
# Enabling Data Enrichment Pipelines for AI-driven Business Products and Services

**HORIZON-CL4-2021-DATA-01-03**

## D1.2 enRichMyData architecture

**Work Package 1**

| | |
|---|---|
| **Type of document:** | Other |
| **Dissemination level:** | Public |
| **Lead beneficiary:** | UB |
| **Authors:** | Cosmin Cara, Cosmin Udroiu, Michele Ciavotta (editors), all other technical partners contributors |
| **Version:** | 1 |
| **Due Date of document:** | 30.09.2023 |
| **Delivery Date of document:** | 29.09.2023 |

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# Document History

| Version | Date | Contributor | Comments |
|---|---|---|---|
| **0.1** | 31.07.2023 | Cosmin Udroiu | Structure and ToC |
| **0.2** | 17.08.2023 | Michele Ciavotta | Architectural view |
| **0.3** | 25.08.2023 | Michele Ciavotta | Tools and Infrastructure services |
| **0.4** | 30.08.2023 | Cosmin Udroiu | ScalR, ReusR and Data Governance |
| **0.4** | 05.09.2023 | Michele Ciavotta | Integration Patterns |
| **0.5** | 08.09.2023 | Michele Ciavotta | Data pipeline example |
| **0.6** | 19.09.2023 | Michele Ciavotta | Implementing Reviewers' remarks |
| **1.0** | 29.09.2023 | Dumitru Roman | Final QA and final version submitted |

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# Executive summary

This document briefly presents the reference architectural view for the enRichMyData project. We began by examining the requirements and challenges outlined in D1.1 and D4.1, while also considering the project objectives, and best practices in DataOps and MLOps. In this context, we identified four core stages encompassing a data pipeline's development cycle, the roles of involved actors, and the inputs and outputs for each stage. Additionally, we considered the nature and current state of development of the tools provided within the project's scope. Furthermore, the reference **architecture** is defined for each stage that identifies the components involved, artifacts, and the data flows. Particular attention has been paid to enabling Human-In-The-Loop (HITL) at the architectural level by design.

The architectural vision proposed in this document is to be read as a high-level blueprint that can direct the development activities of individual tools and services but more importantly the overall integration activity. To facilitate and direct the integration process, several technical aspects of the toolbox tools have been contrasted and **integration patterns** have been identified which are agile and devoted to component flexibility and reusability. These patterns are discussed in this document and explained by means of an illustrative example.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# Contents

1 Introduction ..... 7
2 Deliverable overview ..... 8
2.1 Deliverable purpose, scope and context ..... 8
2.2 Target audience ..... 8
2.3 Deliverable structure ..... 8
2.4 Applicable documents ..... 8
3 enRichMyData architecture ..... 9
3.1 Tools ..... 9
3.1.1 DiscoverR ..... 9
3.1.2 WrappR ..... 9
3.1.3 CleanR ..... 9
3.1.4 LinkR ..... 10
3.1.5 StructR ..... 10
3.1.6 ClassifiR ..... 10
3.2 Infrastructure services ..... 10
3.2.1 ScalR and ReusR ..... 10
3.2.2 Data layer and Data Governance ..... 14
3.2.3 StreamR ..... 15
3.2.4 ResourcR ..... 15
3.2.5 GreenR ..... 15
3.3 Overall architectural view ..... 16
3.3.1 Exploration ..... 18
3.3.2 Design ..... 21
3.3.3 Runtime ..... 23
3.3.4 Continuous Enhancement ..... 27
3.4 Crosscutting functionalities ..... 28
4 Integration patterns ..... 29
4.1 Lightweight integration ..... 29
4.2 Deeper integration ..... 29
4.2.1 Pipeline level integration ..... 29
4.2.2 Platform level integration ..... 30

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

4.3 Pipeline integration pattern via an illustrative example ..... 31
5 Conclusions ..... 33
References ..... 34

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

**List of Tables**

Table 1: Technical requirements supporting the Exploration stage. ..... 19
Table 2: Business requirements supporting the Exploration stage. ..... 20
Table 3: Technical requirements supporting the Design stage. ..... 22
Table 4: Technical requirements supporting the Exploration stage. ..... 22
Table 5: Technical requirements supporting the Runtime stage. ..... 24
Table 6: Business requirements supporting the Runtime stage. ..... 25
Table 7: Technical requirements supporting the Continuous Enhancement stage. ..... 28
Table 8: Business requirements supporting the Continuous Enhancement stage. ..... 28

**List of figures**

Figure 6: Architecture of TAO tool ..... 12
Figure 7: TAO Processing component parameters definition ..... 12
Figure 8: TAO Processing component configuration ..... 13
Figure 9: TAO workflow design ..... 13
Figure 1: enRichMyData Architectural View ..... 16
Figure 2: Reference architecture for the Exploration stage ..... 18
Figure 3: Reference architecture for the Resign stage. ..... 21
Figure 4: Reference architecture for the Runtime stage ..... 23
Figure 5: Reference architecture for the Continuous Enhancement stage ..... 27
Figure 10: Illustrative example of enrichment data pipeline ..... 31

**List of abbreviations**

| Abbreviation | Description |
|---|---|
| **HITL** | Human-In-The-Loop |
| **ML** | Machine Learning |
| **AI** | Artificial Intelligence |

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# 1 Introduction

In the realm of data processing, tools designed for data enrichment play a pivotal role. However, they grapple with significant limitations. These limitations encompass the lack of holistic support for the entire data enrichment process, especially when dealing with complex and AI-driven task (with potential for uncertainty) such as data linking and classification. Furthermore, available solutions often lack the crucial Humans-In-The-Loop (HITL) support required to address and rectify uncertainties in the data enrichment journey. Scalability, repeatability, and resource efficiency also persistently present challenges.

Envisioning a solution to these challenges, enRichMyData plans to deliver an integrated toolbox capable of accommodating the entire lifecycle of data enrichment pipelines. This toolkit includes data discovery tools to explore datasets and enrichment services. It also incorporates advanced data manipulation and cleaning functions. Moreover, the toolbox provides dedicated tools for semantic enrichment, creating linking services, a platform for scalable pipeline execution, a user-friendly pipeline editor, support for streaming data integration, and a tool for assessing the environmental impact of data pipeline deployments.

Regarding the development of the toolbox, the consortium's efforts are directed toward two main objectives. On one hand, they are developing the functionality needed to address technical and research challenges (as identified in Deliverable D1.1 - Technical and Market Requirements [1]) and those of business cases (as presented in Deliverable D4.1 - Business Case Specification [2]). On the other hand, we put substantial effort into providing a clear architectural vision of the toolbox. This vision aims to establish the type and level of interaction between components and users at each step of developing and executing an enrichment pipeline, among other things.

This document presents a concise overview of the reference architectural view for the enRichMyData project. Starting from the requirements and challenges outlined in D1.1 and D4.1, and considering the project objectives, best practices of DataOps [3] and MLOps [4], as well as the nature and current state of development of the tools provided within the scope of the project, four core lifecycle stages were identified to cover the development process of a data pipeline, the actors involved, and the inputs and outputs for each stage. Furthermore, a specific architecture is defined for each stage, identifying the components involved, artifacts, and data flows. Particular attention has been paid to enabling Human-In-The-Loop (HITL) at the architectural level by design.

The architectural vision proposed in this document should be seen as a high-level blueprint that can guide the development activities of individual tools and services, and more importantly, the overall integration activity. To facilitate and guide the integration process, several technical aspects of the toolbox tools have been contrasted, and integration patterns have been identified that are agile and focused on component flexibility and reusability. They are discussed in this document and explained through an illustrative example.

Finally, it's worth noting that, at the time of writing, the architectural view is considered stable. However, following the principles of agile methodologies, it remains open to potential improvements based on feedback from the stakeholders actively involved in its implementation and its use in business cases.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# 2 Deliverable overview

## 2.1 Deliverable purpose, scope and context

enRichMyData will deliver its capabilities as a set of interoperable tools and services that will form the enRichMyData Toolbox. These tools, together with their documentations, are available at https://enrichmydata.github.io/toolbox.

The aims of this deliverable encompass:

1. Illustrating the envisioned lifecycle stages of the project, delineating the process of preliminary data exploration, and of defining and executing data enrichment pipelines.
2. Furnishing a high-level, comprehensive architectural overview of the enRichMyData platform.
3. While the tools within the platform are the focus of deliverables D2.1 - enRichMyData tools v1 [5] and D3.1 - enRichMyData integrated toolbox v1 [6] (which will be delivered, in tandem with this, at month 12), this document will delve into intricate insights of the lifecycle-dependent interactions between the toolbox components, the users and the artifacts (input, output and intermediate data, configuration files, ML models) involved.
4. Elaborating on the integration patterns that various tools can adopt to be seamlessly orchestrated by the ScalR (the orchestration service provided by enRichMyData – see Section 3.2.1) or exploited by the single components of a data pipeline in execution.

## 2.2 Target audience

This deliverable is specifically designed for the stakeholders of the enRichMyData platform, encompassing professionals including software architects, developers, researchers, data workers, and other technical staff.

