The Journal of Systems and Software 112 (2016) 181–198

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Decision architect – A decision documentation tool for industry

Christian Manteuffel [a,*], Dan Tofan [a], Paris Avgeriou [a], Heiko Koziolek [b], Thomas Goldschmidt [b]

[a] University of Groningen, Groningen, The Netherlands
[b] ABB Corporate Research, Ladenburg, Germany

ARTICLE INFO

Article history:
Received 15 December 2014
Revised 14 October 2015
Accepted 20 October 2015
Available online 30 October 2015

Keywords:
Architectural viewpoints
Architecture decisions
Tool-support

ABSTRACT

Architecture decisions are often not explicitly documented in practice, even though explicit capturing and documentation of architecture decisions has been associated with a multitude of benefits. Several decision documentation tools have been proposed but they often do not meet the expectations and needs of the industry. As part of a research collaboration with ABB, we developed a decision documentation tool that aims to ensure industrial applicability. This tool is an add-in for Enterprise Architect and is an implementation of a viewpoint-based decision documentation framework. To validate the add-in, we conducted a case study with architects from ABB. In this study, we assessed to what extent and why software architects intend to use our approach. We therefore investigated the perceived usefulness, perceived ease-of-use, and contextual factors threat influence the architect's intention to use the tool. We found that the tool has a high relevance for software architects, that it increases the quality of decision documentation and that it improves productivity of architects in various ways. With respect to ease-of-use, the study identified the need for better guidelines. With respect to contextual factors, we found several factors that influence the intention to use the add-in in an industrial environment.

1. Introduction

Architecture decisions (AD) are the most important choices an architect has to make (van Heesch et al., 2013). They determine the overall structure, behavior and quality of a software system. The perspective of looking at software architecture as a set of architecture decisions is widely recognized (Jansen and Bosch, 2005; Kruchten et al., 2009). The explicit documentation of these architecture decisions has been associated by researchers with a multitude of benefits, such as avoiding knowledge vaporization, supporting change impact estimation, increasing system understanding, improving knowledge sharing, and facilitating architecture evaluation (Kruchten et al., 2009; Tyree and Akerman, 2005; Van Der Ven et al., 2006). However, in practice, architecture decisions are often not explicitly documented but reside in the architect's mind as tacit knowledge or are only implicit in the models or other documents that the architect creates (Kruchten et al., 2009; Tyree and Akerman, 2005).

There have been attempts to close this gap between industrial practice and the benefits demonstrated by research, through a growing number of architectural knowledge management tools being proposed (Tang et al., 2010). However, most of these tools are developed in academic, non-commercial contexts, and therefore often do not meet the expectations and demands of the industry (Koziolek and Goldschmidt, 2013). In a comparative study of architectural knowledge management tools, Tang et al. (2010) identified several limitations of these tools. First, the tools are often poorly embedded into the architecture design process or existing tool chains. Second, they are often only applicable during specific stages of the architecture lifecycle. Third, they do not integrate well with existing architecture documentation standards, such as the ISO/IEC/IEEE 42010. Furthermore, van Heesch et al. (2012a) argued that existing approaches often "do not frame all concerns of all stakeholders in an adequate and useful manner", which further limits their applicability in industrial projects.

In order to address the aforementioned limitations of existing tools, we have developed a new tool, the Decision Architect, as part of a research collaboration between the University of Groningen (RuG) and ABB. Our intent was to ensure industrial applicability by combining an academic approach that goes beyond the state of the art, with the practical constraints of a large company. Our collaboration included requirements interviews with software architects, an analysis of existing documentation, frequent feedback and an extensive validation of the tool in two industrial case studies. A seamless integration into the software development processes has been achieved by developing the tool as an extension of Enterprise Architect; Enterprise Architect is the modeling tool of choice of ABB's Software Architects. By integrating the decision documentation tool into an existing

* Corresponding author. Tel.: +49 1787282497.
E-mail addresses: c.manteuffel@rug.nl, cm@notagain.de (C. Manteuffel), d.c.tofan@rug.nl (D. Tofan), paris@cs.rug.nl (P. Avgeriou), heiko.koziolek@de.abb.com (H. Koziolek), thomas.goldschmidt@de.abb.com (T. Goldschmidt).

http://dx.doi.org/10.1016/j.jss.2015.10.034
0164-1212/

modeling platform, architects can use the same tool that they use to create models, diagrams and views of the architecture, to document the decisions that constitute these artifacts. This narrows the gap between architecture documentation and architecture decision documentation that still exists in industry today. Furthermore, the tool implements the 42010-based decision documentation framework proposed by van Heesch et al. (2012a) to make sure that the tool supports through different architecture viewpoints, all phases and stakeholders concerns of a typical software architecture project.

A preliminary version of the tool has been evaluated in a first exploratory case study (Manteuffel et al., 2014). In this case study we assessed the status quo of architecture decision documentation at selected business units at ABB, identified architects' expectations of a hypothetical ideal decision documentation tool, and made a preliminary evaluation of Decision Architect. The main findings of that study are as follows (Manteuffel et al., 2014):

- Awareness of decision documentation is increasing at ABB, but still several barriers exist that limit the use of decisions in practice. Such aspects include a lack of tool-support, the absence of a standardized documentation format and limited guidance on how to utilize decision documentation.
- Regarding the ideal tool, architects want a descriptive and efficient approach. Additional features like reporting or decision sharing were requested.
- The developed add-in was well perceived by the architects. For example, the ability to create traces between decisions and other modeling elements was regarded beneficial. However, the first study also identified areas for improvement, such as a clearer separation between the problem of a decision, the outcome, and considered alternatives.

Based on the insights and feedback obtained during the first study, we have evolved Decision Architect. In this paper, we present both the evolved decision documentation tool and a second confirmatory case study, in which we assess to what extent the tool meets the expectations of software architects and is applicable in industrial projects. Furthermore, the study increases the understanding of how decision documentation is perceived in industry. Finally, the study presents lessons learned from the execution of the study.

The paper is organized according to the guidelines for reporting case studies proposed by Runeson et al. (2012). Section 2 begins by introducing the conceptual decision documentation framework, and presents the evolved tool as well as a brief summary of the exploratory study, the resulting adaptations and related work. Section 3 is concerned with the case study design. It briefly presents the research questions and describes case and subject selection, data collection and analysis procedures. The fourth section presents the findings of the case study, focusing on the three key themes: perceived usefulness, perceived ease-of-use and contextual factors. Section 5 assesses the validity of the study. Section 6 critically discusses the findings followed by implications for practitioners and researchers. Finally, the conclusion gives a brief discussion of the implications of the findings to future research into this area.

2. Background

The remainder of this section briefly summarizes the decision documentation framework and presents Decision Architect before discussing related work in the area of decision documentation tools.

2.1. Decision documentation framework

Decision Architect is based on the conceptual decision documentation framework proposed by van Heesch et al. (2012a, 2012b). The framework allows architects to capture important rationale of a system by treating architecture decisions as first-class entities in architecture specifications. The framework uses the conventions of the ISO/IEC/IEEE 42010 and thus, can be easily used in combination with other architectural viewpoints like Kruchten's 4+1 (Kruchten, 1995b). As illustrated in Fig. 1, the framework defines five viewpoints each framing a different set of stakeholder concerns, for instance, identifying decision dependencies in order to estimate the impact of changes, or to assess the degree of which decisions cover the architecturally relevant requirements. The five viewpoints are *decision relationship viewpoint, decision detail viewpoint, decision chronology viewpoint, stakeholder involvement viewpoint*, and *decision forces viewpoint*. For readability, we omit the decision prefix in the rest of the paper when referring to the viewpoints. Due to the scope of this paper, we can only present a brief overview of the decision documentation framework. Please refer to van Heesch et al. (2012a); 2012b) for a detailed description of the viewpoints and its concepts.

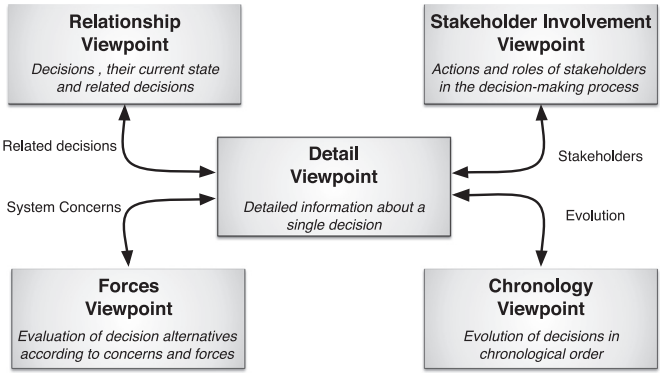

Fig. 1. The five decision viewpoints of the decision documentation framework.

The relationship viewpoint illustrates connections between decisions, e.g., Hibernate was *caused by* the decision to use MySQL. Moreover, the relationship view provides an overview over multiple decisions and their current state. The framework proposes seven decision states: *idea, tentative, discarded, decided, approved, challenged*, and *rejected*. It is possible to adapt the set of decision states to the needs of the respective project (van Heesch et al., 2012a), for example, to reflect a more agile process with fewer decisions states.

The forces viewpoint captures the evaluation of multiple decision alternatives according to a set of architectural forces, such as architecturally significant requirements. The concept of a force is defined as any aspect (development, business, political, social, legal, etc.) that "influences the architect when making a decisions out of multiple alternatives (van Heesch et al., 2012b)".

The stakeholder involvement viewpoint traces the actions of stakeholders in the decision-making process and thus, makes their role and involvement explicit. Stakeholders can be involved in various ways in the decision-making process, for example, they can *formulate, propose, discard, reject, confirm, challenge* or *validate* a decision.

The decision chronology viewpoint describes the evolution of decisions over time. It allows the architect to understand the temporal order in which decisions were, for instance, considered, taken, or rejected. It provides important insights into the process that led to the architecture.

The detail viewpoint presents an overview of a single decision including a description of the underlying problem, the offered solution and the arguments that justify the decision. The detail viewpoint is inspired by existing decision templates like the decision template proposed by Tyree and Akerman (2005).

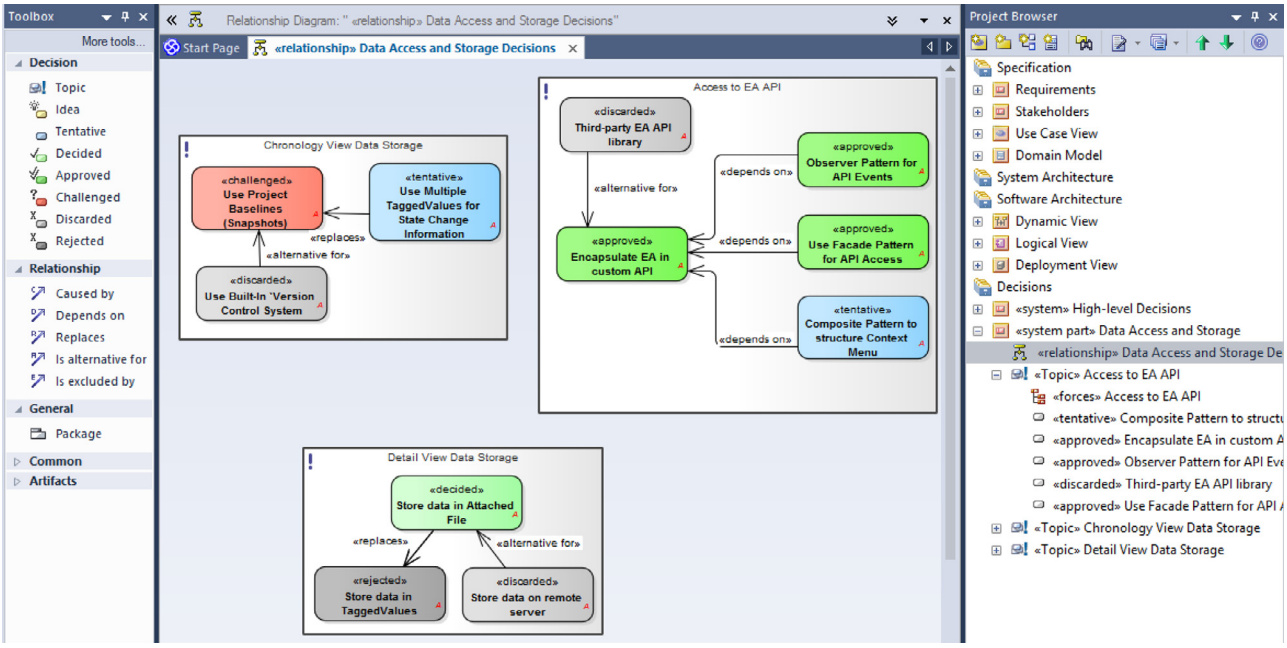

Fig. 2. A decision relationship view in Enterprise Architect. The project browser (right) shows the integration of architecture decisions and other architecture views.

