SousLeSens - A Comprehensive Suite for the Industrial Practice of Semantic Knowledge Graphs

Claude Fauconnet[1(✉)], Jean-Charles Leclerc[2], Arkopaul Sarkar[3], and Mohamed Hedi Karray[4]

[1] SousLeSens, Paris, France
claude.fauconnet@gmail.com
[2] TotalEnergies, Paris, France
[3] Knowledge Graph Alliance, Brussels, Belgium
[4] Université de Technologie de Tarbes, Tarbes, France

Abstract. Over recent decades, the advancement of semantic web technologies has underscored the increasing importance of tools dedicated to developing and managing the foundational components of the semantic web stack. Addressing the evolving needs, a variety of tools have emerged from the research and development projects from academia as well as commercial software vendors. These tools offer a diverse range of services tailored to the management of various aspects of semantic knowledge graphs. Despite this proliferation, feedback from stakeholders involved in public and privately funded projects has highlighted notable shortcomings in existing tools. These gaps become evident in two key areas: firstly, the user experience struggles to scale up to meet industrial-level data practices and knowledge engineering methodologies. Secondly, a lack of interoperability and compatibility among the existing task-specific tools leads to elevated costs and efforts. This paper introduces a novel semantic knowledge management ecosystem embodied in a suite of tools collectively known as 'SousLeSens'. Unlike its counterparts, SLS not only provides comprehensive coverage of typical knowledge engineering tasks while adhering to best practices for ensuring quality but also boasts a purely visual (no to minimum-code) interface. This feature is particularly well-suited for handling large-scale, industry-grade semantic data models. The paper delves into the establishment of requirements for knowledge engineering tools and services, derived from recent stakeholder surveys. It proceeds to present the SLS toolkit, elucidating its architecture and operational protocols. Finally, the paper validates the toolkit's capabilities by comparing it with existing tools against predefined requirements and illustrating various use cases.

Keywords: ontology editor · knowledge graph · SousLeSens · protege · semantics

A. Meroño Peñuela et al. (Eds.): ESWC 2024, LNCS 14665, pp. 162–177, 2024.
https://doi.org/10.1007/978-3-031-60635-9_10

1 Introduction

In the evolving landscape of semantic web engineering tools, there exists a pressing need for solutions that can seamlessly navigate the intricacies of constructing and managing the foundational elements of the semantic web stack. Despite the emergence of various tools from both academic and commercial spheres, feedback from stakeholders has consistently highlighted notable shortcomings [1]. These deficiencies manifest themselves primarily in the struggle to scale user experiences to meet industrial-level data practices and the lack of interoperability among existing, task-specific tools, resulting in increased costs and efforts.

Against this backdrop, this paper introduces 'SousLeSens' (SLS), a collaborative semantic knowledge management (KM) ecosystem specifically designed to address the identified gaps in the current tool landscape. Unlike its counterparts, SLS not only comprehensively covers many typical knowledge engineering tasks, but also lets users to use a visually driven, minimal-code interface tailored for handling large-scale, industry-grade semantic data models. The Semantic Labeling System (SLS) was developed as part of the TotalEnergies Semantic Framework (TSF) project [7] to address the need for a set of tools that comply with the best practices in knowledge engineering while also being user-friendly and easy to learn for professionals in the industry.

In this paper, we first identified the requirements of a comprehensive KM tool by exploring both the past literature and industrial expectations in Sect. 2. Afterwards, in Sect. 3, the methodological principle and technical architecture of SLS is presented. In Sect. 4, three industrial use cases are presented to demonstrate the capabilities of SLS. Finally, a comparative analysis of SLS against other similar products is presented in Sect. 5.

2 State-of-the-art

The realm of KM tools is diverse, mirroring the variety of methods, languages, and formats for knowledge management. The aim of the section is however not to analyse these tools individually but to identify the requirements for a comprehensive KM tool from the past surveys and industrial feedback. SLS gathered these requirements from two sources: 1) review of the existing KM tools and feedback from users of these tools as described in Sect. 2.1 and 2) industrial needs identified through various initiatives and projects as described in Sect. 2.2.

2.1 Existing Knowledge Engineering Tools

In ontology engineering, while many tools address specific phases, only a few provide comprehensive solutions. A recent EU Horizon2020 OntoCommons survey [11] categorized tools, including competency questions for requirement specification, Linked Open Vocabulary (LOV) and IndustryPortal for ontology reuse and publication, Chowlk and Menthor editor for conceptual modelling, various reasoners, e.g., Pellet, Hermit, Fact++, and consistency checkers, e.g., OOPS!,

all of which has targeted focus in some of the activities along the ontology engineering lifecycle. In contrast, a comprehensive KM tool facilitates many KM activities, such as ontology implementation, covering drafting, source editing, evaluation, ontology matching, maintenance, issue management, and usage as well as managing large data volumes. Past surveys [3,8,9,12,13] on ontology editing tools mention comprehensive tools such as SWOOP, Protégé, WebODE, TopBraid Composer, OntoPic Fluent editor, OWLGrEd, OntoEdit, Anzo platform, Cameo modeller, and PoolParty, which comply to the most popular and de facto standards for knowledge engineering, such as OWL and RDF. It can also be observed that ontology editing and knowledge graph management are distinct activities, and the mentioned tools have varied support for both. Graph database systems such as Neo4J, GraphDB, AllegroGraph, Stardog, and RDFox often include dedicated data transformation tools along with a triplestore as their main offering.

