Policy Cloud

Cloud for Data-Driven Policy Management

CLOUD FOR DATA-DRIVEN POLICY MANAGEMENT

Project Number: 870675 Start Date of Project: 01/01/2020 Duration: 36 months

D4.1 REUSABLE MODEL & ANALYTICAL TOOLS: DESIGN AND OPEN SPECIFICATION 1

Dissemination Level	PU
Due Date of Deliverable	31/08/2020, M8
Actual Submission Date	03/09/2020
Work Package	WP4 Reusable Models & Analytical Tools
Task	All
Type	Report
Approval Status	
Version	0.8
Number of Pages	p.1 – p.42

Abstract: Internal architecture of the Integrated Acquisition and Analytics Layer, responsible for the integration of analytical tools in extensible manner, registration of new data sources and applying the required transformation and cleaning on the data fusion pass. Specification and design of the built-in analytics tools for Situational Knowledge, Opinion Mining & Sentiment Analysis, Social Dynamics & Behavioral Data analysis.

PolicyCLOUD has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870675.

Versioning and Contribution History

Version	Date	Reason	Author
0.1	28/07/2020	Initial version	Ofer Biran (IBM)
0.2	16/08/2020	Initial content for sections 1-3	Ofer Biran (IBM)
0.3	23/08/2020	Content for 4, 5, 5.1, 5.2	Ofer Biran (IBM), María Ángeles Sanguino, Jorge Montero (ATOS)
0.4	25/08/2020	Content for 6.3 Content for 4.1, 4.3, 5.3, 6.3	Sandra Ebro, Alejandro Ramiro (LXS), Ofer Biran (IBM), Argyro Mavrogiorgou, Thanos Kiourtis, George Manias, Nikitas Sgouros (UPRC)
0.5	26/08/2020	Content for 6, updates	Ofer Biran, Oshrit Feder (IBM)
0.6	28/08/2020	Updated version after internal review	Ofer Biran (IBM), Rafael del Hoyo (ITA), Pavlos Kranas (LXS)
0.7	01/09/2020	Updated for last review issues	Ofer Biran (IBM)
0.8	02/09/2020	Quality Check performed and addressed comments.	Argyro Mavrogiorgou (UPRC), Ofer Biran (IBM)

Author List

Organisation	Name
IBM	Ofer Biran, Oshrit Feder
LXS	Sandra Ebro, Alejandro Ramiro
ATOS	María Ángeles Sanguino, Jorge Montero
UPRC	Argyro Mavrogiorgou, Thanos Kiourtis, George Manias, Nikitas Sgouros
OKS	Kostas Nasias

Abbreviations and Acronyms

Abbreviation/Acronym	Definition
API	Application Programming Interface
EC	European Commission
EOSC	European Open Science Cloud
GTD	Global Terrorism Database
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
PDT	Policy Development Toolkit
POS	Part-of-Speech
REST	Representational State Transfer
SKA	Situational Knowledge Acquisition

Contents

List of Figures

Executive Summary

This document is the first deliverable of WP4. It provides the architecture of the Data Acquisition and Analytics Layer, the included data analytic and transformation tools, and the cloud platform and software tools that are planned at this point to be used for the implementation of the PolicyCLOUD platform towards the demostration in D4.2 - Reusable Model & Analytical Tools: Software Prototype 1.

The Data Acquisition and Analytics Layer is responsible for:

- The registration of data sources of various kinds and properties
- The ingest processing of the data sources
- The registration of data analytic and transformation tools
- The built-in analytics tools for Cleaning and Data Interoperability, Situational Knowledge, Opinion Mining & Sentiment Analysis, Social Dynamics & Behavioral Data analysis
- The data storage for hot and cold big data and fusion process from hot to cold storage with enablement of seamless analytic on both

In essence, the Data Acquisition and Analytics Layer provides an extensible framework for data source and analytic tools, controlling the full data pass from the data sources through filtering, transformation and initial analytic, to hot storage and then to cold storage while enabling deeper analytics by the registered analytic tools on the ingested data at any time.

1 Introduction

1.1 WP4 Objectives

The major objectives of WP4 as stated in the PolicyCLOUD project proposal and Grant agreement are:

- Provide the framework for data fusion and aggregation – for different data sources types
- Provide data cleaning ensuring quality of information, sources reliability assessment, reliability-based selection of information sources
- Sentiment analysis techniques for policy assessment
- Analyze the social and behavioral data and requirements provided by social science experts for data selection in a given case
- Decoupling the analytical models and tools from the underlying infrastructure and datastores, assuring their reusability.

To fulfill these objectives, WP4 will provide an extensible framework for applying various analytics on various data sources, through the development of the Data Acquisition and Analytics Layer and the basic analytic tools that will be pre-integrated in it.

1.2 WP4 Positioning and Relations to other Work Packages

WP4 is responsible for the Data Acquisition and Analytics Layer, which is the central layer of PolicyCLOUD, between the cloud infrastructure and Policy layers. This layer provides the functionality for data ingestion from various sources while applying filtering and initial analytics, preparing it for deeper analytics on longer term storage (DB, object storage).

From the aspect of Work Packages partitioning, this layer is under the responsibility of WP4 (Reusable Models & Analytical Tools) and its tasks, with a strong relation to Task 3.3 (Cloud Gateways) and Task 3.6 (Data Governance Model, Protection and Privacy Enforcement) of WP3. In Figure 1 we show the conceptual model from the work packages partitioning point of view and the WP4 interfaces to WP3 below and WP5 above, as provided in the Grant Agreement document.

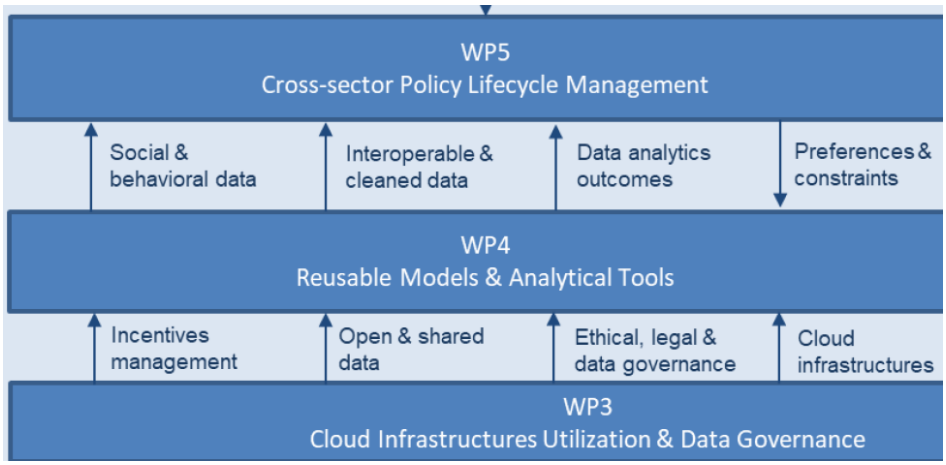

FIGURE 1 - WP4 INTERFACE WITH WP3 AND WP5

2 Data Acquisition and Analytics Layer

2.1 General Architecture and Roles

The Data Acquisition and Analytics layer is designed for extensibility and reusability of analytic functions. New analytics functions (services) can be registered into PolicyCLOUD and reused for applying analytics on new and existing registered data sources. The overall architecture, implementation of the layer's infrastructure (functions platform, registration schemes, layer API) and integration is provided by Task T4.1 - Cross-sector Data Fusion Linking.

The decided design alternative is a registration as serverless functions that are activated on demand, either by a direct PolicyCLOUD user request or by event/rule. There are two types of functions:

1. Ingest analytics / transformation function, which will be used to apply initial analytic and/or transformation on the data fusion path of data sources
2. Rest data analytic function which will be activated upon PolicyCLOUD user action invoked through the Policy Layer, to perform an analysis on specified data source (which was already ingested) to provide analytic results for policy decisions.

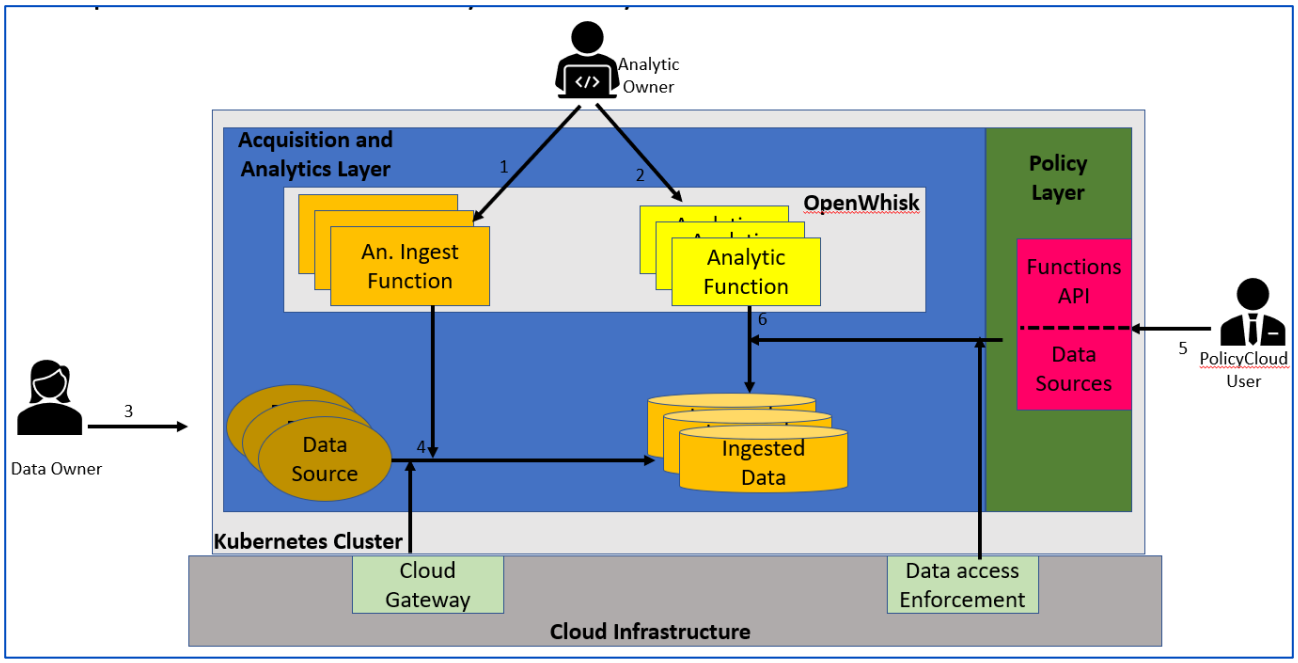

FIGURE 2 - DATA ACQUISITION AND ANALYTIC LAYER

According the numbers on Figure 2:

1. The Analytic Owner is responsible for the functionality of the Analytic Ingest Function which he registers into the PolicyCLOUD platform (and specifically into the Data Acquisition and Analytics layer). On Analytic Ingest Function registration the following parameters should be provided:
    a. API - parameters for data transformation
    b. API – parameters for initial analytic
    c. Expected data format / schema

A PolicyCLOUD administration privilege is required for a registration. Internally, the function will be deployed as OpenWhisk[1] function. At later phase of the project, a registration tool will be developed to make the registration simpler as well as to enforce a scheme of common parameters. The Enhanced Interoperability & Data Cleaning (T4.2) will be a built-in Analytic Ingest Function in PolicyCLOUD.

2. Similar Analytic Owner responsibility. On Analytic Function registration the following parameters should be provided:
    a. API - parameters for data transformation
    b. Category (e.g. the Heatmap function category developed for Software Prototype 1 (D4.2) is a category of Situational Knowledge Acquisition & Analysis)
    c. Expected data format / schema
    d. Purpose definition (for data access enforcement)

3. The Data Owner is responsible for the registration of a Data Source into the PolicyCLOUD platform. There are three types of data sources:
    a. **Streaming** – the data will be continuously streamed from external source into PolicyCLOUD data store, a cloud gateway (T3.3) connector and streaming details should be specified, as well as Analytic-ingest function(s) + parameters that should be applied on the streamed data – for filtering/cleaning purposes, for retrieval of initial analytics (that can be used for instance policy alerts) and for data transformation – keeping in the PolicyCLOUD data store only extracted knowledge, as oppose to the whole raw data.
    b. **Ingest-now** – an external (or local) data that should be ingest into the PolicyCLOUD data store, specify Analytic-ingest function(s) + parameters that should be applied on the ingested data – for filtering/cleaning purposes and for data transformation – keeping in the PolicyCLOUD data store only extracted knowledge, as oppose to the whole raw data.
    c. **External** – an external data source that can be queried / analyse upon demand.

    On a Data Source registration, the following parameters should be provided:
    a. Title
    b. Type (Streaming / Ingest-now / External)
    c. Source specification (streaming details, source data location, source URL, access method, credentials etc.)
    d. Schema / metadata
    e. Permissions - by whom (public / specified user list) and for what purpose the data can be used for (match the purpose definition on the Analytic Function above)
    f. Analytic-ingest function + parameters
    g. Trigger + rule

4. The data of a registered data source will be ingested into PolicyCLOUD according to the data source type: for Streaming type the registered Analytic-ingest function(s) will be triggered whenever a new data is available, for Ingest-now type the data will be ingested upon registration, processed by the registered Analytic-ingest Function(s), and for External type the data will be read (and possibly processed by registered Analytic-ingest function(s)) upon direct user request for analytic.

[1] https://openwhisk.apache.org/

5. A PolicyCLOUD user will get on the platform's dashboard the list of registered data sources, and the list of registered Analytic Functions, and can request to apply an Analytic Function on a Data Source. Access validation will be performed according to the user credentials the permissions registered for the data source and the purpose registered for the Analytic Function.

6. The Policy Layer will interact with the Data Acquisition and Analytics layer to retrieve the registered Data Sources and Analytic Functions, and for applying a selected analytic on a selected Data Source.

