AHP-Based Systematic Approach to Analyzing and Evaluating Critical Success Factors and Practices for Component-Based Outsourcing Software Development

Abdul Wahid Khan 1, Shams Ullah Khan 1, Hathal Salamah Alwageed 2, Faheem Khan 3*, Jawad Khan 4 and Youngmoon Lee 4*

1 Department of Computer Science, University of Science and Technology Bannu, Bannu 28100, Pakistan
2 College of Computer and Information Sciences, Jouf University, Sakaka 72388, Saudi Arabia
3 Department of Computer Engineering, Gachon University, Seongnam 13120, Korea
4 Department of Robotics, Hanyang University, Ansan 15588, Korea
* Correspondence: faheem@gachon.ac.kr (F.K.); youngmoonlee@hanyang.ac.kr (Y.L.)

Abstract: Component-based software development (CBSD) is a difficult method for creating complicated products or systems. In CBSD, multiple components are used to construct software or a product. A complex system or program can be created with CBSD quickly and with money while maintaining excellent quality and security. On the other hand, this research will persuade outsourced vendor companies to embrace CBSD approaches for component software development. We conducted a systemic literature review (SLR) to investigate the success factors that have a favorable impact on software outsourcing vendors’ organizations, and we selected 91 relevant research publications by creating a search string based on the study questions. This useful information was compiled using Google Scholar, IEEE Explore, MDPI, WILLEY Digital Library, and Elsevier. Furthermore, we completed all of the procedures in SLR for the full literature review, including the formulation of the SLR protocol, initial and final data collection, retrieval, assessment processes, and data synthesis. Among the ten (10) critical success factors we identified are a well-trained and skilled team, proper component selection, use of design standards, well-defined architecture, well-defined analysis and testing, well-defined integration, quality assurance, good organization of documentation, and well-organized security, and proper certification. Furthermore, the proposed SLR includes 46 best practices for these critical success factors, which could assist vendor organizations in enhancing critical success factors for CBOSD. According to our findings, the discovered success factors are similar and distinct across different periods, continents, databases, and approaches. The recommended SLR will also assist software vendor organizations in implementing the CBSD idea. We used the analytical hierarchy process (AHP) method to prioritize and analyze the success factors of component-based outsourcing software development and the result of different equations of the AHP approach to construct the pairwise comparison matrix. The largest eigenvalue was 3.096 and the CR value was 0.082, which is less than 0.1, and thus sufficient and acceptable.

Keywords: success factors; CBSD; practices; component based outsourcing; software development; SLR; vendor; SPSS

MSC: 68N01
1. Introduction

CBOSD allows us to construct and integrate product components that improve reusability, quality, and testing ease. The reusability technique decreases software development costs and time by reusing previously produced components. Software development demands the creation of low-cost, high-quality solutions, which is both vital and challenging. CBSD can assist developers in building software more quickly and at a lower cost. Component-based software engineering (CBSE) designs loosely coupled self-contained system components while removing unnecessary relationships. CBSE is concerned with the interconnection of many components to deliver services via several interfaces. Rapid software development is achievable due to this method of developing standard components [1]. Software systems have become more complicated and performance-oriented over time. To generate cost-effective systems, organizations usually use component-based technologies rather than building the entire system from the ground up. The purpose of using component-based technology is to save money on development.

Nonetheless, this industry has become more critical to reducing dependency on the existing market and fulfilling rapidly changing customer wants. The goal to lower development costs is quickly driving the adoption of component-based technology. More functionality can be created with less time and money if this technology is used [2].

A well-known software engineering field is CBSD. Its strategies and approaches include using architecture definition languages (ADLs), middleware, object-oriented design, and software architectures. On the other hand, the software is not the same as industrial equipment. As a result, there is no way to instantly translate rules from traditional architectural disciplines into software architecture. Understanding the component is not tricky in standard architectural fields because it is intuitive and fits well with the basic concepts/principles and engineering designs and models. However, the software components are not in the same boat [3].

Due to financial and time-to-market constraints, software development organizations increasingly rely on third parties to create more complicated systems or software. The third organization’s components will be well-organized, self-contained, high quality, and secure, and they will be merged based on customer needs. CBSD is the name given to this process [4]. Many scholars have put in substantial effort to create a time and cost-efficient software system [5].

CBSD is based on the principle of component reusability. Reusing previously constructed components, reusability is a critical feature that reduces the cost, time, and risk of software development. If an element cannot be reused, the CBSD is rendered unusable as a whole. According to the second definition, the property of any component can be reused with minimum or no alterations [3]. Software reuse, rather than building a software system from the ground up, constructs one from existing software components. Software’s reusability decides whether it may be reused in another application or changed in some way [6].

CBSD is a popular way to acquire more powerful and comprehensive software. CBSD reduces development expenses, shortens development time, and increases the amount produced by eliminating duplication of hard labor [7,8], which approaches the problem with marketable component assembly; similar concepts can be found in other technical disciplines.

Outsourcing or operations to a remote site is known as software outsourcing. In the twenty-first century’s global economy, outsourcing has become a normal business practice. Recently, several multinational firms have outsourced all or part of their software development (SD) work to overseas workers [9]. In today’s world, IT outsourcing is a common practice. According to one survey, the value of IS/IT outsourcing in 2018 was over USD 260 billion, and over 94% of Fortune 500 organizations outsource at least one core business function [10].

We established five (05) research questions (RQs) to investigate these success factors:
RQ1. What are the success factors that have a positive impact on software outsourcing vendor organizations in adapting the concept of a component-based approach in software development?

RQ2. What are the existing solutions/methods for increasing the success factors by the software outsourcing vendor organization, as outlined in the literature?

RQ3. Are the identified success factors of CBSD varying over time?

RQ4. Are the identified success factors of CBSD varying based on diverse study strategies?

RQ5. What will be the global rank of the identified success factors by using the AHP method?

RQ1 and RQ2 are the basic research questions defined for the identification of success factors and their practices/solutions in the CBSD. RQ3 and RQ4 are used for the statistical analysis of the identified success factors showing the significant difference regarding time and study strategies.

The last question, RQ5, is defined to use the concept of AHP to find the importance and weight of the various groups defined in the model development.

Using the 91 research articles shortlisted for the proposed SLR study, we discovered 10 essential success variables as well as 46 practices that would help in improving these success factors. The findings confirm the similarities and differences in the indicated success factors across time, continents, databases, and methods. Software suppliers will be aided in adopting the CBSD paradigm by SLR success factors and practices.

The following sections make up this paper: Section 2 explains the literature review; Section 3 explores the research methods; Section 4 contains the SLR findings and analysis; Section 5 discusses the study’s limitations; and Section 6 discusses our conclusions and future work.

2. Related Work

CBSE research based on community initiatives was suggested by Zhang et al. [11]. The authors created a CBSE solution and validated the Trustee components system using the Cushion and Eclipse tools. The authors describe two techniques: service-oriented software development and CBSE. The Open Services Gateway Initiative (OSGI) is a software-oriented architecture (SOA) and CBSE interface (SOA).

Chouambe et al. [12] are interested in reverse engineering component-based system software models. Most current approaches to this goal concentrate on reverse engineering components while neglecting external dependencies. Furthermore, class and module exchange are based on the description of dependent and existing components. As part of iterative reverse engineering strategies and methodologies, developers must apply component-based software designs and examine alternative tools, processes, approaches, and techniques.

The majority of component-based approaches involve high-level programming development. In such cases, specific component-based models use a specialized programming language to construct the model, which mandates some specified rules. The primary benefit of CBSE is that it is self-contained in terms of individual component development.

According to Hunt and McGregor [13], CBSE is a way for producing and developing systems from new components, which has substantial significance for numerous computer software engineering procedures. CBSE techniques have changed over time, and this field of research is continually expanding to address fundamental challenges, concerns, and convenience in CBSE. Despite the limits of the present software tools, students would obtain practical experience with CBSE approaches if a CBSE course was incorporated into the curriculum.

Abdellatif et al. [14] investigated how CBSS may be integrated with various components and deployed separately. Even though researchers used a variety of CBSE characteristics, putting the CBSE metrics into practice is difficult since some measures overlap
or are poorly defined. The authors described how to measure the complexity of the interface’s parameters and methods. The metrics suggested are primarily for assessing JavaBeans’ reusability component.

Time, cost, and product quality are three essential factors that drive software development and directly impact the software business. According to Khan et al. [15,16], information technology has considerable hurdles, including meeting product deadlines while lowering development time and cost. A reusability approach to software components could help cut software development costs and time in half. Many software companies use CBSE techniques to answer their clients’ requirements for a low-cost, quick-to-market solution. Existing components in the software development process can also help to boost software productivity. Adopting cutting-edge tools is also quite advantageous when software development expenses and time must be kept to a minimum.

The development of model-driven engineering prototypes could have a number of positive effects on the projects’ overall performance. In this case, a number of traditional embedded system concepts and approaches may be useful. To boost quality, CBSE also uses reusable components. Bunse et al. [17] looked into how model-driven prototypes and CBSD techniques might aid embedded system development. For a car microcontroller subsystem, the authors adopted the Marmot technique. Two essential approaches were used to compare and measure the development efforts and size of the model: agile software development and unified process.

Kahtan et al. [18] described the CBD’s future security issues and focused on constructing models by merging current software components. As a result of this strategy, effective customer service is provided as well as the ability to make system improvements as needed. Throughout the process of creating a software system, CBSE provides a wide range of features. CBSE focuses on constructing software systems by reusing high-quality, independent software components, according to Iqbal et al. [19].

