

Microservices Infrastructure Architecture for the Cloud-Based Multi-Agent Group Decision Support Systems for Autonomous Cyberphysical Systems

Anatoliy Melnyk^{1,2} and Bohdan Zimchenko²

¹ Lviv Polytechnic National University, Stepana Bandery St, 12, Lviv, 79013, Ukraine

² The John Paul II Catholic University of Lublin, Al. Raclawickie 14, Lublin, 20–950, Poland

Abstract

In this article, we introduce the microservices cloud-based infrastructure architecture for creating a group decision support system (GDSS) designed for autonomous cyberphysical systems (CPS). Our proposed architecture leverages the capabilities of Microsoft Azure cloud computing services. The proposed architecture model assumes that the deployed GDSS can be simultaneously used by many CPSs, as a means of decision support, via an application programming interface (API), using HTTP requests. The creation of the GDSS, based on both fuzzy and nonfuzzy logic principles, and filling data for it can be done by an expert human. The key structural and logical elements of the model of the proposed architecture are described and revealed, including features of the construction of the decision-making intelligent agents. The peculiarities of the development of the GDSS technology, its modern application, and the nearest prospects for the future were also considered. In addition to these findings, we deliver a thorough analysis of the system's potential limitations and areas for its improvement.

Keywords

Microservices architecture, cloud-based application, group decision support system, cyberphysical system, cloud computing.

1. Introduction

An autonomous CPS is a system that can make decisions and take actions without human intervention. Decision-making challenges are acute in such systems, and many technological approaches are used to achieve autonomy based on the architecture of the CPS and the tasks to be performed. For example, dedicated processors can be used for decision support [1].

The concept of decision support arose primarily from theoretical research of organizational decision-making in the late 1950s and was first implemented in the 1960s [2]. Since then, DSSs have been widely used in various areas of human activity, as well as in CPSs.

Given the fact that the CPS can have a heterogeneous and complex architecture, the approaches to building the architecture of the DSS itself can change. If for a relatively simple CPS, an effective solution may be the creation of a monolithic DSS based on an expert system (ES), then for a more complex CPS, which may consist of many interconnected nodes, the scalability, and maintenance of such DSS may be a difficult and too costly process.

In this case, it makes sense to divide the DSS into smaller monolithic architectural parts that can be interconnected and make decisions within their working area, that is, a set of rules, input data, and a way of making decisions. Such division within one system can be ensured by relying on the approaches of building a microservice architecture. Such approaches can be used for many software solutions, including cloud-based applications. Microservices are the simplest entities and services that accept incoming requests and perform actions. For example, consider a function that gets activated


International Scientific Symposium «Intelligent Solutions» IntSol-2023, September 27–28, 2023, Kyiv-Uzhhorod, Ukraine

EMAIL: aomelnyk@gmail.com (A. Melnyk); bohdan.v.zimchenko@lpnu.ua (B. Zimchenko)

ORCID: 0000-0002-8981-0530 (A. Melnyk); 0000-0002-4574-4068 (B. Zimchenko)

© 2023 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

when a particular event transpires, or a backend service that remains perpetually available. To put it simply, it refers to a function or a collection of functions that can be invoked via a specific API and operated across the network. In some aspects, it resembles a monolithic application.

However, it only executes a fragment of the system's logic and does not perform all its functions. In contrast to a monolith, the resulting application consists of many relatively small, independent services called microservices that communicate over a computer network. A typical microservices architecture of the web-based application is shown in Figure 1.

