This study investigates the process that controls the IT project configuration.

The task addressed is early identification of configuration items (CIs) within an enterprise management information system (IS). Research in this area is aimed at solving the task of early identification of services when refactoring software systems. Up to now, the application of artificial intelligence methods to define CIs has not been studied in detail.

To solve the task of early identification of IS CIs, the density-based clustering algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was adapted. The adapted DBSCAN was used to early define CIs in the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department". As initial CIs, 10 functions and 12 entities in the database of the task were considered. The result is the sets of clusters that describe monolithic, modular, and service-oriented IS architectures.

A comparative analysis of the use of DBSCAN, Divisive Analysis, Agglomerative Nesting, Chameleon, and k-means methods and algorithms for solving the task of early identification was carried out. The criteria "Cumbersome solution" and "Identification of separated CIs" were used for comparison. The application of DBSCAN made it possible to form a solution from one (monolithic and modular architecture) or two (service-oriented architecture) clusters and to detect separated CIs. These values of the proposed criteria are the best for the selected group of clustering methods and algorithms.

Implementing the results makes it possible to automate a procedure for synthesizing the description of IS architecture. This automation would improve the quality of IS development by identifying a set of architectural entities of this IS for its design. This set is much smaller than the set of elementary IS functions

Keywords: information system, configuration element, dense-based clustering, DBSCAN algorithm, Chebyshev distance

UDC 004.65; 004.91
DOI: 10.15587/1729-4061.2026.352878

DEFINITION A CLUSTERIZATION METHOD TO SOLVE THE TASK OF EARLY IDENTIFICATION OF IT PRODUCT CONFIGURATION ELEMENTS

Adrian Kozhanov
PhD Student*
ORCID: https://orcid.org/0009-0002-3586-6886
Maksym Ievlanov
Corresponding author
Doctor of Technical Sciences, Professor*
E-mail: maksym.ievlanov@nure.ua
ORCID: https://orcid.org/0000-0002-6703-5166
*Department of Information Control Systems
Kharkiv National University of Radio Electronics
Nauky ave., 14, Kharkiv, Ukraine, 61166

Received 01.12.2025
Received in revised form 29.01.2026
Accepted date 16.02.2026
Published date 27.02.2026

How to Cite: *Kozhanov, A., Ievlanov, M. (2026). Definition a clusterization method to solve the task of early identification of IT product configuration elements. Eastern-European Journal of Enterprise Technologies, 1 (2 (139)), 77–90. https://doi.org/10.15587/1729-4061.2026.352878*

1. Introduction

A configuration management process is one of the most important ones in the life cycle of information systems (ISs) used to manage enterprises and organizations. The intermediate and final results of this process are recognized as necessary in many aspects of managing an organization and its IT services [1]. The purpose of this process is to systematically control configuration changes, maintain the integrity, and track the configuration throughout the product life cycle [2].

Regardless of the approaches to structuring and describing individual stages and tasks of the configuration management process, the configuration identification task is highlighted in it. This task is associated with determining those system elements that are objects or configuration items ("Configuration item", CI) [1–3]. Solutions to the configuration identification task gradually establish and maintain the current basis for monitoring and accounting for the state of the IT product and its CIs throughout all their life cycles [1].

The theoretical and applied foundations of IT product configuration management systems are generally recognized as formed [1, 4] and are proposed to be represented as a set of the following groups [4]:

– theoretical foundations (in particular, the three-component model);

– best practices (in particular, ISO 20000 and ITIL, as well as their analogs);

– innovations (in particular, DevOps integration, infrastructure as a code, containerization technologies).

Theoretical foundations form a fundamental understanding of configuration management. Best practices provide a fairly effective solution to the problem of identifying, controlling, and accounting for the state of CIs. The innovations considered improve traditional methods of IT product configuration management, enabling scalability, automation, and adaptability of applied solutions [4].

The main disadvantage of such a representation is its conceptual nature. Almost all theoretical foundations and best practices are conceptual provisions and models that very rarely have formal descriptions. In addition, these provisions and models are too generalized and require significant efforts to adapt to the specificity of specific ISs. Such adaptation is usually performed using innovative approaches and methods and leaves a number of issues unresolved. They include security problems, issues of compliance with IT product requirements and complexities that arise during the implementation of various cooperation options, the interaction of the IT product with the surrounding world [4]. Therefore, despite the unity of views on the definition of the configuration identification problem, its solution is still considered at the conceptual level [1].

The reason for the lack of formal solutions should be considered the established representation of CIs as a set of hardware, software, or both. Such a vague representation causes contradictory requirements for the configuration identification problem. On the one hand, such a task should cover every element of hardware and software. On the other hand, CIs should be marked at the highest possible level (it is recommended to use the smallest possible number of CIs in the final IT product) [1]. As a result, in the course of solving the configuration identification task, a complex problem arises of selecting the optimal number of CIs. The optimal number here should be understood as the number of CIs that will require minimal time and resources to perform further work in the IT product configuration management process.

This problem is further complicated by the need to trace the requirements for the IS to the descriptions of individual CIs that make up the description of the architecture of the IS being formed. On the one hand, there is a need to form such CIs (for example, functions) that would unambiguously correspond to the elementary well-formulated requirements for the IS. On the other hand, there is a need to combine individual functions as CIs of the designed IS into separate architectural entities that make up the general description of the architecture of the designed IS.

The use of these architectural entities, whose number is much smaller than the number of individual IS functions, significantly simplifies the transition from the processes of forming and analyzing IS requirements to the processes of designing the same IS. Solving this problem will significantly reduce the time and effort spent on compiling architecture descriptions, planning and managing an IT project to design an IS.

Therefore, conducting research in the field of devising new and improving existing methods for solving the task of identifying IS CIs remains one of the most relevant areas of computer science evolution.

2. Literature review and problem statement

As shown in [4], the existing foundations of IT product configuration management systems are too generalized and do not provide formal descriptions of the solution to the problem of identifying IS CIs. In addition, these provisions and models are too generalized and require significant efforts to adapt to the features of specific IS.

One of the approaches to overcoming this drawback is the proposal to abandon a single description of CI throughout the entire IS life cycle. Such a refusal makes it possible to consider the problem of identifying the IS configuration as a set of the following tasks:

– the task of early identification of CI (the result of the solution of which is a set of CIs as architectural entities of IS, identified based on the results of the analysis of well-formulated requirements for this system);

– the task of design identification of CI (the result of the solution of which is a set of CIs as elements of the IS design, identified based on the results of the analysis of the architectural entities of this system);

– the task of identifying CIs during implementation (development) (the result of the solution of which is a set of CIs as elements of software and/or hardware of the IS, identified based on the results of the analysis of CIs as elements of the design of this system).

This approach simplifies the work on forming CI descriptions at different stages of the IS life cycle. But it leaves the problem of identifying the optimal number of CIs unresolved.

In [4] it was also shown that the solution to the problem of early identification of SI largely depends on the solution of the issue of compliance with the requirements for the IT product. In the general case, this dependence can be defined by the following statement: the task of early identification of SI as architectural entities of the IS must be solved for descriptions of well-formulated requirements for this IS. In this case, the term “architectural entity” should be understood as groups of functional requirements (functional subsystems and tasks, functional modules, resource management circuits, IT services, etc.), which correlate the functions of the IS with the business needs of the users of this IS.

The proof of this statement is the results of the development of a modern concept of research and formal description of service-oriented systems [5]. According to this concept, the solution to most problems of selecting IT services requires the definition of a set of functionally equivalent IT services before starting to solve such problems [5]. If we consider IT services as CIs, then this statement can be formulated as follows: the task of identifying IT services as CIs should be solved first for abstract descriptions of these CIs. Such abstract descriptions should not depend on the specificity of the implementation of these services and non-functional requirements imposed on services. However, specific formal models and methods for creating and processing such abstract descriptions are not given in [5]. In addition, the issue of compliance with the elementary requirements that these abstract descriptions of CIs should meet in [5] remained unresolved.

A similar situation arises during configuration management of large systems (so-called System of Systems), an example of which is an IS. In particular, the possibility of eliminating most cases of negative impact of local solutions on the overall design and quality of a large software and hardware system due to early division of the system into separate elements is described in [6]. However, the issue of formal description of such separate elements and separation of these elements from general system descriptions or descriptions of individual requirements for IS is not solved in [6].

The general solution to the problem of early identification of IS CIs can be found based on the following approaches:

a) functional decomposition of the general description of the IS architecture into separate architectural entities and system elements that make up these entities;

b) synthesis of descriptions of architectural entities and a general description of the architecture from descriptions of individual system elements or requirements for these elements.

An example of using approach a) is described in [7]. In this case, the task of configuration identification is solved in two stages:

– at the first stage, the functional decomposition of the architecture description into separate elements is performed, taking into account the necessary evolutionary changes;

– at the second stage, those decomposition options are selected that satisfy the constraints on the cost of the necessary changes.

