Bridging the Gap: Follow-up Strategies for Effective Software Architecture Implementation

Abdullah A H Alzahrani [1†]
aahzahrani@uqu.edu.sa
Engineering and Computers College – Alqunfuda,
Umm Al Qura University, Makkah, P.O.Box: 715 Saudi Arabia

Abstract
Software architecture are High-level design decisions shaping a software system's components, structure, and interactions. It can be a blueprint for development, evolution, and ongoing maintenance. This research investigates the communication practices employed by software architects and developers to ensure adherence to the designed software architecture. It explores the factors influencing the selection of follow-up methods and the impact of follow-up frequency on successful implementation. Findings reveal that formalized follow-up procedures are not yet a ubiquitous element within the software development lifecycle. While electronic communication, particularly email, appears to be the preferred method for both architects and developers, physical and online meetings are utilized less frequently. Interestingly, the study suggests a potential confidence gap, with architects expressing concerns about developers' ability to faithfully implement the architecture. This may lead to architects providing additional clarification. Conversely, while most developers reported confidence in their software knowledge, overly detailed architecture documentation may pose challenges, highlighting the need for architects to consider alternative communication strategies. A key limitation of this study is the sample size, restricting the generalizability of the conclusions. However, the research offers valuable preliminary insights into the communication practices employed for architecture implementation, paving the way for further investigation with a larger and more diverse participant pool.

Keywords:
software architecture, software styles, software maintenance, software development, software engineering.

1. Introduction

Software architecture represents the high-level design decisions of a software system, typically made during the early stages of the development lifecycle [1]. However, validating the successful implementation of the intended architecture within the source code can be a challenging and complex process.

Modern software development practices utilize various follow-up methods to bridge this gap, including physical meetings, online meetings, and social media communication. Email remains a traditional and important follow-up channel as well. Despite these diverse methods, ensuring accurate software architecture implementation remains a persistent challenge. Effective collaboration between software architects and developers is crucial for achieving this goal, and follow-up practices can play a significant role in facilitating communication and ensuring alignment.

This research aims to investigate the follow-up methods employed by software developers and architects to ensure proper software architecture implementation. The study will explore the factors influencing the choice of follow-up methods and examine the impact of follow-up frequency.

This paper is structured as follows. The first section provides background information on software development and software architecture, followed by a review of relevant existing research. The second section details the research methodology and research questions. The third section presents and discusses the research findings. Finally, the concluding section summarizes the key findings, discusses limitations of the study, and suggests avenues for future research.

2. Related work and background

This section reviews existing literature relevant to the research topic of follow-up practices in ensuring software architecture implementation.

A) Software Development and Architecture

Software development methodologies, such as Agile, involve a Software Development Life Cycle (SDLC) that emphasizes meeting client requirements and quality assurance (QA) practices [2, 3, 4, 5, 6, 7,

Manuscript received July 5, 2024
Manuscript revised July 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.7.1

8]. Software architects and developers share responsibility for ensuring proper software architecture implementation, which can be challenging due to factors like architecture complexity and developer knowledge [9, 10, 11, 12].

B) Communication in Software Development

Effective collaborative communication between developers and other stakeholders is crucial for software development processes [13]. Various communication channels, such as email and meetings, are employed for collaboration and follow-up purposes. However, communication can be hindered by factors like distance or multiple project teams, despite the emphasis on close collaboration within Agile methodologies [14, 15].

C) Previous Research

Several studies have explored aspects of software architecture and communication. Wiese et al. [16] investigated software architects' practices for identifying, managing, and communicating architecture mistakes, highlighting the importance of standardized communication for improvement purposes. Muccini et al. [17] examined the evolving role of software architects in developing machine learning systems, emphasizing the impact of communication and collaboration on overall architecture. Malavolta et al. [18] analyzed architectural languages (ALs) and user needs through a survey, finding that researcher-proposed ALs lack features like communication support desired by professionals.

Bailey et al. [19] investigated developer responses to user feedback through reviews in over 1,700 applications, demonstrating the need for communication channels between developers and users. Ozkaya et al. [20] studied developer understanding of software architecture using the Unified Modeling Language (UML), revealing preferences for specific UML diagrams in visualizing different architectural aspects. Haoues et al. [21] proposed a guide for selecting appropriate software architecture styles, drawing on prior research and expert validation.