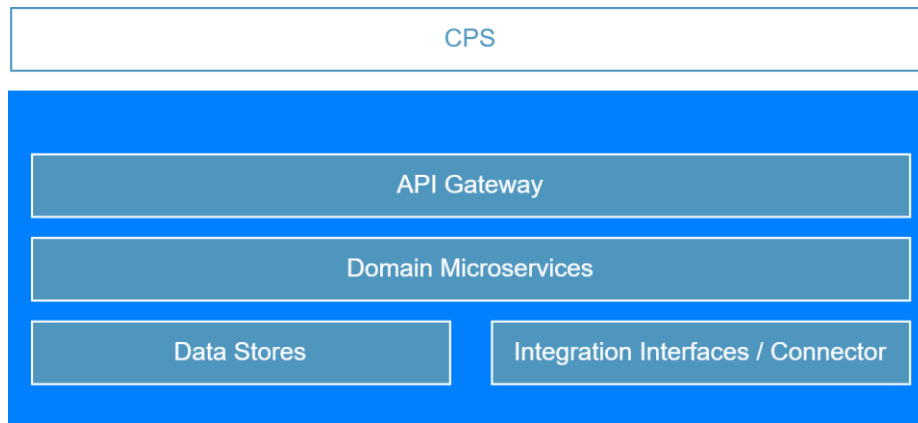


Figure 1: Microservices architecture of the web-based application

API gateway is a communication environment between a CPS and a DSS and mostly is responsible for such functionality as request handling, data transferring, access control, and security policies. Domain microservices are responsible for business logic and operate on the data and resources under their control.

The best practice is to ensure that a database belongs to only one microservice. No other service should be able to directly access the database, only through the owning service's API. This allows each service to manage its own database consistency and structure. There also can be chosen the best database for the specific purpose, regardless of the target's overall system requirements. For example, one service can use a normalized Structured Query Language (SQL) database while another can benefit from her Not only SQL (NoSQL) database. However, one microservice can manage multiple databases.

The main goal of this work is to propose a conceptual model of the cloud-based GDSS with a microservices-based architecture that can communicate with other systems through the communication environment, in particular the Internet. The issue of integrating a GDSS into a CPS can be resolved by using a cloud-based application. This enables engineers and programmers to concentrate more on resolving problems of group decision-making and performance difficulties that may occur. This GDSS can be utilized concurrently by several CPSs.

This paper is organized as follows: Section 2 presents reviews on existing GDSSs. Section 3 describes the decision-making agents of the GDSS in the proposed architecture. Section 4 depicts the elements of the proposed system architecture using Microsoft Azure services for the GDSS. Section 5 overviews the relational database for data, referred to as both fuzzy and nonfuzzy logic, its tables, and their fields. Section 6 presents the discussion. Section 7 describes the conclusions and future works.

2. State-of-the-art

The idea of using computerized DSSs is discussed by James Reason in his work on human error under the heading of intelligent DSSs. James Reason notes that the events following the Three Mile accident have given him less confidence in the effectiveness of some of the existing methods in decision-making. For example, in the Davis-Besse accident, both independent safety parameterization systems failed before and during the event [3].

Computerized DSSs were developed to help decision-makers consider the impact of different ways of thinking and reduce the risk of human error. Because making decisions may involve many considerations, the problem of group decision-making nowadays is acute.

The authors of the paper [4] reviewed several weighting methods related to the problems of GDSS to express the influence of the decision results based on parameters and decision-makers. This work [5] presents a web-based GDSS that supports groups of decision-makers regardless of their location. The proposed GDSS allows the construction of each multi-criteria problem, its alternatives, and the criteria that can be used to evaluate each choice.

In the paper [6], an architecture for the cloud-based GDSS is discussed and actions, that need to be taken to introduce decisional processes into the cloud at infrastructure, platform, service, and business levels, are presented. A new group decision-making model has been developed and presented in this research [7] that enables the implementation of group decision-making procedures with many participants and alternatives. Its primary goal is to create a setting for the Internet-based paradigm. A technique for group decision-making is suggested by the authors in this paper [8]. This approach uses machine learning to automatically classify and choose experts. They use a two-tuple language structure to convey and quantify their judgment opinions. Here, the suggested approach is used to study Yuyuantan Lake in Beijing. The decision-making procedure and outcome are examined and debated in this work.

Two major contributions are provided in this work [9]: the first is a reflection on the current status of web-based group DSSs, specifically on their shortcomings and potential barriers to their adoption by organizations, with an emphasis on situations involving dispersed decision-makers. The second is the suggestion of a conceptual web-based GDSS with a microservices-based architecture that is specifically created for dispersed group decision-making and tries to address the issues found in this work. In this paper [10], novel methods for building priority vectors in GDSSs from components provided by functions discovered through regression modeling are investigated. It proposes that different priorities can be predicted for specific respondents, profiled by each predictor, predicted over time, examined by the importance of the predictor, and evaluated by the significance and quality characteristics known in regression modeling. This research [11] explains a group decision-making procedure based on the analytic hierarchy process. To organize the problem and elicit expert opinions, a Delphi method was set up.

