Architectural Decision Forces at Work: Experiences in an Industrial Consultancy Setting

Julius Rueckert
Andreas Burger
Heiko Koziolek
firstname.lastname@de.abb.com
ABB Corporate Research Center
Germany

Thanikesavan Sivanthi
Alexandru Moga
Carsten Franke
firstname.lastname@ch.abb.com
ABB Corporate Research Center
Switzerland

ABSTRACT

The concepts of *decision forces* and the *decision forces viewpoint* were proposed to help software architects to make architectural decisions more transparent and the documentation of their rationales more explicit. However, practical experience reports and guidelines on how to use the viewpoint in typical industrial project setups are not available. Existing works mainly focus on basic tool support for the documentation of the viewpoint or show how forces can be used as part of focused architecture review sessions. With this paper, we share experiences and lessons learned from applying the decision forces viewpoint in a distributed industrial project setup, which involves consultants supporting architects during the re-design process of an existing large software system. Alongside our findings, we describe new forces that can serve as template for similar projects, discuss challenges applying them in a distributed consultancy project, and share ideas for potential extensions.

CCS CONCEPTS

• **Computer systems organization → Architectures**; • **Software and its engineering → Designing software**.

KEYWORDS

Software Architecture, Architectural Decision Forces, Design Decisions, Experience Report.

ACM Reference Format:
Julius Rueckert, Andreas Burger, Heiko Koziolek, Thanikesavan Sivanthi, Alexandru Moga, and Carsten Franke. 2019. Architectural Decision Forces at Work: Experiences in an Industrial Consultancy Setting. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3338906.3340461

ESEC/FSE '19, August 26–30, 2019, Tallinn, Estonia

ACM ISBN 978-1-4503-5572-8/19/08…$15.00
https://doi.org/10.1145/3338906.3340461

1 INTRODUCTION

Decision making in the context of software architecture is known to be a complex process which goes beyond selecting solution alternatives based on technical requirements. In practice, decisions have to be made according to the context in which the software is designed, implemented, operated, and maintained. In addition to technical requirements, the project setup, the time to market of the product, a company's strategic preferences toward certain technologies, and even an architect's experience with a certain technology might play an important role. Van Heesch et al. [10] refer to any of such influencing factors as *decision forces* and propose to document them as part of a dedicated architectural viewpoint based on ISO/IEC/IEEE 42010 [2]. This way, they claim, decisions become more transparent and traceable. But even without an extensive documentation of forces for all decisions within a software architecture, discussing forces for the most critical decisions as part of an architecture review can be of high value as shown in [11].

Transparency and documenting rationales of architectural decisions is crucial especially for software systems with a long lifespan during which decisions might have to be reconsidered due to evolving decision forces: new technologies become available, operative requirements change, architects build up new expertise, architects leave or join the project, or the strategy of the company changes. Another scenario in which transparency is of high importance is a setup that we refer to as *consultancy setup* in the remainder of the paper. In such a setup, a system is initially designed by a separate team (the consultants) than the one taking care of its productive implementation (the business unit). This is a typical scenario even inside companies, where technology development requires dedicated research or pre-development units that act as consultants to the business units on a short-term project base. For such a setup, the reasoning behind architectural decisions needs to be completely transparent such that they can easily be re-evaluated in an operational context that might be different from the context during design time. Such differences are hard to avoid and can be caused by the fact that not all decision forces were observable during the design or the forces changed between the time of designing and the implementation as part of a product.

In this paper, we describe the lessons learned of using the forces architectural viewpoint proposed by Van Heesch et al. [10] and study its applicability to a consultancy setup. It was applied in a 6-month project in which five consultants in two teams worked with multiple business unit architects on the architecture evolution of an established industrial control system. This work is motivated

by the observation that documenting the decision rationales for a set of design decisions during this project seemed inherently incomplete when trying to summarize the rationales behind the decisions. Initially, we used a minimalistic and pragmatic approach to capture the rationales based on the Y-model [12] that has been used for many projects before. However, in this project, the consultancy teams noticed a disconnect between the decision and rationales documented. To better understand this disconnect and to close the gap, we decided to study and document the decision forces for important decisions that were made. During this process, we noticed that applying the viewpoint from [10] was helpful but showed some challenges caused by the consultancy setup and multiple teams collaborating remotely. We also noticed that the forces discussed so far in the context of the viewpoint were not covering the consultancy setup, which however is common in a corporate environment.

To encourage the use of the forces viewpoint in future industrial projects, we present lessons learned based on the work with the viewpoint as consultants during the project. In addition, we describe additional forces that were observed specific to the consultant setup and that were not discussed in the context of the decision forces viewpoint before. Some of the discussed forces play a crucial role for key design decisions and close the gap in fully understanding the decision process. Finally, we also discuss open questions that could, e.g, help driving the development of adequate tools to support distributed teams in using the viewpoint.

The remainder of the paper is organized as follows. First, in Section 2, the concept of decision forces is introduced as a background for the subsequent sections. Second, we present experiences gained and lessons learned while using the forces architectural viewpoint in a distributed consultancy setting of a real industrial project. In Section 4, we discuss lessons that we learned from applying the viewpoint and open questions we identified. Subsequently, we discuss related work in Section 5 and conclude the paper in Section 6.

2 DECISION FORCES

The notion of architectural decision forces, short *forces*, was introduced by Van Heesch et al. in [10]. In this work, the authors propose the *forces viewpoint*, an addition to the framework of architectural viewpoints they introduced for the documentation of software architectures in [9] based on ISO/IEC/IEEE 42010 [2]. This framework includes four viewpoints: the decision detail viewpoint, the decision relationship viewpoint, the decision chronology viewpoint, and the decision stakeholder involvement viewpoint. The additional forces viewpoint is motivated by the desire to make design decisions more transparent and document decision rationales. It complements the other viewpoints as it acknowledges that architects make decisions not only based on technical arguments but also incorporate many more factors.

