2024

Draft Document

DELIVERABLE 1.2

THIS DOCUMENT IS IN DRAFT FORM AND PENDING OFFICIAL APPROVAL. IT IS SUBJECT TO REVIEW AND MAY BE UPDATED.

D1.2: MVP DEFINITION AND ARCHITECTURE

This project has received funding from the European Union's Horizon Research and Innovation Actions under Grant Agreement № 101093216.

Title:	Document version:
D1.2 MVP Definition and Architecture	0.7

Project number:	Project Acronym	Project Tittle
101093216	UPCAST	Universal Platform Components for Safe Fair Interoperable Data Exchange, Monetisation and Trading

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M9 (September 2023)	M9 (September 2023)	O-PU

*Type: P: Prototype; R: Report; D: Demonstrator; O: Other; ORDP: Open Research Data Pilot; E: Ethics.

**Security Class: PU: Public; PP: Restricted to other program participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	Organization:	Contributing WP:
Shanshan Jiang	SINTEF	WP1
Audun Vennesland	SINTEF	WP1

Contributing Authors (organization):
Shanshan Jiang (SINTEF), Audun Vennesland (SINTEF), Luis-Daniel Ibáñez (University of Southampton), George Konstantinidis (University of Southampton), Semih Yumusak (University of Southampton), Jaime Osvaldo Salas (University of Southampton), Tek Raj Chhetri (University of Southampton), Margherita Forcolin (Maggioli), Kostas Kalaboukas (Maggioli), Sofoklis Efremidis (Maggioli), Despina Psimaris (Maggioli), Vasiliki Nasi (Maggioli), Hanene Jemoui (CeADAR), Julia Palma (CeADAR), Yibrah Gebreyesus (CeADAR), Aditya Grover (CeADAR), Ricardo Simon Carbajo (CeADAR), Eugenia Papagiannakopoulou (ICT ABOVO), Georgios Lioudakis (ICT ABOVO), Mariza Koukovini (ICT ABOVO), Santiago Andres Azcoitia (LS TECH), Evangelos Kotsifakos (LS TECH), Pelayo Fernandez Blanco (LS TECH), Amit K. Srivastava (ALCATEL- LUCENT (Nokia)), Olga Papadodima (NATIONAL HELLENIC RESEARCH FOUNDATION), Paraskevi Tarani (Major Development Agency of Thessaloniki), Anthi Tsakiropoulou (Major Development Agency of Thessaloniki), Lazaros Ioannidis (Open Knowledge Foundation Greece), Panagiotis Philippidis (Open Knowledge Foundation Greece), Nenad Stojanovich (NISSATECH), Milan Vuckovic (NISSATECH), Fernando Perales (JOT INTERNET MEDIA), Miguel Lopez (JOT INTERNET MEDIA), Alexandros Lemperos (CACTUS), Lorenzo Gugliotta (KU Leuven)

Abstract:

This document is deliverable D1.2 MVP Definition and Architecture for the UPCAST project. The document describes the initial UPCAST architecture including the functionality and interfaces of the UPCAST plugins, the final pilot design and functionalities to demonstrate the use of these plugins, and the definition of main features to be delivered in the UPCAST MVP. It also includes an initial description of the vocabulary and data model to be used in UPCAST. In addition, this deliverable is accompanied with a .qea/XMI file that contains architectural models defined in UML and BPMN diagrams, and a website that documents the UPCAST vocabularies with available download serialization in a number of formats (such as .owl, .ttl).

Keywords:

Technical Architecture, Minimum Viable Product (MVP), Pilot Design and Functionality.

REVISION HISTORY

Revision:	Date:	Description:	Author (Organisation)
V0.1	29.06.2023	First version of the document, with suggested structure and content	Shanshan Jiang (SINTEF)
V0.2	15.08.2023	Second version, adding content in many sections.	Shanshan Jiang (SINTEF)
V0.3	31.08.2023	Updated with new structure. Pilot use cases updated following the new approach.	Shanshan Jiang (SINTEF)
V0.4	15.09.2023	Draft content in most sections. Version sent for first internal review.	Shanshan Jiang (SINTEF)
V0.5	19.09.2023	A complete version for second internal review.	Shanshan Jiang (SINTEF)
V0.6	23.09.2023	Addressed most comments from first and second internal reviews	Shanshan Jiang (SINTEF)
V0.7	26.09.2023	Final version for submission, cleaned up and fixed remaining issues.	Shanshan Jiang (SINTEF)

This project has received funding from the European Union's Horizon Research and Innovation Actions under Grant Agreement Nº 101093216.

More information available at https://upcastproject.eu/

COPYRIGHT STATEMENT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 UPCAST Project

UPCAST, Universal Platform Components for Safe Fair Interoperable Data Exchange, Monetisation and Trading, provides a set of universal, trustworthy, transparent and user-friendly data market plugins for the automation of data sharing and processing agreements between businesses, public administrations and citizens. Our plugins will enable actors in the common European data spaces to design and deploy data exchange and trading operations guaranteeing:

- automatic negotiation of agreement terms;
- dynamic fair pricing;
- improved data-asset discovery;
- privacy, commercial and administrative confidentiality requirements;
- low environmental footprint;
- compliance with relevant legislation;
- ethical and responsibility guidelines.

UPCAST will support the deployment of Common European data spaces by consolidating and acting upon mature research in the areas of data management, privacy, monetisation, exchange and automated negotiation, considering efficiency for the environment as well as compliance with EU and national initiatives, AI regulations and ethical procedures. Four real-world pilots across Europe will operationalise a set of working platform plugins for data sharing, monetisation and trading, deployable across a variety of different data marketplaces and platforms, ensuring digital autonomy of data providers, brokers, users and data subjects, and enabling interoperability within European data spaces. UPCAST aims at engaging SMEs, administrations and citizens by providing a transferability framework, best practices and training to endow users in order to deploy the new technologies and maximise impact of the project.

Work package 1, UPCAST concept and MVP definition, addresses the following project objectives:

- Objective 1: Apply models and standards to easily specify data processing requirements in the context of common European data spaces,
- Objective 4: Enable interoperability of data sharing across different entities, platforms and marketplaces,
- Objective 5: Provide a legal and ethical framework for automated contracts, and,
- Objective 8: Pilot and evaluate the platform in Real Market Dataspaces.

These project objectives will be achieved by WP1 through the following sub-objectives:

- Sub-objective 1.1 Establish the vision and direction for the project by defining a Minimum Viable Product (MVP) and agreeing the technical and pilot requirements and usage scenarios to achieve sustainability of the UPCAST set of tools. A methodology to define the requirements and the MVP will be used to stir up the process.

- Sub-objective 1.2 Define the data model and vocabularies for expressing the UPCAST workflows, preferences and other features based on extending existing efforts in GAIA-X and IDS.
- Sub-objective 1.3 Provide a legal framework based on European and National regulations, best practices and ethics guidelines for UPCAST.
- Sub-objective 1.4 Define the UPCAST system architecture using best practices on architecture specification in compliance with the data spaces, along with legal and ethical aspects.

1.2 Purpose of the Document

This document aims to provide the technical design of the initial UPCAST architecture and the definition of the MVP to be demonstrated in the selected business cases in UPCAST pilots. It documents the technical design of the UPCAST architecture and pilot demonstrations using technical models and specifications. It also describes the initial version of vocabularies and data models to be used by the UPCAST plugins and the pilot demonstrations.

This document is accompanied by a .qea/XMI file that contains the architectural models represented in UML and BPMN diagrams used in this document, and a website that documents the UPCAST vocabularies with available download serialization in a number of formats (such as .owl, .ttl).

This document serves as the technical specification and guideline for the development of the MVP and the pilot business cases to be implemented in WP2-WP5.

1.3 Scope of the Document

This document is based on deliverable D1.1 Project Concept Requirements Setup, and describes the result of an iterative process that elaborates and refines the MVP definition, the technical design of the UPCAST architecture, and the design of the UPCAST pilots. It gathers input from the following tasks:

- Task 1.1 (MVP Definition and Requirements for the Data Value Chain): definition of the main features to be delivered in the UPCAST MVP;
- Task 1.2 (Pilot Design and Functionalities): the final pilot design and functionalities based on elaboration of the initial pilot use cases and requirements;
- Task 1.3 (Vocabulary and Data Model): the initial input related to the vocabulary and data model to be used in UPCAST MVP and pilots.
- Task 1.5 (UPCAST Tools Design and Architecture): the initial UPCAST architecture and interface to develop the MVP.

D1.2 documents the UPCAST pilot design and functionality using technical models and diagrams based on the input from UPCAST deliverable D1.1. For an overview of a high-level description of the pilots, their main technical challenges, datasets and requirements, kindly refer to D1.1.

1.4 Structure of the Document

The remainder of the document is organised as follows: Chapter 2 describes ARCADE framework and how it is used to design and document the initial UPCAST architecture. Chapter 3 provides a high-level description of the MVP and its main features from a user

journey perspective. Chapter 4 describes the initial UPCAST conceptual and technical architecture as well as vocabularies. Chapter 5 to Chapter 9 describe the design and functionality of each UPCAST pilot using the views defined in ARCADE framework. These include the roles and stakeholders involved, the reference use cases representing pilot functionality, the UPCAST plugins involved to implement the pilot use cases, and pilot implementation plans. Finally, Chapter 10 concludes the deliverable and proposes future work.

2 ARCADE FRAMEWORK

The UPCAST architecture design and description is based on the ARCADE framework[1]. The ARCADE framework is an architecture description framework based on IEEE standard for system architecture specifications, and it is used in UPCAST to create a holistic system architecture using a set of interconnected viewpoints. Each *viewpoint* specifies a specific *view* of the architecture using *models* describing different aspects regarding the structure and behaviour of the system as illustrated in Figure 1 (the middle box). The ARCADE framework provides guidance on using the different viewpoints, views and models to describe the system architecture. It is a flexible approach, meaning that not all the views and models must be used, but they can be selected according to the context and purpose of the architecture design.

ARCADE defines the following views with respective models:

- Context View: describe the various aspects of the target system's environment, including its stakeholders, intended functionality described in use cases, business processes, the scope of the target system and its environment, and the mapping from the business aspects to the target system.
- Requirements View: document all specific requirements related to the target system.
- Component View: identify and document physical or logical components regarding their data, interfaces and functionality.
- Distribution View: describe the logical distribution of software and hardware components.
- Realisation View: describe the realisation of the system with its subsystems, including their deployment.

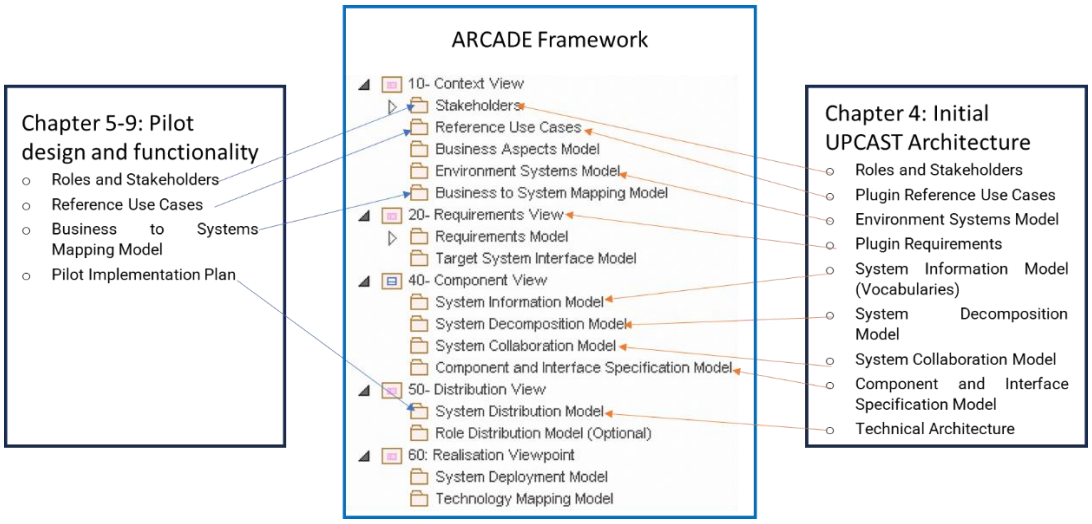

Figure 1: Mapping between D1.2 content and ARCADE framework.

UPCAST uses the ARCADE framework to model the realisation of the UPCAST MVP including the UPCAST architecture and the demonstrating pilots. As part of the preparation of D1.1, an initial version of the Context View, Requirements View and

[1] http://arcade-framework.org/

Component View were created to provide a contextual basis for relevant requirements specification. These views documented the following: As-is and to-be use case models of pilots developed in interviews with pilots; pilot user stories and requirements; plugin requirements. Taking these views as a starting point, D1.2 completes the initial architectural models by instantiating and elaborating the Component View and Distribution View. During this process, the 3 views (Context, Requirement and Component) developed in connection with D1.1 are further elaborated, updated and extended. Furthermore, the pilot design is also elaborated using selected views of the ARCADE framework as illustrated in Figure 1.

The following describes the architecture design and MVP Implementation process with the resulting documentation from different steps (see Figure 1). The bold text in parentheses refers to the ARCADE models used in D1.2 as the result of each step. The process is iterative, meaning that each step may be done in several rounds and does not strictly follow the order of this bullet list.

1. Define the generic roles and stakeholders for the UPCAST plugins and pilots (**Roles and Stakeholders** in Section 4.1 for plugins and Section 5.1, 6.1, 7.1, 8.1 and 9.1).
2. Harmonise and prioritise plugin requirements (**Requirements View** in Section 4.4). These plugin requirements can be traced back to the pilot requirements and user stories.
3. Define the generic UPCAST plugin functions that are needed to satisfy the requirements in the pilots (**Reference Use Cases** in Section 4.2). The goal is to specify generic functionality, therefore, the internal functionality (functions and actors) needed to support the pilot objectives is considered but not modelled explicitly.
4. Assess, harmonise and prioritise functions associated with each plugin to establish methods (functions) the plugin should offer (**Reference Use Cases** in Section 5.2, 6.2, 7.2, 8.2 and 9.2).
5. Determine which functionality should be within and outside of UPCAST scope (**Environment Systems Model** in Section 4.3).
6. Define interfaces (methods/services and input/output) of components needed to realise the UPCAST plugins (**Component View** in Section 4.8).
7. Define the UPCAST technical architecture and specify how the plugins will be deployed and interacted (**Technical Architecture** in Section 4.9).
8. Define which plugins will be used to realise pilot use cases **(Business to System Mapping Model** in Section 5.3, 6.3, 7.3, 8.3 and 9.3).
9. Specify how UPCAST plugins will be deployed/implemented in the pilots (**System Distribution Model**–- Pilot Implementation Plan in Section 5.4, 6.4, 7.4, 8.4 and 9.4).

3 MVP DEFINITION

UPCAST provides support for the management, negotiation, and exploitation of resources through a set of plugins that can be installed in Data Marketplaces or other data platforms that can mediate data transactions between providers and consumers. A resource can be a dataset, a data operation or an artefact (such as a machine learning model).

The UPCAST Minimum Viable Product (MVP) is an implementation of the minimum functionality of the UPCAST plugins (described in Section 4.2) that satisfies the requirements prioritised and selected based on the pilot needs and project vision. The MVP will serve to gather valuable feedback for further development.

In this section, we describe the MVP functionality from a user journey point of view. Figure 2 illustrates the core functionality that is included in the UPCAST MVP as a set of functions performed by or provided to either the Resource Provider, the Resource Consumer, or in some cases both roles. The remaining part of the report will describe in detail how the functionality is offered by the UPCAST Plugins (Context View and Component View), how the UPCAST Plugins will be deployed in a technical infrastructure (UPCAST technical architecture), and how this functionality will be implemented in the UPCAST Pilots (Pilot Implementation Plan).

