TEADAL

D3.1 Gravity and friction-based data governance

Revision: v.1.0

Work package	WP 3
Task	T3.1 T3.2
Due date	30/11/2023
Submission date	30/11/2023
Deliverable lead	POLIMI
Version	1.0
Authors	Pierluigi Plebani, Matteo Falconi, Mattia Salnitri (POLIMI) Alessio Carenini (CEFRIEL)
Reviewers	Ronen Kat (IBM), Victor Casamayor Pujol (TUW)
Abstract	One paragraph
Keywords	Data friction, Data mesh, Federated Data Product

Grant Agreement No.: 101070186
Call: HORIZON-CL4-2021-DATA-01
Topic: HORIZON-CL4-2021-DATA-01-01
Type of action: HORIZON-RIA

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	20/10/2023	1st version for comments	Pierluigi Plebani, Matteo Falconi, Mattia Salnitri
V0.2	10/11/2023	2nd version with updates addressing the comments received by IBM and TUB	Pierluigi Plebani, Matteo Falconi
V0.3	20/11/2023	3rd version with the integration of data catalog contribution	Alessio Carenini
V0.4	21/11/2023	Version for internal review	Pierluigi Plebani
V.1.0	29/11/2023	Final version submitted	Pierluigi Plebani, Matteo Falconi, Alessio Carenini, Mattia Salnitri

Disclaimer

Funded by the European Union

Funded by the European Union (TEADAL, 101070186). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Copyright notice

Project funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	*Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)*	✔
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

EXECUTIVE SUMMARY

This document reports the results of the activities of the first 15 months for WP3. In general, the workpackage aims to define a federated governance model for data lakes that is able, through the concepts of gravity and friction, to evaluate the impact of data movement between different locations within the same data lake, or between data lakes managed by different organizations.

Specifically, this document presents a **data governance model** that allows one to manage data sharing through an approach that combines service design principles and data mesh in a revised and extended version for federated environments.

The proposed governance model is based on a data lake that, unlike those usually proposed in the literature, has an architecture based on zones, in which one called data sharing zone, is responsible to host the so-called **Federated Data Products (FDPs)**. These elements, based on a model that extends the Data Product concept typical of data meshes, enables data sharing and includes governance elements based on policies. As the **definition of policies** is usually a cumbersome activity, it is here proposed an approach that starts from the definition of policies at higher level of abstraction and then, after some transformations, for which the study of automation mechanisms are under development, generates the instructions useful for the enforcement of those policies.

A further contribution to the work is provided by the **data catalog** that allows one to publish and search FDPs. This data catalog includes a metadata model that allows one to store information related to FDPs, how they are created and updated, and how they are shared. It is, therefore, through the data catalog that a potential use of data is possible. In fact, it is assumed that when a FDP is realized, its description is published in the data catalog. At this point, a user may have an interest in this FDP and request access to it. This triggers a process of creating the so-called SFDP (**Shared FDP**), which corresponds to a proxy of the FDP, where, however, only the data agreed by the provider for that particular consumer is visible on the basis of the policies that the two parties define.

In face of a data request, there will be a **pipeline** between the FDP and the SFDP defined on the basis of the agreement. The goal of this pipeline is to transform the dataset exposed by the consumer-independent FDP into a dataset containing only the data visible by the consumer-dependent SFDP. It is important to emphasize that the pipeline is the primary element for handling **gravity and friction**. This is because tasks can be placed at the edge and cloud, or on the provider side and on the consumer side. At present, more emphasis has been given to friction and a computational model of this aspect is proposed.

Funded by the European Union

TABLE OF CONTENTS

Funded by the European Union

LIST OF FIGURES

Funded by the European Union

ABBREVIATIONS

AP	Application Profile
DP	Data Product
DCAT	Data Catalog Vocabulary
DCAT-AP	Data Catalog Vocabulary-Application Profile
ESD	European Data Spaces
EIF	European Interoperability Framework
FDP	Federated Data Product
IDSA	International Data Space Association
RDF	Resource Description Framework
SDMX	Statistical Data and Metadata Exchange
SFDP	Shared Federated Data Product
SPARQL	SPARQL Protocol and RDF Query Language

Funded by the European Union

1 Federated Data Governance

Regardless of the domain and the market in which a company operates, digitalization has increased the amount, and at the same time also the value, of data. As a consequence, data has been recognized as one of the most fundamental assets in a company and a lot of effort has been dedicated to increase the effectiveness and the efficiency of data internally managed.

In fact, it is becoming clearer that the data value can be increased if data are also shared with other companies (Lefebvre 2023). While data sharing is a common practice among organizations to support supply chains, organizations themselves see an opportunity in sharing complementary data assets. For example, for those who produce sensorized products, the data linked to what is collected by these sensors. We then move from sharing transactional data to sharing analytical data (Wixom 2022) but, at the same time, it is important to protect the data so that the value of the asset is not lost. In other words, once the data has been shared, It is important to keep control on the data asset and to retain the how the data is shared ownership of the asset. It is, therefore, important that an organization has full control of the data and, therefore, has assurances that the data is used according to the agreed terms. Especially when the data processed contains personal information, the exchange of the same must take place in compliance with the relevant legislation (e.g., GDPR).

In this context, the European Commission, in line with its data strategy (EC 2020), is promoting the creation of **data spaces** (Giess 2023): federated architectures enabling the data sharing while preserving data sovereignty that, among the several definitions offered in the literature (Hummel 2021), it is considered in this document as the ability for data owners to have a complete control on their data, where control includes, among the others, the decisions on where to store the data, who has the right to access them, according to which purpose. In this context, the need to **share data** between different organizations requires a revisiting of the concepts related to data governance activities. In fact, data governance is usually about making data available and trustworthy to users *within the organization*.

In TEADAL the boundaries of data governance are expanded and consider stakeholders in other organizations among potential users.

As shown in Figure 1, as a basic hypothesis of the project, the organizations involved in data sharing relies on a data lake to manage their data for analytical purposes. Data lake provides the capabilities to ingest, curate, and store heterogeneous data. Moreover, facilities to perform analytics are also provided according to the needs of the company. Each organization has its own policies that govern the life cycle of the managed data. Assuming that this type of organization are members of the same federation, the focus is to define the **federated data governance** model that is valid for all organizations belonging to the federation and specifies how to make data available and trustworthy to users *within a federation*.

Funded by the European Union

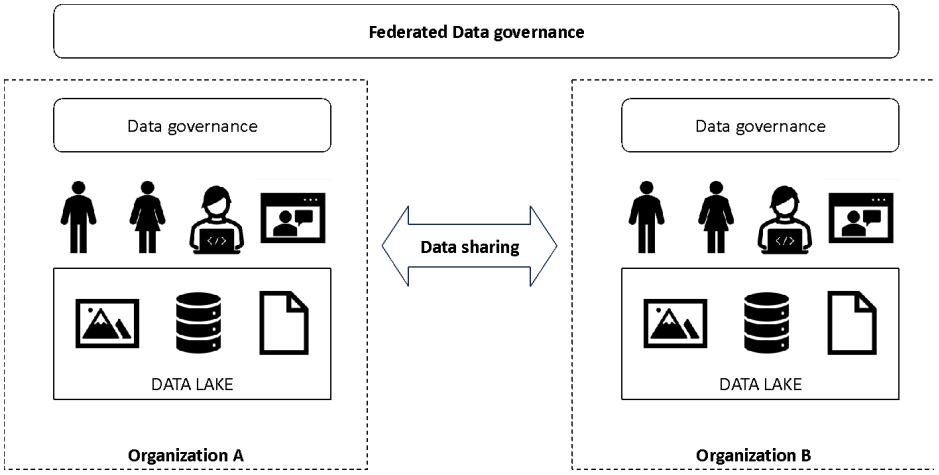

FIGURE 1 - TEADAL GENERAL SCOPE

1.1 DESIGN PRINCIPLES

Basically, the federated data governance model proposed in TEADAL is based on two main pillars: the extension of Data Mesh to a federated setting and the adoption of the service orientation principles. The resulting approach can be conceptualized as in Figure 2. Generally speaking, each organization that wants to share the data has to create the so-called **Federated Data Product (FDP)** (described in details in Section 1.3), a data mesh inspired architectural component in charge of managing the data sharing. The FDPs are designed according to the service oriented principles, thus they will be visible from the outside of the organizations by the other members of the federation. The proposed federated data governance will guide the life cycle of the federated data product: from its definition, to the invocation, till the decommissioning. To this aim, as discussed in Section 1.2, the data lakes are extended towards **TEADAL nodes** which offer additional facilities to address the federated data governance.

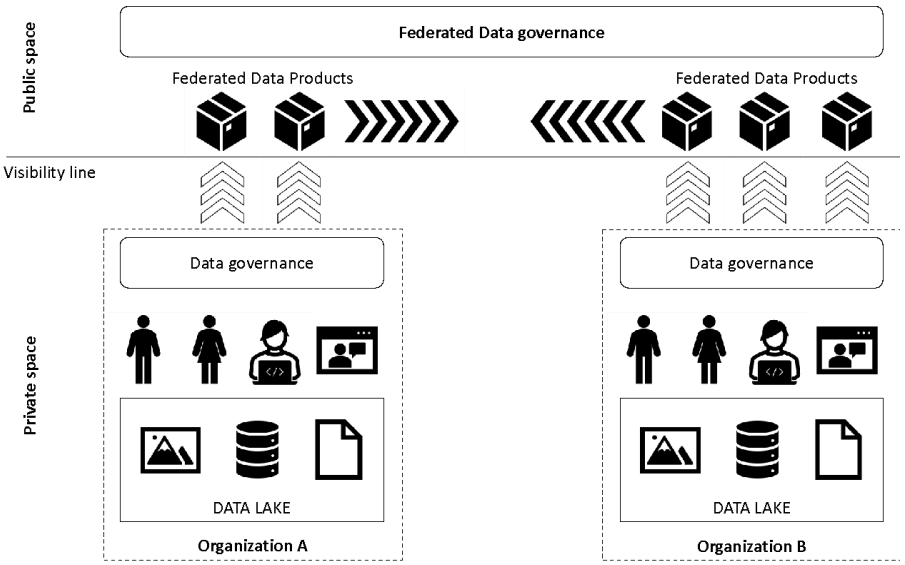

FIGURE 2 - CONCEPTUALIZATION OF THE TEADAL APPROACH TO FEDERATED GOVERNANCE

Funded by the European Union

Data Mesh is an emerging decentralized socio-technical approach to share, access, and manage analytical data in complex and large-scale environments, within or across organizations (Dehghani, 2022). This paradigm is based on the following four principles:

- Domain ownership. Inspired by the driven-domain software design (Evans 2004), the responsibilities of the data are given to the people that are closer to them. Talking about people and not technology implies a distribution of responsibilities that does not depend on the way in which the platform used to manage data is organized, but is aligned with the business.
- Data as a product. Like in the service-oriented solutions, data sets need to be managed having in mind the final consumer, which can be also seen as a customer. For this reason, data needs to be curated, properly described, made visible, and be easily and efficiently accessible.
- Self-service data platform. To avoid situations where the different teams involved in managing their own data products independently develop platforms and applications for this purpose, a common platform offering a set of capabilities to support the data life cycle is offered.
- Federated computational governance. Having different teams that independently manage their data products could lead to a chaotic scenario where every team decides to govern their products according to policies which could clash with each other. This calls for a common data governance based on policies that enactment needs to be automated as much as possible.