2.2 Extensibility and Reusability

2.2.1 Data Source Registration

PolicyCLOUD administration privilege is required for a data source registration. Internally, the data source details will be saved in the PolicyCLOUD store, a unique identifier will be generated for the internal representation, and the following for applying the required ingest processing, according to the data source type:

a. **Streaming** – An OpenWhisk rule and event will be deployed for activation of the specified Analytic-ingest functions whenever data is available (e.g. on a Kafka topic used as a connector). The destination data set is determined by the unique identifier generated at registration time.
b. **Ingest-now** - The data will be ingested into the destination data set, processed by the specified Analytic-ingest function(s).
c. **External** – an Analytic-ingest function can be specified also for External type, in this case whenever a read is performed from the external source (driven by a PolicyCLOUD user requests for applying analytic), the data will be processed by the Analytic-ingest function before being passed to the regular Analytic function. Additional data pre-processing is being pushed down to the external datastore provider and the PolicyCLOUD platform only retrieve the result of this pre-processing, thus never having complete access to the data set itself.

The Data Acquisition and Analytics layer will provide an API to list all the registered Data Sources with their registration parameters, to be exploited by the Policy Layer.

2.2.2 Analytic Function Registration

A PolicyCLOUD administration privilege is required for a registration of Analytic Ingest function or Analytic function. Internally, the function will be deployed as OpenWhisk function. At later phase of the project, a registration tool will be developed to make the registration simpler as well as to enforce a scheme of common parameters. The Enhanced Interoperability & Data Cleaning (Task T4.2) will be a built-in Analytic-Ingest Function in PolicyCLOUD, and the Situational Knowledge Acquisition & Analytics (Task T4.3), Opinion Mining & Sentiment Analysis (T4.4.) and Social Dynamics & Behavioral Data Analytics (Task T4.5) are built-in Analytic Functions in PolicyCLOUD.

The registration scheme of analytic function might be based on the results from the project CrowdHEALTH[2] - Figure 3 shows an example of the attached registration information:

[2] https://www.crowdhealth.eu/

```
{
  "configuration": null,
  "dateCreated": "2020-06-23, Thu, 00:00:00 CEST",
  "description": "The implementation of the exploratory data at-rest analysis for a specific dataset
  "outputGraph": "heat map",
  "endpoint": "http://www.example.com/analytical-tool/aggregationDataAnalysis",
  "uuid": "a3d06b6c-c253-4cf0-8455-911dbb425446",
  "serviceType": {
    "dateCreated": "2020-06-23, Thu, 00:00:00 CEST",
    "segmentTypes": [],
    "description": "Provided by ATOS",
    "name": "Exploratory Dataset Analysis Tool",
    "id": 3
  },
  "name": "Exploratory Dataset Analysis Tool",
  "id": 3,
  "parameters": [
    {
      "valueContraints": [
        {
          "value": [],
          "contraintType": "DEFAULT"
        }
      ],
      "unit": null,
      "valueType": "ENUM",
      "name": "fields",
      "description": "field to be used for the aggregation",
      "id": "1888df82-843a-406a-b78c-e1cddb68cc41",
      "evaluationTypesAllowed": [
        "RANGE"
      ]
    },
```


FIGURE 3 - EXAMPLE OF A FUNCTION REGISTRATION

2.2.1 Analytic Function Invocation

The Data Acquisition and Analytics layer will provide an API to list all the registered Analytic Functions with their registration parameters, enable it to invoke a selected function on a selected Data Source. A common scheme will be provided for Analytic Function invocation. The invocation parameters will include at least the following:

- Analytic Functions ID.
- The input parameters used for the analysis as defined at registration time.
- Datasets Information: Dataset ID, table, etc.
- Output: the data result from the analysis and optionally the type of requested visualization.

3 Data Fusion with Processing and Initial Analytics

The data ingested into the PolicyCLOUD data store from the various Data Source types is being processed / transformed for the following reasons:

- Cleaning for privacy/security issues or for improved accuracy.
- Filtering – to get only relevant data.
- Maintaining only derived insights (by initial analytic) that might be needed for further analytic – as oppose to store the entire data.
- Format transformation – saving the ingested data according to a common format which is expected by the Analytic Functions for applying analytics upon later user requests.
- Immediate alerts – this aspect is relevant more to a Streaming type Data Source. Alerts capability will be provided, in which Analytic-Ingest Function can trigger immediate alert upon processing of new streamed data.

Figure 4 provides an example for initial analytics on a Streaming Data Source scenario:

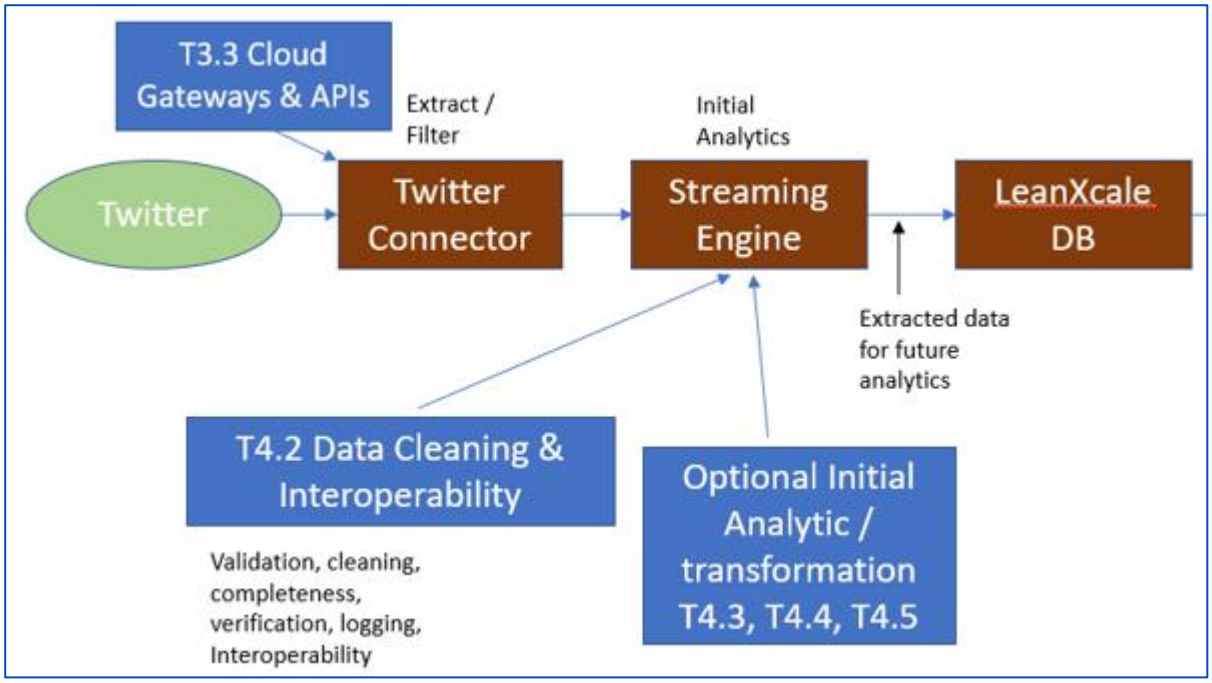

FIGURE 4 - THE STREAMING DATA PATH

In this example, a social network Data Source (Twitter) was registered, and a Cloud Gateway connector (provided by Task T3.3) with the specified filtering ingest data in streaming fashion. Alternative options were considered for the Streaming Engine (as Spark Streaming), and at least for Software Prototype 1 we'll use a solution based on Kafka sub/pub framework connected to OpenWhisk eventing. The Data Cleaning & Interoperability Analytic-Ingest Function is invoked whenever new data is ready and apply the specified cleaning and transformation. Optionally additional Analytic-Ingest Functions are applied before the transformed data is written to the PolicyCLOUD data repository for further deeper analytics. Optionally alerting can be specifying for cases when the initial analysis identifies urgent situation.

3.1 Cloud Gateways (Task T3.3)

The Cloud Gateway and API component will offer unified gateway capabilities to move streaming and batch data from data owners into PolicyCLOUD, supporting both SQL and NoSQL data stores and public and private data. It will act as the only entryway into PolicyCLOUD project allowing multiple functions and microservices to act cohesively and provide a uniform, gratifying experience to each stakeholder. The provided Gateway API will allow building scalable and robust APIs, while simplifying the interaction and data collection from various sources and providers. The main goal of this component is to handle a request by invoking multiple microservices and aggregating the results. It will add dynamic routing parameters and develop custom authorizations logic. PolicyCLOUD's Cloud Gateway and API component will support scalability, high availability and shared state without compromising performance. Moreover, it will support client-side load balancing, so that the overall system can apply complex balancing strategies, caching and batching, coupled with capabilities of fault tolerance, service discovery and handle multiple protocols. On top of this, several mechanisms and microservices will be utilized in order to check and evaluate the reliability of the data provided, by performing filtering of the obtained datasets before providing them to the PolicyCLOUD's internal services and components. Following the Gateway Pattern, it exposes a set of microservices to users via a unified API instead of providing an API for each in included. More details regarding the Cloud Gateways component are documented and can be found in D3.1 [3].

3.2 Enhanced Interoperability & Data Cleaning (Task T4.2)

The Enhanced Interoperability and Data Cleaning component consists of two (2) different sub-components, the Enhanced Interoperability and the Data Cleaning, being responsible for (i) enhancing the data interoperability of all the acquired data based on data-driven design, coupled with linked data and Natural Language Processing (NLP) technologies, and (ii) offering all the appropriate algorithms and techniques for detecting and correcting (or removing) corrupt or inaccurate records from all the collected data, correspondingly. These two sub-components are highly coupled with each other, since all the data that is cleaned by the Data Cleaning are immediately sent into the Enhanced Interoperability sub-component for further preprocessing and utilization, extracting semantic knowledge and high-quality information from all this cleaned data. However, this interaction is bidirectional, since as soon as the Enhanced Interoperability receives the cleaned data by the Data Cleaner and furtherly process it, it sends back to the Data Cleaner all the created data models that it has used for making the data interoperable, so as for the latter to consider them for creating more targeted and effective cleaning rules. The interconnection of these sub-components is depicted in Figure 5.

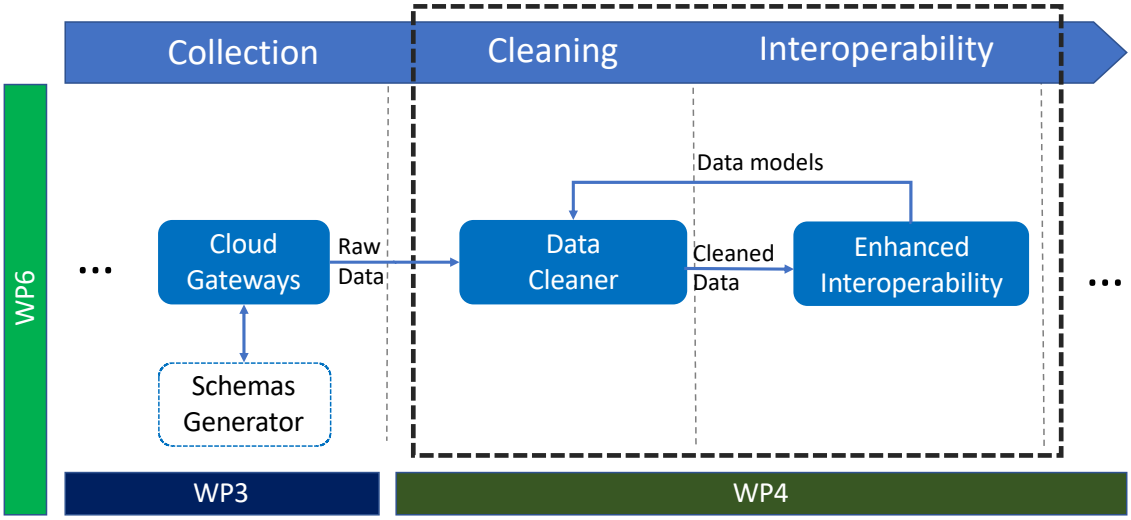

FIGURE 5 - SUB-COMPONENTS POSITIONING

3.2.1 Data Cleaning

The scope of the Data Cleaning sub-component is to deliver the software implementation that will provide the assurance that the provided data coming from several heterogeneous data sources will be clean and complete, to the extent possible. The Data Cleaning sub-component aims to provide all the processes that will detect and correct (or remove) inaccurate or corrupted datasets containing incomplete, incorrect, inaccurate or irrelevant data elements with the purpose of replacing, modifying or deleting these data elements, also known as "dirty" data. Towards this direction, the main goals of this sub-component are to (i) safeguard the level of confidence of the incoming data, (ii) investigate and develop mechanisms in order to ensure information non-repudiation, (iii) investigate and develop mechanisms that ensure information provision when needed, and (iv) design and develop prediction mechanisms that can be used to extrapolate (or interpolate) data in case of temporary disconnected operation of sources.

In order to achieve all the aforementioned, the Data Cleaning sub-component will implement the data cleaning workflow providing all the necessary actions towards the aim of consistent datasets across the PolicyCLOUD platform, as depicted in Figure 6:

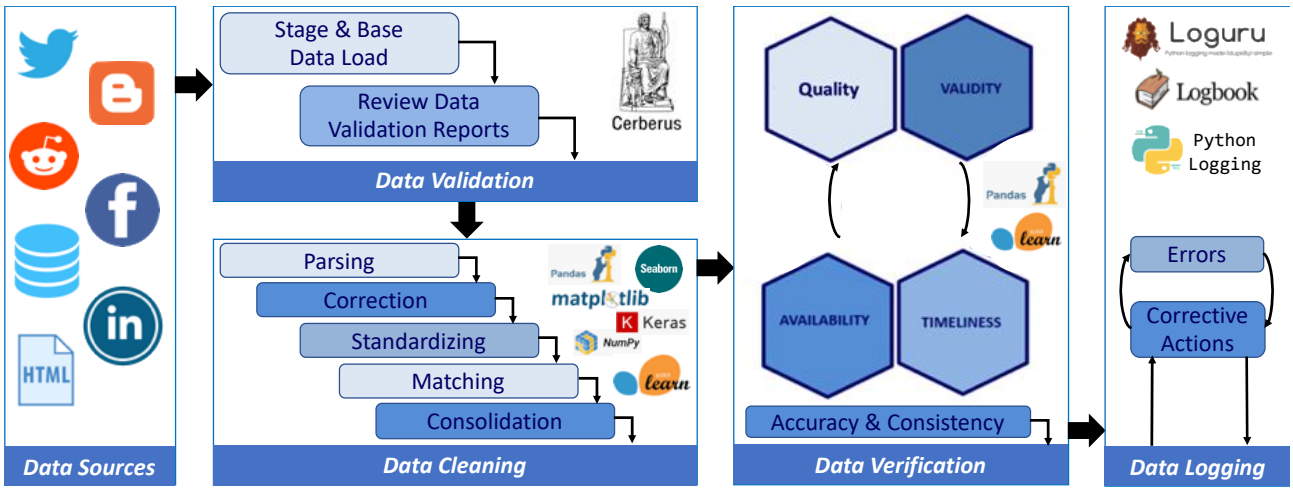

FIGURE 6 - DATA CLEANING WORKFLOW

In deeper detail, the data cleaning workflow comprises of four (4) discrete steps, each one of them being provided as an individual service.