2.2. Decision Architect

Decision Architect (DA) allows documenting and managing architecture decisions according to the five viewpoints defined by the decision documentation framework. Each viewpoint is specialized to capture a particular aspect of the system's decisions and thus, to answer a different set of questions about the system's rationale and to support different architecting activities. DA offers the possibility to trace decisions to other architectural modeling elements, like components, classes or requirements, and to generate decision reports. Although many stakeholders are involved in the architecting process, architecture views are mainly created by architects and therefore, they are the primary users of the tool. Other stakeholders, such as Managers, Requirement Engineers, Software Engineers, Reviewers or Customers, will benefit from DA primarily as consumers of decision views either by using the tool or through exported reports.

DA has been designed with flexibility in mind. A flexible approach for decision documentation is essential (Manteuffel et al., 2014). Therefore, DA supports both capturing of decision after the fact, i.e., after parts of the architecture have been modeled or implemented, and capturing decisions before or while modeling the architecture. As Capilla (2009) argues, the latter scenario leads to a more elaborated process and better reflects the reasoning activity that happens in the architect's mind, while the retrospective documentation of decisions often requires less effort because a lot of rationale is embedded in the design choices. Nevertheless, retrospective documentation has a higher risk of omitting important information. Therefore, we suggest that Decision Architect is used with a combination of both approaches.

The flexibility of DA is further emphasized by the fact that the five decision viewpoints can be used separately or in combination. The selection of viewpoints depends on the stakeholder concerns considered to be important for a project's architecture specification. None of the five viewpoints is mandatory and all of them can be seen as optional with the only requirement that at least one viewpoint is used in a project. However, the results of related studies suggest that the relationship, detail and forces viewpoint are considered most useful (van Heesch et al., 2012a; 2012b; Manteuffel et al., 2014). Furthermore, DA does not enforce strict models in order to better support descriptive models that can cope with the creativity inherent to designing software systems.

DA pays special attention to the seamless integration of decision documentation into the work of a software architect. This close integration has been accomplished by developing Decision Architect as an extension to Sparx Systems' Enterprise Architect, a widely-used general purpose modeling tool for creating UML-based models. An add-in has the advantage of a lower learning curve because users are already familiar with the platform. However, the concepts and features implemented in Decision Architect could possibly be transferred to any other modeling tool that provides a plugin architecture, like Visual Paradigm[1] or Eclipse.[2] Decision Architect freely available online[3] under an industry-friendly open source license.

2.2.1. Decision relationship viewpoint

DA treats decisions as first-class modeling elements. This means that decisions and decision topics are stored in a central project repository just like any other modeling element. Decisions topics (grey box with an exclamation mark) provide the means to associate multiple decision alternatives to a common decision point, e.g., an architectural issue. Topics can also be used to express upcoming or unsolved issues that require a decision in the future. Decisions are represented by a rounded rectangle that displays the name and the state of a decision (cf. Fig. 2). The color of a decision reflects its state, for example, *challenged* decisions are indicated in red to emphasize that this decision represents an unsolved problem and requires further discussion. A challenged decision can be solved in two different ways, either by selecting an alternative option as a replacement or by reaffirming the challenged decision. In the first case, the state transitions from challenged to rejected while in the later the new state is approved. *Discarded* decisions are displayed in an unobtrusive grey while *approved* decisions are visualized in green. Decision states are a property of the decision-making process, which means that a decision has potentially many versions, one for every state change. The relationship viewpoint only shows the latest version of a decision and thus, only provides a snapshot of the decision-making process. The

[1] http://www.visual-paradigm.com.
[2] http://www.eclipse.org.
[3] http://decisions.codeplex.com.

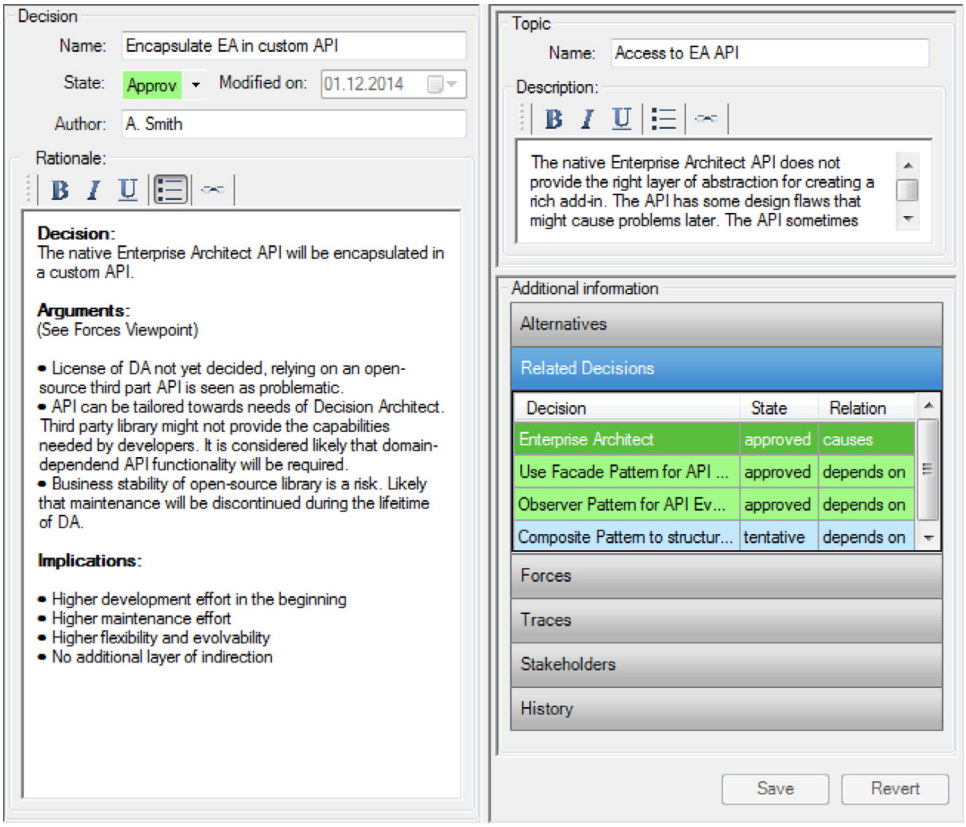

Fig. 3. A concrete decision detail view illustrating the arguments and alternatives of one decision.

chronology and stakeholder involvement viewpoint frame concerns related to the decision-making process.

Fig. 2 shows an excerpt of a relationship view embedded in the user interface of Enterprise Architect. On the left side of the picture is the toolbox, which shows modeling elements and relationships that are relevant to the current diagram, in this case, decisions and decision relationships. On the right side is the project browser, in which all architecture views and architecture elements and diagrams (including decisions and topics) are hierarchically organized. This illustrates how closely the design rationale is embedded in the architecture specification using DA.

The combination of decision relationships and their current decision state provides a good overview and supports system understanding. For example, one of the topics shown in Fig. 2 requires further discussion since a design alternative has been challenged. The relationship viewpoint particularly assists in change impact estimations, architecture reviews and supports decision reuse.

2.2.2. Decision detail viewpoint

The decision detail viewpoint offers the possibility to provide a textual description of the decision's rationale and accumulates related information from other decision views, such as alternatives, forces, traces or stakeholders (cf. Fig. 3). The rationale can be filled in according to a template. However, the template is not predetermined by the tool and can be defined by the user. We decided to capture the textual rationale in an unstructured form in order to allow for company or project-specific decision templates, like Tyree and Akerman (2005) or the minimal Why-approach by Zimmermann (2012). This decision was an explicit trade-off between flexibility and the possibility for automatic information retrieval. The unstructured form increases the effort for a context sensitive search or automated reuse. However, the results from the previous study showed that architects ranked the possibility for custom templates as more important.

The detail viewpoint is particular useful during maintenance and evolution because it provides deeper insights compared to the other views. However, filling out this view also requires more effort and thus, it is not recommended to document every decision in detail. In accordance with Tang et al. (2006), it is often sufficient to document especially those decisions whose rationale will be difficult to reconstruct later on.

2.2.3. Decision forces viewpoint

The forces view captures the analysis of design alternatives for an architectural issue according to all architecturally significant decision forces. The forces view is a matrix that lists the forces on the vertical dimension and the decisions on the horizontal dimension, as shown in Fig. 4. The view provides a good overview of which design alternative best supports the relevant forces. Notably, the forces view does not replace the evaluation of design alternatives but rather captures its results in an intuitive and easy understandable way.

The forces view plays an important role in system understanding. On the one hand, the forces view establishes the backwards traceability to the motivational factors of a decision like requirements, assumptions, or any other architectural element. On the other hand, it provides codified rationale in form of force-ratings that indicate the influence of a force on one or more design alternatives. The rating schema is not predetermined and can be defined by the user, for example, a seven-item scale can be used where +3 indicates a strong positive influence and -3 a strong negative influence.

2.2.4. Decision chronology viewpoint

The chronology viewpoint depicts the evolution of decisions over time. Fig. 5 illustrates the evolution of design alternatives for a particular decision topic. The chronological order is established via the *followed-by* relation. In this case, the oldest decision can be found in the top left and the latest in the bottom right. However, the layout

Add Decision | Add Force | Excel

	Concern	<<discarded>> Third-party EA API library	<<approved>> Encapsulate EA in custom API
License model	Business Context	-1	+3
Dependence on third party	Business Context	-2	+3
Effort of maintenance	Maintainability	+2	-3
Effort to learn and understand API	Maintainability	0	+1
Evolvability of component	Maintainability	-1	+2
API suits developers needs	Functionality	+1	+3

Fig. 4. The decision forces view visualizes the impact of architecturally significant forces on design alternatives.

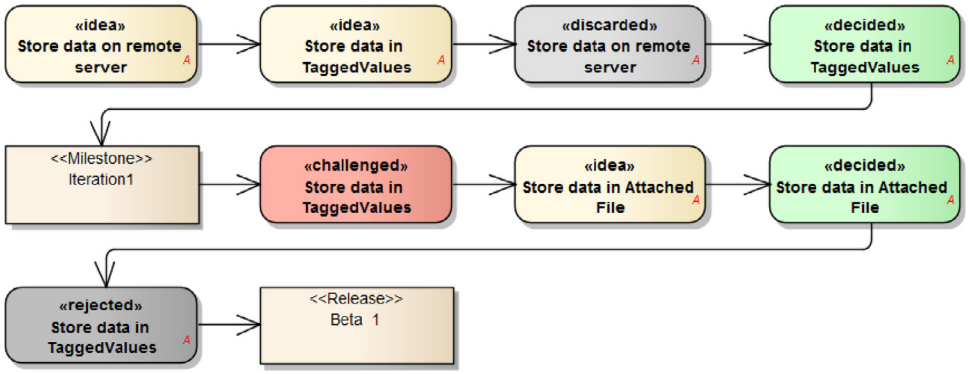

Fig. 5. A decision chronology view showing the evolution of decisions for two milestones.

and orientation depends on the personal preferences of the user. It can be seen that the decision *"Store data in TaggedValues"* was challenged after (Iteration 1) and it has been eventually replaced by the new design alternative *"Store data in Attached File"*. After the new design alternative was chosen, the challenged decisions was marked as rejected. The chronology view can be used to reconstruct the decision process, for example, by new team members to catch up with a project or by managers when reviewing the project.

In order to reduce the effort required to create the chronology view, DA provides the possibility to automatically generate this view for a set of decisions. Therefore, DA keeps a timestamped record whenever the state of a decision is changed. When the user wants to generate the chronology view, these records are used to establish the followed-by relationships. In the last step, the user can use the automatic layout of Enterprise Architect to arrange the elements on the view.

