

# SOFTWARE ECOSYSTEM ARCHITECTURAL CHALLENGES AND MITIGATION STRATEGIES PRIORITIZATION USING ANP ALGORITHM

Inayat Ur Rahman<sup>1</sup>, Sahib Bahadar<sup>2</sup> and Sara Shahzad<sup>3</sup>

<sup>1</sup>Department of School of Computer and Software, Nanjing University of Information Science and Technology, China

<sup>2</sup>School of Electronic and Information Engineering, Nanjing University of Information Science and Technology, China

<sup>3</sup> School of Computer Science, University of Peshawar, Pakistan

## Abstract

*In recent years, software ecosystems (SECO) have emerged as a key paradigm of modern software development, bringing together a large number of entities that collaborate to create, maintain and evolve software products. While contributing to collaboration, innovation, and quick software evolution, SECO also pose complex architectures that may impact on scalability, security, reliability, flexibility, interface stability, and long term sustainability. Although a previous Systematic Literature Review (SLR) identified key architectural challenges and how they might be mitigated in SECO, its relative importance and the order in which to consider its mitigation strategies was unclear. To fill this gap, the present study emphasizes on architectural challenges and the corresponding mitigation strategies for the events identified through analysis of SLR, using Multi-Criteria Decision-Making (MCDM) approach Analytic Network Process (ANP) which is appropriate for complex decision variables. The expert judgment data were gathered by carrying out a structured survey from software practitioners of the software ecosystem and professionals and then analyzed using the ANP based pairwise comparison and priority weight computation. The results indicate that the most important architectural challenge is Scalability of SECO Architecture, then Security, Architecture Flexibility, Reliability of SECO Architecture, and lastly Interface stability. Based upon the above, the capacity of software ecosystems to assist in growth, interaction assurance, adaptability to architecture, service reliability and stable interfaces are key to sustainable development of a software eco-system. With regard to mitigation strategies, Framework is the top performing mitigation methodology, followed by Metrics, Custom Design, Assessment of Ecosystem Health, Sandbox, and Variable Design Decisions. This study further adds value to the software ecosystem research field by adding the findings of the previously mentioned SLR which was published in the field, through quantitative prioritization. The proposed prioritization based on ANPs offers pragmatic suggestions for SECO software architects, researchers, and practitioners towards identifying the most important architectural challenges and choosing corrective actions, i.e., mitigation strategies. The results assist in making informed architectural decisions, proactive risk management, and enhancing the sustainability of software ecosystem architecture.*

## Keywords:

SECO, ANP, MCDM, NumPy, Software Ecosystem Architecture

## 1. INTRODUCTION

Software ecosystems (SECOs) are a new paradigm in the software development sector, in which multiple organizations formally interact through a common platform and technological infrastructure [1]. SECO describes the interactions among software companies, developers, and other stakeholders to design, retain, and develop software products and services. Various definitions of software ecosystems have been proposed in the literature. Following Bosch [2], a software ecosystem comprises a collection of software products and the organizations that produce them, which can connect actors in social and economic

environments. In like manner, a software ecosystem is explained by Jansen, Finkelstein, and Brink Kemper as a community of actors that contributes to software development and communicates with each other via a common software and service market [3]. Software ecosystems tend to have interactions among players that share a common technological platform on which organizations work together and, within a similar time frame, compete. These ecosystems have a central organization that coordinates the underlying platform technology and shares in their success [3]. Due to the fast development of platforms like Android, Apple iOS, Microsoft, and Salesforce, the term software ecosystem has become a major concern among the industry and academia. Such conditions usually result in innovation developed through the cooperation of more than one organization rather than by a single party [4].

Software ecosystems offer various advantages to firms, such as enhanced cooperation, lower development costs, accelerated innovation, and coevolution of software products and services among the firms involved [4] [21]. Nevertheless, such benefits have several challenges that are induced by software ecosystems to their growth and continuity. These challenges are associated with architecture, governance, sustainability, data analytics, dynamic/evolution, and domain-level ecosystem [5].

Among these challenges, architectural challenges are among the most important issues in software ecosystems [6]. Earlier research has identified various architectural issues and recommended possible measures to address them [7][8] [10]. Nonetheless, no prioritization in terms of the relative weight of these challenges and mitigation strategies has been taken seriously in the current research. To fill this gap, a Systematic Literature Review (SLR) was conducted to identify the architectural challenges in software ecosystems and the mitigation approaches reported in the literature. Moreover, the obtained SLR data were processed using the ANP to rank the identified obstacles and their corresponding mitigation measures.

The ANP method allows examining the intricate relationships among variables and identifying the most significant architectural problems that impact software ecosystems. The findings of this paper can offer a prioritized conception of architectural challenges and threat-prevention initiatives, which may assist researchers and practitioners in enhancing the structure of the software ecosystem and decision-making. The rest of this paper will be structured in the following manner. Section 2 discusses the associated work. Section 3 outlines research methodology, i.e., the ANP analysis process. Section 4: Discussion results and analysis of the study. The last section of the paper, Section 5, provides directions for future research.

## 2. RELATED WORK

Software ecosystems (SECO) are interconnected networks of software platforms, software developers, and software services and companies that work together to design and sustain software products and services [12]. Despite the various benefits of software ecosystems, including favorable collaboration, innovation, and cost reduction, there are also numerous challenges that influence their development and sustainability. Such challenges can typically be divided into domains, including architecture, governance, sustainability, data analytics, dynamics and evolution, and domain-specific ecosystems [13]. One such area is architectural challenges, which are among the most significant concerns in software ecosystems. Software architecture refers to the high-level structure of a piece of software and the framework upon which the system components and their interactions are designed and outlined. It determines the division of system components, the distribution of duties, the design of interfaces, and so on, to facilitate communication among various parts of the system. As such, architectural design plays a significant role in determining the quality, maintainability, and scalability of software systems. A number of literature [14] [15] [16] [17] has indicated that one of the leading challenges of developing and maintaining software ecosystems is the architecture problems.

These papers have identified varied architectural counterparts arising from intricate interactions among consumers, platforms, and technological infrastructures in the ecosystem. Moreover, scholars have proposed various mitigation measures to address architectural issues and enhance the stability and performance of software ecosystems [14] [15] [16]. Nevertheless, although various architectural challenges have been identified and mitigation measures determined, the extant literature has not thoroughly examined the relationship between these challenges and the recommended mitigation strategies.

In a bid to fill this knowledge gap, the outcomes of a Systematic Literature Review (SLR) were applied in this research to recognize architectural risks and their respective solutions within software ecosystems [22]. The obtained data are analyzed using the ANP to prioritize the challenges and mitigation strategies identified and rank them according to their significance. The research results of this paper will lead to a prioritized conceptualization of architectural problems in software ecosystems and provide suggestions on the most appropriate techniques for mitigation to enhance the architecture of software ecosystems.

## 3. RESEARCH METHODOLOGY

The research approach for the present study is discussed in this section. The Fig.1 displays the overall research design. The procedures for conducting the study are detailed below.

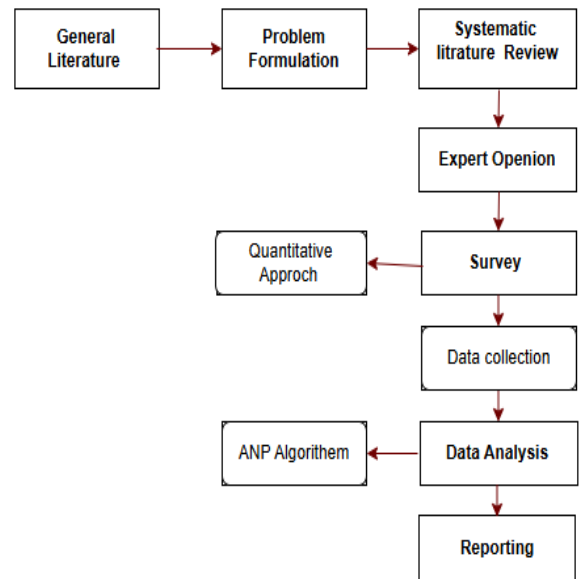


Fig.1. Research Methodology

### 3.1 SYSTEMATIC LITERATURE REVIEW

In our previous research report [11], an SLR was conducted to identify SECO architectural issues and applicable mitigation strategies; a set of closed-ended questionnaires was used to collect information during the interview process. The discovery of the challenges of the architectural form of SECO and the solution of the challenges. These mitigation strategies address several issues simultaneously.

