ASIO: a Research Management System based on Semantic technologies

Jose Emilio Labra Gayo[1][0000−0001−8907−5348], José Barranquero Tolosa[2], Guillermo Facundo Colunga[1][0000−0003−1283−2763], Alejandro González Hevia[1][0000−0003−1394−5073], Emilio Rubiera Azcona[1][0000−0002−0292−9177], Daniel Ruiz Santamarìa[2], Pablo Menéndez Suárez[1], and Paulino Álvarez de Ron Ondina[1]

[1] WESO Research Group, University of Oviedo, Spain
[2] Izertis S.A., Spain

Abstract. In this paper we describe the architecture of a Research Management System based on Semantic technologies. The system is composed from two main modules: ontological infrastructure and research management system which are communicated through an RDF triple store that integrates all the information. The data model is defined in terms of Shape Expressions which are synchronized with Java entities that define the data model. The shapes also act as core layer that can be used to describe the main entities that will be employed and to validate their ontological definitions with test data. In this way, we propose a test-driven development approach for ontological engineering that improves the quality of both the ontologies defined and the data. The semantic architecture is based on a reactive approach which combines both a clean architecture and a stream-based pattern. This paper describes the architecture of the system and the main quality attributes and design decisions that have been taken into account.

Keywords: research management, semantic technologies, linked data, ontology, stream processing, shape expressions.

1 Introduction

There is an increase interest in the development of Research management systems which improve the available services for both researchers, research administrators and citizens in general. However, representing scholarly information is a complex task which requires to take into account the differences between disciplines, the decentralized nature of research advances and collaborations, as well as the increasing amounts of research data that are generated and need to be collected and taken into account. Although several approaches have tackled this problem promoting the use of linked data and semantic web technologies there is still a need to develop research management systems which are adopted by the different institutions and easily integrate their data models.

The HERCULES project[3] has been proposed as a University Research Data Semantics system for Spanish universities with the goal of developing new semantic web technologies that gather new information and integrate multiple nodes with heterogeneous ontologies and vocabularies. The University of Murcia (hereinafter UM) signed an agreement with the Spanish Ministry of Economy, Industry and Competitiveness (MINECO) in 2017 backing the HERCULES project with an 80% of cofinancing from the European Regional Development Fund program (ERDF) within the 2014-2020 period. The purpose of this agreement was to establish collaboration amongst MINECO and the UM, directed towards the improvement of public services and business innovation through the Public Procurement of Innovation. Some goals of the system were to create a research management system with semantic capabilities and infrastructures and create support systems for the detection of synergies in R&D between universities. The HERCULES project was divided in three main sub-projects: semantic architecture and ontological infrastructure (ASIO), research management system (SGI); and data enrichment and methods of analysis (EDMA).

ASIO. The objective of the ASIO project is to provide an ontology that represents the system of Spanish university research. And, at the same time, create a platform that allows you to store, manage and publish data.

SGI. Development of an innovative prototype of a comprehensive research management system that can be integrated into any CRUE university. It will include all management activities related to the research and must be integrated with the developed semantic architecture.

EDMA. Identification, extraction, analysis and evaluation of relevant data sets. Data completion from information from external data sources.

In this paper we present the architecture of the first sub-project called ASIO by its Spanish acronym, which has been proposed by one of the contractor teams formed by Izertis[4] company and the WESO research group[5] from the University of Oviedo, Spain. More information about the ASIO project can be obtained at https://www.um.es/web/hercules/proyectos/asio.

2 System Overview

The architecture of the implemented system is separated into two large blocks. On one side we have the Ontological Infrastructure. And on the other hand we have the Semantic Architecture.

[3] https://www.um.es/web/hercules/
[4] https://www.izertis.com/
[5] http://www.weso.es

The first block focuses on defining an ontological model that represents the Spanish universities research system. And, provide the necessary tools to carry out the development and evolution of this domain model. Within these tools we find continuous integration systems for ontologies, transpilers, systems for RDF validation using Shape Expressions [21] and systems to sync between GitHub [6] and Wikibase [7].

The second block focuses on import, processing, modification operations and data access. To carry out this, this block makes use of the Ontological Infrastructure. So the imported data, whatever the source is, is transformed to RDF and projected onto the Ontology. In addition, in this block, are the storage systems, data-processing modules, access points for SPARQL queries and the web application for end users.

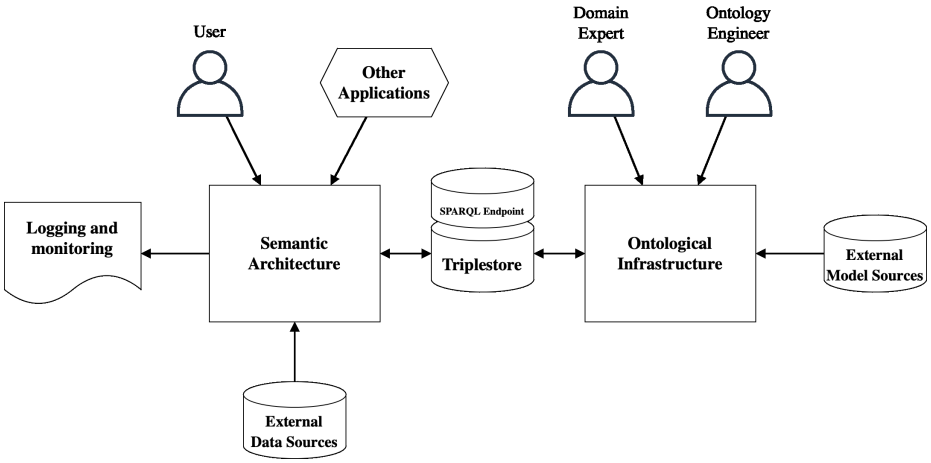

Fig. 1: General view of the different blocks that make up the implemented ASIO system.