CBSD is founded on the “Buy not Build” principle, according to Bauer and Fritz [20]. They observed that putting together components to create a software system is faster than writing a code. Although many of them are obtained through outsourcing companies, the developer may require particular components in order to build at the correct location.

The use of the CBSD concept is thought to benefit Software Outsourcing Vendor Organizations by increasing or justifying crucial success aspects of CBSD. As a result, we decided to undertake an SLR to learn more about the problems of software outsourcing vendors. Furthermore, determining the best solutions or techniques that the software outsourcing vendor’s firm should use to boost success factors is even more critical.

We began participating in many ways. To begin, we have learned that CBSD has a set of success factors according to SLR. Many academics have discussed and defined these success factors separately and not conducted proper SLR study. As far as we know, there is not a model in the literature that leverages SLR to help software vendor organizations use the CBSD concept. Second, our model for component-based outsourcing software development (MCBOSD) would aid software outsourcing organizations in improving their efficiency. In the CBSD software vendor ecosystem, our model is a one-of-a-kind addition. Our research is based on SLR to identify success factors and SLR and practical study to assess solutions/practices in software organizations to address, mitigate, and improve success factors.

We are the first to conduct the SLR technique on the given data. The other authors mentioned in the literature had not conducted SLR but they have generally discussed the success factors and their practices separately, whereas we have conducted secondary study to identify all the discussed success factors and their practices using SLR. Thus, we can claim our research study to be novel.

3. Methodology

We used SLR to find success features and practices that can aid software outsourcing vendor organizations implementing CBSD. SLR is used to find related published work
using a pre-defined search string based on the research questions. The SLR used a pre-defined inclusion/exclusion method/criteria to analyze the obtained data. Data can be collected through survey [21], technically [22] and SLR [23]. The three phases of the SLR are planning a review, conducting a review, and reporting a study. Because SLR used a systematic evaluation process, the outcomes of SLR are more reliable than those of a traditional literature review. We followed a step-by-step method to identify the success factors and practices of CBSD. We chose each article based on its relevance to the topic or search word and then applied inclusion/exclusion criteria to eliminate irrelevant papers from our search. The data extraction from the selected research papers was then carried out systematically, followed by data synthesis and publication-quality assessment as shown in Appendix A. We identified 16 success factors from the proposed SLR, which were further reduced to 10 success factors by combining the related critical success factors for CBSD. The planning and conducting phases of our SLR have already been completed, and we now need to submit the findings of the performed phase. The main goal of this study article is to highlight all of the success factors and how they should be managed using SLR, that we identified as critical success factors for component-based technology based on frequency ≥20%, where other scholars have utilized the similar approach [24]. We also used SLR to find out the practices/solutions for increasing the success factors by the software outsourcing vendor organization. We applied the chi-square test (linear-by-linear association) to identified the success factors of CBSD varying over time and varying based on diverse study strategies and at the end, we applied the AHP method to find the global rank of the identified success factors of the CBSD.

3.1. Search Process and Practice

We followed the procedures below to create the search string for the SLR.

- For the definition of the search phrase, identify the population, outcomes, and intervention.
- Identification of the alternative spellings and synonyms.
- Verification and validation of keywords for search queries in all relevant literature.
- If there is a need to instruct the search engine, employ Boolean operators (AND/OR) for precise results.

Initially, a search string was created to discover relevant research publications in multiple databases. This trial’s search engine is shown in Table 1 and the search string is as below.

(Outsourcing OR “Software outsourcing”) AND “CBSD” AND “Success factors” AND Practices AND vendor)

The final search string is:

((“Component Based Software Outsourcing” OR “Component Based Software development” OR “Component Based Software engineering” OR “Component Based Software solution”) AND (“Success factors” OR Factor OR CSFs OR “Critical Success Factors”) AND (practices OR solutions OR methods) AND (vendor OR suppliers OR contractors OR Provider)).

Table 1 displays the results of the final search query’s search for relevant articles. We chose the relevant research publications based on thorough reading, paper quality, and verification using the inclusion and exclusion criteria listed in Table 1.

| Name of Data Source | Result Found | Primary Selection | Final Selection |
|---------------------|--------------|-------------------|-----------------|
| Google Scholar      | 2830         | 215               | 27              |
| IEEE Xplore         | 56           | 40                | 30              |
| Elsevier            | 9310         | 60                | 8               |
| WILLEY              | 554          | 90                | 15              |
### Publication Selection

#### Inclusion Criteria

- CBSD will be gathered from research publications covering it in depth.
- Papers that explore outsourcing vendor software organization in depth.
- CBSD requires the usage of a software outsourcing vendor.
- Research that looks into the factors that influence CBSD’s success.
- Publications discussing CBSD success factors and their solutions/practices.
- Publications in the English language.
- Publications with a title similar to the title of our study article.
- The publication contains terms relevant to our search query.

#### Exclusion Criteria

- Work that is not related to component-based development.
- Papers unrelated to CBOSD.
- Papers that are not relevant to our search string.
- Paper titles that are not relevant to our search string.
- Paper abstracts that are not relevant to our search string.
- Papers’ documentation that are not relevant to our search string.
- Papers that are duplicated.
- Papers that are not relevant to English language.

#### Publication Selection and Quality Assessment

The process of selecting a research publication based on the study’s title, abstract, and keywords is known as publication selection. We chose 411/12,559 documents for the most part. The following questions were used to evaluate a publication’s quality.

- Is the author able to effectively convey why implementing the CBSD approach can benefit the software vendor organization?
- What strategies does the author use to increase these success factors?

#### Data Extraction

To retrieve data, we utilized the following criteria:

- The title of the study, the names of the authors, and other details regarding the document’s publication such as conference or journal, volume and issue number, number of pages of publication, conference location, and year of publication.
- The papers are relevant to our research questions.

#### Data Synthesis

Through SLR, a list of success factors was identified in 91 relevant publications. The majority of the data synthesis was carried out by the first author of the paper, with the rest of the authors contributing considerably to the validation of the SLR results. We first established 16 success factors for CBSD, which were then examined and validated, and some of these success factors were blended based on their commonalities. Table 2 concludes with a list of 10 critical success factors.
4. Results Discussion

4.1. Success Factors Find through SLR

To respond to RQ1, we discovered success factors for outsourcing software vendor companies by critically assessing research papers and reviewing them using SLR (Table 2). The success factors of the CBOSD identified along with their occurrences in each research paper included in the SLR are highly competent team (training + skills) (41%), well-defined procedure for component selection (31%), proper standard for designing (29%), proper architecture (26%), well-defined analysis and testing (29%), well-defined integration (29%), assurance of quality standards (25%), good organization of documentation (25%), proper security (23%), and proper certification (21%).

The “well-trained and skilled team” is the most critical success factor, as reported in 41% of the research papers. People must have well-defined reuse-oriented professions with suitable training, skills, and rewards [25]. Developers must be thoroughly trained to acquire the essential skills and competence to use the most up-to-date technology and tools, which will make the integration process more effortless in the long run. The vendor should be contacted for support and training if the software component is commercial off-the-shelf (COTS/OTS) [26]. Tools and technology constantly evolve, necessitating new professional knowledge, abilities, and advice. As a result, staff must be appropriately trained to hold the necessary professional expertise [27]. Team members should be prepared to follow the organization’s processes, with incentives for good reuse performance and reuse throughout the organization [27]. All stakeholders in the GSD project should receive proper training on the entire component-based development process [28].

The success factor “well-defined procedure for component selection” is highlighted in 31% of research papers while conducting the SLR study. An efficient and effective software component selection approach is necessary to utilize COTS technology fully. Various COTS selection strategies, processes, and methodologies have been presented. Component selection, composition, and, most crucially, component integration is all required to develop COTS-based systems. Following its successful implementation, the technique expanded in popularity. Today, many software components are commercially available, with component selection to suit all system needs while minimizing the cost of system software [29]. Many enterprises dedicate a significant amount of time to selecting reusable components, as choosing the proper ones has a considerable impact on the project’s outcome. Due to the lack of standardization in the component selection process, each project develops its strategy [30]. CBSE aims to compose, select, and create components. As this technique becomes more common, and as a result, the number of commercially available software components expands, selecting a collection of essential components to meet all of the requirements while keeping costs low becomes increasingly tricky. Choosing the best component systems is a difficult task [30].

“Using standards for designing” is also the critical success factor for CBOSD and reported in 29% of the research articles. A lack of a consistent design could stymie integration.

Table 2. Identification of success factors through SLR.

| Success Factors                                      | Frequency (N = 91) | %   |
|------------------------------------------------------|--------------------|-----|
| Highly competent team (training + skills)            | 37                 | 41  |
| Well defined procedure for component selection      | 28                 | 31  |
| Proper standard for designing                       | 26                 | 29  |
| Proper architecture                                 | 24                 | 26  |
| Well defined analysis and testing                    | 26                 | 29  |
| Well-defined integration                            | 26                 | 29  |
| Assurance of quality standards                       | 23                 | 25  |
| Good organization of documentation                   | 23                 | 25  |
| Proper security                                      | 21                 | 23  |
| Proper certification                                 | 19                 | 21  |

The success factors are identified according to their occurrences in each research paper included in the SLR study.
To maximize reuse and coverage from design to delivery, the component should be made to be standard [25]. Consistency in requirements and architecture design is critical for successful software integration because the requirement phase encompasses the whole development process. The software architecture serves as a blueprint for creating the product [31].

