State-of-Practice in Architectural Change Management for Software-Intensive Systems: An Interview Study

Ifrah Qaisar
Mälardalen University
Västerås, Sweden
ifrah.qaisar@mdu.se

Robbert Jongeling
Mälardalen University
Västerås, Sweden
robbert.jongeling@mdu.se

Jan Carlson
Mälardalen University
Västerås, Sweden
jan.carlson@mdu.se

***Abstract*—Modern software-intensive systems are growing more complex, evolving continuously, and requiring extensive collaboration across diverse domains. Effective documentation and communication of architectural changes are critical to managing the development of such systems. But this task remains challenging due to constraints in time, resources, and standardized practices. This study investigates the practices and challenges related to architectural documentation and communication in the development of software-intensive systems through semi-structured interviews with architects from four partner companies. The findings reveal key challenges, including identifying stakeholders for communication, inconsistent architectural definitions across teams, and knowledge transfer issues due to differing tools and terminologies. Participants also shared their reliance on informal mechanisms and ad hoc processes to address these challenges. Insights from this study highlight the need for more structured, standardized approaches, flexible documentation methods, and improved cross-domain collaboration tools. The results provide actionable recommendations for both practitioners and researchers to enhance documentation practices, improve team alignment, and ensure the effective management of architectural changes in dynamic and large-scale software systems.**

***Index Terms*—Software-intensive systems, architectural changes, semi-structured interviews, cross-domain communication**

I. Introduction

Software-intensive systems are systems where software plays a critical role in enabling interactions with various components such as other systems, sensors, actuators, devices, applications, and users [1]. These systems are often highly interconnected and operate across different domains, including automotive, telecommunications, embedded systems, industrial automation, and business applications. Their complexity arises from the need to manage diverse and interdependent elements within dynamic and evolving environments.

Within this context, architectural and design documentation emerges as a critical component, playing a key role in managing complexity while ensuring system reliability, scalability, and maintainability. Within the context of this study, architectural documentation is primarily intended for internal use during development and maintenance. It serves as a record of design decisions and their rationale while offering a clear view of the system's structure to provide a common point of reference for cross-functional teams, including developers, testers, managers, and other stakeholders. Additionally, it serves as a vital communication tool, enabling stakeholders to understand and align on the design decisions and supporting effective collaboration across teams [2].

Despite the important role of software architecture documentation, it is frequently overlooked by software practitioners due to several reasons such as time and resource constraints, increased complexity of the system and lack of standardized practices [2]. This lack of attention can result in miscommunication of architectural information, misaligned understanding among stakeholders, and difficulties when it comes to maintaining and evolving the system over time.

As part of a broader industry-academia collaboration with multiple partner companies, this interview study explores how architectural changes are documented and communicated within and across teams developing software-intensive systems. Our goal is to understand how software practitioners in these organizations navigate the complexities of architectural documentation, the practices they currently employ, and the challenges they face. By examining these aspects, we aim to provide actionable insights to enhance industry practices and support effective collaboration in this dynamic and evolving field. Our investigation is guided by the following research questions.

RQ1. *What are the current industry practices for documenting and communicating architectural changes in software-intensive systems?*

RQ2. *What challenges do software practitioners face when documenting and communicating architectural changes in software-intensive systems?*

Based on our analysis of the interview transcripts, we highlight key challenges and practices related to documenting and communicating architectural changes in software-intensive systems. Participants shared insights into the varying approaches used across teams or within the teams, and the difficulties faced in aligning goals and managing complexity.

This work was supported by Software Center: www.software-center.se

These insights not only provide a deeper understanding of current industry practices but also emphasize the need for more structured, standardized approaches to improve communication and collaboration. In the following sections, we present these findings and explore their implications for improving architectural documentation processes in dynamic development environments.

The remainder of this paper is organized as follows. In Section II, we describe the design of the interview study. In Section III, we present our research findings and analyses. In Section IV we discuss the findings presented in Section III. Section V highlights the related work and Section VI concludes the paper.

