Architecture of multi-agent systems for generative automatic matching among heterogeneous systems

Zouhair Ibn Batouta[1], Rachid Dehbi[2], Mohamed Talea[2]
[1]LTI Laboratory, Faculty of Science Ben M'Sik, Hassan II University, Casablanca, Morocco
[2]LR2I Laboratory, Faculty of Science Aïn Chock, Hassan II University, Casablanca, Morocco

Article Info

Article history:

Received Jun 25, 2024
Revised Nov 23, 2024
Accepted Dec 2, 2024

Keywords:

Generative automatic matching
Heterogeneous systems
Mathematical metrics
Model-driven engineering
Multi-agent system

ABSTRACT

This paper presents the generative automatic matching (GAM) approach, implemented through a multi-agent system (MAS), to address the challenges of heterogeneity across meta-models. GAM integrates automatic meta-model matching with model generation, offering a comprehensive solution to complex systems involving diverse architectures. The key innovation lies in its ability to automate both the detection of correspondences and the transformation of models, improving the precision and recall of matching processes. The system's scalability and adaptability are enhanced by MAS, allowing for efficient management of diverse meta-models. The approach was evaluated through relational to big data UML meta-models (RBDU) case study. The results demonstrated high accuracy, with precision and recall metrics approaching 1, underscoring the robustness of GAM in managing heterogeneous systems. Compared to traditional methods, GAM offers significant advantages, including automated matching and generation, adaptability to various domains, and superior performance metrics. The study contributes to the field of model-driven engineering (MDE) by formalizing a method that effectively bridges the gap between heterogeneous meta-models. Future research will focus on refining matching heuristics, expanding case studies.

Corresponding Author:

Zouhair Ibn Batouta
LTI Laboratory, Faculty of Science Ben M'Sik, Hassan II University
Av Driss El Harti Sidi Othmane, Casablanca, 20700, Morocco
Email: zouhair.ibnbatouta@gmail.com

1. INTRODUCTION

Model-driven engineering (MDE) is a software development approach that uses high-level models as core elements for both design and implementation, rather than just documentation. MDE promotes automation in various stages of development by leveraging models for generative purposes. However, the approach has led to diverse systems based on heterogeneous meta-models, lacking a universal standard. Examples include relational databases versus NoSQL systems and meta-models for similar domains like C# and Java [1]. A key issue identified in our previous systematic mapping review (TSMR) and multi-criteria analysis [2], [3] is the challenge of interoperability across systems with different meta-models, even when they share similar objectives. For example, migrating from SQL-based relational databases to NoSQL systems is complex, as is transitioning between different NoSQL systems (*e.g.*, key-value store to document store). Similar difficulties arise in application development, where the growing diversity of programming languages and architectures, such as UML and MERISE, makes manual transitions between them cumbersome. The heterogeneity of meta-models and architectures also complicates system transformations in the domain of code generation. Our previous findings [3] revealed that 49% to 85% of the code generation studies reviewed developed their own platforms by creating new domain-specific languages (DSLs). To address the increasing diversity of architectures and meta-models in similar or distinct domains, existing methods-such as static identifier-based techniques (SIB) [4]–[6], signature-based techniques (SIG) [7]–[9], similarity-based techniques (SIM) [10]–[15], and custom-specific language techniques (CSL) [16]–[20]-have aimed to match and establish correspondences between different architectural elements and meta-models of heterogeneous systems. These techniques seek to facilitate model transformation across platforms, technologies, or methodologies. However, they face notable limitations, including reliance on manual matching processes and the inability to automatically generate models across diverse meta-models.

To address the limitations of existing methods, we introduce the generative automatic matching (GAM) approach, a novel methodology that integrates automatic meta-model matching with model generation. GAM offers a comprehensive solution to the challenges posed by heterogeneous systems by automatically detecting and matching correspondences between source and target meta-models. This process begins with the identification and schematization of both systems, ensuring that all key aspects of their meta-models are captured. The core innovation of GAM lies in its ability to automate the detection of similarities between meta-model elements, enabling the seamless transformation of models from the source system to the target system. This automation significantly reduces the complexity and manual effort typically involved in managing heterogeneity across platforms, technologies, and methodologies. In contrast to previous approaches, which often rely on manual matching or address only the model layer, GAM uniquely combines automatic meta-model matching and model generation. Its versatility allows it to work with a wide range of models, regardless of the underlying technology, making GAM a scalable and adaptable solution for heterogeneous environments. This approach represents a significant advancement in managing the complexity and diversity of meta-models.

This paper advances the GAM approach by focusing on two key developments: its implementation using a multi-agent system (MAS) based on foundation for intelligent physical agents (FIPA) standards for intelligent agents [21], and its application in relational to big data UML meta-models (RBDU) case study to demonstrate its robustness in automatic matching. The paper is structured as follows: first, a review of existing approaches to address heterogeneity between systems in various domains is provided, highlighting their strengths and limitations. Next, the methodology of the GAM approach is detailed, including its architecture, mathematical formalism, and multi-agent system structure, alongside communication protocols. The RBDU case study, is then presented to illustrate the system's effectiveness. Finally, the results and discussion section evaluate the approach and outlines future research directions.

2. RELATED WORK

Several existing approaches have attempted to address architectural heterogeneity, but they have notable limitations. Most rely on manual or semi-automated matching processes and use fixed, non-adaptive algorithms. Key techniques, such as SIB [4]–[6], SIG [7]–[9], SIM [10]–[15], and CSL [16]–[20], each have specific drawbacks, including failure with heterogeneous models, limited scalability, and reliance on manual intervention. To better understand these limitations, a strengths, weaknesses, opportunities, and threats (SWOT) analysis in Table 1 summarizes the strengths and weaknesses of these methods, laying the groundwork for the more adaptive and comprehensive GAM approach.

