D2.1 Requirements and use case specification

Document Due Date: 30/04/2020
Document Submission Date: 27/04/2020

Work Package 2: System Architecture, Requirements & Use Cases

Document Dissemination Level:
Public

INODE
Intelligent Open Data Exploration
is funded by the Horizon 2020 Framework Programme of the EU for Research and Innovation.
Grant Agreement number: 863410— INODE — H2020-EU.1.4.1.3.

(This page has been intentionally left blank)

Executive Summary

Deliverable D2.1 provides an analysis of the INODE requirements from both a technical and a user perspective. It presents the use cases that will be deployed in the project, the requirements that stem from these use cases, along with a rich set of technical requirements that need to be addressed in order to provide the desired functionality as initially defined by the INODE service offering.

This deliverable gives an overview of the project context as far as it concerns the objectives and the goals of the task it refers to. The relationship to other WPs and tasks is also presented to show alignment with the other activities of the project. The methodology that is followed in this document starts with the definition of the INODE concept along with an overview of the services that INODE plans to deliver. It continues with the description of the use cases, the elicitation of the requirements and their consolidation. Three use cases are described using a common pattern with a brief overview with respect to their context and needs for data exploration and querying, as well as the visualization of the results, the context within which they will be implemented, the description of the use case scenario as a set of functionalities offered, the various stakeholders that are involved in their execution and the user-centric requirements analysis perspective.

The requirements analysis process relies heavily on the involvement of the stakeholders in the whole value chain that the project brings. The INODE consortium includes all necessary stakeholders of the respective value chain and the whole methodology followed has been aligned with this feature of the project. The consortium includes data infrastructure providers, technology providers as well as service providers, integrators, and end-users who will be recruited in the project through the pilot activities. This approach allows for a credible validation of the INODE concept, along with different deployment configurations and service operations plans.

The overall requirements analysis and consolidation is based on the well-established process of dividing the requirements into two categories: functional and non-functional. The functional aspect of the requirements analysis focuses on what a system must do to produce the required operational behavior. This includes inputs, outputs, states, functions and transformation rules. Functional requirements are the primary source of the requirements that will eventually be reflected in the system specification. A non-functional requirements analysis focuses on what other technical features a system must have in place in order to facilitate the service provision.

This document concludes with a detailed and consolidated requirements list so that the implementation of the INODE platform can be effectively carried out in the next tasks and tracing mechanisms towards validating the requirements fulfillment can be achieved. Moreover, the consolidated list of requirements will facilitate the collaboration and synergies among different system developers and partners throughout the next tasks and work packages of the project.

Project Information

Project Name	Intelligent Open Data Exploration
Project Acronym	INODE
Project Coordinator	Zurich University of Applied Sciences (ZHAW), CH
Project Funded by	European Commission
Under the Programme	H2020-EU.1.4.1.3. - Development, deployment and operation of ICT-based e-infrastructures
Call	H2020-INFRAEOSC-2019-1
Topic	INFRAEOSC-02-2019 - Prototyping new innovative services
Funding Instrument	Research and Innovation action
Grant Agreement No.	863410

Document Information

Document reference	D2.1
Document Title	Requirements and use case specification
Work Package reference	WP2
Delivery due date	30/04/2020
Actual submission date	27/04/2020
Dissemination Level	Public
Author(s)	Bastian Frederic, Mendes de Farias Tarcisio (SIB)
Contributor(s)	Koutrika Georgia, Skoutas Dimitris (ATHENA) Amer-Yahia Sihem, Boumaout Mourad (CNRS) Lücke-Tieke Hendrik, May Thorsten (Fraunhofer) Litke Antonis, Mitropoulou Aikaterini, Papadakis Nikolaos, Papadopoulos Dimitris (Infili) Fabricius Max, Subramanian Srividya (MPE) Massucci Francesco, Rull Guillem (SIRIS) Calvanese Diego, Lanti Davide, Mosca Alesandro (UNIBZ) Braschler Martin, Kosten Catherine, Smith Ellery, Stockinger Kurt (ZHAW)

List of Acronyms and Abbreviations

Acronym	Explanation
ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
Bgee	A multi-species gene expression database.
CLI	Command Line Interface
CORDIS	Community Research and Development Information Service the European Commission repository of EU-funded research
CRUD	Create, Read, Update, Delete
CSV	Comma Separated Values
EOSC	European Open Science Cloud
EPO	European Patent Office
ETL	Extract, Transform, Load
GIS	Geographic Information System
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NL	Natural Language
NLP	Natural Language Processing
NUTS	Nomenclature of territorial units for statistics
OncoMX	A cancer biomarker resource leveraging published literature and genomics data.
OWL	Web Ontology Language
RDBMS	Relational Database Management System
RDF	Resource Description Framework
REST	Representational state transfer
SPARQL	Query language and a protocol for accessing RDF
SQL	Structured Query Language
UML	Unified Modelling Language
VCaaS	Visual Computing as a Service
VKG	Virtual Knowledge Graphs
W3C	World Wide Web Consortium

Table of Contents

1 INTRODUCTION

1.1 Objectives and goals of the deliverable

The current document is the deliverable 'D2.1 Requirements and use case specification' which comprises the major outcome of 'Task 2.1 - Requirements specification' and 'Task 2.2 - Use case specification'.

Task 2.1 specifies the functional and non-functional requirements of INODE. Specific tasks include the identification of the main data sets, what kinds of analysis are typically performed by end-users, and the definition of the visualizations needed to best explore the data. Task 2.2 defines the specific use cases for the three exploration types: by-example, by-analytics and by-natural language.

The first step in such an action is to describe a list of use cases from the partners who will provide the pilots and validate the use cases. These will then be further analyzed and discussed with the help of the technology partners. The use cases are described based on a common template that highlights various characteristics such as their innovative nature, interaction with stakeholders, data that will be provided, query examples, and requirements for the technological base of INODE, etc. The second step is to extract, elicit, and analyze the requirements from the perspective of the various stakeholders of the project, who will influence the design and implementation of the INODE system and its services as a whole.

The primary audience of this document consists of people who will participate in the design and development of the INODE system and services. This audience consists primarily of members of the consortium who will design and implement the components and modules of the system. Additionally, this document is of wider interest to stakeholders that are active in the areas of EOSC, open data initiatives, and data exploration, including researchers participating and contributing to H2020 projects under the aforementioned topics.

1.2 Relation to other Work Packages

The requirements and use cases serve as the basis for designing the overall INODE system architecture (Deliverable D2.2 at project month 12) and are the primary reference point for designing the individual INODE components, such as querying databases in natural language, interactive data exploration, user assistance, and visual data exploration.

2 OVERALL METHODOLOGY APPROACH

The analysis starts with a definition of the INODE concept, including an overview of the use cases, use case diagrams (described through Unified Modelling Language - UML) and the stakeholders involved so as to understand the context of the project and identify the various stakeholders. The consortium partners then give an overview of the technologies that are going to be involved in the project and how they will be used as a basis for the implementation of the INODE platform. Similar projects as well as some background from previous projects are also mentioned in order to present the state-of-the-art and previous achievements that can be used as a starting point. Then, the elicitation of requirements is derived from the definition of the desired functionality of the INODE system given from the perspective of both the functional components and the non-functional attributes that the end system will expose. Finally, the requirements are gathered and consolidated into one list, grouped and coded accordingly, in order to provide the reference for the design, implementation and validation phases of the project.

The following figure (Figure 2-1) summarizes the overall methodology for the requirements analysis approach in INODE.

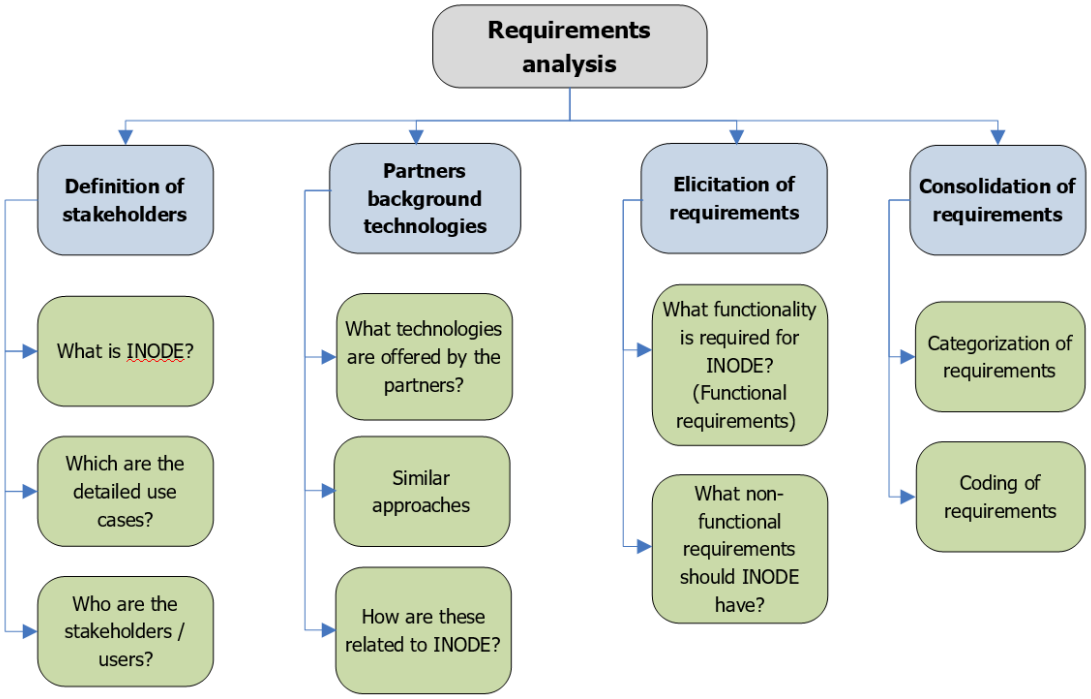

Figure 2-1 *Requirements analysis methodology*

3 USE CASES DESCRIPTION AND REQUIREMENTS

3.1 Project Definition

3.1.1 What is INODE?

To date, the scientific community has a wide variety of tools and APIs for managing and querying big and heterogeneous data sets. However, there are still several limitations that make accessing and combining data from different sources non-trivial, time-consuming, and user unfriendly for the different stakeholders of the data ecosystem. These limitations are further outlined below.

(1) Existing search interfaces are cumbersome and non-intuitive for most users. Specialized query languages such as SPARQL or SQL are not meant to be used by most users. On the other hand, keyword search interfaces are very simplistic and return flat lists of results, which does not help users understand and leverage data. Hence, the ability of users to explore data sets heavily depends on their technical and domain expertise.

(2) Users are not familiar with open datasets. There are two challenges that lie with open data. First, their exact contents and data structure are usually unknown to most users. Second, users may not have well-defined information needs over open data. They may only have some intuition about the data, and not know exactly what to ask, where to look for it, what they need, or how to interpret a result.

(3) Combining data from different data sets is difficult. Combining data residing in more than one dataset can uncover rich information and insights. However, to do so, one needs to identify which datasets can be linked and how - a process that is time-consuming and challenging even for expert users. Some data scientists estimate that up to 80% of their research time is covered by what they call “data jujitsu”, i.e., data preparation and data management tasks.

Many efforts for collecting open data sets may be wasted if they cannot be explored easily. Intuitive data linking, and data exploration primitives are becoming a necessity to unveil the full potential of data sets to all users.

The INODE project addresses the above challenges and targets open gaps in the data exploration service offering of the EOSC-hub. The core principle of INODE is that users should interact with data in a more dialectic and intuitive way, similar to human dialog. To achieve this principle, INODE will offer a suite of agile, fit-for-purpose, sustainable services for exploring open data sets that help users (a) link and leverage multiple datasets, (b) access and search data using natural language, examples, and analytics, (c) get guidance from the system in understanding the data and formulating the right queries, and (d) explore data and discover new insights through visualizations. This is achieved through INODE’s parallel services: **OpenDataDialog** and **OpenDataLink**. OpenDataDialog enables end-users (e.g., scientists) to perform intuitive queries in natural language with the support of data exploration capabilities (e.g., query suggestions, interactive visualizations), helping them to fully grasp the underlying data structures and maximize insights. OpenDataLink is mostly targeted at domain experts who already have the technical skills to navigate through complex datasets but lack an effective toolkit to link, query, and generate structured knowledge from heterogeneous sources. INODE can meet the different demands of autonomous data exploration with

OpenDataDialog by accelerating knowledge extraction through sophisticated semantic queries across large data sets, and with OpenDataLink by providing the necessary data management infrastructures to help users unlock the full potential of multiple data integration.

Our service offering is formed by and will initially respond to the needs of large and diverse scientific communities brought by our three use case providers: (a) Research and Innovation Policy Making, (b) Astrophysics, and (c) Cancer Biomarker Research.

The following subsections give a detailed analysis of the selected INODE use cases, which will lead to the specification of the overall system as well as the implementation and validation/evaluation of the service offering.

A common aspect across our use cases is a detailed characterization of different user roles. User roles reflect a variety of domain, data, and technical expertise. They also reflect different user needs and different ways of exploring data. User roles are referred to with different names in individual use cases. For instance, a domain expert is a policy maker in our first use case, an astronomy scientist in our second use case, and a biologist in our third use case. Any given user in INODE, a domain expert, a use case provider, a data scientist, a novice end-user, may have a mix of roles. INODE will cater to all those roles by providing different levels of guidance, i.e., recommendations and explanations, in expressing exploration queries and pipelines, and enabling different user feedback according to their expertise. For instance, a data scientist will be proficient in expressing complex data processing operations but may know little about the actual domain-specific needs. A domain expert, on the other hand, will have extensive knowledge about the needs in a particular domain and will benefit from guidance in expressing those needs with INODE exploration queries and pipelines. A novice user will need guidance in the entire exploration process, and INODE will provide recommendations and explanations of results, while the user may only be able to provide minimal feedback on data. A system administrator or data curator will benefit from the ability to select subsets of the data for which different configurations and permissions need to be set. While INODE does not focus on access control functionality, it provides the ability to identify different subsets of input datasets, enabling their effective management, access, and exploration.

3.2 Research and Innovation Policy Making

3.2.1 Overall description

The main goal of INODE in the context of Research and Innovation (R&I) policy making is to allow policy makers, who tend to be non-technical users, access a collection of relevant datasets in a homogeneous way so that they can make informed decisions (and move towards what is commonly called evidence-based policy making). This applies to decision-makers both within organisations that fund and perform research and innovation, that is, regional governments, foundations, etc. as funders of R&I, and within universities, research centers, etc. as performers of R&I.

The end-users of this use case domain are therefore policy makers, administrators, and scholars from public organisations, private foundations, corporations, agencies, and companies that (i) are actors in the creation of the future R&I sector, or (ii) have the

responsibility of managing its evolution. Figure 3-1, Figure 3-2 and Figure 3-3 below, show a typical usage scenario for INODE, where a policy maker from a regional government has the following information need *"I need to understand how much the scientific work of the scholars from the public research institutions in my region has also generated a potential for the creation of economic value in the recent years".*

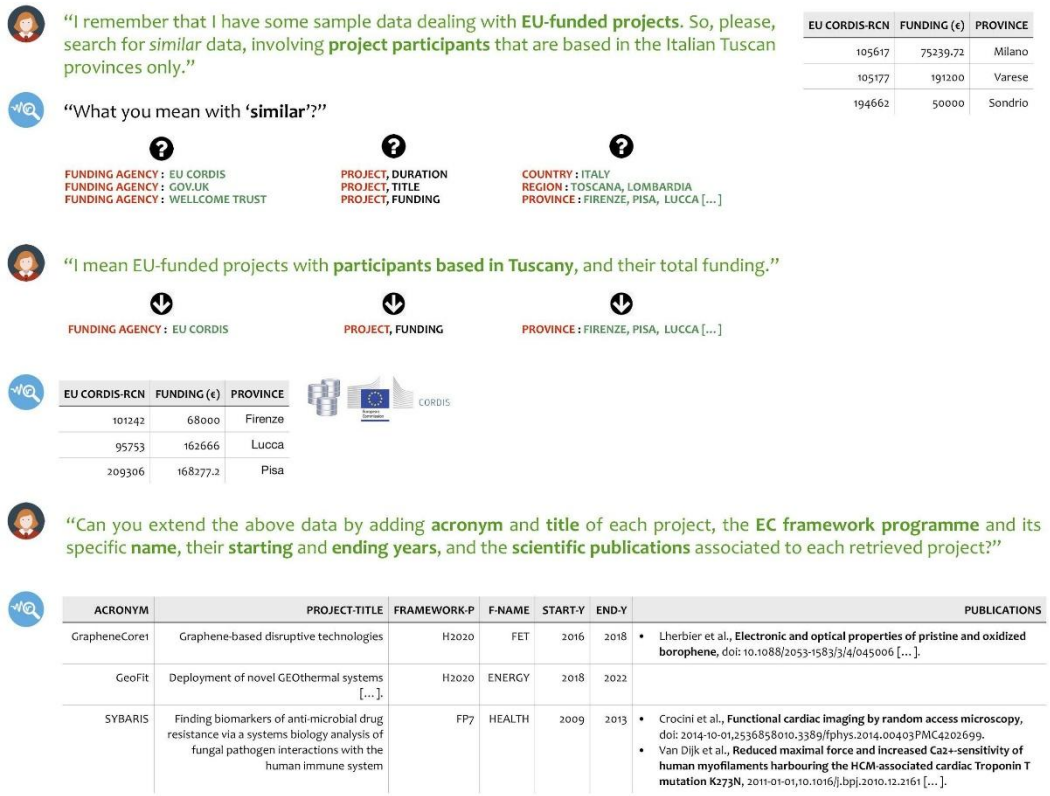

Figure 3-1 *The figure first shows how INODE supports a user in disambiguating a query based on an existing data sample and a knowledge graph that provides structure to the introduced query dimensions. In the second interaction, we see how INODE expands the answers (which can be 'easily' retrieved from the EU-CORDIS repository) thanks to the integration of the OpenAIRE data about EU-funded projects' publications.*

This scenario is tackled by INODE through exploration operators (by-example, by-filter, by-join, etc.), explanations that allow INODE to ask the user questions in natural language, and through recommendations. In the workflow shown in Figure 3-1, the analyst starts with a sample dataset of EU-funded projects and uses **by-example** to find similar projects. The system asks for clarification using **natural language explanations**. The analyst then uses **by-filter** to find partners located in Tuscany among those projects. Finally, the analyst asks for another **by-join** to augment the results with additional attributes such as acronym, title, EC program, start-date and end-date.

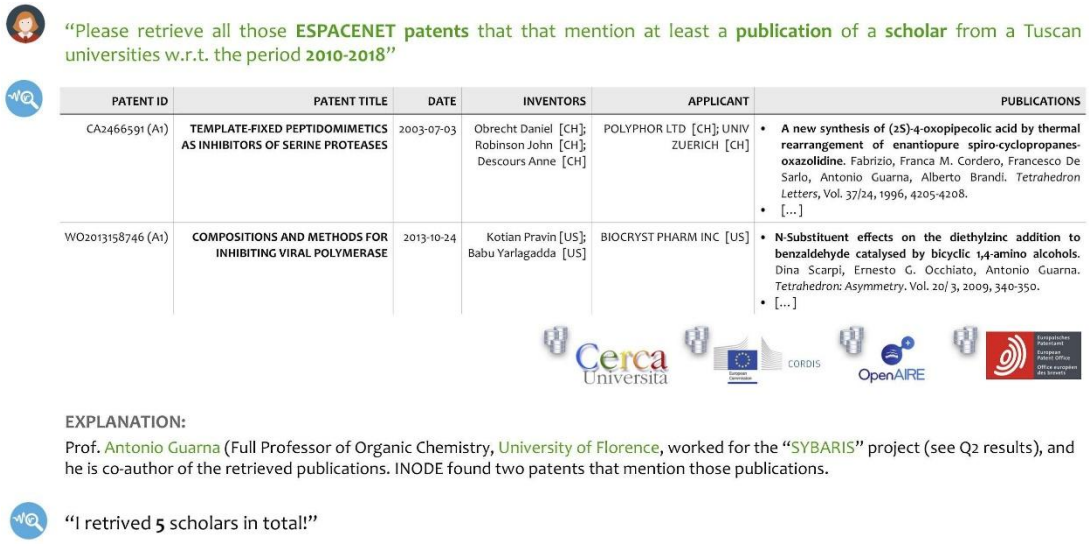

Figure 3-2 *Here INODE relies on the integration with yet another repository that is managed by the European Patent Office (EPO) called ESPACENET. Notice that the above scenario is particularly difficult to deal with. On the one hand, the required data integration among EPO, OpenAIRE, and CORDIS is not straightforward. On the other hand, the currently available open repositories on patents (such as ESPACENET) do not provide structured information about the references to scientific publications yet. Analytically, INODE is finally able to show the requested records by joining the scholars of the Tuscan universities (see, CercaUniversità CINECA) with the previously retrieved data.*

Similarly, in the workflow shown in Figure 3-2, the analyst uses **by-join** to query the open repository on patents (ESPACENET) and join it with the scholars from the CercaUniversità CINECA Tuscan universities who published between 2010 and 2018. The final step of the analysis, illustrated in Figure 3-3, requires further use of **by-filter** to retrieve the data needed to create the ranking of European regions that answers the user's information need. The system uses **natural language explanation** to describe the results retrieved.