- **Data Validation:** Ensures that the rest of the PolicyCLOUD components will operate on clean, correct and useful data. Hence, errors are reported for all the data elements that do not comply on the specified rules. As a result, the Data Validation service will perform data validation of the incoming information data with the purpose of identifying errors associated with the conformance to specific set of constraints. In order to successfully complete this process, the Data Validation service exploits the Cerberus library[3] for implementing all the necessary data validation functionalities, as well as for constructing the needed validation rules. More detail regarding the identified data constraints and rules are described in Section 3.2.1.1. It should be mentioned that as illustrated in Figure 6, the Data Validation service is able to process different kinds of incoming data, referring to (i) streaming data that comes from the Twitter, Facebook, Instagram, LinkedIn, and Reddit, and (ii) stored data that comes from Webpages, Blogs, signals and local Data Stores.

- **Data Cleaning:** Corrects or removes all the data elements for which validation errors were raised, considering missing, irregular, unnecessary, and inconsistent data. Thus, the Data Cleaning service will perform the necessary corrections or removals of errors identified by the Data Validation Service, and depending on the nature of the error, automated cleaning of the information will be performed based on a predefined set of rules. Hence, this service safeguards conformance to mandatory fields and required attributes of the dataset based again on a predefined set of rules or the desired configuration, ensuring the appropriateness and completeness of the incoming data. To successfully perform this process, the Data Cleaning service exploits the libraries of Pandas[4], Matplotlib[5], Keras[6], NumPy[7], Scikit-learn[8], and Seaborn [5] for implementing all the necessary data cleaning functionalities, based on the identified constraints and rules. More detail regarding the identified cleaning actions are described in Section 3.2.1.2.

- **Data Verification:** Checks the data elements of a dataset for accuracy and inconsistencies after the steps of data validation and cleaning are performed. Consequently, the Data Verification service will ensure that all the corrective actions performed by the Data Cleaning service will be executed in compliance to the data models' design of the PolicyCLOUD platform. This service ensures that the data will accurately be corrected or completed and the dataset will eventually be error free, exploiting the libraries of Pandas and Scikit-learn.

- **Data Logging:** Provides history records of the errors identified and the executed corrective actions. Thus, the Data Logging Service will undertake the responsibility of keeping in records the reported error logs from the rest of the services and the actions taken towards the data cleaning of the incoming data. Due to the fuzzy nature of the data cleaning workflow it is mandatory that all error events thrown or actions performed during the processing of the incoming data are held in specific records. The

[3] https://docs.python-cerberus.org/en/stable/
[4] https://pandas.pydata.org/
[5] https://matplotlib.org/
[6] https://keras.io
[7] https://numpy.org
[8] https://scikit-learn.org/stable

PolicyCLOUD moderator (should such a user role eventually be supported) should be able to check over the log entries created, the results of the data cleaning workflow along with detailed information on the error or the action that occurred. In order to successfully implement this process, the Data Logging service exploits the libraries of Loguru[9], Logbook [10], and Python logging for achieving easy and powerful debugging.

3.2.1.1 Data Constraints & Rules

Each acquired dataset of the PolicyCLOUD platform, includes diverse entities, relationships and cardinalities that are captured in the corresponding dataset's data schemas. Based on these characteristics, the corresponding data constraints (i.e. validation rules) are defined. The following list includes the data constraints that are mandatory to be fulfilled and the constraints that are optional. It should be noted that all of these constraints are taken into consideration by the Data Validation service in order to implement the various data validity checks with the aim to provide guarantees for the accuracy and the consistency of the incoming data, ensuring the conformance to specific set of constraints.

- **Mandatory Constraints** that must be fulfilled for each unique attribute:
  - Specific data type (e.g. Numeric, String)
  - Mandatory field
  - Specific value length (e.g. maximum 20 digits)
- **Optional Constraints** that could be fulfilled for each unique attribute:
  - Specific coding standard
  - Value representation (e.g. text formatting "123-45-67" or "1234567" or "123 45 67")
  - Value uniformity (e.g. all times are provided in UTC, all weight values in KGs, etc.)
  - Value range constraints (minimum and maximum values)
  - Pre-defined values (e.g. values selected from a drop-down list)
  - Regular expression patterns - data that has a certain pattern in the way it is displayed, such as phone numbers)
  - Separation of values (e.g. complete address in free form field without any indication where street ends, and city begins)
  - Cross-field validity (e.g. the sum of the parts of data must equal to a whole)
  - Uniqueness - data that cannot be repeated and require unique values (e.g. social security numbers)
  - Logical Error (e.g. female individual with prostate cancer medications prescribed)

3.2.1.2 Cleaning Actions

For the different data constraints and rules that were described in the previous sub-Section (3.2.1.1), the list of the corresponding possible cleaning (corrective) actions are documented in the current sub-Section. The following list includes the cleaning actions that can be used or combined for the described constraints.

- Deletion of value that does not conform to a constraint by:
  - Drop whole entity
  - Drop specific attribute

[9] https://loguru.readthedocs.io/en/stable/api/logger.html
[10] https://logbook.readthedocs.io/en/stable/

- Replacement of value that does not conform to a constraint through:
  - Transformation of wrong data type value
  - Prediction of erroneous/missing value based on the set constraints and rules
  - Prediction of erroneous/missing value based on similar values in the past
  - Creation of a list of features with high percentage of similarity with the same value

3.2.2 Enhanced Interoperability

Mapping and creating interoperable data depend on a method of providing semantic and syntactic interoperability across diverse systems, data sources and datasets. To this end, PolicyCLOUD's Interoperability Component aims to enhance interoperability into PolicyCLOUD project based on data-driven design by the utilization of linked data technologies, such as JSON-LD[11], and standards-based ontologies and vocabularies, coupled with the use of powerful tasks from the domain of Natural Language Processing (NLP), in order to improve both semantic and syntactic interoperability of data and datasets. Through these coupled technologies and methods, the Interoperability Component seeks to provide a state-of-the-art approach to achieve interoperability in data-driven policy making domain. Semantic AI entitles this proposed hybrid approach and is a combination of commonly used semantic techniques coupled with the utilization of NLP tasks and methods. Likewise, the provided Interoperability Component seeks to extract semantic knowledge and good quality information from the cleaned data that will be the input to its system, as shown in the initial architecture of the overall project. This knowledge, shaped in a machine-readable way, will be used in next Tasks for Big Data analytics, Opinion Mining, Sentiment Analysis etc. One of the preliminary steps of this component is to identify relevant, publicly available, and widely used classifications and vocabularies, such as the Core Person Vocabulary provided by DCAT Application Profile for Data Portals in Europe (DCAT-AP), that can be re-used to codify and populate the content of dimensions, attributes, and measures in the given datasets [6]. Hence, this component aims to adopt standard vocabularies and classifications early on, starting at the design phase of any new data collection, processing, or dissemination system. Using for example NLP techniques and tools like Text Classification, Named Entity Recognition (NER), Part-of-Speech Tagging (POS tagging) and even Machine Translation we can identify and classify same entities, their metadata and relationships from different datasets and sources and finally create cross-domain vocabularies in order to identify every new incoming entity [7]. Thus, by implementing a "JSON-LD context" to add semantic annotations to interoperability component's output [8], the system will be able to automatically integrate data from different sources by replacing the context-depended keys in the JSON output with URIs pointing to semantic vocabularies, that will be used to represent and link the data [9]. A standard such as this expresses information by connecting data piece by piece and link by link, allows for any resource (authors, books, publishers, places, reports, people, municipalities, articles, search queries) to be identified, disambiguated and meaningfully interlinked.

3.2.2.1 Linked Data and Semantic Web technologies

Linked Data are designed to support heterogeneous description models, which is necessary to handle divergent and heterogenous data and datasets. The Web of Linked Data is characterized by linking structured data from different sources using equivalence statements. To this end, Linked Data provide a perspective for a different kind of interoperability, based on Web architecture principles. Linked Data interoperability is designed to support heterogeneous description models, which is necessary to handle the very different data from

[11] http://json-ld.org

heterogenous sources. A solution to this gives four key requirements for data to be considered as linked data [10]:

- Use URIs to identify all types of data items, for example, considering a dataset of technical reports, a unique URI will be used for each report.
- Make the URIs accessible globally by using HTTP URIs. Though this, users can look up the identifiers. The latter, will help users of the PolicyCLOUD project to search policies and information needed through the core search engine and catalog of the PolicyCLOUD project, the Data Marketplace, where based on the metadata and semantics attached to the datasets and the policies, the Marketplace APIs will enable the search and retrieval of appropriate information.
- If someone accessed one of the above mentioned URIs, provide useful structured information in RDF.
- Provide links among different data items by including RDF links that point to other URIs. To this end, the discovery of related information will be enhanced.

The W3C has proposed a number of standards to help in the creation of linked data, in particular, the Resource Description Framework (RDF), which provides the standard for linked data resources on the Web. In addition, a number of models have been proposed for detailing the semantics of data described in RDF, in particular, the RDF Schema Model (RDFs) [11], which allows for inductive reasoning on properties and the Web Ontology Language (OWL) [12], which allows for more sophisticated reasoning about data using description logic and providing a sophisticated and enhanced Ontology Mapping. Moreover, a more recent data schema model has also been introduced and recommended since 2014 to promote interoperability among JSON-based web services [8]. Many already deployed platforms or services use JSON as data format. On top of this, the utilization of JSON-LD technology in Enhanced Interoperability component will provide a standard way to add semantic context to the existing JSON data structure, for the purpose of enhancing the interoperability between APIs and across the whole PolicyCLOUD project and data lifecycle. To this end, the utilization of Linked Data and Semantic Web technologies in Enhanced Interoperability component seeks to deliver structured information, which can be used and queried by a flexible and an extensible way to get a better understanding of the provided data and datasets.

3.2.2.2 Natural Language Processing (NLP)

One of the main goals of Enhanced Interoperability component is to create a new interoperability framework for linked data. On top of this, the Enhanced Interoperability component needs to build strong links between datasets. These links rely on the recognition, extraction, and description of entities from the structured and unstructured data, such as persons, objects, concepts, places etc. To this vision, the utilization of NLP will integrate with Linked Data in order to enhance the PolicyCLOUD's interoperability component. Our main focus and goal through the use of NLP is to combine subtasks of this domain with Semantic Web and Linked Data technologies in order to provide a holistic and hybrid approach of achieving interoperability among the PolicyCLOUD project. To this end, the concepts and subtasks of Information Retrieval (IR) and Information Extraction (IE) are core tasks and techniques within the Interoperability Component. NLP through various methods and techniques will enhance the processing and analyzing of raw documents and texts, like Twitter texts, reports etc., hence in the cases that unstructured data need to be processed and analyzed. Semantic Web technologies coupled with NLP tools and techniques, such as NER, IE, POS tagging will enhance the ability of the Interoperability Component to extract entities, semantic relations and annotate raw data, hence to achieve higher semantics and interoperable data. For example, Text Classification, Information Extraction and even Machine Translation tasks will enhance the steps of classification and taxonomy that will be used during the Ontology Mapping phase. Moreover, Text Analysis and processing tasks from the domain of NLP can also be utilized under the scope of extracting required data and information through the raw texts in order to be analyzed, transformed and stored. On top of this, the Enhanced Interoperability component seeks to implement and enable easily constructed, flexible and high-performance NLP pipelines. To this end, the utilization of Linked Data technologies, analyzed in section 4.3.2.2 including JSON-LD files, coupled with NLP services will help to

define the types of the data that pass through services to deliver structured information that can be used and queried in a flexible and an extensible way and to get a better understanding of the data.

3.2.2.3 SEMANTIC ARTIFICIAL INTELLIGENCE (SEMANTIC AI)

Semantic AI addresses the need for interpretable and meaningful data, and it provides technologies to create this kind of data from the very beginning of a data lifecycle. Most machine learning algorithms work well either with text or with structured data, but those two types of data are rarely combined to serve as a whole. Semantic web is based on machine understandable languages (RDF, OWL, JSON-LD) and related protocols (SPARQL, Linked Data, etc.), while on the other hand NLP tasks and services focuses on understanding natural languages and raw texts. To this end, the combination of Semantic Web Technologies and NLP tasks will provide to the project a state-of-the-art approach for achieving semantic and syntactic interoperability and will enhance the ability of the project to combine structured and unstructured data in multiple ways. For example, the utilization of Named Entity Recognition (NER), one of the most widely used tasks of NLP, coupled with the utilization of text mining methods based on semantic knowledge graphs and related reasoning capabilities will enhance the interoperability and the final linking of divergent data and datasets. To this end, this integrated and hybrid approach will ultimately lead to a powerful and more enhanced Interoperability Component. On top of this, semantic data models can bridge this gap. Links and relations between business and data objects of all formats such as XML, relational data, CSV, and also unstructured text can be made available for further analysis. This allows us to link data even across heterogeneous data sources to provide data objects as training data sets which are composed of information from structured data and text at the same time. Linked data based on W3C Standards can serve as an enterprise-wide data platform and helps to provide training data for machine learning in a more cost-efficient way. Instead of generating data sets per application or use case, high-quality data can be extracted from a knowledge graph or a semantic data lake. Through this standards-based approach, also internal data and external data can be automatically linked and can be used as a rich data set for any machine learning task. Finally, the utilization of Semantic AI, in other words the combination of NLP and Semantic Web technologies, will provide the capability of dealing with a mixture of structured and unstructured data that is simply not possible using traditional, relational tools.