## 2.3 Deliverable structure

This deliverable is structured as follows:

- Section 3 is devoted to the presentation of the architectural view, its stages, components, actors, input, and output. The requirements, objectives and best practices underpinning each stage are also presented and discussed.
- Section 4 describes the integration patterns both at platform level and at pipeline level. The proposed integration patterns are explained through an illustrative example.

## 2.4 Applicable documents

This document presents a high-level architecture view of the enRichMyData toolbox and is intended to complement deliverables D2.1 and D3.1, which focus specifically on the tools and integration toolbox, respectively.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# 3 enRichMyData architecture

The objective of this section is to offer the reader a comprehensive architectural perspective in which the various components (tools and services) provided by enRichMyData perform their roles and interconnect with each other. This vision is further elaborated through a specific definition of the process for building a data enrichment pipeline, which involves the integration of human involvement within an ongoing improvement process.

The subsequent subsections sequentially introduce tools (section 3.1), infrastructure services (section 3.2), an architectural view (section 3.3), and cross-cutting functionalities (section 3.4). The inclusion of concise descriptions of the tools and services in this document aims to maximize clarity and self-containment.

## 3.1 Tools

This section provides a concise introduction to the tools that are part of the enRichMyData platform. The purpose of this presentation is to offer self-contained information while highlighting the primary functionalities of these tools and their relevance across the life cycle phases discussed in the preceding section. For a more comprehensive understanding, readers are encouraged to refer to deliverable D2.1.

### 3.1.1 DiscoverR

DiscoverR facilitates comprehensive searches for datasets, ontologies, and enrichment services, offering content insights to support their integration into enrichment pipelines. Users can search for keywords in descriptions of cataloged datasets, ontologies, and services, or explore detailed descriptions via a visual interface. Cataloged resources include well-established knowledge bases like WikiData, DBpedia, and Schema.org, along with data from various pipelines. DiscoverR employs semantic data profiling techniques to enhance descriptions, incorporating ontology usage patterns and statistics based on metadata standards like DCAT. These techniques are applied to both semantic and "non-semantic" sources by inferring semantics through annotation services provided by the LinkR component, ensuring compliance with and promotion of FAIR principles.

### 3.1.2 WrappR

WrappR provides data access using a virtual semantic layer and ensures secure access. WrappR is delivered as a semantic graph database with efficient reasoning, cluster, and external index synchronization support. It provides a variety of different type of APIs and access methods as well as different types of data federation and virtualization. Through semantic data access and integration, WrappR provides a practical, robust, and versatile tool to improve access to data.

### 3.1.3 CleanR

CleanR supports the specification of data manipulation transformations, including data cleaning operations and the generation of knowledge graphs from various data formats. Users specify transformations interactively, while specifications will be stored in a machine-readable format to be replicated and reused. CleanR provides a broad set of data transformations and integrates them with generic linking and extension functionalities provided by the ResourcR. CleanR enables data cleaning and

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

enrichment operations to be shared (as asset, text or executable), managed and incorporated as steps in the data pipelines in the ScalR component.

### 3.1.4 LinkR

LinkR provides capabilities for semantic annotation of structured and semi-structured data using reference knowledge graphs and category schemes. Annotations consist of links from elements of the input data to elements of well-established knowledge bases and ontologies (e.g., WikiData, DBpedia, and Geonames), or user defined knowledge graphs made available through the ResourcR). LinkR supports annotations through ML algorithms recommending annotations and a human-in-the-loop approach enabling fine-tuning the recommendations algorithms and revision of the results, ensure high-quality annotations while minimizing the users' effort even on very large data volumes. Annotations will be converted into data transformations, to be used as part of enrichment pipelines.

### 3.1.5 StructR

StructR is the counterpart of LinkR for unstructured data. It generates structured data from the unstructured input text through semantic annotation, linking and extension. The text is processed by linguistic and semantic tools and concept mentions are identified and disambiguated from context. Furthermore, the text is projected into an embedding space by using representation learning. StructR supports a range of different pre-computed embeddings to represent the text and expand the dataset. Extension with custom annotation services is supported through a labelling interface for creating and editing text annotations which can then be used to build new annotation models in a human-in-the-loop fashion.

### 3.1.6 ClassifiR

ClassifiR supports data classification as a service and complements StructR. Whereas StructR identifies properties of parts of the text, ClassifiR labels the documents as a whole. The labels can be part of standard taxonomies, industry classifications, as well as custom sets of labels for which a classifier is built. Custom classification is supported by an interactive graphical interface which allows users to explore a document corpus and create ontologies through clustering, labelling, and querying. ClassifiR automates the classification process and exposing it through a common endpoint independent of the classification used.

## 3.2 Infrastructure services

Infrastructure services allow for lifecycle management, efficient execution (ResourcR, ScalR), reuse (ReusR), timeseries support (StreamR) and energy footprint analysis (GreenR) of pipelines on Big Data. In the following sections, more details about the infrastructure services are reported. For a more comprehensive understanding, readers are encouraged to refer to deliverable D2.1 and D3.1.

### 3.2.1 ScalR and ReusR

ReusR provides infrastructure components for search and recommendation of assets (e.g., datasets, transformations, etc.) related to setting up and running data enrichment pipelines. It provides user login to data management assets, public/private access to assets and editing, sharing and versioning them. ReusR enables users of the enRichMyData toolbox to edit pipelines and promotes their reuse across use cases.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

ScalR in turn provides infrastructure components for executing cleaning, transformation and linking at large scale. ScalR aims at providing horizontal scalability of data enrichment pipelines using software containers, and support for management of the different procedures associated with the execution of data enrichment pipelines flexibly on heterogeneous computing infrastructures. ScalR provides integrated support for specific data enrichment operations in the form of a pipeline through the development of reusable standard templates for setting up such pipelines. ScalR promotes the reuse and modification of existing data enrichment pipelines by exposing them as an integrated deployable unit, as opposed to ad-hoc, non-reusable pieces of code.

As both ScalR and ReusR services are implemented by TAO (which stands for Tool Augmentation by user enhancements and Orchestration), this tool will be adapted for the enRichMyData project. TAO is an open source, lightweight, generic, extensible, and distributed orchestration framework. It allows to reuse (i.e., integrate) commonly used toolboxes (such as, but not limited to some EarthObservation processing tools like SNAP, Orfeo Toolbox, GDAL, PolSARPro, etc.). This framework allows for processing composition and distribution in such a way that end users could define processing workflows by themselves and easily integrate additional processing modules, without any programming knowledge requirements.

As TAO already offers some basic functionalities that cover the above-mentioned requirements, it will be extended in order to support the full set of ReusR and ScalR functionalities. Thus, the following operations will be possible to be made:

- User management and login.
- Creation of descriptors for the Docker images that are used in the pipelines.
- Creation of processing component descriptors for the tool application or services that are used in pipelines.
- Creation of workflow definitions by chaining and interconnecting the processing components of the tools (the output of a processing component is linked to the input of the next processing component) in order to create the pipelines.
- Reuse of existing workflow components.
- Version control for components (Docker components, processing components or workflows) to have a history of those components.

Thus, for the enRichMyData project, TAO will be modified to decouple the EarthObservation data model from the core of the framework and add additional components that are specific to the tools and data types we have in enRichMyData. These components refer to the input data source, supported input files, internal database structure changes, default docker images and processing component descriptions, web interface etc.

The platform will provide thus a means for orchestrating heterogeneous processing components and libraries to process the enRichMyData specific data. This is achieved in the following steps:

- Preparation of resources (including processing components or tools) and data input.
- Definition of a workflow as a processing chain.
- Execution of workflows, scaling up from one to as many nodes as made available.
- Retrieval and visualization of the results, which allows one to see what is executing in the system and where it is executing, as well as the system resource usage (CPU, memory, storage space).

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

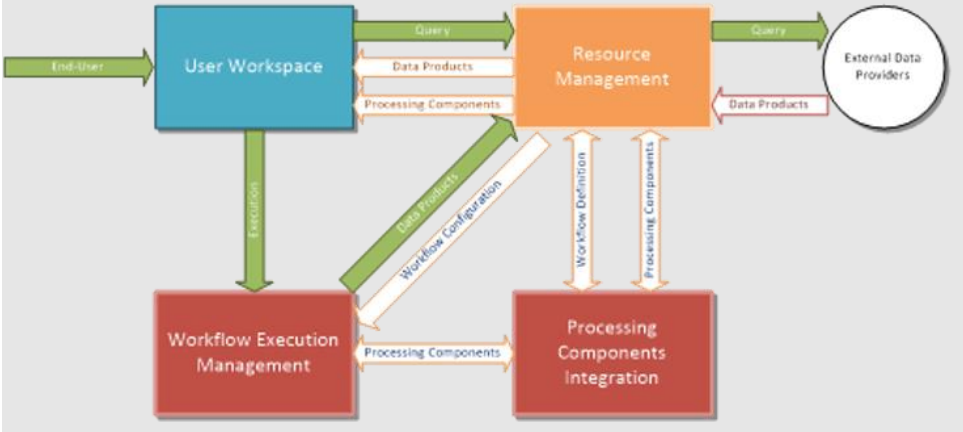


*Figure 1: Architecture of TAO tool*

The processing component, shown in the figure above, represents a standalone application (or tool) defined by the following parameters:

- **Input description:** type of data that the component accepts as an input source (e.g., CSV, JSON, etc.).

