Architecture Design Decisions in Distributed Teams: An Assessment of Tool Support

Mahum Adil[1][a], Ilenia Fronza[1][b], Outi-Sievi-Korte[2][c] and Claus Pahl[1][d]
[1]*Free University of Bozen-Bolzano, Italy*
[2]*Tampere University, Finland*
fi

Keywords: Architecture Design Decisions (ADD), ADD Tools, Distributed Collaborative Design (DCD), Global Software Engineering (GSE).

Abstract: *Background.* Global Software Engineering (GSE) teams develop software artifacts across multiple locations. Designing and maintaining software in such a setting requires continuous collaboration to record design decisions between distributed teams. The literature presents different architecture design decision (ADD) tools to support design thinking and decision-making. However, recent studies highlight the challenges of managing ADDs in a distributed environment. *Aim.* This study aims to present a list of assessment metrics to evaluate whether existing ADD tools support distributed collaborative design (DCD) in the GSE environment. *Method.* We used Goal-Question-Metric (GQM) method and expert survey to define and validate the assessment metrics. *Result.* Six assessment metrics are presented to evaluate ADD tool support for DCD in the GSE environment. *Conclusion.* The tool analysis based on assessment metrics provides insights into the current practices used for ADD process and its conformance to the distributed environment.

1 INTRODUCTION

Since the global pandemic, many software organizations have started working in distributed teams at a larger scale (Matos and França, 2022), which requires software engineers to understand the importance of collaboration fully. In a Global Software Engineering (GSE) setting, software developers work in different geographical and temporal boundaries (Herbsleb, 2007), sometimes including different cultures. As a result, GSE was extensively adopted in unprecedented circumstances by COVID-19 (Marek et al., 2021).

The existing literature highlights the challenge of designing software architecture in distributed teams due to a need for practices to efficiently manage the distributed nature of the architecture design process (Adil et al., 2022; Sievi-Korte et al., 2019). The software architecture represents architecture design decisions (ADD) (Jansen and Bosch, 2005), which comprise software elements evolved in the software development life-cycle (Alexeeva et al., 2016). Related research presented various ADD tools to manage the design decision process (Bhat et al., 2020). However, whether these tools support collaboration in distributed teams while managing software architecture in a GSE environment is still being determined.

To address this issue, we present a list of assessment metrics to evaluate how ADD tools support distributed collaborative design (DCD). Five experienced researchers in the field of GSE research validated the definition of the assessment metrics. To comprehensively review the existing ADD tools, we used a semi-structured literature review presented by Bhat et al. (Bhat et al., 2020) to analyze ADD tools based on the assessment metrics to identify DCD support in a GSE environment. The evaluation of the existing ADD tools highlights the current gap in supporting team collaboration for managing design thinking and decision-making in a distributed environment.

The paper is structured as follows. Section 2 presents background and related work on ADD tools. Section 3 introduces the research objective, and Section 4 presents the research methodology. Section 5 describes the results, which are then discussed in Section 6. Section 7 concludes the paper and shares the future work for the research.

[a] https://orcid.org/0000-0001-6452-6085
[b] https://orcid.org/0000-0003-0224-2452
[c] https://orcid.org/0000-0002-4956-8989
[d] https://orcid.org/0000-0002-9049-212X

Adil, M., Fronza, I., Sieve-Korte, O. and Pahl, C.
Architecture Design Decisions in Distributed Teams: An Assessment of Tool Support.
DOI: 10.5220/0011848000003467
In *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) - Volume 2*, pages 65-74
ISBN: 978-989-758-648-4; ISSN: 2184-4992

2 RELATED WORK AND BACKGROUND

2.1 Related Work

During the past two decades, collaboration has become a fundamental component of the software design process (Larsson, 2003), involving several activities grouped into three phases: problem analysis, solution design, and product development (Shaw, 1995). These activities are complex and require a sound knowledge of design principles and, more importantly, their skillful application (Jolak et al., 2020). Therefore, software design in a collaborative environment largely depends on design thinking and decision practices to support knowledge and architectural planning across all teams. Rodriguez et al. presented a systematic mapping study discussing tools used in GSE (Portillo-Rodríguez et al., 2012). However, the tools categorized as *software design* focus on design diagrams instead of the design decision process. A systematic mapping study on architecture decisions discussed different aspects of architecture decisions, such as decisions based on use cases, quality attributes, or group decisions (Tofan et al., 2014). However, the approaches discussed in the study are only proposed solutions and are not validated empirically in a real-time environment. The evolution of architecture decisions and knowledge management is presented in (Capilla et al., 2016) and (Bhat et al., 2020), which address the success, shortcomings, and acceptance of existing ADD tools in software organizations.