Table 1. SWOT analysis of matching approaches

	Static identifier-based technique (SIB)
Characteristics	Uses unique identifiers (UUIDs) to establish correspondences between model elements. Fast and requires no user configuration, but struggles with heterogeneous models.
Positives	Quick implementation, no user setup.
Negatives	Not suitable for heterogeneous models, poor adaptability, no automatic generation, manual correspondence.
	Signature-based techniques (SIG)
Characteristics	Compares independent models by calculating signatures (or fingerprints) for model elements. Requires user input to define signature functions, limiting its scope.
Positives	Compares independently built models.
Negatives	Requires user-defined identity functions, limited scope, no automatic generation, and manual correspondences.
	Similarity-based technique (SIM)
Characteristics	Uses heuristics to evaluate the similarity between independent model elements. More flexible but relies on fixed heuristics, limiting adaptability.
Positives	Accurate correspondences.
Negatives	Fixed heuristics, no automatic generation, manual correspondence.
	Custom-specific language technique (CSL)
Characteristics	Uses domain-specific languages to integrate semantics into matching algorithms. Flexible for domain-specific models but requires manual specification of algorithms.
Positives	Integrates semantics.
Negatives	Manual algorithm specification, fixed heuristics

As emphasized in the SWOT analysis in Table 1, all existing approaches either require manual matching or lack the capability for automatic model generation. For instance, SIB is fast but unsuitable for heterogeneous models and does not support automatic generation. SIG compares independent models but requires user-defined functions and similarly lacks automatic generation. While SIM is accurate, it relies on fixed heuristics and does not support automatic generation. Lastly, CSL integrates semantics but requires manual specifications and depends on fixed heuristics. To address these limitations, we present a new approach, GAM, which will be detailed in the next section.

3. METHODOLOGY USED

In this section, we present the methodology employed in the design and implementation of our new approach, GAM. This methodology is structured around the following key components: the GAM architecture, which defines the overall structure of the approach; the generative matching meta-model, serving as the foundational framework for matching heterogeneous systems; and the GAM process, outlining the sequential steps for applying the approach. Additionally, the methodology incorporates mathematical formalism to provide a rigorous theoretical basis and a multi-agent system to ensure scalability and dynamic interaction among components. Finally, a case study is included to demonstrate the practical application and validate the effectiveness of the proposed approach.

3.1. GAM architecture

First, we designed the architecture of our approach. GAM is built on two fundamental steps, as shown in Figure 1:

- Meta-model matching: in this step, heterogeneous meta-models are automatically linked. Source meta-models (SMM 1 ... SMM i) are matched with target meta-models (TMM 1 ... TMM j), generating a matching model (MG) that identifies the correspondences between elements.
- Model generation: based on the matching established in the first step, the source models (SM 1 ... SM i) are automatically transformed into target models (TM 1 ... TM j), conforming to the corresponding target meta-models.

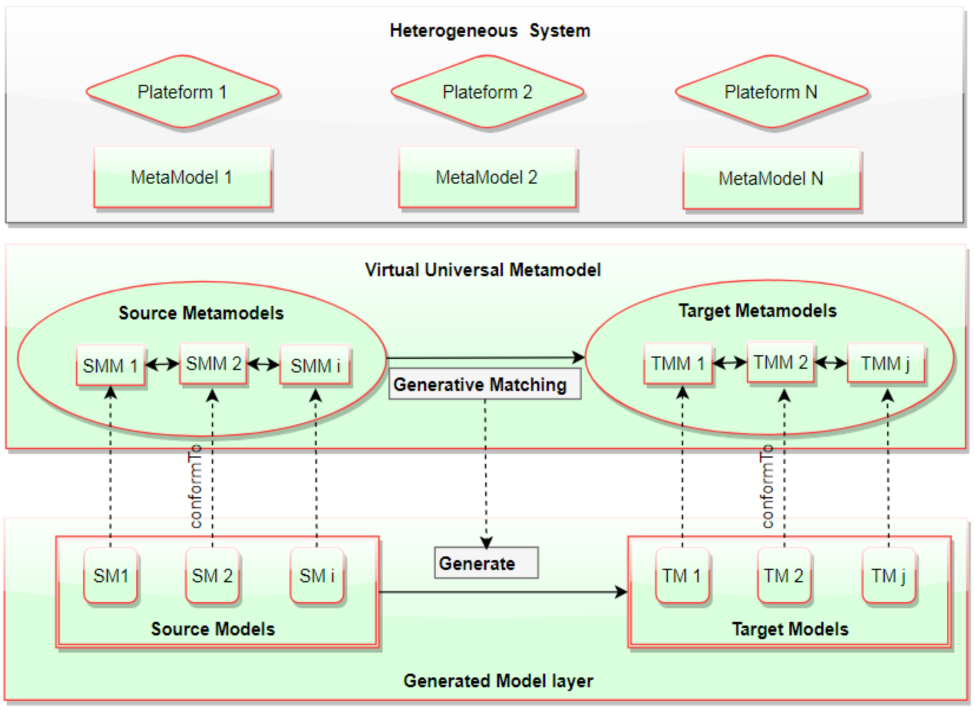


Figure 1. Comprehensive structure of the generative matching approach

3.2. Generative matching meta-model (MMG)

To implement the GAM approach effectively, we designed a generative matching meta-model that identifies key concepts, including elements, relationships, version management, and matching history. This meta-model handles the identification of correspondences between meta-model elements, ensuring consistency and traceability throughout the matching process. Additionally, it provides a structured representation to address the dynamic nature of system changes, allowing for iterative updates and refinements. The design incorporates mechanisms for conflict resolution and supports multiple versions to accommodate evolving requirements. By leveraging this meta-model, the GAM approach achieves a robust and scalable framework for managing complex system heterogeneities.

3.3. GAM process

The GAM process comprises two primary phases: automatic matching and automatic generation. In the matching phase, two meta-models (source and target) are used to automatically generate a correspondence model (MG), which defines the relationships between their elements. In the generation phase, a source model conforming to the source meta-model is automatically transformed into an equivalent target model, utilizing the identified correspondences. The detailed process includes these four key steps, as shown in Figure 2:

- Selecting the source and target meta-models.
- Refining the core generative MMG by adding or modifying relationships and storing refined versions in a cloud repository for easy access.
- Refining the MG model through iterative or manual adjustments using cognitive agents or expert input.
- Generating the target model, with the possibility of further refinement through expert validation or additional iterations.

