*Journal of Information and Data Management, 2026, 17:1*
ISSN: 2178-7107 • doi: 10.5753/jidm.2026.7730
This work is licensed under a Creative Commons Attribution 4.0 International License.

**Special Issue Paper**

# Provenance as the Backbone of Systems-of-Systems Data Governance

**Jessica Monçôres de Almeida** [Universidade Federal Fluminense (UFF) | *jmoncores@id.uff.br* ]
**Débora Pina** [Universidade Federal do Rio de Janeiro (UFRJ) | *dbpina@cos.ufrj.br* ]
**Rodrigo Pereira dos Santos** [Universidade Federal do Estado do Rio de Janeiro (UNIRIO) | *rps@uniriotec.br* ]
**Vanessa Braganholo** [Universidade Federal Fluminense (UFF) | *vanessa@ic.uff.br* ]
**Daniel de Oliveira** [Universidade Federal Fluminense (UFF) | *danielcmo@ic.uff.br* ]

*Universidade Federal Fluminense (UFF), Niterói, RJ, Brazil – Institute of Computing*

**Abstract.** Systems-of-Systems (SoS) integrate independent, autonomous systems (*i.e.*, constituent systems) through well-defined dataflows, thereby enabling benefits such as reuse, scalability, and resilience. However, data governance in SoS remains challenging, particularly due to the lack of effective mechanisms to control and monitor the data lifecycle across constituent systems. Data produced by one constituent system is often consumed, transformed, and stored by others, making it difficult to ensure end-to-end traceability, data quality, and integrity. This manuscript presents `PROVGov-SoS`, an approach grounded in provenance data management to address these challenges. `PROVGov-SoS` explicitly structures inter-system information flows and captures provenance metadata to provide a view of the data lifecycle within an SoS. We evaluate `PROVGov-SoS` through a feasibility study conducted on a real-world SoS, demonstrating its practicality and its potential to enhance data governance by improving transparency and traceability.

**Keywords:** Systems-of-Systems, Provenance, Data Governance

**Received:** 04 March 2026 • **Published:** 27 November 2026

## 1 Introduction

Over the past decade, Systems-of-Systems (SoS) have received increasing attention in both academia and industry [Maier, 1998; Teixeira *et al*., 2023; Cavalcante *et al*., 2024]. An SoS integrates multiple autonomous information systems, known as constituent systems, whose interoperability is achieved through explicitly defined dataflows that enable coordinated interaction [Boardman and Sauser, 2006]. In comparison with traditional standalone systems, SoS represent an architectural evolution in which systems originally designed and operated independently are interconnected to function as a cohesive whole [Cavalcante *et al*., 2024]. A characteristic of an SoS is its collaborative nature: each constituent system maintains operational independence while cooperating with others to achieve global goals (missions) that cannot be accomplished by individual systems. Moreover, SoS are inherently complex and often geographically distributed, reflecting the scale and heterogeneity of their constituent systems. Another defining property of SoS architectures is their adaptability, specifically the ability to add or remove constituent systems without compromising the desired emergent behavior of the overall system. This property captures the core principles of functional independence and interoperability that distinguish SoS from conventional integrated systems [Maier, 1998].

While SoS offer important advantages, *e.g.*, component reuse, architectural resilience, and the ability to integrate heterogeneous technologies, they also introduce challenges related to data governance [Curry and Sheth, 2018]. In an SoS, data may be produced, exchanged, and processed by multiple constituent systems, resulting in a complex data ecosystem in which data origins may span both intra-organizational and inter-organizational boundaries [Curry and Sheth, 2018; Lis and Otto, 2020; Curry *et al*., 2022]. Previous work emphasizes the need for explicit mechanisms to govern the data lifecycle within SoS [Curry and Sheth, 2018]. Such control is particularly critical in the context of auditability and regulatory compliance, where it is necessary to ensure the traceability of data and their transformations over time. Unlike traditional standalone systems, in which data are typically centralized and accessible, an SoS distributes data across its participating systems. As a result, auditing a specific data item in the absence of a clearly defined data derivation path becomes impractical. This lack of visibility into this data derivation path undermines both the reliability of the data and the effectiveness of governance within the SoS. Figure 1 presents an example of an SoS composed of three independent systems, in which the data consumed by a user undergo a sequence of transformations across `System 1`, `System 2`, and `System 3` in that order before being delivered.

Several approaches have been proposed to address SoS monitoring. Vierhauser *et al*. [2016] introduce the ReMinds framework, which proposes a real-time model for monitoring the state of an SoS through the use of a domain-specific language. Kong *et al*. [2020] present a runtime monitoring approach that extracts execution traces from software systems by employing sensors to capture low-level events, such as system calls, interruptions, and context switches, with the goal of minimizing interference with the monitored systems. Calabro *et al*. [2021] investigate the integration of multiple runtime monitoring solutions across different application domains, including urban traffic management and healthcare systems. While these approaches represent important advances in monitoring SoS, they focus on specific concerns, such as observing execution events and evaluating systems' operational status. They do not address challenges related to

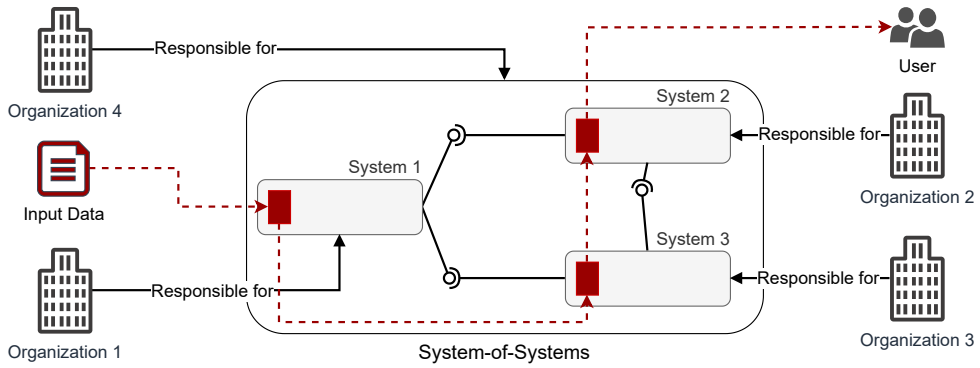


**Figure 1.** Data traceability within an SoS.

monitoring data transformations, ensuring the traceability of information as it flows across constituent systems, or supporting distributed data governance. These aspects are central to guaranteeing data reliability, auditability, and compliance in an SoS, and thus remain unaddressed in the existing literature.

Provenance data [Herschel *et al.*, 2017] naturally emerge as a mechanism for representing data derivation paths in SoS [Allen *et al.*, 2011; Gammack *et al.*, 2016]. Their adoption in this context aligns with a fundamental principle of data provenance: making systems auditable while enabling users to evaluate the reliability and trustworthiness of the data they produce and consume [Moreau *et al.*, 2017]. As such, provenance plays a key role in supporting data governance within SoS. When properly structured and defined in accordance with recognized provenance standards, *e.g.*, W3C PROV [Groth and Moreau, 2013], such metadata enable data to be traced across their entire lifecycle, even as they traverse multiple autonomous systems. This end-to-end traceability provides the level of transparency required to establish trust among constituent systems and to satisfy regulatory, compliance, and business requirements, particularly in critical and sensitive application domains [Fu *et al.*, 2011; Gammack *et al.*, 2016].