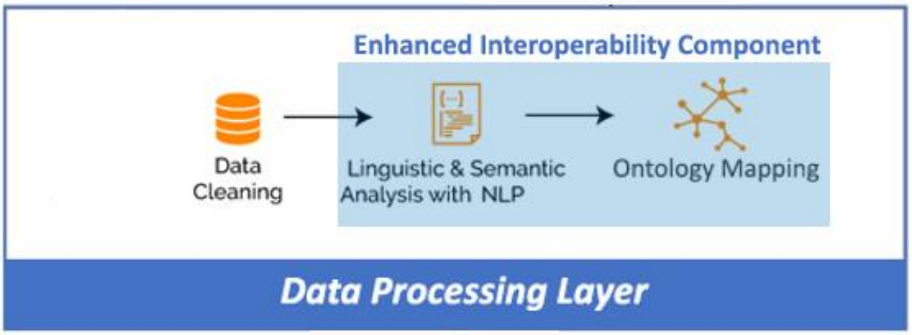

FIGURE 7 - DATA PROCESSING LAYER

As shown in Figure 7 after fetching cleaned data from the Data Cleaning Component, the Enhanced Interoperability Component seeks to implement the hybrid approach of Semantic AI through two core subtasks/phases, the Linguistic & Semantic Analysis with NLP (1st phase) and the Ontology Mapping (2nd phase). To exploit what the Semantic AI offers, data first needs to be structured and annotated. To this end, in the first phase cleaned data will be analyzed, transformed and annotated with appropriate metadata through the utilization of Semantic Web technologies and NLP tasks, such as Named Entity Recognition (NER), Part-of-Speech Tagging (POS Tagging) etc. In the next phase, semantic and syntactic annotated data will be interlinked through

the task of Ontology Mapping. Ontologies are key factors for enabling interoperability in the Semantic Web [13] and have an important role in annotating and organizing the vast wealth of experimental, clinical, and real-world data and their day-to-day usage [14].

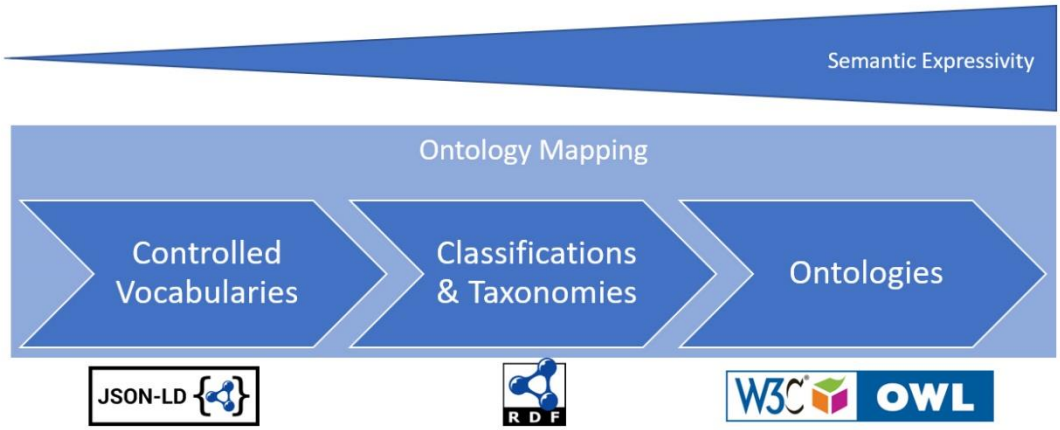

FIGURE 8 - ONTOLOGY MAPPING STEP

As shown in Figure 8 there are several levels of structuring before reaching proper ontologies. At the beginning, the annotation and creation of metadata representations through the utilization of JSON-LD technology is a key point. Afterwards, vocabularies and taxonomies expressed by RDFs are created and in the final step they are correlated and interlinked into ontologies with high semantic expressivity through the utilization of OWL technology.

On top of this, ontologies are central to the Semantic AI because they allow applications to agree on the terms that they use when communicating and furthermore they enable the correlation of divergent data and datasets from various sources. Based on this, raw and unstructured data derived from Twitter, as in Use Case 1 of the PolicyCLOUD project in the concepts of radicalization, can be correlated and integrated with structured data derived from a divergent source, such as Global Terrorism Database (GTD)[12]. To this end, the utilization of ontologies under the scope of Semantic AI facilitates communication by providing precise notions that can be used to compose messages (queries, statements) about the policy making domain. In stakeholders and user level, the ontology helps to understand messages by providing the correct interpretation context. Thus, ontologies, if shared among stakeholders, may improve system interoperability across ISs in different organizations and domains. Moreover, linked data can work as the foundation of a common export format for data in the final service portal of the project, PolicyCLOUD Marketplace. The overall approach that will be followed brings together techniques in modeling, computation linguistics, information retrieval and agent communication in order to provide a semi-automatic mapping method and a prototype mapping system that support the process of ontology mapping for the purpose of improving and enhancing interoperability during the whole data lifecycle in the PolicyCLOUD project.

[12] https://start.umd.edu/gtd/

4 Analytic on Data at Rest

Applying Analytic Function(s) on data from a registered Data Source is driven by a PolicyCLOUD user action. The Analytic Function will be activated as a serverless OpenWhisk function, directly triggered by the user request. The data access operation will be according to the Data Source type:

- **Streaming** – the operation is applied on the data that was already ingested to the PolicyCLOUD store at the time of the user request. The data might reside partially in the LeanXcale database and partially in the object storage, but this will be transparent to the Analytic Function.
- **Ingest-now** – At the time of the user request all the data was already ingested to the PolicyCLOUD store. The data might reside partially in the LeanXcale database and partially in the object storage, but this will be transparent to the Analytic Function.
- **External** – the Analytic Function will fetch the data through the PolicyCLOUD data repository and the Cloud Gateway connector associated with the Data Source upon the user request. The PolicyCLOUD data repository will open a data connection to the external data store and push down the query statement to be processed by the later, and will then transform the data to the common tabular format that the Analytic Function(s) are relied on through the standard JDBC connectivity, making the data access from external sources transparent to them. Ingest transformation by registered Analytic-Ingest Function(s) might be applied to the data before being provided to the Analytic Function.

Figure 9 continues the example for Streaming Data Source scenario for the regular analytic applied on the data that is already ingested (and cleaned, transformed etc.).

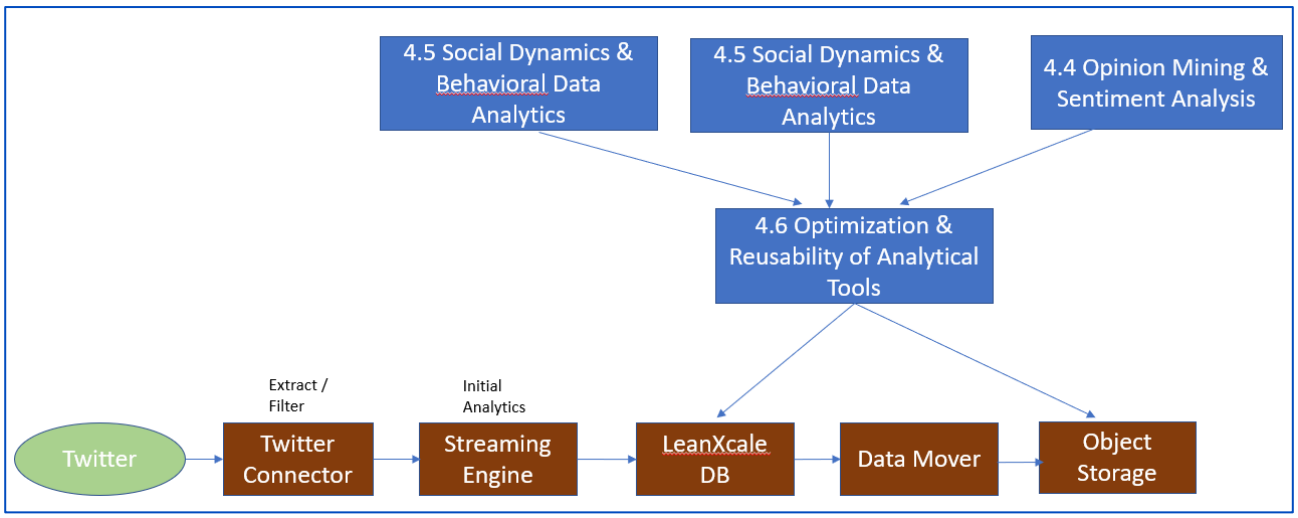

FIGURE 9 - SEAMLESS ANALYTICS ON INGESTED DATA

The Analytic Functions of Situational Knowledge Acquisition & Analysis (Task T4.3), Opinion Mining & Sentiment Analysis (Task T4.4) and Social Dynamics & Behavioral Data Analytics (Task T4.5) are built-in Analytic Functions in PolicyCLOUD, Task 4.6 (Optimization & Reusability of Analytical Tools) will apply further optimization in applying analytics and composition of functions – this will be implemented in the second the second project year and will be described in the next deliverable report (D4.3), similarly for the seamless analytics framework that enables transparent analytic on data in hot and colder data stores of the PolicyCLOUD data repository (LeanXcale data base and object storage).

4.1 Situational Knowledge Acquisition & Analysis (Task T4.3)

The main objective of the Situational Knowledge Acquisition (SKA) component will be derive high-level descriptions from real-world situations. This will be done by means of Machine Learning (ML) techniques and supported with use case-dependant models used for structuring the data.

This component, in a similar way as Opinion and Sentiment Analysis components, will be based on Apache NiFi[13] technology and containerized in a Docker image for its deployment. On the other hand, following the early instructions for reusability and extensibility, it will be provided along a REST interface for allowing the registration and invocation of analytical services in a standardized way in PolicyCLOUD.

4.1.1 Early functionality extracted from pilots

In the context of PolicyCLOUD this knowledge extraction will be materialize in several ways attending the needs elicited from the use case. Worth noting that all the functionalities delivered by the SKA component will have as end user the policy maker, as a result of that they will be described always under the policy maker perspective.

In terms of purpose, at the closure of this document, the main activities identified in the context of Task 4.3 are the following:

- Dataset categorization, the policy maker will be able to obtain text-based information classified into categories. The structure of the categories, how the knowledge is represented, and the categories themselves will be defined and provided by each particular use case. An example of that, for the use case 2, "*Intelligent policies for the development of denomination of origin*", the policy maker will be able to visualize categorized information for report generation based on categories provided by the Aragon Gov.
- Exploratory analysis, the policy maker will be able to visualize main features of the datasets of their interest. As an example of that, for the use case 1, "*Participatory policies against radicalization*", the policy maker will be able to explore the terrorisms attacks extracted from GTD by means of a heat map showing the incidence of attacks by cities and by groups.

4.1.2 SKA component

At this stage of the project, for the SKA component, main efforts have been done in terms of:

- elicitation of requirements, see D2.1 [1] for more details.
- integration with the existing analytical pipelines. See following sub-sections for more details.

4.1.2.1 DATASET CATEGORIZATION

For this first version of the specification, the dataset categorization will be focused on the social media analysis. Therefore, the categorization will be conducted through the streaming data path as Figure 10:

[13] https://nifi.apache.org/

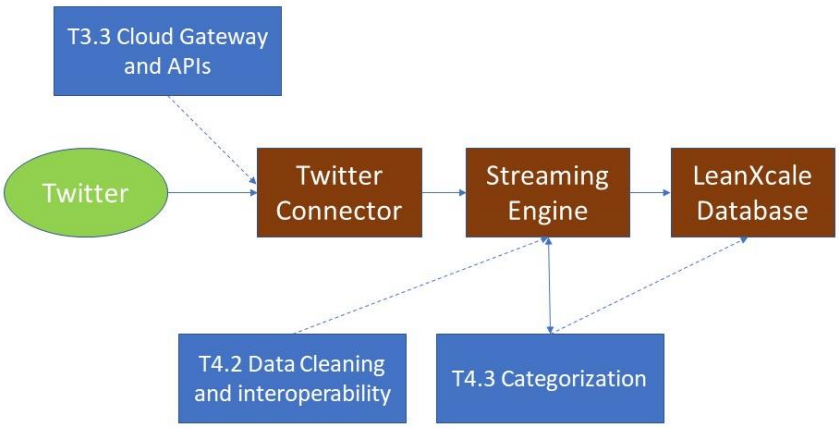

FIGURE 10 - DATASET CATEGORIZATION PIPELINE INTEGRATION

As it is showed in the picture above, data will be collected directly from the streaming engine, analysed and put it back again into the streaming pipeline for its later storage. The discontinue lines mean open issues under discussion (at this stage in the project) for example: what pre-processing steps (data cleaning and interoperability) will be need for the data to be categorized or if the categorization process itself will store the result into the PolicyCLOUD storage.

Finally, and looking inside the Categorization service, following characteristics can summarize the foundations for its development:

- The main objective is the classification of information into defined categories for reporting.
- For this first specification, the categorization will be applied to the text extracted from social networks, more specifically for Twitter.
- Clustering-based techniques will be studied for text classification.
- The categories will be provided by the use cases.
- It can be assigned several categories to the same tweet.

4.1.2.2 EXPLORATORY ANALYSIS

For the case of the exploratory analysis, the analysis will be performed over datasets previously treated and stored. Therefore, the analysis will be conducted through the data at-rest pipeline. See Figure 11 for a summary of it:

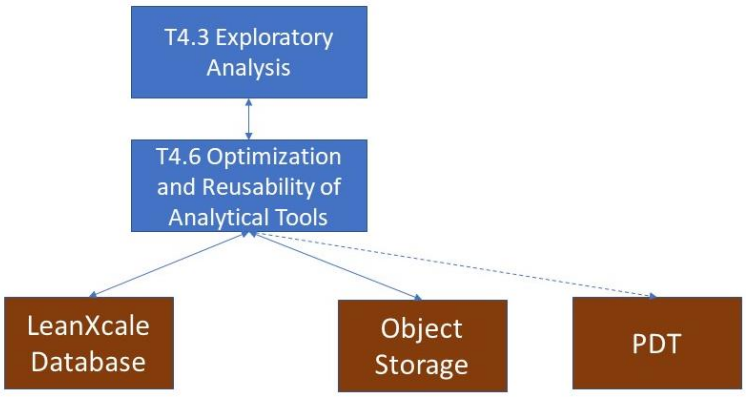

FIGURE 11 - EXPLORATORY ANALYSIS PIPELINE INTEGRATION

In that case, for the exploratory analysis, the investigation on the data will be performed with data extracted from the PolicyCLOUD data storage. Is still under discussion if the results will be stored into the PolicyCLOUD storage or they will be offered directly to the PDT since mostly they will be a data summary in the form of (or oriented to) graphical representation.