2.2 Historical Review of ADD Tools

To analyze the evolution and the shift of focus on collaboration in the design decision-making process, we reviewed the ADD tools given in Bhat et al., (Bhat et al., 2020) semi-structured systematic mapping study, and we distinguished three generations (2006-2019). The first-generation ADD tools focus on capturing architecture elements and visualization processes. The second-generation ADD tools are solution-based research focused on existing limitations in the field. The third-generation ADD tools are advanced automated ADD practices that support the idea of the collaborative design decision process. Moreover, later sections in the study will detail how we structured our research approach to compare these tools with regard to GSE objectively.

The Architecture Design Decision Support System (ADDSS) (Capilla et al., 2006) is a research web-based tool to capture, manage, and store architecture decisions. ADDSS captures architecture decisions following an iterative software development process to store the architecture knowledge evolution for software architects to build possible solutions. However, the tool does not support any communication model to update the team regarding updates in requirements, design decisions, or architecture stored in the tool. *ADD Visualization* (Lee and Kruchten, 2008b) is a research-based visualization tool to represent design decisions to inspect the system's architecture graphically. The tool supports four views to provide a dependency list of design, design decisions visualization, design decision history, and the impact of design decisions. However, the view of the tool is solely based on the architect, which restricts any communication process within the team.

Decisionstick is based on ADD visualization to support three approaches to capture ADD, such as formal identification and elicitation, from existing project documentation and source code annotation (Lee and Kruchten, 2008a). The authors validated the feasibility of the tool with practitioners, whereas no further work was done to enhance the tool. *Decision Management tool* integrates design decisions with modeling (Unified Modeling Language - UML) tool (Konemann, 2009). The idea is to update the modeling diagrams based on the design decisions continuously; however, the tool was not validated in the real-time environment. *Ontology Driven Visualization (ODV)* supports practitioners in ADD process (De Boer et al., 2009): the authors used table and matrix representation of ADD with quality attribute to support product quality evaluation. However, the tool only shows the negative or positive effect of the quality criteria on the architectural decision with no communication process with the software quality analyst to understand the impact of the decision on the software system.

Second Generation (2010-2014). The customized ADD model-based tool *Architectural Design Decisions Management support (ADDMS)* (Chen et al., 2010) manages and organizes architecture knowledge for the team working on the project. Some extensions for the tool have been proposed (Chen and Babar, 2010); however, validation was performed in a controlled environment. The service-oriented architecture design tool *Rationale Visualization* (Shahin et al., 2010) allows the user to explore and visualize the ADDs. The tool is integrated using the Compendium tool to store scope, design drivers, and alternatives for each design decision. The tool was proposed to support collaboration; however, no communication process was used.

Traceability between software architecture and other software artifacts is another solution given in research to main software evolution. The *Language for integrated software architecture (LISA)* toolkit is a research-based solution to build a semi-automatic traceability network between ADDs and software implementation (Buchgeher and Weinreich, 2011). The tool captures traceability in three steps. First, the developer selects the design decision; then, the developer builds the ADD record to specify implementation elements. Finally, a third review serves to ensure compatibility with the software system. The web-based tool *Repertory grid technique (RGT)* (Tofan and Galster, 2014) focuses on the collaborative design decision process. The tool is available online and focuses on three aspects of architecture decisions; to capture, analyze, and make group decisions on ADDs. *Design Practice STream (DPS)* DPS (Nakakoji et al., 2011) supports real-time annotation of video meetings to help new team member(s) to join existing projects and build an understanding of existing ADDS. *Architectural Development Using Architectural Knowledge (ADUAK)* is a web-based research tool proposed by Dhaya and Zayaraz (Dhaya and Zayaraz, 2012) to explore design patterns by using architecture knowledge of the system to enhance ADD process.

To promote the importance of non-functional requirements during the architecture design process, *Architech* (Ameller et al., 2012) suggests alternative ADDs to satisfy non-functional requirements stated for the software system. The research-based tool *Architectural Design Decision Support Framework (ADvISE)* (Lytra et al., 2013) is based on the QOC (Question, Option, Criteria) approach to model ADDs, track possible issues, and build the solution. *Software Architecture Warehouse (SAW)* is a web-based research tool (Nowak and Pautasso, 2013), suitable for distributed collaborative design decision process. However, the tool does not support integration with existing architecture knowledge tools and does not comply with system and software architecture standards (e.g., ISO 40210). *Decision Architect* developed by Manteuffel et al., (Manteuffel et al., 2014) is a plug-in for Sparx System Enterprise Architect to manage design decisions and design rationale for all the developers involved in the project.

