www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(2), 2026, 14-26. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2026.v17i2.1262

Systematic Literature Review on the Use of Cluster Analysis in Software Design Activities

Ángel J. Sánchez-García[1], Álvaro Barradas-Fernández[1], Oscar Alonso-Ramírez[1] and Xavier Limón[1]
[1] Facultad de Estadística e Informática, Universidad Veracruzana, Xalapa, Veracruz, México.
abf2606@gmail.com, {angesanchez, oalonso, hlimon}@uv.mx

Abstract. Cluster analysis is an unsupervised machine learning approach that groups data into homogeneous categories without the need for predefined labels. Although it was not originally developed for software engineering, this technique has increasingly been applied to support various activities in the software design phase. However, information about its use remains scattered across different studies. To address this gap, this work presents a systematic literature review synthesizing the state of the art on the application of cluster analysis in software design. Following a rigorous selection process, 14 primary studies published between 2019 and 2025 were identified from four digital libraries: IEEE Xplore, ACM Digital Library, Springer Link, and ScienceDirect. This review highlights the contexts in which clustering has been applied, emphasizing its predominant role in class decomposition tasks and the frequent adoption of the K-means algorithm, while also documenting the algorithms and tools used during design activities. Furthermore, the analysis discusses the benefits and challenges of adopting cluster analysis in this stage of development. The findings provide software engineering researchers and practitioners with a consolidated overview of the role of cluster analysis in software design, offering insights into its potential, limitations, and directions for future research and practice.
Keywords. Cluster Analysis, Unsupervised Machine Learning, Software Design, Software Engineering, Systematic Literature Review, Class Decomposition, K-means, Design Phase, Software Architecture, Empirical Studies.

Article Info
Received December 2, 2025
Accepted Jan 12, 2026

1 Introduction

According to [1], Software Engineering (SE) is defined as “an engineering discipline that deals with all aspects of software production”. We can understand this as a set of activities divided into processes in order to develop a high quality software product. SE is based on the Software Development Life Cycle (SDLC), an abstract functional model that represents the conceptualization of a system’s need, its realization, use, and evolution [2]. The phases that are most present in software development are specification, definition of requirements, system and software design, implementation and unit testing, and operation and maintenance [1]. The purpose of all these activities, organized in stages, is to be able to build a good quality software project. One of the most important phases in the software engineering process is software design. Software design is the process of “translating software requirements into graphical models which help programmers in the implementation of the software” [3].

The software design stage is a crucial phase in the software development process, resulting in a design specification document for further use in program development [4]. Investing in a solid design early in the process helps prevent serious issues in later stages such as implementation, testing, or deployment, where errors become more costly and complex to resolve. In order to produce a quality product that meets the client's requirements, it is necessary to develop models that meet various aspects of the software and that are the link between the established requirements and the way in which a programmer should develop the software. A flexible, maintainable, extensible, and re-usable software design “enables easier integration of new requirements” [5], so activities usually performed in the software design phase such as the identification of similar design artifacts, Searching

for similar artifacts for reuse and identification of similar or repeated documentation can be automated by cluster analysis algorithms.

On the other hand, artificial intelligence (AI) according to [6] can be understood as "systems or machines that can perform activities that require intelligence when performed by humans". Recently, AI has become a very popular topic in the software-related world. AI has allowed SE to mitigate some problems due to human bias and has favorably impacted the areas of software requirements [7], software design [8], and software testing [9]. By mixing both disciplines, they generate new areas of opportunity for research [10]. However, AI does not have special methodologies for the development of high-quality software, as well as methodologies that adapt to systems that are used for research and experimentation of artificial intelligence itself. For this reason, it is seen that SE can contribute to these needs to AI and therefore the collaboration of both disciplines is necessary to obtain better results in the two areas of research.

The reliability and times necessary to carry out the development of a software project are usually influenced by many factors, mainly the human factor, which can be supported by AI. The need for automation and autonomous decision making have given rise to the application of AI and Machine Learning (ML) in software development activities, however, failures and deficiencies still occur in the resulting software systems [11]. According to [12] Machine Learning 'is the area of study that helps computers learn automatically'. This area of AI offers a wide field of research, since it can provide new techniques that solve automation problems in software product development processes.

Machine Learning comprises three types of learning, supervised learning, unsupervised learning and reinforcement learning [13]. Supervised learning works under supervision and labeled data set, data that already knows the target. Unsupervised learning, unlike the previous one, works without supervision, that is, the data is not labeled and therefore only patterns are searched [13]. Reinforcement learning is very different from the other two types, as it is based on reward-based learning. In other words, doing a good job will result in a positive reward, otherwise a penalty. Reinforcement learning is composed of five steps: agent state, environment, reward, state and action [13].

This work focuses on the second type of learning, unsupervised learning, a technique called Cluster Analysis (CA), is a technique that seeks to group elements according to some similarity metric. According to [14], elements are grouped "according to the degree of similarity between them, they are divided into several groups, in this way similar objects constitute a set. This process is called the Grouping Process". With this collection of information, research areas could be proposed to improve the results of the use of cluster analysis. In addition, it may be useful for software engineers, presenting a new perspective on how the grouping and search of different software artifacts (diagrams, documents, among others) can be optimized in the different phases of the SE. In a scientific utility, this collection of information explains the current state of the application of CA in the phases of SE.