4.2 Opinion Mining & Sentiment Analysis (Task T4.4)

As described in D2.2 [2], the aim of these two components are focused on text analytics for extracting the sentiment of a piece of text, for categorizing any kind of text based on a set of keywords defined by the users, for extracting meaningful information from the text, and more.

These components (and the Situational Knowledge Acquisition component) will be developed using Apache NiFi and will be containerized in a Docker image following the instructions defined in precious sections to be, at the end, an Analytic Function. Inside the image, the service will have an API to be executable with different parameters in the same way as the Situational Knowledge Acquisition component described in the upper section.

4.2.1 Early functionality extracted from pilots

At this phase of the project, these components are not inside the scope of the initial prototype, but crucial advances have been done in collaboration with the pilots to obtain what they need from our analytical components and how they can benefit of them. In the following subsections are shown the output of the discussions with the pilots (at this stage with "UC#1: Participatory policies against radicalization" and "UC#2: Intelligent policies for the development of denomination of origin") along the analytics that are of interest for them. Moreover, all the requirements from the pilots and these two components are written in D2.1 [1].

Figure 12 and Figure 13 represents the desired analytics that the UC#1 aim to obtain after the discussions.

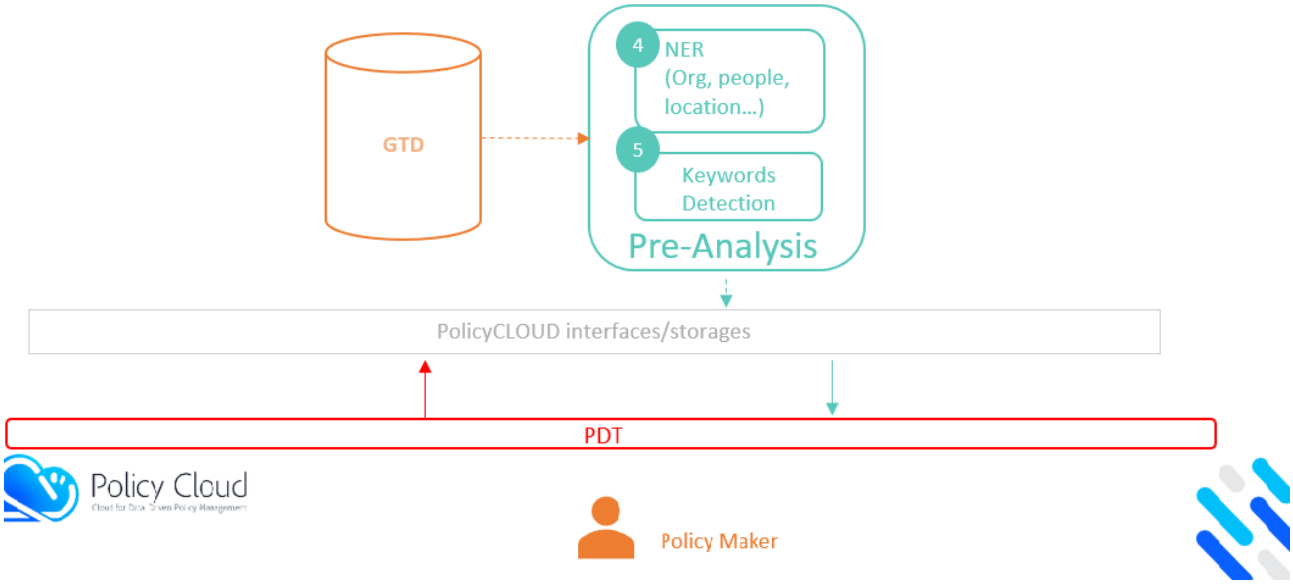

FIGURE 12 - DESIRED ANALYTICS OF DATABASES FOR UC#1

Figure 12 is based on analysing one of the databases that the pilot provides (Global Terrorism Database) and perform some text analytics on it to discover, for example, useful entities or crucial keywords to further investigate them on social media sources. Those are related to the Opinion Mining component.

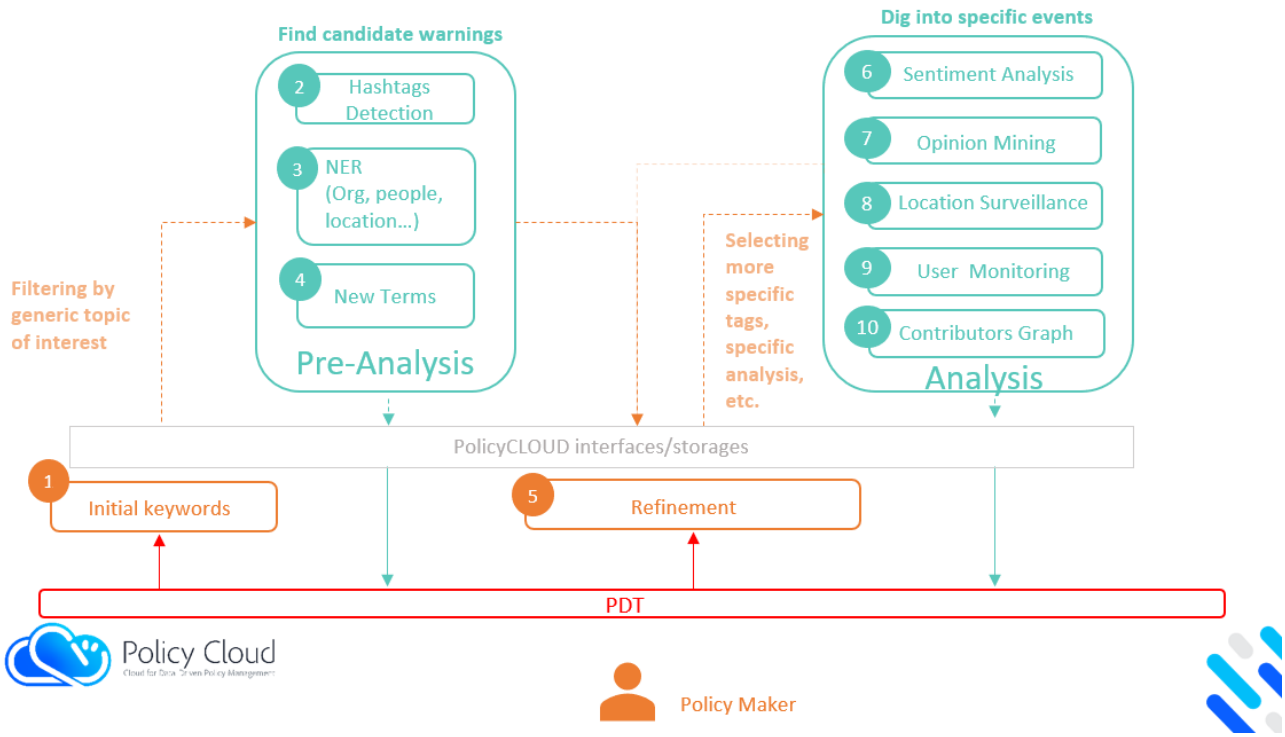

FIGURE 13 - DESIRED ANALYTICS OF SOCIAL MEDIA FOR UC#1

Figure 13 describes the analytics related to social media sources in two phases. First the Policy Maker defines a set of keywords (1) to have an initial analysis over all the data. Then, with the new hashtags (2), entities (3) or terms (4) extracted in the first phase, a second phase is executed to have concrete analysis (6...10) in the targeted data.

The output of the discussions with the UC#2 are presented in Figure 14 and Figure 15.

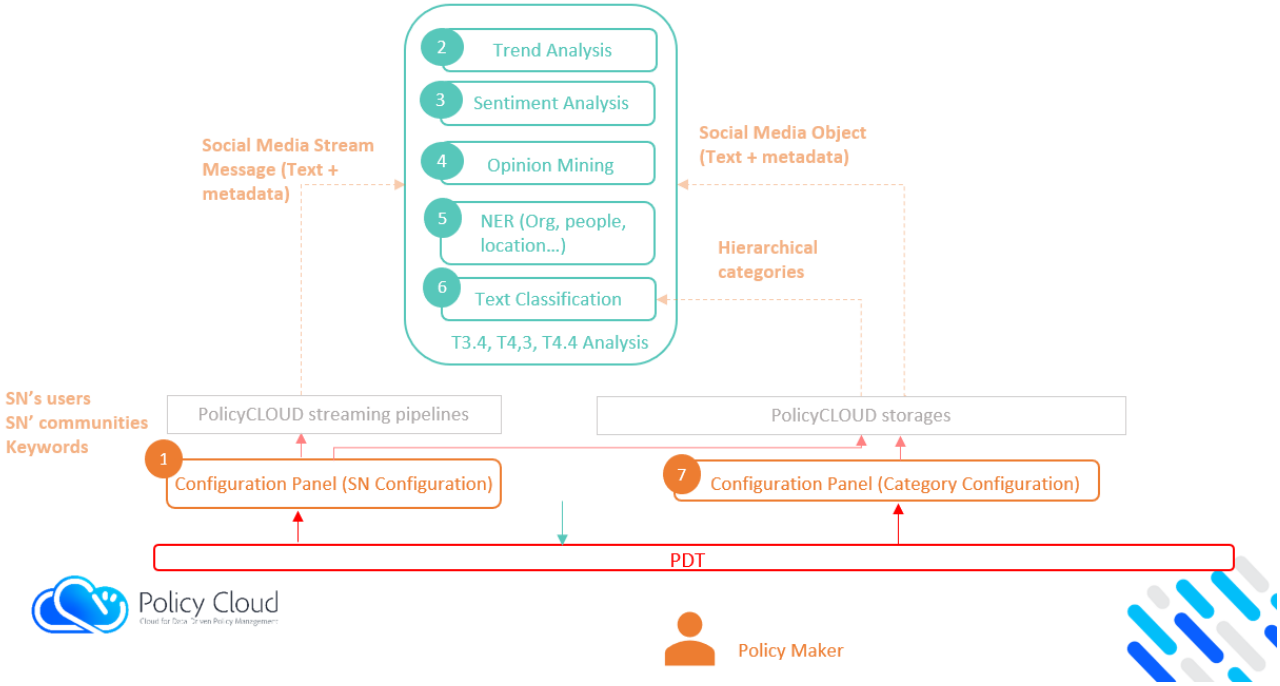

FIGURE 14 - DESIRED ANALYTICS OF SOCIAL MEDIA FOR UC#2

Figure 14 represents the analytics to be performed using social media sources (mainly Twitter) using a Configuration Panel (1) where the Policy Maker defines the keywords to search, and another optional panel (7) to define the categories they want to classify. The analytics (2...6) are related to the Opinion Mining and Sentiment Analysis components.

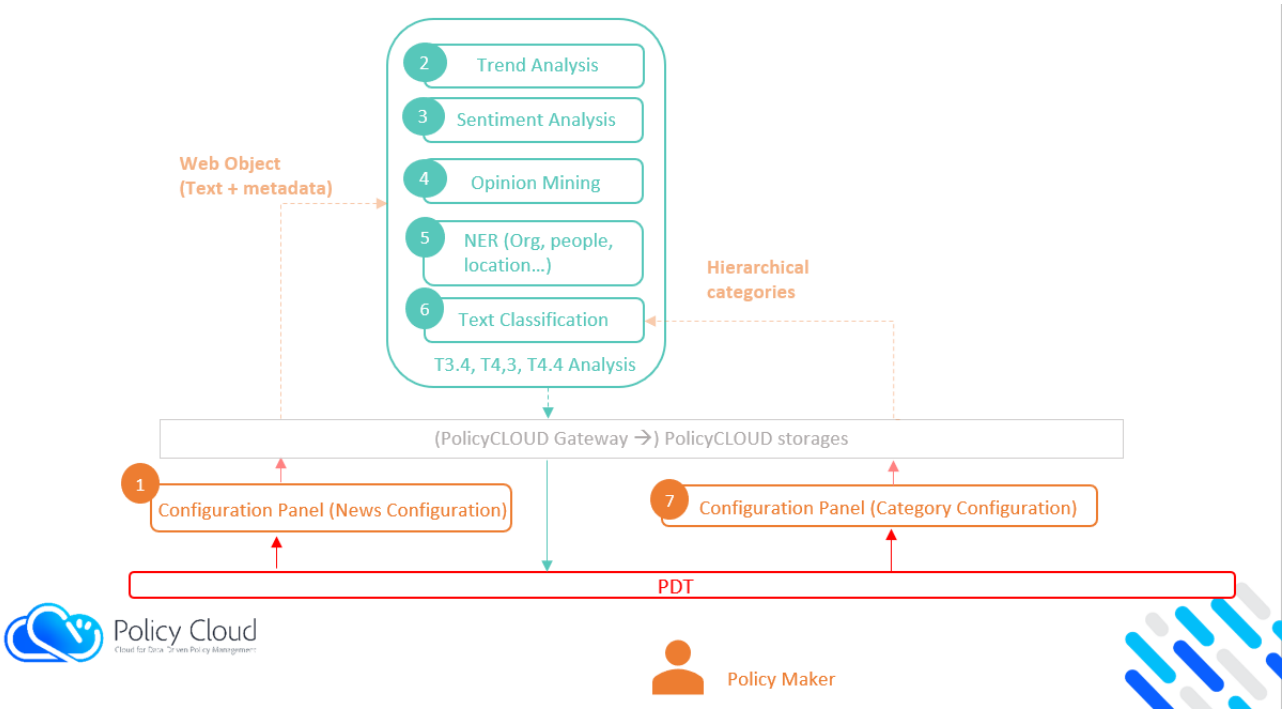

FIGURE 15 - DESIRED ANALYTICS OF RSS FEEDS FOR UC#2

Figure 15 is very similar to the social media analytics described before but using RSS feeds as input. The difference is in the first panel (1) where the Policy Maker defines what RSS feeds will be analysed with some optional keywords to filter them.