2.2.5. Decision stakeholder involvement viewpoint

The stakeholder involvement viewpoint makes the roles of stakeholders in the decision-making process explicit (cf. Fig. 6). Since the viewpoint captures the actions of stakeholders , it can be used to identify conflicts or disagreements between stakeholders. The stakeholder involvement view supports knowledge personalization (e.g. who-knows-what instead of what-is-known) by providing pointers to the persons that were involved in the decision-making process.

Knowledge personalization plays an important role because it is often not feasible or possible to fully document the rationale behind a decision. Using the stakeholder involvement view the architect or the reviewer can directly ask the persons involved in the decision. Knowledge personalization shifts the effort to the future, from documenting rationale to recovering rationale. An inherent risk of this approach is a higher possibility that important knowledge is getting lost, e.g., because it is forgotten or the person becomes unavailable. Therefore, a hybrid approach is often recommended (Farenhorst and van Vliet, 2009) which can be easily achieved in DA using the various decision viewpoints.

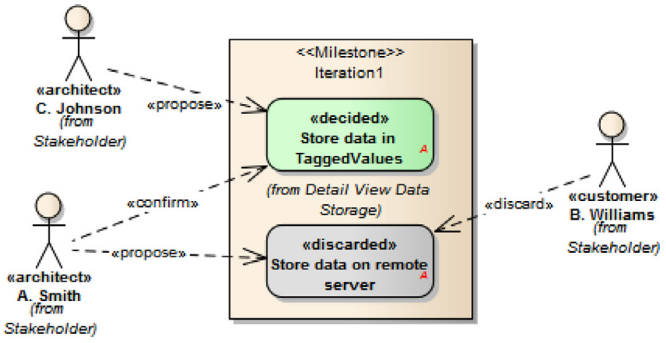

Fig. 6. A stakeholder involvement view showing the actions of three stakeholders in the decision-making process.

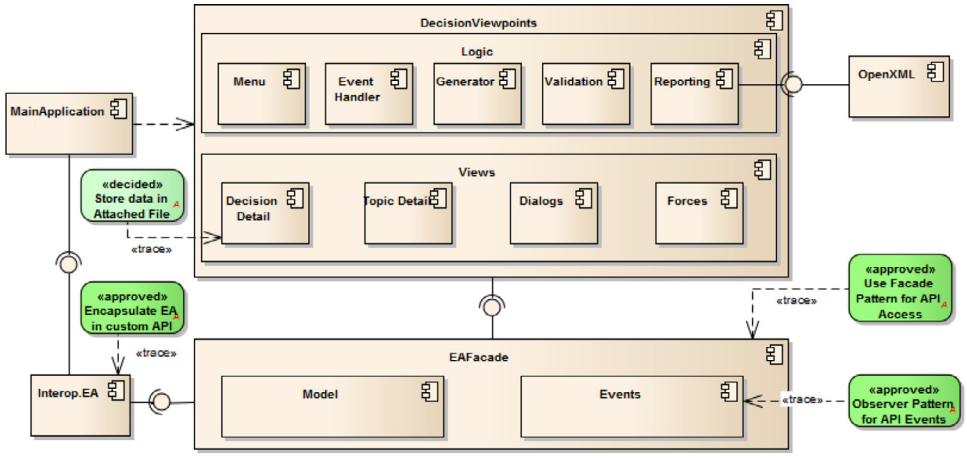

Fig. 7. Trace between decision and baseline component.

2.2.6. Tracing and reporting

One of the most important features of the add-in is the possibility to trace decisions to other modeling elements and thus, enhance the integration of the decision documentation with the rest of the architecture specification. While the forces viewpoint provides traceability to the motivational reasons of a decision, traceability to the design outcome of a decision is equally important (Tang et al., 2007), for example, to support change impact estimation or system understanding. Therefore, the add-in provides a user-friendly way to create traces between decisions and any model elements within Enterprise Architect. So far, DA does not provide the possibility to trace individual attributes of elements or complete diagrams. A trace can be established by connecting a decision and the model element via the trace connector from the toolbox. A trace link can be further refined via stereotypes to provide additional semantics, for example, *result of* or *refined by*.

Moreover, it is possible to hide or show traced decisions in a diagram or to follow trace links. Trace links are bi-directional, so the user can navigate from decisions to design elements and vice versa using the context menu. Fig. 7 shows a component view of Decision Architect including the decisions which affect a particular component. This view is a good example of how other architectural views can be rationalized and enriched by linking them to decisions.

Furthermore, the add-in provides the possibility to generate reports out of selected decision views. Reports play an important role in knowledge sharing since non-technical stakeholders like Managers rarely use modeling tools in their daily work. Three types of documents can be generated. Presentation slides in PowerPoint format, e.g., to be used during architecture reviews; reports in Word format, e.g., to be filed in document management systems or shared with stakeholders that do not have Enterprise Architect; and forces view in Excel format, e.g., as a template during design-space exploration.

2.3. Related work

The importance of preserving architectural design rationale along with the structural aspects of a system has been long acknowledged (Perry and Wolf, 1992; Kruchten, 1995a). In 2004, Bosch recognized that the documentation of architecture decisions and the rationale behind them play a major role in preserving design rationale Bosch (2004). Therefore, he argued that architecture decisions should be treated a first-class architecture elements and thus, should be explicitly documented. Since then, researchers associated the explicit documentation of decisions, and the design rationale that they embody, with a multitude of benefits.

The most frequently used argument in favor of decision documentation is the preservation of architectural knowledge, that if not explicitly captured has a high risk of getting lost (also known as knowledge vaporization). Tang et al. (2006) found in a survey, that it is not uncommon that architects forget details about their own design or that they need to modify systems designed by other architects, who might be unavailable or left the company. The survey findings reconfirmed the importance of preserving design rationale. van Heesch and Avgeriou (2011) stress the importance to not only capture the obvious technical decision-drivers, such as functional and non-functional requirements, but also the non-technical aspects, like time and resources limitations, or political issues, which are frequently being considered in the reasoning process. These non-technical factors justify otherwise counter-intuitive decisions that do not present the best technical solution.

In addition to avoiding knowledge vaporization, decision documentation has also immediate positive effects on the software process, for example, improving the support for change impact estimations. Bratthall et al. (2000) demonstrated in a controlled experiments the advantages that documented design rationale has for system evolution. They found that access to the original design rationale has a positive effect on correctness and speed of change impact estimations. Moreover, studies showed an effect of decision documentation on the reasoning process itself. van Heesch et al. (2013) demonstrated that inexperienced designers are “more systematic in exploring and evaluating solution options” when using decision viewpoints. In an earlier study, van Heesch et al. (2012a) showed that creating decision views, according to their proposed viewpoint specification, support architects in following a more systematic reasoning process.

However, capturing design rationale in an effective and efficient way, is still an open issue; as a consequence, the adoption of documentation approaches in industry is rather low. Indeed, documenting decisions is often considered to be too effort-intensive by practitioners, which makes finding the right balance between tacit and documented knowledge an important issue. Tang et al. (2006) argued that a comprehensive rationale documentation is often not necessary because certain rationale can be reconstructed, especially for non-complex systems. In the same line of reasoning, Falessi et al. (2013) investigated which aspects of design rationale documentation provide the highest value depending on the activity that is performed. A key finding is that the description of the decision, the issue and related requirements has a high value regardless of the activity. Moreover, they found that other aspects of design rationale provide different values for different activities (e.g., considered alternative solutions are less valuable when performing change

impact estimations), therefore they argue that the documentation should be customized according to the activities it should support. This will reduce the amount of effort required from the producer.

Several studies analyzed and compared architectural knowledge management approaches that were proposed by researchers. In 2010, Tang et al. (2010), conducted a study that compared five architectural knowledge management tools based on how they support the various activities in the architecture life-cycle. They found that there is no tool that supports all activities of the architecture life-cycle, i.e. analysis, synthesis, evaluation, implementation, and maintenance. The results also show that the capability to provide different perspectives on AK is limited, as the tools often do not support the concept of views and viewpoints. Furthermore, it was noted that knowledge sharing features are not fully supported by the tools yet. This finding is also supported by an earlier study conducted by Farenhorst et al. (2007). In this study, the authors analyzed five tools from academia and industry from a knowledge sharing perspective. The results show that AK personalization strategies are not sufficiently supported and that the tools do not provide the possibility to generate stakeholder-specific content. The latter was also identified by Tang et al. (2010). In 2009, Liang and Avgeriou (2009) surveyed nine existing tools on their support of typical AK-related use cases. They found that the tools lack the ability to identify stakeholders, to apply decisions to produce the design (synthesis), to evaluate AK, and to support the assessment design maturity. In the following, we discuss six examples of AKM tools and compare those to Decision Architect.

- *SAW*, proposed by Nowak and Pautasso (2013), is a web-based tool that focusses on collaborative decision-making, capturing, and brainstorming.[4] SAW allows the architect to visually explore the design space by following links between knowledge elements, for example, it enables the architect to discover relevant architectural issues and related alternative solutions for that issue. In contrast to DA does SAW not provide any integration into existing tool chains and tracing is limited to AK elements. Moreover, it does not integrate well with ISO 42010.
- *ADVISE* has been developed by the Software Architecture Group at the University of Vienna. Its core focus lies on "modeling reusable architectural decisions and architectural decisions under uncertainty using Fuzzy models[5]". ADvISE has been implemented as an Eclipse plug-in, a tool that is widely used in the software industry. ADvISE also offers a close integration into existing tools used by architects but in contrast to DA it does not provide the ability to trace decisions to existing architecture elements. Moreover, it only provides a single perspective on ADs and does not integrate well with ISO 42010.
- *AREL*, proposed by Tang et al. (2007), is an add-in for Enterprise Architect which focusses on capturing relations between design decisions and architecture elements. Hence, it especially supports traceability-related use cases such as change impact analysis, root-cause analysis, and design verification. Although AREL has been designed to be used in combination with architecture viewpoints, it does not provide any own viewpoints related to decision documentation and thus only offers a limited perspective on ADs. AREL compared to DA offers a more refined model for capturing traces. However, similar to AREL, DA also offers the option to trace decisions to architecture elements as motivational reason (via forces views) and as design outcome (via trace links).
- The *Knowledge Architect* tool suite, proposed by Liang et al. (2010), has been designed to support collaborative architecting. Therefore, the tool focusses on knowledge sharing via a central knowledge repository. The knowledge in the repository is accessed and extended through a set of specialized clients, such as an Excel or a Word plugin that allow the extraction of design rationale in existing documents. Although Knowledge Architect integrates well with the tools used by architects through its various clients, it focusses primarily on capturing AK and only offers limited possibilities to access and visualize AK.
- Capilla et al. (2008) proposed ADDSS, a web-based tool for storing, managing, and sharing architecture decisions. ADDSS compared to DA also captures the evolution of AK and provides the possibility to trace decisions to requirements as well as to candidate architecture models. In addition to DA, it provides access to generic knowledge in the form of patterns and architecture styles. However, the tool only provides a limited integration into the architecting process and existing tool chains, e.g., architecture models can only be added as image.
- *Archium*, proposed by Jansen et al. (2007), takes a different approach by defining an extension for a component-based Java language. This extension allows documenting decisions directly within the source code, combining the implementation of the system and its AK. While Archium offers a close integration in the realization phase of the system, it ignores the fact that architects often work on a higher level of abstraction. Important architecture decisions are often taken long before the first line of source code is written and not all decisions can be logically linked to source code level, such as, executive or non-existence decisions as discussed by Kruchten et al. (2006).
- *PAKME* is a web-based tool that has been built on top of an open source groupware platform (Babar and Gorton, 2007). A design goal of PAKME was to provide support for geographically distributed stakeholders involved in the software architecture process. PAKME offers a template-based approach of capturing design options and decisions. However, the tool provides no possibility to define traces to other architecture elements and only offers limited perspectives on AK.