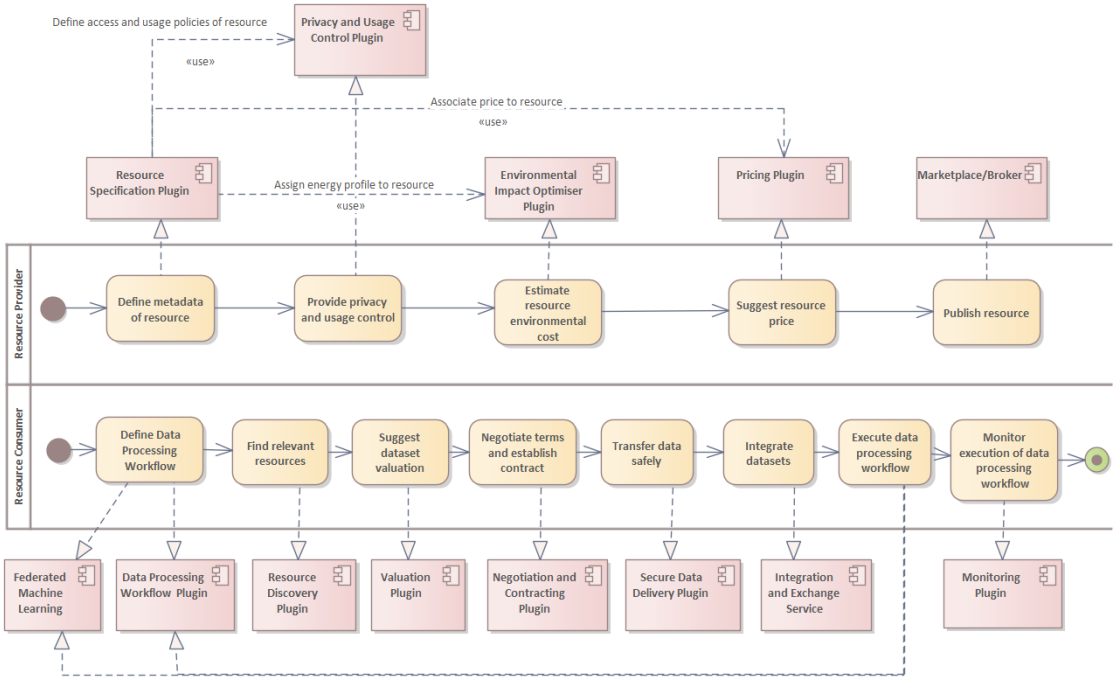

Figure 2: Core functionality included in the UPCAST MVP. The activities included in the MVP are realised by means of the UPCAST Plugins[2] except for "Publish resource" which is provided by a marketplace or broker outside UPCAST.

[2] Only the plugins that are the main entry points for each activity are illustrated in this figure. A plugin may depend on other plugins to deliver the expected functionality. Such dependency is not shown in the figure. In addition, the Negotiation and Contracting plugin is used by both Resource Consumer and Resource Provider, but it is mainly triggered by Resource Consumer, therefore, only the link to the Resource Consumer is shown in Figure 2.

In the MVP definition, we focus on datasets, but some plugins are applicable for other types of resources. The UPCAST plugins are modules that can be deployed on a data marketplace (or other data sharing platform) and offer defined functionality to marketplace users through the marketplace. Plugins interact with each other through well-defined APIs. The users, acting as resource providers or resource consumers, can select a desired plugin from the marketplace and invoke the required functionality through the provided user interface.

Figure 2 shows a representative user journey with activities that involve as many UPCAST plugins as possible. The upper part of Figure 2 illustrates the user journey of a resource provider who wants to publish a dataset[3] using UPCAST plugins. The preparation of a dataset (collection of data, cleaning, and preprocessing) is outside the scope of UPCAST. Therefore, the user journey starts with dataset annotation where the user describes the resource using basic metadata and defines access and usage policies. The user may also assign an energy profile to the resource and associate a price or price range to the resource to facilitate its monetisation. A typical user journey utilising UPCAST plugins is as follows:

RP1. *Define metadata of resource*: Using the Resource Specification plugin, user creates a resource specification and annotates the resource with basic metadata using UPCAST vocabulary and domain-specific vocabularies.

RP2. *Provide privacy and usage control*: Using the Resource Specification plugin, user can further define the privacy and usage control policies in the resource specification with the support from the Privacy and Usage Control plugin.

RP3. *Estimate resource environmental cost*: Using the Resource Specification plugin, user can also create the energy profile for the resource with the resource environmental cost suggested by the Environmental Impact Optimiser Plugin.

RP4. *Suggest resource price*: Using the Resource Specification plugin, the user can assign a price to the resource manually, or with the support of the Pricing plugin, the user will get a suggested price or price range of the resource and understand how the price is formed.

RP5. *Publish resource*: User publishes the resource annotated with the resource specification in a data marketplace or a data catalogue provided by a broker so that others can discover the resource. This functionality is provided by a broker or marketplace that is outside the scope of UPCAST.

The lower part of Figure 2 illustrates the journey of a resource consumer who wants to make use of a dataset. This consumer journey starts with "Define Data Processing Workflow" (RC1) which will utilise datasets from different providers. In some cases, the user does not need to create a data processing workflow (DPW) and will start directly from "Find relevant resources" (RC2). In this case, the DPW can be considered as processing a single dataset. Furthermore, Federated Machine Learning (ML) activity is handled by the Federated Machine Learning plugin. As Federated ML activity is considered as a special type of data processing workflow, we do not address Federated ML separately in the following process.

RC1. *Define Data Processing Workflow*: User defines a series of actions related to pre-processing and processing of datasets using the Data Processing Workflow plugin. A DPW model will be defined, the intended usage and the access and usage policies

[3] In this section, we focus on datasets, but some plugins are applicable for other types of resources.

for the DPW will be specified.

RC2. *Find relevant resources*: User discovers resources to include in the DPW by searching or browsing a Dataset Catalogue or getting suggestions on relevant resources using the Resource Discovery plugin.

RC3. *Suggest dataset valuation*: Using the Valuation plugin[4], the user can get insight on the value of data and understand how the data contributes to a certain task. This can help the user make decisions regarding data monetisation.

RC4. *Negotiate terms and establish contract*: User negotiates with resource providers regarding the terms of access, usage and pricing of the datasets. The result of the negotiation is a contract that states the terms of access and usage, as well as the pricing of the dataset under negotiation. User will get support from the Negotiation and Contracting plugin to facilitate and automate part of the process. In particular, legal assessment is performed to ensure compliance with legal requirements.

RC5. *Transfer data safely*: With the help of the Secure Data Delivery plugin, the dataset contracted will be transferred securely to a trusted environment for its processing by the user.

RC6. *Integrate datasets*: User integrates datasets from various resource providers for the defined DPW using the Integration and Exchange Service. This service is not considered a plugin, but rather a common data processing operation for which UPCAST will offer a solution.

RC7. *Execute data processing workflow*: Using Data Processing Workflow, user starts the DPW execution for the processing of the datasets subject to the terms of access and usage policies that have been negotiated and agreed between the provider and the consumer and are expressed in the negotiation contract.

RC8. *Monitor execution of data processing workflow*: The UPCAST Monitoring plugin monitors the execution of the DPW and notifies user with non-conformance to the DPW specification, such as any access or usage rule violated during the DPW execution.

Some plugins are expected to be essential for performing the tasks for the user journey in line with the UPCAST vision. These "must have" plugins are Resource Specification, Privacy and Usage Control, Data Processing Workflow, Negotiation and Contracting, and Monitoring, while the remaining plugins are optional due to, for example, the operations associated with these plugins are optional in the user journey, or their functionality can be provided by non-UPCAST software (e.g., discovery can be provided by marketplaces).

[4] Note that in UPCAST, the Valuation plugin will be adapted to a specific use case in the Health and Fitness pilot and will not be provided as a general purpose plugin.

4 INITIAL UPCAST ARCHITECTURE AND VOCABULARY

4.1 Roles and Stakeholders

We have defined UPCAST generic roles and stakeholders aligned with standards such as IDSA RAM 4.0[5]. Figure 3 shows these roles for the initial UPCAST architecture.

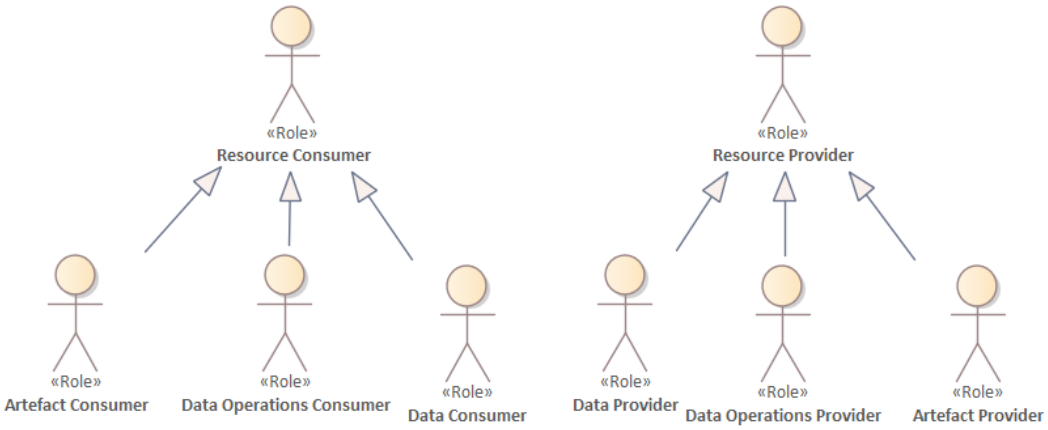

Figure 3: Roles and stakeholders for the initial UPCAST architecture.

In UPCAST, we focus on the Resource Consumer and Resource Provider roles. A Resource Provider makes a resource available for others to use while a Resource Consumer makes use of a provided resource. A resource can be a dataset, a data operation or an artefact. Therefore, the Resource Consumer role is classified into Data Consumer, Data Operations Consumer and Artefact Consumer, and the Resource Provider role is classified into Data Provider, Data Operations Provider and Artefact Provider. In line with the basic roles defined in IDSA RAM 4.0, a Data Provider is an actor that makes datasets technically available for others to use, and a Data Consumer is an actor that makes use of an available dataset. The other roles are not explicitly defined in IDSA RAM, but they are an extension of IDSA RAM regarding general resources that can be shared and exchanged in a data marketplace.

4.2 Plugin Reference Use Cases

This section describes the reference use cases of each UPCAST plugin.

[5] https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3-1-business-layer/3_1_1_roles_in_the_ids

4.2.1 Resource Specification

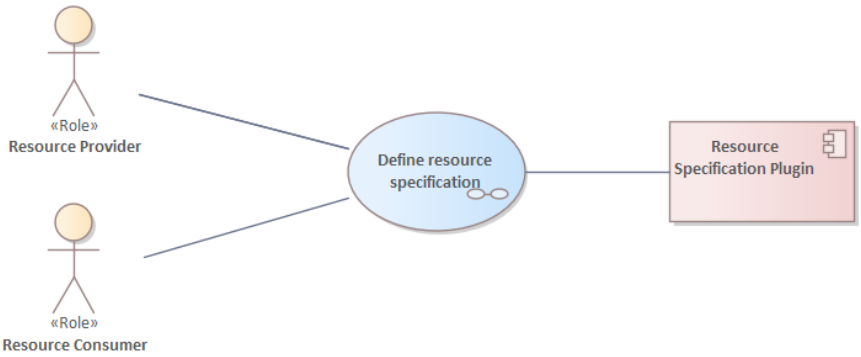

Figure 4: Top-level use case model for the Resource Specification plugin.

Use Case <<Define resource specification>>:

- Precondition/Trigger: User starts the plugin to provide specification of the resource.
- Postcondition: A resource specification is created.
- Description: User starts the resource specification plugin, they can create a new resource.

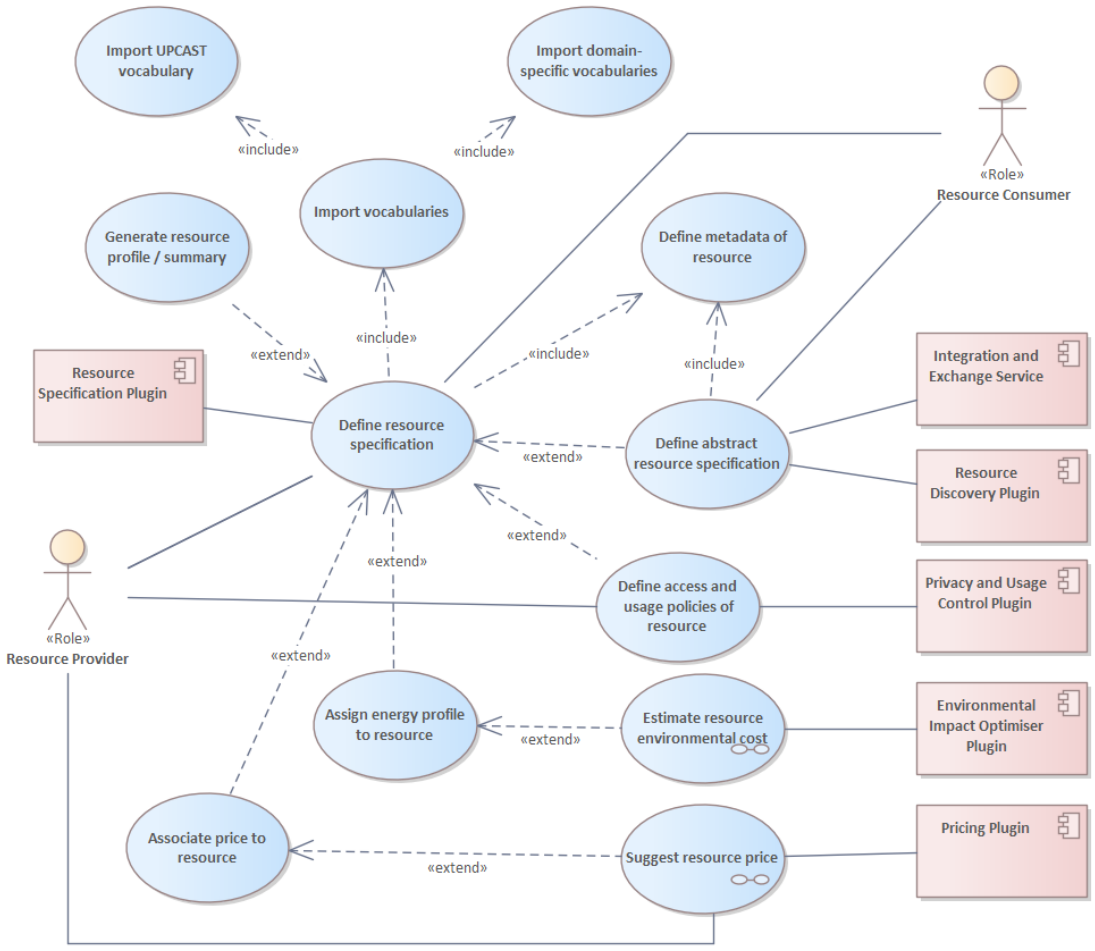

Figure 5: Detail use cases for the Resource Specification plugin.

Use Case <<Import UPCAST Vocabulary>>:

- Precondition/Trigger: A version of the UPCAST vocabulary in machine-readable format (.ttl or .owl) exists. User triggers the use case.
- Postcondition: The UPCAST vocabulary is available to be used for specification of datasets in use case “Define Metadata of Resource.”
- Description: User selects the version of the UPCAST vocabulary they want to use in a .ttl or .owl file. After import, it is available for selection by the user in a configuration section.

Use Case <<Import Domain-Specific Vocabulary>>:

- Precondition/Trigger: The Domain-Specific vocabulary in .ttl or .owl format exist. Triggered by user.
- Postcondition: The Domain-Specific Vocabulary is available to be used for specification of datasets.
- Description: User selects the Domain-Specific Vocabulary they want to import in .ttl or .owl format. After importing the vocabulary is available to be used in the “Define metadata” use case.

Use Case <<Define Metadata of Resource>>:

- Precondition/Trigger: At least one version of the UPCAST vocabulary needs to be loaded in the system.
- Postcondition: A resource specification is created with at least the mandatory attributes of the UPCAST vocabulary. (Optionally, attributes of the Domain-specific-Vocabulary).
- Description: User creates a new resource and describes their metadata using the attributes defined in the UPCAST Vocabulary, except those of access and usage, policy and pricing, and user can also use the attributes defined in any imported Domain-Specific Vocabulary.