To address the gaps in data traceability in SoS, this work proposes `PROVGov-SoS`, an approach that aims at providing data governance in SoS through the capture, persistence, and querying of provenance data. The core premise of `PROVGov-SoS` is to use provenance data as a foundation for fine-grained monitoring of information flows among constituent systems. For that, the approach observes the logs and runtime behaviors of individual SoS components to intercept and extract events related to data transformations and exchanges. The modeling and visualization of provenance information in `PROVGov-SoS` follow the W3C PROV recommendation [Groth and Moreau, 2013], thereby ensuring compliance with established standards and easing interoperability with existing PROV-compliant tools. Provenance data are stored in a graph-oriented database that reflects the structure of provenance graphs and supports both real-time and *post-mortem* queries over data derivation paths.

The main goal of `PROVGov-SoS` is to provide a transparent and fine-grained view of data derivation paths within an SoS, by enabling the traceability and visualization of data transformations over time. From a data governance perspective, this feature enables stakeholders to analyze the data lifecycle in a structured and auditable way, thereby fostering trust, supporting regulatory compliance, and improving decision-making. `PROVGov-SoS` was evaluated through a feasibility study conducted in a real-world SoS, *i.e.*, the Annual Faculty Report system, which comprises academic and administrative systems of the Universidade Federal Fluminense (UFF), as well as external platforms such as Lattes. This SoS integrates data originating from multiple institutional and governmental sources. During the evaluation, provenance data were captured across the data import, transformation, and integration processes from these heterogeneous sources. All generated metadata were structured and persisted in a graph-oriented database in Neo4j, enabling the execution of analytical provenance queries. A set of queries, derived from real issues reported by system users, was executed over the provenance database to assess the practical utility of the approach. The results demonstrate that `PROVGov-SoS` provides clear views of dataflows, supports analysis of information flows across constituent systems, and aids the resolution of user-identified questions and inconsistencies.

This manuscript extends our previous work [Almeida *et al.*, 2025] by offering a more detailed, in-depth explanation of `PROVGov-SoS`, including additional architectural and implementation details. It also provides a background section on SoS. Additionally, it includes a broader experimental evaluation, incorporating new queries that further demonstrate the applicability and effectiveness of `PROVGov-SoS`.

The remainder of this manuscript is organized as follows. Section 2 introduces the theoretical background on provenance data, while Section 3 presents an overview of SoS and a provenance-aware SoS formalism. Section 4 describes the proposed approach, and Section 5 evaluates the use of `PROVGov-SoS` in a real-world SoS scenario. Section 6 discusses related work in the SoS data governance domain. Finally, Section 7 summarizes the main results and outlines directions for future work.

## 2 A Brief Tour of Provenance Data

Provenance data play a crucial role by providing a detailed record of data derivation paths, thereby fostering transparency, reliability, and security [de Oliveira *et al.*, 2018]. In several domains, such as biology, medicine, and engineering, tracking provenance is mandatory to protect rights, ensure

regulatory compliance, assign responsibility, and authenticate information [Hasan *et al.*, 2009]. The W3C PROV is the *de facto* standard for representing provenance data across multiple contexts. It comprises specifications, models, and guidelines that standardize the description, storage, and exchange of provenance data across heterogeneous systems [Gil and Miles, 2013]. At its core lies PROV-DM [Moreau and Missier, 2013] (Figure 2), the PROV Data Model, which defines the following elements of provenance: (i) $E$ is the set of entities, (ii) $A$ is the set of activities, and (iii) $A_g$ is the set of agents. Thus, we define the universe $U = E \cup A \cup A_g$ as the possible constructs.

These constructs are connected through a set of provenance relationships that aim to represent data production and consumption, responsibilities, and temporal constraints. Provenance data are modeled using typed binary relations. Four of the most important relations are: (i) $\mathsf{used} \subseteq A \times E$, (ii) $\mathsf{wasGeneratedBy} \subseteq E \times A$, (iii) $\mathsf{wasAssociatedWith} \subseteq A \times A_g$, and (iv) $\mathsf{wasDerivedFrom} \subseteq E \times E$. Let us also define $t : U \to \mathbb{T}$ as the timestamp function into a totally ordered time domain $\mathbb{T}$. We can then define the following temporal constraints: (i) $(a, e) \in \mathsf{used} \implies t(e) \leq t(a)$ and (ii) $(e, a) \in \mathsf{wasGeneratedBy} \implies t(a) \leq t(e)$.

Provenance data can therefore be represented as a *Provenance Graph* [Sultana and Bertino, 2012], which can be defined as $\mathcal{P} = (U, R)$, where nodes are the constructs in $U$ and edges are defined by $R = \{\mathsf{used}, \mathsf{wasGeneratedBy}, \mathsf{wasAssociatedWith}, \mathsf{wasDerivedFrom}\}$. It is worth noticing that edges are type-constrained and directed. In $\mathcal{P}$, we can define a relation $\prec$ on entities: $e_1 \prec e_2 \iff e_2$ is directly or indirectly derived from $e_1$. Then $(E, \prec)$ is a strict partial order: $\neg(e \prec e)$ and $(e_1 \prec e_2 \wedge e_2 \prec e_3) \implies e_1 \prec e_3$, with provenance assertion $\mathsf{wasDerivedFrom}(e_2, e_1)$. Each activity $a \in A$ may be modeled as a function $f_a : 2^E \to E$, such that $f_a(\{e_1, \ldots, e_n\}) = e'$, with provenance assertions $\mathsf{used}(a, e_i)$ and $\mathsf{wasGeneratedBy}(e', a)$.

The provenance graph $\mathcal{P}$ can be represented in multiple ways, including XML, JSON, and textual documents. PROV-JSON [Huynh *et al.*, 2013] is a serialization format for provenance data based on JSON, a lightweight data interchange format. Designed to be both human-readable and machine-parsable, PROV-JSON eases the integration of provenance data into web applications and JSON-based systems. This format was adopted in `PROVGov-SoS` due to its performance, interoperability, readability, and compatibility.

Despite these advantages, PROV-JSON presents limitations with respect to querying. Therefore, it is necessary to store provenance data in a database whose schema conforms to the conceptual structure of PROV, enabling efficient querying. In `PROVGov-SoS`, provenance data are stored in Neo4j, a graph-oriented database management system. This choice is motivated by the fact that graph-oriented databases naturally match the structure of the provenance graph $\mathcal{P}$. Consequently, no complex transformations or additional mappings are required between the provenance graph (originally expressed in PROV-JSON) and the database schema. Neo4j allows the persistence of the provenance graph in its native form, preserving relationship semantics and navigability among constructs [Wercelens *et al.*, 2019; Almeida *et al.*, 2019].