The “proper architecture” success factor is reported in 26% of the papers, as shown in Table 2. Component development identifies reusable entities and relationships between them, starting with the system requirements. The early design process consists of two steps: first, the specification of a system architecture in terms of functional components and their interactions, which provides a logical perspective of the systems, and second, the specification of a system architecture made up of physical components [32]. Organizations have previously been unable to find reusable software outside their development groups due to a lack of defined interfaces. Architectures such as the Common Object Request Broker (CORBA), Sun’s Enterprise Java Beans, and Microsoft’s Component Object Model (COM) have recently been standardized, allowing for cross-organizational reuse [33].

“Well-defined analysis and testing” is also a critical success factor of CBOSD, as reported in 29% of the research articles as per our SLR finding. On the other hand, components are built as standalone projects, allowing development and testing to take place independently of different components. Although modularity is stressed in structured methods, it is not as prominent as evolution in CBSD, which is the concept of changing or upgrading components with little or no impact on other components [34]. The component severity analysis aids in concentrating testing efforts in areas with the highest potential for reliability improvements. As a result, time and money are saved [35].

“Well-defined integration” is a critical success factor, as reported in 29% of the papers. Special-purpose software is a small but crucial component of a more extensive software system. This crucial computer programmer software is required for designing and integrating existing modules across the domain and developing new modules. Before constructing a new module “component”, it is necessary to assemble new module “components” in a pre-existing component that facilitates inter-component communication [36]. Several of the previously discovered issues start to appear during the integration phase. These challenges increase the effort of global teams and reduce the quality of the final functioning output. More than half of the projects have cost overrun and time overrun due to the complexities and incompatibilities discovered during the integration stage [28].

The “quality assurance” and “good organization of documentation” success factors of CBSD are also critical, as reported in 25% of the research papers. Quality has arisen as a significant concern due to the architectural diversity of component-based software. Developers must trust the component designer. The quality of the component has a significant impact on the quality of the finished product. For a quality estimate, several “Software Quality Models” are required; most models have been updated and developed for specific categories, such as component, commercial off-the-shelf (COTS), and open-source software [37]. A document describing a component’s specification, use, and execution environment should be made public. Organizations aid in the creation of components on a technological level. A configuration management library should store and manage different versions of a component. All changes should be meticulously documented [38].

“Proper security” and “proper certification” are the critical success factors for CBOSD as recorded in 23% and 21% of papers, respectively. Integrating security into the software development process looks to be a challenging task, and evaluating security appears to be even more difficult. The main reason for this is that a non-functional security requirement was not addressed effectively during the early stages of system development, and the operational environment is complex. The research community faces a significant issue in developing a well-established scale against which we can grade a software system’s level of security. Security has always been considered an afterthought, with protec-
tive measures being applied after the software has been constructed [39]. One way to establish trust in a component is to certify it. The process of evaluating a component’s property value levels and issuing a certificate as confirmation that the component meets well-defined standards and is adequate to complete a set of requirements is known as certification. Third-party certification is frequently regarded as an effective method for instilling trust in efforts involving many partners. As a result, it can increase user confidence in software component makers [40].

4.1.1. Analysis of the Success Factors for CBOSD for Software Outsourcing Vendors Organization

We conducted a statistical study on the indicated success factors based on two separate variables. These variables include the decade in which the study was written and the research methods used in the paper. The goal of these analyses is to see if these success factors have remained consistent over time (in decade) and study method, or if they have changed.

Study and Analysis of Success Factors of CBOSD Based on Different Times Decades

In response to RQ3, we looked at success factors throughout various decades from 1990 to 2019 as shown in Figure 1. This time period was separated into three decades. The first decade was 1990–2000, the second decade was 2001–2010, and the third decade was 2010–2020 (2011–2019). The majority of these success determinants are discovered in the second and third decades of life. The success elements of well trained and skilled team were the most crucial success factors in the first decade (1990–2000), as they were reported in 100% of the study articles. In the first decade, quality assurance and proper certification were most reported. The success factors of proper architecture, quality assurance, proper security, proper certification, and availability of repository was described in 14%, 17%, 14%, and 12% of research articles, respectively, in the second decade (2001–2010), indicating that they well-trained and skilled team.

Figure 1. SLR final paper collection from different periods.

Similarly, in the third decade, repository availability was not a significant success criterion. Even while our criterion of ≥20% identifies additional success factors; it is only
referenced in 12% of research publications. There is a substantial decade gap between issues of quality assurance.

As a result, only 1% of these issues were reported in the first decade, rising to 17% in the second decade and 39% in the third decade. The linear-by-linear Chi-Square test is used to detect the “main alteration” among several time decades in order to find significant variations among these success criteria. All of these details may be seen in Table 3.

Table 3 shows that there was just one significant variation between these time decades for the success component “quality assurance” among 10 success factors with p less than 0.05. This success factor was not detected during the decade 1990–2000; it was reported as 0% and 17% in 2001–2010, and 39% in 2011–2019. This shows that, with the exception of the 1990s and 2000s, every other decade considered “quality assurance” to be an important success factor.

### Table 3. List of identified success factors through SLR in two different decades.

| Success Factors                                      | 1990–2000 (n = 8) | 2001–2010 (n = 42) | 2011–2019 (n = 41) | Chi-Square Test (Linear-by-Linear Association) a = 0.05 |
|-------------------------------------------------------|-------------------|--------------------|--------------------|--------------------------------------------------------|
| F | % | F | % | F | % | X² | df | p     |
| Highly competent team (training + skills)             | 8 | 100 | 13 | 31 | 20 | 49 | 0.738 | 1 | 0.390 |
| Well-defined procedure for component selection        | 2 | 25 | 15 | 36 | 11 | 27 | 0.167 | 1 | 0.683 |
| Using standards for designing                        | 2 | 25 | 11 | 26 | 13 | 32 | 0.323 | 1 | 0.570 |
| Proper architecture                                  | 3 | 38 | 6  | 14 | 15 | 37 | 1.495 | 1 | 0.221 |
| Well-defined analysis and testing                     | 3 | 38 | 9  | 21 | 14 | 34 | 0.323 | 1 | 0.570 |
| Well-defined integration                             | 1 | 13 | 11 | 26 | 14 | 34 | 1.669 | 1 | 0.196 |
| Quality assurance                                    | 0 | 0  | 7  | 17 | 16 | 39 | 8.295 | 1 | 0.004 |
| Good organization of documentation                   | 1 | 13 | 10 | 24 | 12 | 29 | 1.000 | 1 | 0.317 |
| Proper security                                      | 2 | 25 | 7  | 17 | 12 | 29 | 0.855 | 1 | 0.355 |
| Proper certification                                 | 0 | 0  | 6  | 14 | 11 | 27 | 2.731 | 1 | 0.098 |
| Improved communication and coordination              | 2 | 25 | 0  | 0  | 10 | 24 | 3.105 | 1 | 0.078 |
| Availability of repository                           | 0 | 0  | 5  | 12 | 5  | 12 | 0.515 | 1 | 0.473 |

Analysis of Success Factors of CBOSD Based on Diverse Study Strategies

In response to RQ4, the Table 4 highlights SLR’s conclusions based on the technique used in the study. SLR discovered a total of 91 papers with a sample size of 91. The SLR approach was used to extract the data from each publication. The following table summarizes the success characteristics discovered using SLR across a variety of methods/strategies. The quality assurance success factor is the significantly different factor among other success factors with the p value of 0.004 showing that the quality is not important in decade 1990 and 2000 with a frequency of 0%. In decade 2001 and 2010, the equality success factor is reported with 17% whereas in decade 2011–2019, the quality success factor is reported with 39%.
| Success Factors                                           | Occurrence in SLR (n = 91) Different Methods | Chi-Square Test (Linear-by-Linear Association) a = 0.05 |
|----------------------------------------------------------|---------------------------------------------|------------------------------------------------------|
|                                                          | CS (n = 14) | I (n = 2) | LS (n = 23) | SLR (n = 7) | LR (n = 8) | ER (n = 1) | T (n = 9) | ES (n = 5) | Report (n = 2) | Other (n = 20) |        |        |        |
|                                                          | F | %   | F | %   | F | %   | F | %   | F | %   | F | %   | F | %   | F | %   | F | %   | F | %   | F | %   | X² | df | p    |
| Highly competent team (training + skills)                 | 2 | 14  | 1 | 50  | 9 | 39  | 7 | 100 | 3 | 38  | 1 | 100 | 3 | 33  | 2 | 40  | 1 | 50  | 8 | 40  | 0.435 | 1   | 0.510 |
| Well-defined procedure for component selection            | 4 | 29  | 0 | 0   | 8 | 35  | 0 | 0   | 4 | 50  | 0 | 0   | 4 | 44  | 1 | 20  | 0 | 0   | 7 | 35  | 0.159 | 1   | 0.690 |
| Using standards for designing                            | 3 | 21  | 0 | 0   | 4 | 17  | 4 | 57  | 1 | 13  | 0 | 0   | 3 | 33  | 2 | 40  | 1 | 50  | 8 | 40  | 3.018 | 1   | 0.082 |
| Proper architecture                                      | 2 | 14  | 0 | 0   | 5 | 22  | 6 | 86  | 0 | 0   | 0 | 0   | 4 | 44  | 2 | 40  | 1 | 50  | 4 | 20  | 0.255 | 1   | 0.614 |
| Well-defined analysis and testing                        | 5 | 36  | 0 | 0   | 6 | 26  | 4 | 57  | 1 | 13  | 0 | 0   | 4 | 44  | 2 | 40  | 1 | 50  | 3 | 15  | 0.587 | 1   | 0.443 |
| Well-defined integration                                 | 1 | 7   | 0 | 0   | 6 | 26  | 4 | 57  | 3 | 38  | 0 | 0   | 3 | 33  | 1 | 20  | 1 | 50  | 7 | 35  | 2.106 | 1   | 0.147 |
| Quality assurance                                         | 3 | 21  | 0 | 0   | 8 | 35  | 4 | 57  | 1 | 13  | 0 | 0   | 3 | 33  | 6 | 60  | 1 | 50  | 3 | 15  | 0.648 | 1   | 0.421 |
| Good organization of documentation                      | 0 | 0   | 1 | 50  | 4 | 17  | 5 | 71  | 3 | 38  | 0 | 0   | 5 | 56  | 1 | 20  | 1 | 50  | 3 | 15  | 0.466 | 1   | 0.495 |
| Proper security                                           | 2 | 14  | 1 | 50  | 7 | 30  | 0 | 0   | 1 | 13  | 1 | 100 | 3 | 33  | 1 | 20  | 1 | 50  | 4 | 20  | 0.045 | 1   | 0.832 |
| Proper certification                                     | 0 | 0   | 2 | 100 | 5 | 22  | 2 | 29  | 0 | 0   | 1 | 100 | 2 | 22  | 1 | 20  | 0 | 0   | 6 | 30  | 1.304 | 1   | 0.253 |

