The Journal of Systems and Software 124 (2017) 56–81

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Platform design space exploration using architecture decision viewpoints–A longitudinal study

U. van Heesch[a,*], A. Jansen[b], H. Pei-Breivold[c], P. Avgeriou[d], C. Manteuffel[d]

[a] HAN University of Applied Sciences, Arnhem, The Netherlands
[b] Philips Innovation Services, Eindhoven, The Netherlands
[c] ABB Corporate Research, Västerås, Sweden
[d] University of Groningen, Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 27 May 2016
Revised 26 September 2016
Accepted 31 October 2016
Available online 10 November 2016

Keywords:
Design space exploration
Architecture decision viewpoints
Technical action research

ABSTRACT

Design space exploration is the simultaneous analysis of problem and solution spaces for a specific domain or application scope. Performing this activity as part of the architectural design is beneficial, especially for software platforms, which are shared across organizations. Exploring the design space of software platforms in a multi-product and multi-domain context is not trivial, and only few methods exist to support this activity systematically.

This paper reports on a longitudinal technical action research (TAR) study conducted to adapt and evaluate architecture decision viewpoints for supporting platform design space exploration. The study was conducted in the context of an ABB project, which was performed to explore the design space for a common software platform for mobile device support in several product-specific software platforms at ABB.

The results indicate that the adapted decision viewpoints are well suitable for dealing with diverging stakeholder concerns, evaluating technological alternatives and uncovering relationships between decisions to be made.

1. Introduction

The essence of architecting is making design decisions that satisfy the most important stakeholder concerns (Kruchten, 2004; Jansen and Bosch, 2005; Perry and Wolf, 1992). When making such decisions, software architects struggle on the one hand with a large number of constraining factors (e.g., quality attribute requirements, interface requirements, constraints, company culture, politics, and other stakeholder expectations) and on the other hand with numerous design options (e.g., off-the-shelf products, software patterns, tactics, idioms, best practices, frameworks, and libraries). In the software engineering literature, the former is referred to as problem space, whereas the latter is called solution space (Shekaran et al., 1994). The union of problem and solution spaces is the design space. A design space explicitly represents design options within a particular domain or project, and the reasons (i.e. problem space elements) for choosing between these options (Zdun, 2007). The term design space exploration refers to the in-depth analysis of the design space of a specific domain or application scope.

Especially in software platform engineering[1] projects, it is important to thoroughly explore the design space when making decisions. In such projects, architects deal with several stakeholder groups and varying (importance of) concerns, which may require different design choices. As opposed to single applications, which need to satisfy a distinct set of concerns, software platforms are used as a basis for several application systems (Taudes et al., 2000). Consequently, they need to support different groups of functionality and they may need to support the construction of application systems with different quality attribute requirements.

We propose an approach for software platform design space exploration based on architecture decision viewpoints. In our previ-

* Corresponding author.
E-mail addresses: uwe@vanheesch.net (U. van Heesch), anton.jansen@philips.com (A. Jansen), hongyu.pei-breivold@se.abb.com (H. Pei-Breivold), paris@cs.rug.nl (P. Avgeriou), c.manteuffel@rug.nl (C. Manteuffel).

http://dx.doi.org/10.1016/j.jss.2016.10.031
0164-1212/

[1] We define software platform as a software environment that is used as a basis for building or running multiple software applications. The platform comprises functionality that is commonly used or shared between those applications. Our definition includes, but is not limited to, operating systems, class libraries, compilers, virtual machines, middleware platforms and frameworks, and meta programming systems.

ous work, we developed architecture decision viewpoints to support a systematic reasoning process (Heesch et al., 2013) and to document architecture decisions and the rationale behind those decisions as an integral part of architecture descriptions (van Heesch et al., 2012a; 2012b; Manteuffel et al., 2016). In this paper, we present an empirical study conducted to adapt and validate architecture decision viewpoints for supporting platform design space exploration. The differences of the study presented in this paper with our previous work are three-fold: 1) we focus on design space exploration 2) we deal with software platforms instead of single application systems 3) we conduct a longitudinal study. A longitudinal study was required, because the results of a platform design space exploration project can only fully be validated after the first application systems are built on top of the platform. Using the goal definition technique suggested in Basili et al. (1994), the goal of the study presented in this paper is formulated as follows:

> Analyze a software platform design space exploration process **for the purpose of** adapting and evaluating architecture decision viewpoints from van Heesch et al. (2012a,b), **with respect to** the level of provided support for platform design space exploration.

The study was conducted as a technical action research (TAR) (Wieringa and Moralı, 2012) project with ABB between January 2012 and February 2015. As part of the project, ABB supported by the researchers adapted and used decision viewpoints to explore and document the design space for a common mobile infrastructure solution for several ABB software platforms, which involved two different business units and multiple sites spread over two continents. The adapted viewpoints were validated in two stages: a first round of validation with the stakeholders was done immediately after the results of the design space exploration were presented to the stakeholders; a second round of validation was done two years afterwards to understand how the projects actually benefited from the design space description, to re-assess the recommendations derived from the design space description, and to elicit additional long-term concerns from the stakeholders.

Technical action research is a rather new research method in the empirical software engineering domain. Therefore, in Section 3, we extensively introduce and discuss the technical action research method. To give the reader a comprehensive impression of how the study was conducted, we also extensively describe the study design and execution, as well as limitations of the study using a reporting structure introduced by Wieringa (2014).

The rest of this paper is organized as follows. In Section 2, we present background information on software architecture, architecture decision viewpoints and design space exploration. Section 3 presents the design of the technical action research study conducted. In Section 4, we present the results of our analysis. In the next section, we present related work. Finally, Section 5 concludes and outlines areas for future work.

2. Background

This section presents background information on software architecture, architectural decision viewpoints and design space exploration.

2.1. Architecture decision viewpoints

Software architecture comprises the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" (ISO/IEC/IEEE, 2011). Apart from the structural aspects of a system, many authors have stressed the importance of also preserving the rationale that went into the system design (Perry and Wolf, 1992; Kruchten, 1995; Garlan et al., 1997).

Building up on the idea to preserve the rationale behind architectural design by treating architectural decisions as first-class entities in architecture representation (Bosch, 2004; Tyree and Akerman, 2005; Jansen, 2008), van Heesch et al. developed a viewpoint-based framework for architecture decisions (van Heesch et al., 2012a; 2012b) using the conventions of ISO/IEC/IEEE 42010 (ISO/IEC/IEEE, 2011). Views of the framework integrate seamlessly into other viewpoint-based architecture descriptions. The framework contains five viewpoints (van Heesch et al., 2012a; 2012b), each of which satisfying different sets of decision-related stakeholder concerns. An explicit definition of stakeholders' concerns in an architecture description is an integral part of viewpoint definitions in ISO/IEC/IEEE 42010. In the following, the viewpoints are briefly explained:

- **Decision forces viewpoint** The forces viewpoint provides means for explicitly analyzing design options in the context of all architecturally significant decision forces. A *decision force* is "any aspect of an architectural problem arising in the system or its environment (operational, development, business, organizational, political, economic, legal, regulatory, ecological, social, etc.), to be considered when choosing among the available decision alternatives (van Heesch et al., 2012b)".
- **Decision relationship viewpoint** Relationship views are used to make connections between architecture decisions explicit. Various types of connections are supported ,e.g. *depends on, caused by*, or *is alternative to*.
- **Decision chronology viewpoint** Chronological views show the evolution of architecture decisions over time. Apart from decisions made, the view also preserves information about design options that were considered but not decided, and other decisions that were discarded over time.
- **Decision stakeholder involvement viewpoint** It describes the roles of specific stakeholders in the decision-making process, capturing which stakeholders proposed, confirmed, or validated specific decisions.
- **Decision detail viewpoint** Decision detail views holistically describe single architecture decisions, including a comprehensive description of the chosen architectural solution and the rationale for choosing this solution.

Originally, the main perceived benefit of decision documentation was preservation of rationale behind decisions made in the past (Bosch, 2004). However, our previous studies have shown that decision view creation also has an immediate benefit for the architects during the design process, as they support a systematic reasoning process (Heesch et al., 2013). In a comparative multiple-case study with junior designers, van Heesch et al. showed that particularly the decision forces viewpoint and the decision relationship viewpoint provide strong support for the decision making process (Heesch et al., 2013). The other viewpoints mainly serve documentation purposes. Their main benefit is the preservation and communication of architectural knowledge, rather than support for the design process.

2.2. Design space exploration

Design space exploration has been traditionally based on determining the problems to be solved, potential solutions to those problems, and the ways to gauge whether the latter solve the former. A typical well-known example of such approaches is Questions-Options-Criteria (QOC) (MacLean et al., 1991), where

questions represent the problems, options map to candidate solutions while criteria are used to determine how well the options fare with respect to the questions at hand. While QOC was originally proposed for the Human-Computer Interaction field, it has been used extensively across the Software Engineering lifecycle. Another popular approach from the field of Software Measurement is Goal Questions Metric (GQM) (Basili et al., 1994). The difference with QOC is that GQM does not look for candidate solutions; it only states the problem (in terms of the goal), then asks a number of questions to refine the goal and finally measures the object of study (e.g. product or process) according to the metrics. In essence, GQM allows for problem space exploration according to the goals and questions but it also provides the metrics to be used for assessing the candidate solutions.

In the software architecture literature, design space exploration is treated as inherent to the activity of architecting. In the well-accepted model of Hofmeister et al. (2007), architecting is comprised of architectural analysis (identifying architecturally-significant requirements), synthesis (finding candidate solutions) and evaluation (assessing the degree to which candidate solutions match the requirements). These three activities map to the exploration of the problem space, the solution space, and the combined design space respectively. Architectural analysis searches through the problem space to come up with potential functional and non-functional requirements, assumptions, risks, stakeholder concerns. Architectural synthesis explores the landscape of potential software patterns, technologies, or any other types of design solutions based on experience, design theory or best practices. Finally architectural evaluation looks at both the problem and the solution space checking the extent to which design solutions match the stated problems. All three architectural activities are used iteratively making compromises in both, problem and solution spaces, until an optimal result is reached. The latter process is also referred to as the Twin Peaks model (Nuseibeh, 2001), which proposes an iterative and incremental exploration of both problem and solution space that allows for the architecture to be shaped by the requirements and the other way round.

During architectural analysis, problem space exploration often takes place through stakeholder workshops. For example, the Quality Attribute Workshop (Barbacci et al., 2003) helps to elicit quality requirements through a discussion among project stakeholders regarding high-level scenarios of system execution and maintenance/evolution. During architectural synthesis, solution space exploration mostly happens by consulting repositories of reusable solutions. For example, the Attribute-Driven Design method (Bass et al., 2012) mandates looking for tactics from established catalogs that satisfy particular quality attributes and patterns that are compatible with these tactics. Finally, during architectural evaluation, design space exploration takes place mostly through expert judgement on the match between problem and solution space. For example, the Decision-Centric Architecture Review (DCAR) method (van Heesch et al., 2014) requests participating stakeholders to weigh reasons in favor against reasons against a decision and eventually vote for the "goodness" of the decisions.

There has been a fair amount of literature applying the aforementioned generic approaches (QOC or GQM or one of their variants), to the three architecting activities of analysis, synthesis and evaluation. Usually such works make use of reusable knowledge both in the problem space (e.g. problem frames or goal models) and the solution space (e.g. software patterns or decision models). As an example, Gross and Yu (2001) aim at supporting design reasoning by supporting designers to select design patterns based on their impact on non-functional requirements (NFR). To this end, they link design patterns to non-functional requirements (represented as goals and related in graphs), thus creating explicit traces between the problem space (goals) and the solution space (patterns). Similarly, Zdun (2007) proposes to explore design spaces made of software patterns by formalizing pattern relationships and further annotating them with effects on quality goals. This allows architects or designers to parse through the design space by navigating from pattern to pattern until a combination of selected patterns optimally meets the quality goals.

3. Study design and execution

In this section, we present the study goal and research questions, the selected research approach, and the data collection and analysis procedures.

3.1. Research goal and research questions

As described in Section 1, the overall study goal is defined as:

> Analyze a software platform design space exploration process **for the purpose of** adapting and evaluating architecture decisions viewpoints from van Heesch et al. (2012a); 2012b), **with respect to** the level of provided support for platform design space exploration.

In the remainder of this article, we adopt Wieringa's classification of research questions into *practical problems* and *knowledge questions* (Wieringa, 2009). Wieringa defines a practical problem as "a difference between the way the world is experienced by stakeholders and the way they would like it (the world) to be"; a knowledge question is defined as "a difference between current knowledge of stakeholders about the world and what they would like to know". The research goal is broken down into the following knowledge questions and practical problems:

- RQ1 (Knowledge question): What are typical stakeholders for software platform design space exploration (producers and consumers)?
- RQ2 (Knowledge question): What are the stakeholders' concerns[2] to be framed by viewpoints used for design space exploration?
- RQ3 (Practical problem): What needs to be adapted in decision viewpoints to support the design space exploration concerns identified in RQ 2?
- RQ4 (Knowledge question): In how far do the adapted viewpoints satisfy the design space exploration concerns identified in RQ2?
- RQ5 (Knowledge question): What is the long-term value of the views of the adapted viewpoints as experienced by the stakeholders in the course of the project?

RQ1 aims at finding typical stakeholder groups of design space exploration, i.e. all persons or organizational groups who influence the design space exploration or who are impacted by the results of the design space exploration. The stakeholders actively involved in the exploration of the design space are referred to as *producers*; the stakeholders who use the results of the exploration are called *consumers* (Tang et al., 2010). RQ2 is about identifying the concerns of the stakeholders identified in RQ1. RQ3 is a practical problem, concerned with the adaptation of existing decision viewpoints to optimally satisfy the concerns identified in RQ2. As explained above, the need for adaptations was expected, because the viewpoints were originally designed for single applications rather than software platforms. RQ4 is about validating the changes, i.e. finding out in how far the adapted viewpoints can satisfy the extended set of stakeholder concerns. RQ5 is intended to understand

[2] The term concern here refers to a stakeholder's interest in an architectural view that is part of an architecture description, rather than a concern in the properties of a software system.

the value of the views for the software projects that used the results of the design space exploration as a basis for application development.

In the next section, we present the research design applied to answer the research questions.

3.2. Research approach

The study was conducted using TAR, as conceptualized by Wieringa and Moralı (2012). TAR is used to validate an artifact that was created to solve a class of problems by using it to solve a concrete problem instance and drawing conclusions with respect to the entire problem class (Seddon and Scheepers, 2012; Wieringa, 2014). A central idea behind TAR is that validation of an artifact does not take place under idealized conditions (which would be the case in a controlled experiment for instance), but in the concrete context of practice. TAR provides a practical value to the organization in which it is used, while allowing researchers to gain new theoretical knowledge (Sjoberg et al., 2007). Therefore, it has higher practical relevance than empirical research methods that idealise conditions to improve the observation of phenomena in isolation (Wieringa and Moralı, 2012; Baskerville and Wood-Harper, 1996; Sjoberg et al., 2007).

Technical action research combines aspects from design science (Hevner et al., 2004; Wieringa, 2009) with action research (Reason and Bradbury, 2001; Baskerville, 1999; McIntyre, 2008). Design science aims at creating or improving artifacts that serve human purposes (March and Smith, 1995; Wieringa, 2009). It makes a generalizability claim in that the artifact is associated with a class of problems, as opposed to action research, which is concerned with gaining knowledge by solving concrete practical problems in organizations (Reason and Bradbury, 2001) by means of active collaboration between an organization and external researchers. Technical action research combines those two aspects in that an artifact is brought into a concrete industrial context to solve a relevant practical problem. In contrast to a case study, which also studies phenomena in an industrial context, (technical) action research is not purely observational. In TAR, the researchers are actively involved in the process of applying the artifact in practice while observing it (Runeson and Höst, 2009; Baskerville and Wood-Harper, 1996; McKay and Marshall, 2001). The main difference between TAR and other forms of action research is that TAR aims at testing a particular (technical) artefact, whereas other action research projects are problem-driven without the pre-determined focus on a particular artifact (Wieringa and Moralı, 2012).