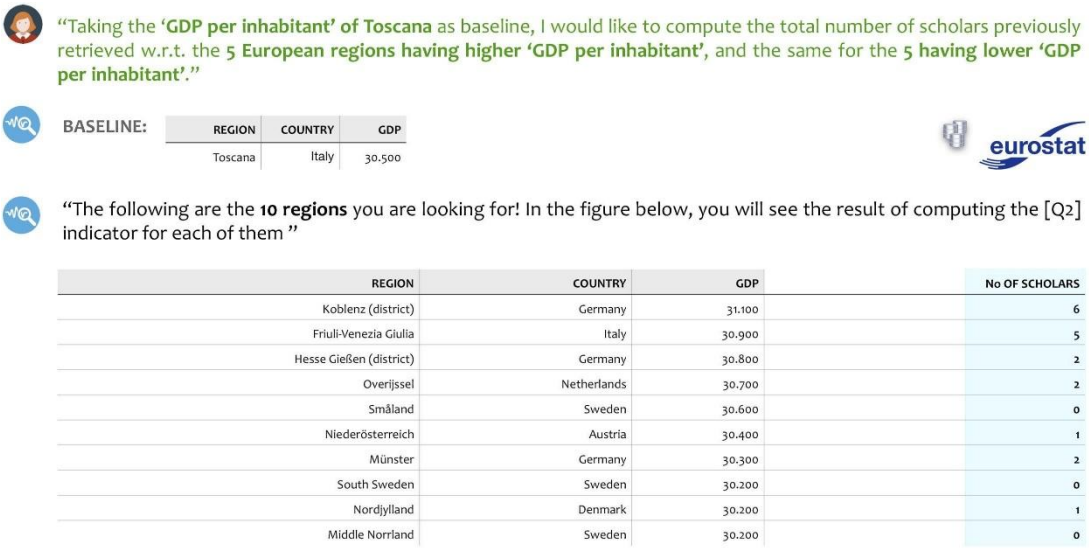

Figure 3-3 *In the last step of the interaction, INODE accesses data from EUROSTAT in order to find the Tuscany-related baseline. Afterwards, INODE retrieves the data* about *European regions in order to recreate the ranking needed and finally computes the* value for the required indicator.

As shown in this example (Figure 3-3), R&I data relevant for R&I decision makers is usually located in disconnected sources, under different formats, and with different structures. This is an ideal scenario for applying solutions based on **virtual knowledge graphs** (VKGs)[1] as intended in INODE (see also Section 4.1.1). In this sense, INODE will provide the tools needed to help bring together all of these disconnected sources in an open and interoperable fashion.

In summary, evidence-based policy making is one of the major areas that INODE will impact, most notably the following ways: (a) Integrating data between research inputs and outputs, will allow for econometric impact estimation; (b) Integrating actor-based data with network-based data (collaborations, mobility, co-authorship, and co-invention) for examining policy impact on networking and knowledge flows; and (c) Integrating institution-based data with geographic data for analysing the impact of policies at both the local and regional level.

3.2.2 Stakeholders definition and mode of interaction

Who are the users/stakeholders?

Policy makers:

The main profile of users in this use case are non-technical officials from universities, public organizations, foundations, etc. that expect to perform queries, in which all of the calculations are already done for them (in SQL terms, these are queries with aggregation functions such as count, sum, etc. with some filters applied to them). In contrast with technical users who prefer to get the raw data and do the calculations themselves, policy makers prefer to have the system do it for them.

Typically, these users are not capable of exploiting SPARQL endpoints (which typically provide access to the -limited- R&I open data sources) since they do not have the technical skills to formulate SPARQL queries, even if they are not after particularly complex results. In the end, what these users expect to obtain by querying those sources is a table with which they can work offline, most typically in Excel. The main goal of these queries is to get a picture of the ongoings within their institution, so they can make informed decisions. This profile is therefore the perfect target of the Natural Language (NL) and exploration services of INODE. NL querying would solve the aforementioned SPARQL knowledge problem, and allow policy makers to pose specific queries to the system by themselves. Exploration would help in cases where they want to investigate some aspect of how the institution is performing, but do not have an exact query in mind. Policy makers could indirectly construct a query, step by step, by applying the exploration operators, starting from an initial set of objects of interest, or by selecting queries that INODE recommends based on the context.

[1] The *virtual knowledge graph* approach to data access and data integration is commonly known in the literature also as *ontology-based data access/integration* (OBDA/OBDI).

Statisticians:

Although not extremely common in many of our projects, technical users within statistical offices are also included within this use-case scenario. Differently from the non-technical users (who, as mentioned, expect the queries to do all the calculations for them), this kind of user prefers to get the relevant raw data from the system and do the analysis themselves with their own tools. In this sense, they are more interested in having an API-like interface, with predefined queries that can be parameterized to get a dump of the data relevant to their study.

Citizens:

Sometimes, the general public is among the data consumers relevant to this use-case. This is typical of public institutions that want to use the system, not just for their internal strategy, but also as a form of public accountability. Citizens would have access to a public portal with informative visualizations, possibly embedded in a narrative that gives them context, following a story-telling approach.

Data administrator:

From a system administration point of view, the most important figure in the present case, is that of the person who manages the data sources. This profile appears when institutions want to have complete control of the system, which means having the data on their own servers and being able to administer them themselves.

System administrator:

As with any system, there should be someone who administers the whole system. Again, this could be SIRIS or the client. It is unclear whether this level of administration would be done through the system itself or from the outside (e.g., by editing configuration files directly on the server's filesystem).

How is every stakeholder/user interacting with application/service?

Table 3-1 *R&I Stakeholders table**.***

Stakeholder/User	***Role***	***Interacting functionalities***
Policy maker	Performs both specific queries and explorations.	Regarding specific queries: 1. Writes a query in natural language. 2. Gets a tabular result together with a NL explanation and some recommendations for related queries. 3. Refines the query, if needed, based on the NL explanation. 4. Once satisfied with the query, downloads the result as a CSV. 5. Selects one of the recommended queries that seems relevant to the study and refines it, if necessary (then goes back to point 2).

		Regarding exploration: 1. Writes a NL query to get an initial set of objects. Alternatively, selects a predefined query from a catalogue (possibly organized by topic, such as teaching, human resources, research, etc.) and refines it. 2. Applies an exploration operator on the result of the NL query (probably by example or by analytics). Optionally, from the perspective of the policy makers user group, it would be nice if it were possible to select how to see the result, for instance, just as a table or through some visualization. Additionally, it would be even better if the system could recommend a visualization based on the type of data. 3. Keeps on applying exploration operators until a relevant result is reached. Then, downloads the data as a CSV. From the perspective of the policy makers user group, it would be also very interesting if it were possible to save the exploration sequence itself. That way it would be repeatable in the future (which could be interesting since the datasets are periodically updated). This could be done by storing previous explorations as part of the user's profile in the system or maybe even downloading it in some format.
Statistician	Downloads relevant portions of raw data from the system	Regarding specific queries: 1. Selects an NL query from a predefined catalogue or writes an NL query and refines it by adding the appropriate filters. 2. Downloads the result as a CSV. Regarding exploration: 1. Writes an NL query to get an initial set of objects. Alternatively, selects a predefined query from a catalogue 2. Applies an exploration operator on the result of the first query, most likely by-example and/or by-analytics 3. Keeps on applying exploration operators until a relevant result is reached. Then, downloads the data as a CSV.

Citizen	Accesses a public website with some story-telling and predefined visualizations.	Interacts with the visualizations by setting filters and in some cases, by clicking on a certain part of the visualization to get more details on certain aspects, and with result summaries explained in natural language.
Data administrator	Edits the mappings between the database(s) and the ontology in the context of the VKG approach.	After an update of an already integrated data source, new attributes may be available (e.g., in CORDIS at some point the VAT of the participants was added). To make this new information available in the system, the data administrator has to: 1. Get access to the mappings 2. Edit them 3. Refresh the system. If a new data source is to be integrated, the data administrator has to: 1. Use the mapping bootstrap functionality to get a first mapping candidate. 2. Refine the proposed mapping. 3. Refresh the system.
System administrator	Manages the overall system	Either logs in and makes changes to the system's configuration from within the system itself (e.g., via an online interface), or just edits configuration files directly on the server's filesystem.

Who will be the first user groups to use and test the system?

Since the main user profile of this use-case is that of policy maker, a group of users matching this profile will be selected. Three people within SIRIS (the use-case provider) that work in collaboration with policy makers and that are not involved in the INODE project have been selected to mimic this profile. The first user group will have a level of technical skills higher than the average policy maker, however we consider that this is required in order to test the first prototype, since it will most likely not be as user-friendly as the final version. For later iterations, with a more refined user interface, we will identify a second group of test users with technical knowledge closer to that of the average end user.

3.2.3 Data provided and query examples

Data provided

Datasets for this use-case come from open data sources in the Research and Innovation field. The main one is the **CORDIS dataset**, which includes data on European-funded research projects, but we also have data from other **European projects such as Erasmus+**. Other available datasets contain data that was collected at a more national level. For example, we

have several open datasets from the Italian university system, including aggregated numbers of students, researchers, courses, etc.

Other datasets of interest (which SIRIS has not yet worked with) include **Patents (the European Patent Office (EPO) has an RDF dump) and research outputs from projects (publications, deliverables, etc.)**. The latter could be obtained from OpenAIRE, which also has an RDF dump, and could connect these research outputs to CORDIS projects.

In terms of size, the datasets that we have dealt with so far are not huge. The following examples are representative of the size of datasets that we have dealt with in the past:

- CORDIS has around 50k projects
- Erasmus+ has 127k projects
- the Italian student data has a dozen tables with numbers of rows around 100k

The EPO and OpenAIRE datasets would be significantly larger. For example, EPO has more than 24 million patents.

The data **update frequency** depends on the data source, but it is not high. It ranges from **every two or three months to once or twice a year**.

The format of the data also depends on the data source. It can be XML, CSV, or in some cases extracted from the web. We have ETL processes that perform webscraping and import the data into a relational database.

Queries (NL + SQL)

These are a set of individual queries executed on the CORDIS dataset from our clients.

Query 1

Natural language: Retrieve the yearly percentage of H2020 and FP7 funding obtained by the region Tuscany.

SQL: See UC2-Q1 in Annex.

Description: This query does two aggregations. The first calculates the sum of the funding obtained by participants in CORDIS projects (filtered by H2020 and FP7 programs) that are located in Tuscany (the NUTS 2 code of Tuscany is “ITI1”, where NUTS is the “nomenclature of territorial units for statistics” used by Eurostat) for each year. The second calculates the total funding given by the European Commission to H2020 and FP7 projects each year. Finally, the query calculates the percentage of funding Tuscany has received from H2020 and FP7 projects each year.

Query 2

Natural language: Calculate the number of projects and funding of the Tuscan participants in H2020 projects, showing also their activity type.

SQL: see UC2-Q2 in Annex.

Description: This query retrieves the participants in CORDIS projects (filtered by the H2020 program) that are from Tuscany and shows the activity type for each of them (i.e. the type of organization), the number of projects in which it has participated, and the total funding received.

Query 3

Natural language: Show, for each Tuscan participant in H2020 and FP7 projects and their partners, the list of programs in which they have participated each year.

SQL: see UC2-Q3 in Annex.

Description: This query lists both Tuscan organizations and partners of Tuscan organizations in CORDIS projects (H2020 and FP7 programs), showing, for each year, the programs in which they participated.

Query 4

Natural language: Count the number of ERC projects of Spain and Portugal for each research domain and year.

SQL: see UC2-Q4 in Annex.

Description: This query calculates the number of ERC projects (within the H2020 program) of Spain and Portugal, grouped by year and research domain. Research domains are derived from the panel that evaluated each program.

Query 5

Natural language: Retrieve the top 100 partners of Tuscan private companies in H2020 and FP7 programs, showing the partner's nationality and the number of collaborations.

SQL: see UC2-Q5 in Annex.

Description: This query creates a ranking with the top 100 partners of Tuscan private companies in the FP7 and H2020 programs. It shows the nationality of each partner (Tuscan, Italian or Non-Italian) and the number of projects they have in collaboration with Tuscany.

3.2.4 Use case detailed description

In this section, we describe the functional requirements in more detail for the policy maker, statistician, citizen, and data administrator profiles. Figure 3-4 shows the overall use case diagram. Further details are provided in the tables below.

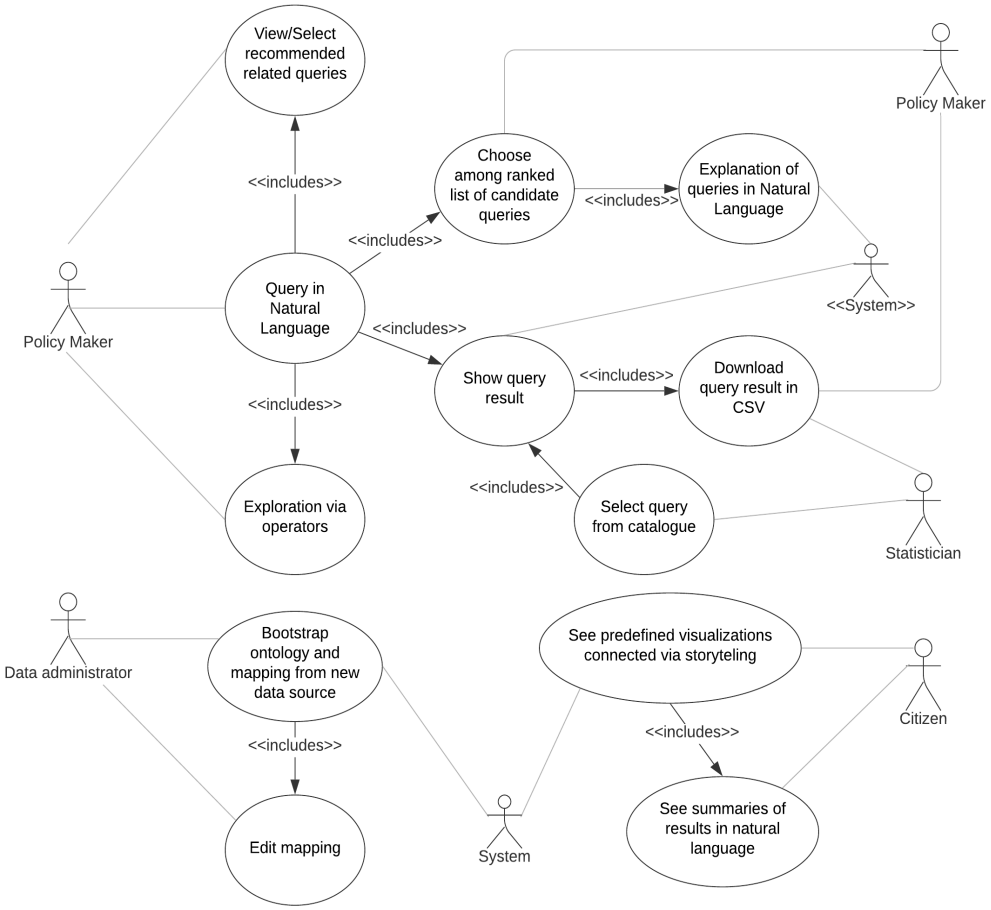

Figure 3-4 *Use case diagram for R&I.*

Table 3-2 *Detailed view of the R&I use case**.***

Name	Query in natural language
Identifier	UC1.1
Description	The user writes a query in natural language. The system executes the query and displays the result.
Goal	To allow the user to formulate a query to the system using natural language and get the result.
Scope	Query answering and exploration.
Preconditions	The user provides the natural language text in English.
Post conditions	The result of the query is displayed.
Actors / Users	Policy maker
Dependencies from other functionalities/steps	None
Exceptions	No candidate query can be generated.

	The query result is empty.
Notes/Comments	None

Name	Choose among a ranked list of candidate queries
Identifier	UC1.2
Description	Due to the ambiguity of a natural language, the system replies with a ranked list of candidate queries that match the user's request.
Goal	To allow the user to disambiguate their initial natural language expression.
Scope	Query answering and exploration.
Preconditions	The user provides the natural language text in English.
Post conditions	The candidate queries are displayed, ranked according to the likeliness of matching the user's intention.
Actors / Users	Policy maker
Dependencies from other functionalities/steps	UC1.1
Exceptions	No candidate query can be generated.
Notes/Comments	None

Name	Explanation of queries in natural language
Identifier	UC1.3
Description	The system explains each candidate query to the user using natural language.
Goal	To provide a natural language description of the candidate queries that correspond to the user's initial natural language expression.
Scope	Query answering and exploration.
Preconditions	The natural language text in English is provided.
Post conditions	The candidate queries are displayed, each of them annotated with a natural language description of its meaning.
Actors / Users	System
Dependencies from other functionalities/steps	UC1.2
Exceptions	No candidate query can be generated.
Notes/Comments	None

Name	Show query result
Identifier	UC1.4
Description	The user writes a query in natural language and, after selecting the appropriate candidate, the result of the query is displayed.
Goal	To show the result of the natural language query to the user.
Scope	Query answering and exploration.

Preconditions	The natural language text in English is provided.
Post conditions	The query result is displayed.
Actors / Users	System
Dependencies from other functionalities/steps	UC1.1, UC1.6
Exceptions	Query result is empty.
Notes/Comments	None

Name	Download query result in CSV
Identifier	UC1.5
Description	The user downloads the query result in CSV format.
Goal	To allow the user to download the result of the disambiguated query in CSV format. Such a query is the result of choosing one of the candidate queries that match the original ambiguous natural language expression issued by the user.
Scope	Query answering and exploration.
Preconditions	A non-ambiguous query is provided.
Post conditions	The result of the query is sent back to the user in CSV format.
Actors / Users	Policy maker, statistician
Dependencies from other functionalities/steps	UC1.4
Exceptions	Query result is empty.
Notes/Comments	None

Name	Select query from catalog
Identifier	UC1.6
Description	The user selects a natural language query from a predefined catalog. The query may have some filters with placeholders that the user can fill in so as to obtain just the data that is relevant. The disambiguation of the query should not be necessary, either because the query is already formulated in a non-ambiguous way, or because the appropriate candidate query is already known by the system.
Goal	To show a predefined catalog of frequently used queries that the user can refine by filling in some of the offered filters.
Scope	Query answering and exploration.
Preconditions	A catalog of queries, potentially with some filters, has been defined in the system.
Post conditions	The user has selected one query from the catalogue and filled in some of its filters.
Actors / Users	Statistician
Dependencies from other functionalities/steps	None
Exceptions	The catalog is empty.
Notes/Comments	None

Name	View/select recommended related queries
Identifier	UC1.7
Description	After performing a query in natural language, the system recommends to the user some other related queries, also formulated in natural language.
Goal	To recommend to the users queries in natural language that are related to the previous search.
Scope	Query answering and exploration.
Preconditions	The user has already issued a query to the system, which has been successfully executed.
Post conditions	A list of natural language queries related to the previous user query is displayed.
Actors / Users	Policy maker
Dependencies from other functionalities/steps	UC1.1
Exceptions	No previous query has been successfully issued by the user.
Notes/Comments	None

Name	Exploration via operators
Identifier	UC1.8
Description	After performing a natural language query to get an initial set of objects, the user can construct, in an iterative way, a pipeline of exploration operators.
Goal	To allow the user to explore the data via the iterative application of operators.
Scope	Query answering and exploration.
Preconditions	The user has already issued a query to the system, which has been successfully executed.
Post conditions	The user has constructed a pipeline of exploration operators, starting with the set of objects resulting from the previously issued query.
Actors / Users	Policy maker
Dependencies from other functionalities/steps	UC1.1
Exceptions	No previous query has been successfully issued by the user.
Notes/Comments	None

Name	See predefined visualizations connected via storytelling
Identifier	UC1.9
Description	Citizens can access a website with a predefined set of visualizations that are put in context by a narrative.
Goal	To communicate to the general public, using data and a storytelling approach that provides context to this data.
Scope	Storytelling.
Preconditions	The visualizations and storytelling have been defined in the system.
Post conditions	The user sees the visualizations and connecting narrative in the browser.