# 3 Systems-of-Systems in a Nutshell

A System-of-Systems (SoS) [Ferreira *et al.*, 2023] is an alliance formed through the collaboration of multiple constituent systems that together provide functionalities beyond those achievable by the systems in isolation. The concept of an SoS arises from the coordination of constituent systems that are independently developed, operated, and evolved, often under different administrative and organizational authorities.

Let us consider the example presented in Figure 1. In this scenario, `System 1` represents the undergraduate information system of a university, `System 2` represents the academic management system, and `System 3` represents a system that manages professors' CVs. These three systems operate independently, but each one of them provides functionalities that enable the implementation of the SoS to support institutional reporting and decision-making. The SoS-level functionality emerges from the interaction and data exchange among these constituent systems, rather than from centralized control.

SoS are typically heterogeneous, as constituent systems may be implemented using different programming languages and frameworks, and may rely on different data models and database systems, which is referred to as operational and managerial independence. Moreover, SoS are usually distributed because each constituent system can be deployed in different environments, *e.g.*, on-premises servers or cloud platforms. From an architectural perspective, SoS are based on loose coupling and interoperability mechanisms, *e.g.*, APIs and shared data formats such as JSON and XML. These mechanisms enable interaction among constituent systems while minimizing dependencies on specific implementations. However, this architectural style introduces a key challenge: auditing the results produced by an SoS, *i.e.*, analyzing the data derivation paths across constituent systems. No single system has complete visibility into how data are produced, transformed, and consumed throughout the SoS.

Data exchange plays a key role in enabling cooperation within an SoS. Data produced by one system serve as input to others, undergoing a sequence of transformations as they traverse the SoS. While this dataflow enables cooperation in an SoS, it also introduces challenges related to accountability and traceability [Cordeiro *et al.*, 2026]. In particular, understanding how data are transformed across constituent systems is essential for ensuring reliability, supporting auditability, and enabling informed decision-making. Bridging this gap requires techniques that capture data derivation paths at the SoS level, rather than within individual systems.

In this context, capturing provenance data in an SoS naturally provides a backbone for traceability in complex, heterogeneous environments. Formally, we can define a provenance-aware SoS as a tuple $\mathsf{SoS}_{prov} = \langle S, L, I, \mathcal{E} \rangle$, where: (i) $S = \{S_1, S_2, \ldots, S_n\}$ is the set of constituent systems; (ii) $L$ is the set of inter-system relationships; (iii) $I$ denotes the set of interoperability mechanisms; and (iv) $\mathcal{E}$ denotes the execution environment in which the SoS executes.

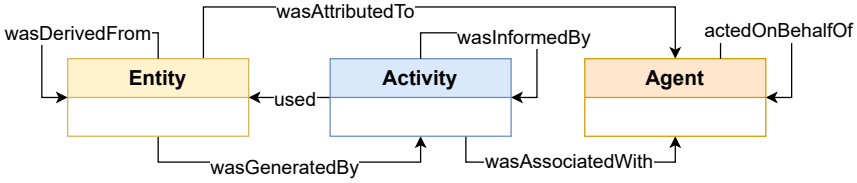


**Figure 2.** PROV-DM, adapted from Moreau and Missier [2013].

Each constituent system $S_i \in S$ is defined as a tuple $S_i = \langle A_i, D_i, P_i, \sigma_i \rangle$, where: (i) $A_i$ is the set of actors (human or computational users); (ii) $D_i$ is the set of data artifacts consumed and produced; (iii) $P_i$ is the set of activities executed within the system; and (iv) $\sigma_i$ is the local provenance schema of $S_i$.

An SoS may adopt a provenance schema $\sigma_{SoS}$ defined over three primitive types aligned with the W3C PROV model: (i) an entity $e \in E$, representing an immutable state of a data artifact; (ii) an activity $a \in A$, representing an action that consumes or produces entities; and (iii) an agent $ag \in A_g$, representing an actor responsible for executing activities. Each constituent system exports its provenance data through a mapping function $\phi_i : \sigma_i \rightarrow \sigma_{SoS}$, which translates local provenance representations into the provenance schema of the SoS. Cross-system provenance is captured through inter-system events defined as tuples $b = \langle S_i, S_j, e, t, c \rangle$, where $S_i$ and $S_j$ denote the source and destination systems, respectively; $e$ is the exchanged entity; $t$ is a timestamp; and $c$ is contextual metadata (*e.g.*, errors and warning messages). These events introduce inter-system edges in the provenance graph, thereby enabling end-to-end provenance tracking across the SoS.

## 4 The PROVGov-SoS Approach

PROVGov-SoS uses provenance data as the backbone of data governance in SoS to ensure that data remain traceable, reliable, and auditable [Moreau *et al.*, 2017]. To do so, PROVGov-SoS is designed to capture and manage provenance data in a distributed, heterogeneous environment, a defining characteristic of SoS, as discussed in Section 3. Figure 3 illustrates the architecture of PROVGov-SoS, which is composed of six main components: (i) the external SoS, (ii) the Provenance Event Broker, (iii) the PROV Mapping Engine, (iv) the Provenance Document Repository (PROV-JSON), (v) the Global SoS Provenance Graph Database, and (vi) the Provenance Query and Export Service.

The execution of PROVGov-SoS begins within each constituent system of the SoS. Using Aspect-Oriented Programming (AOP) [Kiczales *et al.*, 2001], the methods responsible for data transformations are intercepted. Upon each invocation in each constituent system, metadata describing the performed transformation and the affected data are automatically captured, including timestamps and the identifier of the constituent system that executed the method. This information is then encapsulated into a message and forwarded to the *Provenance Event Broker* (step 1 in Figure 3). The *Provenance Event Broker* serves as an intermediary component that receives messages containing metadata about executed transformations and disseminates them to the remaining components of the PROVGov-SoS architecture.

Once the metadata import message is received, the *Provenance Event Broker* forwards it to the *PROV Mapping Engine* (step 2 in Figure 3). This component processes event data from data transformations within SoS constituent systems and structures it into a global provenance graph in accordance with the W3C PROV recommendation. The provenance graph is constructed incrementally as new events are received. Although each constituent system can maintain its own *local* provenance graph, capturing only the internal transformations it performs, the PROVGov-SoS approach integrates these local graphs into a global SoS-level provenance graph. This unified graph captures dependency and chaining relationships among data across multiple systems, thereby providing an integrated view of the data lifecycle across the SoS.

To construct the global SoS provenance graph, PROVGov-SoS uses the concept of a *Bundle*, as defined in the PROV recommendation. In PROV, a bundle is characterized as the "provenance of provenance" [Moreau and Missier, 2013], a special entity that encapsulates a self-contained subgraph of provenance data. In PROVGov-SoS, a bundle is interpreted as a logical transaction executed within the SoS. Each bundle aggregates the data transformations that occur during a transaction, which may span multiple constituent systems within the SoS. This design choice provides fine-grained provenance data, enabling data traceability and supporting future analyses. Within a bundle, actions that generate, modify, or invalidate PROV entities are modeled as activities, as they represent explicit data transformation events. By encapsulating these activities, the bundle preserves the semantic context of each transaction and enables the reconstruction of historical interactions among constituent systems. In this sense, a bundle functions as a transaction capsule, playing a central role in the logical and temporal organization of provenance elements across the SoS.