Fig.1 shows us a general diagram in which we can see how the different actors and components of the system interact. On it you can see how the Domain experts, together with the ontology engineers, collaborate to generate the ontological domain model of ontological domain. Where existing ontologies are reused whenever possible. The semantic architecture then imports, processes, and manages access to the data. Always keeping the reference to the ontology.

[6] https://www.github.com
[7] https://www.wikiba.se

3 Ontological Infrastructure

The ontological infrastructure, as explained in the previous section, defines the domain model of the system. But, it also includes the implementation of the tools that solve existing problems in the development of ontologies. Specifically, the ontological infrastructure is composed by ROH Ontology[8], an ontology development methodology (inspired in good software engineering practice), a transpiler from Shape Expressions to Java, a continuos integration system for ontologies and a synchronization module of RDF between GitHub repositories and the software that powers Wikidata, Wikibase.

The aim of this part of the architecture is, therefore, to allow ontology engineers and domain experts have an environment where can define ontologies that form the backbone of data management systems.

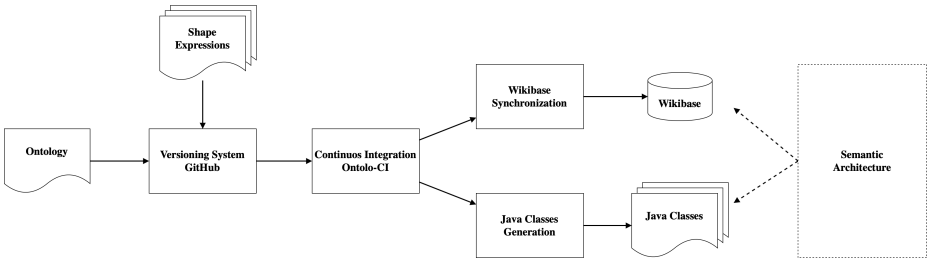

Fig. 2: General view of the components of the Ontological Infrastructure.

3.1 ROH Ontology

The ASIO ontology is designed to address the Research Management of the Spanish University System through the particular case of the University of Murcia but applying an encompassing model capable of addressing the rest of Spanish universities and also others from the European level.

The ontology is split into a central and peripheral modules, loosely inspiring ourselves in Fodor [8]. To do so we need to distinguish between two fundamentally different types of information processing (relying upon information architecture and datasets). In any informational system, in this case an ontology, there must be non-modular processing –or what we call central processing, to distinguish it from modular processing, which is peripheral (our vertical modules). To say that a part of the ontology is core (i.e. involves central processing actions) is, essentially, to say that it is not informationally encapsulated (as the vertical modules). In principle, any part of the system is relevant to confirming any other and we do not draw boundaries within it.

[8] https://www.github.com/herculescure/roh

On one hand, there are highly-specialised information processing datasets aimed at identifying and retrieving data from very specific environments. Informational based datasets in these specific environments involve only a limited type of information. That is why information retrieval tasks having to do with this first type of information are carried out by dedicated parts of the ontology that we call ***modules***. These modules are *domain-specific*, that is, they are responsible only for containing data falling in particular domains (geopolitical, scientific, administrative, staffing, etc.).

On the other hand, there are central informational tasks that involve much more complex and wide-ranging inferences and to which an indefinite amount of background information is potentially relevant. The information processing involved in carrying out these tasks is *domain-general* (conversely to domain-specific) and it concerns our main *university domain* (our ***core***), because we understand *general* here as our general domain.

On the basis of this distinction, we develop an architecture of the ontological organization as involving both very specialized modules (***vertical modules***) and what we call domain-general, non-modular knowledge (***core ontology***). Two properties of modularity in particular, *information encapsulation* and *domain specificity*, make it possible to tie together questions of functional architecture with those of knowledge content.

As far as the ***vertical modules*** are concerned, six modules have been implemented so far, namely:

- geopolitical entities
- administrative entities
- scientific domains
- subject areas
- Spanish universities
- human resources from some national university systems (Spain, Portugal and others)

The aim of this ontological architecture split into core and vertical modules is to facilitate the integration with other possible ontologies from university stakeholders, enabling some customisation through the following procedures:

1. At the ontological core level, some examples of mappings between entities are proposed and
2. illustrated by means of the `owl:equivalentClass` property, which map to equivalent entities in related ontologies, such as VIVO [9] or CERIF [10].
3. With the use of SKOS [11] in vertical modules
4. and their properties `skos:exactMatch`, `skos:closeMatch`, etc. the
5. possibility of mapping the realities of the different stakeholders is enabled.

[9] http://vivoweb.org/ontology/core
[10] http://www.eurocris.org/ontologies/cerif
[11] https://www.w3.org/2004/02/skos/

3.2 Test driven development for ontologies methodology

As part of the Ontological Infrastructure ecosystem, a methodology of test-based ontology development is included. This methodology is inspired by the best practices of software engineering. Where tests are used to direct and ensure correct development of a product.

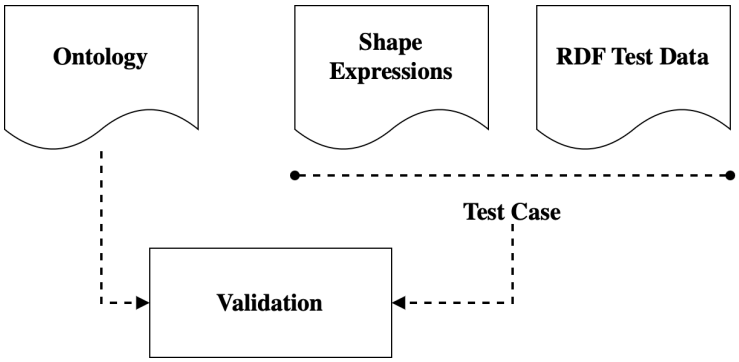

Fig. 3: Test case for an ontology.