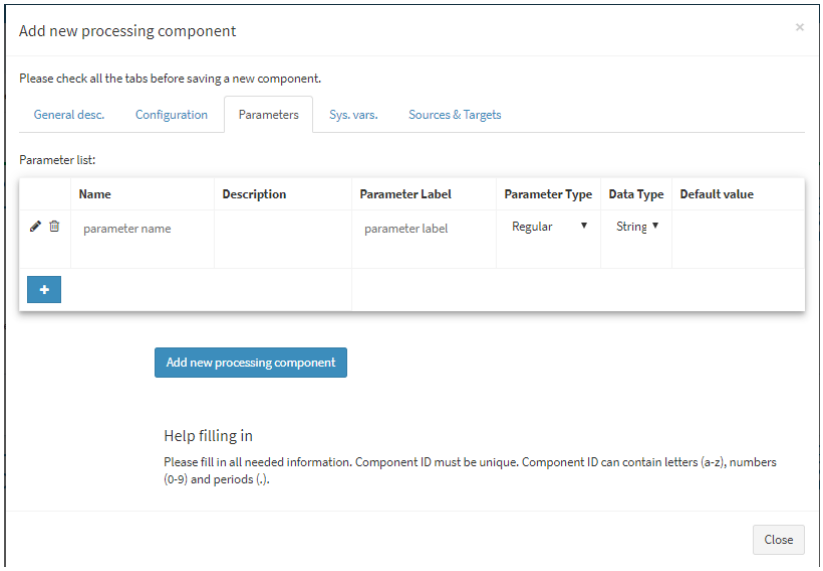


*Figure 2: TAO Processing component parameters definition*

- **Processing operation with execution parameters:** the operation that the component executes with the list of accepted parameters,

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

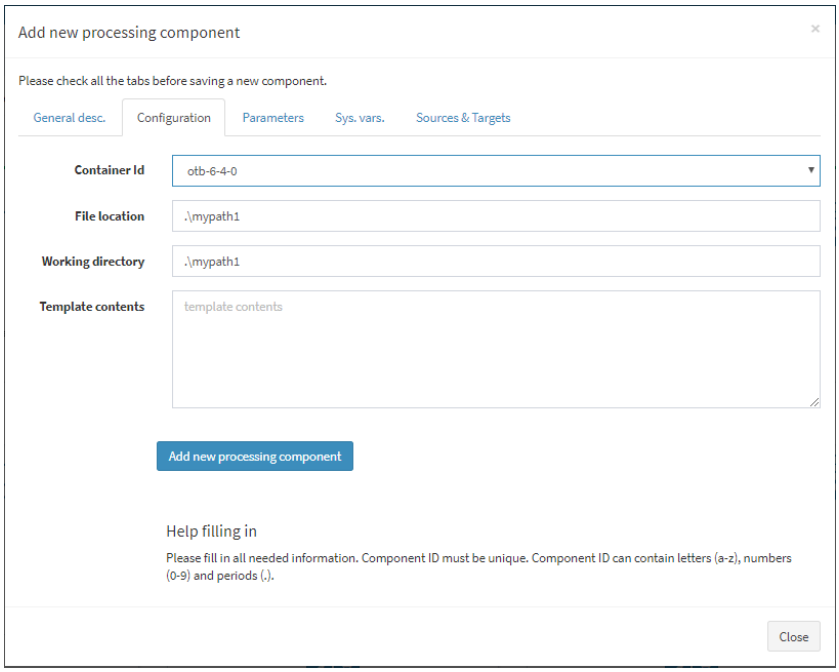


*Figure 3: TAO Processing component configuration*

- **Output description:** type of data provided as a processing operation result. Can consist of one or more files.

From the TAO web interface, the user has the possibility to edit a workflow by adding the desired data source to be used as the initial data provider in the workflow and also to chain the desired processing modules. The chaining of the modules is performed by the user by drag and dropping the available processing components on the drawing canvas and chain them using directed connectors.

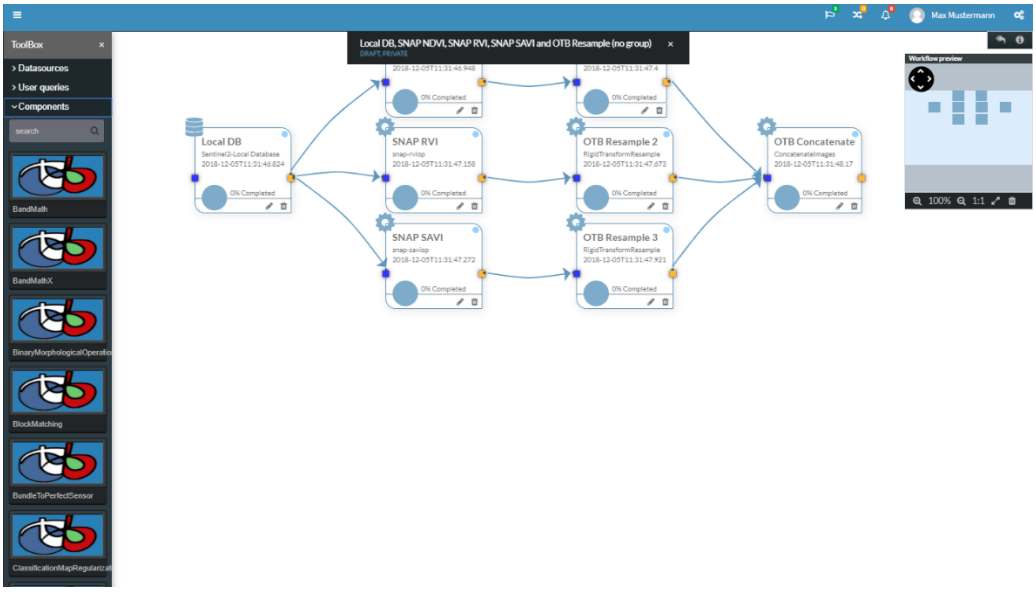


*Figure 4: TAO workflow design*

Another relevant component is the DRMAA (Distributed Resource Management Application API), which provides a standardized access to the DRM systems for execution resources. It is focused on job

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

submission, job control, reservation management, retrieval of jobs, and machine monitoring information. Currently, there are supported DRMAA implementations for local, remote SSH, Torque and SLURM executions. Support for Kubernetes and CWL (Common Workflow Language) implementation is also in progress. The DRMAA implementations are provided as plugins that allow a high flexibility as the current implementation can be changed easily with another.

#### *3.2.1.1 Built-in library*

The TAO framework is built such that it should be as flexible as possible in order to easily accommodate to a variety of components and tools that are used in the execution. This means that it does not come with hardcoded specific tools or services. The services that are built-in are the only ones that implement standards like OGC WPS or export to CWL scripts. Nevertheless, as it has a pluggable architecture, it can be easily extended with newly developed components, according to user needs.

Another option in which the framework can be extended is to include a built-in set of processing components that can perform specific tasks needed in enRichMyData. These components can be included in specific docker images and added as built-in processing components and made available from the internal library of the framework. These components in the built-in library can then be used in workflow creations as any other enRichMyData tool. These will be, for example:

5. Components that perform filtering for rows or column filtering in a tabular data.
6. Components for mapping CSV to JSON files.
7. Components for importing data from RDBMS.
8. Components for exporting data into KG, etc.

Thus, a component mapping CSV to a specific JSON format can be used as intermediate task node between two tools where the output of the first one is CSV but the input of the second is in a JSON format.

This built-in library can be continuously enriched during the project lifetime according to user needs. In the same way, deprecated components can be removed, without any changes in the framework code.

### 3.2.2 Data layer and Data Governance

The data layer is composed from the following components:

#### *3.2.2.1 Configuration and workflow database*

The configuration and workflow database contains the main configuration for users, processing component descriptors, execution nodes, the workflow definitions and all the necessary information needed by the orchestrator for the execution of a workflow like the execution job definition and statuses, the information about the tasks in a certain job execution and events for communicating between the execution tasks and orchestrator. In the case of TAO, the configuration and workflow are stored in a Postgres database.

#### *3.2.2.2 User workspace*

The User workspace is the file system that represents the local repository where all the retrieved or produced data products will be stored. The files in these local processing locations will be used as inputs for the task executions within a workflow job. The outputs after the execution of a certain processing component will be created in this repository. The files in the user workspaces are by default accessible

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

only to the owner user and cannot be accessed by other users. Optionally, a user quota can be set on the user workspace in order to avoid occupying too much disk space that would influence other user's activities.

#### *3.2.2.3 External data sources*

The external data sources represent external repositories that are not under the control of enrichMyData platform. These data sources will be accessed using enRichMyData services and the access will be transparent for the user. Upon creation of a workflow, the user might choose one of these external data sources as the initial source of the workflow data. The data retrieved from external data sources will be saved in the local processing system.