Third Generation (2015-2019). To synchronize architecture decision-related knowledge with design documentation, the Eclipse plug-in *DecDoc* (Hesse et al., 2016) helps a team to capture ADDS and collaborate on design documentation by reflecting on the decision-making process and enhance the knowledge for building solutions. Similarly, *Ontology-based recommender* (Bhat et al., 2017), based on DBpedia ontology, automatically identifies architecture elements from software design documentation and proposes a list of possible solutions to enhance the decision-making process. Since the ontology-based recommender is part of DBpedia ontology, the accuracy of the recommendation can be affected by it. *Quiver* is a web-based design decision recommender (Gopalakrishnan and Biswal, 2017) that captures and stores reference architecture knowledge and architecture artifacts to provide recommendations for design decisions. However, the tool has not been validated to understand the accuracy of its recommendations. *Evolution Visualization for Architecture (EVA)* (Nam et al., 2018) visualizes architecture evolution from multiple facets. The tool enhances team collaboration as the developers can visualize the software architecture and explore the impact of design decisions on the software product. However, empirical analysis has yet to be done to check the performance of the tool in a real-environment.

Recently, researchers shared studies to show the importance of choosing experienced developers with architectural expertise to address any possible shortcomings in software systems (Bhat et al., 2018). To support the idea, the expert recommendation system *ADeX (Amelie - Decision Explorer)* (Bhat et al., 2019) provides a list of experts based on their expertise in the field. The tool uses a bottom-up approach to automatically detect and generate specific views on architecture knowledge to support the collaborative design decision-making process.

3 RESEARCH OBJECTIVE

This study presents a list of assessment metrics to evaluate whether the existing ADD tools support DCD in the GSE environment. This evaluation will help identify the possible gap in existing tools for managing collaborative design thinking and decision-making process in distributed teams. The aim of the study is presented through the three levels of the Goal-Question-Metrics (GQM) measurement model (Basili et al., 2014) where:

1. Goal: to evaluate whether existing ADD tools support DCD in the GSE environment.
2. Question: do ADD tools support the collaborative design decision for managing software architecture in the GSE environment?
3. Metrics: six assessment metrics are defined to evaluate existing ADD tools quantitatively.

To verify the goal achievement of the study, we used the GQM template to provide a structured definition of the measurement (Table 1) by specifying the *purpose* (what object and reason of measurement), *perspective* (what aspect to evaluate and who will do it) and *context* (the environment) characteristics.

Table 1: GQM template for study.

Analyze	ADD tools support for DCD in GSE
For the purpose of	Understanding
With respect to	Tool
From the viewpoint	Researcher
In the context of	ADD process

4 RESEARCH METHOD

To identify the assessment metrics, we used the Goal-Question-Metrics (GQM) measurement model (Basili et al., 2014) (shown in Figure 1) and an expert survey to define metrics empirically evaluated by the GSE field experts. To define the assessment metrics, we used the extended taxonomy of GSE (Britto et al., 2016) and features of GSE tools presented in a systematic mapping study (Portillo-Rodríguez et al., 2012). To validate metrics, we identified GSE authors based on the excellent articles published in the field on Google Scholar. Through the search, we contacted 14 experts working in the ADD/GSE field via email. The experts received a detailed document describing the metrics definition and, for each metric, were asked to approve, disapprove, or suggest to avoid by using an online survey. If they disapproved, they were asked to suggest changes; if they suggested avoiding a metric, they could motivate their answer. We received five responses in total. All the answers given by the experts are kept anonymous, and the credentials are given in Table 2 after their shared consent.

In order to finalize the assessment metrics, all the experts' responses were discussed among the researchers of this study. All the researchers simultaneously agreed on the name and definition the experts approved; however, in case the names or definitions of the assessment metrics were disapproved or suggested to avoid, the experts shared new names or definitions to keep it coherent with the GSE taxonomy. Metrics names and definitions were changed according to the given suggestions and the researchers' common consensus.

4.1 Definition of Assessment Metrics

A detailed description of the identified and validated assessment metrics follows below. The aim is to separate the main concerns of ADD and DCD in a GSE setting using the GQM approach.

Table 2: Expert credentials.