In the case of ontologies, there is no algorithm as such, that can be tested giving it an input and comparing the output of the algorithm with the expected output. However, an ontology can have instances. RDF data that follows the ontology. In our methodology we propose to take these instances as the expected output. And, validate those instances with Shape Expressions. In this way, each test case will be made up of Shape Expressions and RDF data that follows the ontology, as shown in the Fig. 3.

3.3 ShEx-Lite: ShEx to Java transpiler

The goal of ShEx-Lite is to use the Shape Expressions that are used during the validation of the ontology as tests, to generate a domain model in Java. This domain model should be used by the Semantic Architecture. And, in this way, the ontological domain model would be synchronized with the implemented functionalities in Semantic Architecture.

`ShEx-Lite` is available as open source software at GitHub[12]. It has been designed in accordance with the concept *"compiler as an API"*, born with the Roslyn compiler [**?**]. The feature of code generation was designed with the goal of being flexible enough to work with different target programming languages like `Java` or `Python`. The main components of `ShEx-Lite` follow a traditional compiler architecture and are represented in Fig. 4.

[12] https://github.com/weso/shex-lite

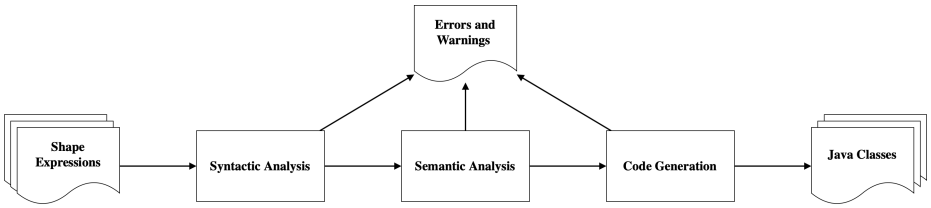

Fig. 4: `ShEx-Lite` internal architecture. SIL stands for ShEx-Lite Intermediate Language. IR stands for Intermediate representation.

- **Parse.** The components of this module aim at reading the source file and performing the syntax validation. The syntax validation of the schemas checks that the schemas defined follow the `ShEx-Lite` syntax. If this is not the case, the compiler lets the user know about the problem and possible solutions.
- **Sema.** At this stage the compiler checks that types are correct and that the invocations and references that occur in the schemas are defined. Also, during this process, if any error or warning is detected, the compiler will notify the user about the problem and possible solutions.
- **IRGen.** This module is the one actually generating target code (`Java`, `Python`, `Any...`). In order to allow adding other languages in the future, ShEx-Lite delegates the specific language checks and mappings and therefore it offers an interface that other language generators will implement. Each one of the code generators is responsible of checking that the constraints represented by the schemas are able to be expressed in the corresponding language. If the schema meets the requierements of the corresponding language, it is the specific code generator the one that produces the code.

3.4 Ontolo-CI: ShEx based ontologies continuos integration system for GitHub

The different software modules that make up a project usually use systems of continuous integration that allow to keep the software continuously tested running the tests are automatically carried out every time there are changes to the software.

Continuous software integration has a large ecosystem of tools such as Travis [13] or Circle-ci [14] that allow us to carry out that purpose. However, When it comes to testing ontologies, there is no tool that allows us to carry out a process of continuous integration of the same. That's when it arises the need to create a system that meets that need.

Ontolo-ci [15] is a continuous integration system for ontologies inspired by Travis. That allows the execution of tests for ontologies automatically on reposi-

[13] https://www.travis-ci.org
[14] https://circleci.com
[15] https://www.github.com/weso/ontolo-ci

tories from GitHub. The tests, carried out by ontolo-ci on the ontology are based on Shape Expressions or competency questions coded as SPARQL queries.

To solve the problem raised above we implemented a rest API that is capable of:

- Listen to the changes that take place on the ontology repository.
- Carry out the execution of the tests.
- Publish the results obtained.

Every time a change occurs on the ontology repository, ontolo-ci picks up that change and runs all the tests that are defined. After this process, ontolo-ci reports the results directly to the repository, as well as to the ontolo-ci database, which is consumed by the web-client. In both places we can watch the results of each test.

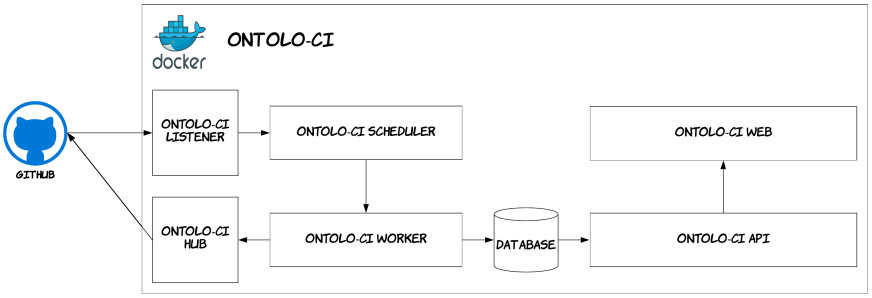

Fig. 5: Diagrama de componentes de Ontolo-CI.

The Fig. 5 shows us the different components of Ontolo-CI. Below is a brief summary of each of them.

- **Listener.** This component is the entry point of the subsystem that will be used by other external systems. To do this, it offers an external interface, called OnWebhook, which is responsible for receiving data on ontology updates from the version control system. In addition, this component is responsible for creating the build objects (a build object represents the build of the project). Build objects are present throughout the ontolo-ci lifecycle. They represent an execution of the tests on a specific state of the version control repository. The listener is in charge of not only creating these objects but also filling them with the information that comes from the version control system.
- **Scheduler.** This component is in charge of deciding when the builds should be executed. In such a way that if you get many builds, you are able to manage the order in which they are to be executed.
- **Worker.** This component is responsible for carrying out the execution of the tests once the build has been filled in with the appropriate tests.

- **Hub.** The hub is the component that communicates with the version control system. It is in charge of obtaining the files that correspond to the new build, as well as posting the results of the tests in the version control system.
- **API.** The api exposes an endpoint to be consumed by the web. Returns the data of the builds that have already been completed.
- **Web.** This component represents a web page where you can see the results of all the builds that have been carried out.