### 3.2.3 StreamR

StreamR provides infrastructure components for streaming and timeseries analytics support in data enrichment pipelines. It pipes data streams from/to appropriate endpoints and ensures high throughput, providing a configurable set of tools for setting up custom streams for new applications.

### 3.2.4 ResourcR

ResourcR provides infrastructure components to support the creation of linking services for a given dataset from a data provider as well as access mechanisms such as search and query. ResourcR enables performant linking and search functionalities with limited effort and expose them as search and linking APIs. The combination of ResourcR and LinkR makes it possible to turn semantic data produced with the toolbox into resources immediately available for reuse.

### 3.2.5 GreenR

GreenR provides infrastructure components to support monitoring of data enrichment pipelines in terms of their environmental impact. It monitors the carbon footprint of the various components in the pipeline and provides the results to the use through a dashboard to log and modulate the environmental impact due to the heavy computations within the pipelines.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

## 3.3 Overall architectural view

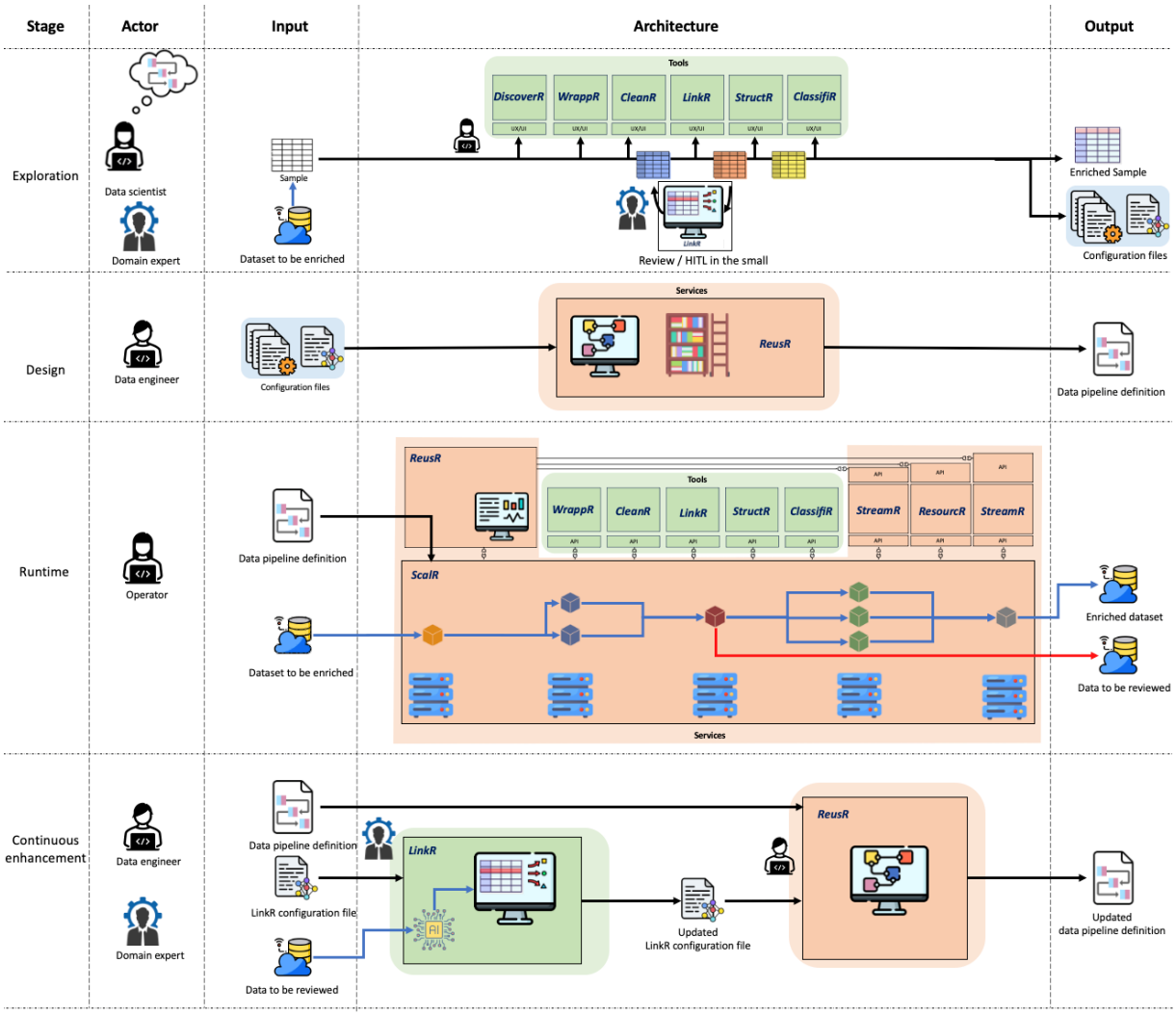


*Figure 5: enRichMyData Architectural View*

The process of developing a data enrichment pipeline in an enterprise environment is generally complex, contemplates several phases, and involves different actors. Within the scope of the enRichMyData project, the requirements and scenarios collected, along with DataOps/MLOps design and integration best practices, were considered to implement a design process aimed at identifying:

1. The main **stages** of the life cycle of a data enrichment pipeline.
2. The main **actors** for each stage. Following a Spiral life cycle model [7] based on User-Centered Design techniques and an agile development approach and starting from the outcomes of Deliverables D1.1 and D4.1, we elicited and normalized for each stage the involved archetypes of users.
3. The **input** and **output** of each stage.
4. The **architectural elements** involved in each phase, i.e.,

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

a. The tools and services of the enRichMyData platform involved in each phase.
b. The relationships among the platform components in each phase.
c. The relationships between the actors/users and the components in each phase.
d. The data flows in each phase.

The result of this work is presented in Figure 5. It proposes an architectural vision to be read as a high-level blueprint that can direct the development activities of individual tools and services but more importantly the overall integration activity. One notable observation from the figure is the distinction in enRichMyData compared to a more traditional DataOps development model following the *Explore-Design-Operate* phases. This is due on a fundamental aspect of the project: the provision of intelligent linking tools flexible enough to adapt to specific data domains. In this context, we recognize the necessity of engaging with domain experts at two crucial points within the project's lifecycle. Consequently, we have expanded the scope of the exploration phase and introduced a dedicated *Continuous Enhancement* stage. This additional phase serves as a pivotal stage where domain experts provide feedback, enabling the tool to refine and enhance the pipeline results before promoting the pipeline to the production stage. This iterative process (runtime – continuous enhancement) is designed to make our intelligent linking tool (LinkR) align progressively with the unique characteristics of each data domain.

Lastly, we emphasize that the architectural definition has been formulated via a rigorous agile process, incorporating a comprehensive review of the requirements documented in D1.1 and D4.1, the main objectives of the project and the best practices of DataOps and MLOps. At the time of writing, the architecture is deemed stable. However, it remains open to potential improvements based on feedback from the stakeholders actively involved in its implementation and its use in business cases.

In the following subsections, the various stages of the life cycle are discussed in detail. For each stage, the goal, actors, input, output, architecture, related project objectives, related challenges, related technical requirements, related business requirements, and related best practices are reported.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

### 3.3.1 Exploration

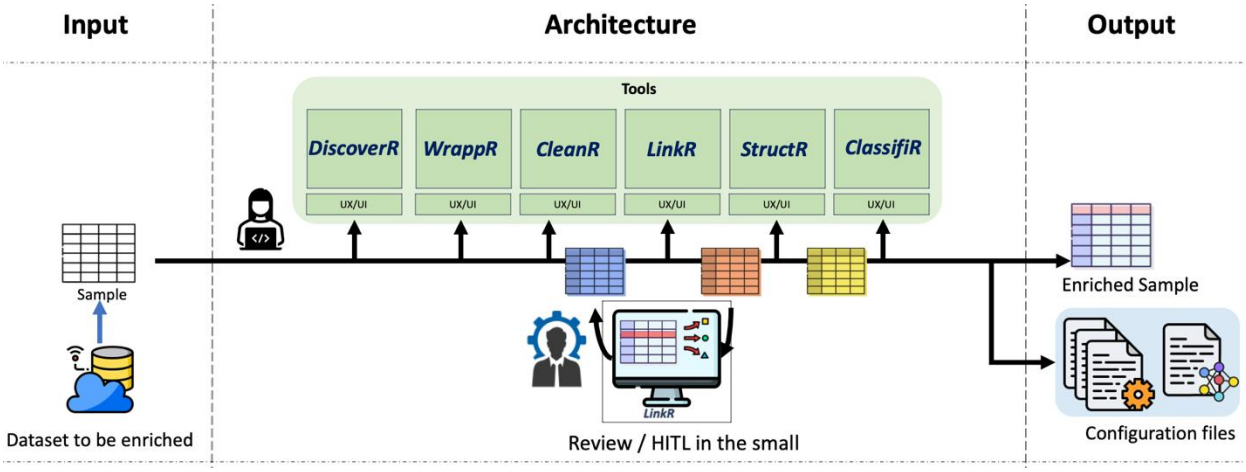


*Figure 6: Reference architecture for the Exploration stage*

**Goal:** The exploration phase is the initial stage where the user identifies the key characteristics of the data at hand and determines the transformations and/or machine learning models to apply for extracting value from the data. During this phase, the user discovers the various steps of the data pipeline that will be designed and implemented in subsequent stages.