Use Case <<Define Access and Usage policies of Resource>>:

- Precondition/Trigger: A resource specification with basic metadata exists.
- Postcondition: The input resource specification is augmented with access and usage policies in the vocabularies.
- Description: User specifies the access and usage policies for the resource at higher level. When the resource is a dataset, user will be able to specify fine grained policies on attributes of the specification that describe the contents of the dataset. E.g., metadata specifies the dataset has a column "Name", with "Name" an entity in the Domain Specific Language (DSL). User can then express

access and usage policies referring to the "Name" column, e.g., "do not use for commercial".

Use Case <<Assign Energy Profile to resource>>:

- Precondition/Trigger: The resource for which the profile will be assigned has been specified with at least basic metadata.
- Postcondition**:** An Energy Profile is added to the resource specification.
- Description: User specifies metadata of the infrastructure on which the resource runs or is located. The metadata is used as input to the Environmental Impact Optimiser (EIO) plugin to estimate the energy consumption. This use case will call the use case <<Perform dataset energy profiling>> of the EIO plugin. The output of the EIO is added to the resource specification. The metadata differs depending on if the resource is a dataset or a processing operation.

Use Case <<Associate price to resource>>:

- Precondition/Trigger: The resource for which the price will be assigned has been specified with at least basic metadata.
- Postcondition: A price is added to the resource specification.
- Description: User may specify the price attribute (assumed to be in the UPCAST vocabulary) manually. Alternatively, for dataset resources a user can send a request to the pricing plugin (having as input the current description of the resource) to get a price suggestion (Use Case <<Suggest resource price>> of Pricing plugin). User may set the pricing attributes following the suggestion.

Use Case <<Generate resource profile/summary>>:

- Precondition/Trigger: The resource for which the profile/summary will be assigned has been specified with at least basic metadata. The Dataset to be profiled is assigned.
- Postcondition: A profile or summary is added to the resource specification.
- Description: User inputs the dataset, selects from a drop-down list the type of profile/summary they want to generate. System generates the selected profile/summary and adds it to the resource specification.

Use Case <<Define Abstract Resource Specification>>:

- Precondition/Trigger: None. Triggered by user or by Discovery and Integration Plugins.
- Postcondition: An abstract resource specification is created.
- Description: The user wants to specify a resource that does not exist or is not known yet (but they want), to either run a discovery task or define the target of

an integration. Abstract resource specifications use the same vocabulary as concrete ones but have variables instead of instances in some variables. In a way they are similar to queries. The steps to define an abstract specification are similar to a concrete one, predicates from the Vocabulary are added to specification, but variables or value ranges may be used instead of constants.

4.2.2 Data Processing Workflow

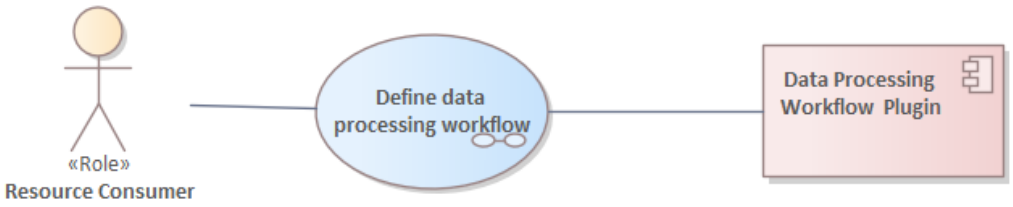

Figure 6: Top-level use case model for the Data Processing Workflow plugin.

Use Case <<Define data processing workflow>>:

- Precondition/Trigger: User wants to implement a service including resources from third-parties.
- Postcondition: Machine- and human- readable DPW model available, or Machine- and human- readable description of the resulting dataset (i.e., the product of the DPW) or artefact (such as a report or a ML model).
- Description: The user specifies a data processing workflow offering some desired functionality. Specifically, a RC defines intended data-centric processes alongside specific access and usage intentions and/or requirements through an easy-to-use no-code design and editing UI. Moreover, the user may define access and usage control constraints at the level of the whole DPW, in case the latter is planned to be advertised as a Resource to be used in other DPWs.

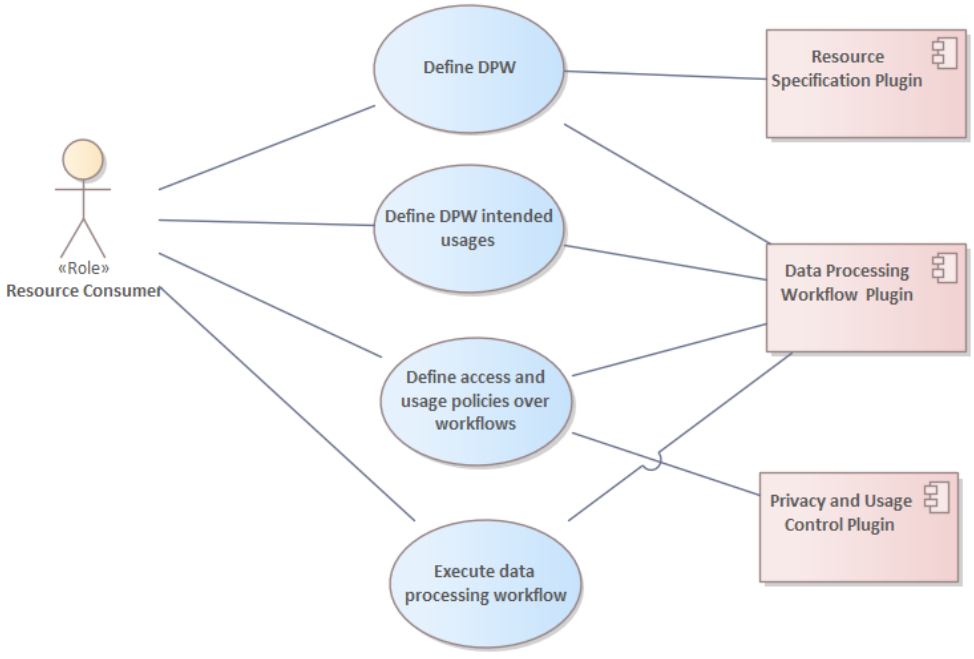

Figure 7: Detail use cases for the Data Processing Workflow plugin.

Use Case <<Define DPW>>:

- Precondition/Trigger: User wants to implement a service including resources from third-parties. Resources can be either already known/discovered or abstract resources from third-parties that will be dynamically discovered and selected according to specific criteria.
- Postcondition: Machine- and human- readable DPW model available
- Description: The user specifies a data processing workflow offering some desired functionality by leveraging its own resources (e.g., data apps and datasets) or resources discovered and bought in an AppStore or Marketplace through an easy-to-use no-code design and editing user interface (UI). Each step of the DPW is defined by specifying the data processing operation to be executed over a specific abstract or concrete resource. For resources that are not yet discovered, user describes the requirements that resources to be discovered have to cover, so as for concrete resources to be mapped to the abstract specifications submitted as queries by the potential resource consumer (cf. Section 4.2.1 use case <<Define Abstract Resource Specification>>).

Use Case <<Define DPW intended usages>>:

- Precondition/Trigger: User wants to implement a service including resources from third-parties.
- Postcondition: Machine- and human- readable DPW model available.
- Description: The user specifies the purposes that the DPW serves and the actor entities that may initiate the DPW. At task level, the user defines the processing operations that will be executed over specific resources. The user may also define constraints (part of their intentions) over the said operations and resources, e.g., for selecting resources satisfying specific criteria.

Use Case <<Define access and usage policies over workflows>>:

- Precondition/Trigger: User wants to implement a service including resources from third-parties, user wants to describe a DPW as a Resource to be advertised.
- Postcondition: Machine- and human- readable description of the resulting dataset (i.e., the product of the DPW) or artefact (such as a report or a ML model).
- Description: The user defines access and usage control constraints at the level of the whole DPW, in case the latter is planned to be advertised as a Resource to be used in other DPWs.

Use Case <<Execute data processing workflow>>:

- Precondition/Trigger: A DPW specification is generated. Negotiation and contracting process has concluded. User triggers the DPW execution.
- Postcondition: The DPW specification starts execution and can be monitored.
- Description: The DPW specification is compiled into an executable model and instructions are inserted into the model so that monitoring messages can be sent to the Monitoring plugin.

4.2.3 Privacy and Usage Control

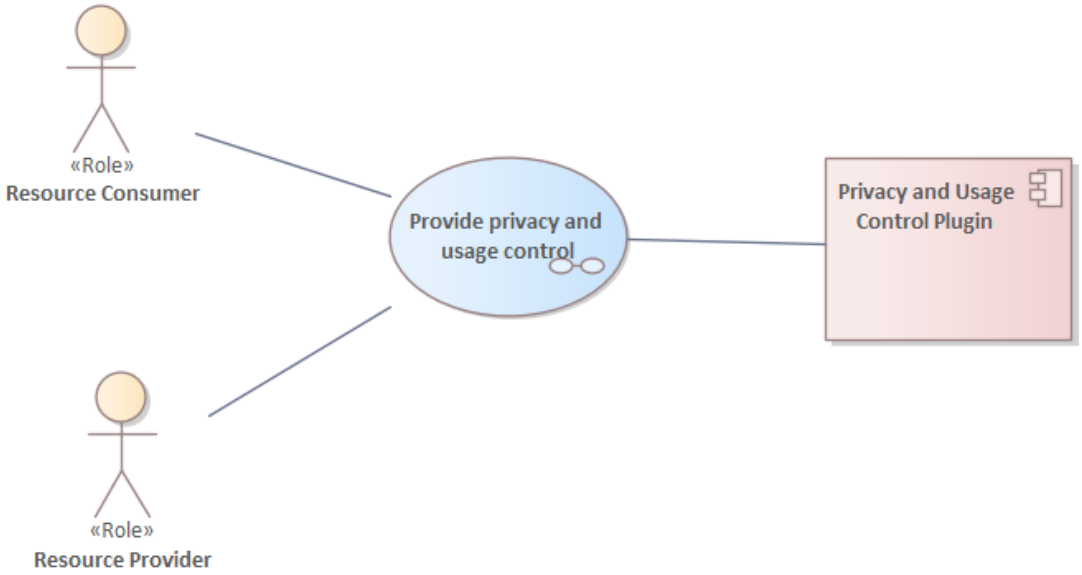

Figure 8: Top-level use case model for the Privacy and Usage Control plugin.

Use Case <<Provide privacy and usage control>>:

- Precondition/Trigger: We have a Resource Provider (RP) and a Resource Consumer (RC) that are paired together by the data marketplace or some other brokerage, to exchange data. In addition: a) RP's constraints defined through Resource Specification; b) RC includes RP's resource in a DPW specification.
- Postcondition: a) RP's constraints defined as Privacy and Usage Control (PUC) rules; b) RC's PUC rules (i.e., organisation-specific and prescribed by the applicable regulations) defined as PUC rules; c), any potentially applicable rules addressing marketplace liability pertaining to it acting as a "data intermediation service"[6], also defined as PUC rules by the marketplace itself; d) Conflicts between RP's constraints and RC's intentions identified; e) RC received authorisation decision/transformed request for accessing a specific resource on the basis of the declared intentions, own ruleset and RP's constraints.
- Description: Upon Resource Specification the RP user defines constraints on their resources which are in turn translated into PUC rules leveraging the UPCAST and domain-specific vocabularies. The RC user defines on the one hand their intentions via the DPW specification, as well as any possible organisation-specific access and usage control rules and rules prescribed by applicable regulations (e.g., GDPR). Thereafter, conflict identification follows between RP constraints, RC intentions and internal rules and derivation of authorisation decisions becomes possible.

[6] https://data.consilium.europa.eu/doc/document/PE-85-2021-INIT/en/pdf

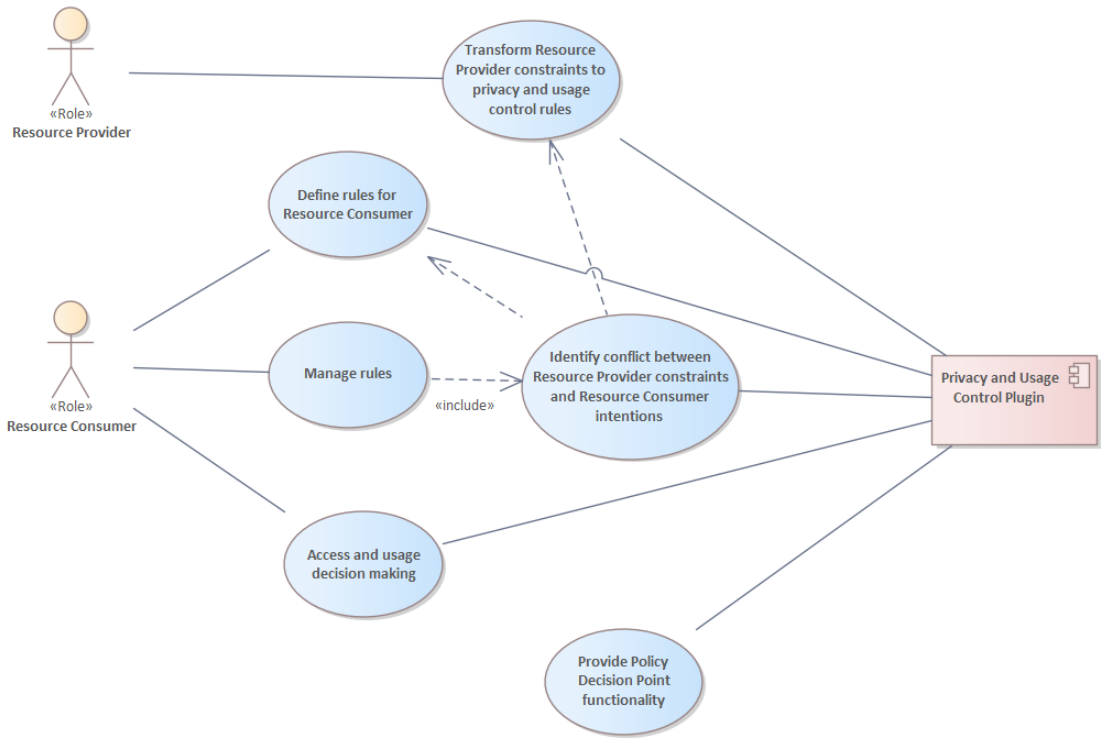

Figure 9: Detail use cases for the Privacy and Usage Control plugin.

Use Case <<Transform Resource Provider constraints to privacy and usage control rules>>:

- Precondition/Trigger: a) RP's constraints defined through Resource Specification (see Use Case <<Define Access and Usage policies of Resource>> in Resource Specification plugin); b) RC includes RP's resource in a DPW specification.
- Postcondition: RP's constraints defined as PUC rules so that they can be compared to RC's intentions in the context of the negotiation process.
- Description: RP's constraints reflected in the respective licences (generated upon Resource Specification) are translated into the underlying semantic policy language for the negotiation to take place. This process is transparent to RP.

Use Case <<Define rules for Resource Consumer>>:

- Precondition/Trigger: RC includes RP's resource in a DPW specification.
- Postcondition: RC's access and usage control rules (i.e., organisation-specific and prescribed by the applicable regulations, as well as DPW-specific) defined as PUC rules.
- Description: The user may specify access and usage control rules over the specified DPW (with the DPW and/or its artefacts constituting potential resources for other DPWs). Additionally, the user defines possible organisation-specific access and usage control rules, along with rules prescribed by applicable regulations (e.g., GDPR). It is noted that access and usage intentions regarding a resource included in the DPW will be also translated into PUC rules, for conflict identification and negotiation to take place.

