*I. J. Mathematical Sciences and Computing,* **2023, 4, 43-49**
Published Online on December 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmsc.2023.04.05

# A Decision-Making Technique for Software Architecture Design

**Jubayer Ahamed***
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: jubayer@aiub.edu
ORCID ID: http://orcid.org/0000-0003-2076-9194
*Corresponding Author

**Dip Nandi**
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: dip.nandi@aiub.edu
ORCID ID: http://orcid.org/0000-0002-9019-9740

Received: 30 July, 2023; Revised: 21 August, 2023; Accepted: 19 September, 2023; Published: 08 December, 2023

**Abstract:** The process of making decisions on software architecture is the greatest significance for the achievement of a software system's success. Software architecture establishes the framework of the system, specifies its characteristics, and has significant and major effects across the whole life cycle of the system. The complicated characteristics of the software development context and the significance of the problem have caused the research community to build various methodologies focused on supporting software architects to improve their decision-making abilities. With these efforts, the implementation of such systematic methodologies looks to be somewhat constrained in practical application. Moreover, the decision-makers must overcome unexpected difficulties due to the varying software development processes that propose distinct approaches for architecture design. The understanding of these design approaches helps to develop the architectural design framework. In the area of software architecture, a significant change has occurred wherein the focus has shifted from primarily identifying the result of the architecting process, which was primarily expressed through the representation of components and connectors, to the documentation of architectural design decisions and the underlying reasoning behind them. This shift finally concludes in the creation of an architectural design framework. So, a correct decision- making approach is needed to design the software architecture. The present study analyzes the design decisions and proposes a new design decision model for the software architecture. This study introduces a new approach to the decision-making model, wherein software architecture design is viewed based on specific decisions.

**Index Terms:** Software Architecture, Decision-making technique, Software Design Activities, Decision Mode, Decision Nature, Decision Makers.

## 1. Introduction

The design of software architecture happens during the first stages of the implementation phase and helps ensure the completion of distinct functional needs, nonfunctional requirements, and business objectives. Therefore, architectural decisions play an important part in determining the success of a project that heavily relies on software. The software design phase holds vital importance within the software development process. During this phase of the software life cycle, the architecture of the upcoming software system is established, and the system is comprehensively explained. The significance of software design in minimizing software expenses is notable. Researchers have made efforts to identify and establish quality measures for the purpose of characterizing software design. Software designers and developers routinely engage in decision-making processes. It is essential for software architects to possess a dependable and detailed methodology for analyzing and selecting architectural choices, while also making sure that the resulting decisions effectively address potential risks and enhance financial benefits. An effective decision-making technique has been defined by its ability to lead the user towards more effective, perhaps optimal, while also being user-friendly. The utilization of an ineffective decision-making methodology may give rise to many challenges, ultimately leading to the choice of a bad alternative. Human beings are actively engaged in decision-making processes, which can vary in their approach and methodology. The human factors involved in the decision-making process and developing a greater

 

understanding of the methodologies applied by software architects in their design methodologies are of the greatest significance. The main objective is to develop a new decision-making model that helps to take the right decisions for a software architecture. This work focuses on the decision-making techniques in software architecture. It helps the decision-makers to take the effective decision. In section 2, we discuss the previous works on decision-making techniques in software architecture and design. The proposed solution and discussion portion will be described in Section 3. In section 4, we set the conclusion part.

## 2. Related Work

*i) Software Architecture*

This research contributed an innovative perspective to software architecture by defining it as a combination of a set of explicit design decisions. With this perspective, architectural design decisions in software architecture are no longer hidden. Some elements were the basis for some design decisions, which were discussed. These included design guidelines, design restrictions, and additional needs. Designers may unconsciously run short of design principles and limitations resulting from earlier design decisions as a system evolves. Because of this, there will be difficulties with this method in the future [1].

N. Medvidovic and R. N. Taylor et al. [2] discussed Architectural Description Languages (ADLs) utilize a formal language for describing software architectures, thus enabling the use of first-class architectural concepts. While ADLs were seeking to show a system's structure, they focused on the development process that concluded in the creation of the structure. Participants discussed a system for keeping track of changes made to an architecture description using an updated management tool. However, neither the idea of a design decision nor the delta was present. Therefore, it could only track random changes and not the relationships between the architect's deliberate decisions.

Current software architecture practices continue to depend on the fundamental principles defined by Dewayne E. Perry and Alexander L. Wolf in their brief but elegant equation "Architecture = {Elements, Form, Rationale}". The fundamental parts of any architectural description in terms of aspects and connectors are three elements, while the nonfunctional properties play a crucial role in determining the ultimate form of the architecture. The objective ofthis work was to establish the fundamental basis for software architecture. The authors proposed a theoretical framework for software architecture that comprises three fundamental constituents, namely, elements, form, and rationale, which were derived from their intuition [3].