An example of a PROV bundle is shown at the bottom of Figure 3, where two different bundles are used to represent separate data transformation transactions occurring within an SoS. Each bundle encapsulates the provenance data associated with an individual transaction, thereby enabling traceable descriptions of data evolution over time. `Bundle A` characterizes the initial data import transaction. Specifically, it describes the creation of the entity (depicted as a yellow ellipse) named `Import/A`, which represents a collection aggregating the entities `Class/A` and `Activity/A`. These entities are explicitly linked to the collection through the hadMember relationship, as defined in the W3C PROV recommendation, thus formalizing their membership within the imported dataset.

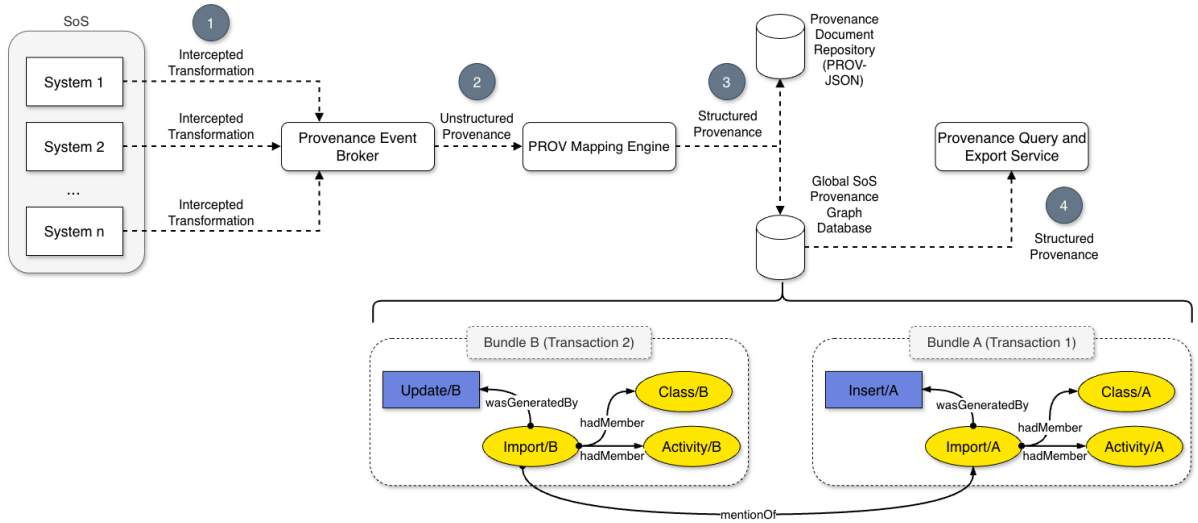


**Figure 3.** PROVGov-SoS architecture.

`Bundle B` captures a subsequent transaction that operates on data from `Bundle A`. In this case, the activity `Update/B` (illustrated as a blue rectangle) modifies the imported data, thereby generating a new entity, `Import/B`. To preserve consistency and traceability across transactions, connections between entities defined in different bundles are established using the mentionOf relationship [Moreau and Lebo, 2013]. This relationship enables an entity in one bundle to be expressed as a specialization of an entity defined in another bundle, thereby ensuring semantic continuity and maintaining explicit provenance links across distinct transactional contexts. One of the key contributions of PROVGov-SoS is the application of the mentionOf relationship to support traceability of transactional data. While the PROV specification defines this relationship in general terms to link provenance descriptions across different bundles without imposing an execution order, our approach introduces a more specialized interpretation. Specifically, mentionOf is used to establish a chronological chain of events involving the same data entity, thereby enabling the reconstruction of its transformation history across multiple transactions. This extended use of the mentionOf relationship has been implemented in the ProvToolbox library [Moreau, 2013] and will be submitted to the community as a proposed enhancement.

Within bundles, relationships play a key role in describing the interactions among entities, activities, and agents. Although the PROV recommendation defines a rich set of relationship types, PROVGov-SoS focuses on the following subset: (i) wasGeneratedBy, which links an entity to the activity responsible for its generation, thereby establishing a direct association between data and their creation process; (ii) used, which connects an activity to the entity it consumes during execution, capturing operational dependencies; (iii) wasInvalidatedBy, which denotes the point at which an entity is invalidated, destroyed, or expired as a result of a specific activity; (iv) hadMember, which represents that a collection-type entity contains other entities as its members; and (v) mentionOf, a ternary PROV-LINKS relationship extended in this manuscript, which expresses a reference to or derivation of an entity from another entity previously defined in a different bundle, thereby enabling semantic chaining across multiple transactions. The mentionOf relationship is particularly important for constructing the global SoS provenance graph, as it enables the linkage of otherwise dispersed provenance information across different SoS components and transactions.

Once all received data have been structured into a provenance graph by the *PROV Mapping Engine*, they are stored in the *Provenance Document Repository* in PROV-JSON format, and the *Global SoS Provenance Graph Database* is updated (step 3 in Figure 3). In the current implementation of PROVGov-SoS, we use Neo4j [Neo4j, 2025] as the graph-oriented database management system. The schema of the *Global SoS Provenance Graph Database* may vary across different SoS, as the semantics of graph nodes and edges depend on the specific application domain and characteristics of each SoS. For example, in the experiments presented in Section 5, we consider an SoS that integrates academic data from multiple constituent systems. The objective in this setting is to analyze how data imports and updates are performed across systems. Accordingly, the graph schema used in this case (Figure 4) consists of `PRODUCT`, `IMPORT`, and `USER` nodes. The graph further captures associations between products affected by import operations and the users involved in these processes. It is worth noting that the generic PROV constructs could be used directly to model such scenarios. However, we deliberately allow users to define an application-specific schema that best fits their analytical needs, and subsequently map each schema element to its corresponding PROV construct (the stereotypes in Figure 4). This design choice provides flexibility while preserving compliance with the PROV standard.

Finally, the *Provenance Query and Export Service* issues a set of database queries using the Cypher language to retrieve the relevant provenance graph or subgraph for a given analysis (step 4 in Figure 3). By supporting PROV-compliant queries and leveraging a graph-based storage model, PROVGov-SoS enables users to analyze and interpret how data are produced, propagated, and transformed over time within the SoS. Moreover, since the stored provenance data comply with the PROV recommendation, they can be

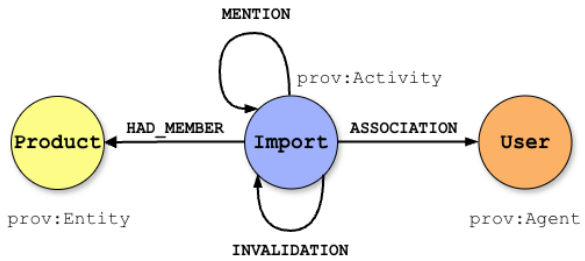


**Figure 4.** An example schema of the *Global SoS Provenance Graph Database*.

readily integrated with existing tools for visualization, auditing, and data analysis [Moreau *et al.*, 2017]. The source code of the current version of PROVGov-SoS is publicly available at https://github.com/UFFeScience/PROVGov-SoS.

