Improving the Consistency and Usefulness of Architecture Descriptions: Guidelines for Architects

Rebekka Wohlrab*†, Ulf Eliasson*‡, Patrizio Pelliccione*§, Rogardt Heldal*¶
*Chalmers | University of Gothenburg, Gothenburg, Sweden
†Systemite AB, Gothenburg, Sweden
‡Vinnter AB, Gothenburg, Sweden
§University of L'Aquila, L'Aquila, Italy
¶Western Norway University of Applied Sciences, Bergen, Norway
Email: wohlrab@chalmers.se, ulf.eliasson@vinnter.se, patrizio.pelliccione@gu.se, rogardt.heldal@hvl.no

***Abstract*— The need to support software architecture evolution has been well recognized, even more since the rise of agile methods. However, assuring the conformance between architecture descriptions and the implementation remains challenging. Inconsistencies emanate among multiple architecture descriptions, and between architecture descriptions and code. As a consequence, architecture descriptions are not always trusted and used to the extent that their authors wish for. In this paper, we present two surveys with 93 and 72 participants to examine architectural inconsistencies, with a focus on how they evolve over time and can be mitigated using practical guidelines. We identified the importance of capturing emerging elements to keep the architecture description consistent with the implementation, and consider the current-state and future-state architecture separately. Consequences of inconsistencies typically arise at later stages, especially if an architecture description concerns multiple teams. Our guidelines suggest to limit the upfront architecture to stable decisions, while paying attention to concerns that matter across team borders. In the ideal case, companies should aim to integrate architects into the teams to capture emerging aspects with time.**

***Keywords*-architectural inconsistencies; architectural conformance; agile architecture; boundary objects; survey; questionnaire; empirical software engineering**

I. Introduction

It is well recognized that software architecture is crucial when developing complex software [1]. A poorly conceived and defined architecture might cause serious implementation flaws, delays on the project, or product delivery extra cost. Architecture knowledge is used in different phases of the architecture lifecycle by a variety of stakeholders [2], to conceive and evaluate the architecture, as well as to implement and maintain it. Architecture knowledge is commonly manifested in architecture descriptions.

Previous research has identified the need to design an architecture also in agile environments, and reconsider suitable approaches to document it in fast-changing contexts [3], [4], [5]. A critical issue is the balance of upfront architecture (that is planned before the start of development) and emerging architecture (that appears as decisions are taken in the course of the development). This balance is needed to take key decisions early on and avoid architectural rework, but to leave room to adapt and evolve the architecture over time [6].

In evolving software engineering contexts, a prevalent issue is the assurance of conformance between architecture and code [1], [7], [8]. Empirical studies have found that the artifacts produced by architects are not used by the intended consumers or at least not to the wished extent [9], [10]. Inconsistencies between architecture descriptions, as well as between architecture descriptions and the implementations can be problematic. A study on the loss of architecture knowledge during system evolution found that more than 70% of the non-conformances between architecture description and source code are due to flaws in the documentation [11]. An analysis of open research issues on inconsistency management arrived at the conclusion that more research is needed on how stakeholders and the general development process influence inconsistencies [12]. A recent study has found that while practitioners are reluctant towards fixing architectural inconsistencies, they see potential in mechanisms to increasing architectural awareness [13]. However, more case studies are needed in the area [13].

This paper contributes towards closing this gap by scrutinizing architectural inconsistencies in evolving software engineering contexts and providing practical guidance to make architecture descriptions more useful and consistent. We conducted an initial survey with 93 respondents to investigate inconsistencies between multiple architecture descriptions, and between architecture descriptions and the implementation. We derived practical guidelines to augment the usefulness and consistency of architecture descriptions. In a second survey with 72 participants, we validated the previous findings and evaluated the guidelines' usefulness.

We focused on the following research questions:

- RQ1: What are the types, reasons, and consequences of architectural inconsistencies?
- RQ2: How do architecture descriptions and their relation to the implementation change over time?
- RQ3: What are guidelines to support practitioners with the management of architecture descriptions?

Our findings suggest that architecture descriptions can be useful both to reason about the future architecture and about the current state. Inconsistencies related to wording and language are perceived as less critical than other types of inconsistencies (e.g., rules and constraints). They occur due to several reasons, e.g., lack of time, lack of knowledge, and communication issues, especially if an architecture description concerns multiple teams. Consequences of inconsistencies typically arise at later stages (e.g., deployment, runtime, or maintenance). The role of consistency, the main users, and the role of architecture changes from the initial creation to the design and development. Over time, developers become more involved and consistency becomes more relevant during the development and design. Our participants see a value in describing both the current-state and the future-state architecture. They aim to refine the architecture during the implementation, but struggle with finding a balance between upfront and emerging architecture. To mitigate these issues, we propose to clarify the purpose of a description and whether the description should refer to the current- or future-state architecture. The upfront architecture should be minimized, but especially elements concerning more than one cross-functional team should be included in architecture descriptions, as these descriptions may act as "boundary objects" between teams. Architects should be integrated into teams to capture emerging aspects, but also have communities of practice to reason about change.

Paper outline: The paper is structured as follows. Section II describes our study's research method. Section III presents an overview of the participants of the surveys. Sections IV (RQ1) and V (RQ2) present the findings of the study. Section VI presents the developed guidelines (RQ3). Section VII presents related work, a comparison with our findings, and relations to the guidelines we propose. We conclude the paper in Section VIII with directions for future research.

