# A Semi-Automated Approach for Resolving Data-Driven Architecture Missmatches

Christos Karathanasis
*Department of Applied Informatics*
University of Macedonia
Thessaloniki, Greece
christoskarathanasisac@gmail.com

Theodoros Maikantis
*Hephaestus Lab, Dept. of Chemistry*
International Hellenic University
Thessaloniki, Greece
teomaik19@gmail.com

Nikolaos Nikolaidis
*Department of Applied Informatics*
University of Macedonia
Thessaloniki, Greece
nnikolaidis.se@gmail.com

Apostolos Ampatzoglou
*Department of Applied Informatics*
University of Macedonia
Thessaloniki, Greece
apostolos.ampatzoglou@gmail.com

Alexandros Chatzigeorgiou
*Department of Applied Informatics*
University of Macedonia
Thessaloniki, Greece
achat@uom.edu.gr

Nikolaos Mittas
*Hephaestus Lab, Dept. of Chemistry*
International Hellenic University
Kavala, Greece
nmittas@chem.ihu.gr

***Abstract*—In contemporary software development, there is a need for delivering solutions that require the integration of multiple software systems, each one relying on different architectural decisions. For instance, an e-shop solutions must communicate with the ERP solution that the company possesses to handle prices, products, and stock. However, such an integration is not always a trivial issue since interoperability problems might arise. A root cause for such interoperability issues is architecture mismatches: e.g., caused by heterogeneity on how data are stored and are expected to be exchanged in the two systems. Interoperability problems can cause delays to the development, require extended communication with different teams, and usually adds complexity to the system. In this paper, we propose a semi-automated AI-based approach and a middleware software solution ("*a connector*") to aid software engineers in "*connecting*" applications with heterogeneous data storing schemas. We have validated our approach and tool with a company that connects ERP systems with e-shops, through a qualitative study.**

***Keywords*—*architecture mismatch, technical debt, heterogenous data, interoperable software, software connectors***

## I. Introduction

In the fast-paced realm of modern business, where data serves as the lifeblood of decision-making [1], the seamless integration of diverse software applications has become a pivotal factor in sustaining operational efficiency and innovation. As businesses increasingly rely on a multitude of software solutions to drive various aspects of their operations through digitalization [2], the challenge of interoperability emerges as a critical bottleneck [3]. As data storage and presentation occur autonomously within distinct systems, the crucial capacity of diverse software systems to communicate and share data becomes essential, exerting a significant impact on the speed, agility, and adaptability of organizations within the constantly evolving technological landscape. A notable example concerns, the pressing need for a data interface between databases and Enterprise Resource Planning (ERP) systems. This imperative need stems from the escalating complexities associated with managing vast datasets and the central role played by ERP systems in orchestrating streamlined organizational processes. However, navigating the intricate terrain of crafting an effective interface proves to be a formidable challenge, characterized by diverse data structures, disparate functionalities, and the demanding requirement for real-time synchronization.

To underscore the critical need for a well-designed interface, envision a software development company seeking to integrate its customer's database with an ERP system. The customer organization needs real-time information to facilitate decision-making, inventory management, customer service, and customer relationship management. All this information is contained within a database that stores customer information, purchase history, and inventory data. The main issue that arises in creating and maintaining a suitable interface is the different logic with which data is stored in the two systems, reflecting different architectural assumptions and choices. This illustrative scenario vividly underscores the strategic advantage of bridging the gap between disparate datasets and the multifaceted functionalities of ERP systems.

