

Towards Interoperability of Systems of Systems Using GraphQL

Eduardo Dantas Luna¹^a, Vitor Pinheiro de Almeida¹^b and Eduardo Thadeu Corseuil²^c

¹*Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil*

²*Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil*
{*eduardoluna, valmeida, thadeu*}@tecgraf.puc-rio.br

Keywords: Digital Twins, System of Systems, Knowledge Graph, API Management, GraphQL.

Abstract: The growing interconnectedness of devices and systems presents a significant opportunity to develop solutions that leverage data from diverse sources. However, integrating data from these heterogeneous systems, which may use different protocols and paradigms, poses a considerable challenge. This paper proposes an innovative solution to address this challenge by introducing an algorithm that generates a GraphQL API management layer. This layer acts as a bridge between disparate systems, enabling seamless data integration and exchange. By leveraging GraphQL's efficient data retrieval capabilities and a knowledge graph to define relationships between data elements, the algorithm automates the creation of a processing layer that simplifies the integration process. The proposed solution offers a promising approach to overcome the complexities of data integration, paving the way for more robust and adaptable data-driven applications.

1 INTRODUCTION

In recent years, the advent of the Internet of Things (IoT) and Web technologies has led to an exponential growth of devices connected via the Web. All these devices create and expose vast amounts of data, which has tremendous potential for developing solutions that require data from distinct sources.

For example, by joining data from different transport data services (such as taxi, bus, metro, boat, etc.), a system could precisely calculate which transport or combinations of transports to arrive at the desired destination with minimal time and money cost. However, connecting data from those sources can be a challenging task.

1.1 The Problems

One of the challenges is how to combine data from multiple systems since systems can describe the same data in different ways or even partially describe the data. An example of such a situation would be two companies that sell clothes, but in one, the money is described in dollars and another in euros; without an explicit description of which currency, it is impossible


to combine this information effectively.


Another concern is combining systems using different data transfer protocols and paradigms. There are different use cases for each paradigm. Therefore, it can be challenging to create a communication between such systems. An example of this involves two paradigms: a movie streaming service and a movie catalogue service. A streaming service needs to send a continuous data flux. In contrast, a catalogue service receives a request and sends a response. If poorly implemented, the streaming service can create more requests than the catalogue can handle.


Additionally to all these issues is the need to develop those data exchange interfaces. To implement the connections between systems, developers from particular systems must learn to deal with different programming languages and technologies to understand how to connect systems. Additionally, if created without standardization, different developers could develop similar interfaces, creating unnecessary rework. Also, these interfaces may be composed of more than just two systems, increasing the difficulty and the time cost to develop those services.

1.2 Objective

This paper addresses the problem of combining data from systems that may use different protocols and paradigms. To achieve this, we will systematically

^a  <https://orcid.org/0009-0002-2663-0732>

^b  <https://orcid.org/0000-0002-6544-9541>

^c  <https://orcid.org/0000-0002-7543-4140>

search the most relevant academic papers and propose a solution based on the findings. The proposed solution will focus on generating a communication layer that can effectively integrate data from diverse sources, addressing the challenges of heterogeneity, protocol differences, and the need for efficient data exchange. By leveraging existing research and proposing a novel approach, this paper seeks to contribute to the development of more robust and adaptable data-driven applications.

1.3 Next Sections

The remainder of this paper is structured as follows: Section 2 establishes a theoretical foundation, defining key concepts used throughout the paper. Section 3 presents a systematic analysis of related academic work, providing insights into existing approaches and their limitations. Section 4 introduces a practical use case from the oil and gas industry, highlighting the challenges of data integration in real-world scenarios. Section 5 details the proposed solution, including the conceptual model and the algorithm for generating the API management layer. Finally, Sections 6 and 7 discuss unresolved issues and potential future work to address these challenges and enhance the proposed solution further.

2 THEORETICAL FOUNDATION

An interface is indispensable for retrieving data from a system. These interfaces are predominantly implemented on the web as Web Applications Programming Interfaces (APIs). An API consists of a collection of interfaces, often referred to as endpoints. To manage these APIs, a system must control multiple aspects of the APIs, such as Authorizations and Rate Limiting; this control system is called API Management. ((Bondel et al., 2021))

API Management has two main parts: the API Gateway and the API Portal. The Gateway communicates services and the API, thereby decoupling the client interface. It also intercepts all incoming requests, routing them to the correct service. It includes many features like caching, scaling, load balance, etc. On the other hand, the API Portal serves as a frontend for both API developers and consumer systems developers. It provides documentation and a guide platform for developers. One of its core features is the endpoint catalogue for each service provided through the multiple APIs. ((Bondel et al., 2021))

