Exploring the Adoption and Effectiveness of Architecture Decision Records in Agile Software Development: An Action Research Study

Bardha Ahmeti
Chalmers | University of Gothenburg
Sweden
gusahmeba@student.gu.se

Maja Linder
Chalmers | University of Gothenburg
Sweden
guskalmas@student.gu.se

Rebekka Wohlrab
Chalmers | University of Gothenburg
Sweden
wohlrab@chalmers.se

ABSTRACT

Software development companies need to document and communicate their software architecture decisions. While a lot of research has been done on software architecture, there exist few established practices that are applied in industry. It is often difficult to establish architecture documentation practices that are lightweight and perceived as useful by agile developers. In recent years, a markdown-based approach called Architecture Decision Records has been proposed. There is little empirical evidence and research on how Architecture Decision Records are used in practice. This poster presents an action research study on the use of Architecture Decision Records in practice. We introduced Architecture Decision Records at a company over the course of several months, and observed challenges and advantages of using them. We found that Architecture Decision Records are perceived as useful by development teams and increase the awareness that documentation is beneficial even in agile development contexts. At the same time, practitioners see a risk in documenting too much or too little and the need for clear guidelines.

KEYWORDS

software architecture, software processes, documentation, action research, empirical research, software engineering

ACM Reference Format:
Bardha Ahmeti, Maja Linder, and Rebekka Wohlrab. 2018. Exploring the Adoption and Effectiveness of Architecture Decision Records in Agile Software Development: An Action Research Study. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Software architecture documentation plays a significant role in developing high-quality software. It involves describing the overall structure of the software, including the relationships between its components, and the reasoning behind the architectural decisions made. These decisions can have significant impacts on the software's functionality, performance, and maintainability, making documenting them particularly crucial. However, the traditional methods of documenting software architecture have been associated with complexity and lengthiness, which can make it challenging to retrieve and maintain accurate information [1]. In the past, agile practitioners often relied on oral communication to discuss architectural concerns. While oral communication has been a common way to keep track of architectural decisions, it is not without limitations [1]. Oral communication is highly dependent on individual memory, and therefore prone to inaccuracies, inconsistencies, and changes over time. Additionally, it requires a lot of attention and can be challenging to coordinate and reach all members across multiple teams, which can lead to knowledge silos and hinder collaboration. Similar challenges have been found by Paasivaara et al. [6] when exploring Scrum of Scrum meetings.

While agile methodologies advocate for a minimalist approach that favors oral communication over extensive documentation, internal documentation remains critical for preserving decisions and avoiding misinterpretations over time. Nevertheless, agile teams often express dissatisfaction with the lack of collaboration and inclusivity in the production of software documentation [2].

One solution to the challenges of documenting software architecture decisions is Architecture Decision Records (ADRs) [3–5]. ADRs are lightweight, markup language documents that reside in the repository and are created near the code. Each ADR captures the essence of a single architectural decision, enabling developers to enhance their architectural knowledge and promote team culture [4]. Although there are potential benefits to using ADRs in software development, there is currently a lack of empirical research on their use, particularly in agile development teams. This means that little is known about their effectiveness in practice.

In line with the theme "Computing Connecting Everyone", we aimed to close the gap between industry and academia, introduce ADRs in a real-world setting, and gather empirical evidence on its use and perceived challenges. This study aims to fill the gap in empirical research on the use of Architecture Decision Records (ADRs) in software development. Through action research, we sought to introduce ADRs to development teams and explore their potential in improving developer involvement in decision-making and addressing challenges related to documenting architectural decisions. By conducting an empirical study on the adoption and effectiveness of ADRs, we aim to contribute to the broader conversation on lightweight documentation in software development and provide

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ADR 003 Centralizing External User Data and Logic in Application.ExternalUser

Context
Application.ExternalUser and Application.Collaboration are two microservices that maintain the logic and data for External Users. [..]

Decision
To simplify the maintenance and usage of External Users, Application.ExternalUser will assume full responsibility for the life-cycle of External Users, including data on invitations and the logic related to their usage. [...]

Status
Proposed

Consequences
Centralizing the data and logic related to External Users within Application.ExternalUser will simplify the maintenance and usage of External Users. It will eliminate the need for other services to duplicate logic. Application.ExternalUser will be the only dependency for services when utilizing External Users.