An organized survey of the most recent fuzzy preference modeling approaches is the focus of this work [12]. Following a brief overview of each technique and taking a look at the fundamentals of existing approaches, the authors highlight some pressing problems and challenges in fuzzy preference modeling. Authors of this work [13] provide a fuzzy best-worst multi-criteria group decision-making strategy. They consider that experts can only rank the other criteria in accordance with some set of criteria because there are numerous ways that only provide the best and worst criteria when integrating expert evaluation, which will cause information loss or distortion. To avoid this, the proposed strategy is discussed.

3. Decision-making agents of the GDSS

The term "agent" describes a software abstraction, idea, or concept, similar to object-oriented programming terms like method, function, and object. The agent concept provides a convenient and powerful way to describe a complex software entity that can operate with some degree of autonomy to perform tasks on behalf of its host [14].

Within the proposed architecture of the cloud-based multi-agent GDSS for autonomous CPS, microservices act as intelligent agents of the GDSS. It is proposed to create an agent as a monolithic unit capable of solving a well-defined, limited problem that can be formulated by decomposing more complex, non-trivial problems. Two main basic types of agents are offered: nonfuzzy agent and fuzzy agent, which are driven by straight IF-THEN logic and fuzzy logic approaches accordingly.

The diagram of the nonfuzzy agent is shown in Figure 2. The main element of the nonfuzzy agent is a service that performs decision-making functions based on the rules laid down in the knowledge base by the human expert and the input data to which these rules must be applied. Also, depending on the agent's tasks, constant values may be used during internal calculations or data conversion. The fuzzy agent is suggested to be created to solve such problems that deal with fuzzy and inaccurate information, so that the solution itself may be not clear or indistinct. It is more complex than the fuzzy agent and can include such structural elements as shown in Figure 3.