In conclusion, this work will provide a general overview for SE practitioners or researchers to know how Cluster Analysis can contribute to the various software design activities.

The rest of this article is structured as follows: Section 2 presents previous studies related to AI and CA in SE. Section 3 describes the research method used for this review. Section 4 shows the results, followed by the discussion of technical implications in Section 5. Section 6 presents the threats of validity. Finally, conclusions and future work are presented in Section 7.

2 Related Work

To the best of our knowledge, no secondary study (systematic literature review, systematic mapping study, or survey) has been identified through manual search that specifically addresses the use of CA in activities of the software design phase. Table 1 provides a summary of the reviews that we consider most closely related to our study.

Table 1. Related work focused on CA in SE (RQ: Research Question, SSV: Search String Validation, R: Reported, NR: Not Reported, QA: Quality Assessment, NS: Number of Studies, DS: Date of Studies)

Study	SSV	QA	NS	DS	RQ
Juziuk et al. [15]	NR	NR	39	1998-2012	RQ1: How are the patterns documented and what pattern templates are used? RQ2: How are the design patterns connected? RQ3: For what types of systems have the design patterns been applied? RQ4: How can the design patterns be classified?
Batarseh et al. [16]	NR	NR	190	1975-2017	RQ1: Is there sufficient intelligence in the SE lifecycle? RQ2: What does applying AI to SE entail?
Robles-Aguilar et al. [8]	NR	NR	36	2015-2020	RQ1: What Software Design activities have been carried out with AI? RQ2: What AI techniques have been used in Software Design? RQ3: In what design activities are AI techniques used and how many reports are of that use? RQ4: What software design artifacts have been made with AI techniques? RQ5: What is the purpose of using AI software design? RQ6: What are the most frequented venues to publish papers on this topic?
Chaudhry et al. [17]	NR	R	58	1995-2023	RQ1. What approaches and algorithms are currently available in clustering? RQ2. What are the benefits and drawbacks of various clustering techniques? RQ3. What are the clustering evaluation measures to consider when selecting a centroid finding method? RQ4. What are the applications or fields where some clustering algorithms outperform others?
Mecarder-Olivares et al. [18]	R	NR	15	2019-2024	RQ1 What clustering algorithms have been used in software development with agile methodologies to identify patterns? RQ2. In which development phases have clustering algorithms been most applied to identify patterns in software development with agile methodologies? RQ3. What types of systems have applied clustering algorithms to identify patterns in software development with agile methodologies? RQ4. What are the advantages of using clustering algorithms to identify patterns in software development with agile methodologies? RQ5. What are the challenges of using clustering algorithms to identify patterns in software development with agile methodologies?

In [15], a review in the area of software design is presented with the aim of reporting methods for detecting, documenting, and classifying design patterns. The authors note that clustering techniques can be applied to identify different types of design patterns. In [16], the authors conducted a review of the application of Artificial Intelligence (AI) in Software Engineering across all phases of the development process from 1975 to 2017. Specifically, for the design phase, only 12 studies applying AI were identified, focusing primarily on tasks such as self-adaptation to dynamic software architectures, applying self-adaptation to autonomous agents, and evaluating architectures. However, these tasks were mainly supported by expert systems or agents, without the involvement of clustering algorithms.

In [8], a systematic literature review (SLR) was conducted on design activities supported by AI in general between 2015 and 2020. The most frequently reported software design activities included the generation of design diagrams, architectural smell detection, design pattern detection, and product line architecture design. Nevertheless, the most common AI techniques were supervised learning methods, particularly for classification tasks, such as artificial neural networks, support vector machines, and decision trees. Notably, this study did report some work using K-means and K-medoids for design pattern classification.

In [17], the authors explored almost 20 years (1995–2023) of research on identifying patterns using unsupervised clustering algorithms. While this work is indeed based on clustering applications, it does not specifically target Software Engineering, and even less so activities within the software design phase.

Finally, in the most recent work [18], the authors conducted an SLR on clustering across different phases of software development. However, their focus was on agile development methodologies. They reported that the most widely used algorithms include K-means, hierarchical clustering, K-medoids, single linkage clustering, and complete linkage clustering,

among others. Their findings, however, were primarily related to software requirements and testing activities within agile methodologies.

A review of these secondary studies on clustering or AI in Software Engineering reveals a clear gap: no existing work provides practitioners with consolidated findings that help them understand clustering algorithms and tools in a way that facilitates software design activities. Moreover, there is a lack of studies that demonstrate how clustering can support the identification of patterns and the detection of similar artifacts in this critical phase of software development.

3 Research method

To carry out the systematic review, the guidelines of the methodology proposed by Kitchenham et al. [19] was followed. This methodology consists of three stages: planning, conduction and reporting.