Country	Affiliation	Academic rank	Involved in GSE research
Germany	Academic	Professor	Since 2003
Spain	Academic	Professor	Since 2004
Ireland	Academic, Applied Research	Professor	Since 2006
Spain	Academic, Software industry	Professor	Since 2006
Sweden	Academic, Software industry	Senior Researcher	Since 2012

4.1.1 M1: Synchronous Collaborative Work

The first dimension of consideration is the tool's support for team communication and collaboration. To do so, we check if the tool supports team members collaborating or communicating simultaneously on a task. Herein, we used GSE taxonomy (Britto et al., 2016) to assess whether synchronous collaborative work is supported by the tool for distributed teams. To categorize, we used binary values, i.e., *Yes* when the synchronous collaboration is supported and *No* when the synchronous collaboration is not supported.

4.1.2 M2: Software Development Methodology

The software development methodology is used to understand the practices incorporated in different sites to conduct GSE projects. The development methodology used in different sites can help to classify the effort associated with the development and coordination with each site. We used GSE taxonomy (Britto et al., 2016) to categorize the two dimensions of software development methodology used in the tool as follows:

- Plan-driven, when the tool requires detailed requirements definition against each team role, which minimizes the collaboration within the team.
- Agile, when the tool does not require detailed requirements definition and the team collaboration is increased to build the solution.

4.1.3 M3: Communication Mode

In GSE projects, the communication between distributed teams is mediated through different elec-

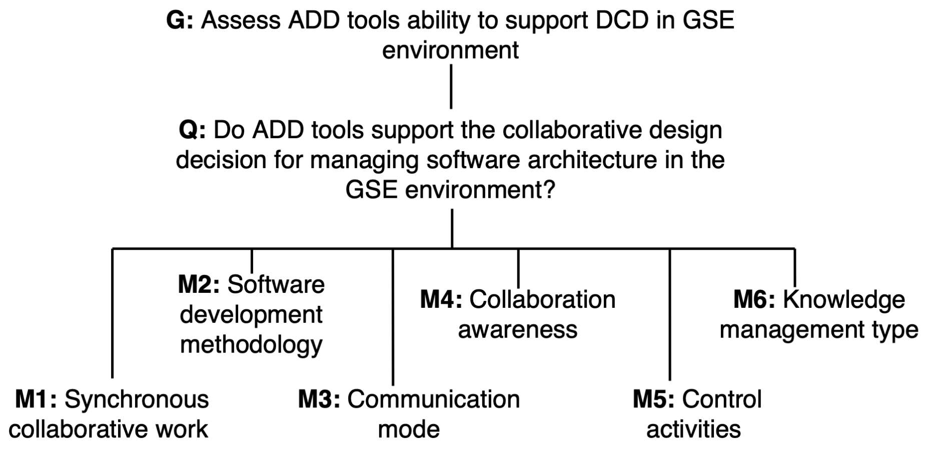
G: Assess ADD tools ability to support DCD in GSE environment
Q: Do ADD tools support the collaborative design decision for managing software architecture in the GSE environment?
M1: Synchronous collaborative work
M2: Software development methodology
M3: Communication mode
M4: Collaboration awareness
M5: Control activities
M6: Knowledge management type

Figure 1: Proposed assessment metrics using the GQM measurement (Basili et al., 2014) and expert survey.

tronic communication media. To understand the mode of communication used in tools, we used a GSE taxonomy (Britto et al., 2016), and communication features listed for GSE tools (Portillo-Rodríguez et al., 2012) to categorize three levels of synchronicity as follows:

- Low, when the tool supports asynchronous communication via, e.g., forum discussion (F), comments on software items (Cmt), or email discussion (E);
- High, when the tool supports synchronous communication via, e.g., audio/ video-conference call (C) or instant text message (T);
- Balanced when the tool supports both synchronous and asynchronous when needed.

4.1.4 M4: Collaboration Awareness

In GSE projects, it is crucial to collaborate with the other team members to participate and avoid any possible conflict (Trainer and Redmiles, 2018) during ADD process. Here we used the awareness feature listed for GSE tools (Portillo-Rodríguez et al., 2012) to define collaboration activities as follows:

- Presence awareness, when the tool allows to know where team members are geographically located (global awareness) or who is connected in a session (session awareness);
- Change awareness, when the tool provides information on who is doing a task independently (visual awareness) or supports asynchronous communication to inform any possible changes in the project (email awareness).