II. Research Methodology

A. Interview Design

In this study, we conducted semi-structured interviews, following Oishi [3] as guidelines for designing the interview process. We developed an interview guide aligned with the research questions to structure the discussion effectively. To maintain the open-ended nature of the interviews and avoid influencing the responses, the guide was not shared with the participants beforehand. The interview guide has been made available online.[1]

The interviews were conducted online using Microsoft Teams, where each session was recorded and later transcribed for analysis. Two individuals facilitated the interviews: the first author of this paper was the interviewer, leading the discussion, while the other (alternating second and third author of this paper) served as the observer, responsible for asking follow-up questions for clarification, monitoring the interview duration, taking notes, and ensuring the discussion stayed on track.

The analysis was carried out in a more relaxed and flexible manner, focusing on identifying key insights and recurring patterns in the data. Each transcript was thoroughly reviewed, and notable points were highlighted and grouped into broad categories that aligned with the research objectives.

B. Participants and Contexts

A total of 12 participants from four companies, for anonymity referred to as Company A, Company B, Company C, and Company D, were interviewed. These companies operate within the domain of software-intensive systems, encompassing diverse fields such as advanced engineering (B), mobility solutions (A), construction equipment (D), and energy-efficient systems (B,C). Three of the companies are located in Sweden and one company is located in Denmark. The participants included five representatives from Company A, two from Company B, four from Company C, and one from Company D. All interview participants held roles as system or software architects, with some also serving in additional capacities, such as product owners and developers. This diversity of roles provided a broad perspective on the practices and challenges related to architectural change documentation and communication within these organizations. When asked about team size and the number of collaborating teams, participants reported that the average team consists of approximately 10 members, typically organized as Scrum teams. However, participants were often unsure about the exact number of collaborating teams, as many are involved in large-scale projects. The details, such as their current job roles, years of experience and over all team structure including total number of teams and number of people per team are mentioned in Table I.

[1]https://doi.org/10.5281/zenodo.14733882

C. Threats to Validity

The relatively small sample size of 12 participants limits the generalizability of the findings. With a small number, the results might not fully represent the broader industry or capture diverse perspectives, especially when considering the large and varied nature of the software industry. We mitigate this threat by selecting experts from organizations operating in different sectors of software-intensive systems, thereby covering a broader range of industry practices. Additionally, by including participants from different companies in the domain of software-intensive systems, we captured insights from different organizational contexts, further enhancing the richness of the data. While the sample size remains a limitation, we aimed to gather in-depth, qualitative insights from those with direct experience in architectural documentation and communication within their respective teams.

Interviewees were invited based on relevance to the topic of architectural documentation and communication in software-intensive systems, the selection was non-random, which introduces a potential threat of selection bias. However, because our study focuses on the specific domain of software-intensive systems within multiple industries, a random selection approach would not have been suitable. To mitigate the potential threat of selection bias, we carefully selected interviewees who were directly involved in architectural documentation and communication within software-intensive systems. We ensured that participants represented a variety of roles, including system and software architects, product owners, and developers, to gather diverse perspectives on the topic. Furthermore, by selecting participants from multiple companies working across different sectors (e.g., engineering, mobility solutions, and energy-efficient systems), we aimed to capture a broad set of practices and challenges, reducing the risk of bias introduced by focusing on a single organization or industry. This purposeful, targeted selection allowed us to gather relevant and diverse insights while maintaining a focus on the specific challenges in the domain of interest.

A notable limitation of this study is the over-representation of participants from Company A, as half of the interviewees were from this organization. While this might skew the findings toward practices and perspectives specific to Company A, it is important to note that Company A is a very large organization with participants coming from different subunits and groups, offering diverse experiences. However, this over-