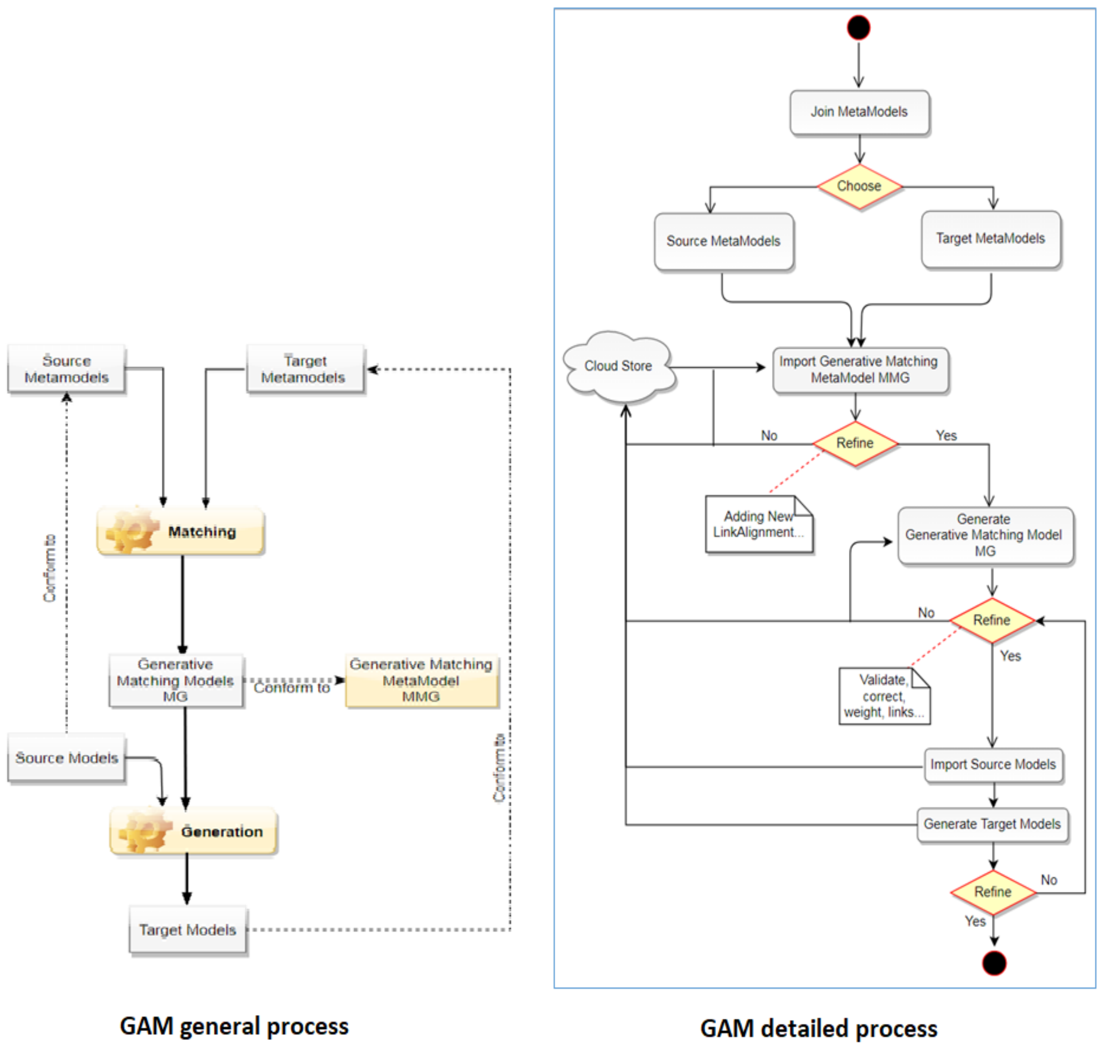


Figure 2. GAM process description

3.4. Mathematical formalism

We developed a mathematical formalism for the GAM MAS approach, grounded in set theory, where each meta-model (MMa) is represented as a set of triplets. These triplets comprise elements from the refined generative matching meta-model (MMG) and the relationships between them. For two meta-models, MMa (source) and MMb (target), the matching model (MG) captures the correspondences between their elements, also represented as triplets. The transformation process between source and target models leverages these correspondences to generate equivalent models, ensuring they conform to their respective meta-models. This set-based formalism provides a structured and rigorous representation of models, correspondences, and transformations.

3.5. Multi-agent system

The concept of multi-agent system (MAS) stems from distributed artificial intelligence (DAI). This approach facilitates the understanding, modeling, and simulation of complex systems composed of multiple agents that exhibit intelligent behavior and interact with both each other and their external environment. MAS is particularly suited for solving problems in a distributed manner [22]–[28]. Each agent operates locally with cooperative behaviors, and through collective self-organization, a global solution emerges from the individual problem-solving efforts of the agents.

3.6. Case study

To evaluate our approach, we conducted the RBDU case study, this case study involved a more complex heterogeneous database system incorporating the UML meta-model. For this, we developed five meta-models representing various database types, including relational databases and three big data NoSQL types (key-value store, document store, and columnar store), in addition to the UML meta-model. These meta-models illustrate the effectiveness of the GAM SMA approach in facilitating model transformation across diverse database systems.

4. RESULTS AND DISCUSSION

4.1. Generative matching meta-model

The generative matching meta-model (MMG) we designed addresses the management of lexical, structural, and semantic similarities to effectively match heterogeneous meta-models. Its key components enable the alignment and transformation of elements across meta-models by employing various similarity measures. This structure significantly improves the overall efficiency of the GAM SMA approach in automating model generation. The core of the MMG consists of several essential components, as illustrated in Figure 3.

