Less is More: Architecture Documentation for Agile Development

Irit Hadar
Information Systems Dept.
University of Haifa
Haifa, Israel
hadari@is.haifa.ac.il

Sofia Sherman
Information Systems Dept.
University of Haifa
Haifa, Israel
shermans@is.haifa.ac.il

Ethan Hadar
CTO, AGT International
Zürich, Switzerland
ehadar@agtinternational.
com

John J. Harrison Jr.
CTO Office
CA Technologies
Framingham, USA
jay.harrison@ca.com

***Abstract*-The use of agile development methodologies in the software industry has increased significantly over the past decade. This has led to efforts to adjust these methodologies to enterprise products and complex systems' development, and specifically to combine the requirement of minimalism with the need for well-defined up-front architecture artifacts. Nevertheless, in many cases, architecture is accompanied with extensive documentation that requires significant effort to review and maintain throughout the development lifecycle. This paper presents a case study that aims at identifying difficulties architects and other stakeholders encounter when dealing with architecture documentation in agile development. The findings indicate that the architecture specification document is usually very long, complex, and not self-explanatory. In order to adjust the architecture documentation to the lean and minimal documentation approach of agile processes, we propose a considerably shorter abstract specification document, requiring reduced documentation efforts and resulting in a lean documentation that is easier to review, update, and communicate.**

***Index Terms*—Software architecture in agile development, architecture documentation.**

I. INTRODUCTION

The use of agile methodologies in industry has increased significantly over the past decade, promoting the value of human-centric software development process. This growing use derives the need to adjust agile methodologies to bigger, more complex system development projects, where architecture plays a significant role. Although there are still believers that there is a conflict between the requirement of minimalism in agile methodologies and the need for well-defined and documented architecture in complex systems [1], it seems that agile architecture is no longer an oxymoron [2]. (For an elaborated discussion on this topic, see [3].)

Agile methodologies emphasize rapid and flexible development. The main characteristics of agile development are short releases, flexibility, and minimal documentation [4]. In order to adjust architecture documentation to the lean and minimal documentation approach of agile processes, we propose an architecture specification document requiring reduced documentation efforts and resulting in a lean documentation that is easier to review, update, and communicate.

The rest of the paper is organized as follows: the next section briefly discusses the content and challenges of architecture documentation. Section 3 describes the empirical study we conducted and its findings, followed by the proposed solution in section 4. Section 5 discusses the proposed solution in the context of architecture documentation challenges. Finally, section 6 concludes.

II. ARCHITECTURE DOCUMENTATION

Architecture is one of the most complex and heavily documented parts of software development. This documentation helps to maintain consistency when making architecture design decisions and facilitates communication with other system stakeholders [5]. One of its main purposes is to provide instruction about how the system is structured, how it works, and the design rationale that led to it [5]. Architecture documentation has to be sufficiently transparent and accessible to be quickly understood, sufficiently concrete to serve as a blueprint for construction and include enough information to serve as a basis for analysis [2]

Current literature on architecture documentation mostly relates to heavy weight development methodologies. In these methodologies, architecture documents usually contain essential architecture information such as design decisions [2][4][6][7], components and connectors descriptions [2][8] and related activities in the development process [4][7]. Much emphasis is made on essential design decisions and design rationale, to allow understanding the circumstances in which these decisions were made [4]. On the other hand, there is a need to broaden the architecture's perspective to non-technical uses [9]. Since growing technologies require cross-organization collaboration, architecture documentation must be accessible to, and understood by, additional stakeholders who are not software developers [10].

Although architecture documentation is clearly important, there are many challenges architects and other stakeholders deal with while handling it. Architecture documents are usually large and complex, and may grow from tens to hundreds of pages; they consist of multiple documents that within and between them encompass many concepts and relationships, multiple views, different levels of abstraction, and numerous consistency issues. Current software architecture documentation approaches cannot efficiently cope with this size and complexity [6]. Jason et al. identified a list of challenges related to architecture documentation [6]. In this work we focus on the following three challenges:

1) Understandability of the documents by architects, reviewers, and other relevant stakeholders in cases of:

- Large and complex systems [7].
- Different backgrounds of stakeholders, which may hinder their understanding of the language and concepts used in the document [6].