## 5 Evaluation of PROVGov-SoS

To evaluate the PROVGov-SoS approach in a real-world scenario, we selected an SoS from UFF called the Annual Faculty Report (*Relatório Anual Docente* – RAD, in Portuguese)[1] as a case study. This SoS is used by faculty members to report all academic activities performed during a given year, which are subsequently audited by Brazilian public authorities. The SoS is composed of five constituent systems (Figure 5): (i) the Lattes Curriculum, maintained by the Ministry of Science, Technology, and Innovation (MCTI); (ii) SIAPE, maintained by the Ministry of Management and Innovation in Public Services (MGI); (iii) idUFF, maintained by UFF for managing undergraduate data; (iv) SIGPROJ, maintained by UFF for managing extension project data; and (v) SisPos, maintained by UFF for managing graduate student data.

To enable a retrospective analysis, we selected data covering the period from 2021 to 2024. The dataset comprises a large and diverse collection of information, including academic publications, undergraduate and graduate course offerings, and records of faculty and students. The evaluation uses the provenance graph generated by PROVGov-SoS to investigate import-related issues commonly reported by end users in this SoS. This analysis demonstrates the effectiveness of PROVGov-SoS in diagnosing real-world anomalies. In this experiment, attribute names were preserved in Portuguese in the queries, but Table 1 provides the corresponding English translations.

The first analysis focuses on a recurring inconsistency reported by users: after certain import processes, the number of credit hours of a course (attribute *ch* in the graph) is recorded as zero, which may represent an error for some courses. To investigate this issue, we formulated a query in the Cypher language[2] and executed it over the provenance database, without relying on extension packages such as APOC[3]. We first executed Query Q1, shown in Listing 1, to identify entities in the provenance graph whose attribute associated with credit hours equals zero and that have not undergone subsequent modifications by other systems. Specifically, we kept only nodes that do not participate in MENTION relationships, *i.e.*, entities that were neither referenced nor

[1] https://app.uff.br/rad/
[2] https://neo4j.com/docs/getting-started/cypher/
[3] https://neo4j.com/labs/apoc/

**Table 1.** Portuguese–English terminology equivalences.

| Portuguese | English |
|---|---|
| conectado | Connected |
| disciplina, disciplinaNome | Course (name) |
| ocorrencias | Occurrences |
| propriedade | Property |
| ch, cargaHoraria | Credit hours |
| nosColetados | Collected nodes |
| qtdAlunos | Number of students |
| aulaPosGraduacao | Graduate class |
| afastamento | Leave of absence |
| Dissertação | Dissertation |
| Estágio Docência | Teaching Internship |
| Tese | Thesis |
| Seminário de Projetos I, II | Project Seminar I, II |
| Dissertação I, II | Dissertation I, II |
| Exame de Qualificação | Qualifying Exam |
| orgao, idOrgao | Department/Institute (ID) |
| quantidade | Amount |

modified in other transactions (*bundles*). The query returns these nodes together with their associated relationships (Figure 6), enabling the isolation of data representing the final system state, with no recorded updates or later corrections. The results can be inspected both visually and in JSON format.

Listing 1: Query Q1 – Detection and analysis of insertion anomalies affecting the recorded credit hours of academic courses.

```
Q1: MATCH (n {ch:"0"})-[*]-(conectado)
    WHERE NOT EXISTS {(n)<-[:HAD_MEMBER]-()
        <-[:MENTION]-()}
    RETURN n,conectado
```

Next, we conducted a second analysis using Query Q2 to identify the names of courses associated with entities whose credit hours are recorded as zero, and to compute the frequency of each course. The goal of this analysis is to determine which courses are most affected by this inconsistency during the annual data import process. Listing 2 presents the query written in Cypher, and Table 2 reports the corresponding results.

Listing 2: Query Q2 – Detection of courses with zero credit hours.

```
Q2: MATCH (n {ch:"0"})
    WHERE NOT EXISTS {(n)<-[:HAD_MEMBER]-()
        <-[:MENTION]-()}
    RETURN n.disciplinaNome AS disciplina,
        count(*) AS ocorrencias
    ORDER BY ocorrencias DESC;
```

Query Q3 further extends the provenance analysis by examining dependency relationships among entities that are not directly connected in the graph but are linked indirectly through transitive closures. This analysis enables the identification of potential chains of updates that data may have undergone after their initial import into the system. Specifically, the query compares different versions of the same entity to detect inconsistencies that may indicate incorrect initial imports followed by subsequent modifications. By analyzing these discrepancies, the case study exposes weak-

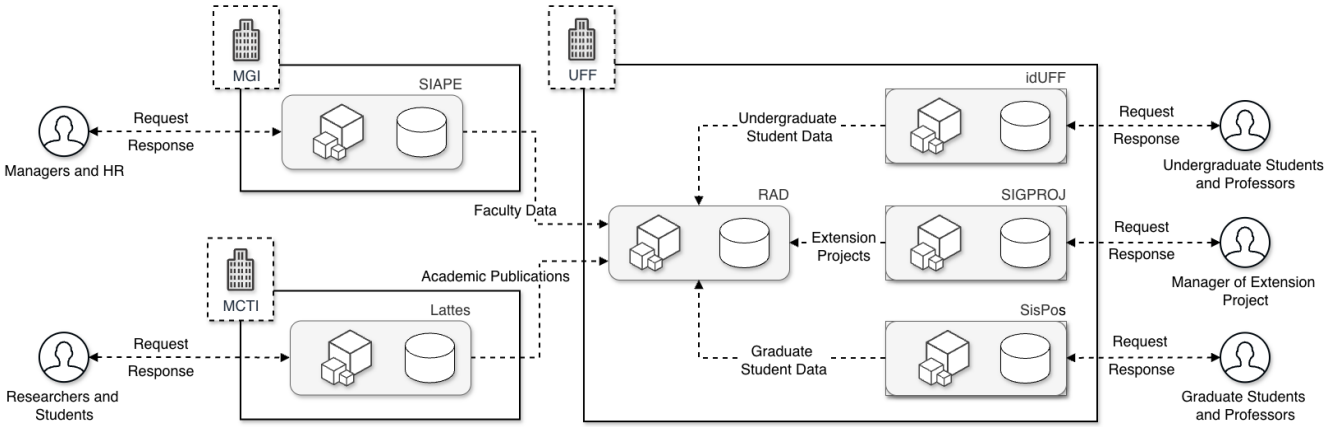


**Figure 5.** The Annual Faculty Report SoS selected as a case study.

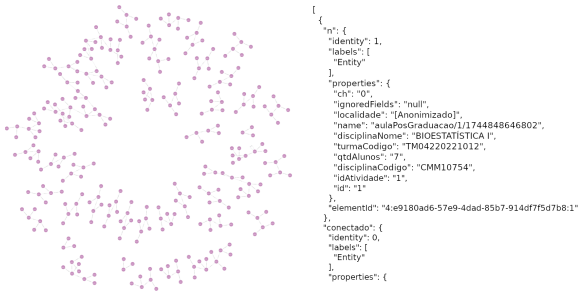

**Figure 6.** Results for Query `Q1` – entities that have not undergone subsequent changes. (Left) Graph resulting from the query; (Right) JSON file with the query result.