The primary aspects of evaluating these tools are their features and usability; the former covers the functionalities, architecture, and support to various activities of ontology engineering and the latter covers various types of user experience. Past evaluations [5,9,11,13] mention many common features that ontology and knowledge graph editors typically support. These are expressivity, language, modularisation and reusing, consistency check, reasoning and inference, merging, mapping, encoding and storage, visualisation, data transformation, query, and third-party support. As Islam and Sheikh [6] noted, there is no standard approach to evaluate the usability of ontology editors. Still, some of the usability criteria that can be found in the small number of past surveys are user experience and visual abstraction, collaboration, multi-lingual support, performance and scalability, methodological guidance, practicality, help and documentation, and exception handling. Easy access and deployment of the tool, along with its openness and extensibility also play a big role in ensuring its usability. The OntoCommons survey [11] collected 77 responses from the major knowledge engineering tool developers using online questionnaires that include a list of common requirements for knowledge engineering tools: collaboration of multiple stakeholders, visualisation, debugging, validation, quality assurance and analytics, API support, OWL support, ontology import, user-friendliness, connected ontologies, tool integration, modularisation, search and reuse ontology, ontology correlations, ontology deployment and document generation, and ontology conceptualisation, understandability of GUI labels. Although the basic features for a comprehensive ontology editor can be found by surveying the existing tools, the development of SLS prioritised those features that can fulfil the requirements gathered from community surveys as mentioned above.

2.2 Industrial Feedback

The introduction of a new process and working method requires the support of new types of tools, as described in the OntoCommons roadmap [1]. The Knowledge Translator report [4] emphasizes the need for easy on-boarding and training for the transition from a human-centric paper standard to a data-centric and

model-centric approach, exemplified by initiatives such as CFIHOS[1], allowing the analysis of digital requirements, source coherence and internal knowledge sharing with external partners.

In 2020 and 2021, TotalEnergies collaborated with stakeholders of the International Association of Oil & Gas Producers[2] (IOGP) and the World Economic Forum (WEF) to formalize the conditions for better data exchanges in the world's energy industries. It is evident in the industrial outlook that sharing reliable information requires addressing the big data challenges (volume, value, variety, velocity, and veracity) and a common way of structuring and exchanging knowledge. The IOGP working group identified not only the fundamentals for common semantics for data structuring and exchange but also a set of recommendations for a suitable KM tool, including web (HTTP protocol) compliance, following the motto: 'Everything is available to everyone and everywhere at any time', open code practice, publishing vocabulary definitions with IRI, obligatory common schema definition, compliance to RDF/OWL formalism, and interactive functional web-applications for query/extraction of information. The industrial stakeholders of the above-mentioned initiatives realised that existing tools for supporting these semantic web requirements often have a high learning curve, necessitating a new kind of tool for easy onboarding and upskilling. In the following sections, we describe how SLS addresses these requirements from the perspective of its core architectures and various tools dedicated to critical KM tasks.

3 Specification of SLS

Developed within the TSF project, SousLeSens[3] (SLS) is a suite of web tools designed to facilitate the creation of semantic knowledge graphs using ontologies. It functions as an adaptable set of tools and a cohesive suite that aims to demystify the complexities associated with the Semantic Web. By offering a user-friendly interface, SLS makes it possible for individuals, including those in industrial sectors, who may not specialize in these technologies, to leverage Semantic Web capabilities. This accessibility is crucial for meeting industrial requirements, particularly in managing data and knowledge, developing digital twins, and supporting decision-making processes.

3.1 Methodological Foundations

The initial requirements for a knowledge graph management tool were chartered by the TSF project in January 2021, as shown in Fig. 1. The SLS development team facilitated the creation of initial prototypes that were periodically reviewed to progressively develop the versatile TSF tool suite. Over time, the support

[1] https://www.jip36-cfihos.org/cfihos-standards/.
[2] https://www.iogp.org/.
[3] https://sls.kg-alliance.org/.

for W3C-recommended formats in a cohesive data management framework provided a flexible solution to TSF for predefined data interoperability in several use cases. This allowed the immediate reuse of internal technical references, expressed in SKOS, by converting them into RDF and OWL formats within TSF. The code and interface integrated methodological governance, ensuring conformity to common structuring principles. This orchestration aligned company activities with semantic knowledge graphs, serving as domain ontologies for reference data domains.

