



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

D4.5 Reusable Model & Analytical Tools: Design and Open Specification 3

Dissemination Level	PU
Due Date of Deliverable	31/08/2022, M32
Actual Submission Date	31/08/2022
Work Package	WP4 Reusable Models & Analytical Tools
Task	All
Type	Report
Approval Status	
Version	1.0
Number of Pages	p.1 – p 105

Abstract: Third and last version of the Internal architecture of the Integrated Acquisition and Analytics Layer, responsible for the a) integration of analytical tools in extensible manner, b) registration of new data sources c) applying the required transformation and cleaning on the data fusion pass, d) (new) integration with external frameworks such as Politika, e) specification and design of the built-in analytics tools for Situational Knowledge, f) Opinion Mining & Sentiment Analysis, g) Social Dynamics & Behavioral Data analysis, and h) Optimization & Reusability of Analytical Tool section.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability. This deliverable is licensed under a Creative Commons Attribution 4.0 International License.



Versioning and Contribution History

Version	Date	Reason	Author
0.1	24/05/2022	Initial version based on D4.3	Yosef Moatti (IBM)
0.2	17/07/2022	Draft complete but for T4.6	Team
0.3	24/07/2022	Draft ready for internal review but for T4.6	Team
0.4	27/07/2022	Internal review completed by Rafael	Rafael
0.5	15/08/2022	T4.6 added by Pavlos	Pavlos
0.6	16/08/2022	T4.6 Internal review completed by Nikitas	Nikitas
0.7	19/08/2022	Internal review completed by Pavlos	Pavlos
0.8	22/08/2022	First draft for final version	Yosef
1.0	31/08/2022	Ready to be published	Team

Author List

Organisation	Name
IBM	Ofer Biran, Oshrit Feder, Yosef Moatti
LXS	Pavlos Kranas, Javier Periera, Alejandro Ramiro
ATOS	María Ángeles Sanguino González, Sergio Salmerón Majadas
UPRC	Argyro Mavrogiorgou, Thanos Kiourtis, George Manias, Nikitas Sgouros
OKS	Kostas Nasias
ICTLC	Martim Taborda Barata, Isabella Oldani

Abbreviations and Acronyms

Abbreviation/Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
AO	Analytic Owner
EC	European Commission
ENISA	European Union Agency for Cybersecurity
EOSC	European Open Science Cloud
GTD	Global Terrorism Database
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
ID	Identification (number/code)
ML	Machine Learning
NFS	Network File System
NER	Named Entity Recognition
NLP	Natural Language Processing
PDT	Policy Development Toolkit
PM	Policy Maker
POS	Part-of-Speech
REST	Representational State Transfer
SAF	Seamless Analytical Framework
SKA	Situational Knowledge Acquisition

Contents

Versioning and Contribution History.....	2
Author List.....	2
Abbreviations and Acronyms	3
Executive Summary	8
1 Introduction.....	10
1.1 Positioning of this deliverable	10
1.2 WP4 Objectives.....	10
1.3 WP4 Positioning and Relations to other Work Packages.....	10
2 Data Acquisition and Analytics Layer.....	11
2.1 General Architecture and Roles	11
2.1.1 Updates since D4.3.....	11
2.1.2 Architecture and analytic functions APIs.....	11
2.2 Extensibility and Reusability	16
2.2.1 Updates since D4.3.....	16
2.2.2 Data Source Registration	16
2.2.3 Analytic Function Registration	16
2.2.4 Analytics Function Invocation	17
2.2.5 Integration of external frameworks.....	17
2.3 Security and Legal concerns.....	21
2.3.1 Updates since D4.3.....	21
2.3.2 Ethical, legal, regulatory, and societal considerations.....	21
2.3.3 Bias and Trade-off Management for Analytic Functions.....	29
2.3.4 Logging Service	32
3 Data Fusion with Processing and Initial Analytics.....	34
3.1 Cloud Gateways (Task T3.3).....	35
3.1.1 Updates since D4.3.....	36
3.2 Enhanced Interoperability & Data Cleaning (Task T4.2)	36
3.2.1 Updates since D4.3.....	37
3.2.2 Data Cleaning.....	37
3.2.3 Enhanced Interoperability	41

4	Analytic on Data at Rest.....	50
4.1	Situational Knowledge Acquisition & Analysis (Task T4.3).....	51
4.1.1	Updates since D4.3.....	51
4.1.2	functionality extracted from pilots.....	51
4.1.3	SKA components.....	52
4.2	Opinion Mining & Sentiment Analysis (Task T4.4).....	56
4.2.1	Updates since D4.3.....	56
4.2.2	Functionality extracted from pilots.....	57
4.2.3	Contextual Information Opinion and Sentiment components.....	58
4.3	Social Dynamics & Behavioral Data Analytics (Task T4.5).....	62
4.3.1	Updates since D4.3.....	62
4.3.2	Introduction.....	62
4.3.3	Simulation Environment for Social Dynamics.....	63
4.3.4	Meta-Simulation Environment for Policy Analysis and Design.....	73
4.3.5	Policy GUIs.....	78
4.3.6	Integration of the Social Dynamics Component with PolicyCLOUD.....	81
4.3.7	Simulation and Policy Models for the Agri-Food Pilot Use Case.....	81
4.3.8	Updated Simulation and Policy Models for the Radicalization Pilot Use Case.....	84
4.3.9	Future steps.....	84
4.3.10	Conclusion.....	85
4.4	Optimization & Reusability of Analytical Tools (Task T4.6).....	88
4.4.1	Updates since D4.3.....	88
4.4.2	User Story and Motivation.....	89
4.4.3	Architecture.....	90
4.4.4	Seamless within Policy Cloud.....	91
5	Cloud Platform and Software Tools.....	96
5.1	Updates since D4.3.....	96
5.2	Kubernetes cluster.....	96
5.3	OpenWhisk cluster.....	97
5.4	LeanXcale database.....	97
5.5	Spark cluster.....	100
5.6	Object storage.....	100

6 Conclusion	101
References	102

List of Figures

Figure 1 - WP4 interface with WP3 and WP5	11
Figure 2 - Data acquisition and analytic layer	12
Figure 3 - Data Source Path	15
Figure 4 - Example of a function registration	17
Figure 5 - Steps of an invocation of an external framework	21
Figure 6 - The streaming data path	34
Figure 7 - Data cleaning workflow	38
Figure 8 - Ontology-Based NER	45
Figure 9 - Data preprocessing layer	46
Figure 10 - Ontology mapping step.....	47
Figure 11 - Annotated tweet through the utilization of Enhanced Interoperability component	49
Figure 12 - Seamless analytics on ingested data.....	50
Figure 13 - SKA-EDA General Architecture.....	53
Figure 14 - SKA-TSF Training Process	55
Figure 15 - SKA-TSF Predicting Process	55
Figure 16 - Aspect-Based Sentiment Analysis General Architecture	59
Figure 17 - Conceptualization process example	60
Figure 18 - Bursty concept example	61
Figure 19 - Overall architecture of the politika simulation environment	65
Figure 20 - I/O specification of the social dynamics simulation.....	66
Figure 21 - Concurrent execution scheme of the social dynamics simulator	71
Figure 22 - Screenshot of simple simulation model in POLITIKA dealing with radicalization	73
Figure 23 - Example policy structure used by the meta-simulation environment	74
Figure 24 - POLITIKA processing pipeline	76
Figure 25 - Screenshot of the tree GUI for a policy model in Politika	79
Figure 26 - Snapshot of the text-based GUI for defining a policy model.	80
Figure 27 - EXample. chart for the Social Dynamics output in the Agri-Food use case (generated by PolicyCLOUD).....	83
Figure 28 - POLITIKA GUI: Management of network data and execution of simulations.....	86
Figure 29 - Visualization example of scaled attribute values for each individual in a population	87
Figure 30 - Visualization example of a connectivity matrix.....	88
Figure 31 - Seamless high-level architecture	90
Figure 32 - Seamless analytical framework with data mover.....	92
Figure 33 - JOIN of 2 seamless datasets	94
Figure 34 - Breakdown of the JOIN.....	94

Figure 35 - LEANXCALE datastore architectural design 98

List of Tables

Table 1: Updated WP4 Legal/Ethical Checklist 29

Table 2: Self-Adjustment of the data cleaning based on dataset domain 39

Executive Summary

This document is the third and final architecture deliverable for WP4. In this summary we first mention what the major additions made during the past year, that is from D4.3 and then give a quick overview of the overall work achievements by end of month 32.

Following are the major changes that were added in the past year: new subsection 2.2.5: “Integration of External Frameworks” was added. It describes the generic integration of external frameworks and how specifically we applied it to the Politika framework.

In subsection 2.3: “Security and Legal concerns”, a lot of new information concerning the Legal/ethical framework was added. This includes the legal questions which are presented to registrants of datasets or analytic functions to DAA and which will guide the registrant to fill in all the needed legal details.

Section 4.1 (SKA analysis) reports of incremental improvements, except for the new situational knowledge analysis based on time series forecasting (SKA-TSF), a new component which has been fully defined and is at the closure of this document under development.

Section 4.2 reports on contextual Information opinion and sentiment components, reporting on the status of the aspect-based sentiment analysis and introducing a new component for the trend analysis.

The description of the Social Dynamics component was extended in multiple directions: section 4.3.5 presents the GUIs that have been developed for defining a policy model during design, section 4.3.6 briefly covers the integration of this component with the PolicyCLOUD platform, and finally sections 4.3.7 and 4.3.8 describes simulation and policy models respectively related to the Agri-Food and the Radicalization use cases.

Following is a summary of the overall achievements of the WP4 tasks:

The Data Acquisition and Analytics layer (DAA) was architected and implemented in a successful way as it was successfully used for all the Use Case scenarios. Furthermore, the DAA was also the natural basis for the integration of external frameworks as detailed in Section 2.2.5.

Task T4.2 comprises two major technologies: Data Cleaning and Enhanced Interoperability. Since they are interlinked we present their major achievements as a whole:

First of all, the design and implementation of all the sub mechanisms, including these components genericity, were completed. The corresponding final software prototypes were released.

The integration with all components and tools of PolicyCLOUD Data Ingestion Pipeline was also successfully completed.

Concerning Enhanced Data Interoperability, the integration and collaboration with T4.4 “Opinion Mining & Sentiment Analysis” for providing an enhanced Entity-Level Sentiment Analysis (ELSA) mechanism was achieved.

Concerning Data Cleaning, an extensive list with generalized cleaning rules and actions, covering multiple domains (going beyond the PolicyCLOUD Use Cases and scenarios) was elaborated and implemented.

Finally end-to-end integration was achieved for all the Use Cases and scenarios

Concerning Situational Knowledge Acquisition and Analysis several analytical tools have been implemented. In general terms, they are focused at i) providing exploratory data analysis based on data visualization and ii) predicting analysis based on time series forecasting. End-to-end integration was achieved for all the use cases and scenarios.

Concerning Sentiment and Opinion Analysis two main capabilities have been provided, i) a sentiment analysis tool for extracting the feeling and opinion from text as well as entities detected in the text and ii) a trend analysis tool for conceptualization, contextualization and monitoring of given entities in social media. End-to-end integration was achieved for all the use cases and scenarios.

T4.5 Concerning Social Dynamics, we implemented this component as a stand-alone web tool referred to as Politika. Politika uses agent-based, social simulation as the primary analysis tool for evaluating policy alternatives based on a special-purpose modeling language, a concurrent simulator and a meta-simulation framework for automatically managing and evaluating simulations of policy alternatives. We then proceeded with the integration of Politika to the rest of the PolicyCLOUD platform through a REST API. Finally, we developed two use cases in Politika as part of the Radicalization and Agri-Food groups of policies examined in PolicyCLOUD.

T4.6 Concerning the Optimization and Reusability of Analytical Tools, we relied on the *Seamless Analytical Framework*, firstly introduced by the BigDataStack EU project. The aim of this framework is to combine and unify the best benefits of two complementary worlds in data management systems: the operational and the analytical datastores. However, the background technology had some limitations when performing complex relational algebraic operations such as the execution of the SQL "JOIN". Under the scope of PolicyCLOUD, we designed a solution that can overcome this limitation by re-writing the source code related with the runtime execution of this complex operator, when it involves federated datasets that are split across both the two involved datastore solutions.

1 Introduction

1.1 Positioning of this deliverable

This document is the third and last design technical deliverable for WP4. It is an incremental update of the PolicyCLOUD deliverable D4.3. As such, it repeats some of the content and adds new developments, and new technologies (e.g., section 4.1). Each of the topics presented in this document is preceded by a short summary of the updates since D4.3.

1.2 WP4 Objectives

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

- Provide the framework for data fusion and aggregation – for different data source types
- Provide data cleaning ensuring quality of information, sources reliability assessment, reliability-based selection of information sources
- Provide 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
- Decouple the analytical models and tools from the underlying infrastructure and datastores, assuring their reusability.

All of these objectives were fulfilled. WP4 provides an extensible framework for applying various analytics on different data sources, through the development of the Data Acquisition and Analytics Layer and the basic analytic tools that have been registered in it.

In this third version (D4.5), we provide an additional section: “Integration of external frameworks” (2.2.5) that was missing in the second version (D4.3) and we also reported the evolution, sometime significant, of the other already introduced technologies.

1.3 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 on the one hand, 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), and on the other hand, various analytical tools that can be used by the upper layers of the PolicyCLOUD platform, in addition we handled a new requirement which is to enable the integration of external frameworks as detailed in section 2.2.5.

From the aspect of Work Packages partitioning, the Data Acquisition and Analytics Layer is under the responsibility of WP4 (Reusable Models & Analytical Tools) and its tasks, it strongly relies on 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. Let's also note the introduction in this past year of the external framework which can now be integrated to the PolicyCLOUD platform.

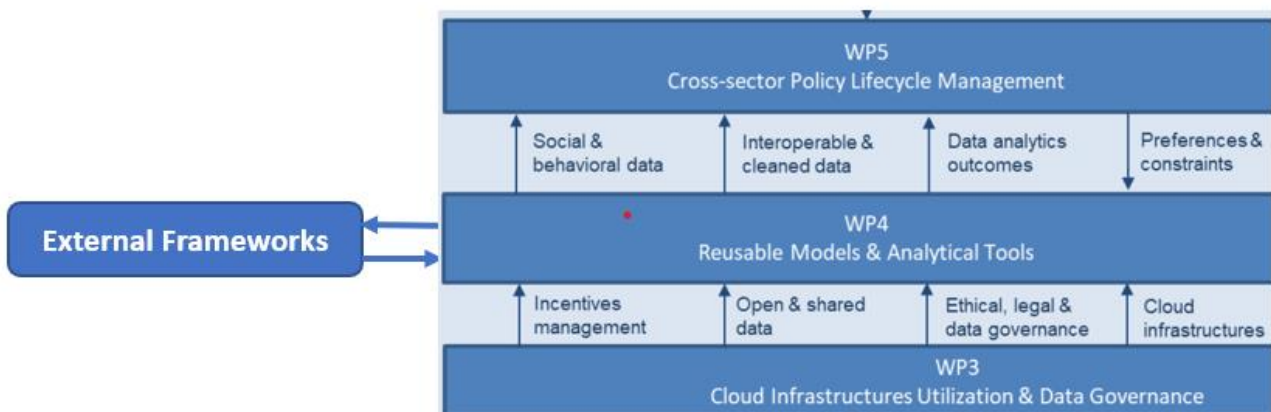


FIGURE 1 - WP4 INTERFACE WITH WP3 AND WP5

2 Data Acquisition and Analytics Layer

2.1 General Architecture and Roles

2.1.1 Updates since D4.3

The Data Acquisition and Analytics layer (DAA) architecture further proved its stability by successfully handling additional Use Case scenarios. Furthermore, we used the very mechanism of analytic function registration and invocation to design and implement a generic architecture for integrating external frameworks within PolicyCLOUD which is one of the main updates of D4.5. This integration has been designed while putting minimal requirements on these frameworks. It has been detailed in subsection 2.2.5 and has been tested with the Politika framework for two Use Cases (see section 4.3.7 and 4.3.8).

2.1.2 Architecture and analytic functions APIs

The DAA is responsible for the orchestration and integration of all the components of the analytic functions, data sources, data repository, as well as providing the external interface for the layer's exploiters – analytic owners, data source owners, and the Policy layer (through which the end user applies desired analytics). Its roles include forming the proper setup of the registered analytic functions and data sources to ensure the correct data ingest process (applying the required ingest-time

analytics/transformation – in both Ingest-now and Streaming modes), (see definitions page 14) and applying the requested analytics on the requested data source. The DAA API Gateway is implemented as a set of serverless functions (in the OpenWhisk cluster), where the state is managed both on the OpenWhisk itself (e.g., analytic functions’ metadata) and the database.

There are two types of functions:

1. Analytics Ingest / Transformation Functions are used to apply initial analytics and/or transformation on the data fusion path of data sources before being stored in PolicyCLOUD backend.
2. Analytics Functions are activated by PolicyCLOUD users through the Policy Layer, to perform an analysis on a specified data source (which has already been ingested / transformed) to provide analytics insights that can be further used in policy decisions.

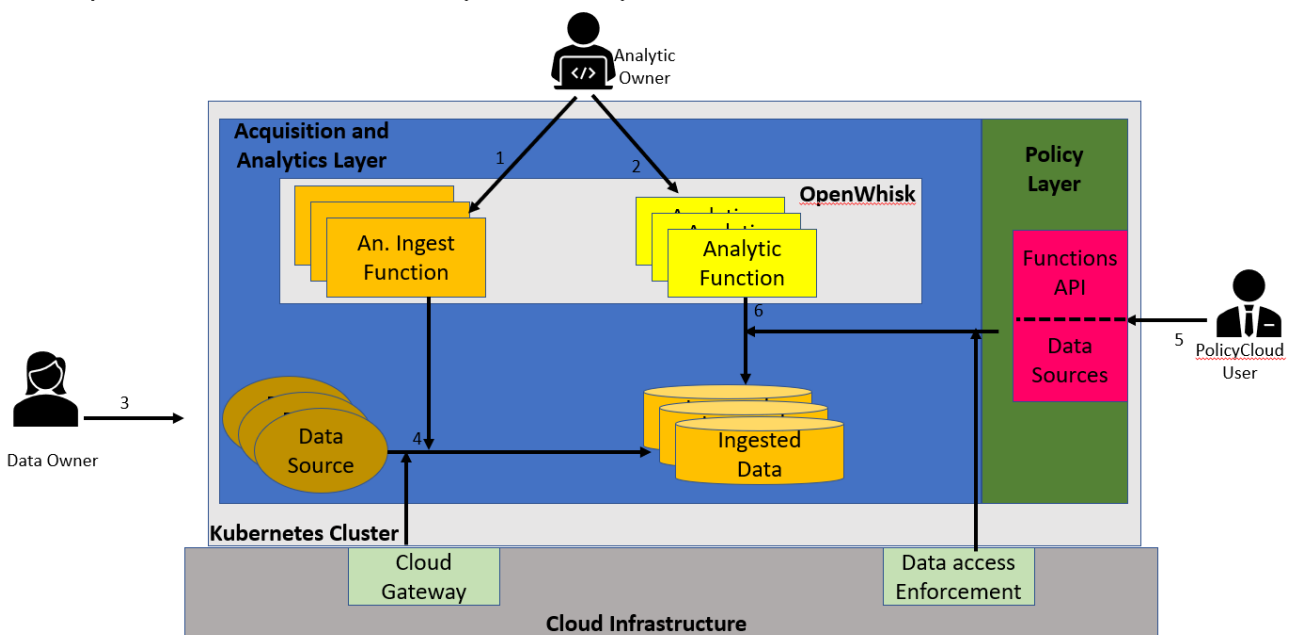


FIGURE 2 – DATA ACQUISITION AND ANALYTIC LAYER

According to the numbers in Figure 2:

1. Analytic Owners are responsible for the functionality of the Ingest Functions which they register on the PolicyCLOUD platform (and, specifically, using the Data Acquisition and Analytics layer API). On Ingest Function registration the following parameters should be provided:
 - a. name: function name
 - b. kind: relates to the function implementation type (e.g., java8, image based, etc.)
 - c. filename: jar filename containing the function code
 - d. main: specify the function main class
 - e. image: image name residing in the container registry service if kind is image (optional)
 - f. function parameters: function parameters (Json)
 - g. type: analytic / analytic-ingest

- h. category: Data-Cleaning / Data-Interoperability / Situational-Knowledge / Opinion-Mining / Social-Dynamic
- i. description: function description and usage
- j. Expected input data schema/format
- k. purpose: for data access enforcement
- l. biasDoc – this parameter permits, and in fact could even oblige (e.g., by checking that this parameter has a minimal required length and/or that an attachment of a defined format and filesize range is included), the Analytic Owners to input bias management documentation, providing information on the specific measures taken to address the risk of biases inherent to the functioning of the Analytics Ingest Function. It could also permit / oblige Analytic Owners to upload or indicate one or more unit test datasets on which the Analytics Ingest Function can be applied, so as to demonstrate that resulting bias measurements do not exceed a given maximum (in other words, that the specific measures taken to address the risk of biases are functioning effectively, as intended);
- m. tradeoffsDoc - this parameter permits, and in fact could even oblige (e.g., by checking that this parameter has a minimal required length and/or that an attachment of a defined format and filesize range is included), the Analytic Owners to input documentation on the management of trade-offs, providing information on the relevant trade-offs encountered, decisions made concerning the balancing of competing requirements (e.g., result precision versus fairness) and measures taken to implement and document those decisions.

PolicyCLOUD administration privileges are required for registration of Analytics Ingest Functions (see Section 2.2.3). Internally, Analytics Ingest Functions are deployed as OpenWhisk¹ functions. Enhanced Interoperability & Data Cleaning (T4.2) are two of the built-in Analytics Ingest Function in PolicyCLOUD.

Each ingest function is expected to read the incoming data from the request's message parameter (JSON string), and to return the transformed message as a JSON string as well. The resulting JSON string might be the input of the next ingest function, or the transformation result written to PolicyCLOUD's backend.

2. Similar responsibilities apply to Analytic Owners regarding the registration of Analytics Functions (see API parameters above (1)).
3. Data Owners are responsible for the registration of Data Sources into the PolicyCLOUD platform. Upon a datasource registration, the relevant data is imported to PolicyCLOUD backend after being transformed by applying a group of pipelined ingest functions on it (e.g., filtering/cleaning), keeping in the PolicyCLOUD data store only extracted knowledge, as opposed to the whole raw data. We have designed PolicyCLOUD to handle the three following different types of datasource registration:

¹ <https://openwhisk.apache.org/>

- a. **Streaming** –data is continuously streamed from one or more external sources (e.g., twitter) into the PolicyCLOUD data store.
- b. **Ingest-now** –external (or local) data which should be ingested into the PolicyCLOUD data store.
- c. **External** – an external data source that can be queried / analysed upon demand, without ingesting data outside of such queries / analyses.

We actually did not experiment with the “External” data source since none of the Use Cases fit this kind of data source.

During the datasource registration, the CloudGateway (T3.3) component is invoked to retrieve the data source content.

On Data Source registration, the following parameters should be provided:

- a. Name
- b. Type (Streaming / Ingest-now / External)
- c. Description
- d. Source specification (streaming details, source data location, source URL, access method, credentials etc.)
- e. Schema / metadata
- f. Permissions – by whom (public / specified user list) and for what purpose the data can be used for (Analytics Functions which are to be applied to a Data Source should have been registered for purposes which are identical or compatible with the purposes identified for that Data Source)
- g. Analytics Ingest Function + parameters if applicable
- h. biasDoc – this parameter permits, and in fact could even oblige (e.g., by checking that this parameter has a minimal required length and/or that an attachment of a defined format and filesize range is included), the Data Owner to input bias management documentation, providing information on the bias detection methods applied to the Data Source, the specific biases identified, and the specific measures taken to address any such biases.
- i. GDPRDoc – this parameter permits, and in fact could even oblige (e.g., by checking that this parameter has a minimal required length and/or that an attachment of a defined format and filesize range is included), the Data Owners to input privacy / data protection management documentation which, in particular, should indicate whether any personal data is included within the Data Source and, if so, the measures taken to ensure that the Data Source can be ingested onto the platform in compliance with the applicable data protection and regulatory framework related to the processing of personal data, as defined by the GDPR and other applicable data protection laws (e.g., to ensure compliance with the principles of lawfulness, transparency and purpose limitation under the GDPR, considering the purposes for which the personal data within the Data Source were originally collected and the purposes for which those personal data will be further processed via the platform).

- j. authDoc – this parameter, permits, and in fact could even oblige (e.g., by checking that this parameter has a minimal required length and/or that an attachment of a defined format and filesize range is included), the Data Owners to input documentation to demonstrate that the registration of the Data Source has been authorised by relevant rightsholders (or that such authorisation is not required under applicable laws).
4. The data of a registered Data Source is ingested into PolicyCLOUD according to the Data Source type: as can be seen in Figure 3, for Streaming Data Sources, the registered Analytics Ingest Function(s) is triggered whenever new data is available; for Ingest-now Data Sources, data is

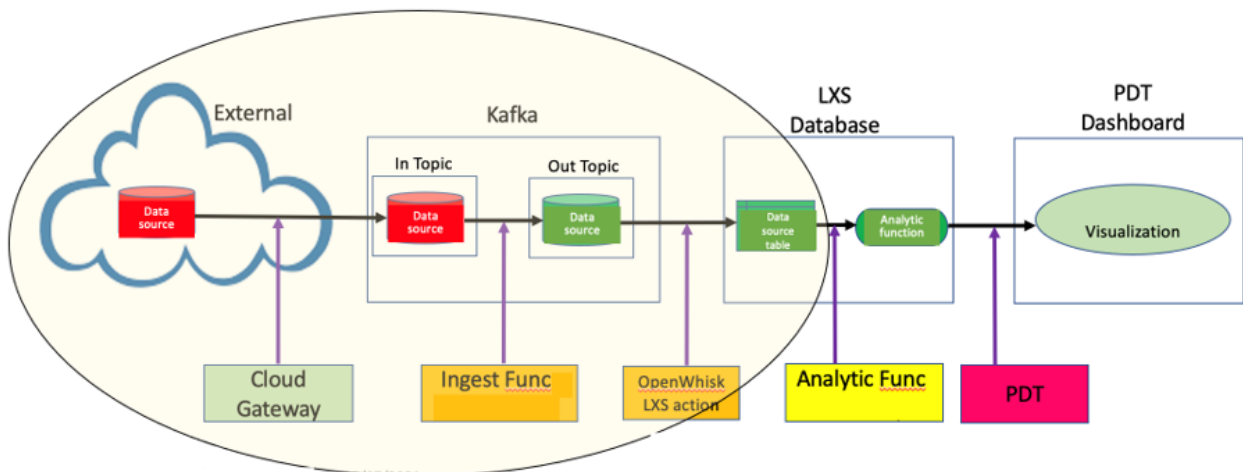


FIGURE 3 - DATA SOURCE PATH

- ingested upon registration and processed by the registered Analytics Ingest Function(s); and for External Data Sources, data is read (and possibly processed by registered Analytics Ingest Function(s)) at the direct request of users for analytics.
5. PolicyCLOUD users can list, through the platform’s dashboard, registered Data Sources and Analytics Functions together with their metadata, description and required parameters. PolicyCLOUD users can apply a registered Analytics Function to a given Data Source. Validation for this application is performed according to the user’s credentials, the permissions registered for the Data Source, and the defined purpose(s) registered for the Analytics Function (as noted, Analytics Functions which are to be applied to a Data Source should have been registered for purposes which are identical or compatible with the purposes identified for that Data Source).
 6. The Policy Layer interacts with the Data Acquisition and Analytics layer to retrieve the registered Data Sources and Analytic Functions, and to apply the selected Analytics Function to the selected Data Source while enforcing permissions.

On applying analytic function on datasource, the following parameters should be provided:

- UserCredentials: (TBD by the enforcement point)
- Function: Analytic function name
- Data-source: data source name

2.2 Extensibility and Reusability

2.2.1 Updates since D4.3

Section 2.2.5 was added as of D4.5. It describes the integration of external frameworks.

2.2.2 Data Source Registration

PolicyCLOUD administration privilege is required for registering a data source. Internally, the data source details are saved in the PolicyCLOUD store, CloudGateway is called to pull the datasource data, kafka topics are created for buffering and interaction with the CloudGateway, OpenWhisk rule and trigger are deployed for activation of the specified Analytic-ingest functions whenever data is available (e.g., on a Kafka topic used as a connector) and according to the data source type also the following:

- a. **Streaming, Ingest-now** – At the time of the user request all the data was already ingested to the PolicyCLOUD store. Over time, the data might reside partially in the LeanXcale database and partially in the object storage, but this is transparent to the Analytic Function (with the seamless component).
- b. **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 may be pushed down to the external datastore provider and the PolicyCLOUD platform only retrieves the result of this pre-processing, thus never having actual access to the data set itself.

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

2.2.3 Analytic Function Registration

PolicyCLOUD administration privileges are required for the registration of Analytics Ingest Function(s) or Analytics Function(s). Internally, these functions are deployed as OpenWhisk functions. Enhanced Interoperability & Data Cleaning (Task T4.2), Situational Knowledge Acquisition & Analytics (Task T4.3), Opinion Mining & Sentiment Analysis (T4.4.) contributed built-in Analytics Functions in PolicyCLOUD. Concerning Social Dynamics & Behavioral Data Analytics (Task T4.5), please refer to subsection 2.2.5 which details the integration of external frameworks within the PolicyCLOUD framework.

Analytic functions are developed by Analytic owners and can be implemented either in python or in java or node.js and should follow Openwhisk action implementation guidelines.

DAA includes a common gitlab structure for functions' code registry and common container registry for functions' docker images with all required dependencies.

Files and images are automatically pulled from the PolicyCLOUD gitlab service during function registration to create the serverless function.

The registration scheme of an Analytics Function includes a list of parameters that must be specified. More details can be found at the D5.4 deliverable. Figure 4 gives an illustration of an attached registration information:

```
{
  "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": [
    {
      "valueConstraints": [
        {
          "value": [],
          "constraintType": "DEFAULT"
        }
      ],
      "unit": null,
      "valueType": "ENUM",
      "name": "fields",
      "description": "field to be used for the aggregation",
      "id": "1888df82-843a-406a-b78c-elcddb68cc41",
      "evaluationTypesAllowed": [
        "RANGE"
      ]
    }
  ]
}
```

FIGURE 4 – EXAMPLE OF A FUNCTION REGISTRATION

2.2.4 Analytics Function Invocation

The Data Acquisition and Analytics layer provides an API to list all the registered Analytics Functions with their registration parameters, allowing to invoke a selected Function on a selected Data Source.