#### 3.1.1 Aims and Objective:

The current study attempts to fill the knowledge gaps in the literature and offer suggestions for understanding the impact of various SECO concerns and their mitigation measures. By establishing the following objectives, the current study's objective will be accomplished

- The research is significant because we identified every threat to the SECO architecture, potential countermeasures, and measured the extent of those threats' effects on the SECO architecture.
- An ANP algorithm will be used for prioritizing in the final stage.

This research study identifies the architectural challenges faced by SECO and proposes mitigation measures to address them, thereby reducing their impact and contributing significantly to the research literature. The study's conclusions will be useful to both the professional and research sectors. The only issues addressed by this study are those relating to Architecture and any associated SECO mitigation measures.

### 3.2 QUESTIONER DEVELOPMENT

Usually, surveys are conducted to take a clear picture of the current circumstances. Also, surveys provide information that is difficult to obtain through observational methods [17]. In the current study, we conducted a complete investigation. Based on the SLR results, a questionnaire was developed. Close-ended questions were used to collect data from the target population who had experience and expertise in working in the SECO

Architecture environment. Respondents find it easier to answer closed-ended questions [18]. A simple draft or Google Form of the questionnaire was developed at an early stage and further refined through various validity tests and a pilot study, as discussed in detail below.

### 3.3 QUESTIONNAIRE LAYOUT

Development is a difficult task in a quantitative study. Each questionnaire section must be developed in accordance with the study's research objectives and goals. The questionnaire is classically divided into several sections. Every section comprises various questions regarding that section's subject [17][18]. The survey questionnaire contained four main sections, i.e., section 1, section 2, section 3, and section 4. The first section focused on the respondent's demographic information. This section contains general information about the respondent. Personal information of the respondent, including their gender, education, and email address, was collected. Similarly, the respondent's working experience and their role and responsibility in software ecosystem-based organizations were obtained. The SECO section includes the organizational-related information. In this section, the questionnaire was designed to collect information about the organization where the respondent works. It may include the organization's size, the number of employees, and the type of project they develop. The third section provides software ecosystem issues that affect software ecosystem Architecture and act as an endogenous construct in the current study. This section measures software ecosystem issues, including interface stability, security, reliability, and so on.

#### 3.3.1 Measurement Scale:

The five-point Likert scale is used in the existing study. The options included in the five-point Likert scale, starting from 1 = 'strongly disagree', 2 = 'disagree', 3 = 'neutral', 4 = 'agree', 5 = 'strongly agree' were used for the items given in the questionnaire [19]. In [19], the author claimed that using a neutral point on the scale has no drawbacks. Moreover, providing the neutral option may result in unbiased data, as respondents are not forced to answer either positively or negatively. Similarly, the center point in a five-point Likert scale (i.e., 3) is right in the middle and impeccably signifies a mixed feeling [20]. The survey questionnaire sample is provided in Appendix A".

#### 3.3.2 Questionnaire Pre-Testing:

Pre-testing the questionnaire is very important before conducting a survey. It helps to examine the questionnaire's errors and limitations. The questionnaire's pre-testing comprises four steps: content validity, questionnaire readability, a pilot study, and a final review. The Fig.6 below represents the steps of questionnaire pre-testing.

#### 3.3.3 Content Validity:

To evaluate and assess the reliability and authenticity of the questionnaire, the first step is to assess its content validity. Content validity should be conducted by domain specialists in the relevant field, or else it can be difficult for respondents. Content validity involves experts reviewing the various items and texts mentioned in this survey questionnaire. Check the questionnaire and confirm whether the respondent can easily understand the question or is confused. The following points were considered in evaluating content validity.

- All the questions that have been used to access the variable are reviewed.
- Scales used in the questionnaire and their appropriateness is reviewed.
- Section arrangement and pitfalls are reviewed and eliminated.

All the changes were made accordingly after the review.

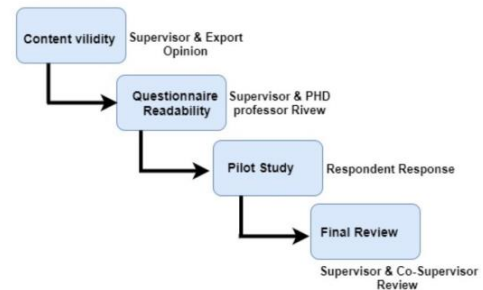


Fig.2. Questionnaire Pre-Testing

### 3.4 QUESTIONNAIRE READABILITY

It is also important to review the questionnaire's readability before proceeding with the survey. To evaluate the questionnaire's readability, grammar, and font, English is also checked. If the questionnaire is not legible, the respondent may not answer appropriately or may get confused by the questions.

Therefore, a readability test was conducted to make the questionnaire easier to understand for the respondent. To evaluate questionnaire readability, the researcher's supervisor is an expert in software engineering, and two other Ph.D. professors have reviewed the questionnaire and evaluated its readability. The following points were taken into consideration while evaluating the content validity. Are the words properly understandable?

- Are the words properly understandable?
- Do all questions can be marked by the respondent easily?
- Does professionalism seem professional in terms of presentation?
- Does the length and clarity of the questionnaire is appropriate?
- Is there any grammatical or writing mistake in the questionnaire?

All the changes were made accordingly after the questionnaire is accessed by experts and pass through the readability phase.

#### 3.4.1 Pilot Study:

A pilot study was conducted to assess the reliability of the survey questionnaire. A preliminary study is conducted before conducting the full survey of the target population. The current study is based on a small-scale pilot to assess the plan and methodology of the study. Moreover, it is also conducted at the early stage to review and revise the items of the questionnaire and analyze the correctness and precision of the questionnaire.

Questionnaires were sent to two SECO Architecture organizations, the software ecosystem organization, and Architecture experts. Data collected from the experimental study were tested using the ANP algorithm. The questionnaire was further modified based on respondents' answers in the pilot study.

Responder is assured that the questionnaire's data will not be shared with anyone else and will be kept confidential.

### 3.4.2 Final Review:

Before performing the survey, it is important to read and understand the final version of the survey questioner. Consequently, after performing all the pre-testing techniques, the final version of the survey questionnaire was reviewed by the supervisor and co-supervisors of the researcher. After the final review phase has been completed, the final version of the survey questionnaire is ready for conducting the final survey. This stage results in minor corrections in the questionnaire and is fully ready for conducting the final survey.

### 3.4.3 Data Collection Process:

The Data collection process depends on the research problem's type in the current research study. For data collection, the survey method was used, as already discussed. The final survey was carried out in the various SECO-based software organizations. Data were collected from 3 March 2023 to May 2023 from various SECO-based software ecosystem organization expert. The population has been wisely chosen and restricted only to IT professionals who have proper knowledge in the SECO-based environment. The survey was distributed to 102 respondents, from which 90 responses were accurate and complete. An (88.2%) response rate was achieved in the final study.

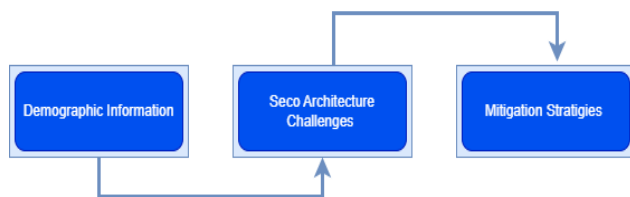


Fig.3. Survey Design

## 3.5 SURVEY RESULTS

Following are the results of the survey:

### 3.5.1 Demographic Profile of Respondents:

For the empirical examination, someone must have comprehensive details on a sample of significant descriptive statistics. Some basic information about the respondent or member and its organizational information is beneficial for effective results while filling the survey questioner. That way, demographic and organization information related to any respondent was also gathered in this research while doing survey. The above information was collecting for more meaningful and authentic results. Therefore, a total of 102 responses was used in this research in the following table respondent's demographic information, such as gender, education, SECO base experience, and position in the organization have shown.

### 3.5.2 Gender

In this study survey, the percentile of male respondents is more than the female respondents. From the total of 102 respondents, 89 (87%) were male, and 12(13%) were female who had expertise regarding the software ecosystem as shown in Table.1.