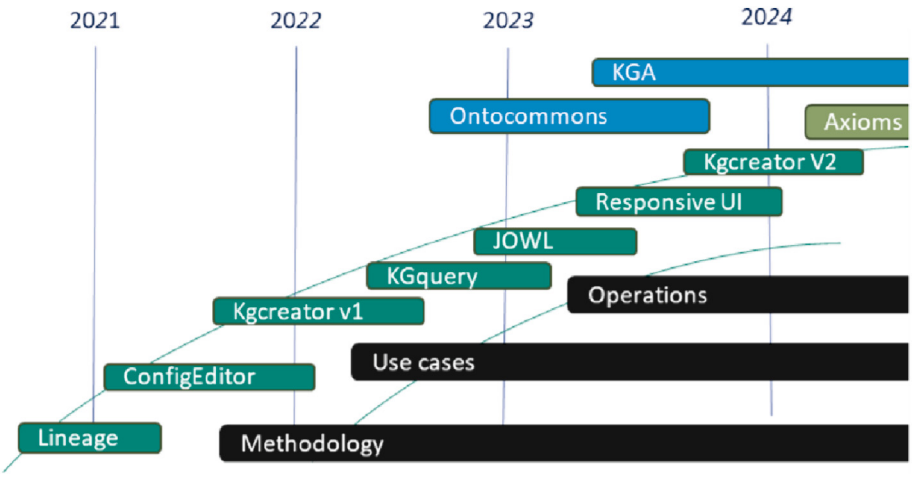

Fig. 1. Timeline of SousLeSens evolution.

The SLS toolset (purposes of various tools under SLS are summarised in Fig. 2) is designed to facilitate the continuous onboarding and upskilling of internal members of TotalEnergies by mirroring the exploration process of the semantic web for newcomers. However, the responsive and interactive design principles of SLS are poised to benefit all users in learning semantic web technologies, addressing common inquiries throughout the learning curve. For example: 1) **Discovering, Comparing, and Choosing Ontologies**: SLS's visualization and manipulation of graphs under the tool Lineage simplifies understanding essential ontology notions, allowing users to explore, mix, and compare different ontologies with visually driven capabilities. 2) **Building Ontology**: SLS provides a user-friendly interface for creating classes, establishing semantic relationships, and respecting constraints. The tool automates SPARQL queries within triplestore databases based on the type of ontology being built. 3) **Associating an Ontology with Data**: SLS aids in generating mappings between ontology objects and data model elements, especially for tabular data. The tool allows users to express, save, and graphically represent mappings for validating their relevance. 4) **Building a Semantic Knowledge Graph (SKG)**: SLS transforms

data from relational databases or CSV files into a knowledge graph using pre-established mappings. The tool is evolving to support the RML standard in its roadmap. 5) **Using SKG**: SLS supports two distinct uses of semantic knowledge graphs, enabling queries through APIs or exploration mode. In exploration mode, users can interact with graphical representations, simplifying queries based on ontology concepts. 6) **Collaborative Work**: SLS's internet-native architecture promotes collaborative work by providing secure user management, data source control, and read-write rights. Metadata addition helps trace modifications and qualify their status. 7) **Combining Tools**: SLS's layered, modular architecture, combined with technical standards, allows it to offer integrated services and act as a client or third-party service provider, facilitating seamless combination with other tools.

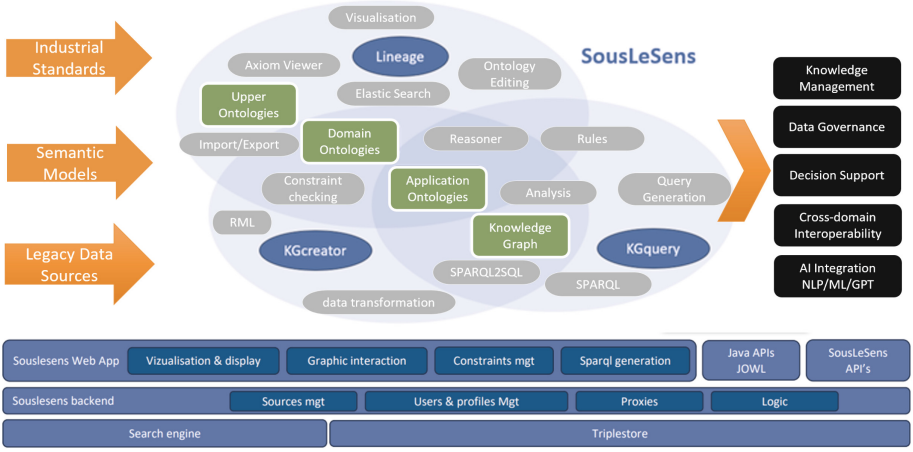

Fig. 2. Overview of SLS mapping different usage of its tools (Lineage, KGCreator and KGQuery) along with the technical stack (at the bottom).

In this context, the technical architecture of SLS was designed based on the following pillars: 1) Rigorous management of semantics included in upper-level ontologies, encoding the constraints in reference languages, such as RDF, RDFS, and OWL; 2) A graphical visual representation, utilizing nodes and edges as a global metaphor for an RDF graph, captures the expressive richness of the triples graph while ensuring understandability; filtering and navigation capabilities within the network elements, coupled with the graph spatialization engine, aid users in comprehending complexity, coherence, inconsistency, and multiple relationships by accommodating the limitations of simultaneous information apprehension by the human eye. 3) Alternative mode of interaction and manipulation based on the generalization of a three-step process: translation of user actions into SPARQL queries, execution of queries on a triplestore internal to SLS or other in reading as well as in writing modes, and representation of query results either in graph mode or in table mode.