Similar to a physical object in space that moves or comes to rest depending on the sum of all externally applied forces, decisions made by software architects are a result of the interactions between a number of observable forces. These can be technical requirements, stakeholder concerns, a development team's experience with a specific technology, their expertise, a business-related concern, or any other factor that influenced the decision. Thereby, forces share some commonalities with the notion of *design constraints* as discussed in [8] but are more openly defined which made them more appealing for our project.

In addition to the examples mentioned by Van Heesch et al., by now, it is well understood that also human factors play an important role in architectural decision making [7] as well as background and judgement of individuals making the decisions [3]. Furthermore, it has been acknowledged that decisions can be subject to cognitive biases, reasoning fallacies, and political games as pointed out by Kruchten [5]. While the latter might be hard to capture in concrete forces, in our opinion, they should be considered when exploring and eliciting forces observed by the team of involved architects for individual decisions. Some of these factors can jointly or individually be observable in form of a force that might be hard to capture at first. Overall, we decided to use the flexible and open notion by Van Heesch et al. to study the decisions in our project, expressing any factor that we observed and that can be attributed to one of the above categories as forces.

3 EXPERIENCE REPORT

In the following, we share our experience during applying the forces viewpoint. For this, we first motivate the project setup and introduce the high-level software architecture. Subsequently, the decision making process in the project setup is discussed, the observed decision forces are presented, and their application to two exemplary design decisions are explained.

3.1 Distributed Consultancy Setup

The project setup considered in this work is typical for corporate research and development (R&D) environments. In our case, it consisted of two consultancy teams that were working remotely from two locations and two architects from the related business units at another two different locations. The consultants were experts from two corporate research centers of the same company. In such a case, the consultancy work can include running pre-studies, evaluating or developing new technologies, proposing initial system architectures, supporting the redesign of existing software systems, or implementing first proof-of-concept systems to show the feasibility of a proposed design. We assume that most aspects of the setup considered here would also be applicable if the consultants are external to the company. In the discussed case, they belonged to the same company but not the same business unit. The business unit architects decided to mandate the consultants with the task to provide input to a planned architecture evolution of an existing software system. The project duration was six month and involved five consultants. For their work, the consultancy teams were granted access to the source code of the software system and its documentation. In addition, they got access to expert developers which are familiar with the system.

Figure 1 illustrates the setup as described above. Architectural decisions in the redesign were made by the consultancy teams according to technical requirements elicited at the beginning of the project. The design was presented to the business unit architects and discussed in regular telephone conferences. The business unit architects expressed their opinions and potential concerns during these meetings and approved or overruled the decisions taken by

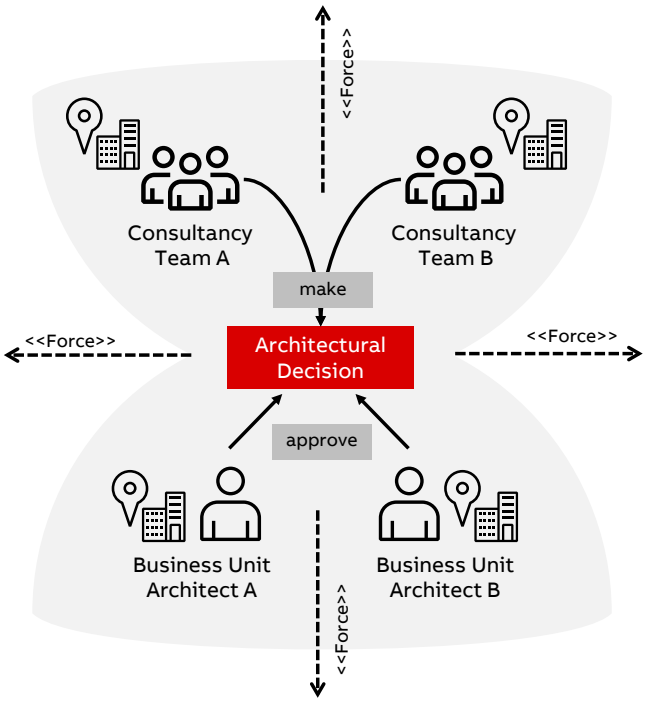

Figure 1: A typical distributed consultancy setup as observed in the reported project with two consultancy teams and two business unit architects.

the consultancy team. Wherever possible and valuable, the consultancy teams ran experiments to study non-functional properties and implications of design decisions. This included, e.g., performance measurements of chosen technologies, libraries, middlewares, and implementation alternatives.

3.2 High-level Software Architecture

Figure 2 illustrates a high-level overview of the architecture evolution done in this project. The goal was to achieve a decoupling of tightly coupled applications and a database of an existing large-scale and established industrial control system with a large customer base. The control system consists of a central database and applications that can have three different roles: (1) collecting data from the field, such as sensors, and writing them to the database, (2) operating on data from the database and writing data back to it, and (3) reading data from the database to perform analytics or display information to a human operator.

The tight coupling of applications to the database was a result of timing requirements and dates back to the time the system was initially designed (a few decades ago). With the redesign, a decoupling layer should be added as a step towards decoupling applications from the database implementation and deployment. The motivation to introduce such a layer was twofold: Firstly, a compatibility and translation layer should be added to avoid binary dependencies of the applications on a specific database version. A practical implication of resolving this direct dependency is that a better decoupling of the database and application development process can be achieved. Secondly, the layer should support an independent deployment of applications and database. This, in turn, enables a path towards modularization as well as resource allocation and scaling of independent application. For the design of the decoupling layer, this second objective implied a necessity to support a complete distributed deployment. For all changes to the architecture, a key requirement was that compatibility of existing applications in terms of the database interface and features they use shall be maintained as is. This also applies to the interface of the database.