**Actors:** The primary actors involved in this phase are the data scientist and the domain expert

**Input:** The user often works with a representative sample of the data that the final data pipeline will process in production.

**Architecture:** Automation in this phase is minimal, and users have the flexibility to interact directly and singularly with the various enRichMyData tools. Specifically, data scientists can either directly use these tools through user-friendly graphical interfaces, if available, or employ rapid prototyping tools like Jupyter notebooks [8] and specialized libraries to interact with the APIs of different enRichMyData tools. As the linking and extension of tabular data holds a central position within the project, the exploration stage provides a more detailed account of the user's interaction with LinkR. In this phase, the user, assuming the role of the domain expert, collaborates iteratively with LinkR to fine-tune the matching model, ensuring it closely aligns with the specifics of the data domain under consideration. It is worth noting that LinkR, although equipped with an intelligent matching model trained on extensive datasets spanning various domains, may encounter challenges when asked to link data featuring distinct structures, especially data that is noisy or originates from unfamiliar domains. To address these challenges, LinkR will engage in dynamic interactions with the human user, presenting cases where it is uncertain of the correctness of its output for review and adjustment. Thereby enhancing the accuracy of the overall linking process (Human-in-the-Loop or HITL). Although not made explicit by the figure, other tools such as ReusourcR and StreamR may also be present at this stage. The former supporting LinkR (ResourcR exposes linking and search functionalities used by LinkR) and the latter for time series analysis.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

**Output:** When considering the transformation and enrichment of data, two scenarios can arise based on the scale and purpose of the task:

1. **Small-scale Data Processing**: If the dataset is relatively small, and there is no need for reusable data processing, the outcome of this phase primarily includes the enriched dataset itself. Additionally, it may include any results generated by the ClassifiR tool models.
2. **Preliminaries for Data Pipeline Design:** On the other hand, if the goal is to design a scalable and reusable data pipeline, this phase yields two key results:
3. **Clear Sequence of Actions**: It involves a well-defined sequence of actions to be executed on the data, each associated with a specific tool. This sequence outlines the step-by-step process for data cleaning, enriching and value extraction.
4. **Tool Configuration Files**: If the chosen tool supports it, a configuration file is generated. This file serves to parameterize the tool and provides a way to record and automatize the actions it will perform within the pipeline. For example, the configuration file produced for LinkR, at a minimum, will include information such as the source column for linking, the knowledge graph (KG) utilized, and the refined matching model.

**Related project objectives**

1. **Objective O3:** Simplify cleaning, linking (to reference resources), and extension of semi-structured data, featuring approaches that enable users to specify such operations visually.

**Related challenges:**

1. **GUI**: Tabular data search provides a GUI to explore information in a table of interest and additional information sources that can be used to augment the information in the table. For these sources, provide information to understand their content and support their usage.
2. **Natural Language Interface**: tabular data search provides language-based interfaces for exploring data augmentation functionalities for a table of interest. Support conversion of instructions given in natural language into tabular data manipulation commands.
3. **Evaluating performance**: Developing effective methods to evaluate the performance of entity linking algorithms is an ongoing challenge, particularly as the datasets and contexts in which entity linking is applied continue to grow in complexity.
4. **Entity linking with limited training data and human-in-the-loop approach**: In many cases, there may be limited training data available for a particular domain or context. In this case, integrating Human-In- The-Loop approaches with active learning to minimize human effort is a key research challenge.

**Related Technical requirements:**

*Table 1: Technical requirements supporting the Exploration stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-DiscoverR-1 | Tabular data search on data about weather, calendar events, organizations |

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

| | | |
|---|---|---|
| 2 | RQ-DiscoverR-2 | GUI for tabular data search |
| 3 | RQ-DiscoverR-3 | Natural language interfaces for tabular data search |
| 4 | RQ-CLEANR-4 | Interactive data cleaning |
| 5 | RQ-CLEANR-7 | Limit human intervention in data cleaning |
| 6 | RQ-LinkR-2 | Uncertain outputs reporting |
| 7 | RQ-LinkR-3 | Uncertain outputs handling |

**Related Business requirements:**

*Table 2: Business requirements supporting the Exploration stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-BC1-5 | Automated access to weather and calendar data |
| 2 | RQ-BC1-6 | Weather data source and extraction of data |
| 3 | RQ-BC1-10 | Filter and correlate datasets |
| 4 | RQ-BC4-2 | Validation of reconciliation, especially of those strings that appear frequently in the dataset |
| 5 | RQ-BC4-4 | Reconciliation scoring |
| 6 | RQ-BC5-14 | User-friendly Interface |

**DevOps/MLOps best practices:**

1. Exploration in DataOps is the pivotal process of analyzing and comprehending datasets, assessing data quality, identifying patterns and relationships, and ultimately informing data processing activities while ensuring that data operations.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

### 3.3.2 Design

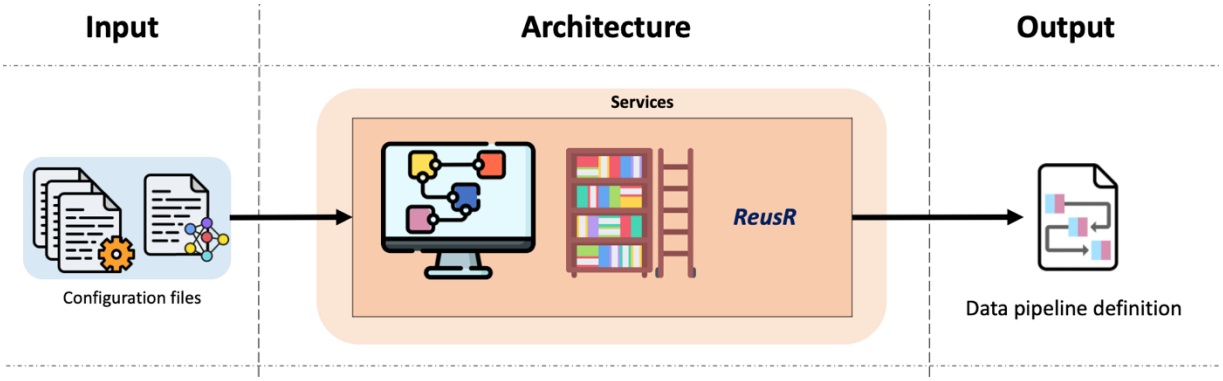


*Figure 7: Reference architecture for the Resign stage.*

**Goal:** The primary objective of this stage is to define the data pipeline, as identified in the previous phase, using a Computer-Aided Design tool called ReusR.

**Actors:** In this phase, the actor must possess the skills of a data engineer, enabling them to design the data pipeline by configuring all its components, including ingestion, preparation, analytics, and export.

**Input:** The input for this stage is essentially derived from the output of the previous phase. The user has access to a clear set of instructions for actions to be performed on the dataset, including cleaning, transformation, linking, extension, and machine learning. Furthermore, the user is provided with information about which tools to employ for these tasks. For some of these tools, a configuration file is also made available to the actor.

**Architecture:** During this stage, the user interacts primarily with a single tool, referred to as ReusR. This service adopts a dataflow programming approach and offers the following functionalities:

1. **Graphical Interface:** Users can interface with ReusR through a graphical interface, which allows them to define a directed acyclic graph (DAG). This DAG consists of transformations (nodes) and the flow of data between these transformations (arcs).
2. **Library of Actions and Transformations:** ReusR provides a comprehensive library of actions and transformations. Examples of actions include reading files from specific directories within a filesystem. Transformations, particularly within the scope of the enRichMyData project, encapsulate the functionality of various tools. Details regarding the creation of transformations and their integration with tools are elaborated upon below.
3. **Data Pipeline Definition Management:** ReusR facilitates the saving and versioning of data pipeline definitions for reuse, ensuring that they can be efficiently managed and reused as needed.

**Output:** The outcome of this stage is the precise definition of the data pipeline, presented in a format that is intelligible to the ScalR service. This defined pipeline corresponds to the one identified during the exploration stage.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

**Related project objectives:**

1. **Objective O3:** Simplify cleaning, linking (to reference resources), and extension of semi-structured data, featuring approaches that enable users to specify such operations visually.