3. Case study design

Decision Architect is aimed at providing a decision documentation tool that meets the expectations of industry and that architects are willing to adopt in their daily work. Therefore, we want to evaluate to what extent and why software architects intend to use the DA. The intention of individual architects to use the tool, can be determined by three separate factors: the usefulness of the add-in, the ease-of-use of the add-in, and contextual factors. According to Davis (1989), usefulness and ease-of-use play a fundamental role in predicting the degree to which an individual would use a new technology. In addition to usefulness and ease of use, the intention to use also depends heavily on contextual factors that are beyond the influence of a concrete technology, such as: the individual's background; social influence; and facilitating conditions (Venkatesh et al., 2003). Those factors are relevant in order to understand the environment in which the add-in provides value for architects and can thus be adopted in practice. Based on the Goal Question Metric approach by Basili et al. (1994), we have defined the following goal of the study:

> *Analyze*the architecture decision documentation activity of software architects *for the purpose of* evaluating the intention to use Decision Architect *with respect to* usefulness, ease-of-use, and contextual factors *from the point of view of* external empirical researchers *in the context of* selected software architects from different business units of ABB.

Based on this study goal, we derived the following three research questions.

RQ 1. *How do software architects perceive the usefulness of the add-in?* Using the definition of perceived usefulness proposed by Venkatesh and Bala (2008), we investigate to what extent

[4] http://saw.inf.unisi.ch/drupal/.

[5] https://swa.univie.ac.at/Architectural_Design_Decision_Support_Framework_(ADvISE).

architects believe that the add-in supports the architecture decision documentation activity and thus, enhances their performance as architects.

RQ 2. *How do software architects perceive the ease-of-use of the add-in?* Based on the definition of perceived ease-of-use proposed by Venkatesh and Bala (2008), we want to know to what extent architects believe that using the add-in will not require much effort.

RQ 3. *What are the contextual factors and how do they influence the intention to use the add-in?* In practice, the use of a tool often depends on external factors, for example, the nature and size of a software project, the preferred way of working of individuals, or corporate culture and guidelines. We ask RQ3 to elicit and understand the context that leads to a successful application of the add-in in an industrial project. Furthermore, this research questions allows us to evaluate to what extent our results are applicable to other projects/companies, since the case study is limited to software architects from ABB.

3.1. Research method

We decided to conduct a confirmatory case study to evaluate the developed add-in in a typical and realistic context as defined by Runeson et al. (2012), in order to obtain an understanding of the intent of architects to use the developed add-in for architecture decision documentation in industrial software projects. Studying aspects of software projects, such as decision documentation, in a realistic context often results in limited control of variables, such as the complexity and duration of the project, the number and completeness of documented decisions, or the business culture within the observed project. The case study method is particular useful for situations with limited control over variables (Yin, 2008).

We could not use experimentation as a research method because findings from an experiment are often bound to specific lab conditions, which are difficult to achieve in real software projects. Likewise, we dismissed surveys as a research method because surveys are suitable to identify characteristics of a population rather than evaluating a tool in a realistic setting. However, it is important to be aware of the characteristics and limitations inherent to case studies. For example, case studies cannot achieve the same scientific rigor as scientific experiments (Runeson et al., 2012). Moreover, the limited control over independent variables limits the internal validity of conclusions (Cavaye, 1996). In this study, we followed the guidelines for conducting and reporting case study research described by Runeson et al. (2012).

We decided for a holistic multiple-case design. In a holistic case design, the case is studied as a whole and not divided into multiple units of analysis (Runeson et al., 2012). According to Yin (2008), a multiple-case design allows for stronger conclusions due to the inherent replication and the comparability of the results across cases. The study analyzes *the activity of a software architect using Decision Architect to document decisions*. The study adopts a holistic approach because the behavior and perceptions of architects depends on their individual context comprised by their project, responsibilities and working attitude and thus, differ on a case by case basis.

Table 1
Mapping between research questions and sources of data.

	RQ1	RQ2	RQ3
Documentation analysis	x	x	
Interviews	x	x	x
Focus group	x	x	x
Background questionnaire			x
Webinar questionnaire	x	x	x

Since the participants of the study are all assigned to different software projects, we consider each participant as an individual case.

3.2. Case selection

Our main criteria for inviting architects to participate in this study were as follows. The architects had to be currently involved in a software project, in which they had the responsibilities of a software architect. Furthermore, they should have considerable experience in Software Architecture and should use Enterprise Architect on a regular basis. For pragmatic reasons, architects that met the criteria were invited based on availability.

3.3. Data collection

To limit the effects of one interpretation and to mitigate validity threats, we used multiple data sources and multiple data collection techniques, allowing for data triangulation to strengthen conclusions (Runeson et al., 2012). According to the classification by Lethbridge et al. (2005), we used the following four techniques: questionnaire, interview, focus group, and documentation analysis. Table 1 shows the coverage of research questions by the data collection techniques. For each case, we executed the data collection procedure illustrated in Fig. 8. First, we asked each participant to fill out an online questionnaire. The questionnaire asked about the professional background, characteristics of the project in which they are currently involved and the current approach to architecture decision documentation within this project.

After the questionnaire we asked the participants to use DA for a project in which they are currently involved. The produced documents presents a glimpse of a person's understanding of a tool (Lethbridge et al., 2005), in this case DA. The participants had one week time for this task. We provided a website with video tutorials as learning material. While they were using the tool, we asked them to note down their observations in an observation form.

The produced decision documentation and the observations serve as input for the interviews. We decided to conduct semi-structured interviews, which "include a mixture of open-ended and specific questions, designed to elicit not only the information foreseen, but also unexpected types of information (Seaman, 1999)". According to Runeson et al. (2012), interviews are one of the most frequently used and most important data collection methods in Software Engineering case studies since "much of the knowledge that is of interest is not available anywhere else than in the minds of the participants

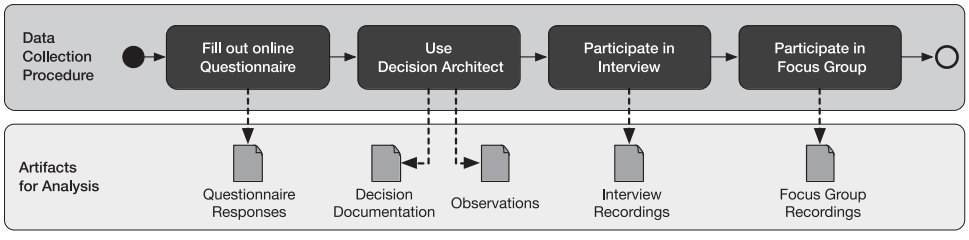

Fig. 8. Overview of the data collection procedure from the perspective of the participant.

Table 2
Role and projects ofparticipating architects.

	Participant 1	Participant 2	Participant 3	Participant 4
Exp. in industry (years)	13	8	5	10
Exp. in SA (years)	6	8	2	7
Exp. in SA (self-assessment)	Very good knowledge in SA	An expert, primarily involved in SA on a daily basis	Good understanding of SA	An expert, primarily involved in SA on a daily basis
Training in decision doc.	Yes	Yes	No	N/A
Project summary	Architecture redesign for an embedded communication platform	Technology research for cloud computing in an industrial environment	Remote access infrastructure design	System for power plant control
Role	Software Architecture Consultant	Lead Architect	Project Leader and Technical Contact for conceptual architecture	Software Architect
Decisions documented	11 (about half)	15 Decisions (about half)	25 (all)	5–6 decisions
Documentation approach	Arc42-template[1] combined with 42010-template[2]	Excel (later Decision Architect)	Why-approach Zimmermann (2012)[3]	PowerPoint slides

[1] http://arc42.org.
[2] http://www.iso-architecture.org/ieee-1471/templates/.
[3] http://www.sei.cmu.edu/library/assets/presentations/zimmermann-saturn2012.pdf.

(Runeson et al., 2012)". Finally, we conducted a focus group, which helps to put the answers that we obtained from the individual interviews into perspective. The group dynamics that occur in such a discussion help to focus on the most important topics, to provide checks and to identify extreme divergent views (Runeson et al., 2012).

Since ABB is multi-national corporation we decided to organize a webinar in order to present Decision Architect to a broader audience of Software Architects within ABB. At the end of the webinar, we asked the participants to fill out an online questionnaire about Decision Architect. The data collected in this questionnaire is used to cross-check our findings.

3.4. Data analysis

We collected mostly qualitative data, i.e., from interviews, focus group, decision documentation and open questions in the webinar survey. Quantitative data obtained in the webinar questionnaire was analyzed using descriptive statistics. Qualitative data was analyzed using the constant comparative method (Adolph et al., 2011). The goal of qualitative data analysis is the identification of generalizations from the data in a systematic way (Runeson et al., 2012). These generalizations emerge in an iterative process (Runeson et al., 2012). In this process the researcher assigns codes, i.e., tags or labels, to chunks of data, e.g., a piece of text or an image, "which are relevant to a particular theme of interest in the study (Seaman, 1999)". A coded chunk of data is called an incident (Adolph et al., 2011), which is one instance of a particular code in the data. Since different subjects sometimes use different descriptions for their experiences and insights, it is important to constantly compare codes and check if they are related to a common theme. During the coding and constant comparison of codes, the researcher gradually gets a better understanding of the data (Adolph et al., 2011). In the following we describe the analysis procedure that we followed during the study:

1. First, all documents were stored in a central study database. We used MAXQDA[6] to support the analysis.
2. In the following step, two researchers individually coded the interview and focus group transcripts. The transcripts were studied and incidents were summarized in a code, e.g., "stakeholder involvement viewpoint supports negotiations". An incident could be a statement from a participant of any length, e.g., a word, sentence or multiple paragraphs. We decided against using a predefined set of codes to uncover concepts that were not anticipated by the researchers.
3. After a document was coded by one researcher, the document was discussed between the two researchers and potential conflicts or different interpretations were resolved.
4. Subsequently, we revisited all already coded documents and compared codes. During this process, codes were refined, removed or merged. If codes had a common theme or topic we combined the codes under a common concept. In our case, a concept is a representation of a pattern of behavior, a recurring statement or an expression of an opinion, for instance, about the usefulness of the add-in. Steps 2–4 were repeated until all documents were coded and no new concepts emerged.
5. In the next step, all codes and incidents related to a particular concept were intensively studied, discussed and summarized into a memo. In this step, codes and incidents from different data sources were again compared against each other. This is a form of triangulation, which is a common procedure to establish validity in qualitative studies (Creswell and Miller, 2000).
6. In the final step, we grouped related concepts that had common theme or subjects into a higher level category that could be associated with one of the research questions, e.g. "job relevance". Overall, 240 incidents were coded, which resulted in 45 concepts that were grouped into seven categories.

3.5. Case and subject description

In this section we describe the subjects of the interviews and focus group as well as the demography of the webinar participants.

3.5.1. Case study subjects

By the time of the study, all case study participants worked for different Business Units of ABB and were involved in different projects. All of them are German. As shown in Table 2, they can be classified as experienced software architects with substantial practical experience. Three of the four participants took part in the first study (Manteuffel et al., 2014). Participants 1 and 2 took part in the focus group.

3.5.2. Webinar subjects

Approximately 25 persons attended the webinar session of which 11 filled out the questionnaire. The exact number of participants could not be determined because the invitation was send to a mailing list. The respondents were working in the following countries:

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

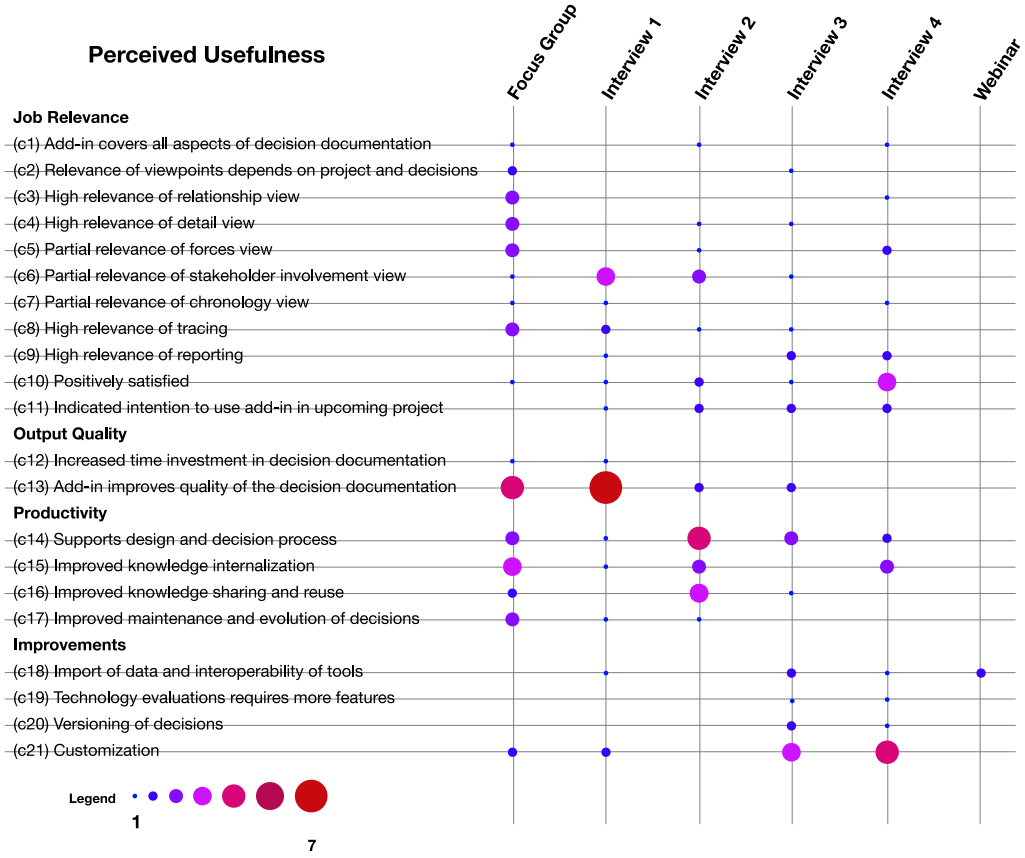

Fig. 9. Matrix showing frequency of incidents per data source and concepts related to perceived usefulness.