Actors / Users	Citizen
Dependencies from other functionalities/steps	None
Exceptions	The website is not available.
Notes/Comments	None

Name	Bootstrap ontology and mapping from new datasource
Identifier	UC1.10
Description	The system assists the data administrator in integrating a new data source by generating an initial ontology and set of mapping rules.
Goal	To create an ontology and mapping based on a new data source structure.
Scope	Data integration.
Preconditions	The parameters necessary to connect to the data source are provided.
Post conditions	An ontology and mapping for the data source have been generated.
Actors / Users	Data administrator, system
Dependencies from other functionalities/steps	None
Exceptions	The data source is not accessible. The specification of the data source is not rich enough to allow the construction of an ontology and mapping.
Notes/Comments	None

Name	Edit mapping
Identifier	UC1.11
Description	The data administrator gets access to the mapping for a particular dataset and edits it via some interface.
Goal	To allow the data administrator to refine existing mappings for the integrated data sources.
Scope	Data integration.
Preconditions	An ontology and mapping are available for a given data source.
Post conditions	The data administrator has edited the mapping, and potentially also the ontology specification.
Actors / Users	Data administrator
Dependencies from other functionalities/steps	UC1.10
Exceptions	No mapping and ontology are available for the given data source.
Notes/Comments	None

Name	See summaries of results in natural language
Identifier	UC1.12
Description	Citizens can see the results of a set of predefined queries explained in natural language.

Goal	To complement the storytelling addressed to the citizen's with a natural language explanation of relevant data.
Scope	Storytelling
Preconditions	The queries to be explained have been defined in the system.
Post conditions	The user sees the natural language explanations in the browser.
Actors / Users	Citizen
Dependencies from other functionalities/steps	UC1.9
Exceptions	The website is not available.
Notes/Comments	None

3.2.5 Requirements Summary

The requirements elicited by this specific use case are provided in the consolidated table (Table 5-1) under Category R1, Group 2 (specifically the requirements R1.2.2-R1.2.4, R1.2.6-R1.2.17 and R1.2.27).

3.3 Astrophysics

3.3.1 Overall description

INODE will allow scientists to react immediately to new events by taking away the painstaking procedure of finding out what other relevant datasets exist, and then going through the process of familiarizing oneself with those to the level where scientific questions can actually be asked. Examples from astrophysics include - but are by far not limited to - supernovas, gamma ray bursts, gravitational wave events, or the recently discovered elusive fast radio bursts. Unpredicted occurrences of new properties or clusters of data naturally make scientists want to compare them against previous data. These are often stored in very different formats and/or locations.

Modern astrophysics collaborations need to analyze dozens of databases at a time. As a consequence, it becomes increasingly challenging for scientists to penetrate the structure of the data and their metadata to generate scientific knowledge. INODE takes away the burden from scientists to design several thousand-character-long nested SQL statements and programming routines by providing them with more intuitive means to interact with the data, such as to simply ask in NL "What is the density of blue and star forming galaxies in a certain region of the sky?". INODE will speed up the ability to test scientific questions and to probe if their detailed pursuit is promising. INODE will provide an interface to data to allow for these questions to be formulated as naturally as possible. Additionally, these questions will be answered in an efficient manner.

On the one hand, unusual objects will be discovered faster, and grouped together more quickly through queries by-example. Once groups of objects have been identified, fundamental questions about their nature can be asked (e.g., "When in cosmic time were these galaxies built, what are their other intrinsic properties like star formation rates or metallicity?"). Additional information, e.g., spectroscopic data for objects discovered in

imaging surveys, or UV/infrared/radio data for objects discovered by gravitational wave emission will be accessible, convenient and available to the entire astrophysical community rather than just experts of the respective databases (Figure 3-5). Even theoreticians or enthusiasts will be able to generate queries without needing to understand the intricacies of the many underlying datasources, test their ideas and generate theories on the nature of newly discovered objects. INODE is expected to speed up the scientific process, and increase the amount of generated knowledge per time unit.

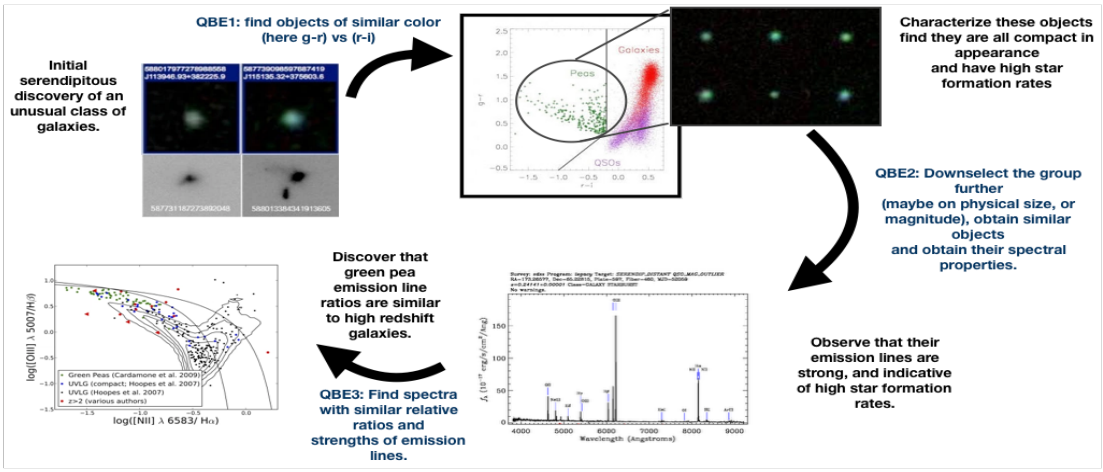

Figure 3-5 *Green Pea galaxies recently gained a fair bit of attention in astronomy as one of the potential sources that drove cosmic re-ionization. First discovered in the Galaxy Zoo* project[2]*, they appeared green and compact in the recorded imaging data and fall into an unusual region of color-color diagrams of galaxies. A consequent selection in color space led to a detection of a large sample of objects that all exhibit strong emission lines and unusually high star formation rates that are much more common in the very young Universe. The figure shows three queries by example (QBE) of analyzing astrophysics data with INODE.*

Here are some specific workflows of an astrophysics analysis: **The analyst is a domain expert who wants to identify galaxies of interest.** The analyst starts with an unusual class of galaxies and uses **by-example** to find groups of similar galaxies. The analyst observes that the results are all compact in appearance and have high star formation rate. The analyst uses **by-facet** to break down that group by density, size, magnitude etc. The analyst uses **by-query** to obtain spectral properties of objects. The analyst observes that their emission lines are strong and asks for **by-analytics** on those distributions to find groups of objects with similar and dissimilar emission lines. The analyst finds spectra with similar relative ratios and strength of emission lines. Once groups of objects have been identified, the analyst can use **by-NL** to answer "When in cosmic time were these galaxies built, what are their other intrinsic properties like star formation rates or metallicity?"

[2] Galaxy Zoo (https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/) is a popular and rather successful implementation of the crowd sourcing idea: It lets, through a very simplified online interface, inspect large quantities of astronomical (and now also other objects), by users drawn from the general public. This allows a much larger number of objects to be screened that would be feasible though the limited group of scientists.

3.3.2 Stakeholders definition and mode of interaction

Who are the users/stakeholders?

We foresee the following principal classes of users:

DB Administrator

Responsible for hosting the databases, access (e.g., Web) and the INODE infrastructure and architecture. Should retrieve data and access permissions from the data curators (see b).

Data Curators

These will be mostly scientists with access to specific astronomical data sets either through their affiliation with a particular project or a specific astronomical institution. Their responsibility is to i) make the data available in a format (e.g., SQL dump) that is ingestible by the tools developed within INODE and to provide appropriate data mappings ii) as well as define access rights. For proprietary or partially proprietary datasets, access needs to be restricted to the subgroup of users (see below) with appropriate access rights. The data curators will carry the responsibility to set the permissions and to update the permissions as proprietary periods are ending. We don't foresee any "write" access to databases. The only level of access control to particular tables will be "is allowed to read" or "is not (yet) allowed to read".

Scientists with proprietary access

These will be mostly scientists with access to specific astronomical data sets either through their affiliation with a particular project or a specific astronomical institution. The scientists will not change the data sets but they will be allowed to access them (read only).

General public

Many/most astronomical datasets are open to the general public or do become public after some proprietary period. The "general public" should be the generic unauthenticated user that has read access to all data that does not fall under some proprietary protection. "General public" specifically also includes non-astronomers.

How is every stakeholder/user interacting with application/service?

The following table give an overview of how the stakeholders interact with the various applications and services of INODE.

Table 3-3 *Astrophysics Stakeholders table.*

Stakeholder/User	***Role***	***Interacting functionalities***
DB administrator	Hosts databases, INODE infrastructure.	Will be notified by data curators as new data sources become available.
Data curator	Responsible for integrating new data sources.	Defines data mappings.

	Responsible for updating already ingested data sources, e.g., after data releases. Appropriately forwarding access control to datasets (proprietary vs. non-proprietary datasets).	Delivers data sources to DB administrators.
Scientists with proprietary access	Performs queries to the DB.	Executes queries in natural language. Receives answers in tabular form (ideally in a new table). Discerns whether the result answers the questions, refines the query if needed. May want to rerun a modified query or a different query based on the results of the first. This may involve visualisations of the data to define/find interesting parameter ranges to define sub populations of datasets.
General public	Same as the Scientist, but with access only to unprotected datasets.	

Who are going to be the first user groups to use and test the system?

The astronomers involved in INODE will be the immediate test users for the system. An effort will be made to extend the group to other astronomers. Allowing access to the general public (or a proxy for the general public, e.g., students) is a goal but will require a relatively mature system.

3.3.3 Data provided and query examples

Data provided

As a first trial dataset a dump of the SDSS database has been provided.

The latest version is DR16. Beyond this, potential datasets encompass the databases from the Dark Energy public data release, the PanSTARRS public data release, proprietary data from the HETDEX project and the eRosita space mission (https://www.mpe.mpg.de/450415/eROSITA).

Queries (NL + SQL)

A number of SDSS queries have been made available to the project that exemplify the interaction of researchers with the SDSS dataset and a sample of such queries are listed below.

Query 1

Natural language: Retrieve the unique object ID and coordinates of 100 photometric objects that have the right ascension between 185 and 185.1, and declination between 15 and 15.1.

SQL: See UC3-Q1 in Annex

Description: This sample query searches the sky position defined by ra and dec, finds unique objects in an ra/dec box and prints the unique object IDs and coordinates of the first 100 records.

Query 2

Natural language: Retrieve the unique object ID and the distance of the galaxies within 1 arcmins of an equatorial point with ra =185.0, dec =-0.5.

SQL: See UC3-Q2 in Annex

Description: This sample query searches the galaxies within 1 arcminutes of ra, dec and order it with respect to the distance.

Query 3

Natural language: Retrieve the photometric properties and redshift of 10 stars from all fields whose U magnitude is between 0 and 15

SQL: See UC3-Q3 in Annex

Description: This query does a table JOIN between the imaging (PhotoObj) and spectra(SpecObj) tables of stars with U magnitude in between 0 and 15.

Query 4

Natural language: Give me the colors of a random 1% sample of galaxies from all fields

SQL: See UC3-Q4 in Annex

Description: This query uses htmID as a random number generator. HtmID is multiplied by a prime number (37) to remove bias and then constrained to be between 650 and 65000 to generate 1% random sample of data.

Query 5

Natural language: finds galaxies that have clean photometry with at least 10 Galaxy Zoo volunteer votes and at least an 80% probability of being clockwise spirals.

SQL: See UC3-Q5 in Annex

Description: This query identifies the galaxies with 80% probability of being clockwise spirals and voted by minimum of 10 Galaxy Zoo volunteer.

3.3.4 Use case detailed description

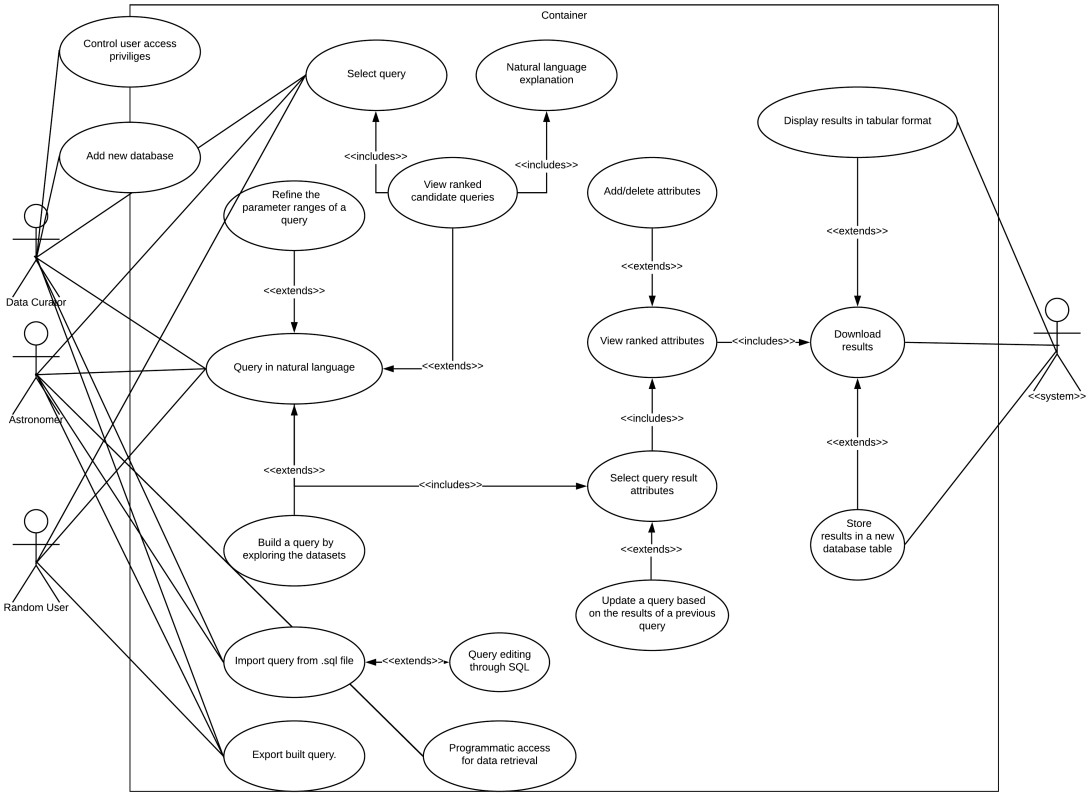

Figure 3-6 Use case diagram for astrophysics.

Table 3-4 *Detailed view of the Astrophysics use case**.***

Name	Querying in natural language
Identifier	UC2.1
Description	The user writes a query in natural language to be executed over the astronomical datasets. As a result, ranked candidate queries generated based on the user input are available along with some NL explanations. In doing so, the system provides other query options “interpreted” by the system. By default, the results of the best ranked candidate query are retrieved in a tabular format. For long result sets an appropriate example subset is displayed.
Goal	To retrieve results of a given query.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user has to write and submit a query in English.
Post conditions	The information requested is displayed.
Actors / Users	Data curator, astronomer, and random user

Dependencies from other functionalities/steps	None
Exceptions	No results are found. The query is too general. It was not possible to generate any candidate query or none of them are significant.
Notes/Comments	Candidate queries are considered significant if they do satisfy any aspect of the user intent.

Name	Selecting query result attributes (the query projection)
Identifier	UC2.2
Description	This use case is to define which attributes are expected by the user to answer her/his question. After selecting a candidate query, the user can be interested in refining the query by adding and/or removing attributes. For example, let us consider the following question: "which galaxies are present in a particular sky area?". The answer for this question can include one or more of the following attributes: A magnitude in a specific band, projected size, various model fit parameters.
Goal	To allow the user to choose which attributes compose the answer for his/her question.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user has to select which attributes will be projected
Post conditions	The information requested is displayed
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.1
Exceptions	None
Notes/Comments	None

Name	Downloading results
Identifier	UC2.3
Description	It provides the possibility to download the results as a ASCII file (various formats), a spreadsheet (e.g. .xlsx), a VOTable, a FITS table, and HDF5 file.
Goal	To allow the user to download the results.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user has to choose the format to export the results.
Post conditions	The file is downloaded
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC2.7, UC2.8, UC2.9, UC2.11
Exceptions	No retrieved results to export.
Notes/Comments	None

Name	Storing results in a new database table
Identifier	UC2.4
Description	It provides the possibility to store the results in a new database table. This allows the researcher to use the result set programmatically to generate plots or drive other routines.
Goal	To allow the user to download the results.
Scope	Data curator, astronomer
Preconditions	The user has to choose the format to export the results.
Post conditions	The file is downloaded
Actors / Users	Data curator, astronomer
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC2.7, UC2.8, UC2.9, UC2.11
Exceptions	No retrieved results to export.
Notes/Comments	None

Name	Displaying results
Identifier	UC2.5
Description	The results are mainly shown in a tabular format. New columns can be added according to UC2.2. When applicable, it can also display possible infographics related to the results. If the result table is long, an appropriate subset (e.g. first 100 rows) should be displayed.
Goal	To allow the user to visualize and read the results, i.e question answer.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user has to submit a query before (see UC2.1).
Post conditions	The information requested is displayed.
Actors / Users	Data curator, astronomer
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC2.7, UC2.8, UC2.9, UC2.11
Exceptions	No results to display.
Notes/Comments	None

Name	Visualizing large controlled vocabularies and ontologies. Give detailed scientific explanation of attributes.
Identifier	UC2.6
Description	The number of large astronomical datasets is growing quickly. A detailed understanding of database schemas and all attributes becomes a burden to the astronomer and an impossibility to the random user. As INODE is specifically supposed to allow users to query multiple datasets, this must be mapped into a common ontology. A precise description of the ontological term meanings and possible caveats shell be displayed here: Not all mappings may be precise. A g band magnitude may mean something different in different datasets. The user must be made aware of this.

