projects within the same company could provide several benefits. In a survey about the decision-making process of architects, van Heesch and Avgeriou (2011) suggests that “decisions from comparable projects can serve as a starting point to develop [an architecture] vision and can furthermore help to make sure that no important considerations were forgotten”, such as stakeholders, architectural problems, architecture concerns, decision forces or candidate solutions. Likewise,Tofan et al. (2013) also found in a survey that architects are eager to compare their decisions to previous decisions that solve a similar problem and that the absence of those decisions are considered to contribute to the perceived difficulty of decisions. Zimmermann (2011) further states that the availability of reusable decisions can help to increase the understanding of decision-making needs and available solutions, support architecture reviews and prioritization of work items, as well as, improve communication between architects.

In the past decade, several approaches for documenting architecture decisions in software projects have been proposed (Tang et al., 2010; Capilla et al., 2015). The majority of these approaches are codification-approaches, in which decisions are explicitly documented like PADME (Babar et al., 2005), EAGLE (Farenhorst et al. (2007a)), ADWiki (Schuster et al., 2007), ADvISE (Lytra et al., 2013), or Decision Architect (Manteuffel et al., 2016). Only a few approaches employ aspects of a knowledge personalization strategy, capturing “who knows what” about a decision rather than documenting the decision itself (Dingsøyr and van Vliet, 2009). Most approaches identify decision-reuse as an important stakeholder concern. For example, the decision management tools proposed by Nowak and Pautasso (2013), Farenhorst and van Vliet (2008) and Manteuffel et al. (2016) all mention support for reusing AK, although their main focus lies on capturing decisions while no concrete guidance is given on how decisions can actually be reused in the architecting process and for which purposes. According to Nowak and Pautasso (2010), the creation of a company-wide repository of reusable decisions and its adaptation into industry practices remains an unsolved problem. Approaches for capturing design reasoning targeted more towards system engineering are Customer-Application-Functional-Conceptual-Realization (CAFCR) (Muller, 2004) or Questions-Options-Criteria (QOC) (MacLean et al., 1991). In CAFCR the customer view and the application view capture the why of the product. The functional view describes the what of the product. The conceptual and realization describe the how of the product. In QOC questions represent the problems, options link to candidate solutions, and criteria are used to determine how well options solve the given problem.

In recent years new approaches have been developed that focus on enabling decision-reuse. For example, Zimmermann (2011) explored the idea of decision-reuse via decision guidance models, which identify typical issues and solutions for a well-defined domain. However, a limitation of their approach is that the effort to create the guidance models is only justified for domains in which issues frequently reoccur, for example, green-field service-oriented architecture projects (Zimmermann, 2011). Another limitation is the extra effort required for creating system-generic guidance models out of system-specific decision models. Recently, the approach received tool-support in the form of a plugin for Enterprise Architect called ADMentor (Zimmermann et al., 2015). Overall, numerous studies are related to the subject of decision-reuse by codifying frequent domain-specific decisions into decision models, for example, Zimmermann et al. (2007) for service-oriented architectures or Lewis et al. (2016) for cyber-foraging systems.

Compared with previous work, this study does not propose a concrete approach for reusing architecture decisions but rather examines the current situation of decision-reuse and its potential within the software-intensive system industry. Understanding the status-quo of decision-reuse, along with the guidelines and recommendations obtained from the findings of this study, is an important step for developing successful decision-reuse approaches that intend to find a larger adoption within the software-intensive systems industry. Moreover, in contrast to previous work, a special emphasis is put on the reusability of system-specific knowledge.

3. Case study design

The design of our study can be described as an exploratory qualitative embedded single-case study. An exploratory study is ideal in situations when the research goal is to seek new insights or generate ideas. In this study we want to explore the reuse of architecture decisions and not confirm an existing theory about decision-reuse or validate a decision-reuse approach. Moreover, we use qualitative data collection techniques because we study human behavior (Seaman, 1999), i.e. to gain an understanding of the underlying motivation, opinions, and communication of architects reusing decisions, as opposed to quantitative data collection techniques that aim to explain causalities or study the order of magnitude of a phenomenon (Runeson and Höst, 2009). A case study approach was adopted to obtain further in-depth information on decision-reuse. In contrast to experiments, a case study is pertinent when it is difficult to study the phenomenon in isolation: in this case the software process, application

domain, and organizational guidelines influence to what extent decisions can and will be reused (Runeson et al., 2012). Moreover, a case study allows us to acquire an in-depth understanding of the phenomenon as opposed to a survey, which would only allow us to generalize over a population. The case study follows an embedded design, which allows us to observe more details in the case because we study multiple units of analysis. This reduces the likeliness to miss important issues and it allows us to discover differences within the case.

3.1. Research questions

The main goal of this study has been formulated according to the GQM template by Basili (1993), which expresses the object of the study, defines the particular angle of the study, and describes the context of the study.

Analyze the architecting process *for the purpose of* understanding *with respect to* reuse of architecture decisions *from the point of view of* architects *in the context of* software-intensive system projects.

The research goal has been decomposed into five research questions:

RQ1 What is the current state of decision-reuse in the software architecting process of a software-intensive systems company?
RQ2 Who are the typical stakeholders for reusing decisions?
RQ3 What are the stakeholders concerns with respect to reusing decisions in the software architecting process?
RQ4 What are the characteristics of an ideal approach for reusing decisions?
RQ5 How do stakeholders assess the cost and benefits of decision-reuse?

We ask **RQ1** to identify practices and approaches of reusing architecture decisions; for example consider how far architects revisit decisions of previous projects for potential reuse and which information is important to them. **RQ2** identifies relevant stakeholders that would produce or consume reusable decision documentation. A typical software-intensive system project involves many different stakeholders that are involved in decision-making, such as, product managers, marketing experts, system architects, software architects, electrical engineers, or mechanical engineers. **RQ3** aims at identifying concerns of the stakeholders identified in RQ2. In this case, a concern is any interest of a stakeholder in decision-reuse; for example, which aspects of decisions are relevant for reuse. We ask **RQ4** to identify how architects imagine the ideal decision-reuse approach. This question allows us to derive use cases and requirements for a decision-reuse approach. **RQ5** explores how stakeholders evaluate the costs and benefits of reusing decisions in a concrete project. This could provide important insights into potential incentives or inhibitors for decision-reuse.

3.2. Case and subject selection

This study investigates decision-reuse in large-scale software-intensive system companies. The case for this study has been selected based on availability, which is common in software engineering research (Runeson et al., 2012). Océ R&D provides the case for this case-study because they are representative for the described study population and they were a collaborator in the ITEA2 PROMES[1] research project. The units of analysis are software architects who are either involved in product development or who regulate the architecting process (i.e., architects with a management role). Software architects working in concrete projects provide us with insights about their on-the-job architecting experience. Architects with a management role will give us insights about the blueprint of the architecting process as well as allocation of resources and people in different phases of the project. Moreover, they provide us with a higher-level long-term perspective on the strategic planning of software architecture related aspects. This high-level view is particular interesting for answering RQ4 and RQ5 since benefits of decision-reuse are not immediately visible or measurable. We aimed for a variation of subjects in architecting experience, architecting responsibilities, and roles in order to get a broad overview, to judge the significance of findings and to identify disagreements in the data.

[1] https://itea3.org/project/promes.html.

3.3. Data collection

We employed three data collection techniques: semi-structured interviews, documentation analysis, and observational techniques. All three data collection techniques have been long established in qualitative software engineering research (Seaman, 1999). Using multiple data sources and data collection techniques allows for data triangulation that strengthens conclusions because it limits the threats to validity caused by the reliance on a single data source or data collection technique (Runeson et al., 2012).

We conducted semi-structured interviews because they are particularly useful for exploratory studies as they do not strictly follow a predefined interview guide but leave room to explore ideas and concepts that emerge during an interview. Due to the slight variations in the roles of the interviewees (i.e., focus on architecting or focus on management) we tailored the interview guides towards the current role of the interviewee, e.g., in interviews with a management role we put more emphasis on assessing the cost and benefits of decision-reuse by asking additional questions and by dedicating more time to this topic during the interview.Table 1 shows that the interviews help in answering all research questions.

A major difficulty in the design of this study was that in practice decisions are rarely reused explicitly and systematically but rather implicitly and ad-hoc. Therefore, asking directly about the decision reuse process would not yield the desired information but rather let interviewees hypothesize about an imaginary reuse process. This would threaten the degree of realism in the study. Instead, we aimed at observing to what extent and how decision-reuse plays a role in practice by asking open questions and by letting the interviewees explain their daily work. This allowed us to infer what role decision reuse plays in practice and to get a holistic view on decision-reuse (Runeson et al., 2012). The complete interview design including questions can be found in the Appendix A.3. A mapping of interview questions to research questions can be found in Appendix A.4. In total, we interviewed six architects and recorded six hours of audio.

Participant observation is a data collection technique in which a researcher collects data by observing subjects in their typical environment, e.g., the researcher joins a development team (Lethbridge et al., 2005). The technique allows the researcher to capture first-hand behavior and interactions in a systematic and unobtrusive way that might not be noticed otherwise (Seaman, 1999). In typical observation studies, the observer is visibly present and collects data with the knowledge of those being observed. Compared to interviews, respondents are more likely to be “comfortable with a team member and to act naturally during observations” (Lethbridge et al., 2005). Observing subjects in addition to interviewing them helps to mitigate the inherent risk of interviews that answers are often given in a socially desirable way

Table 1
Mapping of data collection methods to research questions.

	RQ1	RQ2	RQ3	RQ4	RQ5
Interviews	•	•	•	•	•
Documentation analysis	•	•	•		
Observation	•	•	•		

(Yin, 2008). In this study, one of the authors became part of a newly formed architecture team that was tasked with the redesign and evolution of an embedded systems component (hereinafter: redesign project).

Over the period of one year we observed fifteen design meetings. Data collected consisted of field notes and audio recordings. In total, we recorded seven meetings, which resulted in 11.5 hours of audio. As suggested by Lethbridge et al. (2005), the researcher was not involved in key activities in order to not lose the perspective on the research goal. Instead the researcher performed support tasks, such as writing minutes of meetings. Observing architecture design meetings allows us to witness how architecture decisions are currently being reused and which aspects of decisions are being reused (e.g., candidate solutions, decision rationale, forces, etc.). It also provides us with insights about who is interested in reuse and their concerns. As shown in Table 1, participant observation does not contribute to answering RQ4, since an non-existing ideal approach cannot be observed, nor RQ5, as the method is unfit to elicit costs/benefits assessments from stakeholders' perspective.

In addition to the data collected through observation, we also collected architectural documents related to the redesign project as supplementary data (Lethbridge et al., 2005). Collecting documents from a real project provides us with additional insights about decision-reuse as we expect to find traces of reused decisions (like cross-references to other projects) in the documentation. In particular, it contributes to the question if and which aspects of decisions are being reused. In total, we had access to 28 documents consisting of 11 meeting minutes and 17 general documents (e.g., architectural descriptions, architecture views, plan and vision documents, and memos) Moreover, the collected documents allow us to cross check our findings from the interviews and therefore, broaden our perspective on decision-reuse. As shown in Table 1 the analysis of documents does not contribute to answering RQ4 nor RQ5, since we do not expect to find assessment of the costs and benefits of decision-reuse nor any information on a ideal reuse approaches in architecture documents.

In total, the data collection was performed in between May 2015 and July 2016. The observation of the redesign project started in May 2014 and concluded in May 2015. The observed design meetings took place in a frequency of two to four weeks. The interviews were conducted in June/July 2015.

3.4. Data analysis

The collected qualitative data has been analyzed using the constant comparative method (also called constant comparison). The constant comparative method is a systematic iterative analysis method for qualitative data (Miles et al., 2013). In each iteration the researcher generates, refines or rejects theories (Adolph et al., 2011) by continually comparing new data against previously observed data.

Constant comparison follows a systematic process, in which theories are developed bottom-up in a hierarchical manner. In every iteration, a subset of the data is coded, which is the process of assigning labels (codes), to chunks of data (incidents) that are of particular interest for answering the research questions. We coded the data using a descriptive coding approach in combination with sub-codes. Descriptive codes summarize the basic topic of an incident (e.g., "Reuse Approach") (Miles et al., 2013). The resulting code system will be an inventory of discussed topics. Sub-codes were used to further detail the topics by assigning second-orders tags to the primary tag (e.g., "Reuse Approach:Template") (Miles et al., 2013). For instance, if an interviewee told us how s/he reused architectural knowledge in one of their projects, we assigned the code "State of the Art" to the corresponding incident. If the same incident only talked about reuse based on the architect's experience, we additional assigned the sub-code "State of the Art":"Tacit Knowledge". The advantage of this coding style is that it is applicable to a diverse range of data (e.g., interviews, documents, or field notes). However, descriptive codes are usually neutral (e.g., they do not indicate if the incident is a positive or negative statement about a topic), which makes it necessary to perform an additional analysis step to qualify the coded incidents. Due to the exploratory nature of the study, we decided against a predefined set of codes generated from hypotheses or an existing theory. Instead, we let the codes progressively emerge during the data analysis. This approach is also called inductive coding. Inductive codes are better grounded empirically since the research is not determined to force-fit the data into preexisting codes (Miles et al., 2013). According to Miles et al. (2013), coding is a heuristic method of discovery in which the researcher is forced to deeply reflect on the data's core meaning and therefore, can be considered a vital part of the analysis. In later iterations, the researcher discovers similarities between new codes and previously established codes across data sources. Similar codes and the incidents they represent are carefully compared and analyzed and, if possible, summarized into overarching concepts, which represent early theories about the data. This comparison allows to cross-check different data sources. By following the constant comparative process the researcher further summarizes, refines or rejects concepts. These concepts mature over time upon reviewing additional data and eventually result in a theory that answers the research questions.

In this study, we analyzed the data (i.e., interviews, documents, field notes, and observations) in the following way.

1. The interview and meeting recordings were transcribed by two researchers. The transcribed recordings were crosschecked for accuracy and quality.
2. We applied the constant comparative method, starting with iteratively coding the interviews, observations, field notes, and documents using MAXQDA[2], which is a tool for qualitative data analysis. After each iteration, we harmonized the coding system (e.g., similar codes were combined) and all previously coded data was checked if one of the added codes was also applicable. The harmonized coding system was then used as a basis for the next iteration. Fig. 1 shows an example of a code and the related incident within MAXQDA.
3. After the second step was repeated for all data, we mapped the codes onto research questions and summarized the codes into overarching concepts. For example, the incident "you should automate it so that people that do not share this opinion can easily work with it" was coded with "Low effort tool support", which was summarized in the concept "(4.15) Providing incentives".
4. To increase the reliability of the coding, we asked another researcher who was neither involved in the study design nor in the data collection to code all data using the code system from step 3. He was allowed to make modifications to the code system, e.g., add, remove or rename codes. However, he was blinded to the previously coded incidents, so he could not see how the data has been coded by the other researcher. Afterwards we discussed the changes to the code system and also discussed a random sample of coded incidents to check the level of agreement. Based on the results of the discussion another iteration of constant comparison was performed.