This means that technical action research projects actually comprise two projects: the first project is a research project, in which researchers try to answer research questions (typically the validation of a previously designed artifact); the other project is an engineering project, in which the artifact is applied (and possibly adapted) to solve a concrete problem in an organization. Wieringa refers to the former project as empirical cycle and to the latter as client engineering cycle (Wieringa and Moralı, 2012). Apart from helping the organization with a concrete problem, the client cycle serves the researchers as a means to answer research questions concerned with the validation of the applied artifact.

Fig. 1 shows the empirical and engineering cycles used in TAR. The research cycle starts with an investigation of the research problem. The cycle continues with the research design and validation. The next activity in the research cycle is the research execution. As Fig. 1 shows, in TAR the client engineering cycle is intertwined with the empirical cycle, i.e. the client engineering activities are part of the empirical research design and validation and the research execution is completely performed as part of the client engineering cycle. The client engineering cycle comprises its own problem investigation, which is concerned with the concrete problem that the client organization wants to solve using the artifact under study in the research cycle. The client cycle continues with treatment design, i.e. how will the artifact be introduced, used, and adapted in the organization (Wieringa, 2014). After the validation, the treatments are implemented and finally evaluated in the client context. The implementation evaluation concerns the suitability of the artifact to address the client problem. The entire client cycle can be executed in multiple iterations. When the client activities are finished, the data analysis is conducted as a final activity in the research cycle.

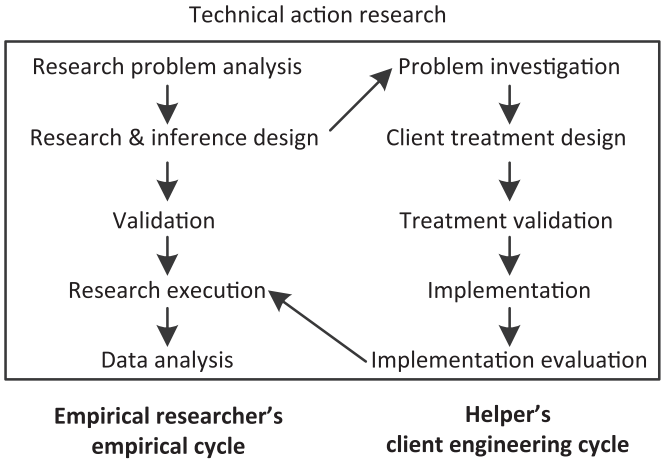

Fig. 1. Empirical and client engineering cycles in TAR (excerpt from Wieringa, 2014).

In absence of established reporting templates for technical action research, we use the activities depicted in Fig. 1 as reporting structure for the remainder of this section. Each activity is described in a separate subsection. For a more detailed description of these activities, we refer the reader to Wieringa (2014).

3.3. Research problem analysis

In this section, we describe the unit of study and the research questions associated with it. The unit of study is the use of decision viewpoints for supporting design space exploration for mobile device support in several software platforms at ABB. As described above, decision viewpoints were originally designed for documenting architecture decisions made in a software project. Their use in this context has been validated in previous empirical studies (van Heesch et al., 2012a; Heesch et al., 2013; van Heesch et al., 2012b). The use of decision viewpoints for exploring the problem and solution spaces for software platforms is a new use case. Previous studies demonstrated the general applicability of the decision viewpoints supporting a rational exploration of the design space for single software applications. Their use in a platform development context introduces several additional challenges, e.g., dealing with multiple coherent stakeholder groups, different sets of functional requirements, different priorities and importance of decision forces, and a wider solution space that is not constrained by the specifics of a single software project. It was thus assumed that the viewpoints were not usable out-of-the-box, but that they had to be adapted and extended during the project to optimally support the additional or different stakeholder concerns. In the next section, we specify the research goal and questions.

3.4. Research & inference design

In technical action research, the research design typically comprises (Wieringa, 2009; Tripp, 2005):

- The acquisition of a client company that is willing to apply the artefact under study
- An agreement on an improvement goal for the client engineering cycle
- An agreement on the researchers' roles and responsibilities in the client company
- And an agreement on the data collection activities.

The following subsections describe those aspects in detail.

3.4.1. Cient acquisition and improvement goal

As opposed to many other (technical) action research projects, in which it is common that the researchers actively acquire clients, in this project, the client approached the researchers. In 2012, ABB started a company-internal project to explore the design space for a common mobile infrastructure platform for several ABB software products. ABB contacted the researchers, because they had made good experiences with decision-centric architecting and they found the decision viewpoints from van Heesch et al. (2012a) a promising basis for supporting design space exploration. Ultimately, the design space exploration was intended to contribute to ABB's strategy for expanding parts of the product portfolio with mobile solutions.

The researchers and ABB agreed on the following client improvement goal statement:

"Enhance ABB's understanding of the design space of mobile applications and their related backend systems in the context of several product platforms at ABB[3] by documenting architecture alternatives, their trade-offs, and an evaluation and recommendation based on architecture decisions viewpoints and architectural prototypes."

3.4.2. Researchers roles and responsibilities

In TAR projects, it is important to clarify the researchers' roles and responsibilities upfront, because they have two roles that need to be distinguished. On the one hand, they act as empirical researchers who aim at answering knowledge questions related to the unit of study. On the other hand, they get actively involved in the client engineering cycle as helpers, who support the client in using and adapting the artefact under study in the client-specific context (Wieringa, 2014).

This study is a joint effort of three researchers and two client delegates. Two of the researchers were involved as helpers in the client engineering process, whereas the third researcher was involved in the data analysis and interpretation only. As suggested by Siebers for empirical research in software engineering (Sieber, 2001), all researchers signed a non-disclosure agreement to protect ABB's intellectual property. In the following, we describe the researchers' responsibilities in their roles as researchers and as client helpers in the context of this research project:

Researchers Design the empirical study including all treatments; obtain information required to answer the research questions posed in Section 3.1, analyze the data and draw conclusions with respect to the validity of the research results for the wider problem class (in contrast to the client-specific problem instance).

Client helpers Support ABB in the use and adaptation of decision viewpoints for design space exploration. Provide support exclusively in the form of training and guidance on how to use decision viewpoints and in developing further the used viewpoint specifications. The actual design space exploration, including the creation of all decision views that correspond to the viewpoint specifications is exclusively done by ABB.

[3] The names of the concrete platforms were removed for confidentiality reasons

3.4.3. Data collection activities

In this section, we describe the data collection activities used during the study. Each activity is briefly described and linked to the research questions posed in Section 3.1.

All activities belonging to the client engineering cycle took place at the ABB site in Västerås, Sweden. The engineering cycle involved two distinct business units, each having multiple sites, product lines, and presence on two continents. The external researchers were not present on-site. All communication and data exchange between the researchers and ABB took place remotely via e-mail, video conferencing, and file sharing.

The data collection took place in two phases. In the first phase, between January and December 2012, the actual design space exploration was performed. The phase finished with a presentation of the design space to all company stakeholders at ABB. Between 2013 and 2014, ABB used the results of the analysis as input for implementing mobile strategies in different software platforms. The second phase started in October 2014. The researchers and ABB delegates met again to plan and conduct a retrospective data collection, required to answer RQ5 about the long-term value and sustainability of the views created in phase one.

Table 1 lists the data collection activities and the assigned research questions. The retrospective interviews with the company stakeholders were conducted in phase two, all other data was collected in phase one.

3.5. Research design validation

In this section, we identify potential validity threats and explain how they were addressed in the study.

An important validity aspect of technical action research projects is the suitability of the project domain. According to Baskerville and Wood-Harper (1996), a domain is suited for action research, if the researcher can get actively involved in the project with a mutual benefit for both, researchers and organization. Moreover, it must be possible to immediately apply knowledge gained during the project in a cyclic process linking theory and practice. This project meets all of those criteria. The researchers were actively involved in the project for adapting decision viewpoints to the special requirements of platform design space exploration. The process was iterative and knowledge gained during one iteration was immediately used in subsequent iterations to improve the usability of the decision viewpoints. The outcome of the study was used by ABB for sub-sequent software projects while the gained knowledge went into an improved decision viewpoint specification.

Following Easterbrook et al., the two main criteria for judging action research designs are authenticity of the problem and authenticity of the knowledge outcomes for the participants (Easterbrook et al., 2008). In this study, the researchers were approached with a problem by the industrial partner ABB, thus the authenticity of the problem cannot be questioned, because the problem is a real-life industrial problem that was defined by the industrial partner prior to the study. To increase the authenticity of the knowledge outcome, the study was designed as a long-term study. There was a period of approximately two years, in which ABB used the study results (i.e. the design space exploration documented in architectural views) before the retrospective validation was conducted. We therefore assume a high level of knowledge outcome authenticity.

While the active involvement of the researchers in the client engineering cycle is a key characteristic of all action research projects, it also casts a major threat to the generalizability of the results. If the artifact under study is primarily used or applied by the researchers, then one might argue that the researchers can use the artifact in a way nobody else can. Wieringa proposes to

Table 1
Used data collection activities.

Activity	Description	RQs
Field notes (Shull et al., 2008)	The researchers took field notes during the entire course of the project. In these notes, they wrote down observations and conjectures that served as input during the data analysis. Field notes were primarily taken during regular meetings with the ABB delegates and during design sessions, in which viewpoint definitions were adapted.	RQ1, RQ2, RQ3, RQ5
Analysis of work artifacts (Shull et al., 2008)	Relevant work artifacts created as part of the client engineering cycle were provided to the researchers for discussions. This includes the created decision views in different stages, presentation material, office documents, and conceptual sketches. Additionally, the two researchers involved in the client engineering cycle had regular process meetings with the ABB delegates.	RQ1, RQ2, RQ3
Interviews with stakeholders	The ABB delegates regularly interviewed different company stakeholders throughout the project. During these interviews, they presented (intermediate) results to the stakeholders, while at the same time interviewing them about their understanding of the decision views, concerns, questions and suggestions for improvement. The interviews were captured in the form of screen casts and audio records. The researchers did not take part in the interviews, but they created question guides (see Mack et al., 2005 for a description of question guides) together with the ABB delegates upfront. The questions in question guides are not asked directly, but they serve the interviewers as orientation to ensure that all relevant information content is elicited from the interview partners.	RQ1, RQ2, RQ4
Retrospective interviews with company stakeholders	As described above, the retrospective interviews were conducted with the company stakeholders two years after the design space exploration was finished. Like the interviews in phase one, the interviews were collaboratively prepared by the two ABB delegates and the researchers. The retrospective interviews were necessary, because viewpoints for design space exploration cannot completely be validated before they were actually used as input for architectural decisions in software projects.	RQ1, RQ2, RQ4, RQ5

mitigate this threat by teaching others to use the artifact so they can apply it in the client cycle (Wieringa and Moralı, 2012). We adopted this proposal. The client delegates followed an introduction on view definition using the viewpoints in the beginning of the client cycle. All views, created during the study, were exclusively created and refined by ABB delegates. The researchers contributed by refining the viewpoint definitions, which capture specifications and guidelines on how to create views.[4]

Another potential threat to validity in all qualitative studies stems from the phenomenon that interviewees tend to answer questions in a socially desirable way or that researchers interpret ambiguous indicators in a favorable way (Yin, 2009; Wieringa and Moralı, 2012). The former threat can be partially mitigated by having others than the researchers perform the data collection; the latter threat by assigning the data analysis activity to researchers not directly involved in the project. As described above, in this project, all interviews with ABB stakeholders were carried out by the ABB delegates rather than the researchers. This significantly reduces the probability of interviewees giving answers only because they are desirable. The risk of researchers' bias during the data analysis was at least partially mitigated by involving another researcher into the analysis. The coding procedure was conducted by two researchers independently, only the resulting code system and concepts were derived in collaboration (please refer to Section 3.7 for details on the data analysis procedure). This reduces the threat of researcher's bias to a certain extend. However, the threat could not totally be eliminated.

3.6. Research execution–client engineering cycle

In this section, we describe research execution activity, which corresponds to the client engineering cycle, as depicted in Fig. 1.

3.6.1. Client problem investigation

In 2012, ABB started a company-internal project to explore the design space for a common mobile infrastructure platform for several ABB software products. The project was started based on the observation that the second generation of standard mobile platforms provides considerable more processing, data networking, and user interaction capabilities than the first generation and is therefore mature enough to be integrated into ABB's project portfolio. For ABB, an integration of mobile solutions opens up several new use cases for existing products, for instance mobile control apps, maintenance managers, and sales support apps.

Before the project, ABB had no standard way of integrating mobile solutions into their platforms. Their main concerns centered around deployment strategies and facilities, security, product scoping and licensing for the different product portfolios. To address these concerns, ABB wanted to investigate the design space for mobile platforms, which includes an identification of the key stakeholders of the different products, their concerns and interrelationships, an identification of key architectural alternatives, and the investigation of specific technological solutions. ABB decided to use architecture decision viewpoints to support this investigation process and to document the outcome of the design space exploration. As the existing decision viewpoints were not optimally suited for software platforms, the viewpoints had to be adapted and views of the adapted viewpoints had to be created.

3.6.2. Client treatment design

In technical action research, a treatment is an artefact under study interacting with a particular problem context (Wieringa and Moralı, 2012). In this case the artifacts under study are decision viewpoints and the problem context is ABB's design space exploration project for mobile integration. As described in Section 3.5, we intended not to create the decision views ourselves, but to teach the ABB delegates how to create them. The following list describes how decision viewpoints were introduced to ABB and how they were used within the project.

Decision viewpoint specifications ABB was provided with the latest decision viewpoint specifications following the conventions of ISO/IEC/IEEE 42010 (ISO/IEC/IEEE, 2011). Viewpoint specifications describe the concerns framed by the viewpoint, typical stakeholders for these concerns and one or more model kinds, which describe the notations, conventions, and modeling techniques to be used on models of these kinds. Please refer to ISO/IEC/IEEE (2011) for a detailed description on viewpoints, views, model kinds and models.

Introductory presentation on decision viewpoints We introduced the decision viewpoints to the two ABB delegates in an online conference meeting. The presentation covered the viewpoint specifications and examples of views of each

[4] For the difference between views and viewpoints, please refer to (ISO/IEC/IEEE, 2011)

viewpoint. Furthermore, the ABB delegates had the chance to clarify questions and uncertainties regarding the view creation.

Adaptations of viewpoint specifications As mentioned earlier, we anticipated that the viewpoints had to be adapted to optimally support platform design space exploration. Changes to the viewpoints were initiated by the ABB delegates when they observed a limitation of the original viewpoints specification. In such cases, the researchers developed extensions to the viewpoints to overcome those limitations. The adapted viewpoint specifications were then discussed with the ABB delegates. In many cases, this process required multiple iterations.

Reviews of created views The decision views created by ABB were regularly reviewed by the researchers. During these reviews, the researchers focused on syntactical and conceptual correctness of the views rather than checking the domain-specific content.

3.6.3. Treatment validation

In TAR, treatment validation is primarily concerned with estimating the effect of the artefact in the problem context and evaluating whether these effects satisfy the stakeholders' improvement criteria (Wieringa and Moralı, 2012). Apart from that, each treatment could have side-effects that are not desired or even harmful with respect to the client's goals. Therefore, as part of the treatment design validation, the researchers evaluate whether the treatments to be applied are as simple and non-invasive as possible to achieve the client goals, while at the same time reducing side-effects to the best possible degree. Other concerns of treatment design validation are repeatability and ethics (Wieringa, 2014).