Goal	To show a visual summarization of ontological term meanings.
Scope	Data curator, astronomer, and random user profiles
Preconditions	ontology/taxonomy/controlled vocabulary
Post conditions	Visual summarization of one or more term semantics
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.5
Exceptions	The file could not be loaded. Term does not exist.
Notes/Comments	None

Name	Building a query by exploring the datasets
Identifier	UC2.7
Description	The user writes a query in natural language. As a result (s)he obtains a subgraph that represents in an abstracted way how the data used to answer a question is structured and related to other data items (i.e. an explicit view of the data schema). Value examples are also displayed when highlighting or selecting a given attribute of a graph node (that can be interpreted as a class). Displaying attribute values is important for the user because (s)he can find out what kind of projected results to expect when executing the final built query. In this subgraph, not only an explicit portion of the data schema to answer the initial question is shown but also the closest neighbor nodes and their attributes. This is done to allow the user to explore the datasets and expand the initial query. Moreover, when expanding the initial query, the user has the possibility to specify attribute values, consequently, (s)he restricts the retrieved results similar to a WHERE clause in SQL.
Goal	To allow the user to interactively build a query based on an initial exploratory question.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user has to write in English and submit a query.
Post conditions	A subgraph that represents how the data are structured to answer an exploratory query and their final results.
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC2.5, UC2.8, UC2.9, UC2.10, UC2.11
Exceptions	No data found to answer the query.
Notes/Comments	None

Name	Refining the parameter ranges of a query
Identifier	UC2.8

Description	The user has written a query in natural language and the result set generally is satisfactory. But now the user wants to modify the parameter range of a query. For instance, different magnitude ranges in different photometric bands are queried. The SQL equivalent would be a modification of (UMIN, UMAX, GMIN, GMAX in the following query): select * from galaxy where u > UMIN and u < UMAX and g > GMIN and g < GMAX
Goal	To allow the user to interactively refine a query based on an initial exploratory question.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user must have already written a query in English and submitted the query.
Post conditions	An updated query result (or appropriate subset thereof) is displayed.
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC2.5, UC2.10, UC2.9, UC2.10, UC2.11
Exceptions	No data found to answer the query.
Notes/Comments	None

Name	Update a query based on the results of a previous query
Identifier	UC2.9
Description	The user has written a query in natural language and the result set generally is satisfactory. Based on the results the user now wants to explore the dataset in a different way. “Based on the previous result, now show me all the objects that have a star formation rate of more than a 100 solar masses per year.”
Goal	To allow the user to interactively extend a query based on an initial exploratory question.
Scope	Data curator, astronomer, and random user profiles
Preconditions	The user must have already written a query in English and submitted the query.
Post conditions	An updated query result (or appropriate subset thereof) is displayed.
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC2.5, UC2.7, UC2.7, UC2.10, UC2.11
Exceptions	No data found to answer the query.
Notes/Comments	None

Name	Exporting built query.
Identifier	UC2.10
Description	It provides the functionality to save the built query into a SQL query file (.sql files).
Goal	To export the built query in a .sql file.
Scope	Data curator, astronomer, and random user profiles

Preconditions	Built a query based on UC2.6 or UC2.9.
Post conditions	Display of the SQL or a downloadable file for the SQL is made available.
Actors / Users	Data curator, astronomer, and random user
Dependencies from other functionalities/steps	UC2.1, UC2.2, UC, 1.7, UC2.8, UC2.9, UC2.11
Exceptions	No data found to answer the query

Name	Importing query from .sql file.
Identifier	UC2.11
Description	It provides the functionality to import a query from a .sql file. The query can be then executed by the user. Edition is enabled within a SQL query editor for the data curator and astronomer user profile.
Goal	To import a query from an .sql file into the INODE system.
Scope	Data curator and astronomer user profiles
Preconditions	SQL query stored as a .sql file.
Post conditions	Possibility to execute or edit the loaded query.
Actors / Users	Data curator and astronomer
Dependencies from other functionalities/steps	UC2.10
Exceptions	Syntax file is not compliant with the rq format. No data found to answer the query.
Notes/Comments	None

Name	SQL
Identifier	UC2.12
Description	It provides a SQL editor with auto-complete feature.
Goal	SQL query edition.
Scope	Data curator and astronomer user profiles
Preconditions	Connection to a data store that supports SQL.
Post conditions	Execution of the query along with results
Actors / Users	Data curator and astronomer user
Dependencies from other functionalities/steps	None
Exceptions	No results found. SQL query syntax error. Connecting to the data store was not possible.
Notes/Comments	None

Name	REST like interface for the programmatically retrieval of data
Identifier	UC2.13

Description	Astronomers often interact with data systems through programmatic interfaces. Ideally a Python module allows the researcher to submit a particular query and to retrieve the result.
Goal	Programmatic interface.
Scope	Astronomer profile.
Preconditions	Connection to the data store.
Post conditions	Execution of the query along with results.
Actors / Users	Astronomer.
Dependencies from other functionalities/steps	None
Exceptions	No results found. Query syntax error. Connecting to the data store was not possible.
Notes/Comments	None

3.3.5 Requirements summary

The requirements that have been elicited from this specific use case are fully listed in the consolidated table (Table 5-1) under Group 2 (R1.2.1 - R1.2.5).

3.4 Cancer Biomarker Research

3.4.1 Overall description

The main purpose of the INODE project in the context of the cancer biomarker research use case is to facilitate and precisely answer queries over multiple cancer-related datasets. These queries are written in natural language. Moreover, the difficulty of answering such queries and "understanding" user intent is tackled by an information discovery functionality that interactively guides the user over the available data and metadata (e.g., ontologies).

Screenshots of the INODE usage for medicine are given in Figure 3-7 and Figure 3-8. For Cancer Biomarker Research, users are scientists, including biologists, bioinformaticians, data scientists, medical doctors, etc. It is important that scientists, without prior training, be able to perform powerful queries across several data sets in ways that cannot necessarily be anticipated. These requirements go far and beyond the query functionality of existing query interfaces. INODE will **accelerate extraction of useful knowledge from these data by enabling sophisticated semantic queries across large data sets**.

Furthermore, there is no conventional representation for some of the data integrated in OncoMX (e.g., biomarker information, provenance information). These representations vary between sources, and are likely to evolve, which makes the user task of knowledge extraction from OncoMX more challenging. For instance, several efforts aim at identifying relevant cancer biomarkers. While OncoMX integrates data from the Early Detection Research Network (EDRN), it is necessary to link this information to the list of FDA approved

biomarkers[3], or to the biomarker qualification opinions from the European Medicines Agency[4]. INODE will **allow scientists to combine the information present in OncoMX with data provided by different sources in an easier and more automated way** by proposing mappings to scientists. There is currently no resource linking all the data available, a difficulty strongly impacting all cancer research, on which the EU spends about 3 billion Euros per year [1]. **We expect INODE to improve cancer biomarker research and to speed up the translation from research to medical application.**

The following figures demonstrate the most important features of INODE. Figure 3-7 shows a natural language query with user assistance. INODE parses the query, provides hints for autocompletion and disambiguates terms. Figure 3-8 visually clusters cancer types retrieved by the natural language query. The user can explicitly choose the distance metric for the cancer types. These examples demonstrate a powerful interplay between user-assisted natural language queries and visual exploration.

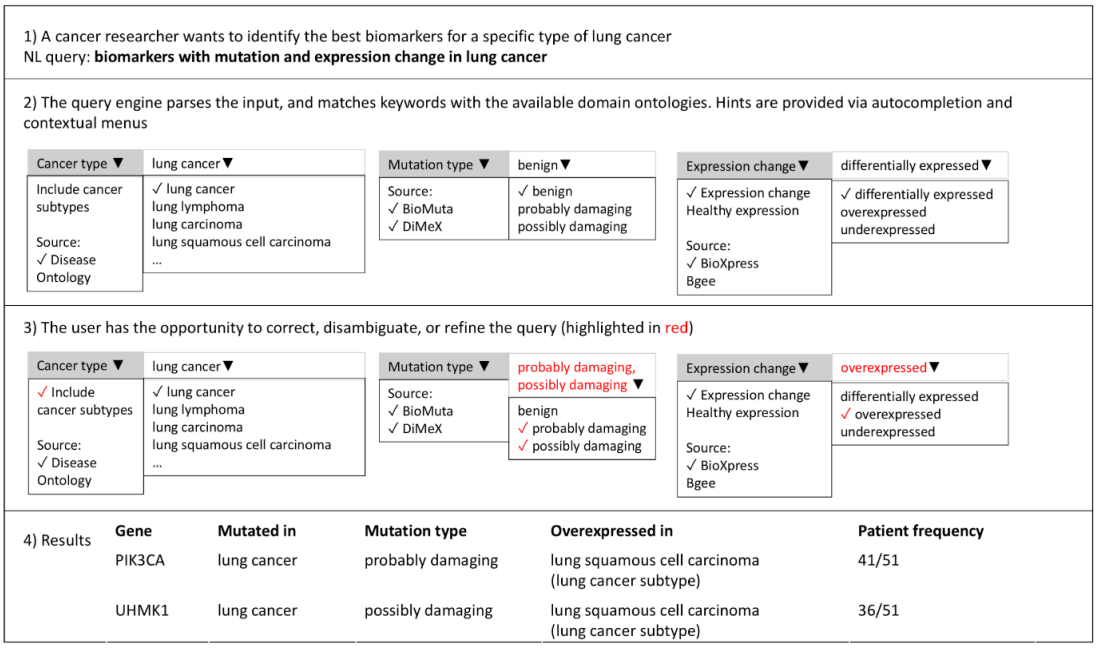

Figure 3-7 *Natural language query interface with user assistance. Step 1: user enters query in natural language. Step 2: INODE parses query and matches keywords against the available ontology. Step 3: INODE provides user assistance by disambiguating terms and suggesting alternatives. Step 4: results are shown.*

[3] https://www.fda.gov/Drugs/DevelopmentApprovalProcess/DrugDevelopmentToolsQualificationProgram/BiomarkerQualification- Program/ ucm535383.htm

[4] https://www.ema.europa.eu/en/human-regulatory/research-development/scientific-advice-protocol-assistance/qualification-novel- methodologies-medicine-development#section2

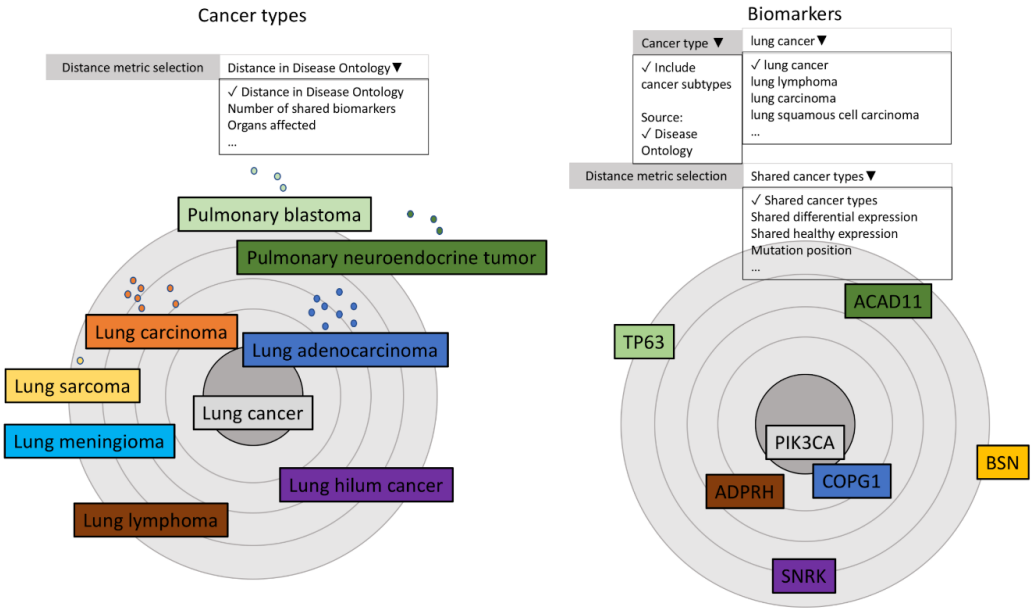

Figure 3-8 *Visualization of cancer types identified with the natural language query in Figure 8. The left-hand side shows various cancer types that are similar to lung cancer. The distance between the diseases can be chosen by the user, e.g., by distance in disease ontology. The right-hand side shows biomarkers related to lung cancer. Here, the cancer type and the distance can be chosen by the user.*

A typical analytics workflow looks as follows: **The analyst is a domain expert (biologist) who wants to identify which markers characterize which cancer type (among c1, c2, c3) for some demographic groups**. The interface given to that analyst is the **OpenDataDialog**. The analyst starts with a group of middle-age male patients in Zurich who have different types of cancer (c1, c2, c3). To expand that set, the analyst calls **by-example** to find k similar sets of patients where the similarity is by cancer type. For each group, the analyst calls **by-facet** to breakdown the group into three groups, one for each cancer type. The analyst then calls **by-analytics** to find a set of k groups for each group whose markers have the same distribution as that group. For each obtained group, the analyst wants to know those with the same demographics. That is obtained by invoking **by-attribute-similarity** on each group to obtain subgroups with the same demographics and cancer type. The analyst can iterate over those groups in the same way as the starting group. The observations resulting from this iteration associate markers to cancer types and demographics.

3.4.2 Stakeholders definition and mode of interaction

Who are the users/stakeholders?

Researchers and biologists

This profile is mainly composed of scientists that are interested in cancer biomarkers and related data with, but not limited to, the following roles: research investigator, rare disease researcher, research associate, biomedical researcher, biomarker researcher, regulatory

scientist, PhD candidate and post-doc in life sciences, biocurator, cancer glycobiologist, cancer program officer, and master or undergraduate student in life-sciences.

In this profile, we do not expect the researcher to know a structured query language such as SPARQL or be familiar with any computer engineering concepts (e.g., a programming language). When querying in natural language (NL), researchers expect results in a tabular and exportable format such as comma-separated values or spreadsheet file. Moreover, infographics (i.e., a visual representation of information or data) to summarize and highlight information from the tabular results are highly appreciated. The main goal of researchers is to query for relevant information and data that can substantially support their research. So far researchers have to manually navigate and understand the OncoMX data from multiple sources and eventually combine them, with a restricted functionality in terms of searching, for example, per gene name such as TP53. They are thus perfect targets for the NL and exploration services of INODE.

Moreover, we should not expect that this user profile will be aware of the exact information available in the database or the question/query (s)he wants to address. Because of this, these users first want to explore the data and then compose the final query to retrieve the results they are looking for. To exemplify a real natural language query that a researcher would ask, we can consider the following question: "obtain a list of the cancer types that my gene is differentially expressed in with a p value cut off of < 0.01". In the OncoMX project, the identification of differentially expressed genes has been previously computed and explicitly stored. With the results of this query, the researcher can explore significant changes between healthy and disease tissues in the expression of a given gene of interest to identify candidate cancer types for experimental investigation.

Bioinformaticians

This profile is mainly composed of bioinformaticians that have interest in cancer biomarkers and related data with, but not limited to, the following roles: data scientist, ontologist, bioinformatics teacher, clinical bioinformatician, computer scientist, knowledgebase developer, post-doc in bioinformatics, and student.

We can expect these users to have a minimal computer engineering background and to be familiar with some programming languages. This background facilitates and enables them to understand a query language or elaborate NL queries that better fit the system. They can also take advantage of advanced unstructured query functionalities (e.g., search for an exact match, apply logical operators, etc.). In contrast with the non-technical users such as the researcher profile, who expect the question-answer engine to perform the entire process for them, bioinformaticians prefer to get the relevant raw data from the system and do the analysis themselves with their own tools. In this sense, they are more interested in having command line interfaces (CLIs) such as the Shell or Application Programming Interfaces (APIs) for programming languages such as Python and R. These interfaces provide predefined queries they can just parameterize to get the relevant data for their work and *bioinformatics pipelines* (a.k.a. workflows).

Data manager

The data manager is responsible for managing the data sources by integrating new ones and updating the existing ones. The data manager is also responsible for transforming the semi-structured data provided by OncoMX into highly-structured data by creating and applying a relational data model. The data manager also defines data transformations and mappings to

semantically enrich the original data models either physically or virtually (e.g., in the context of the virtual knowledge graph approach). The goal of enriching the data model is to enable and improve the question-answer system.

System administrator

The system administrator is mainly responsible for creating and managing the system profiles. (S)he can create a user with multiple profiles. For example, a user could have both the system admin and data manager profiles.

How is every stakeholder/user interacting with application/service?

Table 3-5 *Cancer Biomarker Research Stakeholders table.*

Stakeholder/User	*Role*	*Interacting functionalities*
Researcher and biologist	Free text querying and information discovery. **Access rights:** only read.	**Question & Answer (QA):** Advanced QA 1. Write a query in natural language. 2. Get a tabular result example together with other recommended candidate queries and related NL explanations. 3. Choose a candidate query. 4. Refine the query by offering the possibility of adding new columns/attributes. 5. Execute the final query and results are available in a tabular-like style and it can include infographics that summarises or represents the retrieved results. The results can also be exported into a spreadsheet format or a simple CSV/TSV file. Note that a spreadsheet format can also include infographics. Simple QA It solely contains steps 1 and 5 from advanced QA. **Information discovery:** 1. Writing a NL data exploration query to get a sub-graph that represents the underlying data at the conceptual level ("data schema"). In this subgraph basic information along with examples are shown to the user to improve understandability of the "data schema". Initially solely nodes and edges associated with the NL query are highlighted. Basic information is, for example, all possible attributes related to a given node. 2. Selecting non-highlighted node attributes, and sub-graph nodes enables to expand the original NL query.

		3. If the user executes the expanded query a new sub-graph is generated to continue data exploration. 4. The user is also able to disable nodes and attributes. By doing so (s)he also changes the original NL exploration query. 5. Finally, once the user built the query (s)he was looking for by data exploration, this query is executed. The results are available in a spreadsheet format or a simple CSV/TSV file. Note that a spreadsheet format can also include infographics.
Bioinformatician	Downloads relevant portions of raw data from the system. **Access rights:** mostly read. Create, read, update, and delete operations are solely available to manage the query catalog.	It includes all functionalities of the "Researcher and biologist" profile, and in addition also the following ones: **Querying**: 1. Selects a NL query along with the SPARQL query from a template catalogue. Queries can also be written by using a simplified structured query language, for example, similar to a command line interface (CLI). In addition, an SPARQL editor with auto-complete is available on this user profile as well. 2. Download the result in a CSV-like and/or JSON format. **Regarding exploration:** SPARQL query bootstrap This user profile can take advantage of the data exploration query generator (described in the previous row) to produce a draft SPARQL query that can be further improved for his/her needs. In doing so, the user can write complex queries that are not automatically generated by the system.
Data manager	Integrates data sources, creates and manages relational data models and edits VKG mappings. **Access rights:** create, read, update, and delete.	**Data integration** The OncoMX data are originally available as CSV files in a semi-structured format. The data manager can use external tools for performing his/her tasks, however the INODE system can provide bootstraps to facilitate his/her tasks of data modeling from tabular-like files into OWL/RDF terminological and assertion data. ● After an update of a dataset by the OncoMX team that is already present in the INODE system, new attributes may be available. To update the INODE system, the data manager has to either manually edit the VKG mapping file or the system could automatically generate a draft of the new mappings to be afterwards validated and edited by the data

		manager. To facilitate the data manager job, the system may highlight the mappings that require a modification because of changes in the table columns (a.k.a. attributes). ● If a new data source is added, the data manager has to: 1. Create a relational model or extend the existing one 2. Use the mapping bootstrap functionality to create a draft of the ontology and VKG mappings. 3. Enrich and refine the ontology concepts and properties. 4. Refine and create new mappings to populate the ontology. **Access rights:** create, read, update, and delete. Create, update and delete operations are not allowed for managing user profiles. These operations are actually part of the system admin role.
System administrator	Manages the overall system. **Access rights:** create, read, update, and delete.	Either (s)he makes changes to the system configuration from within the system itself (e.g., via an online interface), or just edits configuration files directly on the server. **Access rights:** create, read, update, and delete. CRUD is solely applied for system profiles. A user exclusively with the system admin profile cannot delete integrated data by a data manager, for example.

Who are going to be the first user groups to use and test the system?

Once a minimal product (alpha version of the system) is available, Bgee team members will be the first users of the system and will assess the technical and usability aspects. We will also ask the OncoMX team if they can test the system.

3.4.3 Data provided and query examples

Data provided

We have been working closely with the OncoMX team from the US by reporting several issues in the original cancer-related datasets. Critical issues have been addressed by the OncoMX team, which allowed us to define the first release of a relational data model. Originally these datasets are comma-separated files that are actually semi-structured data.

The OncoMX raw datasets considered so far and the related MySQL and PostgreSQL database dumps built by us are available to download in the folder at ftp://ftpbgee.unil.ch/inode/.

