Classification Methodology for Architectures in Information Systems: A Statistical Converging Technique

Qishan Yang
Dublin City University
Dublin, Ireland *qishan.yang@insight-centre.org*

Mouzhi Ge
Deggendorf Institute of Technology
Deggendorf, Germany *mouzhi.ge@th-deg.de*

Markus Helfert
Maynooth University
Maynooth, Ireland *markus.helfert@mu.ie*

Abstract

Architectures are critical to the Information System (IS) domain because they represent fundamental structures and interactions of systems. Since analysing architecture similarities is challenging and time-consuming even in one domain, IS architecture classifications are paramount to understanding architectural complexity. However, classification approaches used in existing research commonly rely on manual interventions, and thus architectural classification reliability is hampered. We propose a novel methodology based on component modelling and application of a statistical converging technique, which ensures reliable IS architectural classification and minimises subjective interventions. We demonstrate the methodology by classifying data warehouse architectures.

1. Introduction

Information Systems (IS) architectures are important to organising complex systems [4], and understanding architectures not only assists with systematically organising systems with complicated structures and diversified components [17], IS architectures also support software design and development [6] and facilitate information-sharing for businesses [37]. Many disparate types of architectures in IS exist, and they can be interpreted, for example, from software, hardware, or business perspectives. To analyse their characteristics, classifying the architectures is valuable [11, 36].

Although some studies classify IS architectures such as [26] and [32], related research remains limited with providing reliable architectural classifications. A primary reason for lack of reliability is manual interventions, and clarity is also lacking regarding the number of architectures and variations required to establish a consistent and reliable classification [20]. Insufficient information might be used to describe architectures, and disparate architectures can be described in various ways using third-party or self-defined modelling languages. Such drawbacks further limit architecture classifications. Therefore, the research question of this paper is how to systematically classify IS architectures in a reliable way.

This study therefore proposes a systematic architectural classification methodology that allows replications and achieves reliable results. The methodology contains detailed steps required to classify architectures by analysing modelled architectural components in IS. To achieve reliability, we apply a statistical converging technique to classify IS architectures with controlled manual intervention. We propose to apply this methodology to classify architectures in one domain in IS such as smart city architectures or data warehouse architectures, rather than a cross

domain usage. The methodology is validated and demonstrated by classifying data warehouse architectures.

The paper is orgnised as follows. The next section reviews existing research on architecture classifications. Section three describes the steps required to use the proposed architecture classification methodology. The fourth section illustrates how the methodology applies to classifying data warehouse architectures, and the fifth discusses and evaluates the application. The last section summarises contributions of this article and suggests directions for future research.

2. Related Work

We analysed research related to IS architectures and its classification, identifying a common understanding of classification approaches in IS. According to [28, 33], several types of IS architecture exist, including enterprise architecture (EA), software architecture (SA), and computer architecture (CA). EA relates to the business architecture of an enterprise, and it includes aspects of information and technical architecture. EA combines business processes, information systems, and technology to achieve a common goal by focusing on information flows that an organization requires [15]. It also relates to business operations and execution, analysing an enterprise's design, planning, and implementation for development and execution of strategy from an enterprise-wide perspective. It navigates an organisation's business, information, processes, and technology to execute strategies by using architecture principles and practices [14]. SA relates to the detailed structure of a software system, using principles to develop and document its structure. It combines both software elements and their relationships with specifications for configuration of each element [8]. SA represents a combination of a structured system comprised of architectural characteristics, architecture decisions, and principles for design [29]. CA concerns demonstration of physical components and their interrelationships in a system, which benefits engineers when designing and developing technical systems [27]. This type of architecture emphasises the structure of computers, including their hardware components [23]. It sets regulations to explain computer systems regarding their functionality, organisation, and implementation to describe the abilities and program model of a computer without details of its implementation [7].

Given the importance of the term *architecture* in IS and the range of various IS architectures such as [4] and [36], it is difficult to compare such architectures. Architectures can have subtypes across domains, making understanding them challenging, and thus solid classification is required as a fundamental concept of structuring knowledge [34]. To classify the large variety of IS architectures, we consider the types of architectures and classifications described above, with selected studies in these domains discussed below. [35] classify enterprise service-oriented architectures (ESOAs) to design better systems, satisfy business requirements, and reduce enterprise IT complexity and cost. They classify ESOAs into six sub-styles based on a domain model. [2] use exploratory empirical analysis to classify three enterprise architecture scenarios using the hierarchical clustering algorithm. [17] and [32] propose a system and framework, respectively, to analyse and classify enterprise architectures based on dimensions, which allow managers to make decisions easier when choosing EAs. [26] generate a typology to specify and exemplify responsibilities at each layer of SAs from various perspectives, which offers an overview of responsibility types and allocations in business information systems software and enhances a system's analysability, reusability, and portability. [9] propose a unified taxonomy of Infrastructure as a Service (IaaS) based on fundamental components to analyse and classify them into ordered layers, and they apply the taxonomy to design an IaaS architectural framework. [39] create a hybrid algorithm based on a weighted and directed class as a fundamental entity of clustering, which is primarily for object-oriented software architecture recovery. Recent methodologies are tailored to contexts or sub-domains to classify architectures. Such methodologies require significant manual or empirical interventions to generate classifications,

in which investigators produce disparate classifications. The current study proposes a methodology that generates reliable classifications with less human intervention to reduce bias.