3.5 Hercules-Sync: GitHub-Wikibase synchronization module

Version Control Systems (VCS) have been used by software developers to facilitate the process of working collaboratively on a set of source files. Developers can greatly benefit from the discussion of individual changes and the establishment of an acceptance workflow. Due to the nature of RDF models representing ontologies, they are dynamic entities which may change as a consequence of different reasons [**?**]:

- Changes in the conceptualization of concepts.
- Appearance of new information about concepts.
- Changes in the domain being described.

This makes RDF files a perfect fit for use with version control systems. However, some domain experts may not know how to work with Version Control Systems or with RDF files directly, but could still contribute useful information to the ontology creation and conceptualization. These users can work on an easy-to-use RDF browsing and publication service like Wikidata, where they can make modifications to the underlying content through a web interface. The contents of both the VCS and the publication service need to be synchronized, so both ontology engineers and domain experts can contribute to the growth and evolution of the ontology.

Hercules-Sync therefore seeks to allow a synchronization between RDF files, hosted on GitHub, and a Wikibase instance. Fig. 6 shows an overview of the synchronization system presented in this paper. The initial step is the modification of RDF files by an ontology engineer, and the push of those changes to a VCS (in our case, GitHub). After a new release is created, the WebHook associated with the repository is called with the changes regarding this new release. These changes are passed to the synchronization system, which will be in charge of updating this information to a Wikibase.

Although the complete workflow involves the use of GitHub and WebHooks to synchronize the content to a Wikibase, it is important to note that this is not necessary, and the synchronization system can be invoked independently to synchronize any given RDF data to a Wikibase[16]. A continuación mostramos el camino que siguen los datos desde que sufren una modificación en GitHub hasta que se publican en Wikibase.

[16] For instance, the UniTest Wikibase, available at https://unitest.wiki.opencura.com, has been populated using just the synchronization system and an RDF file.

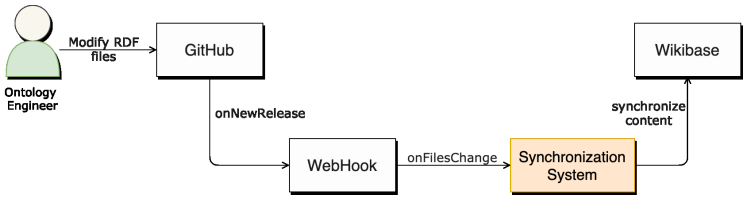

Fig. 6: Complete overview of the synchronization system.

Modifying an RDF file. The overall process begins with the modification of an RDF file. These changes in the file may be distributed amongst many commits in the VCS. In Fig. 7 we provide an example where a subset of an original file is changed multiple times. Some new triples are added to the graph, while others are removed or modified.

Original file

```
:knows owl:inverseOf :isKnownBy ;
       a owl:ObjectProperty
:alice :knows :bob, :carol .
```

Commit A

```
:knows owl:inverseOf :isKnownBy ;
       rdfs:range :Person ;
       a owl:ObjectProperty .
:alice :knows :bob, :carol ;
       rdfs:label "Alice" .
```

Commit B

```
:knows owl:inverseOf :isKnownBy ;
       rdfs:range :Person ;
       a owl:ObjectProperty .
:alice :knows :bob, :carol ;
       rdfs:label "Alice" .
```

Final file

```
:knows owl:inverseOf :isKnownBy;
       rdfs:range :Person ;
       a owl:ObjectProperty .
:alice :knows :bob ;
       rdfs:label "Alice" .
```

Fig. 7: Example evolution of an RDF file.

Making a new release. After a number of commits have been performed and the administrators of the repository decide that the new data is ready to be synchronized and published, a new release in GitHub can be created[17]. After that, the WebHook will be triggered and will notify the synchronization system with information about this new event.

Obtaining the graph diff. Once the synchronization system has received this notification from the WebHook, it will download the original and final RDF files from the repository. These files will then be parsed as graphs using the *RDFLib* library, and compared to obtain the diff between them.

[17] https://help.github.com/en/enterprise/2.13/user/articles/creating-releases

Creating the synchronization operations. After the diff between the two graphs is known, the next step is the creation of the operations that need to be executed on Wikibase to synchronize the changes.

An important step when creating the synchronization operations is to infer the types of each resource that will be added to the Wikibase. First of all, when synchronizing an IRI node we need to know if that node represents a Wikibase property (Px) or an item (Qx) before executing any operations on the Wikibase. Sometimes it is trivial to know when a node corresponds to a given entity type: for example, all the IRIs that are used as a predicate in an RDF triple are properties. However, there are cases when some kind of type of inference needs to be performed (e.g. exploring `rdfs:subPropertyOf` predicates in the graph to identify the properties).

After we know which IRIs correspond to classes and properties, we also need to know which are the ranges of the properties. When creating a new property in Wikibase it is necessary to indicate its data type. Modifying it after its creation, although possible, is not an expected operation and should be done with care. When a property is used as a predicate in a triple, we rely on the type of the object to infer the datatype of the property. If the property is not used as a predicate in the whole graph, we also rely on the `rdfs:range` predicates available for the property. There may be times when the datatype of a property may not be inferred with the information available in the graph. For those cases, we expect its range to be '`wikibase-item`' by default.

Performing the operations on Wikibase. Once the operations have been created, the next step is to execute them on the target Wikibase. Since those operations provide general information to be executed on multiple triplestores, this step also includes some Wikibase-specific tasks:

- **Handling labels, descriptions and aliases**. An important aspect of the Wikibase data model to take into account for the synchronization is its internal entity representation[18]. The description of a node is composed of the following mappings to RDF:
  - Labels are defined as `rdfs:label`, `schema:name` and `skos:prefLabel` predicates.
  - Aliases are defined as `skos:altLabel` predicates.
  - Descriptions are defined as `schema:description` predicates.