Case study: CS; interview: I; literature survey: LS; literature review: LR; experience report: ER; thesis: T; empirical study: ES; frequency: F.
Figure 2 depicts the numerous publications in relation to the various study techniques mentioned in our thesis. The success aspects were highlighted in the literature review, case study, and thesis, according to the data.

The two most essential success factors in the case studies are “well defined analysis and testing” and “proper component selection”, which are reported in 36% and 29% of research publications, respectively. “Good organization of documentation”, “proper certification”, and “availability of repository” are not relevant in the research since they are only mentioned in 1% of the published papers, but they are critical in the case studies. “Proper certification”, “highly competent team (training + skills)”, “good organization of documentation”, and “proper security”, which are all rated at 100%, 50%, and 50%, respectively, are the crucial success criteria. Because they are recorded at 0%, the others are not significant success criteria.

In the literature survey, the essential success elements “using standard for developing”, “correct documentation”, “better communication and coordination”, and “availability of repository” are recorded in less than 20% of papers. The remainder, on the other hand, are key success variables based on the criteria. As indicated below, the set level of ≥20%, “proper component selection”, “well-organized security”, and “repository availability” are not essential success requirements in the case of SRL; however, the others are identified as critical success factors. All other techniques follow the same formula. The important differences between various research methodologies among these 10 crucial success criteria of CBOSD are evaluated using a linear-by-linear association of the chi-square test. Our data reveal that in all of the search methodologies in Table 4, “repository availability” is the most important factor, showing that of the 12 success variables with p less than 0.05, there was only one significant variation between these research methods for the success factor “availability of repository.” This success factor was not detected in the case study, interview, literature survey, experience report, empirical study and reports. It was reported as 0% and 14% SLR, 13% in the literature review, and 56% in the thesis, respectively.

4.1.2. Practices for Addressing the Identified Success Factors for CBOSD through SLR

In response to RQ2, we found a total of 46 solutions/practices for enhancing the established success factors of component-based outsourced software development using
SLR [41–43] from the vendor’s perspective. We used several abbreviations in the tables of solutions/practices below.

- ‘CSF’ is used for “Critical Success Factor”.
- ‘PCSF’ is used for “Practices for Critical Success Factor”.
- ‘P’ is used for a paper-like P-1 denotes paper-1.

Practices for 10 Factors

The practices for crucial success criteria are highly competent team (training + skills), well-defined procedure for component selection, proper standard for designing, proper architecture, well defined analysis, and testing, well defined integration, assurance of quality standards, good organization of documentation, proper security, and proper certification are shown in Tables 5–14.

### Table 5. CBOSDSF 1: Highly competent team (training + skills).

| S. No | Practices and Solutions for Addressing Highly Competent Team (Training + Skills) | Paper Id | Index |
|-------|---------------------------------------------------------------------------------|----------|-------|
| PCBOSDSF for Success Factor 1 | To assemble well-trained teams to guarantee that component reuse protocols are followed. To provide suitable training for existing employees in order to increase their advanced tool abilities. To assemble skill teams with the necessary professional expertise. To provide adequate training for all CBSD staff on the entire CBD process. Organization-wide training programmers should be organized to boost competence and awareness of new breakthroughs. | p-1, p-6, p-11, p-12 p-44 | 5 |

### Table 6. CBOSDSF 2: Well-defined procedure for component selection.

| S. No | Practices and Solutions for Addressing Well-Defined Procedure for Component Selection | Paper Id | Index |
|-------|-------------------------------------------------------------------------------------|----------|-------|
| PCBOSDSF for Success Factor 2 | To choose more effective and efficient components using the optimal performance model (OPM). To choose appropriate components using architectural analysis. To choose the proper component, make a priority list of elements like functionality and cost. Cost, availability of source code, ease of modification and integration, documentation, ease of use, technical support, familiarity with the component, and proof of successful component usage are all considerations to consider. To choose components that meet all of the design criteria. To choose components using COTS Aware Requirements Engineering and Software Architecture (CARE/SA). To choose acceptable components using the Weight Scoring Method (WSM). When choosing components, trade-off analysis should be carefully studied. | p-2, p-8, p-12, p-13 p-17, p-21, p-33, p-43 | 8 |

### Table 7. CBOSDSF 3: Using standards for designing.

| S. No | Practices and Solutions for Addressing Using Standards for Designing | Paper Id | Index |
|-------|---------------------------------------------------------------------|----------|-------|
| PCBOSDSF for Success Factor 3 | Pattern catalogues and design notation such as Unified Modeling Language (UML) will be useful in using standard tools. It lowers the total development cost by keeping the requirements and architecture design under control. To employ a standardized design notation, such as UML. Both component and system modelling can benefit from the usage of Unified Modelling Language. To meet the goals, a component-centric design framework should be used since it fits and combines nicely. | p-1, p-6, p-7, p-8, p-44 | 5 |
### Table 8. CBOSDSF 4: Proper architecture.

| S. No | Practices and Solutions for Addressing Proper Architecture | Paper Id | Index |
|-------|----------------------------------------------------------|----------|-------|
|      |                                                          | p-1, p-2, p-6, p-8, p-9, p-12, p-13, p-18 | 8     |

**PCBOSDSF for Success Factor 4**

- To form a group of experts who will develop the architecture for component components.
- To gather user requirements and determine system specifications so that an acceptable system architecture may be chosen.
- To assemble a well-trained team to choose the best architectural design.
- To employ SAAM (Software Architecture Analysis Methodology) and ATAM (Application Technology Assessment Methodology) (Architecture Tradeoff Analysis Method).
- Cross-organization reuse will be aided by the usage of CORBA, Sun’s Enterprise Java Beans, and Microsoft’s Component object Model (COM).
- To choose components that fit within the current architecture or to justify the design at the outset.
- To determine the component architecture by inspecting the component design.
- To change the system design using trade-off analysis.

### Table 9. CBOSDSF 5: Well-defined analysis and testing.

| S. No | Practices and Solutions for Addressing Well-Defined Analysis and Testing | Paper Id | Index |
|-------|------------------------------------------------------------------------|----------|-------|
|      |                                                                        | p-2, p-8, p-12, p-13, p-14, p-18, p-41 | 8     |

**PCBOSDSF for Success Factor 5**

- To assess whether the component meets the system requirements using the Optimal Performance Model (OPM).
- To employ SAAM (Software Architecture Analysis Methodology) and ATAM (Application Technology Assessment Methodology) (Architecture Tradeoff Analysis Method).
- To put freshly created components through rigorous testing.
- To properly test a component, different test cases and prototypes should be used.
- Separate component testing from the testing of other components.
- Component testing should be done in isolation.
- Component severity analysis will be used to target testing efforts.
- To investigate and quantify the quality of software testability using an empirical analytic technique.

### Table 10. CBOSDSF 6: Well-defined integration.

| S. No | Practices and Solutions for Addressing Well-Defined Integration | Paper Id | Index |
|-------|------------------------------------------------------------------|----------|-------|
|      |                                                                  | p-6, p-12, p-22, p-32 | 4     |

**PCBOSDSF for Success Factor 6**

- To allocate adequate planning tasks to each member of the organization’s team.
- To test components before integrating them during the development phase.
- To integrate software components in a step-by-step manner.
- To adhere to the integration sequence based on the software components’ availability.
Table 11. CBOSDSF 7: Quality assurance.

| S. No | Practices and Solutions for Addressing Quality Assurance | Paper Id | Index |
|-------|----------------------------------------------------------|----------|-------|
|       | Quality may be enhanced by employing good management.    | p-9, p-13, p-21, p-23, p-34 | 5     |
|       | The use of simulation techniques for component evaluation improves quality. |          |       |
| PCBOSDSF for Success Factor 7 | To assure component quality, employ estimation of component reusability. |          |       |
|       | To utilize metrics to improve the system’s quality. |          |       |
|       | To choose the correct documented component. |          |       |

Table 12. CBOSDSF 8: Good organization of documentation.

| S. No | Practices and Solutions for Addressing Good Organization of Documentation | Paper Id | Index |
|-------|---------------------------------------------------------------------------|----------|-------|
|       | To properly format papers and keep track of versions.                     | p-6, p-12, p-44, p-3 | 4     |
|       | To properly format documents as well as the descriptions of their dependencies. |          |       |
| PCBOSDSF for Success Factor 8 | To keep track of all changes in the pre- and post-conditions. |          |       |
|       | To write component documentation using a supported environment. |          |       |