**Table 2.** Results for Query `Q2` – Courses with zero credit hours.

| Course | Number of Occurrences |
|---|---|
| Dissertação | 187 |
| Estágio Docência | 93 |
| Tese | 83 |
| Seminário de Projetos II | 71 |
| Dissertação I | 71 |
| Seminário de Projetos I | 62 |
| Exame de Qualificação | 52 |
| Dissertação II | 43 |

nesses in the data integration process across the SoS and provides insights to support both corrective and preventive measures. Listing 3 presents the Cypher query, and Table 3 reports the corresponding results.

Listing 3: Query `Q3` – Versions of entities that underwent changes in a transitive closure.

```
Q3: MATCH (n {ch:"0"})<-[*..3]->(m)
    WHERE n.disciplinaNome IS NOT NULL AND m
        .disciplinaNome IS NOT NULL
    WITH n, m, [propriedade IN keys(n)
      WHERE (n[propriedade] <> m[propriedade
          ] AND propriedade <> 'name')
     ] AS diffs
    WHERE size(diffs)>0
    RETURN n.name, m.name, diffs
    ORDER BY size(diffs) DESC
```

To conclude the analysis of issues related to course workload inconsistencies, we employ Query `Q4`, which is designed to identify entities whose recorded credit-hour values deviate from the standard pattern observed for a given course. Such deviations are indicative of potential data recording errors that may have been introduced during the data import or integration process. This query first selects the most recent version of each relevant entity in the provenance graph, ensuring that the analysis reflects the final state of the data presented to end users. The selected entities are then grouped by their associated course, and the distribution of credit hours within each group is analyzed. By comparing individual credit-hour values against the pattern for the corresponding course, the query can highlight outliers that may indicate anomalous or inconsistent records. The anomaly detection mechanism implemented in Query `Q4` is grounded in provenance-based reasoning and pattern identification over historical data. By using both versioning information and inter-entity relationships captured in the provenance graph, this query provides a systematic way to identify discrepancies that might otherwise go unnoticed. As such, it supports auditing and validating the data import process, as well as diagnosing integration problems across constituent systems within the SoS. Listing 4 presents the Cypher query corresponding to `Q4`, while Figure 7 illustrates the resulting set of detected anomalies.

**Table 3.** Results for Query `Q3` – pairs of entity versions that experienced subsequent modifications identified through transitive closure analysis.

| n.name | m.name | Detected Differences |
|---|---|---|
| aulaPosGraduacao/4476/1744893030548 | aulaPosGraduacao/4476/1744893030541 | [ch, qtdAlunos] |
| aulaPosGraduacao/5015/1744893055406 | aulaPosGraduacao/5015/1744893055415 | [ch, qtdAlunos] |
| aulaPosGraduacao/5016/1744893055444 | aulaPosGraduacao/5016/1744893055451 | [ch, qtdAlunos] |
| aulaPosGraduacao/5611/1744893108045 | aulaPosGraduacao/5611/1744893108039 | [qtdAlunos, ch] |
| aulaPosGraduacao/30/1744848648811 | aulaPosGraduacao/30/1744848648828 | [qtdAlunos] |
| aulaPosGraduacao/30/1744848648828 | aulaPosGraduacao/30/1744848648811 | [qtdAlunos] |
| aulaPosGraduacao/42/1744848649254 | aulaPosGraduacao/42/1744848649276 | [qtdAlunos] |

Listing 4: Query `Q4` – Identification of courses whose assigned credit hours deviate from the established patterns.

```
Q4: MATCH (n)
    WHERE NOT EXISTS {(n)<-[:HAD_MEMBER]-()
        <-[:MENTION]-()}
    AND n.disciplinaNome IS NOT NULL AND n.
        ch IS NOT NULL
    WITH n.disciplinaNome AS disciplina,
        collect(n.ch) AS cargaHoraria,
       collect(n) AS nosColetados
    WITH disciplina, nosColetados,
       size([ch IN cargaHoraria WHERE ch =
           "0"]) AS zeros
    WHERE toFloat(zeros) / toFloat(size(
        cargaHoraria)) < 0.5
    UNWIND nosColetados AS nos
    WITH nos WHERE nos.ch="0"
    RETURN nos
```

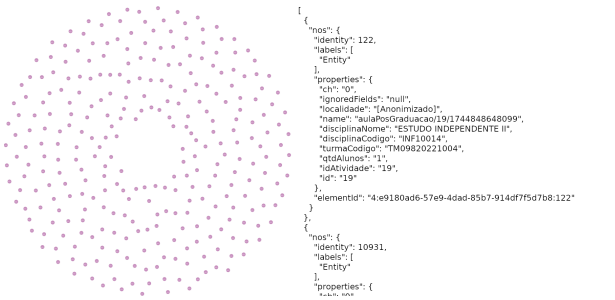


**Figure 7.** Results for Query `Q4` – identification of courses whose assigned credit hours deviate from the established patterns.

Query `Q4` can be further extended to focus specifically on entities that have undergone modifications over time, enabling a more fine-grained analysis of credit-hour inconsistencies. In this extended configuration, the query filters entities with multiple recorded versions in the provenance graph, and examines whether the credit-hour attribute was modified between successive versions and the temporal point at which such changes occurred. This additional level of analysis enables tracing the evolution of credit-hour values throughout the data import and integration pipeline, thereby identifying the precise stage at which an inconsistency was introduced. Consequently, this provenance-driven analysis not only supports post hoc diagnosis of observed errors but also provides actionable insights to improve data integration pipelines, strengthen validation mechanisms, and prevent the recurrence of similar inconsistencies in future import cycles.

The usefulness of the `PROVGov-SoS` provenance data model goes beyond the identification and analysis of data inconsistencies. In particular, it enables deeper insights and supports analytical queries that capture the dynamics of user interaction with services over time. By explicitly modeling the data derivation path and evolution, transformations, and dependencies, the provenance database enables a shift in perspective, moving from examining data as a static artifact to understanding it as a continuously evolving process. Under this approach, the database is no longer interpreted as a fixed representation of a single system state, *i.e.*, a snapshot, but rather as a dynamic and comprehensive historical record that documents the complete lifecycle of data in SoS, from its initial creation and import to subsequent modifications, corrections, and usage. This temporal richness allows users to reconstruct event sequences, identify behavioral patterns, and reason about cause-and-effect relationships that are otherwise difficult to observe.