In this particular case, the treatment had to be suitable for supporting the design space exploration activities of ABB. As described above, our previous studies had shown that decision viewpoints support several design space exploration activities when being used for single-product development. We therefore expected that the existing viewpoints would show the same positive impact in platform design space exploration. However, platform design space exploration has several additional challenges:

- systematically identify and prioritise diverging stakeholder groups and their concerns in this platform design space exploration
- uncover and resolve conflicting stakeholder concerns and priorities
- understand impacts of those concerns (including conflicting impacts) on candidate architectural solutions
- efficiently present the results of the design space exploration to project stakeholders and to
- specify decisions to be made including decision dependencies (technological and chronological)

We planned to address these challenges by anticipating the need for viewpoint adaptations, as described in the treatment design.

Undesired side effects were for instance an unreasonable increase of effort, researchers' activities causing project delays, or a dependency on the researchers for the creation or consumption of decision views. While the first two treatments (provision of decision viewpoint specification and introductory presentation of decision viewpoints) are uncritical, ABB's dependency on the researchers for updating viewpoint specifications bears a potential risk with respect to project delays. ABB mitigated this risk by planning project activities in a way that the view creation was not on the critical project path. That way, the researchers efforts for adapting viewpoint specifications were performed in parallel to ABB's project activities. The reviews of the created views were also performed offline, thus not binding ABB resources apart from the communication of the review results. The dependency on the researchers for creating or interpreting views was avoided by planning the treatments in a way that views were exclusively created by ABB. This also has a positive impact on the repeatability of the study; only the adaptation of decision viewpoints was done by the researchers, which is a natural trade-off given that the decision viewpoints are the artefact under study. The following measures were taken to mitigate ethical concerns to the best possible degree:

- All participants of the study were fully aware of all treatments and the motivation behind the treatments
- ABB, and all ABB employees who took part in the study were free to stop the study at any point without giving reasons
- The researchers signed a non-disclosure agreement to protect ABB's intellectual property, information integrity, and confidentiality.
- All publications of the study results are approved by ABB

In the next section, we describe the implementation within the client cycle. We mention all activities within the project that have relevance for the research project.

3.6.4. Implementation

As stated above, the goal of the client cycle was exploring the design space for a common mobile platform at ABB, which could be used as basis for mobile application support in several ABB software products.

Among the different mobile applications that are being developed and/or to be developed by different business units (BUs), the project focused on the mobile applications that communicate through the internet with a server controlled by a customer running an ABB PC-based software product. In this setup, the customer's IT organization is responsible for its deployment and operations. Two business units were chosen accordingly for understanding the problem and solution space of mobile applications and their related backend systems.

The two ABB delegates interviewed product managers and RD managers at the BUs to collect information. The objective of the interviews was to understand and collect information about products' mobility development and future strategies. These interviews were recorded, and minutes of the meetings were sent out to interviewees for comments if any. The same general questions were used during each interview, but a certain degree of freedom was allowed for getting the required information. The interviews were semi-structured with open-ended questions. Examples of questions that were used in the interviews are:

- What does the mobile strategy look like?
- Which parts of the product are considered to be expanded with mobile solutions?
- What are the business drivers that motivate the development of mobile solutions?
- What kind of business model do you envision for mobile solutions?
- Which quality attributes (e.g., performance, availability, security, etc.) are in focus?
- What are the business constraints when developing mobile solutions?
- What are the technical constraints when developing mobile solutions?
- What do the deployment and update strategies look like?

The entire question guide for the interviews can be found in Appendix D.

After consolidation of the collected information, seven main categories of requirements were identified for the common mobile

Table 2
Consolidated force priority model (product names obfuscated).

		Priorities					
Code	Categories of requirement	XXX	XXX	XXX	XXX	Avg priority	Prio. variance
1	Integrability	10	4	9,5	12	8,9	11,7
2	Interoperability	20	6	6	10	10,5	43,7
3	Security	15	8	4,5	15	10,6	27,6
4	Usability/user experience	20	7,5	8	10	11,4	34,2
5	Availability	10	7	5,5	10	8,1	5,1
6	Performance	5	7,5	4,5	6	5,8	1,8
7	Compatibility	0	7,5	5,5	3	4	10,5
8	Maintainability	10	4,5	7,5	4	6,5	7,8
9	Reusability	0	4,5	5,5	3	3,3	5,8
10	Reliability	5	6,5	5,5	5	5,5	0,5
11	Flexibility	0	5,5	7,5	3	4	10,5
12	Evolvability	0	5,5	6	3	3,6	7,6
13	Deployability	0	7	7,5	2	4,1	13,7
14	Device management	0	3,5	4,5	2	2,5	3,8
15	Network	0	5,5	5,5	6	4,3	8,1
16	Development constraints	0	5	3,5	5	3,4	5,6
17	Costs	0	5	3,5	2	2,6	4,6

platform, i.e., Integration/Interoperation, Quality Attributes, Device Management, Evolution/Deployment, User Interface, Functionality, and other aspects that concern business model, development efforts, resources and expertise. These categories of requirements pinpoint the key considerations when developing the common mobile platform. The results of the survey were documented in a list of initial force priority models, one model for each software product. At that time the models contained only a list of force candidates mapped to the respective product stakeholders. In the course of the project, these models were iteratively refined. The ABB delegates used MS Excel for creating all forces models. Other architectural views were created using Sparx System's Enterprise Architect.[5]

To better understand the problem space and to facilitate required choices of appropriate technologies and architecture solutions for the common mobile infrastructure, the ABB delegates prepared a questionnaire and let stakeholders provide prioritizations for the elicited requirements. In the questionnaire, the elicited requirements were grouped into 17 aspects in total, including Integrability, Interoperability, Security, Usability/User experience, Availability, Performance, Compatibility, Maintainability, Reusability, Reliability, Flexibility, Evolvability, Deployability, Device management, Network, Development constraints, and Costs.

Some of the aspects contain sub-elements. All the aspects and respective sub-elements (if any) had to be answered on a 7-point Likert scale (1 for the lowest prioritized, 7 for highest prioritized). At the end of the questionnaire, the stakeholders were asked to distribute 100 points among the 17 aspects so as to give us an overview of their relative priorities of the requirements. The stakeholders were also asked if there were any additional requirements that they wanted to add. The intention of the questionnaire was two-fold: (1) Further validate the completeness and correctness of the elicited requirements; and (2) Understand the different levels of importance of the requirements to the stakeholders. This would provide input to the consecutive technology evaluations and the choices of architectural decisions/solutions in the project. The full priority questionnaire can be found in Appendix E.

After this step, the existing force priority models contained product-specific forces and priorities of the product stakeholders. The models were consolidated into one additional force priority model (see Table 2) showing an overview of forces, the priorities of the product stakeholder groups and statistical information about average priority and variance per force.

The client project focused on exploring existing mobile Cross Platform Tools (CPT) in the market. These tools allow building mobile applications for multiple mobile platforms with the same code base. Additionally, they provide deployment, integration and device management functionality. To get a comprehensive overview of existing CPTs, relevant technology trend reports were browsed including Gartner's report (Clark et al., 2012) and Vision Mobile's report (Mobile, 2012). There is a large diversity of proprietary programming languages, frameworks and development tools in the mobile market, which is problematic for application producers who need to support multiple mobile platforms. A cross platform technology that would create mobile applications for multiple platforms using a single code base could save cost, time and development effort. The ABB delegates identified 37 technologies and analyzed them in terms of the previously identified forces. Additionally, technology-specific forces such as supported platform, I/O support, device core API support, integration ability, development environment and deployment, license model and cost were used. To support the analysis, the ABB delegates created one decision forces model per technology. Apart from the forces already identified in the force priority models, the additional technology-specific forces were added to the models. In these models, each impact rating of a force was described with a short statement describing the impact rating in more detail. (Fig. 2)

After gathering the information, the ABB delegates classified the technologies into ten different categories based on the architectural styles they use: (1) Mobile web adaptation server, (2) Native browser, (3) Embedded webkit, (4) Embedded virtual machine, (5) Native application, (6) Embedded remote desktop, (7) Embedded webkit with JavaScript API, (8) Cross compiler, (9) Data driven, (10) Embedded webkit with virtual machine. Table 3 shows an example of an analysis of identified architectural styles, showing a short description of each style with pros and cons. The content from the table is taken from representative forces models for each architectural style.

As a sub-sequent step a consolidated decision forces model was created showing all forces and one representative technology for each architectural style. The forces model was used to understand which technologies were most promising in the project context. Fig. 4 shows an excerpt from this model.

Because the impact of some forces on specific technologies was uncertain at that time, the ABB delegates decided to build small prototypes for the most promising architectural alternatives. Apart from filling some white spots in the force model, the prototyping activity was to used to test the feasibility of the technologies for selected mobile platforms. The implemented scenario was a visualization and acknowledgement functionality for an alarm from an alarm list using one code base. The scope of the prototypes included examining multi-platform UI and live-data stream perspectives.

[5] http://www.sparxsystems.com/products/ea/

	Requirement	[redacted]	Rationale
Interopability	Interface with other applications on device	2	Integration is supported with a configurable application interface that allows drag&drop with external apps and application specific customization, e.g to enable context sensive navigation with other applications
	Access to device capabilities	0	Limited to HTML5 and support of the browser and end device
Usability	User experience	3	There are no limitations and the technology framework is open for implementing as good usability as required
	Visual identity	3	Fully scalable displays adopt to every end device and allow developing any kind of visual identity required
	Ease of adoption	3	Using the preferred web browser of the user enables easy adoption
	Consistent user interface	2	Selectors and other native widgets of the end device are used as far as possible
ecurity	Security	3	End-to-end security is supported with encrypted communication. Data security is based on ACL definitions on object type, property, and instance level. In addition also time based security

Fig. 2. Excerpt from a forces model created for one technological alternative (name of technology obfuscated).

Table 3
Architectural styles identified.

Arch. style	Description	Pros	Pros
Embedded webkit with JavaScript API	This model is a hybrid architecture that includes customized webkit and JavaScript libraries that expose device native API to the SDK.	Rich platform diversity; excellent device hardware accessibility; offline use ability	Still need to write some platform dependent code.
Cross compiler	Utilizes an intermediate language to write app logic and UI; a cross platform compiler compiles the code into different native package for each platform.	Rich platform diversity; excellent performance and user experience; no need to connect to Internet when using the application.	A limited UI collection is supported; requires some twist on the generated code.
Data-driven approach	Server has pre-defined data formats that can be used by UI generator in the local device. UI generator utilizes transmitted data according to UI templates.	Provided templates to facilitate UI development; understandable by business personnel; ability to work offline; ability to access to the device hardware.	A limited UI collection is supported in the templates.

Table 4
Partially obfuscated excerpt from the consolidated decision forces model.

	Requirement	[redacted]	[redacted]	Antenna	Monotouch	[redacted]	[redacted]	[redacted]
Interopability	Interface with other applications on device	2	-2	2	-3	X	R	2
	Access to device capabilities	3	2	2	2	0	3	0
Usability	User experience	1	2	3	3	-1	3	3
	Visual identity	0	3	3	3	-2	3	3
	Ease of adoption	3	2	-1	0	-2	-2	3
	Consistent user interface	3	2	3	-3	3	3	2
Security	Security	0	X	3	2	3	1	3
	Integrate with							

To decide on which technology solution to choose, in addition to prototyping, several key decision points were identified based on conflicting forces resulting from the different the business units' requirements. Some examples are:

- Offline support: The mobile platform should allow for mobile applications to work in an offline mode, allowing for maximum productivity in the case of the absence of a data connection. The mobile platform should also allow for background synchronization once the data connection is re-established.
- Cross-platform support: The technology should support different mobile operating systems and versions, as to decouple the evolution of the mobile application and associated server product from the evolution of the mobile devices and their operating systems.
- Supported form factor: The mobile platform supports large and small form factors (laptop, tablet, phone, etc.) that run on different operating systems.

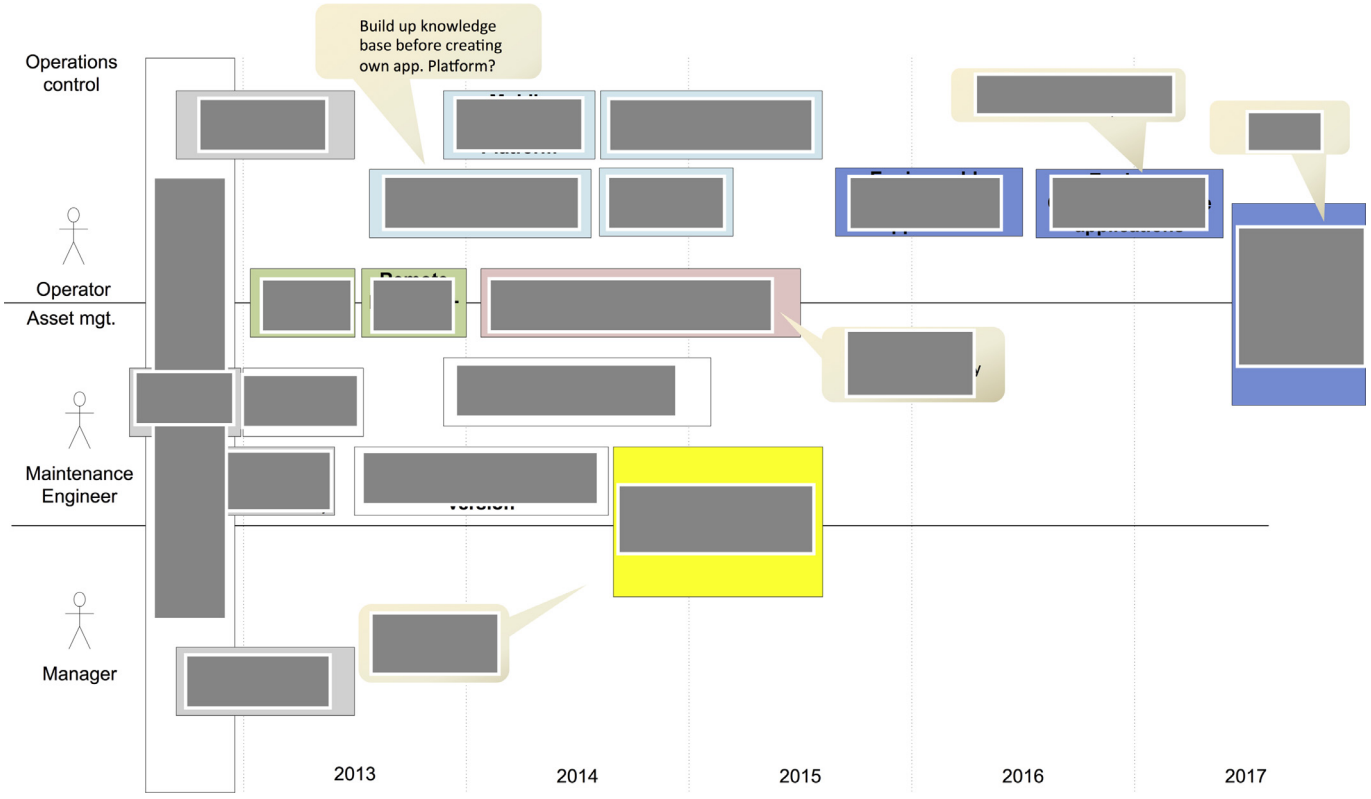

Fig. 3. Obfuscated consolidated decision roadmap model.

Several decision relationship models were created to visualise decision alternatives and their inter-dependencies. Different business units have different issues and maturity levels in their mobile strategies. Therefore, as a final step, multiple decision roadmap models were created to reflect different business units' needs and plans on software reuse, mobile business models, and potential enhancement in mobile strategy with value proposition. The roadmaps were consolidated into one overview model showing business units, their products, milestones and decisions to be made in time. Fig. 3 shows a simplified representation of the consolidated roadmap showing decisions to be made in time relevant to multiple products, stakeholders, milestones and evaluation criteria. The view is highly obfuscated as it mainly contains business-critical strategic decisions.