4.1.5 M5: Control Activities

In distributed teams, it is crucial to monitor the project's progress, track any possible issues, and continuously update the changes in the related documents. We used the control and coordination feature listed in GSE tools (Portillo-Rodríguez et al., 2012) to define control activities as:

- Issue tracking (IT), when the tool supports the identification of any possible issue in the project and team coordination accordingly.
- Version control (VC), when the tool allows keeping track of changes in the related documents and coordinating the changes between distributed teams.
- Build management (BM) when the tool supports the automated source code of the software project.

4.1.6 M6: Knowledge Management Type

In order to enhance coordination and collaboration within distributed teams, knowledge management is used to support knowledge acquisition, sharing, and distribution for the software development process. We used the knowledge management features used in GSE tools (Portillo-Rodríguez et al., 2012), covering:

- Wiki, when the tool includes a platform to share knowledge or seek consult from the rest of the team;
- Document management system (DMS), when the tool allows storing all the related documents of the software project in electronic format with proper indexing in case of information retrieval.
- Blog, when the tool supports brainstorming sessions to exchange knowledge in a distributed environment.

5 RESULTS

Table 3 summarizes the analysis of 20 ADD tools (Bhat et al., 2020) based on the assessment metrics to evaluate DCD support in GSE environment. We address the six criteria one by one in the following.

Synchronous Collaborative Work. In order to manage ADDs in a distributed environment, continuous team communication and collaboration are required for brainstorming, listing all the possible solutions, and establishing consensus on a final set of ADDs for the software system (Yang et al., 2021). In the analysis, we noticed that the support of synchronous collaborative work was captured in 5 of 20 ADD tools discussed in the third generation of tools mentioned in Section 2. However, most of these tools lack empirical validation studies to assess performance in a real-time environment.

Software Development Methodology. Agile practices are used in the software development process to support collaborative design decisions for managing software architecture in a GSE environment (Camara et al., 2020). However, our analysis noticed that no ADD tool was designed to support agile practices for software architecture and development processes.

Communication Mode. The global distance between distributed teams requires more effort to ease communication during software design thinking and decision-making. While the communication challenges in distributed teams are well discussed in literature (Hummel et al., 2013; Sievi-Korte et al., 2019), 12 of 20 tools lack support for communication mode (synchronous and asynchronous) to seek team consensus on the software architecture of the project.

Collaboration Awareness. The ADD process is a group-based activity requiring continuous coordination among teams to make distributed collaboration effective (Bosch and Bosch-Sijtsema, 2010). The analysis shows that 11 of 20 ADD tools support change awareness, which shows how tasks are allocated to each member. Furthermore, 6 of 20 ADD tools do not support collaboration activities, thereby restraining distributed team members from instantaneously increasing shared understanding of the software system.

Control Activities. To monitor the project's progress and track any possible issues in distributed teams, 10 of 20 ADD tools support issue tracking (IT). This feature is used to centralize the project repository and prioritizes issues according to severity. However, distributed teams need to have up-to-date project documentation to resolve the given issues. This allows the distributed teams to review the related documents and systematically propose solutions. In this analysis, only 6 of 20 tools support issue tracking (IT) and version control (VC) to coordinate related changes with distributed teams.

Knowledge Management Type. To support knowledge acquisition, sharing, and distribution in distributed teams, 15 of 20 ADD tools used a document management system (DMS) to store the related documents. However, these tools offer limited integration capabilities and collaboration features for distributed environments.

6 DISCUSSION

This section discusses the results of this work to address the possible barriers of DCD in a distributed environment. It is important to note that some of the findings discussed here reflect known concerns, such as communication and collaboration barriers in GSE (Noll et al., 2011) and benefits of using agile practices in GSE (Jalali and Wohlin, 2012), but are valuable here to provide a comprehensive perspective. Hence, this study acknowledges the existing gaps in the field.

The following observations concern the focus of the reviewed tools and specific features that are consequently well or insufficiently supported.

Active software process model. The results presented in Table 3 show that the plan-driven software process model is actively supported in all generations of ADD tools. We found that 16 of 20 tools follow the plan-driven method since the process activities were predefined and the project tasks were executed in a sequential task-specific manner. As a result, this feature limits the collaboration within the team to discuss any design-related issues. In order to increase collaboration in distributed teams, the agile software process should be used within ADD tools to emphasize the importance of team communication and coordination practices for successful projects (Séguin et al., 2012).

Research based tools are in majority. Regarding the current support of ADD tools for the collaborative design decision process, we found that 13 of 20 tools were prototypes developed by researchers, while two tools shared the prototype on online repositories (e.g., gitHub) and can be used without any license payment. Five of the 20 tools are commercial tools developed by practitioners as part of a software product, and the license can be obtained through payment. The results show that most tools are research-based projects, so validating their effectiveness in real-time is challenging.