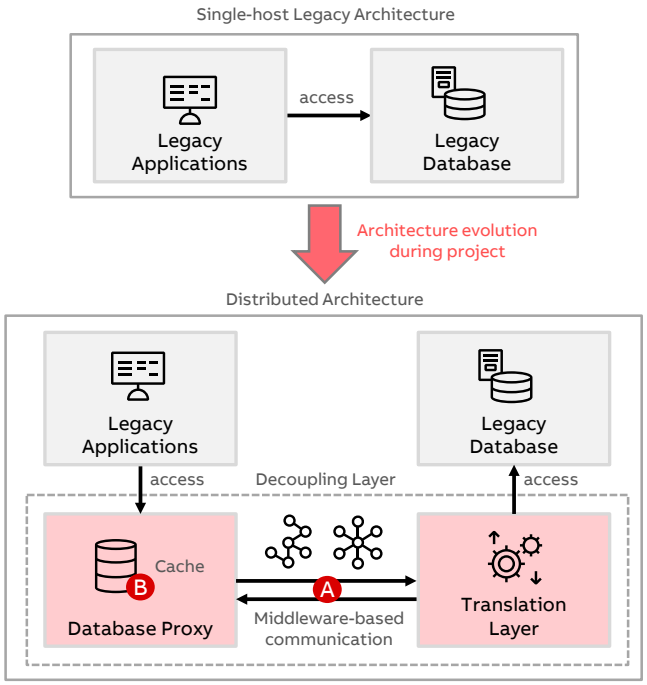

Figure 2: High-level overview of software architecture evolution in the presented project, including two exemplary architectural decisions: (A) the used communication middleware and (B) the used caching approach.

To enable a distributed setup, the decoupling layer was designed to consist of a database proxy component and a translation layer that translates calls of individual applications to the database. Instead of directly issuing database calls, the applications use a database proxy instance which provides a backward-compatible interface to the database. This way, legacy applications can be used without modifications, which was a key requirement stated by the business architects. In a distributed setup, applications would be always deployed together with a database proxy instance, while the translation layer and the database could be hosted separately. The communication between the database proxy and the translation layer was decided to make use of either an existing centralized or decentralized communication middleware. The decision for a specific middleware was a key architectural decision made in the project, which involved a detailed study of technology options by the consultancy team. To comply with the timing requirements of the applications for database access, the database proxy was decided to include a local cache. The decision for a specific caching

solution was a second relevant architectural decision, which involved a study of available technology and implementation options. Here, a number of additional requirements needed to be taken into account, especially concerning data access requirements of applications, data consistency among cache instances and the database, as well as the persistency of data.

Although the architecture presented here was motivated by the specific project, it is of a generic nature and not specific to a product of our company. Yet, it is sufficient to showcase our experience with the decision forces architectural viewpoint and the challenges we faced in the specific industrial use case.

3.3 Decision Making and Documentation

The project started off with an elicitation of functional and non-functional requirements that were ranked according to importance and criticality by the business unit architects. In total, there were about 20 requirements, covering concerns related to the desired decoupling of of legacy applications from the database, timing requirements, reliability, compatibility of legacy applications, and some specific features. Over the course of the project, about a quarter of these requirements and their rankings were adapted based on the input and results of the consultancy teams work. Some of these changes were caused by decision forces that were not intuitive to the consultancy teams at the beginning. The explicit analysis of decisions based on the notion of forces, however, helped to make these decisions more transparent. For this step, the consultancy team discussed selected decisions in retrospective. The team members expressed forces that they observed in addition to the requirements elicited at the beginning of the project. These forces were collected in an Excel sheet following the template used in [10]. As proposed by Van Heesch et al., the architecture-significant requirement were listed as the first set of decision forces, followed by a list of diverse additional forces collected from the teams.

It turned out to be difficult to discuss the relevance of each individual force on a selected architectural decision in an ad hoc manner. While the consultancy team made use of advanced tools for remote collaboration, such as shared whiteboards, the discussion of the viewpoint was time consuming. One observation during the discussion was that the perception of forces across the consultancy team members was very diverse first. As a result, discussing the relevance and names of individual forces themselves took more time than initially anticipated. Due to the rather subjective nature of some of the observed forces, a force-by-force discussion was necessary to agree on the meaning and terminology used for the forces. Even with only five consultants, this showed to be challenging because of the teams being remote to each other and the missing ability to discuss observations in parallel. Based on this experience, a better collaboration method specific to the elicitation of the forces would be desired. We decided to accept this limitation and decided to leverage the possibility of the two teams collaborating more closely in the individual locations. For consultancy setups with completely distributed teams, a face-to-face meeting could be an alternative option. In our case, the two teams continued the discussion and rating of the forces for individual decisions independently. This way, a more efficient face-to-face discussion and decision process were possible, which led to a speed up of the process. Afterwards, results were consolidated in a focused meeting in which the observations were already more consolidated and common terminologies were easier to establish. The steps taken for this distributed assessment are illustrated in Figure 3.