However, the approach considered in [7] is focused mainly on tracking changes made to the architecture description. At the same time, the following issues remain unconsidered:

– initial identification of individual functions as CIs as a result of an analysis of points of view on the architecture of this system;

– formation and development of IS architectural entities as sets of CIs that were selected during the solution of the problem.

The use of this approach to solve the tasks of design identification of CIs and identification of CIs during implementation is considered in [8, 9]. Thus, in work [8], the solution to the problem of design identification of CIs was considered as the allocation of individual software artifacts that ensure reliable operation of the designed software system. As the basis for such allocation, it was proposed to use the description of the meta-architecture of such a system. It was emphasized that the use of abstractions to describe the architecture of a software system simplifies the solution of such a problem. However, the allocation of descriptions of individual architectural entities from the description of the architecture of the CI system as groups of system functions is not considered in work [8].

In work [9], the solution to the problem of identification of CIs during implementation was considered as the task of dividing the description of the architecture of a software system into microservices. To describe the general system architecture, it was proposed to use the Silvera object-oriented language. However, the application of this solution is limited by the following features [9]:

– orientation exclusively on decentralized development of microservices;

– the lack of a description of business logic in the description of the system architecture.

The main limitation of using approach a) is the lack of any universal methods and ways to solve the problems of CI identification. A certain confirmation of this statement is study [7]. It shows that most of the existing approaches to the decomposition of a monolithic architecture into a set of individual microservices can be applied only under certain conditions. However, these conditions are not specified in [7].

The study on the tasks of early division of the system into separate elements showed that these problems arose, mainly, as a result of misinterpretation of requirements or bias of personal experience [10]. Therefore, it is advisable to use artificial intelligence methods for identification of CIs in large systems, in which the subjective influence of an individual analyst is minimized.

Unlike approach a), modern research into methods and tools based on approach b), from the very beginning is based on the use of artificial intelligence methods and tools. Thus, in work [10] examples of application for modeling and tuning of system functions of such artificial intelligence tools as recommender systems and machine learning methods are given. Their application is due in work [10] to the fact that a complete enumeration of all possible configurations is impossible and can cause serious problems with performance. But although the considered examples take into account the factor of variability of descriptions of individual functions, the issues of coherence of these functions and formation of architectural entities from them remain unresolved.

In [11], it is proposed to use a sampler based on deep reinforcement learning to solve the task of system configuration selection. The result of using this sampler is an identified minimal subset of configurations that covers all variants of interaction of functions while minimizing redundancy. But such a solution can be appropriate only in the case of choosing a weakly coupled service-oriented architecture as the basic concept of describing the synthesized IS architecture. For IS intended for managing stable processes of enterprises and organizations, such a solution is excessively complex.

A frequently used artificial intelligence tool used by researchers to solve the tasks of identification of IS within the framework of approach b) are clustering methods. As examples, we can cite the application of clustering algorithms to solve the following tasks:

– identification of microservices during refactoring of the source code of a monolithic software application [12];

– refactoring of complex multi-level monolithic software systems during their decomposition into microservices [13];

– early identification of backlogs of execution teams as CI systems [14].

However, the studies considered do not take into account the connections between individual IS functions, which determine the grouping of functions into IS architectural entities. In general, the question of factors that determine the purpose function, constraints and the method of calculating the distance between individual IS functions remains open.

The use of clustering methods to solve the problem of early identification also requires formalization of the requirements that are imposed on IS. In [16], a systematic review of current scientific research on tools, methods, techniques, and processes for extracting requirements from environments based on the Model-Based System Engineering methodology and generating models based on requirements formulated in natural language was conducted. The results of the review revealed several key conclusions that have important implications for future research. Of these conclusions, the following should be especially noted [15]:

– generation of models based on descriptions made in natural language remains the dominant way to formally describe IS requirements;

– transformation of textual descriptions of requirements into formal descriptions based on language models requires more data than current methods for generating such textual descriptions can provide.

But in [15] the issue of further use of requirements models in further processes of the system life cycle is practically not covered. In particular, the issue of using requirements models as a basis for creating descriptions of CI IS in [15] remained unconsidered.

In [16], a review of modern research into determining the features in solving the problem of integral synthesis of the execution environment composition in dynamic System of Systems was given. The results of this review allowed the authors of [16] to draw the following conclusions:

– the tasks of solving this problem are divided into four categories: modeling and analysis, stable operations, system orchestration, heterogeneity of component systems;

– the solutions to this problem cover seven areas: joint modeling and digital twins, semantic ontologies, integration frameworks, adaptive architectures, middleware, formal methods, stability based on artificial intelligence;

– the tools for solving this problem are dominated by service-oriented frameworks for composition and integration, while modeling platforms support evaluation of results;

– the key problems that limit the solution of this task are the complex interaction between existing tools, limited data, and work flows that arise between individual tools, and the lack of standardized benchmarks for evaluating the result of solving the problem.

But in work [16] practically no frameworks are highlighted, which are recommended to be used to solve the task of ho-

listic synthesis of a system-wide description based on descriptions of individual CIs at the early stages of the IS life cycle.

Our review of the literature [4–16] leads to the following conclusions:

– it is desirable to solve the task of early identification of IS CIs using approach b) because it is impossible to compile a general description of the IS architecture for further decomposition in many cases;

– to solve the specified task, it is best to use formal methods of artificial intelligence and, in particular, clustering methods;

– as derived data for solving the specified task, models of requirements that are put forward for IS should be used, in particular – formal models of functional requirements for IS based on their ontological descriptions.

These conclusions define a problem that requires a scientific and applied solution, such as determining the possibility of solving the task of early identification of IS CIs based on approach b) using such clustering methods and algorithms that do not depend strongly on assumptions about the shape of clusters.

Solving this task could allow us to highlight the advantages and disadvantages of using such methods and algorithms in comparison with other common clustering methods.

3. The aim and objectives of the study

The purpose of our study is to justify the choice of a clustering method, which, according to predefined criteria, can be considered the best for solving the task of early identification of IS CIs for the management of enterprises and organizations. This would allow us to choose an algorithm for the automated solution to the task of early identification of IS CIs, the application of which could provide a solution with a minimum number of architectural entities without losing separate system functions.

To achieve the goal, it is necessary to solve the following tasks:

– to adapt the DBSCAN algorithm to the specificity of solving the task of early identification of IS CIs;

– to check the possibility of solving the task of early identification of IS CIs using the DBSCAN algorithm;

– to conduct a comparative analysis of the results with the results from solving the task of early identification of IS CIs using other clustering methods and algorithms.

4. The study materials and methods

The object of our study is the process of managing the configuration of an IT project. The subject of the study is methods for solving the task of early identification of IS CIs.

The principal hypothesis assumes the possibility of improving the solution to the task of early identification of IS CIs using the DBSCAN density clustering algorithm in comparison with solutions to the same problem obtained using other methods and clustering algorithms.

The choice of a group of density clustering methods is due to the following considerations [17]:

– these methods, unlike many other clustering methods (partitioning methods, hierarchical methods, etc.), can find clusters of arbitrary shape;

– these methods define clusters as dense areas of objects of the studied sample, separated by areas with low density (which best corresponds to the definition of the architectural entity);

– these methods can filter out anomalies (that is, descriptions of individual IS functions, the inclusion of which in the general description of the architecture of this system is inappropriate for reasons previously determined by the analyst).

In [17], it is recognized that there are better clustering methods and algorithms than DBSCAN that make it possible to detect high-quality clusters of arbitrary shape (in particular, the Chameleon agglomerative clustering method). However, the DBSCAN algorithm is simpler to implement and requires less computational effort and time to solve the clustering problem [18]. In addition, one of the advantages of the DBSCAN algorithm is the ability to process data of various natures. As shown in [18] with references to relevant studies: "…DBSCAN was never limited to using Euclidean distance or points in $\mathbb{R}^d$, but was always intended for use with geographic data, polygons, and other types of data."

A description of the DBSCAN density clustering algorithm using pseudocode is given in Fig. 1 [17].

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ϵ-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ϵ-neighborhood of p;
(8)          for each point p′ in N
(9)              if p′ is unvisited
(10)                 mark p′ as visited;
(11)                 if the ϵ-neighborhood of p′ has at least MinPts points,
                     add those points to N;
(12)             if p′ is not yet a member of any cluster, add p′ to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

Fig. 1. Description of the Density-Based Spatial Clustering of Applications with Noise algorithm using pseudocode [17]

In the form shown in Fig. 1, the DBSCAN algorithm does not take into account all the features of solving the task of early identification of IS CIs. Therefore, it is necessary to adapt this algorithm to the features of solving this problem by specifying the definitions of its individual parameters.