Table 13. CBOSDSF 9: Well-organized security.

| S. No | Practices and Solutions for Addressing Well-Organized Security | Paper Id | Index |
|-------|----------------------------------------------------------------|----------|-------|
|       | To employ the Compositional Component Security Assurance (CSA) framework, which is based on the atomic components’ security properties. | p-2, p-5, p-6, p-29 | 4     |
| PCBOSDSF for Success Factors 9 | To pay extra attention to the quality requirements in order to maintain security. |          |       |
|       | To employ the Architecture Tradeoff Analysis Method (ATAM), which focuses on a number of different quality factors (modifiability, availability, security, and performance) |          |       |
|       | To employ a security assessment framework. |          |       |

Table 14. CBOSDSF 10: Trusted and certified.

| S. No | Practices and Solutions for Addressing Trusted and Certified | Paper Id | Index |
|-------|----------------------------------------------------------------|----------|-------|
|       | To choose from an approved environment for a component. | p-8, p-15, p-36, p-56 | 4     |
| PCBOSDSF for Success Factors 10 | To certify the components using the Software Component Maturity Model (SCMM). |          |       |
|       | To establish confidence, certified components should be used. |          |       |
|       | To make each component certifiable in order to overcome the effort of trust components. |          |       |
4.1.3. Analytical Hierarchy Process (AHP)

In response to RQ5, we applied an AHP approach for the MCDM problem, which is used by many researchers for prioritizing and analysis [44,45], and is very important. To prioritize and analyze the specified success criteria of CBOSD, we employed the MCDM approach based on the AHP method. Using a pairwise comparison method, the AHP methodology is generally used to assess relevant weight between multiple criteria, identify and prioritize concerns, and compute weights of criteria inside decision-making problems [46,47]. The three phases of AHP are as follows:

- Dissecting complicated decision-making issues down to their most basic hierarchical structure.
- To identify the priority-weights of issues and their subordinate success criteria, pairwise comparisons are performed.
- Verification of the consistency of the results.

We utilized the AHP Equations (1)–(4) to prioritize and analyze the success factors of component-based outsourcing software development. These equations draw from the discussions by Lipovetsky [48,49] and Saaty [50].

\[
A = \begin{bmatrix}
1 & a_{12} & \cdots & a_{1N} \\
 a_{21} & 1 & \cdots & a_{2N} \\
 \vdots & \vdots & \ddots & \vdots \\
 a_{N1} & a_{N2} & \cdots & 1
\end{bmatrix}
\]

where

\[a_{ij} = 1/a_{ji}, a_{ij} > 0.1\] (1)

\[AW = \lambda_{\text{max}}(W)\] (2)

where A is a set of pairwise comparisons for CBOSD success factors, max denotes the highest eigen vector value, and W denotes the appropriate weight.

\[CI = (\lambda_{\text{max}} - N)/(N - 1)\] (3)

\[CR = CI/RI\] (4)

where N represents the order of the success factors, CI represents the consistency index, and RI represents the random index’s consistency value, which varies depending on the total number of success factors. Although the absolute value of the consistency ratio CR is (0.10), the priority weights of the success criteria are outstanding and acceptable if the absolute value of CR is less than 0.10. To restore consistency, the evaluating technique from step 1 must be repeated if the total value of CR is more than 0.10.

Findings of Pair Wise Comparison, Priority Weights and Checking Consistency

For the preparation level in Tables 15 and 16, we used equation one to construct the pairwise comparison matrix and Equations 3, 4, and 5 to calculate the normalized matrix. The largest eigenvalue was 3.096 and the CR value was 0.082, which is less than 0.1, which is sufficient and acceptable.

Tables 17 and 18 provide the pairwise comparison matrix and normalized matrix for the standards level. The highest eigenvalue was 3.029 and the CR was less than 0.1. As a consequence, the priority weights for success criteria are satisfactory and acceptable. The assessment process is repeated if the CR value is more than 0.1 to increase consistency.

Using the above-mentioned equations, Tables 19 and 20 offer a pairwise comparison and normalized matrix for the integration level success factors. Table 21 shows the highest eigenvalue as 2 and the CR value as 0 (less than 0.1); therefore, the success factor’s priority weights are satisfactory and acceptable once more.
Tables 22 and 23 provide a pairwise comparison and normalized matrix of prominence level with a maximum eigenvalue of 2 and CR value of 0, less than 0.1, suggesting that the priority weights are appropriate and acceptable. The pairwise comparison and normalized matrix of these four levels are shown in Tables 24 and 25, where the maximum eigenvalue is 4.101 and CR is 0.037, which is less than 0.1, showing that the priority weights are suitable and acceptable.

Table 26 details the local and global weights of the success criteria, as well as their priority rankings. The global weights represent the importance of certain success variables in the overall study. Global weights of success factors are the product of local weights of success factors and weights of its concerned level. CSF, for example, has a global weight of 0.087, which is computed as 0.03330.261.

The worldwide ranking values for the additional success factors mentioned above were also established. Figure 3 shows the overall global ranking of these success drivers. Among these 10 critical success variables, CSF1 has the highest worldwide value of 5.449, while CSF7 has the lowest global value of 0.087. The remaining success factors have a higher significance than CSF7 but a lower significance than CSF1. Table 26 delves deeper into the prioritization and ranking of these critical success factors. The highest critical or significant success factor for CBOSD is CSF2 “good organization of documentation”, which has been prioritized as the topmost critical or significant success element for component-based outsourced software development. When comparing the findings of SLR with the AHP approach, we can see that CSF2 is the second most important success factor according to SLR, since CSF2 is categorized as the most important success factor in 31 of the 91 research articles. CSF1 “well-trained and skilled team” is the second most important success factor, whereas CSF7 “quality assurance” is the least important or key success factor. The remainder may be found in Table 26. In a similar way, we may prioritize the identified practices against each of the indicated success metrics.

**Table 15.** Pairwise comparisons in AHP are done using Saaty’s 9-point scale.

| Intensity of Importance | Definition                          | Explanation                                           |
|-------------------------|-------------------------------------|-------------------------------------------------------|
| 1                       | Equally important (EI)              | Both criteria are equally favored by judgement.       |
| 3                       | Weakly important (WI)               | One criterion is marginally favored by judgement.     |
| 5                       | Fairly important (FI)               | One criterion dominates the judgement.                |
| 7                       | Strongly important (SI)             | One criterion is substantially favored over the other.|
| 9                       | Absolutely more important (AMI)     | There is proof that one criterion is preferred over another. |
| 2, 4, 6, 8              | Intermediate values (IV)            | There can be no absolute judgement, hence a compromise is essential. |

**Table 16.** Pairwise comparisons for preparation level success factors.

| S. No | Success Factors | CSF 1 | CSF 2 | CSF 8 |
|-------|----------------|-------|-------|-------|
| 1     | CSF 1          | 1.00  | 0.20  | 2.00  |
| 2     | CSF 2          | 5.00  | 1.00  | 4.00  |
| 3     | CSF 8          | 0.50  | 0.25  | 1.00  |
| Sum   |                 | 6.50  | 1.45  | 7.00  |

**Table 17.** Normalized matrix for preparation level success factors.

| S. No | Success Factors | CSF 1 | CSF 2 | CSF 8 | Priority Weight |
|-------|----------------|-------|-------|-------|-----------------|
| 1     | CSF 1          | 0.15  | 0.14  | 0.29  | 0.1925          |
| 2     | CSF 2          | 0.77  | 0.69  | 0.57  | 0.6768          |
| 3     | CSF 8          | 0.08  | 0.17  | 0.14  | 0.1307          |
\[ \gamma_{\text{max}} = 3.096, \text{ CI} = 0.048, \text{ RI} = 0.58, \text{ CR} < 0.1, \sum = 1.000. \]

Table 18. Pairwise comparisons for standards level success factors.

| S. No | Success Factors | CSF 3 | CSF 4 | CSF 10 |
|-------|-----------------|-------|-------|--------|
| 1     | CSF 3           | 1.00  | 5.00  | 6.00   |
| 2     | CSF 4           | 0.20  | 1.00  | 2.00   |
| 3     | CSF 10          | 0.17  | 0.50  | 1.00   |
|       | Sum             | 1.37  | 6.50  | 9.00   |

\[ \gamma_{\text{max}} = 3.029, \text{ CI} = 0.015, \text{ RI} = 0.58, \text{ CR} < 0.1, \sum = 1.000. \]

Table 19. Normalized matrix for standard level success factors.

| S. No | Success Factors | CSF 3 | CSF 4 | CSF 10 | Priority | Weight |
|-------|-----------------|-------|-------|--------|----------|--------|
| 1     | CSF 3           | 0.73  | 0.77  | 0.67   | 0.72     | 0.72   |
| 2     | CSF 4           | 0.15  | 0.15  | 0.22   | 0.17     | 0.17   |
| 3     | CSF 10          | 0.12  | 0.08  | 0.11   | 0.10     | 0.10   |
|       | Sum             | 1.00  | 1.00  | 1.00   | 1.00     | 1.00   |

\[ \gamma_{\text{max}} = 2, \text{ CI} = 0, \text{ RI} = 0, \text{ CR} < 0.1, \sum = 1.000. \]

Table 20. Pairwise comparisons for integration level success factors.