D) Software Architecture and Code

da S. Carvalho et al. [22] explored the relationship between software architecture and code smells, identifying potential associations between specific code smells and certain architectural styles. Garcia et al. [23] introduced the "SAIN" framework for reverse engineering software architecture from source code, highlighting its potential to reduce maintenance costs but acknowledging maturity limitations.

Nahar et al. [24] explored collaboration challenges between software engineers and data scientists in developing machine learning systems, identifying miscommunication as a key issue. Savarimuthu et al. [25] investigated the impact of communication between users and developers on mobile app development using review data, emphasizing the need for further research in this area. Chen et al. [26] proposed a framework for developers to analyze user feedback and improve mobile apps, further highlighting the importance of communication between users and developers.

E) Communication and Software Maintenance

Several studies by various researchers (e.g., [27, 28, 29, 30]) have explored the use of communication methods like emails, feedback reviews, and social media for software maintenance and improvement purposes, often employing natural language processing and machine learning techniques. This existing research underscores the significance of investigating communication methods in software development.

F) Gap in Existing Research

While extensive research has addressed topics like architecture selection, architectural languages, and developer understanding of architecture, the current study aims to fill a gap in existing knowledge. Specifically, this research investigates the choice and impact of follow-up practices and communication methods employed to ensure proper implementation of software architecture. While some studies (e.g., [31, 32, 33]) acknowledge the importance of collaborative

communication between software architects and developers, the nature of this communication, particularly regarding follow-up practices, has not been sufficiently explored.

3. Research Design

This section outlines the research design employed to investigate the use of follow-up practices in ensuring software architecture implementation.

A) Research Questions

Stemming from the research motivation, this study addresses two central research questions (RQs) focused on follow-up practices:

RQ 1: What factors influence the choice of follow-up methods for ensuring proper software architecture implementation?
RQ 2: Is following up a standard practice for guaranteeing correct software architecture implementation?

B) Participants

To answer these questions, the study involved two participant groups: software developers and software architects. Aiming for generalizability, the participants were anonymous and not affiliated with a specific workplace. Additionally, participant experience levels were varied to capture a broader perspective.

C) Methodology

Surveys were chosen as the primary methodology due to their efficiency in collecting data from a large pool of participants. The targeted participants, software architects and developers, were able to provide anonymous responses through a web-based survey tool.

D) Sample

The survey, distributed via email and text messages, was sent to 200 software developers and architects. A total of 75 participants responded. Figure 1 presents the distribution of participants based on job title and gender. As shown in Figure 1, 59% of participants identified as software developers, and the majority (88%) were male.

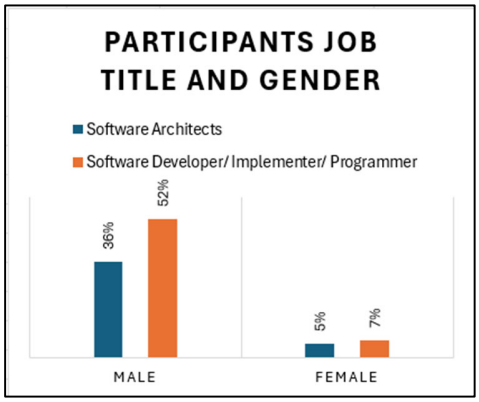

Figure 1: Distribution of participants.

E) Survey Instrument

The survey design consisted of four sections:

1. **Consent**: Obtained informed consent from all participants.
2. **General Information**: Collected demographic data, including job title.
3. **Main Questions**: This section presented relevant questions tailored to the participant's job title (software architect or developer) based on their response in Section 2, creating an interactive format.
4. **Open Question**: Allowed participants to provide additional insights.

F) Measurement Instruments

The majority of questions employed a Likert scale ranging from 0 (never) to 4 (always) to gauge frequency or agreement. One question regarding developer knowledge of software architecture utilized a similar but distinct scale, ranging from 0 (none) to 4 (excellent). Additionally, a single closed-ended question with yes or no answer options inquired about participant adherence to an Agile development model. Figure 2 illustrates that 83% of participants reported working within an Agile environment.