Data mesh approach is more and more adopted by companies to improve data management, with a special focus on data governance to ensure the data ownership. This implies that the data mesh approach is directed to ensure that the data is properly managed inside an organization. This is done by providing a platform that is able to support the policy definition and the policy compliance check, considering the **data product** as the basic architectural element. More in detail, the data mesh approach defines the data product as the architectural quantum, i.e., the most compact architectural entity capable of standalone deployment. It has significant functional unity and encompasses all necessary structural aspects for its operation.

Moving to the second pillar of the approach, i.e., the adoption of service orientation principles, these are applied to a data product to define the basic element that can be shared among the organizations, i.e., the **FDP**. In particular, the service-orientation principles defined by (Erl 2007) are:

- Standardized Service Contract: functional and non-functional aspects must be defined with a format agreed among the actors that are involved in the service life cycle.
- Service Loose Coupling: life cycle of a service should be as much as possible independent from its consumer.
- Service Abstraction: all the information about how the service is implemented must be hidden, thus the service contract should contain only the minimal set of information to allow the consumer to invoke the services.
- Service Reusability: the logic encapsulated by the service is associated with a context that is sufficiently agnostic to any one usage scenario to be considered reusable.
- Service Autonomy: services are deployed in an environment over which they exercise a (preferably an exclusive) control.
- Service Stateleness: services should not retail any state to increase autonomy and keep their loose coupling.
- Service Discoverability: Service contracts are equipped with appropriated metadata that will be correctly referenced when discovery queries are issued
- Service Composability: a service can be designed as a composition of other services.

Funded by the European Union

It is assumed in the TEADAL project that organizations in the federation that wish to share their data do so in the form of FDP. They, therefore, offer a data access service that remains the property of the organization offering the FDP which will control data access and, in general, the life cycle of the FDP. As described below, each federated data product will be appropriately described. The format of the document containing such a description is defined by the federation. Furthermore, the creation and management of the FDP will be the responsibility of the FDP provider itself. It is also important to underline that the FDP is built starting from data resources internal to the organization and for which no information is made explicit to the final consumer. In this way, a federated data product holds the role of line of visibility that distinguishes what is private to the organization and what is public to other members of the federation.

1.2 TEADAL Node Model

The governance of federated data, therefore, concerns the control of the life cycle of a federated data product so that it is created according to the rules defined at the federation level, is made public to the other members of the federation, is accessible only after the definition of access rules and data management between supplier and consumer. In this regard, the TEADAL project hypothesizes that different organizations of the federation will manage their data according to a data lake platform, which – thanks to the tools offered by TEADAL – will be able to manage the life cycle of the federated data product: from creation to dismission. Specifically, the expected TEADAL node model is based on a structure which is inspired by data lake architectures which are organized in **zones**.

A data lake zone represents an area of the data lake that gathers computational and storage resources for a specific objective. The quantity of resources and their characteristics are commensurate with the objectives of the area. The division between zones is therefore functional and not infrastructural.

The data lake is fed by data sources considered external to the data lake itself. Data sources can be related to structured, semi-structured, or unstructured data, accessed in batch or streaming mode.

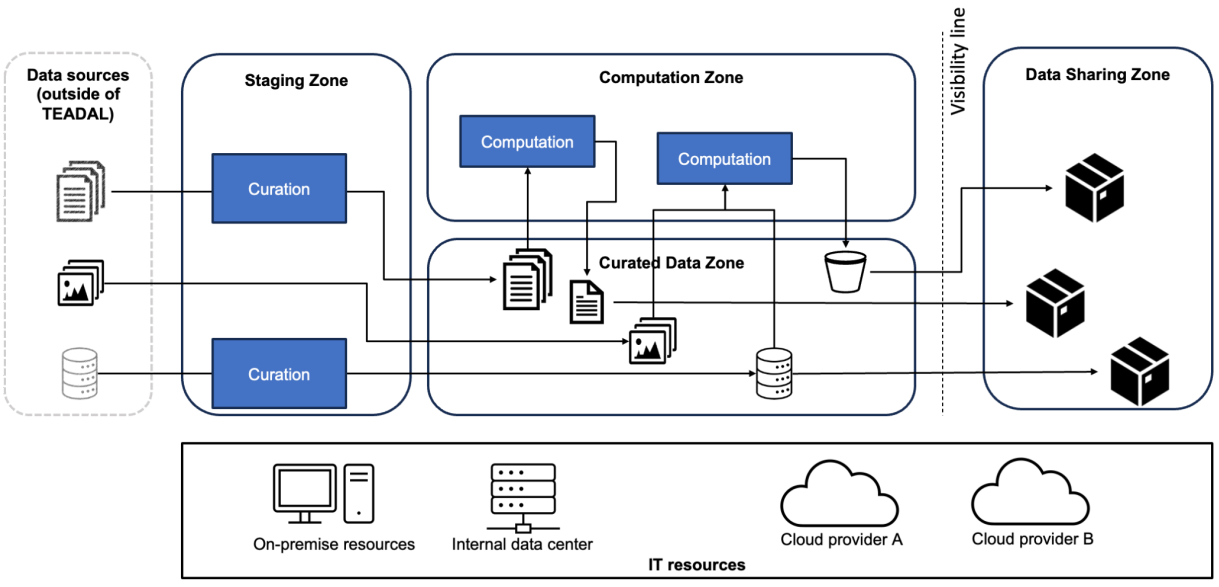

Figure 3 - TEADAL node mode

Funded by the European Union

In particular, considering the data sources external to the considered data lake, the TEADAL node is organized in 4 zones (see Figure 3)[1]:

- Staging Zone: this zone has the objective of extracting the data from the data sources. Thus, here there are connectors to the data sources and the strategies defining how and when the data are read from the data sources are specified. Before loading the data into the data lake, some curation could be operated on the read data to increase their quality (e.g., data cleansing, data harmonization). The result of these transformations are stored in the curated data zone. It is worth noticing that, based on the typical approach of data lakes, which is in this regard different to the data warehouses, data are subjected to minimal interventions and the format of the original data sources are maintained.
- Curated Data Zone: this zone is in charge of storing the curated data, i.e., the data load and cleaned from the data sources. Data stored in this zone can be seen as the data assets that the organization considers relevant for secondary usage. The curated data zone is the place in which the data lake stores the data ready to be used by the other zones. Accessibility is ensured from a management and technical perspective. About the former, all the data sets stored in this zone are properly classified, described, and their description published on a **data catalog**. About the latter, how to access and credentials are specified and the related access control is implemented to make sure that the entries in the data catalog concerning the dataset in the curated data zone are only visible internally.
- Computation Zone: this is the zone in which the data analytics occur. Depending on the needs, in this zone, data from the curated data zone are read and processed. It might happen that the output of a computation constitutes a data asset worth to be placed in the curated data zone. The computation zone hosts the environments for the data analytics as requested by the developers in charge of implementing the related applications. Likewise, the zone relies on a set of resources that can properly sustain the needs of the computational environments.
- **Data Sharing Zone**: this zone represents the distinctive element of the TEADAL proposal as it hosts the resources needed to share the data that the organization wants to make available to other organizations. Based on the TEADAL approach, the sharing data zone has the objective of managing the life cycle of the FDPs that have been defined by the organization.

Returning to the line of visibility defined previously, having the goal of hosting and managing the FDP, the data sharing area represents the only area visible to external consumers. This consequently requires that the data catalog used to manage the datasets for the internal management of the data lake can also be extended to classify the FDPs and, therefore, offer other organizations visibility with respect to these, and only these, FDPs.

A further aspect that characterizes TEADAL's proposal concerns the way in which the data lake can be implemented. Classic data lake solutions provide a substantially centralized solution, in the sense that all zones are hosted on a single infrastructure provider, usually in the cloud. Therefore, the storage and processing resources needed for each zone are often in the same locations. TEADAL's idea of providing a federated extended data lake takes the form of a stretched data lake where the underlying resources managed within zones can be distributed across different locations, both on the cloud and at the edge. Likewise, these resources can be managed by a single organization or multiple organizations, as well as provided by the same vendor or different vendors. For example, as shown in Figure 3, the

[1] For additional details about the internal structure of a TEADAL node and the technology chosen to implement see Deliverable D2.2

Funded by the European Union

underlying IT infrastructure, needed to host the different zones, is composed of a heterogeneity of resources. Some resources can be offered by different cloud providers, which could operate in different locations. Moreover, the same organization could have internal resources dedicated to host what it is needed by the data lake.

This heterogeneity in the management and localization of resources is the main source of criticality that leads to the definition of gravity and friction, as discussed in Section 4. Specifically, friction comes into play when data passes between different organizations via FDP or, even when considering a single organization, resources are managed by different actors (for example, two different cloud providers). In contrast, gravity comes into play when data passes, and possibly transformed, between resources that are located in different locations while being managed by the same operator and, in particular, when some resources are located at the edge and others in the cloud.

1.3 Federated Data Product Model

A FDP represents the architectural element used within a TEADAL node to represent the data served through that node. A FDP is essentially a service which provides access to some data. To this aim, as shown in Figure 4, a FDP exposes an API which specifies how the service consumer, i.e., the data consumer, can access to the data. In TEADAL, we assume that the REST architectural style is adopted to design the interface and, therefore, OpenAPI[2] is adopted as a description language for this interface. Internally, a FDP has the capability to run code, to store data, and to enforce policies. Depending on the specific FDP, those capabilities could be or could be not exploited.

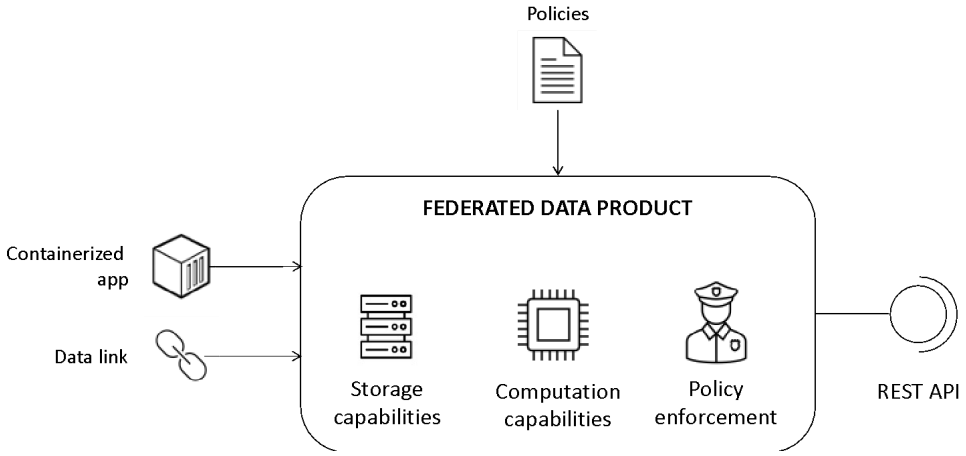

Figure 4 – Federated Data Product Model

A fundamental and mandatory input of a FDP is the link to one or more data sets as they will then be exposed via the API. Such data, that are present in the storage zone of the TEADAL node, can be made available for sharing purposes as they are acquired by the FDP, or they can be subjected to processing (e.g. integrations, transformations) to be compatible with the specified interface. To this end, depending on the complexity of the data processing, it may be necessary to define - through a presumably containerized application to guarantee the flexibility of the system - the code capable of carrying out such processing. Likewise,

[2] https://www.openapis.org

Funded by the European Union

depending on the amount of data that can be processed, the FDP will be able to rely on internal storage to completely or partially store the data that will then be exposed.