II. Research Method

To fulfill our research goals, we conducted two surveys, following Kitchenham and Pfleeger's guidelines for survey research [14]. Surveys are appropriate when the aim is to provide statistical descriptions of an issue by asking questions and analyzing mainly numerical answers [15].

We decided to include several iterations in the research method: Validation focus groups to validate our study design, an initial survey to understand the consistency and usefulness of architecture descriptions, and a second survey focusing on concrete guidelines and findings derived from the first step. Figure 1 depicts the steps of our research method that we describe in the following.

A. Literature Review

In order to get a good understanding of related work and see what research questions are of importance, we searched the literature for studies in the area. We focused on architectural inconsistencies and architecture descriptions, as well as general studies on architecture acknowledgment and agile architecture.

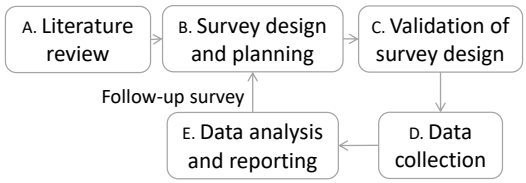

Figure 1. Overview of our research method

B. Survey Design and Planning

We defined the selection criteria of our surveys as follows:

1) industrial practitioners knowledgeable in the areas of software or systems architecture;
2) involved in the architectural process at their company (creators, consumers, or other stakeholders of architecture descriptions);
3) having a role in which they are in contact with architecture descriptions (e.g., developer, tester, architect).

We decided to use convenience sampling due to the limited availability of the target population. We used existing contacts to identify participants fulfilling these criteria, e.g., based on research collaborations or by contacting participating companies in the Software Center initiative[1]. We also posted an announcement in the mailing list of the ISO/IEC/IEEE 42010 architecture standard [16], and in the "*Software Architects and Enterprise Architects*" and the "*97 Things Every Software Architect Should Know*" LinkedIn groups. Moreover, we asked our participants to extend our sample using snowball sampling and sent reminders to participants to increase the response rate of our survey [14].

When composing the questionnaire, we linked included questions to our main research questions for better traceability. The questionnaires can be found online[2]. We included demographic questions, as well as open- and closed-ended questions. Open-ended questions were used to not impose restrictions on the respondents by predefined answers. Closed-ended questions included numerical values, response categories, yes/no answers, and Likert scale questions [17].

C. Validation of the Survey Design

We conducted initial test runs of the first survey internally and with three practitioners from different companies.

Moreover, for both surveys, we hosted validation focus groups. For the first survey, we had four sessions with staff from two automotive companies and one telecommunication company. For the second survey, we conducted four validation focus groups together with two system architects

[1]http://www.software-center.se/

[2]https://tinyurl.com/ybk9gycm and https://tinyurl.com/y849zamk

working in an automotive company for more than 20 years, and an architect from a telecommunications company with 7 years of experience. In the validation focus groups, we walked through the questions in a PowerPoint presentation, and asked for input, e.g., if the terminology was understandable. We recorded all given input and used it to inform the developed guidelines that we validated in the second survey.

For both surveys, we updated the questions and conducted final test runs together with the focus group participants.

D. Data Collection

We used online surveys for the data collection, based on the platform SoSci Survey[3]. The length of the first survey was approximately 20 minutes, and about 10 minutes for the second survey. We filtered out the responses of participants who only clicked through the survey without answering at least 20%. In total, we had 93 complete answers for the first survey. For the second survey, we got 72 replies in total.

E. Data Analysis and Reporting

For closed-ended questions, we analyzed the number of given responses to draw conclusions. For answers given in text, we coded the text of the responses by categorizing replies or parts of replies.

With the coding and statistical summary of the questions, we collected key findings in two documents: One with the results of each question (vertical analysis), and one across questions (horizontal analysis). For instance, we checked whether architects would have different response pattern than respondents with other roles. Findings and hypothesis were revised, tested, and discussed in a number of whiteboard workshops among the authors.

Based on the first survey, we derived guidelines and questions for the second survey. We took identified challenges and insights as a basis and derived guidelines as a way to improve the situation. Together with the participants of the validation focus groups, we discussed the guidelines. We also systematically noted down our insights and discussions, created appropriate graphs and diagrams to visualize the data, and report on our findings in this paper.

F. Research Validity And Limitations

We discuss four types of survey validity [18].

Internal validity is concerned with the relationship between a treatment and its results and whether any unknown factors influenced the outcome of the study. To improve the instrument used in the study, we conducted validation focus groups before the surveys, also discussing factors not captured in the surveys. We asked our participants to respond on a voluntary basis and ensured confidentiality to encourage respondents to answer in a truthful way.

Conclusion validity relates to the certainty that correct conclusions can be drawn about the relation between the measures and the observed outcome. To mitigate threats to conclusion validity, we included researchers and practitioners with different backgrounds in the design of the study. Moreover, we constructed the second survey in a way that we asked general questions in the beginning before presenting our guidelines and potentially biasing the participants. In this paper, we are transparent about our methodology. We systematically analyzed the data by coding free text answers and analyzing questions both individually and in relation to other questions. Moreover, we validated the conclusions with industrial and academic participants.