3.6.5. Implementation evaluation

The implementation evaluation was done both at the end of the client project and two years after the project finished. The former evaluation focused on if the decision viewpoint process helped the business units identify and prioritize stakeholders' concerns, and if the process helped provide clear and reasonable ground for them to make decisions on mobile applications in the future. The second evaluation was performed through retrospective in-depth interviews with focus on how the decision views have supported the stakeholders' decision making process for mobile strategy. The interviews were conducted by the ABB delegates. A question guide (see Appendix F) was used by the delegates to make sure that important topics of interest were covered during the interviews. All interviews were conducted using video conferencing tools as the responsible stakeholders were located at different ABB sites. The results of the evaluation are described in Section 4.

3.7. Data analysis procedure

The data gathered during the study was analyzed using the constant comparative method, originally described as part of grounded theory (Glaser and Strauss, 1967). Using the constant comparative method (also referred to as constant comparison), researchers analyze the collected data in multiple iterations. In each iteration, theories are developed or refined based on the collected data. The theories are constantly compared to new findings made during subsequent artifact analysis. That way, theories either stabilize, or they have to be discarded when new findings contradict them.

Grounded theory and its constant comparative method have increasingly been used in software engineering research as a method for qualitative data analysis in the last years (Urquhart et al., 2010; Adolph et al., 2011). It is particularly suitable if the research is fundamentally exploratory rather than being confirmatory (Urquhart et al., 2010). The knowledge questions in this study aim at gaining new knowledge rather than confirming existing hypotheses or conjectures. Additionally, the data collected is mainly qualitative. Thus, constant comparison is well-suited as a data analysis method.

Fig. 4 provides a model of concepts used in constant comparison. In the following, we describe those concepts as part of the analysis procedure applied during this study:

- **Step1 - Filter study documents**: In the first step, we browsed all documents we created during the study and documents that were provided by ABB. All documents were stored in the study document database. Apart from interview transcripts, field notes and minutes, this data also included relevant presentation material and design documents that related to the research questions.
- **Step2 - Coding and concept identification**: In step two, all documents were coded iteratively. The coding was done by

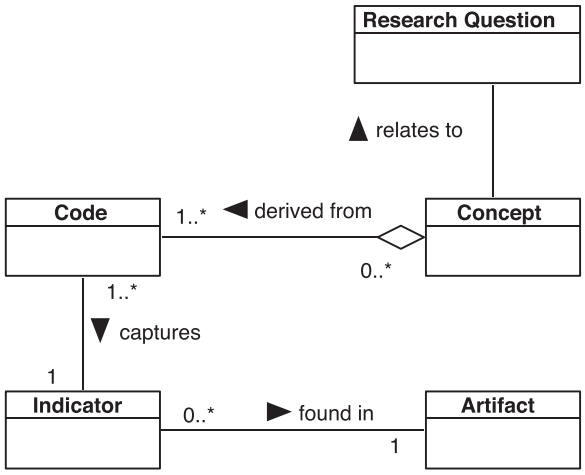

Fig. 4. Conceptual model of constant comparison entities.

two researchers. One researcher had already taken part in the study as researcher and client helper, the second researcher contributed to the study as a data analyst only. We used the qualitative analysis software MaxQDA,[6] which supports collaborative coding and provides several means for tracing codes to indicators in several document types: among others videos, presentations and office files. During the coding procedure, the two researchers independently studied the documents. Each phrase, sentence, or paragraph that indicated a certain behavior (called *indicator* in grounded theory (Strauss, 1987)) was labeled with a code that briefly captures the coded indicator. This approach to coding is called descriptive coding (Miles et al., 2013). The code system was developed from scratch, thus no pre-defined coding system was used. After each coded document, the researchers collaboratively revisited the codes and indicators and derived concepts together. A concept is a representation of a pattern of behavior, suggested by a set of indicators, which were captured using codes (Adolph et al., 2011). At the same time, the two independent coding systems of the researchers were discussed and harmonized. That way the coding system and the concepts were iteratively refined.

Step3 - Final harmonization: After all documents were coded and concepts were derived, the two analyzing researchers performed a final harmonization step. This included checking all indicators and codes belonging to each concept and revising them with respect to homogeneity and consistency.

Step4 - Assign concepts to research questions: After the final harmonization of codes and concepts, each concept was assigned to one or more research questions. This mapping of concepts and research questions served as a basis for the interpretation of results, reported in Sections 4.3.1 – 4.3.5.

4. Results

In this section, we present information on the interviews conducted, the coding system that emerged from the analysis of the data collected, and our interpretation of the collected data.

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

Table 5
Interviewers with project stakeholders.

Interviewee	Position	Duration (min)
Stakeholder 1	System architect	70
Stakeholder 2	Product line manager	43
Stakeholder 3	Product management manager	60
Stakeholder 4	Research program manager	120
Stakeholders 5 & 6	Business manager, Research scientist	74
Stakeholder 7	Product general manager	105

Table 6
Retrospective interviewers.

Interviewee	Position	Duration
Stakeholder 2	R&D for operations	10
Stakeholder 3	Global mobility	70
Stakeholder 6	Corporate research	20

4.1. Interviews

In the first project phase in 2012, the ABB delegates conducted interviews with seven stakeholders holding different positions and responsibilities at ABB. All stakeholders were involved in the mobile platform engineering project. Table 5 presents the interviewees, their position at ABB, and the duration of the interview. All interviews were transcribed and analyzed by the researchers. In total, 9.5 hours of interviews were collected in this phase.

Table 6 presents information on the retrospective interviews conducted in 2015 to gather data on the long term use of the decision views and the progression of the platform development and use. We managed to interview three of the original interviewees. The other interviewees had either left the company or were not available for interviews. The interviewees' positions within the company had changed since the original interviews. Again, the interviews were conducted by the ABB delegates and analyzed by the researchers.

Apart from the transcribed interviews, we analyzed 15 work artifacts (including decision views, project presentations and reports, requirements and design documents, and field notes).

4.2. Coding results

Fig. 5 presents the results from the coding procedure. We present concepts and sub-concepts derived from the codes. For reasons of company disclosure rules, the codes themselves are not shown. In total, roughly 400 incidents[7] were coded in the various documents. The concepts were categorized to make them more clear.

A mapping of concepts to research questions can be found in Appendix A.

4.3. Interpretation

In the following, we present our interpretations of all collected data with respect to the research questions.

4.3.1. Interpretation RQ1

RQ1 was about identifying the typical stakeholder groups that are involved in a platform design-space exploration and whose concerns must be considered when adapting the decision viewpoints. To answer this research question, we considered all parties that were directly involved in this project in one of the following

[7] In grounded theory, incidents are small chunks of data that yield certain theories, which are represented by concepts (Glaser and Strauss, 1967).

Category: Concerns
- What is a decision roadmap for product A?
- What is an efficient overall decision roadmap covering all products?
- What are candidate architectural styles for the platform?
- How do candidate solutions map to the architectural styles?
- What forces impact decision D?
- How sustainable is decision D?
- What decisions are subject to product variability?
- What decisions are influenced by force F?
- What decisions are influenced by force weight W(F)?
- What weight W does force F have for stakeholder H?
- What is the commercial impact of decision D?
  - Impact of cost force
  - Priority of cost force
  - Costs comprised of multiple dimensions
  - Cost of trade-offs
  - Cost is an important force for management
  - Cost overview is a main concern for developer managers
  - Costs were not identified as major concern in the process
- What are key decisions to be made for the platform?
- What is the optimal decision D in the context of forces F?
  - Concern: Support make or buy decisions
  - Decision view reduced bias
- What candidate solutions are considered for the platform?
- What forces need to be considered in the platform decisions?
  - Roadmap presents decision alternatives as guidance
- What decisions are dependent on decision D?
- What forces F have conflicting influences on decision D?

Category: Quality attributes of views
- Process comprehensibility
  - Identification of forces comprehensible
  - Identification of alternatives is comprehensible
  - Identification of decisions points is comprehensible
- Sustainability
- Reliability
- Effort
  - Investment
- Completeness
- Comprehensibility

Category: Long-term use
- COTS features and maturity evolve quickly
- Project situation changes quickly
  - Change: Staff/Mangement
  - Change of project-specific forces
  - Change: Budget cuts
  - Perspective of managing staff impacts design choices
- Investigated additional alternatives
- Impact of design space description
- Roadmap recommendations
- Forces view is consumed once, not used systematically

Category: Stakeholders
- Developers (Product)
- Plaform Users
- Software Architect (Platform)
- Software Architects (Product)
- Platform Owners
- Product Manager

Category: Retrospective Evaluation
- Views allow for a rapid kick-off
- Satisfied with results of design-space exploration
- Likes forces view
- Difference roadmap - project
- Roadmap underestimates effort and difficulty
- A common platform requires more time

Category: Missing Forces
- Time to market
- Flexibility
- Support for specific legacy features
- Value proposition
- License model of technologies
- Commercial robustness of technology

Fig. 5. Code system: resulting categories and concepts.

ways: (a) they were part of the team that explored the design-space for the common mobile platform; (b) they were interviewed during the force elicitation process or provided other input; (c) they participated in decision reviews or the presentation of project results. Additionally, we checked all documents and the interviews for other mentioned stakeholders who did not belong to one of the previous groups. Afterwards, we looked for commonalities between the candidate stakeholders and grouped them according to their common characteristics.

In total, we identified six distinct stakeholder groups that played a major role in the in the platform design-space exploration project: Platform Software Architect, Platform Product Manager, Platform Architecture Reviewer, Platform Software Engineer, Product Software Architect, Product Manager. Due to the nature of platform projects, stakeholders can be either associated with the platform or a platform-dependent product. Platform stakeholders primarily have an interest in the realization of the platform. Product stakeholders are primarily interested in realization of products that are based on the platform. The latter group can also be seen as customers of the software platform. In contrast to single application projects, product stakeholders and their concerns do not pertain to the same system, resulting in heterogeneous demands for the platform.

In addition to the distinction between platform and product stakeholders, we also distinguished between stakeholders being producers of a views, consumers of views, or both. View producers create and maintain architecture views, while consumers use the information presented in a view.

Although results collected in a single case are not necessarily generalizable per se, the described stakeholder types are generic for platform design space exploration projects in companies with separate organizational units for product families. In the following, we briefly summarize the main characteristics and responsibilities of each stakeholder as observed in this study. Stakeholder can be individuals, a team, or an organization. One individual might belong to multiple stakeholder groups.

Type:	Producer & consumer	Platform software architect
Description:		The Platform Software Architect (PSA) is mainly responsible for the architecture of the platform and as such, is in charge of the design-space exploration of the target platform. Despite the typical responsibilities of a Software Architect as defined by Kruchten (2008), the PSA harmonizes the heterogeneous architecture concerns and requirements of the client products. The PSA is the main view producer (i.e. during design-space exploration) and consumer (i.e. during maintenance and evolution).

Type:	Consumer	Platform product manager
Description:		The Platform Product Manager investigates, selects, and drives the development of the platform from a non-technical management perspective, as opposed to the PSA, including planning, strategic and financial aspects and the implementation of the platform in the organization.

Type:	Consumer	Platform architecture reviewer
Description:		The Platform Architecture Reviewer is an expert that reviews the platform's architecture regarding its fitness for purpose. In this review activity, the platform decisions and platform recommendations are evaluated according to the most important architecture concerns and requirements.

Type:	Producer & consumer	Platform software engineer
Description:		The Platform Software Engineer is responsible for implementing the common platform according to the platform architecture. The implementation of a platform is not limited to software development but also includes detailed software design, implementing platform variants and variability mechanisms as well as testing and documentation.

Type:	Consumer	Product software architect
Description:		The Product Software Architect is responsible for a particular system that is built upon the common platform, which means that the product's architecture is constrained by the platform's architecture. Hence, the Product Software Architect must ensure that all product concerns and requirements are properly addressed in the platform.

Type:	Consumer	Product manager
Description:		The Product Manager is the equivalent of the Platform Product Manager on the product-level. This stakeholder drives the development of a particular product that is based on a common platform from a management perspective.

4.3.2. Interpretation RQ2

RQ2 aims at identifying the stakeholders' concerns in platform design space exploration. The study confirmed several concerns that had been identified for the original set of decision viewpoints, while also several new concerns, specific to platform design space exploration could be found. Table 7 shows the existing concerns that were confirmed, related to the viewpoints that frame those concerns:

Table 8 summarizes the additional concerns we found during the study. In Table 9, they are mapped to the stakeholders identified as part of research question one. After that, each of the concerns is briefly explained.

Table 9 provides a mapping of the concerns to the stakeholders identified during the study as part of research question one.

A detailed description of all newly identified concerns can be found in Appendix B.

4.3.3. Adaptation of decision viewpoints (RQ3)

This research question focusses on the adaptation of existing decision viewpoints to satisfy the concerns from RQ2. As opposed to the other research questions, RQ3 is a practical problem. In the remainder of this section, we describe the process we (ABB and researchers) followed to choose and adapt existing viewpoints to frame the additional concerns identified. The validation of the adapted viewpoints is presented in the next section. A full specification of all adapted viewpoints can be found in Appendix C.

Table 7
Confirmed concerns from previous studies.

Code	Concern	Original viewpoint
C1	What forces F_j impact/influence decision D?	Forces Viewpoint (van Heesch et al., 2012b)
C2	What decisions D_k are influenced by force F?	Forces Viewpoint (van Heesch et al., 2012b)
C3	What forces F_l have conflicting influences on decision D?	Forces Viewpoint (van Heesch et al., 2012b)
C4	What decisions are dependent on decision D?	Relationship Viewpoint (van Heesch et al., 2012a)
C5	What concerns C_m does decision D pertain to?	Forces Viewpoint (van Heesch et al., 2012b)

Table 8
Additional concerns identified in the study.

Code	Concern
C6	What forces F_m need to be considered in the platform decisions?
C7	What weight W_b does force F have for stakeholder H?
C8	What key decisions D_c need to be made for the platform?
C9	What decisions D_q are subject to product variability?
C10	What decisions D_m are influenced by force weight WD_n?
C11	What is the optimal outcome for decision D in the context of forces F_q?
C12	What candidate solutions are considered for the platform?
C13	What are candidate architectural styles for the software platform to be developed?
C14	How do candidate solutions map to the candidate architectural styles identified?
C15	What is an efficient decision roadmap for the platform?
C16	How sustainable is decision D?

Table 9
Mapping of stakeholders and concerns.

Stakeholder	Concerns
Product manager	C9, C15, C16
Software architect	C9, C12, C13, C14, C15, C16
Architecture reviewer	C1, C2, C3, C4, C5, C6, C7, C8, C9,C10, C11, C12, C13,C14
Platform software engineer	C5, C6, C7, C9
Platform software architect	C1, C2, C3, C4,C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16
Platform product owner	C15, C16

Adaptation of decision-forces viewpoint. As described in Section 3.6.4, the first activities during the project centered around stakeholder identification from the participating business units, an elicitation of their requirements and other forces, and a prioritisation and weighing of the forces.

The new stakeholders and concerns that had to be framed by the adapted decision viewpoints are described above. For the identification and documentation of requirements and other decision forces, the Decision Forces Viewpoint (van Heesch et al., 2012b) was a natural choice. The forces viewpoint allows to document decision forces and to categorise forces using concerns. The only required change was the set of decision states. Please refer to Appendix C.1.4 for a description of those changes. The decision forces viewpoint satisfies many force-related concerns from Tables 7 and 8 (C1, C2, C3, C5, C6, C8, C9, C11, C12).

Nevertheless, in the original viewpoint, a context is assumed in which a forces view is used for a single application development project with a distinct set of stakeholders and concerns, all related to the same application. For platform design space exploration, the situation is different in that the product-stakeholders' concerns and forces can be very heterogeneous, depending on the demands of the concrete applications that need to be built upon the platform. Some forces are common among all stakeholders; these forces should surely be considered by the platform architects. Under specific circumstances, it can make sense to additionally consider forces that are common among a subset of product-stakeholders only.