Table.1. Summary of Respondent's Demographics

| Demographics       | Respondents        | Frequency | Percentage |
|--------------------|--------------------|-----------|------------|
| Gender             | Female             | 13        | 12%        |
|                    | Male               | 89        | 89         |
| Total              |                    | 102       | 100%       |
| Position           | Developer          | 27        | 26.3%      |
|                    | Analyst            | 5         | 4.9%       |
|                    | Designer           | 33        | 32.4%      |
|                    | Team Manager       | 7         | 7.8%       |
|                    | Project manager    | 8         | 7.8%       |
|                    | Tester             | 5         | 4.9%       |
|                    | CEO                | 2         | 2%         |
|                    | Other              | 14        | 13.7       |
| Total              |                    | 102       | 100%       |
|                    | Diploma            | 3         | 2.9%       |
|                    | Bachelors          | 33        | 32.4%      |
|                    | Post-Graduate      | 47        | 46.1%      |
|                    | PhDs               | 19        | 18.6%      |
| Total              |                    | 102       | 100%       |
| Working Experience | 1-3 years          | 10        | 9.8%       |
|                    | 4-7 years          | 23        | 22%        |
|                    | 8-10 years         | 30        | 29.4       |
|                    | More than 10 years | 39        | 38.2       |
| Total              |                    | 102       | 100%       |

Position is also important to analyze the respondents' position in the Software ecosystem organization. The positions were classified into Developer, Analyst, Designer, Team Manager, Project manager, Tester, CEO, and Others (i.e., Support Engineer, IT technicians, and Director) as shown in Table 4.12. analyst (5%), Designer (33%), team manager (7%), Project manager (8%), CEOs (2%), developer (27%), another such as (support engineers and IT technicians etc. (26%).

### 3.5.3 Educating the Respondents:

Education is also another important element of the demographic profile. The respondent's education level was also recognized to assess the knowledge of everyone. In this study, out of 102 respondents, 33 (32%) were bachelor's degree holders, whereas 47 respondents had postgraduate (46%). However, 3(2%) respondents were diploma certificate holders, and PhD 19 (18%). Mostly, respondents in the current study were bachelor's degree holders.

### 3.5.4 Working Experience:

Since this research is mainly based on the Software ecosystem environment, it is essential to consider the work experience of the interviewees. Most respondents have 1 to 3 years of work experience 10y (9%). Twenty-three 23(22%) of the respondents have 4 to 7 years of work experience. Subsequently, 30 (29%) of work experience were 8 to 10 years and more than 10 years was 39(38%) respectively. Therefore, the demographic results of this study show that the respondents are in a good position. Statistics

also show that the interviewees are well-educated and have extensive work experience in SECO-based organizations.

### 3.5.5 Organization Related Information:

Understand the background of the organization from the data have been collected. The nature of the project and the number of workers in the organization are shown below. It is also essential to examine the nature of SECO-based projects and the number of workers in the organization. The research focuses on SECO-based organizations; therefore, the nature of the projects produced in these organizations has also been recognized.

### 3.5.6 Nature of Project:

In this research, the SECO architecture-based software organizations were mainly focused. Various types of projects were developed in SECO organizations. The type of projects includes web development, software development, and others (i.e., Data-ware house development, app development, telecommunication, IT infrastructure, database administration, data assurance, and game development.) as shown in Table 8. In the current study, 72(70%) of organizations in SECO were Software Architecture developer. In (21%) of projects that were developed in SECO organizations were web-based. Subsequently, other like Android development was 3% of projects in the SECO context and ISO/Apple development were 2%. "Other development like included Data-ware house development, app development, telecommunication, IT infrastructure, database administration, data assurance, and game development" (2%).

### 3.5.7 Number of Employee:

In this section, Table.2 present the number of employee and their specific field.

Table.2. Number of Employee

| Organ info          | Respondents                       | Frequency | Percentage |
|---------------------|-----------------------------------|-----------|------------|
| Nature of project   | Software Architecture Development | 72        | 70.2%      |
|                     | Web Development                   | 21        | 20.2%      |
|                     | Android development               | 4         | 3.9%       |
|                     | IOS/Apple Development             | 3         | 3.9        |
|                     | Others                            | 2         | 2%         |
|                     | Total                             | 102       | 100%       |
| Number of Employees | 5-10 employees                    | 31        | 30%        |
|                     | 10-25 employees                   | 15        | 14%        |
|                     | 26-50 employees                   | 111       | 10%        |
|                     | 51-80 employees                   | 191       | 18%        |
|                     | 80-250 employees                  | 18        | 14%        |
|                     | Above 250 employees               | 15        | 14%        |
|                     | Total                             | 361       | 100%       |

This study includes small and medium-sized organizations SECO. By estimating the number of employees in the SECO organization, the size of the organization can be clearly understood. In this research, (11%) of the organization included 80-250 employees. Whereas the 51-80 employees (8%). 20% and (26%) of employees ranged from 10-25 and 26-50. some other

like more than 250 was (14%) respectively. Complete detail of the number of employees has been presented in table 2.

## 3.6 DATA ANALYSIS APPROACH

Various data analysis tools are used to analyze and evaluate the data collected through a survey. In the current study, the tool used for statistically analyzing the data gathered through the survey was ANP Algorithm framework Jupyter notebook is used for prioritizing the issue and its mitigation. Therefore, this tool is quite appropriate for accessing the results of the data collected through a survey. For analysis for data, three data analysis approach was used, i.e., Descriptive analysis, reliability analysis, and testing hypothesis.

## 3.7 DESCRIPTIVE ANALYSIS

Descriptive analysis is used to determine and describe the respondents' demographic information. It also demonstrates the organizational-related information of the companies from where data is gathered. Various graphical, tabular, and chart representations are used to determine the descriptive frequency. The number of occurrences is also determined by giving the observed variables the frequency.

## 4. DISCUSSION

This study aimed to identify the main challenges from an architectural perspective in software ecosystems and to prioritize the corresponding mitigation strategies, in order to take steps to address them and resolve issues. The main goal of this study is to identify the challenges from an architectural perspective of software ecosystems, and the respective precautions taken to mitigate these challenges, in order to take steps towards them and solve problems. Once the Systematic Literature Review (SLR) has been conducted, the architectural challenges that have been identified and mitigation measures for them have been further investigated with respect to expert-based survey data, and analysed using the Analytic Network Process (ANP). The outcomes of the corrected ANP-MCDM tables are further discussed in this section, and the relative importance for each challenge and mitigation strategy is explained.

In this study, the use of ANP is appropriate because there are several factors of software ecosystem architecture which are linked each other. Scalability, Security, Reliability, Flexibility, Interface stability, Integration, Architectural Variability and Information sharing are not independent challenges. Rather, these elements will interact with each other and will impact the quality, sustainability and success of software ecosystem. Hence, ANP is used in order to identify a prioritised list of architectural questions and mitigation measures while exploring the interdependencies.

## 4.1 RESULT AND FINDING

In this section we have answered RQ3, where the growing impact of the software ecosystem shows that there are multiple challenges that are related to architectural issues and that need to be addressed at different levels. As shown in Table.4 and Table.6, the result of pairwise comparison and super limit matrix led to the finding that the most critical architectural challenge is the issue of Scalability of SECO Architecture. From this discovery, the

following architecture issues become paramount in the context studied: Software’s ability to grow, accommodate larger numbers of users, provide other services, manage more components, and meet expanding platform requirements. The most important priority weight was for the element of scalability (8.52%). The implication of this result here is that modern software ecosystems are constantly changing. Amongst them are many applications and APIs, third party components, cloud services, software developers and organizations. When the architecture isn’t scalable, there will be performance degradation, service bottlenecks, integration issues, sustainability concerns, and so forth. As such, scalability needs to be considered at the architectural level instead of merely as a performance concern. Security comes in second to the most important challenge, with a priority weight of 8.26%. This finding reflects the fact that security is a leading concern in software architectures. SECOs are typically complex projects where multi-party, distributed platforms, shared services, open-source elements and external libraries/lb APIs are involved. If one component fails a whole ecosystem fails. Thus, in architectural design, plat-form governance, component integration and monitoring of eco-systems, a security aspect must be taken into account. The third ranked challenge is Flexibility in Architecture (8.13%) Priority Weight. Flexibility is key since software ecosystems are dynamic and always evolving. Requirements, technologies, services, stakeholders and business models could change over time. For the ecosystem, this flexibility is important because it avoids having to be completely redesigned as changes occur. This finding is ratifying the need for architectural flexibility towards longer-term SECO evolution and sustainability. SECO Architecture was rated as being fairly reliable and had a priority weight of 8.00%. Reliability is important because it assumes that ecosystem participants require robust and reliable services, consistent behavior of participants on the platforms and consistency of the availability of software components. Low reliability can lead to service failure, loss of trust, less user satisfaction and poor performance in the ecosystem. Thus, reliability becomes an indispensable feature for SECO design and management, and should be taken into account as a fundamental architectural requirement. The fifth, with the lowest priority weight was Interface Stability (7.55%). Stable interfaces are essential because they facilitate software systems to communicate with each other, software components, software services, APIs, and third-party applications. Application integration problems, interface compatibility issues, and maintenance challenge can be introduced due to frequent interface changes. Thus, stable interfaces contribute to stability and facilitate the coordination among ecosystem actors. SECO further challenges architecture challenges. Two other ratings had high priority weights: Functionality of SECO Architecture with 6.99% and Longevity of SECO Architecture with 6.99%. The other categories were Integration Applications with 6.74%, Architectural Variability 6.65%, Decentralized Development 6.56%, Information Sharing 6.31%, Independent Platform Development 6.28% and Removing Obsolete Code 5.83%.