Use Case <<Manage Rules>>:

- Precondition/Trigger: RC wants to manage their internal ruleset, RC includes RP's resource in a DPW specification.
- Postcondition: RC's access and usage control rules (i.e., organisation-specific and prescribed by the applicable regulations) defined as PUC rules.
- Description: The user may create, update and delete access and usage control rules. Management of generic organisation-specific and legislation-related rules will be provided by a dedicated UI. Management of rules includes conflict resolution and rules merging, i.e., mechanisms for the elimination of deprecated policies (i.e., overridden by other policies), as well as identification of conflicts between RP access and usage constraints and RC access and usage intentions and subsequent suggestions for conflict resolution. It is noted that DPW-specific access and usage intentions referring both to a specific resource included in the DPW and the whole DPW may be managed directly at the level of the DPW modelling (through DPW specification UI).

Use Case <<Identify conflict between Resource Provider constraints and Resource Consumer intentions>>:

- Precondition/Trigger: RC includes RP's resource in a DPW specification.
- Postcondition: Conflicts between RP's constraints and RC's intentions identified.
- Description: RC access and usage intentions along with organisation-specific and legislation-related rules are checked against RP access and usage constraints and possible conflicts are identified and presented to user in an intuitive and user-friendly way (cf. "Present and resolve negotiation conflicts" use case in Section 4.2.8). This may also include suggestions for conflict resolution to be taken into consideration during the negotiation process.

Use Case <<Access and usage decision making>>:

- Precondition/Trigger: RC includes RP's resource in a DPW specification.
- Postcondition: RC received authorisation decision/transformed request for accessing a specific resource on the basis of the declared intentions, own ruleset and RP's constraints.
- Description: In case that no conflicts leading to access and usage request rejection between RP access and usage constraints and RC intentions have been identified, the user receives the authorisation decision for their request for a specific resource (expressed by means of their intentions), that may be a) accept the request as is, possibly prescribing/forbidding the execution of other actions in the future; b) accept a transformed version of the original request, by means of selection, projection and change of state of fields, possibly prescribing/forbidding the execution of other actions in the future.

Use Case <<Provide Policy Decision Point functionality>>:

- Precondition/Trigger: Other services/plugins require Policy Decision Point (PDP) functionality.
- Postcondition: Authorisation decision/transformed request for accessing a specific resource generated.

- Description: Privacy and Usage Control Plugin provides advanced PDP functionality to other services or plugins (e.g., in the context of DPW execution and /or monitoring). It provides for request transformation where possible; instead of simply allowing or denying an incoming access/usage request, it allows request transformation (e.g., allow access to/usage of parts of the requested data) and/or prescribe/forbid the execution of subsequent actions.

4.2.4 Pricing

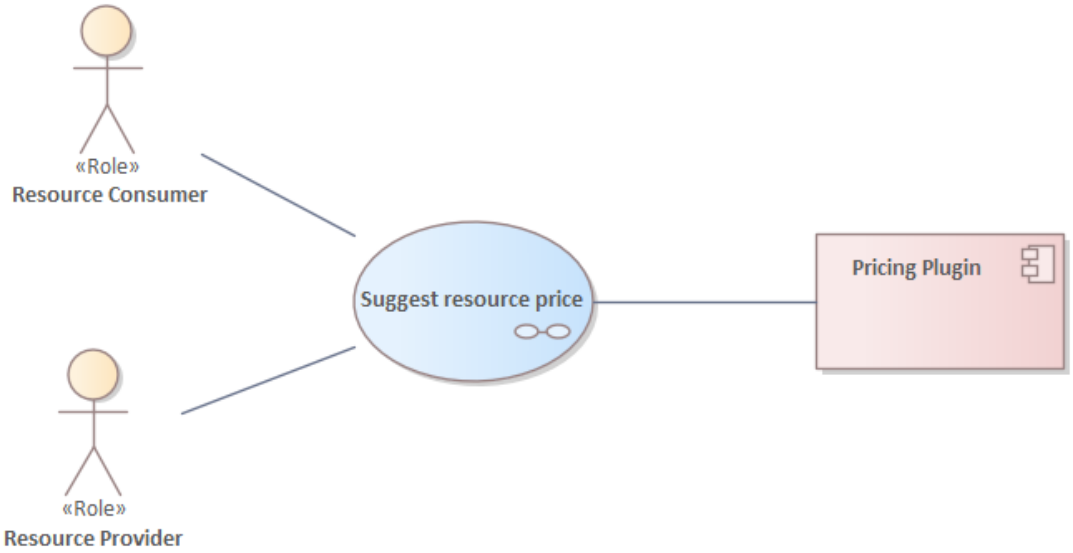

Figure 10: Top-level use case model for the Pricing plugin.

Use Case <<Suggest resource price>>:

- Precondition/Trigger: Resource Specification with basic metadata is defined. The user sends a request to get a price suggestion. Users of the pricing plugin can be, for example, data providers interested in setting the price of their data products or data consumers interested in obtaining a reference of the price of a certain data product they are willing to acquire.
- Postcondition: A price or price range for the resource is provided. Additionally, similar resources may be suggested.
- Description: Plugin users can use the plugin to get a suggested price or price range for selected resource using static or dynamic pricing models. Users may also get explanation of the estimated price. Additionally, they can get suggestions on similar resources.

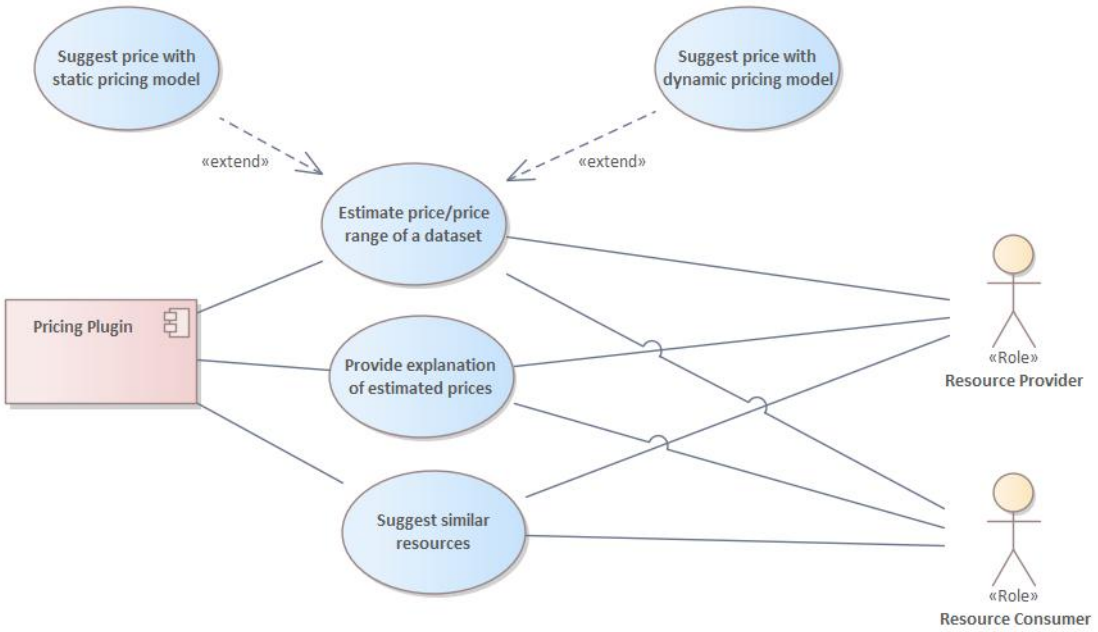

Figure 11: Detail use cases for the Pricing plugin.

Use Case <<Estimate Price / Price Range of a Dataset>>:

- Precondition/Trigger: Resource Specification with relevant metadata is defined. The user sends a request to get a price suggestion for this dataset.
- Postcondition: User gets a specific range of prices for a dataset through the static or dynamic pricing model(s).
- Description: This use case represents the core functionality of the plugin, which is to suggest price or price range for datasets using static or dynamic pricing models. The use case thus is extended by the use cases <<Suggest price with static pricing model>> and <<Suggest price with dynamic pricing model>>, which use different mechanisms and inputs to provide the estimation.

Use Case <<Suggest price with static pricing model>>:

- Precondition/Trigger: Resource Specification with basic metadata is defined. The user sends a request to get a price suggestion for this resource.
- Postcondition**:** User gets a specific range of prices for a dataset and the static pricing model(s) available.
- Description: Plugin users will be returned a specific range of prices of data products in accordance with the description and metadata attributes they submit as inputs. The estimation will be calculated based on regression models fed with market information about data products currently being commercialized in real data marketplaces.

Use Case <<Suggest price with dynamic pricing model>>:

- Precondition/Trigger: Resource definition with UPCAST vocabulary exists, marketplace characteristics are provided. The user sends a request to get a price suggestion for this resource.
- Postcondition**:** Fixed or range of market-driven prices of a dataset available.
- Description: The dynamic pricing models will return a fixed or range of market-driven prices of a dataset, potentially using static price estimations for bootstrapping, and incorporating market characteristics, based on marketplace interactions.

Use Case <<Provide explanation of estimated prices>>:

- Precondition/Trigger: The pricing model(s) and dataset metadata are defined. The user sends a request to get an explanation of the features that are affecting the price suggestion for this dataset.
- Postcondition**:** The relevant features or attributes contributing to the suggested price of the dataset and the degree to which each feature contributes (as a percentage) to the provided range of prices are provided.
- Description: Plugin users may ask for further explanation of a certain estimation provided by the plugin to know what the most relevant features contributing to the price of their data are. For that purpose, the plug-in will return an estimation of price for the description and metadata attributes provided by the user, and a list of the metadata features and / or group of features and their contribution to that estimation (a percentage or an explanatory real number). The end user will be able to graphically visualise the explanation.

Use Case <<Suggest similar resources>>:

- Precondition/Trigger: Resource definition with UPCAST vocabulary exists. The user sends a request to get similar data products to this resource in the market.
- Postcondition**:** The user will be provided information of data products found in commercial data marketplaces that are closer to the resource specified as input.
- Description: Plugin users may ask for data products that are like the resource described as an input and comply with any metadata attribute constraints stated in the request. The output would be a list of data products in the database that conform to those characteristics including, when available, their price.

4.2.5 Valuation

Even though the methodology of the valuation plugin will be designed to be generic and applicable to a wide range of use cases, its design and its implementation will be restricted to the scope of the Health and Fitness use case.

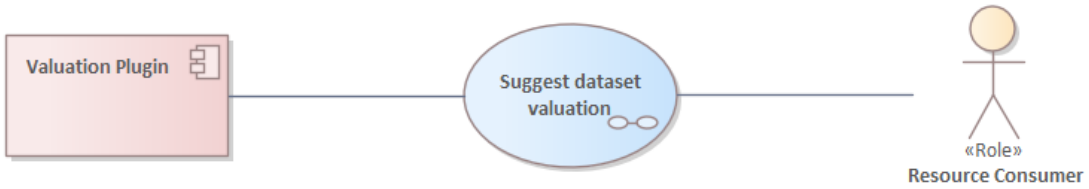

Figure 12: Top-level use case model for the Valuation plugin.

Use Case<<Suggest dataset valuation>>:

- Preconditions/Triggers: Data is available from data provider(s). The user (data marketplace, data aggregator or data consumer) has defined the criteria and methods to valuate data (e.g., improving the accuracy of a model, selecting data with higher spatial or temporal entropy, comparing the similarity to a validation set).
- Postconditions: Data valuation, meaning the relative value of data sources / data from different providers, is calculated, and further information is provided for the output to be analysed.
- Description: This use case calculates Data Valuation (contribution of a selected data to a given purpose) and gives insight on the provided value/contribution. This can be used by data consumers to select which resources fit their purposes (data selection) or by data marketplaces to distribute rewards among data providers that contributed data to a transaction involving a combination of their data.

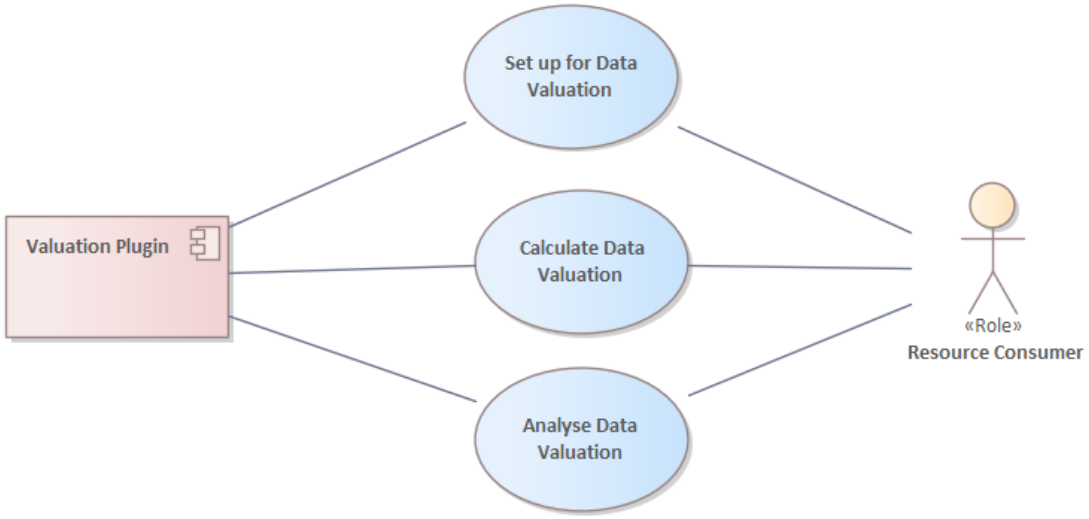

Figure 13: Detail use cases for the Valuation plugin.

Use Case<<Set up for Data Valuation>>:

- Preconditions/Triggers: Data is available from the data providers. Data conforms to the suggested criteria and methods of evaluation (e.g., data can be used by the model or valuation functions provided by data consumers). All methods are

defined, and all the data required to run those methods are available (e.g., a validation set to test the output, if applicable).
- Postconditions: Validity of data is confirmed.
- Description: This use case sets up the data in the right form so that Data Valuation can be calculated.

Use Case <<Calculate Data Valuation>>:

- Preconditions/Triggers: Use case <<Set up for Data Valuation>> is done. A particular data subset is selected for valuation. The user (data marketplace, data aggregator or data buyer) has defined the criteria and methods to valuate data (e.g., data can be used by the model or valuation functions provided by data consumers).
- Postcondition: The contribution of the selected data is calculated according to the criteria defined by users.
- Description: Data Valuation is calculated for a selected data subset based on the predefined methods.

Use Case <<Analyse Data Valuation>>:

- Preconditions/Triggers: Use case <<Calculate Data Valuation>> is done.
- Postconditions: Statistics and explanation of the valuation will be provided for data providers to know more information about the reasons why their data was highly valuable or less valuable than others.
- Description: Data Valuation provides insight on what is the value of the data provider, based on the dataset provided. For this functionality to be useful, it must be adapted to the specific use case.

4.2.6 Environmental Impact Optimiser

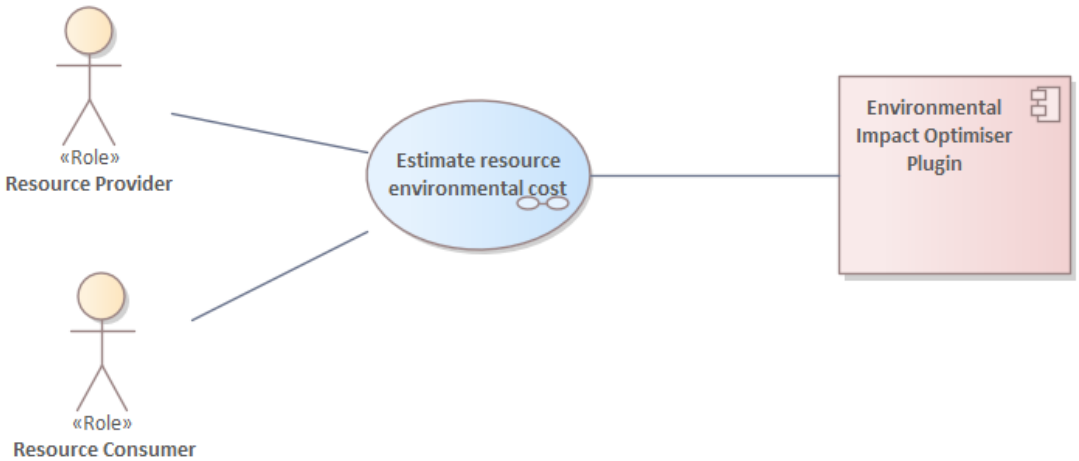

Figure 14: Top-level use case model for the Environmental Impact Optimiser plugin.