TABLE I
PARTICIPANT DETAILS

ID	Company	Current role	Time on current project	Previous experience	Team Structure
P1	A	Product Owner	1 year	Working with architecture of the platform	150 people in functional safety, 2 platform, and 10 application agile release trains.
P2	A	Software Developer and Software Architect	5 years	10 years	Typically works in Scrum teams with 7–9 members. Current team has three core developers.
P3	A	Platform Architect (informally), formally titled Expert System Design Engineer	13 years	16 years in architectural role	Large project involving 500-700 people, organized into various streams and sub-streams. Individual teams typically consist of 8–10 people.
P4	A	System Architect	12 years	Software Developer	Estimated typical team size is 5 to 10 people. Currently involved in around 10 different teams working on various projects, each consisting of 5 to 10 members.
P5	A	Expert Function System Design Developer	8 years	No experience before that	13 people in the team. Collaboration with other teams varies depending on the project, typically involving 3 to 5 teams or more.
P6	B	Software Architect	2.5 years	Architect for approx. 6 years	7 to 10 people per team
P7	B	Software Architect	2 years	Software Architect for 16 years	Several teams spread all over the world. Did not mention the exact number
P8	C	Software Developer and Software Architect	6-7 months	-	Collaboration between teams in India and Europe of 7 to 10 people
P9	C	System Engineer	11-12 years	Software Developer	Scrum team, 7 people per team and 10 different teams. But in this project roles are not assigned
P10	C	Embedded software developer + extra role as architect	7 years	Developer	3 people in current team, no idea about total number of teams
P11	C	Product Owner / Architect	2 years	-	12 people per team
P12	D	Electronic System Architect	1.5 year	Application Engineer	10 people per team, total teams more than 10

'-' means that the participant did not mention any previous experience

representation remains a limitation to keep in mind when interpreting the results.

III. RESEARCH FINDINGS

This section presents the findings derived from the analysis of the interview data, addressing the research questions central to this study. The first section provides a detailed overview of current industry practices for documenting and communicating architectural changes in software-intensive systems, and offers insights into common approaches and methods (RQ1). The subsequent section highlights the challenges reported by interview participants in effectively documenting and communicating architectural changes within and across different teams (RQ2).

A. State of Practice

In documenting architectural changes, it was observed that informal practices, such as descriptive text, basic sketches, or block diagrams, are predominantly used due to their simplicity and ease of application. These methods are often favored for their flexibility in quickly capturing and communicating changes. However, for larger projects with strict compliance requirements, such as those mandated by standards like ISO26262, companies are compelled to adopt more formal techniques, including Unified Modeling Language (UML) and Systems Modeling Language (SysML), to ensure adherence to regulatory standards. The major reason that participants highlighted for the preference for informal documentation practices is the ease of communicating design and architectural changes to team members, particularly cross-functional teams. The participants highlighted several tools, including draw.io, Enterprise Architect, and Confluence, commonly used for architectural documentation across different teams within the same organization.

Architectural changes within teams are generally communicated through informal verbal exchanges, in-person or virtual meetings, and shared collaboration tools such as Confluence and Microsoft Teams. For cross-team communication, in-person or virtual meetings are also the primary coordination methods. However, the way these meetings are conducted varies across organizations and even among teams within the same organization.

Most of these processes are informal, but in some cases, formal review meetings are held, such as those conducted by a Product Change Review Board or a Technical Ownership Group. Participants from one company noted that these formal reviews often occur too late in the decision-making process, after changes have already been finalized. Additionally, they highlighted that meetings with large attendance (50–60 participants) tend to become more informational than consultative, limiting meaningful feedback. To address this, participants from another company shared that such meetings are attended by one representative per team, who is responsible for relaying

the discussed information back to their respective teams.

Several participants from a single company shared that architectural practices in their project, particularly in their team, often revolve around addressing customer requests, which are communicated through stakeholders or product owners. These requests are documented in a "solution document" which outlines the desired feature, stakeholder requirements, and system requirements. This document serves as a collaborative tool where team members contribute their inputs for example details about specific interfaces to ensure alignment across various teams, such as developers, cloud engineers, and testers. This solution document does not contain details about implementation and design. It serves as a living document during development but is not maintained after implementation. Instead, its contents are transferred into other tools that manage the interfaces between different teams, to preserve documentation, leading to information being fragmented across multiple platforms. The participant expressed a preference for a centralized repository where comprehensive feature descriptions, beyond high-level summaries, could be consistently maintained. Moreover, to some extent, they also use the C4 model [4] for high-level documentation of their architecture, primarily as a reference architecture. However, the use of UML modeling semantics within their C4 model is quite basic, offering only a lightweight form of formalism. One participant said the following about C4 model: *"To me C4 model makes sense that you have four different levels, you have the very high level and then slightly lower levels and then all down to the code. If you have that, then you can use that model to communicate with pretty much everyone from the system architects to the implementation to I guess even the users of the whatever you had."*

Overall, communication practices for architectural changes vary significantly depending on the project and the nature of the changes, highlighting the lack of a standardized approach across teams. Changes affecting interfaces are often addressed informally and on a case-by-case basis, which can lead to inconsistencies.