3.2 Software Architecture

The organic design of SLS evolved by gradually aggregating functionalities around the central core concept of visualizing and manipulating graphs. Technical solutions, implementing some original codebases, reflect this development. Notably, the client bears the majority of processing, enabling rich interactions with the graph without frequent server calls-a key for the performance of SLS. The choice of untyped JavaScript, used on both the client and server sides (node.js) was based on its agility in seamlessly moving code between them. TypeScript, a typed form of JavaScript, was not used due to its heavy constraints on typing, which were deemed impractical for our frequently evolving code needs. The architecture of SLS is centered on its NodeJs web server which essentially acts as a flow distributor between its different peripheral components through secure APIs, which gives it the character of an open technical platform as shown in Fig. 3.

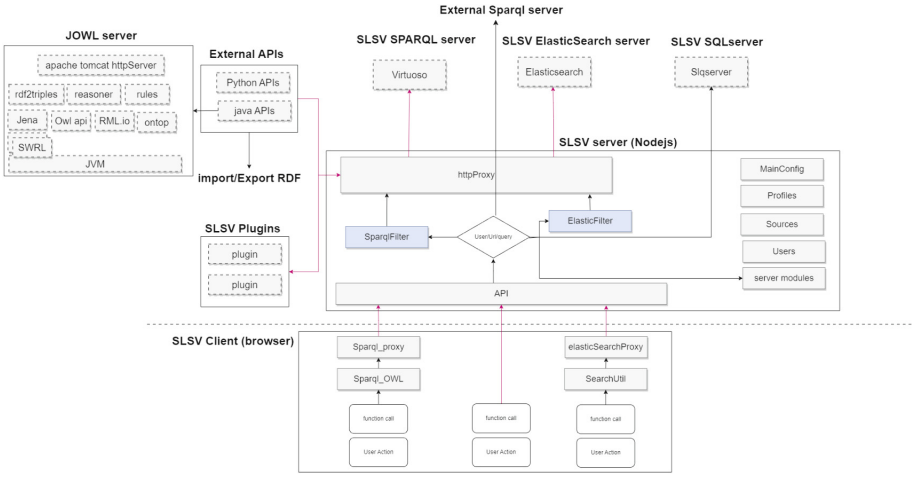

Fig. 3. Technical architecture of SLS

In the following, various core components of SLS are described based on the architecture.

1. **Security and access management**: Each user of the SLS web interface or API is authenticated. Several authentication systems are supported but only one is active within an instance. In addition, requests made to the internal TripleStore or the search engine are filtered according to the user's rights.
2. **Administration and configuration module**: As a collaborative application, SLS rigorously manages the access of different users to different resources and allows very flexible management of access to resources ranging from public sources visible to all to private sources, having restricted write rights for

only a few users. For this purpose, SLS has developed an administrative management module composed of the objects: Users, having different roles and permissions, Source, identifying different KGs, and Profiles, maintaining access rights by mutual association.

3. **Storage layer**: The RDF graphs manipulated by SLS are stored either in its internal triplestore (Virtuoso) or in other triple stores referenced and accessible in SLS depending on the configuration.
4. **Application server layer**: This layer powered by Node.js communicates via APIs in HTTP(S) mode with the user interface and with external clients while respecting the access rules applicable to each client. Requesting APIs triggers various calls, e.g., execution of queries on SPARQL or SQL servers, specific internal commands, and access to third-party APIs (python, java, etc.) depending on the case, always checked in terms of access rights.
5. **The web client layer (browser)** locally manages most of the actions carried out by the user (business logic) and limits the interactions with the server layer for reading data or specific processing. For example, the construction of SPARQL or SQL queries in response to user actions is done dynamically on the client side, unlike the practices of many other applications. SLS incorporates common UI components, such as the *SourceSelector*, for user-authorized source selection, the *jsTreeWidget*, facilitating tree representation and node control actions, and the *NodeInfos* dialogue, displaying predicates of a node, including the ability to add or remove predicates based on user rights and experimental visualization of axioms. These components streamline various interactions within the interface across multiple contexts.
6. **Visual representation of knowledge graphs** stands out in SLS by prioritizing graph drawing and manipulation, a central feature that distinguishes it from similar tools. It utilizes the vis.js[4] JavaScript library, offering a robust API for graph visualization and interaction. Vis.js provides various spatialization algorithms, optimizing graph representation based on factors like the number of relationships each node has, with central nodes having more connections. For example, the configuration for the BarnesHut layout algorithm, which is a quadtree-based gravity model and the fastest non-hierarchical solver. Graph stabilization is achieved through successive iterations, which can be manually interrupted by users. Before graph drawing in SLS, SPARQL queries select triples from the store, transformed into nodes and edges using vis.js API. As RDF graphs in OWL ontologies or SKGs can be intricate, different triplet filtering strategies, detailed in Sect. 3.3, are implemented for clear and useful representations. Once represented on the screen it is possible to interact via the mouse with the nodes and edges of the graph in order either to obtain information to navigate in the graph or to create new predicates.
7. **Plugins**: SLS has a system of plugins allowing users to create specific applications, including their web app, that can be used like other tools and take advantage of the entire environment. The open architecture of SLS supports

[4] https://github.com/visjs.

interaction with external libraries, such as JOWL[5] -a Java web application with REST APIs developed in 2023. This integration allows the execution of OWLAPI and SWRL (reasoner and rules) Java methods within SLS. Additionally, an integrated Python API enables users to choose import/export formats for triplestores and, soon, to interact with SLS-referenced graphs from a Jupyter or other Python environment while respecting user access rights.