Construct validity is about how a theory behind an investigation relates to the observations made. In the area of architecture descriptions and inconsistencies, there is no established theory. There exists a potential threat that respondents answer what they think we expect them to answer and/or want to present themselves in a favorable light (evaluation apprehension). Creating anonymous surveys and assuring that we treat the data confidentially helped to mitigate this bias.

External validity is about the limitations of this study to generalize conclusions to other companies and cases. We aimed to find a representative sample based on different sources, both within our networks and via LinkedIn groups and mailing lists. In the first survey, the respondents came from 27 companies. Still, some companies were more strongly represented than others. Other potential biases are a high proportion of architects and individuals having a positive attitude towards architecture topics. We highlight when the architects' replies stood out compared to the responses from other roles. 51% of the respondents in the first survey were architects, and 61% in the second survey. However, 60% of the architects in the second survey also had another role. For this reason, they have a broader perspective on the topic. Our experienced respondents also answered based on experience from previous roles.

III. Overview of Survey Participants

This section presents the survey participants' origin and context for the initial and the second survey.

A large majority (76% in the first, 64% in the second survey) of the participants work in companies with more than 2000 employees. The dominating domain was automotive with 41% in the first and 33% in the second survey. Telecommunication was selected by 26% in the first survey, and 9.3% in the second survey. The rest of the companies covered domains such as banking, business systems, health care, defense, and aerospace.

In the first survey, we also for the respondents' companies. In total, 27 companies in 12 countries were named.

Our participants cover different roles, and we let the participants choose more than one role. The most popular role among our participants was "architect" (51% in the first, 61% in the second survey). Software designer and

[3]https://www.soscisurvey.de/

system engineer were selected by 26% in the first and 16% in the second survey. Other roles were function owner, Scrum master, CTO, system/product owner, project manager, implementer, and tester.

A majority of our participants were experienced, with 27% having more than 21 years of experience in the first, and 39% in the second survey. In the first survey, 43% had 11-20 years of experience, and 38% in the second survey.

In the second survey, we asked the participants what development methods they used. 87.5% of the participants indicated that they use agile methods, and 9% indicated in the comments that they are in the transition to agile methods.

IV. Architectural Inconsistencies in Practice

This section gives answers to RQ1: *What are the types, reasons, and consequences of architectural inconsistencies?* 85% of our respondents of the first survey reported that they were aware of inconsistencies between the architecture description and the implementation, and 82% were aware of inconsistencies between multiple architecture descriptions.

A. Types of Inconsistencies

The survey participants of the first survey were asked to give examples of inconsistencies in plain text. Based on the coded data, we identified four main types of inconsistencies:

1) Interface specifications and implementation: Interfaces are implemented differently from their specification. A developer mentioned that inconsistencies often emerge due to *"interfaces between components developed in separate organizations."* The organizational and collaborative consequences of developing a product in a distributed manner strongly influence the architecture and the likelihood to have inconsistencies.
2) Rules and constraints: These conditions are set by the architecture, but broken in the implementation.
3) Patterns and guidelines: These should be followed, but are less strictly enforced than rules and constraints.
4) Wording and language: Inconsistencies in wording and language are quite common, but the negative impact of this type of inconsistencies is considered rather low.

In the initial survey, the first three types of inconsistencies were perceived to have a highly negative impact. Inconsistencies on wording and language were not seen as very critical. An architect working at an automotive company stressed that as long as traceability is supported, inconsistencies in wording are not that critical. For instance, there are naming conventions for components at the architectural level which are not followed in the implementation.

B. Reasons for Inconsistencies

We asked our respondents to state why they think inconsistencies occur. 54% of the respondents of the first survey stated that a reason for inconsistencies is a lack of time, and 26% stated that inconsistencies exist due to a lack of resources. These causes are also related to the priorities of investing in (the quality of) architecture descriptions and consuming the created documentation. The reason "lack of interest" in architecture was reported by 30% of the participants of the first survey. An architect stated that he did "*not think developers consider reading architecture descriptions at all, instead they invent own local patterns which are not documented at all.*"

54% of the respondents of the first survey stated that inconsistencies are caused by a lack of knowledge—mainly knowledge of underlying rules and principles of the architecture. Related concerns are issues on assumptions, chosen by 40% of the respondents of the first survey, lack of communication (46%), and lack of a common language (40%). In fact, we found that lack of communication across team boundaries is especially critical: 75% of the respondents of the second survey agreed or strongly agreed with the statement that "*inconsistencies are more likely to arise when a change concerns multiple teams.*"

C. Consequences of Inconsistencies

In the first survey, we got 48 free-text examples of consequences and categorized them into development issues, run-time issues, deployment issues, and maintenance issues. Development issues mostly relate to workarounds and rework motivated by the need to remove inconsistencies. Run-time issues are deadlocks, illegal behavior in the interaction between components, or other defects. Deployment issues relate to complications for staff deploying the system to understand how the product works and how to deploy it. Maintenance issues are connected to the difficulty of finding bugs in the system, or unforeseen dependencies between components that result in the need for architecture rework.

V. Time Perspectives of Architecture Descriptions

This section answers RQ2: *How do architecture descriptions and their relation to the implementation change over time?*

We found that our participants had different perceptions of the purpose and time perspectives of architecture descriptions (i.e., whether it should describe the as-is or to-be architecture) and how these aspects change over time.

A. Development of Architecture Descriptions over Time

We analyzed how the audience of architecture descriptions and the role of consistency change over time.