3.1 Planning Stage

At this stage, research questions are posed, search terms are defined, the search string is constructed, and selection criteria are established.

3.1.1 Research Questions

For this review, five research questions were established, which aim to find relevant information related to the application of Cluster Analysis in software design activities. Table 2 shows the research questions that guide this review.

Table 2. Research questions.

ID	Research question	Motivation
RQ1	What are the activities in the software design engineering where CA has been used?	It is important to know the different activities within software design used in CA to identify which ones have the greatest impact.
RQ2	What are the CA algorithms used in software design activities?	It is important to know the different CA algorithms that are used in software design to provide valuable information on how to use cluster analysis to improve the requirements phase.
RQ3	What tools are used for CA in software design activities?	Knowledge of the CA tools is important, since the choice of the right tool is an important factor in achieving successful results.
RQ4	What are the advantages of CA in software design activities?	Software design is of utmost importance, knowing the advantages of CA in the activities in this phase can provide valuable information to improve the generation and recovery of design artifacts.
RQ5	What are the challenges of CA in software design activities?	Knowing the challenges of CA in software design activities can provide valuable insights into how to address challenges and get the most out of CA.

3.1.2 Data sources

Table 3 shows the selected data sources for the automated search, which were selected from the software engineering research ranking presented by Zhang et. al. [20].

Table 3. Data sources

Data source	Web site
ACM Digital Library	https://dl.acm.org/
IEEE Xplore	https://ieeexplore.ieee.org/
Science Direct	https://www.sciencedirect.com/
Springer Link	https://link.springer.com/

3.1.3 Search Terms

From the research questions, the search terms along with their related terms were selected and they are shown in Table 4.

Table 4. Search terms.

Keyword	Related Terms
Cluster Analysis	Clustering
Machine Learning	-
Design	Software Design
Technique	Method, approach, algorithm
Tools	Software, Application

3.1.4 Search string

With the Table 4 it can be seen that five key terms were found, with cluster analysis and software design being the terms of greatest interest for this research. Seven search strings were constructed with these terms, which were tested through search iterations. For the evaluation of the strings, the Quasi-gold standard was used, using the metrics proposed by Zhang [20] that will be described below.

In order to perform the automated search, various search strings were performed with various combinations of the search terms. To evaluate the search strings, the guidelines proposed by Zhang [20] were used, the relevant articles are used to evaluate the sensitivity and precision, mentioned by the methodology as Recall and Precision respectively; same as are calculated with (1) and (2) respectively.

$$\text{Recall} = \text{Retrieved relevant studies} / \text{Relevant studies.} \quad (1)$$

$$\text{Precision} = \text{Retrieved relevant studies} / \text{Retrieved studies} \quad (2)$$

The seven search strings where the recall and precision metrics were applied based on five relevant studies found manually. The selected search string was the one that reached a recall = 1 (although a recall = 0.8 is acceptable [20]) and precision = 1.8 which is shown below.

"software design" AND ("machine learning" OR "clustering")

3.1.5 Selection Criteria

Table 5 presents the inclusion and exclusion criteria that were used for the selection of primary studies in the automated search.

Table 5. Selection criteria.

ID	Inclusion criteria
IC1	Studies were published between 2019 and 2025.
IC2	Studies written in the English language.
IC3	Full access to the studio is available.
IC4	The title and abstract answer at least one research question.
IC5	The study answers at least one research question.
ID	**Exclusion criteria**
EC1	In multidisciplinary sources, the study is not in engineering or computer science.
EC2	The study is a slide presentation, technical report or chapter of a book.
EC3	The study is duplicated.

For the selection of primary studies, automated searches are performed with previously selected search strings and the results are subsequently filtered by applying the inclusion and exclusion criteria. These criteria were assigned in four stages. Fig. 1 shows the stages for the selection process of primary studies. Table 6 shows the results of selection process of the primary studies.

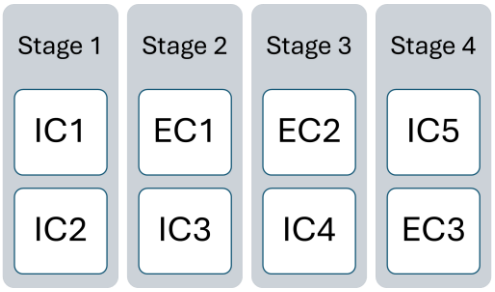

Fig. 1. Primary studies selection process.

3.2 Reporting

The narrative synthesis was conducted by following the procedures outlined by Popay et al. [21]. This process involved analyzing the findings and organizing them according to each of the research questions. To achieve this, the results were grouped by research question, and relevant text fragments were extracted from the primary studies. These excerpts were then tabulated to provide structured responses to the research questions. Once the tabulation was completed, related data were further clustered, resulting in a systematic classification of the evidence in line with the research questions.

4. Results

Table 6 shows the results of the selection process of primary studies by phase, where it can be seen that in the initial search 15,196 studies were obtained. In Table 6 It can be observed that the stage in which most articles were excluded was the first one, primarily due to the year of publication criterion. In stage 3, the criterion that led to the highest number of exclusions was IC4, which involved reviewing the title and abstract in relation to the research questions.

