Software Architecture Evaluation Methods: A systematic Review

Rashid Noor and Dr. Khubaib Amjad Alam
School of Computing
FAST National University of Computer and Emerging Sciences
Islamabad, Pakistan
Mail@nu.edu.pk

Software Architecture Evaluation Methods: A Systematic Review

***Abstract*—With the continuous demand of software systems to solve the daily life problems and instant increase in the use of software systems to solve complex and complicated problems. Numerous efforts has been made to enhance the architecture designs of soft aware systems for accurate and swift results. However there has been a gap between the formal research in this area and no significant accomplishments has been made with research of design optimal software system architectures. To integrate and relate the existing efforts Systematic Literature Review (SLR) has been conducted in this study, analyzed research papers of different platforms. A formal sketch has been given after evaluating the existing work from different sources for the better understanding and to enhance the efficiency and outcomes from the efforts being made. Moreover, this research is conducted to help the technical community to achieve the best maximum results in optimal time period in designing the advance and complex software systems' architecture.**

***Index Terms*—Designing, System Architecture, Software Architecture, System Software, Systematic Literature Review**

I. Introduction

With the passage of time, complexity and complications have been increased in the software systems, as they are being utilized to solve the advance and complex problems[1]. With the continuous increase in the technologies integrated with different software are making those system hard to understand and difficult to adopt the possible changes in the future. Developing and maintaining a software with bigger complexity would result in increase of high level software system descriptions, in the management of that particular software systems these complex descriptions would help[1]. Software Architecture is the main building-block in the development of the software systems, which contains all the information related the software systems[1]. From architecture design relationship of entities, modules, addons, libraries, and packages would be understand so easily, and there we can find the possibilities of possible accommodations of new changes. To identify the potential risk, loose holes, weaknesses, and compromised scenarios Software Architecture can be helpful, moreover to understand the possible trade-offs in the software systems[2], architectural description would be beneficial. Design architecture can be elevated at the time of development of the software systems, to accommodate new changes as per the requirements of making that system optimal in use and suitable as per the requirements of the client[2]. Architecture design can also explain the mathematical equation, simulation bases test, and modelling of user needs in the system. Implementation of architecture design would be high level, different layers of architecture can be used to enhances the quality of software systems, can be implemented at the start, between or even in the final stages of the software systems.[1] Software architecture evaluation is the process of assessing the qualities of a software system's architecture in order to determine its effectiveness and suitability for use[3]. There are a number of methods that can be used to evaluate software architecture, including:

Inspection: This involves reviewing the architecture documentation, code, and other artifacts manually to identify problems or weaknesses.[1][2]

Model-based evaluation: This involves creating a model of the architecture and using tools to analyze the model for certain qualities, such as performance or scalability.[3]

Architecture tradeoff analysis method (ATAM): This is a process-oriented evaluation method that involves stakeholders in the evaluation process and focuses on identifying and balancing the tradeoffs made in the architecture.[1]

Architecture evaluation framework (AEF): This is a set of methods and techniques that can be used to evaluate software architecture against a set of predefined quality attributes.[2]

Quality attribute workshop (QAW): This is a workshop-based evaluation method that involves stakeholders in the evaluation process and focuses on identifying and prioritizing the quality attributes of the architecture.[3]

There are many other methods that can be used for software architecture evaluation, and the choice of method will depend on the specific needs and goals of the evaluation.

In this study, we employed a Systematic Literature Review (SLR) to conduct a survey. The SLR process includes three stages: planning, conducting, and reporting. During the planning phase, we developed a research methodology, identified the study sampling process, defined quality measurement rules, and developed a data gathering strategy. We also documented and replicated the extracted data. The search strategy we followed to collect research material from digital libraries is described in detail. Our research aims to conduct a systematic