Finally, a FDP is also defined based on the policies that must be respected when sharing data. The policy document, in fact, reflects the data governance choices relating to the data that the FDP exposes. These policies can define, among other things, access control rules, transformation requirements before transmission (e.g. encryption, anonymization), data retention rules once shared. Compared to the other two inputs of a FDP, i.e. links to data sources and containerized apps, the policy document is produced by the data provider but must also be visible to the data consumer to make them aware of it. In TEADAL, we adopted Rego[3] as the language to specify the policy rules.

It is important to underline how a FDP is designed by the data provider to define the possibility of access to a specific dataset that the data provider intends to share. Following a service-oriented design logic, the definition of the FDP occurs without necessarily knowing the exact consumer of the data, but only based on a hypothetical profile of a typical consumer.

Once the FDP has been designed, it is made visible through the data catalog which will contain the public description of the FDP composed of the interface description (with OpenAPI) and the policy document (with Rego). Assuming that a data consumer has selected a FDP, the TEADAL approach provides that the FDP is not directly accessible by the data consumer, but the access is mediated by an element called **Shared Federated Data Product (SFDP)**.

A SFDP (see Figure 5) is essentially a service connected to a specific FDP, which has the objective of allowing access to the data offered by the FDP to a given data consumer according to specific access rules defined following an agreement between the data provider and the data consumer

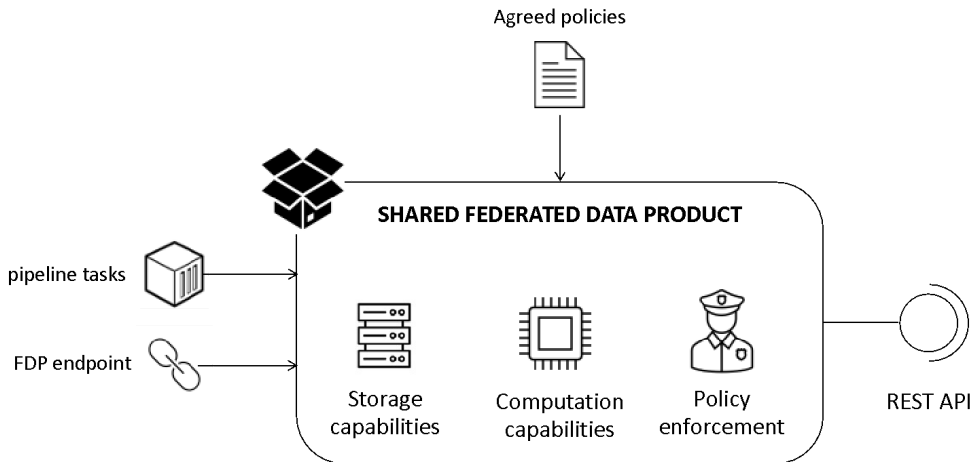

FIGURE 5 - SHARED FEDERATED PRODUCT MODEL

This agreement mainly acts on the definition of the SFDP API, therefore, on the methods and data that the data consumer can actually access. For example, assuming that the data provider has defined a FDP capable of exposing patient data from its hospital defined by

[3] https://www.openpolicyagent.org/docs/latest/policy-language/

Funded by the European Union

name, surname, age, sex, pathology, following the request for such data by another hospital, the agreement could stipulate that only age, sex can be made accessible. Furthermore, the data of not all patients can be accessed but only those with certain pathologies.

Therefore, a SFDP is defined by an API consisting of the data access methods available to the data consumer. Since the SFDP is the FDP's intermediary for a specific data consumer, the relationship between the API exposed by the FDP and that exposed by the SFDP is linked to an operator that we could define as containment. In fact, it is assumed that the SFDP API has a set of methods that are the same as or contained in the method set of the FDP API. Additionally, for each method call, the cardinality and degree of the data returned by the SFDP will always be equal to or less than the cardinality and degree of the data returned by the FDP. In between the FDP and the SFDP some data transformation could be required and they are organized in pipelines as described below.

The agreement between consumer and data provider also affects policies. Although the policies applied to a FDP are generic to all consumers, there may be specific policies that apply to specific consumers. For example, a SFDP might mandate a specific encryption algorithm to be applied in data transmission. Or, it could define data visibility rules depending on the type of end user - consumer side - who will access the data. For example, while a doctor will have access to both gender, age and pathology attributes, an administrator will only have access to age and gender[4].

Finally, from a structural point of view, a SFDP is very similar to a FDP. A SFDP could have storage capacity and computing capacity. Finally, a SFDP has the ability to monitor the application of policies.

As mentioned, the link between FDP and SFDP is very strong. This is because a SFDP does not directly access data but is connected to a reference FDP. Importantly, given a FDP, there can be multiple connected SFDPs: one for each of the data consumers interested in the FDP[5]. In terms of degree and cardinality data exposed by a SFDP can be a subset of those of the FDP. Moreover, data governance policies regulating the access to the SFDP should be stricter than the one defined for the FDP. Based on this, a **pipeline** can be used as the model to represent the set of operations applied to the data from when they are produced by the FDP to the one that is consumed by the SFDP (See Figure 6). This pipeline is specific to each FDP/SFDP pair and should be created by a data steward on the data provider side.

[4] As mentioned in deliverable D2.2, federated identity management based on KeyCloak is considered to ensure this type of control.

[5] At the moment, however, the possibility that the same SFDP is connected to different FDPs is not envisaged as the implications of this type of architectural choice are being studied.

Funded by the European Union

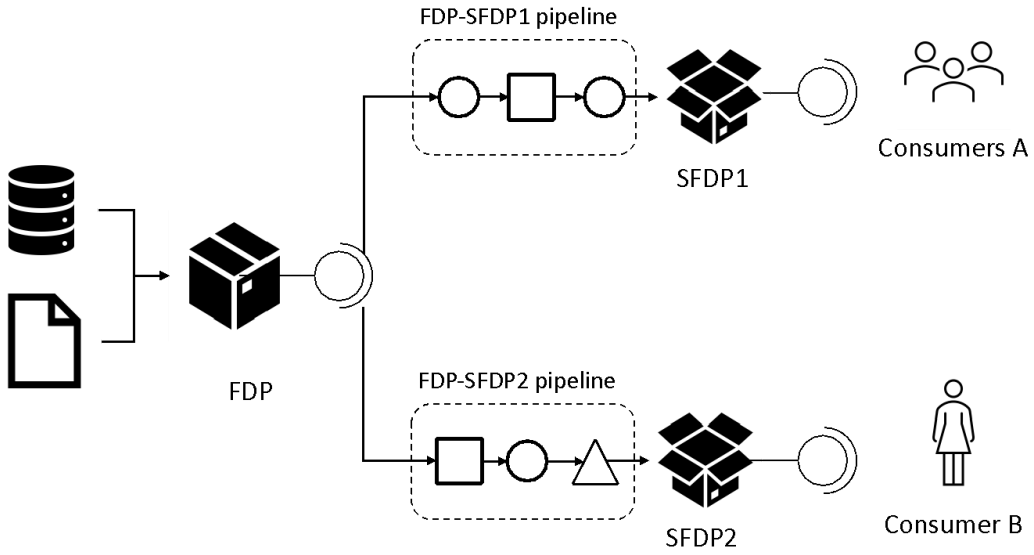

FIGURE 6 - RELATION BETWEEN FDP AND SFDP

With respect to the architecture of the TEADAL node, as reported in Figure 3 and already discussed, the FDPs are part of the data sharing zone. Similarly, the SFDPs and the related pipelines connecting them are part of the same zone.

Funded by the European Union

2 Gravity and Friction-aware Pipelines

Data gravity and friction have been considered in TEADAL as a way, inspired by physics, to model the forces that regulate, respectively, the way in which data can be distributed along the continuum, i.e., vertically from the edge to the cloud, and the possibility to share data with other organizations, i.e., horizontally, where involved resources are not only owned/managed by the organization that has the right to use the data.

These forces are strictly connected to the data governance approach described above as the gravity and friction are directly related to the effort that is required to take data from the sources, change to transform them to be compliant with a FDP, support the transformation along the pipeline, and then offer to the final consumer. In particular, gravity and friction affect the ability of data to move from one compute or storage element to another along the data source→fdp→pipeline→sfdp chain. In fact, for reasons related to the availability of computational and/or storage resources, the node on which the FDP is hosted may not allow the processing of the requested data in a single block, but only through a series of batches. Similarly, the sending of data to locations present on a specific cloud could be prevented due to constraints linked to regulations which require, for example, the storage of data on nodes not belonging to the European Community.

In a first attempt to identify how friction and gravity can be related to the location involved in the data management, as shown in the Figure 7, there are four possible logical locations to have storage and/or compute resources: provider-controlled cloud resources, provider-controlled edge (on-premise) resources, consumer-controller cloud resources, consumer-controller edge (on-premise) resources. Therefore, the datasource→fdp→pipeline→sfdp chain must be implemented appropriately to satisfy all constraints defined by federated data governance, as well as to minimize the influence of the network in the data transmission phase.

According to this setting, **gravity regulates the decision to deploy the resources either on the edge or the cloud,** while **friction regulates the decision to deploy the resources either on the provider or consumer side** (see Figure 7).

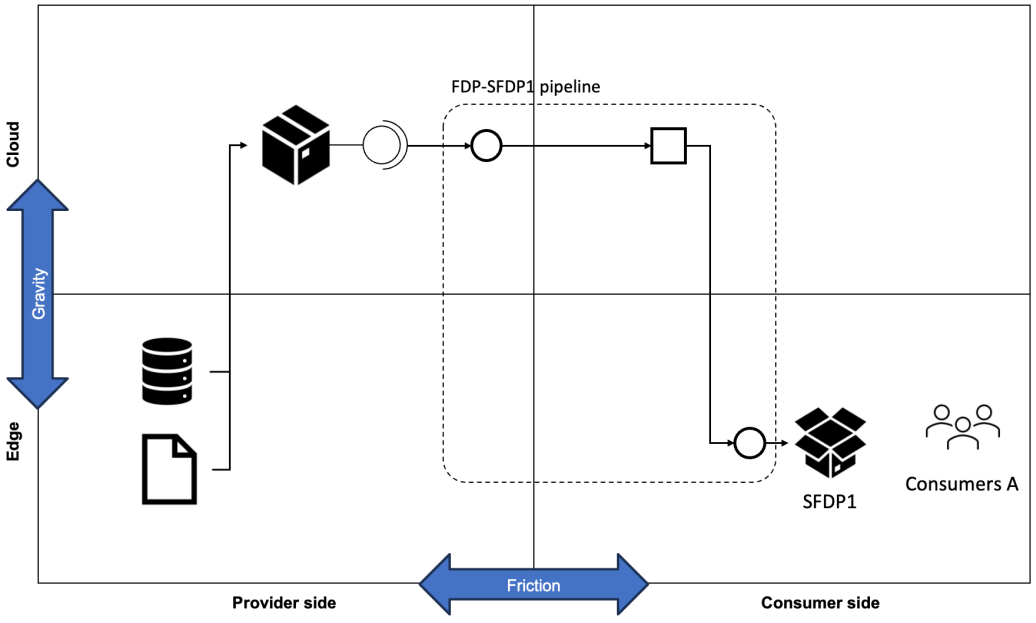

Figure 7 - Deployment of the pipeline and relation with Gravity and Friction

Funded by the European Union

According to this approach, the goal of gravity and friction is to evaluate the effects on the adoption of different pipeline deployment configurations among the admissible ones. In fact, considering the three tasks of the FDP-SFDP1 pipeline in Figure 7, their deployment in one of the four quadrants depends on the resources available to run the task and the possibility to move the data from one quadrant to the other. Only to give an idea, if we consider the case in which the consumer can see only anonymized data and the third task is in charge of this transformation, the entire pipeline must reside on the provider side. Conversely, if the second task performs the anonymization, then the deployment configurations are two: with the third task on the provider side or on the consumer side.