Table 6. Results of the primary studies selection process.

Source of information	Initial search	Stage 1	Stage 2	Stage 3	Stage 4
IEEE Xplore	460	231	231	8	6
SpringerLink	4,422	2,337	2,001	10	1
ACM Digital Library	3,130	1,825	513	4	1
ScienceDirect	7,185	3,464	2,071	6	6
Total	15,197	7,857	4,816	28	14

Table 7 presents the primary studies selected by source, showing that IEEE Xplore and Springer Link were the venues with the highest number of publications. Figure 2 illustrates the distribution by year of publication, where a relatively constant output can be observed; however, as of July 2025, no primary study directly related to the topic was identified.

Table 7. Selected primary studies.

Data source	References
IEEE Xplore	[22][23][24][25][26][27]
ACM Digital Library	[28]
ScienceDirect	[29]
Springer Link	[30][31][32][33][34][35]

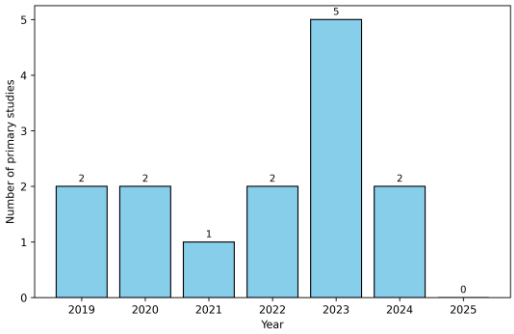

Fig. 2. Number of selected studies per year.

4.1 RQ1. What are the activities in the software design engineering where CA has been used?

As shown in Table 8, the activities in which CA is applied are primarily related to improving the understanding of software systems by decomposing them into less complex subsystems. CA groups the elements required in software design into simpler packages, thereby facilitating both comprehension and management of the system. An additional key application of CA is the identification and prediction of design errors, enabling their prevention or the establishment of contingency plans when necessary.

Table 8. Software design activities in which Cluster Analysis was used.

Activities of software design	Description	Studies
Predict defects in software design.	For error prediction in software design, Cluster Analysis has proven to be a valuable tool. It has been applied to assess the weight of risks and to group them into clusters, thereby facilitating their classification and the identification of preventive measures. In this context, Cluster Analysis is complemented by the XGBoost algorithm: while CA generates diverse datasets, XGBoost identifies patterns within them to predict potential errors. Although XGBoost is not specific to clustering, its role will be discussed in detail later.	[30][31][32]
Recover the software architecture	CA enables the grouping of software system components into clusters, allowing the recovery and better understanding of system architectures. It has been applied to organize and visualize collections of similar design models, as well as to classify software systems into categories with shared features, thereby simplifying the visualization of architectures and supporting their replication in future projects.	[22][27][28][33]
Break classes into packages or subsystems.	CA supports software design by decomposing classes into packages or subsystems that are easier to understand and manage. By segmenting data into homogeneous groups, it facilitates decision-making and the identification of meaningful patterns in software design.	[25][34]
Classify profiles of similar users.	Another application of CA in software design is automatic clustering to group users with similar profiles, thereby improving their management.	[23]

4.2 RQ2. What are the CA algorithms used in software design activities?

CA algorithms play a crucial role in uncovering patterns and structures within complex datasets, particularly when managing large volumes of unlabeled data such as the elements of a software system. These algorithms automate the grouping of data based on similarities, enabling the identification of clusters of elements that share common characteristics. From the analysis of

the primary studies, a variety of Cluster Analysis algorithms applied to software design activities were identified. Among these, the K-means algorithm emerged as the most prominent, and it will be discussed in greater detail below. Table 9 summarizes the CA algorithms employed in software design. Table 10 presents the comparison between the algorithms reported in Table 9.

Table 9. Cluster analysis algorithms identified for software design activities.

Cluster analysis algorithm	Description	Studies
K-means	The K-means algorithm was primarily employed to cluster classes based on their identifiers. It is applied to efficiently group data and uncover underlying structures in software design. The studies describe how metrics and vectors are used to configure the algorithm, which then partitions the vectors into k clusters, with k representing the number of groups defined by the user.	[23][24][25][27][28] [34]
Density-Based Spatial Clustering of Applications with Noise (DBSCAN).	To apply the DBSCAN algorithm, it must be configured with specific parameters, such as group density. Unlike other clustering methods, studies using DBSCAN did not need to predefine the number of clusters, as the algorithm automatically determines them based on data distribution.	[28][34]
Fast Clustering Algorithm (FCA).	The FCA algorithm is applied to the modularization of large-scale software systems. It operates on a dependency matrix derived from the source code and was specifically designed to address the shortcomings of traditional clustering algorithms, which often exhibit inefficiencies in time and space when handling large or very large systems.	[22]
Fuzzy C-means (FCM).	The FCM algorithm was used as part of the software modeling process. The main objective of using FCM was to address the imprecision and uncertainty present in the components when classifying the cluster. Before applying the FCM algorithm, the Empirical Model Decomposition (EMD) was used to extract the components that make up the groups; finally, the FCM was applied to create fuzzy sets that represent the data.	[29]
Mean Shift.	The Mean Shift algorithm was used for the organization of multi-dimensional groups, without the need to specify the exact cluster number. This algorithm helped manage large amounts of data.	[34]
Multi-level Greedy Modularity Clustering (MGMC).	The use of the MGMC algorithm focuses on evaluating the effectiveness of the recovery of the software architecture. The MGMC algorithm can be applied to dependency graphs to cluster them.	[33]

Table 10. Comparison of features reported by the clustering algorithms.