Germany (4), United States (3), Sweden (2), Switzerland (1) and Italy (1). On average, they had 8.1 years[7] of experience in software architecture and 16.23 years[8] of experience in software engineering. Only one of the participants stated that he has not been involved in architectural decision-making. Moreover, we asked them how often they consume architectural decision documentation. Four answered weekly, five stated on a monthly-basis and two of them on a yearly basis. Furthermore, we asked how often they produce decision documentation. Five answered monthly, three yearly and two stated that they have never documented decisions so far. One participant did not answer this question.

3.6. Exploratory case study

An earlier version of Decision Architect has been evaluated in a previous case study (Manteuffel et al., 2014) with the goal of an early exploration of the advantages and limitations of the add-in. Furthermore, the study examined the status-quo of decision documentation at ABB and investigated the ideal architecture decision documentation tool from the perspective of architects, a hypothetical tool that is considered optimal by software architects.

We conducted five evaluation sessions with architects from different business units of ABB. All five participants had substantial practical experience. The participants had between 10 and 24 years of experience in software development and 3–18 years of experience in software architecture.

The study found that awareness for decision documentation is increasing at ABB, but still several barriers exist that limit the use of decisions in practice. Such aspects include a lack of tool-support, the absence of a standardized format and only little guidance on how to utilize decision documentation. We have seen that for some architects, the current approach of documenting decisions does not provide enough direct benefits for their project, compared to the required effort of documentation. The approach towards decision-making is comparable between the business units and is characterized by an intensive phase of preparation. With respect to the ideal decision documentation tool, we found that the ideal tool should be lean and flexible, offer an efficient way of documenting decisions, and should be compatible with existing development processes. Besides documenting decisions, the architects found it very important that the tool interfaces with existing tools, exports stakeholder-specific reports, and provides the possibility to share and reuse decisions.

The preliminary evaluation showed that the architects perceive the add-in very positively. Having different viewpoints for different concerns was appreciated, especially since there was no agreement on the usefulness and relevance of individual viewpoints. Suggestions for improvement included a clearer separation between the problem, alternatives and outcome of a decision.

4. Results

In this section we present the results of our study. The results of the qualitative analysis are organized according to the research questions. For each research question, the associated concepts are presented in a frequency matrix (cf. Figs. 9, 12, and 14). The matrix shows the number of coded incidents per data source for a given concept. According to 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. The presented matrices provide further insights about how the evidence is grounded in data and thus, allows evaluating the presented results and putting them into context. However, neither the frequency of a concept nor its distribution over multiple

[7] Minimum 2, maximum 16, standard deviation 3.73.
[8] Minimum 4, maximum 30, standard deviation 7.58.

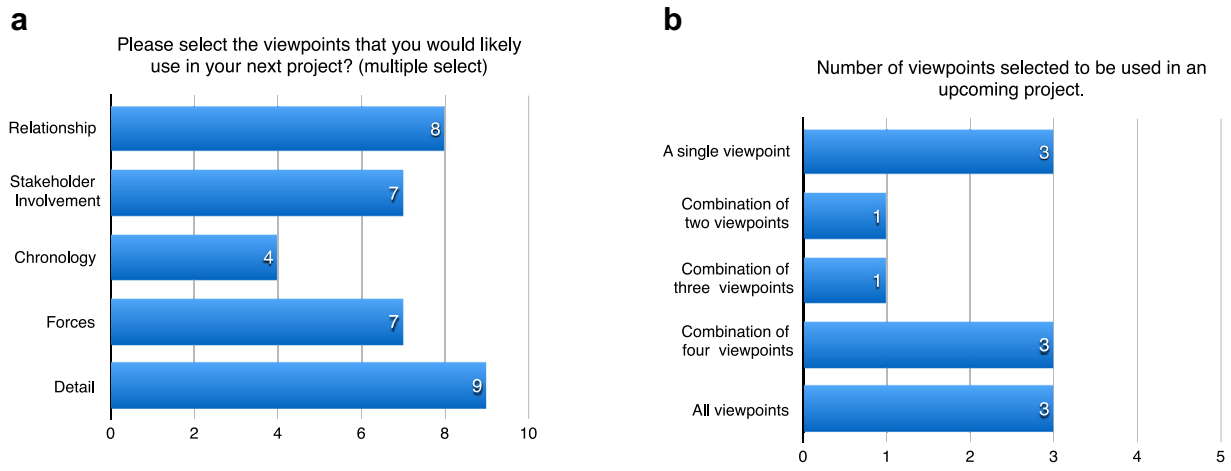

Fig. 10. Webinar results: use of viewpoints (a) and number of viewpoints (b).

data sources necessarily allows conclusions about its importance or significance with respect to the research questions.

4.1. RQ1 – perceived usefulness

The concepts related to perceived usefulness of the add-in were grouped into four categories, as shown in Fig. 9. The first category examines the job relevance of the add-in's features for architects. The second category investigates the quality of the documentation that is produced by the add-in. The third category analyses to what extent and how the add-in influences the productivity of architects. The fourth category presents requested improvements and desired features.

4.1.1. Job relevance

The responses obtained indicate that the add-in covers all important aspects of decision documentation (c1), which can be interpreted that all types of architecture decisions can be expressed and documented in Decision Architect. *"I think conceptually there were not many things that I would like to express and I could not."* and *"[B]asically everything is targeted that is needed for a decision documentation."*

However, the relevance of viewpoints and features depends among others on the nature of the project and the kind of decisions that need to be made (c2). This is also illustrated by the fact that both the relationship viewpoint and detail viewpoint were generally considered to have a high job relevance (c3, c4) by all participants, while the remaining three viewpoints were only partially considered relevant for architects and would only be used in special occasions (c5, c6, c7). Both reporting (c9) and tracing (c8) were considered to be an important addition and highly relevant for their work. *"Following trace links ...helps you to find places where it gets inconsistent and such things"*

The detail viewpoint was considered useful because it resembles the participants' current way of decision documentation and because it provides a textual description of the rationale. (*"I like the detail viewpoint that is more or less the same as we are using with the y-approach."*). The relationship viewpoint was valued for providing a quick overview of decisions. One of participants stated that *"sometimes its enough to look at the relations to understand the rationale behind a decision"*. The stakeholder viewpoint was considered to be relevant for decisions that involve many stakeholders and to keep track of opposing preferences of decisions. *"In general I think I like this view. ...I wouldn't use it for every decision but for some of the decisions when there was a lot of discussion, it helps a little bit."*. The forces viewpoints' advantage is capturing the results of extensive evaluations (*"If you really do an extensive evaluation of a technology, ...you can use the forces view."*) while the chronology viewpoint was considered useful for understanding *"...how the decision process overall evolved"* or how one particular topic evolved.

With respect to the answers obtained after the webinar, Fig. 10(a) shows that the detail and the relationship viewpoint were also considered as the viewpoints most likely to be used in an upcoming project. The forces and stakeholder involvement viewpoint were selected seven times, while the chronology viewpoint was only chosen by four participants. Fig. 10(b) shows that most participants selected four or more viewpoints to be likely used in an upcoming project. However, three out of eleven would only use one of the five decision viewpoints. Fig.10(a) and (b) further supports the modularity concept of Decision Architect, which encourages users to only choose the viewpoints that are relevant in their context depending on decision-related stakeholder concern. Choosing only a subset of viewpoints allows users to balance decision documentation quality (e.g. completeness) and documentation effort.

The responses obtained in the interview and focus group indicated a positive assessment of the add-in by the architects (c10). This is also supported by the fact that all architects indicated their intention to use or to further evaluate the add-in in an upcoming project (c11). According to Fig. 11(a), the results of the webinar also show satisfaction with the features of the add-in. Seven out of 11 respondents rated the tool as likely or very likely to satisfy their needs for decision documentation. Fig. 11(b) shows that six respondents have at least a strong intention to use the tool in an upcoming project.

4.1.2. Output quality

With regards to the primary output of the add-in, the subjects indicated that they will produce more decision documentation because of the add-in. Although, the add-in is not regarded as a time saver but rather as an incentive to invest more time into decision documentation (c12). *"I would just create more documentation because I have this tool."* Moreover, the interviews showed that the tool also improves the overall quality of the decision documentation in three ways (c13). The definition of viewpoints and the framework provide means for standardization. The traces to other architectural elements enrich the documentation and help to keep the documentation consistent. Finally, the architects capture aspects of decisions that they did not consider before, such as dependencies between decisions or stakeholders and their involvement.

4.1.3. Productivity

The third category examines how the add-in affects productivity. According to the participants, the add-in supports the design and decision process in various ways (c14). It was reported that the

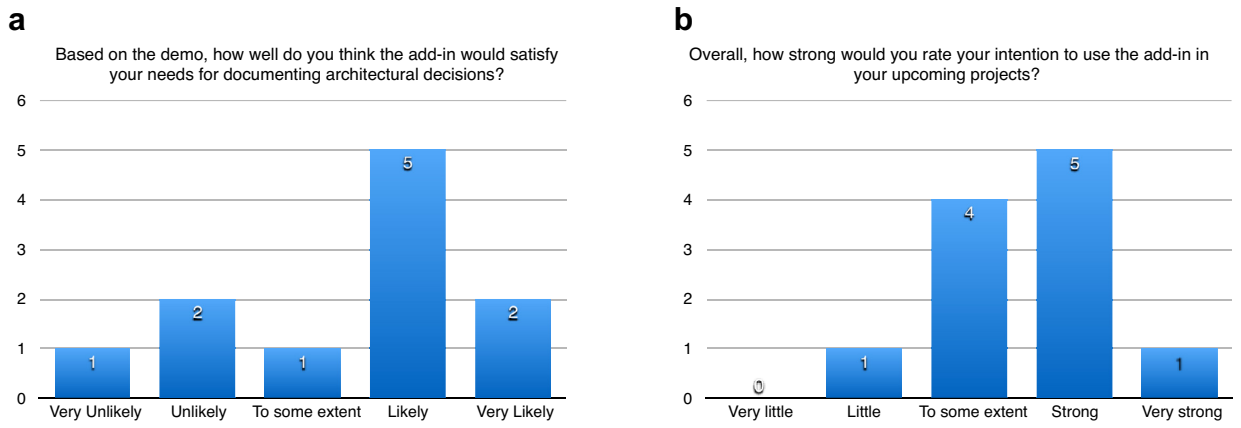

Fig. 11. Webinar results: satisfaction (a) and intention to use (b).