**Related technical requirements:**

*Table 3: Technical requirements supporting the Design stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-LinkR-1 | Integration into existing pipelines |
| 2 | RQ-REUSR-1 | ReusR component usage |
| 3 | RQ-REUSR-2 | ReusR component versioning |
| 4 | RQ-UWS-1 | Workflow definition and management |
| 5 | RQ-UWS-2 | Workflow validation |
| 6 | RQ-SUI-1 | User interface access |
| 7 | RQ-SUI-4 | User interface workflows |

**Related business requirements:**

*Table 4: Technical requirements supporting the Exploration stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-BC1-8 | Pipeline sharing requirements |
| 2 | RQ-BC5-14 | User-friendly Interface |

**DevOps/MLOps best practices:**

1. **Automated Data Pipeline**: Automate data integration, processing, and transformation tasks to reduce manual intervention and the risk of errors.
2. **Version Control**: Apply version control to all aspects of the data pipeline, including code, configurations, and data transformations.
3. **Data Lineage and Dependency Tracking**: Establish data lineage tracking to understand how data flows through the pipeline and identify dependencies.
4. **Change Management**: Implement change management processes to manage changes to data infrastructure, pipelines, and schemas in a controlled and documented manner.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

### 3.3.3 Runtime

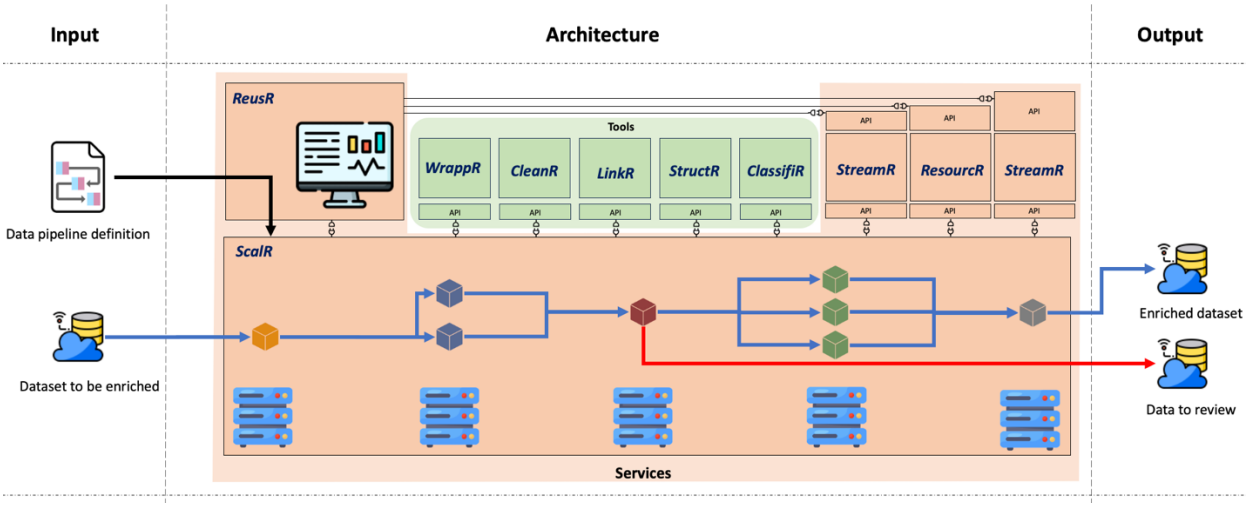


*Figure 8: Reference architecture for the Runtime stage*

**Goal:** In this stage, the previously defined pipeline is brought to life. It undergoes instantiation, operation (including deployment, execution, scaling, and undeployment/redeployment), and continuous monitoring.

**Actors:** The primary actor involved in this phase is typically the infrastructure operator, responsible for both software and computational aspects. Occasionally, they may be assisted by roles like data engineers.

**Input:** At this stage, having a formal pipeline definition is essential. This definition encompasses all components, which are drawn from a library of pre-set or custom-built elements compatible with the infrastructure, as established in the previous stage. It's also crucial to have a clear understanding of the characteristics of the pipeline's processing infrastructure (e.g., nodes, core count, memory) to make informed operational decisions that ensure scalability. The pipeline takes as input a dataset, which can be substantial in size and must follow a known structure to be processed correctly.

**Architecture:** Automation and component complexity are notably high at this stage. A central architectural element is ScalR, designed to instantiate the pipeline within a computational infrastructure like Kubernetes. ScalR handles scaling, replicating pipeline elements as needed, and ensures that the pipeline's state aligns with the user-defined specification. For instance, in case of a pipeline element failure, ScalR can be configured to restart it and resume data processing. Users can provide pipeline definitions directly to ScalR or rely on integration between ReusR and ScalR. The data pipeline elements, once operational, may require tools to execute specific functionalities. In addition, services enhance ScalR's capabilities by offering features such as produced energy consumption estimates (GreenR), time-series analytics (StreamR), and linking and search functionality (ResourcR). Lastly, ReusR is responsible for monitoring various stages of the pipeline lifecycle, providing information and statistics, including metrics like energy consumption during execution.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

**Output:** In this phase, two types of datasets are expected as outputs. The first contains the results generated by the data pipeline, while the second is designed to capture instances where the linking model driving the process exhibits the highest uncertainty. These uncertain instances can be isolated and stored separately to avoid compromising the results of machine learning models and to support further refinement of the linking model.

**Related challenges:**

1. **Scalability:** As datasets grow in size and complexity, it becomes increasingly difficult to efficiently match entities. Developing algorithms and techniques that can handle large-scale datasets is a key challenge in entity linking.
2. **Evaluating performance**: Developing effective methods to evaluate the performance of entity linking algorithms is an ongoing challenge, particularly as the datasets and contexts in which entity linking is applied continue to grow in complexity.
3. **Efficiency**: Efficient algorithms or pipelines, especially for argument extraction, that can handle big data in an efficient way remains an open problem.
4. **Measuring**: Definition of a general scalability benchmarking methodology.
5. **Management**: Definition of a data workflow management system for container-based system.
6. **Real-Time Processing:** Streaming data requires real-time analysis to extract timely insights and enable prompt decision-making. Processing large volumes of streaming multivariate time series data in real-time poses computational and latency challenges.
7. **Model Maintenance and Continuous Learning**: Industrial systems and processes evolve over time, requiring models to adapt and continuously learn from new data. Developing techniques for model maintenance, retraining, and online learning with streaming data is an ongoing research challenge.

**Related project objectives:**

1. **Objective O5:** Support the management of data enrichment pipelines, including creation and operation of data, linking and extension of services, a framework for deployment and execution of pipelines at large scale, and reuse and extension of existing pipelines to deliver a hub of data and services for data enrichment.
2. **Objective O6:** Support data streaming in data enrichment pipelines, featuring support for setting up appropriate endpoints and ensuring high throughput pipeline execution.
3. **Objective O7:** Monitor and reduce energy consumption for executing data enrichment pipelines by using models to estimate and track their carbon footprint.

**Related technical requirements:**

*Table 5: Technical requirements supporting the Runtime stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-LinkR-1 | Integration into existing pipelines |
| 2 | RQ-LinkR-2 | Uncertain outputs reporting |

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

| | | |
|---|---|---|
| 3 | RQ-LinkR-4 | Ability to process a large amount of data |
| 4 | RQ-StructureR-10 | Off-line operation |
| 5 | RQ-ClassifiR-2 | Offline Classification |
| 6 | RQ-ScalR-1 | User workflow execution |
| 7 | RQ-ScalR-2 | Executable tools integration and execution |
| 8 | RQ-ScalR-3 | Web services integration and execution |
| 9 | RQ-ScalR-4 | Tool execution parallelization |
| 10 | RQ-ScalR-5 | Flexibility on heterogeneous components |
| 11 | RQ-ScalR-6 | Horizontal scalability |
| 12 | RQ-SCALR-7 | ScalR Kubernetes execution |
| 13 | RQ-SCALR-8 | Handling big data |
| 14 | RQ-SCALR-9 | Data enrichment pipelines as integrated deployable units |
| 15 | RQ-SCALR-10 | Automatic scalability |
| 16 | RQ-SCALR-12 | ScalR execution monitoring |
| 17 | RQ-GreenR-1 | Live Carbon Intensity |
| 18 | RQ-GreenR-2 | Visual dashboard |
| 19 | RQ-GreenR-3 | Interface with other services |
| 20 | RQ-UWS-1 | Workflow definition and management |
| 21 | RQ-UWS-3 | Workflow execution |
| 22 | RQ-SUI-1 | User interface access |
| 23 | RQ-SUI-3 | User interface dashboard |
| 24 | RQ-SAPI-1 | System access via API |
| 25 | RQ-SAPI-2 | Integrating components via REST or OGC WPS API |
| 26 | RQ-SAPI-3 | Integration with GreenR |

**Related business requirements:**

*Table 6: Business requirements supporting the Runtime stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-BC1-1 | Retrieve and provide the information needed by analysis tools |
| 2 | RQ-BC1-5 | Automated access to weather and calendar data |
| 3 | RQ-BC1-6 | Weather data source and extraction of data |