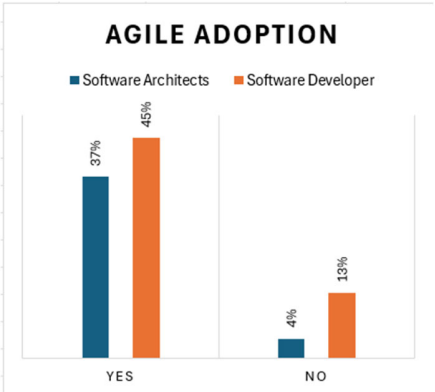

Figure 2: participants beliefs of Agile adoption in workplace.

G) Survey Development and Administration

The survey was designed and implemented using Google Forms. The survey was then disseminated to the target population through email and direct text messages. A simplified version of the survey is provided in Appendix A.

4. Research Findings

This section presents the key findings of the study, categorized into three subsections (A, B, and C) based on the survey instrument's structure. Each subsection delves into the responses received from software architects and developers regarding follow-up practices, architect perspectives on implementation fidelity, and developer knowledge and resource access.

A. Frequency of Follow-up Methods

This subsection examines the frequency with which software architects and developers utilize various follow-up methods (e.g., in-person meetings, email, social media) to ensure proper software architecture implementation.

Overall Follow-up Practices:

Data presented in Figure 3 suggests that 42% of software architects consistently (always or often) follow up with developers regarding architecture implementation. While a similar proportion (46%) of software developers reported using some form of follow-up, the findings from Figure 4 indicate that "sometimes" was the most common response for both groups regarding initiating follow-up requests. This inconsistency suggests that regular follow-up is not a standard practice within the development process. There might be an absence of established follow-up policies, or a reluctance among some developers or architects to engage in frequent follow-up. Further investigation is needed to explore this aspect.

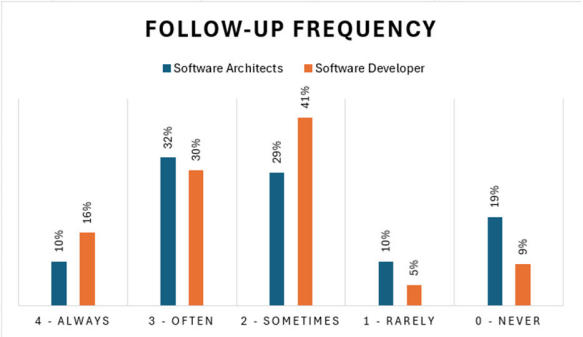

Figure 3: Frequency of Follow-up.

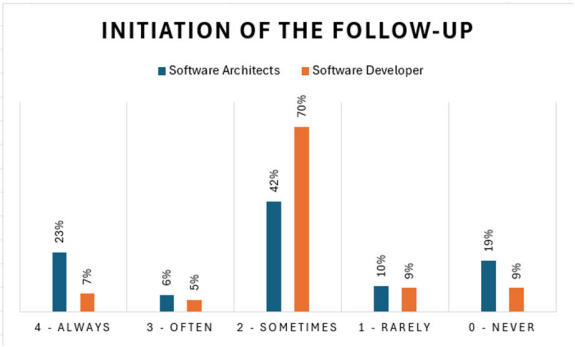

Figure 4: Frequency of Follow-up Requests.

Preferred Follow-up Methods:

Figures 5-8 illustrate the preferred methods for follow-up communication between architects and developers. Both groups utilize email frequently (Figures 5 & 8). Interestingly, software developers appear to favor online meetings (Figure 6) over in-person meetings (Figure 5), while the opposite is true for software architects.

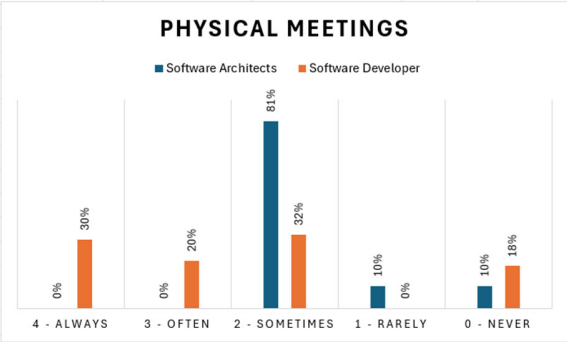

Figure 5: Frequency of Follow-up in Person.