The effectiveness of our generative matching meta-model lies in its extensibility, which allows for flexible adaptation to various domains and architectures. At its core, MMG consists of key components that manage correspondences between heterogeneous meta-models, ensuring that all essential elements are addressed during the matching process. The *Element* component generalizes other elements with attributes such as name, ID, and description, while the *Matching* component efficiently manages relations between source and target meta-models, incorporating version control and refinement. *Source* and *Target* define the respective meta-models, and the *AgentMetamodelHandler* and *AgentElementHandler* facilitate the navigation and manipulation of meta-elements. Additionally, the *AgentTransformer* ensures the proper transformation of source elements into target elements, while *LinkAlignment* defines relationships like aggregation and similarity, which are essential for effectively aligning meta-elements. The Similarity component is crucial, as it captures various types of correspondences—lexical, structural, semantic, and functional—forming the foundation of the GAM process and enabling precise automatic model generation across diverse systems.

4.2. Multi-agent system

By assigning specialized roles to different agents, the system efficiently handles various tasks. Figure 4 illustrates the Contract Net protocol, which outlines the agent societies and their communication. The network comprises specialized agents, each responsible for distinct functions within the GAM MAS approach. The *CoordinatorAgent* oversees coordination among agents, while the *GeneratorAgent* handles model generation based on matching results. The *RefiningAgent* refines the correspondences identified during the process, and the *TransformerAgent*, along with the *MFTransformerAgent* and *FMTransformerAgent*, converts meta-models into mathematical formalism for adaptability. Agents such as the *MatchingAgent*, *MeaningSimilarityAgent*, *TranslationAgent*, and *StructuredSimilarityAgent* calculate various types of similarities, including semantic, structural, and functional. Additionally, agents like the *BasicSimilarityAgent*, *NameSimilarityAgent*, and *DescriptionSimilarityAgent* focus on lexical similarities, while the *FunctionalSimilarityAgent* evaluates functional relationships. This integrated agent society efficiently manages meta-model heterogeneity, enabling automatic model generation and significantly enhancing the overall effectiveness of the GAM MAS approach.

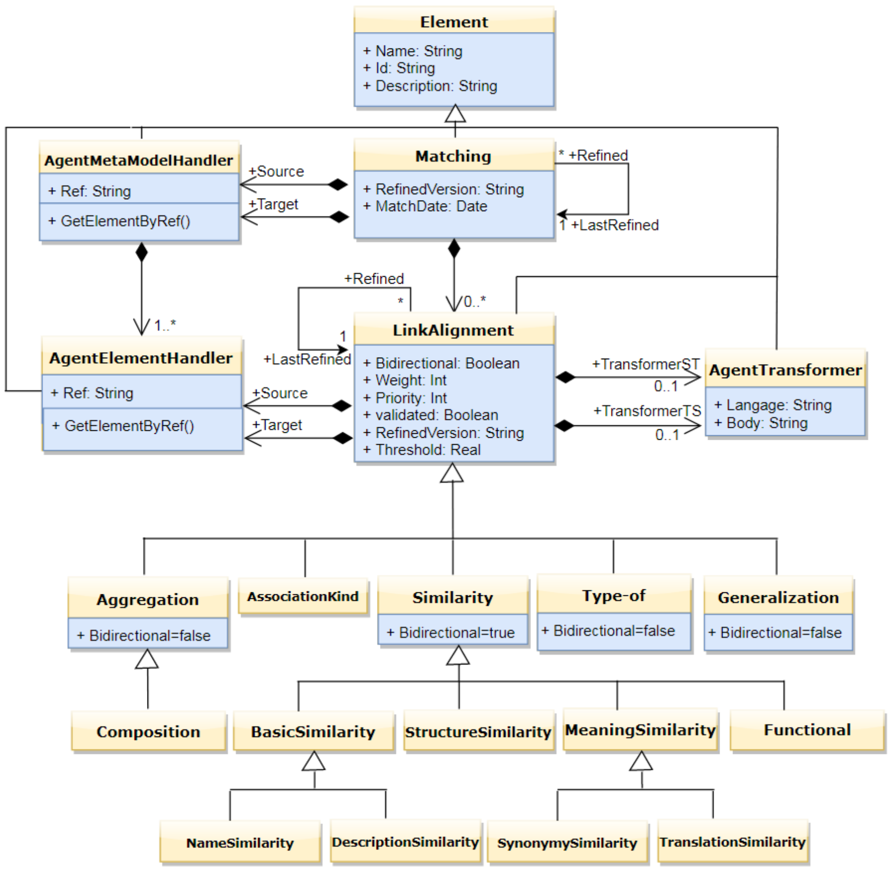


Figure 3. The core of the GAM approach

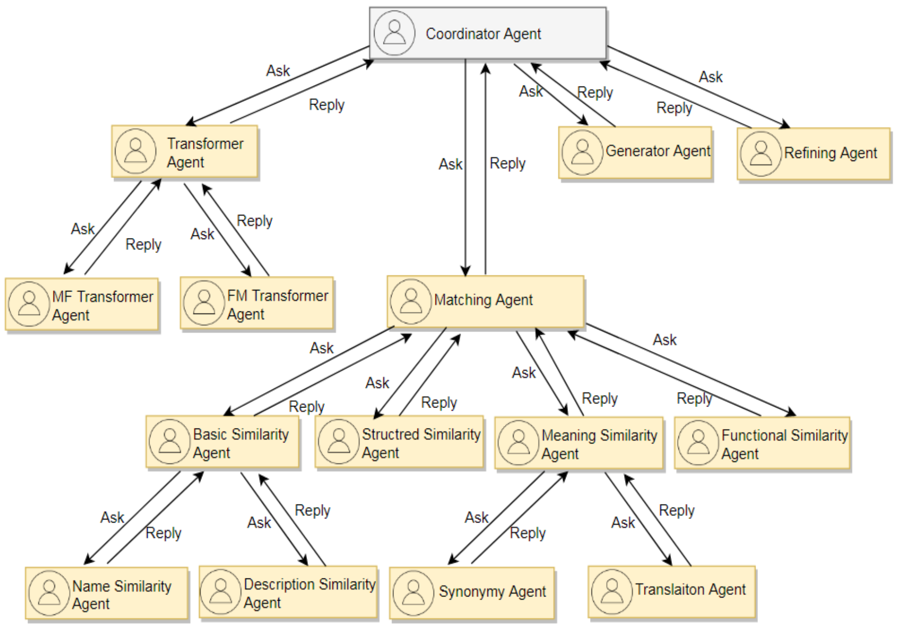


Figure 4. Contract Net protocol (CNP) GAM SMA

4.3. Case study and matching results

In this section, we present the meta-models developed for the case study incorporated in our research, along with the corresponding matching results. The purpose of these case study is to test our approach on different systems and evaluate its effectiveness in enabling matching and model generation between them. The meta-models for the RBDU case study are shown in Figure 5. Through this case study, we aim to assess the efficiency of the GAM MAS approach in managing diverse systems.