2.2.5 Integration of external frameworks

2.2.5.1 INTRODUCTION

For various reasons external frameworks may not be fully integrated within the PolicyCLOUD framework.

For instance, too much effort may be required to port an existing framework to the serverless paradigm. Sometimes also the serverless paradigm is not well suited to the framework.

In these cases, it is still possible, as we will show in this subsection, to integrate the external framework to PolicyCLOUD.

We will take as an example the Politika framework (see section 4.3) for which quite a big effort would have been required to rewrite as serverless functions, not speaking of the fact that Politika comprises important interactive components which are not well suited to be ported to PolicyCLOUD.

Thus, we decided to use Politika as an external framework and we used in this way Politika for two different use cases (for simulations of parameters which influence the marketing of a given Aragon wine versus a main competitor wine and also for simulating the effect of certain policies for the terrorism use case).

The Politika framework (see section 4.3 for details) offers a GUI interface which permits the creation of simulation models. The process of creating a Social Dynamics model is typically an interactive process which does not fit with the PolicyCLOUD architecture. Moreover, this activity is typically not handled by policy makers but rather by Social Dynamics specialists therefore there was no advantage at integrating this process within PolicyCLOUD.

However, once a model has been created, simulations can be invoked by policy makers from PolicyCLOUD who will provide the values of the input parameters for the specific simulations to be executed by Politika.

In order to facilitate the integration with any external analytical framework such as Politika, we have developed during this last phase of the project a generic function that can act as the bridge between the PolicyCLOUD platform and the external analytical framework. In this way, the PolicyCLOUD platform can interact with this function in the same manner as with any other native functions, such as the Situational Knowledge Acquisition & Analytics (Task T4.3) or the Opinion Mining & Sentiment Analysis (T4.4).

The only requirement for the external framework is to implement a REST API which exposes the functionalities required by the end-users of the PolicyCLOUD. The REST web methods are invoked by the generic bridge function to allow the interaction between the two platforms. The generic bridge function invokes the REST web method by providing a list of values of the input parameters that have been previously defined by the provider of the external analytical framework. In the case of Politika, after receiving these input parameters, it feeds them to run the simulation, and upon completion returns a list of output values that are stored similarly to non-external analytics invocations in the datastore that is accessible from PDT of the PolicyCLOUD, by the generic bridge function. This permits the result of Politika's simulation to be visualized using the PDT graph dashboards.

The advantages of enabling the invocation of these external frameworks from PolicyCLOUD is twofold, for both the external framework and the PolicyCLOUD perspective:

First, it extends the built-in functionality of PolicyCLOUD, by enhancing its extensibility since it can now easily hook to external frameworks. This is totally aligned with the major objectives of the project, the extensibility of the platform and the realization of reusable models and analytical tools. Moreover, the policy maker benefits from a single framework from which he/she can invoke external tools such as those offered by Politika, without having to navigate to different UIs.

Secondly, integrating external analytical frameworks to PolicyCLOUD enables them to benefit from all the built-in visualization tools of PolicyCLOUD, as well as from its user community, thus potentially increasing their visibility. Also, joint exploitation paths can be further investigated, as part of the activities

of T71 (“Market Analysis, Business Scenarios & Exploitation) and T7.5 (“Roadmapping & Impact on Adaption”).

2.2.5.2 ARCHITECTURE

We used the Politika framework as the test case for the generic way of integrating PolicyCLOUD with external analytical framework, our goals during the design phase were first to re-use as much as possible the PolicyCLOUD framework capabilities and even more importantly to design a generic architecture that will fit a wide range of external tools.

The sole requirement from the external framework is to implement a REST API which permits:

1. To pass to the framework the input parameters as a dictionary of values, that is a list of pairs where each pair specifies the parameter name and its value. We relied on our data model for defining the input parameters for the analytical functions so that the PolicyCLOUD can interact with the external framework in the same way as it is currently interacting with its regular analytical tools. The model for defining this input is general enough to specify to the external framework any type of parameters and evaluation: string, numeric values, dates, list of key/value pairs etc. In the case of Politika, at registration time, the tool provider defines the input parameters expected from the end user, such as the type of the required service (i.e., simulation) that should be run, the model, version and the additional input parameters.
2. To return to the invoking party the results of the function invocation (using the JSON format).

It is important to understand that when invoking an external framework from PolicyCLOUD, the datasets on which the framework is applied (if relevant) are also external to the PolicyCLOUD framework. Indeed, it would not make much sense to require from an external framework to import from PolicyCLOUD a large quantity of data. Moreover, due to privacy constraints, it might not be permitted for the external frameworks to provide their raw data to a third party, in our case the PolicyCLOUD platform. Specifically, in the case of Politika, no dataset is needed to perform simulation.

Before we describe how an invocation from PolicyCLOUD of a service offered by an external framework is performed from start to end, let’s understand how the communication between the PolicyCLOUD framework and the external framework is done: As a preliminary step, an analytic function whose goal is to act as the bridge between the two frameworks is registered through the DAA. During this registration, the registrant party specifies the format of the input to the function from the PDT, along with some configuration arguments such as the URL of the REST entry point of the external framework. This follows the same procedure as for the registration of regular analytic functions as detailed in 2.2.3. Since the input parameters are defined during the registration, this automatically specifies to the PDT the frontend GUI that should be presented to the invoker of the external framework, typically the policy maker (i.e., visualize a dropdown with the list of values allowed for a given parameter, draw a calendar in case a parameter is a date type, etc.).

As with any other analytical tool, the parameters are of two categories: the function parameters, which are visualized to the end-user and correspond to the input needed by the external framework, and the configuration arguments that are used internally by the analytical bridge function. As a result, upon

invocation, the typical input, constructed by the PDT backend and provided by the DAA layer to this bridge analytical function is:

From the configuration parameters, the URL of the external framework is retrieved and more specifically its REST API entry point along with the type of invocation (i.e., GET, POST etc.). The configuration arguments are followed by the function parameters, which in the case of Politika is a dictionary of values built as follows:

For each dictionary there is a strictly positive natural number p such that each pair consists of a parameter name and an ordered list of p values associated with the parameter name.

The first column of values may define the “base case” while each of the other columns defines an alternative policy that should be simulated and compared to the base case.

In the following list, we detail the various stages of the invocation of an external framework from PolicyCLOUD which are also depicted in Figure 5.

1. As a first step, the policy maker enters at the PDT the input, as defined by the external framework provider during the registration phase. In the case of Politika, this is a dictionary of key/value pairs, but our implementation is generic to allow for any type of parameters that might be needed by any tool.
2. The policy maker invokes the corresponding function by for instance pressing a “run” button
3. The PDT front-end informs the backend that an invocation of a specific function needs to take place with a given user input.
4. The backend constructs the JSON body with all parameters and interacts with the DAA layer to deploy and execute the bridge function into the Openwhisk serverless platform passing it the entered parameters.
5. The DAA uses Openwhisk to execute the requested function, providing it the input parameters given by the end-user, along with all its related configuration arguments that had been defined during the registration process.
6. In the case of an external analytical framework, the bridge function retrieves the URL endpoint of the REST interface of the framework, and synchronously invokes this REST entry point passing it all the parameters defined by the end-user during the runtime.
7. The external framework upon invocation runs the specified service using the passed parameters. Multiple invocations of the services are possible if so, specified by the parameters.
8. The external framework sends back to the invoking bridge function the results of the service invocation(s)
9. The bridge function receives the answers and stores them within the database, using the PDT REST API, as for regular DAA analytic functions.
10. After the PDT backend persistently stores the output results, it notifies the frontend via messages sent through websockets.
11. The PDT front end after having been signaled invokes the visualization function that fits the received answers, which the policy maker can now analyze.

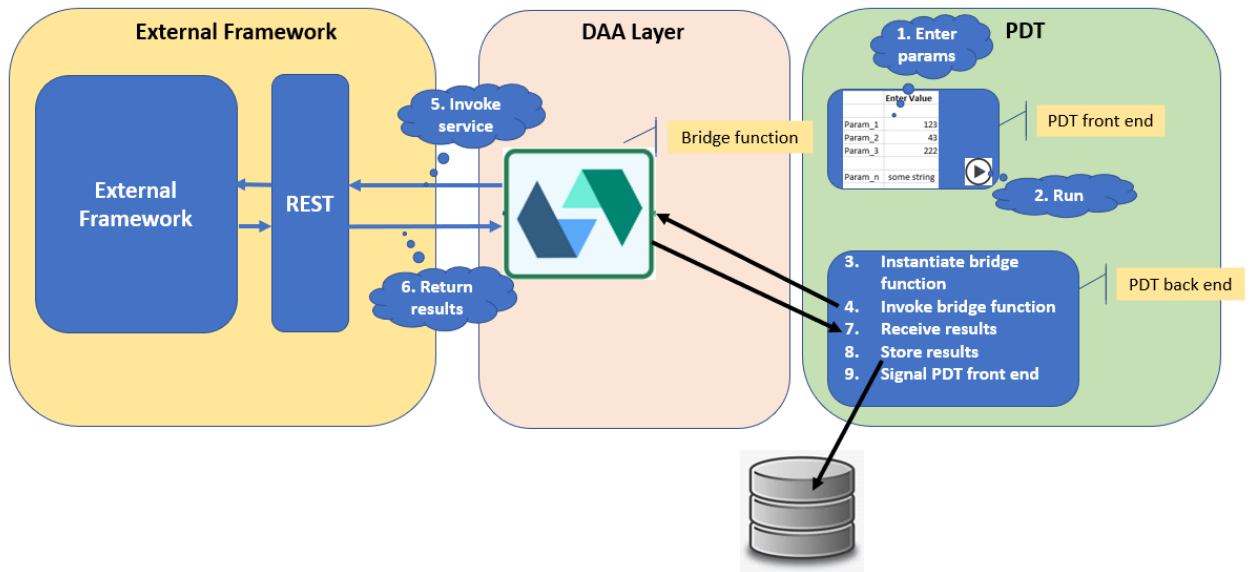


FIGURE 5 - STEPS OF AN INVOCATION OF AN EXTERNAL FRAMEWORK

These steps are identical with the sequence flow of the invocation and execution of a function from the PDT to the serverless platform, as it has been described in deliverable D5.6. The only difference is that our novel bridge function, instead of being executed locally and consuming data from the PolicyCLOUD datastore, communicates with the external analytical framework via its REST endpoints, delegating to it the execution of the desired function and finally retrieves the results in the body of the HTTP response of the REST API.

2.3 Security and Legal concerns

2.3.1 Updates since D4.3

The main updates relevant to security and legal concerns since D4.3 are the measures taken to further consolidate the WP4 Legal/Ethical Checklist, which was last updated in D3.6 [1].

2.3.2 Ethical, legal, regulatory, and societal considerations

As last reported in D3.6 [1], the WP4 Legal/Ethical Checklist maps a refined set of legal, regulatory, ethical and societal requirements applicable to the activities which fall under WP4's scope. For reference, we report the latest update to the WP4 Legal/Ethical Checklist below, which now includes the following set of controls of relevance (updated from the version reported in D3.6, as of the date of this Deliverable):

Control No.	Recommendation / Examples of activities concerned / further explanation	Implementation Status	Notes
1	<p>The platform's analytic components (including analytic-ingest components) must be covered by an appropriate initial and routine training and testing protocol/program.</p> <p>This protocol/program must aim to mitigate the risk that the components may skew knowledge obtained from data (output) in a biased or otherwise erroneous manner. It must also aim to ensure a reasonable degree of statistical accuracy for output generated (e.g., in the case of Enhanced Interoperability, for annotations and connections established).</p>	Ongoing	<p>The specific Consortium members responsible for each relevant tool have been engaged to provide information about the bias/trade-off management measures applicable (or to be applied, where relevant) to the Analytic Functions for which they are responsible. This has been achieved through two lists of questions regarding which they have been prompted to provide input. Initial input has been provided by all relevant Consortium members, which is now under revision and discussion.</p> <p>Input parameters have been defined in the Analytic Function registration process for Analytic Owners to provide information about the bias/trade-off management measures applicable to the Analytic Function they are seeking to register, for the benefit of PolicyCLOUD platform users. These will be incorporated into a larger "User Guide" meant to explain to Analytic Owners how to address each specific input parameter (please refer to Section 2.2.3 of this Deliverable for more on this).</p>

<p>2</p>	<p>Adequate technical and organisational security measures must be implemented in the components developed by WP4, developed as a result of a dedicated security risk assessment targeting potential threats generated, in particular, from reliance on big data analysis.</p> <p>Specific measures aimed at ensuring data integrity (e.g., prevention of external attacks affecting the integrity of data used as input, the functioning of the analytics components, or the output generated by those components) should be considered in this assessment, as well as specific measures aimed at ensuring data confidentiality/availability (where input datasets may be stored on the platform), and restrictions on reuse of personal data (e.g., hashing or encryption).</p> <p>The ability to detect and appropriately respond to security incidents, including personal data breaches, must also be considered.</p>	<p>Ongoing</p>	<p>Matters related to security must be coordinated at the Project-level - ENISA's (draft) "EUCS – Cloud Services Scheme" [2] has been identified as a relevant standard for the Project, and several controls have been extracted from this standard to serve as a security benchmark for the Project. Different Partners (including WP4 Partners) are being engaged to provide information on the status of implementation of these controls regarding the PolicyCLOUD components for which they are responsible. Where a given control has not been implemented and it is feasible to do so given the technical circumstances applicable to the Project, the Partner(s) responsible for such control will be tasked with this implementation.</p> <p>The collection of information on the implementation of security controls and the monitoring of such implementation will primarily be done through the revision, consolidation and implementation of the WP2 Legal/Ethical Checklist, together with all relevant Partners.</p>
----------	--	----------------	--

<p>3</p>	<p>Mechanisms facilitating the auditability of AI systems (e.g., traceability of the development process, the sourcing of training data, and the logging of the AI system processes, outcomes, and positive and negative impacts) must be implemented.</p> <p>These logs should assist in ensuring traceability of automated decisions (i.e., explaining how an algorithm arrived at a given output from a given input), as well as the presentation of correlations established and their rationale to the end-user for confirmation. This information should allow end-users to validate correlations made to some extent.</p>	<p>Implemented</p>	<p>A standard logging service has been implemented in the platform, as a centralised, PolicyCLOUD project-wide component (please refer to Section 2.3.4 of this Deliverable for more information on this).</p> <p>This logging mechanism has been further developed to ensure that it is also able to record events corresponding to the users' invocation of Analytic Functions regarding specific Data Sources. The generated records/logs can be read in combination with the information provided on each invoked Analytic Function (as provided, e.g., in the function parameters, type, category, description and purpose parameters of the function's registration on the Platform) to allow for the tracking of specific actions/operations performed by an invoked Analytic Function on a given Data Source - in other words, to allow traceability of the process followed by an Analytic Function to convert the input (dataset) to the function's output.</p>
<p>4</p>	<p>Any trade-off between requirements, principles or individual rights considered in AI system development should be properly documented.</p>	<p>Ongoing</p>	<p>See notes to #1.</p>

5	Should the use of a selected data source be subject to restrictions (e.g., contractual terms, database copyright protection, database sui generis right protection, copyright protection), that data source should not be used without appropriate permissions from the relevant owners/rightsholders of that data source.	Ongoing	<p>A requirement has been set for Data Owners seeking to register a Data Source to document measures taken to address applicable legal requirements, through adequate fields added to the registration Application Programming Interfaces (APIs). In particular, specific input parameters to be addressed by Data Owners require them to link information / documentation to confirm and/or demonstrate that the registration of the Data Source has been authorised by relevant rightsholders (or that such authorisation is not required under the EU legal framework), to the Data Source upon registration.</p> <p>An explanation of how to address these input parameters, for the benefit of Data Owners, will be included in a “User Guide” (see Section 2.3.3 of this Deliverable for more on this).</p>
---	--	---------	---

6	<p>The Platform should be designed so as to facilitate the exercise of rights granted by the GDPR or other applicable data protection laws to the relevant data subjects (i.e., individuals whose personal data may be stored within cleaned/ingested datasets).</p> <p>This may include measures such as setting up a dedicated e-mail inbox to receive requests for the exercise of these rights, as well as an internal workflow which allows for adequate management of these requests within the timeframe provided by the GDPR (as a rule, requests must be addressed within one month from receipt).</p> <p>Furthermore, the Platform should be designed so as to not create any relevant technical obstacles to the exercise of these rights. In particular, regarding personal data stored within cleaned/ingested datasets, it should be possible to manipulate those personal data as needed to respond to any of the requests described in the "Data Subject Rights" tab.</p>	Implemented	<p>It has been confirmed that the Platform's data repository allows for the execution of relevant data manipulation abilities (better described in D3.6 [1]) needed to ensure the ability to address requests submitted by data subjects, whose personal data may be included in a given Data Source, to exercise their rights under the GDPR.</p>
7	<p>Platform users shall be limited both from a technical and from a contractual point of view in how they can process personal data which are collected and managed through the PolicyCLOUD platform.</p>	Implemented	<p>A Data Governance and Privacy Enforcement mechanism has been developed for the PolicyCLOUD platform.</p>

8	<p>PolicyCLOUD shall only collect personal data, which is adequate, relevant, and limited to what is necessary in relation to the purposes for which they are processed. As such, for each purpose of processing connected to the Project, it shall identify the minimum amount of personal data needed to fulfill such purpose.</p>	Ongoing	<p>The PolicyCLOUD user is in control of the specific data points which will be cleaned/ingested and stored on the platform - attribute/data point filtering is part of the cleaning process. When registering a Data Source, the Data Owner will be able to specify the attributes which are to be further processed as part of the Data source's parameters. This will allow users to configure the platform so that personal data is not unnecessarily collected - in other words, to either prevent or minimize the collection of personal data (e.g., by refraining from collecting relevant identifiers).</p> <p>A requirement has been set for Data Owners seeking to register a Data Source to document measures taken to address applicable legal requirements, through adequate fields added to the registration Application Programming Interfaces (APIs). In particular, specific input parameters to be addressed by registrants require them to link privacy / data protection management information / documentation to the data source upon registration.</p> <p>An explanation of how to address these input parameters, for the benefit of Data Owners, will be included in a "User Guide" (see Section 2.3.3 of this Deliverable for more on this).</p>
---	--	---------	--

9	<p>End-users and PolicyCLOUD must ensure that appropriate steps are taken to verify the accuracy of any personal data collected, and to maintain those personal data up to date over time.</p>	Ongoing	<p>The transformation and “pre-processing” of data, include activities such as:</p> <ul style="list-style-type: none"> · Data validation, to ensure that the rest of the PolicyCLOUD platform components operates on clean, correct and useful data. This also involves assessing the data against defined constraints (which may be mandatory or optional), as a way to provide further assurances as to the accuracy and consistency of data processed via the platform; · Data cleaning, to correct or remove all data for which validation errors were raised during the validation process, including missing, irregular, unnecessary and inconsistent data. This also involves deleting or replacing data which is not aligned with the defined constraints; · Data verification, to check the data for accuracy and any remaining inconsistencies, after the validation and cleaning processes have been completed. <p>Mandatory and optional data constraints can be defined to configure the parameters under which these data validation, cleaning and verification activities operate. This should afford control to Data Owners and other PolicyCLOUD users over the specific data points of a data source to be registered and leveraged via the PolicyCLOUD platform. This will allow the configuration of the platform so that personal data is not unnecessarily collected or processed, thereby allowing unnecessary personal data to be removed from data sources prior to their further storage and processing via the PolicyCLOUD platform. Furthermore, various libraries are</p>
---	--	---------	---

			<p>exploited in connection with the platform’s data validation, cleaning and verification activities to provide greater assurances of data quality (including accuracy, completeness and lack of errors).</p> <p>An explanation of how to address this matter through the input parameters presented to Data Owners upon registration of a Data Source will be included in a “User Guide” (see Section 2.3.3 of this Deliverable for more on this).</p>
10	End-users should also be advised of the possibility of false positive/negative correlations, so that they are incentivised to verify the validity of correlations made.	Ongoing	Disclaimers are to be presented to PDT users to emphasise that output presented to them may not be 100% reflective of the underlying reality (due to the risk of false positive/negative correlations), thereby requiring users to exercise their own critical judgment in interpreting output and using it to inform policymaking.

TABLE 1: UPDATED WP4 LEGAL/ETHICAL CHECKLIST

The main measures taken to progress with the implementation of the WP4 Legal/Ethical Checklist are described in the below sections.

2.3.3 Bias and Trade-off Management for Analytic Functions

In particular, the AI Fairness 360 framework [3] on the matter of bias and trade-off management concerning the built-in Analytic Functions (and Analytic Ingest / Transformation Functions) within the PolicyCLOUD platform, as well as those functions which may be registered on the platform in the future (following the process described in Section 2.2.3 of this Deliverable), two lists of questions have been developed to serve as a guide for Analytic Owners to understand the type of information which should be provided to address the “biasDoc” and “tradeoffsDoc” parameters (described in Section 2.1.2 of this Deliverable) in a manner which is complete and effective towards future platform users. In other words, these lists of questions seek to guide Analytic Owners on how to explain the steps taken to mitigate the risk that their Analytic Functions may skew knowledge obtained from data (output) in a biased or otherwise erroneous manner, and to ensure that any trade-off between requirements, principles or individual rights considered in the development of their Analytic Functions have been duly documented. These lists of questions are provided below:

“biasDoc” (Questions around Analytic Function bias management)

1. Have you carried out any assessment of the possible limitations of your Analytic Function in achieving fair/unbiased results, prior to its registration on the Platform?
 - a. If not, please explain why not (and move to #3);
 - b. If so, please move to #2.

2. In connection with this assessment and the design of your Analytic Function, did you identify any potential (unfair) biases that may be generated/reinforced by the use of your Analytic Function?
 - a. If not, please elaborate on the conclusions reached within your assessment.
 - b. If so:
 - i. Did you rely on an adequate working definition of “fairness”? In particular:
 1. Is your definition of “fairness” commonly used?
 2. Did you consider other definitions before choosing this one?
 3. Did you ensure a quantitative analysis or metrics to measure and test the applied definition of fairness?
 - ii. Did you assess the likelihood that your analytic tool may produce biased/unfair results?
 - iii. Have you made any assessment as to the circumstances under which your analytic tool is most likely to lead to biased results (e.g., type of dataset used, purposes for which the tool is implemented)?
 - iv. Did you consider and document the possible causes for your analytic tool to produce biased results?
 - v. Did you identify who (users/third parties/others) or what (e.g., environment/society at large) may be impacted the most (both directly and indirectly) by such biased results?
 - vi. Did you identify and estimate the likely impact/potential damage and/or harm that biased results generated using your analytic tool may cause?
 - vii. Did you identify the thresholds under which a potentially biased result is acceptable (e.g., an acceptable margin of error)?
 - viii. In conducting the assessment described in response to the previous questions, have you considered potential discrepancies between the development environment and expected deployment contexts?
 - ix. Did you ensure that the identification and the assessment of potential biases (as explained in response to the previous questions) has been duly documented?
 - x. Have you implemented measures to mitigate the risk of generating (unfair) biased results?
 1. If not, please explain why not;
 2. If so:
 - a. Which measures have been implemented to detect and prevent the risk of biased results?
 - b. Which measures have been implemented to monitor that the analytic tool is meeting its goal/purpose and intended application?
 - c. Have you tested the effectiveness of such measures in detecting and preventing the risk of biased results?
 - i. If not, please explain why not;
 - ii. If so, please describe the testing process carried out.

 3. Is your Analytic Function based on an AI model?
 - a. If not, skip to #4;

- b. If so, is this an AI model that (a) has been trained using a corpus of data sets, or (b) is based on the detection of specific combinations of patterns in data sets?
 - i. If not, skip to #4;
 - ii. If so, please address the following questions:
 1. If (a) applies, is the corpus available for inspection and has it been checked for specific biases?
 2. If (b) applies, are these combinations of patterns available for inspection and are they representative of the datasets to which they are applied?
 - a. If so, what is the result of this assessment and what measures have been implemented to limit biases (where detected) ?;
 - b. If not, did you test your AI model for biases with any other tool?
 - i. If not, please explain why not;
 - ii. If so, which other tool did you use and what is the result of the assessment conducted by means of that tool?
4. Did you establish a process for monitoring/reviewing/flagging possible limitations of the Analytic Function in achieving fair/unbiased results?
5. Did you establish a process for third parties (e.g., end users) to report (potential) biases that may result from the use of the Analytic Function?

“tradeoffsDoc” (Questions around Analytic Function trade-off management)

1. In designing and developing your analytic tool, did you identify interests and values relevant to its functioning (e.g., privacy of personal data, accuracy of results, security of data)?
2. Did you assess whether any actual and/or potential trade-offs between those interests and values (e.g., privacy vs accuracy, accuracy vs fairness, explainability vs security) were relevant to the design/development/use of your analytic tool?
 - a. If not, please explain why not;
 - b. If so:
 - i. Which methodology did you use for identifying and quantifying such trade-offs?
 - ii. Which methodology did you use for assessing trade-offs, including the impact that such trade-offs may have on data subjects?
 - iii. Have you considered the existence of technical approaches that may be implemented to minimize trade-offs (considering the level of investment and effort that those approaches, where available, would require)?
 - iv. How did you decide on those trade-offs and which considerations (e.g., benefits vs potential harms) have been taken into account in striking a balance between the competing interests and values at stake? Please describe:
 1. The prioritization criteria and the rationale for the final decision.
 2. The main steps of the approval process that has led to the final decision.
 - v. In conducting the assessment above, have you considered potential discrepancies between the development environment and expected deployment contexts?
 - vi. Did you ensure that trade-off decisions and the approval process that led to those decisions are duly documented?

3. Did you establish a process for monitoring/reviewing/flagging trade-offs on a regular basis?
4. Did you establish a process for third parties (e.g., end users, affected individuals) to report (potential) trade-offs in the use of the analytic tool?

These questions may be further refined and will ultimately be included in a broader “User Guide”, to be developed before the end of the Project. This Guide will seek to instruct Analytic Owners on how to address each of the input parameters described in Section 2.1.2 of this Deliverable when submitting an Analytic Function for registration (and also Data Owners regarding the registration of Data Sources). At this stage of the Project, the registration of Analytic Functions and Data Sources is carried out by PolicyCLOUD’s system administrators on the “backend” of the platform, as there is no user interface available for this registration at present); however, once such a user interface is developed (which may occur at a commercialisation phase for PolicyCLOUD after the Project’s completion), this “User Guide” will be made available to users through that interface (e.g., as a downloadable PDF or embedded in the interface itself).

These questions have also been presented to the Task Leaders responsible for the built-in Analytic Functions, to document the approaches taken on bias/trade-off management for such functions. Some follow-up clarifications have been requested of the Task Leaders following the receipt of their answers, which are currently under review. The final documented results for each built-in Analytic Function will be reported in the upcoming Deliverable D3.9, as part of the final update to the WP4 Legal/Ethical Checklist.

2.3.4 Logging Service

As noted in Section 2.3.2 of D3.6 [1], it is essential for the PolicyCLOUD platform to incorporate adequate technical and organisational security measures, developed because of a dedicated security risk assessment targeting potential threats. To this extent, a secure logging service is critical for security. Among other usages, logging permits to identify security threats, to analyse suspected incidents (forensic analysis), to monitor policy violations and to provide information for unusual conditions. Furthermore, logging is also critical for debugging both performance and functional problems.

Logging further serves to ensure appropriate auditability of the AI-based functions (such as Analytics Ingest Functions and Analytics Functions) which are executed on the PolicyCLOUD platform – in other words, “the ability of an AI system to undergo the assessment of the system’s algorithms, data and design processes” [4]. This serves to identify further legal, regulatory, ethical and societal requirements first identified in Section 2.1.4 of D3.3 [5], notably, “Mechanisms facilitating the auditability of AI systems (e.g., traceability of the development process, the sourcing of training data, and the logging of the AI system processes, outcomes, and positive and negative impacts) should be put in place”.

During this phase of the project, we have now implemented a standard logging service by deploying logstash², arguably the most popular open-source log management tool, on PolicyCLOUD’s Kubernetes infrastructure. The logging service is accessible to all PolicyCLOUD WPs using an HTTP API, and forms a

² open server-side data processing pipeline <https://www.elastic.co/logstash/>

centralized, project-wide infrastructure component. Each logging event is stored in a persistent file system enabled by NFS and ensures resiliency to pod failures and restarts.

For each PolicyCLOUD logging event, this service creates an additional entry in the log file(s). The list of fields that can be logged contains common basic fields, and additional fields which may be modified to fit the message created, so each message type might contain additional different fields. Common basic fields contain: wp, function, type, name, user, message, level.

As part of the Project's efforts to harmonize the approach to data security around relevant standards, this system may be further assessed where needed to ensure that it is aligned (to the extent feasible) with the applicable requirements defined in ENISA's (draft) "EUCS – Cloud Services Scheme" [2] – in particular, **OPS-10 to 16**.

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 opposed to storing 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.