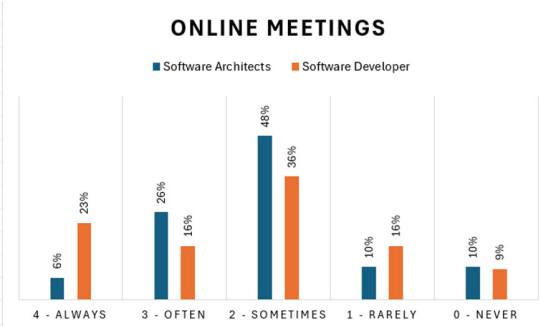

Figure 6: Frequency of Follow-up via Online meetings.

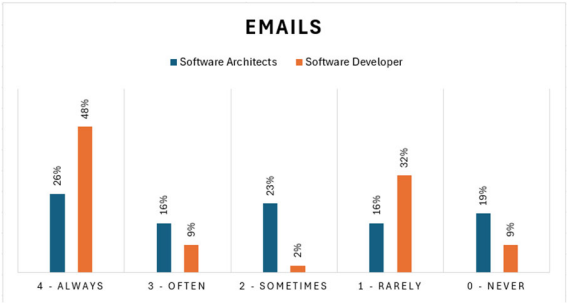

Figure 7: Frequency of Follow-up via Emails

Social media usage for follow-up is also notable, with a significant portion of both groups using it frequently (Figure 8). It is important to acknowledge that experience might influence follow-up method selection (Figures 9 & 10). However, with limited data, further research is needed to confirm this relationship.

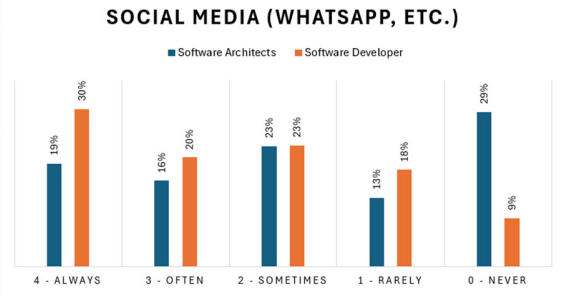

Figure 8: Frequency of Follow-up via Social media.

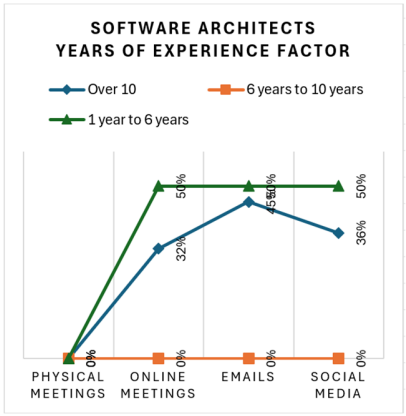

Figure 9: Software architects' experience and the frequent follow-up means

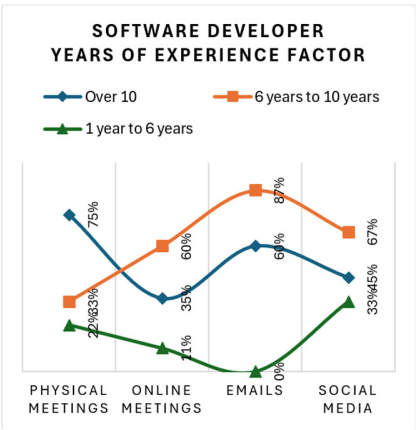

Figure 10: Software developers' experience and the frequent follow-up means

These findings highlight the need for a more standardized approach to follow-up practices within the development process. Establishing clear guidelines and considering communication preferences of both architects and developers could lead to more effective and efficient follow-up strategies.

Formalized follow-up procedures for ensuring adherence to software architecture are not yet widely adopted. Further research could explore potential benefits and barriers to implementing such procedures.

B. Software Architects' Perspectives

This subsection focuses on the opinions of software architects concerning the frequency of correct software architecture implementation by developers and the need for further explanation (Figures 11 & 12). As shown in Figure 11, only 48%

of software architects believe developers always or often implement the architecture correctly. This finding suggests potential concerns about developer understanding or adherence to the architecture. There could be several explanations for this. The initial documentation might be unclear or lack sufficient detail, leading to misinterpretations by developers. Alternatively, developers might not have received adequate training or knowledge sharing opportunities regarding the specific software architecture being implemented.