  All of the predicates above have as objects language-tagged literals. When we encounter any of those predicates in the synchronization process, they are used as label, alias or description values for the given entity.
- **Mapping datatypes**. Although in the creation of synchronization operations we have already annotated the properties with their respective datatypes, there still needs to be a mapping between those datatypes and the ones from the Wikibase data model. For example, coordinates expressed in an RDF file

[18] For more information, see https://bit.ly/2PBU6Fo

using the `geo:wktLiteral` datatype are mapped to the Wikibase `globe coordinate` datatype, and the values from literals must be processed to fit with the format specified by Wikibase.

– **Blank nodes**. Blank nodes can be defined as existential variables, representing the existence of some unnamed resource [**?**]. The approach taken for the synchronization of blank nodes consists of representing each blank node as a Wikibase item, with its label being the UID generated by `rdflib` in the graph canonicalization phase. However, it is important to note that the time to canonicalize blank nodes may increase exponentially on large graphs.

4 Semantic Architecture

The role of Semantic Architecture is to offer functionality over data. Specifically, it plays a fundamental role in the data fetching. On integration from data from multiple sources and formats to RDF. In the storage of RDF data. In the automatic processing of these graphs to discover latent knowledge. In exposing all these data and processes through a web portal. And finally, in exposing a SPARQL endpoint that allows querying on the data of the system.

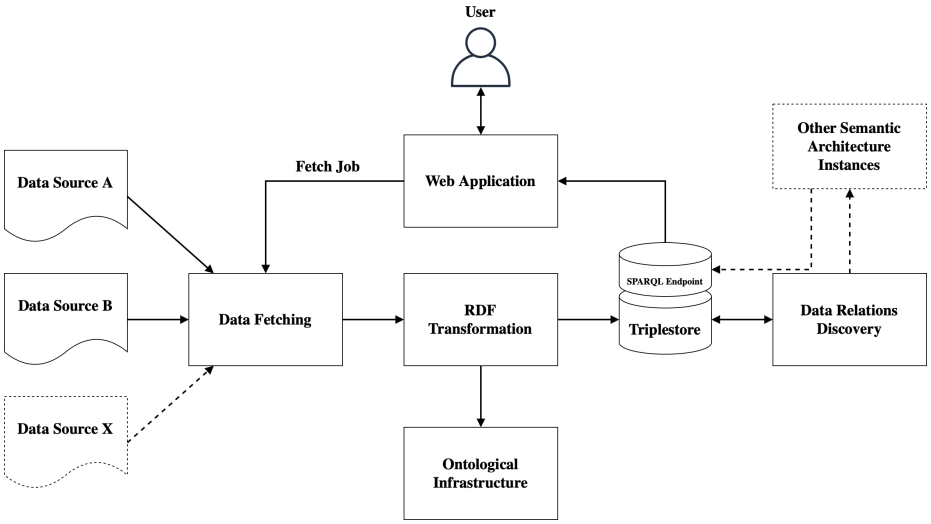

Fig. 8: General view of the components of the Semantic Architecture.

Some aspects that determine its design are the requirements for decentralization, interoperability, efficiency, usability, use of FAIR principles, data quality and replicability. In the Fig. 8 we can see how all the components interact.

As you can see in the diagram of the Fig. 8 each instance of the semantic architecture distributes the responsibilities between different modules. And, syn-

chronization is achieved through the SPARQL standard for data query and the link discovery module to find relationships between data from different nodes.

4.1 Data Fetching

One of the most complex processes in all systems is importing data from heterogeneous sources. The reason, is that each data source is different. And it is very likely that the format in which the data is stored in each source is also different. This is why that each source requires a significant level of customization. To fix this, in the Semantic Architecture it was decided to use the tool Pentaho[19]. This tool allows us create data import processes from different sources. Fig. 9 shows as an example of a Pentaho pipeline.

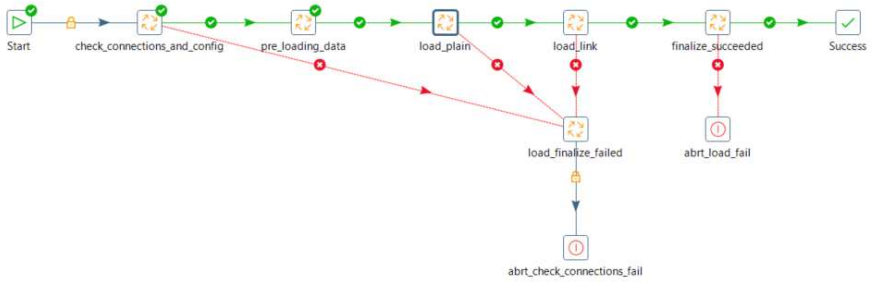

Fig. 9: ETL execution pipeline example.

4.2 RDF Transformation

Once we have the imported data we have to perform a transformation to RDF. For it, what is done is to map each field of the imported data sources with the ontology provided by the Ontological Infrastructure. At this point the process is done through streaming. In order not to block the system during the import processes. As they go performing the transformations, the triples are inserted in the architecture triplestore Semantics.

4.3 Data Storage

The data is stored on Trellis LDP. The reason why Trellis was chosen as a triplestore it is because of the features that it includes as LDP.

[19] https://es.wikipedia.org/wiki/Pentaho

4.4 Automatic Data Relationships Discovery

The main goal of automatic relationship detection is to address three independent but related functionalities. All of them will be discussed in depth in this document: Reconciliation of entities, discovery of links and detection of equivalences.

It is decided to implement an ad hoc solution, for the reconciliation of entities, despite the fact that there are triple store implementations that somehow support it, for example Stardog (entity linkig in knowledge graph) or BalzeGraph (link all the entities).