Cluster algorithm	Use it when	Advantages	Disadvantages
K-means	• Clusters are roughly spherical and well-separated [24].	• Very fast and scalable. • Easy to implement. • Works well with large datasets.	• Requires predefining k [28][34]. • Sensitive to outliers. • Assumes spherical clusters.
Density-Based Spatial Clustering of Applications with Noise (DBSCAN).	• Clusters have irregular shapes. • Detecting noise/outliers is important.	• Automatically detects number of cluster [34]. • Handles arbitrary-shaped clusters. • Good for noisy data [28].	• Struggles with varying densities. • Sensitive to hyperparameters (eps, minPts).
Fast Clustering Algorithm (FCA).	• Working with very large datasets where exact methods are too slow.	• Very fast for massive data. • Often parameter-free or low-parameter [22].	• Sometimes unstable depending on initialization.
Fuzzy C-means	• Objects can belong to	• Captures uncertainty and	• Requires setting cluster

(FCM).	multiple clusters.	partial membership [29].	count. • Sensitive to noise/outliers. Slower than K-means due to fuzzy membership computation.
Mean Shift.	• Clusters are non-spherical and unknown in number. • Data has clear density peaks.	• Does not require specifying number of clusters [34]. • Captures arbitrary cluster shapes.	• Highly sensitive to bandwidth parameter.
Multi-level Greedy Modularity Clustering (MGMC).	• Clustering is applied to graphs/networks.	• Detects hierarchical modular structures. • No need to specify number of clusters [33].	• Applicable mainly to graph-based data, not raw feature vectors.

4.3 RQ3. What tools are used for CA in software design activities?

From the analysis of the primary studies, several tools supporting the use of Cluster Analysis in software design activities were identified. Notable similarities among these tools allowed for their classification. As shown in Table 11, the results were not entirely as expected, since the two tools most directly associated with Cluster Analysis are platforms primarily oriented toward machine learning. In addition, while studies mention the use of programming languages for implementing Cluster Analysis algorithms, only one library specific to Python was reported. Furthermore, the complementary tool identified was the same across studies, providing support mainly for prediction and error detection

Table 11. Tools used for cluster analysis in the design of software.

Tools for CA	Description	Studies
WEKA (Waikato Environment for Knowledge Analysis)	WEKA is a platform used for machine learning, especially in data analysis and error prediction. WEKA can be used to compare the effectiveness of an optimized cluster model that combined logistic regression and tree classifier for the purpose of classifying and predicting faults in the software system.	[31]
Programming languages and libraries for using Cluster Analysis		
Python and sklearn	It is a Python machine learning library that is widely used in developing artificial intelligence and machine learning applications. It is a powerful tool that offers a wide range of functionalities for data processing and creating predictive models.	[29][34]
R project	The use of the R programming language is mentioned for the implementation of the FCM algorithm code, however, there is no specification of the language's own libraries used.	[29][34]
Matlab	MATLAB is a software platform, especially useful for the implementation of machine learning techniques, such as Cluster Analysis.	[31]
Java	The use of the Java programming language is mentioned for the implementation of the Cluster Analysis algorithm code, however, there is no specification of the language's own packages used.	[32]
C++	The use of the C++ programming language is mentioned for the implementation of the Cluster Analysis algorithm code, however, there is no specification of the language's own libraries used.	[32]
Complementary tools for Cluster Analysis		
SHAP (Shapley Additive exPlanations)	SHAP assigns an importance value (positive or negative) to each feature in a particular prediction. These values allow you to summarize important features and associate low and high feature values with an increase/decrease in the output values (prediction).	[30]

4.4 RQ4. What are the advantages of CA in software design activities?

From the analysis of the primary studies, 13 works reported advantages of applying Cluster Analysis in software design activities. These benefits mainly concern improving the understanding of software architectures, enhancing design robustness, and supporting better error prediction. Table 12 summarizes the advantages identified across the studies.

Table 12. Advantages of cluster analysis in the software design.