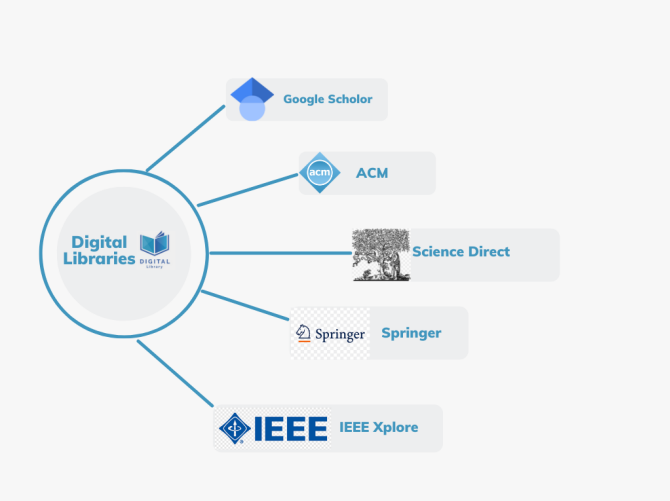

Fig. 1. Digital Libraries

review of previously published studies on improving software evolvability through various techniques and practices at the architectural level. We hope to provide a comprehensive overview of existing research in this area and to identify any challenges and gaps in the field that require further exploration. Our secondary goals are to provide practitioners with an updated understanding of the research themes that have been actively pursued in the field of software evolvability architecting and to assist them in finding relevant studies that meet their needs. Additionally, we aim to help the research community identify areas that need further investigation. Specifically, we have formulated the following research questions:

- What are the methods for software Architecture evaluation?
- What are the frequently used methods and processes to evaluate software Architecture?
- What type of metrics and methods are being used for software Architecture evaluation?
- What are the best practices to evaluate software Architecture?

The aim of this study is to examine the existing literature on the evolution of software architecture and the processes involved in evolving software architecture. Through our findings, we hope to determine the importance of architecture in ensuring the success and effective performance of a software system in terms of efficiency, scalability, enhancement, and optimal performance.

II. Research Method

This research followed a systematic review process, which involves several steps to comprehensively gather and assess all relevant knowledge on a specific topic. These steps include: creating a review protocol, defining inclusion and exclusion criteria, searching for relevant publications, evaluating the quality of the publications, extracting data from the publications, and synthesizing the data. These steps are described in further detail in the following sections.

A. Study selection procedure

- In the first level of screening, we went through titles and abstract of the papers to judge and decide if they fulfill the criteria of inclusion or not, so paper would be added or excluded.
- By defining the inclusion and exclusion criteria it was definite to add or remove papers on the basis of defined criteria.
- If there was enough indication that valid methods were discussed in the papers related to software architecture evaluation so those papers were included.
- By getting the answer of the question that does that paper satisfy the criteria and answers any of the research questions.
- If the study satisfies any of the research questions that is included in the list, moreover if the abstract satisfies the criteria but does not give the answer to any question that study is excluded.

B. Inclusion and Exclusion Criteria

1) Inclusion:

- Any of the articles, journal papers, or conference papers which include the meaningful indication related to the research topic.
- Articles, conference papers,or journal papers which include the abstract which focuses on the research topic.
- Papers cover the range of topics related to software architecture evaluation.

2) Exclusion:

- Papers lie outside the scope or variation of software architecture domain.
- Papers which do not possess the knowledge about software architecture domain evaluation.
- Papers with general introductory statements about software architecture evaluation in general.

C. Search terms

Comprehensive search string was used to find all the relevant results from the internet sources, digital libraries, and databases. Search string was written in such a way that maximum relevant results would be included in order to get the most appropriate and relevant data, that would further help in the research process to complete the desired process and to bring the maximum results. Following were the digital libraries where this search string was used to find out the date:

- ACM Digital Library (http://portal.acm.org).
- IEEE Xplore (http://www.ieee.org/web/publications/xplore/).
- ScienceDirect – Elsevier (http://www.elsevier.com).
- SpringerLink (http://www.springerlink.com).

1) Search String: **(”Software Architecture” AND (Evaluation OR Process* OR Analysis) AND (Methods OR Technique* OR Procedure* OR Practice*)) AND ”Review”**

D. Quality assessment

The following four criteria were used to evaluate the rigor and relevance of the studies included in this review:

- The data analysis is based on evidence or theoretical reasoning rather than unsupported or ad hoc statements.
- The study provides context for the research.The research aims are supported by the design and execution of the study.
- The study describes the method used for data collection.
- All of the included studies met these criteria to ensure the credibility and relevance of the data for synthesis in the review.