In reality, the situation is even more complex, as the forces are not important or not important for specific stakeholders, but there are various degrees of priorities. The original forces viewpoint does not provide any support for different priorities of forces,

Table 10
Excerpt from a force priority model.

Concerns and Forces	Stakeholder 1	Stakeholder 2	Stakeholder 3	Average Prio	Variance
Concern: Deployability					
13.1. Specialized "App" stores	5	4	4	4	0,2
13.2. Private/public cloud solution	2	3	4	3	0,7
13.3. Push updates	2	2	3	2	0,2
13.4. Vendor distribution channel	1	5	3	3	2,7
13.5. Replacement of devices	4	4	4	4	0
Concern: Device Management					
14.1. Device Usage Monitoring	2	3	4	3	0,7
14.2. Application usage monitoring/diagnostics	1	4	3	3	1,6
14.3. Remote Device Management	4	4	3	4	0,2
14.4. Real-time support	1	3	4	3	1,6

nor for stakeholder-specific forces in general; thus, support for deciding which forces need to be considered and how important the forces are, is not provided by the original forces viewpoint. Therefore we decided to adapt the viewpoint by adding another model kind dedicated to the analysis and documentation of stakeholder-specific force priorities. The new model kind can be found in Appendix C.1.3.

Table 10 shows a simplified priority model created during the study. The model shows the forces belonging to the concerns *deployability* and *device management*. Each stakeholder expressed the priority of the forces on an integer scale from 1 for not important to 5 for very important. The outer right columns show the median values of the ratings ("Average Prio") and the variance of the priority ratings. To improve the readability, the columns are coded using traffic light colors. Green implies high importance or low variance, respectively. Red implies low importance or high variance. The information in the model implies that forces 13.1, 13.5, and 14.3 are deemed most important by all stakeholders. However, these values must be interpreted together with the variance. If a force has a high average and a low variance, this implies that the force is a key force that is equally important for all stakeholders (in the example it is force 13.5: "Replacement of devices"). If a force has a high average, but also a high variance then this means that the force is very important to some of the stakeholders while being very unimportant for others. Such forces also require special attention, as they may indicate a variability point in the system (e.g. force 13.4: "Vendor distribution channel" in the example).

The additional model kind for the forces viewpoint addresses concerns related to stakeholder-specific force priorities (C7 and C10). Generally, the adapted forces viewpoint satisfies the majority of concerns in platform design space exploration. This can be explained by the fact that the original forces viewpoint was specifically designed to support and document a rational decision-making process based on explicit decision criteria. The other existing viewpoints (decision relationship viewpoint, chronological viewpoint, stakeholder involvement viewpoint, and detail viewpoint) have greater benefit for preserving the rationale of made decisions rather than providing support for the decision-making process itself. In design space exploration, no decisions are made; instead the focus is on making explicit the problem space (i.e. the forces), the design space (i.e. the candidate solutions) and the inter-relationships between them to support decisions that are made later.

Decision-relationship viewpoint. At that point, the concerns that remained un-addressed were:

C4: What decisions are dependent on decision D?
C13: What are candidate architectural styles for the software platform to be developed?
C14: How do candidate solutions map to the candidate architectural styles identified?
C15: What is an efficient decision roadmap for the platform?
C16: How sustainable is decision D?

Concerns related to decision dependencies (C4) can be framed by the decision relationship viewpoint without any adaptations (van Heesch et al., 2012a). This viewpoint is dedicated to decision relationships; among others, it can be used to capture causal and dependency relationships. It also visualises the current state of decisions. Candidate architectural styles (C13,C14) can also be represented. In the study, ABB created component views using UML component diagrams and deployment views using UML deployment diagrams to sketch the architectural styles for the stakeholders.

New viewpoint: decision roadmap viewpoint. To address concern C15, we decided to develop a new decision viewpoint. Although the decision relationship viewpoint and the decision chronological viewpoint frame concerns related to decision dependencies and chronological order of decisions, the roadmap concern is inherently different in that it has a planning aspect that is not by any means addressed in the existing viewpoints, which document decisions that were already made. The primary objective of decision roadmaps is aligning short-term and long-term objectives of multiple projects and providing a planning framework for those projects typically covering multiple years.

The decision roadmap viewpoint, we created to address these concerns, has only one model kind that integrates with the model kinds of all other decision viewpoints. It shows concepts related to the roadmap for the software platform itself (Platform Milestone, Platform Phase) and concepts related to the software products that use the platform (Product Milestone, Product Phase, Product). As the platform development is typically done in parallel to the product development, the product phases are independent of the platform phases. However, the platform development needs to be synchronised with certain milestones of the product to make sure that it does not delay the products release. Architecture decisions that are not subject to product variability are made as part of the platform project, whereas variability decisions are made for each product individually in the respective product phases.

Fig. 6 shows a conceptual example of a decision roadmap view. The full specification of the viewpoint can be found in Appendix C.

Concern C16 about understanding the sustainability of a decision is particularly challenging as the sustainability of an architectural decision is dependent on many factors. Zdun et al. derived five criteria for sustainable decisions from 10 industrial case studies (Zdun et al., 2013):

Strategic: Are strategic consequences of decisions considered, e.g. long-term impact like maintenance effort and future operations?
Measurable and manageable: Are the decisions impact on functionality and quality attributes measurable?

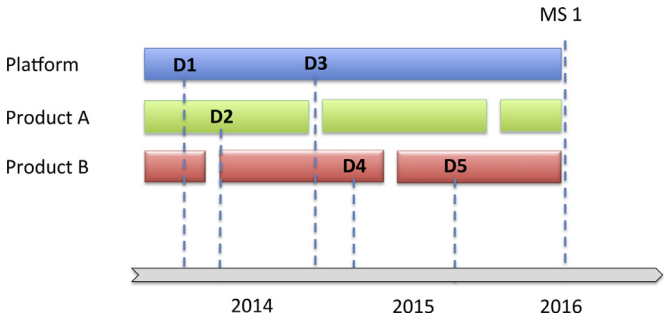

Fig. 6. Conceptual example of decision roadmap view.

Archievable and realistic: Is the rational for decisions explicit and are architectural choices made pragmatically?
Rooted in requirements: Are decisions grounded in the project requirements and constraints?
Timeless: Is decision rationale based on non-volatile argumentation (e.g., using proven patterns and architectural styles)

All of those factors for sustainable decision-making concern the rationale behind architectural decisions. Following Zdun et al.'s suggestions, the more systematic or the more rational a decision was made, the more sustainable it is. The decision forces viewpoint, as explained above, is particularly useful for supporting a rational and systematic reasoning process. This was also shown in previous studies on decision viewpoints (van Heesch et al., 2012b; Heesch et al., 2013). Thus, although no hard measurements for decision sustainability exist, we argue that the degree to which a decision is embedded in the context of the identified decision forces and the elaborateness of the force elicitation have a major impact on the decisions' sustainability. Consequently, decision forces views provide certain support for judging the potential sustainability of decisions. However, clear indicators or measures for the sustainability of decisions, as desired by the stakeholders (concern C16), cannot be provided in decision views. Addressing this concern is subject to future research. As part of the interpretation of RQ4, we will discuss sustainability and other quality attributes of decision views.

4.3.4. Interpretation RQ4

In RQ4, we validate to what extent the adapted decision viewpoints, described in Section 4.3.3, are able to satisfy the platform design-space exploration specific concerns identified in Section 4.3.2 with respect to the stakeholder described in Section 4.3.1. We found that the choice of viewpoints and the implemented adaptations comprehensively satisfy the platform design-space exploration related concerns of the relevant stakeholders. The only exception is C16 (How sustainable is decision D?), as described in Section 4.3.3, which was only identified after the adaptation of the viewpoints. Overall, these results indicate that the decision forces viewpoint is highly suitable for platform prototyping activities, as it summarizes design-space exploration results and helps to understand complex trade-offs. Moreover, it proved to be an effective tool in convincing platform stakeholders of the proposed technology choices. Additionally, the decision roadmap viewpoint was found to be a valuable addition to the decision viewpoints as it helped to plan and coordinate developments for multiple sub-projects relying on the target platform. The decision roadmap viewpoint was a major outcome of the study that all stakeholders deemed highly important. When asked about the decision roadmap view, a participant concluded that "*[t]he output gives you an idea of the time horizons that you have to think about*".

A key activity in the common mobile platform project was the presentation of the (intermediary) results of the design-space exploration to the relevant stakeholders as part of the decision-making process. We observed that those decision reviews rarely focus on technical details but rather stay on an abstract level, partly because the majority of stakeholders work in a management or technical lead position (e.g. Platform Software Architect or Product Manager) that implicate hard time-constraints and partly because they had a high-level of trust in the competence of the view producer so that they did not require further details. A common view amongst interviewees was that the summarizing aspect of the forces viewpoint was considered more important than the comprehensive presentation of details, such as the precise rationale behind the force ratings.

The forces viewpoint combines architecture concerns, forces, candidate solutions and force ratings and thus, allowed stakeholder to get a good overview of a decision and to quickly find answers to their concerns. The stakeholders were able to easily identify the best candidate solution (C11) and also gained a good overview the considered candidate solutions (C12). "*I think it does help. [It] did help me to view all the alternatives, and benefits and drawbacks*".

In the context of the platform design-space exploration project, we found that the forces viewpoint dictates the right amount of information at an appropriate level of abstraction. For example, one interviewee said: "I found that you could kind of look at that quickly [...]. Certainly the color coding is really what gives it that ease of seeing. Look at the deployability across all [candidate solutions], there is an immaturity there. Here is a stunning amount of data on this one slide" A viewpoint inherently incorporates trade-offs in terms of detailed versus abstract information. Too detailed information and the view requires substantial effort to be consumed and can be perceived as overwhelming by stakeholders. Too abstract and it is hardly useful.

Moreover, we found that the stakeholders were able to assess whether the presented information was complete, i.e., if all relevant architectural concerns, decision forces architectural styles and candidate solutions were identified. As one participant expressed this during the interview: "*Overall that was so easy to understand and you could quickly recall something and recognize your overall requirements*". During the discussions with the stakeholders, forces were identified that were not recognized during the force elicitation process such as commercial robustness of the technology or time to market. This clearly shows that the suggested viewpoints are effective for stakeholder communication. Moreover, the newly introduced force priority model enabled the architects to identify the reasons why those forces were not included, for instance, because their priority was not rated high enough or the respective stakeholders did not indicate them during the elicitation interviews.

However, the forces viewpoint was also found to obfuscate information in two ways. First, the rationale leading to a force rating is not visible. Second, it can be challenging to interpret the implications that small differences in force-ratings have on the respective candidate solutions, e.g., what is the difference between a -2 and a -1, or what is the criteria that makes a force-rating X (i.e. the exclusion of a candidate solution) and not -3. However, these issues were not particularly prominent in the interview data and can be mitigated, for example, by providing a legend that explicitly explains the rating scheme and rating criteria, and by referencing the detailed results of the analysis (e.g., external documents like spreadsheets or reports) in order to establish traceability.

The analysis of the stakeholder interviews further showed that apart from the viewpoints' ability to satisfy the identified concerns, the stakeholders are concerned about the quality of the content within views rather than the information content itself. Those qualities are orthogonal to the actual stakeholder concerns framed

by the viewpoint. Five broad themes emerged from the analysis: sustainability, comprehensibility, completeness, reliability, and usability. In the following we list those themes and suggest those to be considered as a first step towards a quality assessment model for architecture views. We argue that these qualities are implicitly considered when choosing a viewpoint or assessing the quality of a view.

Sustainability In contrast to the sustainability of a decision, a recurrent theme in the interviews was the sustainability of a view. When creating architecture views, an architect often must decide which aspects of the system are made explicit and which aspects should be kept abstract. The trade-off between explicitness and abstraction determines to some extent how long the information presented in a view remains accurate and true. A view must be concrete enough so that it conveys enough information to be useful for the relevant stakeholders but it should also be abstract in order to have certain resilience for change.

Comprehensibility The ability of the view to be understood by the stakeholder whose concerns it frames. This includes the ability to identify the concerns answered by a view and understand the concepts and models used. For example, we found that stakeholders could intuitively understand the forces view, even after years had passed since the view was created.

Completeness The extent to which a view presents all information necessary to answer the stakeholder concerns.

Reliability The extent to which the view withstands scrutiny, for example, the stakeholders' ability to assess the correctness and accuracy of information. A mitigating factor both for reliability and completeness is the stakeholders' confidence in the view producer(s). In this project, we found that in all cases, the participants expressed high confidence in the abilities of the view producer and believed that the results were based on a rigorous process. As one participant put it: "My confidence that you have covered the completeness of the design space is more based on my assessment of your capabilities than [having detailed information about the] methodology". In contrast, we observed that the confidence in the accuracy of force ratings was much lower for the technologies that were based on self-assessment of the vendor. A possible adjustment would be to clearly indicate the source of the force rating and, as proposed by one participant, to also show confidence factors, which would it make easier for stakeholder to assess the reliability of the information.

Usability The ease of creating the view. Considerations that impact the usability of a view are: the need for dedicated tools to create and maintain the view, the complexity of the model kind and its visualizations, or the time-wise effort to create the view.

4.3.5. Interpretation RQ5

In this research question, we wanted to find out how the long-term value of the adapted viewpoints was perceived by the stakeholders in the course of the project. Documenting architecture views requires significant effort. Therefore, the question to what extent this effort has positive long-term implications for a project or a company is important. The retrospective interviews were conducted two years after the first interviews and the presentation of the platform project results.

A common view amongst interviewees was that the platform evaluation based on decision viewpoints provided a solid foundation and enabled a rapid kick-off in the product projects. Due to the information presented in the forces view and roadmap view, the individual products were able to make quick decisions on how to move towards the common platform. As one interviewee put it: "It was a really helpful exercise to go through this in the beginning and it sort of helped us to make some decisions early on". These decisions were not only technical in nature but also concerned investments, e.g., one interviewee said: "I think the biggest [help was] in terms of agreeing where we want to invest".

Despite being subject to significant changes, the roadmap envisioned in the decision roadmap view was considered highly useful, especially in the beginning of the project. One interviewee reported that they "simply followed the proposed path". However, other participants reported that the roadmap was very accurate but that the timing was too ambitious. Overall, the roadmap view was highly appreciated as it outlined the transition of individual projects towards the platform

However, we also found that the decision views were not systematically used in all projects. Instead, some interviewees reported that they used the decisions views only in the beginning of the project and once the knowledge was internalized, the decision views were archived. They stated that they used the information implicitly in their projects as something that they "had in the back of [their] mind". They would take decisions with the inherent knowledge of the requirements and recommendations presented in the decisions views but they wouldn't "necessarily go back and refer to the [views] to use [them] in the decision-making" process.

Overall, the interviewees reported that they were satisfied with the results of the platform design-space exploration. The retrospective interviews showed that the decision views had a significant impact on the projects, especially the forces view and roadmap view. With change being the only constant, it was no surprise that there have been significant changes in the project both on the platform-level and product-level. We observed budget changes, staff turnover, changes in focus, scope and priority of products, changes in the market environment, as well as changes in requirements, all of which are common risks associated with software engineering projects. Naturally, all these changes impacted the original platform recommendation. For example, new forces emerged that were initially not considered or forces dropped out or were decreased in priority. Additionally, the considered candidate solutions also evolved in maturity and functionality, especially since primarily COTS technologies were considered for the platform. Considering that the goal of the design-space exploration was to provide recommendations for years ahead, the question of sustainable decisions and sustainable decision views seems to be an important direction for further research. Moreover, further research should be undertaken to investigate the combination of risk-related concerns and decision viewpoints, especially in the case of the roadmap viewpoint.

5. Generalization and limitations

In Section 3.6.3, we already discussed possible threats to the validity of our results related to the study design in general and the treatment design in particular. Concerning generalizability, TAR is -to a certain degree- comparable to case studies, because the study results are interpreted in the context of the organization, in which the study was performed (Engelsman and Wieringa, 2012). Findings that are internally valid in this context are not necessarily fully valid for a broader theory, as sometimes stated by critics of case study research (Yin, 2009). Because single cases do not allow for statistical generalization, in case study research and likewise TAR, analytical generalization is used (Wieringa, 2014; Wieringa and Moralı, 2012). In the remainder of this section, we will revisit the findings for our research questions and discuss their generalizability.