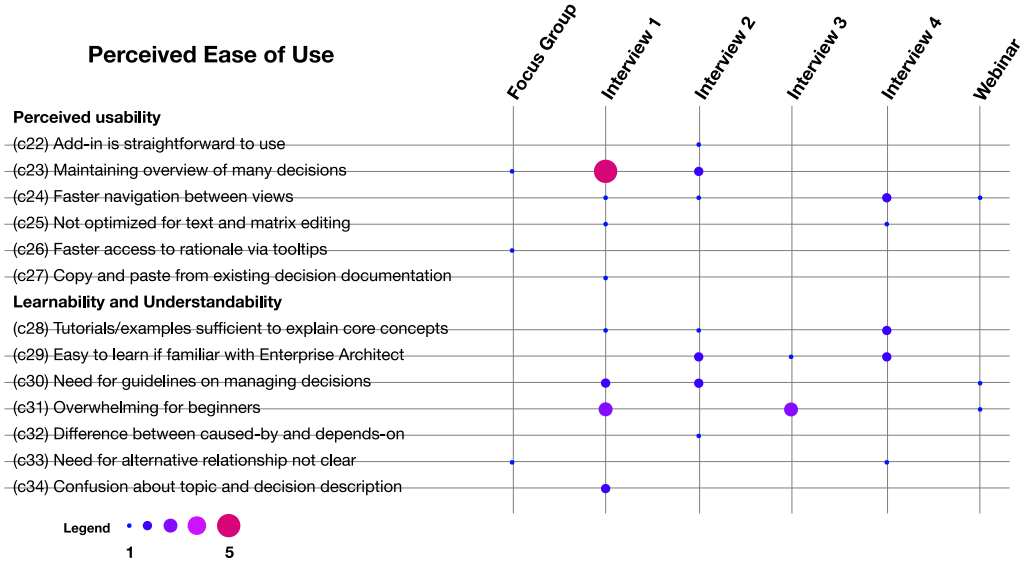

Fig. 12. Concept frequency matrix for RQ2 perceived ease-of-use.

integration of decision documentation into an architectural modeling tool makes it easier to keep track of important thoughts and rationale, for example, by creating placeholder decisions that can be extended with details when the design activity has been completed. *"...I did some conceptual things in the architecture and then I noted in my head that I have to document that [...]. Sometimes I forgot it. [...] Now, I just add the element and I can say directly what was the idea. It saves time because of that."* Furthermore, the integration reduces the risk of losing focus when switching between a design application and a documentation application. *"The main factor that helps me adopting is the fact that it is an Enterprise Architect plugin. So that I don't have to use another tool for all my decisions but I can do it directly where I spend most of my design work. That's a really big effect for me."* It can be stated that the add-in brings the documentation activity closer to the architectural design activity. Moreover, the add-in improves knowledge internalization (c15), knowledge sharing and reuse (c16), as well as maintenance and evolution of decisions (c17). Decisions are presented in smaller chunks via standardized viewpoints that focus on a particular concerns. According to the participants these factors make documentation easier digestible compared to long reports and they allow faster exploration of architectural knowledge.

4.1.4. Improvements

The participants also mentioned features that would further improve their productivity with regards to decision documentation. This includes better tool interoperability, for example by providing the possibility to import data, such as requirements, legacy documentation, or technology evaluations (c18); to have a revision history of decisions that goes beyond the history of decisions states (19); and to have more elaborate formatting features in the forces view, such as conditional formatting or grouping of forces (20). Finally, customization is an important topic (c21). The participants often expressed the need to document additional information, such as priority of forces, confidence factors of decisions, or additional relationships between decisions. *"The option to change names or relationships would allow introducing company policy. That would be interesting for us."* This means that the add-in should offer more customization possibilities and a greater flexibility to express individual concepts, for example, it proposed to rename the term "concerns" to "expectations" in order to convey a more positive way of thinking.

4.1.5. Analysis of decision documentation

In this section we compare the decision views that were created by the participants during the study. The way how the viewpoints, decisions and relationships have been used provide additional insights into the usefulness of the DA. Appendix B shows relationship views that are exemplary for each participant. Appendix A summarizes the participants's decision documents. We can only report on two examples due to confidentiality reasons.

Both architects primarily used the detail and relationship viewpoint. The other viewpoints were only used in a few occasions. Comparing the documentation of the first and second participant,

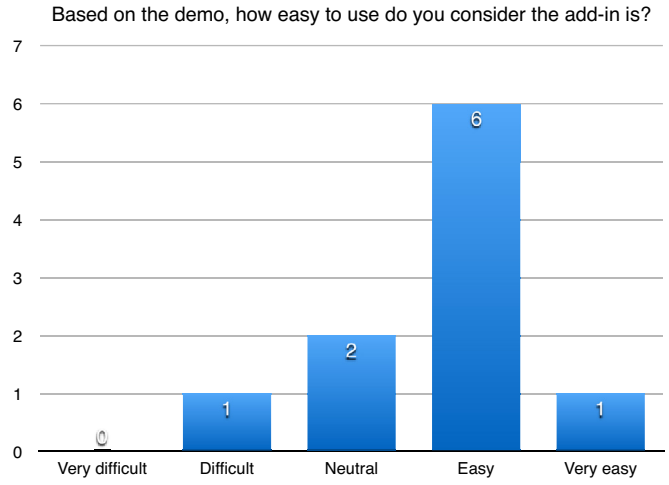

Fig. 13. Questionnaire results: Ease-of-use.

it is notable that both architects used the relationship viewpoint to organize decisions according to subject area, e.g., deployment, data management, performance, communication, security, etc. However, there were considerable differences in how they modeled the relationship views.

The first architect used the topic as a container for the decision alternatives, while the second one connected topics and decisions via relationships. This customization allowed him to model relationships between topics, for example, to indicate the cause for a topic. As motivation for this modeling style, he said: "[F]or most of our decisions we first define a decision point. So we came from a top-down approach, defining the decision point and at some point in time we analyze the actual decisions". The observed differences in modeling the relationship view illustrate how DA is able to support both documentation approaches; after the architecture has been modeled; and during forward engineering before or while modeling the architecture. More importantly, it is another indicator for the need of flexible and descriptive decision documentation approaches.

4.2. RQ2 – perceived ease-of-use

As shown in Fig. 12, the concepts related to perceived ease-of-use were grouped into two categories: perceived usability as well as learnability and understandability. The first category discusses the usability of the tool, which mainly includes concepts related to user interactions. The second category investigates how much effort and which information is required to be able to use the add-in and to what extent the concepts of the add-in are understandable.

4.2.1. Perceived usability

The participants were quite familiar with Enterprise Architect and since the add-in follows the common user interface conventions of EA, we did not receive many comments with regards to usability (c22). The answers obtained in the webinar questionnaire also indicate that the add-in is not difficult to use (cf. Fig. 13). The subjects indicated that the possibility to easily explore the decision documentation and get fast access to the rationale is an important concern. They suggested that the add-in should provide more shortcuts between views to allow faster navigation (c24) and to use tooltips to provide quick access to information (c26). In the add-in, some shortcuts were implemented but as the validation shows there is room for improvement. Another usability impediment is related to maintaining overview of large decision models (c23), which is a pitfall commonly associated with graphical modeling approaches. Furthermore, it was mentioned that the add-in is not optimized for processing text (detail view) or modifying tables (forces view) and thus, does not provide the same user experience as a word processor or spreadsheet application (c26). However, it was considered positive that it is possible to copy and paste text from existing decision documents without losing the formatting (c27).

4.2.2. Learnability and understandability

The learning curve is considered to be low under the premise that the user is already familiar with Enterprise Architect (c29). Since the add-in follows the user interface conventions of Enterprise Architect, the main burden is understanding the purpose of the viewpoints and the meaning of the elements and relationships. It was estimated that it does not take more than a day to be fluently able to use the add-in. If the user is not familiar with EA, it was estimated that it will take approximately three days. With regards to the provided help material, the architects considered the provided tutorials and examples as sufficient to understand the basic features and concepts of the add-in (c28). However, the risk was articulated that the different viewpoints, relationships, etc. can be considered as overwhelming in the beginning (c31), especially for architects that are new to the idea of decision documentation. It was suggested to come up with a beginner mode that offers a simplified user interface that hides information. This user interface could be gradually extended towards an expert mode, which offers the full functionality. This underlines the need for guidelines and best practices (c30). It was noted that an explanation of features is not enough but in order to unlock the full benefits and utility of the add-in, it is necessary to also have guidelines that explain how to create and structure a view in order to gain the most benefits for the development process. These guidelines should also include best practices, for example, that describe how to create overviews or how to manage large numbers of decisions.

In general, the viewpoints seem to be easily understandable but it must be noted that all subjects already familiarized themselves with the add-in in the first case study. However, there were some discussions on the interpretation of relationships, i.e. difference between depends on and caused by(c32), and some questions with regards to modeling of alternatives and topics (c33, c34).

4.3. RQ3 – contextual factors

The last research questions investigates contextual factors that influence the intention to use the tool. Fig. 14 shows the frequency of concepts per data source for this research question.

The fact that Decision Architect has been developed as an add-in rather than a standalone application reduces the barrier of adopting the tool (c35). *"The main factor that helps me adopting, is the fact that it is an Enterprise Architect plugin. So that I don't have to use another tool for all my decisions but I can do it where I spend most of my design work, that's a really big effect for me."*. On the other hand, if Enterprise Architect is not being used, the chances that Decision Architect will be used is much lower. Coupling the tool to a platform can increase adoption but only if this platform is being widely used. We have also seen in the interviews that the preferred style of working (c36) influences how the tool is being evaluated. We identified two dimensions related to the individual's preferred style of working. First, the individual's preference about using graphical modeling and manipulating information via point and click. Second, how often Enterprise Architect is being used. For example, one participant indicated that he only uses Enterprise Architect at the end of the process to document the final results while others use it constantly throughout the development process.

Furthermore, legal (c40) and privacy concerns (c39) were also identified as contextual factors. Legal concerns were related to issues resulting from the explicit documentation of risks, uncertainties, implications, or low confidences. Privacy and confidential concerns are,

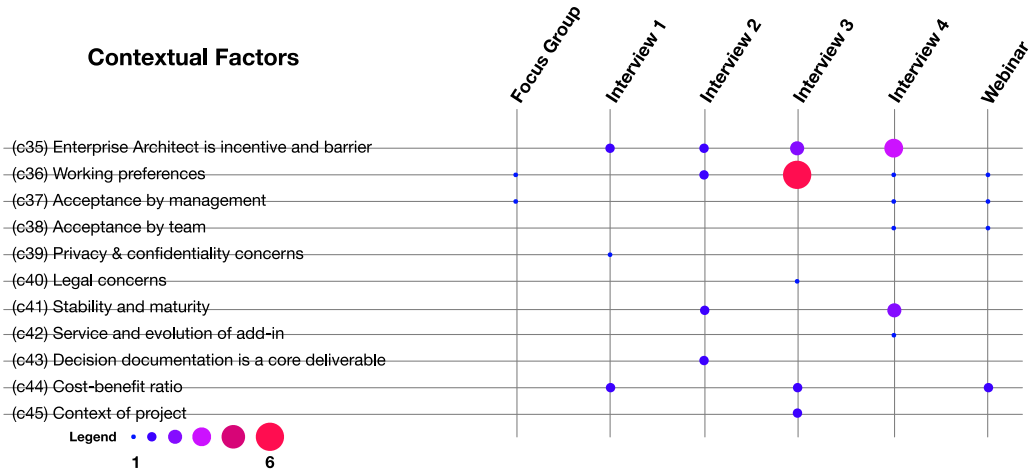

Fig. 14. Concept frequency matrix for RQ3 contextual factors.

for example, related to the explicit traceability of stakeholder's actions introduced by the stakeholder involvement viewpoint.

Moreover, the decision to use a tool in a corporate environment requires the acceptance of the architect's team (c38) as well as the acceptance of management (c37). This also requires that the costs and benefits of using the tool need to be explicit (c44), especially since the benefits of decision documentation are often not immediately visible. *"I need to better understand the value it will bring me"*. An additional factor that is relevant particularly in a commercial environment is that the tool is stable and mature (c41), and that maintenance and continuous evolution are ensured (c42)

The project and its context is also considered an influencing factor (c45), especially if the project ask for a decision document as core deliverable(c43). *"Then it heavily depends on the projects that we have. What is the scope of the project? What are the goals? Who are the stakeholders?"*.

4.4. Summary

With respect to RQ1, the results indicate that the architects perceive the add-in as useful for their work. The features provided are relevant with respect to decision documentation and the tool is able to capture all aspects of decisions. The results show that the add-in improves the quality of decision documentation and benefits the productivity of architects. Better tool-interoperability and support for customization are suggested as improvements.