To illustrate this, the following analyses focus on records of faculty leave of absence. In this case, the evaluation of the corresponding provenance graphs reveals a data-modification scenario that differs substantially from the credit-hour-related inconsistencies discussed earlier, thereby demonstrating the provenance model's flexibility in capturing and explaining different classes of analyses. To investigate this new context, we designed Query `Q5` (shown in Listing 5) with the goal of identifying entities that have undergone one or more modifications over time. This query explicitly targets versioned entities in the provenance graph, allowing the analysis to focus on records whose state has evolved through successive updates. The execution of Query `Q5` over the faculty leave of absence data reveals a different pattern characterized by a high frequency of changes. In the provenance graph, this behavior manifests as long chains of connected nodes, as illustrated in Figure 8. Each node in such a chain corresponds to a different version of the same underlying entity, collectively capturing a detailed history of incremental updates, revisions, or corrections applied to the record over time. In contrast, when the same query, with the source filter changed accordingly, is executed over provenance data associated with graduate course activities, the resulting graphs show a different structure. The absence of similarly long version chains indicates a substantially lower rate of modification, highlighting divergent patterns of data evolution across application domains. This contrast further demonstrates how provenance-based analysis can reveal domain-specific patterns in data usage and maintenance within an SoS.

Listing 5: Query `Q5` – Searching for nodes that have undergone any type of change in the context of leave of absence.

```
Q5: MATCH (n)-[*]-(conectado)
```

```
WHERE EXISTS {(n)-[:MENTION]-()}
AND n.fonte =~ 'siape/afastamento'
RETURN n, conectado
LIMIT 200
```

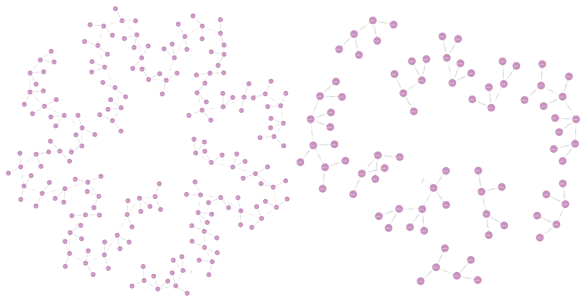

**Figure 8.** Results for Query Q5 – searching for nodes that have undergone any type of change, in the context of leave of absence (left) and graduate course activities (right).

To further deepen the investigation, we developed Query Q6, presented in Listing 6, with the goal of ranking academic departments according to the number of data deletions or invalidations imported into the RAD system. This query traverses the provenance graph to identify and aggregate invalidation events associated with departmental data, thereby providing a quantitative view of their concentration. The resulting analysis is particularly valuable for technical and administrative teams responsible for maintaining the SoS, as it shifts the focus from reactive, case-by-case error handling to a more systematic, proactive diagnostic approach. Rather than addressing individual data issues as they arise, the query enables the identification of departments that repeatedly generate invalid records, which may indicate underlying problems such as inconsistencies in constituent systems, misaligned data schemas, or recurring failures in the data transformation and import pipeline. By highlighting these recurrent sources of invalidated data, Query Q6 supports more informed decision-making regarding where to prioritize debugging, data quality interventions, and process improvements. Table 4 reports the top four departments identified by this query, illustrating how provenance-based analytics can guide targeted corrective actions within the RAD SoS.

Listing 6: Query Q6 – Search for nodes that have been invalidated (grouped by department).

```
Q6: MATCH (n)-[*]-(conectado)
    WHERE EXISTS {(n)-[:invalidated]->(n)}
    AND conectado.idOrgao <> "null"
    RETURN conectado.idOrgao AS orgao, count
        (*) AS quantidade
    ORDER BY quantidade DESC
```

**Table 4.** Results for Query Q6 – Search for nodes that have been invalidated, grouped by department.

| Department ID | Number of Invalidations |
|---|---|
| 473 | 254 |
| 614 | 98 |
| 984 | 78 |
| 10461 | 76 |

These analyses provide practical support for data governance and support development teams. With the provenance data in hand, teams can conduct deeper evaluations to determine whether the high recurrence of changes for a given individual reflects (i) expected behavior inherent to the business process, (ii) the need for better training of users responsible for data ingestion, validation, and maintenance, or (iii) evidence of systemic issues, either related to the usability of a constituent system that leads to frequent manual corrections or to flaws in its operational logic that may be generating incorrect data. Moreover, these insights allow teams to act proactively by focusing efforts on addressing root causes within the most affected sectors, thereby improving data quality at its source.

The evaluation showed the feasibility of the PROVGov-SoS approach for identifying patterns and performing analyses based on captured provenance data. The use of a provenance database implemented in Neo4j proved well-suited to this type of application, given the nature of the data. Graph-oriented databases provide native mechanisms for complex queries, such as path searches between nodes (as in Query Q3) and the detection of recurrent or atypical patterns, capabilities that are essential for anomaly detection in distributed systems such as SoS [Anuyah *et al.*, 2024]. As a result, the application of provenance improved data quality and strengthened governance mechanisms in the SoS context by providing an integrated, auditable, and trustworthy view of the data lifecycle.

## 6 Related Work

The importance of provenance data has been widely recognized across multiple research areas, including databases, data management, and distributed systems, where it has been shown to support transparency, traceability, and reproducibility [de Oliveira *et al.*, 2018]. In the context of SoS, provenance-related concepts have been adopted to support the monitoring, analysis, and understanding of complex system behavior, although often implicitly or indirectly.

Kritzinger *et al.* [2019] investigate SoS monitoring from a user-centered perspective, emphasizing the role of visualization in supporting system understanding. Their work, implemented through the ReMinds framework [Vierhauser *et al.*, 2016], provides multiple visualization techniques that enable stakeholders to monitor the state of an SoS and identify violations or abnormal behavior. The authors' user studies demonstrate that visual representations significantly improve users' ability to reason about system dynamics. While effective for operational monitoring, this approach primarily focuses on system states and violations rather than on the historical evolution of data across constituent systems.

Chreim *et al.* [2024] propose a multilevel hypergraph-based approach to address challenges related to SoS reconfiguration, coordination, and optimization. Their work highlights the need for continuous monitoring mechanisms to evaluate how individual constituent systems contribute to global SoS performance, particularly in environments characterized by uncertainty, heterogeneity, and frequent change. By defining metrics to evaluate system behavior, the proposed framework supports performance optimization and

adaptability, but does not explicitly address provenance data or governance concerns arising from data integration processes.

Kong *et al*. [2020] introduce a runtime monitoring approach that enables observation of SoS behavior without requiring modifications to the source code of constituent systems. This non-intrusive strategy is especially relevant in SoS settings, where systems are independently developed and managed. The approach provides real-time insights into operational behavior and supports compliance checking, but it remains focused on execution monitoring rather than on tracking the lifecycle and transformations of data artifacts.