It is worth noticing that, at this stage, the focus has been mainly dedicated to the exploration of the friction. Exploration and definition of gravity will be part of the next iteration.

Funded by
the European Union

3 Enriching data catalog

Datalakes, often likened to a vast reservoir of raw and unstructured data, gain value and efficiency when coupled with a well-organized and structured catalog. The catalog serves as the backbone, providing a systematic and detailed description of every data asset within the datalake, enabling efficient governance and seamless integration with external services.

Structured descriptions of each item, ranging from datasets to FDP in the context of the TEADAL project, not only facilitate easy discovery but also could provide support to increase the automation in generating the FDP and the SFDP. With a catalog in place, organizations can implement robust data governance policies, ensuring compliance, security, and data quality standards are met consistently. The Catalog also plays a vital role when organizations decide to cooperate in a federation, as it becomes the way users of federated organizations can understand which FDPs are offered for consumption in the context of the collaboration.

One of the main features of a catalog is its ability to streamline and automate governance processes. By providing a clear and standardized view of datasets and Federated Data Products, the catalog enables organizations to enforce policies, track lineage, and monitor access seamlessly. Furthermore, a key aspect in TEADAL is to enrich the data catalog with facilities to create a bridge to the outside world, acting as an enabler for integration with external services. With structured descriptions in place, the datalake becomes not just an isolated reservoir of data, but a dynamic hub capable of interacting with third-party applications, analytics tools, and other external services.

In this section, we will describe the outcomes of the first iteration regarding the descriptions of the assets contained in the TEADAL Catalog and its user interface. The next iterations will enrich the metadata model with the aspects specific to gravity and friction as enablers to proper data sovereignty and energy consumption optimisations.

3.1 Metadata model

The first release of the TEADAL metadata model is mainly targeted at enabling the representation of Datasets and Federated Data Products inside the TEADAL Catalog. We, therefore, focused on the main metadata aspects, namely providing a meaningful description of a digital artifact, representing its status changes over time, and keeping track of how a specific Dataset or Federated Data Product has been generated.

The first aspect we tackled is properly describing the main features of a digital artifact. Data Catalog Vocabulary (DCAT) is an Resource Description Framework (RDF) vocabulary designed to facilitate interoperability between data catalogs published on the Web. The basic idea of DCAT is to model a Catalog, which is composed of several datasets (according to the DCAT definition), each one having different embodiments, as depicted in Figure 8.

DCAT can be extended by means of so-called Application Profiles (AP), which are specifications that re-use terms from one or more base standards, adding more specificity by identifying mandatory, recommended and optional elements to be used for a specific application, as well as recommendations for controlled vocabularies to be used.

The DCAT Application Profile for data portals in Europe (DCAT-AP) is a specification based on the DCAT developed by W3C[6].

[6] http://data.europa.eu/88u/dataset/dfa6ef4a-6d10-44ad-b51a-42bd8dda4476

Funded by the European Union

This application profile is a specification for metadata records to meet the specific application needs of data portals in Europe while providing semantic interoperability with other applications on the basis of reuse of established controlled vocabularies (e.g., EuroVoc) and mappings to existing metadata vocabularies (e.g., Dublin Core, SDMX, INSPIRE metadata).

Since it is focused on data portals, the DCAT-AP use case is the publication of generic datasets without further interest in a fine-grained definition of the purpose or type of the specific dataset.

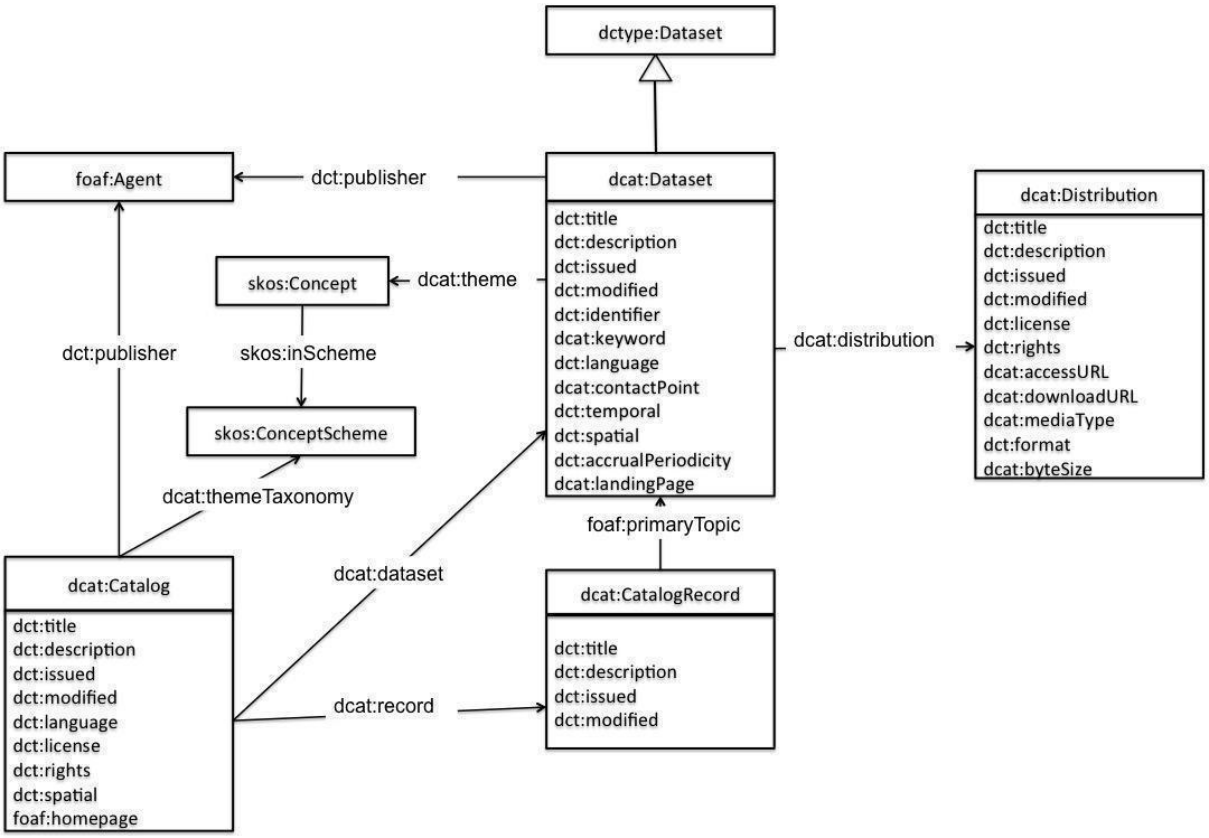

FIGURE 8 - DCAT MAIN CLASSES

In the context of TEADAL project, we decided to create two asset types, whose metadata structure is directly inherited from DCAT-AP and thus compatible also with the International Data Space Association (IDSA) specifications:

- Dataset
- Federated data product

DCAT-AP already provides two concepts which directly map onto such asset types, namely *Dataset* and *Data Service*. In particular, the Data Service concept describes a service which provides access to a specific dataset and can be therefore used to represent a Federated Data Product.

3.2 TRACKING ASSET STATUS CHANGES

The TEADAL Catalog is a digital product which enables collaboration for people belonging to the same organization, and which paves the way for collaboration for people belonging to

Funded by the European Union

different federated organisations. As such, it contains both artifact descriptions which can be considered “ready for publication” as well as descriptions which are in a “draft” status. It is therefore critical to keep track of the status changes, as the value of the “status” attribute defines whether the description can be seen only by its author or whether it is ready to be consumed by a wider audience. We decided to use the PSO Ontology[7] for such a purpose (an example of usage of this ontology is depicted in Figure 9) as it allows stating that a resource can change its status over time as an effect of an action performed by a user.

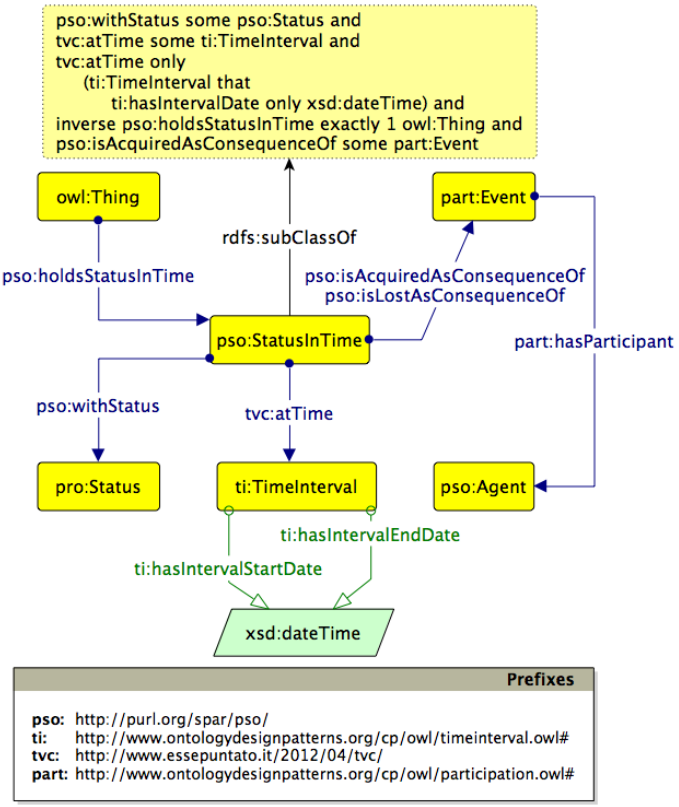

FIGURE 9 - PSO MAIN CLASSES

3.3 TRACKING PROVENANCE

A Federated Data Product can be obtained as a result of processing many Datasets, and keeping track of such processing actions and all their by-products is the basis for ensuring proper data protection and data sovereignty. PROV-O Ontology[8] has been created specifically for such purpose, as it allows establishing links between different artifacts, stating that a specific artifact has been created as a result of a specific activity performed by an agent (which can be a user or a process), as shown in Figure 10.

[7] http://www.sparontologies.net/ontologies/pso

[8] https://www.w3.org/TR/prov-o/

Funded by the European Union

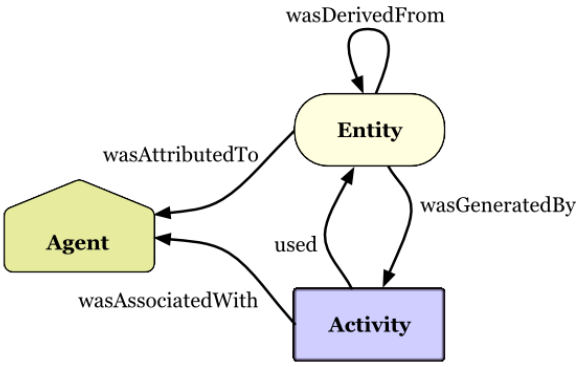

FIGURE 10 - PROV-O MAIN CLASSES

3.4 CATALOG UI

The user interface of the first release of the TEADAL Catalog allows browsing and managing descriptions of digital assets (Datasets, Federated data products and Clinical study proposals). Once logged, users can view the available asset types and the latest items which have been published onto the catalog, as depicted in Figure 11.