Advantages	Description	Studies
Better understanding of software architecture	The use of Cluster Analysis in software design activities allows you to reduce the complexity of a system by replacing sets of artifacts with clusters that group artifacts with similarities, which facilitates the reuse of existing artifacts, the analysis of software systems and the identification of common design patterns.	[22][24][26] [27][28][34] [35]
Better software robustness	The importance of design in the software development process must be highlighted, with the application of CA, facilitating the understanding of the system, its design is facilitated, and therefore, better robustness of the software is achieved.	[24][30][31] [32]
Better error prediction	CA in software design helps detect patterns and group data, clustering with previously detected errors, improving the prediction of possible errors.	[29][30][31] [32]
Automate software design processes	With CA many software design processes such as class decomposition, grouping of similar artifacts, and decomposition of the system into subsystems can be automated.	[25][32]
Greater savings of time and effort for designers	With CA applied in software design activities, as previously mentioned, class decomposition processes can be automated, thereby saving time and effort for designers.	[24][25][32]
Facilitate reconstruction of software architecture	CA provides benefits such as the ability to identify code dependencies, measure the impact of dependencies on the software architecture, and facilitate the reconstruction of the software architecture thanks to the breakdown of components and artifacts, grouped in clusters with similarities.	[35]
Greater efficiency in software programming.	Thanks to the acceleration of software planning and design processes, it causes a chain effect that helps save costs in development and in the efforts and time of developers.	[31]

4.5 RQ5. What are the challenges of CA in software design activities?

The analysis of the primary studies revealed nine works reporting challenges in applying Cluster Analysis to software design activities. These challenges mainly concern the difficulty of determining which tools and algorithms are most suitable for specific project contexts. Table 13 summarizes the challenges identified.

Table 13. Challenges of cluster analysis in the software design.

Challenges	Description	Studies
Identify optimal techniques and tools	The complexity of modern software systems creates a need for effective tools and techniques to manage that complexity. During software design it is difficult to choose the optimal clustering technique to predict defects in software design. It is emphasized that CA generally requires the integration of tools and processes external to Cluster Analysis itself in order to be able to apply it in the most efficient way.	[24][31][32] [34][35]
Predict the optimal number of clusters	The use of the K-means algorithm for CA in various software design activities was noted. One of the challenges identified in the studies concerns predicting the optimal number of clusters (K) from term matrices and	[25][28][34]

	preparing larger datasets for more detailed evaluations. Since the algorithm requires the configuration of parameters such as minimum distance and density thresholds, its application to software system design can be complex, particularly when determining the appropriate number of clusters to separate system components.	
Complement Cluster Analysis with methods to explain results	It is highlighted that CA must be complemented with methods that explain the results, such as principal components analysis in the case of clustering based on metrics and a method to obtain the most descriptive terms of each cluster in the case of clustering based on the chatbot vocabulary	[28]
Generalize results to other projects	Although CA shows to be effective in predicting defects in software design, a significant challenge to its application is that by having to focus the algorithms and tools on a specific case, it becomes a challenge to ensure that the result of the application of Cluster Analysis to the design of a software can be used in a different software design.	[30]
Ensure proper interpretation of results	The challenge of complementing the Cluster Analysis with other methods to explain the results obtained causes another challenge to arise and that is that it cannot be guaranteed that the adequate interpretation of the results obtained is carried out if the appropriate method is not used to complement the explanation of these	[32]
Identify necessary libraries and utilitie	Before starting the clustering process in software design activities, it is necessary to be clear about which programming language will be used, so that in this way it can be identified which libraries and utilities will be optimal for the use of CA.	[22]

5 Discussion of technical implications

The results of this review indicate that CA can play a meaningful role in software design by supporting the comprehension of complex architectures, facilitating modular decomposition, and contributing to error prediction. Grouping components into functional and reusable modules improves maintainability and system evolution, while the identification of anomalies or unusual patterns can enhance reliability and reduce design-related risks. These findings suggest that, when applied with care, CA can provide tangible benefits for both practitioners and researchers in software engineering.

At the same time, several challenges were identified. The complexity of current software systems complicates the choice of appropriate clustering algorithms and tools, since their effectiveness depends on project-specific characteristics. In addition, configuring parameters (such as the optimal number of clusters) remains a non-trivial task that directly impacts the quality of the results. These limitations emphasize the importance of combining CA with complementary techniques and validation strategies to ensure robust and meaningful outcomes. Furthermore, the synthesis obtained through this systematic review can be directly integrated into software design practices in industrial environments. The analyzed clustering techniques provide actionable guidance for structuring and reorganizing software artifacts, supporting early-stage design decisions, enhancing modularity, and identifying architectural inconsistencies. These findings suggest practical opportunities for professionals, such as selecting clustering algorithms according to data characteristics, incorporating automated clustering analysis into DevOps pipelines, and using cluster-based insights to guide refactoring and reengineering efforts. Overall, the reviewed evidence highlights how clustering methods can contribute to more informed, data-driven design processes in software engineering practice.

Finally, the absence of established standards or guidelines for applying CA in software design restricts the ability to generalize results or define best practices. This review, therefore, provides an evidence-based overview rather than prescriptive recommendations. For the research community, the findings highlight opportunities for methodological advances and comparative studies, while for practitioners they offer a reference to better understand both the advantages and the constraints of CA. In this way, the study lays the groundwork for future investigations and for the gradual integration of CA into broader software engineering practices.