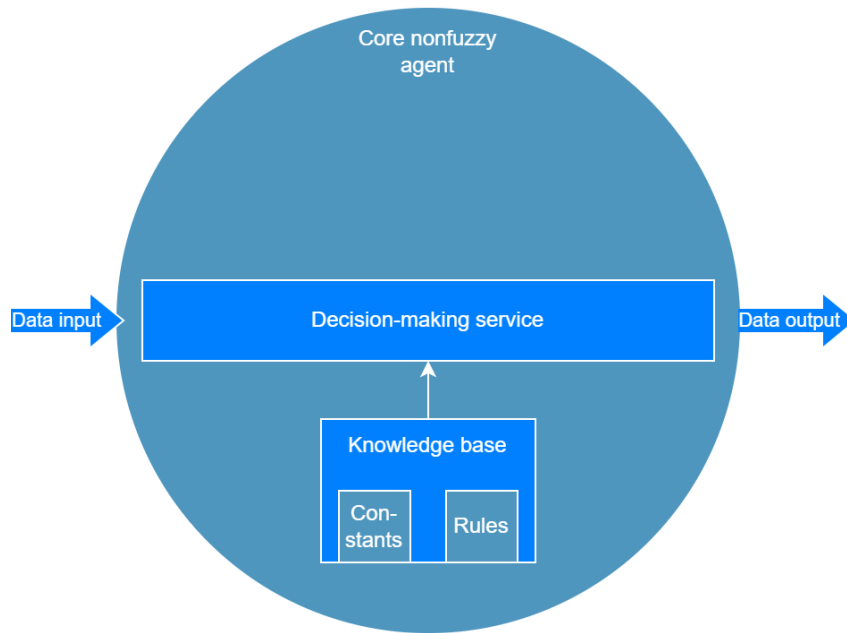


Figure 2: Nonfuzzy agent

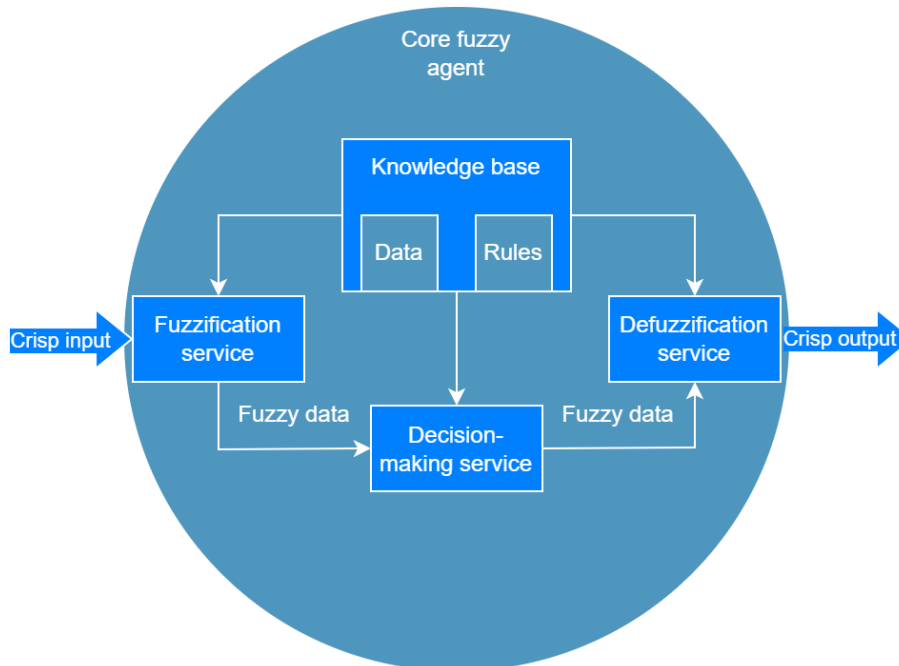


Figure 3: Fuzzy agent

The fuzzification service converts crisp sets to fuzzy sets. By using the input membership function, the input becomes fuzzy. In this method, the numerical inputs of the system are assigned to fuzzy sets with some degree of membership. This degree of membership belongs to the interval $[0,1]$. Each value between 0 and 1 represents the degree of uncertainty about belonging to the values in the set. A decision service operates on rules. First and foremost, there should be defined a set of rules stored in a database that can be applied to a set of input and output data. The rule strength is then determined for each rule by combining the fuzzified inputs according to the rules, resulting in a set of rule strength values. During rule processing, Boolean operators used in rule descriptions are replaced with Zadeh operators, please, check Table 1. All the results are combined and passed to the defuzzification service as a fuzzy set of results to get the output distribution. The defuzzification service then computes the arithmetic mean of the resulting fuzzy set for each output value and uses the output membership function to transform the arithmetic mean into a distinct value. Finally, we get the defuzzified crisp output data.

Table 1
Zadeh operators

Boolean	Fuzzy
AND(x, y)	MIN(x, y)
OR(x, y)	MAX(x, y)
NOT(x)	1 - x

4. Proposed system architecture using Microsoft Azure services

Choosing Azure as the bedrock for the microservices architecture of the GDSS is motivated by the multitude of advantages that this cloud computing platform presents, which are crucial for fulfilling the main goals of our project. Azure's comprehensive array of services and features facilitates the smooth creation and roll-out of serverless applications. The proposed architecture of the GDSS benefits from a monolithic approach that delivers a highly flexible, scalable, and cost-effective framework, smoothly adapting to different requirements that engineers might face while developing the GDSS.

Furthermore, utilizing Azure's cloud framework eliminates the need for the procurement and upkeep of expensive on-premises hardware, thus reducing the total cost of ownership, and simplifying the challenges commonly associated with managing conventional infrastructure. Azure also provides high availability, resilience to failures, and stringent security measures, all of which are indispensable for safeguarding the data and processes involved in evaluating the dependability of power supply systems. The proposed system architecture using Microsoft Azure services for the GDSS, the structural diagram of which is shown in Figure. 4, incorporates several critical components.

The architecture's process commences with the CPS, where interactions with the system are initiated. When a CPS triggers a request, it is dispatched as an HTTP API request to the Load Balancer, setting the process in motion. The Load Balancer is a crucial element that supports the system's scalability and resilience. It acknowledges the CPS's HTTP API request and forwards it to the Crossover service. The Load Balancer improves the overall performance and dependability of the system by ensuring that the load is spread equally.

The Crossover service, a microservice, acts as a bridge between the Load Balancer and other services. It is instrumental in linking various elements of the system and facilitating the exchange of information. Upon receipt of the request from the Load Balancer, the Crossover service liaises with both Redis and one more relational database. Redis stands here for a quick memory bank where the system stashes data that is used a lot, user sessions are managed, and live stats are stored. Meanwhile, the relational DB is used for storing persistent data crucial for the GDSS's operations. The crossover service dispatches downstream API requests to the Multiplex service for task orchestration.