In the study, it was decided to consider the description of the IS architecture as the result of the problem of synthesizing a variant of the description of the rational IS architecture. This problem in [19] was proposed to be described by a game-theoretic model of the following form

$$\Gamma_{IS} = \left\langle \{\Pr, U\}, \{X^{\{\Pr,U\}}\}, \{f^{\{\Pr,U\}}\}, \right. \tag{1}$$

where Γ_{IS} – designation of the game of the Supplier and Consumer of IT services (IS functions); $\{\Pr, U\}$ is the set of

players participating in game Γ_{IS} (Supplier and Consumer, respectively); $\left\{X^{\{\Pr,U\}}\right\}$ – set of strategies for game Γ_{IS}, which is formed on the basis of the set of variants of descriptions of the architecture of the designed IS IT_{acm}; $\left\{f^{\{\Pr,U\}}\right\}$ is the set of payoff functions of game Γ_{IS}, which consists of expressions

$$F_{\Pr} = \sum_{i=c+1}^{e} 1 - \left| K_i^{f_{IS}} / K_i^{f_{\Pr}} \right| \Big/ \left| K_i^{f_{IS}} \right| \to \max$$

and

$$F_U = \sum_{i=c+1}^{e} 1 - \left| K_i^{f_{IS}} / K_i^{f_U} \right| \Big/ \left| K_i^{f_{IS}} \right| \to \max.$$

Game (1) is a finite zero-sum non-coalition game between two players (the Supplier and the Consumer of IT services). To find the outcome of game (1), in [19] it is proposed to use the Nash equilibrium search method for a bimatrix game in pure strategies. The result of using this method is a description of the architecture of the designed IS as a set of individual functions of this IS.

A detailed description of game (A), formal models of descriptions of requirements for individual IS functions, and the Nash equilibrium search method for this game are given in [19]. It should be noted that during the research in [19], it was proposed to limit to the following basic types of IS architectures based on the results of research into the laws of composition of the functional structure of IS:

– functional (subsystems);

– modular;

– service (service-oriented).

Since the functional architecture of IS is considered outdated and is practically not used when designing modern IS, our study proposes to use a monolithic architecture instead. A feature of this architecture is the inclusion of all IS functions into one CI. The use of this architecture is considered appropriate, for example, when stabilizing orchestration scenarios and choreography of individual services within the framework of an automated business process.

One of the shortcomings of the solutions proposed in [19] is the formation of the description of the IS architecture from the descriptions of individual functions of this system. This means that a description of a separate function (or requirements for this function) is selected as a separate IS CI. As a result, the number of individual CIs in large and medium-sized ISs for managing enterprises and organizations can reach several hundred. This significantly complicates the further distribution of the description of the IS architecture between separate teams of IT project developers for the development of this system. Therefore, it is necessary to reduce the number of CIs in the description of the IS architecture. It is for this reduction that clustering methods and algorithms are used to solve the task of early identification. In the general case, this reduction can be described as objective function J, which takes the following form [18]

$$J\left(M,d,U,C\right) = \sum_{i=1}^{c}\sum_{j=1}^{k} u_{ij} d\left(m_j, c^{(i)}\right) \to \min, \qquad (2)$$

where M is the basic set of derived data; m_j is the j-th point (vector) of derived data; k is the number of points (vectors) of derived data; d is the method for determining the distance between points (vectors) of derived data; U is the matrix of partitioning points (vectors) of derived data between clusters $c^{(i)}$; u_{ij} is the element of the partitioning matrix that establishes the belonging of point (vector) m_j to cluster $c^{(i)}$; C is the set of clusters that are the results of solving the task of early identification; $c^{(i)}$ is the i-th cluster, which is one of the results of solving the task of early identification of CIs of IS and describes the identified architectural entity of the IS as a subset of individual functions; c is the number of formed clusters.

The set of constraints for objective function (2) takes the following form [18]

$$u_{ij} = \{0,1\},\ \sum_{i=1}^{c} u_{ij} = 1,\ 0 < \sum_{j=1}^{k} u_{ij} \le k. \qquad (3)$$

These restrictions establish that:

– each derivative point (vector) can belong only to one of the clusters and not to other clusters;

– each cluster contains at least one derivative point (vector), but not less than the total number of derivative points (vectors) k.

To verify the possibility of solving the task of early identification of IS CIs using the DBSCAN algorithm, it was proposed to use the descriptions of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department". This task was implemented as a separate IT program to develop the capabilities of the information and analytical system "University" at the Kharkiv National University of Radio Electronics (Ukraine). Detailed descriptions of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department" are given in [14].

As a description of individual functions, it is proposed to consider each individual work of the data flow diagram with input and output data flows that are associated with our work [14, 20, 21]. The resulting descriptions of the functions of the task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department" are given in Table 1 [21]. The letters "CI" and numbers in Table 1 indicate the identifiers of individual works that describe the corresponding functions of the task as separate initial *CIs*. Elements with the value "1" indicate the facts of using the entity with the identifier given in the column of Table 1 to describe the work, input or output data flow with the identifier given in the row of Table 1. Elements with the value "0" indicate the opposite facts.

The names of the data flow diagram works specified in Table 1 are given in Table 2 [14, 20, 21].

The names of the entities, whose IDs are given in Table 1, are indicated in Table 3 [14, 20, 21].

The solutions obtained using the DBSCAN algorithm were proposed to be compared with the solutions to the same problem obtained on the same data set using the methods and algorithms of hierarchical clustering and the k-means method.

All calculations when checking the possibility of solving the task of early identification of IS CIs using the DBSCAN algorithm were performed in the MS Excel 2016 environment.

Table 1

Descriptions of configuration items for the task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department"

Function description *CI*												
CI	Entity ID											
	1	2	3	4	5	6	7	8	9	10	11	12
*CI*1	1	1	1	1	1	1	0	0	0	0	0	0
*CI*2	1	1	1	1	1	1	1	1	1	0	0	0
*CI*3	1	1	1	1	1	1	1	1	1	0	0	0
*CI*4	1	1	1	1	1	1	1	1	1	0	0	0
*CI*5	0	1	0	1	0	1	1	0	0	1	1	0
*CI*6	0	0	0	0	0	0	0	1	1	0	0	0
*CI*7	0	0	0	0	0	0	0	0	0	0	0	1
*CI*8	1	1	0	1	1	1	1	1	1	0	0	1
*CI*9	1	1	0	1	1	1	1	1	1	0	0	0
*CI*10	1	1	0	1	1	1	1	1	1	1	1	0
Description of input data streams *CI*												
CI	Entity ID											
	1	2	3	4	5	6	7	8	9	10	11	12
*CI*1	1	1	1	0	0	0	0	0	0	0	0	0
*CI*2	1	1	1	1	1	1	1	1	1	0	0	0
*CI*3	1	1	1	1	1	1	1	1	1	0	0	0
*CI*4	1	1	1	1	1	1	1	1	1	0	0	0
*CI*5	0	1	0	1	0	1	1	0	0	1	1	0
*CI*6	0	0	0	0	0	0	0	1	1	0	0	0
*CI*7	0	0	0	0	0	0	0	0	0	0	0	1
*CI*8	1	1	0	1	0	1	1	1	1	0	0	0
*CI*9	1	1	0	1	1	1	1	1	1	0	0	0
*CI*10	1	1	0	1	1	1	1	1	1	1	1	0
Description of the output data streams *CI*												
CI	Entity ID											
	1	2	3	4	5	6	7	8	9	10	11	12
*CI*1	1	1	0	1	1	1	0	0	0	0	0	0
*CI*2	0	1	0	1	0	1	1	1	1	0	0	0
*CI*3	0	1	0	1	0	1	1	1	1	0	0	0
*CI*4	0	1	0	1	0	1	1	1	1	0	0	0
*CI*5	0	1	0	1	0	1	1	0	0	1	1	0
*CI*6	0	0	0	0	0	0	0	1	1	0	0	0
*CI*7	0	0	0	0	0	0	0	0	0	0	0	1
*CI*8	1	1	0	1	1	1	1	0	0	0	0	1
*CI*9	1	1	0	1	1	1	1	1	0	0	0	0
*CI*10	1	1	0	1	1	1	1	1	1	1	1	0

Table 2

Name of the work of the data flow diagram of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department"

Work	
The designation in Table 1	Name
*CI*1	Conversion of the section «Academic work»
*CI*2	Formation of the section «Scientific work»
*CI*3	Formation of the section «Methodological work»
*CI*4	Formation of the section «Organizational work»
*CI*5	Formation of the list of positions and long-term assignments
*CI*6	Formation of a list of recommended works
*CI*7	Formation and maintenance of regulatory information on Key Performance Indicators (KPIs)
*CI*8	Formation of KPI of the lecturer and part of the department's KPI
*CI*9	Formation of a summary table for the academic year
*CI*10	Formation of the source document «Individual Plan»

Table 3

Name of entities of the ER-diagram of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department"

ID	Name
1	Academic_load
2	Academic
3	Department
4	Individual_plan
5	Academic_section
6	Academic_year
7	Section
8	Recommended_works
9	Type_of_work
10	Section_Pos_Assign_Dept
11	PositionsAssignments
12	KPI

5. Results of solving the task of early identification of configuration items in the information system

5. 1. Results of adapting the selected algorithm to the features of solving the task of early identification

To adapt the DBSCAN algorithm to the features of solving the task of early identification of IS CIs, it was necessary to determine:

– features of the representation of the dataset D, the objects of which should be studied using this algorithm;

– features of the representation of the radius parameter ε, which should determine the boundary of the region of existence of the nearest neighbors of the studied objects (the so-called proximity measure);

– features of the choice of values of the *MinPts* parameter, which should determine the density threshold of the nearest neighbors of the studied object.