The main motivation for implementing your own solution is that the solutions mentioned above (regardless of their suitability for the project, which will be discussed below), directly contradict the requirement expressed in the specifications that it must be possible to change the triple store. The selection of one and the other would imply the existence of a vendor lock-in, towards these solutions, so that it would be impossible to change the triple store and continue offering the functionality associated with the reconciliation of entities, provided by them.

On the other hand, these do not seem to fully cover the needs of the project, for example the solution provided by BlazeGraph requires providing a list of variations for the attributes, obviously knowing that, is also knowing in advance which entities are duplicated and what values these have.

In the case of Stardog, this process is oriented rather to the association of external entities with entities within the knowledge graph, which being interesting, does not cover all the requirements expressed in the specifications for this component.

For all this, despite the associated complexity, the option of carrying out an own implementation is chosen, which maximizes the fulfillment of the project requirements, and reduces the vendor lock-in associated with the choice of one of the aforementioned solutions.

In the project's document repository [20] much more extensive documentation can be found for interested readers.

4.5 Web Application

For consumption by end users, a web portal has been designed where you can both consume information and execute actions. This web portal offers the following functionality.

Creation of data fetching jobs. A researcher, or failing that, the administration can request the importation of their data via the web. This runs an import and discovery job that results in the storage of the investigator's information, as well as references to other possible data sources on the node where the task was run.

[20] https://bit.ly/33HRnBo

Data query Through the web interface, a content negotiation is offered that allows modifying the interface depending on the data that is displayed. For example, a map will be displayed for coordinates and a graph for temporary data.

4.6 SPARQL Endpoint

The design of the query federation architecture aims to respond to the requirement for a joint response of the HERCULES network to a query about the system at a global level.

At a high level, the proposed approach is the management, registration and consultation of endpoints from the master node of the network, through a Service Discovery endpoint.

The Service Discovery module is also present in the federated nodes of the network, to manage the node registration and consult the available and active nodes.

This document assumes that this federation will be carried out at the read level (queries), in contrast to data synchronization during import (data replication). Data synchronization on all nodes is ruled out for the following reasons:

- Greater complexity of the data import module.
- Higher processing cost during import.
- Higher cost of information storage.
- Data privacy restrictions between nodes.

The query federation module is further subdivided into three key subsystems:

- QUERY: in charge of executing the query in parallel in all the nodes of the network and optimizing the process.
- IDENTIFICATION: it relies on the discovery library to be able to identify repeated or similar entities.
- AGGREGATION: once the results are received and the similarities are identified, it would be in charge of unifying the data of equivalent entities obtained from different nodes.

Query. Given the volume of information stored, it would be advisable to have an intermediate layer that allows queries to be made more efficiently, instead of querying from the standard SPARQL API, following different approaches recently proposed.

The use of the federation provided by SPARQL is ruled out for the following reasons:

- The federated query process is not robust to the failure of the attached endpoint network.
- The data aggregation process is very limited and is restricted to the functionalities exposed by the query language itself.

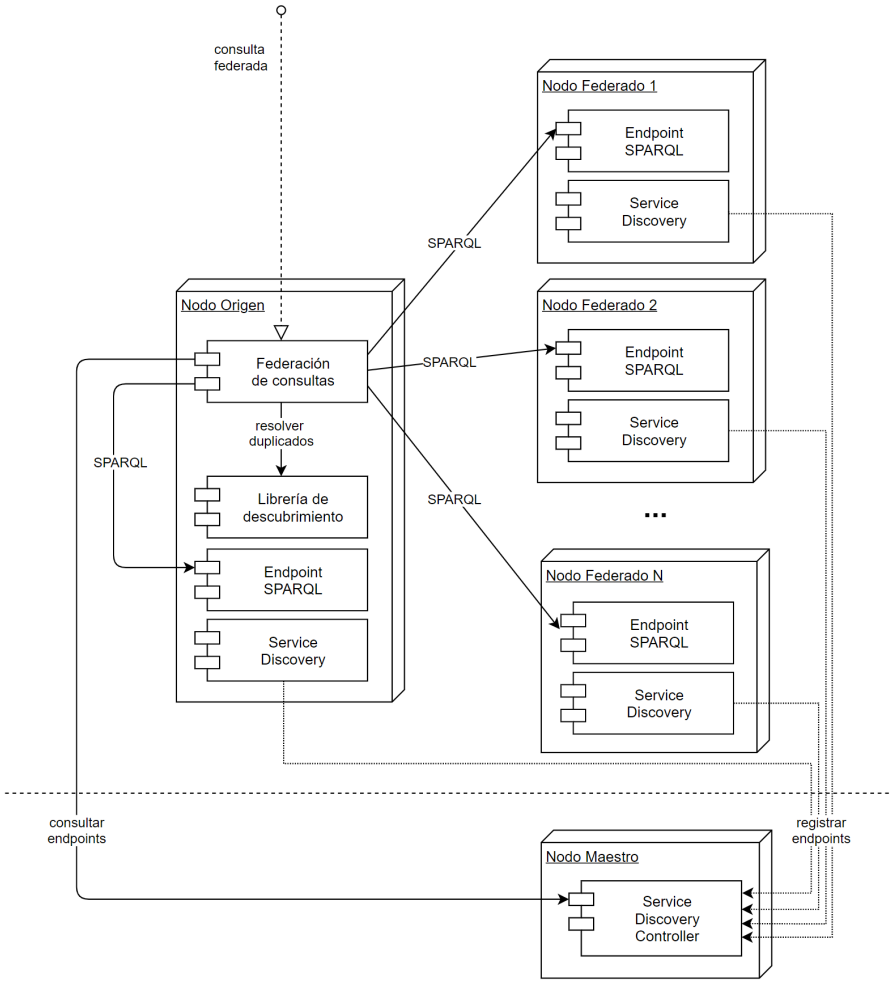

Fig. 10: SPARQL query federation diagram.

- The scalability of the process is limited for large volumes of information and is restricted by the SPARQL endpoints themselves.
- You lose control over the process and do not take advantage of the common data model and the discovery library.