# 7 Conclusion and Future Work

This manuscript uses provenance as a foundational backbone for data governance in SoS. To achieve this, we define a set of provenance entities, activities, and relationships grounded in the W3C PROV recommendation. Provenance records are represented both in PROV-JSON format and in a graph-oriented database. This standardized, dual representation enables SoS stakeholders to trace the complete derivation path of data items, including all transformations applied throughout their lifecycle. Provenance data explicitly links transactions performed by the constituent systems of the SoS, which are encapsulated in conceptual constructs called *bundles*.

Despite the benefits of provenance-centric monitoring for governance and auditing, its adoption in SoS environments is not trivial. Capturing, structuring, and managing provenance data in distributed and heterogeneous environments introduces complexity due to the heterogeneity of constituent systems, data formats, and processing pipelines. We evaluated `PROVGov-SoS` through a case study conducted in a real-world SoS operated by UFF. The results indicate that `PROVGov-SoS` effectively supports auditing activities, enabling, for instance, the detection of inconsistencies in registered pipelines within imported datasets, a recurring issue reported by end users of the evaluated SoS.

Several directions for future work can be listed. First, we plan to develop visualization and interactive navigation capabilities over provenance graphs to enable more intuitive analysis. Second, we plan to explore different levels of granularity in provenance capture, as well as alternative collection strategies that reduce performance overhead and intrusiveness. Third, we aim to design analytical mechanisms for automated anomaly detection and decision support, further strengthening the governance role of provenance data. Finally, we plan to evaluate `PROVGov-SoS` in additional application domains to assess its generality and adaptability.

# Declarations

The authors declare no competing interests.

## Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq (Grant 316510/2023-8), and FAPERJ (Grant 211.583/2019).

# References

Allen, M. D. *et al*. (2011). Provenance for collaboration: Detecting suspicious behaviors and assessing trust in information. In *CollaborateCom*, pages 342–351. IEEE.

Almeida, J., Braganholo, V., and de Oliveira, D. (2025). Governança de dados em sistemas-de-sistemas: Uma abordagem orientada à dados de proveniência. In *XL SBBD*, pages 182–195, Fortaleza. SBC. DOI: 10.5753/sbbd.2025.247059.

Almeida, R., Silva, W. M., *et al*. (2019). Managing data provenance for bioinformatics workflows using AProvBio. *IJCBDD*, 12(2):153–170.

Anuyah, S., Bolade, V., and Agbaakin, O. (2024). Understanding graph databases: a comprehensive tutorial and survey. *arXiv preprint arXiv:2411.09999*.

Boardman, J. and Sauser, B. (2006). System of systems - the meaning of of. In *2006 IEEE/SMC*.

Calabro, A., Daoudagh, S., *et al*. (2021). MENTORS: Monitoring environment for system of systems. In *WEBIST*, pages 291–298.

Cavalcante, E., Batista, T., *et al*. (2024). Looking back and forward: A retrospective and future directions on software engineering for systems-of-systems. *J. Softw. Evol. Process*, 36(10):e2697.

Chreim, A., Chen, Y., *et al*. (2024). Toward supervision of stochastic system of systems engineering: A multi-level hypergraph approach. *IEEE Access*, 12:162844–162860.

Cordeiro, F. *et al*. (2026). An accountability evaluation model for systems-of-information systems. *Journal of Systems and Software*, 232:112652.

Curry, E., Scerri, S., and Tuikka, T. (2022). *Data spaces: design, deployment and future directions*. Springer Nature.

Curry, E. and Sheth, A. (2018). Next-generation smart environments: From system of systems to data ecosystems. *IEEE Intelligent Systems*, 33(3):69–76.

de Oliveira, W. M. *et al*. (2018). Provenance analytics for workflow-based computational experiments: A survey. *ACM Comput. Surv.*, 51(3):1–25. DOI: 10.1145/3184900.

Ferreira, F. H. C., Nakagawa, E. Y., and dos Santos, R. P. (2023). Towards an understanding of reliability of software-intensive systems-of-systems. *Information and Software Technology*, 158:107186.

Fu, X., Wojak, A., Neagu, D., Ridley, M., and Travis, K. (2011). Data governance in predictive toxicology: A review. *Journal of cheminformatics*, 3:1–16.

Gammack, D. *et al*. (2016). Modelling provenance collection points and their impact on provenance graphs. In *IPAW*, pages 146–157. Springer.

Gil, Y. and Miles, S. (2013). PROV model primer. `https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/`.

Groth, P. and Moreau, L. (2013). PROV-Overview. `https://www.w3.org/TR/prov-overview/`.

Hasan, R., Sion, R., and Winslett, M. (2009). Preventing history forgery with secure provenance. *ACM Transactions on Storage (TOS)*, 5(4):1–43.

Herschel, M. *et al*. (2017). A survey on provenance: What for? What form? What from? *VLDB J.*, 26(6):881–906.

Huynh, T. D., Jewell, M. O., *et al*. (2013). PROV-JSON serialization. `https://www.w3.org/submissions/2013/SUBM-prov-json-20130424/`.

Kiczales, G. *et al*. (2001). An overview of AspectJ. In *ECOOP*, pages 327–354. Springer.

Kong, S., Lu, M., Li, L., and Gao, L. (2020). Runtime monitoring of software execution trace: Method and tools. *IEEE Access*, 8:114020–114036.

Kritzinger, L. M., Krismayer, T., *et al*. (2019). A user study on the usefulness of visualization support for requirements monitoring. In *VISSOFT*, pages 56–66. IEEE.

Lis, D. and Otto, B. (2020). Data governance in data ecosystems–insights from organizations. In *Americas Conference on Information Systems (AMCIS)*.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Moreau, L. (2013). ProvToolbox. Java library to create and convert W3C PROV data model representations. `https://lucmoreau.github.io/ProvToolbox/`.

Moreau, L., Batlajery, B. V., Huynh, T. D., Michaelides, D., and Packer, H. (2017). A templating system to generate provenance. *IEEE Trans. on Soft. Eng.*, 44(2):103–121.

Moreau, L. and Lebo, T. (2013). PROV-LINKS. `https://www.w3.org/TR/2013/NOTE-prov-links-20130430/`.

Moreau, L. and Missier, P. (2013). PROV-DM: The PROV data model. `https://www.w3.org/TR/2013/REC-prov-dm-20130430/`.

Neo4j (2025). Neo4j graph database. `https://neo4j.com/product/neo4j-graph-database`. Accessed on 12 Mar. 2025.

Sultana, S. and Bertino, E. (2012). A comprehensive model for provenance. In *IPAW*, pages 243–245.

Teixeira, P. G., Lebtag, B. G. A., *et al*. (2023). Externalizing requirements for achieving operational independence in systems-of-systems: A mapping study. In *SoSe*, pages 1–6.

Vierhauser, M., Rabiser, R., *et al*. (2016). ReMinds: A flexible runtime monitoring framework for systems of systems. *JSS*, 112:123–136.

Wercelens, P., da Silva, W., *et al*. (2019). Bioinformatics workflows with NoSQL database in cloud computing. *Evolutionary Bioinformatics*, 15:1176934319889974.