The results for RQ2 show that architects perceive the ease-of-use as acceptable, although the overall usability of the add-in is limited by Enterprise Architect. We found that architects would prefer faster access to information, for example via tooltips, and additional shortcuts between views. Decision Architect has a low learning curve and its concepts are easily understandable. However architects asked for better guidelines and best practices.

Various contextual factors that further influence the adoption of the add-in in a commercial environment were identified in the results of RQ3. These include, acceptance by peers, continuous maintenance and evolution, overall maturity of the tool, privacy and legal concerns, as well as the context of the project. A factor that largely influences the intention to use is the architect's preferred style of working.

5. Validity

The validity of a study is an indicator of the trustworthiness of the results and to what extent the results are true and not biased by the researcher's subjective point of view (Runeson et al., 2012). *Internal validity* does not apply because we do not examine causalities in our data (Yin, 2008).

Construct validity reflects to what extent the operational measures that are studied really represent what was intended to be studied. To ensure that our operational measures are suitable, we established a research protocol that was reviewed by two researcher with extensive experience in conducting case study research. Both are co-authors of this paper. We created a case study database that contains all collected data both raw and coded, so that the analysis can be verified and traced.

External validity reflects the extent of generalizability of the findings, and to what extent the findings are of interest to other people outside the investigated case. Statistically representative samples can typically not be achieved in case studies (Cavaye, 1996) and thus, the emphasis is usually put on analytical generalization that explains why the findings are representative for other cases with common characteristics (Runeson et al., 2012). The participants of the study were professional software architects with multiple-years of experience in architecting software systems. Moreover, ABB and the studied business units are representative for companies developing software intensive systems. Therefore, we argue that the subject population in the study is representative for the larger population of experienced software architects that have a similar approach to architecting and decision-making as the study participants. There is a risk that the cultural background influenced the results since all participants were German. However, since ABB is an international corporation and since the participants were working in multinational projects in Europe and Asia, we argue that the impact on the results is less substantial. Furthermore, the webinar provided us with perspectives from architects outside of Germany.

Reliability is concerned with the question to what extent the study results are dependent on the specific researchers. We used multiple sources of data (interviews, questionnaires, documentation analysis, and focus groups) to limit the effects of one interpretation of a single data source (Runeson et al., 2012). We also took different perspectives into account and investigated differences between projects and business units, which improves the strength of our conclusions. Furthermore, we addressed reliability by defining a rigorous study protocol and interview guides Runeson et al. (2012). We conducted one pilot-study to train the interviewers and to identify problems with the questions (Runeson et al., 2012). We mitigated the risk of biasing our findings towards a positive outcome by asking open-ended questions and also asked the participant to motivate their answers. However, the risk of a researcher bias also affects analysis. Therefore, the analysis was conducted by two researchers and the findings were discussed with a third researcher, who did not participate in the data

collection, to get an unbiased perspective. All researchers involved in the analysis are co-authors of this paper.

6. Discussion

Based on the results discussed in Section 4, we provide a general discussion of our findings. Furthermore, we discuss implications for researchers and practitioners including lessons learned from this research-industry collaboration.

6.1. General discussion of results

The decision documentation tool presented in this paper aimed at industrial applicability by combining an academic approach that goes beyond the state of the art, with the practical constraints of a large company. The goal of the study was to show to what extent and why architects would adopt this tool.

With respect to perceived usefulness, the results indicate a high job relevance of the tool as well as quality improvements and benefits for productivity. However, the study also showed that the usefulness does not only depend on the main capabilities of the tool but also on supplemental features, such as formatting, copy and paste, importing, or reporting. Overall, the results indicate that the tool is perceived as useful for documenting architecture decisions from the perspective of software architects.

On the question of perceived ease-of-use, this study found that the tool usability is acceptable but can be improved. However, several studies (Davis, 1989; Venkatesh et al., 2003) showed that usefulness has a significantly greater impact on predicting future use than ease-of-use, as long as the benefits from using the tool are substantial. The study also identified factors that will increase or decrease the likelihood of adoption. Of course, many of those factors are not constrained to decision documentation tools but apply in a broader context. However, we believe that it is important to identify the factors that are particularly relevant for such a tool and determine the context under which it works best.

The findings of the this study are consistent with those of the first case study. The first study also identified that architects were generally positive about the tooling, but had different emphasis regarding the different viewpoints. Furthermore, the topic element which was introduced as a direct result of the first case study was well-perceived by the architects (cf. Section 4.1.5).

6.2. Customization and extensibility

Another important finding was that even within the same company there are large variations in project types and working methods that influence the adoption of a decision documentation tool. The results show that the relevance of viewpoints highly depends on the project. For example, the relationship viewpoint was regarded useful in projects with large technology stacks. The forces viewpoint was considered useful for green field architecting with lots of technology evaluations. One interpretation could be that there seems to be difference whether the viewpoint is relevant for the design process or mainly for documentation, like the chronology and stakeholder viewpoint.

This finding has important implications for developing decision documentation tools. It shows these kind of tools need to be versatile and allow for customizations and extensions. These results match those observed in the first case study, which also identified that the ideal decision documentation approach should be flexible.

6.3. Guidelines

On the question of learnability and understandability, this study found that the help material was sufficient to understand the features and concepts of the add-in. Nevertheless, one important finding is that the practitioners also asked for better guidelines that go beyond the functional explanation of the tool.

Those guidelines should focus on the process of creating decision views and how the tool can be used in a particular situation in order to gain the maximum benefits. They could provide indicators on what kind of information and how much information should be documented in order to gain certain benefits. For example, doing change impact analysis would require to create decision relationship views. Furthermore, the guidelines could provide best practices depending on the type of project (e.g., multi-disciplinary development, architecture prototype, or legacy system evolution) or application domains.

Additionally, guidelines could present approaches that vary in terms of agility versus formality. Since agile teams are usually smaller and focus on direct face-to-face communication, an agile decision documentation approach would favor a knowledge personalization strategy, using the stakeholder involvement viewpoint, over a comprehensive documentation of detail viewpoints. Additionally, the documentation could be further simplified, for example, by reducing the number of decisions states and decision relationships.

7. Implications for practitioners and researchers

Looking back at this industry-research collaboration project, we learned a number of generic lessons that are helpful in similar situations.

Industrial collaboration increases relevance: The close collaboration with a partner from industry ensured that the developed tool does not only work under the idealized conditions and assumptions of researchers but also provides real practical value. Such a project provides the opportunity to test a theory under practice conditions, in this case, the decision documentation framework. In these kind of projects, researcher play two roles that need to be strictly separated. On the one hand, they are tool vendors that want to provide a functional tool. On the other hand, they are researchers that want to investigate a problem by providing tool-support.

Design data collection with practitioners in mind: The selection and design of data collection plays an important role in the success of the study and the quality of collected data. Since the architects were participating on voluntary basis they had to make sure that the participation in the study did not conflict with their job obligations. Our experience in this project shows that it is important to keep the effort for practitioners as low as possible and to clearly communicate the required time investment in advance. We also found that supervised data collection methods were better perceived as unsupervised techniques that are not performed together with the researcher. For example, the participants were often willing to spend more time on the interviews then it was agreed upon in advance. In contrast, the online questionnaires were less well-perceived. Also, the self-taught use of the tool in order to document decisions did not work as good as the tool session in the first study, which was conducted together with researchers.

Webinar helps to advertise project as well as to collect additional data: The webinar allowed us to reach a wider audience within the company and thus, to inform more software architects about the tool and the related research. Especially, the open discussion after the webinar was a good opportunity to exchange experiences and to get further insights into the constraints under which software architects work.

In addition to these lessons learned, Koziolek and Goldschmidt (2013) discuss additional lessons learned from the perspective of ABB in an earlier report about the Decision Architect project.

8. Conclusions

In this paper we presented Decision Architect, an architecture decision documentation tool for Enterprise Architect that focusses on industrial applicability. In an industrial case study we demonstrated that Decision Architect is applicable in industrial projects and meets the expectations and demands of the industry.

The findings of the study show that Decision Architect is perceived as useful by practitioners. The offered features have a high relevance for software architects and the tool is able to capture all aspects of decisions. The study shows that DA increases the quality of decision documentation, and it improves productivity of architects in various ways. The research has also shown that Decision Architect is easy-to-use, with the exception that better guidelines and best practices are needed. Additionally, the study presented contextual factors that influence the adoption of the tool in an industrial environment. Especially the fact that the intention to use such a tool is largely influences by the preferred style of working suggest that decision documentation tools must be adaptable and offer a high flexibility in order to gain a wide acceptance. Decision Architect has been evaluated in two case-studies, in which we recorded 15.5 h of audio resulting from nine interviews and one focus group. Taken together, these results suggest that the presented decision documentation approach is applicable in industry.

However, the study also revealed many possibilities for improving Decision Architect. This includes the option for further customizations , the support of structured decision templates for automatic information retrieval, as well as further increasing tool interoperability. Nevertheless, the study also revealed several directions for future research:

(i) What is the impact of Decision Architect on a system's quality?
(ii) How does Decision Architect compare to existing AKM research tools?
(iii) What guidelines and best practices are needed for decision documentation?

Additionally, there is a need to perform further validation studies of Decision Architect in other companies operating in different domains and countries.

Overall, this study provided further insights on how decision documentation is perceived in industry and the reasons why decisions are rarely explicitly documented in practice. The results of this project will help future tool development efforts to better understand the constraints under which software architects work and their needs and expectations with respect to decision documentation.

Acknowledgment

We would like to thank all participants of the case study. Especially, we would like to thank Spyros Ioakeimidis, Antonis Gkortzis, Mark Hoekstra, Marc Holterman, and Hessel van Apeldoorn for their contribution to the add-in as well as Zengyang Li and Werner Buck for piloting our study. This research has been partially sponsored by the ABB Software Research Grant Program and the ITEA2 Project 11013 PROMES.

Appendix A. Summary of decision documentation

Table A.3 provides high-level statistics about the use of elements and viewpoints. Since DA automatically creates a detail view for every

Table A3
Statistics of the decision views created by the study participants.

	Participant 1	Participant 2
Elements		
Decisions	33	38
Idea	2	16
Discarded	20	5
Tentative	3	8
Decided	8	6
Challenged	0	3
Topic	10	12
Decisions per topic (avg/min/max)	3.3 / 2 / 7	2.9 / 2 / 5
Views		
Detail	24	38
Relationship	6	6
Forces	1	1
Chronology	1	0
Stakeholder involvement	1	2

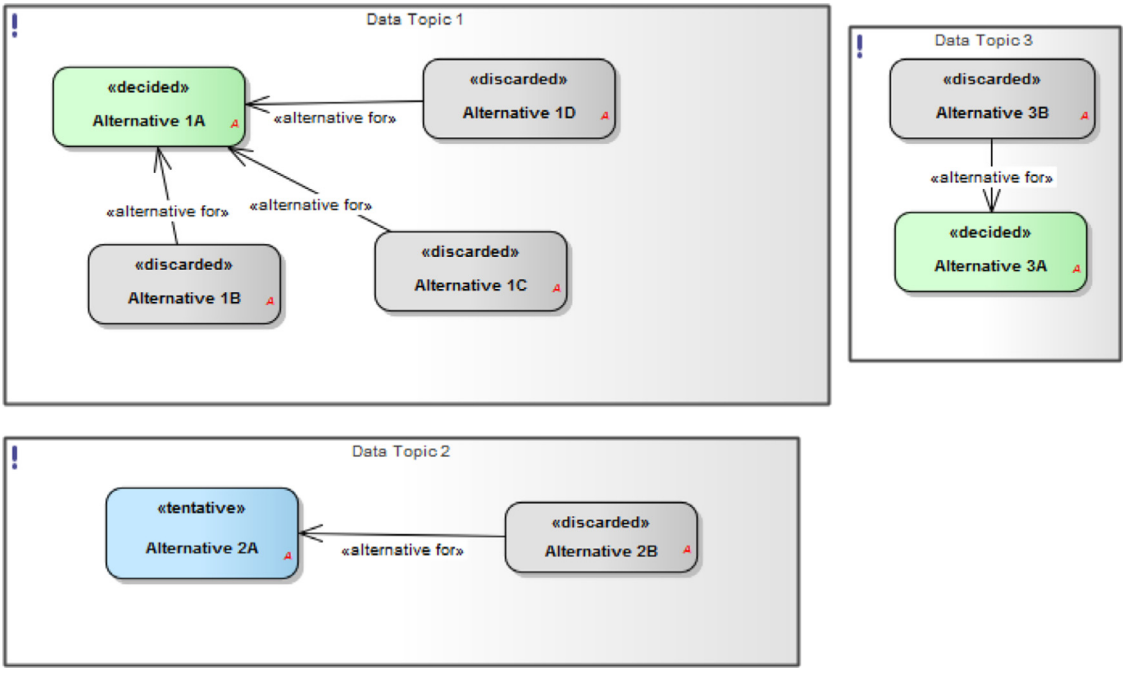

Fig. B15. Relationship view modeled by the first participant. The view has been obfuscated.