The author suggested that software architecture should be seen as a process for choosing where to invest capital into software assets with uncertain and long-term worth. To back up this point of view, the author used choices theory to explain ideas related to software architecture. Lacking a unified and consistent structure, software design is approached in a disconnected and distinctive fashion. Despite the abundance of available information, many people still find it difficult to understand, implement, and acquire software design principles [4].

D. Garlan and D. Perry et al. [5] discussed some major issues of software architecture and design. This paper showed the connection between Design Methods and Software Architecture. Then the architectural design decision should be taken based on this discussion.

This study familiarized the subjects with the current advancements and practical applications in the domain of software architecture and examined the potential developments that are expected to emerge in this area [6]. The authors worked on conceptual integrity, intellectual control, effective reuse of knowledge, experience, design and code, effective project communication; and management of a set of related variant systems.

*ii) Software Design Activities*

Tang A. and Vliet H. et al. [7] discussed about software design choices that led to the development of certain architectural approaches. It was possible for designers to perform a wide range of activities in a variety of orders and variations. Inquirers might, for instance, begin by examining all the different types of design options or by getting deeply into one area of design. The design process itself is a good time to use either a problem- or solution-focused strategy. Two different methods of design exploration were worked on during the research. Both the breadth-first and depth-first exploration strategies were used. The purpose of this research was to evaluate the relative merits of user-driven and solution-driven approaches to solving design challenges, with a focus on the relationship between design complexity and designer experience.

Software design activities help to create the architecture of software. The software design activities are performed by following specific decisions. Those decisions come from the decision-making techniques. The process of design reasoning involved utilizing both design rationale and information to facilitate logic-based reasoning in the context of decision-making. The investigation had shown a carefully organized design rationale constitutes a form of documentation that facilitated the monitoring and assessment of the matters and options under consideration by designers. Numerous techniques for design rationale have been proposed. While design rationale offered explanations for design decisions, it did not indicate the psychological procedures involved in the logic behind them [8].

*iii) Decision-Making Techniques for the software design*

This study conducts a review related to decision-making in the field of software architecture, specifically emphasizing human behaviors and practices [9]. The researchers performed a comprehensive examination of the available body of literature pertaining to decision-making within the domain of software architecture. The investigation was categorized into two discrete areas, specifically decision-making behavior and decision-making practice. A systematic literature review was not undertaken due to the absence of a complete search across all databases. Their capacity to claim comprehensive coverage of all material pertaining to decision-making in software architecture was constrained.

Christiaans H. and Almendra R. et al. [10] showed the decision-making technique in software engineering. The objective of their study was to investigate the software designer's design process and compare the findings with those of the product architects. The authors applied a descriptive model of decision-making to evaluate the processes of three software design teams. In this study, they used three teams (Intuit, Adobe Systems Incorporated, and AmberPoint). Every team followed the same criteria and level of analysis. But their working strategy was different. And lastly, they compared the overall work and tried to find out the best decision-making technique. But the usability approach was absent in the protocols of Intuit and Adobe Systems Incorporated.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides et al. [11] analyzed design patterns to make decisions for software architecture. A design pattern can be defined as a distinct category of design choices. Design patterns refer to specified design choices that hold established functionality and behavior. But selecting the correct design pattern is important. If engineers choose the wrong pattern, then it will create some big issues.

The application of decision-making techniques was first observed during the 1980s. The author has identifieda total of seven technical and business issues and has provided a description of their respective effects to make good architectural designs for the software industry. Most modern architectural rationale systems display shortcomings in addressing realistic considerations, such as the economical utilization of resources and effortless integration [12].

The protocols, to a certain extent, depict the system development organization in terms of a series of actions that convert client requirements into an acceptable design and of the information flow controlled by a single decision-maker who makes both design and development decisions while adhering to time and financial constraints [13]. It is a system for producing decisions.

This study analyzed previous models for architectural design decisions and conducted a comparison of tools basedon established criteria [14]. The primary objective of this scholarly article was two-fold. Firstly, it was aiming to demonstrate that all the models presented in the study were in agreement in terms of capturing the fundamental nature of an architectural design decision. Secondly, it tried to clarify the principal differences among the tools and highlight the specific features that were absent from these tools. Nine ADD models and associated tools were surveyed.

These models were compared in order to make clear their similarities and differences. Finally, they suggested the designing technique and tool based on the comparison.