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

| | | |
|---|---|---|
| 4 | RQ-BC1-7 | Seamless integration of the data enrichment pipeline in JOT software platform |
| 5 | RQ-BC1-8 | Pipeline sharing requirements |
| 6 | RQ-BC1-10 | Filter and correlate datasets |
| 7 | RQ-BC1-22 | Compatibility with the orchestrator for process management |
| 8 | RQ-BC1-23 | Integration with the orchestrator to receive commands and provide status updates |
| 9 | RQ-BC1-24 | Handling of errors and providing detailed error reporting |
| 10 | RQ-BC1-25 | Horizontal scalability |
| 11 | RQ-BC1-27 | Processing of large tabular datasets in the order of 100s of Gigabytes |
| 12 | RQ-BC1-28 | Integration with a relational database for data storage |
| 13 | RQ-BC1-32 | Consideration of scalability, performance, and parallel processing for efficient execution |
| 14 | RQ-BC1-33 | Provision of data access and retrieval APIs for querying and filtering |
| 15 | RQ-BC2-1 | Pipeline sharing |
| 16 | RQ-BC2-4 | Data/Tool Integration |
| 17 | RQ-BC3-10 | On-premises deployment |
| 18 | RQ-BC4-4 | Reconciliation scoring |
| 19 | RQ-BC5-14 | User-friendly Interface |
| 20 | RQ-BC5-15 | Scalable System Architecture |
| 21 | RQ-BC6-3 | Process monitoring and fault detection |

**DevOps/MLOps best practices:**

1. **Automated Data Pipeline**: Automate data integration, processing, and transformation tasks to reduce manual intervention and the risk of human error.
2. **Monitoring and Logging**: Set up comprehensive monitoring and logging for data pipelines to detect issues in real-time and troubleshoot problems quickly.
3. **Data Quality Assurance**: Establish data quality standards and perform data profiling, validation, and cleansing to ensure accurate and reliable data.
4. **Scalability and Elasticity**: Design data architectures and infrastructure to be scalable and elastic to handle growing data volumes and workload demands.
5. **Data Lineage and Dependency Tracking**: Establish data lineage tracking to understand how data flows through the pipeline and identify dependencies.
6. **Containerization**: Package machine learning models and their dependencies into containers (e.g., Docker) for consistent and portable deployment across different environments.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

7. **Orchestration**: Use orchestration tools like Kubernetes to manage containerized machine learning applications, ensuring scalability, availability, and resource allocation.

### 3.3.4 Continuous Enhancement

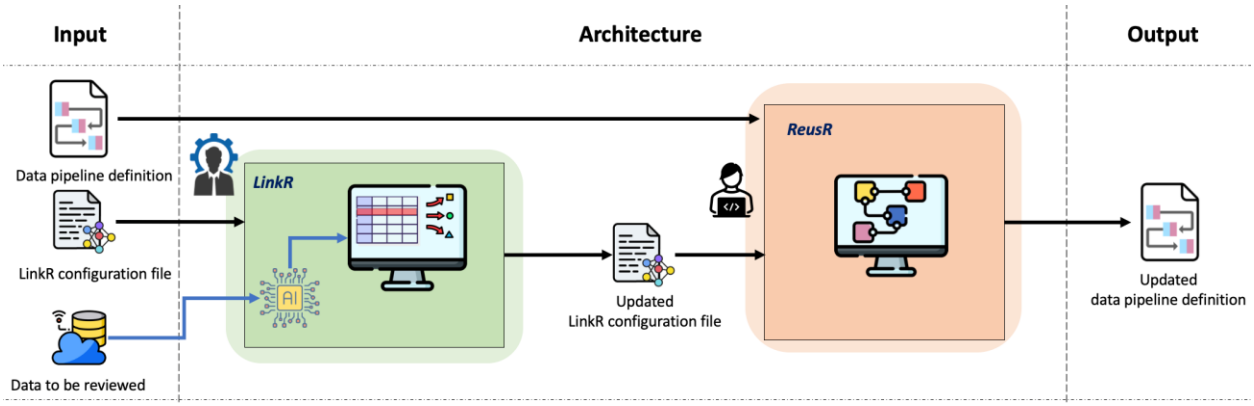


*Figure 9: Reference architecture for the Continuous Enhancement stage*

**Goal:** The primary objective of this phase is to enhance the data pipeline to achieve higher data quality outcomes. To accomplish this, we will utilize the dataset containing uncertain data points, which will undergo analysis in collaboration with human users (Human-In-The-Loop - HITL).

**Actors:** This phase involves two key actors: the domain expert and the data engineer. The domain expert plays a pivotal role in reviewing the most uncertain matches and refining the linking model. In contrast, the data engineer utilizes the refined model to update the data enrichment pipeline definition.

**Input:** During this stage, we leverage the uncertain data generated in the preceding phase. These data are carefully analyzed and processed to present a manageable subset for review by domain experts. As an intermediate outcome, a refined matching model specific to the dataset is generated. This model, a component of the LinkR configuration file, is subsequently employed by the data engineer responsible for pipeline definition, which also serves as input.

**Architecture:** Two essential architectural components are engaged in this phase: the LinkR and the ReusR. The LinkR receives the dataset containing uncertain matches as input, and if the dataset is excessively large, it employs intelligent sampling techniques to reduce the number of uncertain cases presented to the domain expert. The objective of the LinkR in this context is to further fine-tune the underlying matching algorithm model to align with the user's data. This updated model will then be employed by the data pipeline designer to revise the pipeline description.

**Output:** The outcome of this phase comprises an updated version of the data pipeline, ready for use in the runtime phase, thereby contributing to improved data quality and effectiveness in subsequent data processing operations.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

**Related challenges:**

1. **Evaluating performance**: Developing effective methods to evaluate the performance of entity linking algorithms is an ongoing challenge, particularly as the datasets and contexts in which entity linking is applied continue to grow in complexity.
2. **Entity linking with limited training data and human-in-the-loop approach:** In many cases, there may be limited training data available for a particular domain or context. In this case human-in-the-loop approaches with active learning to minimize human effort is a key research challenge.

**Related technical requirements:**

*Table 7: Technical requirements supporting the Continuous Enhancement stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-LinkR-2 | Uncertain outputs reporting |
| 2 | RQ-LinkR-3 | Uncertain outputs handling |
| 3 | RQ-LinkR-4 | Ability to process a large amount of data |

**Related business requirements:**

*Table 8: Business requirements supporting the Continuous Enhancement stage.*

| # | Requirement ID | Requirement Short Title |
|---|---|---|
| 1 | RQ-BC4-4 | Reconciliation scoring |

**DataOps/MLOps best practices:**

1. **Feedback Loops**: Establish feedback loops between data consumers and data producers to continuously improve data quality and relevance.
2. **Feedback and Continuous Improvement**: Collect feedback from data consumers and stakeholders to identify areas for improvement in the DataOps process and make iterative enhancements.

## 3.4 Crosscutting functionalities

In the context of complex distributed software architectures, cross-cutting functionalities refer to aspects of the system that transcend individual components and have a significant impact on the entire architecture. Features such as authorization and authentication fall within this category, and there are several requirements that explicitly mention these features, even though they are outside the specific scope of the project. Nevertheless, as of the time of writing this document, it is plausible to consider that the authentication and authorization features of ReusR and ScalR will be employed for the integration of infrastructure services. For access to the tools within a given pipeline, authorization management (including usernames, passwords, and tokens) will be handled on a per-component pipeline basis (refer to Section 4 for more detailed information on integration patterns).

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# 4 Integration patterns

As has been discussed in previous sections, the actual software architecture proposed by the enRichMyData project depends on the specific stage of the life cycle of a given enrichment pipeline being considered.

## 4.1 Lightweight integration

Almost all stages in the proposed architectural view are characterized by light integration in that the tools are either used separately (Exploration and Continuous Enhancement) or only one tool is used (Design). In these cases, integration is essentially implemented at the data layer (logical data layer) so that all tools can access a certain user's data.

Another possible lightweight integration pattern that can be implemented in the exploratory phase is called portal integration. In this case, the integrator consists of a notebook (e.g., Jupyter) and libraries that can interact with the tools' APIs. The presence of a logical data layer to facilitate the use of different tools is also possible here but a service-oriented approach in which data is passed along with the request would provide more flexibility and safety.

## 4.2 Deeper integration

The architecture envisioned for the runtime phase, on the other hand, is complex and involves virtually all of the tools and services offered by the project. In this scenario, integration is necessarily deeper as the various components are required to communicate with each other and cooperate to implement evolved functionality. In this integration scenario, we distinguish two cases: integration between infrastructure services (platform level) and integration between enrichment data pipeline components and tools (pipeline level).

It is worth noting that there are scenarios where integration is both at the platform level and at the pipeline level as in the case of the integration of the GreenR service. In fact, for this service, integration with the various pipeline components is required so that the service can collect the data needed to calculate the pipeline's carbon footprint, and at the same time, it is necessary for the GreenR to be able to collaborate with the platform to perform monitoring.

### 4.2.1 Pipeline level integration

As mentioned, integration at the data pipeline level concerns the execution of transformation identified during the Exploration phase by specific steps in the pipeline. It is therefore necessary to ask how the use of independent tools can be automated while maintaining a high level of independence, flexibility, and reusability. In what follows, two simple and effective (non-exclusive) integration patterns are presented. These patterns cater to principled approaches to facilitate the integration of external tools (not limited to the ones provided by the project) into ScalR, our agnostic workflow engine used to orchestrate container-based data pipelines.