3. IS Architecture Classification Methodology

The classification methodology proposed in this paper uses statistical converging during an iterative approach to determine saturation results. It ensures classification reliability based on measuring variations between iterations, consisting of the five phases shown in figure 1. The first phase gathers information of selected architectures as inputs, and the second models these architectures using a modelling language. This ensures that all architectures are comparable, since architectures can be described using various modelling languages. The third phase extracts components from the modelled architectures, which are used as inputs in the classification algorithm during the fourth phase. The classification is then evaluated during phase five. The process continues with phase one until a variation threshold is reached by statistical converging.

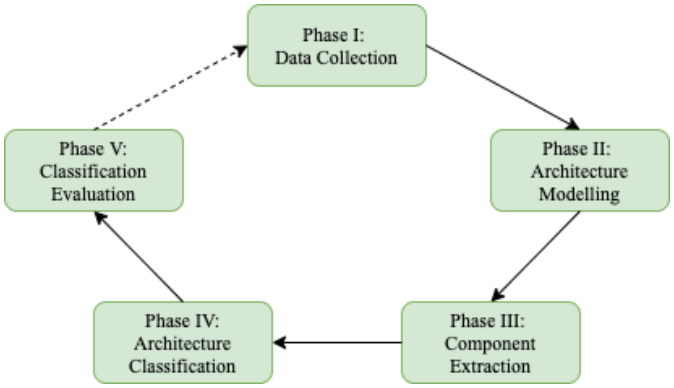

Fig. 1. The phases of the architecture classification methodology

3.1. Phase I: Data Collection

The first phase selects architectures and gathers the data required for modelling. Data might be in various forms, such as text, documents, or architectural models and diagrams, and thus various methods are used to collect the data. Selection of suitable architectures is essential because they represent a range of possibilities and the set of selected architectures determines the scope of final classification. Ideally, data are collected from multiple sources to reduce classification bias. The architectures can be described using disparate modelling languages to demonstrate their structures and components, and thus they are presented using various tools or approaches (e.g., self-defined, process flow, and modelling languages), which complicate the classification. Data are cleaned and transformed into a standardised form for each architecture, manual transformation usually achieves higher quality. Although this offers the opportunity of interpreting architectures, subjective input should be avoided during this stage, which should only be a transformation process of data into standardised form.

3.2. Phase II: Architecture Modelling

Although, modelling IS architectures is challenging and inherently interpretive, automatic modelling tools support and reduce bias during the process, and they increase the reliability of results. Automatic modelling is not further discussed in present research, but it provides an opportunity to boost the modelling process of this methodology in the future. Using a standard modelling language such as ArchiMate is critical to comparing architectures by describing

components and relationships. Once a modelling language is selected, it is used to model all architectures selected during phase one, a step crucial to providing cleansed data in a standardised model for the next phase of extracting IS architectural components.

3.3. Phase III: Component Extraction

After modelling the architectures, we converted them into a machine-readable format to conduct the classification. The output of this phase is a component matrix, which contains a set of components. The matrix is generated by adding components extracted from the modelled architectures. If an architecture has new components, they are added to the matrix, and if an architecture has components that already exist in this matrix, no new components are added. However, in both situations, the architecture is added to the matrix and related component columns are labelled if they contain these components. Although the component matrix may not fully represent an architecture, it offers an applicable method to convert architectures to clustering inputs. The component level must be considered when generating the matrix, since a component might include sub-components, and even sub-components might have sub-components. For example, if two levels of components are used to generate a matrix, only main and sub-components are extracted from each architecture. This decision is made based on requirements; if the architectures are insufficient, the extracted level can be set deeper to enrich the component matrix.

3.4. Phase IV: Architecture Classification

This phase represents the key phase of the process that identifies similarities and differences among architectures by analysing the component matrix. It is designed to retrieve a portion of data to build an initial classification model that is subsequently refined across iterations. We use the criterion of no further additions made during the last two consecutive iterations as a threshold for convergence; if a classification is found (i.e., no changes in comparison to previous results), an additional iteration is used to test statistical convergence. If a classification does not change in two consecutive iterations, classification is complete.