The resulting concepts were discussed and described in detail by the researchers. This list of concepts served as a basis for the interpretation of results reported in Section 4, in which we also reference corresponding codes within the text in superscript (i.e., $^{(x.y)}$) to increase data transparency. Additionally, we also indicate the source of a quote, e.g., *"xyz"*$^{<1>}$ originates from interview 1 (cf. Appendix A.2). According to Hiles (2008), transparency is achieved by being explicit, clear, and open about the production, analysis and data of the study, which is a substantial concern for establishing the quality of qualitative research.

In the following, an example of the analysis process is given. The

[2] http://www.maxqda.com.

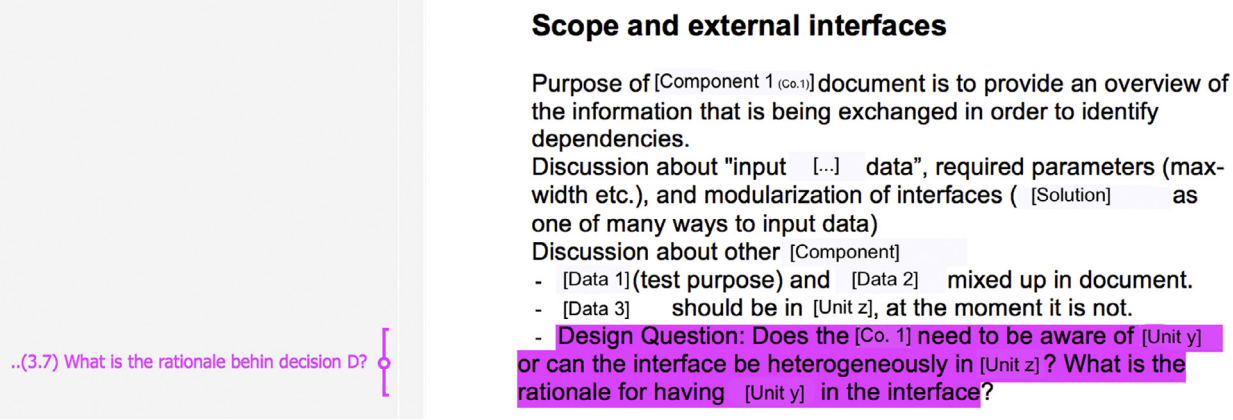


Fig. 1. Coded incident in the minutes of meeting #4.

quotes are taken from different interviews: *"People are busy, so they easily have a too small focus. People should talk to others and ask other to review. The challenge is always that people keep their team too small and they dont get the knowledge in that is available. They tend to start either again or with a too small team. Just invite guys that have a lot of knowledge but they are not in your project."*<6> This reply was initially labeled with "RQ1", "Problem: Lack of communication / knowledge exchange", "RQ4" and "Facilitating communication" as the architect talks about challenges in the context of decision-reuse and also about ways to improve decision reuse. Using the constant comparative method, other passages were found that had a similar subject, e.g., *"I walk around a lot and talk to everyone. I share this vision with everyone always, for example at the coffee machine, everyday."*<6> or *"It is only by talking."*<4> or *"In the current situation by asking someone who has done the project before me, which is already quite a nice way."*<1>. All these passages, in their respective context, refer to the need of personal interactions between architects in order to distribute knowledge about architecture decisions in the company. Thus, the code "(4.13) Facilitate interactions between architects" had been established.

The complete list of codes can be found in Table A.6 in Appendix A.2. Table A.6 also indicates the number of incidents per data source for a given concept and thus, provides further insights about how the evidence is grounded in data. However, neither the frequency of a code nor its distribution over data sources necessarily allows conclusions about its importance or significance with respect to the findings. The coding style and the corresponding size of incidents can result in multiple incidents that are only a few words long or a single incident that spans over multiple paragraphs.

4. Results

In this section, we describe the case and subjects of this study and present and discuss the results of the qualitative analysis. In order to present the results in a structured and meaningful way, we first discuss a conceptual model of decision-reuse. The conceptual model iteratively emerged through the data analysis as it is based on the patterns that we discovered during the analysis of the interviews and our observations of the architecting process in the case company. The rest of the section is organized according to research questions.

4.1. Case and subject description

Océ is a Netherland-based manufacturer of printing and copying hardware and related software. Océ operates worldwide and has research and development centers in Germany, Hungary, Romania, France, Belgium, Canada and Malaysia. The core business of Océ can be described as new product development, i.e., the transformation of a market opportunity into a tangible product. The creation of new products is taking place in large multi-disciplinary development teams, in which development and manufacturing is often distributed over multiple-sites and countries. The product development process is commonly organized according to the phase-gate process (Yazdani, 1999) and model-based development is increasingly practiced. Within the framework of the phase-gate process, sub-teams employ discipline-specific development processes. In the case of embedded software engineering it is commonly a variant of an iterative process, such as the Rational Unified Process (RUP) or Scrum.

The architecture decisions taken in the context of a project are usually cross-disciplinary and therefore, are taken within a group of architects from different domains. As one interviewee puts it: *"At Océ [...] the architectural level is the multi-disciplinary level"*<1>. The architects mentioned that the typical decision-making process usually takes place over a longer period of time and involves *"a lot of discussion between the architects"*<6>. The decision-making process has been described as highly iterative and sometimes feasibility checks are performed and prototypes are created along the way. Eventually, the discussions result in proposals, which are reviewed by a technical committee. *"In the end, decisions are taken in the technical committee."*<6>

The project that we observed as part of this study aimed to redesign and evolve an existing internal component of a product family of wide-format printers. In total four architects were involved in this redesign project. The goal was to make the software architecture of this component reusable for a wide-variety of existing and future products. The internal component is responsible for translating bitmaps into concrete jetting instructions for print nozzles and consists of software, firmware and electronics. Key architectural drivers were performance, print quality, variability, quick setup and flexibility (for development). One of the interviewees was responsible for the redesign project.

The interviewees stated their roles as follows: 2 Software Architects, 1 Domain Architect, 2 Workflow Architects, 1 Department Manager. Since job roles are not universally defined, we asked participants to describe their tasks and responsibilities to make sure that these match the typical role description of an architect, as defined by Kruchten (2008). For examples, typical responsibilities include assessing technical risks and mitigation strategies, taking key decisions about the architecture of the system, participating in project planning, or maintaining the architectural integrity of the system. The various roles mainly differ in focus, for example, the domain architect described his role as follows: "I am more occupied with the multi-disciplinary total aspects [qualities] of the product itself, like productivity or diagnostics". In contrast, the workflow architects' focus is more on the requirements, e.g., "determining the functionalities and costumer value of the product" or "taking the key decisions and validate that they will lead to a working product", while the department manager's role involves making sure that "architecture is at the right level and the

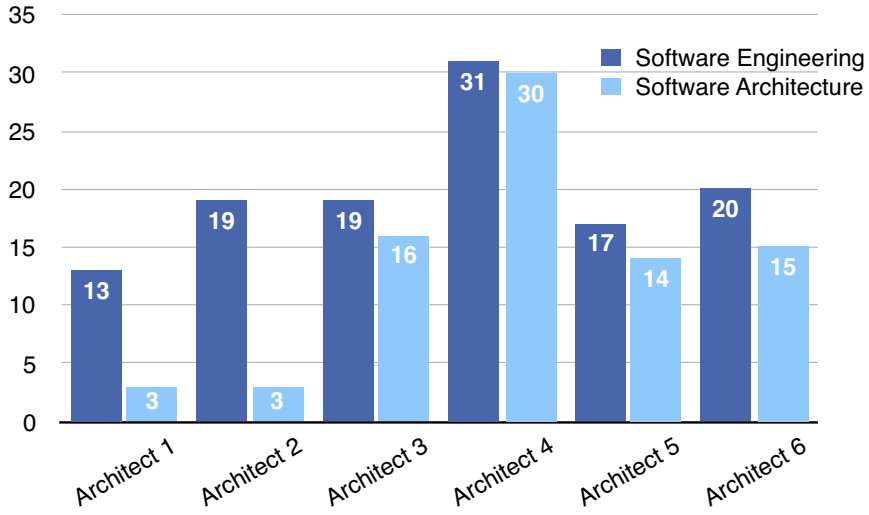


Fig. 2. Experience of the interviewed architects with software engineering and software architecture.

company stays fit from the discipline point of view". As shown in Fig. 2, the participants all have several years of experience with software engineering (min. 13 years, max. 31 years, avg. 19.8 years). With regards to their experience in software architecture , the participants can be divided into two groups: four participants with more than ten years of professional experience and two participants who have started working in the area only a few years ago (min. 3 years, max. 30 years, avg. 13.5 years).

4.2. A conceptual model for decision-reuse

Decision-reuse can be described as the use of a previously made decision in the decision-making process of an architectural problem that is similar to the problem addressed by the original decision. Reusing decisions supports the architect in accumulating knowledge about the problem- and solution-space (also called design-space) by providing reusable knowledge about the problem (e.g., forces, concerns, stakeholders) and at the same time provide insights about previously selected or considered solutions, as well as the rationale for choosing or rejecting those. In that sense, architecture decisions can be seen as a potential source of reusable architectural knowledge.

Architecture decisions usually do not constitute off-the-shelf reusable solutions like generic architectural knowledge entities, such as architecture patterns or reference architectures, since both problem and solution of a decision are closely linked to the original system-specific context. Hence, before a decision can be reused in a different system context, the architectural knowledge embedded in a decision must first be converted from one system-specific context to a different system-specific context. According to Farenhorst and de Boer (2009), this conversion can be performed through abstraction and utilization. Abstraction is the conversion between system-specific to system-generic AK, which is AK that is applicable to a wide variety of systems. Utilization is the conversion from system-generic to system-specific AK, which is AK that is only applicable to one specific system. An example of a decision-reuse approach that is based on the principles of abstraction and utilization is Zimmermann's decision guidance model (Zimmermann, 2011). A decision guidance model abstracts reusable decisions in a system-generic manner, which can be utilized by an architect to reuse decisions in a system-specific context.

However, the results of our study show no evidence of a systematic process of turning system-specific decisions to system-generic decisions(1.1). The architects did not invest in abstracting decisions but rather immediately converted a decision from one system-specific context to a different system-specific context. The system context describes the circumstances that prevailed when taking a decision. It embodies the conditions, the situation and the environment that influenced the decision-making process. It is defined by the problem that had to be solved, the set of forces and their priority that influenced the architect when taking the decision, the stakeholders involved in the process, the considered alternatives and the set of previous decisions that affected the choice of alternatives as well as the final decision with its rational (cf. van Heesch et al. (2012b)). It is important to note that the system context does not only depend on the system-being-developed but also has a temporal dependency, such as, the solution alternatives that were available at a particular point in time. Hence, the system context provides important information that is necessary to fully understand a decision and to assess its reusability and applicability in a different system. As shown in Fig. 3, this immediate conversion implies that the architect understands and evaluates the reused decision as well as the system context in which the decision was taken. The architect compares the original system context to the new system context, internalizes the architecture knowledge embedded in the decision (conversion of explicit to tacit knowledge Farenhorst and de Boer, 2009), and finally applies the knowledge embedded in the reused decision in the decision-making process of a new problem. This ad-hoc reuse approach can rather be seen as a form of knowledge refinement, the conversion from system-specific AK to system-specific AK (Farenhorst and de Boer, 2009), rather than a combination of abstraction and utilization.

The results of this study suggest that organizations do not invest resources on abstracting decisions. In contrast to an ad-hoc decision-reuse, a systematic reuse approach based on an explicit abstraction of a decision into a system-generic format and a subsequent utilization requires an upfront investment that is not guaranteed to pay-off as it remains open if the abstracted decision will be reused after all.

Reuse approaches can be distinguished into *implicit* reuse, *explicit* reuse and *hybrid* reuse, a combination of the first two(1.2). When *implicitly reusing decisions* the architect relies only on his experience. That is, the architects only consider decisions that s/he has taken or which s/he has heard of. These decisions are part of the architect's tacit repository of reusable decisions. Often this type of reuse happens subconsciously and is motivated by the tendency of architects to prefer known solutions. Several studies show that architects often implicitly reuse solutions that they have previously considered when making a decision (van Heesch and Avgeriou, 2011; van Vliet and Tang, 2016). Implicit reuse can be associated with naturalistic decision-making, which is based on intuition and experience. The architect does not exhaustively explore the problem and available solutions but primarily

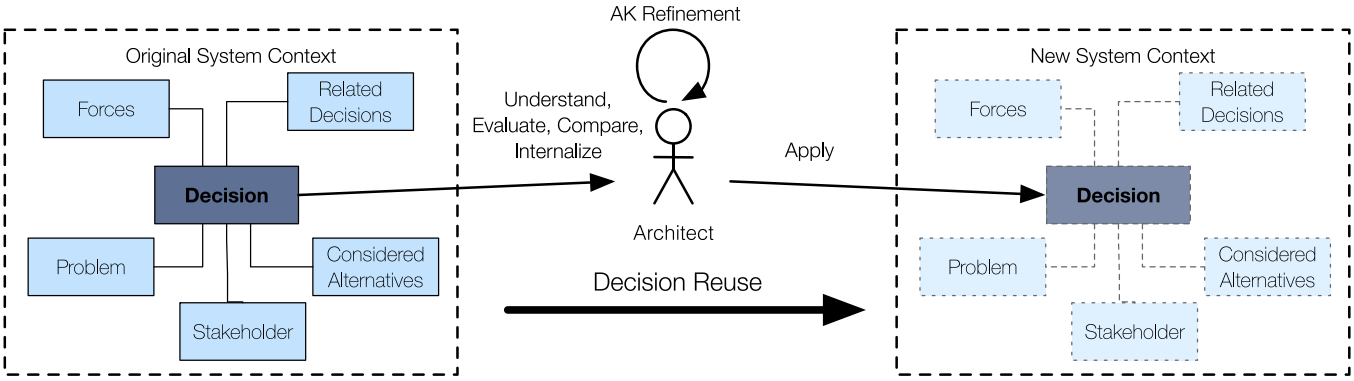


Fig. 3. Ad-hoc decision-reuse as a form of architectural knowledge refinement.

relies on his expertise (van Vliet and Tang, 2016).

Explicit decision-reuse involves the architect actively seeking for reusable knowledge, e.g., by asking colleagues, scanning documentation of past decisions, or using decision guidance models. Explicit decision-reuse can be systematic, like Zimmermann's guidance models (Zimmermann, 2011) that is based on abstraction and utilization, or ad-hoc as we observed in this case study. This style of reuse resembles a rational decision-making approach, which is based on a careful exploration of the problem, alternatives and trade-offs (van Vliet and Tang, 2016). We found that in practice there is no hardline between implicit and explicit decision-reuse. Often architects rely on a *hybrid* form of reuse that is partly based on experience and partly requires the architect to explicitly seek for reusable knowledge. For instance, the architect's experience is often only a starting point for discovering potentially reusable decisions. In many cases detailed knowledge about these candidate decisions vaporized and needs to be actively sought, e.g., by talking to colleagues.