B. Challenges

This section outlines the challenges that the software practitioners who participated in this study face towards effectively documenting and communicating architectural changes, both within individual teams and across multiple teams in the industry. Based on the challenges identified by the participants, we have categorized them into three distinct groups. These categories, along with the number of companies that reported each challenge, are presented in Table II.

1) Knowledge Management and Sharing Challenges: The participants reported that it is *difficult to transfer knowledge between different domains* due to differences in terminology and tools. For instance, SysML (Systems Modeling Language) is commonly used in domains like vehicle development for modeling complex system architectures. However it cannot be utilized in the same way, or at all, in domains like cloud software development, where lightweight tools or code-based solutions, are preferred to meet the specific needs of that domain. These differences often lead to misunderstandings, misalignment of expectations, and difficulties in achieving seamless collaboration. For instance, terminology unique to one domain is not easily understood by teams in another domain, creating a communication gap. Similarly, the use of domain-specific tools can complicate knowledge sharing, as not all teams may be familiar with or have access to the same tools, further hindering the efficient transfer of architectural knowledge and practices.

It is also interesting to note that a few participants mentioned that there is usually a *lack of familiarity among staff with UML* as one participant said: *"I talk UML, if the listeners understand that language, I try to use UML, otherwise I fake it and do something similar. My age is so that UML was sort of a hot when I was young in the trade. So I talk it but the young people have never heard of it. It seems like so. So if you're like in your early 30s or late 20s, you probably haven't seen it before."*

Several participants also reported that there is a *lack of structured mechanism* like centralized knowledge management systems or shared tools for knowledge transfer and tracking changes, between different teams, which often results in inefficiencies and miscommunication. Without standardized processes or tools in place, the exchange of critical information, such as architectural changes or domain-specific insights, becomes inconsistent and fragmented. This lack of structure can lead to misunderstandings, and inconsistencies among different architectural and design documents. Furthermore, the reliance on informal methods, such as ad-hoc discussions or broadcasting of information on teams channels, fails to ensure that all team members or stakeholders receive the necessary information, ultimately affecting the alignment and coherence of collaborative efforts.

One participant reported that *"We see that and I'm doing that myself also, that in MS team channels I normally turn off a lot of notifications because I will be overwhelmed with all the notifications all the time. So I turn it off and then it's easier to forget about them. So therefore even teams channel is not easy to work with."* Another participant reported that *"If people don't check system viewer on time then they can miss important updates leading to inconsistency problems."*

The participants also shared that it can be *challenging to figure out which stakeholders need to be informed about architectural changes*. This difficulty arises due to the complexity of software-intensive systems that involve many teams, domains, and areas of expertise with varying responsibilities. One contributing factor is the ambiguity in ownership; when areas of responsibility are not well-defined, it becomes unclear who needs to know about specific tasks or decisions. Additionally, some teams may implement architectural changes without fully understanding which other teams depend on their work, or even when teams know to involve. Delays in communication or feedback can create further challenges, as concerns or issues may surface after decisions have already been finalized and work has progressed. Without a clear process or a list of

TABLE II
CHALLENGES REPORTED BY PARTICIPANTS

Categories	Challenges	No. of Companies
Knowledge Management and Sharing	Knowledge transfer between different domains due to difference in tools and terminology	3
	Lack of familiarity with UML / SysML	2
	Lack of structured mechanism for communication between different teams	2
	Difficulty in identifying which stakeholders need to be informed about architectural changes	4
Tools and Documentation Challenges	Hard to incorporate informal diagrams into formal models	1
	Architectural documentation is stored in several different formats that leads to inconsistency	2
	Limitations in the current tools that they are using.	2
Organizational Challenges	Misalignment between the goals of system and software architects	1
	Architecture definition is different for different teams.	3
	Lack of a clear and structured organizational hierarchy for communication and decision-making processes.	2

roles and responsibilities, it's easy to miss key people who should be included. As a result, some stakeholders might be left out of important updates or discussions, leading to miscommunication, or conflicts during various activities. The lack of a structured way to identify the right stakeholders makes this issue even harder to manage.