Communication process is not well supported. The support for the communication process in tools

Table 3: ADD tools analysis using the proposed assessment metrics.

Name	Sync. collab. work	Software dev. methodology	Communic. mode	Collaboration awareness	Control activities	Knowledge manage. type
ADDSS	No	Plan-driven	NA	NA	VC	DMS
ADD Visualization	No	NA	NA	Change: Visual	IT, VC	DMS
Decision Stickies	No	Plan-driven	Low: F	Change: Visual	IT, VC	DMS
Decision management tool	No	Plan-driven	NA	Change: Visual	IT	Wiki
ODV	No	Plan-driven	NA	Change: Visual	BM	NA
ADDMS	No	NA	Low: F	Change: Visual, email	IT	DMS
Rationale visualization	No	Plan-driven	NA	Change: Visual	IT	DMS
LISA	No	NA	Low: F	Change: Visual	IT, BM	NA
RGT	Yes	Plan-driven	Low: Cmt	Presence: Session	IT	DMS
DPS	No	NA	NA	NA	IT	NA
ADUAK	No	Plan-driven	NA	NA	IT	DMS
ArchiTech	No	Plan-driven	NA	Change: Visual	IT	DMS
ADvISE	No	Plan-driven	NA	NA	IT, BM	DMS
SAW	Yes	Plan-driven	Low: F, Cmt	Presence: Session	IT	DMS
Decision Architect	No	Plan-driven	NA	Change: Visual	IT, VC	DMS
DecDoc	Yes	Plan-driven	Low: Cmt	Change: Visual	IT, VC	DMS
Ontology-based recommender	No	Plan-driven	Low: Cmt	Change: Visual	IT	DMS
Quiver	No	Plan-driven	NA	NA	IT	NA
EVA	Yes	Plan-driven	Low: F, Cmt	Change: Visual	IT, VC	DMS
ADeX	Yes	Plan-driven	NA	NA	IT, VC	DMS

was another focus point for the study since it encourages continuous information flow within distributed teams. (Alexeeva et al., 2016). According to our analysis, 7 out of 20 tools supported low asynchronous communication to track any issue or changes done in architecture presentation. Furthermore, no tool supported high synchronous or balanced communication process. This leads to a research gap regarding how to support the communication process to control the progress of tasks, track issues and collaborate for decision-making while working in a distributed environment.

Few tools exist for collaborative design decision. To improve architecture presentation and DCD process, 5 out of 20 tools support team collaboration to be continuously aware of the activities performed by team members. This activity is tightly coupled with the communication process used in the tool. In these five tools, the communication support consisted of forum discussions and comments on software items to inform the possible changes in the project. Furthermore, only 2 of these five tools support the session awareness to know who is currently connected within the team. This is another research gap that needs to be addressed by researchers and practitioners, as working in distributed teams is a new normal software development process.

6.1 Threats to Validity

We analyzed possible threats to validity following the guidelines provided by Wohlin et al. (Wohlin et al., 2012) to mitigate any research bias. Concerning the presented results, this study poses research limitations as we only investigated the tools mentioned in Bhat et al., (Bhat et al., 2020). Some existing tools that have not been included here may have been published after the related research. However, to limit this threat, we searched systematic mapping studies and literature reviews through research sources published after 2020 focused on the same area of research. Apart from the tools mentioned in this study, we may have excluded some commercial tools since many commercial tools are not presented through research publications.

Concerning construct validity, a possible threat relates to how the expert survey results are obtained. To avoid any threats to construct validity, we contacted the experts in the field to share their opinion on the assessment metrics through an anonymous survey to ensure data confidentiality and avoid evaluation apprehension. During the survey, we noticed that experts suggested different terms to what was given to them in the initial document. As a result, it helped to avoid any influence of the authors of this paper on the proposed assessment metrics.

External validity concerns the ability to generalize the results for a specific setting. The proposed assessment metrics can be used by other researchers in the GSE field, focusing on tool support for a broader perspective of ADD process in a distributed environment.

7 CONCLUSIONS AND FUTURE WORK

While tool support for architecture design for distributed development in a global software engineering context exists, recent years have emphasized the need to specifically consider distributed collaborative design (DCD)(Yang et al., 2021).