**Service-oriented integration:** In order to take advantage of a certain functionality within the pipeline it will be necessary to use a dummy component that can handle the input data, call the remote system using

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

an API and, once the request is completed, generate the output so that the pipeline can continue its execution. Depending on the type of API implemented by the remote system to be integrated, we can have a synchronous pattern in which the component remains waiting for the remote system to respond or an asynchronous pattern, in which, that is, the component can continue its execution while maximizing the level of parallelism and optimizing pipeline latency. It's important to include an error handling and recovery strategy as part of the overall solution.

- **Benefits**: The main advantage of this approach is the simplicity of integration; there is no need to rewrite or bring specific code into the pipeline. The fact that ScalR is agnostic to the technology used and the specific communication protocol is the basis of this pattern. Another important advantage is that the remote service can be powerfully resource-hungry and slow to start, making it unsuitable for incorporation into a data pipeline.
- **Drawbacks**: The main disadvantage relates to slow network communication and to the often limited bandwidth available. This can make processing large amounts of data virtually impossible. This disadvantage can be mitigated, when possible, by locally installing the service to be invoked near the component that invokes it. This makes it possible to act on the network infrastructure.

**Function-oriented Integration:** This approach departs from the remote service invocation paradigm and champions the encapsulation of critical functionalities within software containers. This approach is grounded on the principles of functional programming and modularization to achieve streamlined and side-effect-free data processing. Specifically, to implement this pattern, the functionality of the tools to be used in the data pipeline to be executed must be extracted, refactored, and encapsulated in the form of (possibly stateless and side-effect-free) data processing functions.

- **Benefits**: This approach harmoniously integrates with containerized data pipelines (implemented by ScalR). Moreover, it eliminates the need for remote service invocation, reducing latency and ensuring swift and efficient data pipeline execution, making it indispensable for real-time data processing. Functions seamlessly integrate with containerized data pipelines. Resource allocation becomes dynamic, adapting seamlessly to workload fluctuations, optimizing overall efficiency. Functions simplify the scalability and elasticity of data processing tasks. The modularized approach allows for the dynamic scaling of individual components within containers, ensuring optimal resource utilization as data workloads fluctuate. Recovery processes are streamlined. The encapsulation of functionalities within containers allows for easier fault isolation and recovery, ensuring data pipelines maintain reliability.
- **Drawbacks**: Implementing this approach necessitates access to the source code of the tool being modularized. Untangling and separating different parts of the tool into containers is only feasible when the source code is available. Moreover, it involves significant effort in untangling and separating different parts of the tool and orchestrating them.

### 4.2.2 Platform level integration

Integration at the platform level mainly concerns the infrastructure services provided by the project. As far as ReusR and ScalR are concerned, the integration is already complete, being implemented by the same tool, TAO. These two services form the core of the data pipeline execution platform provided by the project, the other services will not be integrated but will be used by ReusR, ScalR or both. Based on the analysis of the available services and the architecture of TAO, the integration approach that will most

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

likely be followed is service-oriented. The services of ReusR and ScalR will then interact with the other infrastructure services by means of the APIs that the latter make available.

## 4.3 Pipeline integration pattern via an illustrative example

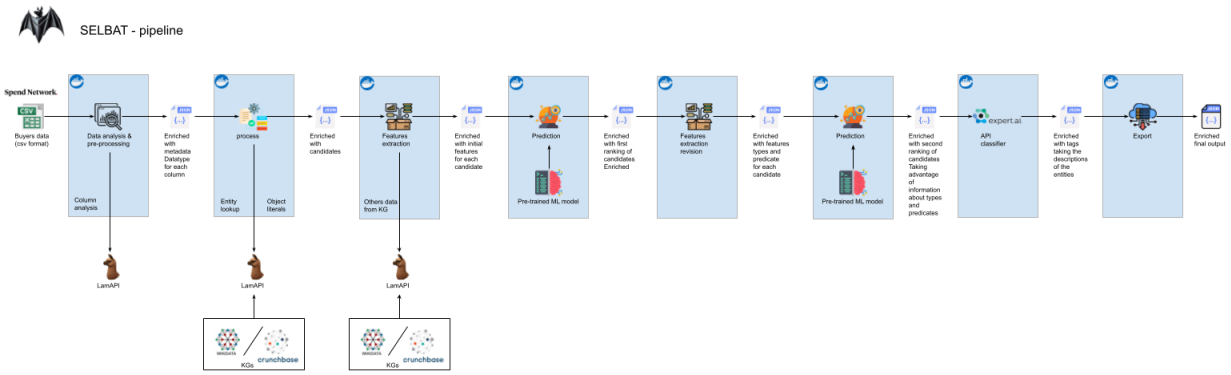


*Figure 10: Illustrative example of enrichment data pipeline*

Figure 10 presented above serves as an illustrative guide to two integration patterns recommended for pipeline-level integration (more details on the specific data pipeline are reported in Deliverable D3.1).

The **function-oriented Integration** approach involves dissecting the core functionalities of SelBat (LinkR). This decision significantly enhances flexibility and promotes maximal component reuse among data pipelines. It's worth noting that the implementation of the function-oriented pattern is somewhat hybrid in the case of the first three pipeline components presented. Specifically, it maintains unchanged the (API-based or service-oriented) interaction between certain functions provided by SelBat and LamAPI (ResourcR). This hybrid approach proves effective in scenarios where high resource demands (CPU and RAM) and large startup times would make encapsulation within a single component impractical. It's also valuable when the tool can be efficiently shared among multiple pipelines. In this context, LamAPI's resource-intensive nature and its ability to serve multiple data pipelines concurrently are advantageous. Moreover, LamAPI's open-source nature offers deployment flexibility, allowing installation either in close proximity to ScalR for reduced latency or in remote configurations to accommodate specific infrastructure needs. On the other hand, components 4 through 6 of SelBat are self-contained and fully implement the proposed function-oriented pattern.

The **service-oriented Integration** pattern is observed in the pipeline component interfacing with the ExpertAI solution, which provides remote classification services. ScalR's agnosticism towards the components it orchestrates is crucial here, as it enables seamless communication with remote services using specific protocols, irrespective of programming languages or libraries employed. This inherent adaptability empowers developers and ensures efficient and adaptable integration across diverse scenarios. In this specific case, the pipeline component essentially comprises a wrapper script that receives input data from the preceding pipeline stage, invokes the remote service, waits for the result, and uses the outcome of the invocation to generate its output. It's important to note that even with this

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

pattern, specific code implementation is required. However, generally, this code tends to be simpler (and therefore quicker to create and test) than the alternative approach.

In conclusion, it's essential to highlight a crucial observation: Regardless of the chosen integration pattern, the reusability of a particular component is not guaranteed. Therefore, during the design and implementation phase of the component, it's vital to decouple functionality as much as possible from the format and domain of input and output data. Additionally, ensuring a high level of configurability for the component is essential to adapt it effectively to different scenarios.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# 5 Conclusions

The objective of this document was manifold. Firstly, it aimed to characterize the lifecycle of a data enrichment pipeline in a manner that by design supports Human-In-The-Loop (HITL) interactions. Secondly, it aimed to identify, for each stage of the proposed lifecycle, a reference architecture. This architecture specified which tools, provided by the project, should be utilized and how. Additionally, it clarified the relationships between the tools, the involved users, the artifacts, and the data flows.

Lastly, the document provided guidance for both tool development and integration efforts. It achieves this by identifying recommended patterns for two key aspects: the integration of services to create an advanced platform for executing data pipelines, and the integration of tool functionality within individual pipelines.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.

# References

[1] M. Palmonari, F. De Paoli, enRichMyData Consortium, “D1.1: Technical and Market Requirements”, 2023. enRichMyData project, M9 Deliverable. Grant agreement ID: 101070284.

[2] F. Perales, C. Parrondo, C. Xuan Chu, E. Kharlamov, Q. Gao, A. Young, I. Makgill, L. Rei, B. Massri, T. Song, “D4.1: Business Case Specification”, 2023. enRichMyData project, M9 Deliverable. Grant agreement ID: 101070284.

[3] H. Atwal, “Practical DataOps: Delivering agile data science at scale”, 2019. Apress.

[4] D. Kreuzberger, N. Kühl, S. Hirschl, “Machine learning operations (mlops): Overview, definition, and architecture”. IEEE Access.

[5] K. Buza, R. Avogadro, B. Elvesæter, enRichMyData Consortium, “D2.1 - enRichMyData tools v1”, 2023. enRichMyData project, M12 Deliverable. Grant agreement ID: 101070284.

[6] C. Udroiu, C. Cara, L. Nicola, enRichMyData Consortium, “D3.1 - enRichMyData integrated toolbox v1”, 2023. enRichMyData project, M12 Deliverable. Grant agreement ID: 101070284.

[7] B. W. Boehm, "A spiral model of software development and enhancement", 1988. Computer 21.5: 61-72.

[8] T. Kluyver, "Jupyter Notebooks-a publishing format for reproducible computational workflows", 2016. Elpub 2016: 87-90.

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON EUROPE RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO 101070284.