While less significant for the odd metrics of scalability, security, flexibility, reliability, and UI stability, these challenges are still important and contribute to and support architectural maintainability, ecosystem co-ordination, and long-term soft-ware quality. The results of the correction show that Scalability,

Security, Flexibility, Reliability and Interface Stability are the most influential architectural challenges in SECO. These challenges should be considered by the researchers, software architects or practitioners in designing, maintaining and improving software ecosystem architectures.

4.2 RESULT OF ANP MCDM

In this section we have discussed the implementation of ANP algorithm which prioritize different software ecosystem Architecture issues and their mitigation strategies.

4.2.1 ANP Algorithm Implementation:

Following are the steps according to the algorithm of ANP as shown in the following Tables. Similarly, we can do pairwise comparison with respect to SECO Architecture issue and their mitigation strategies. Jupiter framework is used to prioritize and find the most and least impact of every risk and its mitigation. Similarly, we can do pairwise comparison with respect to Architecture issues sharing information, Architecture Stability, Integration application, Removing Obsolete Code Architecture Security, Architecture Reliability, Functionality, Independency Architecture, Architecture Flexibility, Decentralized Architecture Longevity, Solidity and Scalability), and its mitigation (Variable Design, Custom Design, Assessment of ecosystem health, Metrics, Framework and Sandbox). According to algorithm after pairwise comparisons, we can get Limit super matrix to get final priorities, table 9 is shows generalized super matrix while Table.4 shows Pairwise comparison of SECO Architecture Challenges, furthermore Table.5 repented Level of Mitigation of each Strategy w.r.t. SECO Architecture Challenges and table 6 shows Level of SECO Architecture Challenges Super limit Matrix of challenges and finally Table.7 represented prioritized matrix for SECO Architecture challenges .

Table.3. Generalized super matrix

|             | Goal     | Criteria | Alternative |
|-------------|----------|----------|-------------|
| Goal        | 1        | weigh22  | 0           |
| Criteria    | weight22 | 1        | 0           |
| Alternative | 0        | weight21 | 1           |

4.2.2 Structural Diagram:

Following diagram show goal criteria and alternative

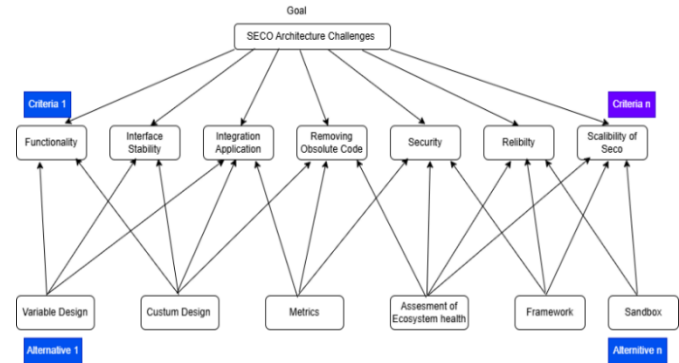


Fig.4. Structural diagram

### 4.3 WEIGHTS MATRIX

#### 4.3.1 Criteria Weights Matrix:

The weights for the criteria are represented in the following matrix:

$$\text{Criteria} = \begin{bmatrix} 1 & \text{weight}_{22} & 0 \\ \text{weight}_{22} & 1 & 0 \\ 0 & \text{weight} & 1 \end{bmatrix}$$

#### 4.3.2 Alternatives Weights Matrix:

The weights for the alternatives are represented in the following matrix:

Weight alternative

$$= \begin{bmatrix} 1 & 0.889246 & 0.896440 & 0.844512 & 0.737683 & 0.977072 \\ 1.124549 & 1 & 1.008091 & 0.949695 & 0.829561 & 1.098765 \\ 1.115523 & 0.991974 & 1 & 0.942073 & 0.822903 & 1.089947 \\ 1.184116 & 1.052970 & 1.061489 & 1 & 0.873502 & 1.156966 \\ 1.355596 & 1.205457 & 1.215210 & 1.144817 & 1 & 1.324515 \\ 1.023466 & 0.910111 & 0.917476 & 0.88662 & 0.864329 & 1 \end{bmatrix}$$

#### 4.3.3 Overall Weights Matrix:

The overall weights matrix is computed as follows:

$$W_{\text{overall}} = W_{\text{criteria}} \times W_{\text{alternatives}}$$

The resultant overall weights matrix is:

$$W_{\text{overall}} = \begin{bmatrix} 1 & \text{weight}_{22} & 0 \\ \text{overall}_{\text{weight}_{22}^{-1}} & 1 & 0 \\ 0 & \text{overall}_{\text{weight}_{21}} & 1 \end{bmatrix}$$

#### 4.3.4 Consistency Ratio (CR) and Consistency Index (CI):

The Consistency Ratio (CR) and Consistency Index (CI) are made as per the Saaty table. Ideally, the values ought to be less than 0.1 to have a uniform matrix.

$$CI = \frac{\lambda_{\text{max}} - n}{n - 1}; \text{ While } CI=1$$

where  $n$  is the order of the matrix.

#### 4.3.5 Unweighted Super Matrix:

The unweighted super matrix is constructed as follows:

$$\text{Sunweighted} = \begin{bmatrix} \text{Risks Matrix} & 0 \\ 0 & \text{Mitigation matrix} \end{bmatrix}$$

#### 4.3.6 Weighted Super Matrix:

The weighted super matrix is obtained by multiplying the unweighted super matrix with the overall weights matrix:

$$S_{\text{weighted}} = W_{\text{overall}} \times \text{Sunweighted}$$

#### 4.3.7 Limit Matrix:

The limit metric is  $L = \lim_{n \rightarrow \infty} (S_{\text{weighted}})^k$ . In the limit matrix, the alternatives' values in each row should be the same.

#### 4.3.8 Pairwise Comparisons Process:

In order to assess the software ecosystem architecture criterion, we use the Analytic Network Process (ANP) which is a

powerful approach that allows us to model complicated relationships and dependencies. This process of pairwise comparisons is important towards coming up with priority weights of each criterion.

### 4.4 PAIRWISE COMPARISONS MATRIX FOR CRITERIA

The initial one is the formation of a matrix of the pairwise comparisons of every criterion.  $C$  is a set of criteria and,  $n$  is the number of criteria.  $A^{(c)}$  is a matrix that is obtained after comparing each of the criteria  $m_i$  to each other criterion  $c_j$  by the use of the Saaty scale:

$$A^{(c)} = \begin{bmatrix} 1 & a_{11} & \cdots & a_{1n} \\ \frac{1}{a_{12}} & 1 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{a_{1n}} & \frac{1}{a_{2n}} & \cdots & 1 \end{bmatrix}$$

where  $a_{ij}$  indicates the value of comparison between the pair of  $c_i$  and  $c_j$ . The values are given by participants in the decision-making process depending on their opinion.

#### 4.4.1 Calculating Priority Weights:

After getting the pairwise comparisons matrices of criteria, we then get the priority weights of criteria.  $W(c)$  is the priority weight vector of criteria and  $\lambda(c)$  is the consistency index. The weighted priorities are calculated in the following way:

$$w(c) = \frac{1}{n} \sum_{j=0}^n \left( \prod_{i=1}^n \lambda(c) \right)^{1/n}$$

where  $\lambda(c)$  is the consistency index, and  $n$  is the number of criteria.

#### 4.4.2 Consistency Ratio (CR) Calculation:

The consistency ratio (CR) is a measure of how consistent the judgments are. It is calculated using the formula:

$$CR(c) = \frac{\lambda(c) - n}{n - 1}$$

Close to zero is considered acceptable consistency.