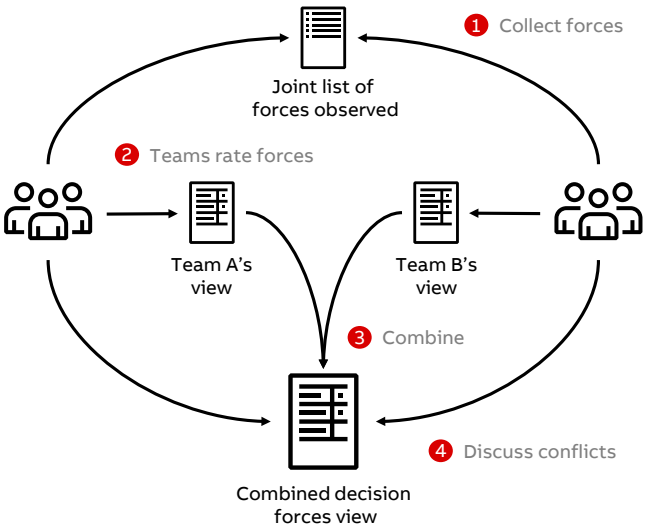

Figure 3: The steps taken to produce a combined decision forces view by the distributed consultancy team.

First, an initial list of observed forces was compiled based on the input of both teams and briefly discusses in a joint session. In our case, an initial list was proposed by one team and then refined together with the other team. After the complete team had a rough overview and understanding of all collected forces and the decision forces concept in general, the two consultancy teams individually continued the discussion and performed the rating for all selected decisions. For the rating, the scale by Van Heesch et al. from [10] was used, ranging from ++ (a force strongly supports a specific decision alternative) to - - (a force strongly opposes an alternative) and *X* (a decision alternative is prevented by a force). In a third step, the individual ratings of the teams were combined into a single Excel sheet. Instead of a single rating, the sheet indicated the two ratings of the teams side by side. In a final step, discussions focusing solely on those ratings that both teams differed in were initiated. During these discussions, inconsistent or unjustified decisions became obvious to the complete team. We observed the latter as one of the main strengths of the overall process related to the decision viewpoint. Based on the rating difference and the discussion, existing forces were rephrased, new ones were added, or ratings were revised by the individual teams. In case of an unsatisfying outcome or remaining questions on why a decision was made, we propose to enter a discussion with the business unit architects to identify additional forces that are not directly observable by the consultants.

For the actual rating process, an important observation was that it took the teams and its members some time to agree on a common understanding of the rating scale. This should not be underestimated when using the method with a larger team, especially if the team collaborates remotely. We observed, e.g., that while the team ratings showed similar tendencies, they often differed by one rating

step. This is not considered a problem but again showed the overall subjective nature of the forces rating. A discussion led to the impression that forces were often observed differently by individual team members. To avoid multiple reduplications, we decided to acknowledge a certain degree of subjectivity of the observations and keep the individual team ratings. For the initial purpose of achieving transparency of decisions, the results seemed sufficient and clearly showed the relevant tendencies of forces to explain specific decisions.

Regarding the rating scale itself, a number of concerns were discussed during our project. For example, it was considered if the scale should be extended to better express the relative strength of forces among each other. This was for example the case when discussing the different technology options for the communication middleware used as part of the decoupling layer, where the interest of the architects in particular parts of the architecture and the existing experience with one technology led to a rather quick decision in favor of that technology. In this particular case, the rating discussion was resolved by making use of the *X* rating to indicate that a decision against an alternative was taken because of a single dominating force. Thereby, this force canceled out other strong forces. Making this observation explicit and discussing it in the team, was an important step for some dissensus decisions to achieve transparency and document decision rationales. In retrospective, it seems obvious what led to some of the decisions when taking into account all observed forces. During the time of discussion and shortly after a decision was made, there was not always a consensus regarding individual decisions. One explanation could be that establishing some of the more objective forces involved a substantial amount of effort, e.g., to execute a performance measurement and comparison of individual technology alternatives. This was the case for the decision for a communication middleware. Here, quantitative performance measurements were available for two of the alternatives that were shadowed by other forces in the final decision. Accepting that such an objective force can be canceled out by another business-architect-related force that seemed rather subjective and not as perceptible to the consultants was not always easy but an important lesson. This also shows a key challenge of the consultancy setup, where not all forces might be observable directly and quantifiable at first. Therefore, we propose to establish a body of potential decision forces that can help preparing for more intangible decisions and being able to document them.

Another discussion regarding the rating of decision forces when applying the method was, whether a weighting factor should be introduced to express the strength of a force relative to other forces, e.g., for the decision on the middleware discussed above. Based on this idea, it was also discussed whether calculating a difference between positive and negative ratings should be used as a quantitative indicator of the combined decision force. However, these two extensions to the methodology were not used as the complexity would have increased and additional challenges introduced. For example, the weighting of forces seemed to be dependent on the individual decisions and could not be quantified easily. Defining adequate weights would have been a challenge on its own. Furthermore, simply calculating the difference between positive and negative forces showed to be not straightforward once *X* ratings were used. Thus, we decided to follow the original rating scale instead and leave a study of the ideas for future work.

While we decided to execute ratings on the basis of the two teams as a whole, one could imagine that team members individually express their observations from which the aggregate for the team can be deduced. However, based on our experience during the project, a discussion to establish a common understanding of the forces and the rating scale are indispensable and would also need to be done in this case. Tool support could be provided, e.g., by extending the previously mentioned and already interesting Enterprise Architect add-in from [6].

3.4 Forces in Industrial Consultancy Setting

In Table 1, we list the architectural decision forces that we observed in the project. A similar forces table discussed by Van Heesch et al. [10] additionally list the architecture-significant requirements as forces. For the sake of brevity, these forces are not listed here because they are specific to the particular project. In the presented project, the list of architecture-significant requirements included aspects such as the degree to which the solution shall decouple individual components, the required compatibility to the existing legacy architecture, the expected timing and availability requirements, as well as requirements regarding data consistency, persistence, and security. Clearly, these requirements played an important role for the architectural decisions as they provided the guardrails for all decisions taken. Within these guardrails, however, the following forces often defined the final decision.