| S. No | Success Factors | CSF 5 | CSF 6 |
|-------|-----------------|-------|-------|
| 1     | CSF 5           | 1.00  | 2.00  |
| 2     | CSF 6           | 0.50  | 1.00  |
|       | Sum             | 1.50  | 3.00  |

\[ \gamma_{\text{max}} = 2, \text{ CI} = 0, \text{ RI} = 0, \text{ CR} < 0.1, \sum = 1.000. \]

Table 21. Normalized matrix for integration level success factors.

| S. No | Success Factors | CSF 5 | CSF 6 | Priority | Weight |
|-------|-----------------|-------|-------|----------|--------|
| 1     | CSF 5           | 0.67  | 0.67  | 0.67     | 0.67   |
| 2     | CSF 6           | 0.33  | 0.33  | 0.33     | 0.33   |
|       | Sum             | 1.00  | 1.00  | 1.00     | 1.00   |

\[ \gamma_{\text{max}} = 2, \text{ CI} = 0, \text{ RI} = 0, \text{ CR} < 0.1, \sum = 1.000. \]

Table 22. Pairwise comparisons for prominence level success factors.

| S. No | Success Factors | CSF 7 | CSF 9 |
|-------|-----------------|-------|-------|
| 1     | CSF 7           | 1.00  | 0.20  |
| 2     | CSF 9           | 5.00  | 1.00  |
|       | Sum             | 6.00  | 1.20  |

\[ \gamma_{\text{max}} = 2, \text{ CI} = 0, \text{ RI} = 0, \text{ CR} < 0.1, \sum = 1.000. \]

Table 23. Normalized matrix for prominence level success factors.

| S. No | Success Factors | CSF 7 | CSF 9 | Priority | Weight |
|-------|-----------------|-------|-------|----------|--------|
| 1     | CSF 7           | 0.17  | 0.17  | 0.17     | 0.17   |
| 2     | CSF 9           | 0.83  | 0.83  | 0.83     | 0.83   |
|       | Sum             | 1.00  | 1.00  | 1.00     | 1.00   |

\[ \gamma_{\text{max}} = 2, \text{ CI} = 0, \text{ RI} = 0, \text{ CR} < 0.1, \sum = 1.000. \]

Table 24. Pairwise comparisons between level wise success factors.

| S. No | Levels       | Preparation | Standards | Integration | Prominence |
|-------|--------------|-------------|-----------|-------------|------------|
| 1     | Preparation  | 1.00        | 5.00      | 4.00        | 7.00       |
| 2     | Standards    | 0.20        | 1.00      | 0.50        | 3.00       |
| 3     | Integration  | 0.25        | 2.00      | 1.00        | 3.00       |
Table 25. Normalized matrix between level wise success factors.

| S. No | Levels       | Preparation | Standards | Integration | Prominence | Priority Weight |
|-------|--------------|-------------|-----------|-------------|------------|----------------|
| 1     | Preparation  | 0.628       | 0.600     | 0.686       | 0.500      | 0.603          |
| 2     | Standards    | 0.126       | 0.120     | 0.086       | 0.214      | 0.136          |
| 3     | Integration  | 0.157       | 0.240     | 0.171       | 0.214      | 0.196          |
| 4     | Prominence   | 0.090       | 0.040     | 0.057       | 0.071      | 0.065          |
|       | Sum          | 1.000       | 1.000     | 1.000       | 1.000      | 1.000          |

**γ** max = 4.101, **CI** = 0.034, **RI** = 0.90, **CR** = 0.037 < 0.1, ∑ = 1.000.

Table 26. Summary of local and global weights success factors and their rankings.

| Categories/Level | Weight | Critical Success Factors                               | Local Weight | Local Rank | Global Weight | Global Rank |
|------------------|--------|--------------------------------------------------------|--------------|------------|---------------|-------------|
| Prominence       | 0.261  | Assurance of quality standards CSF 7                   | 0.333        | 9          | 0.087         | 10          |
|                  |        | Proper security CSF 9                                  | 1.667        | 3          | 0.436         | 7           |
| Integration      | 0.813  | Well-defined analysis and testing CSF 5                | 1.333        | 4          | 1.084         | 4           |
|                  |        | Well-defined integration CSF 6                         | 0.667        | 5          | 0.542         | 6           |
| Standards        | 0.549  | Proper standard for designing CSF 3                    | 2.213        | 1          | 1.214         | 3           |
|                  |        | Well-defined architecture CSF 4                        | 0.525        | 7          | 0.288         | 8           |
|                  |        | Trusted and certified CSF 10                           | 0.311        | 10         | 0.171         | 9           |
| Preparation      | 2.520  | Highly competent team (training + skills) CSF 1        | 0.589        | 6          | 1.485         | 2           |
|                  |        | Good organization of documentation CSF 2               | 2.162        | 2          | 5.449         | 1           |
|                  |        | Well-defined procedure for component selection CSF 8   | 0.396        | 8          | 0.998         | 5           |

Figure 3. Local and global weight comparison of the success factors.

5. Limitations

The writers of all of these research papers were not supposed to describe the genuine reasons why these success criteria have a beneficial impact on CBOSD outsourcing software vendors. These could be because most research papers were literature reviews, surveys,
and experience reports, all of which are liable to publication bias. Our SLR approach may have missed some comparable publications due to the increasing quantity of papers published in CBSD from a vendor perspective. Furthermore, several search engines, such as Google Scholar, did not provide complete access for paper extraction.

6. Conclusions and Future Work

With the help of SLR, we first discovered a list of 16 success factors, from which we combined several factors to arrive at the final list of 10 success factors given in Table 2. These 10 factors are marked as critical: well-trained and skilled team (41%), proper component selection (31%), using standards for design (29%), well-defined architecture (26%), well-defined analysis and testing (29%), well-defined integration (29%), assurance of quality standards (25%), good organization of documentation (25%), well-organized security (23%), and proper certification (21%). The standard 46 practices for these 10 critical success factors were also determined from the literature. The software vendor organization should concentrate on these 46 practices to increase these 10 essential success factors. We applied the linear-by-linear chi-square statistical approach to detect the “main alteration” among several time decades in order to find significant variations among these success criteria. We found that there was just one significant variation between these time decades for the success component “quality assurance” among 10 success factors with \( p \) less than 0.05. This success factor was not detected during the decade 1990–2000; it was reported as 0% and 17% in 2001–2010, and 39% in 2011–2019. This shows that, with the exception of the 1990s and 2000s, every other decade considered “quality assurance” to be an important success factor. At the end, applied AHP method to prioritize and analyze the specified success criteria of CBOSD. Our research aims to make the software outsourcing vendor organization more efficient by incorporating the CBSD concept. Apart from the found and discussed success factors, the future work of our research will validate the identified success factors and find practices for these success aspects through empirical studies [51]. We also intend to undertake a case study in the relevant software vendor’s organization, similar to the Capability Maturity Model Integration (CCMI) model, to identify each vendor organization level of our suggested model and, lastly, to support them in the CBSD approach. Furthermore, we intend to prioritize and analyze these CBOSD success factors in the future by using the Fuzzy TOPSIS technique to find the most significant and critical success factors among those found.

Author Contributions: Conceptualization, A.W.K. and S.U.K.; formal analysis, J.K. and Y.L.; funding acquisition, Y.L.; investigation, A.W.K., S.U.K., H.S.A., F.K., and Y.L.; methodology, A.W.K. and S.U.K.; project administration, S.U.K. and J.K.; resources, H.S.A., F.K., and Y.L.; software, S.U.K.; validation, F.K. and Y.L.; visualization, S.U.K.; writing—original draft, A.W.K. and S.U.K.; writing—review and editing, A.W.K., S.U.K., H.S.A., F.K., J.K., and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation of Korea (NRF) grant 2022R1G1A1003531 and Institute of Information & communications Technology Planning & Evaluation (IITP) grant IITP-2022-2020-0-101741, RS-2022-00155885 funded by the Korea government (MSIT).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. List of Final Papers Selected through slr for CBOSD

1. M. A. Akbar. A. A. Khan: A. W. Khan: S. J. J. o. S. E. Mahmood and Process, Requirement change management challenges in gsd: An analytical hierarchy process approach, 32 (2020), no. 7, e2246.
2. M. O. Khan, A. Mateen, A. R. J. A. J. o. S. E. Sattar and Applications, Optimal performance model investigation in component-based software engineering (cbse), 2 (2013), no. 6, 141-149.140.
36. M. Goulão and F. B. e Abreu, *The quest for software components quality*, Proceedings 26th Annual International Computer Software and Applications, IEEE, 2002, pp. 313-318.

37. A. Makarenko, A. Brooks and T. Kaupp, *Orca: Components for robotics*, International Conference on Intelligent Robots and Systems (IROS), Citeseer, 2006, pp. 163-168.

38. D. Kunda and L. Brooks, *Identifying and classifying processes (traditional and soft factors) that support cots component selection*, (2000).

39. E. Braf, “Organisationers kunskapsverksamheter: En kritisk studie av “knowledge management””, Institutionen för datavetenskap, 2000.

40. D. Kunda, L. J. I. Brooks and S. Technology, *Assessing organisational obstacles to component-based development: A case study approach*, 42 (2000), no. 10, 715-725.

41. J. Gao, *Component testability and component testing challenges*, Proceedings of International Workshop on Component-based Software Engineering (CBSE2000), held in conjunction with the 22nd International Conference on Software Engineering (ICSE2000), Citeseer, 2000, pp.

42. K. C. Ukaoha, O. O. Ajayi, S. C. J. I. o. I. C. Chiemeke and I. Sciences, *Assessing the stability of selected software components for reusability*, 19 (2019), no. 2, 1-16.