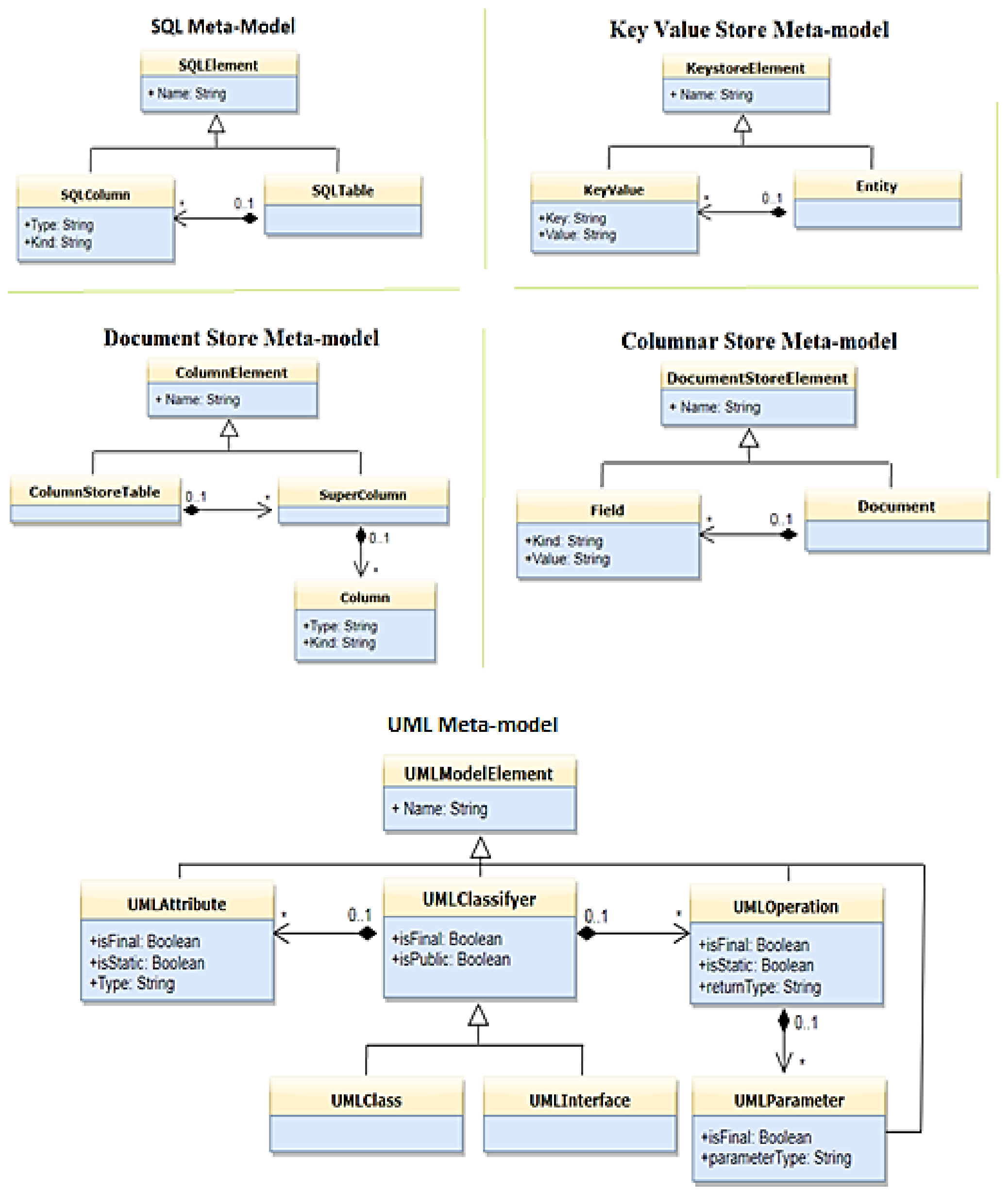
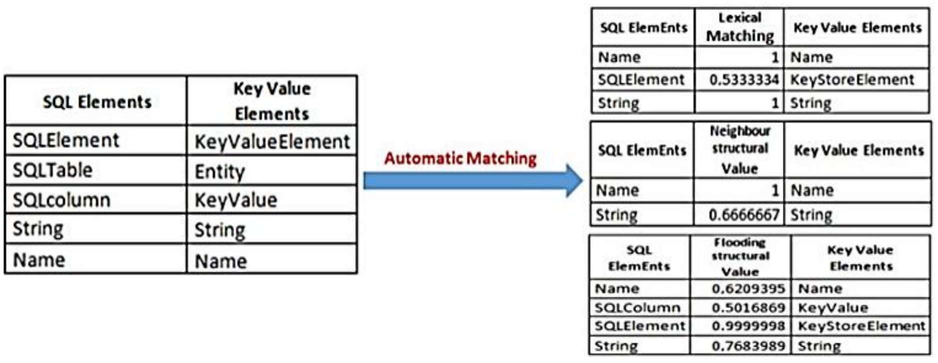


Figure 5. RBDU case studies

The results of applying the GAM SMA approach to the RBDU case study are summarized as follows. We present the matching results obtained from applying our approach to the RBDU case study. Figure 6 illustrates the matches generated between the SQL and key-value store meta-models, while Table 2 provides a detailed summary of the automatic matching results produced by the GAM SMA approach between the SQL source meta-model and the UML target meta-model.