8. **Deployment**: SLS's technical environment is deployed through diverse Docker images, including a node.js web server with SLS code and APIs[6], a Virtuoso triplestore, an ElasticSearch search engine, an SQL server (and soon Postgres SQL) database, a Python environment, and optional third-party components and specific SLS plugins.SLS can be downloaded freely from its GitHub repository[7] under an MIT license. SLS releases follow a standard versioning system. Supporting the easy onboarding, a documentation website[8] (currently containing 'quick start' and many video tutorials) is also made available.

3.3 Tools in SLS

Lineage in SLS facilitates the simultaneous representation and manipulation of graph representations of different ontologies. It enables the selection, editing, evaluation, and comparison of ontologies, allowing users to generate overviews or details for specific tasks. Key features include selecting source ontologies, transforming triples into interactive graph drawings, random navigation on the whiteboard, creation of classes and individuals, visualization of predicates and axioms, drag-and-drop relationship creation, advanced SPARQL query generation, representation of label similarities, label-based graph searches, and object properties selection and presentation. The UI components of Lineage displaying an example ontology shown in Fig. 4. Lineage being a versatile ontology editor, it supports constraints on class and properties in a variety of expressivity, from simple subsumption and transitivity suitable of RDF graph to domain, range, existential, and universal constraints to support full Description Logic (DL) formalism. The graph-drawing complexity is balanced by providing the option to export information displayed in tabular form (CSV) for users accustomed to spreadsheets and relational databases. As SLS is integrated into IndustryPortal [2], users may search for a FAIR ontology from the portal for easy reuse.

KGCreator can be used to simplify the transform complex industrial data into a semantic knowledge graph by serving as a user-friendly alternative to RML or R2ML-based tools. Its straightforward structure, depicted in Fig. 5, proves

[5] https://github.com/souslesens/jowl.
[6] https://sls.kg-alliance.org/api/v1/.
[7] https://github.com/souslesens/souslesensVocables.
[8] http://souslesens.org/index.php/tools/.

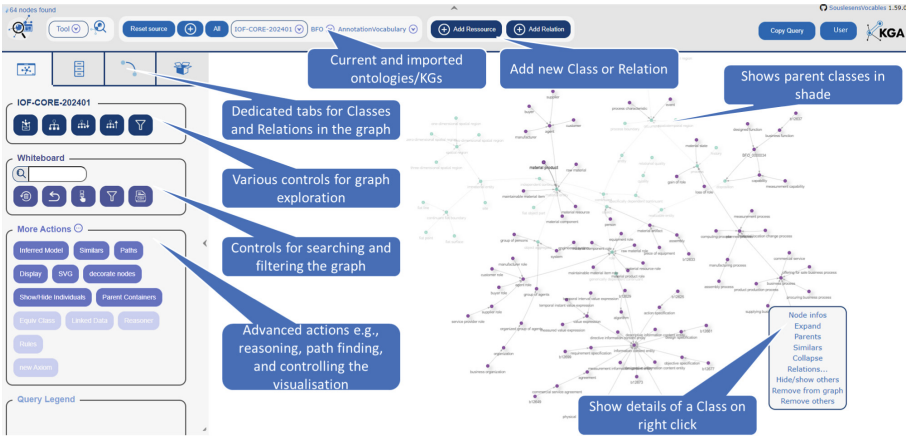

Fig. 4. Lineage tool displaying an ontology with upper ontologies in its import and various UI controls for users to manage, view and edit the ontology.

effective for even non-specialist operators. While KGCreator facilitates interoperability with standards by easily translating to RML, it offers features such as access to tabular data schemas, graphical exhibition of the ontology for column/class associations, detailed mapping for enriched associations, visualization of mappings' completeness and consistency, and simulation of generated triplets for previewing and corrections before storing in the triple store. Additionally, KGCreator supports functions and transformations for data line processing and table mapping, ensuring flexibility in adjusting data structures and avoiding errors. Resource identifiers in KGCreator can take various forms, including real URIs and blank nodes associated with random numbers. Triplet Generation Engine, a backend processing program, utilizes mappings and associated data to generate the knowledge graph in the triple store, offering an API for regenerating the entire sub-graph from previous mapping.

KGQuery allows the use of SKG in exploration mode without having to master the SPARQL language and know the complex structure of the graph, the KGQuery tool offers an interactive query-able abstraction of the KG, which allows users to automatically generate queries by graphically selecting the classes, as shown in Fig. 6. KGQuery is also able to apply filters and perform queries on multiple graphs. The visual manipulation of knowledge graphs through queries relies on correct data typing and predicate construction in alignment with ontology constraints during graph construction. The queryable abstraction of the KG is derived from the actual classes and predicates present in the graph, maintaining fidelity. The graph traversal algorithm, based on shortest path calculations using adjacency matrices, facilitates automatic SPARQL query construction, with results presented in tabular form - following the usual expectations of data users - along with group queries and set operations.