43. A. P. Singh, P. J. I. J. o. C. A. E. Tomar and Technology, *Rule-based fuzzy model for reusability measurement of a software component*, 9 (2017), no. 4, 465-479.

44. Ratneshwer and A. K. J. i. o. c. a. i. t. Tripathi, *lim-cbsc: An integrated maturity model for cbse*, 46 (2013), no. 4, 323-336.

45. R. Gonzales, M. J. J. o. S. Torres, Cybernetics, and Informatics, *Issues in component-based development: Towards specification with adls*, 4 (2006), no. 5, 49-54.

46. Z. Javed, A. R. Sattar, M. S. J. G. J. o. C. S. Faridi and Technology, *Unsolved tricky issues on cots selection and evaluation*, (2012).

47. C. Werner, L. Murta, A. Marinbo, R. Santos and M. Silva, *Towards a component and service marketplace with brecho library*, Proceedings of the IADIS International Conf. WWW/Internet, 2009, pp. 567-574.

48. S. Becker, L. Grunske, R. Mirandola and S. Overhage, “Performance prediction of component-based systems”, Architecting systems with trustworthy components, Springer, 2006, pp. 169-192.

49. S. D. Kim and J. H. Park, *C-qm: A practical quality model for evaluating cots components*, Applied Informatics, Citeseer, 2003, pp. 991-996.

50. M. A. Martinez and A. Toval, *Cotsrc: A components selection method based on requirements engineering*, Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), IEEE, 2008, pp. 220-223.

51. B. Suleiman, *Commercial-off-the-shelf software development framework*, 19th Australian Conference on Software Engineering (aswec 2008), IEEE, 2008, pp. 690-695.

52. J. K. Lee, S. J. Jung, S. D. Kim, W. H. Jang and D. H. Ham, *Component identification method with coupling and cohesion*, Proceedings Eighth Asia-Pacific Software Engineering Conference, IEEE, 2001, pp. 79-86.

53. M. A. Rothenberger, K. J. Dooley, U. R. Kulkarni and N. J. I. T. o. S. E. Bastani and K. Engineering, *Strategies for software reuse: A principal component analysis of reuse practices*, 29 (2003), no. 9, 825-837.

54. A. Alsumait and S. Habib, *Usability requirements for cots based systems*, Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services, 2009, pp. 399-399.

55. A. Alvaro, E. Santana de Almeida and S. J. A. S. E. N. Romero de Lemos Meira, *A software component quality framework*, 35 (2010), no. 1, 1-18.

56. E. Ferguson and E. K. Mugisa, *An approach for incremental certification of software components*, Software Engineering Research and Practice, 2008, pp. 656-660.

57. E. B. Asuquo, *Quality indicators for effective component-based software application modernization*.

58. P. Banerjee, A. J. J. o. I. S. Sarkar and Applications, *Quality evaluation of component-based software: An empirical approach*, 11 (2018), no. 12, 80.

59. I. Crnkovic and M. Larsson, *A case study: Demands on component-based development*, Proceedings of the 22nd international conference on Software engineering, 2000, pp. 23-31.

60. M. Forsell, *Improving component reuse in software development*, Jyväskylän yliopisto, 2002.

61. D. Kunda and L. Brooks, *Identifying processes that support cots-based systems in developing countries*.

62. N. S. J. A. S. e. n. Gill, *Dependency and interaction oriented complexity metrics of component-based systems*, 33 (2008), no. 2, 1-5.

63. M. Palviainen, A. Evesti, E. J. J. o. S. Ovaska and Software, *The reliability estimation, prediction and measuring of component-based software*, 84 (2011), no. 6, 1054-1070.

64. N. Upadhyay, B. M. Deshpande and V. P. Agarwal, *Macbss: Modeling and analysis of component-based software system*, 2009 WRI World Congress on Computer Science and Information Engineering, IEEE, 2009, pp. 595-601.

65. E. Mauda, A. Menolli, A. Malucelli, A. Santin and S. Reinheir, *Quality model for internal security features of software components*.

66. S. Mahmood, *Towards component-based system integration testing framework*, Proceedings of the World Congress on Engineering, 2011, pp.

67. T. Gao, H. Ma, I.-L. Yen, L. Khan, F. J. I. J. o. S. E. Bastani and K. Engineering, *A repository for component-based embedded software development*, 16 (2006), no. 04, 523-552.

68. M. Larsson, “Predicting quality attributes in component-based software systems”, 2004.

69. V. Cortellessa, F. Marinelli and P. Potena, *Automated selection of software components based on cost/reliability tradeoff*, European Workshop on Software Architecture, Springer, 2006, pp. 66-81.
70. M. Ruiz, I. Ramos and M. Toro, *Using dynamic modeling and simulation to improve the cots software process*, International Conference on Product Focused Software Process Improvement, Springer, 2004, pp. 568-581.

71. S. Sedigh-Ali, A. Ghafoor and R. A. Paul, “A metrics-guided framework for cost and quality management of component-based software”, *Component-based software quality*, Springer, 2003, pp. 374-402.

72. D. J. C.-B. S. Q. Kunda, *Stace: Social technical approach to cots software evaluation*, (2003), 64-84.

73. R. P. Simão and A. D. Belchior, “Quality characteristics for software components: Hierarchy and quality guides”, *Component-based software quality*, Springer, 2003, pp. 184-206.

74. I. J. U. h. w. m. m. s. p. p. Crnkovic, *Component-based software engineering: Building systems from software components*.

75. H. Jain, P. Vitharana and F. M. J. A. S. D. t. D. F. A. i. I. S. Zahedi, *An assessment model for requirements identification in component-based software development*, 34 (2003), no. 4, 48-63.

76. G. Kumar and P. K. J. A. S. E. N. Bhatia, *Neuro-fuzzy model to estimate & optimize quality and performance of component based software engineering*, 40 (2015), no. 2, 1-6.

77. H. Yin, “Mode switch for component-based multi-mode systems”, Mälardalen University, 2012.

78. S. S. Thapar, P. Singh and S. I. A. S. E. N. Rani, *Reusability-based quality framework for software components*, 39 (2014), no. 2, 1-5.

79. M. S. Sitaraman, G. Kulczycki, J. Krone, W. F. Ogden and A. N. Reddy, *Performance specification of software components*, Proceedings of the 2001 symposium on Software reusability: putting software reuse in context, 2001, pp. 3-10.

80. S. K. Alaimagal and R. J. A. S. E. N. Srinivasan, *The need for transforming the cots component quality evaluation standard mirage to reality*, 34 (2009), no. 5, 1-4.

81. W. B. Frakes and S. J. I. s. Isoda, *Success factors of systematic reuse*, 11 (1994), no. 5, 14-19.

82. I. Jacobson, M. Griss and P. Jonsson, *Software reuse: Architecture, process and organization for business success*, ACM Press/Addison-Wesley Publishing Co., 1997.

83. B. Tekumalla, *Status of empirical research in component based software engineering—a systematic literature review of empirical studies*, (2012).

84. L. M. Dunn, “An investigation of the factors affecting the lifecycle costs of cots-based systems”, University of Portsmouth, 2011.

85. M. M. Syvertse, “Selection and use of third-party software components: Study of a it consultancy firm”, Institut for datateknikk og informasjonsvitenskap, 2011.

86. S. U. Khan, M. Niazi and R. J. I. s. Ahmad, *Empirical investigation of success factors for offshore software development outsourcing vendors*, 6 (2012), no. 1, 1-15.

87. A. W. Khan and S. U. J. I. s. Khan, *Critical success factors for offshore software outsourcing contract management from vendors’ perspective: An exploratory study using a systematic literature review*, 7 (2013), no. 6, 327-338.

88. S. U. Khan, M. Niazi, R. J. o. s. Ahmad and software, *Factors influencing clients in the selection of offshore software outsourcing vendors: An exploratory study using a systematic literature review*, 84 (2011), no. 4, 686-699.

89. H. Chang, L. Mariani and M. Pezze, *In-field healing of integration problems with cots components*, 2009 IEEE 31st International Conference on Software Engineering, IEEE, 2009, pp. 166-176.

90. S. J. M. U. S. Larsson, *Key elements of the product integration process*, (2007), 78.

91. K. K. Chahal and H. Singh, *A metrics based approach to evaluate design of software components*, 2008 IEEE International Conference on Global Software Engineering, IEEE, 2008, pp. 269-272.

References

1. Khan, A.; Khan, K.; Amir, M.; Khan, M.N.A. A component-based framework for software reusability. *Int. J. Softw. Eng. Its Appl.* 2014, 8, 13–24.

2. Larsson, M. *Applying Configuration Management Techniques to Component-Based Systems*; Uppsala University: Uppsala, Sweden, 2000.

3. Crnkovic, I.; Sentilles, S.; Vulgarakis, A.; Chaudron, M.R. A classification framework for software component models. *IEEE Trans. Softw. Eng.* 2010, 37, 593-615.

4. Chopra, S.; Nautiyal, L.; Malik, P.; Ram, M.; Sharma, M.K. A non-parametric approach for survival analysis of component-based software. *Int. J. Math. Eng. Manag. Sci.* 2020, 5, 309.

5. Abbas, S.F.; Shahzad, R.K.; Humayun, M.; Jhanji, N.Z.; Alamri, M. SOA Issues and their Solutions through Knowledge Based Techniques—A Review. *Int. J. Comput. Sci. Net.* 2019, 19, 8–21.