By looking at decision-reuse from a process perspective we can establish a general decision-reuse process[(1.7-1.12)]. The general decision-reuse process shown in Fig. 4 is based on Markus (2001a), who proposed a general process of knowledge reuse. We adapted this general process to the software architecture domain based on the results of this case study.

The reuse process can be divided into two phases: preparation and execution. Preparation and execution are not likely to take place at the same time and might not be performed by the same person. The preparation phase involves the *documentation* of a decision for reuse and its *distribution*. The execution phase is the actual reuse of a decision in a different system and consists of *discovery, retrieval, evaluation*, and *adaptation*. The process is generic enough that we can map reuse approaches onto it that are based on abstraction and utilization as well as ad-hoc refinement. It also supports explicit, implicit and hybrid reuse. Depending on the concrete reuse strategy some activities might not be performed or are only performed implicitly, e.g., discovering decisions by recalling them from memory.

Documentation is the process of documenting a decision in way that makes it reusable. The documentation approach and the information that needs to be codified depends on the documentation strategy. The strategy determines which knowledge remains personalized and which knowledge must be made explicit. Basically any decision documentation approach is sufficient, e.g., decision views (van Heesch et al., 2012a), decision templates (Tyree and Akerman, 2005). In the case of implicit reuse, this activity concerns the internalization of the existing decision.

Distribution is the process of announcing or publishing a decision in a way that makes it discoverable. According to Markus (2001b), distribution can either be passive, for example, by storing it in a searchable knowledge repository, or active, for example, by announcing it during daily scrum meetings. The goal of distribution is that architects are able to discover a decision.

Discovery is the process of identifying or finding a decision that is applicable for reuse because it solves the same or a similar architectural problem. The discovery activity leads to a set of candidate decisions. However, it is not yet determined if a decision is relevant or that it will help the architect in decision-making. Moreover, the discovery activity does not need to provide the complete decision but might only show some meta-information that is necessary to judge if a decision is potentially relevant and where the remaining information can be found. Decisions can be discovered in multiple ways like searching information systems, asking colleagues or recalling from memory. The way decisions can be discovered depends on the adopted reuse strategy in the preparation phase.

Retrieval is the process of retrieving the full description of a decision, for example, by accessing a decision repository (codification strategy), by interviewing colleagues (personalization), or a combination of the two.

Evaluation is the process of understanding the original context of a decision and assessing whether the decision is applicable in the new context. When reusing a decision, the original context must be evaluated, e.g., what were the forces and their priorities, what trade-offs were made, what was the impact of the decisions on the system (i.e., was it a good or bad decision), etc.

Adaptation is the process of using a decision or parts of it, like a subset of forces or a considered alternative, in the decision-making process of a different decision. The reused decision will be converted from the original context to the new context.

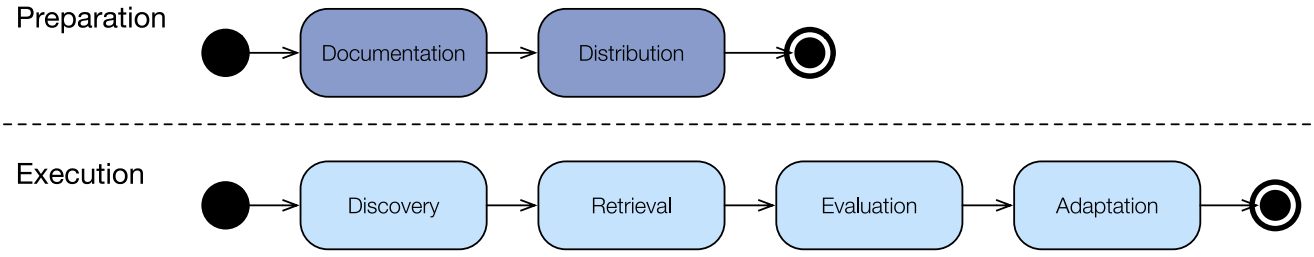


Fig. 4. A conceptual decision-reuse process.

4.3. RQ1 - Current state of reuse

In RQ1 we investigate the current state of decision-reuse. The interviews and observations show that *reuse is an important concern*(1.1) for architects when making decisions and that *decisions are frequently being reused*(1.3). As one of the architects put it: *"I reuse a lot of what I did in previous projects with previous experiences. I sometimes look, how did I do that in 2003, how did I do that in another project"*<3>. The motivation for decision-reuse is manifold(1.4). The architects reported that the main motivation to reach out to other projects is *seeking inspiration*, for example, how a similar architectural problem has been solved before. They are hoping to find *information about considered alternatives and the decision rationale as well as models and simulations* that were used to evaluate alternatives, such as performance models. For example, one architect said: *"We use decisions from previous products to make it better, but we always need to consider the context we are in. It is not one-to-one reuse but it is more what was the reasoning there and also some experience is what I want to have and then, yes, I reuse it"*<3>. Another important motivation for decision-reuse is the fact that new projects usually start as forks of previous products, as shown in Fig. 5. These forking projects require the architect to *revisit important decisions* and, if need be, adapt those decisions to the new context. However, the data showed *no evidence of a systematic reuse* of decisions in the architecting process.

The interviews showed that architects often face *similar decision problems in different projects*(1.5). Yet, the context of the decisions can be different and so could be the actual outcome of the decisions as they might address concerns with a different priority. For instance, one project focused on reducing the cost-price of a printer while another project has the primary goal to improve the quality of prints. In both projects the high-level architecture problems are similar, e.g., software-hardware decomposition, task scheduling, or power management. Nevertheless, many aspects of the decision can be reused even if the system context is slightly different. For example, a reused decision can serve as inspiration or guideline, and models and simulations can be adapted to a new decision problem. As one interviewee put it: *"Some things tend to come back but there are differences each time. They could reuse the methodology. Productivity, timing, power management is always an issue. [...] Right now this [reuse of decisions] is not really happening. It is either from experience or by doing it again."*<1>

Talking about this issue an interviewee stated that many *reappearing problems are solved in the company-specific reference architecture or other reusable artifacts*(1.6). Also from a management perspective reuse of decisions is an important concern since aligning decisions across projects and reusing solutions can save both money and time. As one interviewee stated: *"There are similar problems in multiple projects. [...] It happens, that they take different decisions"*<6>. One architect particularly highlighted that a challenge with regards to reuse in software-intensive system projects is the cooperation between different disciplines(1.17) since there is no common domain language and often no common tooling available. As he puts it: *"[...] group[s] mostly have their own way of working for years, so it is hard to mess around with that. What we do lack is some support for multi-disciplinary interactions and multi-disciplinary corporation"*<2>.

Below we illustrate, using the conceptual model for decision-reuse (cf. Fig. 4), how decisions are being reused and to what extent and how the development process of the case company supports or impedes decision-reuse.

4.3.1. Preparation

Several issues related to the *documentation* of decisions were identified. *Architecture decisions are not represented as first-class entities* in the documentation, although, they play an important role in the development process of Oc . The documentation is solution-oriented specifying the components, interfaces and connectors of the chosen alternative rather than the rationale of the decision, considered alternatives or other information about the decision-making process. Only in a few occasions decisions were elaborately described. However, the interviewees stated that this is the exception and that it is usually very hard to find the rationale documented let alone considered alternatives and the argumentation. One interviewee stated that *"a lot of the discussion is lost and in the end the discussion leads to one or two proposals but whether all rational behind the proposals is documented, I am afraid not. In the end the final proposal is there but not the reasoning how we came to the proposal or why."*<6>

Architecture documentation is not coherent. We found that the majority of documents do not follow a predefined structure and that it is often the responsibility of the architect to decide what needs to be documented. Overall, reuse did not seem to be a concern that was prominent when documenting.

Information about decisions are scattered over multiple documents,

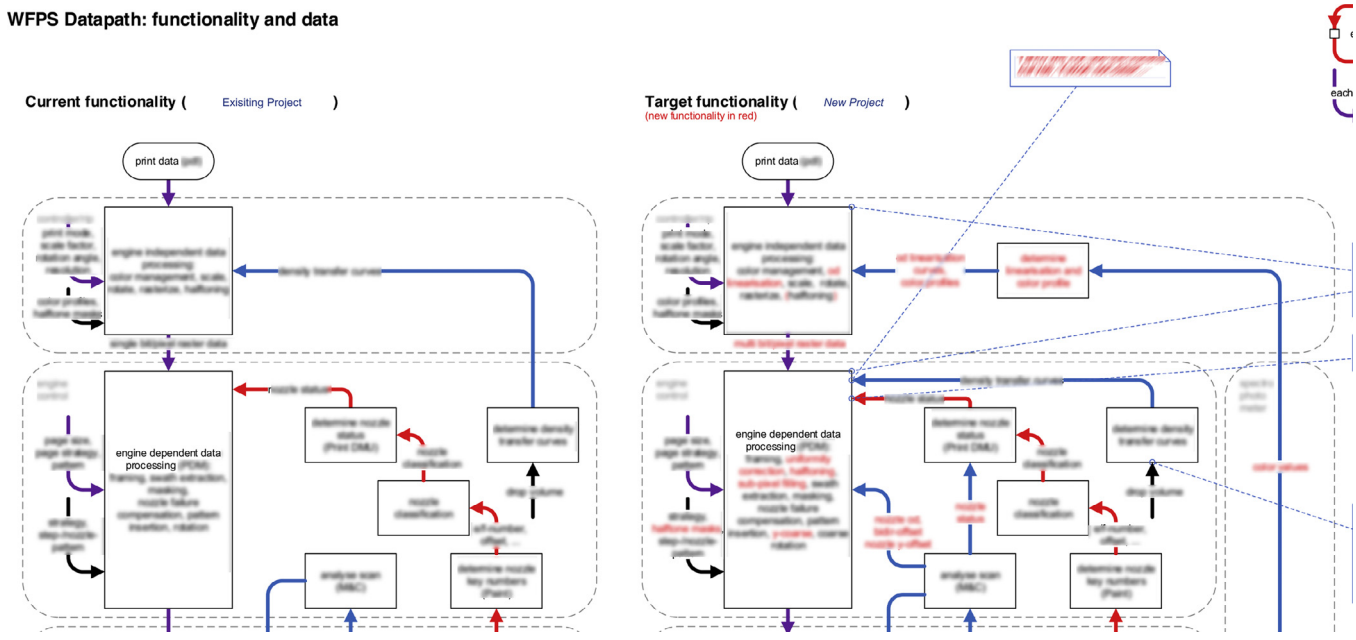
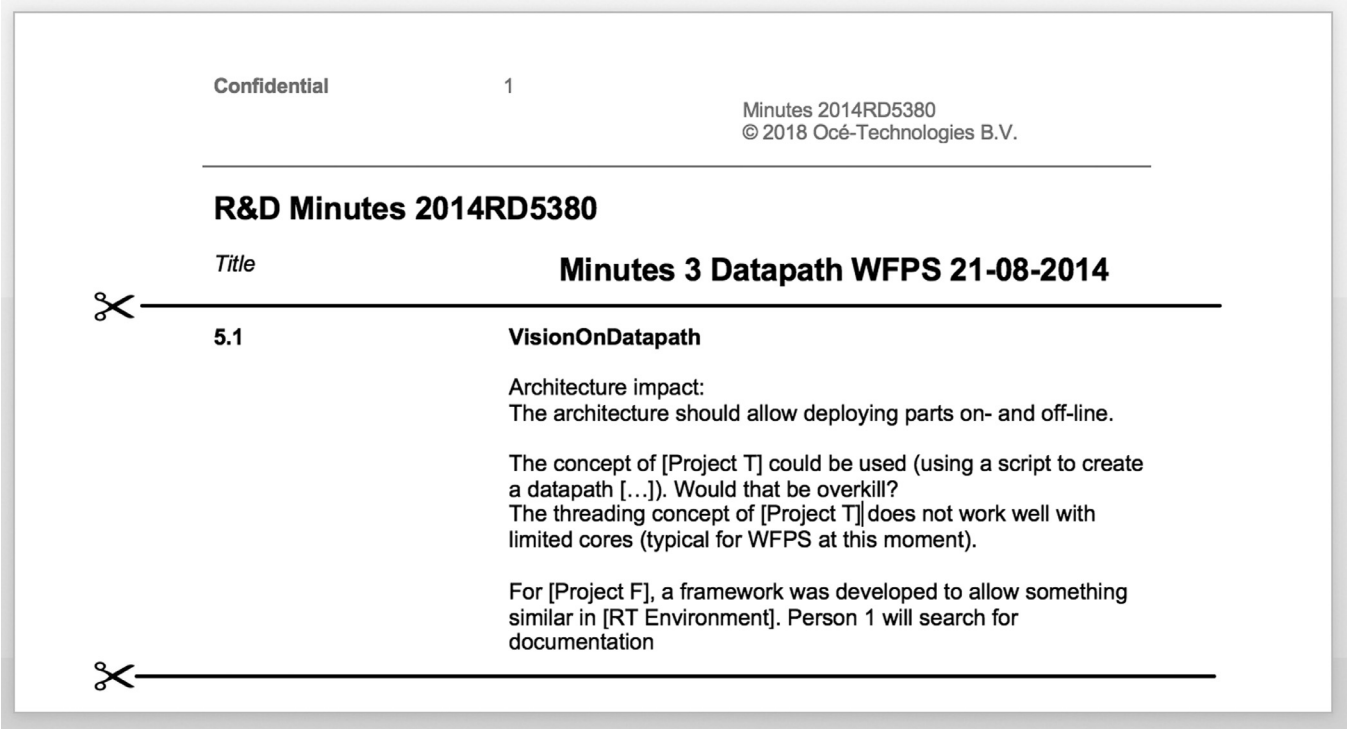


Fig. 5. Excerpt of a document illustrating functionality in the architecture of an existing project and the target functionality in the spin-off project (obfuscated).


Confidential 1

Minutes 2014RD5380
© 2018 Océ-Technologies B.V.

R&D Minutes 2014RD5380

Title **Minutes 3 Datapath WFPS 21-08-2014**

5.1 VisionOnDatapath

Architecture impact:
The architecture should allow deploying parts on- and off-line.

The concept of [Project T] could be used (using a script to create a datapath […]). Would that be overkill?
The threading concept of [Project T] does not work well with limited cores (typical for WFPS at this moment).