The RBDU case study further demonstrated the strength of the GAM SMA approach in handling complex, heterogeneous database systems. Five meta-models, representing relational and big data NoSQL databases (key-value store, document store, and columnar store), were matched and transformed with high accuracy. The successful matches in the case study highlight the robustness of the GAM SMA approach across various domains, with matching precision and recall values approaching 1. In the following section, we will provide a detailed analysis of the evaluation results of the GAM SMA approach, including a synthesis of the final quality metric calculations for the RBDU case study.



SQL Elements	Key Value Elements
SQLElement	KeyValueElement
SQLTable	Entity
SQLcolumn	KeyValue
String	String
Name	Name

SQL ElemEnts	Lexical Matching	Key Value Elements
Name	1	Name
SQLElement	0.5333334	KeyStoreElement
String	1	String

SQL ElemEnts	Neighbour structural Value	Key Value Elements
Name	1	Name
String	0.6666667	String

SQL ElemEnts	Flooding structural Value	Key Value Elements
Name	0.6209395	Name
SQLColumn	0.5016869	KeyValue
SQLElement	0.9999998	KeyStoreElement
String	0.7683989	String

Figure 6. Matching between SQL and key-value store- threshold: 0.5

Table 2. Summary of the matching results between SQL and UML meta-models

		Name Similarity	Neighbor Structural	Flooding Structural	Max	Interpolation Moyenne
SQLElement	UMLModelElement	0.5333334	0.25	1	1	0.59444447
SQLTable	UMLClassifyer	0.2307692	0.2	0.1205332	0.2307692	0.18376747
SQLTable	UMLCLASS	0.125	0	0.007990999	0.125	0.04433033
SQLcolumn	UMLAttribute	0.1666667	0.4444444	0.1895308	0.4444444	0.26688063
Type	Type	1	0.5	0.0451359	1	0.5150453
Type	Returntype	0.4	0.5	0.04240563	0.5	0.31413521
Type	Parametertype	0.3076923	0.5	0.02927022	0.5	0.27898751
Name	Name	1	1	0.6129308	1	0.87097693
String	String	1	0.5714286	0.6036351	1	0.72502123

4.4. GAM evaluation

The evaluation was conducted using well-established quality metrics commonly applied in machine learning and artificial intelligence, including recall, overall accuracy, F-measure, and precision [29]–[33]. Table 3 presents the results of these metrics after generating correspondences using the GAM MAS approach in the RBDU case study. The quality metrics calculated for both case studies demonstrate the overall effectiveness of the GAM MAS approach. The RBDU case study exhibited strong performance across multiple big data NoSQL meta-models, with precision consistently reaching 1. These high-quality metrics confirm that the GAM SMA approach is capable of handling both simple and complex systems, delivering reliable and accurate results.

Table 3. Quality measurement results for the big data RBDU section

		Measures			
Meta-model couples	Heuristic	Recall	Precision	F-Measure	Overall
(SQL, Key-Value)	NameMatching	0.6	1	0.75	0.6
	Neighbour Structural	0.5	1	0.666666667	0.5
	Flooding Structural	1	1	1	1
(SQL, DocumentStore)	NameMatching	0.8	1	0.88888889	0.8
	Neighbour Structural	0.6	1	0.75	0.6
	Flooding Structural	0.8	1	0.88888889	0.8
(SQL, Columnar)	NameMatching	0.875	1	0.93333333	0.875
	Neighbour Structural	1	0.8	0.888888889	0.75
	Flooding Structural	0.375	1	0.545454545	0.375
FINAL Values	NameMatching	0.85714286	1	0.92307692	0.85714286
	Neighbour Structural	0.764705882	0.866666667	0.8125	0.647058824
	Flooding Structural	0.647058824	1	0.785714286	0.647058824

The metric results obtained after generating matches using the GAM SMA approach for the big data meta-models in the RBDU case study. We opted to separate the evaluation results for big data meta-models from those of the SQL/UML pair, shown in Table 4, to analyze the impact of meta-model domain similarity on the generation results. The final metrics for the RBDU case study are summarized in Table 5, where the weighted sum method was applied, assigning a weight of 1 to each pair to calculate the overall results.

Table 5 display the final metric results, reflecting the average values calculated by the corresponding functions. It is noteworthy that all metrics are close to 1, underscoring the high quality and accuracy of the results obtained. Our evaluation demonstrates that the GAM SMA approach significantly outperforms existing methods such as SIB, SIG, SIM, and CLS, as highlighted in the SWOT analysis in Table 1. Unlike these methods, which either lack automatic model generation or rely on fixed heuristics, GAM MAS integrates both matching and generation processes, making it highly adaptable to a wide range of systems and architectures. The use of a MAS enhances scalability and flexibility, allowing for efficient management of diverse meta-models. The case study confirmed GAM MAS's effectiveness in addressing the challenges of automatic matching and model generation between heterogeneous meta-models. All quality metrics approached values close to 1, highlighting the high precision and reliability of the correspondences and transformations achieved.

Table 4. Quality measurement results for SQL/UML pair

		Measures			
Meta-model couples	Heuristic	Recall	Precision	F-measure	Overall
(SQL, UML)	Maximum Similarity	0.666666667	1	0.8	0.666666667

Table 5. Final quality metrics GAM SMA

		Measures			
Case study	Function	Recall	Precision	F-Measure	Overall
RBDU	Final Similarity	0.80952381175	1	0.89230769	0.80952381175

4.5. Limitations

The GAM MAS approach provides a robust solution by integrating automatic meta-model matching with model generation, effectively addressing the complexities of heterogeneous systems and technologies. This significantly improves precision and recall metrics, with successful implementation demonstrating important implications for managing heterogeneity across different development systems and architectures. It enhances the efficiency and accuracy of creating interoperable systems, especially in complex environments with diverse systems and meta-models. The high quality of results, reflected in metrics nearing 1, highlights the reliability and effectiveness of the approach in producing accurate correspondences. However, several areas for future research remain. First, improving the automatic matching heuristics is a key priority. For instance, weight and threshold calculations could be refined through advanced techniques, such as the Rock method for weight determination or fuzzy logic. Integrating new heuristics and testing them could further optimize the matching accuracy. Another critical direction is applying the generative automatic matching approach to address data layer interoperability challenges [34], particularly between big data NoSQL and relational databases. The versatility and adaptability of GAM MAS offer numerous research opportunities across a wide range of domains. Future work could extend its application to areas such as IT governance, e-learning, e-healthcare, IoT, search engines, and the Semantic Web. These expansions would not only enrich the knowledge base but also enhance the capabilities of the agents, broadening the scope of the research. Additionally, we plan to develop a comprehensive GUI application on the .NET platform to facilitate broader adoption and practical use of GAM SMA, ensuring it remains a flexible and effective tool for tackling future challenges in automatic matching and system interoperability.