Figure 6 provides an example for initial analytics on a Streaming Data Source scenario:

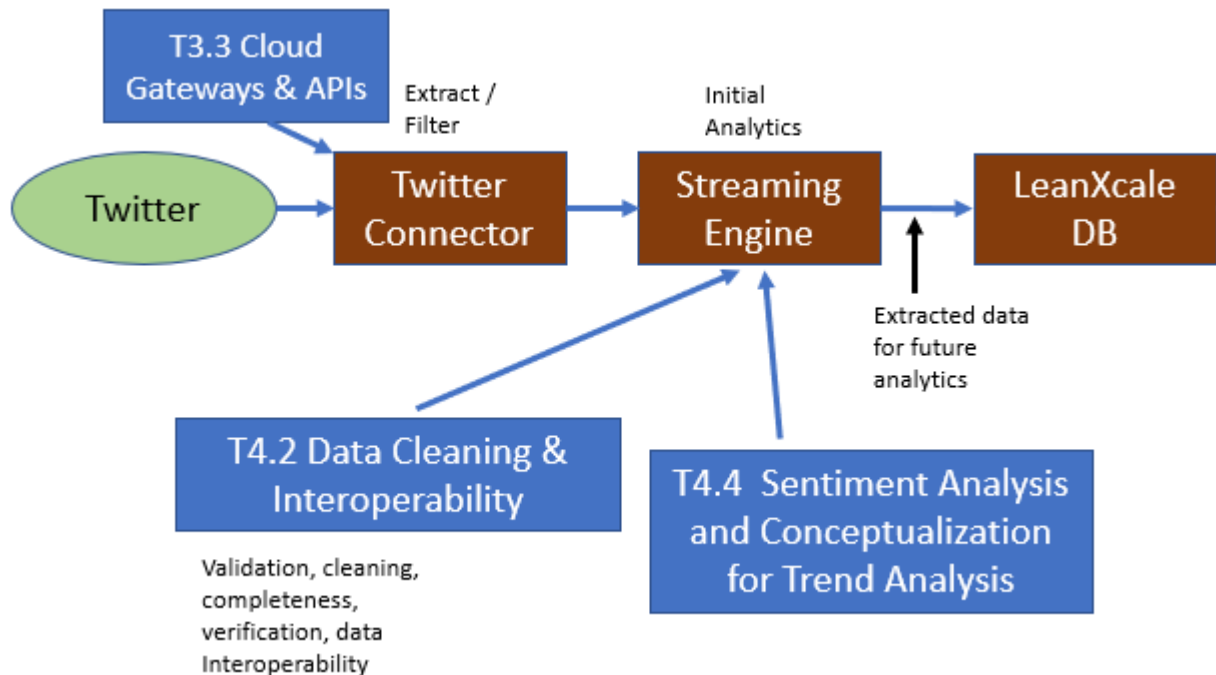


FIGURE 6 - THE STREAMING DATA PATH

In this example, a social network Data Source (Twitter) has been registered, and a Cloud Gateway connector (provided by Task T3.3) with the specified filtering ingests data in streaming fashion. Alternative options were considered for the Streaming Engine (as Spark Streaming), and for Software Prototype 1 we have used a solution based on Kafka sub/pub framework connected to OpenWhisk eventing. These positive experiences of applying this architecture to the use cases of the project during the past two and a half years gave us confidence that this architecture is both sound, flexible and performant. The Data Cleaning & Interoperability Analytic-Ingest Function is invoked whenever new data is ready and applies the specified cleaning and transformations. Optionally, additional Analytic-Ingest Functions can be applied before the transformed data is written to the PolicyCLOUD data repository for further analytics.

3.1 Cloud Gateways (Task T3.3)

The Cloud Gateway and API component offers 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. The Cloud Gateway acts as the only entryway into the PolicyCLOUD project allowing multiple functions and microservices to act cohesively and provide a uniform, gratifying experience to each stakeholder. The provided Gateway API allows 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, by enabling the acquisition of multimodal data from various sources and providers and finally by aggregating the results. PolicyCLOUD's Cloud Gateway and API component supports scalability, high availability, and shared state without compromising performance. Moreover, it supports client-side load balancing, so that the overall system can apply complex balancing strategies, caching and batching, service discovery and handle multiple protocols and implementing adapters for different technologies. On top of this, several mechanisms and microservices are used 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. Moreover, this component describes and maps incoming data into project's internal flexible data schemas for both data in-flight and at-rest. Realizing the value of real time data insights, streaming data ingestion is also supported. Following the Gateway Pattern, it exposes a set of microservices to users via a unified API instead of providing an API for each one. The latter enhances the generalization and applicability of this mechanism in a wider range of domains, scenarios and datasets. What is more, as introduced in D3.5 [6] the Cloud Gateways and APIs component integrates with the project's Interim Repository and will act as the project's dataset repository where each stakeholder and data provider has the ability to upload its own datasets. More details regarding the design and implementation of the Cloud Gateway and APIs component are documented and can be found in D3.1 [7], D3.4 [8], and D3.5 [6] which have been submitted on M08, M20, M22 respectively, and in the last iteration of this series of deliverables D3.7 that is delivered at M32.

Furthermore, authentication mechanisms have been applied at the Gateway level, instead of having to implement authentication mechanisms for each microservice individually. Using federated identity and OpenID connect on OAuth2 protocol, the authentication server can be a separate component or third-party service, but the authentication and authorization process takes place in the Cloud Gateway & API component. In addition, the Attribute-based access control (ABAC) has been selected as the main authentication and authorization model to be utilized. This model uses attributes, rather than roles, to grant user access. Hence, access decisions are made based on attributes (characteristics) of subject or the user making the access request. This enables enhanced flexibility to allow/deny access to critical resources by keeping user attributes up-to-date and propagating changes in real time to the authorization mechanism.

3.1.1 Updates since D4.3

In D4.5 an extra mention and note is given to the integration of the Cloud Gateway component with the project's authentication mechanism and to the implementation of a generalized set of sub-mechanisms, integrations and microservices that enhance the overall generalization and applicability of the Cloud Gateways and APIs component, as well as of the whole PolicyCLOUD platform.

In that context, major changes and enhancements of the Cloud Gateways component were implemented:

1. An integration mechanism with the Interim Repository, as reported in D6.6 Integration Plan, [9] in which data providers have the ability to upload their own files that will be further fetched and processed by the Cloud Gateway's microservices.
2. Two different and generalized sets of microservices (Streaming microservices, Static microservices) were utilized for obtaining data from external data sources along with filtering and mapping mechanisms as suggested by schemas of the project.
3. The integration of Cloud Gateways with the ABAC mechanism. to further enhance the ability and flexibility to manage the access to critical resources of the PolicyCLOUD project.
4. The utilization of the Traefik tool³ which is a popular HTTP reverse proxy and load balancer software. Traefik instead of requiring manual route configuration for each microservice, bundles to the registry service or orchestrator API and generates all routes automatically so that this services to be available for public and ready to use

These improvements and implementations were reported in deliverable D3.7 Cloud Infrastructure Incentives Management and Data Governance Design and Open Specification 3 at section 3. We choose not to replicate the information in this deliverable.

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.

³ <https://traefik.io/traefik/>

3.2.1 Updates since D4.3

3.2.1.1 ENHANCED INTEROPERABILITY

In D4.5 the finalized version of the Enhanced Interoperability component is detailed in. In particular we report on the generalization and wider applicability of Enhanced interoperability. In D4.1 and D4.3 we introduced the initial designed and specified functionalities, whereas in D4.5 we detailed the workflows and pipelines of the Enhanced Interoperability component.

More specifically, the generalized nature of the SemAI hybrid mechanism and approach is presented with specific focus on the internal functionalities and mechanisms that allow this component to be applied in a wider range of various scenarios and heterogeneous data sources. These functionalities have a major impact on the successful aggregation, analysis, and exploitation of data across the whole policy making life cycle within the PolicyCLOUD platform. The latter further enhances the interoperability of the processed data and of the extracted policies.

3.2.1.2 CLEANING MECHANISM

In the previous deliverables (D4.1 and D4.3) we reported the first and the second stable versions of the Data Cleaning component, which had been thoroughly evaluated with diverse use case scenarios. In D4.5 we conclude with the finalized version of the Data Cleaning component, where in essence we have finalized the core rules and constraints that can be used by this component in order to be applicable into different scenarios. Hence, this final version of the component states its generalized nature, being able to support different scenarios (including both the scenarios that are set into the context of the PolicyCLOUD platform, and the scenarios that may be introduced from the external environment of the platform), where data needs to be checked upon missing, irregular, unnecessary, and inconsistent values.

3.2.2 Data Cleaning

The scope of the Data Cleaning sub-component is to deliver the software implementation that provides the assurance that all the provided data coming from several heterogeneous data sources will be clean and complete, to the possible extent. Towards this direction, the Data Cleaning sub-component aims to provide all the processes that 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. In deeper detail, 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 that the ingested information is not repeated, (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 implements the data cleaning workflow providing all the necessary actions towards the aim of consistent datasets across the PolicyCLOUD platform, as depicted in Figure 7:

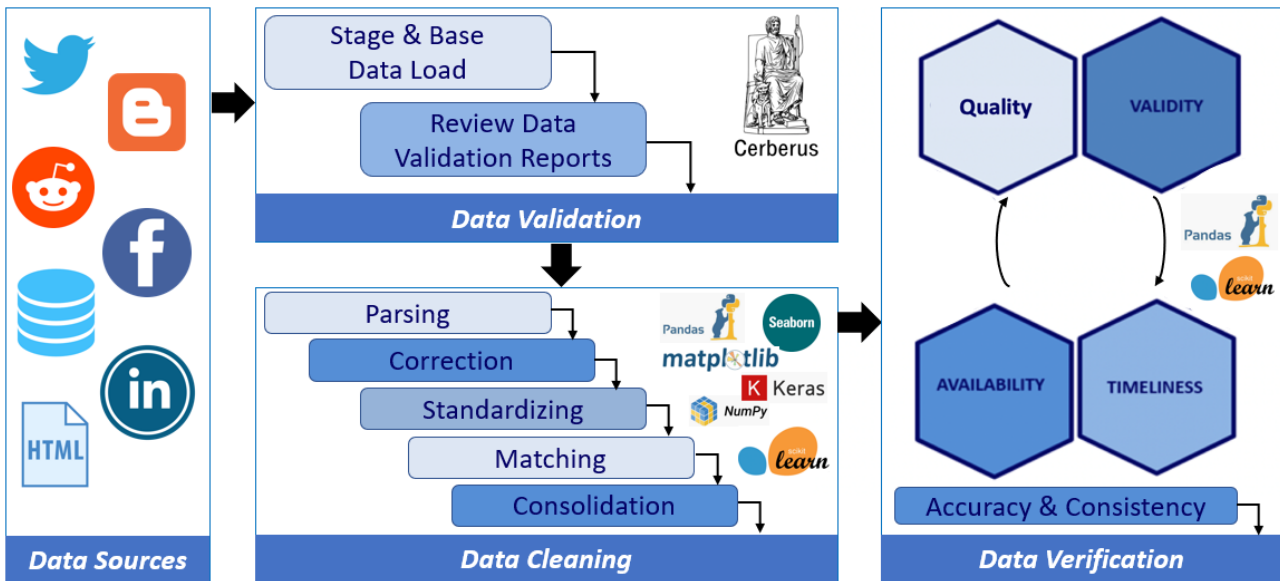


FIGURE 7 – DATA CLEANING WORKFLOW

In deeper detail, the data cleaning workflow consists of three (3) discrete steps, each one of them being provided as an individual service. However, prior to these steps, the initial action is to ingest the data from the various underlying data sources, referring to (i) streaming data that comes from Twitter, Facebook, Instagram, LinkedIn, or Reddit, and (ii) stored data that comes from Webpages, Blogs, or local Data Stores. All this process is achieved through the PolicyCLOUD gateway component. As soon as the Data Cleaning sub-component receives the data from the ingestion process, the ingested dataset domain is identified by discovering and analyzing the semantics of the ingested data, and thus the Data Cleaning actions to be performed are being adjusted. In this context, the Retina API [11] is exploited to convert all the different ingested data into semantic fingerprints, following the paradigm of Semantic Folding of the Retina API. During this task, a data-encoding mechanism is being exploited, for inputting language semantics into Hierarchical Temporal Memory (HTM) networks [12], a theory that describes the human neocortex as a 2D sheet of modular, homologous microcircuits that process any kind of information in a consistent data format called Sparse Distributed Representations (SDRs). In that case, words, sentences, and whole texts are semantically compared to each other, with the whole system only requiring small amounts of training data. Through this, sparse distributed word vectors deriving from the ingested data (semantic fingerprints) are dynamically positioned in a topographical, two-dimensional semantic map in a way that semantically related word vectors are placed close to each other. Thus, in combination with the approach of [13] this process compares the data concepts by overlaying their semantic fingerprints and estimating their distances. As a result, the semantic similarity identification of all the available data is determined, calculating the application domain in which the data belongs to.

Shortly, for the identified application domains in Table 2, the data cleaning process adjusts the cleaning “aggressiveness” accordingly. In the context of PolicyCLOUD, the data cleaning process is able to differentiate between the various pilot domains, including radicalization, winery, and smart cities. Nevertheless, the overall process is trained to also consider the domains of healthcare, finance, industry, security, and education, thus supporting its generalized nature. The overall handling of each of the various domains follows a similar approach for each of the use cases, as described below.

Domain	Cleaning Aggressiveness	Cleaning Actions		
		Missing	Erroneous	Duplicates
Radicalization	Medium	Drop	Transform	Drop
Winery	Medium	Drop	Transform	Drop
Smart Cities	High	Drop	Transform	Drop
Healthcare	High	Drop	Drop	Drop
Financial	Medium	Drop	Predict	Drop
Industrial	Medium	Drop	Predict	Drop
Security	Medium	Drop	Drop	Drop
Educational	Low	Predict	Transform	Drop

TABLE 2: SELF-ADJUSTMENT OF THE DATA CLEANING BASED ON DATASET DOMAIN

After the adjustment of the proposed approach, the rest of the services initiate their tasks:

- Data Validation:** This service 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 with the specified rules. As a result, the Data Validation service performs data validation of the incoming data with the purpose of identifying errors associated with the conformance to a specific set of constraints. In order to successfully complete this process, the Data Validation service exploits the Cerberus library⁴ for implementing all the necessary data validation functionalities, as well as for constructing the needed validation rules. More details regarding the identified data constraints and rules are described in Section 3.2.1.1. It should be mentioned that as illustrated in Figure 7, the Data Validation service is able to process different kinds of incoming data, referring to (i) streaming data that primarily derives from social media platforms (i.e., Twitter, Facebook, Instagram, LinkedIn, and Reddit), and (ii) stored data that comes from Webpages, Blogs, signals and local Data Stores.
- Data Cleaning:** This service 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 performs the necessary corrections or removals of errors identified by the Data Validation Service. Depending on the nature of the identified errors, automated cleaning of the information is 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

⁴ <https://docs.python-cerberus.org/en/stable/>

service exploits the libraries of Pandas⁵, Matplotlib⁶, Keras⁷, NumPy⁸, Scikit-learn⁹, and Seaborn [14] for implementing all the necessary data cleaning functionalities, based on the identified constraints and rules. More details regarding the identified cleaning actions are described in Section 3.2.1.2.

- **Data Verification:** This service 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 ensures that all the corrective actions performed by the Data Cleaning service are executed in compliance with the data models' design in the context of the PolicyCLOUD platform. This service ensures that the data is accurately corrected or completed so that the dataset eventually becomes error free, exploiting the libraries of Pandas and Scikit-learn.

3.2.2.1 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. To this end, it should be noted that in order to extend the overall functionalities of the Data Cleaning to support a generalized concept (i.e., support different kinds of scenarios), the most common constraints have been used as the basis for the generalization part of the component. The following list includes the mandatory and optional data constraints and can be applied in different kinds of scenarios. 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 the given 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)
 - Predefined 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)

⁵ <https://pandas.pydata.org/>

⁶ <https://matplotlib.org/>

⁷ <https://keras.io>

⁸ <https://numpy.org>

⁹ <https://scikit-learn.org/stable>

- 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)

3.2.2.2 CLEANING ACTIONS

For the different data constraints and rules that were described in the previous sub-Section (3.2.1.2), the list of the corresponding possible cleaning (corrective) actions is documented in the current sub-Section. The following list includes the cleaning actions that can be used or combined for the described constraints, and can be applied in various scenarios, following the rules defined in Table 2.

- Deletion of value that does not conform to a constraint by:
 - Dropping a whole entity
 - Dropping a specific attribute
 - Dropping a specific value or a set of values not conforming to cross-field validity (e.g., sum of values not adding up correctly)
- 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

3.2.3 Enhanced Interoperability

Data interoperability is the ability to merge data without losing meaning and is realised as a process that identifies the structural, syntactical, and semantic similarity of data and datasets and turns them into interoperable domain-agnostic ones.

In practice, data is said to be interoperable when it can be easily reused and processed by different applications, allowing different information systems to work together and share data and knowledge.

Hence, Semantic Interoperability is a key enabler for the policy makers as it enhances the exploitation of Big Data and improves data understanding.

This is done by extracting and taking into account parameters and information that were not initially apparent, thus creating efficient and effective policies in terms of good governance. The latter demonstrates the need for the modern stakeholders to implement techniques, mechanisms, and applications that focus their operations on the concept of data interoperability and more specifically on Semantic Interoperability [15], for increasing their performance and enhancing their entire policy making approach.

Mapping and creating interoperable data depends on a method for providing semantic and syntactic interoperability across diverse systems, data sources and datasets. To this end, PolicyCLOUD's Interoperability Component enhances interoperability into PolicyCLOUD project based on data-driven design by the utilization of linked data technologies, such as JSON-LD¹⁰, and standards-based ontologies and vocabularies, coupled with the use of powerful Natural Language Processing (NLP) tasks, in order to improve both semantic and syntactic interoperability of data and datasets.

Through these coupled technologies and methods, the Interoperability Component provides a state-of-the-art approach to achieve interoperability in the 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 extracts semantic knowledge and good quality information from the cleaned data that are the input to its system, as depicted in Figure 8. This knowledge, shaped in a machine-readable way, is 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 [16]. 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 [17]. Thus, by implementing a "JSON-LD context" to add semantic annotations to interoperability component's output [18], the system is able to automatically integrate data from different sources by replacing the context-dependent keys in the JSON output with URIs pointing to semantic vocabularies, that are used to represent and link the data as authors in [19] have introduced. A standard such as this expresses information by connecting data piece by piece and link by link, allowing for any resource (authors, books, publishers, places, reports, municipalities, articles, search queries) to be identified, disambiguated and meaningfully interlinked.

3.2.3.1 LINKED DATA AND SEMANTIC WEB TECHNOLOGIES

Linked Data are designed to support heterogeneous description models, which is necessary to handle divergent and heterogeneous 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 provides 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 heterogeneous sources. A solution to this gives four key requirements for data to be considered as linked data as presented and suggested in [20]:

- Use URIs to identify all types of data items, for example, considering a dataset of technical reports, a unique URI is used for each report.

¹⁰ JSON-LD is a **lightweight Linked Data format** <http://json-ld.org>

- Make the URIs accessible globally by using HTTP URIs. Through this, users can look up the identifiers. The latter helps 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 World Wide Web Consortium (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) [21], which allows for inductive reasoning on properties and the Web Ontology Language (OWL) [22], 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 [18]. 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 provides 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 the Enhanced Interoperability component seeks to deliver structured information, which can be used and queried by a flexible and extensible way to get a better understanding of the provided data and datasets.

3.2.3.2 NATURAL LANGUAGE PROCESSING (NLP)

One of the main goals of the Enhanced Interoperability component is to create a new interoperability framework for linked data. In addition, 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 objects, concepts, places etc.

In that context, the utilization of NLP integrates with Linked Data in order to enhance PolicyCLOUD's interoperability component. Our main focus and goal for using 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, Information Retrieval (IR) and Information Extraction (IE) are core tasks and techniques within the Interoperability Component. NLP through various methods and techniques enhances 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 enhances 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 Named Entity

Recognition tasks enhance the steps of classification and taxonomy that is used during the Ontology Mapping phase, as further analyzed in next Section. Moreover, Text Analysis and processing tasks from the domain of NLP are also 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 implements and enables easily constructed, flexible and high-performance NLP pipelines. To this end, the utilization of Linked Data technologies, analyzed in section 3.2.3 including JSON-LD files, coupled with NLP services helps 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.

3.2.3.3 SEMANTIC ARTIFICIAL INTELLIGENCE (SEM AI)

SemAI addresses the need for interpretable and meaningful data and 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 focus on understanding natural languages and raw texts.

To this end, the combination of Semantic Web Technologies and NLP tasks provides the project a state-of-the-art approach for achieving semantic and syntactic interoperability and enhances the ability of the project to combine structured and unstructured data in multiple ways. Based on this approach raw and unstructured data derived from Twitter can be correlated and integrated with structured data derived from a divergent source. The latter has been applied in two different Use Cases and domains of the PolicyCLOUD project as introduced in D6.11 [23]. More specifically, in Scenario A: Radicalization incidents and Scenario C: Trend analysis of the Use Case 1, as introduced in D6.11 Use Case Scenarios Definition & Design, and in the concepts of radicalization analysis, data derived from Twitter were correlated with data from the Global Terrorism Database (GTD)¹¹, while in the context of Scenario B: Opinion on social networks of the Use Case 2 and agrifood industry, data derived from Twitter were correlated with data from websites and other sources. To this end, the generalized nature and the wider applicability of this component have been further investigated and evaluated both in the sense of data and domain interoperability.

In that context, 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 enhanced the interoperability and the final linking of divergent data and datasets. What is more, to further enhance the generalization of this task an ontology-based NER approach has also been followed and implemented. The ontologies that are used to further train the NER task and finally provide the proper URIs to the identified entities can either be introduced by the policymaker, or widely applied ontologies can be used. The overall sub-mechanism for this enhanced functionality is depicted in Figure 8. The overall functionality of this task is summarized as follows. The NER performs tasks to locate nouns in the sentences based on the output of the spaCy Part-Of-Speech Tagger, then the identified nouns are lemmatized with the lemmatizer available in WordNetLemmatizer

¹¹ Global Terrorism Database <https://start.umd.edu/gtd/>

provided by the nltk library. What is more, SPARQL is used to further annotate the identified and lemmatized nouns with either URIs derived from the given ontology or from widely used ontologies and knowledge bases, such as the Wikidata. Hence, the final recognized entities are linked with specific identifiers and to be further used and analyzed by the analytical component of the PolicyCLOUD project. As a result, this integrated and hybrid approach ultimately leads to a powerful and more Enhanced Interoperability Component for achieving and providing an enhanced interlinking between divergent data and datasets, as in the two above presented Use Cases. In addition to this and based on all the aforementioned functionalities and implementations, the Enhanced Interoperability component can be applied to each new use case and dataset. Thus, to support any newly introduced scenario of the PolicyCLOUD platform. From the early beginnings of the design and specification of this specific component, as introduced in previous deliverable D4.1, to the finalization and the delivery of the 2nd Software prototype, as reported in the context of D4.4, the general utilization of this component was one of the core requirements and prerequisites of its design.

Ontology-based NER

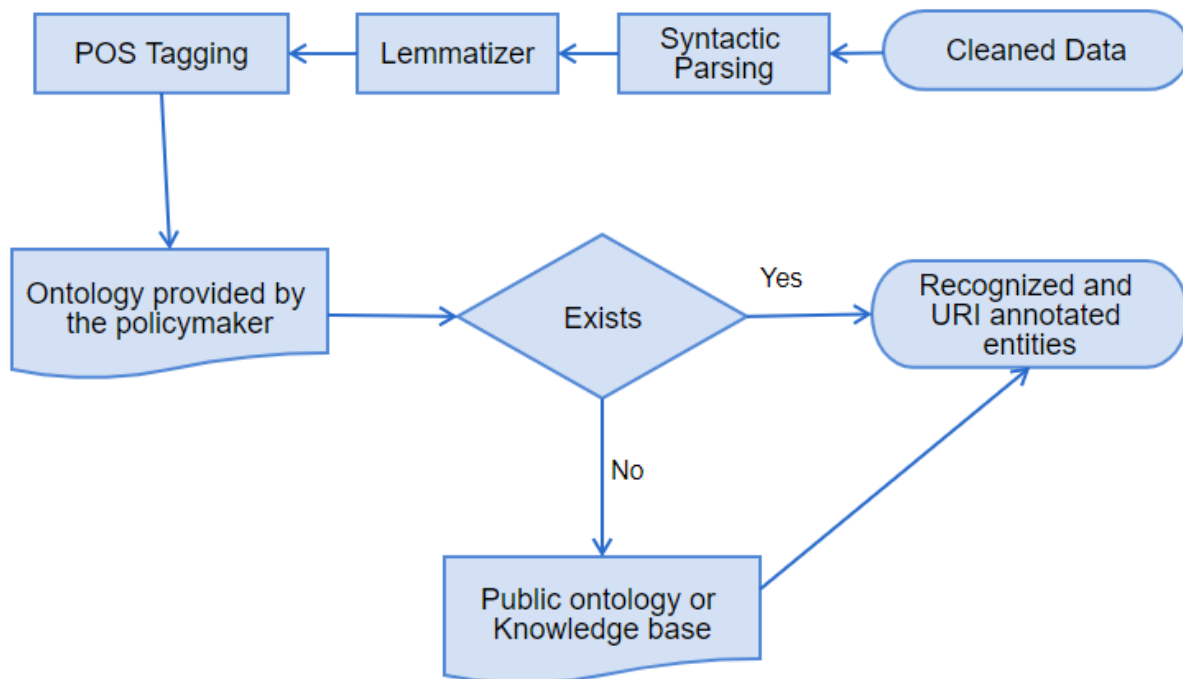


FIGURE 8 - ONTOLOGY-BASED NER

Moreover, this scenario-aware approach 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 help 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 SemAI, in other words the combination of NLP and Semantic Web technologies, provides the capability of dealing with a mixture of structured and unstructured data that is simply not possible using traditional, relational tools.

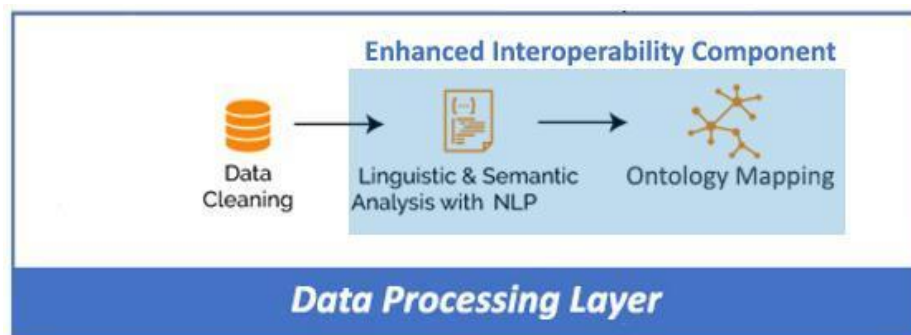


FIGURE 9 – DATA PRECESSING LAYER

As shown in Figure 9 after fetching cleaned data from the Data Cleaning Component, the Enhanced Interoperability Component seeks to implement the hybrid approach of SemAI 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 is analyzed, transformed and annotated with appropriate URI metadata and controlled vocabularies can be identified and designed through the utilization of Semantic Web technologies coupled and enhanced by the utilization of NLP techniques, such as Named Entity Recognition (NER), Part-of-Speech Tagging etc, through the utilization of advanced and multilingual NLP tools such as spaCy¹², nltk¹³ and CoreNLP¹⁴. The selection and utilization of these tools is also aligned with the overall generalization and wider applicability of the Enhanced Interoperability component. The overall design and implementation of this component has been based on the utilization of general and schema-agnostic rules and functionalities, without introducing any constraint in the wider applicability of this component.

The main objective of this subcomponent of SemAI is the identification and recognition of entities, which are further used for interconnection and interlinking with widely used knowledge bases. Moreover, classifying named entities into predefined categories, such as places, organizations, dates etc, enables the identification, design and utilization of proper widely used and controlled vocabularies and standards.

¹² <https://spacy.io/>

¹³ <https://www.nltk.org/>

¹⁴ <https://stanfordnlp.github.io/CoreNLP/>

In addition, the sub-task of Named Entities Linking (NEL) allows the annotation of translated data with URIs pointing either into corresponding widely used and known knowledge databases, such as Wikidata¹⁵ and DBpedia¹⁶, or into predefined and introduced-by-the-policymaker ontologies and knowledge databases, as also highlighted previously. Moreover, an automatic topic identification and dataset classification task ensures that datasets topics of interest are properly identified.

Proper topic analysis is performed to organize and fully understand the large collections of text data and the correlations among them. In the next phase, semantic and syntactic annotated data are interlinked through the task of Ontology Mapping. Ontologies are key factors for enabling interoperability in the Semantic Web [24] 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 [25].

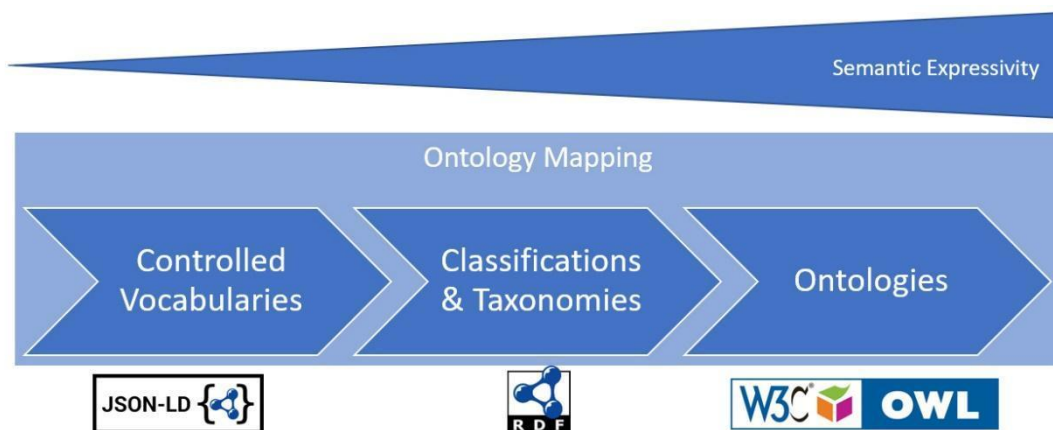


FIGURE 10 – ONTOLOGY MAPPING STEP

Under this scope, an Ontology and Structuring Mapping service is utilized in order to interlink not only URI annotated data with proper ontologies, but also to interlink and correlate datasets among them.

Successful annotation, transformation and mapping of data and corresponding ontologies in terms of semantic and syntactic interoperability of data is one of the key elements of SemAI. To this end, one of the main objectives of the Ontology Mapping subcomponent is to save correlated, annotated and interoperable data in JSON-LD format and as linked ontologies. Hence, the retrieval of semantic facts for the support of the corresponding data schema models is feasible. Moreover, this subcomponent seeks to map concepts, classes, and semantics defined in different ontologies and datasets and to achieve transformation compatibility through extracted metadata. In addition, a data modeling subtask by standard metadata schemas is defined in order to specify the metadata elements that should accompany a dataset within a domain. To this end, semantic models for physical entities / devices (i.e., sensors related to different policy sectors) and online platforms (e.g., social media) are identified.

These models are based on a set of transversal and domain-specific ontologies and could provide a foundation for high-level Semantic Interoperability and rich semantic annotations across policy sectors, online systems and platforms. As shown in Figure 10, there are several levels of structuring before

¹⁵ https://www.wikidata.org/wiki/Wikidata:Main_Page

¹⁶ <https://www.dbpedia.org/>

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 the OWL technology.

On top of this, ontologies are central to the SemAI 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. To this end, the utilization of ontologies under the scope of SemAI hybrid mechanism facilitates communication by providing precise notions that can be used to compose messages (queries, statements) about the policy making domain. At the stakeholders and user level, the ontology helps to understand messages by providing the correct interpretation context.

Thus, ontologies, when shared among stakeholders, improve system interoperability across Information Systems 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. To this end, the utilization of ontologies under the scope of SemAI facilitates communication by providing precise notions that can be used to compose messages (queries, statements) about the policy-making domain. At the 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 information systems in different organizations and domains.