Producing an architecture classification required 10 steps, shown in Figure 2. In step 1, the component matrix provides input into the procedure. In step 2, theoretical expected iterations, $E(x)$, are set to 6, the number of segments is set to N, the threshold for confirmations (T) is set to 2, which means classification should be stable within two consecutive iterations, and the number of segments for testing is set to E. The architectures then needed to be distributed randomly into N segments. During step 4, M segments are chosen to further create an initial classification, and thus an initial classification is generated in step 5 using unsupervised learning algorithms (e.g., hierarchical clustering, K-Means, etc.), since the architectures are unlabelled across some classes. Step 6 assesses whether there exists an unused segment for testing. If there are unused data, several additional steps are followed, but if no data are left for testing, the process jumps to step 10 to output the architecture classification and terminate. During steps 7 and 8, the process loops to test and update the classification using an unused segment at each iteration. During step 9, the updated classification is compared with the previous one, and if the current classification meets the condition of two continuous confirmations, the process moves to step 10 to output the classification and terminate the process. If the classification does not meet the condition, the process returns to step 6.

3.5. Phase V: Classification Evaluation

This phase verifies the quality of results from phase IV, during which an architecture classification is generated. Several approaches can be applied to conduct the evaluation. For our method, we evaluated similarities based on [1], and this intra-cluster similarities should be as high as possible and inter-cluster similarities should be as low as possible. However, finding thresholds

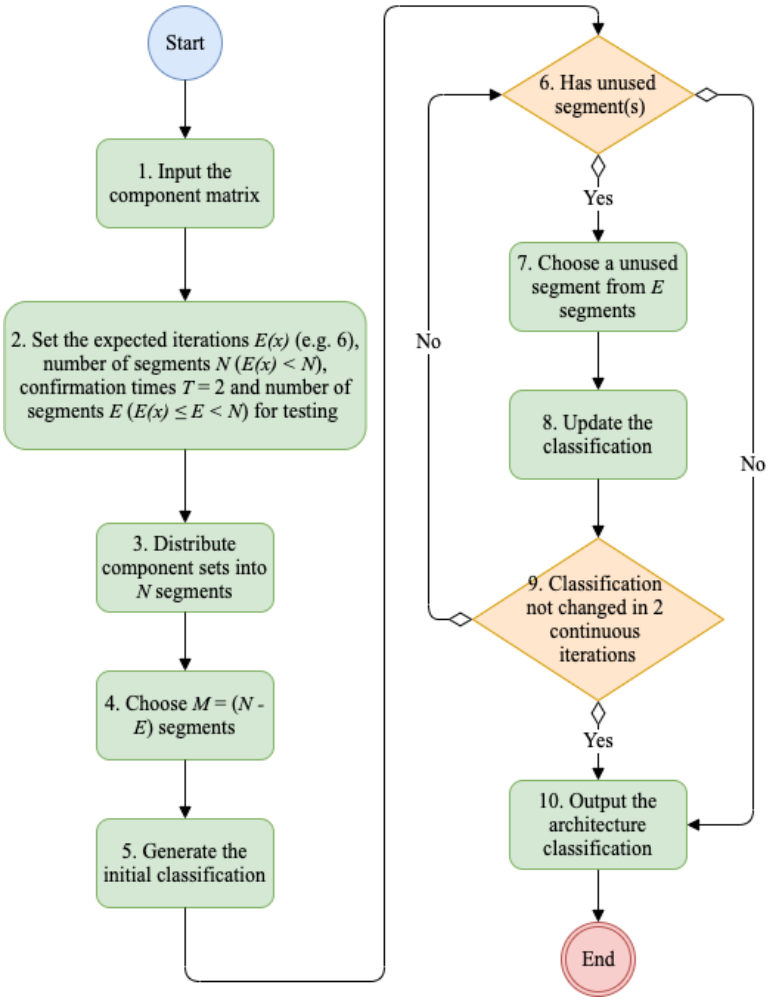

Fig. 2. The process flow in Phase IV: Architecture Classification

of similarities is laborious with this method, and it is difficult to verify whether the similarities are sufficiently high or low. According to [5] and [21], several approaches can be used to evaluate clustering, such as purity, entropy, accuracy, normalised mutual information (NMI), and adjusted rand index (ARI), which require labelled architectures to compare both clustered and labelled results. Therefore, it is efficient to label part of the architectures instead of all of them to test the classification.

After phase V, two scenarios are possible. The first is that the flow returns to phase I if the classification generated does not match requirements. For example, the classification might not fulfil the criterion of two continuous confirmed tests. More architectures are then collected from other resources and the phases are re-executed. The second is the entire process terminating because requirements are met, no test data remained, or no more data were found.

4. Application with Classifying Data Warehouse Architectures

This section discusses validating the classification methodology and demonstrating it based on Figure 1. Data warehouse architectures (DWHAs) are selected as a case study to validate the reliability of a generated classification, with selection based on four criteria: (1) The DWHA domain contains a large variety of architectures proposed to serve disparate contexts, and we thus have sufficient architectural samples for analysis; (2) Although existing research attempts

to classify DWHAs using significant manual interventions, it lacks evidence of classification reliability, and thus it is valuable to classify DWHAs automatically by considering reliability and convergence; (3) In many applications, DWHA is paramount to building systems, and thus it is important to understand how they are built from their classification; (4) Since the DWHA domain has a number of architectures comparable to other domains, such as Big Data and smart cities, we argue that if this methodology can be validated in the DWHA domain, it can be generalised to classify other architectures.