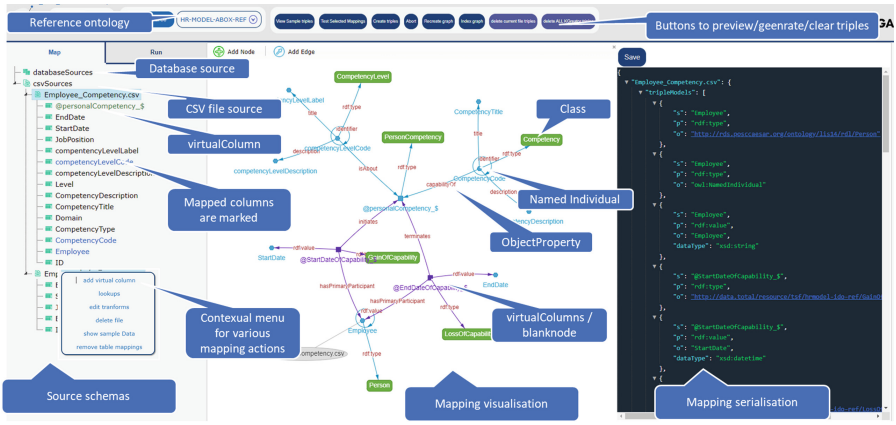

Fig. 5. KGCreator interface to create and manage mapping from heterogeneous sources (e.g., database, CSV) to ontology and generate triples using these mappings.

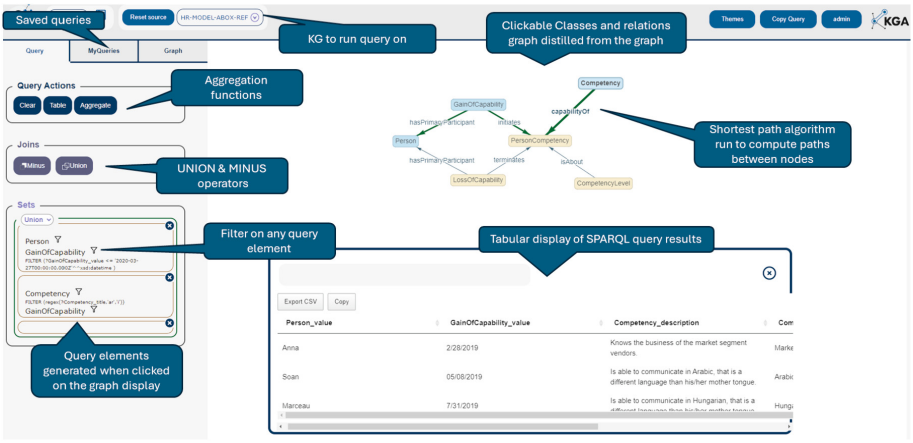

Fig. 6. Example of automatic SPARQL query generation using the graphical interface of KGQuery.

4 Validation

In this section, we first provide three industrial use cases to demonstrate the practical use of SLS in semantic modelling and exploitation of data. Furthermore, we discussed the essential capabilities of SLS in comparison to other tools against the scope, coverage, and requirements identified in Sect. 2.

4.1 Case Studies

As a fundamental tool, SLS supports many data activities in TSF. The following sections present three use cases where SLS plays major roles in standardising

industrial data for cross-organisation exchange, supporting business decisions, and valorising geoscience data by integrating AI tools for oil discovery.

Lineage for Reusing Semantic Standards for Industrial Data. OntoCommons Roadmap [1] identifies industrial needs for enhanced data standards, such as logic-based formats, integration of cross-cutting standards, and multi-domain stakeholder involvement. The development of the CFIHOS RDL[9] prototype during the READI JIP[10] in Norway (2018-2022), aligned with the Industrial Domain Ontology[11] (IDO) structure, exemplifies progress in this area, fostering major industry adoption under ISO standardization. The lack of interoperability among standardized ontologies and the demand for clear presentations of CFIHOS RDL values emphasize the importance of semantization, providing benefits like easier accessibility and flexibility for extensions. Lineage proves valuable in facilitating collaboration in ontology setup and validation phases through its filtering capabilities and visual representations, aiding in the detection of defaults. TotalEnergies leverages Lineage for digital technical data interoperability and NLP processing for noise reduction and enhancing machine learning confidence in used labels, based on application ontologies in the Geosciences and Reservoir Well domain, which are built following ISO 23726 standard family for ontology and ISO/IEC 81346 standard for managing digital data. Lineage's key applications include versatile source comparisons, collaborative work on a writable semantic layer, ensuring coherence in the domain knowledge graph, developing a collaborative CFIHOS vocabulary in IDO through SLS, and facilitating easy integration of business applications with reference domain ontologies across the organization.

KGCreator and KGquery for Business Decision Support. KGCreator and KGQuery play a pivotal role in TotalEnergies' business decision support, particularly in addressing challenges during major installation shutdowns, as showcased in the Dalia asset's life cycle extension project. In this project, SLS, equipped with KGCreator and KGQuery, optimized planning sequences by considering criteria from various Information Systems like SAP, Document Management System, and Primavera planning tool. The digital approach transformed the project by facilitating real-time insights across disciplines and systems, with a key strength in linking each Work Breakdown Structure item with pertinent data sources for efficient categorization and filtering. The application of KGCreator and KGQuery within SLS resulted in streamlined decision making, efficient change management, simplified project reviews, automated compliance checks, and impact analysis, bringing tangible benefits such as reduced standby risks, improved operational efficiency, comprehensive documentation, and continuity preservation.