API Management is often used in a system architecture called microservices. A microservice archi-

itecture uses multiple decoupled systems instead of a monolith system. By using this type of architecture API Management acts as a system that orchestrates these various systems. Since API Management is used to connect multiple systems, it can be used to build systems composed of interaction with other systems. These systems are referred to as Systems of systems (SoS) and are often employed in various scenarios, including Digital Twins. A Digital Twin is a digital representation of a physical system that mirrors real-world behaviours by integrating and presenting updated information across the various technologies that compose the Digital Twin. ((Shi et al., 2016; Anacker et al., 2022; Olsson and Axelsson, 2023))

An API can use several paradigms, including REST, SOAP and gRPC. However, this thesis primarily explores the GraphQL paradigm. GraphQL, which stands for Graph Query Language, is a standard that defines ways to query and handle data. Advantages of GraphQL include: efficient data retrieval since it allows for precise data requests, avoiding over-fetching and under-fetching; It has a single endpoint in contrast to other paradigms; Strongly typed data; It can aggregate data from multiple data sources. (GraphQL Foundation, 2024)

The architecture of GraphQL is built around two core components: the GraphQL Server and the query language. The server is responsible for processing and executing queries using three main elements: types, fields and resolver functions. The types and fields are used to define how the data is structured in the API. The types and fields also defines all possible queries a client can make. Resolver functions are blocks of code responsible for executing the query, each query defined in the schema corresponds to a resolver function. The query language closely mirrors the type definition language, utilizing defined queries combined with several operands to fetch data described by the query. (GraphQL Foundation, 2024)

Similar to how GraphQL is a standard, there exists a standard for REST APIs known as OpenAPI. OpenAPI files serve as descriptor files that REST API libraries extensively utilize to provide detailed information about the API. These files encompass data such as the url of all endpoints, the response codes implemented by each endpoint, the input and output schemas, and whether specific parameters are required, among other details. (Initiative, 2024)

A knowledge graph is a sophisticated data structure frequently employed to model, organize, manage, and analyze heterogeneous and intricate datasets. Owing to its graph-based architecture, it encapsulates a complex abstraction of knowledge on a particular domain and delineates the interrelationships among

various data entities. (Ramonell et al., 2023)

3 RELATED WORK

This paper undertakes a systematic search for the most relevant academic papers to be reviewed. This systematic search is illustrated by Figure 1. The search process was conducted using the Findpapers library (Grosman, 2024), which allows users to create queries based on specific keywords to retrieve academic papers. The data sources utilized by this library include ACM, arXiv, bioRxiv, IEEE, medRxiv, PubMed, and Scopus.

3.1 Methodology

Given the focus of this thesis on Systems of Systems (SoS) and related aspects of Interoperability, a query was formulated using two groups of keywords. The first group included: Integration, Interoperability, Digital Twins, API Management, Representational State Transfer (REST), GraphQL and Federated Systems. To specifically address the SoS focus, a second group of keywords related to System of Systems was created. A conditional AND was applied between these two groups and a time constraint was set to include only papers published after 2019.

The initial query returned a substantial number of papers. To refine the results, an additional filter was applied using a third group of keywords with AND NOT conditions. This third group is comprised of authentication, security, authorization, and cybersecurity. This filtering process reduced the number of papers to 111.

Further refinement was necessary due to the presence of papers without DOIs and those behind paywalls, which were inaccessible. Consequently, the number of available papers was reduced to 48. A word count analysis was then performed on the remaining papers to verify that their primary focus was indeed on Systems of Systems and Interoperability. The article was selected if it had more than 5 occurrences of 'Interoperability' and more than 6 occurrences of 'System of Systems', this two conditions narrowed down the number of papers to 7. These are the 7 papers: (Pickering et al., 2020; Mittal et al., 2020; Mohsin and Janjua, 2018; Weinert and Uslar, 2020; Neureiter et al., 2020; Căndeia et al., 2023; Anacker et al., 2022)

Since this systematic selection process was executed to find papers focusing on System of Systems and Interoperability, it was also added two papers related to the System of Systems, Interoperability and

GraphQL (De F. Borges et al., 2022; Li et al., 2024), a Survey about Digital Twins and System of Systems (Olsson and Axelsson, 2023) and a survey about GraphQL (na Mera et al., 2023).