#### 4.4.3 Pairwise Comparisons Matrix for Alternatives:

We form pairwise comparison matrices of alternatives as in the case of criteria.

$$A^{(a)} = \begin{bmatrix} 1 & a_{12} & \cdots & a_{1m} \\ \frac{1}{a_{12}} & 1 & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{a_{1m}} & \frac{1}{a_{2m}} & \cdots & 1 \end{bmatrix}$$

where  $A^{(a)}$  represents the alternative matrix and  $m$  is the number of alternatives.

#### 4.4.4 Calculating Priority Weights for Alternatives:

Likewise, priority weight of the alternatives  $W(a)$  and consistency index  $\lambda(a)$  are determined.

$$w(a) = \frac{1}{m} \sum_{j=1}^m \left( \prod_{i=1}^m a_{ij} / \lambda(a) \right)^{1/m}$$

$$CR(a) = \frac{\lambda(a) - m}{m - 1}$$

#### 4.4.5 Unweighted Super Matrix Construction:

Using the priority weights, we randomly form the un-weighted super matrix  $A(w)$  which is a combination of the criteria and alternatives.

$$A(w) = \begin{bmatrix} A^C & 0 \\ 0 & A^{(a)-1} \end{bmatrix}$$

#### 4.4.6 Weighted Super Matrix Construction:

The weighted super matrix  $A(w^0)$  is the multi-ply result of  $A(w)$  multiplied by the priority weight vector of criteria  $W(c)$ .

$$A(w^r) = \begin{bmatrix} w^C \cdot A^C & 0 \\ 0 & w^a \cdot (A^{(a)})^{-1} \end{bmatrix}$$

#### 4.4.7 Limit Matrix:

Lastly, the limit matrix  $L$  is created whereby the values within any row in the alternatives are equal.

$$L = \begin{bmatrix} L_{11} & \cdots & L_{1m} \\ \vdots & \ddots & \vdots \\ L_{n1} & \cdots & L_{nm} \end{bmatrix}$$

where each entry  $L_{ij}$  (where  $i=1, \dots, n_i = 1, \dots, n$  and  $j=1, \dots, m_j = 1, \dots, m$ ) is given by:

$$L_{ij} = \frac{W_i^{(c)} \cdot A_{ij}^{(c)}}{W_j^{(a)} \cdot A_{ij-1}^{(a)}}$$

The subsequent parts will explore the findings and discussion based on this comprehensive ANP based analysis.

### 4.5 CONSISTENCY RATIO (CR) CALCULATION

In order to warrant reliability of the pairwise comparison made in the process of ANP, we compute the Consistency Ratio (CR). The CR also assists in determining the consistency of the judgments given by the participants.

#### 4.5.1 Criteria Consistency Ratio (CRc):

For the criteria, the Consistency Ratio (CRc) is calculated using the following steps:

#### 4.5.2 Pairwise Comparisons Matrix for Criteria:

$C$  will denote the set of criteria and  $n$  will denote the number of criteria.  $A(c)$  is a pairwise comparisons matrix constructed according to Saaty scale:

$$A_c = \begin{bmatrix} 1 & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ 1/a_{1n} & \cdots & 1 \end{bmatrix}$$

#### 4.5.3 Calculating Priority Weights:

The weights of primacy of criteria  $W(c)$  are determined to be:

$$w(c) = \frac{1}{n} \sum_{j=1}^n \left( \prod_{i=1}^n a_{ij} / \lambda(c) \right)^{1/n}$$

$\lambda(c)$  = The index of consistency.

#### 4.5.4 Consistency Index (CIc):

The formula used to calculate The Consistency Index (CIc) is given below:

$$CI(c) = \frac{\lambda(c) - n}{n - 1}$$

#### 4.5.5 Random Index (RI):

The order of the matrix is a constant (ex: in a 3x3 matrix, RI = 0.58). The Consistency Ratio (CRc) is calculated using it:

RI = Matrix order lookup value.

#### 4.5.6 Consistency Ratio (CR):

$$\text{The Consistency Ratio (CRc)} = CR(c) = \frac{CI(c)}{RI}$$

When the CRc value approaches zero this shows satisfactory consistency.

#### 4.5.7 Alternatives Consistency Ratio (CRa):

On the same note, in case of the alternatives, it is done through establishing pair-wise comparisons matrices, calculation of priorities weights, consistency index, random index and lastly, the Consistency Ratio (CRa). The strict method also means the judgments that are presented during the pairwise comparisons are held constant making the ANP-based decision making process more reliable.

### 4.6 SUPER MATRIX CONSTRUCTION

The aggregation of the criteria and alternative information, which results in the Super Matrix, illustrates the weighted relationships among all elements. Its construction follows a number of steps:

#### 4.6.1 Unweighted Super Matrix:

The Unweighted Super Matrix  $S(u)$  is developed as a result of combining the pairwise comparisons matrices relating to criteria  $A(c)$  and alternatives  $A(a)$ , as well as their respective weights  $W(c)$  and  $W(a)$ :

$$Su = \begin{bmatrix} 0 & w^C \cdot A^C \\ w^a \cdot (A^{(a)})^{-1} & A^a \end{bmatrix}$$

where,  $W^{(c)}$  and  $W^{(a)}$  represent column vectors with weights of criteria and alternatives respectively.

#### 4.6.2 Weighted Super Matrix:

To create the Weighted Super Matrix  $S(w)$ , we add the priority weights of the criteria  $W(c)$  to the Un-weighted Super Matrix:

$$Sw = \begin{bmatrix} 0 & w^C \cdot A^C \\ w^a \cdot (A^{(a)})^{-1} & w^C \cdot A^C + w^a \cdot A^a \end{bmatrix}$$

#### 4.6.3 Limit Matrix:

Limit Matrix  $L$  is derived by normalizing the Super Matrix of weighted:

$$L = \frac{S^u}{\text{Column sum of } S^w}$$

The limit matrix thus produced is an important factor in determining the total ranking and priority weighting. The matrices  $S(u)$ ,  $S(w)$ , and  $L$  reflect the dynamic conditions of information integration during the ANP process. The relationships, priorities, and weights are encapsulated in these matrices and serve as the basis for further analysis and decision-making.

#### 4.7 LIMIT MATRIX

Limit Matrix, also referred to as  $L$ , is an important result in the Analytic Network Process (ANP) methodology. It is based on the Weighted Super Matrix and is widely used to determine overall rankings and priority weights.

We construct the Limit Matrix by normalizing the Weighted Super Matrix ( $S(w)$ ):

$$L = S^{(w)} / \text{Column Sums of } S^{(w)}$$

The limit matrix  $L$  is structure as follows

$$L = \begin{bmatrix} 0 & \cdots & L_{1,n} \\ \vdots & \ddots & \vdots \\ L_{n,1} & \cdots & 0 \end{bmatrix}$$

where  $l_{ij}$  is the value at the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the Limit Matrix, which in the Analytic Network Process (ANP) represents the long-term influence of one element on another.

It is important to ensure that the values of the alternatives in each row are identical, meaning that each alternative has been evaluated equally by the criteria. Such homogeneity is one of the main features of the ANP methodology, providing a balanced display of the decision's elements. Once the Limit Matrix has been built, the research check that indeed the values of the alternatives in rows are all equal. Any inconsistency beyond this homogeneity may reflect the decision model's inconsistencies. The Limit Matrix can serve as a basis for further analyses, such as determining the overall rankings and the priority weights.

#### 4.8 ALGORITHM

There are four steps used in ANP approach

Step 1: construction of model and structuring of problem: The problem must be clearly specified and decompose into rational system in the form of network.

Step 2: Pair wise comparisons and priority vectors: Comparison of the elements of the decision at each cluster level is done using a control criterion. The control criteria concept is a key element of the ANP approach. The influence  $e$  of one element on the other

is represented by the use of the Eigen vector. The relative importance can be calculated using Satya's scale.

Step 3: Super matrix formation: Markov chain processes and super matrix designs are not different. Global priorities with independent influence may be secured by placing the local priority vectors in suitable columns of the matrix. Each matrix represents the cluster relationships within a super matrix.

Step 4: Synthesis of the criteria and alternatives: Weight of priorities and the best alternatives: The weights of the criteria and the best alternatives could be obtained in the normalized super matrix.

#### 4.9 JUPYTER NOTEBOOK

Jupyter notebook is an open-source web app that enables the creation and sharing of documents composed of live code, equations, visualizations, and explanatory text. It belongs to the list of the most popular data science and scientific computer tools. Jupyter notebook provides a writing and execution experience in code cells, enabling you to write and run code. Code can be written in languages such as Python, R, and Julia. Each cell can have the code. You can run one or the whole notebook, and it is not hard to experiment, iterate, and analyses data.