81% of the respondents of our second survey stated that the role and purpose of architecture change over time. 57% stated that the importance of consistency increases over time and 36% reported that it decreases over time.

There are differences between the creation phase of the architecture and the phase in which the implementation of the software or system starts. 69% of the respondents

Table I
AUDIENCE, ROLE OF CONSISTENCY AND OF EMERGING ELEMENTS OVER TIME (NUMBERS IN PERCENT, SECOND SURVEY)

	During initial creation of the architecture	During development and design
Architects are the main users	56%	28%
Developers are the main users	22%	54%
Inconsistencies are not a big issue	38%	14%
Need stability/correctness/ consistency among arch. descr.'s	61%	86%
Emerging elements should be planned for and captured	86%	90%

stated that they observed that "*there is an initial description of the architecture which is refined when the product is designed and developed*." Table I shows how the audience, role of consistency, and importance of emerging elements were perceived to change over time, based on the second survey. We found that architecture descriptions are slightly more commonly used by architects in the initial stage: 56% of the survey respondents saw architects as the main users of architecture descriptions in the beginning, while 22% considered developers the main users. 86% of the respondents stated that in the beginning, one needs to plan for emerging elements. An architect working at an automotive company stated that with time, the goal is to further increase the collaboration between architects, developers, and other architecture stakeholders. The intention is to align architecture and implementation. 54% of the participants of the second survey reported that during the design and development, the architecture is mainly used by developers, and 28% saw architects as main stakeholders. While both roles seem to be involved from the initial creation to the development and design, the trend is that developers become more involved over time. In the comments, several respondents mentioned that other stakeholder groups should not be forgotten.

38% of the respondents of the second survey stated that during the initial creation of the architecture, inconsistencies are not a big issue, while 14% of the respondents stated that inconsistencies are not a big issue during the development and design. In the later phases, consistency appears to play a more important role. 86% of the respondents of the second survey stated that during the development and design, an architecture description should be consistent with other architecture descriptions, and 88% that the architecture description should be consistent with the implementation.

Based on the presented information, we understood that roles and the importance of consistency change over time. Developers become more central users during the development and design and the importance of consistency increases after the initial creation of the architecture.

Table II
WHAT POINT IN TIME DOES/SHOULD THE ARCHITECTURE DESCRIBE? (NUMBERS IN PERCENT, FIRST SURVEY)

	What does the architecture description *currently* describe?	What *should* the architecture description describe?
To be developed	47%	63%
Current state	46%	46%
Other	13%	11%

B. Time Perspective of Architecture Descriptions

To understand how the relation of architecture descriptions and the implementation changes over time, we were interested in understanding what state of the system the architecture description currently describes and should describe. Table II shows the answers of the respondents of the first survey related to these questions. There is a balance between the future and the current perspectives: 47% of the respondents indicated that it is concerned with the system to be developed, while the current state was picked by 46%. 63% of the participants reported that an architecture description *should* describe what should be developed in the future, while 46% selected that it should be the current state.

On one hand, respondents see that the architecture should reflect the system at the current point in time for various purposes. In the first survey, 69% named that an architecture description should communicate architecture decisions and also 69% stated that it should define components and connectors. On the other hand, practitioners considered that architecture descriptions should be a blueprint for future development (48%, first survey) or to analyze the impact of changes (43%, first survey).

In the second survey, 88% of the respondents stated that a description of the future architecture is needed, and 79% stated that they need a description of the current architecture during the development and design.

Several participants of our study also commented that an architecture description is used for long-term knowledge management, e.g., of a series of products. A requirements engineer, architect, and system engineer from a telecommunications company stressed that an architecture description should "*capture the thoughts, ideas, patterns, and assumptions that went into the architecture, so that future development can follow the same ideas and patterns*."

To conclude, we saw that architecture descriptions are and can be used to describe the current-state system or software, as well as the future-state system or software. For both time perspectives, there are purposes and respondents stating that the perspectives are valuable. We used this understanding during the development of the guidelines (Section VI).

C. Interplay Between Architecture and Implementation

We aimed to understand whether the architecture description is designed first or whether it follows the design and

implementation. In the first survey, 60% of the participants stated that the architecture is designed upfront and 49% indicated that the architecture emerges from the implementation.

We noticed a difference between roles. In the first survey, 81% of the architects indicated that the design and implementation follow the architecture rules and principles. Of the respondents with no architecture role, 54% agreed with the statement that the design and implementation follow the architecture rules and principles.

In the second survey, 90% stated that during the design and development, emerging elements of the architecture appear and should be captured in the architecture description. 79% of the participants of the second survey agreed or strongly agreed with the statement that "*an initial architecture created by the architects should be the starting point for the implementation*."

To conclude, our participants saw that the architecture should be refined based on aspects that appear during the design and development. An initial architecture description is typically used as a starting point for implementation. In the second survey, 76% agreed or strongly agreed with the statement that they "*struggle with finding a balance between upfront architecture and architecture emerging from the development*."

VI. Guidelines for Practitioners

This section answers RQ3: *What are guidelines to support practitioners with the management of architecture descriptions?* An overview of our guidelines and our participants' ranking of their value is shown in Figure 2. In the following, we present the rationales behind the guidelines and insights from the second survey.