[9] https://www.jip36-cfihos.org/cfihos-standards/.
[10] https://readi-jip.org/.
[11] https://www.iso.org/standard/87560.html.

SLS for AI-Powered Oil Discovery and Ocean Exploration. TotalEnergies' pilot study integrates SLS and advanced AI tools like NLP, GPT, and LLM with the well domain ontology (see Sect. 4.1) to revolutionize oil discovery and ocean exploration. NLP addresses challenges in managing complex informational portfolios, while GPT enriches the knowledge graph with additional facts. The experiment focuses on SLS for metadata management, entity reconciliation, and acculturation support to enhance data value and mitigate challenges in processing data labels and business objects across contexts. The proposed demonstrator, leveraging diverse databases while preserving semantics, aims to rationalize source databases, enhance data usage, and virtualise data access. For generative AI projects, SLS provides a reference vocabulary-based semantic guidance and automated verification by reasoning as an essential service to address interoperability challenges and inaccurate data. An SLS python API facilitates easy access to SLS RDF graphs and third-party AI tool libraries in the Jupiter python environment. In conclusion, the modular elements of SLS offer a comprehensive approach to combine generative AI and ontologies for data and industrial engineering document valorisation. The proposed benchmarking initiative will emphasize knowledge graphs' role in enhancing LLMs' accuracy for question answering on enterprise SQL databases, as posited by Sequeda et al. [10].

5 Comparative Analysis

SLS aims to be one of the most widely used modern ontology editors that supports OWL and RDF. It therefore includes tools that span various stages of ontology engineering, notably Lineage for ontology implementation and KGCreator and KGQuery for ontology maintenance and use. This combination of features within a single ecosystem is uncommon among other tool systems, with the exception of TopBraid and PoolParty. Notably, Protégé, in its standard offering, does not include knowledge graph management. Various features of Lineage, such as the visual whiteboard for conceptual to formal drafting of ontology, mapping, matching and comparison facility for handling ontology import stack in a modular fashion, along with in-place guidance to help users build coherent ontologies typical to the other ontology editors, SLS stands out from other editors, e.g., Protege, TopBraid Composer, WebODE, SWOOP, OntoPic and Fluent with its unique feature of purely visual, no to minimal code environment for ontology editing - although many of them provide ad hoc graph visualisation. This purely visual design becomes a more distinguishable feature of SLS in terms of KGCreator and KGQuery, considering no comprehensive knowledge management system contains such a novel interface. Collaboration is a primary feature of SLS and covers every knowledge engineering activity under its support. Although collaboration is also provided by some of the other editors, e.g., WebProtege, TopBraid, Anzo, SWOOP, and OntoEdit, the user experience for these editors will be vastly different in comparison to SLS's multi-user simultaneous manipulation of visual knowledge graphs. Lastly, in past and ongoing projects, SLS's visual abstraction of many complexities of end-to-end knowledge engineering has proven to be particularly suitable for industrial users in

Table 1. Comparison of tools. Columns: A - coverage of KM activities (e - editing ontology, r - reasoning KG, v - visualizing KG, p - population of data in KG, q - query, u - use), B - modularisation/import resolution, C - search and reuse ontology, D - Consistency check, E - reasoning/inference, F - rules, G - mapping, H - model repository, I - conceptualisation, J - data transformation, K - visual query, L - collaboration, M - methodological guide, N - User-friendliness (no-code environment, ease of learning, third-party integration)

Tools	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Protégé	e,r,v	**Y**	N	**Y**	**Y**	**Y**	N	N	N	N	N	N	**Y**	N
WebProtege	e,r,v	**Y**	N	**Y**	N	**Y**	N	**Y**	N	N	N	**Y**	**Y**	N
OntoPic	p,q,v	N	N	N	N	N	**Y**	N	N	**Y**	N	**Y**	N	**Y**
TopBraid	e,r,v,p,q	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	N	N	**Y**	N	**Y**	N	N
Fluent	e,r,v	**Y**	N	**Y**	**Y**	**Y**	**Y**	N	N	N	N	N	**Y**	N
OWLGrEd	e,v	**Y**	**Y**	N	N	N	N	**Y**	**Y**	N	N	N	N	N
OntoEdit	e,r	**Y**	**Y**	**Y**	**Y**	N	N	N	N	N	N	N	N	N
Anzo	p,q,v,u	**Y**	**Y**	**Y**	N	N	**Y**	**Y**	N	**Y**	N	**Y**	N	**Y**
PoolParty	e,v,p,u	**Y**	**Y**	N	**Y**	N	**Y**	**Y**	**Y**	**Y**	N	**Y**	N	**Y**
SLS	e,r,v,p,q,u	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

co-modelling, co-creating and co-analysing vast amounts of data for common understanding, system-wide integration and exchange, supporting practical benefits in decision support driven by SLS's easy integration with other AI-based tools.