4.1. Phase I: Data Warehouse Architecture Collection

To collect DWHAs, a meta-analysis literature review was conducted to search for them. The literature review was organised into the 5 steps recommended by [30]: (1) identify a problem by observing phenomena, (2) identify search terms to retrieve related publications from the literature, (3) search at least two electronic databases using the terms identified in step 2, (4) screen titles and abstracts to determine eligibility for inclusion, (5) analyse the full text of related articles. In our case, the topic was determining how to generate a reliable classification. Related online resources (i.e., Google Scholar, ACM, dblp, IEEE, and Scopus) were searched using the keywords *data warehouse architecture*. Publications involving *data lake architecture* and *big data warehouse architecture* were searched as supplementary materials. Although many papers were identified during the search, many were also inaccessible. Originally, 158 papers were collected, but after selection and analysis, 116 were used for further analysis. The collected architectures are not instances an architecture.

4.2. Phase II: Data Warehouse Architecture Modelling

After the DWHAs were collected, they were restated by modelling them from a component perspective to further conduct the classification. Various modelling languages can be used to diagram architectures, such as unified modeling language (UML), business process modelling and notation (BPMN), and ArchiMate [40]. We used ArchiMate to model the DWHAs with the following reasons: Compared to ArchiMate, UML and BPMN are mostly focused on software and business processes, On the other hand, ArchiMate is based on concepts from the IEEE 1471 standard [13] and is supported by various tool vendors and consulting firms [22], also it is an open-source modelling toolkit for all levels of enterprise architects [3], and it can be combined with TOGAF [16]. Since some of the DWHAs used in this study have already been modelled using other or self-defined modelling languages, they should be re-modelled using ArchiMate to unify them. Some DWHAs have been only briefly described and diagrammed, and thus it is necessary to model their components explicitly. Nonetheless, some DWHAs, which were either duplicates or whose architectures and components could not be interpreted, were discarded. As a consequence, 68 DWHAs were modelled using ArchiMate, which were derived from 6 data sources with a diversity of structures and components.

4.3. Phase III: Data Warehouse Architecture Component Extraction

This phase converted the DWHAs from a graphical to a digital format that represented a machine-readable pattern. The first step included retrieving a modelled DWHA, extracting components, and marking the related slot in the matrix if a component already existed in the matrix, or adding a new column for a new component if it did not exist in the previous DWHAs. For example, in table 1, the first column contains the ID of the modelled DWHAs. In the first row, the DHWA contains structured data, and we thus placed a 1 into the related cell of the structured data. If the column of the structured data was not included while the first DWHA included the component, we added the component to the table. Before conducting these processes, the component level should be predetermined since a component might contain sub-components and

sub-components might contain sub-sub-components. The component level is determined based on purposes and requirements, but after it is determined, it should be used to extract components across all architectures. In our case, the component level was set to 2, which means components and their sub-components were extracted and added to the matrix. However, if a component had sub-components and its sub-components had sub-components, these sub-sub-components at the third level were not considered. Therefore, each row in the matrix stood for a modelled DWHAs, and if it had several components, related slots were assigned a 1. If a DWHA did not have components but others did, related slots were assigned a 0. Samples of such modelled DWHAs appear in Table 1.

Table 1. Samples of data warehouse architectures in the component matrix

DWHA ID	Structured Data	Unstructured Data	E	EL	ETL	...
1	1	0	0	1	0	...
2	1	0	0	0	1	...
3	1	1	0	1	0	...
4	0	1	0	0	0	...
5	1	0	1	0	0	...
...	...	...	...	...	...	...

4.4. Phase IV: Data Warehouse Architecture Classification

This phase automatically classifies architectures into classes based on the component matrix. Since the number of classes was unknown and there was no label that indicated the classes of these DWHAs, we thus use unsupervised learning for clustering. Different types of algorithms can be applied, such as hierarchical clustering, DBSCAN, K-Means clustering, and fuzzy clustering. In our case the algorithm hierarchical clustering was chosen because it allows easy observation of relationships in the architectures at multiple levels. This clustering also does not require presetting of the number of clusters or thresholds to classify DWHAs, thus requiring less manual intervention, and it flexibly presents classification within a hierarchical or taxonomical structure [31].