Use Case <<Estimate resource environmental cost>>:

- Precondition/Trigger: The vocabulary and resource specification are complete with the plugin-specific requirements and there is access to the infrastructure where datasets are created, stored, modified and processed.
- Postcondition**:** Estimated environmental impact of a dataset and its processing through a data workflow provided.
- Description: The environmental impact optimiser will include efficient AI models, monitoring and visualisation tools to estimate the environmental impact of a data processing workflow, by profiling datasets based on their potential impact and calculating energy metrics such as estimated carbon footprint.

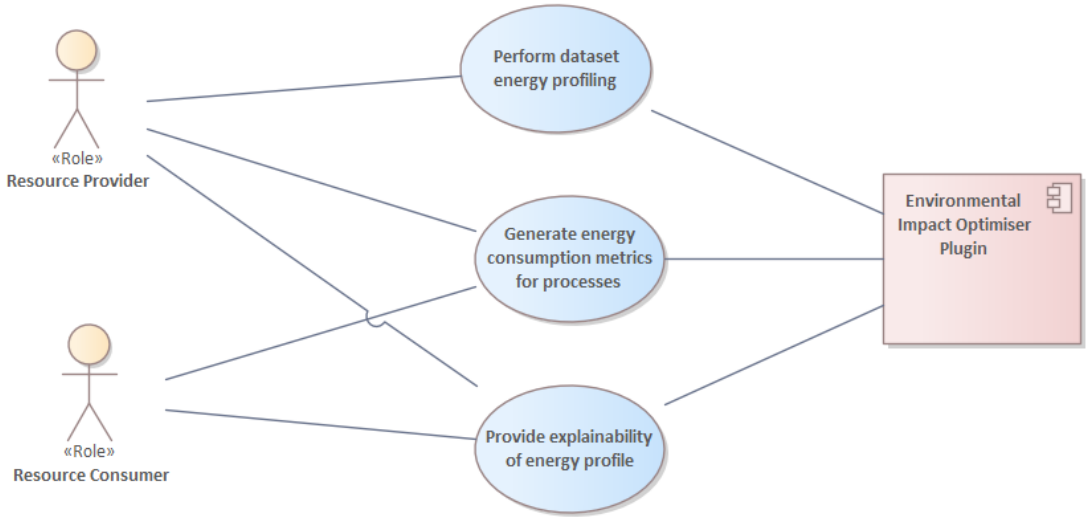

Figure 15: Detail use cases for the Environmental Impact Optimiser plugin.

Use Case <<Perform dataset energy profiling>>:

- Precondition/Trigger: Resource definition with UPCAST vocabulary, hardware information, set of standard operations defined.
- Postcondition: The estimated energy efficiency metrics provided, which are aggregated to create the energy profile.
- Description: The environmental impact optimiser will estimate the energy consumption and intensity of a resource based on its metadata and hardware information where the resource is created, stored or processed. This will be used to create an energy profile for each resource.

Use Case <<Generate energy consumption metrics for processes>>:

- Precondition/Trigger: Access to hardware infrastructure, set of standard operations defined.
- Postcondition**:** Actual power consumption of the processes provided.
- Description: The environmental impact optimiser will generate energy consumption metrics of processes applied to the resource in the data processing workflow and aggregate the energy consumption.

Use Case <<Provide explainability of energy profile>>:

- Precondition/Trigger: Energy profile, dataset metadata exist.
- Postcondition**:** The features contributing to the assigned energy profile of the dataset available.
- Description: Using explainable AI (XAI) techniques, the end user will obtain transparent feedback on the algorithm's decision to assign a particular energy profile to a dataset.

4.2.7 Resource Discovery

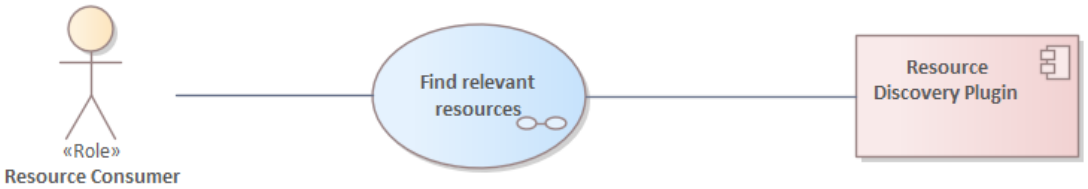

Figure 16: Top-level use case model for the Resource Discovery plugin.

Use Case <<Find relevant resource>>:

- Precondition/Trigger: Resource catalogue includes resources specified in the UPCAST vocabulary. The consumer triggers the use case by asking for resources.
- Postcondition: relevant resources identified.
- Description: a consumer may find relevant resources through browsing for resources, searching for resources and discovery of related or recommended resources.

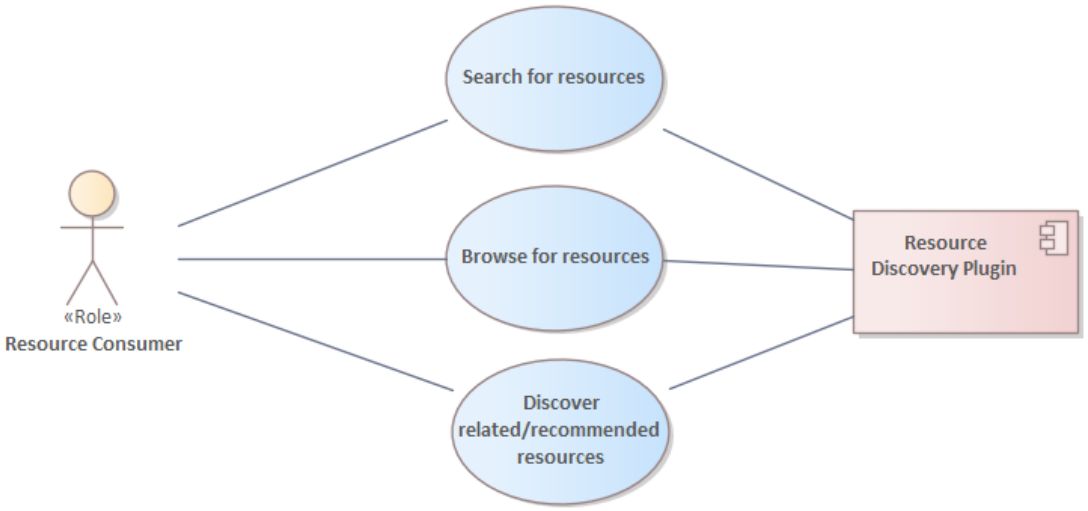

Figure 17: Detail use cases for the Resource Discovery plugin.

Use Case <<Search for resources>>:

- Precondition/Trigger: Resource catalogue contains resources specified by the UPCAST vocabulary. The consumer triggers the action by sending a search request to the plugin.
- Postcondition: The search is logged.
- Description: The consumer initiates a search by inputting a search text. Upon receiving the search request, the system triggers a comprehensive search within the specified resources. As a result, the user is presented with a well-organised and relevant list of resources that matches their search criteria.

Use Case <<Browse for resources>>:

- Precondition/Trigger: Resource catalogue includes resources specified in the UPCAST vocabulary. The consumer triggers the action by sending a request to retrieve a list of resources.
- Postcondition: Resource found/not found.
- Description: The consumer is provided with a curated list of resources through a sophisticated faceted search request. The facets presented are tailored to the underlying data model, offering the user an intuitive and efficient way to navigate and explore the available resources.

Use Case <<Discover related/recommended resources>>:

- Precondition/Trigger: Resource catalogue includes resources specified in the UPCAST vocabulary. A timer-based or an addition of a new dataset trigger the creation/update of a relevant resources graph for each resource. The consumer triggers the retrieval of resources with a request.
- Postcondition**:** Linked resources graph is created/updated.
- Description: The consumer initiates a discovery process by submitting an input query or a resource. In response, a comprehensive list of highly relevant resources is generated. To ensure up-to-date and dynamic results, the relevant resources graph is continuously updated as new datasets arrive. This ensures that users have access to the most current and valuable information available.

4.2.8 Negotiation and Contracting

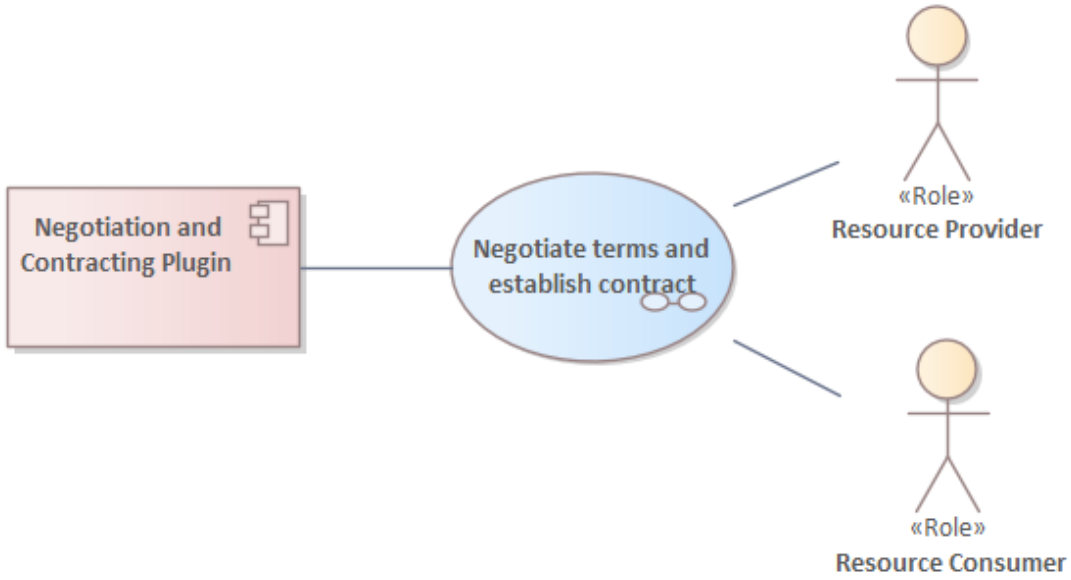

Figure 18: Top-level use case model for the Negotiation and Contracting plugin.

Use Case <<Negotiate terms and establish contract>>:

- Precondition/Trigger: RC has defined the DPW including resources owned by third parties, RP has specified a resource and the corresponding access and usage constraints, RP and RC have been matched for negotiation and agreement, RP and RC have defined negotiation terms that consist of 1) sets of alternative possible values for each field that they are willing to negotiate, and 2) rules that describe the relationships between values across fields, GUIs to both sides are provided.
- Postcondition: Negotiation outcome available, Machine-processable contract available, Natural language contract available.
- Description: Once RP and RC have been matched for negotiation and agreement, the plugin verifies the DPW against the RP constraints, RC intentions, legal constraints and organisation-specific policies, pricing and environmental impact constraints; in case no conflict has been identified (Use Case <<Identify conflict between RP constraints and RC intentions>>, also conflicts related to environmental impact and pricing), an agreement is automatically reached. Otherwise, the system will highlight the conflicts, try to find a (set of) optimal solutions, which will then be sent back to the RC. Following this, a negotiation process will be initiated consisting in a sequence of counter-offers going back and forth between RP and RC. The RC may choose one of the alternatives presented by the system, manually edit the terms, or ask the system for a new alternative. This counter-offer can then be accepted or changed by the RP. If the RP accepts the counter-offer, an agreement is reached and the system can proceed to contracting. Otherwise, the RP must provide a counter-offer, taking the role of the RC in the previous step. This back and forth is repeated until the RP accepts an offer. Ultimately, RP has the final say on whether the negotiation goes through by either accepting, rejecting or sending another counter-offer. For this purpose, RP defines through the negotiation and contracting plugin UI the

negotiation range for each statement in the resource specification, while RC may also fine-tune the DPW specification in order to reflect their own negotiation ranges. The case concludes with the generation of machine-readable and natural language contracts, in case RP and RC have reached an agreement. In any case the negotiation outcome is presented to both parties.

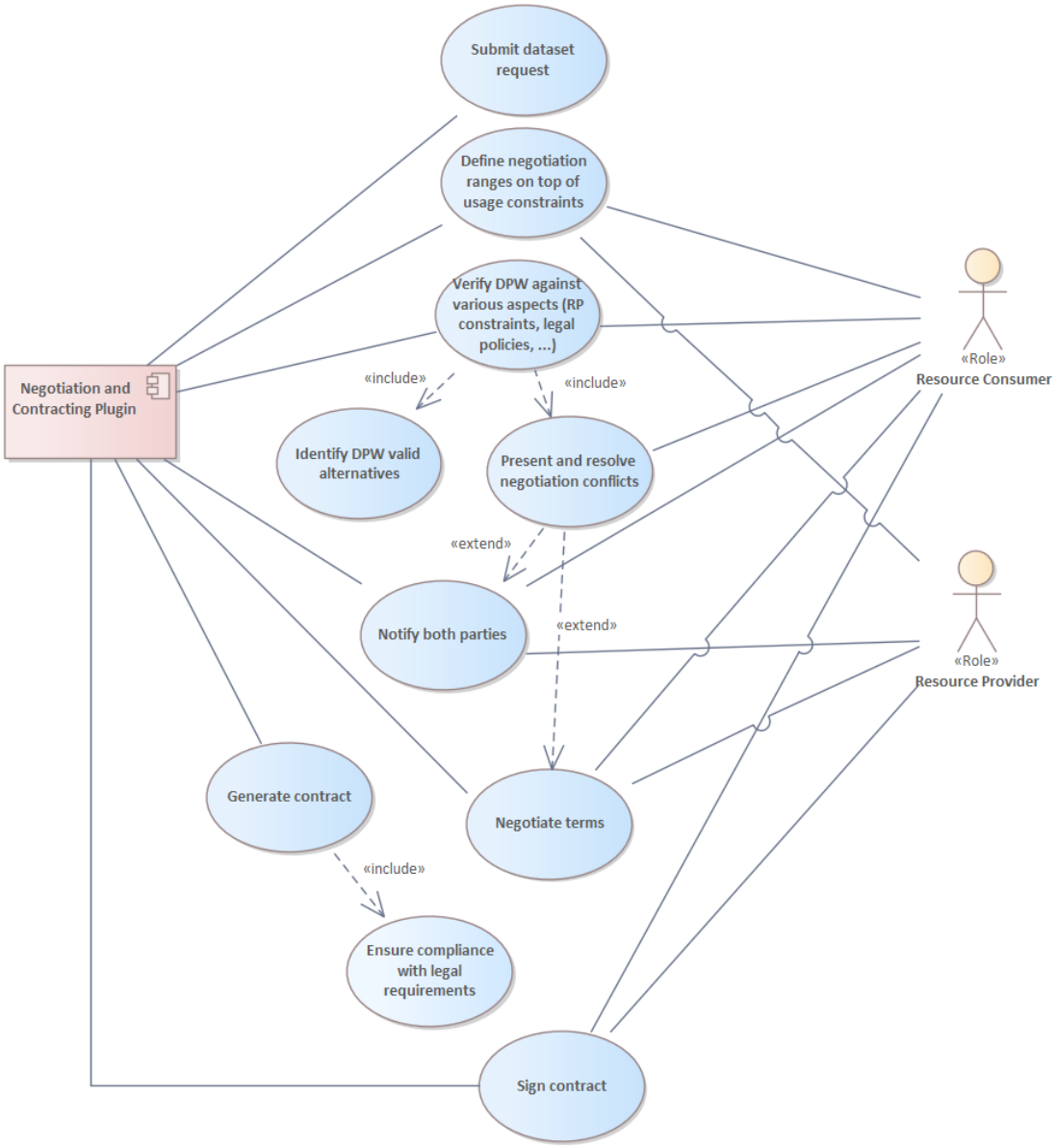

Figure 19: Detail use cases for the Negotiation and Contracting plugin.