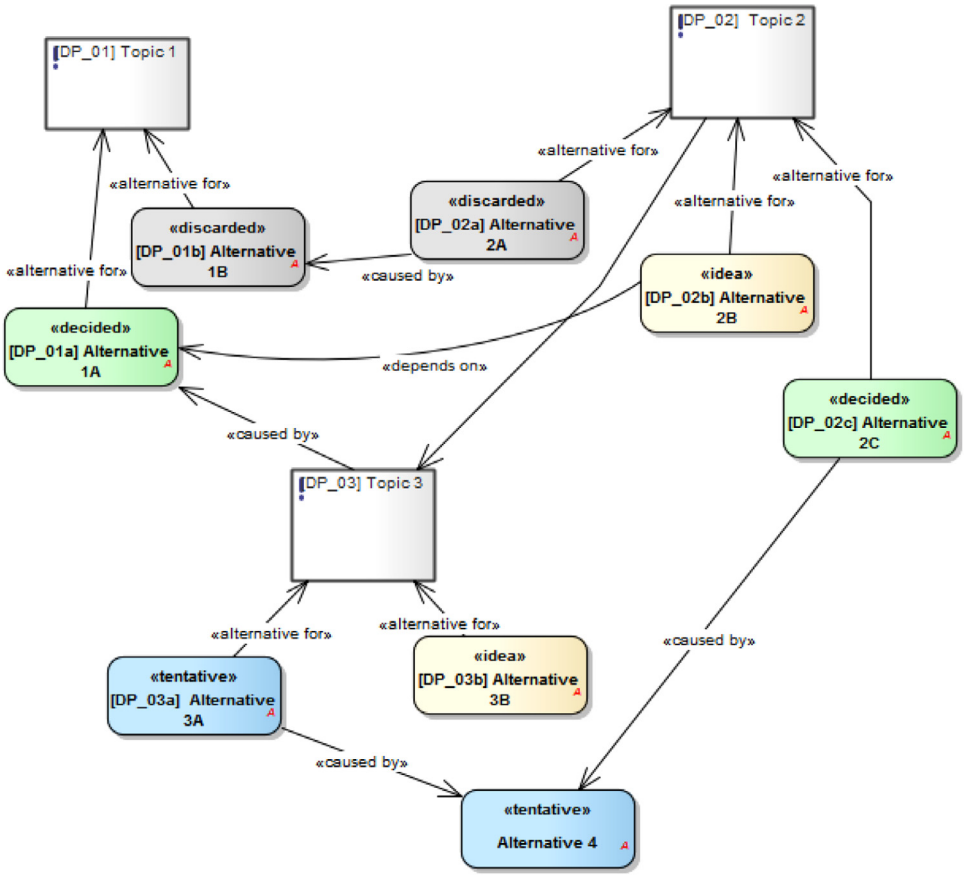

Fig. B16. Relationship view created by the second participant. Besides decision relationship, this view also contains relationships between topic elements. The view has been obfuscated.

decision by default, we only counted those detail views in which the participant provided additional information.

Appendix B. Examples of decision documentation

See Figs. B.15 and B.16.

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. doi:10.1007/s10664-010-9152-6.

Babar, M.A., Gorton, I., 2007. A tool for managing software architecture knowledge. In: Sharing and Reusing Architectural Knowledge – Architecture, Rationale, and Design Intent, SHARK/ADI'07:ICSE Workshops 2007. Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture ICSE Workshop Second Work, p. 11.

Basili, V.R., Caldiera, G., Rombach, H.D., 1994. The goal question metric approach. Encycl. Softw. Eng.. John Wiley & Sons, pp. 528–532.

Bosch, J., 2004. Software architecture: the next step. Softw. Archit., 3047. Springer, pp. 194–199.

Bratthall, L., Johansson, E., Regnell, B., 2000. Is a design rationale vital when predicting change impact? – a controlled experiment on software architecture evolution. In: Proceedings of PROFES 2000 – Product-Focussed Software Process Improvement, pp. 126–139.

Capilla, R., 2009. Embedded design rationale in software architecture. In: Proceedings of the 2009 Joint Workshop of IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. IEEE, pp. 305–308. doi:10.1109/WICSA.2009.5290826.

Capilla, R., Nava, F., Montes, J., Carrillo, C., 2008. ADDSS: architecture design decision support system tool. In: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. IEEE, pp. 487–488. doi:10.1109/ASE.2008.84.

Cavaye, A.L.M., 1996. Case study research: a multi-faceted research approach for IS. Inf. Syst. J. 6 (3), 227–242. doi:10.1111/j.1365-2575.1996.tb00015.x.

Creswell, J.W., Miller, D.L., 2000. Determining validity in qualitative inquiry. Theory Pract. 39 (3), 124–130.

Davis, F.D., 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. 13 (3), 319. doi:10.2307/249008.

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

Farenhorst, R., Lago, P., van Vliet, H., 2007. Effective tool support for architectural knowledge sharing BT. In: Software Architecture. Springer, Berlin, Heidelberg, pp. 123–138.

Farenhorst, R., van Vliet, H., 2009. Understanding how to support architects in sharing knowledge. In: Proceedings of the 2009 ICSE Workshop on SHAring and Reusing Architectural Knowledge, SHARK'09, pp. 17–24.

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

Jansen, A., Bosch, J., 2005. Software architecture as a set of architectural design decisions. In: Proceedings of the 5th Joint Workshop of IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, WICSA'05. IEEE Computer Society, pp. 109–120.

Jansen, A., Van Der Ven, J.S., Avgeriou, P., Hammer, D.K., 2007. Tool support for architectural decisions. In: Proceedings of the 2007 Joint Workshop of IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, WICSA'07, pp. 1–10.

Koziolek, H., Goldschmidt, T., 2013. Tool-driven technology transfer to support software architecture decisions. In: Proceedings of SE14, pp. 1–6.

Kruchten, P., 1995. Architectural blueprints—the "4+ 1" view model of software architecture. IEEE Softw. 12 (6), 42–50.

Kruchten, P., Capilla, R., Dueñas, J.C., 2009. The decision view's role in software architecture practice. IEEE Softw. 26 (2), 36–42.

Kruchten, P., Lago, P., van Vliet, H., 2006. Building up and reasoning about architectural knowledge. In: Proceedings of the Second International Conference on Quality of Software Architecture, QoSA'06. Springer-Verlag.

Kruchten, P.B., 1995. 4+1 view model of architecture. IEEE Softw. 12 (6), 42–50. doi:10.1109/52.469759.

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. doi:10.1007/s10664-005-1290-x.

Liang, P., Avgeriou, P., 2009. Tools and technologies for architecture knowledge management. In: Liang, P., Avgeriou, P. (Eds.), Software Architecture Knowledge Management. Springer, Berlin, Heidelberg, pp. 91–111. doi:10.1007/978-3-642-02374-3_6.

Liang, P., Jansen, A., Avgeriou, P., 2010. Collaborative software architecting through knowledge sharing. Collab. Softw. Eng. 343–367.
Manteuffel, C., Tofan, D., Koziolek, H., Goldschmidt, T., Avgeriou, P., 2014. Industrial implementation of a documentation framework for architectural decisions. In: Proceedings of the 2014 Joint Workshop of IEEE/IFIP Conference on Software Architecture (WICSA), pp. 225–234.
Nowak, M., Pautasso, C., 2013. Team situational awareness and architectural decision making with the software architecture warehouse. In: Proceedings of the 7th European Conference ECSA, July 1–5, 2013, Montpellier, France. In: Volume 7957 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 146–161. doi:10.1007/978-3-642-39031-9_13.
Perry, D.E., Wolf, A.L., 1992. Foundations for the study of software architecture. ACM SIGSOFT Softw. Eng. Notes 17 (4), 40–52.
Runeson, P., Höst, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, Inc., Hoboken, New Jersey doi:10.1002/9781118181034.
Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. 25 (4), 557–572.
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.
Tang, A., Babar, M.A., Gorton, I., Han, J., 2006. A survey of architecture design rationale. J. Syst. Softw. 79 (12), 1792–1804. doi:10.1016/j.jss.2006.04.029.
Tang, A., Jin, Y., Han, J., 2007. A rationale-based architecture model for design traceability and reasoning. J. Syst. Softw. 80 (6), 918–934. doi:10.1016/j.jss.2006.08.040.
Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. Software, IEEE 22 (2), 19–27.
Van Der Ven, J.S., Jansen, A., Avgeriou, P., Hammer, D.K., 2006. Using architectural decisions. In: Proceedings of the 2nd International Conference on Quality of Software Architecture. Karlsruhe University Press.
van Heesch, U., Avgeriou, P., 2011. Mature architecting – a survey about the reasoning process of professional architects. In: Proceedings of the 2011 9th Joint Workshop of IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, pp. 260–269.
van Heesch, U., Avgeriou, P., Hilliard, R., 2012a. A documentation framework for architecture decisions. J. Syst. Softw. 85 (4), 795–820.
van Heesch, U., Avgeriou, P., Hilliard, R., 2012b. Forces on architecture decisions – a viewpoint. In: Proceedings of the 2012 Joint Workshop of IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, pp. 101–110.
van Heesch, U., Avgeriou, P., Tang, A., 2013. Does decision documentation help junior designers rationalize their decisions? A comparative multiple-case study. J. Syst. Softw. 86 (6), 1545–1565.
Venkatesh, V., Bala, H., 2008. Technology acceptance model 3 and a research agenda on interventions. Decis. Sci. 39 (2), 273–315. doi:10.1111/j.1540-5915.2008.00192.x.
Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D., 2003. User acceptance of information technology: toward a unified view. MIS Q. 27 (3), 425–478.
Yin, R.K., 2008. Case Study Research: Design and Methods, 4th Sage Publications Ltd..
Zimmermann, O., 2012. Making Architectural Knowledge Sustainable – Industrial Practice Report and Outlook (Invited IEEE Software talk at SEI Saturn 2012).

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. His primary research focus are software architecture decisions, particularly architecture decision management in the domain of embedded systems. He is a member of the ACM and the IEEE Computer Society.

Dan Tofan did his M.Sc. studies in computer science at the Gheorghe Asachi Technical University of Iasi in Romania, and Ph.D. studies in software architecture at the University of Groningen in Netherlands. Previously, he worked in various software engineering roles at a leading provider of telecom billing solutions. Currently, he is a hands-on software engineer with software architecture and project management responsibilities for a portfolio of internal core software applications at a global leader in bio-analytical testing. Dan's research interests include software architecture, architectural decisions, and empirical software engineering.

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, evolution, patterns and link to requirements.

Heiko Koziolek received the Ph.D. degree in computer science from the University of Oldenburg, Germany, in 2008. He is a Principal Scientist and Global Research Area Coordinator at ABB Corporate Research in Ladenburg, Germany. His contributions received Best Paper Awards at WICSA, ICPE, SPLC, and WOSP. He has served as co-program chair of the software architecture conferences QoSA and SATURN, as well as reviewer for several conferences and journals, including ICSE and IEEE TSE. His research interests include performance engineering and software architecture. He is a member of the ACM, the IEEE Computer Society, and the German Computer Science Society.

Thomas Goldschmidt is a principal scientist at ABB Corporate Research in Ladenburg, Germany. He joined ABB in 2010 and since then has been leading and working on research in the area of model-driven development, domain-specific languages as well as software architecture in the automation domain. Additionally, he has been lead architect in a research project on bringing automation software to the cloud. In 2010, he received a Ph.D. from Karlsruhe Institute of Technology for his thesis entitled "View-Based Textual Modelling".