While the necessity for integration is evident, the prevailing state of affairs often involves manual integration processes, introducing inherent challenges. The current landscape heavily relies on human expertise to navigate the intricacies of both databases and ERP systems. Both the creation and upkeep of such solutions are demanding and time-intensive challenges for a company. In terms of development, there is a need to write code in the form of queries and services. The queries extract data based on the sophisticated logic stored in the ERP and then this data must undergo processing to align with the targeted logic for transmission. Afterwards, the data is transmitted, and the e-commerce store update is executed. Executing this entire process demands a profound understanding of the operational logic of both systems, necessitating collaboration between the ERP management team and the e-commerce store management team. The resulting code includes business logic and is highly tailored, exclusively addressing the nuances of this specific interface. We can see that such a solution is high demanding in terms of effort and person-hours but offers no further value other than the targeted use case. As a result, the tailor-made software solution being non-reusable by its creator results in a higher selling price, while reusability or ease of creation would have led to a lower cost. Though feasible, this manual intervention introduces inefficiencies, elevates the risk of errors, and demands extensive time and resources. These issues are part of the Architectural Technical Debt (ATD) for a service-oriented architecture [4]. The ensuing problems stemming from manual integration underscore the urgency for a more efficient and automated solution. Finding a seamless balance between the intricacies of development and the imperative need for automated integration will undoubtedly enhance the overall efficiency and reliability of the system.

In recognition of these challenges, the integration process can be revolutionized through automation. Leveraging advanced technologies, particularly Machine Learning models, presents a transformative approach to system integration. The innovative idea revolves around training models on the intricacies of the database and the ERP system, offering a potential solution to the complexities posed by manual processes.

The subsequent sections of the paper are structured as follows: Section 2 gives an overview of the background information and related work. Section 3 presents the proposed approach in terms of architecture and methodology, while Section 4 presents the tool that has been created. Finally, Section 5 presents the results of our validation efforts and Section 6 concludes the work and discusses potential limitations.

## II. Related Work

In this section, we present existing studies and background information regarding our research area. First, we present studies regarding the architectural technical debt, and how it is connected with the interoperability of data. Moreover, we present the notion of heterogenic data and the problem they introduce to microservices; finally, we present some existing approaches to similar problems.

### A. Architectural Technical Debt in Microservices

De Toledo et al. [4] by studying a big project, that used the microservices architecture, with about 100 services in total tried to find ATD issues. Through the analysis of documents and interviews, they came up with a list of issues, and they mapped each one to the notions of Interest and Principal. One of the most noteworthy findings was the issue regarding the communication of the services since it was noted as one of the most important ones. This issue can become more intense when the business logic is inside the communication layer. Moreover, De Toledo et al. [5] investigated the most known microservice-specific ATD issues. Through interviewing 47 practitioners they found out the issues encountered in migration, in the future, and more difficult to resolve. From this study, we can see that the microservice coupling, insufficient metadata, and data sharing/synchronization are the most recognizable. This goes to show the importance of the connection between services and the data they share. A similar study [6], with the use of 25 interviews resulted in the same outcomes. They found in total of 16 ATD issues, including insufficient metadata in messages, lack of communication standards among microservices, inadequate use of APIs, and use of business logic in communication among services.

### B. Microservices with Heterogenic Semantic Data

Sharing knowledge coming from different and heterogeneous environments and services is becoming more and more common. The study of Nace and Aissani [7] shows exactly that, and the need that semantic communication has for moving forward. They also conducted a throw review of two web solutions JAVA RMI and CORBA, by also stating that a more improved solution is needed. Nagarajan et al. [8] also investigated and noted the need for semantic communication, but they also proposed a new approach with predefined mappings. However, the mapping and matching issues are still noted as a big problem that needs to be addressed in the future.

With the need to solve the semantic data in the communication of services, Viennot et al. [9] proposed Synapse. This platform lets services share data in a semantic and scalable way while supporting a wide variety of SQL and NoSQL databases. Similarly, Marques-Lucena [10] proposed an approach for semantic web services, along with a mapping strategy. However an easy-to-use integration for the mapping between the elements is still missing.

### C. Connection of Heterogenic Data Services

The problem of heterogeneous data emerged from the beginning of services and even more in the last years with microservices, and a lot of studies have been conducted in order to find the best way to communicate. Huf and Siqueira [11], tried to find out how heterogeneous services can be composed. They studied 66 documents, published from 2005 to 2018, and despite the large number, there are still several open issues. Resende and Feng [12], recognizing the problem of heterogeneous data in service-oriented architectures, tried to introduce the Service Data Object specification in order to resolve part of this issue. Moreover, Tan et al. [13] proposed a solution in order to solve this problem, by providing a high-level taxonomy. This solution was also compared with four existing ones, but none of them is better just each of them providing different features.