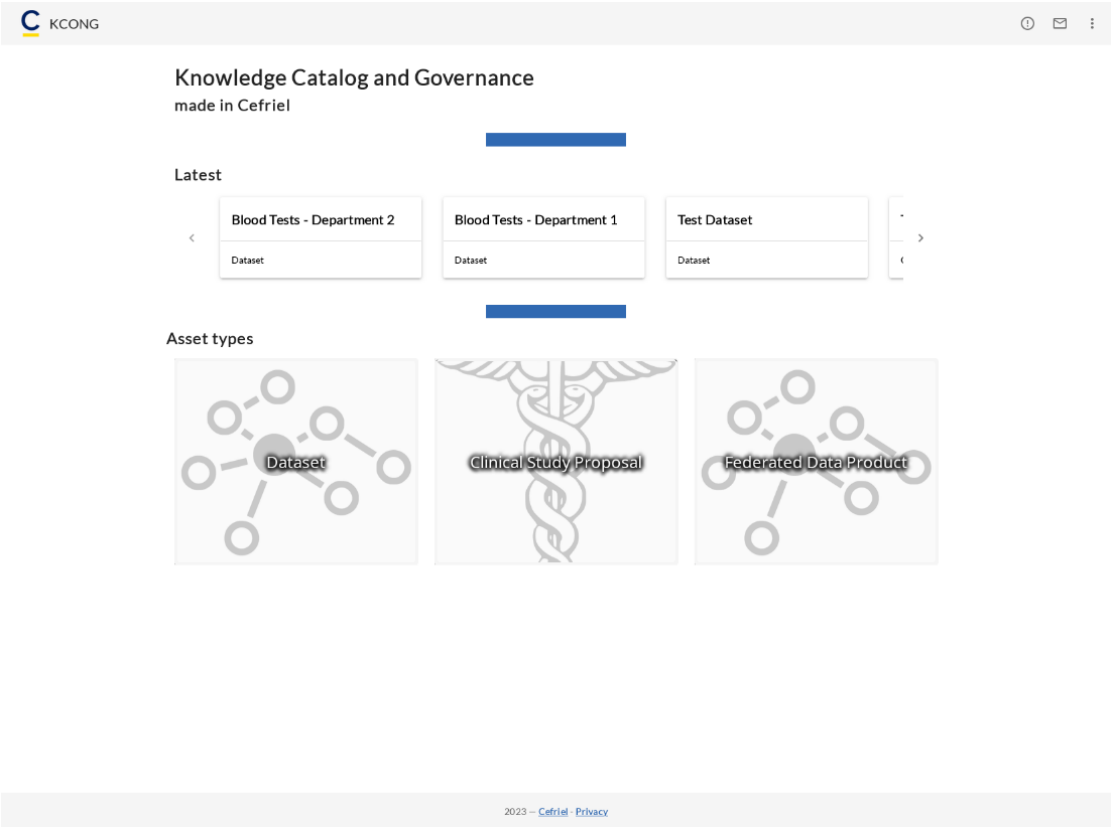

FIGURE 11 - TEADAL CATALOG HOME PAGE

Users can then explore the items belonging to a specific asset type and can perform full-text search (based on the title and the description of an asset) and faceted search (based on a predefined set of filters), as shown in Figure 12.

Funded by the European Union

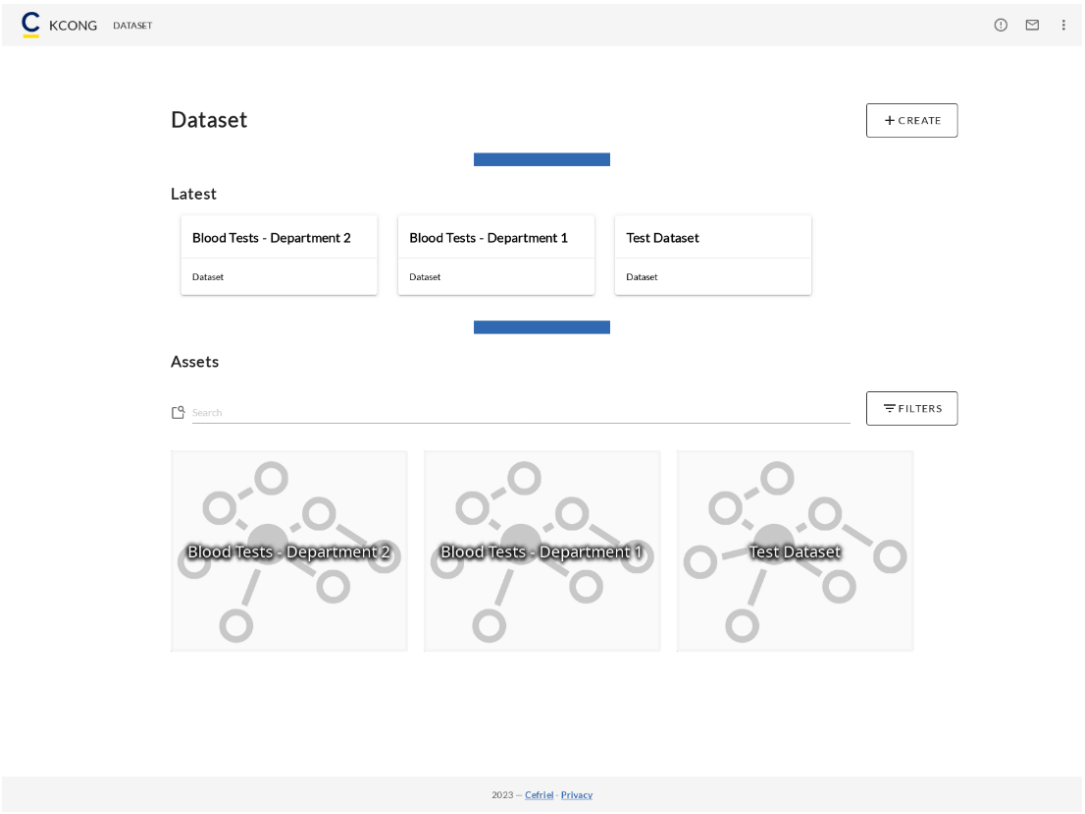

FIGURE 12 - EXPLORING THE ITEMS BELONGING TO AN ASSET TYPE

Viewing and inserting data in the TEADAL Catalog is completely form-based, as depicted in Figure 13.

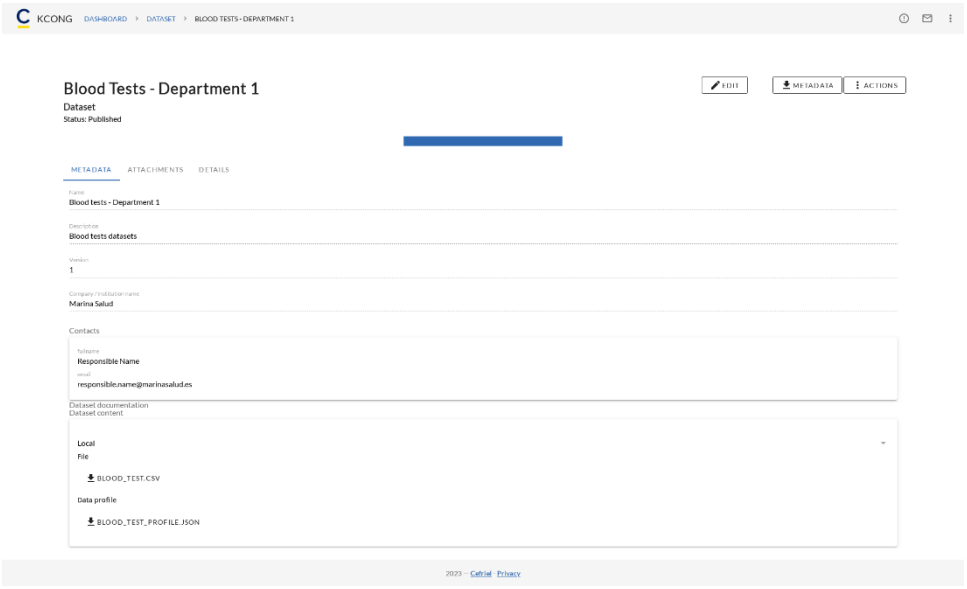

FIGURE 13 - DESCRIBING A NEW DATASET

Funded by the European Union

The Catalog collects metadata in JSON format, and the form which is visible to the user is generated automatically according to the JSON Schema of the metadata model. JSON metadata is then converted in RDF (using DCAT-AP, PSO and PROV-O ontologies described before) according to a template, and the resulting RDF graph is then inserted in an RDF repository, as shown in Figure 14. Such repository contains all the RDF metadata of the asset which have been approved for publication, and can be queried via SPARQL query language.

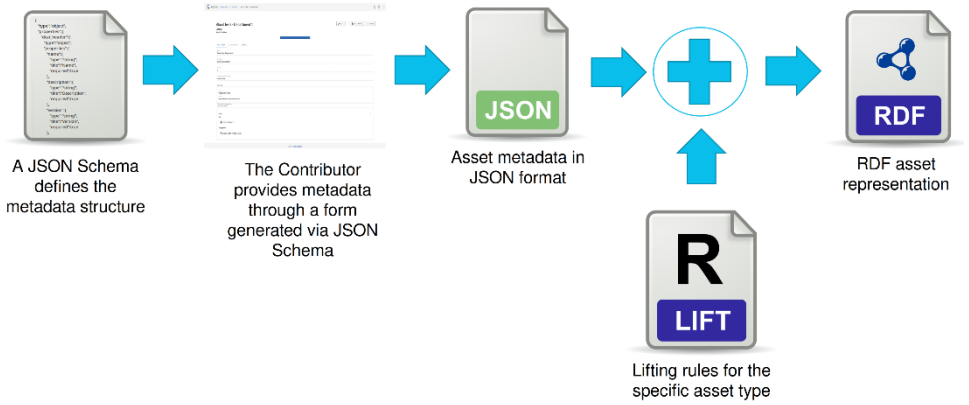

Figure 14 - Generation of RDF metadata from the UI form

3.5 Lifecycle management

Life cycle management in the scope of the TEADAL Catalog can be applied to track the evolution of a single asset, or to enable a particular workflow process. The former refers to the definition of the different states in which an asset can be (e.g., creation, approval, revision, removal), where the transition between each pair of states should be carefully tracked by the data provider. The latter includes the situation where the coordination of multiple services is defined as a workflow. For instance, the deployment of a particular service should be triggered upon completion of another service/process.

We decided to use BPMN for such purpose, as it allows modeling processes encompassing both tasks requiring human intervention and calls to external services. As such, it allows implementing workflows for authorising the publication of the description of an asset, and it is also suited for more complex processes requiring the orchestration of several services, which need to react to the event of a new item being published in the catalog.

The user interface of the TEADAL Catalog provides a Personal Dashboard (shown in Figure 15) where each user can check the publication status of his assets descriptions, and a Task page where he can take decisions in case his intervention is required by a workflow process.