##### 4.9.1 Key features and Advantages of Jupyter Notebook:

Interactive Computing: In jupyter notebook, a code cell is executable by default, i.e., you can execute a code cell and simultaneously see the output. Code and Text: Colored and Visible Mixtures: In the case of the Jupyter notebook, one may combine both code, equations, visualizations, and explanatory text in the same document. This is why it may be regarded as a strong collection of reports, tutorials, and data-based presentations quite easily. Visualization and Plotting: Jupyter notebook can work with data visualization libraries such as Matplotlib, Seaborn and Plotly. You can create interactive plots, charts, and graphs in your notebook. Data Exploration and Analysis: Jupyter notebook is a location where an individual can explore and analyze data. With libraries such as Pandas, NumPy, and SciPy, data manipulation and transformation are performed. Sharing and Collaboration: Sharing the Jupyter notebook files with other participants is easy, making it easy to collaborate with other members and exchange project and research results. It is possible to store notebooks as files that can be carried around, or access them via platforms such as GitHub, JupyterHub or Jupyter Notebook Viewer. Extensions and Widgets: The Jupyter notebook extensions are add-ons that do not work extremely well. You can also customize your notebook space, allowing you to add custom widgets, interactive components, and extensions. Jupyter notebook : This is a server that can be configured as either a local or remote server and can also be accessed via a web browser. It has an end-user-friendly interface in notebook creation, editing, and execution. As mentioned in sections 4.3.8 and 4.4, the Jupyter notebook can be installed using the Python package manager, pip, or the popular Python distribution, Anaconda, which includes Jupyter notebook.

Table.4. Pairwise comparison of SECO Architecture Challenges

| Risks                            | Information Sharing | Interface Stability | Integration Applications | Removing Obsolete Code | Security | Reliability of SECO | Functionality of SECO | Independent Platform Development | Flexibility in Architecture | Decentralized Development | Longevity of SECO | Architectural Variability | Solidity of Architecture | Scalability of SECO | EV    |
|----------------------------------|---------------------|---------------------|--------------------------|------------------------|----------|---------------------|-----------------------|----------------------------------|-----------------------------|---------------------------|-------------------|---------------------------|--------------------------|---------------------|-------|
| Information Sharing              | 1                   | 0.835561            | 0.93668                  | 1.081315               | 0.763942 | 0.788519            | 0.902201              | 1.003814                         | 0.776036                    | 0.961169                  | 0.902201          | 0.948767                  | 0.877963                 | 0.74085             | 6.31% |
| Interface Stability              | 1.1968              | 1                   | 1.121019                 | 1.294118               | 0.914286 | 0.9437              | 1.079755              | 1.201365                         | 0.92876                     | 1.150327                  | 1.079755          | 1.135484                  | 1.050746                 | 0.88665             | 7.55% |
| Integration Applications         | 1.0676              | 0.892045            | 1                        | 1.154412               | 0.815584 | 0.841823            | 0.96319               | 1.071672                         | 0.828496                    | 1.026144                  | 0.96319           | 1.012903                  | 0.937313                 | 0.790932            | 6.74% |
| Removing Obsolete Code           | 0.9248              | 0.772727            | 0.866242                 | 1                      | 0.706494 | 0.729223            | 0.834356              | 0.928328                         | 0.717678                    | 0.888889                  | 0.834356          | 0.877419                  | 0.81194                  | 0.685139            | 5.83% |
| Security                         | 1.309               | 1.09375             | 1.226115                 | 1.415441               | 1        | 1.032172            | 1.180982              | 1.313993                         | 1.015831                    | 1.25817                   | 1.180982          | 1.241935                  | 1.149254                 | 0.969773            | 8.26% |
| Reliability of SECO              | 1.2682              | 1.059659            | 1.187898                 | 1.371324               | 0.968831 | 1                   | 1.144172              | 1.273038                         | 0.984169                    | 1.218954                  | 1.144172          | 1.203226                  | 1.113433                 | 0.939547            | 8.00% |
| Functionality of SECO            | 1.1084              | 0.926136            | 1.038217                 | 1.198529               | 0.846753 | 0.873995            | 1                     | 1.112628                         | 0.860158                    | 1.065359                  | 1                 | 1.051613                  | 0.973134                 | 0.821159            | 6.99% |
| Independent Platform Development | 0.9962              | 0.832386            | 0.933121                 | 1.077206               | 0.761039 | 0.785523            | 0.898773              | 1                                | 0.773087                    | 0.957516                  | 0.898773          | 0.945161                  | 0.874627                 | 0.738035            | 6.28% |
| Flexibility in Architecture      | 1.2886              | 1.076705            | 1.207006                 | 1.393382               | 0.984416 | 1.016086            | 1.162577              | 1.293515                         | 1                           | 1.238562                  | 1.162577          | 1.222581                  | 1.131343                 | 0.95466             | 8.13% |
| Decentralized Development        | 1.0404              | 0.869318            | 0.974522                 | 1.125                  | 0.794805 | 0.820375            | 0.93865               | 1.044369                         | 0.807388                    | 1                         | 0.93865           | 0.987097                  | 0.913433                 | 0.770781            | 6.56% |
| Longevity of SECO                | 1.1084              | 0.926136            | 1.038217                 | 1.198529               | 0.846753 | 0.873995            | 1                     | 1.112628                         | 0.860158                    | 1.065359                  | 1                 | 1.051613                  | 0.973134                 | 0.821159            | 6.99% |
| Architectural Variability        | 1.054               | 0.880682            | 0.987261                 | 1.139706               | 0.805195 | 0.831099            | 0.95092               | 1.05802                          | 0.817942                    | 1.013072                  | 0.95092           | 1                         | 0.925373                 | 0.780856            | 6.65% |
| Solidity of SECO                 | 1.139               | 0.951705            | 1.066879                 | 1.231618               | 0.87013  | 0.898123            | 1.027607              | 1.143345                         | 0.883905                    | 1.094771                  | 1.027607          | 1.080645                  | 1                        | 0.843829            | 7.19% |
| Scalability of SECO              | 1.3498              | 1.127841            | 1.264331                 | 1.459559               | 1.031169 | 1.064343            | 1.217791              | 1.354949                         | 1.047493                    | 1.297386                  | 1.217791          | 1.280645                  | 1.185075                 | 1                   | 8.52% |

Table.5. Level of Mitigation of each Strategy w.r.t. SECO Architecture Challenges

| Mitigation                     | Variable Design | Custom Design | Assessment of ecosystem health | Metrics  | Framework | Sandbox  | Weights |
|--------------------------------|-----------------|---------------|--------------------------------|----------|-----------|----------|---------|
| Variable Design                | 1               | 0.889246      | 0.896440                       | 0.844512 | 0.737683  | 0.977072 | 14.70%  |
| Custom Design                  | 1.124549        | 1             | 1.008091                       | 0.949695 | 0.829561  | 1.098765 | 16.53%  |
| Assessment of ecosystem health | 1.115523        | 0.991974      | 1                              | 0.942073 | 0.822903  | 1.089947 | 16.40%  |
| Metrics                        | 1.184116        | 1.052970      | 1.061489                       | 1        | 0.873502  | 1.156966 | 17.41%  |
| Framework                      | 1.355596        | 1.205457      | 1.215210                       | 1.144817 | 1         | 1.324515 | 19.93%  |
| Sandbox                        | 1.023466        | 0.910112      | 0.917476                       | 0.864329 | 0.754993  | 1        | 15.04%  |