Queries (NL + SQL)

Queries Q1 to Q5 focus on the healthy expression dataset of which we (Bgee team) are the owners. The Q6 query is addressed over the differential gene expression dataset. Q1 to Q6 queries have different levels of complexity as described in the "Query characteristics" column in the table below.

Table 3-6 *Cancer Biomarker Research Queries.*

	Natural language text	**SQL query of reference**	**Query characteristics**	**#Results**	**Results (example)**
Q1	List the names and Ensemble identifiers of healthy expressed human genes	see UC1-Q1 in Annex	2 projections on 2-way join tables	32398	# ensembl_gene_id, gene_symbol 'ENSG00000000003', 'TSPAN6' 'ENSG00000000005', 'TNMD' 'ENSG00000000419', 'DPM1' 'ENSG00000000457', 'SCYL3'
Q2	Human anatomic entities at young adult developmental stages	see UC1-Q2 in Annex	2 projections on 3-way join tables with where-clause on stage name + LIKE operator	4	# name, id 'testis', 'UBERON:0000473' 'temporal lobe', 'UBERON:0001871' 'liver', 'UBERON:0002107' 'corpus callosum', 'UBERON:0002336'
Q3	Healthy anatomical entities where the apoc1 gene is expressed	see UC1-Q3 in Annex	2 projections on 3-way join tables with where-clause on gene_symbol	74	# id, name 'UBERON:0000007', 'pituitary gland' 'UBERON:0000082', 'adult mammalian kidney' 'UBERON:0000451', 'prefrontal cortex' 'UBERON:0000458', 'endocervix' 'UBERON:0000473

					', 'testis'
Q4	"healthy human anatomical entities where the apoc1 gene is highly expressed". The same question more precisely could be: "top 10 healthy human tissues with the highest expression of the APOC1 gene"	see UC1-Q4 in Annex	1 projection on 5-way join tables with where-clause on gene_symbol and species name including group by and order by statements with the MAX aggregation function	10	# name 'liver' 'right lobe of liver' 'left adrenal gland cortex' 'right adrenal gland' 'right adrenal gland cortex'
Q5	Visualize the expression of a gene across multiple developmental stages. **Why to answer this question**: to explore changes in expression of a gene of interest over the course of an organism's life cycle	see UC1-Q5 in Annex	4 projections on 5-way join tables with where-clause on gene_symbol . **Remark**: Gene names may imply some ambiguity. To solve this ambiguity one way is to project the species name. Moreover, although it is not explicitly mentioned in the query text, we should only consider healthy tissues.	302	# Tissue, Stage, Score 'corpus callosum', 'adolescent stage (human)', '15500.00' 'pituitary gland', 'human adult stage (human)', '11900.00' 'adult mammalian kidney', 'human adult stage (human)', '9090.00' 'endocervix', 'human adult stage (human)', '6800.00' 'brain', 'human adult stage (human)', '10400.00'

Q6	Obtain a list of the cancer types that my gene is differentially expressed in with a p value cut off of < 0.01	see UC1-Q6 in Annex	1 projection on 3-way join tables with where-clause on gene_symbol and pvalue	7	# name 'breast cancer' 'colorectal cancer' 'kidney cancer' 'lung cancer' 'stomach cancer' 'thyroid cancer' 'uterine cancer'

3.4.4 Use case detailed description

In this subsection, we focus on describing the functional requirements in further detail for the Bioinformatician and Research/Biologist profiles. This is due to the fact that these profiles are the most relevant for the cancer biomarker research case study. Figure 3-9 shows a general use case diagram that involves the UC3.1, UC3.2, UC3.3, UC3.4, and UC3.5 use cases below.

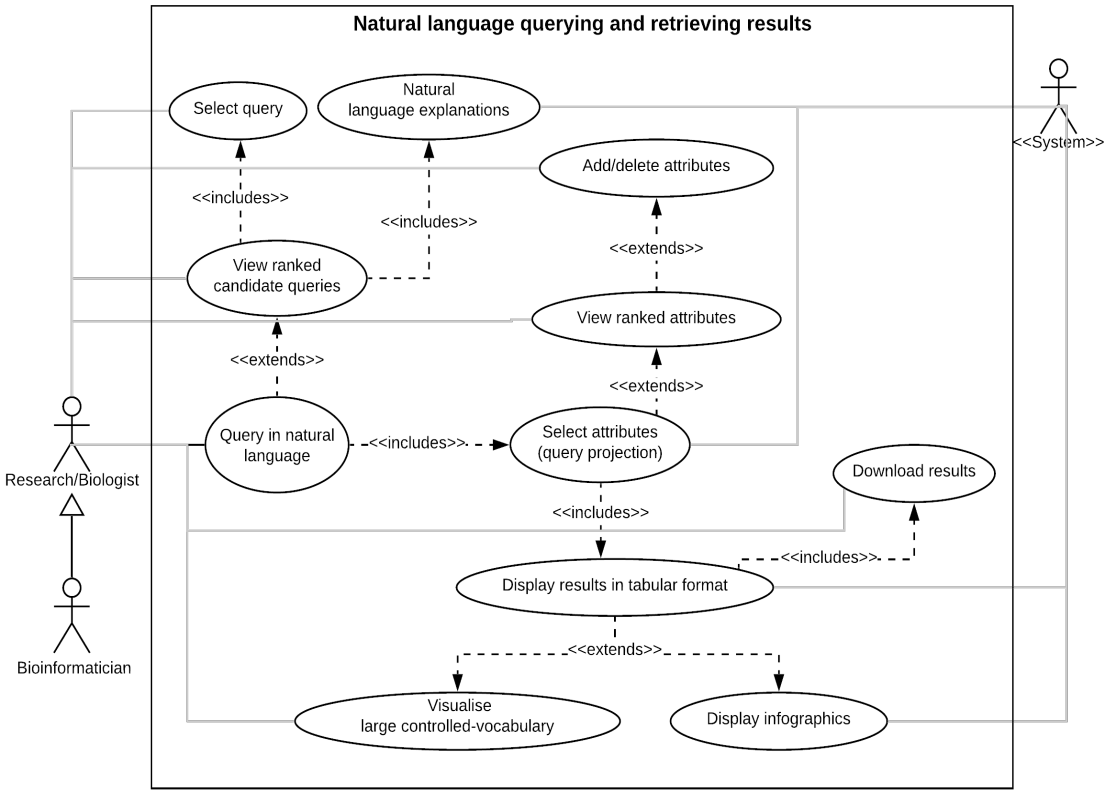

Figure 3-9 *The UML use case diagram for of the Cancer Biomarker Research use case illustrates an overview of the query in natural language and related requirements for the Research/biologist and bioinformatician profiles.*

Table 3-7 *Detailed view of the Cancer Biomarker Research use case.*

Name	Querying in natural language
Identifier	UC3.1
Description	The user writes a query in natural language to be executed over the cancer-related datasets. As a result, ranked candidates queries generated based on the user input are available along with some NL explanations. In doing so, the system provides other query options "interpreted" by the system. By default, the results of the best ranked candidate query are retrieved in a tabular format.
Goal	To retrieve results of a given query.
Scope	Bioinformatician, and researcher and biologist profiles.
Preconditions	The user has to write and submit a query in English.
Post conditions	The information requested is displayed.
Actors / Users	Bioinformatician, researcher and biologist
Dependencies from other functionalities/steps	N/A
Exceptions	No results are found. The query is too general. It was not possible to generate any candidate query or none of them are significant.
Notes/Comments	Candidate queries are considered significant if they do satisfy any aspect of the user intent.

Name	Selecting query result attributes (the query projection)
Identifier	UC3.2
Description	This use case is to define which attributes are expected by the user to answer her/his question. After selecting a candidate query, the user can be interested in refining the query by adding and/or removing attributes. For example, let us consider the following question: "which species are present?". The answer for this question can include one or more of the following species' attributes: a taxonomic identifier, a description, common and scientific names. Attributes can also be ranked to facilitate the choice mainly when several of them are available.
Goal	To allow the user to choose which attributes compose the answer for his/her question.
Scope	Bioinformatician, researcher and biologist profiles
Preconditions	The user has to select which attributes will be projected
Post conditions	The information requested is displayed
Actors / Users	Bioinformatician, researcher and biologist
Dependencies from other functionalities/steps	UC3.1
Exceptions	At least one attribute must be added. The attribute cannot be added for the question answer.
Notes/Comments	None

Name	Downloading results
Identifier	UC3.3

Description	It provides the possibility to download the results as a CSV file, JSON file or a spreadsheet (e.g. .xlsx) by also including infographics when it is applicable.
Goal	To allow the user to download the results.
Scope	Bioinformatician, researcher and biologist profiles
Preconditions	The user has to choose the format to export the results.
Post conditions	The file is downloaded
Actors / Users	Bioinformatician, researcher and biologist
Dependencies from other functionalities/steps	UC3.1, UC3.2, UC3.10, UC3.11 and UC3.6
Exceptions	No retrieved results to export.
Notes/Comments	None

Name	Displaying results
Identifier	UC3.4
Description	The results are mainly shown in a tabular format. New columns can be added according to UC3.2. When applicable, it can also display possible infographics related to the results.
Goal	To allow the user to visualize and read the results, i.e question answer.
Scope	Bioinformatician, researcher and biologist profiles.
Preconditions	The user has to submit a query before (see UC3.1).
Post conditions	The information requested is displayed.
Actors / Users	Bioinformatician, researcher and biologist.
Dependencies from other functionalities/steps	UC3.1, UC3.2, UC3.10, UC3.11 and UC3.6
Exceptions	No results to display.
Notes/Comments	None

Name	Visualizing large controlled vocabularies and ontologies.
Identifier	UC3.5
Description	In biology, controlled vocabularies and ontologies have been adopted to reduce semantic heterogeneities. Usually, these ontologies are not applied as a data schema to structure data. For example, we can mention the UBERON ontology for anatomical entities in animals. Because of the size and complexity (e.g. part of, is a, property chain axioms) of these ontologies, they are not easy to be visualized and understood by the user. This means the user cannot easily retrieve the essential information (s)he is looking for. For example, how to visualize and improve understandability that **"**red nucleus" anatomical structure is part of "the midbrain tegmentum" that is part of the "midbrain". And finally to know that the midbrain is a "regional part of the brain". The ontology terms will often be part of the query results. Hence it is important to the user to better understand the semantics of the retrieved ontological terms rather than solely showing term labels.
Goal	To show a visual summarization of ontological term meanings.
Scope	Bioinformatician, researcher and biologist.
Preconditions	ontology/taxonomy/controlled vocabulary
Post conditions	Visual summarization of one or more term semantics
Actors / Users	Bioinformatician, researcher and biologist.

Dependencies from other functionalities/steps	UC3.4
Exceptions	The file could not be loaded. Term does not exist.
Notes/Comments	None

Name	Building a query by exploring the datasets
Identifier	UC3.6
Description	The user writes a query in natural language as a result (s)he obtains a subgraph that represents in abstracted way how the data used to answer a question are structured and related (i.e. an explicit view of the data schema). Value examples are also displayed when highlighting or selecting a given attribute of a graph node (that can be interpreted as a class). Displaying attribute values are important for the user because (s)he can find out what kind of projected results to expect when executing the final built query. In this subgraph, not only an explicit portion of the data schema to answer the initial question is shown but also the closest neighbor nodes and their attributes. This is done to allow the user to explore the datasets and expand the initial query. Moreover, when expanding the initial query, the user has the possibility to specify attribute values, consequently, (s)he restricts the retrieved results similar to a WHERE clause in SQL.
Goal	To allow the user to interactively build a query based on an initial exploratory question.
Scope	Bioinformatician, researcher and biologist profiles.
Preconditions	The user has to write in English and submit a query.
Post conditions	A subgraph that represents how the data are structured to answer an exploratory query and their final results.
Actors / Users	Bioinformatician, researcher and biologist
Dependencies from other functionalities/steps	None
Exceptions	No data found to answer the query.
Notes/Comments	None

Name	Exporting built query.
Identifier	UC3.7
Description	It provides the functionality to save the built query into a SPARQL query file (.rq files).
Goal	To export the built query in a .rq file.
Scope	Bioinformatician, researcher and biologist profiles
Preconditions	Built a query based on UC3.6 or UC3.9.
Post conditions	A SPARQL query file (.rq file).
Actors / Users	Bioinformatician, researcher and biologist.
Dependencies from other functionalities/steps	UC3.6, and UC3.9
Exceptions	No data found to answer the query
Notes/Comments	None

Name	Importing query from .rq file.
Identifier	UC3.8
Description	It provides the functionality to import a query from a .rq file. The query can be then executed by the user. Edition is enabled within a SPARQL query editor for the bioinformatician user profile.
Goal	To import a query from an .rq file into the INODE system.
Scope	Bioinformatician, researcher and biologist profiles.
Preconditions	SPARQL query stored as a .rq file.
Post conditions	Possibility to execute or edit the loaded query.
Actors / Users	Bioinformatician, researcher and biologist
Dependencies from other functionalities/steps	UC3.3, UC3.4, UC3.9, and UC3.7
Exceptions	Syntax file is not compliant with the rq format. No data found to answer the query.
Notes/Comments	None

Name	SPARQL query editor.
Identifier	UC3.9
Description	It provides a SPARQL editor with an auto-complete feature.
Goal	SPARQL query edition.
Scope	Bioinformatician profile.
Preconditions	Connection to a data store that supports SPARQL.
Post conditions	Execution of the query along with results
Actors / Users	Bioinformatician
Dependencies from other functionalities/steps	None
Exceptions	No results found. SPARQL query syntax error. Connecting to the data store was not possible.
Notes/Comments	None

Name	Query template catalog and API interfaces.
Identifier	UC3.10
Description	Queries are stored along with a description. A query can be set up as an editable form where data values can be modified. For more advanced users, they can directly edit the templates based on SPARQL or simplified query languages. CRUD operations over the query catalog are also available. The queries can also be executed and parameterized through predefined interfaces such as RESTFul APIs and/or APIs in Python and R languages.
Goal	Managing a query template catalog.
Scope	Bioinformatician profile.
Preconditions	Connection to the data store.
Post conditions	Save query and editable template and/or run the query.
Actors / Users	Bioinformatician
Dependencies from other functionalities/steps	UC3.9
Exceptions	Query syntax error.
Notes/Comments	None

Name	Simplified query language.
Identifier	UC3.11
Description	Bioinformaticians are used to command line interfaces (CLI). Providing a simplified query language based on CLI commands can make it more user-friendly for them. An example of an existing SPARQL-CLI interface is http://spang.dbcls.jp .
Goal	CLI-based query language
Scope	Bioinformatician profile.
Preconditions	Connection to the data store.
Post conditions	Execution of the query along with results
Actors / Users	Bioinformatician
Dependencies from other functionalities/steps	None
Exceptions	No results found. Query syntax error. Connecting to the data store was not possible
Notes/Comments	None

3.4.5 Requirements summary

The requirements elicited by this specific use case are provided in the consolidated table (Table 5-1) under Category R1, Group 2 (specifically the requirements R1.2.2, R1.2.3, R1.2.5-R1.2.8, R1.2.10, R1.2.11, R1.2.13, R1.2.14, R1.2.17 - R1.2.26).

4 TECHNICAL AND SYSTEM LEVEL REQUIREMENTS

4.1 Integrated Query processing (WP3)

4.1.1 Technology Elements involved

All Tasks

UNIBZ plans to contribute to the development of all the query processing tasks through its own technology *Ontop* [2], a popular Virtual Knowledge Graph (VKG) system implemented in Java. A VKG is a virtual representation in the form of a graph of the information coming from multiple, possibly heterogeneous, data sources. Such representation relies on the RDF language as the format to represent the data as a graph, and on OWL 2 QL as the language to represent ontologies. Both are W3C standards for VKGs. The RDF graph is virtual in the sense that it is not a copy of the data sources that are physically stored somewhere, but rather each query (formulated in SPARQL, the W3C standard query language for RDF) over the VKG is translated by Ontop into a query over the original data sources [3]. This is done through reformulation techniques well studied in the literature [4] as well as a number of optimizations performed by Ontop itself [5], so as to reduce the overhead introduced by virtualization and translation to a minimum. The link between the VKG and the data sources is realized through a *domain-specific ontology*, which provides a vocabulary for SPARQL queries abstracting from storage details, and through a set of *mappings* relating elements in the ontology to queries over the data sources.

Ontop is not a prototype tool, but a large and well-established software artifact that relies on and interacts with several technologies[5]. Most notably, it supports all major open and commercial RDBMSs (e.g., PostgreSQL, Oracle, DB2, Microsoft SQL Server, etc.). Ontop can be used in several different ways: as a SPARQL endpoint to query through HTTP, as an API to enrich other Java applications, or simply as a shell tool.

Task 3.1 - Query execution over rich types of data sources

In this task the goal is dual. On the one hand, we want to provide support for querying data sources that go beyond RDBMSs, for instance NoSQL sources such as MongoDB. With respect to this, we are also considering the possibility of having RDF itself as a data source format, e.g., in the form of Triple Stores (see Task 3.2). On the other hand, we want to support complex datatypes, such as the GIS geospatial types, which are nowadays broadly supported by all major RDBMSs and also by GeoSPARQL (a flavor of SPARQL for GIS). With respect to the latter goal, we began working on supporting *all* SQL and SPARQL datatypes. This is non-trivial, as it requires defining suitable conversion tables between datatypes, which come into play during the query reformulation phase.

[5] https://ontop-vkg.org/

Task 3.2 - Source Federation

In this task, the goal is to render Ontop able to perform query processing tasks over multiple federated data sources, as opposed to a single relational data source. For the federation layer, we plan to use techniques that build upon existing federation middleware, in particular Denodo, Dremio, Teiid, and Spark.

Task 3.3 - Data Analytics

For this task, Ontop needs to be able to express complex analytics (e.g., statistics, clustering, or graph analysis) in terms of the ontological vocabulary, with special attention to the various data dimensions, and techniques and to include them in query reformulation. To achieve this, we are enriching Ontop with SPARQL aggregate functions (e.g., COUNT, MIN, MAX, etc.). Doing so requires a substantial rethinking of the whole reformulation process. For instance, one of the optimizations performed by Ontop is to always push the UNION operators to the root of the reformulated queries. In the presence of aggregates, this optimization cannot be applied blindly since this would affect the soundness of query results. Note that the challenge in this specific example is how to reconcile performance of query answering with support to aggregate functions.

Task 3.4 - Answer Justification

We want the users to be able to reconstruct why a certain answer was returned by Ontop. Such justification is not only in terms of where the data comes from (data provenance), but also in terms of which ontology axioms and mapping assertions were involved during the query reformulation phase (respectively, ontology provenance and mapping provenance). To support this task, we plan to rely on existing techniques based on provenance, and in particular we will rely on ProvSQL, which is at the moment the only working provenance tool developed in the context of RDBMSs. ProvSQL only currently supports PostgreSQL, and no other tool exists for other types of data sources that would be robust enough. We plan to deal with this by providing an answer justification based on mappings and ontology provenance, and only a basic form of data provenance.

4.1.2 Relation to INODE

Query processing is a core component of the INODE architecture, since it is a key service used by almost all other components. In particular, the Data Abstraction Layer in the INODE architecture is essentially provided by Ontop. Note that the tasks and technologies discussed in the previous section directly refer to services and artifacts both lying below (e.g., ontology, data sources) and above (e.g., query execution, answer justification) this Data Abstraction Layer, namely in the Data Access Layer and Business Logic Layer, respectively.

4.1.3 Requirements implied by the technology elements

The requirements that are introduced by these specific technology elements are presented in the consolidated table (Table 5-1) under Group 3 (as requirements coded R1.3.1 to R1.3.4).