To solve the task of early identification of IS CIs, the assumption was adopted about recognizing as IS individual CI functions that are indivisible at a given level of representation. Each such function is defined as the implementation of an elementary well-formulated requirement put forward for IS. Accepting this assumption makes it possible to consider the basic description of IS architecture as a set of individual IT services that implement the corresponding IS requirements to meet the business needs of employees of the IS consumer organization. Data flow diagrams were used to model these IS requirements and, accordingly, descriptions of IS functions. Therefore, in our study, the dataset D is represented as a set of objects CI_j, which are descriptions of IS CIs. Each CI_j description is formally represented as a set [14]

$$CI_j = \{W_j, X_j, Y_j\}, \quad (4)$$

where W_j is the description of the j-th function of IS as a set of entities or classes; X_j is the description of all input flows of the j-th function of the IS as sets of entities or classes that form these flows; Y_j is the description of all output flows of the j-th function of IS as sets of entities or classes that form these flows.

Such a description of IS CIs allows us to use the modified Chebyshev distance to determine distances between functions. This distance is calculated from the following formula [14]

$$d_{\infty}\left(i_p, i_q\right) = \max_{1 \le l \le m} \sum_{k=1}^{n} i_{plk} \oplus i_{qlk}, \quad (5)$$

where i_{plk} is the value of variable (0 or 1) that determines the absence or presence of a description of the k-th entity or class in the description of the function, input or output stream i_{pl} CI i_p; i_{qlk} is the value of variable (0 or 1) that determines the absence or presence of a description of the k-th entity or class in the description of the function, input or output stream i_{ql} CI i_q; n is the maximum value of the entity or class identifier participating in the descriptions of the compared functions, input or output streams CI i_p and i_q; $\oplus$ is the operation "sum modulo 2".

A detailed description of the features of the formation of the modified Chebyshev distance is given in [14].

In the general case, the value of the radius parameter ε is set based on the values of the elements of the D_{CI} distance matrix between the objects of dataset D. However, it should be taken into account that these distances for each specific IS and different subject areas may differ significantly from each other. Therefore, it was proposed to perform minimax normalization of values for the elements of matrix D_{CI} according to the following formula

$$d'_{CI_{ij}} = \left(d_{CI_{ij}} - \min\left|D_{CI}\right|\right) \Big/ \left(\max\left|D_{CI}\right| - \min\left|D_{CI}\right|\right), \quad (6)$$

where $d'_{CI_{ij}}$ – normalized element of distance matrix D_{CI}; $d_{CI_{ij}}$ – element of distance matrix D_{CI}; $\min\left|D_{CI}\right|$ – element of distance matrix D_{CI} with the minimum value; $\max\left|D_{CI}\right|$ – element of distance matrix D_{CI} with the maximum value.

Such normalization made it possible to represent the radius parameter ε of the DBSCAN algorithm as an indicator that characterizes the relative magnitude of the difference between the descriptions of IS CIs data structures. The value of this indicator for each pair (CI_i, CI_j) was proposed to be calculated from the following formula

$$\varepsilon_{ij} = d'_{CI_{ij}} * 100\%. \quad (7)$$

Based on (7), the value of indicator ε was proposed to be determined in the range from 0% to 100%. Thus, the choice of value for indicator ε allowed the analyst to set the desired value of the difference between the descriptions of CI data structures that can be included in the same architectural element. This means that as a result of using the DBSCAN algorithm to solve the task of early identification of CIs in IS, descriptions of architectural entities (functional subsystems, functional modules, IT services, etc.) will be formed from descriptions of functions that process similar data structures.

When setting possible values for parameter *MinPts*, the features of existing descriptions of IS architectures at the level of individual functions were taken into account. These features are as follows. First, for descriptions of service-oriented architectures, the possibility of independent operation of an IT service that implements a separate IS function should be taken into account. Based on this, it is advisable to include this function in the description of the architectural entity of the IS if there is at least one other function that will interact with this function. A formal description of this condition for integrating a new service into the IS is given in [22].

Secondly, for descriptions of modular and monolithic IS architectures, one should take into account the need to operate each specific IS function only in conjunction with other functions of the module or subsystem. Therefore, it is advisable to include a description of a specific function in the description of the architectural entity of the IS only if there are at least two other functions of this system that interact with this function (transmit or receive data).

Based on the identified features of existing descriptions of IS architectures, it was proposed to choose the following values of parameter *MinPts* when using the DBSCAN algorithm to solve the task of early identification of IS CIs:

– $MinPts = 1$ in the case of making a decision on the feasibility of synthesizing a description of a service-oriented IS architecture;

– $MinPts = 2$ in the case of making a decision on the feasibility of synthesizing a description of a modular or monolithic IS architecture.

5. 2. Checking the possibility of solving the task of early identification of configuration items using the selected algorithm

To verify the possibility of solving the task of early identification of IS CIs using the DBSCAN algorithm, it was proposed to use dataset D, given in Table 1 [21]. In the process of forming the input data for the DBSCAN algorithm, a matrix of distances between the objects of this dataset DCI was formed using formula (5). The values of elements in matrix D_{CI} are given in Table 4.

Table 4

Matrix of distances D_{CI} between configuration items in the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department"

CI	CI1	CI2	CI3	CI4	CI5	CI6	CI7	CI8	CI9	CI10
CI1	0	6	6	6	7	8	7	6	7	9
CI2	6	0	0	0	7	7	10	4	3	4
CI3	6	0	0	0	7	7	10	4	3	4
CI4	6	0	0	0	7	7	10	4	3	4
CI5	7	7	7	7	0	8	7	7	6	4
CI6	8	7	7	7	8	0	3	7	7	9
CI7	7	10	10	10	7	3	0	8	9	11
CI8	6	4	4	4	7	7	8	0	2	4
CI9	7	3	3	3	6	7	9	2	0	3
CI10	9	4	4	4	4	9	11	4	3	0

The results of normalization of elements in distance matrix D_{CI} according to formula (6) are given in Table 5.

Table 5

Normalized distance matrix D'_{CI}

CI	CI1	CI2	CI3	CI4	CI5	CI6	CI7	CI8	CI9	CI10
CI1	0	0.545	0.545	0.545	0.636	0.726	0.636	0.545	0.636	0.818
CI2	0.545	0	0	0	0.636	0.636	0.909	0.363	0.272	0.363
CI3	0.545	0	0	0	0.636	0.636	0.909	0.363	0.272	0.363
CI4	0.545	0	0	0	0.636	0.636	0.909	0.363	0.272	0.363
CI5	0.636	0.636	0.636	0.636	0	0.726	0.636	0.636	0.545	0.363
CI6	0.726	0.636	0.636	0.636	0.726	0	0.272	0.636	0.636	0.818
CI7	0.636	0.909	0.909	0.909	0.636	0.272	0	0.726	0.818	1
CI8	0.545	0.363	0.363	0.363	0.636	0.636	0.726	0	0.181	0.363
CI9	0.636	0.272	0.272	0.272	0.545	0.636	0.818	0.181	0	0.272
CI10	0.818	0.363	0.363	0.363	0.363	0.818	1	0.363	0.272	0

First, the solution to the task of early identification of IS CIs for the situation $MinPts = 1$ was considered. This situation arises as a result of choosing the description of the service-oriented architecture as the basic variant of the description of IS architecture.

For the first execution of the DBSCAN algorithm for the case $MinPts = 1$, $\varepsilon_{ij} = 20\%$ was set (the difference between the descriptions of IS CI data structures according to (7) does not exceed 20%).

At the beginning of the execution of the DBSCAN algorithm, all CIs were marked as not attended. Then, a random sequence of CIs was generated using an online random number generator (http://castlots.org/generator-sluchajnyh-chisel/). This sequence takes the following form: CI7, CI4, CI5, CI6, CI1, CI8, CI3, CI10, CI2, CI9.

CI7 was considered first. It was marked as attended. No element was found in the 0.2-vicinity of CI7. Therefore, the element CI7 was marked as noise.