For [Project F], a framework was developed to allow something similar in [RT Environment]. Person 1 will search for documentation

Fig. 6. Obfuscated minutes excerpt from Meeting #3. A participant remembers a similar decision in Project F (*discovery*) and agrees to search for documentation (*retrieval*).

e.g., in separate memos. These documents often do not link to each other, which makes it hardly possible to identify all documents that contain information about a decision.*"The problem is that there is no template for such a document. You would end up with quite a lot of documents that have parts of the information."*<2> The observation that decisions are not explicitly, comprehensively, and concisely documented is typical in industry and has been reported by other industrial case studies, such as Manteuffel et al. (2014). Moreover, the focus on a solution-oriented documentation may also be related to the involvement of multiple disciplines in the development process since other disciplines might not acknowledge architecture decisions in the same way as software architecture. Moreover, the interviews indicated that architects often do not find the information that they were looking for in the documentation(1.16.) For example, *"[a colleague] asked me about a [decision] in a component. And then I looked back how good did I describe that [decision]. I could have done a better job, so I know why he asked."*<3> and *"You ask [a colleague about] some documentation or rationale, and most of the time he just explains it to you. [F]rom the documentation you wont get the rationale, you will only find the specification."*<5>

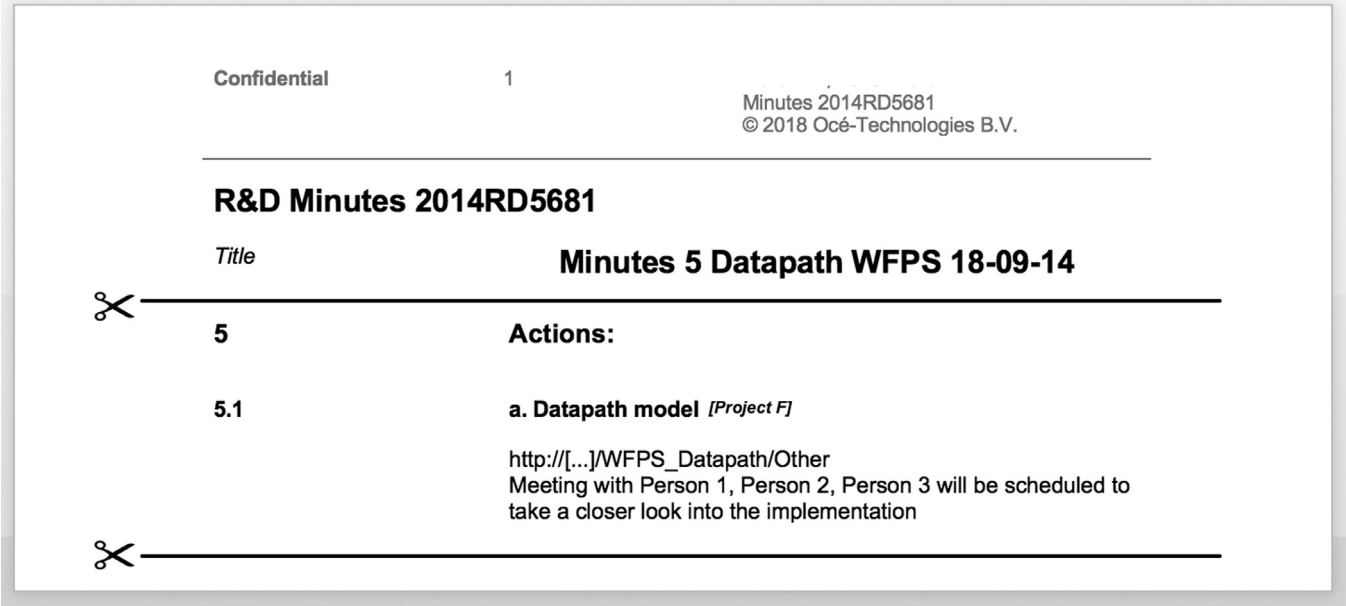
Documents are stored in a variety of information systems. Although decisions are not explicitly documented, documents still hold information about decisions. However, there are multiple information systems in use, which makes it difficult to find the corresponding document. One example is an unstructured company-wide archive that primarily serves as a storage for all sort of documents including memos and meeting minutes. Another example is a project-specific information system that takes specific project-related documents, such as architecture specifications or milestone reports. Additionally, decisions can be found Microsoft Sharepoint or Wikis, which are often dedicated to sub-projects or reusable assets.

The goal of decision *distribution* is to make architects aware of decisions and thus, it is closely linked to corporate knowledge sharing. Our findings show that multiple mechanisms are used to facilitate knowledge sharing, which includes: *advocating personal interactions; frequent meetings; shared office spaces; careful team composition; roaming experts; and intranet.*

The majority of decisions are announced in meetings and personal interactions (like desk-visits or phone calls). It was mentioned that Océ actively advocates personal interactions and meetings as part of their development process. For example, the majority of employees related to a product development team share an office space. Another example are architecture steering group meetings to align decisions across product-lines. In addition to that team composition is an important factor to increase decision reuse as one interviewee mentioned. *"I always look at who is needed in a project so that I can be sure that they are going to do the right things. Because the knowledge is in the people and not because it is nicely documented somewhere and they can easily get it. So if I have a new clean team, there is no way that they can setup a better software for a project or discuss architecture. Experienced individuals are very important."*<6> Another approach to increase knowledge-sharing are domain experts, which move between projects as consultants and thus, take the decision knowledge with them to the next project. In a few but rare occasions, decisions were also announced via the intranet (news bulletin). However, one interviewee also stated that many opportunities for reuse are still missed due to a lack of communication(1.15) despite the company's effort to increase knowledge sharing. *"There are similar choices in multiple projects and sometimes there is hardly any communication between these guys."*<6> and *"People are busy, so they easily have a too small focus. They should talk to others."*<6>

4.3.2. Execution

The interviews showed that decisions are primarily *discovered* based on either *experience (recall from memory)* or by *talking to colleagues who are experts in a particular field*. Again, the experience of the architect plays an important role for *discovery*. As one of the interviewees stated: *"I guess [architect reuse knowledge] a lot. But it is in the people. It is the experience of the architects. Of course they have documents that they use but most of the reuse is because people know"*<6>. The interviewees stated that they often *recall a decision from memory* since they were responsible for it or otherwise involved in the decision-making process. It is important to note that this does not imply that they remember the complete decision with all of its aspects but rather that they have a hunch


Confidential 1

Minutes 2014RD5681
© 2018 Océ-Technologies B.V.

R&D Minutes 2014RD5681

Title **Minutes 5 Datapath WFPS 18-09-14**

5 **Actions:**

5.1 **a. Datapath model** *[Project F]*

http://[...]/WFPS_Datapath/Other
Meeting with Person 1, Person 2, Person 3 will be scheduled to take a closer look into the implementation

Fig. 7. Obfuscated minutes excerpt from Meeting #5. Further investigation is needed to *evaluate* if the decision taken in Project F is applicable in the new project.

that something similar has been decided before. This hunch is the start for further investigations *(retrieval)* (cf. Fig. 6 and 7).

Another common way of discovering decisions is through *discussions with colleagues*, via desk-visits, hallway discussions, or during meetings. A general concern expressed by the interviewed architects is the effort associated with finding suitable decisions for reuse *(discovery)* and processing the documentation to find the relevant information *(retrieval)*(1.14).

The interviews show that architects *do not systematically scan existing documentation* from other projects for potential decisions. Two issues were identified that hinder systematic scanning. First, it is *not clear where to search* since there are multiple information systems that might contain relevant information. Second, the *search is not considered powerful enough* as it does not allow for a targeted search of decisions, e.g., by problem, by solution, etc. As one architect stated this issue: he must know that the documents exists in order to find it. Talking about this issue an interviewee said: *"There could be very good things done in other projects and I would only know them because I would talk to people and not because I would find the information very quickly.[...] There are the projects I know, where I know my way around and then I can refer to those. But there is no structured way of finding information"*<1>. The interviews show that it is *hardly the case that architects explicitly search for decisions* that can be reused. In most cases, reuse is limited in the sense that architects *only reuse familiar decisions*, in which they were involved or they have heard about from colleagues. This leaves a large amount of potential decisions, and thus important architectural knowledge, ignored.

After identifying potential decisions, detailed information about a decision are *retrieved* by searching in the document management systems for a particular document; by asking experts who were involved in the decision-making process or worked on a part of the system that has been affected by the decision; by keeping personal archives (e.g. file archives or email archives) related to a subject or architectural problem that lies within the responsibility of the architect. Overall we found that due to the incompleteness of decision documentation (no rationale, no requirements, etc.), *architects employ a hybrid approach when retrieving a decision* using both codified and personalized knowledge by searching for documentation and by asking experts. However, the interviews also indicate a preference of architects towards personalized knowledge as the documentation often does not contain the information they are looking for.

The interviews showed that architects do not follow a systematic approach for *evaluating* the context of a decision and *adapting* it to a new architectural decision problem. While evaluation can be supported by a systematic process, such as, checklists, the adaption of a decision is more complex since it depends on what information the architect wants to reuse. Talking about this issue an interviewee said: *"So we use the work from previous products lines to make it better, but we need always to take on the context we are in. So it is not one-to-one reuse but it is more what was reasoning there and also some experience (outcome) is what I want to have and then, yes I use it"*<3>. In Section 4.5 we present the stakeholder concerns related to evaluating the context of a decision. The concerns illustrate which information architects are looking for when evaluating decisions for reuse.

4.3.3. Discussion

The observation that architects rarely reuse decisions but rather tend to use them as inspiration is closely related to the software-intensive systems domain. In the studied domain new products often contain innovations coming from other disciplines, like chemical, hardware or mechanical improvements, which then need to be reflected in the software realm. Although the high-level architectural decomposition remains the same (since it has been solved in a reference architecture) the innovations still have a huge impact on the software. Since the solution to these architectural problems usually require some sort of custom solution, systematic reuse like guidance models or system-generic AK like patterns are less applicable. This is also reflected in the interviews, which indicate that architects at Océ rarely consult system-generic AK sources(1.13), such as patterns (5 out of 6 architects indicated no use or very little use of system-generic AK). In mono-disciplinary components of the systems, such as a web-based remote user interface, systematic reuse and generic AK tends be more applicable.

Our results also show that experience plays an important role in the decision-making process. This supports the argument of van Vliet and Tang (2016) that decision-making in software architecture is not entirely rational and to some extent based on intuitions and experiences with a tendency of architects to reuse known solutions. This is further supported by the following excerpts from the interviews: *"Most of time we only discuss the most promising [alternative] and if someone says it is not going to work we should do it [differently][.] we work it out"*<2>; *"It doesn't happen that often that you really compare alternatives. You work out two or three and then you say, now we are going to compare."*<4>. Likewise, van Heesch and Avgeriou (2011) also found that architects prefer

solutions they are familiar with as they impose a manageable risk since shortcomings are often well understood and can be mitigated. Moreover, unfamiliar alternatives require substantial extra effort to analyze, which conflicts with the strict time-to-market characteristics of the observed industry.

The reliance on experience and knowledge personalization has the inherent risks that experts might leave the company or fail to remember. Although, it was stated that this happens rarely and that most of the time large parts of the decision can be reconstructed. In many cases the available documentation helps experts to remember sufficient details. Overall, knowledge vaporization has more severe consequences for decision discovery when it is not possible to discover a decision because it was forgotten. As long as a decision can be discovered, knowledge pertaining to this decision can be, at least partially, recovered.

4.4. RQ2 - Typical stakeholders

RQ2 aimed at identifying typical stakeholder groups that have a stake in decision-reuse. We distinguish between two roles: producer of reusable decisions and consumer of reusable decisions. The stakeholders that are involved in the preparation phase of decision-reuse are *producers* while the stakeholders who actively reuse decisions are *consumers* (Tang et al., 2010). The categorization of stakeholders into producers and consumers follows ISO/IEC/IEEE (2011) standard for architecture specifications. Table 2 presents the stakeholders that we identified in this case study. The case study showed that architecture decisions usually concern cross-disciplinary problems, while mono-disciplinary decisions are regarded as high-level design. Based on this notation of architecture, the presented stakeholder cannot be uniquely associated with a single domain.

4.5. RQ3 - Concerns of stakeholders

RQ3 aimed at identifying concerns of stakeholders with respect to decision-reuse. The term “concern” here refers to a stakeholder’s interest in an architecture description as defined in ISO/IEC/IEEE 42010. The study found several stakeholder concerns related to reusing architecture decisions. Table 3 provides an overview of the identified concerns. However, due to the exploratory nature of this study we do not claim that the list of concerns is exhaustive or prioritized. Moreover, the concerns are linked to the activities of the decision-reuse process. Concerns may play a role in more than one activity. For example, C1 plays a role when documenting and distributing a decision since the stakeholders needs to know whether a decision is a key decision or not. Likewise, it is important during discovery to identify all key decisions of a particular project. Several of the concerns that we identified were also identified by earlier studies related to the documentation and utilization of architecture decisions (i.e., C1, C3, C6, C7, C8, C9 (van Heesch et al., 2012a; van Heesch et al., 2012b). A complete description of the concerns is shown in Appendix A.1.

Table 3
Stakeholder concerns related to reusing architecture decisions.

ID	Concern
C1	What were key decisions in project X (for subsystem S)?
C2	Which decisions D_i addressed problem P?
C3	Which decisions D_1 considered to use alternative A?
C4	Which projects PR_i share problem P?
C5	What decision D_i has been reused in the decision-making process of decision $_2$?
C6	Which stakeholders SH_i were involved in decision D?
C7	What is the rationale behind decision D?
C8	Which alternatives A_i were considered for decision D?
C9	What forces F_i influenced decision D?
C10	Which trade-offs T_i were made for decision D?
C11	What is the high-level system context of decision D?
C12	Which model M was used for decision D?

4.6. RQ4 - Characteristics of an ideal approach

Any decision reuse approach that does not purely rely on personalized knowledge, i.e., the sole experience of an architect and his peers, requires some level of decision documentation. While decision documentation approaches and their ideal characteristics have been discussed in several studies (Farenhorst and van Vliet, 2009; Hoorn et al., 2011; Manteuffel et al., 2014; Capilla et al., 2015), our study focusses particularly on the demands that arise from the context of software-intensive system projects.

Software-intensive system projects can be characterized by a multi-disciplinary and distributed development environment, involving numerous stakeholders with conflicting interests, and a competitive business environment that requires a short time-to-market. As shown in Table 4, for each of these characteristics certain demands arise. We will discuss these demands and present recommendations that solve aspects of the demand, following the structure presented in Table 4.

The results of RQ1 show that both *documentation* and *discovery* have a large potential for improvement. It is important note that due to the design of the study the list of characteristics, demands, and recommendations is not exhaustive. However, practitioners can turn to these recommendations and in combination with the stakeholder concerns identified in RQ3, they serve as a first step to derive concrete requirements for an approach that facilitates decision-reuse.