Figure 12 further supports this notion, revealing that 42% of software architects frequently need to explain the software architecture to developers. This finding suggests potential communication gaps between architects and developers. The frequent need for clarification could be a consequence of the aforementioned issues with documentation clarity or developer knowledge. Alternatively, it might indicate a lack of ongoing communication channels where developers can readily ask questions or seek clarification during the development process.

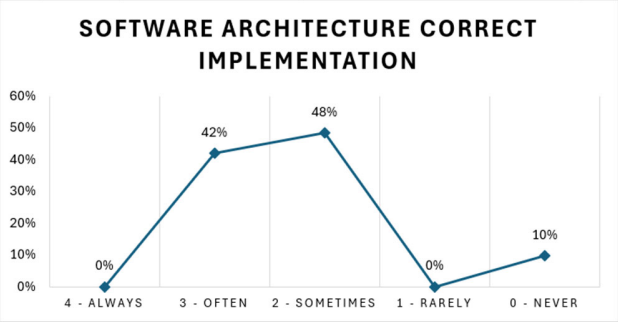

Figure 11: Software Architects opinions on correct implementation of software architecture.

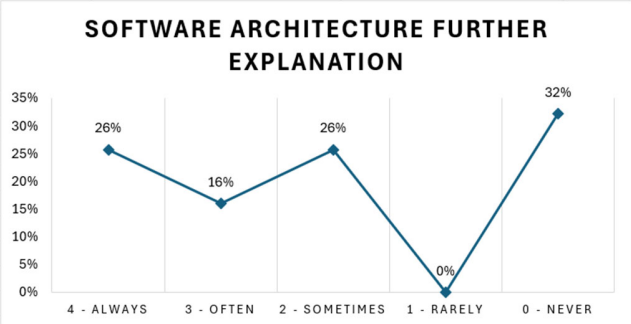

Figure 12: opinions Frequency of need for software architecture explanations to developers.

C. Software Developers' Perspectives

This subsection examines software developers' self-reported knowledge of software architecture and the frequency with which they receive detailed documentation (Figures 13 & 14). Figure 13 illustrates the distribution of self-reported knowledge levels among developers. While 57% report high or excellent knowledge, 16% indicate low or no knowledge. This variation in self-reported knowledge could be attributed to factors such as experience level, prior training, or the complexity of the specific software architecture itself. Developers with less experience or those working on a particularly complex architecture might feel less confident in their understanding compared to their more experienced colleagues.

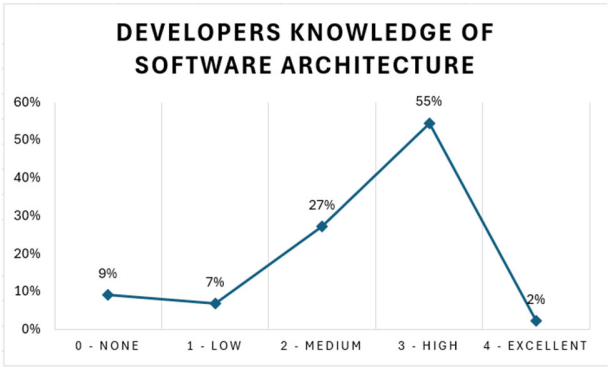

Figure 13: Software Developers Rate of Knowledge in Software Architecture

The data in Figure 14 reveals a disparity in receiving detailed architecture documentation. While 37% of developers receive it frequently, 39% report rarely or never receiving it. This uneven distribution could contribute to the inconsistencies observed in follow-up practices and potential knowledge gaps among developers. Unequal access to detailed documentation could lead to confusion and hinder developers' ability to correctly implement the software architecture. Furthermore, the lack of readily available documentation might necessitate more frequent follow-up from architects to provide necessary clarification, potentially contributing to the inconsistencies observed in follow-up practices.

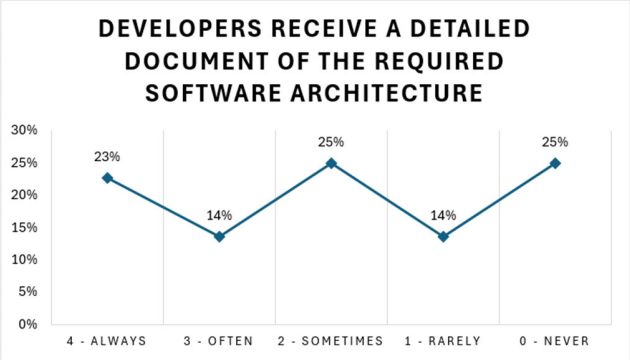

Figure 14: Frequency of providing software developers with detailed document of the required software architecture

The findings in this section suggest a need for further investigation into communication gaps and knowledge disparities between software architects and developers. Addressing these issues through improved documentation, regular communication channels, and knowledge-sharing initiatives could contribute to a more efficient and effective software development process.