4.2 Data Linking and Modelling (WP4)

4.2.1 Technology elements involved

Task 4.1 - Data-driven Mapping

UNIBZ plans to adopt Ontop, a technology discussed in Section 4.1.1, to bootstrap an initial ontology and a set of VKG mappings starting from a database. The generation of the ontology and mappings will be flexible, and will also allow one to only generate the mappings in the case where an ontology is already available to the user. Such mappings will provide the semantic link between the database information and the elements in the ontology. At the moment, the bootstrapping component of Ontop is not flexible, because it only generates the ontology and mappings in combination and does not allow for the reuse of an existing ontology. In order to provide more flexibility, we plan to rely on techniques coming from *Schema Matching*, a research field where the goal is to find correspondences across different relational database schemas. Such techniques need to be extended to the richer setting of mappings between an ontology and data sources, where in addition to having mapping queries that are typically more complex than the direct correspondences in Schema Matching, one needs to deal with the generation of suitable object identifiers at the ontology level from values extracted from the data sources.

Task 4.2 - Task-driven Mapping

UNIBZ plans to extend Ontop so that it can deal with requests (i.e., queries) that require *(i)* concepts not present in the ontology but available in the data sources, or *(ii)* concepts present in the ontology but not yet mapped to the data sources. Observe that to address both scenarios we rely on the functionalities provided by data-driven mappings.

Task 4.3 - Knowledge graph construction based on text

Infili Technologies and ZHAW are planning to contribute to the development of automated entity extraction and graph construction technologies, by collaboratively developing the components of a unified pipeline. The entity extraction components will be developed in a complementary manner, with the outputs of each partner's system being combined in the pipeline. The entity-relation-entity triples output from each system will be consolidated before a unified knowledge graph is constructed. ZHAW plans to build these components by developing deterministic NLP algorithms; Infili aims to do this through its product portfolio of statistical NLP techniques, and more specifically by fine-tuning its family of NLP analytics products (e.g., Noima). This leverages the advantages of both approaches: the higher-recall output of the Infili system can be refined using the higher-precision NLP algorithms developed by ZHAW.

The Infili suite leverages web crawling techniques to collect and process thousands of data sources, including unstructured text (web sites, blogs, forums, etc). It then uses information extraction techniques to isolate the valuable information from these sources, and to collect named entities and their in-between direct and latent semantic relationships. The platform is primarily used for landscaping entities (enterprises, individuals/products) ecosystems, by applying graph databases for entity linking and statistical NLP techniques for syntactic parsing and Part-of-Speech tagging, entity extraction, sentiment analysis etc. The objective at Infili is

to enrich the suite's toolkit throughout the project, by integrating state-of-the-art deep-learning models for language modelling, text comprehension, and summarization.

ZHAW will be using Python's Natural Language ToolKit (NLTK), which contains both Stanford Parser and WordNet interfaces, along with built-in data structures for linguistic analysis (such as dependency graphs and phrase-structure parse trees). The Stanford Parser component is a wrapper for a Java-based parser, which will be run as a server, in parallel to the entity extraction system. New unstructured text documents will be sent to a locally-running server for pre-processing before the entity extraction procedure. The main body of the system will use customised graph and tree algorithms written using common Python libraries, such as NumPy.

The graph construction will be based on the open-source version of the Neo4j graph database management system, an ACID-compliant transactional database with native graph storage and processing, which is optimal for deep or variable length traversals and path queries. Neo4j is implemented in Java but is also accessible from software written in other languages (e.g., Python) using relevant libraries. It uses the Cypher language for query expression, through a transactional HTTP endpoint, or through the binary "bolt" protocol. However, its recent versions also allow the user to create Cypher queries in pure Python using Pypher, a suite of lightweight Python objects.

Regarding NLP-related tasks, the platform already exploits the functionalities of the most popular PoS and NER libraries, namely Stanford Core NLP, Spacy, and NLTK. ZHAW also makes use of these tools in its own systems, ensuring that all components will be built on a reliable foundation of industry-standard tools. NLTK's WordNet interface also provides a hierarchical dictionary/thesaurus reader, which can be traversed for the hypernymy and synonymy information required for restructuring the knowledge graph. Driven by research interests, Infili will also explore Pytorch implementations of state-of-the-art general-purpose architectures (BERT, ELMO, XLNet) for Natural Language Understanding (NLU), leveraging the libraries of pre-trained NLP models as well as development frameworks such as HuggingFace and AllenNLP.

4.2.2 Relation to INODE

The Data-driven and Task-driven components aim to provide support to domain experts in the task of ontology construction. In order to process queries, the Ontop system requires three elements as input: (1) the data sources, (2) the ontology, which provides the target vocabulary (as well as an encoding of the domain knowledge), and (3) the VKG mappings linking the data sources to the ontology. The ontology and the mappings are usually specified manually by the domain experts, in interaction with the data source providers, who are aware of the specific details of how data is organized and structured at the sources. Due to this interaction, writing the ontology and the mappings is usually a tedious and time-consuming activity. Through the Data-driven and Task-driven approaches we provide support to the domain experts in this task, by automating the creation of a first basic version of the ontology and mappings. The domain experts can use these as a starting point, and adapt them according to their needs. The challenge for INODE is to automatically generate an ontology and mappings that are ready-to-go, thus requiring minimal to no intervention by the domain experts.

The development of the Entity Extraction and Knowledge Graph components will facilitate a system that extracts useful information from unstructured text, and transforms it into a format suitable for complementing an existing structured database. The knowledge graph

component of this system will allow the retrieval and query-processing components of INODE to leverage additional data that is complementary to existing structured data. This will be done with the intent of building a system that runs efficiently at scale, but is robust enough to handle a wide range of unstructured text domains, and thus all the use cases of INODE. Further features that could be added to the knowledge graph include a synonymy component that could assist with processing natural language queries in other parts of the system.

Although Infili's NLP analytics platform was originally intended for Greek Web sources, its NER-related capabilities could be easily re-purposed to allow for entity and relation extraction from English unstructured texts without major modifications. One anticipated challenge is NER performance (e.g., in terms of accuracy, F1-score etc.) of the existing pre-trained models on domain-specific corpora (e.g., biomedical), where lower performance is expected. To this end, the rules-based approach of ZHAW will be leveraged to increase the precision of the entities and relations, leading to a higher-precision output. Such rules will likely need to be extended, in order to cover a wide range of text domains and communication styles – however the design of the system will take this into account, and synthesizing additional rules should be a simple process.

There is an issue of co-reference resolution (i.e., finding all expressions that refer to the same entity in a text), which will likely be encountered in both the R&I Policy Making and Cancer Biomarker Research datasets. However, this can be solved by using neural models that check all spans in the document as potential mentions and learn distributions over possible expressions to which each span refers to.

4.2.3 Requirements implied by the technology elements

The requirements that are introduced by these specific technology elements are presented in the consolidated table (Table 5-1) under Category 1 Group 4 (as functional requirements coded as R1.4.1 to R1.4.8) and under Category 2 Group 2 (as non-functional requirements coded as R2.2.8, R2.2.9, and R2.2.10).

4.3 Data Access and Exploration (WP5)

4.3.1 Technology elements involved

Task 5.1 - by example and Task 5.2 - by analytics

The design of the different by-example exploration operators such as by-filter, by-join or by-sim that we have proposed is based on two different approaches, each of which involves a different technology.

- *All-Python*: data is presented in the form of CSV files, and loaded into memory in data frames (using Pandas).
- *Embedded*: This approach is based on a SQL/SPARQL backend, where the data is stored in a relational database (we use the PostgreSQL DBMS), and combined with an in-memory logic in Python.

The implementation of the two approaches will allow us to compare their performances.

Task 5.3 - by natural language

The goal of this task is to translate natural language (NL) questions to SQL or SPARQL. We will perform a **"dual-strategy-approach":**

- *Non-neural network-based approach*: We will extend our existing research prototypes SODA and Bio-SODA [6] (a non-machine learning approach) and adapt it to the use cases of INODE. ("low risk approach")
- *Neural network-based approach*: We will experiment with state-of-the-art machine learning algorithms that require large amounts of training data to translate from NL to SQL. This can be considered a "high risk approach".

By running this dual-strategy-approach, we make sure to have a running system on INODE-data in the short term and also have the freedom to experiment with new algorithms which can be used in the medium to longer term of the project.

The first approach is implemented in Java to leverage enterprise-scale software development technology. The second approach is implemented in Python to leverage state-of-the-art machine learning packages.

4.3.2 Relation to INODE

Task 5.1 - by example and Task 5.2 - by analytics

We investigated two implementations of exploration operators: an all-Python in-memory implementation, and an embedded-SQL implementation. A first version of the all-Python implementation for all operators (by-filter, by-facet, by-subset, by-superset, by-overlap, by-join, and by-sim) is running and we are in the process of finalizing the embedded-SQL implementation.

Each operator (by-filter, by-facet, by-subset, by-superset, by-overlap, by-join, and by-sim) takes as input an example itemset, and a set of attributes and returns one or several itemsets, each of them is "related to" the example itemset. These operators allow users to:

- Explore in more detail and in depth the example itemset (by-facet, by-filter, by-subset).
- Extend the schema of the example itemset (by-join).
- Expand the size of the example itemset (by-superset).
- Explore related itemsets outside of the example itemset (by-overlap).
- Explore similar itemsets outside of the example itemset (by-sim).

Task 5.3 - by natural language

To enable a hybrid approach of translating from NL to SQL/SPARQL, the two approaches (neural network-based and non-neural network based) need to be integrated via a **REST-API**. In particular, the Java-code needs to be efficiently called via Python and vice versa. Moreover, the REST-API needs to be callable by other services described in Tasks 5.1 and 5.2 as well as by the User Assistance Services and the Multi-Modal Discovery Services.

4.3.3 Requirements implied by the technology elements

The requirements that are introduced by these specific technology elements are presented in the consolidated table (Table 5-1) under Category 1 Group 5 (as functional requirements coded as R1.5.1 to R1.5.4), Group 6 (as functional requirements coded as R1.6.1 to R1.6.8) and under Category 2 Group 2 (as non-functional requirements coded as R2.2.6 and R2.2.7).

4.4 User Assistance services (WP6)

4.4.1 Technology elements involved

Task 6.1 - Exploration insights

ATHENA and CNRS are planning to contribute to the development of methods that provide exploration hints, i.e., recommendations, to the user.

ATHENA's *data-based query recommendation* approach starts from a user query and generates query recommendations based on statistical properties of the dataset and the query results. The main idea is to cluster the query results and find a set of new queries that describe these clusters. The user can refine their exploration by submitting one of the queries for further analysis. Result clustering takes the similarity of the results as well as the correlation of the attributes that describe them into consideration. A decision tree is used to produce splits of the dataset. The decision tree boundaries are used to produce the recommended queries.

Furthermore, ATHENA's prior work on *semantic recommendations* [7] uses a knowledge graph to recommend interesting tweets for users. Based on the user activity, the approach identifies semantically similar tweets from other users, and users with similar interest profiles as well. The foundation of this method is a knowledge graph that represents all user topics of interest as a variety of concepts, objects, events, persons, entities, locations and the relations between them. The method uses the knowledge graph and graph theory algorithms in order to construct user interest profiles by retrieving semantic information from tweets based on which recommendations are performed. The main principles of this process can be applied in INODE to implement a semantic approach to query recommendation.

CNRS's prior work on guided exploration of user groups [8] uses Reinforcement Learning with the purpose to find target users. Exploration is modelled as an iterative decision-making process, where an agent is shown a set of user groups, chooses users from those groups, and selects the best action to move to the next step. Reinforcement learning is applied to discover an efficient exploration strategy from a simulated agent experience, and propose the learned strategy to recommend an exploration policy that can be applied to the same task for any dataset. This framework accepts a wide class of exploration actions and does not need to gather exploration logs.

Task 6.2 – Explanations

ATHENA's work on natural language explanations for SQL queries comprises Logos [9], a system that generates natural language explanations for SQL queries. To do so, it considers that the database schema graph can be represented as a graph where nodes represent the database relations and attributes, and edges represent the relationships between the nodes (e.g., attribute belongs to relation, foreign key-primary key relationships connecting relations). Edges and nodes are annotated with labels that explain nodes and their relationships intuitively in natural language. These can include the names of the nodes (if they are descriptive enough) and annotations of the nodes and edges produced by a database designer or the user. An SQL query is also represented as a graph that captures the semantics of the query, and extends the database schema graph to include value and function nodes. Given an SQL query, Logos can synthesize different types of NL explanations using the graph, the labels, additional templates that enrich the explainability mechanism of the system and a template mechanism that dictates how to synthesize NL phrases.

4.4.2 Relation to INODE

Task 6.1 - Exploration insights

Exploration insights or recommendations help the users navigate data and complete their search and exploration tasks more easily.

Especially at the early stages of a dataset's deployment through INODE, user logs may not be available. In general, a cold-start situation is a situation where we do not have logs that provide information about how different users access the data, e.g., queries they perform, results they choose, etc. and/or when there is no such information for a particular user. To be able to provide meaningful recommendations in such situations, INODE will employ a set of approaches that aim at a holistic solution. All methods will be tested with the three use cases.

ATHENA's data-based query recommendation approach fits well into contexts where user logs are not available. In INODE, this approach will be enhanced to work on big and diverse datasets. For example, the astrophysics dataset is large and has mainly numerical attributes; CORDIS, on the other hand, has more textual content. The data-based query recommendation approach needs to scale to large query result sets described by several attributes.

CNRS is developing an adaptation of Reinforcement Learning (RL), an unsupervised method that does not necessitate the use of logs. This adaptation is based on simulating a human agent that interacts with the environment to learn an exploration policy. The method uses a Markov Decision Process to model states and actions. The actions are the exploration operations we have defined. An action is applied to a state and the RL method learns state feature weights during the simulation.

Infili and ATHENA will extend the knowledge-based approach to generate recommendations by exploiting the relationships and entities of knowledge graphs, starting with CORDIS. The major strength of this family of methods is that there are no cold-start problems (i.e., when a user appears on the platform for the first time or needs to make a new search). On the contrary, these case-based approaches rely on specific domain knowledge regarding how certain features meet users' needs and are usually leveraged for query expansion tasks, which

exploit the knowledge source to expand the original query with additional terms that are specifically relevant to the query's scope.

CNRS and ATHENA are also developing an approach based on soliciting user feedback on data and training a classifier to learn the parameters of the next exploration action. A user's exploration actions form a pipeline or sequence. The goal is to find the parameters that maximize the coverage of unlabeled data. Additionally, this method will be extended to solicit feedback on exploration operations and queries.

As INODE starts collecting user queries in logs, we will extend our methods to leverage information from the logs and generate recommendations, not only at the level of individual actions or queries, but also at the level of sets or sequences of actions, i.e., pipelines. For example, given the user's current set of queries, the system may give pipeline recommendations and pipeline completions based on user and query similarities.

These methods will also look into building different user roles (such as novice, expert, data scientist). For example, we expect the novice user to need a simple pipeline, while the domain expert needs a pipeline that focuses on attribute correlations, and the data scientist needs a pipeline that explores relationships between tables (joins etc).

To this end, collaborative filtering approaches could be leveraged for query suggestion, aiming at the exploitation of previous users' experience, in order to drive other users with similar information needs in the right direction. More specifically, the formulation of this problem implies that users with similar profiles (e.g., similar past searches, expertise) can benefit from "successful" queries that allowed other users to find the information they were looking for. Two main types of collaborative filtering algorithms can be used for log-based query suggestion: *(i)* memory-based, which use the whole data to suggest queries based on similarity measures and *(ii)* model-based, which construct a model in advance to depict the different needs of each user profile. The former are usually simple to deploy but have limited scalability and the latter are usually more efficient in terms of prediction speed, but can be difficult to train and fine-tune as the application domain data evolves.

Task 6.2 – Explanations

Generating natural language (NL) explanations of queries for the users is an important part in INODE that aims at bridging the gap between the user queries, on the one hand, and how the system interprets these user queries on the underlying data, on the other hand. ATHENA has adapted Logos to generate NL explanations over the CORDIS data with satisfying results. We are creating a large set of initial queries to test the system. One challenge with generating NL explanations arises when dealing with databases that have obscure or very complex schemas. The astrophysics data are a good example. For these, we are planning on extending our approach on learning templates from existing NL queries. Generating NL explanations for parts of a query (that are the more ambiguous) is also important. As we work on extending to other datasets and queries, we will also extend to handling SPARQL queries formulated over an ontology. These resemble, in philosophy, SQL queries over a database schema, so we expect to map them to query graphs and have Logos work on them.

4.4.3 Requirements implied by the technology elements

The requirements that are introduced by these specific technology elements are presented in the consolidated table (Table 5-1) under Category 1 Group 6 (as functional requirements coded as R1.6.1 to R1.6.19), and under Category 2 Group 2 (as non-functional requirements coded as R2.2.11 and R2.2.12).

4.5 Multi-modal discovery services (WP7)

4.5.1 Technology elements involved

Fraunhofer IGD develops the "Visual Computing as a Service" (VCaaS) platform, which we intend to use and extend in the development of the multi-modal discovery services. The VCaaS platform provides a good starting point for WP7 developments in INODE, as it provides the means to create, run and monitor complex data-driven workflows, for example in the cloud. A workflow is a repeatable process, which, in most cases, consists of various activities carried out in a specified order to transform input information into output results. As different multi-modal data types have to be treated differently, the VCaaS platform provides the flexibility we need to pursue the discovery therein. In fact, VCaaS does not only provide one single runtime, but multiple, each featuring different advantages. The “Executor” engine is a RESTful service capable of updating workflows while they are loaded. The benefit of this approach is that it enables a more rapid workflow design, as time and energy are saved: Datasets load less often, algorithms run fewer times, and larger datasets. Another workflow engine supported by VCaaS is the “steep” engine, which is advantageous when dealing with big data (for example, earth observation satellite imagery), as it allows recursive actions and optimizes parallelization with multiple agents in cloud-native scenarios.

To ease user interaction with the VCaaS platform, the Whiteboard has been created. It provides a web-based visual programming interface to create such workflows. As complex workflows are likely to include dependencies among activities (for example, when combining two results into a single one), and workflow designers need to know and understand these dependencies, workflows are laid out on a 2D canvas, where these dependencies become visually apparent. Besides the actual design of workflows, the Whiteboard does not only provide features to execute and monitor these workflows, but also to interactively visualize the results, to collaboratively edit and annotate workflows. Such features might prove beneficial in the later stages of the INODE project.

4.5.2 Relation to INODE

In general, the Whiteboard and VCaaS already support exploratory processes, as explorations are sequences of partially interdependent actions, which are carried out on data, and therefore captured in a workflow. However, VCaaS currently lacks semantic web support, i.e., RDF and SPARQL support have to be added.

Furthermore, preliminary work suggests that the type of user interface currently employed (a free 2D canvas with boxes and arrows connecting them) is too complex for the prospective target user groups identified by the use case partners. Thus, adaptations and simplifications

are foreseen, which tailor the current user interface for laypersons, resulting in a more query-response style (like classic web search engines).

Task 7.1 - Visual guidance and exploration of search results

One goal of this task is to enable users to explore search results. In doing so, the user is able to validate whether a query yields the expected results. The comparison of actual data to the expected outcomes poses two challenges. First, results from linked big data sources like federated triple stores may return large result sets which contain data of various modalities, posing an inherent challenge to even understand the query result. To support the user in understanding the results, we will provide tools to visualize the query result, inspect parts of it, as well as visualizations of the underlying data schema, as these help the user to gain an understanding of it. The second challenge stems from the comparison of the result to expected outcomes. The expected outcomes are formally specified only in a few cases; in most cases, the expected outcome exists in the users' head only, either as visceral or conscious thoughts. To extract that knowledge, we intend to provide tools for memorizing, arranging, annotating and structuring query results, which in turn will ease the comparison of the actual result to the expected. Finally, users draw conclusions from the outcomes, which inform the decision on how to continue and thus can be used to feed into the next iteration.