6. Ajayi, O.O.; Chiemeke, S.C.; Ukaoha, K.C. Comparative analysis of software components reusability level using gfs and ANFIS soft-computing techniques. In Proceedings of 2019 IEEE AFRICON, Accra, Ghana, 25–27 September 2019; pp. 1–8.

7. Wang, B.; Chen, Y.; Zhang, S.; Wu, H. Updating model of software component trustworthiness based on users feedback. *IEEE Access* 2019, 7, 60199–60205.

8. Mcllroy, M.D.; Buxton, J.; Naur, P.; Randell, B. Mass-produced software components. In Proceedings of 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany, 7–11 October 1968; pp. 88–98.

9. Khan, A.W.; Khan, S.U.; Khan, F. A Case Study Protocol for Outsourcing Contract Management Model (OCMM). *J. Softw.* 2017, 12, 348–354.

10. Benazeer, S.; Bruyn, P.D.; Verelst, J. Applying the concept of modularity to it outsourcing: A financial services case. In *Workshop on Enterprise and Organizational Modeling and Simulation*; Springer: Cham, Switzerland, 2017; pp. 68–82.
11. Bai, Y.; Chen, L.; Yin, G.; Mao, X.; Deng, Y.; Wang, T.; Lu, Y.; Wang, H. Quantitative analysis of learning data in a programming course. In *International Conference on Database Systems for Advanced Applications*; Springer: Cham, Switzerland, 2017; pp. 436–441.

12. Chouambe, L.; Klatt, B.; Krognmann, K. Reverse engineering software-models of component-based systems. In Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering, Athens, Greece, 1–4 April 2008; pp. 93–102.

13. Islam, F. An Effective Approach for Evaluation and Selection of Component. *Comput. Eng. Intell. Syst.* 2017, 8, 1–6.

14. Abdellatif, M.; Sultan, A.B.M.; Ghani, A.A.A.; Jabar, M.A. A mapping study to investigate component-based software system metrics. *J. Syst. Softw.* 2013, 86, 587–603.

15. Karthika, O.R.; Rekha, C. An Analysis of Maturity Model for Software Reusability – A Literature Study. *Int. J. Sci. Res. Comput. Sci. Appl. Manag. Stud.* 2018, 7, 1–5.

16. Bunse, C.; Gross, H.-G.; Peper, C. Embedded system construction—evaluation of model-driven and component-based development approaches. In *International Conference on Model Driven Engineering Languages and Systems*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 66–77.

17. Mahmood, K.; Ahmad, B.J.C.E.; Systems, I. Testing strategies for stakeholders in Component Based Software Development. *Comput. Eng. Intell. Syst.* 2012, 3, 101–105.

18. Matsumura, K.; Yamashiro, A.; Tanaka, T.; Takahashi, I. Modeling of software reusable component approach and its case study. In Proceedings of the Fourteenth Annual International Computer Software and Applications Conference, Madrid, Spain, 10–14 June 2016; pp. 307–313.

19. Iqbal, S.; Khalid, M.; Khan, M. A Distinctive suite of performance metrics for software design. *Int. J. Softw. Eng. Its Appl.* 2013, 7, 197–208.

20. Naur, P. *Software Engineering-Report on a Conference Sponsored by the NATO Science Committee Garimisch, Germany*; Scientific Affairs Division, NATO: Brussels, Belgium, 1968.

21. Farooqi, M.M.; Shah, M.A.; Wahid, A.; Akhunzada, A.; Khan, F.; Ali, I. Big data in healthcare: A survey. In *Applications of Intelligent Technologies in Healthcare*; Springer: Cham, Switzerland, 2019; pp. 143–152.

22. Ahmad, S.; Mehmood, F.; Khan, F. and Whangbo, T.K., 2022. Architecting Intelligent Smart Serious Games for Healthcare Applications: A Technical Perspective. *Sensors* 2022, 22, pp.810–819.

23. Khan, S.U.; Khan, A.W.; Khan, F.; Khan, M.A.; Whangbo, T. Critical Success factors of Component-Based Software Outsourcing Development from Vendors’ Perspective: A Systematic Literature Review. *IEEE Access* 2021.

24. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; Technical Report by UK Economics and Physical Sciences Research Council. 2007.

25. Griss, M.L. CBSE Success Factors: Integrating architecture, process, and organization. In Component-Based Software Engineering: *Putting the Pieces Together*; Addison Wesley: Upper Saddle River, NJ, USA, 2001; pp. 143–160.

26. Ilyas, M.; Khan, S.U. An empirical investigation of the software integration success factors in GSD environment. In Proceedings of the 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA), London, UK, 7–9 June 2017; pp. 255–262.

27. Ilyas, M.; Khan, S.U. Software integration in global software development: Challenges for GSD vendors. *J. Softw. Evol. Process* 2017, 29, e1875.

28. Ilyas, M.; Khan, S.U. Practices for software integration success factors in GSD environment. In Proceedings of the 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, Japan, 26–29 June 2016; pp. 1–6.

29. Kaur, A.; Mann, K.S. Component selection for component based software engineering. *Int. J. Comput. Appl.* 2010, 2, 109–114.

30. Khan, M.O.; Mateen, A.; Sattar, A.R. Optimal performance model investigation in component-based software engineering (CBSE). *Am. J. Softw. Eng. Appl.* 2013, 2, 140, 141–149.

31. Sauer, J. Architecture-centric development in globally distributed projects. In *Agility across Time and Space*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 321–329.

32. Crnkovic, I. Component-based software engineering—New challenges in software development. *Softw. Focus* 2001, 2, 127–133.

33. Ravichandran, T. Organizational assimilation of complex technologies: An empirical study of component-based software development. *IEEE Trans. Eng. Manag.* 2005, 52, 249–268.

34. Sharp, J.H.; Ryan, S.D. A theoretical framework of component-based software development phases. *ACM SIGMIS Database Adv. Inf. Syst.* 2010, 41, 56–75.

35. Vijayalakshmi, K. Reliability improvement in component-based software development environment. *Int. J. Inf. Syst. Change Manag.* 2011, 5, 99–123.

36. Chopra, S.; Sharma, M.; Nautiyal, L. Analogous Study of Component-Based Software Engineering Models. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* 2017, 7, 597–603.

37. Latif, M.; Ahmed, M.; Tahir, R.M.; Ashraf, M.K.; Farooq, U. Estimation and Enhancement of the Component Reusability in Component Based Software Development. In Proceedings of the MDSRC—2017 Proceedings, Wah, Pakistan, 27–28 December 2017.

38. Ratneshwer, Tripathi, A.K. IIM-CBSE: An integrated maturity model for CBSE. *Int. J. Comput. Appl. Technol.* 2013, 46, 323–336.

39. Mir, I.A.; Quadri, S.M.K. Analysis and Evaluating Security of Component-Based Software Development: A Security Metrics Framework. *Int. J. Comput. Netw. Inf. Secur.* 2012, 11, 21–31.
40. Goulão, M.; e Abreu, F.B. The quest for software components quality. In Proceedings of the 26th Annual International Computer Software and Applications, Los Alamitos, CA, USA, 27 June–1 July 2022; pp. 313–318.
41. Liu, W.; Pang, J.; Yang, S.; Li, N.; Du, Q.; Sun, D.; Liu, F. Research on security assessment based on big data and multi-entity profile. In Proceedings of 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2–14 March 2021; pp. 2028–2036.
42. Akbar, M.A.; Khan, A.A.; Khan, A.W.; Mahmood, S. Requirement change management challenges in GSD: An analytical hierarchy process approach. J. Softw. Evol. Process 2020, 32, e2246.
43. Zarbakshnia, N.; Wu, Y.; Govindan, K.; Soleimani, H. A novel hybrid multiple attribute decision-making approach for outsourcing sustainable reverse logistics. J. Clean. Prod. 2020, 242, 118461.
44. Wieland, G.; Zeiner, H. A Survey on criteria for smart home systems with integration into the analytic hierarchy process. In International Conference on Decision Support System Technology; Springer: Cham, Switzerland; pp. 55–66.
45. Khan, A.W.; Khan, M.U.; Khan, J.A.; Ahmad, A.; Khan, K.; Zamir, M.; Kim, W.; Ijaz, M.F. Analyzing and evaluating critical challenges and practices for software vendor organizations to secure big data on cloud computing: An AHP-based systematic approach. IEEE Access 2021, 9, 107309–107332.
46. Khan, A.W.; Zaib, S.; Khan, F.; Tarimer, I.; Seo, J.T.; Shin, J. Analyzing and Evaluating Critical Cyber Security Challenges Faced by Vendor Organizations in Software Development: SLR Based Approach. IEEE Access 2022, 10, 65044–65054.
47. Chatterjee, P.; Chakraborty, S. A comparative analysis of VIKOR method and its variants. Decis. Sci. Lett. 2016, 5, 469–486.
48. Lipovetsky, S. An interpretation of the AHP eigenvector solution for the lay person. Int. J. Anal. Hierarchy Process 2010, 2, 158-162.
49. Lipovetsky, S. An interpretation of the AHP global priority as the eigenvector solution of an ANP supermatrix. Int. J. Anal. Hierarchy Process 2011, 3, 70–78.
50. Saaty, T.L. A scaling method for priorities in hierarchical structures. J. Math. Psychol. 1977, 15, 234–281.
51. Laila, U.e.; Mahboob, K.; Khan, A.W.; Khan, F.; Taekeun, W. An Ensemble Approach to Predict Early-Stage Diabetes Risk Using Machine Learning: An Empirical Study. Sensors 2022, 22, 5247.