Funded by the European Union

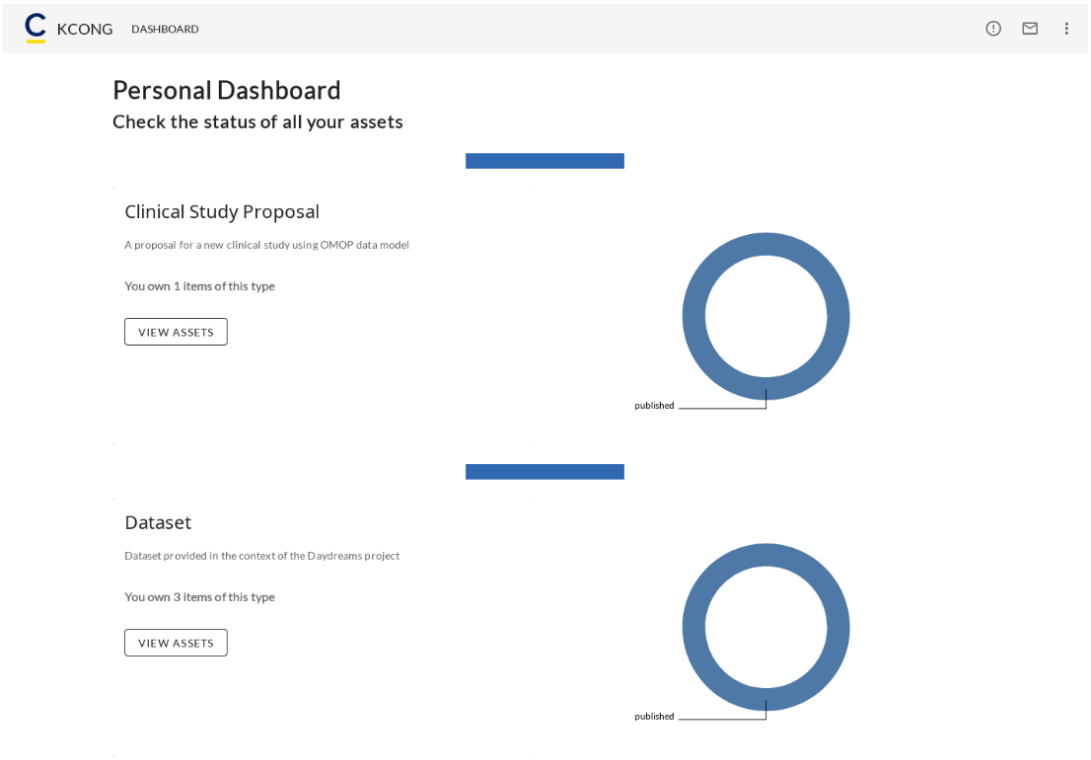

FIGURE 15 - PERSONAL DASHBOARD IN THE TEADAL CATALOG

Funded by the European Union

4 Friction Model

First introduced by (Edwards 2010), the term Data Friction refers to "the costs in time, energy and attention required simply to collect, check, store, move, receive, and access data". Similarly to what happens in physics, Data Friction arises at the interface of two "surfaces", i.e. two points where data is moving (e.g. a IoT sensor to a computer). Data Friction restricts and impedes the natural movement of data and requires costs and effort to overcome it. Poorly managed, low-quality data leads to the inability to use and re-use it effectively, ultimately resulting in friction when it comes to data sharing.

From a technological standpoint, Data Friction is mainly caused by the lack of a proper data sharing infrastructure and proper data management practices (Bates 2019). (Murray-Rust 2008) observed how, while the technological progress made data collection easier, data sharing and re-usability in the scientific community are hampered by the lack of licenses and standards. Even within the same discipline, unified models and platforms supporting data sharing are often lacking (Leonelli 2013). (Edwards et al. 2011) observe also how poor metadata practices restrict the movement of data. In a survey about data sharing practices in the scientific community, (Tenopir et al. 2015) observe how the majority of the respondents do not use metadata standards to describe their data and almost half of the respondents do not use metadata at all. Furthermore, efforts in cost and time to properly curate the collected datasets are often prohibitive for researchers and not covered by the research-funding institutions (Tenopir et al. 2015, Alter 2015).

Based on this literature and on the federated governance model introduced in Chapter 1, we define data friction as the effort to move data from the FDP to the SFDP, i.e., to move the data from the provider side to the consumer side along the TEADAL pipeline that connects the two.

Before discussing in detail how the friction is evaluated, it is required to define some basic elements with a formal notation.

A data pipeline is modeled as a sequence of actions implemented through capabilities that allows operations (e.g., transformation) to be applied to the input data object. The pipeline connects the FDP to the SFDP. The FDP is fed with the datasets managed by the data provider, while the SFDP is offered by the provider and used by the consumer.

In symbols, a pipeline p is defined as an ordered set of capabilities c_i, i.e.:

$$p = \{c_1, \ldots, c_n\}$$

where, given two capabilities c_i and c_j and $i, j \in [1, n] : i < j$, then the capability c_i has to be executed prior to the capability c_j. A data pipeline can be also represented as a function obtained through the composition function (represented with the operator o, e.g. $g(f(x)) = (g\ o\ f)(x)$) of the functions of its capabilities $c_n\ o \ldots o\ c_1$. Given a data pipeline p and a FDP d, the result of the execution of the pipeline on the data object can be expressed as $p(d) = c_n\ o\ \ldots\ o\ c_1(d)$.

Note that, in our definition, the capabilities of a data pipeline p can be executed only in a sequential way, one after another according to the order relationship of p. This pipeline model does not cover those cases where two capabilities can be executed simultaneously: in such

Funded by the European Union

cases, we may assume that the capabilities executed simultaneously can also be executed in series, in an arbitrary order. In future work, additional structures can be discussed.

Given two data pipelines p_1 and p_2, $p_1 \cap p_2$ corresponds to the capabilities that are shared by both the pipelines. This means that, in a system, the capabilities used to implement and execute a data pipeline can be shared with other pipelines.

For instance, let's consider a data object d_0 representing the anagraphic information of a hospital's patients. Through various pipelines, it is possible to derive a variety of data objects from d_0. Among these data objects, let's consider $d_1 = p_1(d_0)$, containing the number of patients per ZIP code area in a JSON array, and $d_2 = p_2(d_0)$, containing the same information as d_1 but in CSV format. We can assume that p_1 and p_2, up to a certain point, will share the same capabilities to transform d_0 in the respective outputs.

Given a data pipeline $p^x = \{c_1, ..., c_x, ..., c_n\}$, the set $\overleftarrow{p}^x = \{c_1, ..., c_x\}$ represents the portion of the pipeline that will be deployed at the provider side. Conversely, the set $\overrightarrow{p}^{x+1} = \{c_{x+1}, \ldots, c_n\}$ represents all the capabilities that will be deployed at the consumer side.

Since, given a pipeline p, every capability $c \in p$ must be implemented either by the provider or by the consumer, and given two capabilities $c_i, c_j \in p$, $i < j$, if c_j is implemented by the provider then also c_i must be implemented by the provider, then:

$$p^x = \overleftarrow{p}^x + \overrightarrow{p}^{x+1}$$

This notation is useful to capture the fact that in a data pipeline connecting a provider and a remote consumer, the transmission of the data object between them will occur at a certain step x. That is, given a pipeline $p_x = \{c_1, ..., c_x, c_{x+1}, ..., c_n\}$ executed up to c_x at the provider side and from c_{x+1} to c_n at the consumer side. Between c_x and c_{x+1} the data object will be transmitted.

$$\overleftarrow{p}^x = \{c_1, \ldots, c_x\}$$
$$\overrightarrow{p}^{x+1} = \{c_{x+1}, \ldots, c_n\}$$
$$\widehat{p}^x = \overleftarrow{p}^x, c_t, \overrightarrow{p}^{x+1}$$

The transmission of the data object can be represented as an additional capability in the pipeline, properly placed in the pipeline. This additional capability is called *transmission capability* and represented with c_t and the pipeline enriched with this capability is indicated as $\widehat{p}^x$.

Depending on the step of the pipeline in which the transmission occurs, the transmitted data object will have certain characteristics. The characteristics of the data object transmitted over the communication channel will affect its transmission, making it more or less convenient in terms of time, cost, or resource usage. For instance, a large, not aggregated data object will be heavier to transmit than the same data object after its aggregation. Another characteristic that can affect the transmission of a data object is the presence (or not) of cryptographic techniques to protect the communication. For the sake of simplicity, among the

Funded by the European Union

characteristics affecting the transmission we will only consider the size of the data object. A larger data object will be more difficult to transmit, and vice versa.

The same data pipeline can be deployed in different ways depending on which capabilities are implemented by the data provider or by the data consumer. Depending on how the pipeline is deployed, the transmission capability c_t will be executed at a certain point. A *deployment configuration* x of a data pipeline p identifies the capabilities implemented by the data provider and the data consumer in that specific deployment. In other words, a deployment configuration x identifies at which point of the pipeline p the transmission c_t occurs. Given a pipeline p, assuming $c_p \subseteq p$ are the capabilities that must be executed at provider side and $c_c \subseteq p$ the capabilities that must be executed at consumer side, the number of possible deployment configurations for a given p is

$$1 + |p| - |c_p| - |c_c|$$

The definition of a pipeline requires the execution of three steps:

- Agreement phase: the goal of this phase is to define the data pipeline needed to transform the data object as exposed by a FDP offered by the data provider into the data object as exposed by a SFDP requested by the data consumer. Once the pipeline is defined, depending on the possible constraints regarding which are the capabilities that must be placed at provider or consumer side, a set of possible deployment configurations is identified.
- Implementation phase: the goal of this phase is to implement those capabilities at provider and consumer side that have not been already implemented. Depending on the chosen deployment configuration p^x, the data provider will have to implement the capabilities in $\overleftarrow{p}^x$ and the data consumer will have to implement the capabilities in $\overrightarrow{p}^{x+1}$. An exception can be made for the two limit cases, i.e., $x = 0$ and $x = n$ where, respectively, the data provider or the data consumer will not have any capability to implement.
- Execution phase: the goal of this phase is to execute all the capabilities in p in order to effectively transform d_{FDP} in d_{SFDP} and, depending on the chosen deployment configuration, transmit the data object from provider to consumer.

4.1 Effort in Data Sharing

Within the TEADAL project, the effort is defined as the amount of work that an actor, namely the data provider or the data consumer, has to do in order to perform a capability. The effort required by a capability depends not only on the characteristics of the capability itself but also on which actor performs it. Some capabilities can be more easily performed by an actor or another, or they can be performed only by a certain actor.

For instance, removing sensitive information from some data d produced by the FDP can be done only by the data provider: in this case, for every other actor the effort will be infinite since they cannot perform the capability.

The effort required to perform a capability consists of two parts: an implementation effort and an execution effort, which are described next.

4.1.1 Implementation Effort

Funded by the European Union

During the data sharing process, there might be some capabilities needed for the execution of the pipeline that are not present in the data provider's data lake nor in the data consumer's data lake. In order to implement those capabilities, an implementation effort is required.

The implementation effort is defined as the amount of work that a data engineer has to do in order to implement a capability, i.e., writing code or to configure low-code/no-code solutions.

The implementation effort depends on the complexity of the capability: a more complex capability will be more difficult to implement, requiring more effort. In symbols, given the capability c, we express the effort to implement the capability successfully as $E_I(c)$.

An estimate of the implementation effort required by a capability can be obtained by applying software cost estimation models such as SLIM or COCOMO (Leung 2002).

4.1.2 Execution Effort

The execution effort is defined as the amount of work that an actor has to do in order to execute a task. Contrary to the implementation effort, the execution effort depends both on the complexity of the capability and on the size of the data d on which the task is executed. The same task will require more effort when executed on a larger dataset.

In order to make explicit the federated data product size in the definition, a unitary effort $u(c)$ is defined. The unitary effort represents the amount of effort required to execute the capability c on a single data item of the data object. Therefore, the total execution effort required by the capability c, represented by $E_E(c)$, is equal to the unitary effort $u(c)$ needed for one data item times the total amount of data items of the data object d:

$$E_E(c) = u(c) * sizeof(d)$$

The unitary effort required to execute a capability $u(c)$ can be estimated experimentally and it is assumed, at this point, that has a linear trend with respect to the amount of data.