F1-F3 categorize the forces related to the main stakeholders of the solution and the development process. F4 lists forces that are specific to the project being an architecture evolution project. F5 names general business-related forces that concern the competitiveness of the developed solution, while F6 focuses on forces specific to the project. Finally, F7 names forces observed in the selection of 3rd-party technologies, such as commercially available or open-source libraries and middlewares.

Forces F1, F2, and F6 are specific to the presented consultancy setup of the project. Some of the initially made decisions by the consultancy teams were later revoked because of strong unanticipated forces related to the business architects. We observed these forces are mainly due to very focused interest of the architects in particular parts of the proposed architecture, making previously made decisions obsolete, or differences in the judgment among the business architects. Here it is to note that these observed forces are very subjective and expressed from the viewpoint of the consultants. The actual forces leading to the observed forces are likely more complex and not easy to understand as outsiders to the actual business unit. The actual reason for a force might differ from its observation. It could be caused by, e.g., past experiences of the architects, a strategy that the consultants are not aware of, or the decision not to focus on a specific part of the architecture that is of interest to the architect expressing the force.

Another observation was that forces in category F5 and F6 should not be underestimated. They provide the framework of the overall project. Competitiveness is a key concern for every company that competes with others in the market for customers. Product management sets out strategic goals for future products that also research

Table 1: Decision forces observed in the considered consultancy setup, exclusive the (non-)functional requirements.

Code	Description	Concern(s)
F1	**Consultancy team related**	
F1.1	Experience with technology	Development time
F1.2	Imbalance of experience across distributed teams	Development time
F2	**Business architect related**	
F2.1	Architects' interest/priorities in a specific parts of solution	Stakeholder interest
F2.2	Different opinions of architects representing separate stakes	Stakeholder interest
F3	**Development team related**	
F3.1	Strategic knowledge development	Competitiveness
F4	**Architecture evolution related**	
F4.1	Alignment to legacy architecture and components	Legacy concerns / Compatibility
F4.2	Use of legacy development environment and setup	Legacy concerns / Compatibility
F5	**Business related**	
F5.1	Technology/product road map	Competitiveness
F5.2	Business vision, integration with other products, new architecture	Competitiveness
F5.3	Time to first proof of concept	Competitiveness
F6	**Project related**	
F6.1	Time budget for project	Development time
F6.2	Challenge development team and business with new technologies	Innovation
F7	**3^{rd}-party technology related**	
F7.1	Knowledge on performance	Performance
F7.2	Expected license costs	License costs
F7.3	Support model	Maintenance/Features
F7.4	Operational experience	Ease of operation

and development project need to be aligned with. Similarly, the project itself is typically time-bound and has a fixed development budget. Finishing a first proof-of-concept implementation to showcase the value of the investment taken by running the project, thus, is extremely important. By showcasing a proof of concept, it is important to convince stakeholders to further invest in a particular solution or abandon it in favor of an alternative path.

For the forces in category F7 it is to note that for using 3rd-party technologies in an industrial setting, the actual features of a technology might only play a minor role in the decision to use it. The license model, expected costs, as well as the support and active development of, e.g., a library or middleware usually are stronger forces to be considered. Furthermore, although the performance of a technology might be excellent it might not be considered as long as no comprehensive and reliable empirical proofs exist. Similarly, a specific technology might not be selected if the effort to find or educate corresponding R&D personnel is considered to be too high. Depending on the importance and available time, empirical studies might be done as part of a project or skipped in favor of less rated but well-studied alternatives.

While some of the forces discussed above are more generic, a number of new forces were identified that are specific to the consultancy setup. Other listed forces are specific to the project being an architecture evolution effort, where the existing software architecture has to be considered in all decisions. With that, Table 1 can serve as template for similar projects in an industrial setting to ease the phase of force elicitation and inspire identifying similar forces specific to a new setting.

3.5 Exemplary Design Decisions

To illustrate the effect of the different forces, we present two exemplary design decisions and their corresponding forces. In both cases, decisions have been made regarding technology alternatives to be used within the project. The first decision concerned the communication middleware used to implement the decoupling layer. The second one concerns the used caching solution within the decoupling layer (also see Figure 2). The considered technology alternatives were either commercially available implementations, open-source implementations, or in some cases even a custom implementation that would have to be implemented by the consultants as part of the project. To illustrate different expressions of forces and how they jointly led to a decision, the actual technologies are not as important. Therefore, we refrained from any details here. In the two examples presented in the following, the architecture-significant requirements are omitted. While for other decision they played a more crucial role, for these two particular decisions they did not discriminate among the technology options. Interestingly, only the forces introduced in the previous section led to a decision in these two cases. Table 2 shows an extract of the architectural decision forces viewpoint of the project, highlighting the strong forces that contributed to the final decision.

Decision A - Communication middleware: In case of the communication middleware, a decision in favor of Technology A was made. The decision was made in two steps after the consultancy teams evaluated the individual technology options and discussing their findings with the business architects. In a first step, Technology B was ruled out as a result of a number of forces. The technology was new to most of the members of the project. The missing experience regarding the technology and the lack of reliable information on its performance specific to the intended usage played an important role in the decision. In addition, the business architects realized that using Technology B would require a more fundamental study of how to model and map data of the legacy system to the concept. It became clear that the architects had general interest in the new technologies themselves. However, they decided to avoid spending

too much time on this particular part of the architecture. The architects' wish to focus on different parts of the architecture in the project, thus, prevented the further consideration of Technology B.