Most of the needs analysed at this stage are based on social media data, in particular, RSS feeds and Twitter. But this is not restricted to that, other kind of data from databases, opinions, and more, could be analysed. Next is described in more details how to create different workflows, or pipelines, using Apache NiFi to fulfil the needs presented by the pilots.

4.2.2 Opinion Mining component

Figure 16 describes a workflow for opinion mining (categorization and entity extraction) using RSS feeds as input data, in this case those are Spanish RSS feeds, but at the end the official language to be analyzed will be English. Those RSS feeds will be generating data every time and will be sent to Kafka. This part of the pipeline could be externalized to retrieve data from the PolicyCLOUD datastore in batch mode. In the middle we will have a preprocessor and analytic functions developed in Python using spaCy[14] to match the keywords defined by the user (by an ontology, a taxonomy, a set of words for each category, or others) and get potential useful entities or categorize the text correctly based on the requirements defined. Other solutions and interesting frameworks will be analyzed to perform all the analytics defined in the scope of the Opinion Mining component. The output will be sent again to Kafka to be used by other components or will be stored directly in the PolicyCLOUD datastore to later be visualized in the dashboard.

[14] https://spacy.io/

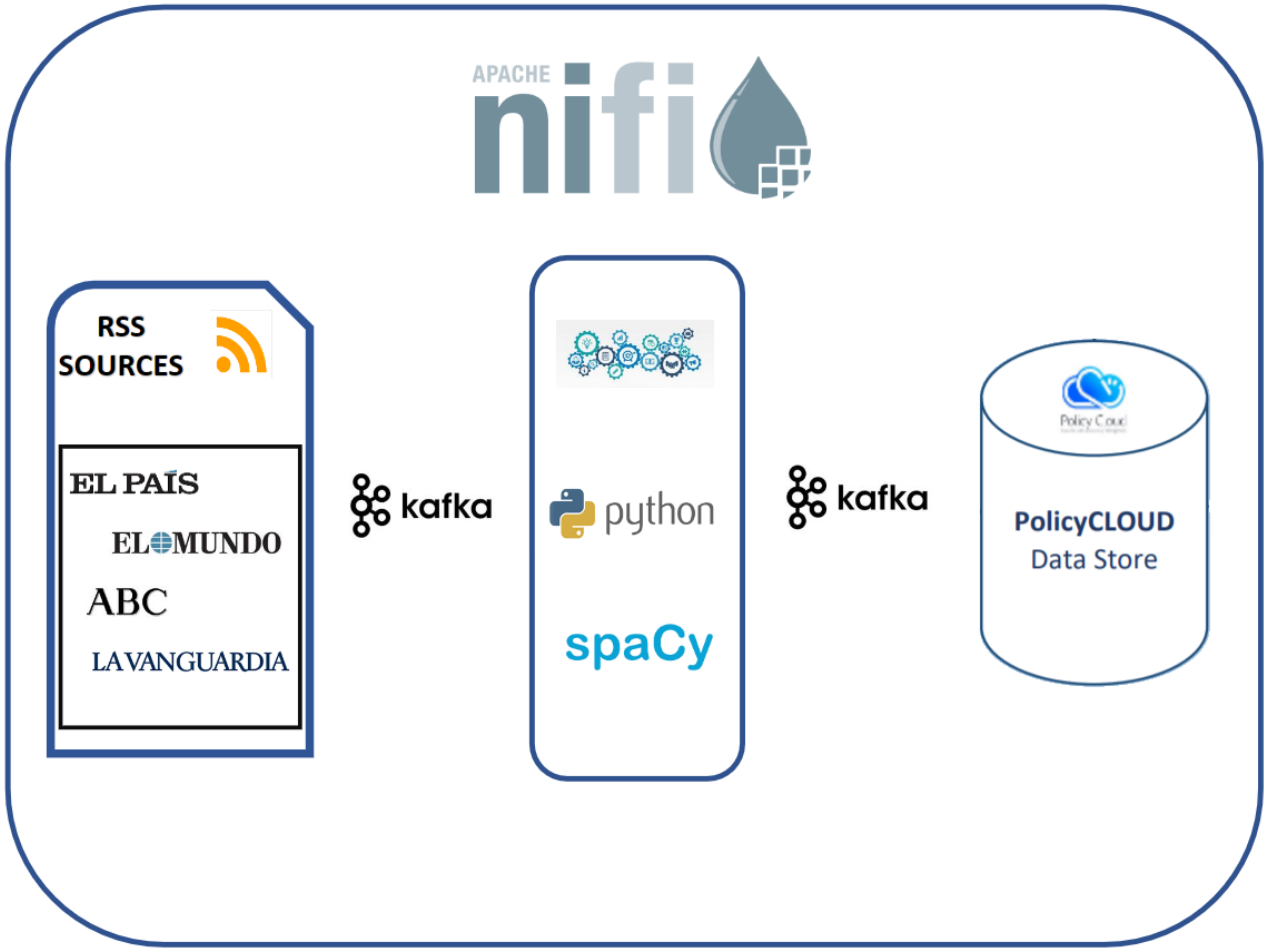

FIGURE 16 - EXAMPLE WORKFLOW FOR OPINION MINING COMPONENT USING APACHE NIFI

4.2.3 Sentiment Analysis component

On the other hand, going deeply in the Sentiment Analysis component, Figure 17 shows a workflow from the point of view of Apache NiFi components. "ConsumerKafka" and "PublishKafka" processors do the same as explained before with the Opinion Mining component, but it could be changed to a database processor to read from or write to the PolicyCLOUD datastore. In the middle is the "InvokeHTTP" processor that contains a Python application in the form of REST services and containerized in a Docker image that is called from the pipeline. This microservice allows us to change, update or add more functionalities without restarting the full workflow, just redeploying the new image.

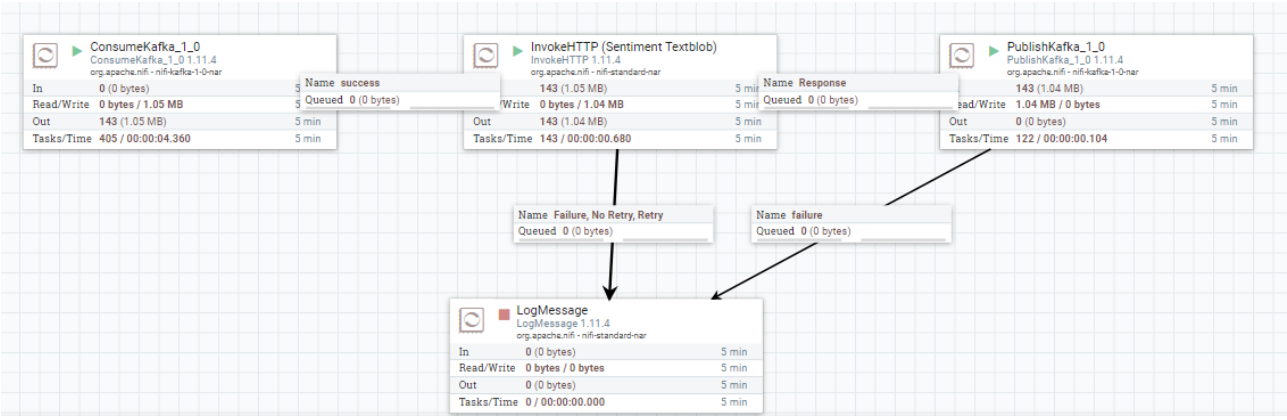

FIGURE 17 - EXAMPLE WORFLOW FOR SENTIMENT ANALYSIS COMPONENT USING APACHE NIFI

At this moment, two python libraries have been studied to generate the sentiment of a text. Both have an internal translation to English that may be a degradation of the accuracy for other languages and need to be optimized in next phases to minimize that.

Next is listed the two libraries with an example of the outputs they provided:

- **VADER** (Valence Aware Dictionary and sEntiment Reasoner)[15] [4]: this is a lexicon and rule-based tool to perform the sentiment, mainly in social media sources. The output of this library is based on three values for positive, negative and neutral sentiments, and other called "compound" that is the normalized value. "Compound" value is in the range [-1,1] going from the most negative sentiment to the most positive sentiment, respectively.
    - Example: {'neg': 0.026, 'neu': 0.492,'pos': 0.482,'compound': 0.9798}
- **TextBlob**[16]: it has natural language processing tasks, like spaCy, but in addition it performs the sentiment in the text. In this case, the output is defined by the "polarity", that is a value in the range [-1,1], it means that if the polarity is "-1" the sentiment is very negative, and if it is "1" the sentiment is very positive. TextBlob gives an additional output for subjectivity, this is very useful to know if the text is about personal opinions or generic texts.
    - Example: Sentiment(polarity=0.95, subjectivity=0.95)

4.2.4 Next steps

As commented before, these two components are in a good way and in close contact and discussions with the pilots to fulfil their needs, even without being included in the first prototype. Next steps will be focused in the integration with the other components of PolicyCLOUD to be included in the next prototype, continue working in covering all the requirements from the other pilots, and improving UC#1 and UC#2.

[15] https://github.com/cjhutto/vaderSentiment
[16] https://textblob.readthedocs.io/en/dev/quickstart.html

4.3 Social Dynamics & Behavioral Data Analytics (Task T4.5)

Politika, the PolicyCLOUD social dynamics component, provides an online environment for agent-based modeling and simulation able to simulate the social effects of various policies applied on a population of autonomous and interconnected agentes. Politika is under development using the Elixir/Phoenix/Postgres Ecosystem. It uses a server-side, Model-View-Controller (MVC) pattern that leverages the ability of functional programming environments to support big data processing in a distributed setup. Politika´s development is based on the Elixir/Erlang[17] programming environment because such an environment provides a well-tested infrastructure that supports the development of these types of apps on the web.

Figure 18 depicts the overall architecture of Politika. As this figure indicates the environment includes a user- and server-side system. The user-side system allows multiple users to concurrently interact with Politika through a Javascript web client interface. Using this interface they can:

- specify, edit or delete a simulation
- browse the specifications of simulations stored in the system
- execute simulations
- upload/download data to/from a simulation
- examine raw simulation results
- compute and visualize simulation analytics

All these operations serve multiple users concurrently except from simulation execution, which, due to possibly high computational load, runs one simulation at a time by default. In this case, all user requests for simulation execution are placed in wait mode during which they poll the simulator for availability with a fixed frequency. Such a user request starts executing automatically when polling succeeds in finding an available slot.

On the server side all user requests are processed by a web server based on the Phoenix web framework[18]. The Phoenix system interacts with three independent components consisting of the Social Dynamics simulator built in Elixir, the Analytics component also built in Elixir and the Database system that uses the Ecto[19]/Postgres environment.

[17] https://elixir-lang.org/
[18] https://www.phoenixframework.org/
[19] https://github.com/elixir-ecto/ecto

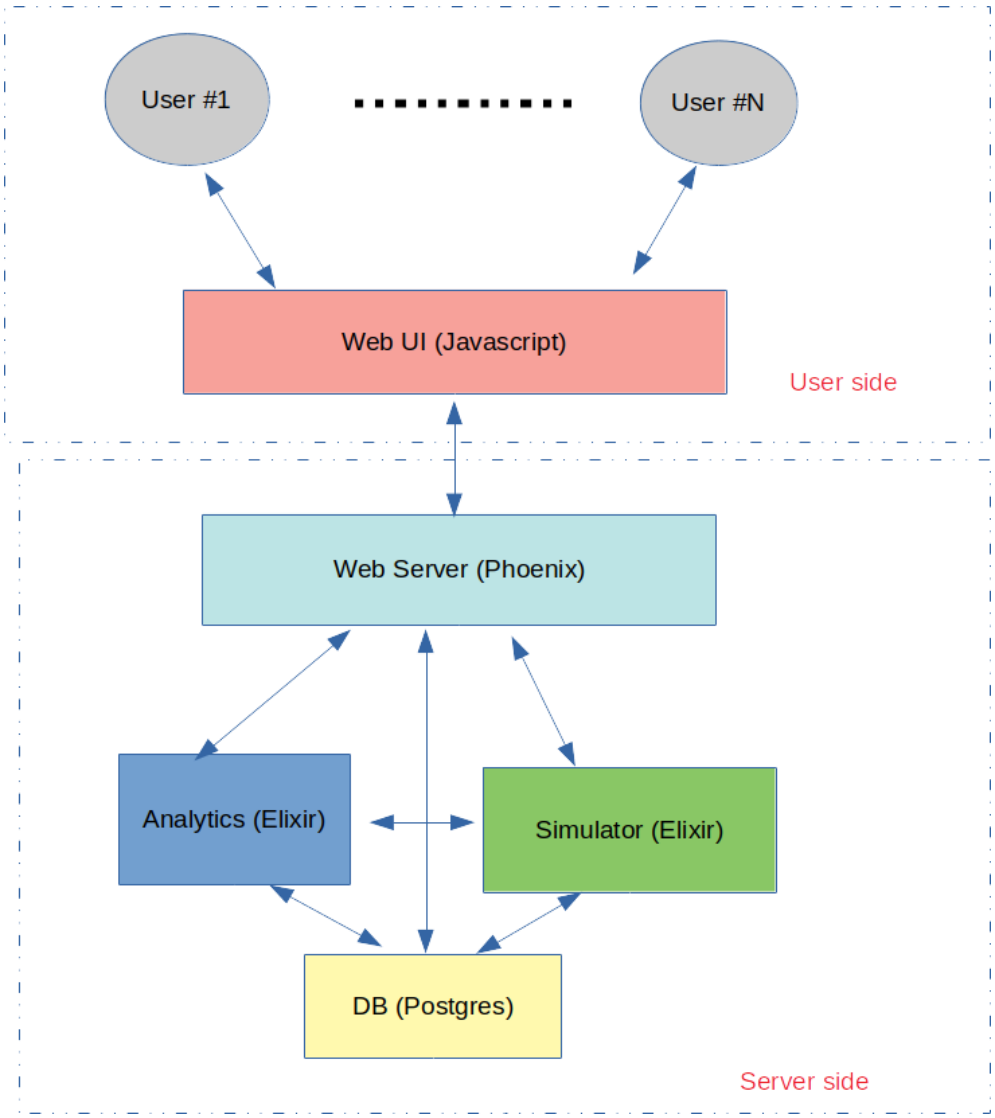

FIGURE 18 - POLITIKA OVERALL ARCHITECTURE

4.3.1 Design of the Social Dynamics Simulator

Figure 19 describes the design of the Social Dynamics simulator. Bidirectional red arrows indicate data sources which the simulator can read and/or update. Unidirectional black arrows indicate user-defined data sources which the simulator can only read.