.

5. Conclusion

This study investigated various follow-up methods employed by software architects and developers to ensure the implementation of software architecture. It explored factors influencing the choice of follow-up methods and the impact of follow-up frequency.

The findings revealed that following up on architecture implementation is not a routine practice in software development. While software architects initiate follow-up more frequently, developers sometimes take the initiative as well. Interestingly, emails emerged as the preferred follow-up method for both parties. Physical meetings were less favored by architects, while online meetings were less appealing than social media platforms. However, developers seemed to prefer meetings for follow-up discussions.

The research also highlighted a potential lack of confidence among software architects, who expressed concerns about developers' adherence to the architecture. This may lead them to provide additional explanations. Conversely, while most developers reported confidence in their software knowledge, detailed architecture documentation could present challenges, suggesting a need for adaptation by architects.

A key limitation of this study is the sample size. With only 75 participants, generalizing the conclusions may be difficult. Nevertheless, the research provides valuable initial insights into the follow-up practices between software architects and developers, laying the groundwork for further exploration.

Acknowledgment

This research journey would not have been possible without the unwavering support of a remarkable network. I extend my deepest gratitude to my esteemed family and friends whose constant encouragement, patience, and unwavering belief in me fueled my determination throughout this process. I am also incredibly thankful to Umm Al Qura University for providing the exceptional environment and invaluable resources that facilitated my exploration of software development within the vast field of software engineering. Their guidance proved instrumental in shaping this project. To all of you, my heartfelt thanks.

References

[1] M. C. Oussalah, *Software architecture 1*. Wiley Online Library, 2023. Accessed: Jun. 06, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118930960

[2] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, "Agile software development: Methodologies and trends.," Int. J. Interact. Mob. Technol., vol. 14, no. 11, 2020, Accessed: Jun. 06, 2024. [Online]. Available: https://pdfs.semanticscholar.org/2fef/154748093288894dbd0b98db1b9b54731c71.pdf

[3] E. Laaraib et al., "A Methodology for Incorporating Quality Assurance Practices during Software Development Life Cycle," Int J, vol. 10, pp. 2296–2301, 2021.

[4] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, Agile Software Development Methods: Review and Analysis. arXiv, 2017. Accessed: Jun. 05, 2024. [Online]. Available: http://arxiv.org/abs/1709.08439

[5] V. R. Basili, "Software development: A paradigm for the future," in [1989] Proceedings of the Thirteenth Annual International Computer Software & Applications Conference, IEEE, 1989, pp. 471–485. Accessed: Jun. 05, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/65127/

[6] H. S. Bok and K. S. Raman, "Software Engineering Productivity Measurement using Function Points: A Case

Study," J. Inf. Technol., vol. 15, no. 1, pp. 79–90, 2000, doi: 10.1177/026839620001500108.

[7] I. C. S. S. E. T. Committee, IEEE Standard Glossary of Software Engineering Terminology, vol. 729. IEEE, 1983.

[8] I. Sommerville, Software Engineering, 10th edition. Boston: Pearson, 2015.

[9] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," ACM SIGSOFT Softw. Eng. Notes, vol. 17, no. 4, pp. 40–52, Oct. 1992, doi: 10.1145/141874.141884.

[10] D. Spinellis and G. Gousios, Beautiful architecture: leading thinkers reveal the hidden beauty in software design. O'Reilly Media, Inc., 2009. Accessed: Jun. 07, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=h34pwy005nYC&oi=fnd&pg=PR5&dq=beautiful+architectures&ots=YWhejCmTV2&sig=YKo_a69ks28hrobI1yry1t6Rq5g

[11] S. Herold, C. Knieke, M. Schindler, and A. Rausch, "Towards improving software architecture degradation mitigation by machine learning," in The Twelfth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2020), Nice, France, October, 26-29 2020, 2020. Accessed: Jun. 06, 2024. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1484805