The definition of execution effort and the equation mentioned above also applies to the transmission capability c_t, since the only characteristic affecting the transmission of the data object is its size. The transmission capability is a special instance of a capability, that is applied whenever a data item needs to be transmitted. In this capability, the unitary effort $u(c_t)$ is the effort required to transmit a single data item of the data object and, the bigger the data object size d, the higher the effort to transmit it.

4.2 Friction in Data Sharing

Before initiating the data sharing process, the data provider and the data consumer agree on which data to send and how that data should be presented. If the FDP exposes an interface that is not does not directly provide the data as agreed, a set of capabilities encapsulated in a pipeline have to be deployed and executed. Some of these capabilities may have been already implemented in a previous exchange, and thus does not need to be implemented again and can only be executed.

Data friction μ is defined as the discrepancy between the total amount of capabilities requested for the exchange and the amount of requested capabilities that have already been

Funded by the European Union

performed (for other consumers) before the start of the data sharing process (for the current consumer).

As per the effort, the data friction consists of two parts: an implementation friction and an execution friction.

4.2.1 Implementation Friction

Considering a pipeline p and its deployment p^x, the capabilities that are already implemented by the provider are identified as $C_{Provider}$. This could happen if the same FDP is used by different consumers, each of them through their specific SFDP. In this case, it might happen that several pipelines could have some capabilities in common, thus it is not required to reimplement them. it is worth noticing that we assume that only the provider has the ability to implement the capabilities. The role of the consumer is only to call the SFDP and, if needed, to host and run in its resources the capabilities defined as deployable at consumer side.

The implementation friction μ_I is defined as the discrepancy between the total amount of capabilities in the data pipeline and the amount of capabilities that have already been implemented due to the existence of those capabilities in other pipelines. To overcome the implementation friction, the sum of the provider's and consumer's effort must be greater or equal than the total effort required to implement the capabilities.

The implementation friction can be expressed in symbols as follows:

$$\mu_I = \sum_{c \in p \ / C_{Provider}} E_I(c) / \sum_{c \in p} E_I(c)$$

According to the equation defined above, μ_I will be lower the more capabilities in p have already been implemented.

To overcome the implementation friction, the provider's effort has to be equal (or higher) than the total effort required to implement the capabilities in $p^x / C_{Provider}$, thus:

$$E_I \geq \mu_I \sum_{c \in p} E_I(c)$$

4.2.2 Execution Friction

Once implemented, the capabilities in p have to be executed in order to provide the data object as requested by the consumer. Depending on the chosen deployment configuration p^x, the effort to perform the data exchange will change. Some pipeline capabilities may not need to be executed if there are semi-processed data within the data lakes: the capabilities already executed by the provider are defined as $C'_{Provider}$ whereas the capabilities already executed by the consumer are identified by $C'_{Consumer}$.

The execution friction μ_E is defined as the discrepancy between the total amount of capabilities in the pipeline and the number of capabilities that have already been executed

Funded by the European Union

considering the execution of the other pipelines connecting the same FDP with the SFDP of other consumers.

Since the execution effort of a capability depends also on the size of the data object, and each capability will be executed on the same data object, μ_E can be expressed in terms of the unitary effort $u(c)$ required by each capability:

$$\mu_E = \sum_{c \in p^x / (C'_{Provider} \cup C'_{Consumer}} u(c) / \sum_{c \in p^x} u(c)$$

According to the equation above, the more the capabilities already executed in the past, the lower the μ_E .

In highly dynamic contexts, where the basic data object d is constantly updated with new data, it is unlikely that some capabilities will already be executed. In these scenarios, μ_E will likely be close to 1. Conversely, the less frequent the updates to the data object d the higher the probability that some tasks have already been executed in previous data exchanges. As in the previous phase, the effort to move the data from the FDP to the SFDP according to the defined pipeline will be:

$$E_E \geq \mu_E \sum_{c \in p^x} E_E(c)$$

Funded by the European Union

5 Friction-aware Policy Model

A fundamental element of data governance concerns the definition and enforcement of security policies that regulate access and processing rights to data.

However, at the same time there is a difficulty in combining business needs that are the source of the definition of security policies, with the technological needs of their implementation. High-level of abstraction policies, not formally formulated, are interpreted and implemented by those in charge of managing IT, whose job is to try to translate them into rules and system calls capable of verifying them. Unfortunately, the communication is one-way, i.e., from decision makers who define security policies to IT and security experts who implement the security policies. It is very challenging to verify whether what is prescribed by the business level security policies is actually implemented.

One of the purposes of TEADAL is to provide a tool that can bridge this gap to allow those involved to define policies without going into the merits of the IT structure. This is done using high-level abstraction language that is understandable to those without technical knowledge. The tool provided by TEADAL will lead to the creation of policies at an infrastructure level, described in a language compatible with the technical solutions adopted, and which can be implemented.

To meet the needs defined and, therefore, to avoid the severe consequences described above, we created a three levels framework. Each level targets a different level of details and it approaches the definition of privacy policies taking a different perspective. This allows to cover multiple aspects and many details without over complicating the specification language. We defined the following three levels:

- business level: offering a graphical notation to define a policy
- technical level: based on rego, a datalog-compliant declarative language, used in OPA[9]
- enforcement level: enabling the enforcement of the rego rules in the adopted data lake node .

In particular, specific focus in these first phases of the project has been done to the management of **privacy policies** as its correct enforcement is key to any data sharing system or federation. Wrong privacy policies, not to mention their incorrect or weak enforcement, lead to sever consequences with monetary loss, reputation loss and, in case of law infringements, also very robust fines.

5.1 Business level

The business level targets privacy policies form an organizational perspective. In this level, concepts as actors, and relations between them are used to define privacy policies. In particular, a privacy policy is defined as a set of authorizations between a source actor (who grants the authorization), and a target actor (who receives the authorization), to access or distribute a set of resources that are processed for a given purpose while respecting some constraints.

This level of specification of privacy policies is meant for decision makers, who might have a limited knowledge of the technical enforcement and are more focused on the broad picture of

[9] *For additional information about OPA, please refer to https://www.openpolicyagent.org/docs/latest/policy-language/"*

Funded by the European Union

the flow of authorizations and responsibilities of each actor in the stretched federation. We, therefore, opted for a graphical modelling language that will be used to define privacy policies and the concepts described above.

Figure 16 shows an example of a diagram created with the graphical modelling language we defined for the definition of business level privacy policies. The diagram extends the authorization view of STS-ml (Dalpiaz 2016), a goal-oriented modelling language.

Actors, represented in pink solid circles represents active entities of the federation that can grant or receive authorizations.

*Resource*s represent the most fine-graded level of sharable pieces of data, which in TEADAL federation can be dataset or part of dataset, for example, in a relational database can be relations, columns (projections of relations) or rows (selection of relations). Resources are link to only one actor with the *ownership* relation, which assigns to the actor the full responsibility and authorization of the resource in TEADAL federation. An actor that owns a resource is responsible for the quality of data provided .

Two actors can be linked with the *belongs* relation, which represents a social relationship between the two of them. This relation specifies that the source actor is part of the set of actors represented by the target actor. For example, an employee belongs to a department which, in turn, belongs to an organization.

Authorizations are represented with two large white boxes and a set of smaller boxes on top and on the bottom of them. They connect a source actor, who grants the authorization, and a target actor, who receives the authorization. The lower main box contains the resources targeted by the authorization, i.e., the resources for which the authorization is granted. The upper main box contains the purposes for which the authorization is valid.

The upper set of boxes specifies the authorization types that will need to be enforced when the resources are shared with the target agent. The R box, if checked, grants the authorization to read the resource, while the D box, if checked, grant the authorization to further distribute the resource. The *S:X* box specifies the maximum amount of time (X) the resources can be stored in target actor premises, while the *G:X* box, if checked, specify the geographical location where the resource must be stores.

It is worth specifying that constraints on the purpose of utilization of the resource and on the usage, cannot be enforced by the source actor if the target actor belongs to an external organization. In this case, such constraints are enforced using the agreement between the source agent and the target agent, for example with the definition of penalties.

The set of boxes on the bottom part of the authorization specifies constraints that must be met before the data is shared. The E box, if marked, specifies that the resource will be encrypted before being transmitted to the target actor, a checked A box specifies that data will be anonymized, while a checked P box specifies that the resource will be pseudo-anonymized. The C box, if marked, specifies that consent need to be present, for the purpose specified in the upper main box.

Funded by the European Union

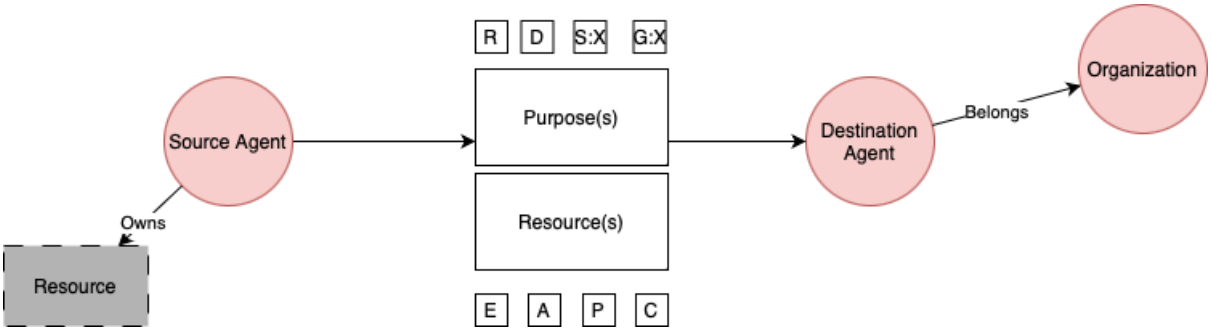

Figure 16 - Example of a diagram created with the business level privacy policy modelling language

Figure 17 shows an example of a diagram defined using the modelling language specified above, of an authorization taken from the Marina Salud pilot. This diagram specifies that Research Doctor authorizes Marina Salud to read all his/her resources and that the resources are encrypted before transmitting them. The “all resources” resource is a placeholder that can be used to indicate all the resources connected to the source actor with a *owns* relation. Authorizations are inherited by actors that *belong to* the authorized one. In this case Doctor and Nurse actors inherit the authorization. In other terms, Doctor and nurse actor are authorized to read all resources of research doctor for the purpose of attending patients, data will be encrypted.

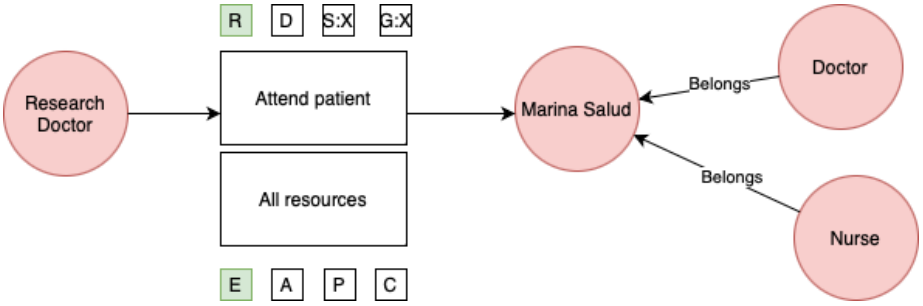

Figure 17 - example of a diagram for the Marina Salud case study

Figure 18 shows an example of a privacy policies derived from the Marina Salud pilot, that consist in a chain of authorizations. In this case an external hospital, identified as Nino Jesus actor, authorizes Marina Salud hospital to read and distribute clinical data resources for the purpose of lung cancer trials. The resource provided is encrypted and consent have been collected for this authorization. Marina Salud in turn authorizes an external laboratory to read the clinical data resource. This resource will be anonymized and encrypted before being accessible by the external laboratory. The laboratory will store the received data in the European Economic Area (EEA) for maximum 5 Months. The nurse actor, since it belongs to the external laboratory, will inherit the same authorization.