E. Data Extraction Form

Systematic Literature Review_ Data Extraction Form

Study Identifier		Notes:		
Title				
Year				
Author(s)				
Publication Venue	○ Journal	○ Conference	○ Other_____	
Research Question				
RQ1	RQ2	RQ3	RQ4	
Fulfillment of inclusion Criteria				
IC1	IC2	IC3	IC4	IC5
Quality Assessment score				
QC1	QC2	QC3	QC4	
Multi criteria evaluation Method				
Solution Type	Single	Hybrid		
Uncertainty Consideration	Yes	No		
Criteria				
Primary Criteria				
Secondary Criteria				
Criteria Type	Subjective	Objective	Combined	
Application Domain	IaaS	PaaS	SaaS	General
Validation				
Contribution	Integration	Utilization	Extension	New method
Description / Summary				

III. Conclusion

As technology and business progress, creating and evolving software systems with desired quality attributes is becoming more complex. As it is often not desirable to start from scratch, researchers are working to develop effective approaches for evolving software systems. This review seeks to provide a comprehensive overview of existing studies on analyzing and achieving software evolvability at the architectural level. 82 primary studies have been identified, covering a range of approaches with a specific focus on architecture-centric activities in the software lifecycle. Although the approaches vary in terms of terminology, descriptions, artifacts, and activities, they share common goals and focus.

Our conclusions is summarized in following points:

1) Quality considerations during design: approaches are divided into three sub-categories: focused on quality attribute requirements, focused on quality attribute scenarios, and focused on influencing factors. These techniques aim to identify key quality attributes early in the design phase, but often focus on quality attributes in general, rather than specifically on evolvability.
2) Architectural quality evaluation: techniques are used to refine quality attribute requirements and scenarios as the architecture takes shape. These approaches are further divided into experienced-based, scenario-based, and metric-based. These studies often focus on particular quality attributes, such as adaptability, and do not cover evolvability subcharacteristics.
3) Economic valuation: these approaches provide insight into the business consequences of architectural decisions and help teams choose among options. Most studies focus on a single quality attribute, like stability or flexibility, and may not be adequate when multiple evolvability subcharacteristics are involved.
4) Architectural knowledge management: these approaches improve architectural integrity by adding architectural knowledge from different sources to the documentation.
5) Modeling techniques: these techniques add value by modeling software artifacts and their relationships, along with traceability and visualization of the impact of architecture evolution. They do not explicitly focus on evolvability, but can improve the control and improvement of software architecture evolution by modeling the relationships among inter-dependent software artifacts.

IV. Future Work

Future research can be conducted on the other areas of social networks like community detection, behaviour detection and influence detection to check which machine learning models perform best for these kind of researches in social networks. Some other machine learning models can also be considered for better analysis as in our experimentation we only used five supervised machine learning algorithms. Mostly, the work done in sentiment analysis is on textual data there maybe further studies conducted for sentiment analysis on the videos and audio data. As in social network analysis live data matters more than the old data because data is dynamic changes after each interval of time. So the model that is trained in online learning performs better than the model which is trained on batch learning because this is more useful for future predictions.

REFERENCES

[1] Rafael Ferreira Barcelos and Guilherme Horta Travassos. *Evaluation Approaches for Software Architectural Documents: a Systematic Review.* 2006.

[2] Aldeida Aleti et al. *Software architecture optimization methods: A systematic literature review*. 2012.

[3] Hongyu Pei Breivold, Ivica Crnkovic, and Magnus Larsson. *A systematic review of software architecture evolution research*. 2012.

Rashid Noor received the B.S. degree in software engineering from Riphah International University, Islamabad, Pakistan, in 2021. He is currently pursuing M.S. degree with the School of Computing, FAST National University of Computer and Emerging sciences, Islamabad, Pakistan. His research interests include graph neural networks, social network analysis and data modelling with advance algorithms, systematic analysis of software development techniques.