Next, the element CI4 was considered. It was marked as attended. In the 0.2-vicinity of CI4, the elements CI2 and CI3 were found. Therefore, a new cluster C1 = {CI4} was built. Since CI2 had not yet been attended, it was marked as attended. In the 0.2-vicinity of CI2, the elements CI3 and CI4 were found. Therefore, it was decided to include CI2 in the cluster C1 = {CI4, CI2}. Since CI3 had not yet been attended, it was marked as attended. In the 0.2-vicinity of CI3, elements CI2 and CI4 were found. Therefore, it was decided to include CI3 in the cluster C1 = {CI4, CI2. CI3}.

Next, element CI5 was considered. It was marked as attended. No element was found in the 0.2-vicinity of CI5. Therefore, element CI5 was marked as noise.

Next, element CI6 was considered. It was marked as attended. No element was found in the 0.2-vicinity of CI6. Therefore, element CI6 was marked as noise.

Next, element CI1 was considered. It was marked as attended. No element was found in the 0.2-vicinity of CI1. Therefore, element CI1 was marked as noise.

Next, element CI8 was considered. It was marked as attended. Element CI9 was found in the 0.2-vicinity of CI1. Therefore, a new cluster C2 = {CI8} was built. Since CI9 had not yet been attended, it was marked as attended. In the 0.2-vicinity of CI9, element CI8 was found. Therefore, it was decided to include CI9 in cluster C2 = {CI8, CI9}.

Next, in this sequence, it was necessary to consider element CI3. But it was already marked as attended. Therefore, element CI3 was not considered again.

Next, element CI10 was considered. It was marked as attended. In the 0.2-vicinity of CI1, no element was found. Therefore, element CI1 was marked as noise.

Next, in this sequence, it was necessary to consider element CI2. But it was already marked as attended. Therefore, element CI2 was not considered again.

Next, in this sequence, it was necessary to consider element CI9. But it was already marked as attended. Therefore, element CI9 was not considered again.

Thus, as a result of the first execution of the DBSCAN algorithm for the values of parameters $MinPts$=1 and $\varepsilon_{ij} = 20\%$, two architectural entities were formed, which are clusters C1={CI4, CI2, CI3} and C2={CI8, CI9}. All other CIs were marked as noise.

Since a significant number of CIs as a result of the first use of the DBSCAN algorithm were identified as noise, it was proposed to change the value of parameter $\varepsilon_{ij} = 50\%$ to run this algorithm for the same random sequence of CIs a second time.

First, CI7 was considered. It was marked as attended. In the 0.5-vicinity of CI7, element CI6 was found. Therefore, a new cluster C1 = {CI7} was built. Since CI6 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI6, element CI7 was found. Therefore, it was decided to include CI6 in cluster C1 = {CI7, CI6}.

Next, element CI4 was considered. It was marked as attended. In the 0.5-vicinity of CI4, elements CI2, CI3, CI8, CI9, and CI10 were found. Therefore, a new cluster C2 = {CI4} was built. Since CI2 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI2, elements CI3, CI4, CI8, CI9, and CI10 were found. Therefore, it was decided to include CI2 in cluster C2 = {CI4, CI2}. Since CI3 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI3, elements CI2, CI4, CI8, CI9, and CI10 were found. Therefore, it was decided to include CI3 in cluster C2 = {CI4, CI2, CI3}. Since CI8 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI8, elements CI2, CI3, CI4, CI9, and CI10 were found. Therefore, it was decided to include CI8 in cluster C2 = {CI4, CI2, CI3, CI8}. Since CI9 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI9, elements CI2, CI3, CI4, CI8, and CI10 were found. Therefore, it was decided to include CI9 in cluster C2 = {CI4, CI2, CI3, CI8, CI9}. Since CI10 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI10, elements CI2, CI3, CI4, CI5, CI8, and CI9 were found. Therefore, it was decided to include CI10 in cluster C2 = {CI4, CI2, CI3, CI8, CI9, CI10}. Since CI5 had not yet been attended, it was marked as attended. In the 0.5-vicinity of CI5, element CI10 was found. Therefore, it was decided to include CI5 in cluster C2 = {CI4, CI2, CI3. CI8, CI9, CI10, CI5}.

Next in this sequence, it was necessary to consider element CI5. But it was already marked as attended. Therefore, element CI5 was not considered again.

Next, in this sequence, it was necessary to consider element CI6. But it was already marked as attended. Therefore, element CI6 was not considered again.

Next, in this sequence, it was necessary to consider element CI1. It was marked as attended. No element was found in the 0.5-vicinity of CI1. Therefore, element CI1 was marked as noise.

Next, in this sequence, it was necessary to consider element CI8. But it was already marked as attended. Therefore, element CI8 was not considered again.

Next, in this sequence, it was necessary to consider element CI3. But it was already marked as attended. Therefore, element CI3 was not considered again.

Next, in this sequence, it was necessary to consider element CI10. But it was already marked as attended. Therefore, element CI10 was not reviewed again.

Next, in this sequence, it was necessary to review element CI2. But it was already marked as attended. Therefore, element CI2 was not reviewed again.

Next, in this sequence, it was necessary to review element CI9. But it was already marked as attended. Therefore, element CI9 was not reviewed again.

Thus, as a result of the second execution of the DBSCAN algorithm for the values of parameters $MinPts = 1$ and $\varepsilon_{ij} = 50\%$, two architectural entities were formed, which are clusters C1 = {CI7, CI6} and C2 = {CI4, CI2, CI3, CI8, CI9, CI10, CI5}. CI1 was marked as noise.

Since one CI as a result of the first use of the DBSCAN algorithm was identified as noise, it was proposed to change the value of parameter $\varepsilon_{ij} = 60\%$ and run this algorithm for

the same random sequence of CI a third time. The choice of this value of ε_{ij} is due to the fact that distance $d_\infty(CI_p, CI_q)$ is the median for distance matrix D_{CI}.

The first to consider was CI7. It was marked as attended. In the 0.6-vicinity of CI7, element CI6 was found. Therefore, a new cluster C1 = {CI7} was built. Since CI6 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI6, element CI7 was found. Therefore, it was decided to include CI6 in the cluster C1 = {CI7, CI6}.

Next, element CI4 was considered. It was marked as attended. In the 0.6-vicinity of CI4, elements CI1, CI2, CI3, CI8, CI9, and CI10 were found. Therefore, a new cluster C2 = {CI4} was built. Since CI1 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI1, elements CI2, CI3, CI4, and CI8 were found. Therefore, it was decided to include CI1 in the cluster C2 = {CI4, CI1}. Since CI2 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI2, elements CI1, CI3, CI4, CI8, CI9, and CI10 were found. Therefore, it was decided to include CI2 in the cluster C2 = {CI4, CI1, CI2}. Since CI3 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI3, elements CI1, CI2, CI4, CI8, CI9, and CI10 were found. Therefore, it was decided to include CI3 in the cluster C2 = {CI4, CI1, CI2, CI3}. Since CI8 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI8, elements CI1, CI2, CI3, CI4, CI9, and CI10 were found. Therefore, it was decided to include CI8 in cluster C2 = {CI4, CI1, CI2, CI3. CI8}. Since CI9 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI9, elements CI2, CI3, CI4, CI5, CI8 and CI10 were found. Therefore, it was decided to include CI9 in cluster C2 = {CI4, CI1, CI2, CI3. CI8, CI9}. Since CI10 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI10, elements CI2, CI3, CI4, CI5, CI8 and CI9 were found. Therefore, it was decided to include CI10 in cluster C2 = {CI4, CI1, CI2, CI3. CI8, CI9, CI10}. Since CI5 had not yet been attended, it was marked as attended. In the 0.6-vicinity of CI5, elements CI9 and CI10 were found. Therefore, it was decided to include CI5 in the cluster C2 = {CI4, CI1, CI2, CI3. CI8, CI9, CI10,CI5}.

Next, in this sequence, it was necessary to consider element CI5. But it was already marked as attended. Therefore, element CI5 was not considered again.

Next, in this sequence, it was necessary to consider element CI6. But it was already marked as attended. Therefore, element CI6 was not considered again.

Next, in this sequence, it was necessary to consider element CI1. But it was already marked as attended. Therefore, element CI1 was not considered again.

Next, in this sequence, it was necessary to consider element CI8. But it was already marked as attended. Therefore, element CI8 was not considered again.

Next, in this sequence, it was necessary to consider element CI3. But it was already marked as attended. Therefore, element CI3 was not considered again.

Next, in this sequence, it was necessary to consider element CI10. But it was already marked as attended. Therefore, element CI10 was not reattended.

Next, in this sequence, it was necessary to review element CI2. But it was already marked as attended. Therefore, element CI2 was not reattended.

Next, in this sequence, it was necessary to review element CI9. But it was already marked as attended. Therefore, element CI9 was not reattended.