The two remaining technologies were further discussed between the consultancy teams. Interestingly, each of the two teams had some expertise and experience with one of the two technologies. This imbalance implied that a decision for either one of them would likely lead to one of the teams taking over the core development of this part of the architecture. Yet, due to the force being equally strong for both alternatives, it did not play a role for the final decision. Furthermore, we observed different forces in favor of Technology C due to it being part of the general technology road map and being aligned with the envisioned long-term future architecture of the business unit. However, as its application would have required more changes and general considerations for the legacy application during the adaptation, other forces dominated the decision. One part of the business unit was already using Technology A for a different use case and had detailed knowledge on its performance as well as operation. Therefore, the rather pragmatic decision for this "safe" option was taken. Due to the above line of reasoning for Technology C, the consultancy team experienced the final decision as being made rather by the architects' prejudice. In retrospective, also the overall forces observed support the decision. Thus, here, more strategic forces were given less weight than the forces in favor of a timely implementation within the project.

Decision B - Caching solution: The decision for a caching solution was partially influenced by a single requirement requesting that the cache access time should be in a similar order of magnitude than for the legacy architecture. The lack of upfront performance measurements for the intended setup made a discussion in favor of Technology A and B hard. Due to the limited possibilities how these technologies could be integrated into the legacy architecture and the flexibility of aligning Technology C with the existing architecture, a decision for Technology C was taken. Although the consultants were in favor of Technology A and B, the architects finally expressed their interest in Technology C, superseding other forces and canceling out the alternatives. With that, Decision B illustrates the importance of considering forces depending on stakeholder interests, although they might initially not be observable as prominently as other more objective and technical forces.

4 LESSONS LEARNED AND OPEN QUESTIONS

With this work, we wanted to share our lessons learned in using the decision forces viewpoint in a typical distributed project with a consultancy setup. We wanted to understand if it can help achieving a better transparency of architectural decisions and make their documentation more explicit. In the following, we summarize the key lessons.

Applicability and value. After studying the work by Van Heesch et al. [10], the application of the viewpoint was rather straightforward. As the notion of forces is very generic and also intuitive, it seemed easy to apply to the architectural decisions made as part of the project. The challenges we observed were caused mostly by the project setup that implied a number of specific forces but also required aligning on the observed forces and their ratings for individual decisions across the distributed consultancy teams. However, we quickly noticed the benefit of discussing factors influencing the decisions on the more abstract level of forces and reflecting on some of the previously made decisions.

The observed gap between decisions and rationales became obvious once we considered more and more forces. Especially, the non-technical forces were not sufficiently captured when only using the Y-model [12], where we seemed to over-simplify the rationales behind a decision. Other pragmatic documentation approaches, such as Architecture Haiku [4], are expected to have shown similar limitations in this case because its focus on functional requirements and quality attributes. One example for this is the observation that the consultancy team tried to make decisions based on objective measures wherever possible. In case this was not possible, studies of alternative technologies were performed or even performance tests executed to provide a solid basis for a decision. The business architects, however, steered some of the decisions in different directions. They were interested in new technologies and discussing new approaches but also used their experience and interest in particular parts of the solution to pragmatically steer decisions that, as a result, sometimes seemed not fully apparent to the consultants. Understanding these decisions and naming the likely forces behind them showed to be of great value for the project. This transparency also allows re-evaluating critical design decisions once the results of the consultancy project are to be used in an operational context in that some forces inevitably will differ.

Project-setup-specific forces. A main lesson learned in applying the forces viewpoint was that architectural decisions are heavily influenced by non-technical forces that, in the discussed project, were manifold as shown in Section 2. Most interestingly, a number of forces were observed that exist because of the distributed setup, such as differences in the experience across teams in different locations. But also forces that were observed as preferences or interest by individual stakeholders in particular parts of the solution are quite specific to the consultancy setup of the project. Such a force might not be observable in a project with well-aligned architects sharing a common understanding of long-term strategic goals. In the consultancy case, strategies might not be obvious to the external consultants. Also, experience with the particular software system might have to be slowly built up in the consultancy team first, before particular forces might become explainable. As a result, some of the decision forces we observed and reported might not even be captured completely and accurately due to this fuzziness. Related to this observation, some strong forces seemed to suddenly materialize for a number of decisions. Most likely, they were present all of the time but could not be recognized by the consultants from the beginning.

Force elicitation. The previous observation on forces highlights another important lesson: A number of non-technical forces, which one might call as the more subjective forces, were not perceived equally by the members of the consultancy teams. It took several discussions during the phase of forces elicitation to come up with the list of forces presented in Section 2. Some of the forces seemed to be perceived rather fuzzy by different team members that even were not always sure how to name them properly first. Finding the

Table 2: Forces observed for exemplary design decisions. Cells indicate the individual ratings of the two consultancy teams (*Team A/Team B*). Red and green cells indicate dominant forces that led to the made decision. The rating scale follows Van Heesch et al. [10] indicating the strength of a force in supporting/opposing a decision alternative: ++ (strong support), + (moderate support), blank (neutral), - (moderate opposition), - - (strong opposition), X (prevented by force), ? (currently unclear).