Additionally, three technological challenges have been identified, still to be solved in the SPARQL ecosystem:

- Metadata management: this challenge involves the unification and synchronization of a common metadata catalog, which in the case of ASIO is solved through the centralized common ontology in the master node.
- Cache of results: Currently this challenge is being addressed through the use of Elasticsearch and Redis, from the Discovery Library module. As next steps, this module will be refactored to share the functionality with the final query federation module.
- Adaptive query processing: mainly linked to query planning / optimization, remaining outside the scope of this project as it is part of the core of the query planner of the triplestore used.

Identification. This subsystem makes use of the Discovery Library, from the Entity Reconciliation module, to be able to identify repeated or similar entities.

Agregation. The simplest version of the federation provides the results directly, without adding. In the case of identical duplicates, all nodes that have the entity are reported, without repeating it. In case of similar results (same entity, with different content), all the variants and their origin are listed.

This simplified version is of interest to be able to explicitly analyze the content offered by each node, although it would be highly recommended to have an entity merge module, reusing the services already developed in the data import module, specifically the Merge Event Processor, used in the Discovery Library.

Federated queries without aggregation are used in the Discovery Library in order to query the entities hosted in different nodes of the network during the import process, through the Data Fetcher. For the next release, it is planned to refactor the implementation of the Discovery Library, in which there is currently a high degree of coupling between the functionalities of both modules.

5 Related Work

The vivo platform was proposed as a mechanism to enable collaboration between scientists [5]. It was based on semantic web technologies with the VIVO-ISF (Integrated Semantic Framework) providing a set of ontologies for the system. OpenVIVO [15] was later proposed as an demonstration of the VIVO project where anyone with an ORCID account can join. VIVO has been adopted by a large number of institutions, specially in the United States, but also in other

countries around the world. On the other hand, euroCRIS[21], is an international organization of research information systems whose mission is to promote cooperation and share research information through the CERIF, the Common European Research Information Format. Although CERIF data model is based on XML, an initial ontology was developed for CERIF. The need to connect research management systems using linked open data was already proposed by [16,23].

The OpenAIRE (Open Access Initiative for Research in Europe) [22] initiative started as a project to promote Open Access and gradually moved to Open Science offering a research graph of linked data that relates publications, funders, research infrastructures, etc. [1].

The FAIR principles [24] promote the adoption of research data management systems that provide findable, accessible, interoperable and reusable data. The FAIR principles have been related to the linked open data guidelines [12] and the intersection between research data management, FAIR and open data has also been proposed by [13]. In a report developed by European Commission Expert Group on FAIR data, wikidata was mentioned as one of the technologies enabling FAIR data [14]. More recently the TRUST mnemonic has been proposed to promote trustworthy digital repositories which offer transparency, responsibility, user focus, sustainability and technological capabilities [18]. The ASIO model presented in this paper starts from the Spanish University system with a more modest goal but trying to offer a flexible system that can be later adopted by other research institutions, adopting FAIR and linked data principles and a scalable architecture. From a semantic point of view, the ASIO ontology maps its concepts to the VIVO and CERIF ontologies enabling future interoperability with those models. The reproducibility crisis of research has fostered the appearance of new initiatives like research objects [2–4,11,20]. The ASIO technology will give first class support to publish research objects following FAIR principles.

Combining ontological representations of science with research data platforms leads to the development of science knowledge graphs. ResearchSpace is a platform designed at the British Museum to help collaborations between researchers [19]. The Science Knowledge Graph Ontologies is a suite of ontologies that model research findings in various fields of modern science and which enable the development of a knowledge graph for science [7].

Shape Expressions (ShEx) have been proposed as a concise, human-friendly language to describe and validate RDF data [21]. There have been multiple applications of ShEx to different domains already [22], although as far as we know, the use of ShEx schemas to define data models and generate domain models in Java is new. The software architecture proposed is based on the event sourcing and CQRS patterns proposed by Young [25] and Dahan [6] for high performance applications. The pattern has already been employed at [10]. The ontological infrastructure usage of SKOS to map concepts between ontologies has also been proposed in [9]. We have adopted a test-driven approach for ontology

[21] https://eurocris.org/
[22] https://www.openaire.eu/

development inspired by [17] but validating the ontologies using test data plus shapes.

6 Conclusions

We have presented the ASIO semantic architecture for research data management. The main quality attributes that we aim to fullfil are interoperability, scalability and reusability. Following the linked data platform principles we also aim to offer a decentralized solution were the different universities can gradually join the HERCULES project. A prototype system is being developed where the main components are glued together.

Aknowledgements. The HÉRCULES Semantic University Research Data Project is backed by the Ministry of Economy, Industry and Competitiveness with a budget of 5.462.600,00 euros with an 80% of cofinancing from the 2014-2020 ERDF Program. It has also been partially funded by the Spanish Ministry of Economy and Competitiveness (Society challenges:TIN2017-88877-R).