2) Locating relevant architecture knowledge

- Finding the relevant document within a set of architecture documents (including formal documents, informal documents, emails, etc.) [11][12].
- Locating the relevant knowledge within a set of architecture documents [6][12].

3) Trust

- Keeping architecture documentation up to date, otherwise stakeholders will lose confidence in its credibility [6][13].

Looking at these challenges through the lens of agile development methodologies further emphasizes the need to reduce the length and complexity of architecture documentation. One of the main values defined in the agile manifesto is "Working software over comprehensive documentation". It does not mean that there is no space for documentation in agile development, but that documentation needs to support software and not otherwise. Thus, heavy, long architecture specification documents do not adapt well to the lean, flexible and minimalistic approach of agile development. Selic [5] refers to architecture documentation in agile development as a high abstraction level document. This kind of document needs to be stripped off unnecessary technological details and closely coupled to application concepts and requirements, including design rationale and rejected design alternatives [5]. Bass et al. [2] claim that in agile development it is preferred not to document information if it is not needed; all documentation should have an intended use and audience in mind.

III. The Empirical Study

A. Settings

This study was conducted in a large, internationally distributed software firm. At the beginning of the research, this firm practiced both agile and non-agile development methodologies, aiming to broaden the use of agile development to all divisions within this firm.

The main objective of this study was to develop an architecture specification document format aligned with the agile characteristics. To do so we needed to understand the existing architecture specification document, and identify the challenges of it use within the firm in agile development. Aiming to learn a phenomenon and identify its characteristics, rather than corroborating predetermined hypothesis, calls for the use of qualitative research methods and tools [14]. Accordingly, we used in this study the grounded theory approach [ibid].

As a first step, we conducted preliminary interviews with several architects responsible for architecture documents' creation, and with several senior architects who serve as reviewers of these documents. Next, we collected and analyzed architecture documents and artifacts. In parallel, we observed and recorded architect teams' discussions, adding these data to our iterative analysis. Based on these first steps, we identified several issues and challenges architects face with regard to architecture documentation. This enabled us to construct and distribute a questionnaire for collecting further data, focusing on these issues. We received 31 filled questionnaires from architects, all practicing agile development. Analyzing the additional data we further refined and validate our findings, identifying and understanding the main challenges architects face while working with the architecture specification document used at that time in agile development.

Based on the findings, we developed a document template and a supporting tool. The development was done iteratively, implementing the developed document template and tool and refining them based on feedbacks from the field.

The phase of evaluation and validation is still ongoing, however data collected thus far already indicates an improvement of stakeholders' satisfaction and reduced effort.

B. Findings

Data analysis revealed documentation-related challenges architects working in agile development deal with, in the context of the categories presented in section 2.

Understandability: The current architecture document is perceived to be very long, complex, and sometimes not self-explanatory. One of the senior architects, who serves as an architecture reviewer, stated: "*Sometimes I need to review a 300 pages architecture document. It is difficult to read and understand such a long documents. I cannot read it all since I have other responsibilities, and I have at least two such reviews to perform every month*". The length of the architecture document was partly a result of it consisting vast information that was very important for the project in general, but did not focus on the aspects relevant to the architecture, as another architecture reviewer stated: "*In many of its parts, the document does not contain information potentially interesting to me, so I am usually ignoring these parts of the document*". This was also confirmed by the questionnaire: 42% of the architects who participated in the survey reported that they do not understand parts of the architecture documents they read.

Locating relevant architectural knowledge: as discussed, the architecture document was very long and complex. It contained much knowledge that was usually not reused, since it was quite difficult to access and find. In the interviews architects referred to this issue repeatedly. For example: "*I don't usually look for information in architecture documents from previous versions or other projects. I use what I remember, or I reinvent*". In the survey, 48% of the architects stated that it is hard to look for information in the current architecture document, and 42% stated that architecture documentation from previous projects is hard to read and follow.

Keeping documentation up to date: this challenge was already identified and discussed in our previous work [13]. We concluded then that: (1) despite of the importance emphasized in literature of keeping architecture documentation updated, not all architects perceive architecture documentation maintenance as important; and (2) architecture documentation maintenance is usually practiced inconsistently in ad-hoc processes and [13].