(a) Time-to-market

Time-to-market is an important concern at Oc . *“It is always a balance between time-to-market and the time to get a real good product out.”*<6> The high time-pressure to release a product means that decision documentation has a lower priority as it does not directly contribute to the product. *“When we have to change something in the software we had so much time pressure that we didnt update the documents. So at the end, documents are completely out of date and nobody reads them.”*<5> *“Because at the time you are [making a decision], the barrier to document it is too high. [...] I dont want to write a memo about that because it would probably take two hours.”*<5>

The two statements illustrate that it is important to reduce the effort to document and maintain decision documentation but also to make it very easy and fast to discover reusable decisions. RQ1 showed that

Table 2
Typical stakeholders for decision-reuse.

Stakeholder	Role	Description
Architect(2.1)	Producer	The architect, as a producer, is responsible for documenting and distributing a decision. This stakeholder has the goal to make decisions reusable.
	Consumer	As a consumer, the architect is in the process of taking a decision and is seeking for a reusable decision.
Reviewer(2.2)	Consumer	The reviewer is a consumer of decisions. The primary goal of the reviewer is to reuse decisions as source of information in the review process, e.g., to compare a decision to past solutions or to assess how another project solved a comparable problem.
Product-line Architect(2.3)	Consumer	The product (line) architect is responsible for a product-family or product-line. As such, this stakeholder’s goal is to align the architecture of multiple products and bundle the efforts of multiple teams. Therefore, it is necessary that synergies, like shared architectural problems, can be identified. This stakeholder aims at planning and streamlining the reuse of decisions.

Table 4
Overview of characteristics & recommendations.

Characteristic	Reuse activity	Recommendation
(a) Time-to-market	Documentation	Guidance
		Split documentation
		Value-based documentation
		Incremental documentation
	Discovery	Decision overviews
		Comprehensive search
		Relating decisions
(b) Multi-disciplinary	Documentation	Discipline-independent tooling
		Flexibility
	Discovery	One source of information
(c) Distributed development	Documentation & discovery	Informationsystem support

architects complain that architecture documentation is often too long, which makes it hard to find the right information, or it does not contain the right information at all, i.e., design rationale or decision alternatives. AS one architect puts it: *"Finding it should be easier then doing it again."*<1>.

a.1) Documentation

a.1.1) Guidance A common concern expressed by the interviewees was the need for guidance that instructs architects when to document what for which purpose. Guidance can also increase the level of standardization of decision documentation, which is important to know what information is actually recorded and can be found in the documentation. Several approaches were discussed that potentially offer guidance(4.6), which include decision templates like the one proposed by Tyree and Akerman (2005), decision viewpoint-based approaches as proposed by van Heesch et al. (2012a), or simple checklists that let architects choose their own format but ensures a certain standard with regards to decision documentation.

a.1.2) Value-based Documentation The results of RQ1 suggest that architects believe that a large amount of a decision can be reconstructed based on a few hints. *"So there is probably some pragmatic balance between structured documentation of architecture decisions or knowledge and whats in the head of the people."*<6> It can therefore be argued that in the context of decision-reuse, documentation can be kept to a minimum as a long as the decisions can be discovered. The idea of a value-based documentation of decisions has also been explored by Falessi et al. (2013), who claim that a full documentation of decisions "is typically too onerous for systematic industrial use as it is not cost-effective" and therefore, should be tailored to the intended use case. van Heesch et al. (2012a) argue in a similar way, that any decision documentation produced should be based on the concerns of intended stakeholders. Also Manteuffel et al. (2014) found in a study that architects do not want a tool that enforces complete documentation but should offer the possibility to document only certain aspects of a decision. However, we need to gain a better understanding of which aspects of a decision should be documented so that it can be discovered and be reconstructed. For example, a hybrid decision documentation approach, combining codification and personalization, which briefly describes the problem and solution, and captures 'who knows what' about a decision might be promising, as also suggested by Farenhorst et al. (2007b).

a.1.3) Split Documentation A strategy discussed in the interviews to reduce the effort of documenting and maintain decisions is to split decision documentation into two parts: a brief summary that is being kept up-to-date and extended information that are not being maintained(4.3). *"What I think would be the right way, is a very limited set of documentation that you really maintain and [it] should focus on helping somebody [...] on finding its way through the knowledge. All other things should be archived and not maintained."*<1> This split reduces the effort of documentation and its maintenance while still allowing architects to discover potentially reusable decisions. The decision summary refers to further information pertaining to a decision(4.5). A prerequisite for these kind of decision overview documents is a system that allows to create trace links between documents, e.g., based on unique IDs or URIs.

a.1.4) Incremental Documentation To further lower the effort of documentation one interviewee suggested that when reusing decisions an incremental documentation could be often sufficient by only documenting the differences compared to the original decision(4.4). *"So we have got this documentation. What we used to do was to create an addenda to that on how things changed in the actual project. So we have a general description of action control. And then I would write an addenda on that saying now we using action control [in the following way]."*<2>

a.2) Discovery

a.2.1) Decision Overviews To reduce the effort associated with decision discovery, it has been suggested that decision documentation should be brief and centered on the most important information(4.1) by providing an overview. For example, one interviewee proposed to create mandatory decision overviews for every project at the end of certain milestones, e.g., in the form a decision relation viewpoint as proposed by van Heesch et al. (2012a). These overviews provide a summary of the key architectural decisions(4.2) and make it easy to discover important decisions in a project that qualify for reuse (cf. Concern C1). Moreover, the documentation of decisions after-the-fact is considered to reduce the effort of documentation (Jansen et al., 2008). This recommendation can be combined with split documentation and value-based documentation.

a.2.2) Comprehensive search Information systems (IS) can support decision-reuse in various way, for example, by providing a central repository of AK. The interviews showed that several IS are in use but that it is too difficult to find information. Therefore, a comprehensive search functionality(4.11) that allows to semantically search for decisions, e.g., based on the decision problem, considered alternatives, or related forces, is an important requirement for an effect IS that aims to increase reuse.

a.2.3) Relating Decisions A further suggestion to ease discoverability it the possibility to link decisions to related decisions. These decision relationship allow architects to find potential decisions that are useful in the context of the reused decision and therefore, might also be relevant when reusing a particular decision.

b) Multi-disciplinary Due to the involvement of multiple disciplines in the development process, a decision-reuse approach for the embedded systems industry needs to consider additional aspects compared to a mono-disciplinary project. According to one architect *"the complexity in the software architecture [in software-intensive systems] is in the multi-domain part, the interactions and the different information to keep them consistent. [T]he scope is not only the software here but it is multi-domain."*<5> *b.1) Documentation*

b.1.1) Discipline-Independent Tooling In a multi-disciplinary environment it is difficult for a tool to interface with all discipline-specific tools (e.g., CAD[3] in mechanical engineering; IDEs[4] in software engineering; circuit simulator in electrical engineering) due to the lack of a common plugin architecture. However, decisions concerning the architecture of a system often involve the input from multiple-disciplines and therefore, any tool must be available to all disciplines involved in the development process(4.7), for example, either as a standalone tool or by integrating with general-purpose tools, such as Microsoft Office, that are used by all disciplines.

b.1.2) Flexibility Decision-documentation approaches for software-intensive system project requires a high degree of flexibility(4.8); it should be versatile and not limit the use of multiple modeling approaches (e.g. UML) since there is no common graphical notation that is able to express the concerns of all disciplines (e.g., interfaces between software and mechanical components). Depending on the problem and involved disciplines, a decision can sometimes be easily documented in

[3] Computer-aided Design.
[4] Integrated Development Environments.

textual form, sometimes a graphical form is more appropriate, and in other situations a matrix is more suitable. Moreover, some decisions can be documented in a single sentence, while other require an elaborate documentation. Farenhorst et al. (2007b) also emphasize flexibility due to the creative nature of architecting, tools should be descriptive rather than being prescriptive.

b.2) Discovery

b.2.1) One source of information Due to involvement of multiple disciplines, architectural knowledge is often replicated and compartmentalized across disciplines, making it difficult to judge whether a certain decision is the latest and only version, especially with regards to *discovery* and *retrieval*. A single source of information is desirable(4.7), especially, with regards to propagating changes in decisions as illustrated by the following story. *"What we did in the past is that we use CAD data from mechanics. They export the CAD data and we use it directly in the embedded software, so the distance between two sensors is based on data from the CAD. In other projects we had to ask for the distance between two sensors, they would measure it tell us it is 5cm. Then we documented this in another document stating the distance is 5cm. And then finally they decided to change the position and of course all the data was not updated anymore. So now we dont have to put the number 5 everywhere but the data is in the CAD data. Instead of documenting it in Microsoft Word we prefer to refer to models. CAD is a model for us."*<5> The story shows that it doesn't have to be a centralized repository of decisions(4.9) as proposed by Liang et al. (2010) but also that in a multi-disciplinary settings a distributed approach could be a solution.

Another aspect to facilitate reuse is to provide convincing incentives(4.15) for architects. For example, by fostering a company reuse culture, by offering low effort tool-support, or by offering rewards for reusing decisions or making decisions reusable, e.g., as part of the compensation system. *"Well in general it helps if you encourage people to do something if you reward them because that drives the motivation."*<1> *"You need to convince people that it is worth doing it this way. And you should automate it so that people that do not share this opinion can easily work with it."*<5>

c) Distributed Development A company-wide approach for reusing decisions in a distributed development environment must take into account that experts, teams and projects are located at multiple development sites, countries and timezones.

c.1) Documentation & Discovery

c.1.1) Information System Support A solution to improve the documentation and discovery of decisions in a distributed setting requires some information system support(4.10), which can act as a global AK repository(4.9) or allows to link local AK repositores. Notifications related to certain topics or projects (e.g. RSS feeds, Mailing lists) that are relevant for a particular architect are a desirable feature as it is difficult to stay up-to-date on relevant subjects. Farenhorst et al. investigated tool-support to improve knowledge sharing

Farenhorst et al. (2007b)

Besides recommendations that aim at codification of decisions, we also observed strategies at Oc that aim to improve decision-reuse and knowledge-sharing through personalization. Personalization remains important for knowledge-sharing due to the lower effort required compared to codification, as acknowledged by Capilla et al. (2015), although it has the inherent risk of knowledge vaporization. Moreover, personalization can be better combined with agile development practices that tend to reduce the amount of documentation in order to be able "to react to changes in its environment faster than the rate of these changes (Kruchten, 2013)". As shown in RQ1, frequent changes, coming from innovations in other disciplines, require a higher degree of agility in the software domain. This need for agility is one reason to emphasize personalized knowledge as codified knowledge might be outdated by the pace of these changes. We identified three strategies that are used at Océ: a) Facilitating social interactions, e.g, by encouraging frequent meetings, offering shared office space, and co-locating teams of the same project(4.13); b) Careful composition of project teams that ensures that certain knowledge is available(4.12) (cf. Section 4.3.1); c) Roaming experts that act as architecture consultants and join a project for a period of time to assist with a particular architectural problem.

4.7. RQ5 - cost and benefits of decision-reuse

RQ5 examined how architects perceive the costs and benefits of reusing decisions. The interviews show that architects perceive several benefits from reusing decisions. Facilitating decision-reuse helps to train inexperienced architects in decision-making(5.1) by exposing them to architecture decisions that were made by experienced architects, as illustrated by the following quotes: *"These [decision views], if I would have them available in the next project this would really help me."*<1>;*"Yes, that is true. I think good documents [to see how an experienced architect would do sth.] as a reference that helps."*<3> As discovered by van Heesch and Avgeriou (2010) inexperienced architects often do not follow a systematic reasoning process. Exposing them to reusable decisions guides them in how to reason about alternatives and how to make informed, well-balanced trade-offs. Moreover, decision-reuse promises to save time when making decisions(5.2) by reducing the effort associated with architectural analysis and synthesis. The interviewees stated that an extensive body of past decisions is particularly helpful in the beginning of a project and that it in general supports projects(5.12). *"You gain time to market without a doubt. Everything would immediately be available; what has been done in that area or with respect to these topics. That helps a lot."*<6> Even if a decision is only partially reusable (e.g., when the decision-context is not exactly the same), it provides inspiration for solving architectural problems(5.4) by providing examples of reasoning processes from other architects. From a product-line architect's perspective a systematic decision-reuse approaches saves resources and effort when aligning decisions across projects(5.3). And lastly, the availability of reusable decisions can facilitate architecture evaluations in several ways, especially when decision-based evaluation methods are being employed, such as Decision-Centric Architecture Reviews (DCAR) (van Heesch et al., 2014). In DCAR, architects can use a catalog of past decisions to elicit forces relevant for a review by examining which forces were previously considered relevant as well as to identify decisions relevant for the review (e.g., in case the system is part of a product family or based on a previous system). Moreover, architecture evaluations are facilitated by allowing the reviewer to compare the to-be-reviewed decision to similar decisions made in the past(5.6) and thus, to identify potentially omitted design options, decision-forces, arguments, as well as consequences and implications on the system.

On the other hand, architects also expressed their worries regarding decision-reuse. The interviewees showed a hesitance towards a systematic reuse of decisions because of they fear that it might decrease innovation(5.7). It was expressed that if decisions are being reused, the architects would invest less in finding original solutions and rather select well-known solutions from the past. *"Reuse is always important but it shouldn't block you in making changes."*<6> However, several studies already show that architects are biased by their experience when making decisions by favoring solutions they have considered before (van Vliet and Tang, 2016). Therefore, it could also be argued that with a systematic reuse approach the architect considers a larger set of potential solutions exceeding his own experience. Another concern expressed by the interviewees is that reuse might decrease the understanding of a problem(5.8) as the architects would spend less effort on the analysis of architectural problems. While this might be the case for off-the-shelf reusable solutions, it is questionable if a similar effect would occur when reusing decisions. As shown in the conceptual model, reusing a decision always requires to transfer knowledge from one system context to another system context, in which the architect evaluates and adapts a decision. In order to perform these steps, the architect needs a thorough understanding of the problem. Another concern articulated is that decisions can be reused in the wrong way or for the wrong purposes(5.13) as illustrated by the following quote: *"[I see that people use] the available technology in the wrong way and then develop something else for the way it was actually supposed to be used."*<2>.

Architecture decisions are considered an important corporate asset by one of the interviewed architects(5.5), underlining that architectural

knowledge reuse is a frequent and important organizational concern (Markus, 2001b). Nevertheless, it was argued that in product development the most important concern is to get the product to the market as fast as possible. This focus on immediate benefits complicates the introduction of systematic decision-reuse within an organization(5.9), as one interviewee said: *"It is all a matter of, do they spend time on it, documentation is very important and still it is very difficult to really get everything nicely documented. Because people are really extremely busy.It is not immediately important to the project success. So short term [goals] often prevail."*<6> The mandatory reuse preparation phase, in which decision need to be documented and distributed, does not have direct advantages for a project (w.r.t to reuse), but only for potential future projects. However, once a certain number of decisions has been prepared for reuse, the actual reuse of decisions has immediate benefits for a project. Investing in decision-reuse is considered a long-term investment that does not have an immediate return of value and, depending on whether a decision will be reused or not, possibly has no return at all. Still, the point has been raised that due to the increasing size and complexity of software-intensive system projects, not investing in reusable architectural knowledge is considered a risk for the company. This is in line with observations from (Hoorn et al. (2011)), who found that benefits of documenting decisions are only visible in the longer term; in the short term the most important thing is to meet the deadline, and move on to the next project.