With the increasing big data, Sowe et al. [14] noted that this problem is still present and more intense. Kong et al. [15] tried to present the problems for connecting ERP systems with E-Commerce. They proposed a methodology based on web services but the issue of data heterogeneity still remained. Similar methodologies have been proposed but without having a lot of traction and without solving the mapping issue [16][17].

## III. Proposed Methodology

In this section, we present the proposed methodology which can help with the connection of heterogeneous data software. The four points and steps of this solution are the following:

- Initially, by utilizing artificial intelligence we train a model specific to the needs of the service we want to connect.
- Secondly, we use the trained model, which can generate SQL queries leveraging natural language, thereby supporting the creation of the SQL query which will select the data to be sent from one application to another.
- Third, again with the help of artificial intelligence, we provide a way that implements the mapping between the data returned by the previous SQL query step and the JSON file that the second application can receive and consume.
- Finally, after the previous two steps have been successfully completed, the application produces a project that implements the connection and is ready for use.

### A. AI Used Tools

In the last years, an increasing use of AI models has been used in software. Even though the AI model evaluation varies a lot in different studies [18][19], we cannot overlook their accuracy in simple demonstrative problems and repetitive tasks [20]. In the context of our approach, for easier the connection of heterogeneous data software components, we rely on two different AI “co-pilots”.

First of all, ***Vanna AI*** is a natural language processing model specialized in generating SQL queries[1]. It can be tailored to each project while training can take place by importing data related to the database schema, with the use of the documentation as well as by importing SQL queries. Usually, a combination of the above methods is required to adequately train a model and achieve greater accuracy. Once the model is ready, the user can enter queries in natural language and receive the corresponding SQL queries. In case the users are not satisfied with the resulting queries, further model training can take place.

Vanna provides an API and a Python library through which it can be used. In general, the API makes it possible to create new models, train existing ones and, of course, execute queries on them. It is worth noting that Vanna can generate queries for any type of database.

Finally, ***Hugging Chat*** is a natural language processing model from Hugging Face[2]. We should note that it has a similar function to the well-known ChatGPT system. The reason this particular model was chosen is the open-source nature of the project and the free API without any restrictions on the number of calls. In addition, a Python library (hugchat) is provided which facilitates the use of the API.

## IV. Proposed Tool

To automate the proposed methodology and bring it closer to the developers, we developed the Data Mismatch Connector tool. This tool was created as a web application with the use of the Blazor framework and the Radzen library in C#. We note that the code of the application[3], along with the accompanying Python scripts for Hugging Chat[4] and Vanna[5] can be found online. The main functionalities of the application are: a) the training of the model; b) the SQL query generator; c) the C# class generator; and d) the generation of the mapping.

In order to present the tool, we selected a simple use case, with dummy tables, that could be part of a real-world case. When opening the application, the first step is to connect to the database, to be able to access the tables and perform the SQL queries. After this step, the user can create or select a Vanna AI model, that is going to be used in the next views of the application. When creating a new model, the user has three options to feed the model with data. We should note that more information will always help with the accuracy of the model. The three options for training the model which are visible in Figure 1, are: 1) by providing the table/view names and documentation for each one, 2) by providing SQL queries for the tables, and 3) through documentation in the form of free text.

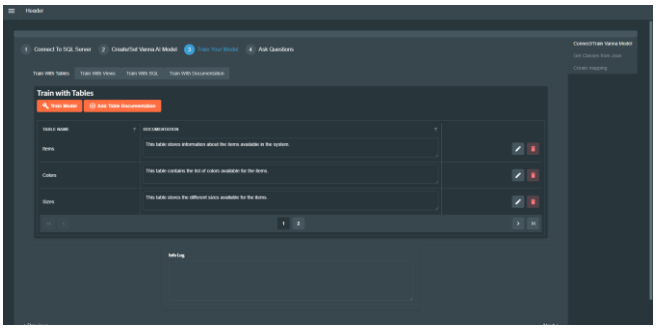


Fig. 1. Model training.

Another useful feature of the application is the generation of SQL queries, and at this point, the user can test only this functionality in order to know if the model is well-trained (see Figure 2). We should note that the model can be retrained at any moment.