The Multiplex service is another microservice that accepts downstream API requests from the Crossover service. It is tasked with managing tasks and ensuring they are processed correctly. When the Multiplex service receives a request from the Crossover service, it initiates the necessary operations and returns a job ID to the Crossover service. This job ID serves as a reference for tracking the status and results of the task.

The Evaluator service is tasked with long polling API requests from the Crossover service. It plays a crucial role in monitoring the status of jobs and retrieving results. The Evaluator service interfaces with both the Multiplex and Result-Fetcher services to perform these operations. It ensures that the system is constantly updated with the status and results of ongoing tasks.

The Result-Fetcher service is tasked with ensuring that the results of the jobs initiated by the Multiplex service are correctly retrieved and made available for further processing. After making their final decisions, the Fuzzy Agent and Non-Fuzzy Agent communicate their findings to the Result-Fetcher. After retrieving the results, they are then streamed back to the Evaluator service, ensuring a steady supply of data and prompt decision-making. The Result-Fetcher service plays a crucial role in the system's data flow, ensuring that data is accurately and efficiently retrieved from storage and made available for further processing.

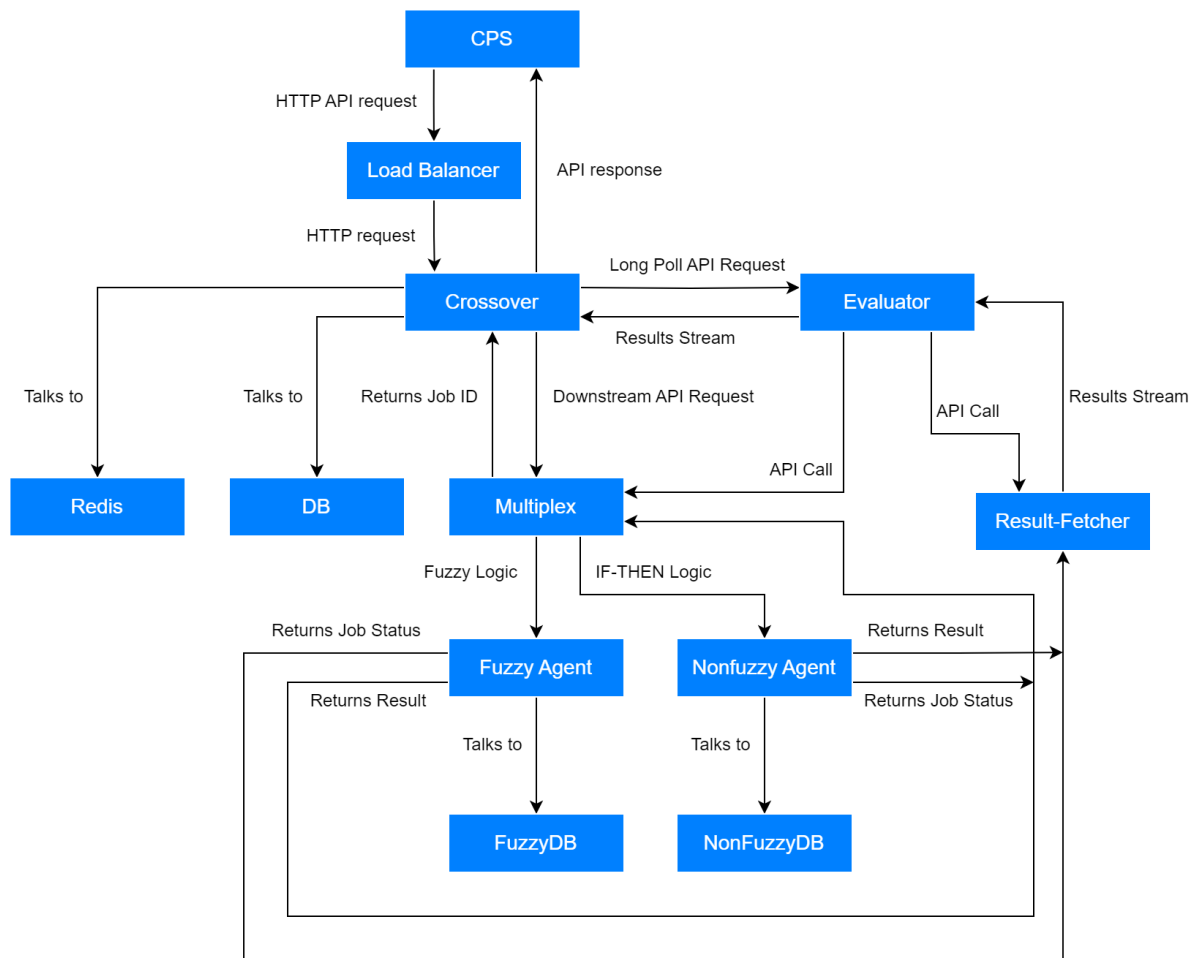


Figure 4: Proposed System Architecture using Microsoft Azure services

The system's intelligent agents are the Fuzzy Agent and Non-Fuzzy Agent. They tackle specific issues using fuzzy logic and IF-THEN logic respectively. These agents have an important role in the decision-making process. They examine the data, then implement the appropriate decision-making logic, and finally make decisions based on the previous analysis. The determinations made by these agents orchestrate the system's operations, assuring its efficient and effective functionality. Both agents are linked to their individual databases, which harbor data pertinent to the decision-making process. Each of these nodes signifies a segment of the system, and they collaboratively operate as outlined to carry out the system's overall function. The architecture is engineered to be resilient, scalable, and proficient, capitalizing on the potency and adaptability of Azure cloud services. The multi-layered, multi-agent strategy, in conjunction with the use of both fuzzy and non-fuzzy logic, pledges to yield an efficacious decision-making system for intricate, interlinked CPSs.