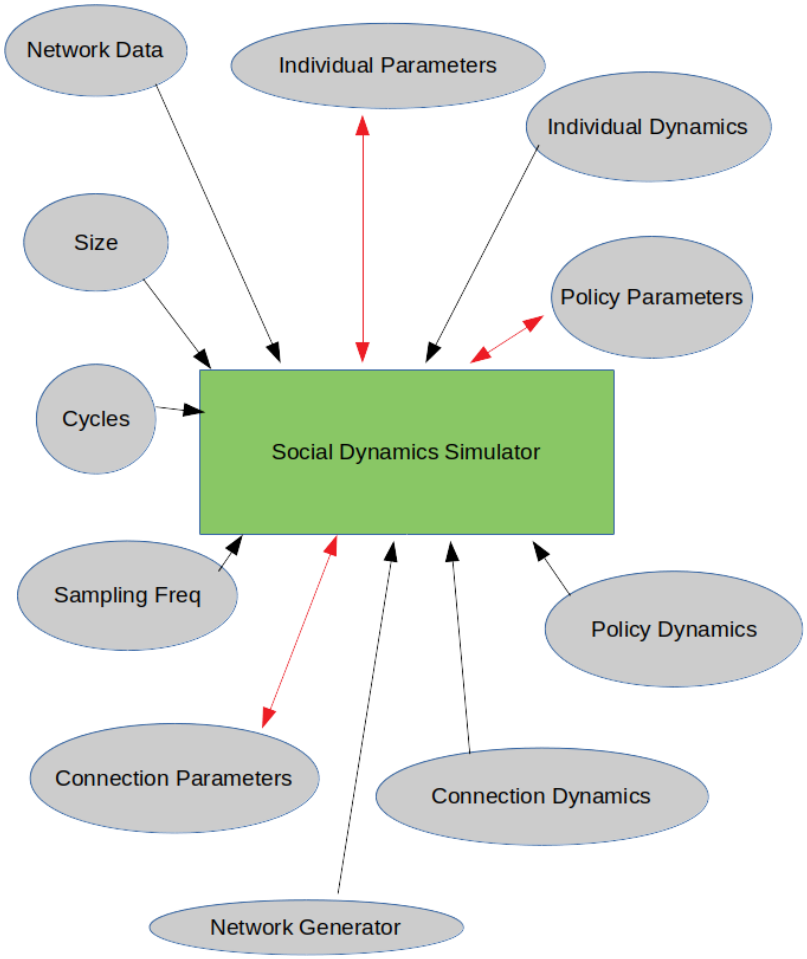

FIGURE 19 - DIAGRAM FOR THE SOCIAL DYNAMICS SIMULATOR

The I/O specification for this component consists of the following data sources:

- A list of policy parameters and their initial values. For example, in a public health policy scenario regarding a pandemic disease can include the epidemiological data (infection probability, death rates, infection/immunity population percentages etc), the public health system parameters (number of ICUs, available medical staff etc) and various key performance indices associated with a policy. At the end of each simulation cycle the simulator updates these parameters.
- A set of rules for policy dynamics specifying how the policy parameters change during the simulation.
- A list of parameters and their initial values that describe the characteristics of each individual agent in the population. For example, in our public health policy scenario, these can include the individual's age, health condition etc. During each simulation cycle the simulator updates these parameters for each individual in the population.
- A set of rules for individual dynamics describing how parameters for each individual are updated during the simulation. Furthermore, these rules describe how the connectivity for each individual change during the simulation (e.g. creation/deletion of individual connections). Finally, these rules describe population changes (e.g. addition/removal of individuals from the population).
- A list of connection parameters and their initial values that describe the connection characteristics between agents in the population. For example, in our public health policy scenario, these can include the contact strength of each connection, its frequency etc. Each individual can be connected to multiple

other individuals. Each connection is unidirectional. During each simulation cycle the simulator updates the parameters of all the connections for each individual in the population.

- A set of rules for connection dynamics describing how both connections and their parameters change during the simulation.
- The size of the population on which the policy will be simulated.
- The number of cycles for which the simulator will run.
- The sampling frequency (in simulation cycles) with which parameters for each individual are sampled, so that a history for the parameter values of all individuals at various cycle intervals can be built for further analysis.
- The output of a network generator component. This is used whenever the user wants to simulate policy effects on a population generated from a known network model such as a random graph or various power-law networks.
- Network data describing in a JSON format the individual and connection parameters of an actual population that can be uploaded/downloaded to/from the simulator.

Parameters are specified as keyword/value strings. For example, let's take a look at the following list of parameters:

```
max_icu_units:    100,    incubation_period:    5,    infection_period:    10,
immunity_period:          10,          cur_infection_death_prob:          0.4,
baseline_infection_death_prob: 0.4, critical_prob: 0.1, s: 0, e: 1, i: 2, r:
3, d: 4, distancing_prob: 0.5, connection_attempts: 5, cur_icu_units: 0
```

These parameters represent the policy parameters for a containment policy in an epidemiological context. Strings ending in `:' are keyword names for parameters, while the rest are the parameter values.

Rules are specified as IF..DO constructs. For example:

```
IF this.cur_icu_units > 0 and
  this.cur_infection_death_prob > this.baseline_infection_death_prob
DO
 this.cur_infection_death_prob:
   max(this.cur_infection_death_prob – 0.1*this.cur_infection_death_prob,
     this.baseline_infection_death_prob)
```

represents a policy dynamics rule that decreases the current infection death probability if it is above the baseline probability provided there are icu units available.

4.3.1.1 Simulator Execution Scheme

All types of dynamics (policy, individual or connection) are specified as lists of IF..DO rules. In the case of individual dynamics these rules utilize:

- parameter values for the individual involved
- macroscopic connection data (e.g., number of incoming/outgoing connections for the individual, existence, filtering and/or comparison of certain connection parameter values in incoming connections, results of function evaluations on parameters of incoming connections (e.g. sum, threshold, mean)

As a result, these rules update the parameters of the individual involved and/or manage outgoing connections (e.g. remove/create outgoing connections). In the case of connection dynamics these rules utilize parameters of individuals at both ends of the connection along with attributes of the specific connection to update the attributes of that connection. Finally, policy dynamics rules utilize current values of policy parameters while also computing macroscopic population parameters based on parameters of individuals to update policy parameters.

The need to create computationally efficient simulations through concurrency requires the design of an execution scheme that guarantees proper updating of parameters. In particular, this scheme needs to:

1. provide immutable state for each parameter that avoids situations in which parameters accidentally change as a result of various side effects during computation

2. ensure that at each point in time only one process updates a parameter, otherwise the results of the updating can be undetermined.

The first requirement is satisfied through the use of functional programming and in particular Elixir as the development environment. This is the case because in functional programming all variables have immutable state as each variable update results in a new copy of the variable. Therefore, data objects generated at different times remain distinct.

The second requirement requires a careful structuring of all computations involved that can provide correct updating of all variables but also ensure the autonomous behavior of each individual in the simulation. To this end, connection dynamics rules update only the parameters for each connection, while parameters for individuals are updated only through the application of individual dynamics rules for each individual. Therefore, in its outgoing connections each individual has read/generate/remove permissions via its individual dynamics rules and read/write permissions via its connection dynamics rules. Each individual has only read permissions on its incoming connections. This scheme allows the simulator to run the individual and connection dynamics rules for each individual concurrently as depicted in Figure 20:

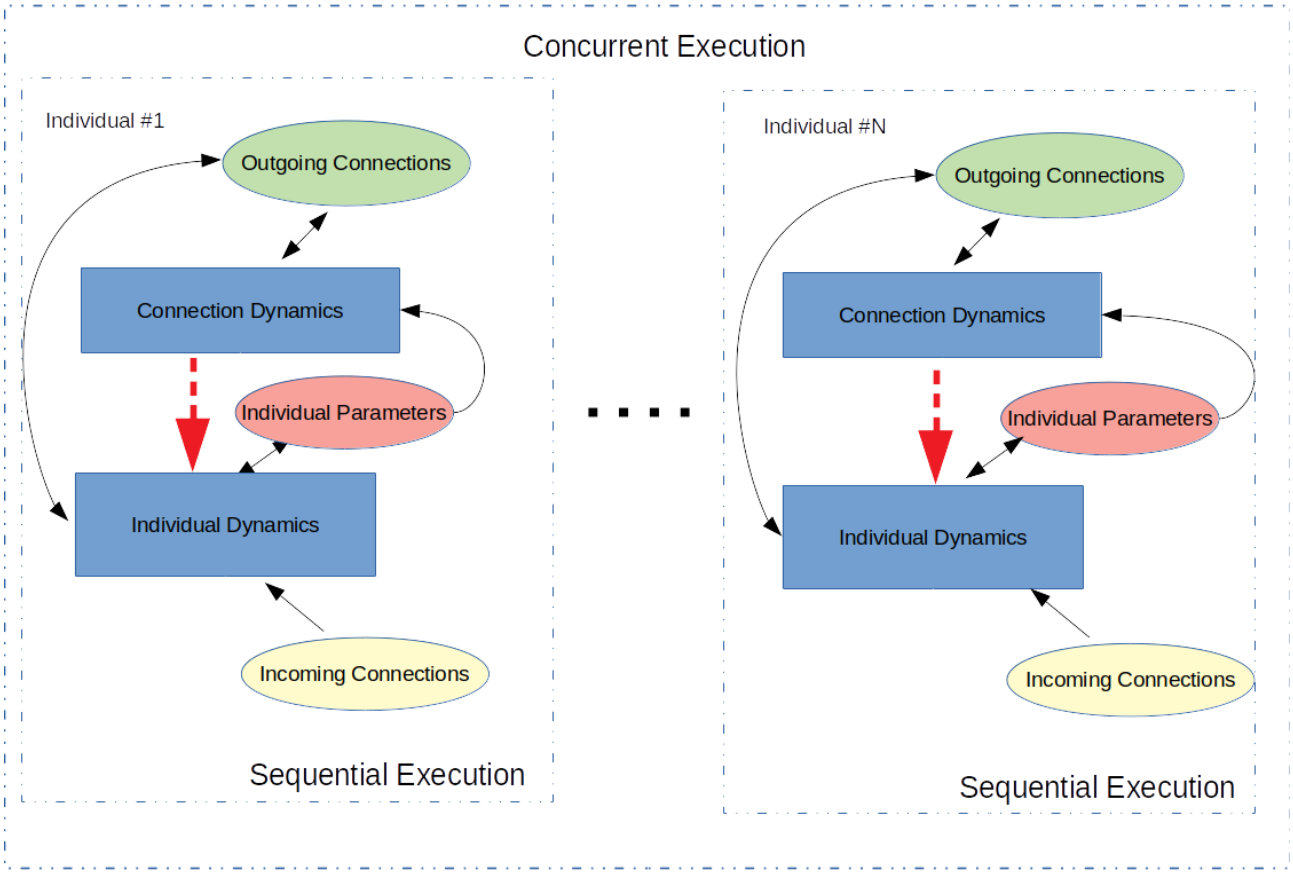

FIGURE 20 - EXECUTION SCHEME OF THE SOCIAL DYNAMIC SIMULATOR

Red arrows indicate execution order in the sequential execution case. Unidirectional arrows indicate data sources that can only be read by a dynamic subsystem. Bidirectional arrows indicate data sources that can be read and/or be updated by such a subsystem. As this figure shows during such execution each individual runs its connection and individual dynamics rules sequentially in the order specified by the simulation designer. This ensures that only one rule can update a parameter at each time, while also allowing the designer to code for scripted behaviors in individuals that assume a specific execution order of their relevant rules (e.g. coding of steps in financial transactions). Finally, policy dynamics rules are executed sequentially in the order specified by the designer. They can only read parameters of individuals while they can read and update policy parameters. Overall, this scheme ensures the proper updating of individual, connection and policy parameters in a concurrent execution environment, because for each one of them there is a unique way of modifying it at each point in time.

The simulator takes into consideration the fact that a computing system has a finite set of schedulers that can execute concurrent tasks, as this is related to the number of processor cores available. Therefore, at the start of each cycle the simulator divides the population into chunks and executes concurrently the dynamics for the individuals at each such chunk. Chunk size is chosen so as to optimize the size of the execution queue for each scheduler. Furthermore, in order to avoid artefacts resulting from the use of fixed chunks during the simulation that imply a standard execution order among the individuals in the population (e.g. the dynamics of Individual #1 are computed always before that of Individual #2), the simulator generates randomly a new set of chunks at the start of each simulation cycle.

4.3.2 Design of the Social Dynamics Analytics

The Analytics component in Politika provides a suite of generic analysis and visualization capabilities. The goal is to reduce the need for using other specialized software for such tasks and provide real-time analytics and visualization during simulation. As

Figure 18 indicates, this component receives analytic/visualization requests from the user via the web server component. It then retrieves the appropriate simulation data either directly from the simulator, if it is a currently executing simulation, or from the database, if it concerns stored simulation results, and proceeds with computing the analytic request and sending the results back to the user. The component employs a websocket-based scheme to update analytics and visualization parameters during user interaction. Visualization capabilities are currently being implemented using the D3.js[20] visualization library. Current analytic/visualization capabilities under development include:

- real-time network visualization through tracking of the position and connectivity of each node during simulation
- graphing of the values of the parameters of each individual in a simulation
- computation and display of average values for the parameters of all individuals in a simulation.

4.4 Optimization & Reusability of Analytical Tools (Task T4.6)

This will be implemented in the second the second project year and will be described in the next deliverable report (D4.3).

[20] https://d3js.org/

5 Cloud Platform and Software Tools

In this section we'll give short description of the cloud platform and software tools that are used for the implementation of the PolicyCLOUD platform, and explanation of their suitability and the reasoning for their selection.