2) Tools and Documentation Challenges: The participants pointed out that informal diagrams and sketches are often used to explain architectural changes clearly, especially in meetings or discussions. These quick drawings, using e.g. Visio or draw.io diagrams, are easy to create and help get ideas across quickly. However, trying to *include these informal visuals into more formal tools like Sparx Enterprise Architect is a challenge*. The process of converting informal diagrams into the structured formats required by formal tools can be time-consuming and may lose important context or details. Several participants mentioned that these tools don't easily support importing informal sketches, creating a gap between informal and formal documentation. This disconnect makes it difficult to ensure consistency where it is needed, as the information from the informal sketches might not be captured or might be interpreted differently when transferred into formal documentation. This gap not only makes communication harder but also adds extra work to align informal ideas with the formal documentation needed for project tracking or compliance. Participants stressed the need for tools that could better integrate informal and formal documentation, making the process smoother and more efficient.

The participants further highlighted that *architectural documentation is often stored in multiple formats*, such as MS Word documents, presentations, and informal diagrams. This diversity in formats makes it challenging to maintain alignment and consistency across the documentation. Many teams rely on manual processes to update and manage these different documents, which is often time-consuming and prone to errors. As a result, keeping all these representations of the system in sync becomes a significant effort, particularly when updates or changes are made in one format but not reflected in others. This lack of integration or standardized approach creates gaps in the documentation, making it harder for teams to have a clear, unified view of architectural changes. Participants expressed the need for more streamlined methods or tools to automate and synchronize the documentation process, reducing the burden of manual updates and ensuring consistency across all formats.

A few participants pointed out the *limitations of the tools they are currently using*, mentioning issues such as limited modeling capabilities, difficulty in navigation, and a lack of integration with other tools. One participant said about the modeling tool that has been used in their company that *"It's a requirement administration tool. They claim to be a modeling tool but it's quite a poor tool for what it claims to be. Essentially, you can only use it to document what has already been done, with the major artifacts being the requirements. Designing a system solely through requirements, however, is not an ideal approach. This highlights the gaps in our architectural thinking."* While another participant from other company acknowledges that the tool works "okay" and outputs good data, but he does not seem very enthusiastic about it. He mentions that *"the visual output (like the diagrams and documents) generated from the tool isn't very pleasant to look at."* His statement suggests that the tool while functional, might not be the best tool for communication and visualization due to its aesthetic limitations or perceived lack of usability.

These constraints make it harder to effectively capture and manage architectural changes. For instance, the tools do not offer the necessary features to model complex systems accurately, forcing teams to work around these limitations. Additionally, participants found navigating through the tools to be cumbersome, which slows down their workflow. The absence of seamless integration with other tools further complicates the process, as it requires manual data transfer and increases the risk of inconsistencies across different platforms. A few participants emphasized the need for more advanced modeling tools capable of automatically propagating design changes to multiple artifacts. They highlighted that such tools

could play a crucial role in resolving consistency issues by ensuring alignment between design artifacts. Another participant highlighted the need of a tool that helps users easily find and follow relevant processes, particularly in complex situations like mergers or component integration, without navigating a cumbersome system. This tool should provide clear guidance, making it easier to identify the appropriate steps and improve consistency. He mentioned that *"I am indicating that I have high hopes with regards to processes because that is sort of the guardrails. I am seeing that. Some parts of the organization might not be as interested in processes as other and I think maybe that's because the availability of the processes."*