Funded by the European Union

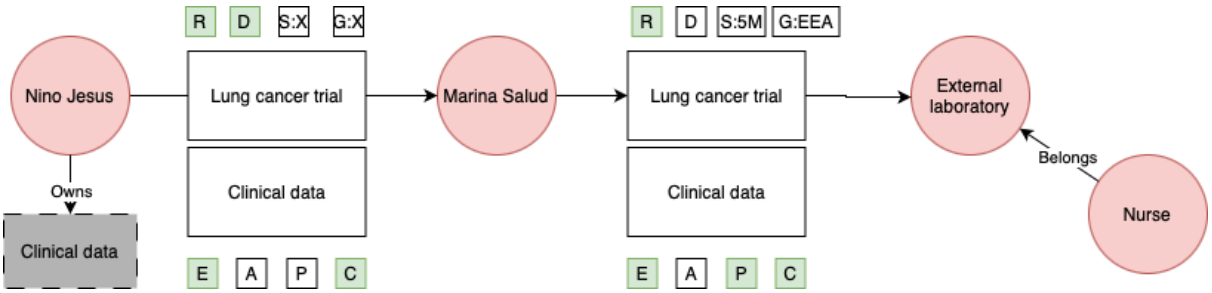

Figure 18 - example of a diagram for the Marina Salud case study

As specified above, the modelling language defined in this chapter will be used to design privacy polices of the TEADAL stretched federation. In particular, it will be used to define privacy policies of the Federated Data Product (FDP), defined in Section 1.3, and of the Sharded Federated Data Product (SFDP).

5.2 Technical level

The technical level aims to specify privacy policies in a human- and machine-readable manner. Privacy policies defined on technical level are strictly bound on the technological solution(s) that is chosen to enforce them. Consequently, also the scope of the policies is limited by the scope of enforceability of the policies.

For what concerns the TEADAL's stretched federation, we opted to specify privacy policies with a declarative logic programming language called Rego, which is based on Datalog. Being Rego a declarative language, it can be used to clearly define privacy policies in a decidable manner.

Datalog is syntactically a subset of Prolog[10], but generally uses a bottom-up rather than a top-down evaluation model. In Datalog with respect to Prolog, there are no symbols of function and there is a non-procedural model of evaluation. Datalog is a series of rules, each rule is composed by a head, also called Left-Hand Side (LHS), on the left of ":-" and a body, also called Righ-Hand Side (RHS), on the right of ":-" .

A rule has the following structure:

$$P :- P1,\ P2,\ P3, P4,\ ...,\ P_n.$$

Each P_i is called a *fact* and it is an instance of a predicate composed by:

- Its name.
- A list of arguments between round brackets:
  - *constants*,
  - *variables*,
  - symbol *do not care* (_) which cannot appear in the head.

To guarantee safety, all the variables in the LHS must appear in the RHS. LHS is true if

[10] https://en.wikipedia.org/wiki/Prolog

Funded by the European Union

RHS is true. In a rule, the notation: P1, P2, . . ., Pn means intersection ∩. The union of rules P = R ∪ S is expressed by the following notation:

$$P(X,Y) :- R(X,Y).\ P(X,Y) :- S(X,Y).$$

The difference of rules P = R – S, is expressed by the following notation:

$$P(X,Y) :- R(X,Y),\ \neg S(X,Y).$$

Recursive queries are expressed by the following notation:

$$P(X,Y) :- R(X,Y).\ P(X,Y) :- S(X,Z),\ R(Z,Y).$$

Rego extends Datalog to support structured document models such as JSON. Its queries are assertions on data stored in OPA, these queries can be used to define policies that enumerate instances of data that violate the expected state of the system. The policies written in Rego are easy to read and write. Rego focuses on providing powerful support for referencing nested documents and ensuring that queries are correct and unambiguous. Rego is declarative so policy authors can focus on what queries should return rather than how queries should be executed. These queries are simpler and more concise than the equivalent in an imperative language.

Listing 1 shows an example of a piece of Rego code. Line 1 defines that, by default, no authorizations are granted. The code makes use of the *input* variable, which allows to get information on the request performed to access data and information that the requester of the data might provide. Lines 2-7 define an authorization which specifies that if the request method is “GET” and the path of the resource requested is “getSalary” and the user requesting the resource is a manger, then the authorization is granted.

```
1  default allow = false
2  allow = true {
3      input.method == "GET"
4      input.path = ["getSalary", user]
5      managers := data.managers[input.user][_]
6      contains(managers, user)
7  }
```

Listing 1 Example of Rego code

Using Rego is possible to define privacy policies that can be interpreted by a Policy Decision Point (PDP) that will then be enforced in the stretched federation. The modeling language defined at business and technical level are deeply different and can be used to represent and define different aspects of privacy policies of a federation. The two modeling languages have been defined to share some concepts, allowing to transform a business level privacy policy in a technical level policy, in a semi-automated fashion. In other terms, it is possible to generate part of the Rego code, that will be interpreted by a PDP, from the business level modeling language.

Funded by the European Union

Being Rego a declarative logic programming language, it lets the developer free to choose the structure of the authorization depending on the PDP implementation that is chosen. Consequently, the transformation rules may widely vary based on the chosen structure of authorization. Below, an example of transformation is reported, based on one of the possible structures of authorization that may be chosen for TEADAL.

Listing 2 shows an example of a Rego authorization derived from the left authorization of the business level privacy policy defined in Figure 3. It is composed of two Rego files, the first one, rbacdb.rego, contains the definition of the actor targeted by the authorization relation in the figure. It specifies that the actor is allowed to access in reading mode (line 3), the resource clinical data of Marina Salud (line 4). The second Rego file, service.rego, specifies the constraints. Lines 7-12 specify that if the *external laboratory* user (line 9) tries to access the resource *clinical data* (line 8), then the output resource, i.e., the resource exposed with the SFDP to the external laboratory, will be *encrypted* and *pseudo-anonymized* (line 7). Furthermore, the resource will contain some meta data that specifies constraints on the time limit of storage (line 10) and on geographical location (line 11). Lines 13-16 specify that the consent is granted if in the request of the resource clinical data, the specified purpose is *lung cancer trial*.

```
      rbacdb.rego
1     external laboratory: [
2           {
3              "methods": http.read,
4              "url_regex": "^/httpbin/marinaSalud/clinicalData"
5           }
6        ]
      service.rego
7     encrypted, pseudo-anonymised if {
8        regex.match("^/httpbin/marinaSalud/clinicalData*", http_request.path)
9        http_request.user == "external laboratory"
10       output.timelimit == “5 months”
11       output.geographicallimit == “EEA”
12    }
13    consent if {
14       regex.match("^/httpbin/marinaSalud/clinicalData*", http_request.path)
15       http_request.purpose == "lung cancer trial"
16    }
```

Listing 2 Example of a rego authorization in TEADAL stretched federation

Funded by the European Union

5.3 Enforcement level

The enforcement level specifies the security mechanisms that will be used to enforce the specifications defined at the technical level. The two levels are closely related since the security mechanisms defined in the enforcement level must be able to interpret the specification on the technical level. In TEADAL the Open Policy Agent (OPA) PDP will be deployed. OPA can parse and interpret privacy policies defined in Rego.

OPA, being a PDP, covers only the interpretation of the privacy policies. A Policy Enforcement Point (PEP) and an executor of the required transformation defined in the authorization will be defined in the next period of the project.

Funded by the European Union

6 Concluding remarks

The results of WP3 described in this document have allowed to define the general architecture of the TEADAL data lake as well as inspiring the definition of the approaches for the TEADAL pipelines between data locations studied in WP4. At the same time, the proposed model is compatible with the model of observability in data sharing studied in WG5.

In the continuation of the activities, the proposed governance model will enrich the policy definitions, data catalog, and fricition solutions described in this document, as well as propose a gravity model. In addition, particular attention will be paid to the study of the impact of the proposed solutions in terms of reducing energy consumption.

Funded by the European Union

REFERENCES

(Alter 2015) G. C. Alter and M. Vardigan. Addressing global data sharing challenges. Journal of Empirical Research on Human Research Ethics, 10(3):317–323, 2015.

(Bates 2018) J. Bates. The politics of data friction. Journal of Documentation, 74(2):412–429, 2018

(Dehghani 2022) Dehghani, Z.: Data Mesh: Delivering data-driven value at scale. O'Reilly Media (2022)

(EC 2020) European Commission: A European strategy for data (2020)

(Edwards 2010) P. N. Edwards. A Vast Machine: Computer Models, Climate Data, and the Politics of Global Warming. MIT Press, 2010.

(Edwards et al. 2011) N. Edwards, M. S. Mayernik, A. L. Batcheller, G. C. Bowker, and C. L. Borgman. Science friction: Data, metadata, and collaboration. Social studies of science, 41(5): 667–690, 2011

(Erl 2007) T. Erl, SOA: Principles of Service Design, Prentice hall, 2007

(Evans 2004) E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2004.

(Gieß 2023) Gieß, A., Möller, F., Schoormann, T., Otto, B.: Design options for data spaces. In: ECIS 2023 (2023), https://aisel.aisnet.org/ecis2023_rp/287

(Hummel 2021) Hummel, P., Braun, M., Tretter, M., Dabrock, P.: Data sovereignty: A review. Big Data & Society 8(1), 2053951720982012 (2021). https://doi.org/10.1177/2053951720982012

(Lefebvre 2023) Lefebvre, H., Flourac, G., Krasikov, P., Legner, C.: Toward cross-company value generation from data: Design principles for developing and operating data sharing communities. In: Design Science Research for a New Society. pp. 33–49 (2023)

(Leonelli 2013) S. Leonelli, N. Smirnoff, J. Moore, C. Cook, and R. Bastow. Making open data work for plant scientists. J. of Experimental Botany, 64(14):4109–4117, 2013

(Leung 2002) H. Leung and Z. Fan. Software cost estimation. In Handbook of Software Engineering and Knowledge Engineering: Volume II: Emerging Technologies, pages 307–324. World Scientific, 2002.

(Murray-Rust 2008) P. Murray-Rust. Open data in science. Nature Proceedings,, 2008

(Tenopir et al. 2015) C. Tenopir, E. D. Dalton, S. Allard, M. Frame, I. Pjesivac, B. Birch, D. Pollock, and K. Dorsett. Changes in data sharing and data reuse practices and perceptions among scientists worldwide. PloS one, 10(8):e0134826, 2015

(Wixom 2020) Wixom, B.H., Sebastian, I.M., Gregory, R.W.: Data shar-ing 2.0: new data sharing, new value creation (Oct 2020) https://cisr.mit.edu/publication/2020_1001_DataSharing_WixomSebastianGregory

(Dalpiaz 2016) Dalpiaz, Fabiano, Elda Paja, and Paolo Giorgini. Security requirements engineering: designing secure socio-technical systems. MIT Press, 2016.

Funded by the European Union