The overall approach that is followed brings together techniques in the domains of modeling, computational linguistics, information retrieval and agent communication to provide an automated mapping method and a prototype mapping system that support the process of ontology mapping to improve and enhance interoperability during the whole data lifecycle in the PolicyCLOUD project.

The novelty of the proposed Ontology Mapping subcomponent is not solely the use of formal application ontologies as an initial mechanism to achieve meaningful Semantic Interoperability, but also the utilization of divergent domain ontologies to support the formal application ontologies mapping process, integrated into an architectural framework.

On top of this, the SemAI hybrid mechanism introduces a multi-layer and hybrid mechanism for Semantic Interoperability across diverse policy related datasets, which facilitates Semantic Interoperability across related datasets both within a single domain and across different policy making domains. This requirement relates to local-regional public administrations and business domains, but also goes beyond the national borders.

Moreover, IT systems and applications interoperability, sharing and re-use, and interlinking of information and policies, within and between domains are essential factors for the delivery of high quality, innovative, and seamless policies. Under this framework, SemAI and its required steps and subcomponents were presented to facilitate its adoption in the data-driven policy making domain. Achieving high levels of Semantic Interoperability in the data can help organizations and businesses to

turn their data into valuable information, add extra value and knowledge to them and finally achieve enhanced policy making through the combination and correlation of several data, datasets, and policies. For instance, the overall functionality of Enhanced Interoperability can be summarised as in Scenario B of Use Case 2. The NER sub mechanism performs tasks to locate nouns in the sentences of a tweet. Afterwards, the identified nouns are lemmatized and further processed. Then, SPARQL is used to further annotate the identified and lemmatized nouns with URIs either derived from the ontology initially specified by the stakeholder or from widely used ontologies and knowledge bases. Hence, the final recognized entities are linked with specific identifiers to be further used and analyzed by the Sentiment Analysis task, as introduced in Section 4.2. The final resulted and annotated tweet is depicted in Figure 11.

```
[{'text': 'It s winewednesday and that calls for bottle of 2017 Embruix Vall Llach [ Garnacha Merlot Syrah Cariñena Cabernet Sauvignon ] from Priorat Spain for only $ 60 Book your reservation below now and we will see you this evening', 'entities': [{'label': 'Garnacha', 'iri': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#AzereGarnacha', 'typeURI': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#RedWine', 'type': 'PRODUCT'}, {'label': 'Merlot', 'iri': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#MerlotGrape', 'typeURI': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#WineGrape', 'type': 'PRODUCT'}, {'label': 'Syrah', 'iri': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#EstecilloSyrahLegado', 'typeURI': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#RedWine', 'type': 'PRODUCT'}, {'label': 'Cariñena', 'iri': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#DO_Cariñena', 'typeURI': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#Region', 'type': 'PRODUCT'}, {'label': 'Cabernet Sauvignon', 'iri': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#CabernetSauvignonGrape', 'typeURI': 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#WineGrape', 'type': 'PRODUCT'}, {'label': 'Priorat', 'iri': 'http://www.wikidata.org/entity/Q15377', 'type': 'GPE'}, {'label': 'Spain', 'iri': 'http://www.wikidata.org/entity/Q29', 'type': 'GPE'}]}
```

FIGURE 11 - ANNOTATED TWEET THROUGH THE UTILIZATION OF ENHANCED INTEROPERABILITY COMPONENT

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 is activated as a serverless OpenWhisk function, directly triggered by the user request. The data access operation is according to the Data Source type:

- **Streaming, Ingest-now** – At the time of the user request all the data was already ingested to the PolicyCLOUD store. Over time, the data might reside partially in the LeanXscale database and partially in the object storage, but this is transparent to the Analytic Function (with the seamless component).
- **External** – for external datasets, data processing requests (e.g., SQL aggregation queries) is pushed down to the external datastore provider and the PolicyCLOUD platform only retrieves the result of this pre-processing, thus never having complete access to the data set itself. Figure 12 continues the example for the Streaming Data Source scenario for the regular analytic applied on the data that is already ingested (and cleaned, transformed etc.).

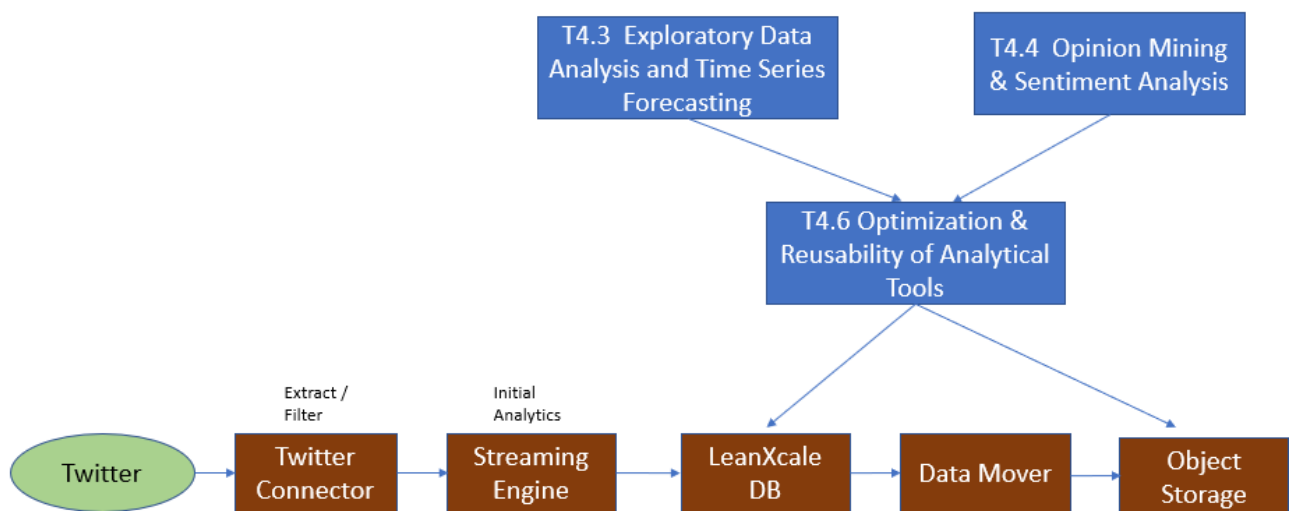


FIGURE 12 – SEAMLESS ANALYTICS ON INGESTED DATA

While the Social Dynamics & Behavioral Data Analytics (Task T4.5) is accessed as an external framework, The Analytic Functions of Situational Knowledge Acquisition & Analysis (Task T4.3) and Opinion Mining & Sentiment Analysis (Task T4.4) are built-in Analytic Functions in PolicyCLOUD, Task 4.6 (Optimization & Reusability of Analytical Tools) will apply further optimization by enabling function composition as well as the seamless analytics framework that enables transparent analytic on data in hot and colder data stores of the PolicyCLOUD data repository (LeanXscale database and object storage), Please refer to section 4.4 for details.

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

The main objective of the Situational Knowledge Acquisition (SKA) component is analyzing and estimating outcomes from real-world situations to influence the Policy Maker into the decision-making process. This will be done by means of Exploratory Data Analysis (EDA) and Machine Learning (ML) techniques and supported with use case-dependent models used for structuring the data.

Therefore, the two following components materialize the scope of the Situational Knowledge Acquisition and Analysis task:

- The SKA-EDA component is focused on providing exploratory data analysis tools for graphical summarization of datasets, key analysis for allowing the policy maker to graphically discover any particularity of their dataset that might be useful for the decision-making process.
- On the other hand, the SKA-TSF component aims to support the policy maker with further and complex analysis in charge of predicting future outcomes based on historical data.

4.1.1 Updates since D4.3

The major changes with regard to D4.3 attend to the following aspects:

- Fully defined functionality. As an outcome of co-creation workshops, centered around the use cases, and a better understanding of the platform from the pilot's side, the requirements of the task T4.3 have been fully completed. More detail on the new specifications with regards to the last specification defined in D4.3 can be found in the following section.
- PolicyCLOUD Integration. The integration of SKA-EDA with PolicyCLOUD's infrastructure has been defined, realized, and tested. This integration encompasses the OpenWhisk, PDT and PolicyCLOUD central data repository.
- Generalization: The SKA-EDA component has been refactorized for making the EDA analysis general enough to be applied to several use cases/scenarios.

4.1.2 functionality extracted from pilots

In the context of PolicyCLOUD this knowledge extraction is materialized in several ways attending to the needs elicited from the use cases. It is worth noting that all the functionalities delivered by the SKA component have as end user the policy maker, thus they always are described from 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 for year 3 are the following:

- Enlarging the SKA-EDA component focused on exploratory data analysis, the policy maker is now able to apply exploratory data analysis to two new scenarios:
 - For the “Use case 2: Intelligent policies for agri-food industry” (Scenario A - Sales prices datasets), the policy maker is able to monitor the sale price of wine as specified in different specialized websites.
 - Optionally, For the “Use Case 3: Urban policy making through analysis of crowdsourced data” (Scenarios C), the policy maker is able to analyze the data based on cross-datasets. The PC visualizes data by category, type, territory and time, which results in a set of graphical visualizations that show, for example, # of incidents, # of incidents per year, # of incidents per geographical location, # of incidents per category per location, etc.
- Developing a SKA-TSF for Predictive Analysis: More specifically for the “Use Case 3: Urban policy making through analysis of crowdsourced data” (Scenario B- Supervised Predictive Analysis), the policy maker is able to predict the number of incidents based on the SOFIA Call Center, is provided.

4.1.3 SKA components

Along this last year, for the SKA components, main efforts are focused on providing the following components:

- SKA-EDA (Exploratory Analysis) V2 will be developed, integrated and deployed
- SKA-TSF (TimeSeries Forecasting) V1 will be developed, integrated and deployed

4.1.3.1 EXPLORATORY ANALYSIS

As it was fully defined in D4.4, the exploratory data analysis component (SKA-EDA) is a tool for allowing the end user graphical exploration of datasets. Thus, the SKA-EDA is conceived as an exploratory data analysis tool in charge of providing variable distributions of attributes in the dataset by collecting datasets, applying some transformations, calculating some distributions, and providing summary data (in the form of json file) or oriented to graphical representation. In a summary, it allows the following variable distributions:

- Frequency distribution, frequency of values for a variable or joint frequencies for more than one variable. The results (json file) are typically portrayed as a horizontal bar chart.
- Spatial Distribution, list of events/signals that happen in the specific values of a geographical variable. The results (json file) are typically portrayed as a heat map
- Accumulated distribution, accumulated value of a numerical variable across several categories. The results (json file) are typically portrayed as a bar chart
- Temporal Evolution, list of values of a numerical variable across several categories. The results (json file) are typically portrayed as a line chart.

Deepening in the SKA-EDA component, the Figure 13 shows the general architecture of the SKA-EDA from a component point of view.

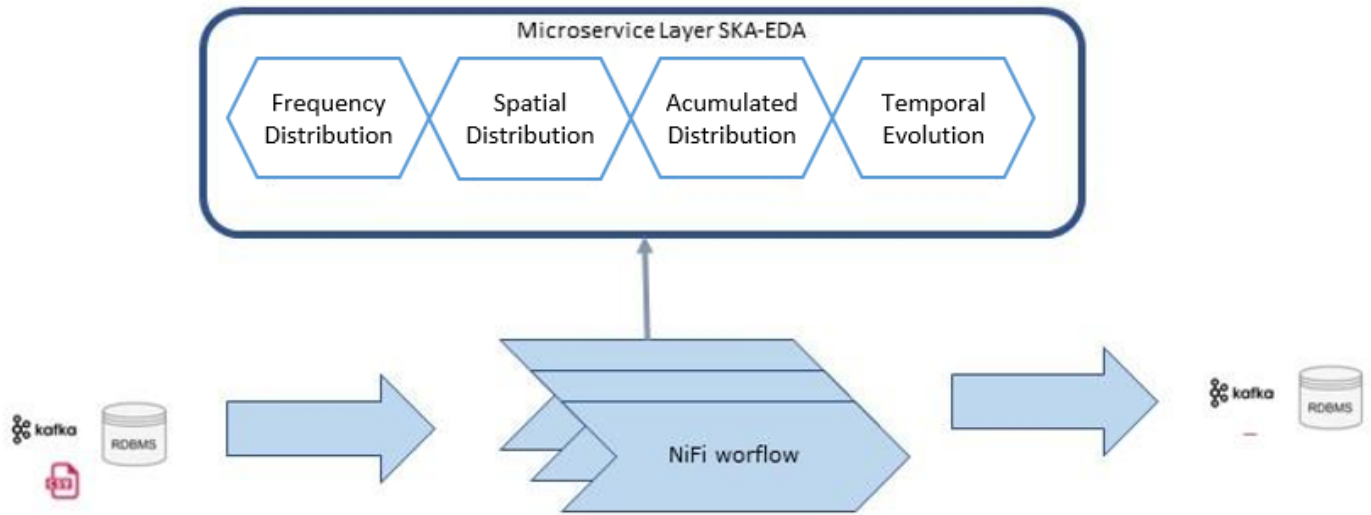


FIGURE 13 - SKA-EDA GENERAL ARCHITECTURE

As it can be seen, the SKA-EDA tool is defined by the following layers:

- A set of NiFi-based pipelines, for routing the data from the PolicyCLOUD storage to the PDT PolicyCLOUD storage with internal invocation to the different python microservices.
- A layer consisting of a set of Python microservices in charge of providing distributions for the attributes present in a dataset. This is the core of the SKA-EDA tool and provides REST API for the variable distributions before being commented:
 - Uni-Frequency distribution: Frequency of occurrence for a single variable
 - Bi-Frequency distribution: Frequency of occurrence for two variables
 - Geographical distribution: Graphical representation of the number of events that happen in the specific geographical positions
 - Accumulated distribution: Sum the value of a specific numeric variable across several categories
 - Temporal Evolution: List of values of a specific numeric variable across several categories

Regarding the scenarios, the SDK-EDA is applied to:

- Use Case 1 (Participatory policies against Radicalization). More specifically to “Scenario A: Radicalization Incidents” interested in monitoring the occurrence of radicalization incidents in a given area.
- Use case 2 (Intelligent policies for agri-food industry). More specifically to “Scenario A - Sales prices datasets”, the policy maker can monitor the sale price of wine as specified in different specialized websites
- Use Case 3 (Facilitating urban policy making and monitoring through crowdsourcing data analysis). More specifically to the “Scenario A: Visualization” interested in obtaining diverse visualizations of the data received with regards to 6 different sectors (Road Infrastructure, Environment and Air Quality, Waste Collection and Waste Disposal, Transport and Parking,

Cleanliness of Public Spaces, Violation of Public Order). Optionally, SKA-EDA might be used for “Scenario C: Environment and Air Quality Cross Analysis” focused on providing cross-analysis between the data coming from the Sofia Call Center and data from the air quality sensors in the city.

4.1.3.2 Predictive analysis

In the context of advanced analysis, predictive analysis is the capability of forecasting future and in general unknown results based on historical data. As it was mentioned in D5.1, this technique in its essence tries “to discover the relationship between independent variables and relationship between dependent and independent variable” [26] and encompasses multiple disciplines as statistical analysis, machine learning and operations research [27]. Therefore, a new component, SKA-TSF is provided in the context of the Situational Knowledge Acquisition and Analysis task for applying stochastic-based techniques for forecasting univariate time series datasets.

In order to converge to an end-to-end forecasting system, a procedure for learning model development and deployment is followed. This procedure, composed of the following steps, is based on common life cycle forecasting models:

1. Dataset preprocessing: Univariate time series exploring and transformation
2. Parameter estimation: Time series component decomposition and analysis which support the model selection and configuration (selection of the hyperparameters)
3. Model training: Fitting the model with the best parameters and for a specific training set
4. Predicting outcomes: Forecasting new data with a pre-trained model
5. Deploy the model and integration into a full-grown system

Regarding steps 1 and 2 (manually supervised) the models used for the univariate time series forecasting in PolicyCLOUD have been previously studied, selected, configured, and fitted based on an exhaustive exploratory analysis of the training time series (the original training datasets transformed into time series) used for the prediction. As part of this time series exploratory analysis, main components (trend, seasonality,..) [28] were fully identified and analyzed (based on seasonal_decompose model [29], graphical validation with ACF/PACF plot [30], ADFuller test [28]) to select both: the most suitable model approach and the adjustment of their hyperparameters. Once the approach was selected and the parameters were adjusted, the training process gives as result a set of trained models ready to be used for the prediction. Two models have been used: Prophet [31] for its seasonality and trend management and Seasonal Arima [32] for being one of the most widely used approaches.

Regarding step 3 and 4, a set of services are put available as SKA-TSF core. These services will facilitate end users to be enrolled in the forecasting process (it is still under discussion which of these services will be fully integrated into the end-to-end scenarios planned in PolicyCLOUD):

- Training Service: A process for fitting a specific model for a specific training dataset and based on a pre-determined hyperparameters. In summary, this process retrieves a (training) dataset, based

on the name passed as input parameter, performs the needed transformations and finally configures and fitting a particular model with concrete hyper-parameters (see Figure 14).

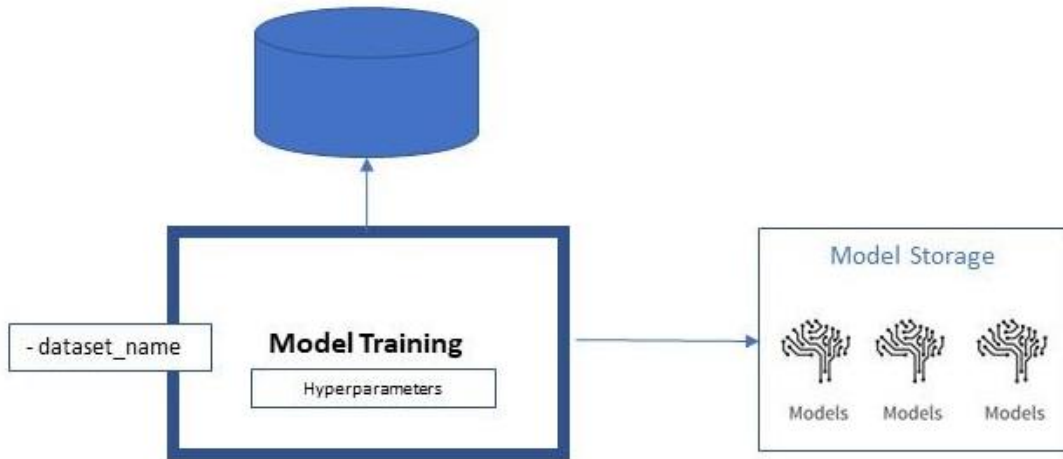


FIGURE 14 - SKA-TSF TRAINING PROCESS

- **Predicting Service:** Based on a selected trained model, given the name/id as parameter, the process is able to return predicted values (in the form of a json file) for a horizon forecast passed as input (as shown in Figure 15). It's worth mentioning that the predicting process will throw reliable outcomes for the future within the next 6 months from the last period (in our case, last day) of the trained dataset used in the selected model. Therefore, it is up to the user the responsibility of keeping the model freshly re-trained with recent data for obtaining optimal results close to recent days.

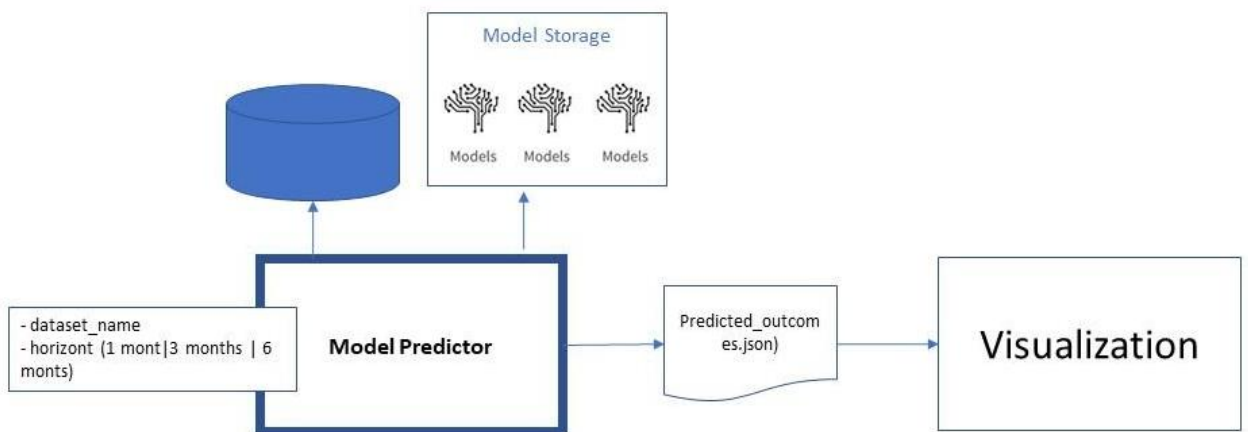


FIGURE 15 - SKA-TSF PREDICTING PROCESS

As part of the deployment and integration (step 5), this SKA-TSF component might be supported by an external tool/framework able to manage lifecycle machine learning-based models, or at least the model storage capability. The models managed in PolicyCLOUD can be either stored to its internal datastore, or if necessary, to external data management systems.

In terms of PolicyCLOUD integration, a similar approach to that followed by SKA-EDA, based on NiFi workflows, can be followed for the integration of the SKA-TSF services. Otherwise, Python runtimes for Apache OpenWhisk can be also provided.

For future improvements on the component, PolicyCLOUD might include a continuous and fully automated procedure for the deployment of models. Because the data might drift over time giving new distributions as a result, PolicyCLOUD might detect new hyper-parameters and retrain existing models over updated datasets. This retraining functionality might include two processes:

- A process for automatically detecting the best model parameters based on the new training dataset. In the case of prophet, parameters such as: `seasonality_prior_scale`, `holidays_prior_scale`, and in the case of Seasonal Arima: the arima order (p: Order of the autoregressive part, d: Degree of first differencing involved, q: Order of the moving average part) and the seasonal order (Seasonal AR specification, Seasonal Integration order and Seasonal MA, Seasonal periodicity)
- Apply the Training service, before commenting, for fitting again the model based on these new detected parameters and the new training dataset.

Regarding the scenarios, this predictive analysis might be crucial for some of the use cases raised in PolicyCLOUD. In particular, the SDK-TSF is applied to:

- “Use Case 3: Urban policy making through analysis of crowdsourced data” (Scenario B-Supervised Predictive Analysis), the policy maker will be able to predict the number of incidents based on the SOFIA Call Center, will be provided.

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

As described in D2.2 [33], the aim of these two components is focused on text analytics for extracting both the sentiment of a piece of text (following different approaches) and the contextual information of the text by identifying relations between entities in the given texts as well as the evolution of appearance for some terms along time.

4.2.1 Updates since D4.3

One of the main differences in terms of specification is that the sentiment analysis planned in D4.1 included a standalone version of the sentiment analysis. As of D.3, after integration in the PolicyCLOUD framework, the sentiment analysis is one of the steps of a stream-ingest workflow which 1) retrieves tweets filtered by keywords, 2) performs the sentiment analysis and 3) writes the results into the PolicyCLOUD data repository. Therefore, the sentiment analysis component has been redesigned to be part of a more complete workflow implemented together with other WP4 tasks.

From the specification in D4.3 additional updates have been included:

- The Sentiment analysis component has been fully defined considering the requirements from T4.3 as well as the integrations carried out and those to come. The following services have been initially developed and under integration
 - Aspect-based Sentiment Analysis: A new approach based on entity-based sentiment analysis has been developed, by means of the inclusion of the library aspect-based sentiment analysis.
 - This Aspect-based Sentiment Analysis is under integration, being almost complete at this stage (still waiting for some dependencies related to some of the pilots of the application). For this functionality, a similar approach to the one described in previous deliverables will be followed.
 - Aspect-based Sentiment Aggregation: a service for aggregating sentiment analysis results has been designed and is being implemented.
- Trend analysis component has been already designed considering all the requirements from the different scenarios to which it will be applied. This component is currently under development and its integration is about to start.

4.2.2 Functionality extracted from pilots

In the context of PolicyCLOUD the opinion mining and sentiment analysis features are materialized in several ways attending to the needs elicited from the use case. We will describe the functionalities of these components from the end user perspective, that is from the policy maker's perspective

At the closure of this document, the main activities identified in the context of Task 4.4 for the next year are:

- Opinion mining & sentiment analysis functionality, including the trend analysis, will be applied in the following scenarios:
 - For "Use case 2: Intelligent policies for agri-food industry" (Scenario B: Opinions on social media) the policy maker will be able to get statistics for several purposes:
 - For analyzing sentiment in a defined (considering time windows and appearance of terms) set of social media publications: These statistics can include both overall sentiment of the text analyzed as well as the sentiment of the text regarding certain terms related to the use case. Different statistics related to the sentiment identified towards a series of entities from the use case will be provided.
 - For analyzing trend in a defined (taking into account time windows and appearance of terms) set of social media publications: Using an ontology with the various use case entities (as well as their types and relations), the policy maker can differentiate between text which address specific use case entities (e.g., a certain wine, winery or region) and text which address all the elements of a given level of abstraction (e.g., all the wines, wineries or regions). This output will throw information that helps to measure and understand the presence of the selected elements in social media texts.

- In “Use Case 1: Participatory policies against Radicalization” (Scenario D: Assessment of online propaganda) a similar approach to the one presented in “Use case 2: Intelligent policies for agri-food industry” (Scenario B: Opinions on social media) is being carried out, where texts from social media are analyzed in order to evaluate both the sentiment towards some terrorist organizations, components, etc. as well as the impact of their presence in social media publications in a given time window. The main goal of this analysis is to identify potential radicalization efforts as well as potential relations between concepts in this environment. Nevertheless, this use case is still being discussed in order to properly define how to address it due to the sensitivity of its data.

Most of the requirements relate to social media data, in particular Twitter, but PolicyCLOUD is not restricted to that case, since texts coming from databases, opinions, and more, can be analyzed.

4.2.3 Contextual Information Opinion and Sentiment components

4.2.3.1 ASPECT-BASED SENTIMENT ANALYSIS COMPONENT

Aspect-Based Sentiment Analysis Component aims at the identification of sentiment (based on polarity indication) towards a set of entities present in a text (e.g., “wine” and “cheese” from the text “I love wine and cheese”). For this purpose, as it was introduced in D4.4, the component here proposed uses libraries that rely on the BERT sentiment analysis model [34]. The results provided by this component are structured around the different entities proposed as input, providing a sentiment polarity for each one of these entities that are present in the text.

Additionally, the sentiment aggregation functionality is currently under development to be adapted to the different use case scenarios where this will be applied. In this sense, the results provided by the aggregation must reflect the sentiment regarding a set of entities defined based on their appearance in different social media contributions over a defined period.

Regarding the architecture (shown below), both the Aspect-Based Sentiment Analysis and the sentiment aggregation, are integrated within the Sentiment analysis contribution, which relies on a set of nifi workflows in order to cover all the interactions required with other PolicyCLOUD components (e.g. PolicyCLOUD storage, PDT, etc.). In this sense, the functionality is to be offered as a set of services. The proposed functionalities in this section are offered as microservices (sentimentAspect, aggregator), in addition to the previously offered services regarding sentiment analysis (sentimentByTextblob, sentimentByVader, preprocessing, ...). Following picture depicts the general architecture of the Sentiment Analysis component.

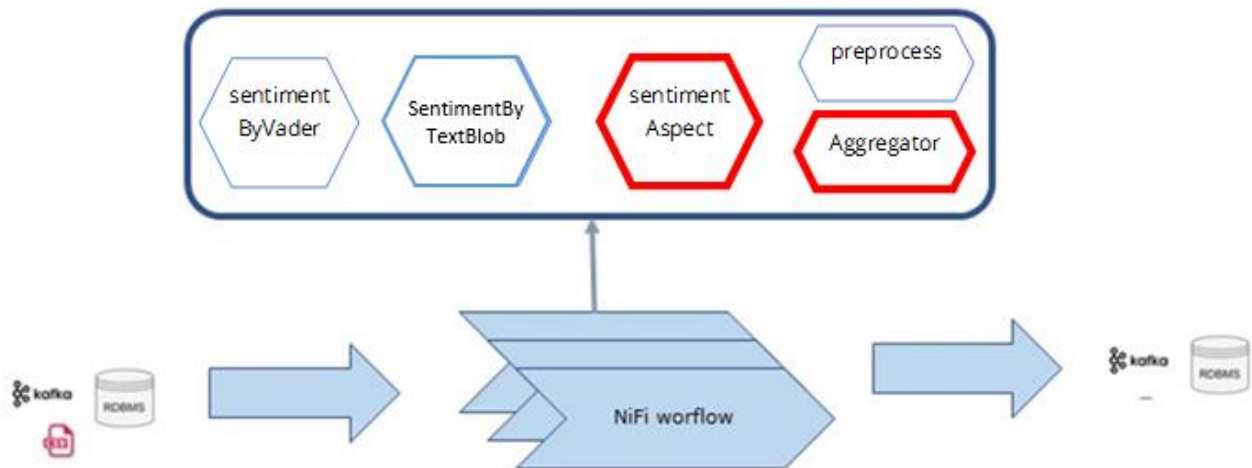


FIGURE 16 - ASPECT-BASED SENTIMENT ANALYSIS GENERAL ARCHITECTURE

Aspect-based Sentiment Analysis has been designed for the following scenarios:

- Use Case 1 (Participatory policies against Radicalization): “Scenario D: Assessment of online propaganda” to analyze the polarity of the social media content towards the terrorist groups and their members studied in this use case.
- Use case 2 (Intelligent policies for agri-food industry): “Scenario B: Opinions on social media” to evaluate the impact and sentiment generated by the different products (wines) in social media contributions. Additionally, this analysis is meant to be done not only with the products themselves but also with their producers (wineries) and areas of production in order to provide an overall view of the image that these actors have in social media.

4.2.3.2 TREND ANALYSIS COMPONENT

The Trend Analysis component proposed in this document aims at the contextualization of given entities in social media. This contextualization is expected to be carried out in two directions:

- Related entities analysis: which aims at offering information of the entities that appear linked with the entities analyzed in social media. Additionally, implicit appearance of concepts (i.e., explicit appearance of related entities) is being addressed. To carry out this analysis, it is required to get access to the modeling of the domain to be able to study the different links between the entities given in this domain.
- Impact over time: which aims at the study of the evolution of the presence of given entities in social media over time. To carry out this analysis it is needed to have all the entities that are related to the domain of application.

To carry out these analyses, a set of several processes have been designed:

- Conceptualization process: which aims at extracting all the knowledge (i.e., all the entities and their relations) from the social media interactions using the available knowledge from the domain of application (as an ontology). Figure 17 depicts this transition from the ontology of the use case to the identification of links that will be used in the conceptualization process.