3) Organizational Challenges: In the context of software-intensive systems, *misalignment between system and software architects* is a recurring challenge highlighted by participants, primarily stemming from differing priorities and communication gaps. System architects often focus on high-level goals, viewing architecture as a conceptual arrangement of boxes and arrows, which can seem flexible and easy to change. In contrast, software architects and developers prioritize quality concerns such as testability, maintainability, and the real-world impact of changes on the codebase. This difference in focus can create friction. For example, a system architect might modify the connection (an arrow) between two components in a design, considering it a minor change. However, for the software architect, this could translate to significant rework, such as revising dependencies or testing, potentially requiring months of effort. Ultimately, the key issue remains ensuring effective communication and collaboration as a participant mentioned that *"With the introduction of agile in this industry, it felt like we removed software architects, it sort of disappeared and wasn't a role here for a few years. I think that meant that the leverage shifted towards the system architects and that wasn't super good. To me, the main problem is communication with the overall architects, with the actual software architects and understand how changes actually in what do they induce in the code base and how do they influence testability and all that stuff that I optimize towards."* System architects must recognize the downstream effects of their decisions on implementation, while software architects need a channel to convey these impacts effectively, creating a shared understanding across roles.

Three participants from three different companies highlighted the fact that the *different parts of the organization have different definitions of what a software architecture is.* In some cases, the responsibilities of architects are not clearly defined which leads to the lack of consistent architectural thinking across organization that ultimately reflects in the architectural documentation and hence causes inconsistency issues.

Another key challenge identified was the *lack of a clearer, structured organizational hierarchy* to improve communication and decision-making processes within large organizations. Without a well-defined structure, coordination becomes difficult, leading to inefficiencies and delays in addressing architectural changes.

IV. Discussion

In this section, we discuss the implications and lessons learned from this study, for both software practitioners and software engineering researchers.

Our findings highlight several critical challenges that practitioners face in the context of software-intensive systems, as shown in Table II. Three of these challenges were reported by the highest number of participating companies, making them particularly significant.

1) **Difficulty in identifying which stakeholders need to be informed about architectural changes:** As emerging technologies in software-intensive systems increasingly require cross-domain collaboration, we observed that the interviewed practitioners often prefer informal diagrams and communication mechanisms over formal methods to communicate architectural and design changes within and across teams. This preference is due to informal methods being simpler and more effective for conveying information. Regarding cross-team communication, practitioners rely on shared software interfaces, updating these interfaces whenever changes occur. These modifications are communicated to other teams through platforms such as Microsoft Teams channels. However, with a large number of teams, some geographically distributed, practitioners face difficulties identifying the relevant stakeholders to communicate changes. This challenge risks critical information not reaching all necessary parties, leading to confusion or misalignment during change implementation.
2) **Varying definitions of architecture across different teams:** It was also noted that all participating companies maintain a basic, high-level architecture, usually referred to as reference architecture, which is rarely modified. On top of this core framework, different teams independently build their own architectural or design models, often unstructured and informal in nature. The approach to architecture and design varies significantly across teams, reflecting a flexible yet sometimes disjointed process. While such methods allow teams to adapt to their specific project needs, this variability leads to inconsistent documentation and communication breakdowns. Misalignment in how architecture is defined across teams undermines the overall coherence of the project, complicating collaboration and documentation efforts.
3) **Knowledge transfer between different domains, arising from differences in tools and terminology:** This challenge was reported by several participants from all participating companies. In software-intensive systems, teams often operate within different domains, each using its own specialized tools and terminology. To facilitate collaboration, practitioners rely on shared interfaces that define the incoming and outgoing data points. However, beyond these interfaces, effective communication is challenging due to the lack of a common language or

tools across domains. This fragmentation leads to miscommunication and inconsistencies when architects and engineers from various domains attempt to collaborate.

Overall, reliance on informal documentation, such as basic diagrams and text, alongside informal communication channels, creates significant challenges, leading to inconsistent documentation. To address these issues and maintain consistency, practitioners currently employ manual processes such as manual review processes, version locking, assigning personnel to oversee document updates, handling consistency at the unit level, and maintaining an appropriate level of abstraction to minimize frequent updates. The individual making the changes is solely responsible for document updates, with no automated consistency checks, nor a well-defined manual processes in place to ensure accuracy.