6 Threats to validity

To mitigate potential bias in the search, selection, and analysis of studies, several practices were implemented. An introductory search was first conducted on the research topic, mainly focusing on Machine Learning and Cluster Analysis, to become

familiar with the terminology used in this domain. This allowed for a more accurate selection of terms during the manual search phase, which was performed following the methodology proposed by Zhang [20]. The evaluation of the search string was carried out using the four data sources, ensuring that the automated search was based on a validated query.

Nevertheless, some threats to validity remain. A primary concern relates to the automated search, as article accessibility was restricted by the institutional subscription. Consequently, potentially relevant studies available only through other subscriptions were not included. Another limitation is that the most recent studies published after the cutoff date may not have been considered, which could affect the completeness of the review.

Additional threats include the lack of venues directly dedicated to the topic, which limited the scope of potentially relevant studies. To mitigate this, primary studies were selected individually through manual search before optimizing the automated search. Moreover, there was a disproportionate distribution of studies across search engines, which could not be controlled or reduced, as it depends on third-party indexing policies and the disciplinary scope of each database. For example, Springer Link, being multidisciplinary, yielded fewer relevant studies, further constrained by restricted access.

7 Conclusions and Future work

This study reports the process and results of a systematic literature review aimed at identifying the application of Cluster Analysis (CA) in software design activities, following the guidelines of Kitchenham [19], the methodology of Zhang [20], and the narrative synthesis approach proposed by Popay [21]. To guide the review, five research questions were formulated and addressed, providing insights into the current state of CA in this context. The findings suggest that CA can be a valuable tool in software design, provided it is applied appropriately and its inherent challenges are considered. Specifically, CA facilitates the identification of related groups of characteristics, enhances the understanding of complex architectures, supports the detection of relevant patterns, and enables the grouping of components into functional and reusable modules (improving both maintainability and evolution). Furthermore, CA contributes to reengineering and system adaptation by clarifying software structure and organization. It may also aid in error prediction by detecting anomalies or unusual patterns in data.

However, the findings also highlight challenges. The complexity of modern software systems complicates the selection of optimal CA algorithms or tools, and the results are not always generalizable across different systems. Additionally, the absence of guidelines or standards for CA implementation in software design prevents the identification of recommended practices. As such, this work does not propose a set of best practices but rather provides an evidence-based account of the current state of CA in software design. The review is subject to limitations that may affect the depth of the findings, including restricted access to certain studies and the cutoff date for primary study selection.

As future work, three additional reviews are planned, following the same methodology, to investigate the application of CA in requirements engineering, construction activities, and software testing. This study is part of a broader monographic effort to analyze the role of CA across software engineering, targeting both practitioners and researchers interested in CA and software engineering.

References

1. Sommerville, I. (2011). *Software engineering* (9ª ed.). Pearson Education/Addison-Wesley.
2. ISO/IEC/IEEE. (2017). *Systems and software engineering — Software life cycle processes (ISO/IEC/IEEE 12207:2017)*.
3. Elmasry, I., Wassif, K., & Bayomi, H. (2021). *Extracting software design from text: A machine learning approach*. In *Proceedings of the 2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS)* (pp. 486–492). IEEE.
4. Chancheaw, S., & Prompoon, N. (2015). *Consistency verification between software design models and user interface design based on components relationships*. In *Proceedings of the 2015 2nd International Conference on Information Science and Security (ICISS)* (pp. 1–5). IEEE.
5. Sharma, T. (2012). *Quantifying quality of software design to measure the impact of refactoring*. In *Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops* (pp. 266–271). IEEE.
6. Kurzweil, R. (2012). *How to create a mind: The secret of human thought revealed*. Penguin.
7. Aguilar, A.R. Ocharán-Hernández J.O., and Sánchez-García, A.J. (2020). A systematic mapping study of artificial intelligence in software requirements. Res. Comput. Sci., vol. 149, no. 11, pp. 179–188.
8. Robles-Aguilar, A., Ocharán-Hernández, J. O., Sánchez-García, A. J., & Limón, X. (2021). *Software design and artificial intelligence: A systematic mapping study*. In *Proceedings of the 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)* (pp. 132–141). IEEE.