Thus, as a result of the third execution of the DBSCAN algorithm for the values of parameters *MinPts* = 1 and ε_{ij} = 60%, two architectural entities were formed, which are clusters C1 = {CI7, CI6} and C2={CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}. No elements that should be marked as noise were detected during the third execution of the DBSCAN algorithm.

An attempt to execute the DBSCAN algorithm for the values of parameters *MinPts* = 1 and ε_{ij} = 70% led to the selection of one architectural entity C1, which included all the CIs (monolithic architecture).

The set of results from solving the task of early identification of IS CIs using the DBSCAN algorithm for the case *MinPts* = 1 is given in Table 6.

Table 6

Results of solving the task of early identification of configuration items during the synthesis of a description of a service-oriented architecture (*MinPts* = 1) of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department"

Value of ε	Solution
ε = 20%	C1={CI4, CI2, CI3}
	C2={CI8, CI9}
	CI1, CI5, CI6, CI7, CI10 is noise
ε = 50%	C1={CI7, CI6}
	C2={CI4, CI2, CI3, CI8, CI9, CI10, CI5}
	CI1 is noise
ε = 60%	C1={CI7, CI6}
	C2={CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}
ε = 70%	C1={CI1, CI2, CI3, CI4, CI5, CI6, CI7, CI8, CI9, CI10}

The set of results from solving the task of early identification of IS CIs using the DBSCAN algorithm for the case *MinPts* = 2 is given in Table 7. When solving the problem, the previously determined sequence of randomly selected elements was used: CI7, CI4, CI5, CI6, CI1, CI8, CI3, CI10, CI2, CI9.

Table 7

Results of solving the task of early identification of configuration items during the synthesis of a description of the modular architecture (MinPts = 2) of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department"

Value of ε	Solution
ε = 20%	C1={CI4, CI2, CI3}
	CI1, CI5, CI6, CI7, CI8, CI9, CI10 is noise
ε = 50%	C1={CI4, CI2, CI3, CI8, CI9, CI10, CI5}
	CI1, CI6, CI7 is noise
ε = 60%	C1={CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}
	CI6, CI7 is noise
ε = 70%	C1={CI1, CI2, CI3, CI4, CI5, CI6, CI7, CI8, CI9, CI10}

Thus, as final variants of the synthesized description of architecture of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department" it was proposed:

a) variant 1 (ε = 70%) – description of a monolithic architecture, which includes all CIs;

b) variant 2 (*MinPts* = 2, ε = 60%) – description of a modular architecture, which consists of cluster C1={CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5} (functions CI6 and CI7 are proposed to be excluded from this task);

c) option 3 ($MinPts$ = 1, ε = 60%) – a description of a service-oriented architecture, which consists of clusters C1 = {CI7, CI6} and C2 = {CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}.

The descriptions of options 2 and 3 were selected according to the criterion of the minimum presence of CIs in the architecture descriptions, defined as noise.

5. 3. Comparative analysis of features in using clustering methods and algorithms to solve the task of early identification

To identify the features of using the DBSCAN clustering algorithm to solve the task of early identification of IS CIs, it is proposed to compare our results with the results obtained using other clustering methods. As such results, it is proposed to consider:

– the solution obtained as a result of using the method, the basis of which is the Divisive Analysis (DIANA) clustering method [14];

– the solution obtained as a result of using the Agglomerative Nesting (AGNES) clustering method [20];

– he solution obtained as a result of using the Chameleon hierarchical clustering method [22];

– the solution obtained as a result of using the k-means clustering algorithm [21].

In all cases, the same problem was used to test the specified methods, which was also solved during the testing of the DBSCAN algorithm in our study.

The solution from the DIANA method is the cluster dendrogram shown in Fig. 2 [14].

The result of applying the agglomerative clustering method AGNES (using the nearest neighbor algorithm) is the dendrogram of clusters shown in Fig. 3 [20].

The result of applying the hierarchical clustering method Chameleon is the dendrogram of clusters shown in Fig. 4 [23].

The result of using the k-means clustering algorithm is three clusters [21]:

a) cluster 1 (C_1) whose elements are CI1 and CI5 with the center in the conditional element CI11;

b) cluster 2 (C_2) whose elements are CI2, CI3, CI4, CI8, CI9 and CI10 with the center in the conditional element CI12;

c) cluster 3 (C_3) whose elements are CI6 and CI7 with the center in the conditional element CI13.

It is proposed to evaluate the obtained solutions using the following criteria:

– "Cumbersome solution";

– "Identification of separated CIs".

The criterion "Cumbersome solution" characterizes the number of clusters that are provided to the analyst as a result of solving the task of early identification of IS CIs. The results of the comparative analysis of the obtained solutions to the task of early identification according to this criterion are given in Table 8.

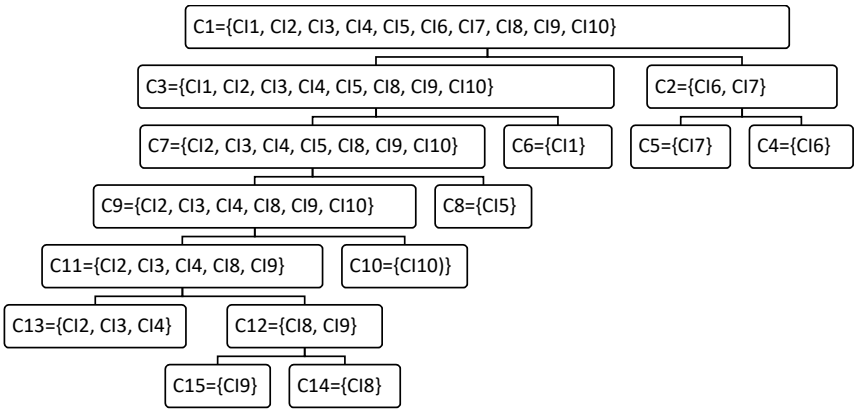

Fig. 2. Dendrogram of clusters of configuration items, formed as a result of applying the Divisive Analysis method

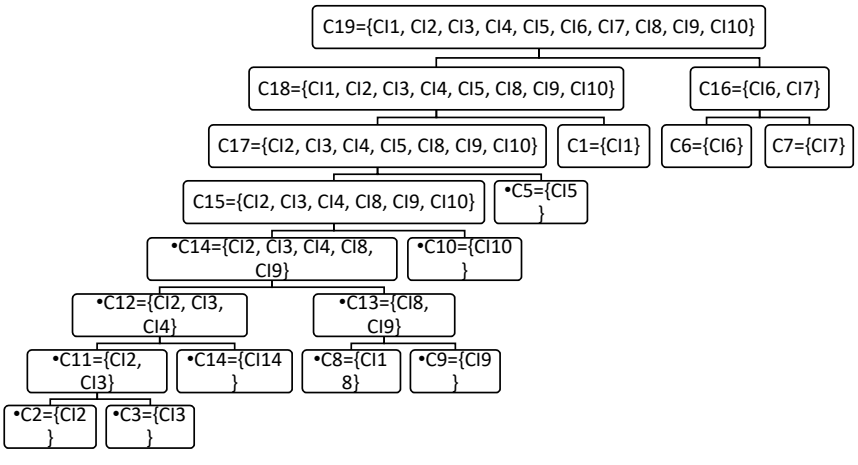

Fig. 3. Dendrogram of clusters of configuration items, formed as a result of applying the Agglomerative Nesting method

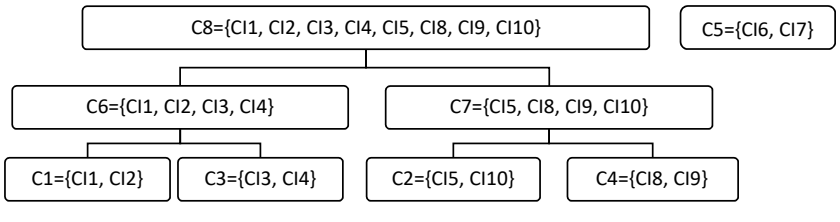

Fig. 4. Dendrogram of clusters of configuration items, formed as a result of applying the Chameleon clustering method

Table 8

Results of comparative analysis of the obtained solutions to the early identification problem according to the criterion "Cumbersome solution"

Designation of methods/algorithms	DB	DI	AG	Ch	k
Description of monolithic architecture					
Number of clusters	1	15	19	8	3
Description of modular architecture					
Number of clusters	1	15	19	8	3
Description of service-oriented architecture					
Number of clusters	2	15	19	8	3

In Table 8 and below, the following notations are adopted: DB – DBSCAN density clustering algorithm; DI – DIANA hier-

archical clustering method; AG – AGNES hierarchical clustering method (using the nearest neighbor algorithm); Ch – Chameleon hierarchical clustering method; k – k-means partitioning algorithm.