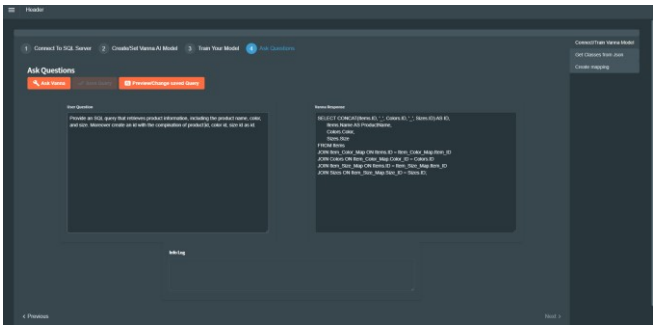


Fig. 2. Generation of SQL queries.

Moreover, the tool provides the creation of classes based on the query that will be needed to send (see Figure 3). So, the user needs to add the data format that needs to be sent from one service to the other, in the form of JSON. The tool will create the C# classes that are needed in order to later create the corresponding objects to fill and send from one service to the other. We should note that the user can change and finetune the generated classes.


Fig. 3. Generation of C# classes.

The fourth and final view of the tool is related to the mapping of the classes and generation of the project (see Figure 4). In order to do the mapping, the Hugging chat needs the SQL query that was generated in the previous steps, along with the class that it was generated from the provided JSON. In the last step, the user should describe the mapping that is needed. This description can be as extended as the user likes, in order to specify details of the projects and idioms that might

[1] https://vanna.ai
[2] https://huggingface.co
[3] https://github.com/ChristosKarathanasisac/integration_builder_project
[4] https://github.com/ChristosKarathanasisac/hugging_chat_api
[5] https://github.com/ChristosKarathanasisac/vannaIntegrationApi

exist specific to each use-case. However, form most cases a simple prompt like the following should suffice.

"Create a method that uses an Integration Data object with the data being located in the table."

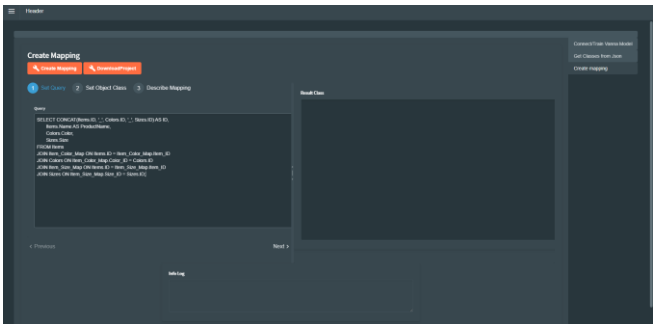

Fig. 4. Generation of mapping.

When the user is satisfied with the description and the information that was provided, the application allows downloading of the generated project. The generated project is in the form of a Windows Service, written in C#. Figure 5 provides the code for the Integration Data, from the generated project, which is used for the communication.

```
namespace IntegrationWinService
{
    public class Payload
    {
        public string product_id { get; set; }
        public string name { get; set; }
        public string color { get; set; }
        public string size { get; set; }
    }

    public class IntegrationData
    {
        public string type { get; set; }
        public List<Payload> payload { get; set; }
    }
}
```

Fig. 5. Example of the generated class.

Figure 6 provides the functionality for the mapping of the data received from the database to the Integration Data object. Finally, Figure 7 lists the functionality for making the actual call to the database, by utilizing the mapping functions (Mapping Fun).

```
private IntegrationData MappingFun(DataTable d)
{
    var integrationData = new IntegrationData();
    integrationData.type = "your_type"; // replace with your desired

    integrationData.payload = new List<Payload>();

    foreach (DataRow row in d.Rows)
    {
        var payload = new Payload
        {
            product_id = row["ID"].ToString(),
            name = row["ProductName"].ToString(),
            color = row["Color"].ToString(),
            size = row["Size"].ToString()
        };

        integrationData.payload.Add(payload);
    }

    return integrationData;
}
```