5. Evaluation of validity

Our study is subject to limitations which can be categorized into construct validity, external validity, and reliability following Runeson et al. (2012). *Internal validity* is not a concern for this study because we did not examine causal relations (Runeson et al., 2012).

Construct validity indicates to what extent the operational measures that are studied really represent what was intended to be studied (Runeson et al., 2012). Interview studies often suffer from poorly worded or difficult to understand questions, which lead to ambiguous responses (Verner et al., 2009). Following the suggestion of Lethbridge et al. (2005), we conducted a pilot interview with a software architect that works in a comparable domain. The pilot interview showed that the questions are clear and understandable. Moreover, interview studies are subject to inaccuracies due to poor recall and reflexivity (Verner et al., 2009), the phenomenon that interviewees tend to answer questions in a socially desirable way or do not answer truthfully because they fear negative consequences, e.g., for their job (Runeson et al., 2012). We aimed to mitigate this risk by ensuring confidentiality of everything discussed during the interview, stating the voluntariness of participation and that there is no obligation to answer, as well as the fact that there are no right or wrong answers. We also asked open-ended questions and asked interviewees to motivate their answer in order to eliminate this bias.

A common limitation of observational studies is the Hawthorne effect, which refers to the fact that subjects tend to change their behavior simply because they are being observed (Lethbridge et al., 2005). We factored potential distortions of the results due to this effect into the analysis. Furthermore, we ensured to not bias the participants of the observed design meetings, for example, by exposing them to the research goals and questions.

Reliability is an aspect of validity that is concerned with the dependence of the data and the analysis on specific researchers (Runeson et al., 2012). The use of qualitative data is susceptible to several limitations of reliability due to the interpretative and subjective nature of qualitative data analysis. The biggest limitation of our study is the fact that the constant comparative method and observations were conducted by an individual researcher due to resource limitations. Therefore, the interpretation of the architect's responses and the observations depend on the researcher's belief system. Though, the study was exploratory and did not aim to validate or reject an a-priori hypothesis, we cannot ensure that ambiguous indicators were interpreted in a favorable way to the study's goal, e.g., the importance of reuse as a concern to architects might be overestimated. We mitigated this concern by regularly discussing intermediate results of the coding procedure and the final results between the authors of this study. Moreover, we performed an additional coding step with an unbiased researcher that re-coded all data in order to increase the reliability and quality of the coding. In addition, we used several data sources which allowed us to cross-check our findings (Creswell and Miller, 2000). For example, the observation of architectural design meetings over a period of one year gave us in-depth insights and allowed us to put the interviews into perspective, which lowered the risk of being biased by one person (Runeson and Höst, 2009).

Moreover, we allow an assessment of validity and reliability by being explicit and open about all research decisions and activities as suggested by Creswell and Miller (2000). We also appended the interview guide and the final code system to increase the level of transparency as suggested by Hiles (2008). Transparency is an important concern for establishing the quality of qualitative research by being explicit, clear, and open about the production, analysis and data of the study (Hiles, 2008).

External validity is concerned with the extent to which findings and conclusions can be transferred to other contexts and to which extent they help to derive useful theories. As critics sometimes state (Yin, 2008), findings of case studies are limited in the sense that they are not necessarily fully valid in a broader context. Because this study is qualitative, the results are not statistically representative and due to the single-case design statical generalization is not possible. Instead, we use analytical generalization to discuss the findings for our research questions and their generalizability (Runeson and Höst, 2009; Yin, 2008). Another limitation of the study is the geographical location of the company and the nationality of the interviewed architects. Although Océ operates globally and has a internationally diverse workforce, we conducted our case study at the R&D site in the Netherlands and all interviewees were Dutch. Consequently, in order to eliminate any factors related to country or culture, the study needs to be replicated in a different setting.

In RQ1, we present the current state of decision-reuse in Océ including company-specific approaches and tools. Consequently, neither the description of the current state of decision-reuse nor the discussed approaches are generalizable to other companies. Nevertheless, the absence of a systematic decision documentation approach and the resulting difficulties to discover decisions are typical problems in the software industry as pointed out by (Capilla et al., 2015). In addition, a key finding of RQ1 is the tendency of architects to reuse decisions in an ad-hoc manner, immediately converting decision from one system-specific context to another system-specific context. This finding is not company-specific as it deals with the fundamental way of working of architects. The conceptual model of decision-reuse that we present is generic enough to be independent of case-specific factors and therefore, can be used to map existing decision-reuse approaches. The stakeholders identified in RQ2 were described in a abstract manner independent from the roles we observed in Oce. However, we acknowledge that the profiles are heterogenous and have different meanings in different companies (e.g., software architect). Nevertheless, the presented classification can be used to map our stakeholders to actual stakeholders in other companies. The concerns presented for RQ3 are applicable to the wider domain of embedded systems engineering. This is because the identified concerns are generic enough and do not include case-specific factors. Several of the concerns were identified in previous studies (van Heesch et al., 2012a; van Heesch et al., 2012b), which further supports their generalizability. An exception to this are the concerns C15 ("Which model was used to evaluate alternatives for decision D?") and C16 ("What are open or upcoming problems P in architecture R?"). C15 is specific to organizations that employ the model-based system engineering methodology in which domain models are used as a means for information exchange and experimental simulation. C16 only applies to situations in which a reference architecture is shared by several simultaneously developed products. The generalizability of the findings of RQ4 (ideal approach) and RQ5 (cost and benefits of decision-reuse) are limited by the degree to which the characteristics of an ideal approach as well as the benefits and limitations of decision-reuse

depend on the case-specific experiences of the subjects.

6. Conclusions and future work

The main goal of the study was to investigate the decision-reuse in software-intensive system projects. The study set out to describe the current state of decision-reuse, to identify the responsible stakeholders and their reuse-related concerns, to characterize the ideal approach for reusing decisions and to analyze how stakeholders assess the costs and benefits of decision-reuse.

As a first step, we introduced a conceptual model for decision-reuse, and based on this model, we have illustrated the current state of decision-reuse at Océ. The results demonstrate that reuse is an important concern and architects frequently reuse decisions when architecting. However, due to the lack of explicit decision documentation and inadequate possibilities to discover decisions, architects are confined to decisions they already know or have heard about. Overall, the evidence suggests that a lot of architectural knowledge embodied in decisions is not utilized by architects and has to be re-acquired. Improving the documentation and discoverability of decisions holds a large potential to increase the amount of decision-reuse and thus, to spread architectural knowledge within a company, improving the architecting process.

Despite its exploratory nature, this study offers insight into how architects reuse decisions. A major finding was that architects do not abstract decisions into a system-generic form before they reuse them but instead do an ad-hoc conversion. Likewise, we found that organizations do not spend resources on abstracting architectural knowledge into a system-/generic reusable form. Taken together, the findings suggest that abstracting decisions into a system-generic form is often not necessary and might not yield a positive return on investment, especially if there is not a high degree of decision reuse.

Moreover, our study identified important reuse-related concerns of stakeholders. Notwithstanding the fact that we only observed a single company, the identified concerns offer valuable insights to improve architectural documentation with regards to decision-reuse. These concerns should inform the design of any decision reuse approach.

With regards to an ideal decision-reuse approach, we found that the reuse activities of documentation and discovery have a large potential for improvement and dedicated tool-support. Nevertheless, our study also highlights the importance of hybrid approaches that supports codification as well as personalization of knowledge. Furthermore, our study emphasizes the importance of architectural knowledge-sharing, especially, with respect to reusing decisions across projects and sites.

A final significant finding, is that decision documentation is not only important for system understanding or in the context of architecture reviews but also to support architects in upcoming projects.

This research will serve as a base for future studies on decision-reuse and enhances our understanding of the role of architecture decisions in the architecting process. Moreover, the findings further support the importance of managing architectural knowledge and especially, of documenting decision rationale and alternative solutions.

This study further raises questions in need of further investigation including: what is an optimal approach for reusing decisions in industrial projects; how to determine in advance which decisions are candidates for reuse; and what documentation is optimal to balance the effort of documenting decisions compared to the effort of retrieving decisions? Moreover, we need more empirical evidence to evaluate the cost, benefits and implications of decision-reuse.

Acknowledgment

We would like to thank all participants of the case study and Michael Stal for participating in the pilot study. We thank our colleague Chen Yang for his valuable support in the data analysis of this study. This research has been sponsored by the ITEA2 project 11013 PROMES.

Appendix A

A1. Description of stakeholder concerns

C1 What were key decisions in project X (for subsystem S)? Identifying the key architecture decisions that were taken for a particular system or subsystem provide a good starting point for reusing AK that already exists within the company. For example, architects working on system S' are looking at decisions in system S because it resembles S', it is in the same family, or it is a followup project. Answering this concern helps architects to discover decisions that can be reused or provide inspiration that supports the architecting process.

C2 Which decisions $D_i$addressed problem P? The interviews showed that architects are interested to learn how other projects solved an architectural problem, which is similar to the one they are working on. Especially, since a set of architectural problems reappear from project to project as shown in RQ1, architects are interested in applying similar solutions. *"Most of the time, I look once or twice at previous solutions. For example, double storage of counters, I want to know how [ProjectA] solved this"*<3>

C3 Which decisions $D_i$considered to use alternative A? Being able to discover decisions in which a particular alternative has been considered provides an important basis for reusing architectural knowledge. From these decisions, architects increase their understanding about an alternative and learn about the arguments that lead to the selection or rejection of it. When using an alternative, architects want to know if the alternative has been used in previous projects and if the choice of the alternative had a positive or negative impact on the project in hindsight, for example, were there any unforeseen negative or positive consequences?

C4 Which projects PR_ishare problem P? Aligning products from a product portfolio point-of-view promises to save resources and effort. Therefore, knowing which projects share similar architectural problems is an important concern that allows the identification of potential synergies across projects and enables cross-project decision-reuse. *"So most of the times when you want to improve, you discuss with the projects that are developing to see if there is some synergy, if they also want to achieve something in that direction? And then you cooperate and you find the common functionality."*<5> It is in the interest of the company to align the architectures of different projects as much as possible in order to facilitate reuse and shorten development time. Therefore from a portfolio point of view it is important to be aware of solved or upcoming problems that affect multiple projects. *"Because then we have huge projects, they all have their own choices and implementations and designs and it is all different. in the end it will be a disaster. I really believe that we should have an alignment in architecture choices across projects, to save resources in the end."*<6>

C5 What decision D_1 has been reused in the decision-making process of decision D_2**?**

This concern asks which decisions were reused by the architects when making a particular decision. The need for this concerns is mainly related to establishing traceability between decisions for impact analysis and system understanding. *"You want to change something which you dont know why it is, so you have to go back into the first design to understand why it is that way. Maybe because it was reused or maybe because it was the easiest thing to do or maybe it has a functional reason."*<5>

C6 Which stakeholders SH_iwere involved in decision D?

It is widely acknowledged that a large part of knowledge cannot be easily documented (Farenhorst and de Boer, 2009), so called tacit knowledge.

Table A1
Frequency and distribution of codes per data source.

Code	Minutes†	Documents†	Interview 1	Interview 2	Interview 3	Interview 4	Interview 5	Interview 6
(1.1) Decision-reuse in ind. practice				•	•		•	•
(1.2) Implicit vs. explicit reuse			●	•	●	●	●	●
(1.3) Frequency of reuse			●		•	•	•	•
(1.4) Motivation for reuse		•	•	•	•		•	•
(1.5) Recurrence of decision problems			•	•	•	•	•	•
(1.6) Recurring problems solved in reuse components		•	•	•			•	•
(1.7) Documentation			•	●	●	●	●	●
(1.8) Distribution			•	•	●		•	•
(1.9) Discovery			•	●	•	•	•	•
(1.10) Retrieval	•		•	•	●	•	•	•
(1.11) Evaluation	•		•	•	•		•	•
(1.12) Adaptation			•	•	•			•
(1.13) System-generic AK is rarely reused			•	•		•	•	•
(1.14) Problem: Effort of finding/processing documentation			•	•	•		•	
(1.15) Problem: Lack of communication / knowledge exchange								•
(1.16) Problem: Decision Documents lack relevant information			•	•	•	•	•	•
(1.17) Problem: Multi-disciplinary cooperation				•				
(2.1) Architect			•	•				•
(2.2) Reviewer			•	•				•
(2.3) Product-line Architect							•	•
3.1) What were key decisions in project X (for subsystem S?)		•	•				•	

(*continued on next page*)

Table A1 (*continued*)

Code	M†	D†	1	2	3	4	5	6
3.2) Which decisions D_i addressed problem P?	•	•	•	•	•			•
3.3) Which decisions D_i considered to use alternative A?		•	•			•	•	•
3.4) Which projects PR_i share problem P?	•	•				•	•	•
3.5) Which decision D_1 has been reused in the decision-making process of decision D_2?							•	
3.6) Which stakeholder SH_i were involved in decision D		•	•	•	•			
3.7) What is the rationale behind decision D?	•	•	•	•	•		•	•
3.8) Which alternatives A_i were considered for decision D?		•	•	•			•	
3.9) What forces F_i influenced decision D?	•	•	•	•	•	•	•	•
3.10) Which trade-offs T_i were made for decision D?	•	•	•	•	•	•	•	•
3.11) What is the high-level system context of decision D?		•			•			
3.12) Which model M was used for decision D?			•			•	•	
(4.1) Focus on the important AK (e.g. rationale)			•	•	•		•	•
(4.2) Overview of decisions			•	•				
(4.3) Reduce documentation maintenance effort			•	•			•	
(4.4) Incremental Documentation				•				
(4.5) Reference (trace) documents			•	•	•	•	•	•
(4.6) Guidance			•		•			•
(4.7) Multi-disciplinary tool availability			•	•			•	
(4.8) Flexibility of documentation			•					
(4.9) Collaborative central repository of AK and AE			•				•	•
(4.10) Information System Support			•	•			•	•
(4.11) Powerful Search			•	•	•	•	•	
(4.12) Assign experienced members on a project				•				•

(*continued on next page*)

Table A1 (*continued*)

Code	M†	D†	1	2	3	4	5	6
(4.13) Facilitate interactions between architects			●		●	●		●
(4.14) Explicit documentation reviews			●					
(4.15) Providing incentives			●	●			●	●
(5.1) Benefit:Training junior architects			●			●		
(5.2) Benefit: Reuse saves time			●		●	●	●	●
(5.3) Benefit:Align Decisions across project to save resources					●	●		●
(5.4) Benefit:Reuse decisions as inspiration for problem-solvin			●	●		●		
(5.5) Benefit: Decisions are corporate assests								●
(5.6) Benefit:Comparing decisions for architecture evaluation			●					
(5.7) Risk:Reuse decreases innovation			●	●				●
(5.8) Risk:Reuse knowledge decreases understanding of a problem			●					
(5.9) Cost: Short-term benefits prevail			●				●	●
(5.10) Cost:Effort of documentation			●	●	●			●
(5.11) Cost:Willingness to invest in reuse			●		●	●		●
(5.12) Benefit: Provide guidance (general proj. support)			●					●
(5.13) Risk: Mis-reuse				●				

† Documentation analysis of the redesign project

Therefore, a complete documentation is often not feasible nor desirable as it requires too much effort. Therefore, being able to identify the stakeholders that were involved in the decision is important in order to fill in the gaps left by the documentation. Being able to answer this concern is necessary when retrieving decisions.The following line from the redesign project exemplifies the concern: *"In case more information is needed please contact [PERSON1] or [PERSON1]"*$^{<M>}$. Also several architectural views were created in the redesign project that indicate the responsible architects of components.