In this paper, we comprehensively reviewed ADD tools in terms of their support for DCD in distributed environments by building on a semi-structured literature review presented by Bhat et al. (Bhat et al., 2020) to identify the tools. To do so, we identified, defined, and validated six assessment metrics with the support of experts of GSE. Based on the assessment metrics analysis, existing ADD tools are not designed to incorporate agile practices for the software design decision process. Furthermore, among the considered tools, only five support collaborative practices to manage software architecture in a distributed environment. This shows the need for synchronous collaborative work to model ADDs and support DCD in existing tools, showing the need for additional research to prevent knowledge vaporization.

In the future, we aim to go beyond this tool review by conducting a systematic mapping study to analyze ADD approaches and tools to discover which design thinking and decision-making practices are currently used in software organizations. Based on the results, our long-term goal is to design an ADD tool that overcomes the identified limitations to enhance the collaborative design in distributed teams.

REFERENCES

Adil, M., Fronza, I., and Pahl, C. (2022). Software design and modeling practices in an online software engineering course: The learners' perspective. In *CSEDU (2)*, pages 667–674.

Alexeeva, Z., Perez-Palacin, D., and Mirandola, R. (2016). Design decision documentation: A literature overview. In *European Conference on Software Architecture*, pages 84–101. Springer.

Ameller, D., Collell, O., and Franch, X. (2012). Architech: Tool support for nfr-guided architectural decision-making. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 315–316. IEEE.

Basili, V., Trendowicz, A., Kowalczyk, M., Heidrich, J., Seaman, C., Mnch, J., and Rombach, D. (2014). *Aligning Organizations Through Measurement: The GQM+Strategies Approach.* Springer Publishing Company, Incorporated.

Bhat, M., Shumaiev, K., Biesdorf, A., Hohenstein, U., Hassel, M., and Matthes, F. (2017). An ontology-based approach for software architecture recommendations. *AMCIS 2017*.

Bhat, M., Shumaiev, K., Hohenstein, U., Biesdorf, A., and Matthes, F. (2020). The evolution of architectural decision making as a key focus area of software architecture research: A semi-systematic literature study. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 69–80.

Bhat, M., Shumaiev, K., Koch, K., Hohenstein, U., Biesdorf, A., and Matthes, F. (2018). An expert recommendation system for design decision making: Who should be involved in making a design decision? In *2018 IEEE International Conference on Software Architecture (ICSA)*, pages 85–8509. IEEE.

Bhat, M., Tinnes, C., Shumaiev, K., Biesdorf, A., Hohenstein, U., and Matthes, F. (2019). Adex: A tool for automatic curation of design decision knowledge for architectural decision recommendations. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 158–161.

Bosch, J. and Bosch-Sijtsema, P. (2010). Coordination between global agile teams: From process to architecture. In *Agility Across Time and Space*, pages 217–233. Springer.

Britto, R., Wohlin, C., and Mendes, E. (2016). An extended global software engineering taxonomy. *Journal of Software Engineering Research and Development*, 4(1):1–24.

Buchgeher, G. and Weinreich, R. (2011). Automatic tracing of decisions to architecture and implementation. In *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, pages 46–55. IEEE.

Camara, R., Alves, A., Monte, I., and Marinho, M. (2020). Agile global software development: A systematic literature review. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pages 31–40.

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., and Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, 116:191–205.

Capilla, R., Nava, F., Pérez, S., and Dueñas, J. C. (2006). A web-based tool for managing architectural design decisions. *ACM SIGSOFT software engineering notes*, 31(5):4–es.

Chen, L. and Babar, M. A. (2010). Supporting customizable architectural design decision management. In *2010 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems*, pages 232–240.

Chen, L., Babar, M. A., and Liang, H. (2010). Model-centered customizable architectural design decisions management. In *2010 21st Australian Software Engineering Conference*, pages 23–32. IEEE.

De Boer, R. C., Lago, P., Telea, A., and Van Vliet, H. (2009). Ontology-driven visualization of architectural design decisions. In *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, pages 51–60. IEEE.

Dhaya, C. and Zayaraz, G. (2012). Development of multiple architectural designs using aduak. In *2012 International Conference on Communication and Signal Processing*, pages 93–97. IEEE.

Gopalakrishnan, A. and Biswal, A. C. (2017). Quiver—an intelligent decision support system for software architecture and design. In *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pages 1286–1291. IEEE.

Herbsleb, J. D. (2007). Global software engineering: The future of socio-technical coordination. In *Future of Software Engineering (FOSE'07)*, pages 188–198. IEEE.

Hesse, T.-M., Kuehlwein, A., and Roehm, T. (2016). Decdoc: A tool for documenting design decisions collaboratively and incrementally. In *2016 1st International Workshop on Decision Making in Software ARCHitecture (MARCH)*, pages 30–37. IEEE.