5.1 Kubernetes cluster

The first design question for the PolicyCLOUD platform was related to the underlying virtualization platform: virtual machines or containers? We chose the container-based solution for the following reasons:

- Efficient application deployment and overall CI/CD cycle
- Excellent suitability for a serverless function-based platform, with the extensibility and reusability requirements of PolicyCLOUD
- Suitability for micro-services architecture, which enable easy deployment and separation of the PolicyCLOUD components
- Portability over cloud providers
- Growing popularity and eco system development, especially in the open source communities

Once the container direction was decided, the Kubernetes[21] container management platform was a clear choice. Kubernetes is the leading open source container management platform used in production of growing number of enterprises as the base of private on-prem cloud, as well as offered by almost all cloud providers as a managed dedicated cluster. It provides a framework to run distributed systems resiliently by taking care of scaling, load balancing and failover (e.g. if a container goes down, another container automatically restarts) for the containerized applications and provides deployment patterns that drastically simplify application deployment and management.

All software components of PolicyCLOUD will be deployed on a Kubernetes cluster which will be the underlying application management platform.

5.2 OpenWhisk cluster

Apache OpenWhisk[22] is an open source light-weight serverless platform that provides capability to deploy functions (written in any language) and specify triggers and rules by which the functions are executed. It offers a simple programming model where the function developer can concentrate only on the mere logic of the function, while the deployment and activation details are taken care transparently. It is based on containers as the functions' wrapper, and deploys and integrates perfectly on a Kubernetes environment. As described in sections 2.2.2 , all analytic functions in PolicyCLOUD will be deployed as OpenWhisk functions, with appropriate triggers and activation rules, addressing the extendibility and reusability requirements. Additional important capability of OpenWhisk is the composition of functions to be activated per trigger/rule, which enable to specify multiple analytics to be applied in series to a data source.

[21] https://kubernetes.io
[22] https://openwhisk.apache.org

5.3 LeanXcale database

LeanXcale database is an integral part of the PolicyCLOUD data repository, along with the *object store.* It is a relational datastore, which ensures transactional semantics and provides ACID (Atomicity, Consistency, Isolation, and Durability) properties and is offers a JDBC interface, supporting standard SQL queries. Moreover, it provides an internal key-value store that can be used in cases that very low latency is a requirement in order to support data ingestions in very high rates, achieving a very high throughput. It consists of three main pillars: the data storage itself, the scalable transactional manager and the query engine. Each of those pillars can be scale out independently, thus being able to support very high loads. Its support for transactions is based on the recently adopted *snapshot isolation* paradigm that removes the need for data locking often used by traditional implementations, and therefore, it allows the concurrent support for operational (OLTP) and analytical (OLAP) processing on the same dataset. This means that the analytical functions of the PolicyCLOUD can perform analytics on real data, as data are being ingested into the platform, thus removing the necessity of performing cost-expensive ETLs for migrating data from the operational datastore to a data warehouse that can be used for analytics. Finally, it provides polyglot capabilities and can be used as a common endpoint to retrieve and pre-process data from external data sources in a seamless manner.

Figure 21 depicts the main architectural components of the LeanXcale datastore.

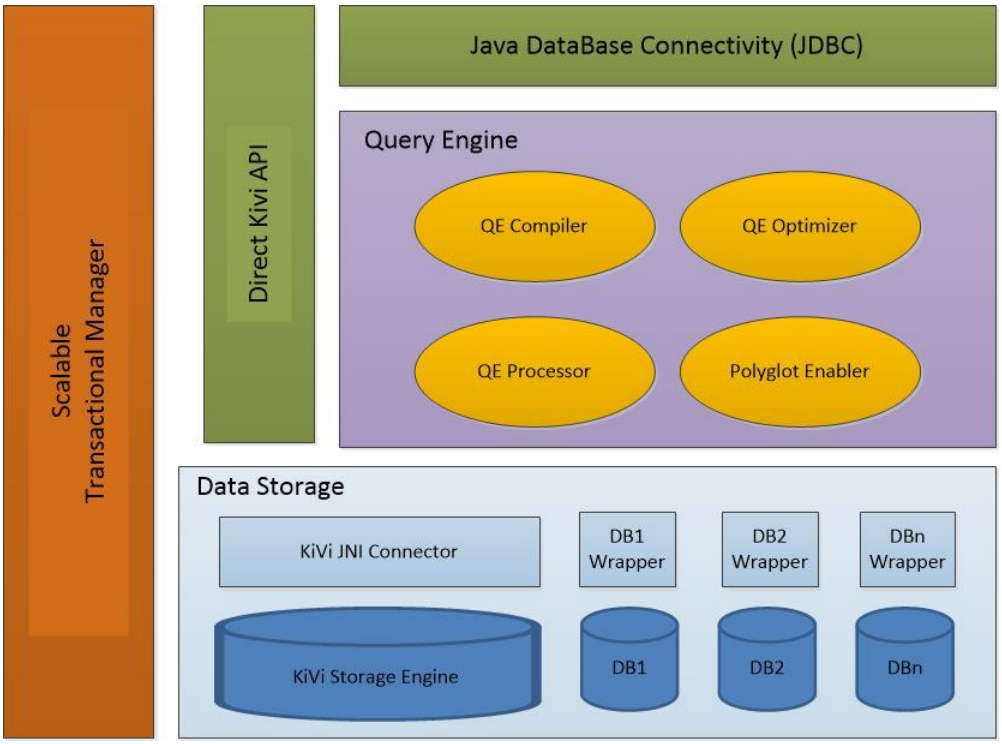

FIGURE 21 - LEANXCALE DATASTORE ARCHITECTURAL DESIGN

In the architectural design, we can see the three main pillars of the solution. At the bottom layer, there is the data storage component. Its central component is the KiVi storage engine, which is the one responsible for persistently store data items to the disk. It allows data items to be stored in a tabular format, providing additionally support for data indexing. Internally, it comes with its own query processing that not only allows for the traditional get/put operations that can be used in traditional key-value stores, but additionally, it exposes a rich interface that supports the majority of the SQL standard operations. The data analyst and application developer can benefit for letting the Storage Engine perform operations like scans, filtering, ordering on columns

that are indexed and aggregations, thus letting the engine do the pre-processing, without having to return the whole dataset to the application or the analytics layers. It has been implemented in C programming language; however, it provides a JNI Connector that wraps this functionality over a Java-based implementation that can be used instead. Finally, as the whole LeanXcale solution provides polyglot support, allowing the application developer to access data from external data sources, we can assume that those datastore providers are also part of an extended Data Storage layer. For each one of those, a specific wrapper has been implemented in order to allow the transparent access by the upper layers of the solution. The wrappers implement the same interface that can be used by the *query processor* of the *query engine.* This allows to execute queries that can join datasets that they are either stored internally in the PolicyCLOUD repository, or they resign in external data stores.

On top of the Data Storage component, there is the Query Engine of LeanXcale. It supports statements writing in standard SQL language and internally it provides a parallel OLAP engine that can be used to efficiently execute analytical queries. The latter allows for inter and intra query and operation processing, thus making it possible for a query statement to be executed in a distributed manner. When a SQL statement is received, then the query engine compiler transforms it to a structural format: the SQL operations are being transformed to a tree of operations. Then, the *optimizer* applies various transformation rules that produce different representations of the tree, which returns an equivalent result with the original one. It then calculates the cost of execution of each of those operations and decides about the optimal query execution: the execution that would require the minimum I/O access in the storage level and will make the minimum calculations in the query engine side. When the optimal execution plan has been decided, it is sent to the *processor* that establishes the data pipeline of the operations for data retrieval and starts retrieving data. It is important to mention that due to the intra-operation parallelism of engine, each of the operations itself can be executed in a distributed manner, while the data pipeline of the operations can be also deployed to be executed in a different node. This allows the LeanXcale data base to exploit a huge level of parallelism while executing a query, and additionally, execute the load in nodes that are not busy and does not consume many resources at that time. As it can be depicted from Figure 21, query engine comes with the *polyglot enabler* component which interacts with the *wrappers* of the external data stores, thus, the *query processing* can establish a data pipeline where one of the operators can retrieve data from one external source, in a seamless way.

Moreover, the *transactional manager* can be depicted vertically in the overall architecture and ensures transactional semantics across the stack. It relies on the *snapshot isolation* paradigm, which removes the need for adding and maintaining locks on data items, which is usually the case in traditional relational datastores that rely on the *two-phase locking* protocol. By not having to maintain all these locks introduced by transactions, we removed the inherit bottleneck that those protocols suffer, and as a result, our solution can scale out adequately and can server operational workloads coming in very high rates. Both the query engine and the storage engine interact with the *transactional manager*, which allows us to ensure transactional semantics even if the data ingestions are accessing the data storage directly, and the data analytics are being served by the query engine.

Regarding data connectivity, there are two ways for accessing the LeanXcale datastore: Either by the direct KiVi API, or by a JDBC implementation. The former exposes a MongoDB-like interface and access the storage by bypassing the query engine. This removes the inherit footprint that the query engine natively introduces and can be used in cases when there is the need to support data ingestion in very high rates. It makes possible for the application developer and the data analyst to exploit the unique characteristics of the storage layer; however, it comes with a drawback: its API is not compliant with a standard, as no interfaces of that type does. In order to overcome this, an implementation of the standard JDBC is also provided that allows access via the query engine of the datastore and its parallel OLAP engine. This allows the integration with popular analytical frameworks that are currently used for ML/DL algorithms that the majority of the analytical functions of PolicyCLOUD exploit. All those frameworks are compatible with JDBC and can push the execution of the query statement down to the datastore level, which gives the benefit of not having to transmit the whole dataset via network to the level of the analytical processing, and perform all pre-processing for data retrieval down to the nodes where the data

is actually resigned. In combination with the level of parallelism that the OLAP engine supports, it allows for effective query execution in a standardized manner.

The LeanXcale datastore can be deployed either directly on the machines (bare metal installation) or via a container orchestration tool such as docker. The solution has been containerized, which facilitates the deployment over a Kubernetes cluster. It is important to mention that when using Kubernetes, the whole installation can be deployed in a single pod, or several pods can accommodate different components of the datastore. Moreover, as it has been previously stated, each of the components can scale out independently. This allows for a great flexibility regarding deployments, as the database administrator can scale the data storage and query engine nodes in cases that need to support in increased data load, or the transactional manager components in case that need to support and increased operational workload with hundreds of thousands of transactions per second. The maintenance can be facilitated with the automation process provided by Kubernetes.

5.4 Spark cluster

Apache Spark[23] is highly popular open source analytic engine for large-scale data processing. It achieves high performance for both batch and streaming data and powered by numerous open source libraries such as SQL, streaming and machine learning to manipulate and query data. The Spark cluster will be used in PolicyCLOUD in conjunction with the LeanXcale database to provide seamless analytic on hot and cold data at rest, where the Spark SQL with optimization exploited from the BigDataStack EU[24] project will be used for queries on the colder data in the object storage.

5.5 Object storage

Object Storage is a storage architecture providing high capacity with low cost, and a definite choice for big data that is once-written, i.e., is not going to be modified after being written (to modify an object, it needs to be deleted and re-written). An object can have metadata (that is used e.g. to increase analytics performance by skipping irrelevant objects), and in contracts to the object itself, its metadata can be modified. The object storage platform is accessed through RESTful HTTP of PUT/GET/POST/DELETE operations on objects. In contrast to file system, there is a flat namespace. Object storage is offered today by almost all cloud providers e.g. Amazon S3, IBM COS, Google Cloud Storage. Object Storage was a clear choice in PolicyCLOUD for the colder big data store.

[23] https://spark.apache.org/
[24] https://bigdatastack.eu/

6 Conclusion

We provided in this document the architecture of the Data Acquisition and Analytics Layer and included components. This is a result of requrement analysis and design sessions conducted in the scope of Work Package 4 and in the global project scope for inter-layers aspects. The document includes the architecture of the overall layer, the included data analytic and transformation tools, and the cloud platform and software tools that are planned at this point to be used for the implementation. The architecture details the are provide here are subject to change as the project evolve due low level details discovered as the lower level design and the actual implementation progress, which may affect the higher-level design decisions, and especially when the planned technologies are applied to the actual PolicyCLOUD use cases.

References

[1] PolicyCLOUD *D2.1 - STATE OF THE ART & REQUIREMENTS ANALYSIS*. Pavlos Kranas. June 2020

[2] PolicyCLOUD *D2.2 - CONCEPTUAL MODEL & REFERENCE ARCHITECTURE*. Panayiotis Michael. 2020

[3] PolicyCLOUD D3.1 - Cloud Infrastructure Incentives Management and Data Governance Design and Open Specification 1

[4] Hutto, C.J. & Gilbert, E.E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media* (ICWSM-14). Ann Arbor, MI, June 2014.

[5] Seaborn: statistical data visualization, https://seaborn.pydata.org/.

[6] DCAT Application profile for data portals in Europe (DCAT-AP), https://op.europa.eu/en/web/eu-vocabularies/dcat-ap.

[7] Hellmann S., Lehmann J., Auer S., and Brümmer M., Integrating NLP using Linked Data,12th International Semantic Web Conference, pp. 21-25, 2013.

[8] JSON-LD - JSON for Linking Data, http://json-ld.org.

[9] Xin J., Afrasiabi C., Lelong S., Adesara J., Tsueng G., Su A. I., and Wu C., Cross-linking BioThings APIs through JSON-LD to facilitate knowledge exploration, BMC bioinformatics, 19(1), 30, https://doi.org/10.1186/s12859-018-2041-5, 2018.

[10] T. Berners-Lee, Linked Data - Design Issues, www.w3.org, 2006.

[11] D. Brickley, and R. V. Guha, Resource Description Framework (RDF) Schema Specification 1.0: W3C, 2000.

[12] D. L. McGuinness, and F. Van Harmelen, OWL Web Ontology Language Overview. W3C recommendation, World Wide Web Consortium, 2004.

[13] T. Berners-Lee, The semantic web, Scientific American, Vol. 284(5), pp. 35–43, 2001.

[14] Groß A., Pruski C., and Rahm E, Evolution of biomedical ontologies and mappings: overview of recent approaches. Computational and structural biotechnology journal, Vol. 14, pp. 333-340, 2016.