In RQ1, we identified typical stakeholders for platform design space exploration. The role designations of the stakeholder groups (e.g. "platform product manager"), as well as their task and responsibility profiles were taken over from ABB. Typically, such profiles are very heterogeneous among different companies. Even the role "software architect" has very different connotations in different companies. Consequently, neither the stakeholder role names, nor their profiles are generalizable to other companies. Nevertheless, the profile descriptions we provided can be used to map our stakeholders to the specific stakeholder profiles in other companies.

With the exception of concern C15 (*What is an efficient decision roadmap for the platform?*), the stakeholder concerns we identified in RQ2 are generalizable to other software platform design space exploration projects. This is because the concerns are abstract enough to be independent of company or case-specific interests. The decision roadmap concern is specific to situations, in which the platform design space exploration is performed where several software products need to be planned simultaneously with the platform development project, and where a synchronization of those projects is possible and desired by a common or shared project governance authority.

RQ3 frames a practical problem (adaptation of decision viewpoints). Thus, the concept of generalizability is not applicable here. The suitability of the adaptations, however was evaluated as part of subsequent research questions. The generalizability of the results of RQ4 (validation of the adapted viewpoints regarding the concerns) is limited by the degree to which the concerns are case-specific. Before, we already discussed the limitations concern the generalizability of C15 (efficient decision roadmap). To satisfy this concern, we developed an additional viewpoint (called *Decision Roadmap Viewpoint*). Although the viewpoint specification deliberately abstracts from ABB-specific concepts to a high degree, the suitability of the resulting viewpoint cannot be claimed for a broader context. The same holds true for RQ5 (long-term value of the created views as experienced by the stakeholders). The views created by ABB during the study are specific to the concrete platform development project, their long-term value is dependent on the project context and even on the representatives of the software products that were interviewed (i.e. different project see different long-term value). However, as described above, one aspect that was mentioned by all interviewees is that a thorough platform evaluation based on decision viewpoints provides a solid fundament for subsequent software product development. Naturally, design space exploration is a valuable input for architecture decision making, because it allows for more informed decisions taking a wide range of pros and cons and possible candidate solutions into account. Whether or not upfront platform design space exploration is of financial value, however, depends on the concrete business context.

6. Related work

The approach used in this paper can be seen as a specific instance of Multiple Attribute Decision Making (MADM). As it refers to making preference decisions (e.g., evaluation, selection, and prioritization) over the available alternatives that are characterized by multiple, usually conflicting, attributes (Hwang and Yoon, 1981). MADM is a branch of the field of multiple criteria decision making (MCDM), as MCDM also includes multiple objective decision making problems where the objective is to find the best alternative given a set of conflicting objectives (Yoon and Hwang, 1995). A taxonomy and overview of MCDM is provided in Ribeiro et al. (2011), which is based on Yoon and Hwang (1995); Shu-Jen et al. (1992).

Examples of MADM methods include; scoring methods like weighted average (Figueira et al., 2005; Yoon and Hwang, 1995), comparative methods like AHP (Saaty, 1980) and ELECTRE methods (Figueira et al., 2005), and ranking methods like TOPSIS (Hwang and Yoon, 1981). These methods can also be combined, as the HAM (Ribeiro et al., 2011) method demonstrates, which combines elements of the AHP and the weighted average method.

Attribute hierarchy-based evaluation of architectural designs (AHEAD) method introduces Attribute-Driven-Design (ADD) (Bass et al., 2002) which can refine and verify requirements, elicit candidate technologies, preliminarily determine discerning criteria and implement the prototype before implementing AHP. Supplementing ADD with AHP enables more accurate reflection of stakeholder's priorities for the evaluation criteria; neutralized researcher bias; and facilitated team acceptance of the evaluation result and confidence in the recommendation. However, it does not provide a method to evaluate functional requirements and it cannot analyze the costs and benefits. Moreover, AHEAD needs to prototype the alternatives before implementing the AHP. Compared to these approaches, the approach used in this paper differs in that it outlines, but does not prescribe, how consequent decisions should be made.

The decision forces view, which was used as a basis in the presented approach, has been evaluated earlier in Heesch et al. (2013). This paper confirms some of the earlier findings of Heesch et al. (2013). Firstly, the forces view can indeed be used to target prototyping activities, as it summarizes research results on several design options and helps to understand trade-offs. Secondly, it also proved to be a useful tool in convincing stakeholders of the proposed technology choices, as rationale about the available and chosen trade-offs was made explicit.

The approach of this paper is used to perform a technology evaluation in a platform context. There are several technology evaluation methods available. TechSuRe (Jansen et al., 2011) is a method to assess sustainability (risks) when introducing a new technology (software) into a software intensive system. This method assesses potential technology from three perspectives: time, risk, cost and benefit. The cost and benefit are estimated through Net-Present Values and Return-on-investment. While the time and risks are estimated through several high level indicators on a three-level discrete scale, subjectively quantified by the assessors. Another approach is STEIM (Brown and Wallnau, 1996), which is a framework which identifies technology deltas created by new technology using genealogy and habit models. The main difference with the approach presented here is that our approach is not solely focussing on technology choices, but concerns in general.

A pattern is a proven solution to a problem in a context, resolving a set of forces (Zimmermann et al., 2008). Hence, patterns (Alexander, 1979; Gamma et al., 1995; Buschmann et al., 1996; Schmidt and Buschmann, 2003) contain descriptions of the forces and how a solution balances them. The basic idea to capture rationale for candidate architectural solutions using forces is similar to the concept we use in the decision forces viewpoint. Furthermore, so called pattern languages (Buschmann et al., 2007) exist, which describe the relationships between individual patterns and describe how forces influence the choices that could be made for composing patterns together. This is very similar to our approach, with the significant distinction that in our approach we are dealing with unproven solutions in a problem context that is not completely captured in existing patterns.

7. Conclusions and future work

In this research project, we investigated the use of architecture decision viewpoints for supporting design space exploration of software platforms. Our results indicate that especially the de-

cision forces viewpoint, with a few adaptations, is a good match for supporting platform design space exploration, as it focuses on connecting elements from the problem space to elements from the solution space and allows to reason about potential trade-offs and variability points in the platform to be developed.

The Decision Roadmap Viewpoint is a first step towards long-term decision planning for complex multi-application platform projects. It has shown its ability to capture a decision roadmap based on the level of information gained during the design space exploration. However, the goal to provide recommendations valid for multiple years with highly sustainable decision recommendations was not entirely met, as too many project contextual factors changed after the design space exploration was finished.

In the course of the study, we identified multiple directions for further research:

Quality attributes of views: As described in Section 4.3.4, we found that apart from the content shown in decision views, stakeholders have a vital interest in being able to judge some meta qualities of views. Meta qualities, in this context, are for instance comprehensibility of views, completeness of view, and reliability and sustainability of the view content. These qualities are content-independent attributes of views that cannot be achieved through viewpoint definitions; they rather depend on information sources used to gather view content information, skills and knowledge of the view producers. We plan to investigate more how such meta qualities can be measured and expressed.

Decision sustainability: Decision sustainability was a major concern of many company stakeholders. Especially for design space exploration conducted to provide recommendations for years ahead, sustainable rationale is important. Unfortunately, the sustainability of decisions is hard to judge objectively. We plan to conduct further research for identifying reliable indicators for the sustainability of architecture decisions and architecture decisions views.

Risk-related concerns: Especially the decision roadmap viewpoint, designed to provide guidance on context and time in which platform decisions should be finalized and implemented is subject to several contextual, organizational, and technological risks (e.g. budget changes, staff turnover, changes in focus, further development of technologies). As we found out during the retrospective interviews, these risks are not sufficiently addressed in the current viewpoint definition. Risk themes and risk evaluation procedures must be developed to accompany the information shown in the roadmap views to give view consumers more guidance on how to use and revise decision views in the longterm to make sure that the information shown remains valid.

Acknowledgments

We would like to thank the participating architects and product managers at ABB for their openness to share and discuss their concerns, and the relevant product/architecture roadmaps.

Appendix A. Mapping of coding concepts to research questions.

The following list shows a mapping of concepts to the research questions. In cases, where a concept relates to multiple research questions, only the primary research question is shown.

- RQ1 (Typical stakeholders for platform design space exploration)
  - Developers (Product)
  - Stakeholders
  - Plaform Users
  - Software Architect (Platform)
  - Software Architects (Product)
  - Platform Owner
  - Product Manager
- RQ2 (Stakeholder concerns to be framed by adapted viewpoints)
  - What weight W does force F have for stakeholder H?
  - What is the optimal decision D in the context of forces F?
  - What is the commercial impact of decision D?
  - What is an efficient overall decision roadmap covering all products
  - What is a decision roadmap for product A?
  - What forces need to be considered in the platform decisions?
  - What forces impact decision D?
  - What forces F have conflicting influences on decision D?
  - What decisions are subject to product variability?
  - What decisions are influenced by force F?
  - What decisions are dependent on decision D?
  - What candidate solutions are considered for the platform?
  - What are key decisions to be made for the platform?
  - What are candidate architectural styles for the platform?
  - Roadmap presents decision alternatives as guidance
  - Priority of cost force
  - Impact of cost force
  - How sustainable is decision D?
  - How do candidate solutions map to the architectural styles?
  - Decision view reduced bias
  - Costs were not identified as major concern in the process
  - Costs comprised of multiple dimensions
  - Cost overview is a main concern for developer managers
  - Cost of trade-offs
  - Cost is an important force for management
  - Concern: Support make or buy decisions
- RQ4 (Concern satisfaction of adapted viewpoints)
  - Views allow for a rapid kick-off
  - Sustainability
  - Suggestion for improvement
  - Satisfied with results of design-space exploration
  - Roadmap underestimates effort and difficulty
  - Reliability
  - Process comprehensibility
  - Missing Force: Value proposition
  - Missing force: Time to market
  - Missing Force: Support for specific legacy features
  - Missing Force: License model of technologies
  - Missing force: Flexibility
  - Missing Force: Commercial robustness of technology
  - Likes forces view
  - Investment
  - Identification of forces comprehensible
  - Identification of decisions points is comprehensible
  - Identification of alternatives is comprehensible
  - Forces
  - Evaluation of decision views
  - Effort
  - Difference roadmap - project
  - Comprehensibility
  - Completeness
  - A common platform requires more time
- RQ5 (Long-term value of views as experienced by the stakeholders)
  - Sustainability
  - Roadmap recommendations
  - Project situation changes quickly
  - Perspective of managing staff impacts design choices

- Investigated additional alternatives
- Impact of design space description
- Forces view is consumed once, not used systematically
- COTS features and maturity evolve quickly
- Change: Staff/Mangement
- Change: Budget cuts
- Change of project-specific forces

Appendix B. Detailed descriptions of additional concerns identified during the study.

Code: C6	What forces F_a need to be considered in the platform decisions?
Description:	We found that the stakeholders deem importance to the differentiation between the forces for the applications to be built on top of the platform and the forces for the platform itself. Many of the decision forces that influence architectural decisions in software platform design are less specific than forces for single software applications. Example forces for platform decisions elicited in the study are: • Access to mobile device capabilities • Interface capabilities with other applications on device • Use experience • Specialized app store capabilities In single application architecture, the first force on device capabilities, for instance, would specifically address the capabilities needed in a specific application (e.g., access to the camera, or access to the GPS module), rather than generally addressing access to available capabilities.

Code: C7	What weight W_b does force F have for stakeholder H?
Description:	Although software platforms are usually designed for use over a long period of time and for software applications that are not known at design time, several potential users of the platform were already known. At ABB, six product groups of two business units were considered as potential first users of the platform. Representatives of each of these BUs were interviewed for their requirements for the platform, which were to be considered as forces in the platform decisions. Naturally, the diversity of the BUs' application domains also causes a difference in the importance of certain requirements. Some requirements are very important for all BUs, while others are only important to a subset of the BUs, or not important at all for single BUs. Consequently, depending on the importance of specific requirements, the corresponding forces have higher or lower weight when being considered in the architectural decisions for the platform.

Code: C8	What key decisions D_c need to be made for the platform?
Description:	Key decisions are those decisions that are impacted by the most important forces and that have the greatest influence on the remaining solution space after the decision was made. Examples of key decisions in the study were which mobile platform to support, whether or not to support offline use of applications (i.e. temporary non-availability of an internet connection), and whether or not to use a cross platform compiler for the development of the platform.

Code: C9	What decisions D_d are subject to product variability?
Description:	Variability refers to the ability of a software artifact for being customized, configured, or extended for use in a specific context (Bachmann and Clements, 2005). When developing a common platform for multiple software products, it may be required to make decisions that are specific to a concrete software product that will be based on the platform, e.g. in cases where it is not possible to find a reasonable trade-off between multiple conflicting forces (Pohl et al., 2005; Chen and Babar, 2010). Such decisions require a special treatment, e.g., to exclude a specific feature from the platform so that it can be implemented by those products only that require the feature.

Code: C10	What decisions D_m are influenced by force weight WF_n?
Description:	In van Heesch et al. (2012b), we described that a common stakeholder concern is to understand which decisions were impacted by a specific force. In software platform engineering, as described above, forces can have different weights as assigned by different (groups of) stakeholders. In the study, some stakeholders expressed the need to understand which decisions were impacted by a specific force weight, i.e. which decisions would have been made differently, if a stakeholder had expressed a lower or higher importance for a specific force.

Code: C11	What is the optimal outcome for decision D in the context of forces F_q
Description:	This concern is about getting support for choosing the best option for a decision to be made in the context specific forces and force weights. These forces are not necessarily all known forces; during architectural synthesis, it can be beneficial to select among design options based on a sub-set of forces only and then evaluate later, if the decision outcome conflicts with other (important) forces (Hofmeister et al., 2007).

Code: C12	What candidate solutions are considered for the platform?
Description:	This concern deals with the identification and documentation of candidate architectural solutions for the platform. In the case study, by far the most candidate solutions concern existing technologies or frameworks (COTs). As part of the solution space exploration, ABB performed a thorough market analysis of mobile technologies. In the first step, this analysis was done independently of the identified forces. Only later, the technologies were examined closer with respect to their suitability as candidate solutions for specific design problems in the platform engineering process.

Code: C13	What are candidate architectural styles for the software platform to be developed?
Description:	Architectural styles express "a fundamental structural organization schema for software systems" (Rozanski and Woods, 2005) including pre-defined element-types, their responsibilities and relationships. In the study, architectural styles were used as a means to reason about the impact of architectural styles on quality attributes independent of concrete architectural solutions imposing such specific styles. Examples of architectural styles identified in the study are: Native application, embedded webkit with Javascript API, and cross compiler.

Code: C14	How do candidate solutions map to the candidate architectural styles identified?
Description:	This concern is about connecting candidate solutions (see concern C11) to candidate architectural styles (concern C12).

Code: C15	What is an efficient decision roadmap for the platform?
Description:	Software platform engineering projects are long-term projects that usually endure multiple years and that impact or contain multiple other software projects or sub projects (e.g. the software systems that are to use the platform as a basis). Therefore, it has been a common practice to define a roadmap for such projects. Roadmaps provide a means to anticipate future technology developments and they provide a framework that helps to plan and coordinate related developments (Phaal et al., 2010). A decision roadmap, as used here, is a roadmap containing decisions to be made, the time or time-span in which the decisions should be made, the project or sub-project responsible for making or enforcing the decisions, and decision dependencies.