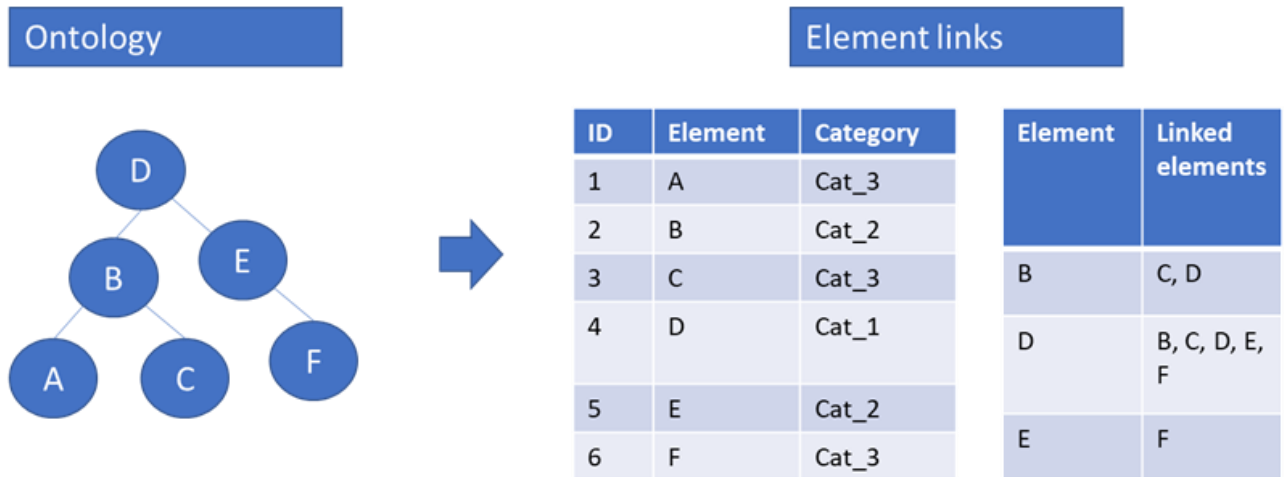


FIGURE 17 - CONCEPTUALIZATION PROCESS EXAMPLE

- Contextual entities process (for Related entities analysis): aims at offering a view containing all the entities that appear in social media together to an entity (or set of entities) provided as input

In this sense, several aggregation approaches have been identified based on three variables:

- If the input of the analytic (i.e., filtering criteria) is a given entity of the domain or if the input is a category or type of the domain (in this case, all the entities of that type will be analyzed). This can provide important insights to compare different entities of the same nature (e.g., social media presence of all the wineries).
- If implicit mentions are going to be counted for a given element: In this component, an “implicit mention” can be understood as the appearance of an entity that is linked to another one considered of a higher level of abstraction. An example for this is considering the appearance of a wine in a tweet as an implicit mention to the winery that produces that wine. In case the implicit mentions are considered, the information of the presence of an entity in social media is calculated considering its presence by itself and by the presence of all the elements that depend on it (e.g., the presence of a winery is calculated considering the mentions to the winery as well as the mentions to the products generated in that winery). In case the implicit mentions are not considered, the presence of an entity in social media is calculated considering only the explicit mentions in social media to that entity (e.g., the presence of a winery is calculated only considering explicit mentions to the name of that winery, evaluating this way the impact of the brand of the winery itself).
- If entities external to the ones of the domain of application are going to be identified. This can be useful to provide a wider context to understand which entities external to the domain are usually present in social media with entities from the domain of study.

After evaluating this with the use cases, the following use cases were identified for their implementation:

- Trend Aggregation I: Input is a category, external entities to be considered, implicit mentions to be considered.
 - Trend Aggregation II: input is a category, external entities to be considered, no implicit mentions to be considered.
 - Trend Aggregation III: Input is a category, external entities not to be considered, implicit mentions to be considered.
 - Trend Aggregation IV: input is a category, external entities not to be considered, no implicit mentions to be considered.
 - Trend Aggregation V: the input is an entity (and the output contains elements from lower levels of abstraction) external entities to be considered, no implicit mentions to be considered.
 - Trend Aggregation VI: the input is an entity (and the output contains elements from lower levels of abstraction) external entities not to be considered, no implicit mentions to be taken into account.
- Bursty concept process: This process is focused on the analysis of the evolution over time of the presence in social media of a given entity. In this case, the number of appearances of entities during two different time windows is to be calculated and compared to evaluate the trend of that entity on social media. Figure 18 shows a small example of bursty concept analytics

Element	Time window	Number of appearances
A	1/1/2022-1/2/2022	10
A	1/2/2022-1/3/2022	20
A	1/3/2022-1/4/2022	5

Element	Time window 1	Time window 1	Number of appearances
A	1/1/2022-1/2/2022	1/2/2022-1/3/2022	+100%
A	1/2/2022-1/3/2022	1/3/2022-1/4/2022	-75%

FIGURE 18 - BURSTY CONCEPT EXAMPLE

Regarding the scenarios, the trend analysis is planned for the following scenarios:

- Use Case 1 (Participatory policies against Radicalization): “Scenario C: Trend Analysis” in order to monitor the spread of extreme political and religious entities through media so they will be able to identify clouds of keywords and monitor the evolution of the topic on social media
- Use case 2 (Intelligent policies for agri-food industry): “Scenario C: Trend Analysis” for identifying trends and evolution regarding specific entities and focus on promotional measures to adjust the market to these trends.

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

4.3.1 Updates since D4.3

The Social Dynamics component, referred to as Politika, has evolved from the version described in D4.3 as follows:

1. Two new alternative GUIs have been designed for the definition of a policy in Politika from the PolicyCLOUD frontend:
 - a. a tree-based GUI in which alternatives are defined in terms of objectives (as internal tree nodes) and simulation steps (as leaves)
 - b. a text-based GUI allowing the user, in one screen, to define all the policy parameters used for simulating and evaluating policy alternatives.
2. A novel REST API has been developed that allows the integration of Politika to the rest of the PolicyCLOUD platform.
3. A new simulation and policy model has been developed that allows the design of policies seeking to increase the purchase motivation of selected wine brands in relation to their competitors in a specific population. These models along with the Politika infrastructure form part of the pilot policies for supporting the agri-food sector in Aragon.
4. The radicalization model developed in D4.3 has been extended for possible inclusion to the pilot anti-radicalization toolbox offered by PolicyCLOUD.

4.3.2 Introduction

PolicyCLOUD introduces a data-driven policy lifecycle management methodology that addresses the complete policy lifecycle: modelling, analysis, simulation, testing, monitoring, and compliance assessment. The Social Dynamics component focuses on the simulation part of the PolicyCLOUD methodology. It is used to estimate the social effects of various policies via social simulation. The data generated by this component can be used to inform policy design and analysis on the possible social implications of various policies with respect to various models of the populations of interest.

Agent-based models and their associated simulation tools have become quite valuable in the analysis of the interactions between individuals or groups in social dynamics as they can capture feedback between the behavior of heterogeneous agents and their surroundings [35]. In this paradigm, agents act independently according to prescribed rules and adjust their behavior based on their current state, on that of other agents and on the environment. Consequently, emergent patterns and dynamics can be observed arising from local interactions between the agents. Based on these features an agent-based modeling and simulation environment must provide effective state representations for individuals and their connections. These are usually implemented using a graph-based model. Furthermore, it must provide efficient execution schemes for running the simulations given the large number of individuals typically involved in them. The situation becomes more complex in the case of online meta-simulation

environments as the execution environment needs to handle, in real-time, variable simulation loads reliably, efficiently, and securely, while also providing online capabilities for developing, compiling and reasoning about simulation models.

We describe the second version of the prototype for the Social Dynamics component of the PolicyCLOUD project, referred to as Politika. Politika is an interactive, web-based meta-simulation framework for policy analysis and design. It uses agent-based, social simulation as the primary analysis tool for evaluating policy alternatives. Politika decomposes each policy in a hierarchy of goals, objectives and simulation steps following the methodology and terminology used in policy analysis [36] [37] [38]. In addition, it provides a series of analytic tools that investigate the relation of simulation outcomes to policy goals and operationalize the criteria selected by the policymakers to recommend policy decisions. Furthermore, by offering a common modeling and execution environment for simulation-based analytics it provides a standard basis that facilitates inspection and comparison of different models for social dynamics. As a result, Politika seeks to enhance the transparency, efficiency and effectiveness of the policy design process. Furthermore, Politika seeks to increase the computational efficiency of policy analysis. This is achieved using concurrency in the simulator that exploits processor core availability in the server during processing and can scale up through the formation of server clusters according to simulation requirements.

Currently, we are focused on validating the Social Dynamics component through its integration in the use cases of the PolicyCLOUD project. We are especially interested in the design of intelligent policies for increasing commercial success for wine brands with denominations of origin. In this case, the Social Dynamics component is used to guide the design of policies for increasing the average purchase motivation for wines from the Aragon region. To this end, we simulate and evaluate various scenarios that take into consideration parameters such as the price, popularity, quality, advertisement effort and levels of social influence for Aragon wines and their competitors to explore different policies for achieving this goal. We also have integrated Politika in the cloud architecture used in PolicyCLOUD. During policy design, Politika can simulate the social effects of implementing a policy. Therefore, it can provide valuable simulation results data that can be further analyzed by the policy-making components of PolicyCLOUD. Currently, we are exploring the integration of Politika as an external tool able to cooperate with the Data Acquisition layer.

4.3.3 Simulation Environment for Social Dynamics

4.3.3.1 ARCHITECTURE

The Social Dynamics simulation component provides an online environment for Agent-Based Modeling and Simulation (ABMS) [39] seeking to simulate the social effects of various policies applied on a population of autonomous and interconnected agents represented as a graph. Figure 19 depicts the overall architecture of the meta-simulation environment. 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 a simulation, examine raw simulation results, and compute and visualize

simulation analytics. All these operations serve multiple users concurrently except from simulation execution, which, due to possibly high computational load, and possibly limited server resources, runs one simulation at a time by default. In this case, all pending user requests for simulation execution are placed in wait mode during which they poll the simulator for availability with a fixed frequency. A user request in wait mode starts executing automatically when polling succeeds in finding an available slot. In terms of implementation, on the server side all user requests are processed by a web server based on the Phoenix web framework [40]. The Phoenix system interacts with three independent components consisting of the Social Dynamics simulator built in Elixir [41] which in turn is built on Erlang [42], the Analytics component that includes the meta-simulator environment also built in Elixir and the Database system that uses the Ecto/Postgres [43] environment.

Throughout this section we examine the design of a hypothetical example policy for controlling radicalization in a population. The example assumes that the radicalization process consists in the progressive adoption of extreme political, social or religious ideals in the population through social influence. In this model, each individual has an initial radicalization status, a real number between -1 (non-radicalized) and +1 (fully radicalized). Each individual connects to a number of other individuals through a number of outgoing, directed connections. Each such connection has a contact strength that indicates whether the individual regards this connection as friendly or not. Contact strength is a real number between -1 (enemy) and +1 (friend). Furthermore, each connection has an influence representing the level of social influence that an individual exerts on the individuals he connects with in terms of radicalization. At each simulation cycle, the influence is computed as the product of the radicalization status of the individual and the contact strength of the connection. Therefore, a radicalized individual is expected to make his friends more radicalized while making his enemies less radicalized. At each simulation cycle each individual updates his radicalization status by adding to his current radicalization status the sum of the influences he receives through all his incoming edges. At the end of the simulation, individuals with a radicalization status greater than 0.5 are considered radicals, those with less than 0 are considered conformists (non-radicals) and the rest are considered radical sympathizers. The example compares the social outcomes of applying a policy that restricts the interactions of radicals with the rest of the population with the outcomes of a base case of applying no such policy. Restricting the interaction of radicals is modelled as a reduction of the contact strength of the connections of the radicals (those with a radicalization status greater than 0) with their friends (those who are connected with a radical with a contact strength greater than 0).

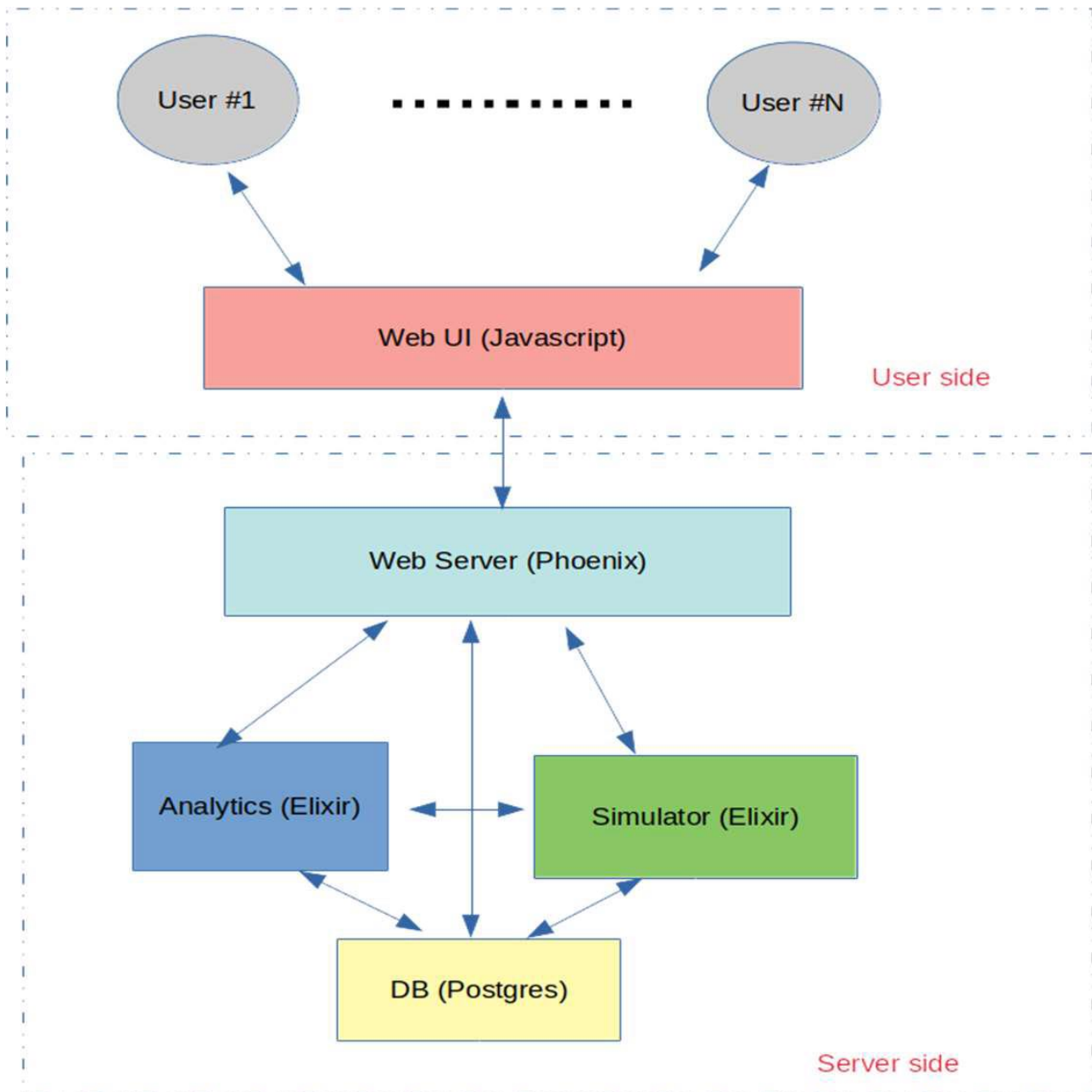


FIGURE 19 - OVERALL ARCHITECTURE OF THE POLITIKA SIMULATION ENVIRONMENT

4.3.3.2 MODELLING ENVIRONMENT

Figure 20 describes the contents of the simulation models fed to the Social Dynamics simulator. The modelling input for this component consists of the following data sources:

A list of policy attributes and their initial values

For example, in a radicalization control policy scenario these can include the proportion of radicals, radical sympathizers and conformist (non-radicalized) individuals in the population as in:

radicals: 0.15, sympathizers: 0.20, conformists: 0.65

At the end of each simulation cycle the simulator updates these attributes based on the model for policy dynamics.

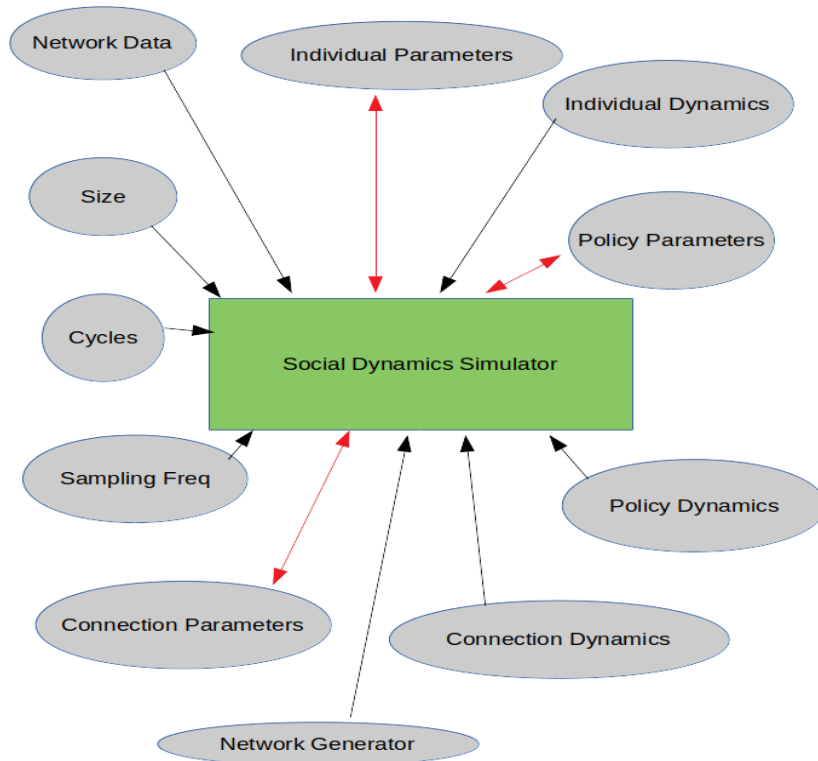


FIGURE 20 - I/O SPECIFICATION OF THE SOCIAL DYNAMICS SIMULATION

In Figure 20, bidirectional red arrows indicate data sources which the simulator can read and/or update and unidirectional black arrows indicate user-defined data sources which the simulator can only read.

A set of rules for policy dynamics

These rules specify how the policy attributes change during the simulation. In general, the values of such policy attributes are computed based on aggregation of individual attributes. Furthermore, policy dynamics rules can be used to model exogenous influences on the population such as the addition of individuals through migration or the imposition of quotas and other policy measures.

A list of individual attributes and their initial values

These describe the characteristics of each individual agent in the population. For example, in our radicalization-related policy scenario, these can include the individual's current radicalization status, education and income level. During each simulation cycle the simulator updates these attributes for each individual in the population.

A list of edge attributes and their initial values

These describe the relations between agents in the population. Each such relation is represented as an edge with attributes between two individuals in a graph. Each edge contains a set of attributes that characterize such a relation and may change during simulation. For example, in our radicalization-related

policy scenario, these attributes include the strength of each connection and its influence over the radicalization status of the individual at the receiving end of the edge. In general, each individual can be connected to multiple other individuals. Each connection is unidirectional. Therefore, bi-directional links are modelled as a pair of unidirectional links with the same attributes and initial values but opposite directions between two nodes. During each simulation cycle the simulator updates the attributes of all the connections for each individual in the population.

A set of rules for individual dynamics

These describe how attributes for each individual are updated during the simulation. Some of the attributes of each individual are influenced by an aggregated function of the attributes of its incoming edges. For example, the radicalization status of an individual is influenced by the sum of the products of the strength times the influence of all its incoming edges. Furthermore, these rules describe how the connectivity for each individual changes during the simulation (e.g., creation/deletion of individual connections). Changing network structure and connectivity with individual dynamics rules allows the user to model how graph topology evolves based on individual decisions at a local level. Finally, individual dynamics rules can add new individuals to the population thus allowing the modelling of local reproduction phenomena.

A set of rules for connection dynamics

These describe how both connections and their attributes change during the simulation. Such dynamics depend on the attributes of the nodes at both ends of the connections and on the policy-relevant attributes.

The size of the population on which the policy will be simulated.

The number of cycles for which the simulator will run.

The sampling period (in simulation cycles)

This defines the period with which the attributes for each individual are sampled. Sampled values for each attribute form a history for changes in the attribute values of all individuals at various cycle intervals. Such a history can be later processed by various analytics or visualization methods.

The specifications 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. The features of such a model are provided as a set of key-value pairs. For example:

```
max_edges: 5, min_edges: 1, directed?: false, degree_distribution: :homogeneous
```

These are instructions to the network generator to create a population where each individual has at least one and at most five outgoing unidirectional edges and where the specific number is randomly selected from a uniform distribution between these extreme values.

Network data in JSON format

These can describe the individual and connection attributes of the members of an actual population dataset that can be uploaded/downloaded to/from the simulator. The modeling language used by the system represents individual, connection or policy attributes as keyword/value strings. For example:

```
radicals: 0.10, sympathizers: 0.18, conformists: 0.72
```

These represent the current proportion of radicals, radical sympathizers and conformists. Strings ending in `:` are keyword names for attributes, while the rest are the parameter values.

4.3.3.3 SOCIAL DYNAMICS SIMULATOR

The simulator provides a wrapper over the state of each individual in the process that governs its updating in a concurrent environment. During each simulation cycle it spawns a set of concurrent processes, one for each individual, that run the individual and connection dynamics rules for it and update its state using the wrapper.

All types of dynamics (policy, individual or connection) are specified as lists of IF..DO rules of the form IF <IF-part of rule> DO <DO-part of rule>

that the simulator compiles into Elixir functions. In the case of individual dynamics rules, these functions have, in pseudocode, the general form:

```
function (this, global, cycle) {  
  if <IF-part of rule> then  
    updated_this = <DO-part of rule>  
  return updated_this  
else  
  return this  
}
```

where this and updated_this refers to the current state of the individual executing this rule and the one resulting from executing the do-part of the rule, respectively. In addition, global refers to the current state of the policy and cycle is the current simulation cycle number. In order to avoid indeterminate updating of agent states in a concurrent rule execution environment, each individual dynamic rule can only update the state of the individual executing the rule. Conceptually, individual dynamics rules utilize:

- attribute 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 attribute values in incoming connections
results of aggregated function evaluations on attributes of incoming connections (e.g., sum, threshold, mean).

As a result, these rules update the attributes of the individual involved and/or manage outgoing connections (e.g., remove/create/update outgoing connections). In order to model reproduction phenomena in the population, the execution of an individual dynamics rule can add new nodes in the population as a side effect. In order to be able to manage these side effects consistently in a concurrent environment, the simulator models the population as a separate concurrent process with its own state

that receives messages from individuals to update its state. The population state includes a list of the concurrent processes corresponding to the list of individuals in the population that is updated every time an individual is added/removed from the population.

Connection dynamics rules are compiled into functions that have the following form:

```
function (this, global, cycle, edge, distal) {
  if <IF-part of rule> then
    updated_edge = <DO-part of rule>
  return updated_edge
else
  return edge
}
```

Compared to the previous case, such functions have two additional arguments/variables: `edge` and `updated_edge` that refer to the state of the specific connection that runs the rule and the one resulting from executing the do-part of the rule, respectively. In addition, `distal` refers to the state of the individual at the receiving end of the connection. In order to avoid indeterminate updating of agent states in a concurrent rule execution environment each connection dynamics rule can only update the state of the connection executing the rule and not the state of the nodes at the receiving end of the edge. Conceptually, connection dynamics rules access attributes 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 are compiled like the individual dynamics rules as:

```
function (this, cycle, size) {
  if <IF-part of rule> then
    updated_this = <DO-part of rule>
  return updated_this
else
  return this
}
```

This is because the policy state is implemented as a unique, special individual with its own attributes that correspond to the policy attributes and with no connections to the rest of the population. Conceptually, policy dynamics rules utilize current values of policy attributes while also computing aggregated population attributes based on attributes of individuals to update policy parameters at the end of each simulation cycle. This is why the function takes a third argument referred to as `size` with a value equal to the size of the current population. In addition, policy dynamics rules can add new individuals in the population. In this case, each new individual is responsible for forming connections with the rest of the population through its individual dynamics rules.

For example, the code

```
IF
this.radicalization_status > 0 and edge.contact_strength > 0
DO
edge.contact_strength: random(0, 1) * edge.contact_strength
```

represents a connection dynamics rule that randomly decreases the strength of connections a radicalized individual has with other individuals during a radicalization containment scheme. The state

of the connection is represented with the edge variable, while the state of the individual from which the edge originates with the `this` variable.

The simulator permits the concurrent simulation of the behavior for all individuals based on BEAM, the Erlang runtime system with built-in support for concurrency and distribution of processing tasks in different nodes of a server cluster. The need to create computationally efficient simulations through concurrency requires the design of a global execution scheme at the population level that guarantees proper updating of attributes in all individuals and their connections. In particular, this scheme needs to:

- provide immutable state for each attribute that avoids situations in which attributes accidentally change as a result of various side effects during computation
- ensure that at each point in time only one process updates an attribute, otherwise the outcome of the updating process can be undermined

The first requirement is satisfied using functional programming and in particular Elixir as the development environment. This is the case because in functional programming all variables have an 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 parameters but also ensure the autonomous behavior of each individual in the simulation. To this end, each connection dynamics rule updates only the attributes for the connection it is applied to, while attributes 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 21. As this figure shows during such execution each individual runs first its connection and then its individual dynamics rules sequentially in the order specified by the simulation designer for each rule type. This ensures that connection dynamics rules always use the values of the individual attributes set at the end of the previous simulation cycle, thus facilitating monitoring of their dynamics and providing consistency in their behavior. In addition, it ensures that only one rule can update an individual/connection attribute 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., when coding the steps in a financial transaction). Finally, policy dynamics rules are executed sequentially in the order specified by the designer at the end of each simulation cycle since they utilize aggregated individual attributes. Such rules can only read attributes of individuals while they can read and update policy attributes. Overall, this scheme ensures the proper updating of individual, connection and policy attributes in a concurrent execution environment, because for each one of them there is a unique way of modifying it at each point in time.

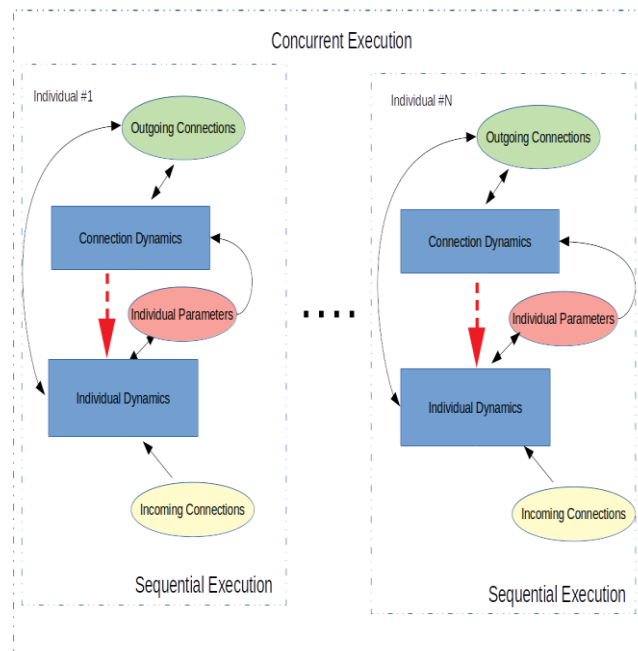


FIGURE 21 – CONCURRENT EXECUTION SCHEME OF THE SOCIAL DYNAMICS SIMULATOR

Figure 21 which depicts the concurrent execution scheme of the Social Dynamics Simulator for a number of individuals #1...#N, the red arrows indicate execution order in the sequential execution case. The unidirectional arrows indicate data sources that can only be read by a dynamic subsystem. Finally, bidirectional arrows indicate data sources that can be read and/or updated by such a subsystem.

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 based on the list of concurrent processes corresponding to individuals in the population state. Chunks are executed sequentially, while the dynamics for the individuals at each such chunk are executed concurrently. For example, if chunks A and B consist of 100 individuals each, the simulator executes concurrently all the individuals in chunk A followed by the concurrent execution of all the individuals in chunk B. Chunk size is chosen so as to optimize the size of the execution queue for each scheduler. Furthermore, in order to avoid artifacts resulting from the use of fixed chunks during the simulation that result in a fixed 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.