Fig. 6. Example of generated function for mapping.

```
namespace IntegrationWinService
{
    class IntegrationBuilder
    {
        private string _dataSource = @"LENOVO-CHRISTOS\SQLEXPRESS";
        private string _initialcatalog = @"DB1";
        private string _url = "url";
        public void StartUpdate()
        {
            try
            {
                string sqlQuery = @"SELECT CONCAT(Items.ID, '_', Colors.ID, '_', Sizes.ID) AS ID,
                                   Items.Name AS ProductName,
                                   Colors.Color,
                                   Sizes.Size
                            FROM Items
                            JOIN Item_Color_Map ON Items.ID = Item_Color_Map.Item_ID
                            JOIN Colors ON Item_Color_Map.Color_ID = Colors.ID
                            JOIN Item_Size_Map ON Items.ID = Item_Size_Map.Item_ID
                            JOIN Sizes ON Item_Size_Map.Size_ID = Sizes.ID;";
                string error = "";
                DataTable dt = GeneralUtils.GetDataTable(sqlQuery, _dataSource, _initialcatalog, out error);

                IntegrationData data = MappingFun(dt);
                string json = GeneralUtils.CreateJsonString(data);

                string repsonse = GeneralUtils.SendRequestPostNew(_url, out error, json);

            }
            catch (Exception exc)
            {

            }
        }
```

Fig. 7. Example of generated function for query.

## V. Validation

To evaluate the proposed methodology and tool, we have performed an initial exploratory empirical study, that was designed and reported based on the guidelines of Runeson et al. [21].

### A. Study Design

To study the proposed methodology and tool, with respect to its acceptance in the industry, in terms of real-world systems, we have formulated the following research questions:

***RQ1***: Does the methodology meet the expectations of the practitioners?

***RQ2***: Does the developed tool meet the expectations of the practitioners?

***RQ1*** will provide insights into the methodology and overall architecture, while ***RQ2*** will provide feedback on the usability of the tool. The validation was conducted during a 1-day workshop to one company in Greece, that specializes in ERP and E-Commerce solutions. First, we conducted interviews with members of the company (developers and managers) in order to find out the status of the existing solutions with regard to the interoperability between components that rely on different databases. After that, the participants were presented with our proposed solution and tool, with the use of a mock database. Moreover, they were asked to use the tool, involving a real-world database and an actual use-case of the company. After spending some time for familiarizing themselves with the developed tool, they were given a questionnaire to evaluate it. Regarding the evaluation of the tool, the System Usability Scale (SUS) instrument was used [22]. Finally, the workshop closed with interviews regarding the evaluation of the methodology as a whole.

### B. Results and Discussion

Navigating the outcomes of our study, we present the results of two crucial tests alongside the insights derived from the System Usability Scale (SUS). These results offer a pragmatic evaluation of our methodology and its associated tool, providing useful user perspectives. Based on user feedback, we explore the implications these results carry for practitioners in the field of software development.

#### 1) Acceptance of Methodology (RQ1)

The exploration of the acceptance of the proposed methodology in response to RQ1 offers insights shaped by participants with respect to the AI practices. The prevailing popularity of

AI practices, currently considered "mainstream" in the technological landscape, underscores the relevance and timeliness of the developed methodology. The adaptability of AI approaches to solving complex problems has been exemplified, contributing to the tool's overall positive reception. Notably, the widespread use of natural language queries, particularly demonstrated through the effective application of ChatGPT, signifies a notable advancement. This innovative aspect allows users to seamlessly translate natural language queries into code, whether it be SQL or services in this case, marking a significant stride in technological sophistication. The technologies employed and their implementation reflect a level of advancement and modernity, aligning with current trends in AI. The combination of these cutting-edge elements ultimately manifests in the result, showcasing the effectiveness of the methodology in addressing contemporary challenges and positioning it as a promising solution in the current landscape of AI practices. With all of these in mind, the participants liked the proposed approach to tackle the mismatch problem with AI. Even though they were reserved in the beginning, they saw the value of both AI models and the resulting connector service in terms of reusability and cost.