Carvajal L., Moreno A., Isabel MI., and Seffah A. et al. [15] have discussed some usability guidelines for software architecture and design. Their guidelines highlighted the review of software design elements. The authors provided a comprehensive analysis of the composition of the proposed design items, connecting them with the established software development process and software architecture applied in each respective application. To make a good usability approach, the guidelines or rules were very important. Usability approach used for software development and software design. Without a good usability approach, usability issues will occur. Then it will have a bad impact on the software's architecture. The overall architecture of the software will be bad for usability problems.

*iv)Problem Classification*

From the above discussion, we try to find out some limitations. After finding the limitations, we show the limitations in a structured form. The limitations have been discussed briefly in table 1.

We try to classify some of the issues. During system evolution, designers may unexpectedly encounter conflicts with design rules and restrictions that have emerged from previous design decisions. This procedure may potentially give rise to future challenges or complications [1]. The absence of a notion of a design decision and delta has been noticed. Therefore, it maintained a capacity to simply observe unpredictable changes and not the interrelated nature of the architectural decisions that the architect was looking for [2]. The researchers conducted a comparative analysis of various decision-making techniques in order to determine the most effective approach. However, the incorporation of the usability approach was not evident in the protocols of Intuit and Adobe Systems Incorporated [3]. Selected wrong design pattern is a big issue for software architecture [4].

The overall findings obtained from the analysis indicate that the negative results can be identified as inefficient decision-making processes in the design of the software architecture. These issues cause inefficient and negative results in the architecture of a software. So, decision-makers will need to follow an effective model or technique that will help them to avoid those issues and make appropriate decisions for making the architecture of a software.

Table 1. Some limitations of the related literature

| **Description** | **Problem/Gap** |
|---|---|
| An innovative method of software architecture, where software architecture was viewed as a compilation of a series of explicit design choices. This viewpoint made architectural design choices a visible component of software architecture [1]. | Throughout the evolution of a system, designers may unintentionally conflict with design rules and constraints arising from prior design choices. This will be created some issues with this technique in the future. |
| N. Medvidovic and R. N. Taylor et al. [2] discussed Architectural Description Languages (ADLs) utilize a formal language for describing software architectures, thus enabling the use of first-class architectural concepts. | The concept of a design decision and delta was absent. Hence, it had the capability to only monitor capricious shifts and not the interrelationships among the design choices that the architect wanted to achieve. |
| This study analyzes the existing state of research on decision-making in software architecture, with a focus on human behaviors and practices [9]. | A comprehensive search of all databases was not conducted to perform a systematic literature review. Their ability to assert that all literature on decision-making in software architecture had been covered was limited. |
| Christiaans H. and Almendra R. et al. [10] showed the decision-making technique in software engineering. The objective of their study was to investigate the software designer's design process and compare the findings with those of the product architects. | They compared the overall work and tried to find out the best decision-making technique. But the usability approach was absent in the protocols of Intuit and Adobe Systems Incorporated. |
| E. Gamma, R. Helm, R. Johnson, and J.Vlissides et al. [11] analyzed design patterns to make decisions for the software architecture. | Selecting the correct design pattern is important. If engineers choose the wrong pattern, then it will be created some big issues. |

## 3. Proposed Solution and Discussion

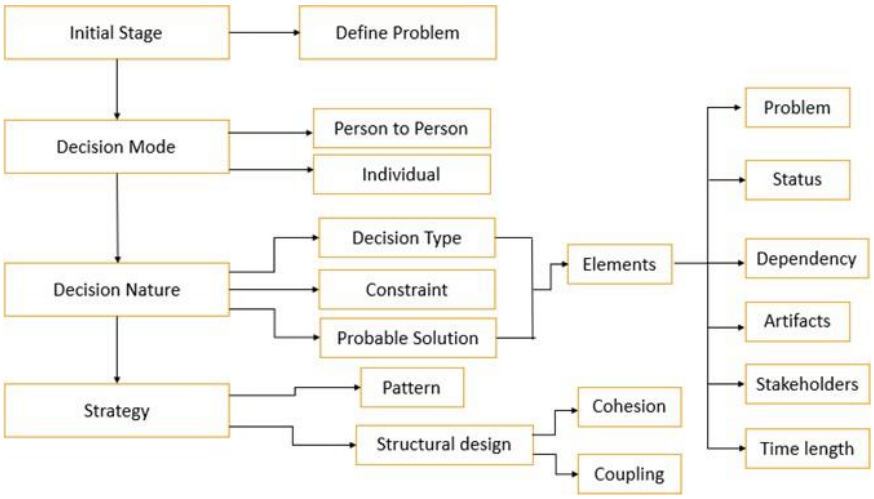


Fig.1. Block diagram of our proposed decision-making technique for software architecture design

In this part, our proposed model has been shown. There are four phases in this model. These are Initial Stage, Decision Mode, Decision Nature and Strategy. These four phases are divided into different types of sub- phases. After completing one phase, the process will follow the next phase. Now, we are going to discuss all the phases & sub-phases in Table 2.