Compilation of rules into Elixir functions presents a security threat for a web-based simulator since it could allow the execution of malicious code at the server hidden in one of the dynamic rules. In order to increase the possibility of detecting such threats, the simulator performs a static security analysis of the code of the rule that is turned into a function. In particular, the simulator maintains a list of Elixir and Erlang built-in functions that are considered to be safe. These include arithmetic (e.g., +, *) and comparison (e.g., max, min, ==) built-in Elixir functions that are further enriched with many simulator-specific functions we have implemented mainly for:

- computing aggregate functions over individuals and over their incoming edges
- allowing the generation or deletion of edges from an individual that satisfy relevant criteria
- adding/removing individuals in the population that satisfy relevant criteria

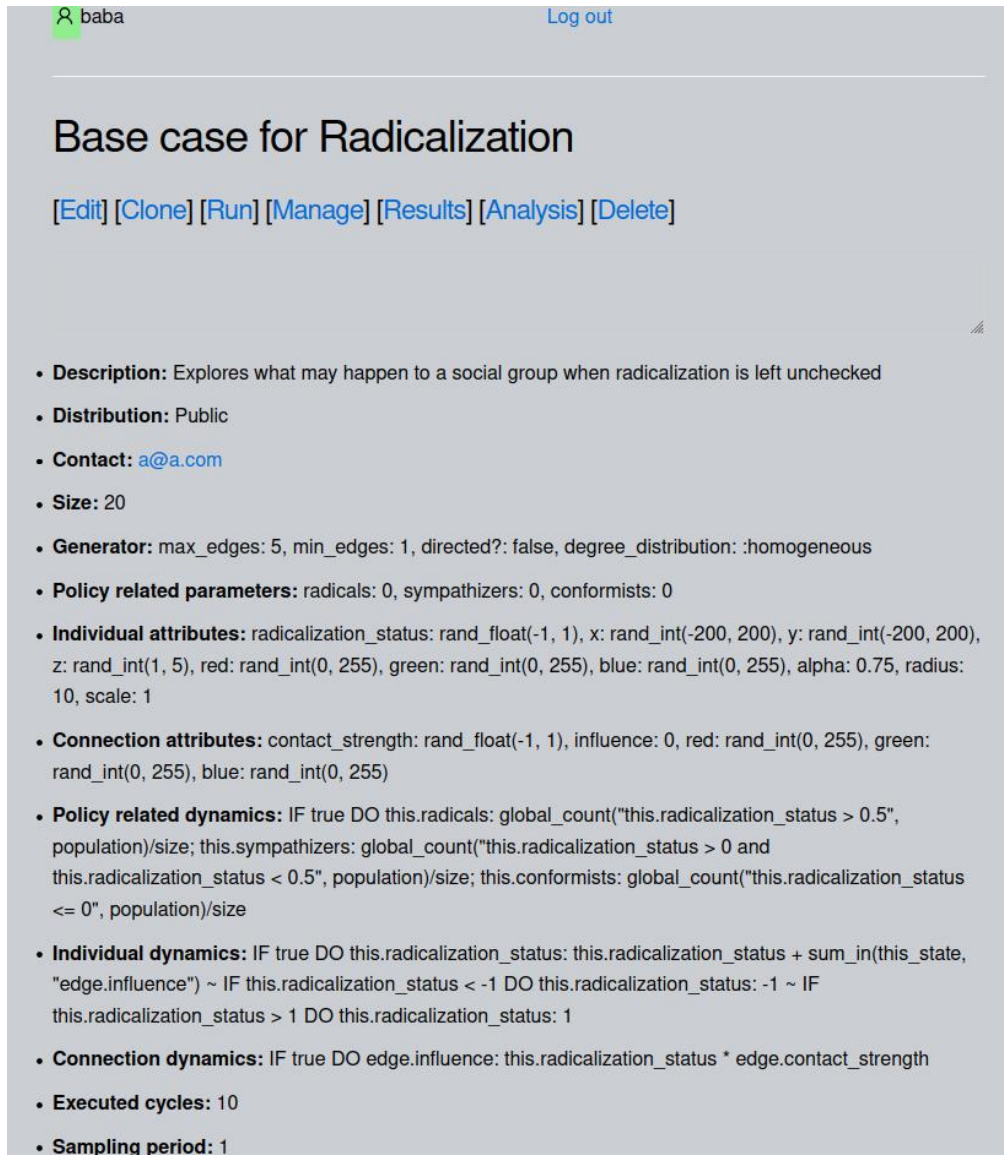
If a non-allowed function is detected, compilation stops and the particular simulation task is aborted.

Furthermore, we seek to increase the reliability of our web-based simulator by monitoring the use of server resources during simulation. More specifically the simulator examines the available memory for operations at the end of each cycle. If this is below a critical threshold that is specific to the server running the simulation, no new cycle of the simulation runs, the results of the current execution cycle are stored in the database and the user is informed about the number of cycles that the system was able to execute and the breach of the memory threshold. This allows the system to abort simulations that utilize population and connection magnitudes beyond the capabilities of the servers running the system. At its current state, Politika runs on a minimal server with two cores and 6 MB RAM. It can reliably execute the simulation model represented by Figure 22 (in the next sub-section) for a population of up to 30K individuals.

4.3.3.4 EXAMPLE OF A SIMULATION MODEL

Figure 22 provides a screenshot of one of the simulation models in Politika related to the base case of the radicalization example we have been using in this section. In this example, the policy-related dynamics rules compute the proportion of radicals, sympathizers and conformists in the population. The individual dynamics rules update the radicalization status of each individual based on its current value and on the aggregate of the influence it receives from the nodes that seek to interact with him (i.e., its incoming nodes). In addition, they clamp the radicalization status of the individual to the range $[-1, 1]$. The connection dynamics rules compute the influence that an individual has on those he seeks to interact with as the product of his radicalization status times the strength of his contact with the other individual. As Figure 22 shows on the top menu, the user can examine all parts of the simulation model, execute it, view its results or clone it if he wants to create a new simulation model based on it. He can also adapt the population to his needs interactively through the Manage option. Some of the individual and connection attributes in the model (x , y , z , red, green, blue, alpha and radius) are used for visualizing the current state of the social network during simulation.

4.3.4 Meta-Simulation Environment for Policy Analysis and Design



The screenshot shows a user interface for a simulation model. At the top left, the user is identified as 'baba' with a profile icon. At the top right, there is a 'Log out' link. The main heading is 'Base case for Radicalization'. Below the heading are several action buttons: [Edit], [Clone], [Run], [Manage], [Results], [Analysis], and [Delete]. The main content area contains a list of parameters and dynamics for the simulation model:

- **Description:** Explores what may happen to a social group when radicalization is left unchecked
- **Distribution:** Public
- **Contact:** [a@a.com](#)
- **Size:** 20
- **Generator:** max_edges: 5, min_edges: 1, directed?: false, degree_distribution: :homogeneous
- **Policy related parameters:** radicals: 0, sympathizers: 0, conformists: 0
- **Individual attributes:** radicalization_status: rand_float(-1, 1), x: rand_int(-200, 200), y: rand_int(-200, 200), z: rand_int(1, 5), red: rand_int(0, 255), green: rand_int(0, 255), blue: rand_int(0, 255), alpha: 0.75, radius: 10, scale: 1
- **Connection attributes:** contact_strength: rand_float(-1, 1), influence: 0, red: rand_int(0, 255), green: rand_int(0, 255), blue: rand_int(0, 255)
- **Policy related dynamics:** IF true DO this.radicals: global_count("this.radicalization_status > 0.5", population)/size; this.sympathizers: global_count("this.radicalization_status > 0 and this.radicalization_status < 0.5", population)/size; this.conformists: global_count("this.radicalization_status <= 0", population)/size
- **Individual dynamics:** IF true DO this.radicalization_status: this.radicalization_status + sum_in(this_state, "edge.influence") ~ IF this.radicalization_status < -1 DO this.radicalization_status: -1 ~ IF this.radicalization_status > 1 DO this.radicalization_status: 1
- **Connection dynamics:** IF true DO edge.influence: this.radicalization_status * edge.contact_strength
- **Executed cycles:** 10
- **Sampling period:** 1

FIGURE 22 – SCREENSHOT OF SIMPLE SIMULATION MODEL IN POLITIKA DEALING WITH RADICALIZATION

4.3.4.1 POLICY DESIGN PROCESS AND ITS DECOMPOSITION

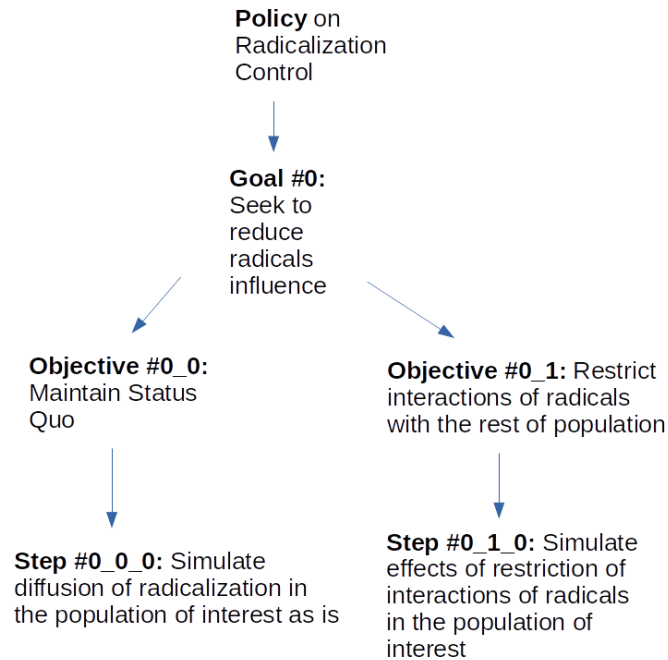


FIGURE 23 – EXAMPLE POLICY STRUCTURE USED BY THE META-SIMULATION ENVIRONMENT

We envision policy design as the process that forms policy alternatives and transforms estimates of their outcomes into policy recommendations. The role of social dynamics simulation tools in this case centers on analyzing each alternative to estimate its social outcomes. Assuming the Politika users to be practicing policy designers, we seek to follow the actual terminology and methodology used by policy analysts [36] [37] [38] [39] and model policy design as a tree hierarchy at the root of which sits a Policy node having a set of alternative Goals as its children. Each Goal contains an abstract description of the desired outcomes of a policy. Under each Goal hangs a set of alternative Objectives for achieving this Goal. An Objective corresponds to a specific methodology for achieving a goal. Each Objective is decomposed into a sequence of Steps, each of which represents a policy execution step in the methodology of the parent Objective. We assume that the execution of each step can be simulated, thus providing a value range for its possible outcomes. For example, Figure 23 describes the design of a hypothetical policy for controlling radicalization in a population. This specific Policy consists of one Goal (#0) for reducing the influence of radicals over the population. Two Objectives are analyzed with regards to this goal. The first one (#0_0) is the base case of maintaining the status quo. Essentially it models the problem that the policy is seeking to address and provides a base of comparison for the rest of the analysis. The other one (#0_1) seeks to restrict the influence of radicals to the public. Each one of these objectives has a Simulation Step below it. The #0_0_0 Step provides a simulation model of the evolution of radicalization in the population of interest with no policy applied. The second one (#0_1_0) provides a simulation model of the social dynamics that ensue when a policy seeks to restrict radicals' interaction with the rest of the population.

4.3.4.2 DESIGN PARAMETERS

Associated with a policy is a set of design-specific parameters that can be defined at various levels in the tree hierarchy and following a top-down propagation in the hierarchy. These include:

1. *The population model relevant to the policy*

In the case of social dynamics we assume that the population can be described as a graph in which individuals correspond to nodes and their relations as edges. In order to make comparisons and reason about policy alternatives, it is often the case that their analysis should be based on the same population model. Consequently, we assume that the population model should be defined at the Policy level and its definition will be adopted by all the nodes below it. For example, in our radicalization control policy we can define our population model at the Policy level as in section (The specifications of a network generator component). As a result, all the simulation Steps will be constrained to use the same model in their network generator component when creating their population of individuals.

2. *The set of policy-relevant attributes*

We assume that policy-relevant attributes should be defined at the Policy level to determine the relevant policy dynamics that will be simulated and compared in all levels below. For example, at the level of Policy we can define our set of policy-related parameters as in section 4.3.3.2 (A list of policy attributes and their initial values) This forces every simulation model at the Steps level to use the same policy parameters for the simulation outcomes to be comparable.

3. *The number of rounds and sizes of populations on which each alternative will be tested*

We assume that both parameters should be defined at the Goal level. For example, specifying the number of rounds at 10 will force each Objective below the particular Goal to schedule the simulations of the dynamics of 10 different populations generated using the population model defined at the Policy level. In this case, a number of 100 for population size will force each Step below the current Goal to generate a population of 100 individuals for each of the 10 different simulation rounds. Both numbers can be set randomly from a range of values supplied by the designer. For example, the designer may want to investigate whether population size affects the simulated policy outcomes and for this he may select a different population size chosen uniformly from a range between 500 to 500 for each of the 10 rounds of simulations in each simulation Step.

Figure 24 describes the top-down propagation of design parameters for the policy depicted in Figure 23. In this case, the blue arrows depict the top-down propagation of the design parameters in the hierarchy. According to Figure 24 the values for the population model and policy attributes defined at the Policy level are assigned to the network generation and policy attributes components in the simulation models for both Steps #0_0_0 and #0_1_0, while the values for population size and simulation rounds at Goal #0 are similarly propagated to both Steps #0_0_0 and #0_1_0.

We should note that the designer can override our recommendations and provide values for all the design parameters at any level in the tree hierarchy. For example, he can specify population size at a Step if he wishes. In case of conflict, the node at the current level will use the parameter value closest to it in the path connecting it to the root of the tree. For example, in Figure 23 if the population size is defined

in Goal #0 and redefined in Objective #0_1 then Step #0_1_0 will use the size defined in Objective #0_1 as this is closer to it.

4.3.4.3 MODELLING OUTCOME ESTIMATION TO POLICY RECOMMENDATION AS A META-SIMULATION PROCESS

Given a decomposition of the policy design process as a tree hierarchy, the Meta-Simulation environment automatically generates a bottom-up processing pipeline for transforming simulation outcomes of the various alternatives into policy recommendations. Figure 24 provides an example of this pipeline, indicated with the red arrows, for the tree hierarchy described in Figure 23.

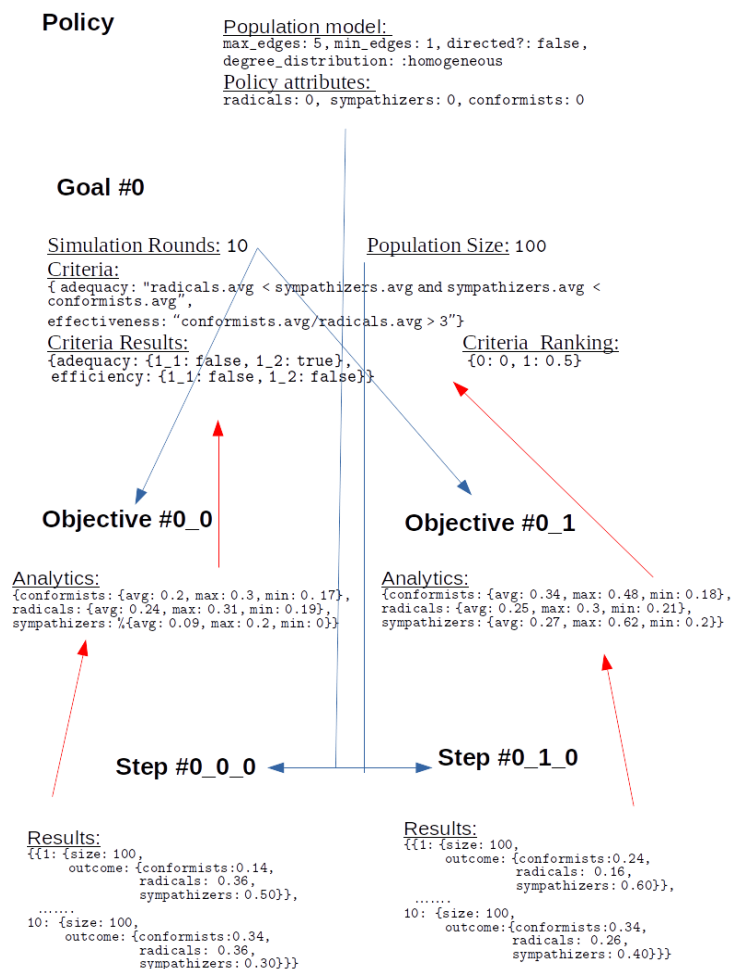


FIGURE 24 - POLITIKA PROCESSING PIPELINE

In Figure 24, blue arrows indicate the top-down propagation of design parameters in the hierarchy and red arrows indicate the bottom-up processing of simulation analytics.

In particular, when the user chooses to execute the policy design process tree then every Step in the leaves of this tree runs the simulation model it has been associated with using the design parameters defined for it. Each Step maintains the results of all the rounds of simulations it has executed along with the population size in each one, both indexed under the round number for each. The results of each Step are then fed to the Objective above it in order to compute at this higher level a set of analytics for each of the policy-relevant attributes defined in the design. For this example, this set includes the average value of the attribute in question along with its minimum and maximum value after all the simulation rounds.

For example (see Figure 24), executing Step #0_1_0 in the radicalization control policy with:

- policy-related parameters of radicals, sympathizers and conformists defined at the Policy node
- population model defined at the Policy node
- population size of 100 and 10 rounds of simulation defined at Goal #0 node

produces a map of results for the policy parameters with keys indicating the specific round on which these results were computed as in

```
{{1: {size: 100,
outcome: {conformists:0.24,
radicals: 0.16,
sympathizers: 0.60}}, ... }
```

This map is fed to Objective #0_1 which now computes an analytics map with keys equal to the list of policy attributes and values equal to their statistics for the 10 rounds of simulations that were executed as in:

```
{conformists: {avg: 0.34, max: 0.48, min: 0.18},
radicals: {avg: 0.25, max: 0.3, min: 0.21},
sympathizers: {avg: 0.27, max: 0.62, min: 0.2}}
```

Analytics maps from each objective are then fed to a set of criteria defined by the policy designer at the Goal above them. Each criterion is defined as a key/value map where key is the name of the criterion and value contains a logical expression involving the policy-related parameters defined for the policy. For example, in Figure 24 we define two criteria in Goal #0. The first criterion named adequacy provides a logical expression that checks whether the average value for the radicals computed in the objectives below it are less than that of their sympathizers, and the one for their sympathizers is less than that of the conformists. The second criterion named effectiveness provides a logical expression that checks whether the average value for the conformists computed in all of the objectives below in Figure 25 is at least three times greater than the one for the radicals. The meta-simulator automatically evaluates each criterion on the analytics maps provided by its objectives and generates a criteria results map that for each objective describes whether it has passed this criterion or not. For example, in Figure 24 both Objectives #0_0 and #0_1 have failed the effectiveness criterion and only one (#0_1) has passed the adequacy criterion.

After evaluating each objective on the set of criteria defined for the specific Goal, the meta-simulator assigns a criteria-ranking map for the objectives based on the proportion of criteria that each one has

satisfied. In our example, this ranking map has the form {0: 0, 1: 0.5} for the two objectives of the specific Goal. The designer can now consult this map to find out which of the two objectives is better for implementing Goal #0.

4.3.5 Policy GUIs

Two novel GUIs have been developed for defining a policy during design: a tree based one that closely follows the structure described in the previous section and a text-based one that reflects the GUI used in the PolicyCLOUD analytical functions that communicate with the Politika tool.

4.3.5.1 TREE-BASED GUI

Figure 25 shows the form of this GUI in Politika. The policy is depicted as a tree structure in which the green (root) node contains the description of the policy and the red node contains the description of a policy goal. The particular goal in the Figure has two alternatives depicted as blue nodes and corresponding to the different policy objectives for reaching the goal and below each objective lies a yellow node representing the models referred to as steps in the previous section that will be used to simulate each alternative. By clicking on any of these nodes the user can see and edit the contents for it in the text area at the bottom of the screen. Using the menu at the top of the screen, he can then update, delete or execute this element or add another element below it. The results of the execution are shown in the text area at the bottom of each screen. For example, the specific scrollable text area in Figure 25 depicts part of the contents of one of the Objectives indicated with the blue squares in the Figure for the Agri-Food use case described in section 4.3.7.

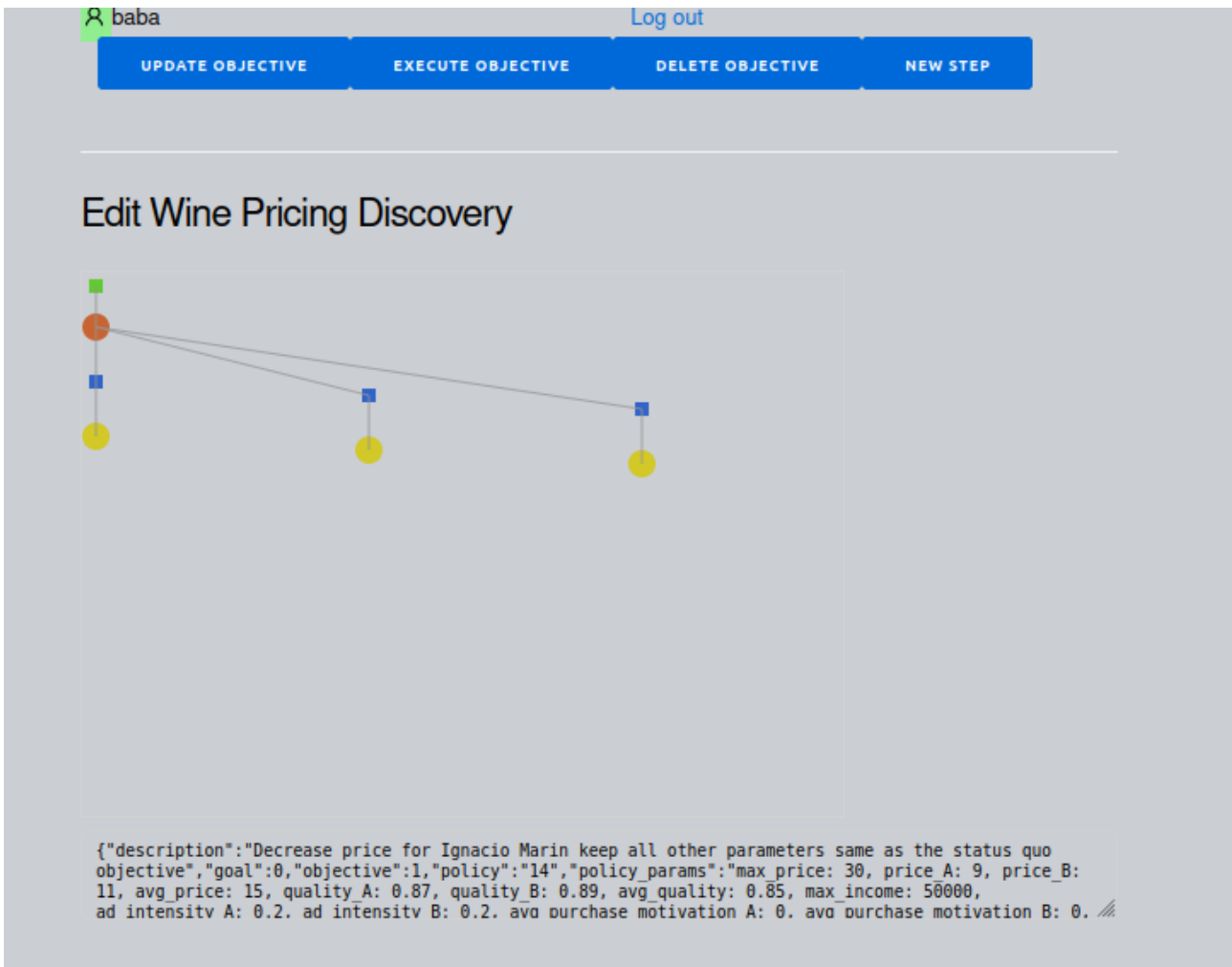
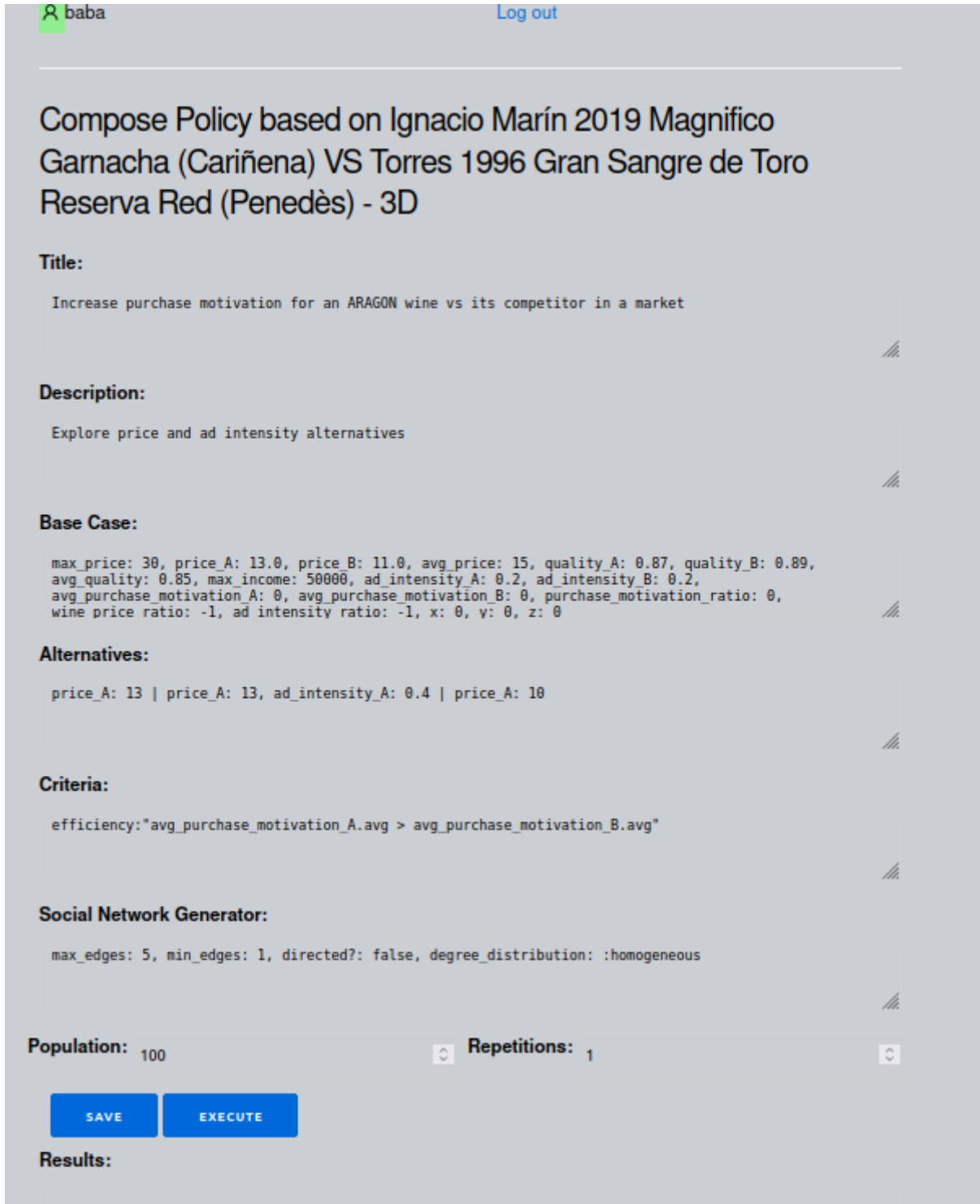


FIGURE 25 - SCREENSHOT OF THE TREE GUI FOR A POLICY MODEL IN POLITIKA

4.3.5.2 TEXT-BASED GUI

Figure 26 depicts a text-based GUI in Politika for describing in one screen the contents of a policy model with one goal. The user can access this GUI from a simulation model which will be used to compute the outcomes of the alternatives. The user inputs the name and description of the new policy using the similarly named text areas, while the base case text area is completed automatically by the initial values of the policy parameters of the associated simulation model through which the user accessed the GUI. These initial values are assumed to provide the base case for the policy, i.e., the social dynamics of the specific population when no policy is applied to it. In the alternatives text area the user describes the various policy alternatives she wants to try. Each alternative is represented as a comma-separated list of initial values of the policy parameters that are different from the base case, while the parameters not mentioned in each alternative are assumed to receive their values from the base case. Alternatives are separated with the '|' character. In the criteria text area, the user describes the criteria that will be used to evaluate each alternative, while the social network generator area contains the model of the population that will be used for simulating all the alternatives defined for this policy. For ensuring a common basis for evaluation and comparison purposes, all alternatives will be simulated with the same population model. The user inputs the size of the population and the number of cycles that will be used

during the simulation of all the alternatives for ensuring a common basis for comparison and evaluation purposes again. Finally, the user can either save or execute the policy model and the results will be shown in the similarly named text area.



The screenshot shows a web interface for defining a policy model. At the top left, there is a user profile icon and the name 'baba'. At the top right, there is a 'Log out' link. The main content area is titled 'Compose Policy based on Ignacio Marín 2019 Magnifico Garnacha (Cariñena) VS Torres 1996 Gran Sangre de Toro Reserva Red (Penedès) - 3D'. Below the title, there are several sections for defining the policy model:

- Title:** Increase purchase motivation for an ARAGON wine vs its competitor in a market
- Description:** Explore price and ad intensity alternatives
- Base Case:** max_price: 30, price_A: 13.0, price_B: 11.0, avg_price: 15, quality_A: 0.87, quality_B: 0.89, avg_quality: 0.85, max_income: 50000, ad_intensity_A: 0.2, ad_intensity_B: 0.2, avg_purchase_motivation_A: 0, avg_purchase_motivation_B: 0, purchase_motivation_ratio: 0, wine_price_ratio: -1, ad_intensity_ratio: -1, x: 0, y: 0, z: 0
- Alternatives:** price_A: 13 | price_A: 13, ad_intensity_A: 0.4 | price_A: 10
- Criteria:** efficiency:"avg_purchase_motivation_A.avg > avg_purchase_motivation_B.avg"
- Social Network Generator:** max_edges: 5, min_edges: 1, directed?: false, degree_distribution: :homogeneous
- Population:** 100
- Repetitions:** 1

At the bottom, there are two blue buttons: 'SAVE' and 'EXECUTE'. Below the buttons, there is a 'Results:' section which is currently empty.

FIGURE 26 - SNAPSHOT OF THE TEXT-BASED GUI FOR DEFINING A POLICY MODEL.

4.3.6 Integration of the Social Dynamics Component with PolicyCLOUD

Following the specifications described in section 2.2.5, Politika implements a REST API which exposes the functionalities required by the end-users of the Social Dynamics component of PolicyCLOUD. The REST web methods are invoked by the generic bridge function to allow the interaction between the two platforms.

4.3.7 Simulation and Policy Models for the Agri-Food Pilot Use Case

In Politika we have developed a simulation model for consumer behavior towards the purchase of wine in a population. The model assumes that price and quality are the main factors influencing consumers when purchasing wine. In addition, consumers can be influenced by their exposure to wine-related advertising/marketing campaigns and the wine preferences of their social circle. Based on these assumptions we define the following set of parameters of interest for estimating the purchase motivation for a particular brand of wine (e.g., A) in a specific region:

- actual price for A
- quality (in a scale of 0 to 1) of A as determined by its average rating in a series of online reviews
- estimate of the average price of wines sold in the region of interest
- estimate of the maximum price for wine that is acceptable for an average consumer (for example this can be double the average price of wines in the region of interest)
- average quality of the wines sold in the region of interest (0 to 1)
- average income of the population in the region of interest
- maximum income of the population in the region of interest (e.g., this can be the maximum income within two standard deviations of the mean assuming a normal income distribution)
- average relative exposure of individuals to the advertising campaign for A (0 to 1).

We assume that average ad exposure is proportional to the relative size of the ad budget for A compared to its competitors but also to the type of ad campaign used (e.g., targeted or undifferentiated). We further assume that the population in the region of interest is represented as a social network where each node corresponds to an individual. For each individual (e.g., X), each outgoing edge is labeled with a weight representing the influence that X exerts on the wine purchasing decisions of one of its social connections. Each individual has a set of attributes that are relevant towards A. These include X's:

- Income ranking (in a scale of 0 to 1) as determined by the ratio of its income to the maximum income for the region.
- Sensitivity to the price of A as determined by the product of the difference of 1 minus X's income ranking times the ratio of the current price of A to the maximum wine price in the region. According to this estimate, poor individuals are more sensitive to the price of wines compared to richer ones.
- Sensitivity to the quality of A as determined by the product of X's income times the ratio of the current quality of A to the average quality of wines in the region. According to this estimate, rich individuals are more sensitive to the quality of wines than poor ones.