In Table 1, we provided a comparative analysis of the SLS against the requirements identified in Sect. 2, considering only the default configuration, i.e., without any plug-in for every product. As described in Sect. 2, the requirements are derived from two sources: 1) user surveys on existing tools, and 2) industrial feedback. The tools presented in Table 1 include many other features, which are sometimes unique to them. These features are not included for them not being part of baseline requirements. Therefore, the comparison matrix in Table 1 is not an evaluation of the overall efficacy of the tools, as each one of them is built with its purpose and serves its users with some unique features. Moreover, the comparison is conducted by the authors conjointly with some industrial stakeholders and users, which lacks the critical mass of a typical survey. We want to conduct a wide survey to compare several existing tools against SLS in the future.

SLS being more focused on ease of use for industrial operators, it does not yet provide some of the more rigorous language supports that are available in other tools. e.g., Protégé. These languages and grammars, including Manchester Syntax Format for writing complex axioms, SWRL, DataLog, or SPIN rules for rule-based inference, and SHACL for data validation. The mappings created by KGCreator is also not exchangeable in standardised formats such as RML and SSSOM. We plan to include these features in the future versions of SLS.

6 Conclusion

This paper introduced SousLeSens (SLS) as a solution that addresses practical industrial needs and facilitates easy onboarding of industries into semantic web technologies, especially at a time when traditional data technologies are no longer sufficient to manage present and future data challenges. SLS, born out of industry necessity, now aspires to foster broad adoption of semantic KM practices. The future work will focus on key areas for improvement and expansion, including collaboration support, enhanced graphic axiom creation tools, specific ontology creation tools, reasoner capabilities for KGcreator mappings validation, refining the virtual SQL query system in KGquery, and reshaping the user interface for better ergonomics and a responsive design. Looking ahead, SLS envisions growing its user community to drive its development roadmap, along with fostering collaboration between academic and industrial perspectives through its open-source initiatives. The establishment of a dedicated working group within the Knowledge Graph Alliance[12] (KGA) will play a pivotal role in achieving these goals and ensuring the continuous evolution of SLS to meet the evolving needs of its user base[13].

Acknowledgment. This work was funded by the EU H2020 Ontology-driven Data Documentation for Industry Commons (OntoCommons) under grant agreement no. 958371 and was supported by TotalEnergies and SousLeSens. We would like to thank Karim Ounnoughi from Akkodis Group, Xavier Garnier and Nicolas Chauvat, from Logilab, Amine Karoui from UTTOP, and Pierre Jallais from TotalEnergies for their valuable technical contributions to this work.

References

1. Adamovic, N., et al.: Ontocommons roadmap v1, January 2023. https://doi.org/10.5281/zenodo.7544509
2. Amdouni, E., Sarkar, A., Jonquet, C., Karray, M.H.: IndustryPortal: a common repository for FAIR ontologies in industry 4.0. In: The 22nd International Semantic Web Conference. Athens, Greece (2023)
3. Buraga, S.C., Cojocaru, L., Nichifor, O.C.: Survey on web ontology editing tools. Trans. Autom. Control Comput. Sci. **NN**(ZZ), 1–6 (2006)
4. Goldbeck, G., et al.: The Translator in Knowledge Management for Innovation - A Semantic Vocation of Value to Industry (2023)
5. Gyrard, A., Datta, S.K., Bonnet, C.: A survey and analysis of ontology-based software tools for semantic interoperability in IoT and WoT landscapes. In: IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings, vol. 2018-Janua, pp. 86–91 (2018). https://doi.org/10.1109/WF-IoT.2018.8355091
6. Islam, N., Sheikh, G.S.: A review of techniques for ontology editor evaluation. J. Inf. Commun. Technol. **9**(2), 104–114 (2015)

[12] https://www.kg-alliance.org/.
[13] https://ontocommons.eu.

7. Leclerc, J.C., Tetard, G., Keraron, Y., Fauconnet, C.: Use of ontologies to structure and manage digital technical data of industrial assets: first steps towards an ecology of knowledge in multi-energies industry. In: CEUR Workshop Proceedings, vol. 3240 (2022)
8. Rahamatullah Khondoker, M., Mueller, P.: Comparing ontology development tools based on an online survey. In: WCE 2010 - World Congress on Engineering 2010, vol. 1, pp. 188–192 (2010)
9. Rastogi, N., Verma, P., Kumar, P.: Analyzing ontology editing tools for effective semantic information retrieval. Int. J. Eng. Sci. Res. Technol. **6**(5), 40–47 (2017). http://www.ijesrt.com
10. Sequeda, J., Allemang, D.T., Jacob, B.: A Benchmark to Understand the Role of Knowledge Graphs on Large Language Model's Accuracy for Question Answering on Enterprise SQL Databases. ArXiv **abs/2311.0** (2023). https://api.semanticscholar.org/CorpusID:265150432
11. Skjæveland, M.G., Slaughter, L.A., Kindermann, C.: OntoCommons D4.3 - Report on Landscape Analysis of Ontology Engineering Tools, April 2022. https://doi.org/10.5281/zenodo.6504670
12. Slimani, T.: Ontology development: a comparing study on tools, languages and formalisms. Indian J. Sci. Technol. **8**(24) (2015). https://doi.org/10.17485/ijst/2015/v8i1/54249
13. Youn, S., Arora, A.: Survey about Ontology Development Tools for Ontology-based Knowledge Management. University of ..., pp. 1–26 (2009). http://suanpalm3.kmutnb.ac.th/teacher/FileDL/supot22255310501.pdf