Table 2. Proposed decision-making technique's phases & sub-phases for software architecture design

| Phase 1: Initial Stage | Phase 2: Decision Mode | Phase 3: Decision Nature | Phase 4: Strategy |
|---|---|---|---|
| Sub-Phase:<br>a) Define Problem: In this phase, the project team or decision-makers will need to identify the problem. If they identify the problem correctly at first, then they will choose the correctdecision-makingtechnique. | Decision-makers will need to identify the project mode.<br>Sub-Phases:<br>a) Person to Person: This type of project is called the "group" project.<br>b) Individual: This type of project is called a "single" project. | Sub-Phases:<br>a) Decision Type: After identifying the problem, decision makers will select the decision type based on the problem.<br>b) Constraint: Set rules and regulations properly based on the problem.<br>c) Probable Solution: Think about the solution based on the problem and decision type. | After completing Phase 2 and Phase 3, the decision-makers will select the strategy.<br>Sub-Phases:<br>a) Pattern: This is the pattern-based strategy. There are many types of patterns in software engineering. Example: Behavior pattern, Structural pattern, etc.<br>b) Structural Design: This is the structural design-based strategy. There are some structural design types.<br>Types are:<br>i) Cohesion: It represents the relationship within a module.<br>ii) Coupling: It represents the relationships between modules |
| | | After completing all of these sub-phases, then a new module will take place. This is called "Elements". Elements are one type of sub-phase. This phase is based on the define problem, decision type, constraint, and probable solution. In this sub-phase, there are eight different elements. These are the problem, status, dependency, artifacts, stakeholders, and time length. | |

There are four phases in our proposed model. The phases are initial stage, decision mode, decision nature and strategy. We need to identify the problem in the initial stage. After that, we need to identify the mode of the decision. Decision mode defines what type of people will take the decision and play the role of the decision makers. Either a group of people will conduct this and play the role of the decision makers, or an individual will conduct this. After completing this phase, the next phase will come. The third phase is decision nature. There are some sub-phases in this phase. Those sub-phases are decision type, constraint and probable solution. Decision makers will need to set the typeof decision. After that, they will follow the rules and regulations of the decision type. Because if they don't follow the rules and regulations, then the decision may not be valid in some cases. That is why the constraint is very important. Then the decision makers will find out the possible solution types in the probable solution sub phase. All of these subphases depend on the elements. The elements are the problem, status, dependency, artifacts, stakeholders and time- length. Decision makers will complete the sub-phases based on these elements in the third phase. After completing the third phase, the final phase of our proposed model will arrive. The final phase is "Strategy". Decision makers willchoose the strategy in that phase to design the software architecture. There are two types of strategy in this phase. Those are pattern-based strategy and structural design-based strategy. There are different types of software patterns. We can design a model of software architecture using these patterns. Also, we can solve the software design problem by using patterns. So, this is one of the valuable strategies for software architecture. Another strategy is structural design. This strategy introduces the cohesion and coupling for the software. Cohesion and Coupling are very important factors in software architecture and design. Cohesion defines the relationship within the modules. Coupling defines the relationship between the modules. Decision makers need to identify which one is more and which one is less. Because a good software architecture has low coupling and high cohesion. It will be easier to maintain. Also,the overall software will perform well.

Our proposed model will assist the decision-makers to take the correct decision in the software's design. Using this model, the decision-makers will find the answer to the following questions:

i) How they select specific techniques for their software's architecture design.
ii) How they follow specific strategies.
iii) How they make certain decisions.

The proposed model is straightforward and well-structured. The beginning of the next phase is dependent upon a successful conclusion of the previous phase by the decision-makers. By following a sequential approach and successfully completing each phase before moving on to the next, the whole process is more likely to produce the correct results. It also means that those responsible for decision-making will make appropriate choices regarding the architectural design of their software. It is essential to make accurate choices in software design in order to ensure the best possible results. The accuracy of the design is dependent upon making the appropriate decision. Without making the appropriate choice, the design of any software will not be perfect. Also, the overall performance of the software will be inefficient. Proper

decisions are crucial for software design. When applying the proposed model's concept, obtaining the right decisions for the software team will be significantly easier.