*2) Acceptance of Tool (RQ2)*

***Mock Database***. During the mock database testing phase, developers and managers were presented with the proposed tool. Feedback from this phase revealed that the performance of the tool was very much aligned with their expectations. The participants found the tool to be effective in simulated scenarios, with its accuracy and responsiveness to be very satisfying. Given that this testing was performed in a less complex mock database, the performance although satisfactory, needed to be proven in more complex and demanding tasks.

***Real Database***. The evaluation of the developed tool's real-world applicability involved rigorous testing on one of the company's actual ERP and E-Commerce databases. This comprehensive evaluation provided insightful and overall satisfactory outcomes, shedding light on the tool's capacity to handle the intricacies of real-world data. The tool not only demonstrated its technical prowess but also showcased its effectiveness in seamlessly integrating with and adapting to the dynamic nuances of a live business environment.

However, as with any complex technological solution, challenges surfaced during the testing phase, particularly in relation to the AI model's requirement for a notable amount of training data to achieve optimal performance. This aspect of the evaluation process, while resource-intensive, was anticipated, considering the inherent intricacies involved in working with a sizable and complex dataset. The substantial effort invested in training the AI model proved to be a strategic investment, as the model, once adequately trained, exhibited remarkable efficiency in generating queries that reflected a nuanced understanding of the underlying database content.

Moreover, the evaluation unveiled an additional challenge related to the natural language querying functionality of the tool. This particular aspect required a certain level of user expertise to formulate queries effectively. Despite the initial hurdle, users with a foundational understanding of the system were able to successfully leverage the tool. This underscores the adaptability of the tool to users with varying levels of expertise, emphasizing its potential usability across a broad spectrum of end-users.

In essence, the multifaceted evaluation not only highlighted the tool's technical prowess but also emphasized its adaptability, efficiency, and potential for widespread use in real-world scenarios, thus establishing it as a promising solution for businesses operating as Database Integration Providers.

***System Usability Scale (SUS)***. The System Usability Scale (SUS) instrument was utilized to quantitatively assess the usability of the developed tool. Participants, when providing ratings on the various aspects measured by SUS, demonstrated an overall favorable perception in areas directly related to their concerns. Conversely, in aspects less directly related to their primary concerns, the SUS scores were more average. This nuanced evaluation indicates that, specifically in the dimensions of the tool most pertinent to the participants' priorities, the ratings were notably positive. Overall, these findings suggest an improvement in current practices, reinforcing a positive and favorable perception of the tool's usability among the practitioners involved in the study.

In a more thorough examination users expressed satisfaction with the tool's accessibility, emphasizing that specialized knowledge was not a prerequisite for operation. Furthermore, the seamless integration of the tool's functionalities contributed to a smooth user experience, garnering positive remarks from participants. Most users found the tool easy to operate, indicating a high degree of user-friendliness. Despite its user-friendly nature, some participants suggested that additional documentation or user assistance might be beneficial for first-time users. The tool was generally perceived as non-cumbersome, featuring a streamlined design that facilitated ease of use. Once demonstrated, users reported that the tool became very easy to use, indicating its potential for quick adoption and proficiency. While the tool instilled a degree of confidence in its output, participants emphasized that it cannot entirely replace human expertise. However, the tool was recognized as non-complex and consistent in its operation, contributing to its positive reception. Depending on individual use cases, participants expressed a desire to use the tool more frequently, underlining a favorable inclination towards its integration into their workflows.

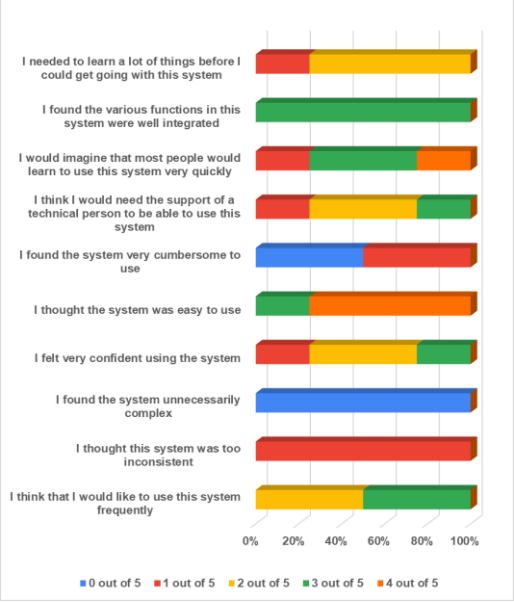


Fig. 8. Usability of Proposed Tool.