Table.6. Level of SECO Architecture Challenges Super Limit

| Risk                             | Information Sharing | Interface Stability | Integration Applications | Removing Obsolete Code | Security | Reliability of SECO | Functionality of SECO | Independent Platform Development | Flexibility in Architecture | Decentralized Development | Longevity of SECO | Architectural Variability | Solidity of SECO | Scalability of SECO | Weight |
|----------------------------------|---------------------|---------------------|--------------------------|------------------------|----------|---------------------|-----------------------|----------------------------------|-----------------------------|---------------------------|-------------------|---------------------------|------------------|---------------------|--------|
| Information Sharing              | 1                   | 0.835561            | 0.93668                  | 1.081315               | 0.763942 | 0.788519            | 0.902201              | 1.003814                         | 0.776036                    | 0.961169                  | 0.902201          | 0.948767                  | 0.877963         | 0.74085             | 6.31%  |
| Interface Stability              | 1.1968              | 1                   | 1.121019                 | 1.294118               | 0.914286 | 0.9437              | 1.079755              | 1.201365                         | 0.92876                     | 1.150327                  | 1.079755          | 1.135484                  | 1.050746         | 0.88665             | 7.55%  |
| Integration Applications         | 1.0676              | 0.892045            | 1                        | 1.154412               | 0.815584 | 0.841823            | 0.96319               | 1.071672                         | 0.828496                    | 1.026144                  | 0.96319           | 1.012903                  | 0.937313         | 0.790932            | 6.74%  |
| Removing Obsolete Code           | 0.9248              | 0.772727            | 0.866242                 | 1                      | 0.706494 | 0.729223            | 0.834356              | 0.928328                         | 0.717678                    | 0.888889                  | 0.834356          | 0.877419                  | 0.81194          | 0.685139            | 5.83%  |
| Security                         | 1.309               | 1.09375             | 1.226115                 | 1.415441               | 1        | 1.032172            | 1.180982              | 1.313993                         | 1.015831                    | 1.25817                   | 1.180982          | 1.241935                  | 1.149254         | 0.969773            | 8.26%  |
| Reliability of SECO              | 1.2682              | 1.059659            | 1.187898                 | 1.371324               | 0.968831 | 1                   | 1.144172              | 1.273038                         | 0.984169                    | 1.218954                  | 1.144172          | 1.203226                  | 1.113433         | 0.939547            | 8.00%  |
| Functionality of SECO            | 1.1084              | 0.926136            | 1.038217                 | 1.198529               | 0.846753 | 0.873995            | 1                     | 1.112628                         | 0.860158                    | 1.065359                  | 1                 | 1.051613                  | 0.973134         | 0.821159            | 6.99%  |
| Independent Platform Development | 0.9962              | 0.832386            | 0.933121                 | 1.077206               | 0.761039 | 0.785523            | 0.898773              | 1                                | 0.773087                    | 0.957516                  | 0.898773          | 0.945161                  | 0.874627         | 0.738035            | 6.28%  |
| Flexibility in Architecture      | 1.2886              | 1.076705            | 1.207006                 | 1.393382               | 0.984416 | 1.016086            | 1.162577              | 1.293515                         | 1                           | 1.238562                  | 1.162577          | 1.222581                  | 1.131343         | 0.95466             | 8.13%  |
| Decentralized Development        | 1.0404              | 0.869318            | 0.974522                 | 1.125                  | 0.794805 | 0.820375            | 0.93865               | 1.044369                         | 0.807388                    | 1                         | 0.93865           | 0.987097                  | 0.913433         | 0.770781            | 6.56%  |
| Longevity of SECO                | 1.1084              | 0.926136            | 1.038217                 | 1.198529               | 0.846753 | 0.873995            | 1                     | 1.112628                         | 0.860158                    | 1.065359                  | 1                 | 1.051613                  | 0.973134         | 0.821159            | 6.99%  |
| Architectural Variability        | 1.054               | 0.880682            | 0.987261                 | 1.139706               | 0.805195 | 0.831099            | 0.95092               | 1.05802                          | 0.817942                    | 1.013072                  | 0.95092           | 1                         | 0.925373         | 0.780856            | 6.65%  |
| Solidity of SECO                 | 1.139               | 0.951705            | 1.066879                 | 1.231618               | 0.87013  | 0.898123            | 1.027607              | 1.143345                         | 0.883905                    | 1.094771                  | 1.027607          | 1.080645                  | 1                | 0.843829            | 7.19%  |
| Scalability of SECO              | 1.3498              | 1.127841            | 1.264331                 | 1.459559               | 1.031169 | 1.064343            | 1.217791              | 1.354949                         | 1.047493                    | 1.297386                  | 1.217791          | 1.280645                  | 1.185075         | 1                   | 8.52%  |

Table.7. Prioritize Order of SECO Architecture issue mitigation

| Sl. No | Mitigation                     | Relative Weights |
|--------|--------------------------------|------------------|
| 1      | Framework                      | 1.201677         |
| 2      | Metrics                        | 1.049667         |
| 3      | Custom Design                  | 0.996864         |
| 4      | Assessment of Ecosystem health | 0.988863         |
| 5      | Sandbox                        | 0.907258         |
| 6      | Variable Design                | 0.886457         |

Super limit matrix for architectural challenges has been discussed in section 4.7, and Table.6 highlights the architectural super limit matrix.

This section provides a detailed analysis of the results for reported risks and their corresponding mitigation strategies. SEC Architecture MCDM tools (ANP Algorithm, Jupyter notebook framework) have been used extensively to produce results, reducing the need for extensive human intervention. Pairwise comparisons of reported risks and their mitigation strategies are used to determine relative impact using the survey approach, and the data are then prioritized using the ANP algorithm.

The ANP-MCDM outputs for prioritizing the architectural challenges and their mitigation strategies for the SECO are presented in this section. The tables were corrected based on data from the survey and arranged using the pairwise comparison method. The comparison values indicate which factor is more important than another. The value of 1 represents equality, values >1 represent that the row factor is more important relative to the column factor, and values <1 indicate the row factor has less importance relative to the column factor.

The pairwise comparison of SECO architecture challenges has been corrected and shown in Table.4. The results show that Scalability of SECO Architecture is the highest priority and so is also the most important challenge in the present study. This discovery aligns with the characteristics of the modern software ecosystems, which grow continuously, add more and more users, add more and more services and rely on more and more platforms, exerting a strong pressure on architecture. Without proper scalability management, other problems like poor reliability, weak performance, integration bottlenecks, and decreased sustainability may also manifest.

The corrected SECO Architecture Challenges Super Limit Matrix is shown in Table 6. The weight column on this table is a verification of the final priority ranking of the architectural challenges. Based on the corrected values, the most important value is Scalability of SECO Architecture with 8.52%, followed by Security with 8.26%, Flexibility in Architecture with 8.13%, Reliability of SECO Architecture with 8.00%, and finally, Interface Stability with 7.55%. The results indicate that the most important issues in SECO architecture are architectural growth, protection, adaptability, dependability, and stable interaction.

It also reveals that other factors that affect SECO architecture include Solidity of SECO Architecture, Functionality of SECO Architecture, Longevity of SECO Architecture, Integration Applications, Architectural Variability, Decentralized Development, Information Sharing, Independent Platform Development, and Removing Obsolete Code. These factors may not have been given the highest priority weights, but they do play

a supporting role in the overall function and sustainability of the ecosystem. Thus, they should not be neglected during the actual architectural design process. Table 5 shows the revised mitigation strategy matrix. Variable Design, Custom Design, Assessment of Ecosystem Health, Metrics, Framework, and Sandbox are all considered mitigation strategies in this study. This table is intended to compare the mitigation strategies and determine which strategy is more important in relation to addressing SECO architecture challenges.

The corrected mitigation results show that Framework is the most important mitigation strategy. A framework-based approach offers architectural guidance, reuse of practices, governance and standard procedures in solving SECO architectural challenges. It facilitates systematic scalability, security, reliability, flexibility and interface stability in software architecture. Thus, it is important to understand that structured architectural support is needed to enhance software ecosystem sustainability and quality, as this is strongly reflected in the high ranking of Framework.

- **Metrics:** Metrics are significant since they enable measurable measures to assess architectural quality, scalability, security, reliability, maintainability and ecosystem health. If the architecture is not measurable, it is challenging to understand if it is improving or if a mitigation approach is working. Thus, evaluating the architecture based on the measurement will help to make decisions based on evidence.
- **Custom Design:** This outcome indicates that software ecosystems with context-specific architectural solutions are needed. All ecosystems have different SECOs with varying platforms, stakeholders, technologies, organizational needs and business objectives, so it is important to understand that one solution might not be appropriate for all ecosystems. With custom design, software architects can design solutions based on the ecosystem's needs.
- **Assessment of Ecosystem Health:** This approach is useful for assessing the maturity, sustainability, performance and stability of the software ecosystem. It's not the top one, but it does come in handy if you want to monitor it and continuously improve it. Ecosystem health assessment can be used to help the decision makers to discover weak areas and then to make architectural plans for the future.
- **Sandbox:** Sandboxing can be used to test, isolate and validate components until they are fully integrated in the ecosystem. By using this approach, it is possible to mitigate security, integration, and reliability risks as new parts or services are introduced. The corrected results, however, indicate that sandboxing is more of a mitigating measure and not the main one.
- **Variable Design:** It is the least important relative weight in the corrected Table 7, but it still comes in handy for supporting architectural alternatives and design flexibility. However, variable design may be useful to architects when addressing varying needs in their ecosystems, and the relative importance of the variable when compared to Framework, Metrics, Custom Design, Assessment of Ecosystem Health, and Sand-box is not very high.