## 4. Conclusion

The use of decision-making strategies holds significant importance in the process of creating an appropriate software architecture. The process of generating software architecture decisions requires more than a basic integration of knowledge and information in order to provide software outputs. It is vital that these choices be followed by a thoroughly defined design rationale. The process of software architecture design necessitates the active participation of multiple stakeholders and encompasses a wide variety of activities. Initially, we attempted to identify some issues with the decision-making technique. After identifying the issues, those issues were analyzed and classified. Then, we proposed a new decision-making technique model. We made an effort to provide a model that is easier to use and simpler than previous models. Our proposed model includes four phases. Initial Stage, Decision Mode, Decision Nature, and Strategy are the four primary elements under consideration. The four phases are further subdivided into numerous subphase categories. Upon conclusion of a particular phase, the subsequent phase will be executed.

The phases of the proposed model are particularly well-structured. Because it follows an ordered sequence. There is no option for ignoring any stages. The proposed model is simple to understand. The proposed model is not being implemented anywhere due to a lack of suitable resources. We will seek out the necessary space and resources to implement this model. We want to implement the proposed model into action and analyze its results in the near future.

## References


[1] Jansen A. and Bosch J., "Software Architecture as a Set of Architectural Design Decisions", Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005 IEEE.

[2] N. Medvidovic and R. N. Taylor, "A classification and comparison framework for software architecture descriptionlanguages.", IEEE Transactions on Software Engineering, 26(1):70–93, 2000.

[3] D.E. Perry and A.L. Wolf, "Foundations for the Study of Software Architecture," ACM Software Eng. Notes, vol. 17, no. 4, 1992, pp. 40–52.

[4] Richard N. Taylor and Andr é van der Hoek, "Software Design and Architecture: The once and future focus of software engineering", Future of Software Engineering (FOSE'07), IEEE computer society, 2007.

[5] D. Garlan and D. Perry, "Introduction to the special issue on software architecture", Computer Science, IEEE Trans. Software Eng., Published 1 April 1995.

[6] N. Medvidovic and R. N. Taylor, "Software Architecture: Foundations, Theory, and Practice", ICSE'10, Cape Town, May 2- 8 2010.

[7] Antony Tang and Hans van, "Design Strategy and Software Design Effectiveness", Swinburne University of Technology,IEEE, 2012.

[8] J. Lee and K. Lai, "What is Design Rationale?" in Design Rationale - Concepts, Techniques, and Use, T. Moran and J. Carroll, Eds., New Jersey: Lawrence Erlbaum, pp. 21-51, 1996.

[9] Tang A., Razavian M., Paech B. and Hesse T., "Human Aspects in Software Architecture Decision Making: A Literature Review", IEEE International Conference on Software Architecture, 2017. doi: 10.1109/ICSA.2017.15.

[10] Christiaans H. and Almendra R., "Accessing decision-making in software design", Elsevier Ltd, 2010.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns - Elements of Reusable Object-Oriented Software.", Addison Wesley, 1994.

[12] J. Lee, "Design Rationale Systems: Understanding the Issues," IEEE Expert, vol. 12, pp. 78-85, 1997.

[13] U. Van Heesch, P. Avgeriou, and R. Hilliard, "Forces on architecture decisions-a viewpoint," in Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on, 2012,pp. 101-110.

[14] Shahin M., Liang P. and Khayyambashi M.,"Architectural Design Decision: Existing Models and Tools", IEEE/IFIP WICSA/ECS,2005.

[15] Carvajal L., Moreno A., Isabel MI., and Seffah A.," Usability through Software Design", IEEE Transactions on Software Engineering, vol. 39, no. 11, November 2013.


### Authors' Profiles

**Jubayer Ahamed** completed his B.Sc. in Computer Science from American International University-Bangladesh, Dhaka, Bangladesh. He obtained his MSc from American International University-Bangladesh, Dhaka, Bangladesh. Currently, Jubayer Ahamed is working as a Lecturer in the Department of Computer Science (CS) at American International University- Bangladesh (AIUB). His research interest includes Software Engineering.

**Prof. Dr. Dip Nandi** pursued his B.Sc. in Computer Science from American International University-Bangladesh, Dhaka, Bangladesh. He obtained his MSc from the University of Melbourne and Ph.D. from RMIT University, Australia. Currently, Dr. Dip Nandi is the Associate Dean of the Department of Computer Science, American International University-Bangladesh. Hisresearch interests include Software Engineering, Technology E-learning, Theory of Learning and Human-Computer Interaction.

**How to cite this paper:** Jubayer Ahamed, Dip Nandi, "A Decision-Making Technique for Software Architecture Design", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.9, No.4, pp. 44-49, 2023. DOI: 10.5815/ijmsc.2023.04.05