*3) Practical Implications*

The acceptance of the methodology (RQ1) and the positive evaluation of the developed tool's usability (RQ2) carry significant practical implications for both end-users and practitioners in the field.

The user-friendly nature of the developed tool, as highlighted by participant feedback, implies that individuals with varying levels of technical expertise can seamlessly incorporate the tool into their workflows. The absence of a steep learning curve and the tool's accessibility make it a valuable asset for users across different domains. The suggestion for additional documentation or user assistance for first-time users serves as a practical insight, emphasizing the importance of providing supplementary resources to enhance the onboarding process. Moreover, the tool's potential for quick adoption and proficiency, suggests that organizations can efficiently integrate it into their existing processes without significant training overhead. This rapid assimilation aligns with the current demand for agile and adaptable tools in dynamic work environments. The recognition that the tool cannot entirely replace human expertise underscores the importance of maintaining a balanced approach, combining the strengths of the tool with human insights. This acknowledgment informs practitioners about the tool's role as a supportive, non-complex, and consistent resource, emphasizing collaboration rather than replacement. In terms of the methodology's acceptance, the alignment with mainstream AI practices positions it as a relevant and timely solution. The demonstrated ability to translate natural language queries into code reflects a practical advancement, streamlining processes and enhancing the efficiency of solving complex problems.

## VI. Conclusions

During our research, we successfully identified key challenges in the field of software integration, focusing on architecture mismatch, heterogeneous data, and data interoperability. The identification of the aforementioned problems provides valuable insights into the intricacies of software integration and highlights the necessity for a novel and automated approach to address real-world challenges. In response to the identified challenges, we formulated a comprehensive methodology leveraging state-of-the-art AI techniques. Additionally, to validate and demonstrate the practical applicability of our approach, we implemented the methodology through the development of a dedicated tool. In order to access our work, we rigorously evaluated the effectiveness of our tool by conducting tests in collaboration with a company specializing in data integration solutions. The testing process comprised two distinct phases. Initially, a proof-of-concept test was executed using mock data to demonstrate the feasibility of our solution. Subsequently, a second test was performed using real-world data and databases, aiming to validate the practical applicability of our methodology in a real-world setting.

The results from the database testing phase demonstrated the tool's noteworthy performance in both mock and real simulated scenarios, showcasing its effectiveness, accuracy, and responsiveness. The real database testing phase, conducted on the company's live ERP and E-Commerce databases, provided insightful outcomes, highlighting the tool's capacity to handle real-world data intricacies. Despite challenges related to the AI model's training requirements, the investment in training proved strategic, resulting in impressive efficiency in generating nuanced queries. The System Usability Scale (SUS) results highlighted the tool's favorable usability, positively impacting current practices and aligning with user priorities. In conclusion, the study's findings emphasize the adaptability, efficiency, and significant potential for widespread use of the developed tool, positioning it as a promising solution for businesses functioning as Data Integration Providers in the contemporary technological landscape.

## Acknowledgment

This work has been partially funded by the European Union's Horizon Europe Framework Programme under grant agreement No 101132663 (project SKILLAB), and partially by the University of Macedonia Research Committee as part of the "Principal Research 2023".

## References

[1] Jordon, A. (2023). Data Governance: A Pillar of Modern Business Intelligence.

[2] Parviainen, P., Tihinen, M., Kääriäinen, J., & Teppola, S. (2017). Tackling the digitalization challenge: how to benefit from digitalization in practice. International journal of information systems and project management, 5(1), 63-77.

[3] Nilsson, J., & Sandin, F. (2018, July). Semantic interoperability in industry 4.0: Survey of recent devel-opments and outlook. In 2018 IEEE 16th international conference on industrial informatics (INDIN) (pp. 127-132). IEEE.

[4] S. Soares de Toledo, A. Martini, A. Przybyszewska and D. I. K. Sjøberg, "Architectural Technical Debt in Microservices: A Case Study in a Large Company," 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), Montreal, QC, Canada, 2019, pp. 78-87, doi: 10.1109/TechDebt.2019.00026.