Document creation: this is a newly identified challenge, which had emerged from the interviews and was later confirmed by the survey. Software architects face difficulties when creating the architecture document: "*It takes a lot of time to write an architecture document, especially when writing it for the first time*", and keeping it up to date: "*Requirements changes uncovered during the build phase are usually not incorporated into the architecture documentation*." In the survey, 52% of the architects stated that it is hard to create an architecture document, and 62% stated that they spend a lot of time and effort to create this document. However, using a template for document creations helps architects think about possibilities that they were not aware off (68%).

IV. THE PROPOSED SOLUTION

A. *Abstract Architecture Specification Document*

The Abstract Architecture Specification (AAS) document we propose includes the most relevant and updated information regarding the architecture solution designed for the current release, as well as the status of the product's existing architecture. The shaping principle of this document is to keep it short by employing elevator speech concepts. The document consists of four major sections (Table 1): product overview, product goals for upcoming release, product architecture overview as a whole interweaved with the sprint changes, and non-functional requirements. The embodied architecture blueprint is presented with four conceptual layers, catering for different stakeholders (system components layer, common and cross concern components layer, external integration components layer and functional components layer). The document captures relevant snapshots of blueprints and a short explanation on the components involved. For the functional components layer, for example, the snapshot of the functional architecture layer extracted from the architecture model is captured as a blueprint and added to the document. The relative components are described in text as well, according to the blueprints sections. A list of acronyms used in the document and links to related documents are also provided. In order to keep it short and focused, we propose to keep it template-oriented; architects will insert relevant data in predefined fields rather than generating unstructured freeform text.

TABLE I. AAS DOCUMENT STRUCTURE

Section	Description	Content
Product Overview	Product summary description *Limited to 100 words*	1. Product goals and business value. 2. Problems the product solves. 3. Target company types and target users. 4. Links to relevant documentation of this or similar products.
Product Goals for Upcoming Release	Description of the product objectives for the upcoming release *Limited to 100 words*	1. Planned deliverables. 2. Links to relevant documentation of this and/or similar products.
Product Architecture Overview	General description of the system architecture at the current release and the changes anticipated for the upcoming release *Limited to 1000 words*	1. Major components used in the current release. 2. Planned changes in existing components for upcoming release. 3. New components that will be used in the upcoming release. 4. Related architecture blueprints.
Fundamental Requirements	A table describing the project goals and architecture impacts of the non-functional requirements	1. Coverage of the current release: Quality attributes that the product supported in the past. 2. Planned coverage for the new release: Quality attributes that will be supported in the future.
Acronyms and Definitions	Acronyms used in this document including:	1. Brief definition for each acronym. 2. Relevant links if more explanation is required.

The AAS document, authored by the architects, allows facilitating a common language for architects. It simplifies communication between architects and other stakeholders. For example, it streamlines information so that reviewers can focus on the content and the design decisions rather than on lengthy narrative statements, blurring the design intent and product architecture structure. Reviewing a short and focused document leads to constructive and targeted comments from reviewers. Reviewers can also easily follow up the changes made in the architecture in different releases, and assess the evolution of the architecture over time. To make it easier for the architects to create such a document, we automate this process by creating an Abstract Specification Tool.

B. *Abstract Specification Tool*

The main purpose of the Abstract Specification Tool is to help the architects in organizing relevant information regarding the architecture while creating design and architecture blueprints, thus reducing the effort of documentation. This tool provides a 'form to fill' and is bundled with an architecture modeling tool. Both facets share the same database, which allows the information in the Abstract Specification Tool to be updated as necessary, either via the forms or directly via the modeling tool. This reduces the information the architect is required to manually insert, captures the knowledge and design decisions during the design phase, and enables the reuse or restructuring of the information later on. For example, when a new component is added to the blueprint in the modeling tool, the architect inserts all required information within the modeling tool, and the relevant data is created by associated queries as part of the Abstract Specification Tool.

Another feature of the tool includes enabling architects to coordinate their work using a checklist. The checklist is integrated within the tool and guides the architects throughout the architecture development. It evolved over time, based mainly on reviewers' comments. The outcome of the tool is generated architecture documentation.

V. Discussion

Four main challenges architects face regarding architecture documentation were identified; three of them had been previously discussed in literature. Aiming to meet these challenges and overcome related difficulties, we proposed a solution in the form of an abstract architecture document and a supporting tool. In this section we discuss how features of this solution correspond with the challenges and difficulties identified.