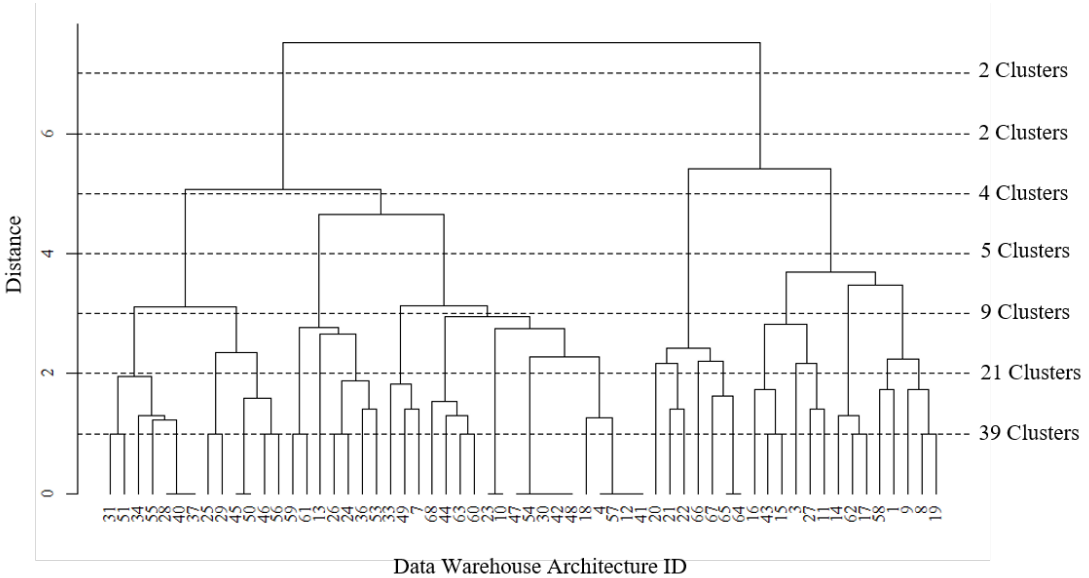

Fig. 3. An example to generate the clustering when the number of DWHAs is 60

The initial component matrix of DWHAs had 47 dimensions. After further analysis, 25 dimensions were selected because eliminated dimensions, such as reporting tools and metadata systems, were non-essential. The component matrix was pre-processed to 68 DWHAs with 25 dimensions and input into the program shown in figure 2. To adhere to the two continuous tests and manage each segment easily, the 68 DWHAs were divided randomly into 14 segments; each segment had 5 DWHAs, except the last, which had 3. For balance, 7 segments were used to create an initial classification and the remaining 7 were used for testing, which is greater than 6, the expected number of iterations. The ending condition was achieved when classification had not changed for two continuous tests, or all data were used. To illustrate how to generate a set of classes, an example is given with 60 DWHAs in figure 3.

Relationships between distances and the number of clusters can be represented as [1, 2, 3, 4, 5, 6, 7] $\Rightarrow$ [39, 21, 9, 5, 4, 2, 2], in which distance is a parameter that measures dissimilarity, or deviation, between clusters. For example, if the distance is 1, the number of clusters, or classes, is 39. To observe the pathway of how the number of clusters changed with distances (e.g., distance 1, distance 2, etc.), each segment (e.g., 5, 10, 15, etc.) was measured and results were tabulated, shown in table 2. After the initial input (35) when distances were 3, 4, 5, and 6, the architectures could be divided into at least two continuously confirmed stable clusters, which generated 9, 5, 2, and 2 clusters when inputs were 50, 60, 45, and 55, respectively. When distances were 1 and 2, the architectures could not be allocated into stable clusters using two confirmations. When distances were 8 and 9, the clusters nearly fixed to 1, which did not classify the architectures and provided less useful information during further analysis. Therefore, the architectures were classified into 9, 5, or 2 clusters.

Table 2. Clusters with different numbers of DWHAs and distances

Distance	5	10	15	20	25	30	35	40	45	50	55	60	65	68
D 1	5	9	13	15	19	21	26	28	30	32	34	39	39	41
D 2	3	5	8	11	13	14	16	17	19	20	20	21	24	25
D 3	1	2	3	4	5	7	8	9	9	9	9	9	9	9
D 4	1	1	1	2	2	3	3	4	4	5	5	5	5	5
D 5	1	1	1	1	2	2	2	2	2	2	3	4	3	4
D 6	1	1	1	1	1	1	1	1	2	2	2	2	2	2
D 7	1	1	1	1	1	1	1	1	1	1	1	2	2	2
D 8	1	1	1	1	1	1	1	1	1	1	1	1	1	2
D 9	1	1	1	1	1	1	1	1	1	1	1	1	1	1

4.5. Phase V: Evaluation for the Data Warehouse Architecture Classification

Results from phase IV suggested that the architectures could be clustered into 9, 5, or 2 clusters, depending on the distances, but how to select better or more appropriate clusters remained unknown even though they met the two continuous confirmation criterion. During this phase, the clusters were evaluated to provide clearer results. To analyse the clusters from a trend perspective, changes were made to clusters regarding distance and different numbers of DWHAs were used. The x-axis represents the number of DWHA inputs, with increments of 5, and the y-axis represents the number of resulting clusters generated by the hierarchical clustering algorithm. Each curve represents a trend of clusters with different data inputs and distances. From top to bottom, the slopes of the lines are metamorphic, sharp to gentle inclines, which suggests that it was more difficult to obtain a stable classification with lower distances because more clusters affected stability more easily than less clusters did.