Hummel, M., Rosenkranz, C., and Holten, R. (2013). The role of communication in agile systems development. *Business & Information Systems Engineering*, 5(5):343–355.

Jalali, S. and Wohlin, C. (2012). Global software engineering and agile practices: a systematic review. *Journal of software: Evolution and Process*, 24(6):643–659.

Jansen, A. and Bosch, J. (2005). Software architecture as a set of architectural design decisions. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pages 109–120. IEEE.

Jolak, R., Savary-Leblanc, M., Dalibor, M., Wortmann, A., Hebig, R., Vincur, J., Polasek, I., Le Pallec, X., Gérard, S., and Chaudron, M. R. (2020). Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication. *Empirical Software Engineering*, 25(6).

Konemann, P. (2009). Integrating decision management with uml modeling concepts and tools. In *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, pages 297–300. IEEE.

Larsson, A. (2003). Making sense of collaboration: The challenge of thinking together in global design teams. In *Proc. of the 2003 Intl. ACM SIGGROUP Conference on Supporting Group Work*.

Lee, L. and Kruchten, P. (2008a). Customizing the capture of software architectural design decisions. In *2008 Canadian Conference on Electrical and Computer Engineering*, pages 000693–000698. IEEE.

Lee, L. and Kruchten, P. (2008b). A tool to visualize architectural design decisions. In *International Conference on the Quality of Software Architectures*, pages 43–54. Springer.

Lytra, I., Tran, H., and Zdun, U. (2013). Supporting consistency between architectural design decisions and component models through reusable architectural knowledge transformations. In *European Conference on Software Architecture*, pages 224–239. Springer.

Manteuffel, C., Tofan, D., Koziolek, H., Goldschmidt, T., and Avgeriou, P. (2014). Industrial implementation of a documentation framework for architectural decisions. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 225–234. IEEE.

Marek, K., Wińska, E., and Dabrowski, W. (2021). The state of agile software development teams during the covid-19 pandemic. In *International Conference on Lean and Agile Software Development*, pages 24–39. Springer.

Matos, J. and França, C. (2022). Pandemic agility: Towards a theory on adapting to working from home. In *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, pages 66–75.

Nakakoji, K., Yamamoto, Y., Matsubara, N., and Shirai, Y. (2011). Toward unweaving streams of thought for reflection in professional software design. *IEEE software*, 29(1):34–38.

Nam, D., Lee, Y. K., and Medvidovic, N. (2018). Eva: A tool for visualizing software architectural evolution. In *Proceedings of the 40th international conference on software engineering: companion proceeedings*, pages 53–56.

Noll, J., Beecham, S., and Richardson, I. (2011). Global software development and collaboration: barriers and solutions. *ACM inroads*, 1(3):66–78.

Nowak, M. and Pautasso, C. (2013). Team situational awareness and architectural decision making with the software architecture warehouse. In *European Conference on Software Architecture*, pages 146–161. Springer.

Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., and Beecham, S. (2012). Tools used in global software engineering: A systematic mapping review. *Information and Software Technology*, 54(7):663–685.

Séguin, N., Tremblay, G., and Bagane, H. (2012). Agile principles as software engineering principles: An analysis. In *International Conference on Agile Software Development*, pages 1–15. Springer.

Shahin, M., Liang, P., and Khayyambashi, M. R. (2010). Improving understandability of architecture design through visualization of architectural design decision. In *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, pages 88–95.

Shaw, M. (1995). Making choices: A comparison of styles for software architecture. *IEEE Software*, 12(6).

Sievi-Korte, O., Richardson, I., and Beecham, S. (2019). Software architecture design in global software development: An empirical study. *Journal of Systems and Software*, 158:110400.

Tofan, D. and Galster, M. (2014). Capturing and making architectural decisions: an open source online tool. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, pages 1–4.

Tofan, D., Galster, M., Avgeriou, P., and Schuitema, W. (2014). Past and future of software architectural decisions–a systematic mapping study. *Information and Software Technology*, 56(8):850–872.

Trainer, E. H. and Redmiles, D. F. (2018). Bridging the gap between awareness and trust in globally distributed software teams. *Journal of Systems and Software*, 144:328–341.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Yang, Z., Xiang, W., You, W., and Sun, L. (2021). The influence of distributed collaboration in design processes: an analysis of design activity on information, problem, and solution. *International Journal of Technology and Design Education*, 31(3):587–609.