The other goal of this task is the provision of guidance to assist users in their exploration process of query results. A process is guided, if advice is given or information is presented that helps overcoming a problem or difficulty. In the context of visual analytics, Ceneda et al provide a more formal definition, which we adopt: "Guidance is a computer-assisted process that aims to actively resolve a knowledge gap encountered by users during an interactive visual analytics session". Guidance is particularly relevant in situations where the user is unable to identify, judge or execute an action. The assistance of how to execute an action is discussed in Section 4.3 and the assistance in identification of actions to take is discussed in Section 4.4. Assistance support to judge the results of a query action thus is the main priority of this task. Ceneda et al. describe the degrees of guidance which ranges from "orienting" over "directing" to "prescribing". While Section 4.3 describes prescribing guidance in the sense that the user is guided through a fixed pipeline of steps, this task focuses on orienting guidance within the visual result presentation. That is, in the visual representation of a query result, we want to help the user to explore and build a mental map of it. Providing tools to find the relationships between data subsets, patterns, or attributes in the data is the first step, suggesting the right tools at the right time is the second one, which heavily depends on the outcomes of the questionnaires filled out by the end users.

Task 7.2 - Interactive manipulation and optimization of queries

The goal of this task is to enable the user to steer the exploration process without formulating queries at all. This will be realized via a direct manipulation mechanism in the visual result presentations. As this task starts in M13, we expect to gather more requirements in the first months of T7.2.

Task 7.3 - Integrated seamless query-response loop

The goal of this task is to reduce the overhead implied by switching between different exploration and query manipulation modes. Another goal of this task is to provide additional

components, which target specific user groups. As T7.3 starts in M13, we expect to gather more requirements in the first few months of T7.3.

4.5.3 Requirements implied by the technology elements

The requirements that are introduced by these specific technology elements are presented in the consolidated table (Table 5-1) under Category 1 Group 7 (as functional requirements coded as R1.7.1 to R1.7.19).

5 REQUIREMENTS CONSOLIDATION AND CATEGORIZATION

This section provides a consolidated view of the aforementioned described requirements so as to arrive at a unique coding of all individual requirements that can be tracked throughout the lifecycle of the project. The following table gives an overview of the elicited requirements of the INODE project. We emphasize that the requirements are derived not only from the use cases, but also from a range of other factors including the partners' background projects, state-of-the-art projects, and requirements expressed by the project stakeholders. The selection of requirements has been also driven by:

- The need to address, design and implement the innovative features of the INODE framework as listed in the Description of Action giving a particular emphasis on the various key performance indicators (KPIs) that have been defined for the success of the project.
- Specific tangible improvements that must be realized on essential INODE components for being able to deliver the INODE service suite offering.
- The functionalities to be delivered in order to support the project's three use cases.

The overall requirements analysis coding and consolidation has been based on the well-established process of dividing the requirements into functional and non-functional ones.

Functional requirements:

The **functional** view of a requirements analysis process focuses on what the system must do to produce the required operational behavior. It includes inputs, outputs, states, and transformation rules. Functional view information includes:

- system functions
- tasks or actions to be performed,
- inter-function relationships,
- functional constraints,
- interface requirements.

Non-functional requirements:

The **non-functional** view of a requirements analysis process focuses on what other technical features the system must have in order to facilitate the service provision. These include among others:

- scalability,
- performance,
- reliability and availability,
- manageability and flexibility,
- security,
- openness and extensibility requirements.

Overall requirements consolidation and coding

The approach that has been followed for the coding, provides a distinction between Functional (denoted as R1 Category) and Non-functional Requirements (denoted as R2

Category), while the requirements have been grouped accordingly into various Groups, as designated in the following table (Table 5-1).

The first column of the table below provides the unique coding for each requirement that will make it traceable within the lifecycle of the project. The second column gives the description of each requirement, while the third column describes any dependencies or comments related to each requirement. The fourth column provides the details on the origin of each requirement, thus linking it with the requirements summary of the use cases or the subsection related with the "Requirements implied by the technology elements" that have been provided by the various INODE partners. Moreover, requirements that have been derived from other factors such as the KPIs that are globally identified in the description of the project have been appropriately indicated in the specific column. The fifth column denotes the relevant stakeholders for each requirement. A rationale behind the content of this specific column is provided in Table 5-2.

Table 5-1 *Consolidated and coded requirements.*

Category R1: Functional Requirements				
Group 1: INODE Generic Functional Requirements				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R1.1.1	The platform should provide registration and sign-in functionalities for the following roles: users, administrators.		All use cases	Service providers & Integrators
R1.1.2	The platform should provide analytics functionalities.	R1.3.3	All use cases	Service providers & Integrators
R1.1.3	The platform should provide a dashboard in order to present results of analysis.	R2.2.2	All use cases	Service providers & Integrators
R1.1.4	The platform should provide real-time data processing functionalities.	R2.2.1	All use cases	Service providers & Integrators
R1.1.5	The platform should be able to access online data, e.g. from web sites and open data repositories.	R1.3.1	All use cases	Service providers & Integrators
R1.1.6	The platform should be able to collect and store data from new data sources.	R1.2.4, R1.3.2	All use cases	Service providers & Integrators
R1.1.7	The platform should be able to connect to different backends for querying data.		All use cases	Service providers & Integrators
R1.1.8	The platform should provide user interaction functionalities.	R1.2.2, R1.2.6	All use cases	Service providers & Integrators
R1.1.9	The platform should provide composable exploration operators.	R1.6.4, R1.7.1	All use cases	Service providers & Integrators
Group 2: INODE Use Case Specific Requirements				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**

R1.2.1	INODE should accept simplified natural language questions, efficiently translate such questions into a valid database-specific query language (SQL in the case of SDSS database) and retreive relevant properties of astronomical objects in the form of numerical tables.		UC3-Astrophysics	Scientists with proprietary access
R1.2.2	INODE should allow a relatively untrained user (e.g. investigator) to perform meaningful queries quickly. Untrained may mean to be untrained in the domain (e.g. astronomy), untrained in the data schemas, database models and/or untrained in structured query languages (e.g. SQL, SPARQL).	R1.1.8	UC3-Astrophysics UC1.1-Cancer Research UC2-R&I Policy Making	Scientists with proprietary access General public Bioinformatician Researcher/Biologist Policy maker
R1.2.3	INODE should provide disambiguation mechanisms to ask back where questions were not clear enough or where specific answers require more precise questioning.	R1.2.5	UC3-Astrophysics UC1.1/UC1.2-Cancer Research UC2-R&I Policy Making	Scientists with proprietary access Bioinformatician Researcher/Biologist Policy maker
R1.2.4	INODE should allow to bring data from multiple datasets together seamlessly such that investigators do not have to learn schemas of new databases painstakingly before being able to ask meaningful questions and to correctly interpret the results.	R1.1.2, R1.1.5, R1.1.6	UC3-Astrophysics UC2-R&I Policy Making	Data Curator, Scientists with proprietary access Policy maker
R1.2.5	INODE should allow queries to be developed in a conversational sense: “Give me all objects like ...” -> “Now show me the following properties...” -> “Downselect according to a specific parameter ....” -> “Give me all the spectra of those...” - > “Show me the derived star formation rates ....”	R1.2.2	UC3-Astrophysics UC1.1-Cancer Research	Scientists with proprietary access General public Bioinformatician Researcher/Biologist
R1.2.6	INODE should provide the possibility to help any non-technical clients (end users) autonomously querying by natural language (NL) and by exploration. In the context of cancer research, this means to be able to interactively build a query by exploring the datasets and based on an initial NL query.		UC2-R&I Policy Making UC1.6-Cancer Research	End user/Policy maker Bioinformatician Researcher/Biologist

R1.2.7	Given the extremely low technical skills of a typical user, it is likely that exploration services will be yet too complex in most of the cases, therefore INODE should prioritise (in terms of development) the development of NL querying over exploration.		UC2-R&I Policy Making UC1.6-Cancer Research	End user/Policy maker Bioinformatician Researcher/Biologist
R1.2.8	INODE should provide functionalities in terms of supporting the generation of mappings between the database, the ontology, and in the formulation of the ontology itself.	R1.3.4	UC2-R&I Policy Making UC1-Cancer Research	Data manager (UC1) Data administrator (UC2)
R1.2.9	INODE should provide analytical queries in SQL terms, that would be queries with aggregation functions such as count, sum, etc. with some filters applied to it.		UC2-R&I Policy Making	Non-expert user/Policy maker
R1.2.10	INODE should deliver the replies in a tabular format that can be exported into several file formats.	R1.2.17	UC2-R&I Policy Making UC1.3/UC1.4-Cancer Research	Non-expert user/Policy maker Statistician (UC2) Bioinformatician Researcher/Biologist
R1.2.11	INODE should provide an API-like interface, with predefined queries that users can just parameterize to get a dump of the data that is relevant to their study.	R1.2.19	UC2-R&I Policy Making UC1.10-Cancer Research	Statistician Bioinformatician
R1.2.12	The platform should enable access to a public portal with visualizations, possibly embedded in a narrative that puts them in context, following a story-telling approach.		UC2-R&I Policy Making	Citizen
R1.2.13	The system should show provide query results in tabular format.		UC1.4-Cancer Research UC2	Bioinformatician Researcher/Biologist
R1.2.14	The system should visualize the query result as infographics (e.g. diagrams)		UC1.4-Cancer Research UC2	Bioinformatician Researcher/Biologist

R1.2.15	The system should suggest data visualisations (e.g. diagrams) based on the contents of the query result		UC2-R&I Policy Making	Researcher/Biologist Bioinformatician Policy maker
R1.2.16	The platform should save the exploration sequence itself, so that it is repeatable if the dataset has been updated		UC2-R&I Policy Making	Policy maker
R1.2.17	The platform should provide a download option for query results in several formats (e.g. XLSX, CSV, JSON, XML) by also providing the schema (e.g. xsd) when it is applicable. A XLSX spreadsheet file may include data visualizations.		UC1.3-Cancer Research UC2-R&I Policy Making	Non-expert user/Policy maker Statistician (UC2) Bioinformatician Researcher/Biologist
R1.2.18	The platform should display on demand how portions of the underlying data are structured at a conceptual level.	R1.2.23, R1.2.7	UC1.4/UC1.5-Cancer Research	Researcher/Biologist Bioinformaticien
R1.2.19	The platform should allow for query processing (e.g. creating, editing and deleting a query) in the query template catalogue.	R1.2.21	UC1.10-Cancer Research	Bioinformaticien
R1.2.20	The platform should enable the user to select and execute a NL query from a template catalogue	R1.2.19	UC1.10-Cancer Research UC2	Bioinformaticien
R1.2.21	The platform should provide a SPARQL editor with auto-completion features.		UC1.9-Cancer Research	Bioinformaticien
R1.2.22	The platform should provide simplified query language functionalities as a command line interface (CLI)		UC1.11-Cancer Research	Bioinformaticien
R1.2.23	The platform should allow the visualisation of large controlled vocabularies and ontologies.		UC1.5-Cancer Research	Researcher/Biologist Bioinformaticien
R1.2.24	The platform should enable the user to select specific query result attributes (query projection).	R1.2.5, R1.2.2, R1.2.3	UC1.2-Cancer Research	Researcher/Biologist Bioinformaticien
R1.2.25	The platform should provide exporting functionalities of the built queries.	R1.2.20, R1.2.21, R1.2.22	UC1.7-Cancer Research	Researcher/Biologist Bioinformaticien

R1.2.26	The platform should allow the user to import queries from .rq files.	R1.2.21	UC1.8-Cancer Research	Researcher/Biologist Bioinformaticien
R1.2.27	The platform should recommend natural language queries related to the user's previous search so as to assist in further exploring the data.	R1.2.6	UC2-R&I Policy Making	Policy maker
Group 3: Integrated Query Processing Services Requirements				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R1.3.1	The platform should be able to deal with rich datatypes in datasources, as well as heterogeneous datasources.	R1.1.5	Task 3.1	All users
R1.3.2	The platform should be able to perform query processing tasks over multiple federated data sources.	R1.1.6	Task 3.2	All users
R1.3.3	The platform should allow the user to perform complex analytics tasks.	R1.1.2	Task 3.3	All users
R1.3.4	The platform should be able to provide a justification to the answers produced. Such justification should contain information relative to the ontology axioms, to the mappings assertions, to the database tables, and also to individual tuples used to build the answer. The information should be presented in a compact and understandable way.		Task 3.4	All users
Group 4: Data Linking and Modeling Services				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R1.4.1	INODE should provide functionalities in terms of supporting the generation of mappings between the database, the ontology, and in the formulation of the ontology itself.		Task 4.1, Task 4.2	Developers, Domain Experts, Scientists

R1.4.2	The platform should facilitate novel user needs that are not met by the available data by enriching the existing mapping with the integration of available (but not yet integrated) data sources (task-driven mapping).	R1.1.6	Task 4.1	Domain Experts and Scientists
R1.4.3	The platform should allow the user to view the mutual information from new data sources by correlating them to existing ontologies (data-driven mapping).		Task 4.2	Domain Experts and Scientists
R1.4.4	The platform should leverage ML/DL methods for automated entity and relation extraction from unstructured data (text), leading to automated knowledge graph creation (KG creation based on text).	R1.1.2	Task 4.3	Domain Experts
R1.4.5	The platform should expand the developed ML/DL methods for automated entity and relation extraction to facilitate task-driven mapping (KG creation based on structured data).	R1.1.2	Task 4.4	Domain Experts
R1.4.6	The platform should provide fine-tuned methods for reliable entity extraction that are suitable for domain-specific corpora.		Task 4.3	All users
R1.4.7	The platform should be able to adapt the output knowledge graph to fit the requirements of other components within the system (such as altering its structure or contents to fit the current information need).		Task 4.3	Domain Experts, End Users, Developers
R1.4.8	The platform should improve the quality of the extracted ontologies, dealing with underlying binding syntactic phenomena such as anaphora resolution.		Task 4.3	Domain Experts and End Users
Group 5: Data Access & Exploration Requirements				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R1.5.1	Several interpretations of by-example exploration should be made available and composable.	R1.1.9	Task 5.1	All users

R1.5.2	Several interpretations of by-analytics exploration should be made available and composable.	R1.1.9	Task 5.2	All users
R1.5.3	Intermediate steps of query processing should be shown to better understand how results were derived		Task 5.3	Developers and end users
R1.5.4	NL queries and SQL/SPARQL should be logged for collecting training data for ML algorithms		Task 5.3	Developers and end users
Group 6: User Assistance Requirements				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R1.6.1	The platform should provide assistance to all types of users when exploring data.	R1.2.6	Task 5.1, Task 5.2, Task 6.1, Task 6.2	All users
R1.6.2	The platform should provide guidance in building and executing exploration pipelines.		Task 5.1, Task 5.2, Task 6.1	All users
R1.6.3	The platform should help users choose operators at each step of the exploration pipeline.		Task 5.1, Task 5.2, Task 6.1	All users
R1.6.4	The platform should help users select attributes for exploration operators.		Task 5.1, Task 5.2, Task 6.1	All users
R1.6.5	The platform should generate natural language explanations for queries that are ambiguous	R1.2.3	Task 6.2, Task 5.3 (for queries that are issued by users)	All users
R1.6.6	The platform should generate natural language explanations for parts of complex queries that are ambiguous	R1.2.3	Task 6.2, Task 5.3 (for queries that are issued by users)	All users

R1.6.7	The platform should generate natural language explanations for different datasets	R1.2.3	Task 5.3 (for queries that are issued by users) Task 6.1 (for query recommendations)	All users
R1.6.8	The platform should generate natural language explanations for results	R1.2.3	Task 5.1, Task 5.2, Task 5.3 (results of queries that are issued by users as described in these tasks)	All users
R1.6.9	The platform should allow a domain expert to provide annotations for parts of a dataset so that the platform can generate more meaningful NL explanations		Task 6.2	domain expert
R1.6.10	The platform should allow a user to provide feedback and corrections for NL explanations generated by the system		Task 6.2	All users
R1.6.11	The platform should leverage user-provided annotations to generate more meaningful NL explanations		Task 6.2	All users
R1.6.12	The platform should provide query recommendations so as to assist in further exploring the data	R1.2.2, R1.2.27	Task 6.1	All users
R1.6.13	The platform should provide query recommendations in cold-start situations (no user information)	R1.6.12	Task 6.1	All users
R1.6.14	The platform should provide query recommendations in cold-start situations (no user logs)	R1.6.12	Task 6.1	All users
R1.6.15	The platform should provide query recommendations for different datasets	R1.6.12	Task 6.1	All users
R1.6.16	The platform should allow a user to provide feedback on query recommendations generated by the system		Task 6.1	All users

R1.6.17	The platform should leverage user-provided feedback to generate query recommendations		Task 6.1	All users
R1.6.18	The platform should generate natural language explanations for queries to enable a bi-directional conversational experience between the user and the system.	R1.2.5	All use cases.	Service providers & Integrators
R1.6.19	The platform should log queries and user feedback for improving user experience		All use cases.	Service providers & Integrators
Group 7: Multi-Modal Discovery Services Requirements				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R1.7.1	The user should be able to steer the exploration process without formulating queries at all.	R1.2.7	Task 7.2	All users
R1.7.2	The user should be able to manipulate the visual result presentations to change queries	R1.2.12, R1.2.14	Task 7.2	All users
R1.7.3	The user should be able to remove irrelevant search results from the screen		Task 7.2	All users
R1.7.4	The user should be able to promote relevant search results		Task 7.2	All users
R1.7.5	The user should be able to provide relevance feedback on search results	R1.1.8	Task 7.2	All users
R1.7.6	User session context should be incorporated into the query generation for more focused search		Task 7.2	All users
R1.7.7	Retrieval of ontologies via OWL format from the data base must be available		Task 7.1	All users
R1.7.8	Retrieval of data in RDF, CSV, JSON-LD from the data base must be available		Task 7.1	All users
R1.7.9	Access to data base schema or data base ontology should be possible		Task 7.1	All users

R1.7.10	Access to additional documentation of a data base / data schema / data ontology should be possible		Task 7.1	All users
R1.7.11	Result set meta-information should be available, including basic meta-information like (estimated) number of results, and total number of attributes,	R1.2.13	Task 7.1	All users
R1.7.12	Result set ontology pointers should be available, so that column/cell-related meta-information can be accessed.	R1.2.13	Task 7.1	All users
R1.7.13	Data should be retrievable in portions (in a “Paging” fashion) form the data bases	R1.2.13	Task 7.1	All users
R1.7.14	The scale of measurement must be available for each column in the result set. In other words, the query result has to provide hints to the visualization system so that it can infer whether a data variable is of nominal, ordinal, interval, and ratio scale as defined by Stevens.	R1.2.14	Task 7.1	All users
R1.7.15	Data scale meta information should be available, such as, but not limited to: (a) The full enumeration of available options for nominal data, (b) the ordering of elements for ordinal data. (e.g. age groups: „Younger than 18“,“18-66“,“Older than 67“), and (c) numerical bounds as well as inclusivity and exclusivity for interval and ratio scales.	R1.2.14	Task 7.1	All users
R1.7.16	Retrieval of ontologies from the data base must be available	R1.2.23	Task 7.1	All users
R1.7.17	Retrieval of portions of the ontology in case of very large ontologies should be available	R1.2.23	Task 7.1	All users
R1.7.18	User Assistance may provide additional information to enable mapping between NL utterance and translated SPARQL response	R1.2.2	Task 7.1	All users
R1.7.19	Relevance rating of individual query terms should be possible.	R1.7.1	Task 7.2	All users
Category R2: Non-Functional Requirements				
Group 1: Scalability				

Code	Requirement Description	Relationship with other requirements	Origin of requirement	Relevant stakeholders
R2.1.1	INODE must be able to scale with respect to more data sources. In this way, the number of data resources should not be limited in a hard way.	R1.1.6	All use cases	Infrastructure Providers, Service Integrators
R2.1.2	The query engine should be able to respond to user queries from the use cases, and moreover should be able to handle multiple users of the use cases at once, such that results are obtained quickly even under load.	R1.1.4	All use cases	Infrastructure Providers, Service Integrators
R2.1.3	The analytics components of the platform should be able to deal with the computational and memory limitations posed by large datasets.	R1.4.7	Task 4.3	Infrastructure Providers, Service Integrators, Developers
R2.1.4	INODE should be capable of accessing Terrabytes of astronomical data available in SDSS database and should also be able to scale with inclusion of other databases for example from other astronomical surveys like HETDEX, EUCLID, etc.	R1.3.2	UC3-Astrophysics	Data Curator, Scientists with proprietary access
Group 2: Performance				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R2.2.1	INODE deploys various big data analytics frameworks that have demands in computational power. They should be regularly evaluated during development, such that they are shown to be accurate with real-time data.	R1.4.5	All use cases	Infrastructure Providers, Service Integrators
R2.2.2	Statistics and reports that should be displayed through the dashboard is a process with high performance needs.	R1.1.3	All use cases	Infrastructure Providers, Service Integrators
R2.2.3	The platform should provide at least 50% reduction in time/effort spent by a user to complete successfully an exploration task	R1.2.6	KPI1	End users