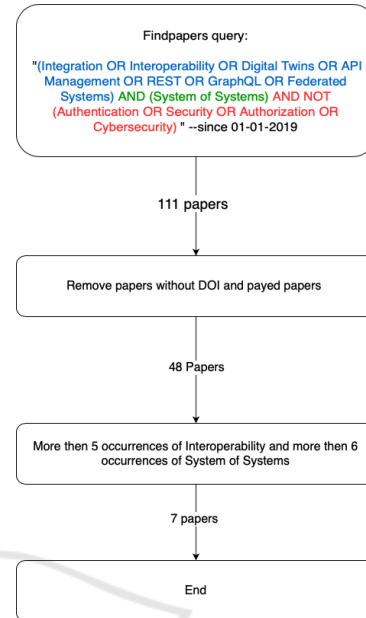


Figure 1: Systematic Research Diagram.

3.2 Preliminary Analysis

(Căndeia et al., 2023) focus on implementing Internet of Things (IoT) and System of Systems (SoS) technologies in various smart applications, such as Industry 4.0, smart cities, and healthcare. They emphasize the importance of interconnectivity and interoperability for seamless communication and data exchange between devices. The authors also highlight the need for reliable quality of service (QoS) to ensure the performance and effectiveness of these applications. (Mittal et al., 2020) present the Simulation, Experimentation, Analytics, and Testing (SEAT) framework, designed to facilitate the development and testing of autonomous systems within a multi-domain environment. The framework emphasizes the composability and interoperability of simulation and analytical applications, enabling the integration of diverse tools and capabilities. (Li et al., 2024; De F. Borges et al., 2022) introduce a framework for using GraphQL. However, one of them uses a framework that leverages ontologies to generate a GraphQL server, and the other uses syntactical analysis. Both of them can query heterogeneous data sources, providing an integrated view of the data.

(na Mera et al., 2023) and (Mohsin and Janjua, 2018) provide comprehensive reviews of GraphQL

and SOA-based software architecture modeling approaches for SoS, respectively. (na Mera et al., 2023) highlight GraphQL's growing adoption and potential research areas, while (Mohsin and Janjua, 2018) delve into the limitations of existing SOA-based modelling techniques for SoS, emphasizing the need for dynamic service identification, composition, and provisioning at runtime. (Olsson and Axelsson, 2023) survey the current knowledge on digital twins in the context of SoS, emphasizing the need for further research to address challenges such as data sharing and integration. (Anacker et al., 2022) review the literature on SoS and patterns, aiming to understand their definitions, classifications, and applications in systems engineering.

(Pickering et al., 2020) propose a time-constrained SoS discovery process and canvas, demonstrating its application in an agricultural case study involving an automated asparagus harvester. The authors emphasize the importance of understanding the relationships between constituent systems and their properties to manage emergent properties effectively. (Weinert and Uslar, 2020) outline the demand and challenges for a structured SoS approach in the agriculture domain, proposing a reference designation-based system architecture documentation approach to address the heterogeneous infrastructure and lack of interoperability. (Neureiter et al., 2020) discuss extending the concept of Domain-Specific Systems Engineering (DSSE) to SoS, highlighting the need for interoperability and compatibility between models from different domains. They present a case study on integrating electric vehicle models with a Smart Grid model to analyze emergent behaviour caused by simultaneous charging.

3.3 Comparison

After analyzing the related work, a methodical comparison was devised. Since this study focuses on the implementation aspect of Systems of Systems (SoS) and Interoperability, six questions were formulated to compare the four implementation-focused papers (Cândeia et al., 2023; Mittal et al., 2020; Li et al., 2024; De F. Borges et al., 2022). The questions are the following, and the answers to those questions are on the Table 1.

1. **Q1:** Does it talk about joining data from multiple sources?
2. **Q2:** Does it propose a methodology?
3. **Q3:** Is the paper related to GraphQL?
4. **Q4:** Does it have an API paradigm restriction?

5. **Q5:** Does it provide a unified vocabulary to integrate APIs?

The comparison of implementation papers in 1 reveals a diverse landscape of approaches to address the challenges of data integration in systems of systems. It is possible to observe that by the table, most of the problems presented in the Subsection 1.1 are tackled by (De F. Borges et al., 2022) and (Li et al., 2024), both relating to GraphQL. This analysis highlights the need for further research and development to create comprehensive solutions that address the diverse challenges of data integration in complex systems.