First guideline: Our findings indicate that architectural inconsistencies are indeed commonly observed by practitioners. Inconsistencies with a less severe negative impact are typically related to wording and language. A majority of respondents reported a lack of time and interest as reasons for inconsistencies. A lack of time and resources indicates that the task of counteracting inconsistencies is not seen as a top priority issue. To improve the quality and usefulness, our participants suggested to include only "*important*" aspects and tailor the descriptions more towards the intended customers. 60% of the respondents of the second survey agreed or strongly agreed that "*it is not always apparent who the consumers of an architecture description are.*" When aiming to create an architecture description that is used by practitioners, it needs to be clearly specified who the consumers are and what the purpose is [9]. We capture this aspect in the first guideline:

(G1) *To define the content of an architecture description, clearly state the purpose and the intended audience. This information can help you to define the level of abstraction and what elements should be included and excluded in the description.*

84% of the respondents of the second survey considered this guideline very or extremely valuable. A respondent considering the guideline "*not at all valuable*" stated: "*Whatever the intended audience was when the document was written is probably different from the real audience later on.*" We acknowledge the need to reassess the audience later and consider it not only in the beginning, but in frequent intervals. An architect working in the health care sector agreed with the guideline and stated that "*this is definitely true, but it is a challenge.*" One respondent stressed that the guideline is also included the ISO standard 42010 [16].

Second guideline: Some participants stated that an architecture description should reflect the current state of the system, but also future concerns are of interest (e.g., when understanding the impact of changes). In fact, we found quite an equilibrium between the two time perspectives—both with respect to what architecture descriptions currently describe and what they should describe. Both are relevant for practical purposes and have their justifications. However, in order to mitigate inconsistencies, these two concerns should be clearly separated. This brings us to our second guideline:

(G2) *There should be a clear distinction between the architecture of the system as it is now and as it is planned for the future. The two time perspectives should not be mixed.*

In our second survey, 72% of our respondents ranked this guideline as extremely or very valuable. We received comments stating that both dimensions can be part of the same document, but still separated. Other respondents stated that they partly mix the perspectives and describe the current architecture in more detail than the future architecture. Two respondents stated that a timeline including the transition from the current to the future state can be beneficial, since "*the transition architecture is sometimes critical.*"

Third guideline: The views of whether an architecture description should be an upfront specification or emerge from the implementation differed a lot between respondents. If there is no upfront architecture, ad-hoc decisions are likely to be taken due to a lack of knowledge and communication, which leads to undesirable rework in later phases. Our findings suggest that if there exists too much upfront documentation, developers potentially ignore it and do not regard it as useful. 90% of the respondents of the second survey stated that emerging elements of the architecture should be captured during development and design.

We found that inconsistencies between the interface specifications and the implementation were the most common type of inconsistencies, and commonly arise when interfaces connect components developed by different organizational units. For architecture elements used across teams, it is especially crucial to have a reliable, useful architecture description. 75% of the respondents of the second survey stated that inconsistencies are more likely to arise when a change concerns multiple teams. This leads us to the

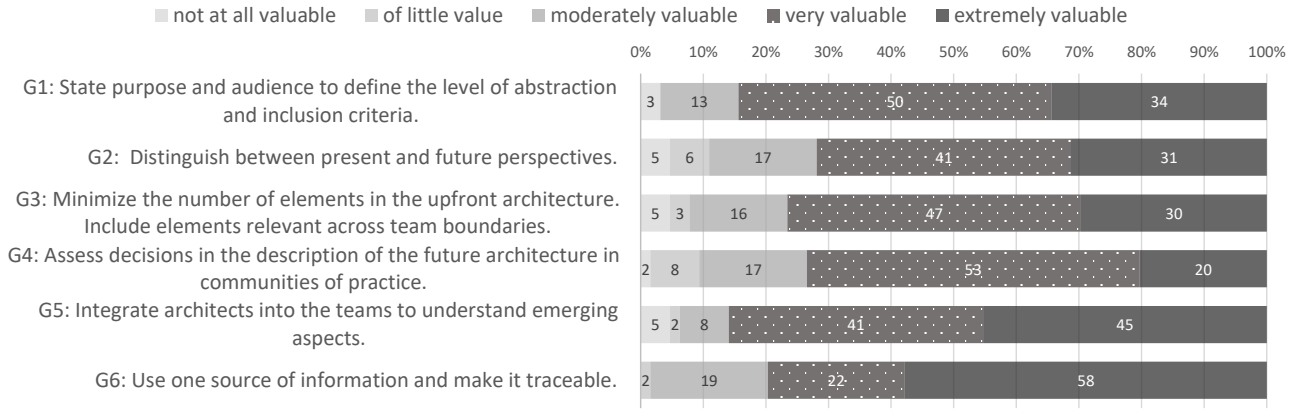

Figure 2. Ranking of our guidelines' value in percent

definition of the third guideline:

(G3) *Minimize the number of elements in the upfront architecture description as much as possible and ensure it is in line with its purpose. Make sure that you include key elements that concern more than one cross-functional team.*

77% of the participants of the second survey ranked this guideline very or extremely valuable. Some critical comments related to the term '*minimize*.' Three respondents stressed that a "*minimum viable architecture description*" with "*architecturally significant elements*" is needed. An architect from an automotive company stated that it might not be relevant to arrive at a low number of elements in the architecture description, but stressed the importance of including elements of "*high impact on multiple teams*." Moreover, this guideline was considered "*essential to achieve agile architecture*" and "*very good guidance*."