9. Bhandari, K., Kumar, K., & Sangal, A. L. (2023). *Artificial intelligence in software engineering: Perspectives and challenges*. In *Proceedings of the 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC)* (pp. 133–137). IEEE.
10. Khomh, F., Adams, B., Cheng, J., Fokaefs, M., & Antoniol, G. (2018). *Software engineering for machine-learning applications: The road ahead*. *IEEE Software, 35*(5), 81–84.
11. Simon, P. (2013). *Too big to ignore: The business case for big data*. John Wiley & Sons.
12. Gupta, V., Mishra, V. K., Singhal, P., & Kumar, A. (2022). *An overview of supervised machine learning algorithms*. En *Proceedings of the 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)* (pp. 87–92). IEEE.
13. Chen, H., & Liu, C. (2013). *Research and application of cluster analysis algorithm*. In *Proceedings of the 2013 2nd International Conference on Measurement, Information and Control* (Vol. 1, pp. 575–579). IEEE.
14. Juziuk, J., Weyns, D., & Holvoet, T. (2014). *Design patterns for multi-agent systems: A systematic literature review*. En *Agent-Oriented Software Engineering* (pp. 79–99). Springer.
15. Batarseh, F. A., Mohod, R., Kumar, A., & Bui, J. (2020). *The application of artificial intelligence in software engineering: A review challenging conventional wisdom*. En *Data democracy* (pp. 179–232). Elsevier.
16. Chaudhry, M., Shafi, I., Mahnoor, M., Vargas, D. L. R., Thompson, E. B., & Ashraf, I. (2023). *A systematic literature review on identifying patterns using unsupervised clustering algorithms: A data mining perspective*. *Symmetry, 15*(9), 1679.
17. Mercader-Olivares, I. E., Martínez-Moreno, P., Vergara-Camacho, J. A., Sánchez-García, A. J., & Ocharán-Hernández, J. O. (2024). *Cluster analysis in the identification of patterns in software development with agile methodologies: A systematic literature review*. In *Proceedings of the 2024 12th International Conference in Software Engineering Research and Innovation (CONISOFT)* (pp. 147–155). IEEE.
18. Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). *Evidence-based software engineering and systematic reviews*. CRC Press.
19. Zhang, H., Babar, M. A., & Tell, P. (2011). *Identifying relevant studies in software engineering*. *Information and Software Technology, 53*(6), 625–637.
20. Popay, J., Roberts, H., Sowden, A., Petticrew, M., Arai, L., Rodgers, M., Britten, N., Roen, K., & Duffy, S. (2006). *Guidance on the conduct of narrative synthesis in systematic reviews*. ESRC Methods Programme.
21. Teymourian, N., Izadkhah, H., & Isazadeh, A. (2022). *A fast clustering algorithm for modularization of large-scale software systems*. *IEEE Transactions on Software Engineering, 48*(4), 1451–1462.
22. Mao, Y., Li, B., Zhu, X., & Ma, J. (2023). *Software design based on k-means algorithm and artificial intelligence technology*. In *Proceedings of the 2023 International Conference on Mobile Internet, Cloud Computing and Information Security (MICCIS)* (pp. 200–203). IEEE.
23. Arora, D., Kumar, U., Jain, S., & Gupta, A. (2019). *UML modeling for preserving sensitive information based on k-means clustering approach*. In *Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI)* (pp. 110–117). IEEE.
24. Hammad, M., Hassan, A. E., & Hamdi, M. (2021). *Automatic class decomposition using clustering*. En *Proceedings of the 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)* (pp. 78–81). IEEE.
25. Parthasarathy, S., Bagavathilaksmi, R., Rajkumar, P., & Devi, S. (2023). *Exploring research opportunities to apply data mining techniques in software engineering lifecycle*. En *Proceedings of the 2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI)* (pp. 1–5). IEEE.
26. Khedmatkon, F., Hasheminejad, S. M. H., & Malak, J. S. (2024). *Automated software design using machine learning with natural language processing*. En *Proceedings of the 2024 14th International Conference on Computer and Knowledge Engineering (ICCKE)* (pp. 25–30). IEEE.
27. Cañizares, P. C., López-Morales, J. M., Pérez-Soler, S., Guerra, E., & de Lara, J. (2024). *Measuring and clustering heterogeneous chatbot designs*. *ACM Transactions on Software Engineering and Methodology, 33*(4), 1–43.
28. dos Santos Ferreira, M. V., Rios, R., & Rios, T. N. (2022). *SCI-FTS: Using soft clustering on intrinsic mode functions to model fuzzy time series*. *Software Impacts, 11*, 100230.
29. Esteves, G., Figueiredo, E., Veloso, A., Viggiato, M., & Ziviani, N. (2020). *Understanding machine learning software defect predictions*. *Automated Software Engineering, 27*(3), 369–392.
30. Johnson, F., Oluwatobi, O., Folorunso, O., Ojumu, A. V., & Quadri, A. (2023). *Optimized ensemble machine learning model for software bugs prediction*. *Innovations in Systems and Software Engineering, 19*(1), 91–101.
31. Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). *Applications of AI in classical software engineering*. *AI Perspectives, 2*(1), 1.
32. Sozer, H. (2019). *Evaluating the effectiveness of multi-level greedy modularity clustering for software architecture recovery*. En *Software Architecture – ECSA 2019* (pp. 71–87). Springer.
33. Rahman, M. S., Khomh, F., Hamidi, A., Cheng, J., Antoniol, G., & Washizaki, H. (2023). *Machine learning application development: Practitioners' insights*. *Software Quality Journal, 31*(4), 1065–1119.
34. Elyasi, M., Simitcioglu, M. E., Saydemir, A., Ekici, A., Ozener, O. O., & Sozer, H. (2023). *Genetic algorithms and heuristics hybridized for software architecture recovery*. *Automated Software Engineering, 30*(2), 19.