When we inquired about ideal solutions that the interviewees would imagine to address these challenges, several interesting ideas were shared. Participants emphasized the need for a clearer, structured hierarchy to improve communication and decision-making processes in large organizations. They also suggested developing better strategies for communicating architectural changes to all involved teams, including those no longer active on the project, to address unexpected issues effectively. Another proposed solution was the creation of a core metamodel that could be translated across different domains, enabling improved traceability between them. Additionally, participants highlighted the potential of AI tools, such as ChatGPT or Copilot, to generate documentation for open-source software, thereby reducing the administrative burden on practitioners.

Based on the findings of our study, we present the following implications for both industry practitioners and researchers.

- There is a clear need for more flexible documentation methods that can accommodate fast changing requirements of agile development practices while also meeting the regulatory requirements where required.
- Enhancing cross-domain communication and knowledge sharing should be prioritized, possibly through the development of common terminologies or translation mechanisms.
- Research into effective communication methods for coordinating architectural changes across multiple teams, departments, or organizations involved in large and complex software systems could provide significant value to the field. Such methods would help ensure that architectural changes are clearly communicated, properly understood, and consistently implemented across diverse and distributed groups.
- The development of integrated and shared documentation tools across multiple teams that can centralize information while maintaining flexibility could help overcome many of the identified challenges.
- Further investigation into lightweight tools and well-defined processes and their applicability in real-world scenarios will provide valuable insights to improve architectural documentation practices.

In conclusion, while current practices for documenting and communicating architectural changes in software-intensive systems have demonstrated adaptability to project-specific needs, significant opportunities remain to improve standardization, cross-domain collaboration, and tool integration. Future research should focus on addressing these challenges to enhance the overall effectiveness of architectural change management in complex software systems.

V. Related Work

We believe that understanding current industry practices and challenges in documenting and communicating software architecture within software-intensive systems is a crucial step toward developing effective architectural communication strategies. Our work is closely related to the study by Wan, Zhang, and Xia [5], which explores a wide range of challenges in software architecture practices, spanning various stages of the development life-cycle (requirements, design, construction, testing, and maintenance). It focuses on architectural styles, documentation, analysis, evaluation, conformance, and refactoring challenges. Our study narrows the focus to communication and documentation challenges during software architecture and design activities, particularly in the context of software-intensive systems, providing a more targeted exploration on these aspects.

Another study by Ivanov [2] aligns closely with our study as both emphasize the critical role of software architecture documentation in creating system understanding, stakeholder collaboration, and effective decision-making. While our study focuses on the challenges and practices specific to software-intensive systems, such as cross-domain collaboration and informal communication, this paper provides general guidelines and tools for improving documentation quality. The insights complement our findings by highlighting strategies to address inconsistencies, maintain up-to-date documentation, and improve cross-team communication, reinforcing the need for adaptive and structured documentation approaches in dynamic development environments.

In another study [6], Babar highlights the tension between agile principles and architectural needs, noting the perception among some agile practitioners that architectural processes are overly formal, while architecture proponents emphasize the importance of sound practices. Through an empirical study involving practitioners experienced in both agile and plan-driven approaches, the study investigates how agile adoption impacts architecture-related practices in large-scale software-intensive systems and explores solutions for bridging the gap between agile and architecture centric approaches. A challenge identified in common with our work is the use of informal architectural documentation due to the introduction of agile practices.

Nahar et al. [7] conducted an interview study focusing on collaboration challenges between data scientists and software engineers during the development of ML-enabled systems,

highlighting the critical role of effective communication, documentation, and process management. Although this study focuses on challenges specific to ML-enabled systems, its broader themes of interdisciplinary collaboration, interface documentation, and organizational practices offer valuable insights. These findings are particularly relevant to understanding and addressing communication barriers in software-intensive systems, which are central to our study.

Ovaska et al. [8] highlight the critical role of software architecture as a coordination mechanism in multi-site development projects, emphasizing the importance of maintaining a shared understanding of architectural decisions and managing interdependencies across distributed teams. The findings regarding the role of architectural documentation, proactive communication, and shared project contexts align closely with our investigation into best practices for architectural documentation and communication. These insights provide a foundation for understanding how effective documentation and communication can mitigate barriers in complex software development environments.