R2.2.4	The platform should provide at least 80% precision of the majority of natural language queries for specific uses cases that are developed in close cooperation with the three use case providers.	R1.2.5	KPI2	End users
R2.2.5	The platform should provide at least 30% reduction in performance overhead of federated INODE queries	R1.3.2	KPI4	Infrastructure Providers, Service Integrators
R2.2.6	All by-example and by-analytics explorations should be optimized to provide interactive times.	R1.2.6	Task 5.1 , Task 5.2	All users
R2.2.7	Query results should be shown within less than a minute. If not possible, partial results should be shown or a progress with roughly estimates about the response time	R1.2.13	Task 5.3	End users
R2.2.8	The system should allow for the efficient syntactic parsing of unstructured text, along with supporting large heap-based graph libraries for efficient knowledge graph modification and traversal.	R1.4.4	Task 4.3	Infrastructure Providers, Service Integrators
R2.2.9	The platform should allow for the training of the latest deep-learning models with intense computational and in-memory processing requirements.	R1.4.4, R1.4.5	Task 4.3	Infrastructure Providers, Service Integrators, Developers
R2.2.10	The analytics components (e.g. ML/DL models) of the platform should provide results in less than one minute.	R1.4.4, R1.4.5	Task 4.3	Infrastructure Providers, Service Integrators, Developers
R2.2.11	Query recommendations should be optimized to provide interactive times.	R1.6.12	Task 6.1	All users
R2.2.12	Natural language explanations should be generated for the majority of common structured queries.	R1.6.5, R1.6.6	Task 6.1	All users
Group 3: Security and Privacy				

Code	Requirement Description	Relationship with other requirements	Origin of requirement	Relevant stakeholders
R2.3.1	INODE platform should respect data access policies.	R1.1.1	UC3-Astrophysics	System administrator
R2.3.2	INODE should be capable of managing different user profiles distinguishing between user roles.	R1.1.1	All use cases	System administrator
R2.3.3	INODE, in accordance with privacy policies, should store privacy covered data in a protected way.	R1.6.19	All use cases	System administrator
R2.3.4	Access to protected data should be possible only to authorized operators.	R2.3.1	All use cases	System administrator
R2.3.5	The applications and technologies used in INODE must respect all regulations concerning the ethical aspects, especially those related with data protection and privacy.	R1.1.1	All use cases	System administrator
R2.3.6	INODE should cover with state of the art technologies all the aforementioned security aspects.	R1.1.1	All use cases	System administrator
R2.3.7	INODE should be compatible with the FAIR principles as well as the security principles of EOSC	R1.1.1	All use cases	System administrator, EOSC Community
Group 4: Reliability and Availability				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R2.4.1	INODE should have a high availability and reliability (e.g. more than 90% in regular operation during the pilots) that can be monitored, measured and audited.		All use cases	Infrastructure Providers, Service Integrators
R2.4.2	In case of failures, measures have to be taken in order to overcome these in short notice and additional measures for preventing their occurrence.		All use cases	Infrastructure Providers, Service Integrators
Group 5: Manageability and flexibility				

Code	Requirement Description	Relationship with other requirements	Origin of requirement	Relevant stakeholders
R2.5.1	INODE should provide a 50% reduction in time/effort spent on integrating new data sources with INODE compared to integrating data without INODE.	R1.3.2	KPI3	End users
R2.5.2	INODE should have a high manageability and flexibility even for users that are not considered experts.	R1.2.2	All use cases	Service providers, Integrators, EOSC Community
R2.5.3	Common management attributes such as add/delete/update should be intuitive and easy to be performed.	R1.2.19	All use cases	Service providers, Integrators, EOSC Community
R2.5.4	Users should be able to “Search anywhere” in order to learn quickly about individual pieces of information		All use cases	End users
Group 6: Modularity				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R2.6.1	The INODE architecture should follow a layered and modular approach.		All use cases	Service providers, Integrators
R2.6.2	The INODE modularity level should allow enough independence of all modules so as if any module needs to be replaced, this will have no consequences to the other modules.		All use cases	Service providers, Integrators
Group 7: Openness and Extensibility				
Code	**Requirement Description**	**Relationship with other requirements**	**Origin of requirement**	**Relevant stakeholders**
R2.7.1	The various components of INODE should be portable across major operating systems.		All use cases	Service providers, Integrators, EOSC Community
R2.7.2	The various components of INODE should be interoperable with other services implementing common and open standards		All use cases	Service providers, Integrators, EOSC Community

R2.7.3	The core components of the INODE framework should be extensible to new types of data structures.	R1.1.6	All use cases	Service providers, Integrators, EOSC Community
R2.7.4	INODE APIs should rely on open standards and built upon other existing open standards where possible.		All use cases	Service providers, Integrators, EOSC Community
R2.7.5	INODE should provide programming interfaces for application developers to gather real-time and historic data.	R1.6.19	All use cases	Service providers, Integrators, EOSC Community
R2.7.6	INODE should reuse existing open source software and tools, where it is appropriate and possible according the license.		All use cases	Service providers, Integrators, EOSC Community
R2.7.7	The architecture of INODE must be open, extensible, providing the ability to add new functional components.		All use cases	Service providers, Integrators, EOSC Community

Table 5-2 *A short description of the different stakeholders that are mentioned in the consolidated requirements table.*

Stakeholder	Explanation
All users	The full set of users that will have access to the INODE platform
Bioinformaticians	A subset of end-users that have interest in cancer biomarkers and related data with the following roles: data scientist, ontologist, bioinformatics teacher, clinical bioinformatician, computer scientist, knowledgebase developer, post-doc in bioinformatics and student.
General public/ Citizens	General public consuming information from the INODE platform (usually through public institutions), having access to a public portal with embedded visualizations in a story-telling approach.
Data administrators /Data managers	Responsible for hosting the databases, managing the data sources by integrating new ones and updating and transforming the existing ones, access (e.g., Web) and the INODE infrastructure and architecture.
Data Curators	Scientists with access to specific data sets either through their affiliation with a particular project or a specific institution, responsible for setting the permissions and updating them as proprietary periods are ending.

Developers	Responsible for maintaining the system, fixing bugs and integrating new features.
Domain Experts	Analysts mainly interested in identifying, generating and exploiting mappings between the extracted ontologies of a specific dataset.
End users	The superset of non-technical users (e.g.policy makers, administrators, bioinformaticians, researchers, scholars etc.) that perform data exploration activities to gain insights.
EOSC Community	Users having access to the European Open Science Cloud (EOSC) hub, a trusted digital platform for the scientific community.
Infrastructure Providers	Stakeholders that will offer their existing infrastructure systems to be utilized for or connected to the INODE platform.
Non-expert user/Policy maker	A subset of end-users that access a collection of relevant datasets in a homogeneous way so that they can make informed decisions.
Researcher/Biologist	A subset of end-users composed of scientists that have interest in cancer biomarkers and related data with the following roles: data scientist, ontologist, bioinformatics teacher, clinical bioinformatician, computer A subset of end-users, scientist, knowledgebase developer, post-doc in bioinformatics and student.
Scientists with proprietary access	A subset of end-users with access to specific astronomical data sets either through their affiliation with a particular project or a specific astronomical institution.
Service providers & Integrators	Responsible for defining the platform's service level targets.
Statisticians	A subset of end-users that acquire raw data from the platform and do the analysis themselves with their own tools.
System administrators	Responsible for creating and managing the system user profiles.

6 CONCLUSIONS

This document presented a first approach to define the use cases and to identify the functional and non-functional requirements, actors (i.e., stakeholders), and the main technologies used to address the requirements. The presented requirements analysis was conducted from both the business and the technical partners of the consortium. Use case providers detailed real-life industrial requirements that if addressed by INODE will enhance existing products and solutions. At the same time, technology providers defined a list of requirements that will ensure that INODE will utilize innovative algorithms and modules, avoiding technical obstacles and limiting the possibility of deviating from the project's objectives and initial vision.

The requirements collected from the three use case providers will be used to create a system that is reasonably complex and feasible to implement, being compatible with existing state-of-the-art IT infrastructures. In addition, the listing of the available technology elements involved in the project along with their role, key features, and relation to the platform's envisioned functionalities, resulted in an additional set of high-level requirements that will drive the design task forward. This deliverable aimed at consolidating the aforementioned requirements from different actors and components, so that the implementation of the platform can be effectively carried out and collaboration or synergies among different system developers and partners can be facilitated. All INODE partners contributed to this endeavor, achieving a consensus among the consortium members on the next phase of the system design: the system architecture.

REFERENCES

[1] Eckhouse, S., Lewison, G., & Sullivan, R. (2008). Trends in the global funding and activity of cancer research. Molecular oncology, 2(1), 20-32.

[2] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. Semantic Web J., 8(3):471– 487, 2017. doi: 10.3233/SW-1602.

[3] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: A survey. In Proc. of the 27th Int. Joint Conf. on Artificial Intelligence, (IJCAI 2018), pages 5511–5519. IJCAI Org., 2018. doi: 10.24963/ijcai.2018/777.

[4] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. J. on Data Semantics, 10:133–173, 2008. doi: 10.1007/978-3-540-77688-8_5.

[5] Roman Kontchakov, Martin Rezk, Mariano Rodriguez-Muro, Guohui Xiao, and Michael Zakharyaschev. Answering SPARQL Queries over Databases under OWL 2 QL Entailment Regime. In Proc. of the 13th International Semantic Web Conference (ISWC 2014), pages 552-567, Lecture Notes in Computer Science 8796, Springer, 2014. doi: 10.1007/978-3-319-11964-9_35.

[6] Ana Claudia Sima, Tarcisio Mendes de Farias, Erich Zbinden, Maria Anisimova, Manuel Gil, Heinz Stockinger, Kurt Stockinger, Marc Robinson-Rechavi, Christophe Dessimoz:Enabling semantic queries across federated bioinformatics databases. Database 2019:, : baz106 (2019)
https://academic.oup.com/database/article/doi/10.1093/database/baz106/5614223.

[7] Danae Pla Karidi, Yannis Stavrakas, and Yannis Vassiliou. Tweet and Followee Personalized Recommendations Based on Knowledge Graphs. Journal of Ambient Intelligence and Humanized Computing, 9(6):2035-2049, 2018.

[8] Mariia Selezniova, Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, Eric Simon. Guided Exploration of User Groups. Under review, 2020.

[9] Andreas Kokkalis, Panagiotis Vagenas, Alexandros Zervakis, Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. 2012. Logos: a system for translating queries into narratives. In Proceedings of the 2012 ACM SIGMOD International Conference on Management.

LIST OF FIGURES

LIST OF TABLES

ANNEX: SQL QUERIES

This Annex provides the SQL code of the queries mentioned in the Use Cases.

UC2: Research and Innovation Policy Making

Query UC2-Q1:

```
select
    q1.year,
    (cast(q1.member_funding as numeric)/cast(q2.total_funding as
numeric))*100
        as percentage_of_funding
from
(
    select
        p.start_year as year, sum(pm.ec_contribution) as member_funding
    from
        unics_cordis.projects p
            join
        unics_cordis.project_members pm
            on pm.project=p.unics_id
            join
        unics_cordis.nuts_hierarchy nh
            on nh.nuts_3=pm.nuts3_code
    where
        nh.nuts_2='ITI1' and p.framework_program in ('H2020','FP7')
    group by
        p.start_year
) as q1
join
(
    select
        p.start_year as year, sum(p.ec_max_contribution) as total_funding
    from
        unics_cordis.projects p
    where
        p.framework_program in ('H2020','FP7')
    group by
        p.start_year
) as q2
on q1.year=q2.year;
```

Query UC2-Q2:

```
select
    inst.name as participant,
    aty.description as activity_type,
    count(distinct p.unics_id) as num_projects,
    sum(pm.ec_contribution) as funding_received
from
    unics_cordis.projects p
        join
    unics_cordis.project_members pm
        on pm.project=p.unics_id
        join
    unics_cordis.nuts_hierarchy nh
```

```
        on nh.nuts_3=pm.nuts3_code
        join
    unics_common.institutions inst
        on inst.unics_id=pm.institution_id
        join
    unics_cordis.activity_types aty
        on aty.code=pm.activity_type
where
    nh.nuts_2='ITI1'
    and p.framework_program='H2020'
group by
    inst.name, aty.description;
```

Query UC2-Q3:

```
select
    inst.name as projec_member,
    etu.description as nuts2,
    p.start_year as year,
    string_agg(prog.title, ' | ') as programmes
from
    unics_cordis.projects p
        join
    unics_cordis.project_members pm
        on pm.project=p.unics_id
        join
    unics_cordis.nuts_hierarchy nh
        on nh.nuts_3=pm.nuts3_code
        join
    unics_cordis.eu_territorial_units etu
        on etu.nuts_code=nh.nuts_2
        join
    unics_cordis.project_programmes pp
        on pp.project=p.unics_id
        join
    unics_cordis.programmes prog
        on prog.code=pp.programme
        join
    unics_common.institutions inst
        on inst.unics_id=pm.institution_id
where
    (nh.nuts_2='ITI1'
    or exists(
        select 1
        from
            unics_cordis.project_members pm2
                join
            unics_cordis.nuts_hierarchy nh2
                on nh2.nuts_3=pm2.nuts3_code
                join
            unics_cordis.project_members pm3
                on pm3.project=pm2.project and
pm3.institution_id!=pm2.institution_id
                join
            unics_cordis.nuts_hierarchy nh3
                on nh3.nuts_3=pm3.nuts3_code
        where
            pm2.institution_id=pm.institution_id
            and nh2.nuts_2!='ITI1' and nh3.nuts_2='ITI1'
    ))
    and p.framework_program in ('H2020','FP7')
group by
```

```
    inst.name, etu.description, p.start_year;
```

Query UC2-Q4:

```
select
    p.start_year as year,
    rd.description as research_domain,
    cou.name as country,
    count(distinct p.unics_id) as num_erc_projects
from
    unics_cordis.projects p
        join
    unics_cordis.erc_calls ercc
        on ercc.ec_call=p.ec_call
        join
    unics_cordis.project_erc_panels p_p
        on p_p.project=p.unics_id
        join
    unics_cordis.erc_panels pnl
        on pnl.code=p_p.panel
        join
    unics_cordis.erc_research_domains rd
        on rd.code=pnl.part_of
        join
    unics_cordis.project_members pm
        on pm.project=p.unics_id
        join
    unics_common.countries cou
        on cou.alpha_2=pm.country
where
    p.framework_program in ('H2020')
    and pm.country in ('ES','PT')
group by
    p.start_year, rd.description, cou.name;
```

Query UC2-Q5:

```
select
    sq2.partner_id,
    sq2.partner_name,
    sq2.partner_nationality,
    sq2.num_projects
from (
    select
        sq.*,
        row_number() over(order by sq.num_projects desc, sq.partner_name
asc) as rnk
    from (
        select
            q.partner_id, q.partner_name, q.partner_nationality,
            count(*) as num_projects
        from
            (
                select distinct
                    partner_inst.unics_id as partner_id,
                    partner_inst.name as partner_name,
                    case
                        when nh2.nuts_2='ITI1' then 'Toscana'
                        when partner.country='IT' then 'Italian'
                        else 'Non-Italian'
                    end as partner_nationality,
```

```
                    p.unics_id as project_id
                from
                    unics_cordis.project_members pm
                        join
                    unics_cordis.nuts_hierarchy nh
                        on nh.nuts_3=pm.nuts3_code
                        join
                    unics_cordis.projects p
                        on p.unics_id=pm.project
                        join
                    unics_cordis.project_members partner
                        on partner.project=pm.project and
                           partner.institution_id!=pm.institution_id
                        join
                    unics_cordis.nuts_hierarchy nh2
                        on nh2.nuts_3=partner.nuts3_code
                        join
                    unics_common.institutions partner_inst
                        on partner_inst.unics_id=partner.institution_id
                where
                    pm.activity_type='PRC'
                    and nh.nuts_2='ITI1'
                    and p.framework_program in ('H2020','FP7')
            ) as q
        group by
            q.partner_id, q.partner_name, q.partner_nationality
    ) as sq
) as sq2
where sq2.rnk<=100;
```

UC3: Astrophysics

Query UC3-Q1:

```
select TOP 100 objID, ra, dec
from PhotoPrimary
where ra > 185 and ra < 185.1 and dec > 15 and dec < 15.1
```

Query UC3-Q2:

```
select top 100 g.objID, gn.distance
from Galaxy AS g
join dbo.fGetNearbyObjEq(185.,-0.5, 1) as gn
  on g.objID = gn.objID
order by distance
```

Query UC3-Q3:

```
select top 10 p.objid, p.ra, p.dec, p.u, p.g, p.r, p.i, p.z,
       s.class, s.z
from PhotoObj as p
join SpecObj as s on s.bestobjid = p.objid
where p.u between 0 and 15 and s.class = 'star'
```

Query UC3-Q4:

```
select top 100 u, g, r, i, z from Galaxy
```

```
where (htmid*37 & 0x000000000000FFFF) < (650 * 1)
```

Query UC3-Q5:

```
select top 100 g.objid, zns.nvote, zns.p_el as elliptical,
       zns.p_cw as spiralclock, zns.p_acw as spiralanticlock,
       zns.p_edge as edgeon, zns.p_dk as dontknow,
       zns.p_mg as merger
from Galaxy as g join ZooNoSpec as zns on g.objid = zns.objid
where g.clean=1 and zns.nvote >= 10 and zns.p_cw > 0.8
```

UC1: Cancer Biomarker Research

Query UC1-Q1:

```
select distinct ens.ensembl_gene_id, ens.gene_symbol
from  oncomx.healthy_expression as he join xref_gene_ensembl as ens on
he.ensembl_gene_id = ens.ensembl_gene_id;
```

Query UC1-Q2:

```
select distinct a.name , a.id
from (healthy_expression as c join stage as st on st.id=
c.uberon_developmental_id ) join anatomical_entity as a on a.id =
c.uberon_anatomical_id
where
st.name like '%young%' and
st.name like '%adult%'
```

Query UC1-Q3:

```
select distinct anat.id, anat.name from
healthy_expression as he join
xref_gene_ensembl as xref on he.ensembl_gene_id = xref.ensembl_gene_id join
anatomical_entity as anat on he.uberon_anatomical_id = anat.id
where
xref.gene_symbol="apoc1"
```

Query UC1-Q4:

```
select  anat.name from
healthy_expression as he join
xref_gene_ensembl as xref on he.ensembl_gene_id = xref.ensembl_gene_id join
anatomical_entity as anat on he.uberon_anatomical_id = anat.id join
stage as st on st.id = he.uberon_developmental_id join species as s on
xref.speciesId = s.speciesId
where
xref.gene_symbol="apoc1" and
s.speciesCommonName = "human"
group by anat.name order by max(expression_score) desc limit 10
```

Query UC1-Q5:

```
select distinct  a.name as Tissue, s.name as Stage, he.expression_score as
Score , sp.speciesCommonName from healthy_expression as he join stage as s
on he.uberon_developmental_id = s.id JOIN xref_gene_ensembl AS g ON
g.ensembl_gene_id = he.ensembl_gene_id join anatomical_entity  as a on
he.uberon_anatomical_id = a.id join species as sp on g.speciesId =
sp.speciesId
where g.gene_symbol = 'TP53'
```

Query UC1-Q6:

```
select distinct d.name from differential_expression as de join project_study
as ps on ps.study_id = de.study_id  join
disease as d on d.id = ps.doid
where de.gene_symbol ='TP53'
and de.pvalue < 0.01
```