C7 What is the rationale behind decision D?

The interview showed that access to design rationale is considered very important when reusing architecture decisions. As one if the interviewee stated: "The insight why a decisions was taken is very important for reuse. Architects should understand where and why certain choices were made in projects in the past. If that is lost that really hampers success in the future.". When reusing decisions it is important to understand the reasoning behind a decision, not only for selecting a certain alternative but also why other alternatives were rejected.

C8 Which alternatives $A_i$were considered for decision D? The alternatives considered in previous decisions sometimes present reusable solutions. If the alternative cannot be reused off-the-shelf, it still might provide a source for inspiration. This does not only apply to the selected alternative but is also relevant for rejected ones as illustrated by the following statement: "We had to understand why they didn't do it and we found out because [one of the considered solution] did not had a [particular] capability [at that time]". In contrast to C3, which aims at the evaluation of a particular alternative, C8 aims at inspiration (e.g., identifying potential alternatives).

C9 What forces $F_i$influenced decision D? When reusing a decision it is important to be aware of the forces that influenced the architects when they made the decision, since they determine the context in which the original decision was made. The selection and prioritization of forces explains why a certain alternative was favored over another solution. A force is a broad concept, capturing anything that impacts the architect, such as, requirements, architectural concerns, expertise of the team, as well as business and projects constraints. Especially forces that are not directly related to the system (e.g., requirements, architectural concerns) are often hard to find in the documentation, although, they fundamentally shape the decision. As one interviewee reported: *""Sometimes a decision has nothing to do with architecture. For example, no one wanted to make a particular component. However, there was one team that had some spare time and agreed to take it over. So you get some features or components in a subsystem that you would have expected in a different subsystem. When you dive into the architecture you could experience that this is a bit strange. Why was it solved that way?""*$^{<3>}$

C10 Which trade-offs T_i were made for decision D?

The interviews showed that architects frequently make trade-offs, which need to be made explicit in the documentation. *"So we have to reconsider the product and we have to do cost-price action and the cost-price action hits the architecture. Maybe I can reduce the cost by half but then the productivity of the printer goes down."*<5> When reusing decisions it is important that the reusing architect is aware of potential trade-offs that were made when making the decision. In practice the considered alternatives are not able to address all forces equally well and therefore, do not perfectly solve an architectural problem. Instead the architect must decide which of the alternatives optimally balances the relevant forces and which trade-offs are acceptable. For example, the architect might trade-off productivity in order to lower the cost-price or to deliberately accumulate technical debt in order to reduce the time-to-market.

C11 What is the high-level system context of decision D?

When looking for existing decisions the reader might not be familiar with the system context of the decision (e.g., unfamiliar terminology). Therefore, it is important to provide a brief description that explains the system context for a particular decision. This description might show where the decision problem is located within the system by showing related components or steps within a sequence. *"When I want to know on a subject I sometimes only need the problem around. For example, when you have the failure correction and this mechanism. I wanted to know more about the concept. I had troubles to find out about the concept but I could find a lot of documents on how to solve this or that, but where is now the concept? That is sometimes not written down and that could help us."*<3>

C12 Which model M was used for decision D?

Océ employs the model-based system engineering methodology, in which domain models are used as a means for information exchange and experimental simulation.Therefore, the decision for a certain alternative is often supported by a model, which evaluates aspects of the solution, such as a performance model. *"If you talk about big data handling, there is a lot of things you can do with a really simple model. If you have resolution, you have pages per minute. If you stuff that into the model you can quite quickly come up with a data rate and see if it fits on a disk or if you need something [else]."*<4> These models often embed a high degree of architectural knowledge and represent reusable entities by themselves.

A2. Code system

Table A2
Mapping of Interview Questions (cf. Appendix A.3.1) to Research Questions.

Interview Questions	RQ1 - Status Quo	RQ2 - Stakeholders	RQ3- Concerns	RQ4 - Ideal	RQ5 - Benefits/Costs
1a	•				
1b	•				
1c	•	•	•		
1d	•				
2a	•		•		
2b	•		•		
2c	•		•		
2d	•		•		
2e	•				
3a	•		•		
3b	•		•		
3c	•		•		
3d	•		•		
4a		•	•	•	
4b				•	•
4c			•		•
4d					•
4e		•			

A3. Interview guides

The interview guides were defined upfront by the three authors in an iterative process. The interview guide was split into four blocks that focused on a particular area of discussion. This first block aimed at understanding the current architecting and decision-making process. It provides us with background information about the interviewees, their concrete role and their understanding of software architecture. Understanding the background of the interviewee is important to put other answers into perspective (e.g., the interviewees understanding of architecture decisions when asking about reusing decisions). The second block of questions identifies current reuse of architectural knowledge. It establishes the current state of decision-reuse and helps to identify which information about a decision is most frequently being reused. It primarily contributes to answering RQ1. The third block of questions discusses recurring architecture problems, solutions, and decisions in order to identify their decision-reuse potential by examining to what extent they recur in the architecting process and *how* they can be reused. Moreover, this block aims to identify which information is most relevant for reuse. This third block of questions helps in answering RQ1,RQ2 and RQ3. The last block of questions aims at identifying concerns related to decision-reuse, the ideal approach for decision-reuse and the cost and benefits of reusing of decisions; thus, it helps to answer RQ2, RQ3, RQ4 and RQ5. The interview guide does not equally cover the research questions. More questions are devoted to RQ1 and RQ3 compared to RQ2, RQ4, and RQ5. The reason is that the research questions vary in scope and complexity. For example, RQ1 asks about the status quo of reuse while RQ3 aims to elicit a list of stakeholders involved in decision-reuse.

- Explain about the study, what we are looking for, and how they will benefits from the results.
- We want to explore to what extent architecture decisions can be reused across projects. Therefore, your views and experiences on architecture decision-making in the context of large-scale embedded system projects.
- Reusing architecture decisions from previous projects (e.g., decompositions of systems aspects, selection of technologies or components, or the application of patterns)
  - promises to support the architect,
  - to reduce the effort associated with architecture analysis, synthesis and evaluation
  - and to increase the quality of decisions.

  The information that you give us in the interview will be used
  (a) to gain a better understanding of the current situation of decision-reuse in industry;
  (b) to identify prerequisites and requirements of a systematic decision-reuse approach that meets the needs of software architects;
  (c) and to identify stakeholders relevant to decision-reuse, their concerns as well as incentives that motivate or inhibitors that discourage decision-reuse in industry.
- There are no right or wrong answers to this. We are keen to gain a wide variety of opinions.

A3.1. Interview guide for software architects

Block 1:(≈ 10min) *Understanding the current architecting and decision-making process*

1a) Could you describe your role as a software architect?
- What are your typical tasks, responsibilities and duties?
- What is not included in your responsibilities?
- How long do you usually stay in a project?

1b) Please describe the typical software architecture process in your projects?
- How do you usually do architectural analysis, synthesis and evaluation?
- In which phases do you interact with other disciplines?

1c) Please describe the process that you follow when making architecture decisions (even if it is not structured, what are typical steps)?
- How do you identify the architectural issues that require a decision?
- How do you identify the forces (requirements) that primarily influence the decision outcome?
- How do you identify and evaluate architectural alternatives for a problem?
- How many alternatives do you usually identify?
- Who is usually involved in the decision-making process and how do you reach consensus?
- Who takes the responsibility for taking the decision?
- How far do you document decisions and which aspects?

1d) In the beginning of a project, to what extent are you familiar with the problem domain?
- Problem-domain: Understanding and awareness of requirements and how they influence the architecting process
- When are you able to sufficiently estimate/anticipate the problem that will/might occur in a project?

Block 2:(≈ 10min) *Identifying current reuse of architectural knowledge*

The second block of questions aims to establish the current state of decisions reuse and to identify the decision reuse potential by examining to what extent architectural issues, alternatives and decisions reoccur in the architecting process and how those can be reused.

2a) To what extent do you consult documentation from past projects when making architecture decisions?
- Could you give a rough estimate, e.g., in x out of 10 decisions?
- What information are you looking for?

2b) To what extent do you consult software architects from other projects (finished and ongoing) when making decisions? (RQ1)
- Could you give a brief estimate, e.g., in x out of 10 decisions?
- What information are you asking for?

2c) To which extent do you consult other information sources when making decisions, e.g., internet, intranet, blogs, etc.?

2d) To what extent do you consider patterns and technologies (libraries, framework, etc.) when architecting? (RQ1)

2e) How far are your decisions influenced by your experience, for example, by decisions that worked well in the past ?

- Could you give a brief estimate, e.g., in x out of 10 decisions?

Block 3:(≈ 10min) *Identifying reoccurring problems, solutions, and decisions*

3a) Are there architectural issues or decision problems that occur in the same or a similar way, in every project?. These decisions can be related to the system's decomposition, properties, trade-offs, development process, etc.
- How far do those decision topics have the same or a similar outcome?

3b) To what extent are forces, such as functional requirements and non-functional requirements, recurring form project to project?
- Are the requirements to some extent similar and only the concrete values change?
- Are there typical forces that you consider when making a decision?

3c) To what extent are stakeholders and their concerns similar from project to project?

3d) How often do you think that an architectural solution has been already considered or evaluated in a previous project?
- To what extent do you see the potential to reuse solutions from previous projects?
- To what extent would you trust such solutions?
- If you could not reuse the solution, would you see any advantage or possibility reusing the evaluation of the alternative (if present) , e.g., performance models or pugh-matrices? If yes, how?

Block 4:(≈ 15min) *Identify concerns related to decision-reuse, ideal approach of decision-reuse and the cost benefits of reusing of decisions.*

- For the next questions, please imagine a typical project in which your are the responsible Software Architect.
- I would like to ask you to imagine an approach for reusing decisions from previous projects.
- And by decisions I am referring to the complete ecosystem of a decisions, which includes the problem, associated requirements, interested stakeholders, the rationale, considered alternatives, evaluations, you name it.
- This approach should be the best possible solution in your eyes.

(4a) Could you briefly describe the approach that came to your mind?
- How would you use the approach during your daily work as an architect?
- What kind of capabilities and features would the approach support?
- Which information would you be most interested in for reuse?
- What kind of use cases for decision reuse came to your mind?
- Who would use this approach?
- How would you envision tool-support for this approach?

(4b) What are the benefits and costs of this ideal decision-reuse approach?
- What would be the greatest incentive (or motivation) of reusing decisions?
- What would be the largest inhibitor of reusing decisions?

(4c) To what extent would the ideal approach support you when you are first assigned the role of a software architect in a new project?

(4d) How much effort would and could you spend on making your decisions reusable for other architects?

(4e) What kind of incentives would you propose to convince your colleagues to make their decisions reusable?

(4f) Who else would benefit from reusable decisions?

A3.2. Interview guide for software architects (Management role)

Block 1:(≈ 10min) *Understanding the current architecting and decision-making process*

(1a) Could you describe your role and responsibilities in Oce?
- What is not included in your responsibilities?

(1b) In how far are you concerned with software architecture as part of your job?

(1c) Could you describe the role of a software architect in your department?
- What are their typical tasks, responsibilities and duties?
- What is not included in their responsibilities?
- Are there significant differences compared to SA in other departments?

(1d) Please describe the ideal software architecture process in your department?
- How is the architecting process managed?
- How are architects supported?
- How do architects and managers interact and in what topics?

(1e) What is the typical process that they should follow when making architecture decisions?
- Are managers involved in making architecture decisions and what are typical concerns?
- Who takes the responsible for taking the decision?
- In how far do they document decisions and which aspects?
- Is documentation of architecture decisions an important issue for management?

(1f) How important do you consider reuse of architecture decisions (or knowledge in general)?

(1g) Would you consider reusable decisions as corporate assets?

Block 2:(≈ 10min) Identifying current reuse of architectural knowledge

The second block of questions aims to establish the current state of decisions reuse and to identify the decision reuse potential by examining to what extent architectural issues, alternatives and decisions reoccur in the architecting process and how those can be reused.

(2a) To what extent do architects in your department reuse architectural knowledge?
(2b) In how far do you facilitate reuse of architecture knowledge in your department?
- What tools and approaches do you provide to support reuse?
- Which sources for reusable architectural knowledge do you provide?
  - Access to existing documentation of past project
  - Reference Architectures or pattern catalogues
  - Other information sources (internet, intranet, blogs, etc.)
- Are there knowledge bases in Oce that cut across departments or individual projects?

(2c) Can you quantify the benefits and costs of reusing architecture knowledge?

Block 3:(≈ 10min) Identifying reoccurring problems, solutions, and decisions
The following set of questions aims to identify the potential of decision reuse

(3a) Are there architectural issues or decision problems that occur in the same or a similar way, in every project? These decisions can be related to the system's decomposition, properties, trade-offs, development process, etc.
- In how far do those decision topics have the same or a similar outcome?

(3b) To what extent do you see the potential to reuse solutions from previous projects?
- To what extent would you trust such solutions?

(3c) If you could not reuse the solution, would you see any advantage or possibility reusing the evaluation of the alternative (if present) , e.g., performance models or pugh-matrices? If yes, how?
(3d) To what extent are forces, such as functional requirements and non-functional requirements, recurring form project to project?
- Are the requirements to some extent similar and only the concrete values change?
- Are there typical forces that you consider when making a decision?

(3e) To what extent are stakeholders and their concerns similar from project to project?

Block 4:(≈ 15min) Identify concerns related to decision-reuse, ideal approach of decision-reuse and the cost benefits of reusing of decisions.

(4a) How would you assess the improvement potential for reusing architecture decisions?
(4b) How can decision reuse be increased in your department?
- How could management (better) support reuse?
- What information would you be most interested in for reuse?