[12] N. Ford, M. Richards, P. Sadalage, and Z. Dehghani, Software Architecture: The Hard Parts. O'Reilly Media, Inc., 2021. Accessed: Jun. 06, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=OX1EEAAAQBAJ&oi=fnd&pg=PP1&dq=%22Software++Architecture:++The+Hard+Parts%22&ots=eS4q_nkmUQ&sig=PAZm7Wh-F7ZqtjfcMOUypg2pPI8

[13] S. Berczuk and B. Appleton, Software configuration management patterns: effective teamwork, practical integration. Addison-Wesley Professional, 2020. Accessed: Jun. 07, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=kmfnDwAAQBAJ&oi=fnd&pg=PR11&dq=%22Software+Configuration+Management+Patterns:+Effective+Teamwork,+Practical+Integration%22&ots=cHCqEw-u2r&sig=qYtsrS1SyTSEKIjGK0797TP703o

[14] P. A. Laplante and M. Kassab, What every engineer should know about software engineering. CRC Press, 2022. Accessed: Jun. 07, 2024. [Online]. Available: https://www.taylorfrancis.com/books/mono/10.1201/9781003218647/every-engineer-know-software-engineering-phillip-laplante-mohamad-kassab

[15] A. Goldberg, "Collaborative software engineering," J. Object Technol., vol. 1, no. 1, pp. 1–19, 2002.

[16] M. Wiese, A.-F. Brand, and A. Van Hoorn, "Learning from Each Other: How Are Architectural Mistakes Communicated in Industry?," in Software Architecture, vol. 14212, B. Tekinerdogan, C. Trubiani, C. Tibermacine, P. Scandurra, and C. E. Cuesta, Eds., in Lecture Notes in Computer Science, vol. 14212. , Cham: Springer Nature Switzerland, 2023, pp. 319–336. doi: 10.1007/978-3-031-42592-9_22.

[17] H. Muccini and K. Vaidhyanathan, "Software architecture for ML-based systems: What exists and what lies ahead," in 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN), IEEE, 2021, pp. 121–128. Accessed: Jun. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9474391/

[18] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," IEEE Trans. Softw. Eng., vol. 39, no. 6, pp. 869–891, 2012.

[19] K. Bailey, M. Nagapan, and D. Dig, "Examining User-Developer Feedback Loops Facilitated by Mobile Application Stores", Accessed: Jun. 07, 2024. [Online]. Available: http://dig.cs.illinois.edu/papers/FeedbackPaper.pdf

[20] M. Ozkaya and F. Erata, "A survey on the practical use of UML for different software architecture viewpoints," Inf. Softw. Technol., vol. 121, p. 106275, 2020.

[21] M. Haoues, A. Sellami, H. Ben-Abdallah, and L. Cheikhi, "A guideline for software architecture selection based on ISO 25010 quality related characteristics," Int. J. Syst. Assur. Eng. Manag., vol. 8, no. S2, pp. 886–909, Nov. 2017, doi: 10.1007/s13198-016-0546-8.

[22] L. P. da S. Carvalho, R. Novais, and M. Mendonça, "Investigating the Relationship between Code Smell Agglomerations and Architectural Concerns: Similarities and Dissimilarities from Distributed, Service-Oriented, and Mobile Systems," in Proceedings of the VII Brazilian Symposium on Software Components, Architectures, and Reuse, in SBCARS '18. New York, NY, USA: Association for Computing Machinery, Sep. 2018, pp. 3–12. doi: 10.1145/3267183.3267184.

[23] J. Garcia et al., "Constructing a shared infrastructure for software architecture analysis and maintenance," in 2021 IEEE 18th International Conference on Software Architecture (ICSA), IEEE, 2021, pp. 150–161. Accessed: Jun. 06, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9426737/

[24] 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, Pittsburgh Pennsylvania: ACM, May 2022, pp. 413–425. doi: 10.1145/3510003.3510209.

[25] B. T. R. Savarimuthu, S. A. Licorish, M. Devananda, G. Greenheld, V. Dignum, and F. Dignum, "Developers' Responses to App Review Feedback – A Study of Communication Norms in App Development," in Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XIII, vol. 12298, A. Aler Tubella, S. Cranefield, C. Frantz, F. Meneguzzi, and W. Vasconcelos, Eds., in Lecture Notes in Computer Science, vol. 12298. , Cham: Springer International Publishing, 2021, pp. 57–75. doi: 10.1007/978-3-030-72376-7_4.