Code: C16	How sustainable is decision D?
Description:	Developing a company-internal software platform is a major financial investment that does not have an immediate benefit, because the platform itself cannot be sold to customers. Instead, the business value results from a long-term reuse of the platform in multiple software projects, which saves efforts when using the platform as a basis. In order to maximize the platform project's return on investment, the sustainability of the decisions made is a major concern, not only of the financial stakeholders. In industrial automation, ABB's business domain, it is even more important to make sustainable decisions, because high investments in machines and devices require software to be maintainable cost-efficiently and over the complete life-cycle of the systems (Weiss et al., 2012). While the sustainability of a technological decision could also be a force to be considered when making decisions, this concern is about the stakeholders interest to understand the sustainability of a specific decision.

Appendix C. Adapted viewpoint specifications.

C.1. Design exploration forces viewpoint

The design exploration forces viewpoint (DE Forces Viewpoint) is an extension of the original forces viewpoint (van Heesch et al., 2012b). As opposed to the original viewpoint, the DE Forces view contains an additional model kind, which governs the creation of models describing forces and stakeholders' assigned priorities to forces.

Furthermore, views using this viewpoint make explicit the relationships between (candidate) architecture decisions and the forces that influenced the architect when making the decisions out of multiple alternatives. A *force*, as we define it is "is any aspect of an architectural problem arising in the system or its environment (operational, development, business, organizational, political, economic, legal, regulatory, ecological, social, etc.), to be considered when choosing among the available decision alternatives." (van Heesch et al., 2012b).

Forces originate from several sources, most importantly from requirements, business constraints, and architecture principles, but also from other "intentions" imposed upon the system; including personal preferences or experience of the architect(s) and the development team; and business goals such as quick-time-to-market, low price, or strategic orientations towards specific technologies. In software platform development, many forces stem from stakeholders of the software products that plan to use the platform as a basis for their systems.

Table C.11
Concerns of the DE Forces Viewpoint.

Code	Concern
C1	What forces F_j impact/influence decision D?
C2	What decisions D_k are influenced by force F?
C3	What forces F_l have conflicting influences on decision D?
C4	What decisions are dependent on decision D?
C5	What concerns C_m does decision D pertain to?
C6	What forces F_m need to be considered in the platform decisions?
C7	What weight W_b does force F have for stakeholder H?
C8	What key decisions D_c need to be made for the platform?
C9	What decisions D_q are subject to product variability?
C10	What decisions D_m are influenced by force weight WD_n?
C11	What is the optimal outcome for decision D in the context of forces F_q?
C12	What candidate solutions are considered for the platform?

Table C.12
Typical stakeholders of the DE Forces Viewpoint and their concerns.

Stakeholders	Concerns
Platform Software Architect	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12
Platform Software Engineer	C5, C6, C7, C9
Architecture Reviewer	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10 ,C11, C12
Product Manager	C8
Product Software Architect	C9, C12

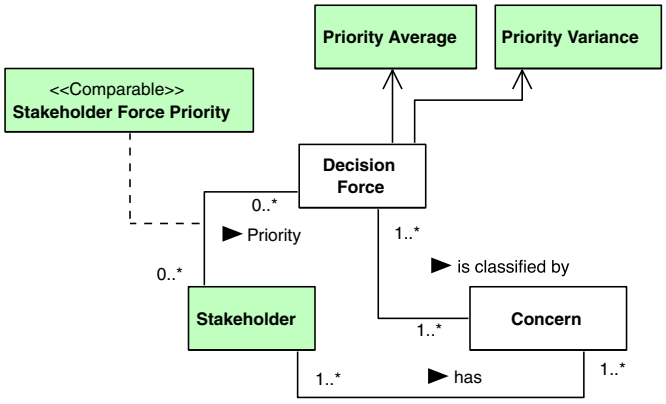

Fig. C.7. Force priority model kind.

C.1.1. Concerns

Table C.11 lists the decision-related concerns[8] framed by the decision forces viewpoint. These decision-related concerns are a subset of a larger set of concerns identified and described in Section 4.3.2.

C.1.2. Stakeholders

Views of the DE Forces Viewpoint are generally intended for stakeholders interested in decisions and decision forces part of design space exploration. The main stakeholders for this viewpoint are platform architects, but also key stakeholders of software products using the platform.

Table C.12 shows the stakeholders along with their main decision-related concerns with respect to the DE Forces Viewpoint. Similarly to the decision-related concerns, the full list of stakeholders can be found in Section 4.3.1.

C.1.3. Force priority model kind

The DE Forces Viewpoint consists of a two model kind: A model kind for force priorities, and a model kind for decision forces assignment. This section describes the Force Priority Model Kind. The next section elaborates on the Decision Forces Model Kind.

Fig. C.7 depicts the Force-priority Model Kind. It presents the conceptual elements for architecture models that adhere to it. This model is part of a shared metamodel, which is used by all viewpoints of the decision documentation framework. Together with well-defined correspondence rules, the shared metamodel ensures consistency among the views of different viewpoints. In the following, each of the elements used in Fig. C.7 is briefly described.

Models adhering to the Force Priority Model Kind contain information about existing forces, the architectural concerns they are classified into, and priorities assigned to forces by various stakeholders.

Forces are classified by one or more concerns. A stakeholder could for instance be concerned about integrability, while concrete forces classified by this concern could be "Integrate with existing

[8] The term *decision-related* concern is used to refer to concerns pertaining to decision documentation (as opposed to any other types of stakeholder concerns).

Table C.13
Excerpt from a decision forces model.

Concerns and Forces	<<Discarded>> Monotouch	<<Recommended>> Antenna
Concern: Deployability		
13.1. Specialized "App" stores	X	R
13.2. Private/public cloud solution	-3	R
13.3. Push updates	0	R
13.4. Vendor distribution channel	X	R
13.5. Replacement of devices	2	3
Concern: Usability		
17.1. User experience	3	3
17.2. Visual identity	3	3
17.3. Ease of adoption	0	-1
17.4. Consistency of user interface	-3	3

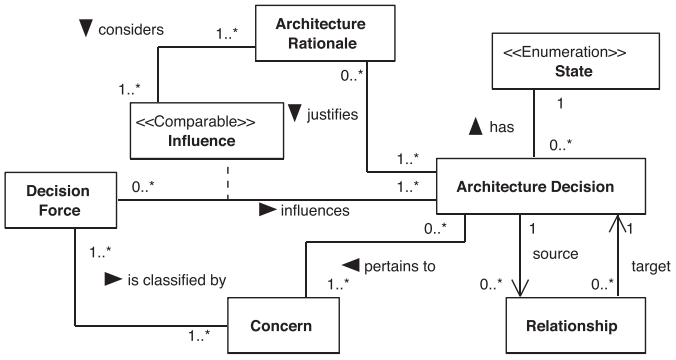

Fig. C.8. Decision forces model kind.

backend system", or to "Allow integration of security infrastructure".

The force priority association expresses a stakeholder's priority for a specific force. The priority is expressed by a value within a fixed value range that allows for a mathematical comparison (e.g. numbers from 1 to 5). These values are used to calculate an average, which expresses the average priority of a force as assigned by a all stakeholders. Additionally, the priority variance is calculated as a measure of the distribution of importance ratings.

For an example of a force priority model, please refer to Table 10.

C.1.4. Decision forces model kind

The decision forces model kind is identical to the original forces view model kind described in Heesch (2012). In the following, we will briefly summarize the main concepts. The classes *Decision Force* and *Concern* were already described as part of the force priority modelkind. A force has an influence on one or more architecture decisions. The influence relationship is qualified by the *Influence* class. The influence should be quantified and qualified, e.g. using a numeric scale from -3 to $+3$ expressing if a force has a very negative influence on a decision (-3), or a very positive $(+3)$, for instance. The influence of the forces on an architecture decision should be considered in the decision's *Architecture Rationale*. In platform design space exploration, architecture decisions can have one of the following states (for a complete set of decision states, please refer to van Heesch et al., 2012a):

Tentative This state is used for decisions which are seriously considered by the architect.

Discarded A discarded decision is a formerly tentative decision that was not decided, e.g. because it was prevented by some major forces, or a design option that was not chosen among the available design options.

Recommended A solution that is recommended for the platform.

Note that, as opposed to the original viewpoint framework, no decision has the state *Decided*. This is because the result of a platform design space exploration is not an application-specific architectural design, but a collection of architecture knowledge items that serves the platform architects as input for the concrete architecture. Eventually, relevant parts of the models have to be transferred to the platform architecture decisions models.

Table C.13 shows a simplified excerpt from a decision force model created during the study. It only shows forces classified by the concerns deployability and usability. Furthermore, only the two candidate decisions *Monotouch* and *Antenna* are shown. The influence scale used by ABB during the study was -3 for very negative

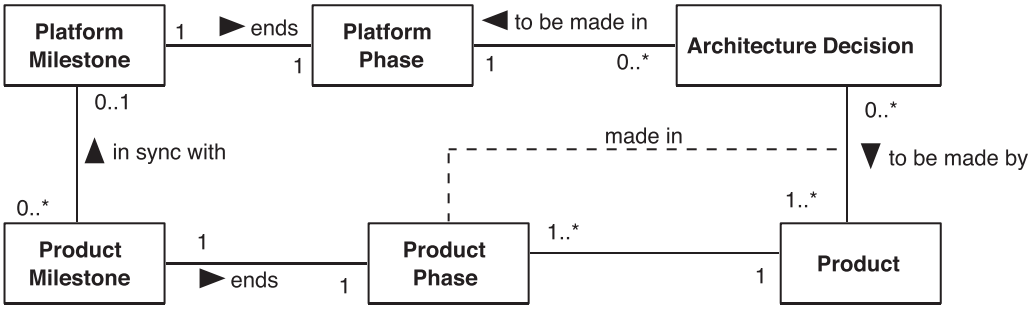

Fig. C.9. Decision roadmap model kind.

Table C.14
Concerns of the decision roadmap viewpoint.

Code	Concern
C4	What decisions are dependent on decision D?
C8	What key decisions D_c need to be made for the platform?
C9	What decisions D_q are subject to product variability?
C15	What is an efficient decision roadmap for the platform?

Table C.15
Typical stakeholders of the Decision roadmap viewpoint and their concerns.

Stakeholders	Concerns
Platform software architect	C4,C8,C9,C15
Platform product owner	C15
Product manager	C9,C15
Product software architect	C9, C15

influence to +3 for very positive influence. Additionally, the letter "R" represents that a force excludes a candidate solution from being recommended. In the example, force 13.1 about specialised app stores excludes Monotouch as indicated by the letter "X". Furthermore, the letter "R" indicates that a forces requires a specific candidate solution to be chosen, because it is the only solution out of all considered alternatives that adheres to the force (in the example, several forces require Antenna to be chosen). In the example Antenna has the state recommended, because it is the most convincing solution among the considered alternatives.

C.2. Decision roadmap viewpoint

Views of the decision roadmap viewpoint provide a decision roadmap for the platform engineering project and the product projects that plan to build up on the platform. The primary objective of the viewpoint is to match short-term and long-term goals of the platform stakeholders and the stakeholders of the software products. It provides a planning framework that can be used by project managers and architects to identify project dependencies, key decisions to be made, and to synchronize development efforts in a way that gained knowledge can be optimally exploited and risks can be minimised on a global level.

C.2.1. Concerns

Table C.14 shows the concerns framed by the decision roadmap viewpoint.

C.2.2. Stakeholders

Decision roadmap views are non-technical views meant to provide decision support for architects and managers. Table C.15 maps the concerns framed to typical stakeholders.

C3. Model kind

The decision roadmap viewpoint has only one model kind that integrates with the model kinds of all other decision viewpoints. It shows concepts related to the roadmap for the software platform itself (Platform Milestone, Platform Phase) and concepts related to the software products that use the platform (Product Milestone, Product Phase, Product). As the platform development is typically done in parallel to the product development, the product phases are independent of the platform phases. However, the platform development needs to be synchronised with certain milestones of the product to make sure that it does not delay the products release. Architecture decisions that are not subject to product variability are made as part of the platform project, whereas variability decisions are made for each product individually in the respective

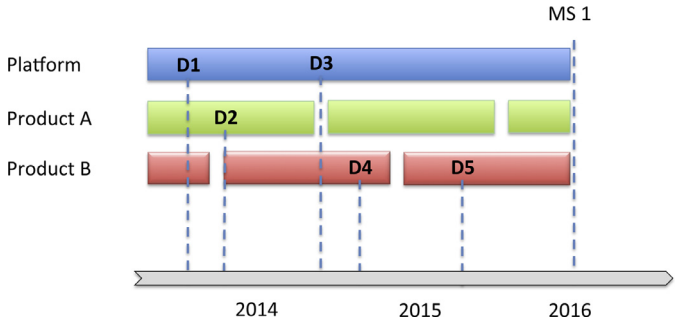

Fig. C.10. Conceptual example of decision roadmap view.

product phases. Fig. C.10 shows a conceptual example of a decision roadmap. The decision roadmap created during the study is confidential and has a level of complexity that is not beneficial for exemplifying the view.

In the figure, the platform project, as well as each product development project are represented by horizontally arranged bars. Breaks between the bars represent milestones in the product development projects. The view shows two architecture decisions that need to be made for the platform (D1 and D3), and several product-specific architecture decisions caused by variability points (D2,D4, and D5). Additionally, the view contains a horizontal time bar at the bottom. The platform milestone MS1 is in sync with milestones of product A and B.

Appendix D. Question guide for interviews conducted to elicit initial requirements.

The following questions were used as a guide to elicit requirements from the platform stakeholders.

D.1. General questions

1. What does the mobile strategy look like?
2. Which parts of the product are considered to be expanded with mobile solutions?
3. What are the business drivers that motivate the development of mobile solutions?
   (a) What does the business environment look like?
   (b) What are the main market differentiators?
   (c) Who are the main stakeholders (roles) considered in mobile solutions?
   (d) Which are the driving requirements that motivate the development of mobile solutions? e. What kind of business model do you envision for mobile solutions?
4. What kind of business model do you envision for mobile solutions?
5. Which quality attributes (e.g., performance, availability, security, etc.) are in focus?
   (a) From which business needs are these quality attributes derived?
   (b) How are they quantified in requirements?
6. What are the business constraints when developing mobile solutions?
   (a) What is the time plan (time-to-market)?
   (b) What are the main customer demands?
   (c) Are there any specific standards that need to be followed?
   (d) Are there any other constraints and concerns, e.g., cost, regulations?

7. Wha are the technical constraints when developing mobile solutions?
   (a) Are there any constraints on the usage of commercial off-the-shelf products?
   (b) Are there any specific requirements on interoperation with other systems?
   (c) What kind of devices should be supported? Consumer devices or specialized hardware?
   (d) Are there any specific requirements on hardware/software platform?
8. What are the main use cases/ scenarios of using the mobile solutions?
9. Any data for NPV calculation, e.g., revenue, market share?
10. Who is the architect/technical lead that we should interview to learn more?

D.2. For those who have ready-developed mobile solutions

Apart from eliciting requirements, the following set of questions is used to collect initial candidate solutions and the rationale for choosing them.

1. How does the licensing strategy look like?
2. What are the main architectural requirements?
3. How does the mobile version differ from the traditional product, e.g., scope of functionality, quality attributes in focus, etc?
4. What kind of App development approach do you take? Native, embedded webkit, interpreter, server side adaptation, etc.
5. Which cross-platform framework/enterprise integration platform is used?
   (a) Which are the main drivers that motivate the selection of the platform?
   (b) What are the main criteria considered when choosing among the diverse platforms?
   (c) What are the main concerns and tradeoffs, if any, with regard to the selected platform?
6. How does the deployment and update strategy look like?

Appendix E. Questionnaire used to gather force priorities from platform stakeholders

The following questionnaire was used to gather force (requirement) priorities from stakeholders. Detailed requirements are obfuscated because of company disclosure rules.