Use Case <<Submit dataset request>>:

- Precondition/Trigger: RC has defined the DPW including resources owned by third parties, RP has specified a resource and the corresponding access and usage constraints, RP and RC have been matched for negotiation and agreement, GUIs to both sides are provided.
- Postcondition: RP informed that their dataset is included in a DPW specified by a RC.
- Description: Once RP and RC have been matched for negotiation and agreement, RC submits a request for the matching resource to the RP.

Use Case <<Define negotiation ranges on top of usage constraints>>:

- Precondition/Trigger: RC has defined the DPW including resources owned by third parties, RP has specified a resource and the corresponding access and usage constraints, RP and RC have been matched for negotiation and agreement, GUIs to both sides are provided.
- Postcondition: DPW updated with new RC and RP constraints.
- Description: RP defines through the negotiation and contracting plugin UI the negotiation range for each statement in the resource specification, while RC may also fine-tune the DPW specification in order to reflect their own negotiation ranges. Users may define that certain parts of their contracts are non-negotiable. Both RP and RC may define negotiation terms that consist of 1) sets of alternative possible values for each field that they are willing to negotiate, and 2) rules that describe the relationships between values across fields (e.g., set a price for general use versus a lower price if only for research purposes).

Use Case <<Verify DPW against various aspects>>:

- Precondition/Trigger: RC has defined the DPW including resources owned by third parties, RP has specified a resource and the corresponding access and usage constraints, RP and RC have been matched for negotiation and agreement, GUIs to both sides are provided.
- Postcondition: Negotiation conflicts identified and presented, valid DPW alternatives identified and presented.
- Description: Once RP and RC have been matched for negotiation and agreement, RC verifies the DPW against the RP constraints, RC intentions, legal constraints and organisation-specific policies, pricing and environmental impact constraints. DPW verification results in identification of conflicts and suggestions for conflict resolution reflected in valid DPW alternatives.

Use Case <<Identify DPW valid alternatives>>:

- Precondition/Trigger: RC has verified DPW.
- Postcondition: Valid DPW alternatives identified.
- Description: DPW verification will come up with valid DPW alternatives if any. These may include alternative processing purposes compliant with the RC constraints, alternative actors authorised to perform the DPW tasks, access to and usage of a subset or specific parts of a resource, access to and usage of alternative types of resources with respect to the ones originally included in the DPW.

Use Case <<Notify both parties>>:

- Precondition/Trigger: Negotiation process in progress.
- Postcondition: RP notified about negotiation conflicts and updated RC terms, RC notified about negotiation conflicts and updated RP terms, Both parties visually informed about the negotiation outcome.
- Description: Both parties get automatically informed every time the other party has updated its terms, so as to appropriately proceed to further steps. Both

parties are informed about the negotiation outcome with all appropriate visualisations.

Use Case <<Present and resolve negotiation conflicts>>:

- Precondition/Trigger: RC has verified DPW.
- Postcondition: Negotiation conflicts presented and resolved.
- Description: Possible conflicts found will be presented to the user, as well as valid DPW alternatives resolving the conflicts, so that the user may in turn resolve the conflicts either manually by editing the DPW or automatically by adopting any of the valid DPW alternatives.

Use Case <<Negotiate terms>>:

- Precondition/Trigger: Negotiation process in progress.
- Postcondition: Accepted/rejected terms or counter-offer.
- Description: Users may accept, reject, or continue negotiations.

Use Case <<Generate contract>>:

- Precondition/Trigger: Negotiation process has successfully concluded.
- Postcondition: Negotiation outcome available, Machine-processable contract available, Natural language contract available.
- Description: Once negotiation has concluded with terms accepted by both parties, the machine-readable and the natural language (with boilerplate text) contracts are generated.

Use Case <<Ensure compliance with legal requirements>>:

- Precondition/Trigger: Negotiation process has successfully concluded, Machine-processable contract available, Automatically generated natural language contract available.
- Postcondition: Natural language contract assessed and available.
- Description: Once negotiation has concluded with terms accepted by both parties and the natural language (with boilerplate text) contract is generated, the latter is assessed by DPOs and/or legal services of the involved entities; essentially, this compliance assessment constitutes a human task complementing the automatic DPW verification and transformation in order to ensure compliance with legal requirements.

Use Case <<Sign contract>>:

- Precondition/Trigger: Negotiation process has successfully concluded, Contract has been generated.
- Postcondition: Signed contract by both parties.
- Description: The machine-readable and the natural language (with boilerplate text) contracts are communicated to both parties which proceed to signing the contract.

4.2.9 Integration and Exchange

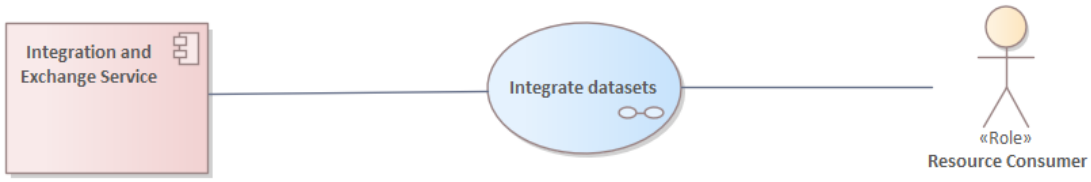

Figure 20: Top-level use case model for the Integration and Exchange plugin.

Use Case <<Integrate datasets >>:

- Precondition/Trigger: There is a DPW defined that includes data integration/exchange as one of its components.
- Postcondition**:** User has access to an integrated solution as defined in the DPW.
- Description: User needs data that can be found in structured sources (this may be in local sources or remote sources provided by a data provider). The user then needs to integrate data as part of their DPW, which usually means answering queries in a structured query language (such as SQL or SPARQL). There are at least two approaches to answering these queries, each of which have different advantages and disadvantages. This use-case is to provide users with systems and algorithms that allow them to answer these queries.

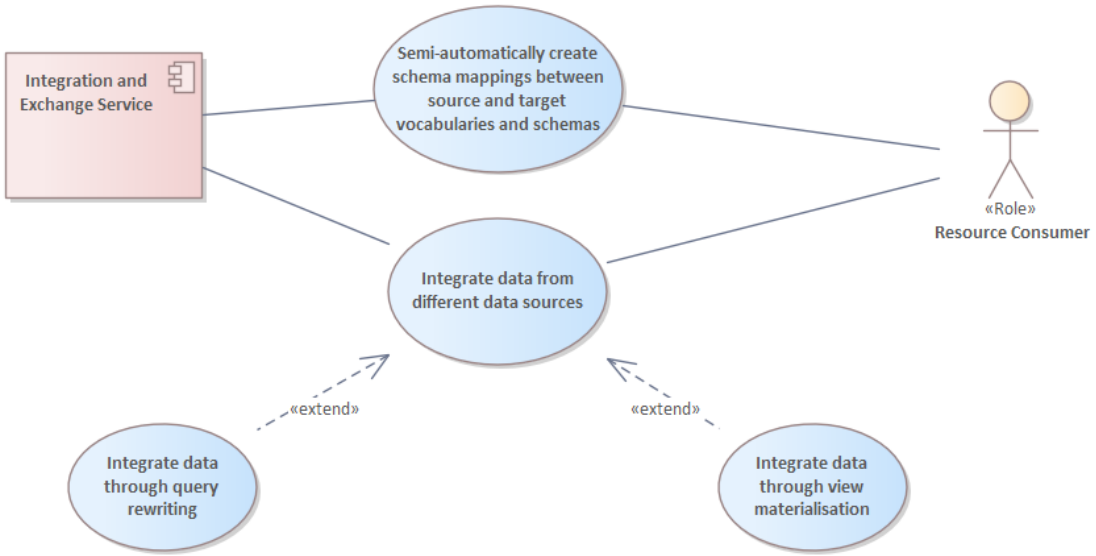

Figure 21: Detail use cases for the Integration and Exchange plugin.

Use Case <<Semi-automatically create schema mappings between source and target vocabularies>>:

- Precondition/Trigger: Data consumer specifies a source and a target. The source and target vocabularies and schemas exist.
- Postcondition: Consumer is given a schema mapping between source and target.
- Description: The plugin takes the resource specification of the source, the resource specification of the target as defined by the data consumer, and

automatically computes a (partial) mapping between source and target. This is semi-automatic because the plugin will also provide an intuitive graphical interface to allow the data consumer to define mappings in a more fine-grained manner.

Use Case <<Integrate data from different data sources>>:

- Precondition/Trigger: Data consumer specifies a source(s) to integrate and provides source to target mapping.
- Postcondition: Data Consumer is given access to integrated solution.
- Description: There are two distinct approaches to data integration/exchange: forward chaining to infer new data (and construct a warehouse), and backward chaining, where we rewrite queries in order to directly query data from each source. The data consumer will be able to choose the approach that better suits their needs, perhaps with some guidance from the plugin.

Use Case <<Integrate data through view materialisation>>:

- Precondition/Trigger: A DPW contains integration as one of its steps, data consumer has specified sources to integrate and source to target mappings.
- Postcondition: New data is generated from the source, and the consumer is given access to this data.
- Description: This use case generates new data from existing data according to constraints defined in mappings. This infers new data from existing one, which is then used to populate a data warehouse (view materialisation).

Use case <<Integrate data through query rewriting>>:

- Precondition/Trigger: A DPW contains integration as one of its steps, data consumer has specified sources to integrate and source to target mappings.
- Postcondition: Data is queried directly from the sources, and the consumer is given access to this data.
- Description: This use case rewrites queries (usually in a structured language such as SQL or SPARQL) to extract the data directly from existing sources.

4.2.10 Secure Data Delivery

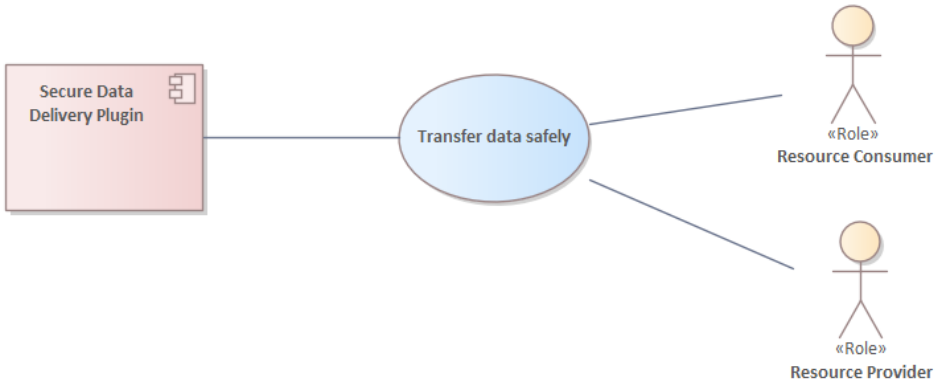

Figure 22: Top-level use case model for the Secure Data Delivery plugin.

Use Case <<Transfer data safely>>:

- Precondition/Trigger: Agreement for data sharing established. Triggered by Resource Consumer using the Data Processing Workflow plugin.
- Postcondition: Data has been transferred according to the agreement in a secure manner.
- Description: The plugin delivers data securely with secure data transfer protocols and user-friendly interfaces that facilitate efficient data exchange.

4.2.11 Monitoring

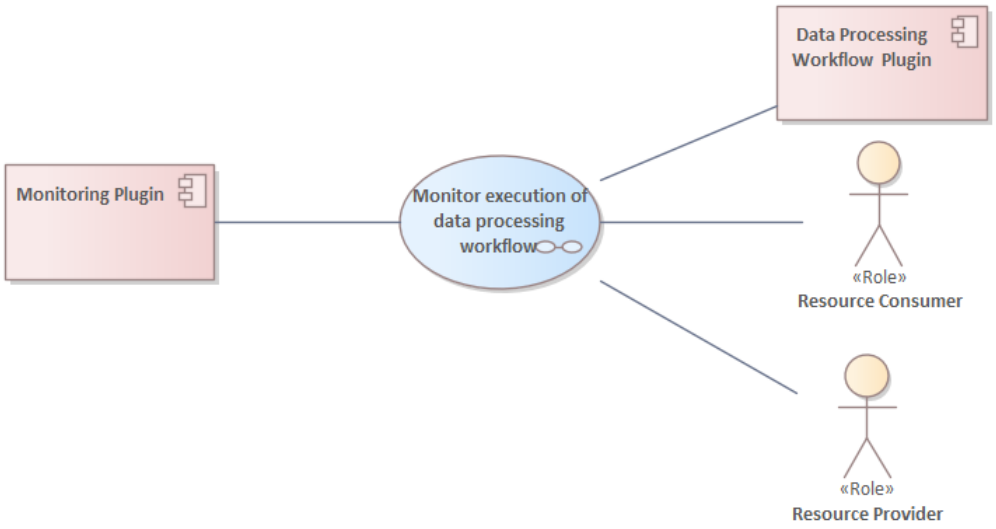

Figure 23: Top-level use case model for the Monitoring plugin.

Use Case <<Monitoring Data Processing Workflow[7]>>:

- Precondition/Trigger: Dataset availability
- Postcondition: Persistence of all log monitoring data
- Description: Logging and persistence of data that have been collected through the data processing workflow. Monitoring data that relate to all steps of the involved UPCAST operations, including dataset specification, annotation with usage and access policies, advertisement, discovery, negotiation, execution, are logged.

[7] Although the Monitoring plugin is mainly used for monitoring of the execution of a data processing workflow by a Resource Consumer, the plugin can also be used for monitoring a dataset execution by a Resource Provider.

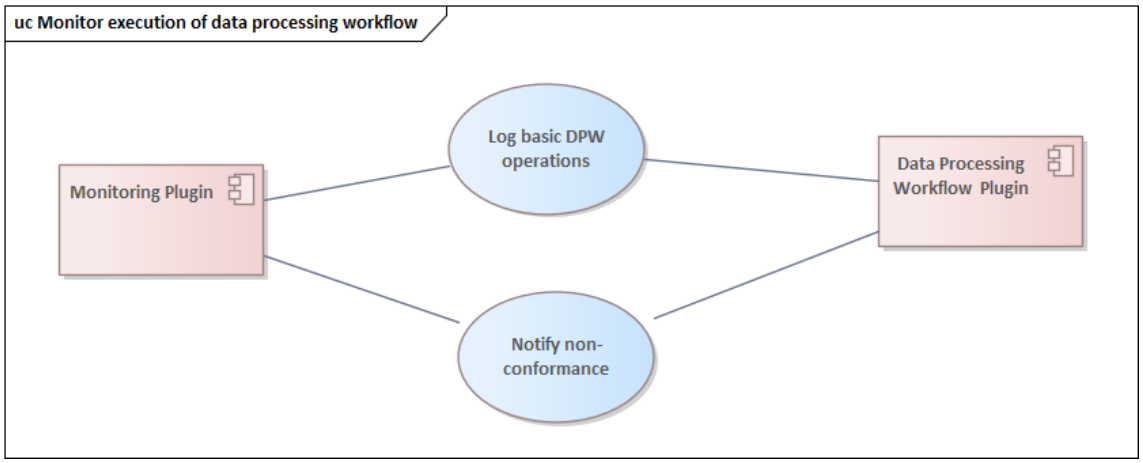

Figure 24: Detail use cases for the Monitoring plugin.

Use Case <<Log basic DPW operations >>:

- Precondition/Trigger: A DPW operation has been carried out.
- Postcondition: A permanent record of the DPW operation has been completed.
- Description: Every event that is generated after each DPW operation is permanently logged in a persistency store.

Use Case <<Notify non-conformance >>:

- Precondition/Trigger: Violation of access or usage policies.
- Postcondition: Alert notification generated.
- Description: An alert notification is generated if during execution of the data processing workflow any access or usage rule is violated based on the collected monitoring data and the policies that have been agreed between the dataset provider and the consumer.

4.2.12 Federated Machine Learning

Federated Machine Learning is a special type of data processing. Three roles are involved with specific focus in this context:

- **Resource Provider (RP):** An entity that offers machine learning models, datasets, or computing resources to be utilised within the federated machine learning process.
- **Resource Consumer (RC):** An entity that seeks to utilize resources provided by Resource Providers to improve its local machine learning model.
- **Data Marketplace Administrator (DMA):** An administrator responsible for managing the Data Marketplace, including onboarding Resource Providers and Consumers, ensuring compliance, and facilitating transactions.