References

1. Alexiou, G., Vahdati, S., Lange, C., Papastefanatos, G., Lohmann, S.: OpenAIRE Lod Services: Scholarly Communication Data As Linked Data (2017). https://doi.org/10.5281/zenodo.293836
2. Bechhofer, S., Buchan, I., Roure, D.D., Missier, P., Ainsworth, J., Bhagat, J., Couch, P., Cruickshank, D., Delderfield, M., Dunlop, I., Gamble, M., Michaelides, D., Owen, S., Newman, D., Sufi, S., Goble, C.: Why linked data is not enough for scientists. Future Generation Computer Systems **29**(2), 599–611 (feb 2013). https://doi.org/10.1016/j.future.2011.08.004
3. Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., Mina, E., Corcho, O., Gómez-Pérez, J.M., Bechhofer, S., Klyne, G., Goble, C.: Using a suite of ontologies for preserving workflow-centric research objects. Journal of Web Semantics **32**, 16–42 (may 2015). https://doi.org/10.1016/j.websem.2015.01.003
4. Brinckman, A., Chard, K., Gaffney, N., Hategan, M., Jones, M.B., Kowalik, K., Kulasekaran, S., Ludäscher, B., Mecum, B.D., Nabrzyski, J., Stodden, V., Taylor, I.J., Turk, M.J., Turner, K.: Computing environments for reproducibility: Capturing the "whole tale". Future Generation Computer Systems **94**, 854–867 (may 2019). https://doi.org/10.1016/j.future.2017.12.029
5. Corson-Rikert, J., Cramer, E.J.: Vivo: Enabling national networking of scientists. In: IASSIST (2010)
6. Dahan, U.: Clarified CQRS, https://udidahan.com/2009/12/0/
7. Fathalla, S., Auer, S., Lange, C.: Towards the semantic formalization of science. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. ACM (mar 2020). https://doi.org/10.1145/3341105.3374132
8. Fodor, J.A.: The Modularity of Mind. MIT Press (1983)
9. Frosterus, M., Tuominen, J., Pessala, S., Hyvönen, E.: Linked open ontology cloud: managing a system of interlinked cross-domain lightweight ontologies. International Journal of Metadata, Semantics and Ontologies **10**(3), 189 (2015). https://doi.org/10.1504/ijmso.2015.073879

10. García-González, H., Fernández-Álvarez, D., Labra-Gayo, J.E., de Pablos, P.O.: Applying big data and stream processing to the real estate domain. Behaviour & Information Technology **38**(9), 950–958 (may 2019). https://doi.org/10.1080/0144929x.2019.1620858
11. Giraldo, O., García, A., López, F., Corcho, O.: Using semantics for representing experimental protocols. Journal of Biomedical Semantics **8**(1) (nov 2017). https://doi.org/10.1186/s13326-017-0160-y
12. Hasnain, A., Rebholz-Schuhmann, D.: Assessing FAIR Data Principles Against the 5-Star Open Data Principles. In: Lecture Notes in Computer Science, pp. 469–477. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-98192-5-60
13. Higman, R., Bangert, D., Jones, S.: Three camps, one destination: the intersections of research data management, FAIR and open. Insights the UKSG journal **32** (2019). https://doi.org/10.1629/uksg.468
14. Hodson, S., Jones, S., Collins, S., Genova, F., Harrower, N., Laaksonen, L., Mietchen, D., Petrauskaité, R., Wittenburg, P.: Turning fair data into reality: interim report from the european commission expert group on fair data (2018). https://doi.org/10.5281/ZENODO.1285272
15. Ilik, V., Conlon, M., Triggs, G., White, M., Javed, M., Brush, M., Gutzman, K., Essaid, S., Friedman, P., Porter, S., Szomszor, M., Haendel, M.A., Eichmann, D., Holmes, K.L.: OpenVIVO: Transparency in scholarship. Frontiers in Research Metrics and Analytics **2** (mar 2018). https://doi.org/10.3389/frma.2017.00012
16. Joerg, B., Ruiz-Rube, I., Sicilia, M.A., Dvořák, J., Jeffery, K., Hoellrigl, T., Rasmussen, H.S., Engfer, A., Vestdam, T., Barriocanal, E.G.: Connecting closed world research information systems through the linked open data web. International Journal of Software Engineering and Knowledge Engineering **22**(03), 345–364 (may 2012). https://doi.org/10.1142/s0218194012400074
17. Keet, C.M., Lawrynowicz, A.: Test-driven development of ontologies. In: Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678. pp. 642–657. Springer-Verlag, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-319-34129-3_39
18. Lin, D., Crabtree, J., Dillo, I., Downs, R.R., Edmunds, R., Giaretta, D., Giusti, M.D., L'Hours, H., Hugo, W., Jenkyns, R., Khodiyar, V., Martone, M.E., Mokrane, M., Navale, V., Petters, J., Sierman, B., Sokolova, D.V., Stockhause, M., Westbrook, J.: The TRUST principles for digital repositories. Scientific Data **7**(1) (may 2020). https://doi.org/10.1038/s41597-020-0486-7
19. Oldman, D., Tanase, D.: Reshaping the knowledge graph by connecting researchers, data and practices in ResearchSpace. In: Lecture Notes in Computer Science, pp. 325–340. Springer International Publishing (2018). https://doi.org/10.1007/978-3-030-00668-6-20
20. Pasmín, O.X.G., Corcho, O., Castro, A.G.: SMART Protocols: seMAntic represenTation for experimental protocols (2014), http://oa.upm.es/36778/
21. Prud'hommeaux, E., Gayo, J.E.L., Solbrig, H.: Shape Expressions: An RDF Validation and Transformation Language. In: Proceedings of the 10th International Conference on Semantic Systems (2014). https://doi.org/10.1145/2660517.2660523
22. Thornton, K., Solbrig, H., Stupp, G.S., Gayo, J.E.L., Mietchen, D., Prud'hommeaux, E., Waagmeester, A.: Using shape expressions (ShEx) to share RDF data models and to guide curation with rigorous validation. In: The Semantic Web, pp. 606–620. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-21348-0-39

23. Wiljes, C., amd Florian Lier, N.J., Paul-Stueve, T., Vompras, J., Pietsch, C., Cimiano, P.: Towards linked research data: An institutional approach. In: Castro, A.G., Lange, C., Lord, P., Stevens, R. (eds.) 3rd Workshop on Semantic Publishing (SePublica). CEUR Workshop Proceedings, vol. 994, pp. 27–38 (2013)
24. Wilkinson, M.D., Dumontier, M., Aalbersberg, I., Appleton, G., Al., E.: The FAIR Guiding Principles for scientific data management and stewardship. Sci Data (2016). https://doi.org/doi:10.1038/sdata.2016.18
25. Young, G.: CQRS and Event Sourcing, http://codebetter.com/gregyoung/2010/02/13/cqrs-and-event-sourcin