Based on the analysis above, a top-bottom rule was imposed to discover an appropriate cluster under certain distances. We started with a distance of 1, finding that trend lines for distances

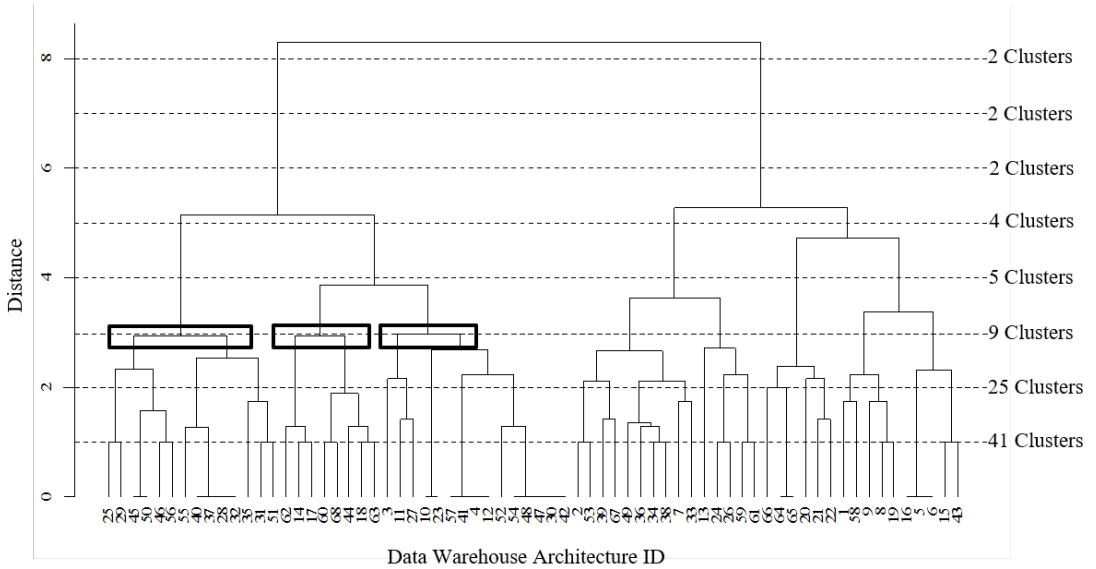

Fig. 4. The hierarchical clustering for the data warehouse architecture classification

1 and 2 sharply and synchronously increased along with the number of DWHAs, which cannot achieve a stable status. When the distance assigned was 3, 4, or 5, the trend and number of clusters in these classifications underwent two stages, from increasing gradually, or fluctuating finely, to nearly stable. The number of clusters in these classifications was manageable for further analysis. If the distance was set to 6, 7, 8, or 9, the trend and number of clusters changed slightly but would offer limited value during subsequent analyses. Therefore, according to the continuous confirmation criterion and the top-bottom scheme, a distance of 3 was considered an appropriate cluster distance, and thus the DWHAs were classified into 9 classes. After inputting 40 DWHAs, the output classification was stable, and thus the 9 classes were largely reliable.

4.6. Classification Results for Data Warehouse Architectures

Results suggested that the DWHAs could be classified automatically into 9 clusters with a distance of 3. When inputting 40, 45, and 50 DWHAs into the clustering algorithm, the number of clusters remained unchanged and the two continuous confirmation criterion was met, and thus the entire process terminated. To further evaluate results, more DWHAs were input into the algorithm to verify the methodology. The remainder of the 18 DWHAs were used as test data to assess the clustering result, which comprised 26.47% of all DWHAs and conformed to the test data size proposed by [10] and [18]. Hierarchical clustering with all DWHAs are shown in figure 4. Since the architectures were split randomly into 14 segments, results generated at each step might be slightly different were the process repeated.

To investigate the characteristics of the DWHAs, they were identified by their components. From observing the DWHAs in each cluster, some had a type of DWHA while others had mixed types due to inadequate information about some DWHAs, and some dimensions (e.g., data stage and operational data store) might have affected clustering. Subsequently, 9 typical DWHAs were identified and generalised, including hub-and-spoke, data mart bus, centralised, independent, federated, virtual, distributed, big, and data lake DWHA architectures, which are explained by [38]. To evaluate results generated by the classification methodology, we conducted a focus group, inviting 5 DWHA experts to review the 9 DWHAs in the 9 clusters. All of the experts reported positive feedback and confirmed that the clusters are useful to identifying typical DWHAs and investigating common components in a cluster. Therefore, the methodol-

ogy could be extended to classify other IS architectures (e.g., enterprise architectures) if they can be modelled. It can also be used to verify distances or similarities of architectures across different numbers of clusters.