Fourth guideline: In the initial survey, 80% of the participants indicated that documenting assumptions could help to improve architecture descriptions. We found that many of the consequences of inconsistencies are observed in later phases of the system's lifecycle, in which architectural rework requires more effort. Therefore, decisions included in the architecture description should be evaluated carefully. The participants of our validation focus groups suggested experiments and prototyping for this purpose.

Communities of practice can be suitable forums to analyze and discuss architecture decisions in groups of experts. An architect from a telecommunications company underlined that discussions in these communities should lead to actionable decisions.

This leads us to the definition of the fourth guideline.

(G4) *If you create a description of the future architecture, carefully assess decisions before setting them into stone to reduce assumptions that can become inconsistencies. Establish a community of practice for architects to reason about change.*

This guideline was ranked very or extremely valuable by 73% of the participants. Several respondents argued that decisions should not be "*set into stone*." Instead, an architect from a telecommunications company argues that decisions can be on multiple confidence levels and should be more or less binding. Another respondent stated that decisions that are expensive to change "*are worth thinking carefully about early on*." The term '*community of practice*' seems to have certain connotations in some of the respondents' companies. Two respondents stated that they prefer the terms '*forum*' or '*community of impacted users*.'

Fifth guideline: Architects are typical users of architecture descriptions. However, to counteract the antipattern of the ivory tower [19] also other roles should get involved. 54% of the respondents of the second survey stated that developers should be the main users of architecture descriptions during the design and development. It is important to iteratively improve architecture descriptions and capture emerging aspects, as also confirmed by our second survey. The top mentioned suggestion in the first survey was to leverage feedback from developers to keep the descriptions up to date.

This leads us to the definition of the fifth guideline:

(G5) *Integrate architects into cross-functional teams to collect feedback from developers to identify inconsistencies and capture emerging aspects of the architecture as the system evolves.*

86% of the participants ranked this guideline as very or extremely valuable. An architect from an insurance company saw little value in the guideline and stated that "*if architecture just arises then there isn't a planned architecture*." It should be noted that both emerging and upfront architecture are needed, which is why G3 is concerned with upfront architecture planned in advance. Many participants stressed the importance of G5 and reported that architects should be involved in daily development. Two respondents commented that feedback should not only be collected from developers, but also "*users, managers, contractors, hardware support, and financial analysts*."

Sixth guideline: Today, a variety of tools is used to create

and manage architecture descriptions. An architect from a telecommunications company suggested to version them together with the source code to enable a more natural connection between architecture and implementation. An automotive architect explained that it can be beneficial to use a common systems engineering tool for the high-level architecture, but also for the design on a lower level. Accessibility and traceability are important properties to ensure that an architecture description is used.

(G6) *To avoid inconsistencies, use one accessible source of information. Make the architecture description traceable for others to keep it up to date.*

This guideline was ranked extremely valuable by 58% of our respondents and very valuable by 22%. Many of the comments related to the challenging nature of implementing this guideline, the need for better modeling tools, versioning mechanisms, and lightweight support for traceability.

VII. Related Work and Relation with Guidelines

Agile processes and software architecture can be successfully combined and be mutually supportive [4]. A challenge when creating architecture descriptions in agile contexts is the balance of up-front and emerging architecture [20]: Practitioners should try to avoid making decisions too early [20], but need to make some decisions early on to avoid architectural rework later.

In this paper, one of our main focuses lied on the relation between architecture descriptions and the implementation. In the architecture life cycle, we focus on the stage of *architectural implementation* [21] and developers as the main consumers of information. We can confirm the observation that the issue of aligning architecture and implementation is a prevalent concern, also in agile methods [5], [3].

Architectural inconsistencies can result in architectural debt [22]. If inconsistencies are not counteracted but remain as wrong assumptions in the architecture descriptions, technical debt emerges. In fact, the importance of consistency has been proclaimed since the beginnings of the field of software architecture [23]. As the architecture evolves over time, inconsistencies are likely to arise, which requires architects to also consider the perspective of time [24]. Our suggested guidelines can help to prevent architecture decay [25] and architecture erosion [23], [26], as they work towards a better alignment of architecture descriptions and implementation. Architectural drift emerges as a result of an obscure architecture that stakeholders have become insensitive to [23]. Better understanding the architecture description's purpose, its relation to the implementation, and connecting architects and developers can help mitigating these problems.

Our guidelines have a connection to related work in multiple ways. An overview is shown in Table III. Guideline G1 ("state purpose and audience") is the basis to all subsequent guidelines and stages, but nevertheless often neglected in practice. We found that in our participating companies, but also in related empirical studies (e.g., [27], [28]), architecture descriptions were used for very different purposes, e.g., communication with developers, or more formal analysis of planned changes. Our findings indicate that practitioners are not always aware of the purpose of their architecture descriptions. Still, the purpose is crucial to decide on what elements should be part of the architecture.