4 USE CASE

Our use case involves multiple systems within the oil and gas industry, which could collectively be used to create a digital twin of an industrial plant. The key challenge lies in the heterogeneity and complexity of these systems, making data integration a non-trivial task. These systems are currently interconnected through peer-to-peer connections, which are increasing exponentially, leading to a complex and potentially unsustainable network architecture. The proposed solution aims to address this challenge by providing a scalable and efficient way to integrate data from these diverse systems, enabling the creation of a comprehensive digital twin that can accurately represent the real-world behaviour of the industrial plant.

5 PROPOSED SOLUTION

After analyzing the work related to this paper, the solution proposed by this paper is an algorithm that can generate a processing layer between systems that assists in joining data from multiple data sources. The following subsections illustrate how.

5.1 Conceptual Model

The conceptual model is illustrated by Figure 2. It has three main layers: the micro-services, the API Management and the outside layer. The Micro-services layer encompasses all applications and services managed by an organization. The API management layer has the API Gateway and API Portal that expose all endpoints and information of the services. And the outside layer is applications or systems that consume data from the organization's service. This conceptual model will be employed to explore the creation of interfaces between services within a System of Systems

Table 1: Comparison of implementation papers.

	(Cândeia et al., 2023)	(Mittal et al., 2020)	(Li et al., 2024)	(De F. Borges et al., 2022)
Q1	No	No	Yes	Yes
Q2	No	Yes	Yes	Yes
Q3	No	No	Yes	Yes
Q4	Yes	No	No	No
Q5	No	No	Yes	Yes

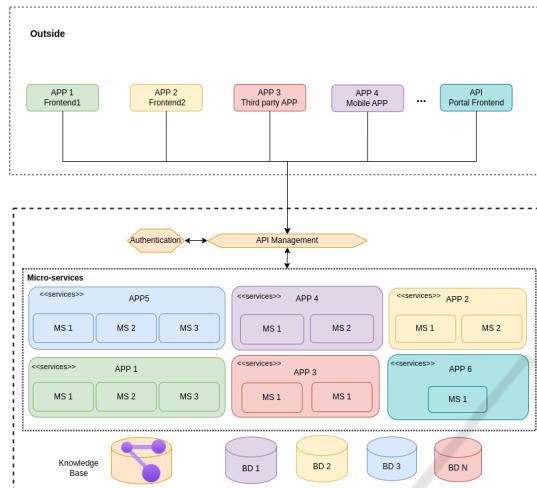


Figure 2: Conceptual Model.

(SoS) framework to develop a Digital Twin. The solution of this paper is focused on generating the API Management layer.

5.2 Server Generation Algorithm

The solution of this study will use GraphQL API Management since other related works like (De F. Borges et al., 2022; Li et al., 2024) have shown results using it. The algorithm will generate a GraphQL API Management based on two main information sources: A Knowledge Graph and the OpenAPI files for each service. These information sources will be employed to define how to merge the data from multiple applications effectively.

The OpenAPI files will be used to extract information regarding the URLs and the input and output format of each endpoint. On the other hand, the Knowledge Graph will have definitions of how to combine data from each endpoint by defining which pieces of data are equivalent in each service.

The algorithm will take all the information sources as input, and for each service in the OpenAPIs, it will create GraphQL Types, Fields and Resolvers for the correspondent service. These resolvers will be simple endpoint calls defined by the OpenAPI file. By doing it this way, all services can be queried through the GraphQL API Management and

also these services can be used afterwards to build new services inside GraphQL.

After generating all the GraphQL Types, Fields and Resolvers, the next step is to create new GraphQL Types, Fields and Resolvers based on each equivalence between services in the Knowledge Graph. After all this process, the algorithm will output a GraphQL Server code that can be instantiated.

6 DISCUSSION

The proposed solution offers a novel approach to address the challenges of data integration in systems of systems. By leveraging GraphQL and knowledge graphs, the algorithm automates the generation of a processing layer that seamlessly connects disparate data sources. This automation not only reduces the development time and effort required for building such integrations but also enhances the flexibility and adaptability of the system. The use of GraphQL as the API management layer ensures efficient data retrieval and a strongly typed schema, contributing to the overall robustness and maintainability of the solution.

However, there are potential limitations to consider. The effectiveness of the algorithm relies on the accuracy and completeness of the knowledge graph, which defines the relationships between data elements from different sources. Inaccurate or incomplete knowledge graphs could lead to incorrect or incomplete data integration. Additionally, while GraphQL offers numerous advantages, it may not be the optimal choice for all use cases. For instance, in scenarios where real-time data streaming is critical, other protocols like gRPC might be more suitable.