[26] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: mining informative reviews for developers from mobile app marketplace," in Proceedings of the 36th International Conference on Software Engineering, Hyderabad India: ACM, May 2014, pp. 767–778. doi: 10.1145/2568225.2568263.

[27] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in 2015 IEEE international conference on software maintenance and evolution (ICSME), IEEE, 2015,

pp. 281–290. Accessed: Jun. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7332474/

[28] F. Kooti, L. M. Aiello, M. Grbovic, K. Lerman, and A. Mantrach, "Evolution of Conversations in the Age of Email Overload," in Proceedings of the 24th International Conference on World Wide Web, Florence Italy: International World Wide Web Conferences Steering Committee, May 2015, pp. 603–613. doi: 10.1145/2736277.2741130.

[29] F. Kooti, H. Yang, M. Cha, K. Gummadi, and W. Mason, "The emergence of conventions in online social networks," in Proceedings of the International AAAI Conference on Web and Social Media, 2012, pp. 194–201. Accessed: Jun. 07, 2024. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14267

[30] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do twitter users say about software?," in 2016 IEEE 24th international requirements engineering conference (RE), IEEE, 2016, pp. 96–105. Accessed: Jun. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7765515/

[31] G. G. Lucassen, "Dynamics of Software Product Management & Software Architecture," Master's Thesis, 2014. Accessed: Jun. 07, 2024. [Online]. Available: https://studenttheses.uu.nl/bitstream/handle/20.500.12932/18340/Thesis%20FINAL.pdf?sequence=2

[32] O. G. Fragoso-Diaz, R. Santaolaya-Salgado, and S. De Gyves-Avila, "Web Services for Software Development: The Case of a Web Service That Composes Web Services," in 2008 The Third International Conference on Software Engineering Advances, IEEE, 2008, pp. 31–36. Accessed: Jun. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4668084/

[33] H.-K. Kim, "Diversity of Mobile Distribution Systems," Int. J. Smart Home, vol. 7, no. 3, pp. 355–364, 2013.

Appendics : Appendix A

I. Survey on opinions of architects and developers.

#	Question	Type of response
1.	What is your job title?	• Software architect • Software developer
2.	Specify your experience range?	• 1 year to 6 years • 6 years to 10 years • Over 10
3.	Do you employ Agile development methodology in your workplace?	• Yes • No
4.	Is there any sort of follow-up to ensure the implementation of software architecture?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always
5.	Do you initiate the follow-up?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always
6.	How often do you follow up using physical meetings?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always
7.	How often do you follow up using online meetings?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always
8.	How often do you follow up using emails?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always
9.	How often do you follow up using social media (WhatsApp, etc.)?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always

II. Survey on opinion of architects.

#	Question	Type of response
1.	How often developers implement the required software architecture correctly?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always
2.	How often do you need to explain software architecture to developers?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always

III. Survey on opinion of developers.

#	Question	Type of response
1.	Rate your knowledge of software architecture in general?	• 0 - None • 1 - Low • 2 - Medium • 3 - High • 4 - Excellent
2.	Do you receive a detailed document of the required software architecture?	• 0 - never • 1 - rarely • 2 - sometimes • 3 - often • 4 – always

.

Abdullah A H Alzahrani's educational journey began with a Bachelor of Science (BSc) from King Abdulaziz University (KAU) in Jeddah, Saudi Arabia, completed in 2007. He furthered his studies at the University of Essex in Colchester, UK, obtaining both a Master of Science (MSc) in 2011 and a Doctor of Philosophy (PhD) in 2016.

Dr. Alzahrani's professional career commenced at Umm Al-Qura University in Makkah, Saudi Arabia, in 2008. Since then, he has been affiliated with the Engineering and Computing College at Alqunfuda, where he currently holds the esteemed position of Associate Professor. Software engineering serves as his primary research focus.

Dr. Alzahrani's leadership qualities are evident in his past administrative roles. He served as the Vice Dean of the Engineering and Computing College at Alqunfuda from 2016 to 2019. Subsequently, from 2019 to 2020, he held the position of Vice Dean for Development and Entrepreneurship at the College of Computing in Al Lith, another branch of Umm Al-Qura University. In February 2020, his expertise was recognized with an appointment as the Vice Dean for Information Technology Deanship in Makkah, a position he held until 2023.