Table III
Guidelines for architecture descriptions and the discussed related work

	Guideline Description	References
G1	State purpose and audience to define the level of abstraction and inclusion criteria.	[27], [28]
G2	Distinguish between present and future perspectives.	[29], [30]
G3	Minimize the number of elements in the upfront architecture. Include elements that are relevant across team boundaries.	[20], [31], [32]
G4	Assess decisions in the description of the future architecture in communities of practice.	[20], [33], [24]
G5	Integrate architects into the teams to understand emerging aspects.	[34], [35], [36]
G6	Use one source of information and make it traceable.	[27], [37]

G2 is concerned with the distinction between current- and future-state architecture, which is used for enterprise architecture [29]. Related work advised to create a read-only description of the current state and future-state models for project teams developing the design of the future architecture [30]. We found that in some companies, a future-state architecture description is not required at all. It depends on the position in the system's life-cycle, the size and structure of the organization, and the number of planned changes.

Guideline G3 ("minimize the number of elements, but focus on elements across team boundaries") relates to Waterman's advice to design only for the immediate future and to delay decision making [20]. We recommend to include elements that concern more than one cross-functional team, which also connects to the relation of software architecture and social debt [31]. Architectural decisions that concern more than one team require good architecture communicability, which is why they should be transparently documented in an architecture description [31]. In a previous study [32], we found that architecture models and descriptions are examples of "boundary objects" between multiple cross-functional teams, which are used to create a common understanding across sites while preserving each team's identity.

G4 ("assess decisions, validate architectural decisions in communities of practice") relates to connecting architecture experts in communities of practice to reason about change [33]. Related work suggested to create an architecture road-map and plan for the system's evolution [24], and to prove the architecture with code iteratively [20]. However, in practice, besides implementing decisions in the actual system, our findings indicate that it can be beneficial to use

other means of analysis to validate the feasibility of planned changes (e.g., simulations).

G5 ("integrate architects into teams to capture emerging aspects") is in line with the recommendation to connect architects more closely with the development teams and capture emerging aspects on-demand [34]. It has been suggested to make architects responsible for promoting decisions and communicating knowledge to the teams [35]. Agile teams that did not have an architect had deteriorated code quality [36]. Due to resource constraints, it is not always possible to integrate an architect into every cross-functional team, as stated by an architect from a telecommunications company.

Guideline G6 ("use one source of information") is concerned with having a traceable and common way of storing architecture descriptions. Related work has used a whiteboard to create and document the architecture in a collaborative manner [27], which might not be scalable for large-scale development contexts. Tool support can also help to ensure consistency between architecture descriptions and the implementation. A systems architect working at an automotive company recommended to use a systems engineering tool across teams and ensure that teams can trace their artifacts to the architecture description. Supporting G6 and improving traceability can also support collaboration between (distributed) stakeholders [37].

VIII. Conclusions and Future Research

In this paper, we presented two industrial surveys with 93 and 72 practitioners examining inconsistencies related to architecture descriptions, their development over time, and guidelines to improve the consistency and usefulness of architecture descriptions. Our findings indicate that the alignment of architecture and implementation is still a prevalent issue in practice. While inconsistencies related to wording and language are not seen as very critical, practitioners observe severe consequences of inconsistencies with interface specifications, rules and constraints, and patterns and guidelines. These consequences typically arise at later stages of the development lifecycle. Most commonly, inconsistencies arise due to a lack of time, knowledge, or communication. Especially across team borders, inconsistencies are likely to arise. The importance of consistency increases after the initial creation of the architecture. Developers become more involved in this stage. Our respondents see a value in describing both the current-state architecture and the future-state architecture. Moreover, we found that one aims to refine the architecture during the design and development. However, finding a balance between upfront and emerging architecture is challenging.

Based on these insights, we developed and successfully evaluated guidelines to create and maintain consistent and useful architecture descriptions. We suggest to clearly state the purpose and intended audience of an architecture description and separate the current- and future-state architecture. The upfront architecture description should be kept small, but include elements that are relevant across team borders. For the future architecture, decisions should be assessed and reasoned about in communities of practice for architects. Moreover, we recommend to integrate architects into cross-functional teams and capture emerging aspects of the architecture. Finally, we suggest to use one accessible source of information and make the architecture description traceable.

Practitioners can benefit from the presented findings and guidelines by understanding how inconsistencies develop over time and applying our guidelines to their contexts. Researchers can benefit from empirical evidence of the state of the practice of architecture descriptions and inconsistencies. The presented research can serve as the motivation for future studies, e.g., to apply and validate our guidelines in industrial case studies. Several of our guidelines can be supported by more concrete practices and approaches.

Methods and techniques to analyze and capture emerging aspects in architecture descriptions could be of value to practitioners, especially with a focus on supporting non-architect stakeholders to leverage and modify architecture descriptions. It is also interesting to explore how current-state and future-state architectures can be developed in parallel, including the mentioned timeline to transition between them. The concept of boundary objects requires further investigation, to identify what information should be in- and excluded in the architecture description. Our findings indicate that tool support, the lack of suitable modeling tools for architecture descriptions, and the need for lightweight traceability are challenging areas and warrant research.

Acknowledgments

We would like to thank all participants for their kind support. Moreover, the work acknowledges support by the Vinnova projects ASSUME, NGEA and NGEA step 2, the Software Center (software-center.se), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] M. Shaw and P. Clements, "The golden age of software architecture," *IEEE Software*, vol. 23, no. 2, pp. 31–39, March 2006.

[2] M. A. Babar, *Supporting the Software Architecture Process with Knowledge Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 69–86.

[3] P. Abrahamsson, M. A. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Software*, vol. 27, no. 2, pp. 16–22, March 2010.