The criterion "Identification of separated CIs" characterizes the possibility of detecting such CIs that do not interact with other elements (functions, services, etc.) of the IS in terms of data. This means that isolated CIs do not provide other CIs of the IS or do not receive a significant amount of data for further processing from other CIs. The results of the comparative analysis of the obtained solutions to the early identification problem using this criterion are given in Table 9.

Table 9

Results of comparative analysis of the obtained solutions to the early identification problem using the criterion "Identification of separated CIs"

Designation of methods/algorithms	DB	DI	AG	Ch	k
Description of monolithic architecture					
Identification of separated CIs	No	No	No	–	–
Description of modular architecture					
Identification of separated CIs	Yes (marked as noise for cases $\varepsilon = 20\%$, $\varepsilon = 50\%$, $\varepsilon = 60\%$)	No	No	Yes (as a separate cluster C5)	No
Description of service-oriented architecture					
Identification of separated CIs	Yes (marked as noise for cases $\varepsilon = 20\%$, $\varepsilon = 50\%$)	No	No	Yes (as a separate cluster C5)	No

According to the selected criteria, the DBSCAN algorithm was recognized as the best for use in solving the task of early identification of IS CIs. It allows for all types of IS architectures to obtain solutions with a minimum number of clusters and detects separated CIs (marked as noise). Of the considered methods and algorithms of hierarchical clustering, the Chameleon method is the closest to the DBSCAN algorithm according to the selected criteria. However, this method requires calculations that, in comparison with the other considered methods and algorithms, have the greatest computational complexity.

Separately, it is necessary to consider the problems that arose when using the k-means algorithm to solve the task of early identification. In [21], a comparison was made of the clusters obtained as a result of applying this algorithm and the clusters obtained as a result of applying the DIANA method. According to the results of this comparison, it was recognized that:

– cluster C3 obtained as a result of applying the k-means algorithm coincides with cluster C2, highlighted in the dendrogram shown in Fig. 2;

– cluster C2, obtained as a result of applying the k-means algorithm, coincides with cluster C9, highlighted in the dendrogram, which is shown in Fig. 2;

– cluster C1, obtained as a result of applying the k-means algorithm, has no direct analogues in the dendrogram, which is shown in Fig. 2.

The analysis carried out in [21] proved that the initial decision to highlight the cluster with the center CI1 was erroneous.

The results of comparing the content of the clusters highlighted as a result of using the selected clustering methods and algorithms are given in Table 10.

Table 10

Results of comparing cluster content obtained using selected clustering methods and algorithms

Designation of methods/algorithms	DB	DI	AG	Ch	k
Description of monolithic architecture					
Cluster content	C1 = {CI1, CI2, CI3, CI4, CI5, CI6, CI7, CI8, CI9, CI10}	C1 = {CI1, CI2, CI3, CI4, CI5, CI6, CI7, CI8, CI9, CI10}	C19 = {CI1, CI2, CI3, CI4, CI5, CI6, CI7, CI8, CI9, CI10}	There is no cluster	There is no cluster
Description of modular architecture					
Cluster content	C1 = {CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}	C3 = {CI1, CI2, CI3, CI4, CI5, CI8, CI9, CI10}	C18 = {CI1, CI2, CI3, CI4, CI5, CI8, CI9, CI10}	C8 = {CI1, CI2, CI3, CI4, CI5, CI8, CI9, CI10}	C2 = {CI2, CI3, CI4, CI8, CI9, CI10}
	Noise (CI6, CI7)	C2 = {CI6, CI7}	C16 = {CI6, CI7}	C5 = {CI6, CI7}	C3 = {CI6, CI7}
	No analog	No analog	No analog	No analog	C1 = {CI1, CI5}
Description of service-oriented architecture					
Cluster content	C1 = {CI7, CI6}	C2 = {CI6, CI7}	C16 = {CI6, CI7}	C5 = {CI6, CI7}	C3 = {CI6, CI7}
	C2 = {CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}	C3 = {CI1, CI2, CI3, CI4, CI5, CI8, CI9, CI10}	C18 = {CI1, CI2, CI3, CI4, CI5, CI8, CI9, CI10}	C8 = {CI1, CI2, CI3, CI4, CI5, CI8, CI9, CI10}	C2 = {CI2, CI3, CI4, CI8, CI9, CI10}
	No analog	No analog	No analog	No analog	C1 = {CI1, CI5}

The results given in Table 10 confirm the conclusion drawn in [21] about the incomplete coincidence of the clusters obtained using the k-means algorithm with the clusters obtained using the DBSCAN algorithm and the DIANA, AGNES and Chameleon methods. The main reason for such incomplete coincidence should be recognized as the subjectivity of the choice of individual CIs, which act as the initial centers of clusters in the k-means algorithm. Other clustering methods and algorithms selected for comparative analysis overcome this subjectivity in the following ways:

– construction of a dendrogram that includes all possible variants of clusters (hierarchical methods DIANA, AGNES, Chameleon);

– inclusion of elements in clusters according to the criterion of maximum connectivity between individual CIs (Chameleon);

– inclusion of CIs in clusters according to the criterion of maximum similarity of descriptions of their data structures (Chameleon, DBSCAN).

6. Discussion of results based on using the density clustering algorithm to solve the task of early identification

During the research, the density-based clustering algorithm DBSCAN was adapted to solve the task of early identification. This task is solved during the distribution of configuration items between architectural entities during the synthesis of the description of the IS architecture. The result of solving the task is defined architectural entities (functional subsystems, models or tasks; services, etc.), which combine individual IS functions to facilitate further IS design.

The essence of adapting the DBSCAN algorithm to the features of solving the task of early identification of the IS CI is as follows:

– to determine the features of the representation of the derived dataset D, a formal description of its element (4) is proposed, which is based on the description of a separate function and requirements for it in the form of data flow diagrams and "entity – relationship;

– to determine the distance between the descriptions of individual IS functions, it is proposed to use the modified Chebyshev distance (5), which takes into account the similarity of the structural descriptions of such functions;

– it is proposed to define the radius parameter ε as an indicator that characterizes the relative magnitude of the difference between the descriptions of the data structures of IS (7);

– it is proposed to set the value of the *MinPts* parameter based on the type of desired IS architecture ($MinPts = 1$ for service-oriented architecture, $MinPts = 2$ for modular architecture).

Further, the adapted DBSCAN algorithm was used to solve the task of early identification of CIs for the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department". This task was solved both for the case of choosing a service-oriented IS architecture and for the case of choosing a modular IS architecture. To select the final solutions to the task, it was proposed to use the criterion of minimal presence in the descriptions of the CI architecture, defined as noise.

Our results of solving the task of early identification of CIs have made it possible to conduct a comparative analysis of the results obtained using the DBSCAN algorithm and similar results from the clustering methods and algorithms DIANA, AGNES, Chameleon, and k-means. The criteria "Cumbersome solution" and "Identification of separated CIs" were proposed for the analysis. In addition, an analysis of the coincidence of individual clusters built by using the specified methods and algorithms was conducted.

According to the results of our analysis (Tables 8–10), the solution obtained using the DBSCAN method is recognized as the best in terms of its practical application for the synthesis of a description of the architectural entities of IS. According to the results of the coincidence analysis, it was found that the results obtained using the DBSCAN algorithm do not actually differ from the results obtained using the DIANA, AGNES and Chameleon methods. The difference between the obtained results and the results of the k-means algorithm is explained by the subjectivity of the choice of individual CIs, which act as the initial centers of clusters in this algorithm.

It should also be noted that an additional advantage of the DBSCAN algorithm, compared to the Chameleon method [23] or the graph-analytic method described in [7], is the refusal of DBSCAN to use a graph representation of the description of the IS architecture. An applied consequence of this refusal is the possibility of implementing the DBSCAN algorithm as a set of SQL queries to the repository of descriptions of individual functional requirements for IS or descriptions of individual functions of this IS. The costs of additional memory for storing a graph describing the relationships between individual IS functions become unnecessary.

The main limitations of applying our research results in further work and theoretical studies are:

– use of data flow diagrams and "entity-relationship" as the main sources of information about the functions and data structures of the IS being built;

– use of the method of determining the distance between CIs proposed in [14] in the clustering method;

– the need to construct, at least at the conceptual level, an "entity-relationship" diagram for the IS being built.

The main drawback of this study is the need for a detailed description of each functional requirement and individual functions of the designed IS. This description should be based on the definition of data entities and their attributes. In this case, the incompleteness of this description or the use of natural language for this description could lead to an inaccurate solution to the task of early identification.

Further advancement of this study may tackle several tasks. One of those is conducting research aimed at identifying the features of using the DBSCAN algorithm to synthesize a description of IS architectures based on descriptions of requirements and individual functions of this IS in the form of UML diagrams. Such studies could make it possible to determine the degree of universality of using the DBSCAN algorithm to solve the task of early identification of IS CIs. The second option is the further development of the DBSCAN algorithm to improve the method of synthesizing the description of the architecture of the designed IS. In particular, we are considering the possibility of using the DBSCAN algorithm to detect and eliminate other inconsistencies between the descriptions of individual requirements and IS functions.