Requirements	Value(1 for the lowest prioritized, 7 for highest prioritized)
1. Integrability	
1.1. XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
1.2. XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
1.3. XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
2. Interopability	
2.1. Interface with other applications on device XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
2.2. Access to device capabilities XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
3. Security XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
4. Usability/user experience XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
4.1. Visual identity XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
4.2. Ease of adoption XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
4.3. Consistent user interface XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
5. Availability XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
6. Performance XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
7. Compatibility XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
8. Maintainability XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
9. Reusability XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
10. Reliability XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
11. Flexibility XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
12. Evolvability	
12.1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
12.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
13. Deployability	
13.1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
13.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	

13.3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
13.4. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
13.5. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
14. Device Management	
14.1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
14.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
14.3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
14.4. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
15. Network	
15.1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
15.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
15.3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
15.4. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
16. Development constraints	
16.1.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
16.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17. Costs	
17.1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17.1.1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17.1.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17.1.3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17.2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17.3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
17.4. Service cost	

Are there any additional requirements that you want to add here? Please fill in here if any, and give your weighting points.	

Categories	values (1 - 100)
1. Integrability	
2. Interopability	
3. Security	
4. Usability/user experience	
5. Availability	
6. Performance	
7. Compatibility	
8. Maintainability	
9. Reusability	
10. Reliability	
11. Flexibility	
12. Evolvability	
13. Deployability	
14. Device Management	
15. Network	
16. Development constraints	
17. Costs	
Points left to be assigned	**100**

2.3. Device diversity
The mobile platform should be able to deal with a high diversity in mobile devices (including ruggedized device) and mobile operating systems.
Put 100 points into the following matrix depends on OS and type of devices.

	Android	iOS	Windows
Phone			
Tablet			
Laptop			

Appendix F. Question guide used during the retrospective interviews.

1. Was the design exploration process clear and reasonable in terms of
   (a) How we came to the main concerns?
   (b) How we came to the technology alternatives?
   (c) How we identified decision points and their dependencies?
2. How is the understandability of the decision views? (i.e., if stakeholders are able to interpret the decision views themselves after an introduction, or do they require someone to explain the captured information to them.)
3. Did we focus on the right concerns for you? If not, which concerns were missing?
4. Did we focus on the right technology alternatives? If not, which ones were missing?
5. Are the tradeoffs between the different technology alternatives clear? If not, which ones were unclear and in how far were they unclear?
6. Are the tradeoffs between the different architectural style alternatives clear? If not, which ones were unclear and in how far were they unclear?
7. In how far did the decision views help you making decisions and in how for will they help when it comes to mobile applications in the future?
8. How do the results impact your strategic roadmaps? What information was missing?
9. Any other feedback and suggestions for improvement?

References

Adolph, S., Hall, W., Kruchten, P., 2011. Using grounded theory to study the experience of software development. Empirical Software Eng. 16, 487–513.

Alexander, C., 1979. The Timeless Way of Building. Oxford University Press.

Bachmann, F., Clements, P.C., 2005. Variability in Software Product Lines. Software Engineering Institute, Pittsburgh, USA Technical report. CMU/SEI-2005-TR-012

Barbacci, M.R., Ellison, R., Lattanze, A.J., Stafford, J.A., Weinstock, C.B., Wood, W.G., 2003. Quality Attribute Workshops (QAWs). Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA Technical report. CMU/SEI-2003-TR-016

Basili, V.R., Caldiera, G., Rombach, H.D., 1994. The Goal Question Metric Approach. Encyclopedia of Software Engineering. John Wiley & Sons, Inc..

Baskerville, R.L., 1999. Investigating information systems with action research. Commun. AIS 2, 4.

Baskerville, R.L., Wood-Harper, A.T., 1996. A critical perspective on action research as a method for information systems research. J. Inf. Technol. 11, 235–246.

Bass, L., Clements, P., Kazman, R., 2012. Software Architecture in Practice, third ed. Addison-Wesley.

Bass, L., Klein, M., Bachmann, F., 2002. Quality Attribute Design Primitives and the Attribute Driven Design Method. In: van der Linden, F. (Ed.), Software Product-Family Engineering. In: Lecture Notes in Computer Science, vol. 2290. Springer Berlin Heidelberg, pp. 169–186. doi:10.1007/3-540-47833-7_17.

Bosch, J., 2004. Software Architecture: The Next Step. In: Oquendo, F., Warboys, B., Morrison, R. (Eds.), Software Architecture. In: Lecture Notes in Computer Science, vol. 3047. Springer Publishing Company, pp. 194–199.

Brown, A.W., Wallnau, K.C., 1996. A framework for evaluating software technology. IEEE Software 13, 39–49.

Buschmann, F., Henney, K., Schmidt, D., 2007. Pattern-Oriented Software Architecture: On Patterns and Pattern Languages, 5. John Wiley & Sons, Inc..

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., 1996. Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons, Inc..

Chen, L., Babar, M.A., 2010. Variability Management in Software Product Lines: An Investigation of Contemporary Industrial Challenges. In: Proceedings of the 14th international conference on Software product lines: Going Beyond. Springer-Verlag, pp. 166–180.

Clark, W., Finley, I., Chuang, S., 2012. Magic Quadrant for Mobile Application Development Platforms. Gartner-Burton IT1 Research.

Easterbrook, S., Singer, J., Storey, M.-A., Damian, D., 2008. Selecting Empirical Methods for Software Engineering Research. In: Guide to advanced empirical software engineering. Springer, pp. 285–311.

Engelsman, W., Wieringa, R., 2012. Goal-oriented Requirements Engineering and Enterprise Architecture: Two Case Studies and Some Lessons Learned. In: Proceedings of the 18th international conference on Requirements Engineering: Foundation for Software Quality, Springer-Verlag, pp. 306–320.

Figueira, J., Greco, S., Ehrgott, M., 2005. Multiple Criteria Decision Analysis: State of the Art Surveys. Springer.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Garlan, D., Monroe, R., Wile, D., 1997. Acme: An Architecture Description Interchange Language. In: Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research. IBM Press, pp. 7–22.

Glaser, B., Strauss, A., 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research.. New York. Aldine Publishing.

Gross, D., Yu, E., 2001. From non-functional requirements to design through patterns. Requirements Eng. 6, 18–36.

van Heesch, U., Avgeriou, P., Hilliard, R., 2012. A documentation framework for architecture decisions. J. Syst. Software 85, 795–820.

van Heesch, U., Avgeriou, P., Hilliard, R., 2012. Forces on Architecture Decisions - a Viewpoint. In: Proceedings of the Joint 10th Working IEEE/IFIP Conference on Software Architecture & 6th European Conference on Software Architecture, IEEE. xx–xx

van Heesch, U., Eloranta, V.-P., Avgeriou, P., Koskimies, K., Harrison, N., 2014. Decision-centric architecture reviews. IEEE Software 31, 69–76.

Heesch, U.v., 2012. Architecture Decisions: The Next Step: Understanding, Modeling, Supporting and Reviewing Architecture Decisions. University of Groningen Ph.d. thesis.

Heesch, U.V., Avgeriou, P., Tang, A., 2013. Does decision documentation help junior designers rationalize their decisions? a comparative multiple-case study. J. Syst. Software 86, 1545–1565.
Hevner, A., March, S., Park, J., Ram, S., 2004. Design science in information systems research. MIS Q. 28, 75–105.
Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., America, P., 2007. A general model of software architecture design derived from five industrial approaches. J. Syst. Software 80, 106–126.
Hwang, C.-L., Yoon, K., 1981. Multi attribute decision making.. Oper. Res. 22, 22–34.
ISO/IEC/IEEE, 2011. ISO/IEC/IEEE 42010, Systems and Software Engineering — Architecture Description. ISO/IEC/IEEE.
Jansen, A., 2008. Architectural design decisions. University of Groningen, The Netherlands Ph.d. thesis.
Jansen, A., Bosch, J., 2005. Software Architecture as a Set of Architectural Design Decisions. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, pp. 109–120.
Jansen, A., Wall, A., Weiss, R., 2011. Techsure - a Method for Assessing Technology Sustainability in Long Lived Software Intensive Systems. In: Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2011). IEEE, pp. 426–434. doi:10.1109/SEAA.2011.66.
Kruchten, P., 1995. The 4+ 1 view model of architecture. IEEE Software 12, 42–50.
Kruchten, P., 2004. An Ontology of Architectural Design Decisions in Software Intensive Systems. In: Proceedings of the 2nd Groningen Workshop on Software Variability, pp. 54–61.
Kruchten, P., 2008. What do software architects really do? J. Syst. Software 81, 2413–2416.
Mack, N., Woodsong, C., MacQueen, K., Guest, G., Namey, E., 2005. Qualitative Research Methods: A Data Collector's Field Guide. FLI.
MacLean, A., Young, R., Bellotti, V., Moran, T., 1991. Questions, options, and criteria: elements of design space analysis. Human-Comput. Interact. 6, 201–250.
Manteuffel, C., Tofan, D., Avgeriou, P., Koziolek, H., Goldschmidt, T., 2016. Decision architect–a decision documentation tool for industry. J. Syst. Software 112, 181–198.
March, S., Smith, G., 1995. Design and natural science research on information technology. Decis. Support Syst. 15, 251–266.
McIntyre, A., 2008. Participatory Action Research, 52. Sage.
McKay, J., Marshall, P., 2001. The dual imperatives of action research. Inf. Technol. People 14, 46–59.
Miles, M.B., Huberman, A.M., Saldaña, J., 2013. Qualitative Data Analysis - a Methods Sourcebook, third ed. Sage Publications, Inc..
Mobile, V., 2012. Cross-platform developer tools 2012 - the quintessential report on the landscape of 100+ cross-platform developer tools. http://www.visionmobile.com/product/cross-platform-developer-tools-2012/.
Nuseibeh, B., 2001. Weaving together requirements and architectures. Computer 34, 115–117.
Perry, D., Wolf, A., 1992. Foundations for the study of software architecture. ACM SIGSOFT Software Eng. Notes 17, 40–52.
Phaal, R., Farrukh, C., Probert, D., 2010. Roadmapping for Strategy and Innovation-aligning Technology and Markets in a Dynamic World. University of Cambridge, Institute for Manufacturing.
Pohl, K., Böckle, G., Linden, F., 2005. Software Product Line Engineering: Foundations, Principles, and Techniques. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.
Reason, P., Bradbury, H., 2001. Handbook of Action Research: Participative Inquiry and Practice. Sage.
Ribeiro, R.a., Moreira, A.M., van den Broek, P., Pimentel, A., 2011. Hybrid assessment method for software engineering decisions. Decis. Support Syst. 51, 208–219.
Rozanski, N., Woods, E., 2005. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Eng. 14, 131–164.
Saaty, T., 1980. The Analytic Hierarchy Process. McGraw-Hill.
Schmidt, D., Buschmann, F., 2003. Patterns, Frameworks, and Middleware: Their Synergistic Relationships. In: Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, pp. 694–704.
Seddon, P.B., Scheepers, R., 2012. Towards the improved treatment of generalization of knowledge claims in is research: drawing general conclusions from samples. Eur.J.Inf.Syst. 21, 6–21.
Shekaran, C., Garlan, D., Jackson, M., Mead, N., Potts, C., Reubenstein, H., 1994. The Role of Software Architecture in Requirements Engineering. In: Proceedings of the First International Conference on Requirements Engineering, IEEE, pp. 239–245.
Shu-Jen, C., Hwang, C.L., Hwang, F.P., 1992. Fuzzy Multiple Attribute Decision Making: Methods and Applications. Springer-Verlag.
Shull, F., Singer, J., Sjøberg, D., 2008. Guide to Advanced Empirical Software Engineering. Springer Publishing Company.
Sieber, J.E., 2001. Protecting research subjects, employees and researchers: implications for software engineering. Empirical Software Eng. 6, 329–341.
Sjoberg, D.I., Dyba, T., Jorgensen, M., 2007. The Future of Empirical Methods in Software Engineering Research. In: 2007 Future of Software Engineering, IEEE Computer Society, pp. 358–378.
Strauss, A., 1987. Qualitative Analysis for Social Scientists. Cambridge Univ Pr.
Tang, A., Avgeriou, P., Jansen, A., Capilla, R., Babar, M.A., 2010. A comparative study of architecture knowledge management tools. J. Syst. Software 83, 352–370.
Taudes, A., Feurstein, M., Mild, A., 2000. Options analysis of software platform decisions: a case study. MIS Q. 24, 227–243.
Tripp, D., 2005. Action research: a methodological introduction. Educacao e pesquisa 31, 443–466.
Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. IEEE Software 22, 19–27.
Urquhart, C., Lehmann, H., Myers, M., 2010. Putting the theory back into grounded theory: guidelines for grounded theory studies in information systems. Inf.Syst.J. 20, 357–381.
Weiss, R., Krogmann, K., Durdik, Z., Stammel, J., Klatt, B., Koziolek, H., 2012. Sustainability Guidelines for Long-living Software Systems. In: Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM), IEEE Computer Society, pp. 517–526.
Wieringa, R., 2009. Design Science as Nested Problem Solving. In: Proceedings of the 4th international conference on design science research in information systems and technology, ACM, pp. 8–20.
Wieringa, R., Moralı, A., 2012. Technical Action Research as a Validation Method in Information Systems Design Science. In: Proceedings of the 7th international conference on Design Science Research in Information Systems: advances in theory and practice, Springer-Verlag, pp. 220–238.
Wieringa, R.J., 2014. Design Science Methodology for Information Systems and Software Engineering. Springer Heidelberg New York Dordrecht London.
Yin, R.K., 2009. Case Study Research: Design and Methods, Applied Social Research Methods Series, Vol 5, fifth ed. Sage Inc..
Yoon, K.P., Hwang, C.-L., 1995. Multiple Attribute Decision Making: An Introduction. SAGE Publications, Incorporated. URL: http://books.google.se/books?id=Fo47SWBuEyMC&lpg=PR5&ots=etl7ItKxBW&lr&hl=nl&pg=PR5#v=onepage&q&f=false
Zdun, U., 2007. Systematic pattern selection using pattern language grammars and design space analysis. Software-Practice Experience 37, 983–1016.
Zdun, U., Capilla, R., Tran, H., Zimmermann, O., 2013. Sustainable architectural design decisions. IEEE Software 30, 46–53.
Zimmermann, O., Zdun, U., Gschwind, T., Leymann, F., 2008. Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method. In: Proceedings of the seventh Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE Computer Society, Los Alamitos, CA, USA, pp. 157–166. doi:10.1109/WICSA.2008.19.

Dr. Uwe van Heesch is a lecturer and researcher at the HAN University of Applied Sciences in Arnhem, the Netherlands. Before joining the HAN, Uwe worked as a project manager and lead architect at Capgemini Germany. He received his PhD from the University of Groningen in 2012. His research interests include (agile) software architecture, software design for mixed-paradigm applications, architecture decision modeling, and architecture evaluation. Dr. van Heesch is a passionate member of the European pattern community and was (co-) organizer and reviewer for several scientific conferences. He has published many peer-reviewed articles in international journals and conference proceedings.

Dr. Anton Jansen is a senior architecture consultant at Philips Innovation services since 2015. Before this, he was senior scientist at ABB corporate research in the software architecture & usability (SARU) group in Vasteras, Sweden. Between 2002 and 2009, he was a member of the Software Engineering and Architecture (SEARCH) research group at the University of Groningen, the Netherlands, where he received a Ph.D. in Software Architecture in 2008. Between 2006 and 2008, he worked as a postdoc on the Dutch Joint Academic and Commercial Quality Research & Development project GRIFFIN. His research interests are software architecture and architectural knowledge.

Dr. Hongyu Pei Breivold is a Principal Scientist at ABB Corporate Research. She obtained her Ph.D in Software Engineering at Mälardalen University, Sweden, in 2011, with topic on software architecture evolution. She is also an adjunct researcher at Mälardalen University. She has published more than 30 peer-reviewed articles in journals, conferences and workshops. She is active in academia as program committee member, track co-chair, and industry-research chair in international conferences. Her main research interests are software evolution, Cloud Computing, Internet-of-things technologies and their applications in industry.

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

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