[5] S. S. De Toledo, A. Martini, P. H. Nguyen and D. I. K. Sjøberg, "Accumulation and Prioritization of Architectural Debt in Three Companies Migrating to Microservices," in IEEE Access, vol. 10, pp. 37422-37445, 2022, doi: 10.1109/ACCESS.2022.3158648

[6] de Toledo, S. S., Martini, A., & Sjøberg, D. I. (2021). Identifying architectural technical debt, principal, and interest in mi-croservices: A multiple-case study. Journal of Systems and Software, 177, 110968.

[7] Nacer, H., & Aissani, D. (2014). Semantic web services: Standards, applications, challenges and solutions. Journal of Net-work and Computer Applications, 44, 134-151.

[8] M. Nagarajan, K. Verma, A. P. Sheth, J. Miller and J. Lathem, "Semantic Interoperability of Web Services - Challenges and Experiences," 2006 IEEE International Conference on Web Services (ICWS'06), Chicago, IL, USA, 2006, pp. 373-382, doi: 10.1109/ICWS.2006.116.

[9] Viennot, N., Lécuyer, M., Bell, J., Geambasu, R., & Nieh, J. (2015, April). Synapse: a microservices architecture for heter-ogeneous-database web applications. In Proceedings of the tenth european conference on computer systems (pp. 1-16).

[10] C. Marques-Lucena, J. Sarraipa, C. Agostinho and R. Jardim-Goncalves, "Model-driven approach for the interoperability of enterprises' services information exchange," 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Rome, Italy, 2016, pp. 789-799.

[11] Huf, A., & Siqueira, F. (2019). Composition of heterogeneous web services: A systematic review. Journal of Network and Computer Applications, 143, 89-110.

[12] Resende, L., 2007, June. Handling heterogeneous data sources in a SOA environment with service data objects (SDO). In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (pp. 895-897).

[13] R. Tan, R. Chirkova, V. Gadepally and T. G. Mattson, "Enabling query processing across heterogeneous data models: A survey," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 2017, pp. 3211-3220, doi: 10.1109/BigData.2017.8258302.

[14] S. K. Sowe, T. Kimata, M. Dong and K. Zettsu, "Managing Heterogeneous Sensor Data on a Big Data Platform: IoT Services for Data-Intensive Science," 2014 IEEE 38th International Computer

Software and Applications Conference Workshops, Vasteras, Sweden, 2014, pp. 295-300, doi: 10.1109/COMPSACW.2014.52.

[15] Kong, Z., Wang, D., & Zhang, J. (2008). A strategic framework for enterprise information integration of ERP and e-commerce. In Research and Practical Issues of Enterprise Information Systems II: IFIP TC 8 WG 8.9 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2007) October 14–16, 2007, Beijing, China (pp. 701-705). Springer US.

[16] Liu, Z. F., Lu, Z. Y., & Zhang, C. L. (2010, May). Application Research on Manufacter E-commerce and ERP Integration. In 2010 International Conference on E-Business and E-Government (pp. 3204-3207). IEEE.

[17] Li, G., Qian, X., & Ye, C. (2009, October). Integration model of Cooperation E-commerce based on Web Services. In International Technology and Innovation Conference 2009 (ITIC 2009) (pp. 1-4). IET.

[18] Vaithilingam, Priyan, Tianyi Zhang, and Elena L. Glassman. "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models." In *Chi conference on human factors in computing systems extended abstracts*, pp. 1-7. 2022.

[19] Imai, Saki. "Is github copilot a substitute for human pair-programming? an empirical study." In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pp. 319-321. 2022.

[20] Nguyen, Nhan, and Sarah Nadi. "An empirical evaluation of GitHub copilot's code suggestions." In Proceedings of the 19th International Conference on Mining Software Repositories, pp. 1-5. 2022.

[21] P. Runeson, M. Host, A. Rainer, and B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, 2012

[22] Brooke, J.: Sus: a “quick and dirty’usability. Usability evaluation in industry 189(3), 189–194 (1996)