5. Relational database for the the decision-making agents

According to the classic microservices architecture pattern, each microservice has a connection to a separate database that only it has access to. Thus, in the presented architecture, it is suggested that each decision-making agent uses an isolated database system, which should contain all the necessary data sets and rules for solving the problem, that should be utilized by the specific agent. The databases for each agent, as well as the agents themselves, are designed by engineers, programmers, and experts at the stage of development of the GDSS application. Although the real database schema can be significantly different for two different agents of the same type, which is related to the nature of the problem to be solved, however, it is possible to single out similar features of the construction of relational database schemas for agents. The proposed relational database used by the fuzzy agent consists of four tables. The diagram is shown in Figure 5.

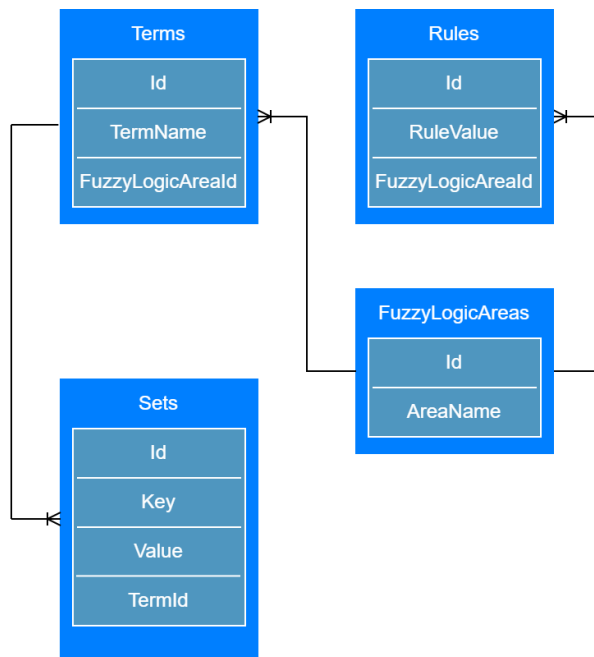


Figure 5: Relational database diagram for the fuzzy agent

The FuzzyLogicAreas table record has an AreaName property that corresponds to the topic area it solves. This table can contain many records because even a basic problem can be decomposed into parts and the separation of data and rules into categories can be performed. It has a one-to-many relationship with Rules and Terms. A Term table record has a TermName String property that represents a linguistic variable. A term table stores a set of linguistic variables for a particular subject. This table has a one-to-many relationship with the Sets table.

The Sets table record has a key string property, which is the linguistic value that can be applied to the corresponding linguistic variable, and its numeric value, which is the Value double property. A rule table record has a RuleValue String property that represents a rule in a specified format using a linguistic variable and its corresponding linguistic value. Rules are defined as IF-THEN constructs.

The database schema for the nonfuzzy agent is shown in Figure 6 and is similar to the database schema designed for the fuzzy agent, except that there is no need for Sets and Terms tables since nonfuzzy logic operates on inputs and constants that the agent may need for some calculations or data transformation. For this purpose, the database provides a Constants table that has Key and Value records and is related to the table LogicAreas with the many-to-one relationship.