[4] G. Booch, "An architectural oxymoron," *IEEE Software*, vol. 27, no. 5, pp. 96–96, Sep. 2010.

[5] M. A. Babar, "An exploratory study of architectural practices and challenges in using agile software development approaches," in *WICSA 2009*, 2009, pp. 81–90.

[6] M. Waterman, J. Noble, and G. Allan, "How much up-front? a grounded theory of agile architecture," in *ICSE'15*, 2015, pp. 347–357.

[7] H. P. Breivold, I. Crnkovic, and M. Larsson, "A systematic review of software architecture evolution research," *Information and Software Technology*, vol. 54, no. 1, pp. 16–40, 2012.

[8] T. Mens and S. Demeyer, Eds., *Software Evolution*. Springer-Verlag Berlin Heidelberg, 2008, vol. 1.

[9] R. Heldal, P. Pelliccione, U. Eliasson, J. Lantz, J. Derehag, and J. Whittle, "Descriptive vs prescriptive models in industry," in *MODELS 2016*, 2016, pp. 216–226.

[10] U. Eliasson, R. Heldal, P. Pelliccione, and J. Lantz, "Architecting in the automotive domain: Descriptive vs prescriptive architecture," in *WICSA 2015*, 2015, pp. 115–118.

[11] M. Feilkas, D. Ratiu, and E. Jürgens, "The loss of architectural knowledge during system evolution: An industrial case study," in *ICPC 2009*, vol. 00, May 2009, pp. 188–197.

[12] G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," *Handbook of Software Engineering and Knowledge Engineering*, vol. 1, pp. 329–380, 2001.

[13] N. Ali, S. Baker, R. O'Crowley, S. Herold, and J. Buckley, "Architecture consistency: State of the practice, challenges and requirements," *Empirical Software Engineering*, vol. 23, no. 1, pp. 224–258, feb 2018.

[14] B. A. Kitchenham and S. L. Pfleeger, *Personal Opinion Surveys*. London: Springer London, 2008, pp. 63–92.

[15] F. J. Fowler, Jr., *Survey Research Methods*, 3rd ed., ser. Applied Social Research Methods. Thousand Oaks, CA: SAGE Publications, 2002.

[16] *ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description*, ISO/IEC/IEEE, December 2011.

[17] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 5–55, 1932.

[18] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer, Berlin, Heidelberg, 2012, vol. 9783642290.

[19] P. Kruchten, "What do software architects really do?" *Journal of Systems and Software*, vol. 81, no. 12, pp. 2413–2416, 2008.

[20] M. Waterman, "Agility, risk, and uncertainty, part 1: Designing an agile architecture," *IEEE Software*, vol. 35, no. 2, pp. 99–101, March 2018.

[21] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. A. Babar, "A comparative study of architecture knowledge management tools," *Journal of Systems and Software*, vol. 83, no. 3, pp. 352–370, 2010.

[22] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012.

[23] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, oct 1992.

[24] E. Poort, "Just enough anticipation: Architect your time dimension," *IEEE Software*, vol. 33, no. 6, pp. 11–15, Nov 2016.

[25] M. Riaz, M. Sulayman, and H. Naqvi, "Architectural decay during continuous software evolution and impact of 'design for change' on software architecture," in *ASEA 2009*, 2009.

[26] L. De Silva and D. Balasubramaniam, "Controlling software architecture erosion: A survey," *Journal of Systems and Software*, vol. 85, no. 1, pp. 132–151, 2012.

[27] B. Weitzel, D. Rost, and M. Scheffe, "Sustaining agility through architecture: Experiences from a joint research and development laboratory," in *WICSA 2014*, 2014, pp. 53–56.

[28] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, June 2013.

[29] X. Wang, X. Zhou, and L. Jiang, "A method of business and IT alignment based on enterprise architecture," in *SOLI 2008*. IEEE, 2008, pp. 740–745.

[30] M. Lankhorst, "6 ways to organize your architecture models (part 1)," *BiZZdesign Blog*, 2018, accessed on February 5, 2019.

[31] D. A. Tamburri and E. D. Nitto, "When software architecture leads to social debt," *WICSA 2015*, no. September 2005, pp. 61–64, 2015.

[32] R. Wohlrab, P. Pelliccione, E. Knauss, and M. Larsson, "Boundary objects in agile practices: Continuous management of systems engineering artifacts in the automotive domain," in *ICSSP 2018*. ACM, 2018, pp. 31–40.

[33] T. Kähkönen, "Agile methods for large organizations - building communities of practice," in *Proceedings of the Agile Development Conference*. IEEE, 2004, pp. 2–10.

[34] R. Weinreich and I. Groher, "The architect's role in practice: From decision maker to knowledge manager?" *IEEE Software*, vol. 33, no. 6, pp. 63–69, Nov 2016.

[35] R. Britto, D. Smite, and L. Damm, "Software architects in large-scale distributed projects: An ericsson case study," *IEEE Software*, vol. 33, no. 6, pp. 48–55, Nov 2016.

[36] S. Frey, L. Charissis, and J. Nahm, "How software architects drive connected vehicles," *IEEE Software*, vol. 33, no. 6, pp. 41–47, Nov 2016.

[37] R. Wohlrab, E. Knauss, J.-P. Steghöfer, S. Maro, A. Anjorin, and P. Pelliccione, "Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture," *Requirements Engineering*, Nov 2018.