- Susceptibility to the advertising/marketing campaign for A (0 to 1).
- Tendency to follow the attitudes towards A of its social circle (0 to 1).
- Perceived influence for A from X's social circle. This is computed as the average purchase influence for A stemming from its social circle.

Based on these attributes the model estimates X's purchase motivation for A as a linear combination of:

- X's price sensitivity for A.
- X's quality sensitivity for A.
- The product of X's advertising susceptibility for A to the intensity of A's advertising campaign.
- The perceived influence for A from X's social circle.

Based on this simulation model the policy maker can design a policy for improving the purchase motivation of a specific wine versus a competitor in Politika. To this end, the policy maker provides data from specialist wine sites (e.g., prices, quality) and Wikipedia (e.g., max income) for two competing wines, e.g. A and B, related to the parameters of interest in our simulation model.

The policy maker can then define, simulate and evaluate various alternatives for pricing and/or advertisement effort for A in order to discover the mix that could improve A's average purchase motivation in the population with respect to B in a specific population based on a set of evaluation criteria defined by the policy maker.

We validate the computed relation ($>$, $=$, $<$) of the purchase motivation for A versus B in the base case using as proxy the relation between the number of ratings for them in wine specialist sites (if A has a higher number of ratings than B this means that more people have purchased A therefore, we can assume that currently the purchase motivation for A is greater than the one for B). Simulation results are visualized as charts that facilitate direct comparisons between the alternatives explored. These charts are generated by linking the Social Dynamics component with the Visualisation tool in PolicyCLOUD via an analytic function. For example, Figure 27 displays:

1. in the x-axis the ratio of price of X vs Y defined for each alternative
2. in the y-axis the ratio of the advertisement intensity of X vs Y for each alternative
3. the ratio of purchase motivation for X vs Y computed via simulation for each alternative as a sphere with an area and color proportional to the value of this ratio.

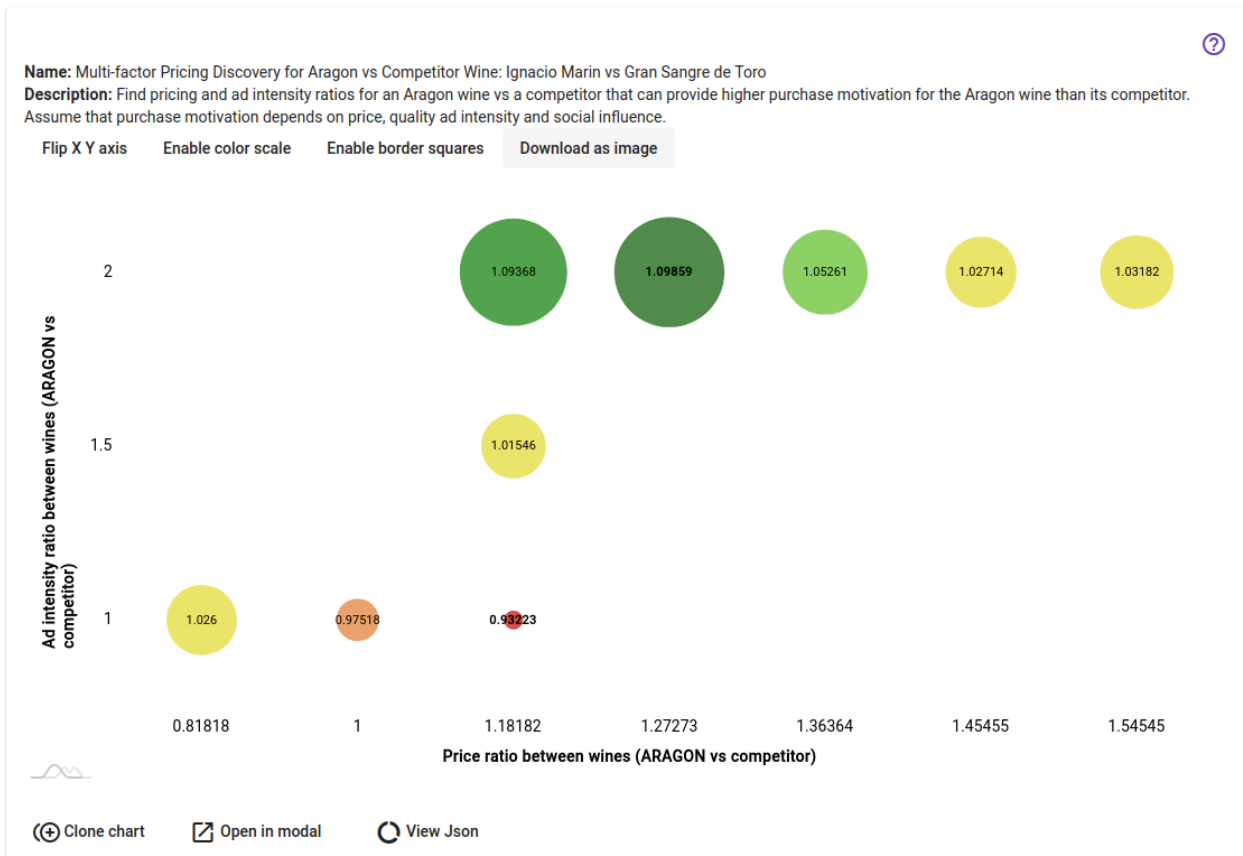


FIGURE 27 - EXAMPLE. CHART FOR THE SOCIAL DYNAMICS OUTPUT IN THE AGRI-FOOD USE CASE (GENERATED BY POLICYCLOUD)

As the chart in Figure 27 shows, the best policy alternative among the nine that were simulated occurs for a price ratio of 1.27273 between X versus its competitor Y and for a doubling of the intensity of the ad effort for X versus Y. At this point we compute an almost 10% higher purchase motivation for X versus Y (a 1.09859 value for the ratio of the average purchase motivation of X versus Y). This is much better than the base case that estimates the current status between these two wines with no policy applied. For this base case we compute an average purchase motivation for X lower than that of Y (a 0.93223 ratio of average purchase motivations) at a price ratio of 1.18182, assuming the same intensity of the ad effort between X and Y. Therefore, for the best policy alternative compared to the base case we were able to increase both the purchase motivation for X (almost 18%) and the price for X (almost 7%) but with a doubling of the intensity of the ad effort for X versus Y. The identification and estimation of such trade-offs between policy variables is one of the advantages of using the Social Dynamics component in PolicyCLOUD for policy design.

4.3.8 Updated Simulation and Policy Models for the Radicalization Pilot

Use Case

We have updated the simulation model in Section 4.3.3.4 in order to capture important aspects of policies for radicalization control. In particular, the policy parameters of the original model have been extended with the following parameters and their meaning:

- **restriction_threshold:** Individuals with a `radicalization_status` above `restriction_threshold` are restricted, Under restriction an individual cannot communicate with her social circle and, therefore, cannot receive or apply social influence to its social network.
- **restriction_duration:** The number of time units for which a restricted radical remains restricted
- **restriction_cost_per_head:** The cost of restricting an individual for a unit of time
- **policy_cost:** The overall cost of applying a restriction policy over the simulation period computed at each time unit as the number of restricted radicals times the `restriction_cost_per_head`.

The goal of the policy in the updated scenario is to meaningfully decrease the percentage of radicals in the population from its initial value with minimal cost. This can be done by an and/or combination of enforcing a restriction regime (based on discovering appropriate values for the `restriction_threshold` and `restriction_duration` parameters) and of reducing marginalization in the social group (by exploring appropriate values for the average number of connections for each individual). Figure 28, Figure 29 and Figure 30 provide visualization examples done in the Social Dynamics component for managing network data, displaying the values of individual attributes in the population, and providing a connectivity matrix between all the individual nodes in the radicalization example.

In particular, Figure 28 depicts part of the social network graph used in one of the simulations. Each node corresponds to an individual and has its own color. Each edge represents a directed connection between two individuals with its own color as well. The contents of each node clicked are shown in the text area below the graph, which the user can edit and then update the attributes of an individual with the UPDATE button at the Node row in the Figure. The user can delete a clicked node with the REMOVE button of the Node row or create a new individual with the NEW button at the end of the Node row in the Figure. Furthermore, the user can create a new edge between two nodes with the NEW button at the Edge row or delete an existing one with the REMOVE button at the same row.

Figure 29 provides an example for a chart that describes the values of the numerical attributes for each individual node in the social network graph. The structure of the specific chart type is described at the top of the Figure.

Figure 30 describes the connectivity (adjacency) matrix between the nodes of an example social network.

4.3.9 Future steps

Future steps by the end of the project will focus on further improving Politika following the feedback on the use of Politika with the actual use cases of the PolicyCLOUD project and on developing GUIs that facilitate policy designers to interact with the system. These will focus on changes in the PolicyCLOUD

interface to Politika that can improve the ‘look and feel’ of the forms used in such interactions based on the development of a profile for their average user.

4.3.10 Conclusion

Policy design is typically modeled as following a five steps process [36] [37] [38] [39] simply described as:

1. Define a policy problem
2. Identify feasible alternatives.
3. Use a set of criteria to compare alternatives.
4. Predict the outcome of each alternative.
5. Make a recommendation.

Our meta-simulation environment is consistent with this model. In particular, step 1 is implemented by providing a base case simulation model of the social dynamics with no policy intervention and serves to describe the problem under investigation. Step 2 is implemented by representing the goals and methodology of each alternative as a Goal-Objective-Step hierarchy. Step 3 is implemented via the ability to specify criteria at the Goal level. Step 4 is implemented with the execution of the simulation Steps in the design hierarchy. Finally, Step 5 is implemented by evaluating and ranking the criteria specified at the Goal level. In general, it is possible for a PolicyCLOUD user to access any of the implementations of these steps through the definition of appropriate functions that exploit the REST API mechanism provided by PolicyCLOUD.

Most of the current ABMS systems [39] offer either an Application Programming Interface over a general-purpose programming language (e.g., NetLogo [45] using a dialect of Logo, Repast HPC [46] using C++) or a special-purpose modelling language (e.g., Gama [47] using Gaml). In Politika we have opted for the second approach and designed a simple rule-based language for agent-based modeling, while providing an efficient execution environment that hides algorithmic and implementation complexity of preparing and executing a simulation from the modeler. The logic behind this decision is to facilitate examination and comparisons between different policy alternatives from users that lack sufficient programming knowledge as is often the case in policy design. Although simple by design, the language can express models of endogenous social dynamics arising from local interactions, exogenous dynamics arising from global interventions on the network or combinations of the two.

Agent-based models for social dynamics are receiving a lot of attention because of their ability to simultaneously model interactions at the individual level, at the society level and between these two levels [35]. Criticisms for this paradigm often focus on the ad-hoc character of the models that are developed and on the difficulty of obtaining empirical datasets of populations in which to apply these models [48]. However, these models are mostly driven by processes and mechanisms inspired by a social or behavioral sciences theory, so they have theoretical underpinnings. Furthermore, there are indications that their outcomes are consistent with empirical data [49].

With regards to modeling policy alternatives there is always an ad-hoc character in their creation since policy analysis and design is essentially a political process, and, as such, it reflects political opinions in framing and solving problems [50]. Politika seeks to make this process transparent, efficient and effective

by providing a web-based framework for modelling, comparing and ranking different alternatives. Such a framework can facilitate examination of models and comparative analysis of their assumptions, mechanics and outcomes. This will make political influence behind goals and objectives transparent and enhance the quality of discourse around each policy. To the best of our knowledge there exists no other ABMS meta-simulation environment for policy analysis and design.

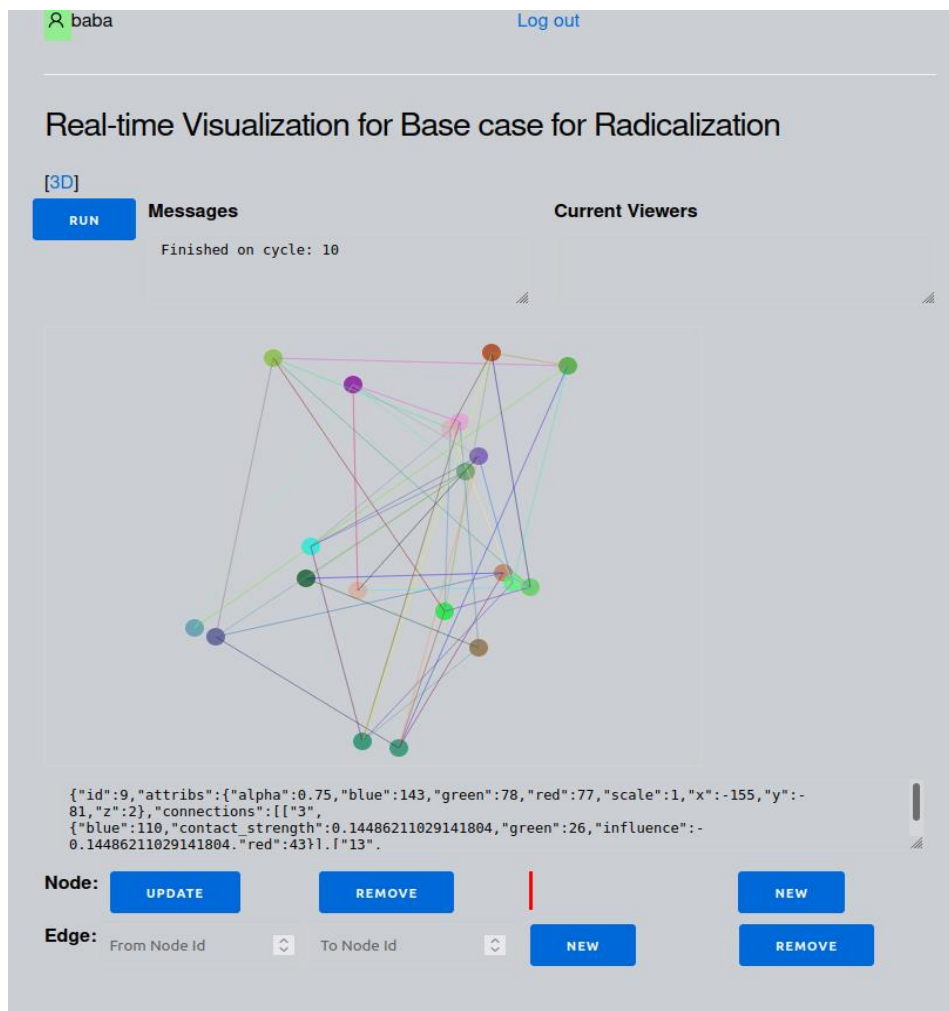


FIGURE 28 - POLITIKA GUI: MANAGEMENT OF NETWORK DATA AND EXECUTION OF SIMULATIONS

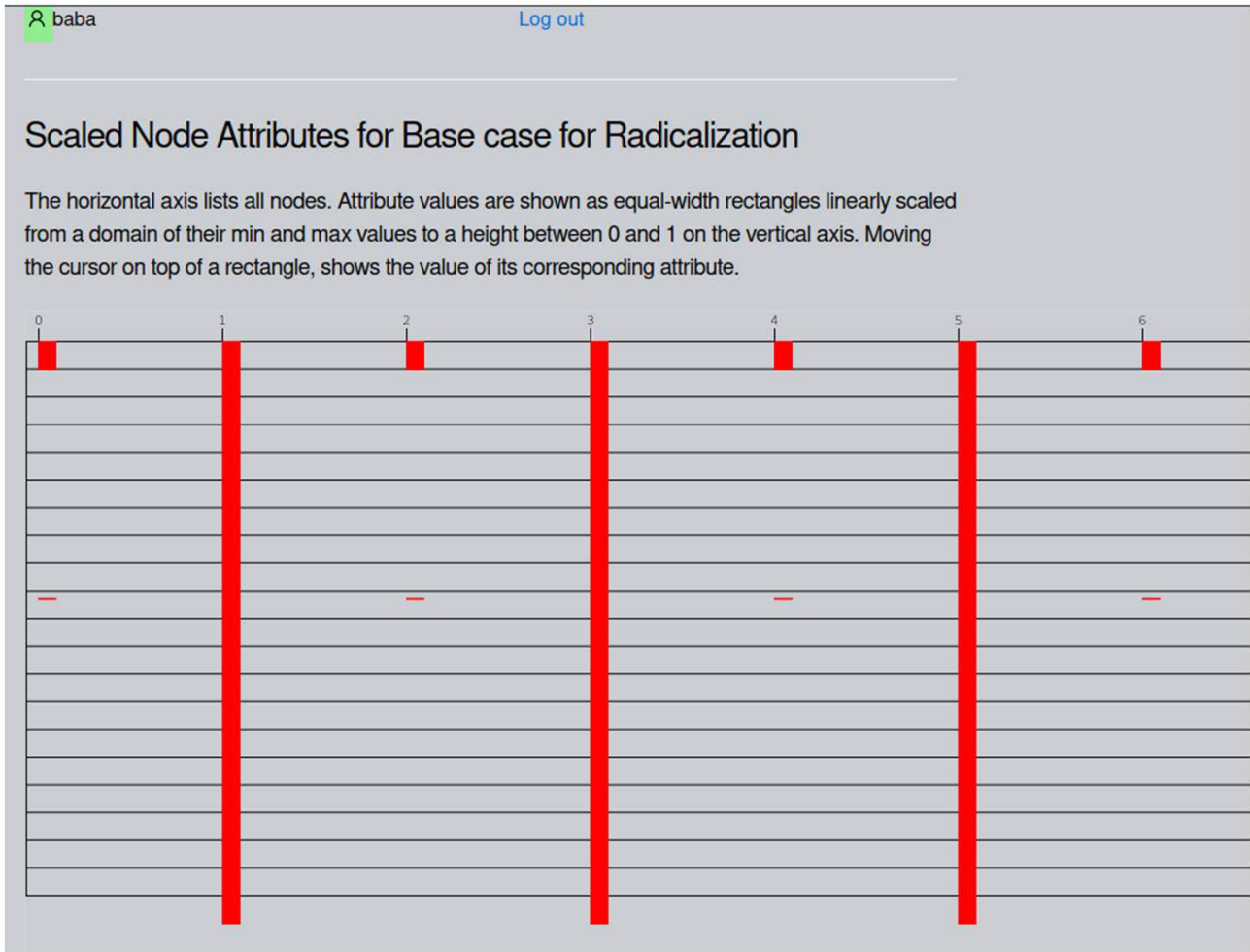


FIGURE 29 – VISUALIZATION EXAMPLE OF SCALED ATTRIBUTE VALUES FOR EACH INDIVIDUAL IN A POPULATION

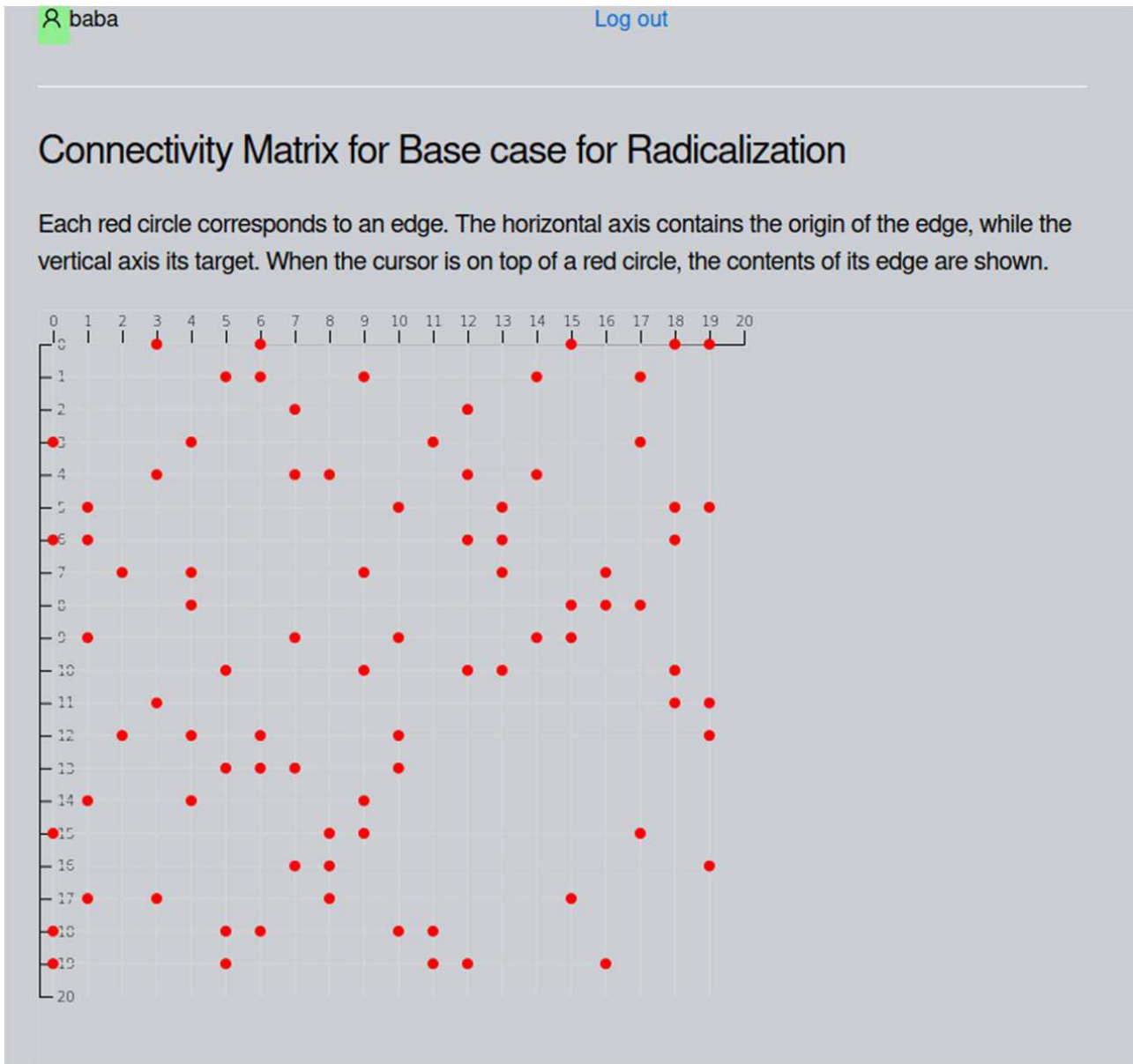


FIGURE 30 – VISUALIZATION EXAMPLE OF A CONNECTIVITY MATRIX

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

4.4.1 Updates since D4.3

This section has been further evolved and extended from its previous version in D4.3. It now provides more information regarding the user stories and our motivation, while the subsection concerning the architecture has been also updated. We provide more details on the advancements that took place during this last phase of the project, giving concrete examples on how we altered the implementation of the SQL *join* relational algebraic operation. Finally, we integrated the data management layer of the PolicyCLOUD with external ETL pipelines AWS Glue.

4.4.2 User Story and Motivation

The context of this section is based and partially reproduces the content of Chapter 7 “Seamless Data Analytics Framework” of the WP4 Scientific Report and Prototype Description – Y3 of the H2020 BigDataStack project [51].

Streaming data is one of the typical data handled by PolicyCloud (see section 2.1.2). This may demand high-rate ingestion within the LeanXcale (LXS) operational data store of data coming from different sources. Thanks to its ultra-scalable transactional manager and its API which permits to directly ingest data on the storage layer bypassing the SQL query engine while still enforcing transactional semantics, LXS can handle highly intensive workload, while at the same time ensuring online analytical processing (OLAP).

Here two critical observations must be made: a) when dealing with high volumes of Data, old or historical data typically becomes ‘cold’, that is no longer subject to modifications. b) Object stores for ‘cold’ data are now considered superior to traditional databases in terms of cost, simplicity, and scalability. These advantages may however be offset by the need of highly performant OLAP analytics on ‘warm’ data.

The basic idea of the “seamless” component is to present a traditional relational database that ensures transactions, such as LeanXcale, and an Object Store as a single logical data store that will provide real-time data warehouse capabilities. One of the implications of seamless is the need to move under the hood historical data slices from the operational data store to the object store.

In general, presenting data spread over two physical data stores as a single logical dataset is complex since the data retrieved for a given query from each of the two stores must be merged at the application level. This is a non-trivial task both in terms of interoperability of data stores as well as of efficiency.

This demands from the application level to be able to perform operations that do not normally belong to the application level and which a database can do more efficiently. Moreover, moving data from one data store to the other introduces significant data consistency concerns that would typically require operation freeze during the data migrations. This is needed because of race conditions that typically occur when concurrent transactions retrieve data from two stores, while a data slice is being moved from the one to the other at the same time.

The Seamless Analytical Framework (SAF) was designed to provide a common interface for the application developer to query the dataset, without having to know neither where the data are actually stored nor the query semantics of the different data stores. The SAF provides a common JDBC implementation that supports standard SQL statements for the end-user to retrieve information and hides the specificities of the data stores. Moreover, it ensures data consistency when moving data from the operational data store to the Object Store, without requiring any freeze. The SAF relies on the transactional management component of LeanXcale to ensure that data records will not be retrieved from both LeanXcale or Object Store even when data is being migrated.

Finally, it supports all standard SQL commands regardless of the complexity introduced by the need to handle the distributed execution of the command. The SAF hides from the end-user all the lower-level

details about data management and retrieval, thus presenting the operational database part and the object store part of the dataset as a single logical dataset. It provides an interface with which the data analyst or application developer can submit standard SQL statements, whose execution will be done seamlessly.

In addition, as previously mentioned, the SAF periodically and seamlessly moves the historical parts of the dataset from the database to the object storage.

With SAF in place, the data scientist can benefit from the advantages of the two data stores as it allows her to perform analytics on fresh data, as data is being currently ingested. Thus, the policy maker can use the AI algorithms provided by the data scientist to perform analytics and retrieve useful results, relying on always up-to-date data, without having to perform analytics on outdated data that has been previously migrated to a data warehouse or an object store. Thus, policy making can benefit from the two distinct worlds of the data management layer: the operational and the analytical ones.

4.4.3 Architecture

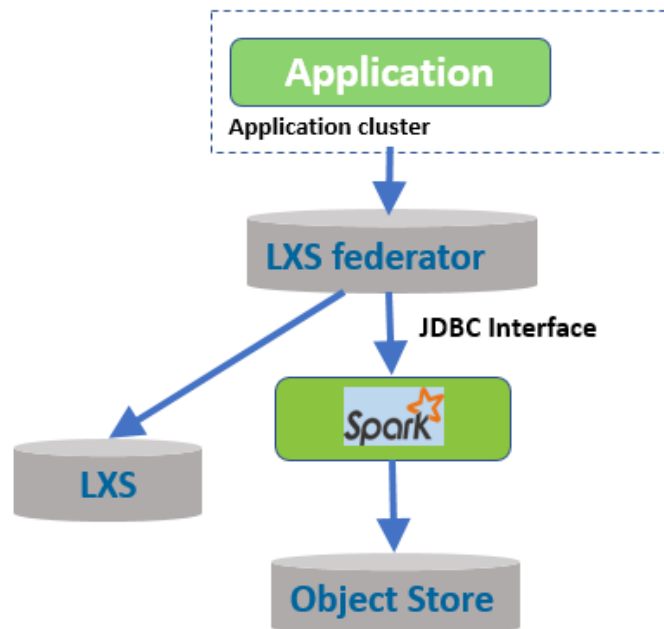


FIGURE 31 – SEAMLESS HIGH-LEVEL ARCHITECTURE

Figure 31 - Seamless high-level architecture gives a high-level conceptual view of the seamless architecture. It shows a query execution over a federated dataset residing both in LeanXcale and the Object store. Typically, the logical dataset is split between the two different physical datastores: the operational data store of LeanXcale and the analytical object store. However, from the application or data scientist point of view, there is a single logical database, whose access is provided by a standard JDBC connection. As a matter of fact, the application or data scientist can submit a standard SQL query via the standard JDBC connectivity mechanism to the *query federator*. The latter is responsible for executing the query to both physical datastores, merge the intermediate results and return a dataset

that is equivalent to what it would have been returned if the overall data was stored in a single physical data store.

It is important to highlight at this point that the target datasets that can benefit from the *seamless analytical framework* are those for which the values of one of their keys or indexes (single-columned or compound) are monotonically incrementing (or decrementing). Those are datasets whose primary keys are auto-generated from a *sequence* or IoT data that usually contains a *timestamp* field. Based on the timestamp, the dataset can be split periodically in different ranges of these values (we call the dataset between Value1 and Value2 as a *data slice*). *Data slices* are moved periodically from the operational data store to the object store and the *query federator* keeps internally which is the current maximum value of the slice that is contained in the object store. As a result, it can restructure the input query using a filter condition on the column that has been *sliced*, execute the two independent queries ad-hoc, whose results will not be overlapping, and finally unify the two intermediate results.

Depending on the type of operations in the query plan of the submitted statement, the *query federator* unifies the intermediate results differently. Firstly, operations that require the physical access of a data table are moved as low as possible down to the query tree. All other operations are executed by the *query federator* itself and cannot be pushed down to be executed remotely. For scan operations (with or without filter conditions), the final result will be the union of the two intermediate results. For aggregated operations (i.e. sum, count, avg, min and max) the result will be calculated as follows: the final min will be the min of the two intermediate mins, the final max will be the max of the two intermediate maxes, the final count will be the sum of the two intermediate counts, the final sum will be the sum of the two intermediate counts, and the final avg will be the final sum divided by the final count. Regarding the *join* operator, this will be translated into 4 individual and non conjugated joins. Two will be executed locally: one in the operational data store and one in the object store. The remaining two will be translated to a *bind join* execution, and the values that take part in the equity join will be sent from the operational data store to the object store. Finally, the *query federator* will unify all 4 intermediate results and will return back to the client the equivalent result set,

We refer the reader to [51] for additional details. In particular this reference explains how SQL JOINS queries are handled efficiently when they target datasets which are split between the LXS data store and the object store.

4.4.4 Seamless within Policy Cloud

As depicted in Figure 32, the seamless analytical framework consists of various components. First, the vanilla Object Store, along with the vanilla main building blocks of the LeanXscale operational data store. For the latter, we distinguish its storage layer from the query engine. For the former, we notice an additional layer using Apache Spark that facilitates data retrieval by exposing a standard JDBC connectivity mechanism.