7. Conclusions

1. The density-based clustering algorithm DBSCAN has been adapted to the specificity of the task of early identification of IS CIs. The essence of this adaptation is to establish a method for forming a dataset, determine the rules for calculating distances between descriptions of individual IS CIs, and clarify the rules for forming the values of the DBSCAN algorithm parameters. The results of the adaptation make it possible to use the DBSCAN algorithm to solve the task of early identification of IS CIs taking into account the number and proximity of neighbors for each CI.

2. The adapted DBSCAN algorithm was used to solve the task of early identification of CIs. This verification was carried out during the planning of the IT project for the development of the functional task "Formation and maintenance of an individual plan of a scientific and pedagogical employee at a department". A data flow diagram and an "Entity – Relationship" diagram were used as a descrip-

tion of the functions of the task. The results of solving the task of early identification of CIs make it possible to form a description of the architecture depending on the degree of difference in the descriptions of the data structures of individual CIs. As final options, it was proposed to choose a description of monolithic architecture as a set of all CIs; a description of modular architecture as a single architectural entity (cluster C1 = {CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}) with the exception of CI6 and CI7 as noise; a description of service-oriented architecture as two architectural entities (clusters C1 = {CI7, CI6} and C2 = {CI4, CI1, CI2, CI3, CI8, CI9, CI10, CI5}).

3. A comparative analysis of results from solving the task of early identification of CIs obtained using the density-based algorithm DBSCAN, the hierarchical clustering methods DIANA, AGNES, and Chameleon, as well as the k-means algorithm, was carried out. For comparison, the criteria "Cumbersome solution" and "Identification of separated CIs" were proposed. For the solution obtained using the DBSCAN algorithm, the maximum value of the criterion "Cumbersome solution" is 2; the value of criterion "Identification of separated CIs" is set to "Yes". These values are the best among the compared solutions. In addition, an analysis of the coincidence of individual clusters developed using the specified methods and algorithms was carried out. According to the results of the comparative analysis, the DBSCAN algorithm was recognized as the best of those hierarchical clustering methods that participated in our analysis. The use of this method makes it possible to obtain descriptions of architectural IS entities as clusters consisting of descriptions of individual IS functions similar in descriptions of their data structures.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Authors' contributions

Adrian Kozhanov: Methodology, Validation, Formal analysis, Investigation, Writing – original draft; **Maksym Ievlanov**: Resources, Data Curation, Writing – review & editing, Supervision.

References

1. Quigley, J. M., Robertson, K. L. (2019). Configuration Management. Auerbach Publications. https://doi.org/10.1201/9780429318337
2. Bourque, P., Fairley, R. E. (2014). Guide to the Software Engineering Body of Knowledge. Version 3.0. IEEE Computer Society, 335. Available at: https://dl.acm.org/doi/10.5555/2616205
3. IEEE/ISO/IEC 15288-2023. ISO/IEC/IEEE International Standard - Systems and software engineering--System life cycle processes. Available at: https://standards.ieee.org/ieee/15288/10424/
4. Farayola, O. A., Hassan, A. O., Adaramodu, O. R., Fakeyede, O. G., Oladeinde, M. (2023). Configuration management in the modern era: best practices, innovations, and challenges. Computer Science & IT Research Journal, 4 (2), 140–157. https://doi.org/10.51594/csitrj.v4i2.613
5. Reiff-Marganiec, S., Tilly, M. (Eds.) (2012). Handbook of Research on Service-Oriented Systems and Non-Functional Properties. IGI Global Scientific Publishing. https://doi.org/10.4018/978-1-61350-432-1
6. Cadavid, H., Andrikopoulos, V., Avgeriou, P., Broekema, P. C. (2022). System and software architecting harmonization practices in ultra-large-scale systems of systems: A confirmatory case study. Information and Software Technology, 150, 106984. https://doi.org/10.1016/j.infsof.2022.106984
7. Faitelson, D., Heinrich, R., Tyszberowicz, S. (2017). Supporting Software Architecture Evolution by Functional Decomposition. Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development, 435–442. https://doi.org/10.5220/0006206204350442
8. Shahin, R. (2021). Towards Assurance-Driven Architectural Decomposition of Software Systems. Computer Safety, Reliability, and Security. SAFECOMP 2021 Workshops, 187–196. https://doi.org/10.1007/978-3-030-83906-2_15
9. Suljkanović, A., Milosavljević, B., Inđić, V., Dejanović, I. (2022). Developing Microservice-Based Applications Using the Silvera Domain-Specific Language. Applied Sciences, 12 (13), 6679. https://doi.org/10.3390/app12136679
10. Felfernig, A., Le, V.-M., Popescu, A., Uta, M., Tran, T. N. T., Atas, M. (2021). An Overview of Recommender Systems and Machine Learning in Feature Modeling and Configuration. Proceedings of the 15th International Working Conference on Variability Modelling of Software-Intensive Systems, 1–8. https://doi.org/10.1145/3442391.3442408
11. Abolfazli, A., Spiegelberg, J., Palmer, G., Anand, A. (2023). A Deep Reinforcement Learning Approach to Configuration Sampling Problem. 2023 IEEE International Conference on Data Mining (ICDM), 1–10. https://doi.org/10.1109/icdm58522.2023.00009
12. Sellami, K., Saied, M. A., Ouni, A. (2022). A Hierarchical DBSCAN Method for Extracting Microservices from Monolithic Applications. The International Conference on Evaluation and Assessment in Software Engineering 2022, 201–210. https://doi.org/10.1145/3530019.3530040

13. Krause, A., Zirkelbach, C., Hasselbring, W., Lenga, S., Kroger, D. (2020). Microservice Decomposition via Static and Dynamic Analysis of the Monolith. 2020 IEEE International Conference on Software Architecture Companion (ICSA-C), 9–16. https://doi.org/10.1109/icsa-c50368.2020.00011
14. Ievlanov, M., Vasiltcova, N., Neumyvakina, O., Panforova, I. (2022). Development of a method for solving the problem of it product configuration analysis. Eastern-European Journal of Enterprise Technologies, 6 (2 (120)), 6–19. https://doi.org/10.15587/1729-4061.2022.269133
15. Santos, J. L., Martins, L. E. G., Molléri, J. S. (2025). Requirements extraction from model-based systems engineering: A systematic literature review. Journal of Systems and Software, 226, 112407. https://doi.org/10.1016/j.jss.2025.112407
16. Ashfaq, M., Sadik, A. R., Das, T., Waseem, M., Mäkitalo, N., Mikkonen, T. (2026). Runtime composition in dynamic system of systems: A systematic review of challenges, solutions, tools, and evaluation methods. Journal of Systems and Software, 232, 112661. https://doi.org/10.1016/j.jss.2025.112661
17. Han, J., Kamber, M., Pei, J. (2012). Data Mining: Concepts and Techniques. Waltham: Morgan Kaufmann Publishers.
18. Schubert, E., Sander, J., Ester, M., Kriegel, H. P., Xu, X. (2017). DBSCAN Revisited, Revisited. ACM Transactions on Database Systems, 42 (3), 1–21. https://doi.org/10.1145/3068335
19. Evlanov, M. (2016). Development of the model and method of selecting the description of rational architecture of information system. Eastern-European Journal of Enterprise Technologies, 1 (2 (79)), 4–12. https://doi.org/10.15587/1729-4061.2016.60583
20. Vasyltcova, N. V., Panforova, I. Yu. (2022). Research on the use of hierarchical clustering methods when solving the task of IT product configuration analysis. Management Information System and Devises, 178, 37–49. https://doi.org/10.30837/0135-1710.2022.178.037
21. Ievlanov, M., Vasiltcova, N., Neumyvakina, O., Panforova, I. (2024). Use of clustering methods to solve the problem of identifying configuration items in IT project. PROJECT MANAGEMENT: INDUSTRY SPECIFICS, 3–38. https://doi.org/10.15587/978-617-8360-03-0.ch1
22. Evlanov, M. V., Vasyltsova, N. V., Nykytiuk, V. A. (2011). Formalyzovannoe opysanye uslovyi yntehratsyy IT-servysov v ynformatsyonnuiu systemu upravlenyia predpryiatyem. Visnyk Akademii mytnoi sluzhby Ukrainy. Seriya «Tekhnichni nauky», 2 (46), 87–96. Available at: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=vamsutn_2011_2_12
23. Ievlanov, M., Vasiltcova, N., Panforova, I., Moroz, B., Martynenko, A., Moroz, D. (2024). Comparison of solutions to the task of IT product configuration items early identification using hierarchical clusterization methods. Eastern-European Journal of Enterprise Technologies, 3 (2 (129)), 20–33. https://doi.org/10.15587/1729-4061.2024.303526