5. CONCLUSION

This paper presented the implementation of the GAM approach using a MAS architecture. GAM MAS represents a novel paradigm that combines automatic matching and model generation to address the heterogeneity of meta-models. Our approach facilitates the generation of models through automatic matching between various heterogeneous systems.

The evaluation using quality metrics and case studies demonstrated the validity and effectiveness of our approach. Specifically, the RBDU case study highlighted the robustness and adaptability of GAM SMA. This work fills a significant knowledge gap by providing a formalized method for meta-model matching and generation, contributing to the field of model-driven engineering. Compared to existing methods, GAM SMA offers several advantages:

The GAM approach introduces significant innovations compared to existing methods. Unlike traditional approaches that often rely on manual matching and lack mechanisms for automatic model generation, GAM seamlessly integrates both processes, ensuring greater efficiency and consistency. Furthermore, GAM demonstrates exceptional adaptability by handling a wide range of meta-models and technologies, effectively addressing the limitations of methods such as static identifier-based technique (SIB), SIG, SIM, and CSL technique, as highlighted in the SWOT analysis. In addition, GAM enhances key performance metrics, offering improved precision and recall, which underscores its potential to transform software development practices by delivering more accurate and reliable results.

Future research will focus on enhancing the heuristics for automatic matching and integrating additional case studies to further validate the approach. We also plan to develop a comprehensive GUI application using the .NET platform to facilitate broader adoption and practical application of GAM SMA, which will further enhance automatic matching and model generation in various domains such as AI, IT governance, E-learning, E-healthcare, internet of things (IoT), search engines, chatbots, and the Semantic Web. Another significant area for future research lies in applying generative automatic matching to address data layer interoperability and migration issues. Our team is currently working in this field, aiming to overcome interoperability problems between big data NoSQL and relational databases. GAM MAS can help automate system matching structure generation and explore data layer transformation.

REFERENCES

[1] Z. I. Batouta, R. Dehbi, M. Talea, and H. Omar, “Generative matching between heterogeneous meta-model’ systems based on hybrid heuristic,” *Journal of Information Technology Research*, vol. 12, no. 2, pp. 53–71, 2019, doi: 10.4018/JITR.2019040104.

[2] Z. I. Batouta, R. Dehbi, M. Talea, and O. Hajoui, “Multi-criteria analysis and advanced comparative study between automatic generation approaches in software engineering,” *Journal of Theoretical and Applied Information Technology*, vol. 81, no. 3, pp. 609–620, 2015.

[3] Z. I. Batouta, R. Dehbi, M. Talea, and O. Hajoui, “Automation in code generation: Tertiary and systematic mapping review,” *Colloquium in Information Science and Technology, CIST*, vol. 0, pp. 200–205, 2016, doi: 10.1109/CIST.2016.7805042.

[4] X. Yaozong, S. Xuebin, Z. Shuhua, Z. Qiujun, and J. Weinan, “Static Analysis Method of C Code Based on Model Checking and Defect Pattern Matching,” in *2023 IEEE 5th International Conference on Power, Intelligent Computing and Systems, ICPICS 2023*, 2023, pp. 567–573. doi: 10.1109/ICPICS58376.2023.10235566.

[5] F. A. Somogyi and M. Asztalos, “Systematic review of matching techniques used in model-driven methodologies,” *Software and Systems Modeling*, vol. 19, no. 3, pp. 693–720, 2020, doi: 10.1007/s10270-019-00760-x.

[6] J. Ren *et al.*, “Matching algorithms: fundamentals, applications and challenges,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 3, pp. 332–350, 2021, doi: 10.1109/TETCI.2021.3067655.

[7] T. Lyons and A. D. McLeod, “Signature methods in machine learning,” *arXiv preprint arXiv:2206.14674, 2022*, 2022.

[8] C. Cuchiero, G. Gazzani, and S. Svaluto-Ferro, “Signature-based models: theory and calibration,” *SIAM Journal on Financial Mathematics*, vol. 14, no. 3, pp. 910–957, 2023, doi: 10.1137/22M1512338.

[9] M. T. Shafiq and S. R. Lockley, “Application of signature-based matching for IFC model comparison,” *International Journal of Construction Management*, vol. 22, no. 9, pp. 1765–1774, 2022, doi: 10.1080/15623599.2020.1742630.

[10] P. Yang, H. Wang, J. Yang, Z. Qian, Y. Zhang, and X. Lin, “Deep learning approaches for similarity computation: a survey,” *IEEE Transactions on Knowledge and Data Engineering*, 2024, doi: 10.1109/TKDE.2024.3422484.

[11] Z. Pan, G. Pan, and A. Monti, “Semantic-similarity-based schema matching for management of building energy data,” *Energies*, vol. 15, no. 23, 2022, doi: 10.3390/en15238894.

[12] M. Auch, M. Weber, P. Mandl, and C. Wolff, “Similarity-based analyses on software applications: a systematic literature review,” *Journal of Systems and Software*, vol. 168, 2020, doi: 10.1016/j.jss.2020.110669.

[13] D. K. Po, “Similarity based information retrieval using Levenshtein distance algorithm,” *International Journal of Advances in Scientific Research and Engineering*, vol. 06, no. 04, pp. 6–10, 2020, doi: 10.31695/ijasre.2020.33780.