In another study, Hadar et al. [9] focus on adapting architectural documentation to align with agile principles by proposing a leaner and more concise documentation approach. The study highlights the complexities of maintaining up-to-date and effective documentation, particularly in agile and cross-domain collaborative settings. The proposed solution of a lean, abstract specification in this paper complements our findings on the need for more structured and light-weight documentation strategies to enhance collaboration, reduce miscommunication, and maintain consistency in architectural documentation.

Prause and Durdik [10] presents the results of structured interviews focusing on architectural design and documentation in agile development. The experts interviewed acknowledged the challenges associated with both, emphasizing the need for improvement. The paper offers an analysis of these challenges and explores their origins, proposing "reputation" as a strategy to enhance documentation and establish a clearer architectural design. An evaluation of the proposed solutions based on expert opinions is also included. The paper concludes by suggesting that these challenges are not exclusive to agile development and calls for further research to refine the proposed solutions, with plans for a larger study involving more industry experts.

VI. Conclusion

This study underscores the critical role of documenting and communicating architectural changes in managing software-intensive systems, which are characterized by their continuous evolution and collaborative nature. Through interviews with software and system architects from four companies developing software-intensive systems, we identified significant challenges that impact these practices, including difficulties in stakeholder identification, inconsistent architectural definitions across teams, and knowledge transfer barriers due to differing tools and terminologies.

Practitioners rely heavily on informal documentation and communication methods, which, while flexible and practical in dynamic environments, often lead to inconsistency and misalignment. Current mitigation strategies, such as manual review cycles, version control, and assigning responsibility for updates, reflect an ad hoc approach to addressing these challenges. However, these practices fall short of ensuring alignment and scalability in large projects.

Participants emphasized the need for structured solutions, such as clearer organizational hierarchies, a core metamodel that is translatable across domains, and AI-based tools to automate documentation and minimize administrative effort. Based on these findings, we derive insights that highlight the importance of flexible documentation strategies, standardized processes, and advanced tools to improve communication and ensure consistency.

We believe that the findings are of interest to both practitioners and researchers. For practitioners, the study suggests prioritizing flexible documentation strategies and cross-domain collaboration mechanisms. For researchers, the results identify several directions for future inquiry, including exploring lightweight formal methods, integrated documentation tools, and scalable communication strategies. By addressing these challenges, we can improve the documentation and communication of architectural changes, thereby supporting the effective evolution and maintenance of software-intensive systems in increasingly dynamic and distributed settings.

References

[1] L. Barolli and O. Terzo, "Complex, intelligent, and software intensive systems," 2020.

[2] M. Ivanov, "Software architecture documentation – guidelines and tools: Studying guidelines and tools for documenting software architecture effectively to facilitate communication and decision-making," *Journal of Artificial Intelligence Research and Applications*, vol. 4, no. 1, pp. 82—92, May 2024.

[3] S. M. Oishi, *How to conduct in-person interviews for surveys*. Sage Publications, 2003.

[4] S. Brown, *Software architecture for developers*. LeanPub, 2013.

[5] Z. Wan, Y. Zhang, X. Xia, Y. Jiang, and D. Lo, "Software architecture in practice: Challenges and opportunities," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 1457–1469.

[6] M. A. Babar, "An exploratory study of architectural practices and challenges in using agile software development approaches," in *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*. IEEE, 2009, pp. 81–90.

[7] N. Nahar, S. Zhou, G. Lewis, and C. Kästner, "Collaboration challenges in building ML-enabled systems: Communication, documentation, engineering, and process," in *Proceedings of the 44th international conference on software engineering*, 2022, pp. 413–425.

[8] P. Ovaska, M. Rossi, and P. Marttiin, "Architecture as a coordination tool in multi-site software development," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 233–247, 2003.

[9] I. Hadar, S. Sherman, E. Hadar, and J. J. Harrison, "Less is more: Architecture documentation for agile development," in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 121–124.

[10] C. R. Prause and Z. Durdik, "Architectural design and documentation: Waste in agile development?" in *2012 international conference on software and system process (icssp)*. IEEE, 2012, pp. 130–134.