Figure 1: Example of an Architecture Decision Record (ADR)

practical insights into the use of ADRs. The research questions focus on identifying the challenges in documenting architectural decisions in an agile context, assessing the impact of ADRs on improving developer involvement in decision-making, and exploring the adoption process of ADRs in software development:
RQ1: What are the main challenges in documenting software architectural decisions in an agile team?
RQ2: To what extent can ADRs alleviate the identified challenges in documenting software architectural decisions in an agile team?
RQ3: To what extent can ADRs improve architecture decision involvement from the developers' perspective?

Our methodology involves action research, where we worked closely with two development teams to introduce and evaluate the use of ADRs in their software development process in two cycles. This study was conducted in collaboration with a software development company that has expressed interest in exploring new methods for documenting software architecture decisions. We worked with two development teams at the company. The teams are composed of employees with varying levels of experience, age, and roles, including back-end and front-end developers, testers, and architects with development responsibilities. We collected data through observations, interviews, and focus groups with team members, and analyzed the data using thematic analysis.

On the poster at womENcourage, we plan to describe the context of the study in detail, present our findings, and present lessons learned for how we can further bridge the gap between research on architecture documentation and industrial practices.

2 RELATED WORK

Architecture Decision Records are an approach to documenting architectural decisions. ADRs were first described a decade ago by Nygard in a blog post [5]. Ever since, they have been adopted by companies around the globe. It is only recently that ADRs are brought up in literature.

Keeling [3] states that ADRs have become increasingly popular among agile teams as a means of bridging the gap between architecture principles and agile practices. Further, he suggests that ADRs appeal to agile teams because they provide a simple yet effective way of documenting design decisions and their rationale, which helps teams understand the trail of decisions leading to a design and improve their ability to scale up their organizations, handle staff turnover, and evolve the system over time.

In his book "Design It" [7], Keeling emphasizes the importance of capturing architecture design decisions using a lightweight, text-based template, namely ADRs. He argues that this is a developer-friendly approach that documents design decisions as they are made, which makes it easier to share and analyze these decisions, while retaining a history of decisions provides context for the current architecture relative to its evolution. According to Keeling, by using ADR templates, teammates can be trained in architectural thinking, and peer review of design decisions can be enabled using standard development tools and an existing peer review workflow.

3 PRELIMINARY FINDINGS

We performed preliminary interviews and workshops with 7 developers, architects, and team leads. The company was positive towards introducing ADRs. Figure 1 shows an example of an ADR that was developed at the company. It can be seen that it deals with the logic and data management in two microservices. In a previous version, the logic and data management was split between different microservices. That split had negative effects on the system's maintainability and modifiability. With the proposed ADR, it will be centralized and only managed in the microservice called "Application.ExternalUser".

Several interviewees thought that the action research study could help with onboarding, raising the perceived need for documentation of architecture decisions, and creating clarity with respect to architectural concerns in the company. For example, a developer stated that *"it's good to make it less person-dependent. If someone leaves the company and has worked a lot in those specific parts of the system, maybe a new employee has no idea about the thoughts when developing it."* When womENcourage takes place, we will have finished the study and can report on the findings, both positive and negative.

REFERENCES

[1] I. Hadar, S. Sherman, E. Hadar, and J. J. Harrison Jr., "Less is More: Architecture Documentation for Agile Development," IEEE, San Francisco, CA, USA, 25-25 May 2013.
[2] C. J. Stettina, W. Heijstek and T. E. Fægri., "Documentation Work in Agile Teams: The Role of Documentation Formalism in Achieving a Sustainable Practice," IEEE, Dallas, TX, USA, 13-17 August 2012.
[3] M. Keeling., "Love Unrequited: The Story of Architecture, Agile, and How Architecture Decision Records Brought Them Together," IEEE Software, The Pragmatic Designer, 20 June 2022.
[4] M. Keeling., "The Psychology of Architecture Decision Records," IEEE Software, The Pragmatic Designer, November/December 2022.
[5] M. Nygard., "DOCUMENTING ARCHITECTURE DECISIONS," Cognitect, November 15, 2011. Accessed: Feb.03, 2023. [Online]. Available: https://www.cognitect.com/blog/2011/11/15/documenting-architecture-decisions
[6] M. Paasivaara, C. Lassenius, VT. Heikkilä., "Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?". In Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement 2012 (pp. 235-238).
[7] M. Keeling "Design It!: From Programmer to Software Architect." 2017:1-358.