[14] Z. Yuan, L. Yan, and Z. Ma, “Structural similarity measure between UML class diagrams based on UCG,” *Requirements Engineering*, vol. 25, no. 2, pp. 213–229, 2020, doi: 10.1007/s00766-019-00317-w.

[15] S. Melnik, H. Garcia-Molina, and E. Rahm, “Similarity flooding: A versatile graph matching algorithm and its application to schema matching,” in *Proceedings - International Conference on Data Engineering*, 2002, pp. 117–128. doi: 10.1109/ICDE.2002.994702.

[16] F. Wrede, C. Rieger, and H. Kuchen, “Generation of high-performance code based on a domain-specific language for algorithmic skeletons,” *Journal of Supercomputing*, vol. 76, no. 7, pp. 5098–5116, 2020, doi: 10.1007/s11227-019-02825-6.

[17] L. N. Lyadova, A. O. Sukhov, and M. R. Nureev, “An Ontology-Based Approach to the Domain Specific Languages Design,” in *15th IEEE International Conference on Application of Information and Communication Technologies, AICT 2021*, 2021. doi: 10.1109/AICT52784.2021.9620493.

[18] G. Czech, M. Moser, and J. Pichler, “A systematic mapping study on best practices for domain-specific modeling,” *Software Quality Journal*, vol. 28, no. 2, pp. 663–692, 2020, doi: 10.1007/s11219-019-09466-1.

[19] K. Panayiotou, C. Doumanidis, E. Tsardoulias, and A. L. Symeonidis, “SmAuto: a domain-specific-language for application development in smart environments,” *Pervasive and Mobile Computing*, vol. 101, 2024, doi: 10.1016/j.pmcj.2024.101931.

[20] G. K. Halley, L. Vanfretti, and M. De Castro, “Interactive model transformations from the common information model (CIM) to modelica,” in *2024 9th International Conference on Smart and Sustainable Technologies, SpliTech 2024*, 2024, pp. 1–5. doi: 10.23919/SpliTech61897.2024.10612559.

[21] FIPA, “FIPA ACL message structure specification,” fipa.org, 2002. Accessed: Oct. 06, 2024). [Online], Available: http://www.fipa.org/specs/fipa00061/SC00061G.pdf

[22] G. Dodig-Crnkovic and M. Burgin, "A systematic approach to autonomous agents," *Philosophies*, vol. 9, no. 2, 2024, doi: 10.3390/philosophies9020044.

[23] A. Amirkhani and A. H. Barshooi, "Consensus in multi-agent systems: a review," *Artificial Intelligence Review*, vol. 55, no. 5, pp. 3897–3935, 2022, doi: 10.1007/s10462-021-10097-x.

[24] R. Calegari, G. Ciatto, V. Mascardi, and A. Omicini, "Logic-based technologies for multi-agent systems: a systematic literature review," *Autonomous Agents and Multi-Agent Systems*, vol. 35, no. 1, 2021, doi: 10.1007/s10458-020-09478-3.

[25] D. Calvaresi, Y. Dicente Cid, M. Marinoni, A. F. Dragoni, A. Najjar, and M. Schumacher, "Real-time multi-agent systems: rationality, formal model, and empirical results," *Autonomous Agents and Multi-Agent Systems*, vol. 35, no. 1, 2021, doi: 10.1007/s10458-020-09492-5.

[26] R. C. Cardoso and A. Ferrando, "A review of agent-based programming for multi-agent systems," *Computers*, vol. 10, no. 2, pp. 1–15, 2021, doi: 10.3390/computers10020016.

[27] W. Du and S. Ding, "A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications," *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3215–3238, 2021, doi: 10.1007/s10462-020-09938-y.

[28] Maciej Serda *et al.*, "ESG, Competitive advantage and financial performances: a preliminary research," *Uniwersytet śląski*, vol. 7, no. 1, pp. 969–986, 2020.

[29] A. K. Chopra, S. H. Christie, and M. P. Singh, "An evaluation of communication protocol languages for engineering multiagent systems," *Journal of Artificial Intelligence Research*, vol. 69, pp. 1351–1393, 2020, doi: 10.1613/JAIR.1.12212.

[30] A. Arabiat and M. Altayeb, "Enhancing internet of things security: evaluating machine learning classifiers for attack prediction," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 5, pp. 6036–6046, 2024, doi: 10.11591/ijece.v14i5.pp6036-6046.

[31] Bharti, N. S. Gill, and P. Gulia, "Exploring machine learning techniques for fake profile detection in online social networks," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 3, pp. 2962–2971, 2023, doi: 10.11591/ijece.v13i3.pp2962-2971.

[32] W. Abdullah and A. Salah, "A novel hybrid deep learning model for price prediction," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 3, pp. 3420–3431, Jun. 2023, doi: 10.11591/ijece.v13i3.pp3420-3431.

[33] I. Slimani *et al.*, "Automated machine learning: the new data science challenge," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 4, pp. 4243–4252, 2022, doi: 10.11591/ijece.v12i4.pp4243-4252.

[34] A. Erraji, A. Maizate, M. Ouzzif, and Z. Ibn Batouta, "Migrating data semantic from relational database system to NOSQL systems to improve data quality for big data analytics system," *ECS Transactions*, vol. 107, no. 1, pp. 19495–19503, 2022, doi: 10.1149/10701.19495ecst.

BIOGRAPHIES OF AUTHORS

Zouhair Ibn Batouta is a computer science professor and a researcher at the LTI Laboratory, Hassan II University, Morocco. His research focuses on the fields of computer science and software engineering. He can be contacted at email: zouhair.ibnbatouta@gmail.com.

Rachid Dehbi is a professor and researcher at the LR2I Laboratory, Hassan II University, Morocco. His expertise includes various areas of computer science and software engineering. He can be contacted at email: dehbirac@yahoo.fr.

Mohamed Talea is a professor and the head of the LTI Laboratory at Hassan II University, Morocco. He has contributed significantly to the fields of computer science and software engineering. He can be contacted at email: taleamohamed@yahoo.fr.