6. Discussion

Researchers who have examined the effects of GDSS on CPS are somewhat separated from others who are developing techniques that can be used with cloud-based GDSS. Behind the scientific success of the several methodologies that were used at the end of the 20th century to evaluate them, the GDSS's failure was evident, a fact that cannot be denied. It is crucial to remember that Watson, DeSanctis, and Poole [15] conducted research in 1988 comparing the effects of using a GDSS with groups simply using paper and pencil and those that used no help at all. It was determined that there were no benefits to utilizing a GDSS.

They assumed that the GDSS's failure might be related to the fact that it was a new form of application and even went so far as to say that additional research is necessary before giving up on the idea of letting groups use GDSS independently without technical support or other assistance.

This story has persisted more subsequently, and in 2016 van Hilleberg and Koenen [16] investigated the precise causes of businesses' delays in implementing GDSS. They concluded that there are now no incentives for businesses to invest in GDSS because there is no proof of their genuine advantages. Additionally, they discovered that the presence of a facilitator is essential for the system to function, but that it incurs excessive costs, making it harder for GDSS that are applied to

more generic decision settings to succeed. However, due to their more specialized context, GDSS-like spatial DSSs are more likely to succeed.

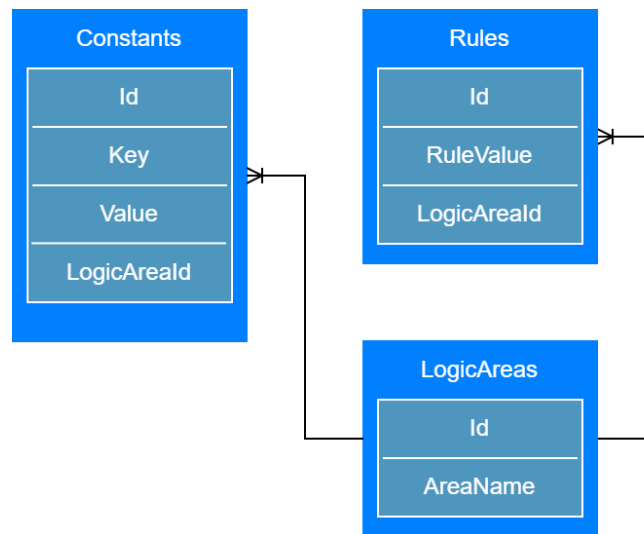


Figure 6: Relational database diagram for the nonfuzzy agent

Therefore, the researchers must rethink some of the strategies that have been developed for GDSSs. Supporting group decision-making in what is a typical human process requires more than the knowledge of methods. At its core, it necessitates a comprehension of the process and the dynamics of group decision-making so that these various strategies and methodologies can cater to the needs of the central figures in the scenario: the decision-makers. To address the decision-makers' requirements, it is crucial to honor the decision-making process and understand that during the period in which the entire process unfolds, it isn't only the state of the decision that experiences changes and alterations, but the decision-makers themselves also evolve and adjust their ideas and preferences. For this reason, it's important to realize that time can often be the crucial element that facilitates this maturation, thus leading to better-quality decisions.

The development of technologies of decentralized systems and architectures, which can be applied to both engineering solutions and software, significantly accelerates the development of GDSS as a decision-making tool in CPSs nowadays. The development and maintenance of such systems are becoming increasingly cheaper, which attracts new investments in these technologies, which contributes to their more widespread use in various architectural forms. Therefore, it can be safely assumed that GDSS for a CPS can become a promising direction of research in the near future.

7. Conclusions and future work

In our proposed architecture, we have integrated a multi-agent system, composed of both fuzzy and nonfuzzy logic principles, to facilitate the process of group decision-making. This architecture comprises several critical components that ensure the effective functioning of the GDSS operations and efficient management of data. These include the Load Balancer for efficient request handling, the Crossover service for bridging various services, and a set of intelligent agents for problem-solving. The integrated data layer includes Redis and MySQL databases that bolster the system's capability for rapid data access and reliable structured data management.

Despite its innovative design, the development of a successful cloud-based GDSS presents its unique set of challenges. The inherent complexity of the microservice architecture demands meticulous attention to details such as fault tolerance, data consistency, and infrastructure management. While these considerations present a steep learning curve, they are undeniably critical for the stability, reliability, and overall efficiency of the system.