7 CONCLUSION

This paper proposes an innovative algorithm for generating a GraphQL API management layer that facilitates data integration in systems of systems. By combining the strengths of GraphQL and knowledge graphs, the algorithm automates the creation of a processing layer that seamlessly connects disparate data sources.

Building upon works like (Li et al., 2024) and (De F. Borges et al., 2022), which sought to address the persistent technical challenges in the software engineering process, our research contributes a novel approach that resonates with organizations grappling with similar issues. This is particularly salient given the observations made in study (na Mera et al., 2023), which underscores the need for innovative solutions, such as code generative tools, to tackle complex elements like paging and nested queries. Our findings, therefore, offer a potentially transformative pathway for enhancing software engineering practices across a multitude of domains.

While potential limitations exist, the proposed solution offers a promising avenue for addressing data integration challenges in complex systems, contributing to the development of more efficient, adaptable, and robust data-driven applications. Future work could implement the algorithm mentioned in this paper. Another possibility is the creation of such Knowledge Graphs through various methods such as syntactical analyses, semantic analyses, or even using an LLM to generate the knowledge graph.

REFERENCES

- Anacker, H., Günther, M., Wyrwich, F., and Dumitrescu, R. (2022). Pattern based engineering of system of systems - a systematic literature review. In *17th Annual System of Systems Engineering Conference (SOSE)*, page 178–183.
- Bondel, G., Landgraf, A., and Matthes, F. (2021). Api management patterns for public, partner, and group web api initiatives with a focus on collaboration. *Proceedings of the ACM on Programming Languages*, 5(OOPSLA):1–28.
- Cândeia, C., Cândeia, G., and Staicu, M. (2023). Impact of iot and sos in enabling smart applications: A study on interconnectivity, interoperability and quality of service. *Procedia Computer Science*, 221:1226–1234.
- De F. Borges, M. V., Rocha, L. S., and Maia, P. H. M. (2022). Micrographql: a unified communication approach for systems of systems using microservices and graphql. In *2022 IEEE/ACM 10th International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS)*, pages 33–40.
- GraphQL Foundation (2024). Introduction to graphql. <https://graphql.org/learn/>. Accessed: 2024-05-22.
- Grosman, J. (2024). Findpapers: A tool for helping researchers who are looking for related works. <https://github.com/jonatasgrosman/findpapers>. Accessed: 2024-05-22.
- Initiative, O. (2024). Openapi specification v3.1.0. <https://spec.openapis.org/oas/latest.html>. Accessed: 2024-05-22.
- Li, H., Hartig, O., Armiento, R., and Lambrix, P. (2024). Ontology-based graphql server generation for data access and data integration. *Semantic Web*.
- Mittal, S., Kasdaglis, N., Harrell, L., Wittman, R. L., Gibson, J., and Rocca, D. (2020). Autonomous and composable m&s system of systems with the simulation, experimentation, analytics and testing (seat) framework. In *Proceedings of the 2020 Winter Simulation Conference*, pages 2305–2316. IEEE.
- Mohsin, A. and Janjua, N. K. (2018). A review and future directions of soa-based software architecture modeling approaches for system of systems. *Service Oriented Computing and Applications*, 12(3):183–200.
- na Mera, A. Q., Fernandez, P., Garc'ia, J. M., and Ruiz-Cort'es, A. (2023). GraphQL: A systematic mapping study. *ACM Comput. Surv.*, 55(10):202:1–202:35.
- Neureiter, C., Binder, C., Brankovic, B., and Lastro, G. (2020). Extending the concept of domain specific systems engineering to system-of-systems. In *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, pages 391–396. IEEE.
- Olsson, T. and Axelsson, J. (2023). Systems-of-systems and digital twins: A survey and analysis of the current knowledge. In *2023 18th Annual System of Systems Engineering Conference (SoSE)*. IEEE.
- Pickering, N., Duke, M., and Lim, S. H. (2020). A time constrained system of systems discovery process and canvas - a case study in agriculture technology focusing on an automated asparagus harvester. In *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, pages 67–74. IEEE.
- Ramonell, C., Chacón, R., and Posada, H. (2023). Knowledge graph-based data integration system for digital twins of built assets. *Automation in Construction*, 156:105109.
- Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). A survey on edge computing for the internet of things. *IEEE Internet of Things Journal*, 3(5):637–646.
- Weinert, B. and Uslar, M. (2020). Challenges for system of systems in the agriculture application domain. In *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, pages 355–360. IEEE.