5. Discussions

In this paper we define the terminology of IS architecture in a broader scope. IS architectures cover all the architectures that are related to technology, process and people. Thus we consider architecture of data warehouse is a specific domain of IS architecture. Differing the scope of EA and IS architecture is out of scope of this paper. We consider IS architecture as a general scope that the proposed methodology can be applied in.

The proposed methodology brings three main contributions to the existing literature: (1) The methodology allows us to increase the reliability of resulting classifications; (2) We are able to indicate the number of architectures required and the degree of variation among them; (3) Iterative evaluation, using statistical convergence, allows us to increase classification reliability.

Three guidelines were derived from this study's results. First, we recommend modelling architectures using ArchiMate because it has been used widely for that purpose [3], which accommodate ArchiMate models and output classifications. Second, the hierarchical clustering algorithm is recommended to derive architectural outputs. Third, a small number of architecture inputs might not result in a converging trend. Therefore, the proposed converging methodology performs better in domains with a large number of architectures, such as Big Data or smart city architectures. Research results should be evaluated under their respective applications based on requirements [24]. There exist many methods regarding evaluation of artefacts, each offering its own advantages and disadvantages [12, 19, 24]. In the current study, a combined evaluation method was applied to validate the proposed methodology, in which qualitative and quantitative research methods (e.g., case study, interviews, and focus group) were used to evaluate the methodology. Based on recommendations from [25], validity was chosen as the primary dimension measured during evaluation, which can be assessed by measuring an artefact's reliability.

6. Conclusion

This paper proposes a methodology to derive reliable classifications automatically for IS architectures, such as enterprise or software architectures. The classification methodology uses five phases, which are data collection, architecture modelling, component extraction, architecture classification, and evaluation. Given the importance of the architecture classification phase, a step-by-step process is described to guide using the proposed methodology. To validate the methodology, an experiment is conducted to produce a reliable classification for data warehouse architectures, which suggests that the methodology is effective at digitising architectures from models to machine-readable patterns and automatic classification, assuring high reliability. The methodology provides an iterative approach to detecting the number of architectures required to generate a reliable classification that can used to cater the classification of new architecture. As future works, we plan to evaluate the methodology in a broader enterprise architecture context.

References

1. Ahmad, A., Amin, M. R., and Chowdhury, F. (2018). Bengali document clustering using word movers distance. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–6. IEEE.
2. Aier, S., Riege, C., and Winter, R. (2008). Classification of enterprise architecture scenarios-an exploratory analysis. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 3(1):14–23.
3. ArchiMate (2019, October 13). Archi open source ArchiMate modelling. *Archimatetool.*

Retrieved from Available at: https://www.archimatetool.com/.
4. Branco, F., Gonçalves, R., Moreira, F., Au-Yong-Oliveira, M., and Martins, J. (2021). An integrated information systems architecture for the agri-food industry. *Expert Syst. J. Knowl. Eng.*, 38(4).
5. Brodić, D., Amelio, A., and Milivojević, Z. (2017). An approach to the language discrimination in different scripts using adjacent local binary pattern. *Journal of Experimental & Theoretical Artificial Intelligence*, 29(5):929–947.
6. Capilla, R., Jansen, A., Tang, A., Avgeriou, P., and Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, 116:191–205.
7. Clements, A. (2006). *Principles of computer hardware*. Oxford University Press.
8. Clements, P., Garlan, D., Little, R., Nord, R., and Stafford, J. (2003). Documenting software architectures: views and beyond. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 740–741. IEEE.
9. Dukaric, R. and Juric, M. B. (2013). Towards a unified taxonomy and architecture of cloud frameworks. *Future Generation Computer Systems*, 29(5):1196–1210.
10. Forsyth, A. W., Barzilay, R., Hughes, K. S., Lui, D., Lorenz, K. A., Enzinger, A., Tulsky, J. A., and Lindvall, C. (2018). Machine learning methods to extract documentation of breast cancer symptoms from electronic health records. *Journal of pain and symptom management*, 55(6):1492–1499.
11. Genkin, M. and McArthur, J. J. (2023). B-SMART: A reference architecture for artificially intelligent autonomic smart buildings. *Eng. Appl. Artif. Intell.*, 121:106063.
12. Helfert, M., Donnellan, B., and Ostrowski, L. (2012). The case for design science utility and quality-evaluation of design science artifact within the sustainable ict capability maturity framework. *Systems, Signs and Actions: An International Journal on Information Technology, Action, Communication and Workpractices*, 6(1):46–66.
13. Hoidn, P. and Schwidder, K. (2014). Enterprise it architectures-it architecture standards, togaf and omg in more detail, key architecture work products. *University of Zurich. Zurich. Switzerland*.
14. Jnr., B. A., Petersen, S. A., and Krogstie, J. (2023). A model to evaluate the acceptance and usefulness of enterprise architecture for digitalization of cities. *Kybernetes*, 52(1):422–447.
15. Kotusev, S., Kurnia, S., and Dilnutt, R. (2023). Enterprise architecture artifacts as boundary objects: An empirical analysis. *Inf. Softw. Technol.*, 155:107108.
16. Lankhorst, M. and van Drunen, H. (2007). Enterprise architecture development and modelling–combining togaf and archimate. *Via Nova Architectura*, 21.
17. Lantow, B., Jugel, D., Wißotzki, M., Lehmann, B., Zimmermann, O., and Sandkuhl, K. (2016). Towards a classification framework for approaches to enterprise architecture analysis. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 335–343. Springer.
18. Levy, M., Raviv, D., and Baker, J. (2019). Data center predictions using matlab machine learning toolbox. In *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*, pages 0458–0464. IEEE.
19. Mani, N. (2018). *A configuration-based domain-specific rule generation framework for process model customization*. PhD thesis, Dublin City University.
20. Mollajan, A., Iranmanesh, S. H., and Tavakoli-Moghaddam, R. (2023). A systems approach to improve reliability of a contract by modularising contract's information flow architecture: a new contribution to risk mitigation in projects management. *Enterp. Inf. Syst.*, 17(4).
21. Mukherjee, S., Asnani, H., Lin, E., and Kannan, S. (2019). Clustergan: Latent space