SAF uses those components, without the need to further extend their codebase. Instead, it consists of three additional building blocks that a) allow the query execution over the federated dataset that is stored in both stores and b) move historical or “cold” data from the operational store to the object store

in a seamless manner. For the first purpose, the implemented *Query Federator* uses the core codebase of the *Query Engine* and extends its base functionality that now allows the execution of such queries. For the second purpose, the *Data Mover* has the responsibility to retrieve the data slice of the now ‘cold’ dataset and migrate it by storing it to the object store. Finally, the SAF orchestrator triggers the process of moving data from one store to the other. More details about the sequence of actions that are being executed during this process can be found in [51] .

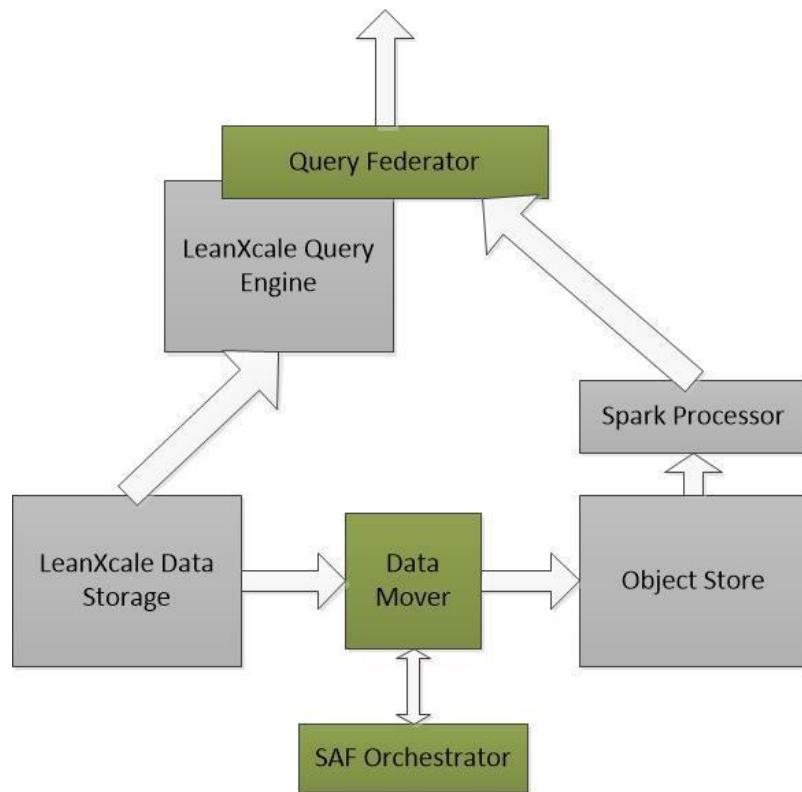


FIGURE 32 – SEAMLESS ANALYTICAL FRAMEWORK WITH DATA MOVER

In order to make use of SAFunder PolicyCLOUD, a series of tasks has been performed in advance. Regarding the Query Federator, its codebase has been upgraded to be compatible with the current release version of the LeanXcale datastore. As it has been mentioned, this component makes use of core libraries of its query engine. At the time when the initial version of SAF was delivered under the scope of the H2020 BigDataStack, the release version of the database was 1.5.5. At the time when this deliverable is submitted, LeanXcale has distributed to PolicyCLOUD its current master release version 1.8.34. The main change between the two versions is that the latter is now using a more recent version of the Apache Calcite query processing framework, which is the core of the query engine. This affects the provision of *table functions* that are used internally by the *query federator* to implement the corresponding functionalities. Other changes are related with the provisioning of the TCP protocol for the communication between the clients and the query engine (that is the query federator as the client application and the vanilla query engine of LeanXcale) which improves the overall performance and allows for long running query executions.

Moreover, the LeanXcale Data Storage engine has been upgraded to a newer version that is included in the current distribution provided to PolicyCLOUD. As a result, its internal API for accessing the storage has been modified. LeanXcale provides a direct java-based API that encapsulates the specifics for accessing the data storage, as its method signatures are considered stable. However, for components that need to access the storage layer, such as the *data mover*, they need to be upgraded and make use of the current release version of this API. The current 1.8.34 version of LeanXcale is a major update from the previous one, and therefore, some methods have been deprecated. As *data mover* is accessing the storage using Apache Spark internally, it would require making use of the current version of the corresponding connector. As the connector implements the standard interface of Apache Spark, it ensures backward compatibility with the previous version of the direct API and no major change in its codebase is foreseen.

Finally, the *SAF Orchestrator* communicates with the *Data Mover* in order to start the process of migrating data from the relational database to the object store. Internally, it makes use of a scheduler that periodically sends messages to the *Data Mover* that includes the definition of the data slices to be moved, while also handles failures of the migration process by applying a re-do strategy. The scheduler is configured via the use of a static configuration file. This comes with an inherent drawback: if the database administration wants to change this configuration, she will need to update the file and restart the specific process. In H2020 BigDataStack, we envisioned that this process should happen dynamically by using an extended SQL DDL statement. However, this task was considered as low priority and finally was not delivered. In PolicyCLOUD, we have designed this functionality that will affect the codebase of the *SAF Orchestrator*. However, this can happen in a later phase as it does not affect the migration of the overall SAF to PolicyCLOUD.

In the scope of the H2020 BigDataStack project, SAF was firstly demonstrated with a specific use case in which IoT sensor data was stored in the persistent datastore. IoT data are usually *insert-only*, they contain a timestamp field and after a (use case specific) period has passed can be considered as historical. At this point they are eligible to be migrated to the object store. The implementation at that point supported typical query operations that can be found in data management systems, such as *scans, filters, projections, sorts, limitations* and *aggregations* (with *groupby* support). In PolicyCLOUD we have similar scenarios with data being collected from social media or datasets that consist of a single table and are frequently updated by inserting additional data items with newer timestamps. As a result, SAF can be used to optimize the query processing needed by the analytical tools and exploit the benefits of our component. Our goal is to validate our approach by using it for scenarios pertaining to different domains and prove that it is domain agnostic.

Furthermore, during the last phase of the H2020 BigDataStack project, we also achieved to implement the *JOIN* operation at the *Query Federation* level. This allowed the application of SAF in use cases that utilize a relational schema. The requirement for the data tables that can be split across the two stores remains the same: they need an increment column (i.e., a timestamp field) and the table must support insert and select statements. This is a common case for the majority of the user scenarios, which extends the applicability of our solution to wider domains. However, our background technology had an inherent constraint that forbade us from using it when accessing big volumes of data and this constraint has been removed in our current release developed under the PolicyCLOUD as we describe below.

As mentioned in the previous subsection, in the case of the *join* operator, the latter is being broken down into 4 individual and non-conjugated operators. Let us take the example of two data tables that have been both split among the LeanXcale and the Object store, as depicted in Figure 33

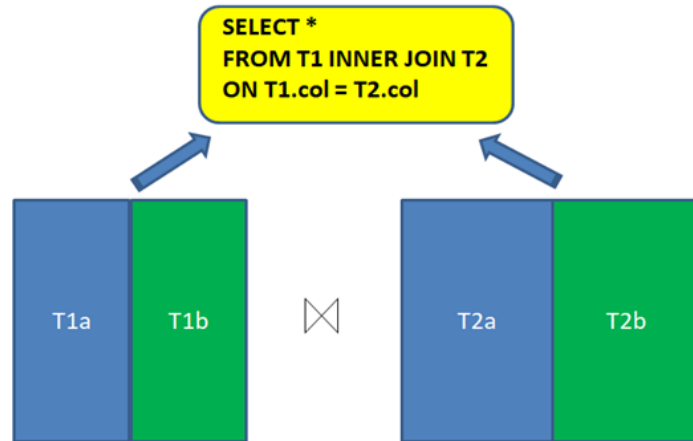


FIGURE 33 – JOIN OF 2 SEAMLESS DATASETS

The single *join* of the two tables can be transformed to the union of the following 4 *joins*:

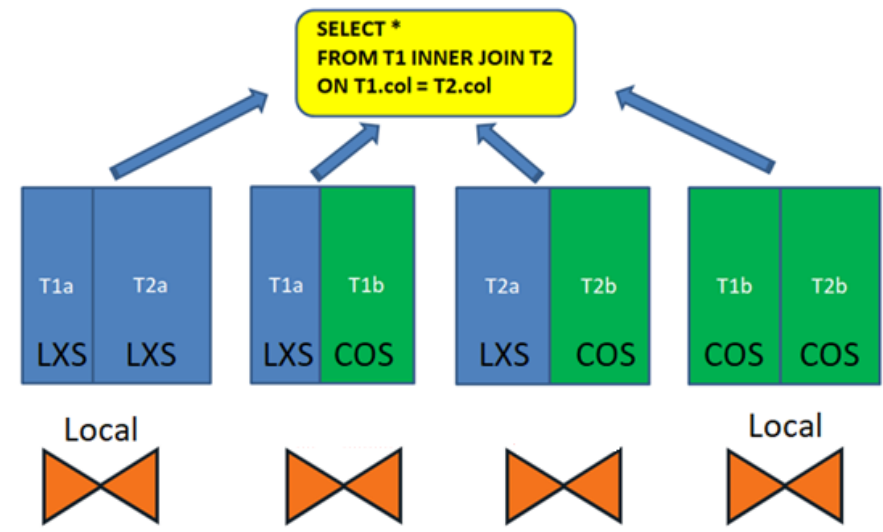


FIGURE 34 – BREAKDOWN OF THE JOIN

As depicted in above Figure 34, 2 of them will be executed locally either within LeanXcale or within the Object Store, while the remaining two exploit the use of the *bind join* implementation. The latter demands that the values of the columns of the records of the left part of the join (which contains data stored in LeanXcale) will be sent to the right part of the join (which contains data stored in the object store) and let the latter execute that part of the join. This is being done by transforming this operator to an *IN* clause. As a result, a list of values is sent from the *query federator* to the object store, using a JDBC connection over Spark. The main obstacle of this design principle that forbade us from using our implementation in real life scenarios involving big volumes of data, is that Spark internally uses a *Set* of this values. This implies that the Java Virtual Machine will keep one *reference* to the *Set* itself, and one

reference to each of its objects. When it comes to big data, the number of objects contained in the *Set* is very high, so *StackOverflowExceptions* can easily occur in the Java Virtual Machine.

In order to overcome this, we designed an alternative: instead of sending a list of objects for the *IN* operator, we are creating a temporal data table in the object store, store the data there, execute the hybrid *join* operator now locally in the object store, and drop the table after the *query federator* has gathered all the results. This is much more expensive, as it now involves 4 different operations (create table, store data, execute join, drop table) instead of a single one. I/O access is first required to store the data before executing the join. This is not critical since the 4 subjoins are executed in parallel, and since the longest one is the local join at the object store. This is because it contains the majority of the data. As a result, it will take more time for this operation to finish, compared to the overall time for creating the temp table, storing the intermediate data, executing the local join and finally dropping the table at the end.

ETLs targeting PolicyCLOUD with Seamless

Another implementation that helped PolicyCLOUD to reach the objectives of the task T4.6 is the integration of the data management layer with the AWS Glue. Taken from its official website documentation, *AWS Glue is a fully managed ETL (extract, transform, and load) service that makes it simple and cost-effective to categorize data, clean it, enrich it, and move it reliably between various data stores and data streams.* AWS Glue is also serverless, and thus, it is quite similar with the data ingestion pipelines that PolicyCLOUD natively provides. This gave us the idea during this last phase of the project to enable the integration of the PolicyCLOUD with the AWS Glue to allow the use of our integrated solution with customers already familiar with the AWS technology. As a matter of fact, we now allow data providers to either use the functionality provided by the PolicyCLOUD platform in order to ingest data that can be later exploited by the analytical tools or re-use their own ingestion pipelines (the AWS Glue ETLs) to direct the data ingestion to the data management layer of PolicyCLOUD.

AWS Glue provides its own *connectors* with target datastores. The *connectors* need to implement the DataSource API v2 of Spark, which is an open API to allow seamless data connectivity with proprietary data management systems. As already mentioned in the previous subsection, our *Seamless Analytical Framework* makes use of the Spark connector for the *data mover* to retrieve the *data slices* from the operational data store of LeanXcale. For this to be achieved, LeanXcale provides a Spark *connector* that allows Spark to execute queries. However, the Spark *connector* previously implemented the DataSource API v1. During this last phase of the project, we advanced our implementation to support the DataSource API v2. Having that in place allowed the integration of the data management layer of PolicyCLOUD with AWS Glue.

5 Cloud Platform and Software Tools

In this section we'll give a 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 Updates since D4.3

By the near end of the project, we do not report any significant evolution from D4.1 for the Cloud Platform and the Software tools. This proves the soundness of our initial choices.

In each of the subsections we report how our initial architecture choices as reported in D4.1 were indeed good fits for the PolicyCLOUD as of the submission of this deliverable.

5.2 Kubernetes cluster

The first design question for the PolicyCLOUD platform 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 ecosystem development, especially in the open-source communities

Once the container direction was decided, the Kubernetes¹⁷ container management platform was a clear choice. Kubernetes is the leading open-source container management platform used in production of a 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.

As of D4.3 we can report that all software components of PolicyCLOUD have been deployed on a Kubernetes cluster which was chosen as the underlying application management platform. In particular, this choice eased the integration of the “seamless” technology (see section 4.4 in this document).

¹⁷ Kubernetes is an open-source container-orchestration system for automating computer <https://kubernetes.io>

5.3 OpenWhisk cluster

Apache OpenWhisk¹⁸ is an open source lightweight 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 of transparently. It is based on containers as the functions' wrapper and deploys and integrates perfectly on a Kubernetes environment. As described in section 2.1.2, all analytic functions in PolicyCLOUD are deployed as OpenWhisk functions, with appropriate triggers and activation rules, addressing the extensibility and reusability requirements. Additional important capability of OpenWhisk is the composition of functions to be activated per trigger/rule, which enables to specify multiple analytics to be applied in series to a data source.

By the near end of the project, we can report that this choice has proven itself, in particular in view of the many fully integrated use cases that were demonstrated during the various reviews and co-creation workshops in 2021 and 2022 where in particular Apache OpenWhisk was instrumental to the implementation of the data demonstrated pipelines.

5.4 LeanXscale database

The LeanXscale database is an integral part of the PolicyCLOUD data repository, along with the *object store*. It is a relational data store, which ensures transactional semantics and provides ACID (Atomicity, Consistency, Isolation, and Durability) properties and offers a JDBC interface, supporting standard SQL queries. Moreover, it provides an internal key-value store that can be used in cases where very low latency is a requirement to support data ingestions at very high rates, achieving a very high throughput.

The database consists of three main pillars: the data storage itself, the scalable transactional manager and the relational query engine. Each of those pillars can be scaled 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 in traditional implementations, and therefore allows 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 the operational data store that has all recent updates, and not on a data warehouse that typically holds an outdated snapshot of the dataset. Analytics take place as data are being ingested into the platform, thus removing the necessity of performing cost-expensive ETLs for migrating data from the operational data store to a data warehouse that can be used for these purposes. Finally, it provides polyglot capabilities for federated query processing over different data stores and can be used as a common endpoint to retrieve and pre-process data from external data sources in a seamless manner.

Figure 35 depicts the main architectural components of the LeanXscale data store.

¹⁸ Apache OpenWhisk is an open source, distributed Serverless platform that executes functions <https://openwhisk.apache.org>

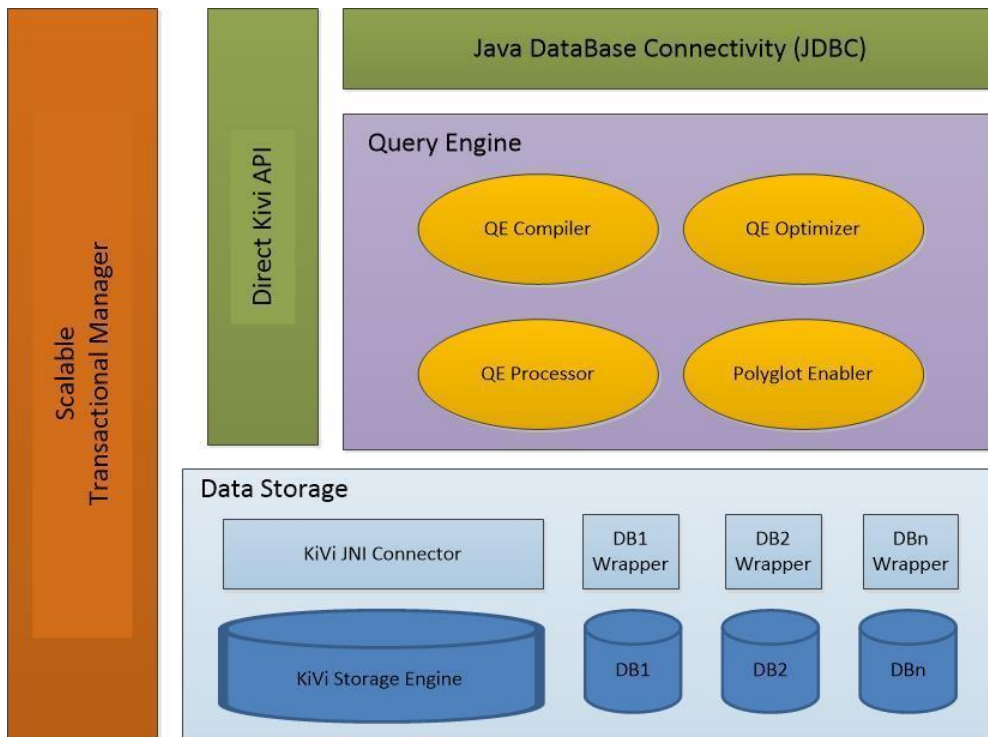


FIGURE 35 - LEANXCALE DATASTORE ARCHITECTURAL DESIGN

In the architectural design, we can see the three main pillars. At the bottom layer lays the data storage component. Its central component is the KiVi storage engine, which is responsible for persistently storing data items to the disk. It allows data items to be stored in a tabular format, providing additional 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 take advantage of it 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. The KiVi storage engine is implemented in the C programming language; however, it also provides a JNI Connector that wraps this functionality over a Java-based implementation. 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 to allow 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 users to execute queries that can join datasets that are either stored internally in the PolicyCLOUD repository, or they are persistently stored in external data stores.

On top of the Data Storage component, there is the Query Engine of LeanXcale. It supports statement 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 transformed into 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 demand 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 can be executed in a distributed manner: for query operations that can be pushed down entirely to the storage engine (i.e. scans, filters, aggregations etc.) they are executed in a distributed manner among the data nodes, while for operations that need to be implemented in the relational query engine (i.e. joins) their execution is also distributed among the nodes of the query engine. This allows the LeanXcale database 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 depicted in Figure 35, the 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* is depicted vertically in the overall architecture since it 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 inherent bottleneck that those protocols suffer, and as a result, our solution can scale out adequately and can serve operational workloads coming at very high rates. Both the query engine and the storage engine interact with the *transactional manager*, which ensures transactional semantics even if the data is directly ingested in the data storage and the data analytics are served by the query engine.

Regarding data connectivity, there are two ways for accessing the LeanXcale datastore: Either with the direct KiVi API, or with a JDBC implementation. The former exposes a MongoDB-like interface and accesses the storage by bypassing the query engine. This removes the inherent footprint that the query engine natively introduces and can be used in cases when there is the need to support data ingestion at very high rates. It makes it possible for the application developer and the data analyst to exploit the unique characteristics of the storage layer; however, it comes with a drawback as its API is not standard. In order to overcome this, we provided specific connectors for a variety of popular frameworks that can be used instead, such as Apache Spark, Kafka, NiFi etc. Moreover, 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 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, either by using JDBC connections or by using the LeanXcale connectors. That way, we can avoid the transmission of the whole dataset to those frameworks and instead, perform all pre-processing for data retrieval locally at the nodes where the data is stored. 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 bare metal or over a container orchestration tool such as docker. The LeanXcale datastore has been containerized, which facilitates the deployment over a Kubernetes cluster. It is important to mention that when using Kubernetes, the whole datastore installation can either be deployed within a single pod or over several pods. Moreover, as previously stated, each of the components can scale out independently. This allows for a great flexibility of deployments, as the database administrator can scale the data storage and query engine nodes for supporting increased data load, or the transactional manager components for supporting increased operational workload with hundreds of thousands of transactions per second. The administrator activities that might be required for fault-tolerance or to scale out the deployment can be facilitated with the automation process provided by Kubernetes.

5.5 Spark cluster

Apache Spark¹⁹ is a highly popular open-source analytic engine for large-scale data processing. It achieves high performance for both batch and streaming data and is powered by numerous open-source libraries such as SQL, streaming and machine learning to manipulate and query data. By D4.3 Spark has already been used in PolicyCLOUD in conjunction with the LeanXcale database to provide seamless analytic on hot and cold data at rest (see section 4.4), where the Spark SQL with optimization exploited from the BigDataStack EU²⁰ project can be used for queries on the colder data in the object storage.

5.6 Object storage

Object Storage is a storage architecture providing high capacity at low cost and is a definite choice for big data that is once-written, i.e., is not going to be modified after being written (to modify any part of an object, it needs to be fully deleted and re-written). An object can have metadata (that is used e.g., to increase analytics performance by skipping irrelevant objects), and in contrast 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 and the objects are accessed in 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 as now demonstrated in section 4.4.

¹⁹ <https://spark.apache.org/>

²⁰ <https://bigdatastack.eu/content/seamless/>

6 Conclusion

In this final version of the Design and Open Specification we provided important updates at the architecture level with the “Integration of External Frameworks” (2.2.5), for the legal/ethical framework with the details of how a registrant, as well as for data analytics with new technologies such as trend analysis (4.3).

The Reusable Model & Analytical Tools work package allows a flexible exploitation and management of policy-relevant dataflows in PolicyCLOUD. This is done by enabling the practitioner to register datasets and specify a sequence of transformations and/or information extraction through registered ingest functions. Once a possibly transformed dataset has been ingested, additional insights can be retrieved by further applying registered analytic functions.

The architecture presented in this deliverable makes PolicyCLOUD an extensible framework which positions it well as an analytic ecosystem. We have presented several essential ingest and analytic functions that are built-in within the framework. They include data cleaning, enhanced interoperability, and sentiment analysis generic functions; in addition, a “trend analysis” function is being created as a new built-in function.

Thanks to the extension of the architecture presented in 2.2.5, PolicyCLOUD has also the ability to tap on the analytic capabilities of external tools. This was demonstrated with a Social Dynamics tool implemented in conjunction with PolicyCLOUD and we described how this stand-alone tool can be integrated with the PolicyCLOUD platform to enrich the platform with policy modeling, design and simulation capabilities.

Furthermore, PolicyCLOUD is supported by a tailor-made legal and ethical framework derived from privacy/data protection best practices and existing standards at the EU level, which regulates the usage and dissemination of datasets and analytic functions throughout its policy-relevant dataflows.

The architecture has been successfully used with all the families of pilots as described in D6.11 [23].

Our next steps in the last four coming months of the project will be to finetune all the task implementations in view of the continued feedback from the use case scenarios.

References

- [1] A. Audino, "PolicyCLOUD D3.6 – PolicyCLOUD’s Societal and Ethical Requirements & Guidelines," 2021.
- [2] "EUCS – Cloud Services Scheme," [Online]. Available: <https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme>. [Accessed 22 7 2021].
- [3] "AI Fairness 360," 9 March 2020. [Online]. Available: <https://developer.ibm.com/open/projects/ai-fairness-360/>.
- [4] "Ethics guidelines for trustworthy AI," 8 April 2019. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai>.
- [5] A. Audino, "PolicyCLOUD. D3.3 – PolicyCLOUD’s Societal and Ethical Requirements & Guidelines," 2020.
- [6] G. Ledakis, "PolicyCLOUD D3.5 - Cloud Infrastructure Incentives Management and Data Governance Software Prototype 2," 2021.
- [7] "PolicyCLOUD D3.1 - Cloud Infrastructure Incentives Management and Data Governance Design and Open Specification 1," 2020.
- [8] G. Ledakis, "PolicyCLOUD D3.4 - Cloud Infrastructure Incentives Management and Data Governance Design and Open Specification 2," 2021.
- [9] P. Michael, "PolicyCLOUD D6.6 - Integration Plan, Panayiotis Michael, August 2021.," 2021.
- [10] G. Ledakis, "PolicyCLOUD D3.2 - CLOUD INFRASTRUCTURE INCENTIVES MANAGEMENT AND DATA GOVERNANCE SOFTWARE PROTOTYPE 1," 2020.
- [11] "Retina-API Documentation," 2022. [Online]. Available: <https://www.cortical.io/science/>.
- [12] L. G. Boudreau, "Contractive Autoencoding for Hierarchical Temporal Memory and Sparse Distributed Representation Binding" 2018. [Online]. Available: https://urldefense.proofpoint.com/v2/url?u=https-3A_scholarworks.rit.edu_cgi_viewcontent.cgi-3Farticle-3D10897-26context-3Dtheses&d=DwlFaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=AARi9MNWMUipP7UCHjQ6buM84IE9X6BoHlnRsmEbl4&m=-Gz0Ly-zmOu_BfFTmozRZI5f8PgUik-aEcV_SKE6U8AFLNKugxqQMx_wRuQiUYC2&s=XKESpbqh5dWfY280sJ8DpLIPnz-H12CzSFZydpXASa4&e=

- [13] A. e. a. Kiourtis, "Aggregating the syntactic and semantic similarity of healthcare data towards their transformation to HL7 FHIR through ontology matching," *Int. J. Med Inform*, Dec 2019.
- [14] "Seaborn: statistical data visualization,," [Online]. Available: <https://seaborn.pydata.org/>.
- [15] Motta, G., Puccinelli, R., Reggiani, L., and Saccone, M., "Extracting Value from Grey Literature: processes and technologies for aggregating and analyzing the hidden «big data» treasure of organizations," vol. 12, no. 1, 2016.
- [16] "DCAT Application profile for data portals in Europe (DCAT-AP)," [Online]. Available: <https://op.europa.eu/en/web/eu-vocabularies/dcat-ap>.
- [17] Hellmann, S., Lehmann, J., Auer, S., & Brümmer, M., "Integrating NLP using linked data", in International semantic web conference, pp. 98-113, 2013. Springer, Berlin, Heidelberg..
- [18] Wang, X., Sun, Q., & Liang, J., "Json-ld based web api semantic annotation considering distributed knowledge", IEEE access, 8, pp. 197203-197221, 2020.
- [19] 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.," in *bmc bioinformatics*.
- [20] T. Berners-Lee, "Linked Data - Design Issues,," 2006. [Online]. Available: www.w3.org.
- [21] D. Brickley, and R. V. Guha, "Resource Description Framework (RDF) Schema Specification 1.0: W3C," 2000. [Online]. Available: <https://www.w3.org/TR/2000/CR-rdf-schema-20000327/>.
- [22] D. L. McGuinness, and F. Van Harmelen,, "OWL Web Ontology Language Overview. W3C recommendation,," 2004. [Online]. Available: <https://www.w3.org/TR/owl-features/>.
- [23] S. Javier, "PolicyCLOUD D6.11 USE CASE SCENARIOS DEFINITION & DESIGN," June 2022. [Online]. Available: <https://zenodo.org/record/6726412#.YwtmLNNByUk>.
- [24] Berners-Lee, T., Hendler, J., & Lassila, O., "The semantic web", Scientific american, 284(5), pp. 34-43, 2001.
- [25] 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.
- [26] Manjiri Mahadev Mastoli, Urmila R. Pol, Rahud D. Patil, "Machine Learning Classification Algorithms for Predictive Analysis in Healthcare," *IRJET*, vol. 6, no. 12, Dec 2019.
- [27] M. John, Predictive Analysis with SAP.

- [28] Holmes, E. E., M. D. Scheuerell, and E. J. Ward, Applied time series analysis for fisheries and environmental data. Edition 2021. Contacts eeholmes@uw.edu, warde@uw.edu, and scheuerl@uw.edu, 2021.
- [29] "statsmodels 0.14.0," [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html.
- [30] F. M. Zakria Muhammad, "Forecasting the population of Pakistan using ARIMA models," *Pakistan Journal of Agricultural Sciences*, vol. 46, no. 3, pp. 214-223, 2009.
- [31] "Prophet," [Online]. Available: <https://facebook.github.io/prophet/>.
- [32] "Seasonal ARIMA models," [Online]. Available: <https://otexts.com/fpp2/seasonal-arima.html>.
- [33] P. Michael, "PolicyCLOUD D2.2 - CONCEPTUAL MODEL & REFERENCE ARCHITECTURE," 2020.
- [34] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam, "Exploiting BERT for End-to-End Aspect-based Sentiment Analysis," 2019.
- [35] Meike Will, Jürgen Groeneveld, Karin Frank, Birgit Müller, "Combining social network analysis and agent-based modelling to explore dynamics of human interaction: A review, Socio-Environmental Systems Modelling," vol. 2, Feb 2020.
- [36] William N. Dunn, Public Policy Analysis: An Integrated Approach (6th ed.). Routledge, 2017.
- [37] E] Eric Patashnik, Eugene Bardach, A Practical Guide for Policy Analysis (6th ed.). Sage, New York, NY, 2020.
- [38] Alex Schwartz, Rachel Meltzer, Policy Analysis as Problem Solving, Routledge, 2019.
- [39] Sameera Abar, Georgios K.Theodoropoulos, Pierre Lemarinierc, Gregory M.P.O'Hare, "Agent Based Modelling and Simulation tools: A review of the state-of-art software," *Computer Science Review*, vol. 24, 2017.
- [40] [Online]. Available: <https://www.phoenixframework.org/>.
- [41] [Online]. Available: <https://elixir-lang.org/>.
- [42] [Online]. Available: <https://www.erlang.org/>.
- [43] [Online]. Available: <https://github.com/elixir-ecto>.
- [44] M. Mintrom, Contemporary Policy Analysis, Oxford University Press, 2012.

[45] [Online]. Available: <https://ccl.northwestern.edu/netlogo/>.

[46] [Online]. Available: https://repast.github.io/repast_hpc.html.

[47] [Online]. Available: <https://gama-platform.github.io/>.

[48] Bhakti Stephan Onggo, Levent, Yilmaz, Franziska Klügl, Takao Terano, Credible Agent-Based Simulation – An Illusion or Only A Step Away?. In 2019 Winter Simulation Conference (WSC). ACM, 2019.

[49] Alexey Voinov, Firouzeh Taghikhah, Tatiana Filatova. Where Does Theory Have It Right? A Comparison of Theory-Driven and Empirical Agent Based Models. Journal of Artificial Societies and Social Simulation 24, Article 4, 2021.

[50] P. Cairney, The Politics of Policy Analysis, Palgrave Macmillan, 2021.

[51] Y. Moatti, "BigDataStack D4.3 WP4 Scientific Report and Prototype Description," 2020.