		Communication middleware			Caching solution		
		<decided>	<discarded>	<discarded>	<discarded>	<discarded>	<decided>
		Technology A	Technology B	Technology C	Technology A	Technology B	Technology C
F1.1	Experience with technology	++/+	-/-	++/+			
F1.2	Imbalance of experience across distributed teams	-/-		-/-			
F2.1	Architects' interest/priorities in a specific parts of solution	/++	X	+/-	X	X	++/++
F2.2	Different opinions of architects representing separate stakes						
F3.1	Strategic knowledge development		/-	+/+			
F4.1	Alignment to legacy architecture and components	+/++	/-	-/- -	-/	-/-	+/++
F4.2	Use of legacy development environment and setup	+/-	/-	/-			
F5.1	Technology/product road map	+/		+/++			
F5.2	Business vision, integration with other products, new architecture	+/		+/++			/+
F5.3	Time to first proof of concept	/+	/-	/-	+/+	+/	-/
F6.1	Time budget for project	/+		/+	/-	/-	
F6.2	Challenge development team and business with new technologies	/-	/+	/+			
F7.1	Knowledge on performance	++/+	/-	+/+	-/+	-/+	
F7.2	Expected license costs	++/+	-/-	-/-			
F7.3	Support model	/+	/++	/++	/+	/++	/-
F7.4	Operational experience	++/++	/-	/-			

right terminology to describe the forces and aligning on a common understanding required some time, which should be considered in the planning. Forces were added, rephrased, grouped, joint, or even deleted before the team decided to go ahead and apply it to the different decisions. The fact that the teams could only collaborate remotely also showed to be challenging. Here, a collaboration tool specific to the force elicitation task would be desirable.

In addition to the process of identifying forces based on the discussion within the consultancy team, a way to directly identify forces early enough, e.g., together with the elicitation of technical requirements would be desirable. Here, a standard catalog of questions could help to specifically ask the business architects for forces they expect and that they think might be hard to anticipate by the consultants. Such a catalog could include questions for expected complexity of different parts of the solution, problems encountered in previous projects, particular interests in technologies, and the expected impact of different project results. Although, experienced consultants might anyway ask these questions, having a checklist of such questions could greatly help in making even less experiences consultants mindful of other potential forces and reducing the risk of overlooked forces. Also, the answers of the architects can provide an initial indication that could be used to assign indication weights to the forces according their expected relevance once they become observable in the course of the project. This could help focusing the attention on more critical forces and avoiding wasting time on, e.g., time-consuming measurement studies to establish a weak force that is likely to be canceled out by a more dominant force.

Potentially critical forces could be monitored and re-evaluated over time to timely stop activities that will likely become obsolete.

Overall, the temporal dynamics in observable forces played an important role in our process and should be considered. Forces might be hidden at a given state, increase or diminish over time or with insights on decision alternatives, or can even become preventive and cancel out other forces. Table 3 illustrates a case in which

Table 3: Exemplary heat map of potential dynamics in strength of forces for an individual decision over time.

	Project time					
Force	M1	M2	M3	M4	M5	M6
A	Hidden		Neutral		Prevented	
B	Neutral		Discriminating		Prevented	
C	Hidden				Exclusive	

three forces are observed with different strengths over the course of the project. While Force B was visible the whole time, in the third month, it becomes discriminating, meaning that it causes a decision towards one of the alternatives at that time. Force A is observed starting from the third month but is neutral and does not have an influence on this particular decision. Force A would be shown as a blank cell in the decision forces viewpoint (cf. Table 2). Force C was not observable in the first four month. It is first observed in the fifth month, but from there on prevents other forces and becomes the only relevant force for a decision. Based on our observation, the goal should be to identify as many potential forces as possible early on, monitor existing forces, be sensitive to upcoming forces, and willingly foster the rising of preventive forces that are inevitable as early as possible to avoid unnecessary work on other forces.

Force rating and agreement. The rating of forces and agreeing on the rating across the consultancy teams showed to be another challenge of the viewpoint when applied in a distributed setup. The problem was not that the consultants did not agree at the end but that discussing the ratings and aligning the individual subjective observation of some forces showed to be too time-consuming already in a team of five consultants. Splitting up the discussion into per-location ratings and combining these ratings helped to come to a rating that showed clear tendencies for the strength of individual forces. We decided to keep the individual team ratings and work with the overall tendencies. However, as also visible in Table 2, in some cases the ratings differed or were observed even in opposite directions. Thus, for the rating process, a live voting tool could be of great value for remote teams. This way, individual team members could rate the forces according their observations before a meeting and the team could directly enter focused discussions to align ratings that diverge substantially. Team members could also start to take others' ratings into account and, thereby, better calibrate to the team's overall rating scale. This could help in executing a more focused rating process for a large number of forces or even take on-the-fly decisions based on a rating, which was observed as a bottleneck for the use of the viewpoint.

Cost/benefit of viewpoint. Overall, one could pose the question, whether the effort of using the viewpoint is justified by the benefit for a project. We agree that this is a valid question and that using the viewpoint becomes more challenging for more complex project setups. However, due to the great benefit we observed in gain of transparency in decisions for the consultancy teams, we would say that the benefits outweigh the costs if the viewpoint is applied in the right manner. We decided to only use it for critical design decisions, similar to the architectural review method by Van Heesch et al. [11], and we decided to stop the elicitation and rating once we observed a strong enough tendency of agreement.

To increase the benefit of using the viewpoint, it would be interesting to further study how the viewpoint could be made more accessible also to the business architects. There is a high chance that experienced architects might see the process as an academic exercise, since most of the forces might be obvious to the architects themselves. Here, it is valuable to increase the awareness for common fallacies as the ones described by Kruchten in [5] and considering them as forces. This could help convincing even experienced architects of the value of the viewpoint.

5 RELATED WORK

Within this section, we summarize the most relevant related work for our work.