Looking ahead, future work on this topic will aim to delve deeper into the nuances of implementing a GDSS. The primary objectives will be to bring this conceptual model to fruition and identify any potential issues that might compromise the success of the GDSS. Subsequently, we aim

to develop robust solutions to these challenges. Also, we anticipate the need to further refine the architecture to suit different use cases and application domains.

As organizations continue to evolve, the requirement for such flexible, efficient, and internet-accessible GDSSs is set to rise. Hence, researchers delving into this topic must consider the dynamics of group decision-making and the functional context when formulating their methodologies. Our work serves as a preliminary measure in this regard. By presenting this novel architecture, we hope to foster further innovations in this significant field.

Additionally, we foresee the need for comprehensive testing and evaluation of this model under real-world scenarios. This can involve implementing prototype systems and conducting user acceptance testing to gather valuable feedback and improve the system's functionality. Furthermore, exploring options for seamless integration with existing organizational IT infrastructures can open up new avenues for the practical application of our proposed GDSS architecture.

In conclusion, the development of cloud-based GDSSs based on microservices architecture presents a novel direction for future research. As we move forward, the insights gleaned from this work will provide a valuable foundation for the continued exploration and development of effective and efficient GDSSs.

8. References

- [1] A. Melnyk, V. Melnyk, Specialized processors automatic design tools – the basis of self-configurable computer and cyber-physical systems, in: 2019 IEEE international conference on advanced trends in information theory (ATIT), IEEE, 2019, p. 326–335.
- [2] P. G. W. Keen, M. S. Scott Morton, Decision support systems: an organizational perspective, Addison-Wesley, 1978.
- [3] J. Reason, Human error, Cambridge University Press, Cambridge, 1990.
- [4] H. Hamdani, R. Wardoyo, K. Mustofa, Weighting model for group decision support system: A review, Indones. J. Electr. Eng. Comput. Sci. 11.3 (2018) 962–974.
- [5] L. Conceição, D. Martinho, R. Andrade, J. Carneiro, C. Martins, G. Marreiros, P. J. Novais, A web-based group decision support system for multicriteria problems, *Concurr. Comput. Pract. Exp.* 33.23 (2019).
- [6] C. Radu, C. Căndea, G. Căndea, C. B. Zamfirescu, Towards a cloud-based group decision support system, *Recent Dev. Comput. Collect. Intell. Springer* 513 (2014) 187–196.
- [7] J. A. Morente-Molinera, G. Kou, I. Javier Pérez, K. Samouylov, A group decision making support system for the Web: How to work in environments with a high number of participants and alternatives, *Appl. Soft Comput.* 68 (2018) 191–201.
- [8] Y. Yang, Y.-T. Bai, X. Wang, L. Wang, Group decision-making support for sustainable governance of algal bloom in urban lakes, *Sustainability* 12.4 (2020).
- [9] J. Carneiro, P. Alves, G. Marreiros, P. J. Novais, Group decision support systems for current times: overcoming the challenges of dispersed group decision-making, *Neurocomputing* 423 (2020).
- [10] S. Lipovetsky, Predictor analysis in group decision making, *Stats* 4.9 (2021) 108–121.
- [11] Y. Blanco, Consensus building in a group decision-making process, *Pesqui. Oper.* 40.2 (2020).
- [12] W. Guan, W. Lingjiu, L. Yushenga, Y. Xiaopingc, A review on fuzzy preference modeling methods for group decision-making, *J. Intell. & Fuzzy Syst.* 40.6 (2021) 10645–10660.
- [13] S. Guo, Z. Qi, A Fuzzy Best-Worst Multi-Criteria Group Decision-Making Method, *IEEE Access* 9 (2021) 118941–118952.
- [14] M. Wooldridge, N. R. Jennings, Intelligent agents: Theory and practice, *Knowl. Eng. Rev.* 10.2 (1995) 115–152.
- [15] R. T. Watson, G. DeSanctis, M. Scott Poole, Using a GDSS to facilitate group consensus: some intended and unintended consequences, *MIS Q.* 12.3 (1988) 463–478.
- [16] J. V. Hillegersberg, S. Koenen, Adoption of web-based group decision support systems: Experiences from the field and future developments, *Int. J. Inf. Syst. Proj. Manag.* 4.1 (2016).