In order to provide policy-aware federated machine learning, this plugin consists of two components that utilises the functionality of the Privacy and Usage Control plugin:

- **Differential Privacy Component (DPE):** A component responsible for applying differential privacy techniques to data shared within the federated learning process, ensuring privacy-preserving data aggregation.
- **Policy Management Component (PMC):** A component that enforces data sharing policies and access controls within the federated learning process.

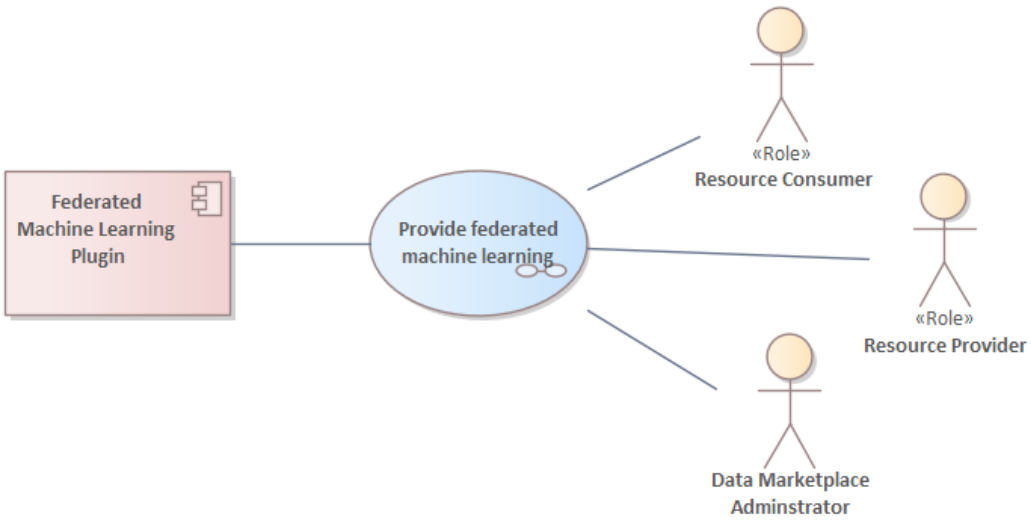

Figure 25: Top-level use case model for the Federated Machine Learning plugin.

Use Case <<Provide federated machine learning>>:

- Precondition/Trigger: User (in the role of Resource Consumer) wants to utilise federated machine learning.
- The Resource Provider has made resources (e.g., model weights, data partitions) available.
- Postcondition: Federated Machine Learning functionalities are provided to user. Policy enforcement is ensured in the process.
- Description: This plugin allows for Data Marketplace Administrator to configure the plugin to make it ready for use, and for Machine Learning users to train, deploy, evaluate and update federated Machine Learning models.

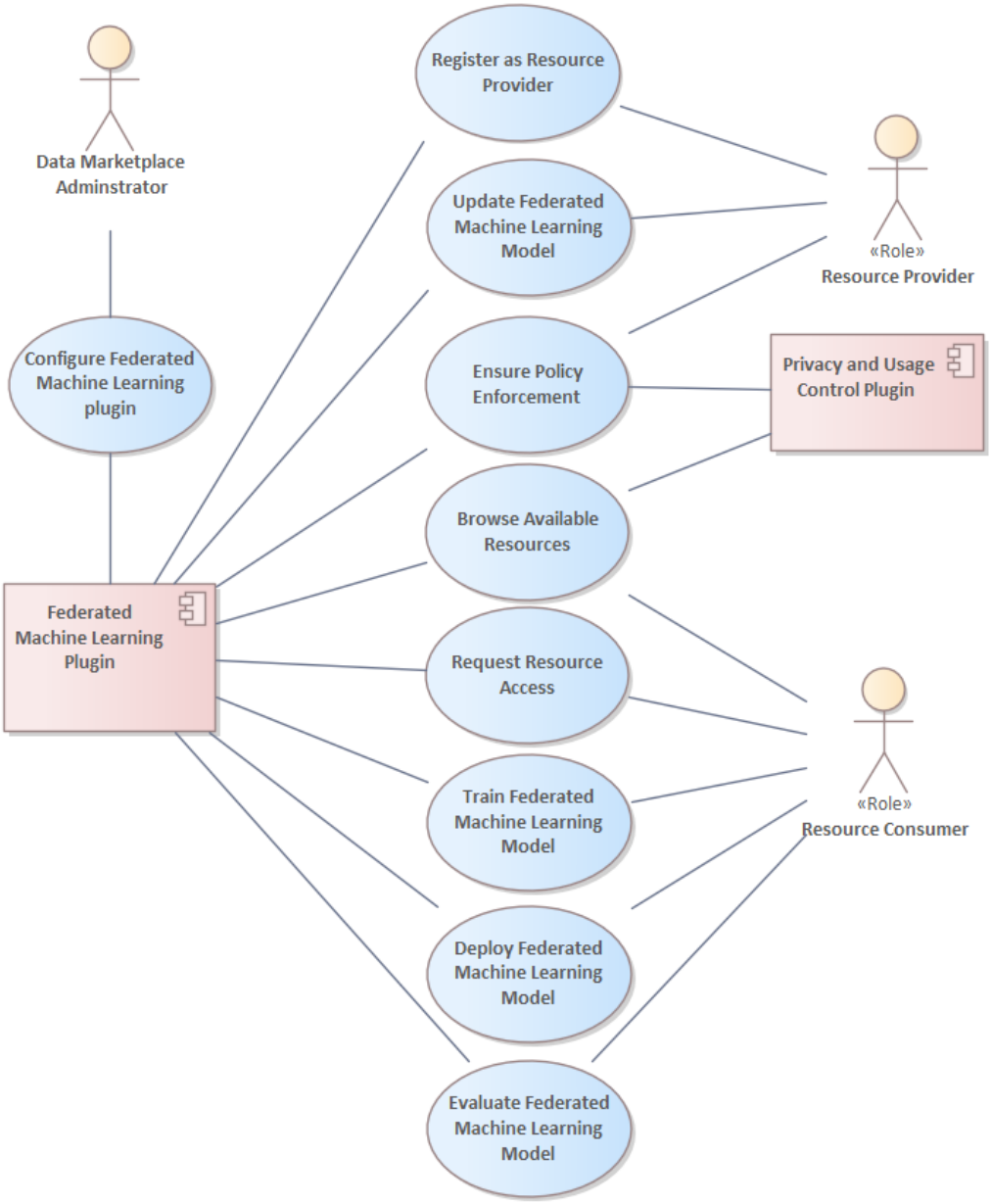

Figure 26: Detailed use cases for the Federated Machine Learning plugin.

Use Case << Register as Resource Provider>>:

- Precondition/Trigger: The Data Marketplace is operational.
- Postconditions:
    - The Resource Provider's profile is created on the Data Marketplace.
    - Resource providers register their datasets on the data marketplace and specify their data usage policies, including restrictions, consent requirements, and the level of data sharing allowed.
    - Policies are encoded in a machine-readable format including granular access controls.

- Description: The Resource Provider registers on the Data Marketplace, providing details about their available resources.

Use Case <<Ensure Policy Enforcement>>:

- Precondition/Trigger:
    - The data marketplace is operational and accessible.
    - Resource provider have uploaded/configured datasets to the marketplace.
    - Data usage policies have been defined and associated with datasets.
- Postconditions:
    - Data access and usage are compliant with the predefined data usage policies.
    - The data marketplace enforces data usage policies during every step of the federated learning process.
    - Policies are enforced during data access, client selection, model distribution, local training, model updates, aggregation, and model deployment.
    - Violations of data usage policies are detected and appropriately handled.
    - Auditing and monitoring logs are maintained for compliance verification.
- Description: This outlines the interactions and processes involved in enforcing data usage policies within a policy-aware federated machine learning plugin operating within a data marketplace. The goal is to ensure that data sharing and processing adhere to predefined policies to maintain data privacy, security, and regulatory compliance.

Use Case << Browse Available Resources>>:

- Precondition/Trigger:
    - The Data Marketplace is operational.
- Postconditions:
    - The Resource Consumer has a list of available resources.
- Description: The Resource Consumer searches and browses available machine learning resources on the Data Marketplace

Use Case << Request Resource Access>>:

- Precondition/Trigger:
    - The Resource Consumer has an account on the Data Marketplace.
    - Resources matching their needs are available.

    - Resource Consumer raise a request for resource access.
- Postconditions:
    - A request is sent to the selected Resource Provider.
- Description: The Resource Consumer requests access to specific machine learning resources provided by a Resource Provider.

Use Case <<Configure Federated Machine Learning Plugin>>:

- Precondition/Trigger: The Federated ML plugin is installed and integrated into the machine learning environment. Data sources or devices that will participate in federated learning are registered.
- Postcondition: The Federated Machine Learning plugin is ready for use with the specified configuration.
- Description: Administrators configure the Federated Machine Learning plugin, specifying parameters, participant devices, and security settings.

Use Case <<Train Federated Machine Learning Model>>:

- Precondition/Trigger: The federated machine learning model is configured. Input data sources have compatible data.
- Postcondition**:** A federated model is trained, and the aggregated data model is available for further evaluation.
- Description: Users initiate federated model training, distribute model update to participants devices and aggregate results.

Use Case <<Deploy Federated Machine Learning Model>>:

- Precondition/Trigger: A trained federated model is available. The production environment is prepared to receive and serve the model.
- Postcondition: The federated model is deployed and serving predictions in the production environment.
- Description: Administrators deploy the trained federated model to the production environment, ensuring it is accessible for inference.

Use Case <<Evaluate Federated Machine Learning Model>>:

- Precondition/Trigger: A trained federated model is available. A test dataset is prepared for evaluation.
- Postcondition: Evaluation metrics (e.g., accuracy, F1-score) are calculated, and the model's performance is documented.
- Description: Data scientists and evaluators assess the performance of the federated model using the test dataset.

Use Case <<Update Machine Learning Model>>:

- Precondition/Trigger: A trained federated model exists. New data and trained local models are available.
- Postcondition: The federated model incorporates updates from participating data sources, potentially improving model performance without exposing raw data.
- Description: Resource Provider initiate model updates, securely transmitting their local model updates to improve the federated model's performance.

4.3 Environment Systems Model

Figure 27 illustrates how the UPCAST plugins interface with its environment. It shows how Resource Consumers and Resource Providers will use the plugins (represented by their top-level use cases). The Resource Discovery plugin will utilise the resource catalogue provided by marketplaces or brokers to find relevant resources, while the Pricing plugin will obtain market prices of a resource from marketplaces. The Privacy and Usage Control plugin will check the relevant rules and regulations defined by Regulatory or legal authorities.

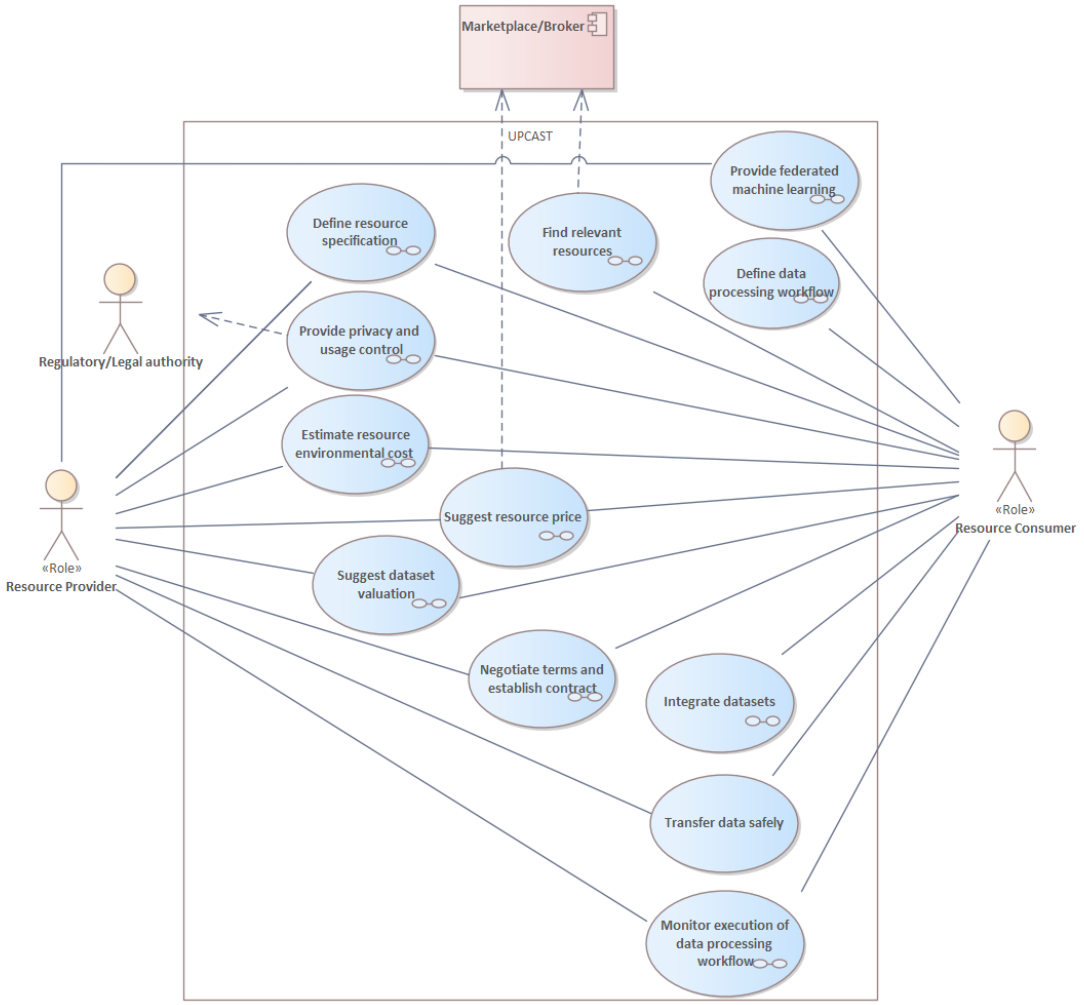

Figure 27: Environment system model.

4.4 Plugin Requirements

The plugin requirements defined in D1.1 have been harmonised and prioritised as a result of elaboration and further work in D1.2. The priorities of some plugin requirements have been adjusted in this process. Other changes to the requirements as documented in D1.1 are summarised below.

- Added requirements for the Federated Machine Learning plugin as this is a new defined plugin (REQ_FL_F_1 to REQ_FL_F_12 and REQ_FL_NF_1 to REQ_FL_NF_7).
- Moved REQ_RD_F_10 (profile generation) and REQ_RD_F_11 (Sample/Preview generation) from the Resource Discovery plugin to the Resource Specification plugin (REQ_RS_F_9 and REQ_RS_F_10).
- Moved REQ_EE_F_8 (The energy profile should be used as a feature to influence its price) from the Environmental Impact Optimiser plugin to the Pricing plugin as REQ_PR_F_11).

The following tables (Table 1 to Table 12) show the updated plugin requirements, one table per plugin. The requirement ID has the following convention: REQ_<<plugin_abbreviation>>_<<requirement_type>>_<<requirement_number>>. For <<requirement_type>>, F stands for functional requirements, NF stands for non-functional requirements.

In addition, system-wide requirements (Table 13) are presented at the end of this section. These system-wide requirements are relevant for the UPCAST ecosystem as a whole.

Table 1: Requirements for the Resource Specification plugin.