Understandability: The proposed solution makes it easier to understand the document, since the document in its new form is short, focused and structured according to a predefined and consistent template. Although the previously used document form in this firm was also template oriented, it included: (1) other topics, which are important to the project as a whole, but not necessary for understanding its architectural information. These topics are documented in other documents (e.g., SRS) and when relevant, links to these documents are provided, thus also avoiding redundant information and integrity issues. (2) The proposed template and tool help the architects to focus on the required information only, and do not allow them to provide lengthy explanations and unnecessary details.

Locating relevant architecture knowledge: The tool provides document users with easier knowledge location process, since it provides links to necessary information in other documents, and provides the architects access to the components' database used in previous versions of the current project and of other projects, making potentially reusable knowledge more accessible.

Keeping documentation up to date: The current version of the tool does not yet fully support architecture document update. However, the next step in this ongoing research is to define a coherent process to architecture maintenance and accordingly add this function to the tool.

Document creation: The tool supports the architects in preparing the document. It guides the architects through all parts of the document by listing the exact information required, by automatically filling information that already resides in other systems (e.g., the modeling tool), and via the checklist that guides the architects to think about additional aspects of the project they have not considered.

VI. Conclusion

The main concepts of agile development are flexibility, minimalism and collaboration. These are achieved when constructing an abstract architecture specification document using a supporting speciation tool for creating a short and focused architecture document. It is nimble and flexible to changes, bundled and integrated with a modeling tool; and is supported by evolving checklists. This short and focused approach supports the minimalism in the agile development and is assisted by an automatic tool and a template for creating the document, facilitating common language, supporting collaboration between stakeholders and fostering easy access to, and location of, architecture knowledge.

References

[1] G. Booch, "An Architectural Oxymoron," IEEE Software, vol.27, no. 5, pp. 95-96, 2010.

[2] L. Bass, P. Clements and R. Kazman, "Software Architecture in Practice". 3rd ed., Addison-Wesley Professional, 2013.

[3] I. Hadar and S. Sherman "Agile vs. Plan-Driven Perceptions of Software Architecture", 5th Int. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), ICSE, June 2012.

[4] J. Tyree and A. Akerman, "Architecture Decisions: Demystifying Architecture". IEEE Software, vol. 22, no. 2, pp. 19-27, 2005.

[5] B. Selic, "Agile Documentation Anyone?" IEEE Software, vol. 26, no. 6, pp.11 – 12, 2009.

[6] A. Jansen, P. Avgeriou and J.S. van der Ven, "Enriching Software Architecture Documentation," The Journal of System and Software. vol. 82, pp. 1232-1248, 2009.

[7] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord and J. Stafford, "Documenting Software Architectures: Views and Beyond." Addison Wesley, 2002.

[8] A. Tang, Y. Jin and J. Han, "A Rationale-Based Architecture Model for Design Traceability and Reasoning", Journal of Systems and Software, vol. 80, no 6, pp. 918-934, 2007.

[9] N. Medvidovic, E.M. Dashofy and R.N. Taylor, "Moving Architectural Description From Under the Technology Lamppost", Information and Software Technology. vol. 49, no. 1, pp. 12–31, 2007.

[10] K. Smolander, M. Rossi and S. Puraro, S. "Software Architectures: Blueprint, Litrature, Language or Decision?" European Journal of Information Systems, vol. 17, no. 6, pp. 575–588, 2008.

[11] R. C., de Boer and H. van Vliet, "Architectural Knowledge Discovery With Latent Semantic Analysis: Constructing a Reading Guide for Software Product Audits", Journal of Systems and Software, vol. 81, no. 9, pp. 1456-1469, 2008.

[12] S. Sherman, I. Hadar and M. Levy, "Enhancing Software Architecture Review Process via Knowledge Management", The Sixteenth Americas Conference on Information Systems, AMCIS, Lima, Peru, August 2010.

[13] S. Sherman and I. Hadar, "Identifying the Need for a Sustainable Architecture Maintenance Process", 5th Int. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), ICSE, June 2012.

[14] A. Strauss and J. Corbin, "Grounded Theory Methodology: An Overview", In Denzin, N.K. and Lincoln, Y.S. (Eds.), Handbook of qualitative research, ch. 1, Thousand Oaks: Sage.1994.