(4c) What would be the greatest incentive (or motivation) for reusing decisions?
(4d) What would be the largest inhibitor for reusing decisions?
(4e) How much effort would you allow architects to spend on making decisions reusable for other architects?
(4f) What kind of arguments would you propose to convince architects to make their decisions reusable?

A4. Mapping of interview questions to research questions

A5. Participant invitation letter

Dear LASTNAME,

We would kindly invite you to participate in a case study about reusing software architecture decisions at Océ.

As part of this study, we would like to invite you to a one-off interview with a researcher to tell us about your views and experiences on architecture decision-making in the context of large-scale embedded system projects. In particular, we want to explore to what extent architecture decisions can be reused across projects. Reusing architecture decisions from previous projects (e.g., decompositions of systems aspects, selection of technologies or components, or the application of patterns) promises to support the architect, to reduce the effort associated with architecture analysis, synthesis and evaluation and to increase the quality of decisions. The information that you give us in the interview will be used

(a) to gain a better understanding of the current situation of decision-reuse in industry;
(b) to identify prerequisites and requirements of a systematic decision-reuse approach that meets the needs of software architects;
(c) and to identify stakeholders relevant to decision-reuse, their concerns as well as incentives that motivate or inhibitors that discourage decision-reuse in industry.

There are no right or wrong answers to this we are keen to gain a wide variety of opinions. If you are interested in taking part in this study, please send us a list of time slots that would work for you. The interview will approximately take 45 min.

The study is part of the ITEA2 PROMES research project, and is conducted by the Software Engineering and Architecture group of the University of Groningen in collaboration with Oce: Christian Manteuffel , Paris Avgeriou, and Roelof Hamberg. If you have any questions about the study then please do contact us.

Thank you very much for reading this letter,

Yours sincerely,

Christian Manteuffel, Paris Avgeriou, Roelof Hamberg

References

Adolph, S., Hall, W., Kruchten, P., 2011. Using grounded theory to study the experience of software development. Empir. Softw. Eng. 16 (4), 487–513. http://dx.doi.org/10.1007/s10664-010-9152-6.

Anvaari, M., Zimmermann, O., 2014. Towards reusing architectural knowledge as design guides. Proceedings of the SEKE 2014. pp. 181–186.

Babar, M., Gorton, I., Jeffery, R., 2005. Toward a Framework for Capturing and Using Architecture Design Knowledge. Technical Report June. University of New South Wales.

Basili, V.R., 1995. Applying the Goal/Question/Metric Paradigm in the Experience Factory. In: Norman, E.F., Robin, W., Whitty, Y.I. (Eds.), Software Quality Assurance and Measurement:Worldwide Perspective. International Thomson Computer Press, pp. 21–44.

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A., 2015. 10 Years of software architecture knowledge management: practice and future. J. Syst. Softw. http://dx.doi.org/10.1016/j.jss.2015.08.054.

Creswell, J.W., Miller, D.L., 2000. Determining validity in qualitative inquiry. Theory Pract. 39 (3), 124–130. http://dx.doi.org/10.1207/s15430421tip3903_2.

Dingsøyr, T., van Vliet, H., 2009. Introduction to software architecture and knowledge management BT - software architecture knowledge management. In: Babar, M.A., Dingsøyr, T., Lago, P., van Vliet, H. (Eds.), Software Architecture Knowledge Management. Springer-Verlag, Berlin Heidelberg, pp. 1–17.

Falessi, D., Briand, L.C., Cantone, G., Capilla, R., Kruchten, P., 2013. The value of design rationale information. ACM Trans. Softw. Eng. Method 22 (3), 1.

Farenhorst, R., de Boer, R.C., 2009. Knowledge management in software architecture: state of the art. In: Ali Babar, M., Dingsøyr, T., Lago, P., van Vliet, H. (Eds.), Sotware Architecture Knowledge Management. Springer, Berlin, Heidelberg.

Farenhorst, R., Lago, P., van Vliet, H., 2007. Eagle: effective tool support for sharing architectural knowledge. Int. J. Coop. Inf. Syst. 16 (03n04), 413–437. http://dx.doi.org/10.1142/S0218843007001706.

Farenhorst, R., Lago, P., van Vliet, H., 2007. Effective tool support for architectural knowledge sharing. In: Oquendo, F. (Ed.), Proceedings of Software Architecture: First European Conference, ECSA 2007 Aranjuez, Spain, September 24–26, 2007. Springer, Berlin, Heidelberg, pp. 123–138. http://dx.doi.org/10.1007/978-3-540-75132-8_11.

Farenhorst, R., van Vliet, H., 2008. Experiences with a wiki to support architectural knowledge sharing. 3rd Workshop on Wikis for Software Engineering (Wikis4SE).

Farenhorst, R., van Vliet, H., 2009. Understanding how to support architects in sharing knowledge. Sharing and Reusing Architectural Knowledge, 2009. SHARK '09. ICSE Workshop on 17–24.

van Heesch, U., Avgeriou, P., 2010. Naive architecting - understanding the reasoning process of students. In: Babar, M.A., Gorton, I. (Eds.), Proceedings of the 9th Working IEEE/IFIP Conference on 4th European Conference Software Architecture, ECSA 2010, Copenhagen, Denmark, August 23-26, 2010. 6285. Springer, Berlin, Heidelberg, pp. 24–37. http://dx.doi.org/10.1007/978-3-642-15114-9_5.

van Heesch, U., Avgeriou, P., 2011. Mature architecting - a survey about the reasoning process of professional architects. Proceedings of the Software Architecture (WICSA), 2011.IEEE. pp. 260–269.

van Heesch, U., Avgeriou, P., Hilliard, R., 2012. A documentation framework for architecture decisions. J. Syst. Softw. 85 (4), 795–820. http://dx.doi.org/10.1016/j.jss.2011.10.017.

van Heesch, U., Avgeriou, P., Hilliard, R., 2012. Forces on architecture decisions - a viewpoint. Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012. pp. 101–110.

van Heesch, U., Eloranta, V.P., Avgeriou, P., Koskimies, K., Harrison, N., 2014. Decision-Centric architecture reviews. IEEE Softw. 31 (1), 69–76.

Hiles, D.R., 2008. Transparency. In: Given, L.M. (Ed.), The Sage Encyclopedia of Qualitative Research Methods. SAGE Publications, Inc., pp. 891–893. http://dx.doi.org/10.4135/9781412963909.

Hofmeister, C., Kruchten, P., Nord, R.L., Obbink, H., Ran, A., America, P., 2007. A general model of software architecture design derived from five industrial approaches. J. Syst. Softw. 80 (1), 106–126.

Hoorn, J.F., Farenhorst, R., Lago, P., van Vliet, H., 2011. The lonesome architect. J. Syst. Softw. 84 (9), 1424–1435. http://dx.doi.org/10.1016/j.jss.2010.11.909.

ISO/IEC/IEEE, 2011. Systems and software engineering–Architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) 1–46. http://dx.doi.org/10.1109/ieeestd.2011.6129467.

Jansen, A., Bosch, J., Avgeriou, P., 2008. Documenting after the fact: recovering architectural design decisions. J. Syst. Softw. 81 (4), 536–557.

Kruchten, P., 2008. What do software architects really do? J. Syst. Softw. 81 (12), 2413–2416. http://dx.doi.org/10.1016/j.jss.2008.08.025.

Kruchten, P., 2013. Contextualizing agile software development. J. Softw.: Evol. Process 25 (4), 351–361. http://dx.doi.org/10.1002/smr.572.

Lethbridge, T.C., Sim, S.E., Singer, J., 2005. Studying software engineers: data collection techniques for software field studies. Empir. Softw. Eng. 10 (3), 311–341. http://dx.doi.org/10.1007/s10664-005-1290-x.

Lewis, G.a., Lago, P., Avgeriou, P., 2016. A decision model for cyber-foraging systems. Proceedings of the 13th Working IFIP/IEEE Conference on Software Architecture. http://dx.doi.org/10.1109/WICSA.2016.38.

Liang, P., Jansen, A., Avgeriou, P., 2010. Collaborative Software Architecting through Knowledge Sharing. In: Mistrík, I., Grundy, J., Hoek, A., Whitehead, J. (Eds.), Collaborative Software Engineering. Springer, Berlin, Heidelberg, pp. 343–367.

Lytra, I., Tran, H., Zdun, U., 2013. Supporting consistency between architectural design decisions and component models through reusable architectural knowledge transformations. In: Drira, K. (Ed.), Software Architecture. ECSA 2013. Lecture Notes in Computer Science, vol 7957. Springer, Berlin, Heidelberg, pp. 224–239. http://dx.doi.org/10.1007/978-3-642-39031-9_20.

MacLean, A., Young, R.M., Bellotti, V.M.E., Moran, T.P., 1991. Questions, options, and criteria: elements of design space analysis. Hum.-Comput. Interact. 6 (3), 201–250.

Manteuffel, C., Tofan, D., Avgeriou, P., Koziolek, H., Goldschmidt, T., 2016. Decision architect - A decision documentation tool for industry. J. Syst. Softw. 112, 181–198. http://dx.doi.org/10.1016/j.jss.2015.10.034.

Manteuffel, C., Tofan, D., Koziolek, H., Goldschmidt, T., Avgeriou, P., 2014. Industrial implementation of a documentation framework for architectural decisions. Proceedings of the IEEE/IFIP Conference on Software Architecture (WICSA), 2014. pp. 225–234.

Markus, L.M., 2001. Toward a theory of knowledge reuse: types of knowledge reuse situations and factors in reuse success. J. Manag. Inf. Syst. 18 (1), 57–93. Article

Markus, L.M., 2001. Toward a theory of knowledge reuse: types of knowledge reuse situations and factors in reuse success. J. Manag. Inf. Syst. 18 (1), 57–93. Article

Miles, M.B., Huberman, A.M., Saldaña, J., 2013. Qualitative Data Analysis: A Methods Sourcebook. SAGE Publications, pp. 69–104.

Muller, G., 2004. CAFCR: a multi-view Method for embedded systems architecting : balancing genericity and specificity. University of Delft, Delft. (Ph.D. thesis)

Nowak, M., Pautasso, C., 2010. Architectural decision modeling with reuse : challenges and opportunities. Proceedings of the SHARK '10 of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge. pp. 13–20. http://dx.doi.org/10.1145/1833335.1833338.

Nowak, M., Pautasso, C., 2013. Team situational awareness and architectural decision making with the software architecture warehouse. Proceedings of the 7th European Conference, ECSA 2013, Montpellier, France, July 1–5, 2013. Springer, Berlin Heidelberg, pp. 146–161. http://dx.doi.org/10.1007/978-3-642-39031-9_13.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164. http://dx.doi.org/10.1007/s10664-008-9102-8.

Runeson, P., Höst, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering. John Wiley & Sons, Inc., Hoboken, NJ, USA. http://dx.doi.org/10.1002/9781118181034.

Schuster, N., Zimmermann, O., Pautasso, C., 2007. ADkwik: Web 2.0 collaboration system for architectural decision engineering. Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering, SEKE, 2007. pp. 255–260.

Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw, Eng. 25 (4), 557–572.

Soliman, M., Riebisch, M., 2014. Modeling the interactions between decisions within software architecture knowledge. In: Avgeriou, P., Zdun, U. (Eds.), Software Architecture, 8627. Springer International Publishing, pp. 33–40. http://dx.doi.org/10.1007/978-3-319-09970-5_3.

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., Babar, M.A., 2010. A comparative study of architecture knowledge management tools. J. Syst. Softw. 83 (3), 352–370.

Tofan, D., Galster, M., Avgeriou, P., 2013. Difficulty of architectural decisions: a survey with professional architects. Proceedings of the Seventh European Conference on Software Architecture, ECSA'13. Springer-Verlag.

Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. IEEE Softw. 22 (2), 19–27.

Verner, J., Sampson, J., Tosic, V., Bakar, N.A., Kitchenham, B.A., 2009. Guidelines for industrially-based multiple case studies in software engineering. Proceedings of the Third International Conference on Research Challenges in Information Science. Fez. pp. 313–324. http://dx.doi.org/10.1109/RCIS.2009.5089295.

van Vliet, H., Tang, A., 2016. Decision making in software architecture. J. Syst. Softw. http://dx.doi.org/10.1016/j.jss.2016.01.017.

Yazdani, B., 1999. Four models of design definition: sequential, design centered, concurrent and dynamic. J. Eng. Des. 10 (1), 25–37. http://dx.doi.org/10.1080/095448299261407.

Yin, R.K., 2008. Case Study Research: Design and Methods, 4. Sage Publications Ltd.

Zimmermann, O., 2011. Architectural decisions as reusable design assets. IEEE Softw. 28 (1), 64–69. http://dx.doi.org/10.1109/MS.2011.3.

Zimmermann, O., Gschwind, T., Küster, J., Leymann, F., Schuster, N., 2007. Reusable architectural decision models for enterprise application development. In: Szyperski, C.A., Reussner, R., Stafford, J.A. (Eds.), Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Berlin Heidelberg, pp. 15–32. http://dx.doi.org/10.1007/978-3-540-77619-2_2.

Zimmermann, O., Wegmann, L., Koziolek, H., Goldschmidt, T., 2015. Architectural decision guidance across projects problem: problem space modeling, decision backlog management and cloud computing knowledge. Proceedings of the Twelfth Working IEEE/IFIP Conference on Software Architecture.Montreal. pp. 85–94. http://dx.doi.org/10.1109/WICSA.2015.29.

Dr. Paris Avgeriou is Professor of Software Engineering at the University of Groningen, the Netherlands, where he has led the Software Engineering research group since September 2006. Before joining Groningen, he was a post-doctoral Fellow of the European Research Consortium for Informatics and Mathematics. He has co-organized several international conferences and workshops (mainly at ICSE). He sits on the editorial board of IEEE Software and Springer Transactions on Pattern Languages of Programming (TPLOP). His research interests lie in the area of software architecture, with strong emphasis on architecture modeling, knowledge, technical debt, patterns and link to requirements.

Christian Manteuffel is currently pursuing his Ph.D. at the Software Engineering and Architecture research group of the University of Groningen in the Netherlands. He obtained his M.Sc. in computer science at the University of Groningen in September 2013 with highest distinctions and he is a graduate of the Honours College of the University of Groningen. His primary research focus are software architecture decisions, particularly architecture decision management in the domain of embedded systems.

Roelof Hamberg holds a Ph.D. degree in Physics from the University of Leiden. He joined Philips Research Labs Eindhoven in 1992 to work on perceptual image quality modeling and evaluation methods. Later he joined Océ as in-product control software developer, digital system architect, and department manager in R&D. In 2006 Roelof joined the Embedded Systems Institute (TNO-ESI) as research fellow, doing applied research in the field of system behavior and systems architecting. He was involved in research projects with Océ, Vanderlande, NXP, and European partners. Currently, he works as a product architect and project leader with Océ.