clustering in generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4610–4617.

22. Opengroup (2020, October 13). ArchiMate certification. *The open Group*. Retrieved from https://www.opengroup.org/certifications/archimate.
23. Parhami, B. (2019). Computer architecture for big data. *Encyclopedia of Big Data Technologies, S. Sakr and A. Zomaya (eds.)*.
24. Peffers, K., Rothenberger, M., Tuunanen, T., and Vaezi, R. (2012). Design science research evaluation. In *International Conference on Design Science Research in Information Systems*, pages 398–410. Springer.
25. Prat, N., Comyn-Wattiau, I., and Akoka, J. (2014). Artifact evaluation in information systems design-science research-a holistic view. In *PACIS*, page 23.
26. Pruijt, L., Wiersema, W., and Brinkkemper, S. (2013). A typology based approach to assign responsibilities to software layers. In *Proceedings of the 20th Conference on Pattern Languages of Programs*, pages 1–14.
27. Rai, L. and Kang, S. J. (2008). Rule-based modular software and hardware architecture for multi-shaped robots using real-time dynamic behavior identification and selection. *Knowledge-Based Systems*, 21(4):273–283.
28. Rechtin, E. and Maier, M. W. (2009). *The art of systems architecting*. CRC press.
29. Santos, D. S., Oliveira, B. R. N., Kazman, R., and Nakagawa, E. Y. (2023). Evaluation of systems-of-systems software architectures: State of the art and future perspectives. *ACM Comput. Surv.*, 55(4):67:1–67:35.
30. Siddaway, A. (2014). What is a systematic literature review and how do i do one. *University of Stirling*, 1:1–13.
31. Sisodia, D., Singh, L., Sisodia, S., and Saxena, K. (2012). Clustering techniques: a brief survey of different clustering algorithms. *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, 1(3):82–87.
32. Sultanow, E., Brockmann, C., Schroeder, K., and Cox, S. (2016). A multidimensional classification of 55 enterprise architecture frameworks. *Twenty-second Americas Conference on Information Systems*.
33. Sunyaev, A. (2020). Information systems architecture. In *Internet Computing*, pages 25–49. Springer.
34. Swain, D. E. (2022). Knowledge architectures: Structures and semantics. bedford, denise, abingdon-on-thames: Routledge, 2021. 544 pp. (ISBN 9780367219444). *J. Assoc. Inf. Sci. Technol.*, 73(6):892–896.
35. Tang, L., Dong, J., Zhao, Y., and Tsai, W.-T. (2010). A classification of enterprise service-oriented architecture. In *2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, pages 74–81. IEEE.
36. van Dinter, R., Tekinerdogan, B., and Catal, C. (2023). Reference architecture for digital twin-based predictive maintenance systems. *Comput. Ind. Eng.*, 177:109099.
37. Van Engelenburg, S., Janssen, M., and Klievink, B. (2019). Design of a software architecture supporting business-to-government information sharing to improve public safety and security. *Journal of Intelligent information systems*, pages 1–24.
38. Yang, Q., Ge, M., and Helfert, M. (2019). Analysis of data warehouse architectures: modeling and classification. *21st International Conference on Enterprise Information Systems*, 2.
39. Zhang, Q., Qiu, D., Tian, Q., and Sun, L. (2010). Object-oriented software architecture recovery using a new hybrid clustering algorithm. In *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, volume 6, pages 2546–2550. IEEE.
40. Zhi, Q. and Zhou, Z. (2022). Empirically modeling enterprise architecture using archimate. *Comput. Syst. Sci. Eng.*, 40(1):357–374.