Manteuffel et al. [6] realized the decision forces viewpoint as part of an add-in for the tool Enterprise Architect by Sparx System. The add-in is used to document architectural decisions based on different architectural viewpoints. The authors evaluated the add-in in an exploratory case study with industrial software architects. The evaluation showed that the forces viewpoint was considered useful with some requests for further features to ease working with the viewpoint. In contrast to the authors' work, we are not focusing on tool support but rather evaluated the use of the viewpoint as such. Furthermore, we discuss its challenges in a typical, not yet considered, consultancy setup with geographically distributed teams. For maximum flexibility and to reduce the number of required tools in the team, we decided to use a generic Excel representation to document the forces instead of the Enterprise Architect add-in.

In Van Heesch et al. [11], a later work by some of the authors of the original forces paper, the use of decision forces in the context of architecture reviews is proposed and discussed. The authors describe a lightweight method to execute focused and time-bound architectural reviews and share experiences for its use in industry use cases. For this, selected design decisions are individually evaluated by internal or external reviewers in a half-day face-to-face meeting. Following a management and architect presentation, relevant design decisions and their forces are collected. Subsequently, the participants of the review prioritize the decisions according their criticality. The top seven to ten decisions are then used during the decision evaluation, where the decisions and the completeness of relevant forces are challenged. The outcome of this evaluation is summarized in a report afterwards that architects can use as additional decision documentation. The approach and reported use case evaluations show the strength of the force concept with a focus on architectural evaluations. While not following the steps of the

proposed method, our use of forces to question architectural decisions during the design process shows similarities to the method. However, the method does not consider the challenges related to the consultancy setup presented in this work.

Harrison et al. [1] discuss the problem of architectural decision making in large open-source software (OSS) projects, where similar to a consultancy setup, a team of geographically distributed architects/developers need to collaborate and agree on taken decisions. Interestingly, they point out that contributors to an OSS project might have different motivations and agendas if they work for different companies. This shares some commonalities with a consultancy setup, where architects work with experts external to their team, potentially having different opinions and motivations in participating in the project. Based on an analysis of architecture documentation from forty-four OSS projects, the authors point out that decision making, communication, and transparent documentation are the major challenges in such projects. They summarize that the decision making in these projects followed three potential patterns: decisions were taken by a single person in the role of a lead architect, an architecture review board existed that reviewed and controlled changes to the architecture, or a less organized approach was taken were voting on decisions are likely to have played a role. Overall, the authors note that no particular decision making practice was observed that takes into consideration the distributed nature of the development team, which is a focus of this paper.

Overall, none of the above works discuss a consultancy setup, the influence of remote teams collaborating, or the specific decision forces we experienced in this setup.

6 CONCLUSION

We applied the *decision forces viewpoint* in a distributed industrial project with a consultancy setting. Overall, this helped to improve the transparency of critical design decisions and contributing to clear documentation of the decision rationales. With that, made decisions can be re-evaluated once decision forces change or related architecture decisions need to be made. Based on the lessons we learned in this project, we encourage the use of the viewpoint in industrial settings. Most of the observed challenges could be addressed by an improved tool support for distributed teams and by trying to identify strong forces early in the design process. For this, continuing our effort in providing templates of likely forces and the development of a question catalog for the the forces elicitation would be of great value.

ACKNOWLEDGMENTS

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the ITEA project REVaMP^2 funded by the German Federal Ministry of Education and Research1 (01|S16042B).

REFERENCES

[1] N. B. Harrison, E. Gubler, and D. Skinner. 2016. Architectural Decision-making in Open-Source Systems – Preliminary Observations. In *IEEE International Workshop on Decision Making in Software ARCHitecture (MARCH)*.

[2] ISO. 2011. ISO/IEC/IEEE 42010, Systems and Software Engineering: Architecture Description. (2011).

[3] J. Jia, P. Zhang, and L. F. Capretz. 2016. Environmental Factors Influencing Individual Decision-making Behavior in Software Projects: a Systematic Literature Review. In *ACM International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*.

[4] M. Keeling. 2015. Architecture Haiku: A Case Study in Lean Documentation. *IEEE Software* 32, 3 (may 2015), 35–39.

[5] P. Kruchten. 2013. Games Software Architects Play. https://resources.sei.cmu.edu/asset_files/Presentation/2013_017_001_47712.pdf

[6] C. Manteuffel, D. Tofan, H. Koziolek, T. Goldschmidt, and P. Avgeriou. 2014. Industrial Implementation of a Documentation Framework for Architectural Decisions. In *IEEE/IFIP Conference on Software Architecture (WICSA)*.

[7] A. Tang, M. Razavian, B. Paech, and T.M. Hesse. 2017. Human Aspects in Software Architecture Decision Making: a Literature Review. In *IEEE International Conference on Software Architecture (ICSA)*.

[8] A. Tang and H. Van Vliet. 2009. Modeling Constraints Improves Software Architecture Design Reasoning. In *IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA)*.

[9] U. van Heesch, P. Avgeriou, and R. Hilliard. 2012. A Documentation Framework for Architecture Decisions. *Journal of Systems and Software* 85, 4 (apr 2012), 795–820.

[10] U. van Heesch, P. Avgeriou, and R. Hilliard. 2012. Forces on Architecture Decisions - A Viewpoint. In *IEEE/IFIP European Conference on Software Architecture (ECSA)*.

[11] U. van Heesch, V.-P. Eloranta, P. Avgeriou, K. Koskimies, and N. Harrison. 2014. Decision-Centric Architecture Reviews. *IEEE Software* 31, 1 (jan 2014), 69–76.

[12] O. Zimmermann. 2012. Making Architectural Knowledge Sustainable–The Y-Approach. In *Presentation at SATURN Conference*. https://resources.sei.cmu.edu/asset_files/Presentation/2012_017_001_31349.pdf