The prioritized order of SECO architecture issue mitigation strategies is shown in Table 7, based on the relative weights of the

algorithms. The corrected Table 7 indicates that Framework is the most significant with a relative weight of 1.201677, while the most insignificant is Variable Design with relative weight of 0.886457. This ranking reinforces the fact that the best way to mitigate is through using Framework, and Metrics and Custom Design are also valuable ways to solve architectural issues.

Overall, the corrected ANP-MCDM results give a clear prioritization of SECO architecture challenges and mitigation. The results show that in terms of architectural scalability, it is the most important factor to be paid attention to, followed by security, flexibility, reliability, and interface stability. For mitigation, architectural guidance, based on framework should be emphasized, complemented by metric-based assessment and custom architectural design. The results offer practical tips for software ecosystem architects, researchers, and organizations to better govern software ecosystems, minimize software risks, and increase the sustainability of software ecosystems.

## 5. CONCLUSION AND FUTURE WORK

The architecture of software ecosystems is a critical aspect to the success, sustainability and long-term evolution of modern software systems. The priority of the identified key challenges in the SECO architectural space was done by Analytic Network Process, based on a Systematic Literature Review done in this study. It is seen from the corrected ANP-MCDM results that the most critical architectural challenge is ‘Scalability of SECO Architecture’ followed by ‘Security’, ‘Flexibility in Architecture’, ‘Reliability of SECO Architecture’ and ‘Inter-face Stability’. These findings suggest the need for software ecosystems to have architectures that can accommodate growth, secure distributed entities, cope with change, ensure dependable service provision and guarantee stable interfaces between ecosystem actors.

The results of the mitigation strategy showed that Framework is the most effective mitigation strategy, followed by Metrics, Custom Design, Assessment of Ecosystem Health, Sandbox, and Variable Design. It implies that SECO architectural challenges can be addressed more effectively in the context of structured frameworks, measurable architectural indicators, and context-specific design practices. A framework-based mitigation strategy can provide the back bone for architectural governance, design uniformity, coordination of components, and long-term planning.

The results of this study are important for research and practice. The study is of value to researchers as it includes a quantitative prioritization of SECO architectural challenges and mitigation strategies based on ANP. The findings can be of value to practitioners as a way to identify issues that ought to be addressed and mitigation strategies that could offer greater support. The study also emphasizes the need for systematic decision making approaches in software ecosystem architecture since architectural challenges are interdependent and simple ranking approaches do not adequately evaluate the challenges.

The research can be further extended in future by applying the model of ANP to the greater data set and other software ecosystem domains. There could be more quantitative tools to validate results as well. Further research can also be conducted to provide support for the ANP-based prior-ization using automated tools; given that the ANP process involves multiple comparisons and

becomes complex with the number of criteria and alternatives. Building on this, further empirical research with industry experts could provide further evidence to the generalizability of the results and to support the creation of more reliable software ecosystem architecture mitigations.

## REFERENCES

- [1] R. Rajab and M. Alnoukari, “DevOps Integration with Capability Model Maturity Integration: A Systematic Mapping Review”, *IEEE Access*, Vol. 13, pp. 1-12, 2025.
- [2] G. Yaya, Soureya, Amougou, Ngoumou, Ngossaha and M. Justin, “Adaptive Software Development: A Comprehensive Framework Integrating Artificial Intelligence for Sustainable Evolution”, *International Arab Journal of Information Technology*, Vol. 22, No. 2, pp. 45-58, 2025.
- [3] E. Onagh and M. Nayebe, “Extension Decisions in Open Source Software Ecosystem”, *Journal of Systems and Software*, Vol. 212, pp. 1-7, 2025.
- [4] K. Saarni, M. Raatikainen and S. Hyrnsalmi, “Failures in Software Ecosystems: A Systematic Literature Review”, *Proceedings of International Conference on Engineering, Technology and Innovation*, Vol. 8, pp. 1-8, 2025.
- [5] A. Raj and R. Deora, “AI and ML Powered Feature Prioritization in Software Product Development”, *International Journal of Data Mining and Knowledge Management Process*, Vol. 15, No. 1, pp. 23-30, 2025.
- [6] S. Hegazy, “Overcoming Experimentation Challenges in Software Ecosystems of Large Product and Service Organizations: A Participatory Action Research Study”, *Journal of Systems and Software*, Vol. 213, pp. 1-7, 2025.
- [7] J. Sun, “Collaboration Challenges and Opportunities in Developing Scientific Open-Source Software Ecosystem: A Case Study on Astropy”, *Proceedings of the ACM on Human-Computer Interaction*, Vol. 9, No. 7, pp. 1-33, 2025.
- [8] Z. Yang, “Ecosystem of Large Language Models for Code”, *ACM Transactions on Software Engineering and Methodology*, Vol. 35, No. 1, pp. 1-30, 2025.
- [9] R. Feitosa, “Actionable Framework for Understanding and Improving Social and Human Factors that Influence the Requirements Management in Software Ecosystems”, *ACM Transactions on Software Engineering and Methodology*, Vol. 36, No. 2, pp. 1-28, 2026.
- [10] A. Katal, “Evolution from Monolithic to Microservices Architecture: A New Era in Software Architecture”, *Advancements in Optimization and Nature-Inspired Computing for Solutions in Contemporary Engineering Challenges*, Vol. 9, pp. 235-279, 2025.
- [11] V. Dixit and D. Kaur, “A Systematic Review for Sustainable Software Development Practice and Paradigm”, *Journal of Computational Analysis and Applications*, Vol. 33, No. 6, pp. 112-125, 2024.
- [12] S. Amorim, “Extensibility in Ecosystem Architectures: An Initial Study”, *Proceedings of International Workshop on Software Ecosystems*, Vol. 15, pp. 11-15, 2015.
- [13] S. Amorim, “Flexibility in Ecosystem Architectures”, *Proceedings of International Conference on Software Architecture*, Vol. 20, pp. 22-29, 2014.
- [14] I. Carvalho, “HEAL ME: An Architecture for Health Software Ecosystem Evaluation”, *Proceedings of*

- International Workshop on Software Engineering for Systems-of-Systems on Distributed Software Development, Software Ecosystems and Systems-of-Systems*, Vol. 12, pp. 45-52, 2017.
- [15] B. Schwichtenberg and G. Engels, "SECOarc: A Framework for Architecting Healthy Software Ecosystems", *Communications in Computer and Information Science*, Vol. 1269, pp. 95-106, 2020.
- [16] J.C. Carver, E. Hassler, E. Hernandez and N.A. Kraft, "Identifying Barriers to the Systematic Literature Review Process", *Proceedings of International Symposium on Empirical Software Engineering and Measurement*, Vol. 8, pp. 203-212, 2013.
- [17] National Institute for Health Research (NIHR), "Surveys and Questionnaires: Revision2009", Available at:[https://www.rdsyh.nihr.ac.uk/wpcontent/uploads/2013/05/12\\_Surveys\\_and\\_Questionnaires\\_Revision\\_2009.pdf](https://www.rdsyh.nihr.ac.uk/wpcontent/uploads/2013/05/12_Surveys_and_Questionnaires_Revision_2009.pdf), Accessed in 2009.
- [18] M. Hyman and J. Sierra, "Open-Versus Close-Ended Survey Questions", *NMSU Business Outlook*, Vol. 14, No. 2, pp. 1-5, 2016.
- [19] S. Sadick, "The Influence of Service Quality on Customer Satisfaction: A Case Study", *International Journal of Indonesian Business Review*, Vol. 1, No. 1, pp. 29-37, 2022.
- [20] R. Feitosa, "Actionable Framework for Understanding and Improving Social and Human Factors that Influence the Requirements Management in Software Ecosystems", *ACM Transactions on Software Engineering and Methodology*, Vol. 36, No. 2, pp. 1-28, 2026.
- [21] I. Ur Rahman, A. Ur Rahman, S. Shahzad and S. Ur Rahman, "Software Ecosystem Architectural Challenges and Mitigation Strategies: A Systematic Literature Review", *Scientific Journal of Computer Science*, Vol. 2, No. 2, pp. 203-223, 2026.