ID	Requirement	Priority
REQ_RS_F_1	User must be able to describe a dataset using the UPCAST vocabulary	Must have
REQ_RS_F_2	User must be able to describe a data processing operation, either implemented as containerised software or including human processing, using the UPCAST vocabulary	Must have
REQ_RS_F_3	User must be able to validate that the resource description generated with the plugin includes all fields required by the specification of the UPCAST vocabulary.	Must have
REQ_RS_F_4	User must be able to upload a Domain-Specific vocabulary for describing resources, and use it to add further descriptions in the same way as the UPCAST vocabulary (RS_F_1).	Must have
REQ_RS_F_5	User can import attributes output from the UPCAST plugins “Privacy and Usage Control”, “Valuation and Pricing” and “Environmental Impact” to augment the description of a resource	Must have

REQ_RS_F_6	Define data operation custom parameter range	Could have
REQ_RS_F_7	Manage catalogue of own resources	Should have
REQ_RS_F_8	User must be able to visualise the description of a dataset both in RDF form and in a suitable graphical form	Must have
REQ_RS_F_9	Given a dataset the plugin must be able to return a profile of the dataset	Must have
REQ_RS_F_10	The plugin should provide functionality to provide users with a reduced sample of the dataset	Could have
REQ_RS_NF_1	The tool Resource Specification plugin must be usable by people with only a basic understanding of ontologies and vocabularies	Must have
REQ_RS_NF_2	UPCAST vocabulary should align as much as possible with existing vocabularies from Data Space community: IDSA Information model and DCAT	Must have
REQ_RS_NF_3	Manage a joined vocabulary of at least 100 classes and 100 properties	Must have
REQ_RS_NF_4	Assuming knowledge of the values of the descriptive properties, and excluding the time generating output from the other UPCAST plugins, user must be able to complete the description of a resource in 20 minutes or less.	Must have

Table 2: Requirements for the Data Processing Workflow plugin.

ID	**Requirement**	**Priority**
REQ_DPW_F_1	The definition of all entities involved in DPWs should be supported at the required detail level, based on the semantic foundation provided.	Must have
REQ_DPW_F_2	Usage preferences should be able to be suitably formalized, providing adequate expressiveness	Must have
REQ_DPW_F_3	Access and usage constraints should be able to be expressed in the process models in order to consistently reflect compliant execution	Must have
REQ_DPW_F_4	The definition of conditional execution of tasks and conditional data flow depending on context, purpose or intra-workflow dependencies should be supported	Should have
REQ_DPW_F_5	The definition of the entity that performs a processing step, but also the entity that initiates a DPW, should be supported	Must have
REQ_DPW_F_6	The definition of the operations performed through a data processing step should be supported	Must have

REQ_DPW_F_7	The definition of the data asset which is accessed or upon which an operation is performed should be supported	Must have
REQ_DPW_F_8	The definition of the data exchanged between processing steps should be supported	Must have
REQ_DPW_F_9	DPW models should be able to accurately represent cases of cross-domain data sharing at various levels (organisations, states, regulatory domains, etc.)	Must have
REQ_DPW_F_10	A GUI shall be made available to model DPWs.	Must have
REQ_DPW_F_11	The DPW tool should support the definition of data workflows comprising atomic actions and decisions through a GUI	Should have
REQ_DPW_F_12	The DPW tool should be able to execute a workflow	Could have
REQ_DPW_F_13	The DPW should be seamlessly integrated with a monitoring service to collect information on the progress of the execution of a workflow	Should have
REQ_DPW_F_14	The DPW should provide a control interface to allow a user to intervene (stop, pause, resume, inspect) on the execution of a workflow.	Could have
REQ_DPW_NF_1	The DPW should support the secure execution of a workflow	Could have

Table 3: Requirements for the Privacy and Usage Control plugin.

ID	**Requirement**	**Priority**
REQ_PUC_F_1	The plugin should provide the means for system operation in accordance to access and usage control policies. Moreover, policy based access control should be inline with Attribute Based Access Control (ABAC) paradigm	Should have
REQ_PUC_F_2	Access and usage control policies should be fine-grained, following hierarchical organisation of related entities; that is, policies should be defined in different granularities of the underlying concepts, particularly the resources to be protected.	Must have
REQ_PUC_F_3	All applicable parameters must be taken into account, such as roles, attributes, contextual parameters, the purpose under which the underlying action should be executed, prior actions (history), etc.	Could have
REQ_PUC_F_4	Access and usage control policies must be machine-readable, so that they can be processed by a policy engine	Must have

REQ_PUC_F_5	The plugin must incorporate a policy engine, able to make decisions as regards access and usage of data and other resources	Must have
REQ_PUC_F_6	Policies must support the definition of complementary / forbidden actions that must/ must not take place upon and/or prior to their enforcement	Must have
REQ_PUC_F_7	The plugin should provide for implicit definition / propagation of policies, i.e., rules defined for high-level concepts should be propagated to more specific ones without the need to add new specific rules	Could have
REQ_PUC_F_8	The plugin should provide for advanced decision making. Policy decision point should provide for request transformation where possible; instead of simply allowing or denying an incoming access/usage request, it should provide for request transformation (e.g., allow access to/usage of parts of the requested data) and/or prescribe/forbid the execution of subsequent actions	Should have
REQ_PUC_F_9	The plugin should provide for advanced conflict resolution and rules merging, i.e., mechanisms for the elimination of deprecated policies (i.e., overridden by other policies), as well as for conflict resolution between data provider and data consumer access and usage constraints (consistent enforcement of data provider constraints, without compromising core policies)	Should have
REQ_PUC_F_10	The plugin should provide a graphical user interface for the specification of access and usage control rules (Data Consumer side)	Should have
REQ_PUC_F_11	The plugin should provide the ability to data subjects/providers to define their data usage constraints in a machine-readable form through a graphical user interface	Should have
REQ_PUC_F_12	By means of access and usage control rules, it should be possible to explicitly determine what is the necessary accuracy (detail level) of information that is necessary for a certain purpose and under certain conditions.	Should have
REQ_PUC_F_13	Access and usage control should make possible that a data structure can be collected or processed in parts, based on certain criteria; that is, there must be selective handling of different parts of such structure	Could have
REQ_PUC_F_14	The plugin should enable decisions about data collection, processing, storage and communication to be made on the basis of the underlying purpose	Must have
REQ_PUC_NF_1	Policies must support the specification of (i) permissions, i.e., actions that are allowed to take place; (ii) prohibitions, i.e., actions that are prohibited to take place; (iii) obligations, i.e., actions that must take place	Should have

REQ_PUC_NF_2	The plugin should provide for ensuring the compliance with data protection legislation; for instance, the plugin should allow the definition of policies containing any applicable GDPR-related rules as adapted to the specific organisation's needs	Must have
REQ_PUC_NF_3	The plugin should foster context-awareness: it should provide the means for semantic definitions of applicable contextual parameters, while decision making on data collection, processing, storage and communication should be able to consider contextual aspects	Could have
REQ_PUC_NF_4	Access and usage control should provide the means for adjusting the detail level of data that are collected, processed, stored and communicated, in a manner as automatic as possible	Should have
REQ_PUC_NF_5	Access and usage control rules should define what data are considered proportional for a certain purpose and under certain conditions	Could have
REQ_PUC_NF_6	The plugin should provide for semantic information management: the underlying information model should (i) support a variety of concepts, including data types, purposes, roles, operations, attributes, organisations, context, etc.; (ii) provide coherent semantic definitions of the underlying concepts; (iii) support the hierarchical organisation of concepts, including different semantics, such as generalisation / particularisation, inclusion, etc.	Should have
REQ_PUC_NF_7	The plugin must provide mechanisms for specifying the compatibility between collection and processing purposes and provide means for defining prevention rules regarding incompatible purposes	Should have
REQ_PUC_NF_8	The plugin should enable transformation of governance metadata to the underlying machine-readable access and usage control policies and vice versa	Should have

Table 4: Requirements for the Pricing plugin.

ID	**Requirement**	**Priority**
REQ_PR_F_1	The plugin should provide similar products to a data product specified by the users found in real commercial data marketplaces.	Must have
REQ_PR_F_2	The plugin requires the characteristics of data products including metadata fields included in commercial data marketplaces such as their description, categories, units, update rate, geographical and time scope, etc.	Must have
REQ_PR_F_3	The plugin should provide flexible functionality to label data products of a data marketplace using the categories	Should have

	and criteria of a “source” data marketplace based on their descriptions.	
REQ_PR_F_4	The pricing plugin will return a range of prices for a data product specified by the user based on the estimation of different price regressors fitting the price of similar commercial products in data marketplaces, already stored in the plugin database.	Should have
REQ_PR_F_5	Using Explainable AI, the pricing plugin will be able to inform the buyers about the most relevant features when building its price predictions, producing a list of relevant features and a percentage of impact in the prediction of the price.	Should have
REQ_PR_F_6	End users will be able to access this functionality through a REST API	Must have
REQ_PR_F_7	The plugin should record the history of user transactions/interactions for pricing, security, logging, insights and transparency.	Should have
REQ_PR_F_8	The plugin must support an admin user that manages all aspects of the plugin including the database, manipulating data, updating/enriching datasets, training models, interconnections, and permission management	Should have
REQ_PR_F_9	The database and architecture will be centralised with a central server providing all the functionality, data, models, and API responses.	Must have
REQ_PR_F_10	The plugin should incorporate market conditions and marketplace interactions to produce a dynamic price range for datasets.	Should have
REQ_PR_F_11	The energy profile of a dataset should be used as a feature to influence its price	Should have
REQ_PR_NF_1	The plugin must be usable by end users with only domain-specific knowledge to get price ranges and similar products in the database.	Must have
REQ_PR_NF_2	The architecture of the plugin and its various management scripts should be well documented and available/easily accessible to the plugin admin.	Could have
REQ_PR_NF_3	Must be able to support at least 10.000 price references	Should have
REQ_PR_NF_4	Must support data from at least 10 data marketplaces	Should have
REQ_PR_NF_5	The plugin must respond in a few seconds to applications through the REST API	Should have
REQ_PR_NF_6	The pricing plugin must include security mechanisms to prevent abuse and misuse from users based on the activity registered through the REST API	Should have

REQ_PR_NF_7	The plugin needs to be compliant with the EU regulations (such as GDPR, Data Act, AI Act)	Must have
REQ_PR_NF_8	The plugin should be designed in a modular and maintainable way, allowing for flexibility to add and/or update the machine learning models, labels etc, enabling extensibility.	Should have

Table 5: Requirements for the Valuation plugin.

ID	Requirement	Priority
REQ_VAL_F_1	The plugin will provide functions to calculate the relative value of a set of N data sources for a certain ML task given by a model and a valuation function, or for specific valuation functions that use relevant data resources to evaluate as inputs.	Must have
REQ_VAL_F_2	The plugin will be implemented using a Python library that can be integrated with new ML models and valuation functions by users.	Must have
REQ_VAL_F_3	The plugin will provide functions to carry out exact calculations of the Shapley value of data sources, and different approximation algorithms.	Must have
REQ_VAL_F_4	Shapley approximation algorithms will include tunable parameters to balance precision and execution time.	Must have
REQ_VAL_NF_1	The plugin must be usable by ML developers with only domain specific knowledge to get price ranges and similar products in the database.	Must have

Table 6: Requirements for the Environmental Impact Optimiser plugin.

ID	Requirement	Priority
REQ_EE_F_1	The plugin should generate an energy profile of a dataset	Must have
REQ_EE_F_2	The plugin should display relevant energy consumption metrics, in a graphical manner, based on processes applied to the datasets	Must have
REQ_EE_F_3	The plugin should use explainability techniques to explain the factors contributing to the energy consumption	Should have
REQ_EE_F_4	The plugin requires hardware information such as server, platform (physical/cloud) and data centre characteristics	Must have
REQ_EE_F_5	The plugin should continuously improve its energy profiling as the dataset quantity increases, and requires dataset metadata	Should have
REQ_EE_F_6	The plugin should calculate the energy cost of storing and updating a dataset	Must have

REQ_EE_F_7	The plugin should model the energy footprint of atomic operations related to the access of resources	Should have
REQ_EE_NF_1	The plugin must be usable only by internal stakeholders, other plugin developers and relevant data marketplaces	Must have
REQ_EE_NF_2	The plugin should be compatible with different operating systems, environments and platforms, including both physical and cloud infrastructure	Must have
REQ_EE_NF_3	The plugin performance should not degrade significantly when monitoring large or complex dataset processes	Should have
REQ_EE_NF_4	The plugin needs to be compliant with the EU regulations (such as GDPR, AI Act)	Must have
REQ_EE_NF_5	The plugin should be designed in a modular and maintainable way, allowing for easy updates and bug fixes	Could have

Table 7: Requirements for the Resource Discovery plugin.

ID	**Requirement**	**Priority**
REQ_RD_F_1	Users can search datasets with keywords on a dataset catalogue	Must have
REQ_RD_F_2	Users can search datasets with facets on a dataset catalogue	Must have
REQ_RD_F_3	Users must be able to search based on an abstract (incomplete) description of a dataset. This should be equivalent to an OR keyword/facet search using the attributes specified in the abstract description.	Must have
REQ_RD_F_4	Users must be able to search with keywords on top of an existing Data Processing Operation Catalogue.	Must have
REQ_RD_F_5	Users must be able to search with facets on top of an existing Data Processing Operation Catalogue.	Must have
REQ_RD_F_6	Users must be able to search based on an abstract (incomplete) description of a data processing operation. This is equivalent to an OR keyword/facet search using the attributes specified in the abstract description	Must have
REQ_RD_F_7	Given a Data Processing Workflow, users can trigger a search for alternatives for any resource in the workflow	Should have
REQ_RD_F_8	User should be able to connect remotely to multiple catalogues (assuming they are available on a Web server) to browse and search from all of them.	Could have
REQ_RD_F_9	Given a Data Processing Workflow, and a set of resource catalogues, the plugin should be able to recommend resources that could replace or augment the ones in the Data Processing Workflow	Should have

REQ_RD_F_10	Given an owned dataset, the plugin should enable searching for datasets in a catalogue that can augment the input dataset in terms of join/union operations.	Could have
REQ_RD_NF_1	The plugin must be usable by people with only domain specific knowledge	Must have
REQ_RD_NF_2	The plugin must support at least 100.000 resources when used on a local catalogue	Could have
REQ_RD_NF_3	The plugin must support connection to at least 5 remote catalogues	Could have

Table 8: Requirements for the Negotiation and Contracting plugin.

ID	Requirement	Priority
REQ_NE_F_1	Users must be able to make contracts semi-automatically that follow the IDSA standard (with some variance allowed).	Must have
REQ_NE_F_2	System should be able to detect (some) conflicts between offer and request contracts.	Must have
REQ_NE_F_3	Users should be able to negotiate with other parties whenever there is a conflict in their respective contracts.	Must have
REQ_NE_F_4	Users must be able to accept, reject, or continue negotiations.	Must have
REQ_NE_F_5	System should evaluate privacy and usage settings from all parties.	Must have
REQ_NE_F_6	System should evaluate the environmental impact of the relevant datasets.	Must have
REQ_NE_F_7	System should evaluate the pricing of all relevant datasets and inform all parties.	Must have
REQ_NE_F_8	System should provide a visualisation of the result of the negotiation (i.e., agreement, rejection).	Must have
REQ_NE_F_9	Users must be able to generate contracts that contain and use boilerplate text whenever the contract contains clauses that can only be expressed through natural language.	Should have
REQ_NE_F_10	Once an agreement has been reached, the system should draft an agreement contract that can be reviewed by all parties.	Must have
REQ_NE_F_11	Whenever a conflict is detected between contracts, the system must highlight said conflicts so that all parties can review them.	Should have

REQ_NE_F_12	Users may define that certain parts of their contracts are non-negotiable, so the system must immediately reject contracts that conflict with any of these parts.	Should have
REQ_NE_F_13	System must ensure that all negotiations are carried out while respecting and protecting all users' privacy and confidentiality.	Should have
REQ_NE_F_14	Users must be able to edit policies in contracts with the help of a graphical interface.	Must have
REQ_NE_NF_1	Plugin must be usable by people with domain-specific knowledge (not necessary knowledge of how the contracts are processed).	Should have

Table 9: Requirements for the Integration and Exchange plugin.

ID	**Requirement**	**Priority**
REQ_IE_F_1	System must be capable of integrating data by forward chaining (chasing) and backward chaining (query rewriting).	Must have
REQ_IE_F_2	Users must be able to choose whether the data to integrate comes from a local source or a remote source.	Must have
REQ_IE_F_3	Users must be able to define and create schema mappings with help of a graphical interface.	Must have
REQ_IE_F_4	Users must be able to define constraints and functional dependencies for target database.	Must have
REQ_IE_F_5	Users must be able to integrate data that may be structured under different standard formats (e.g., TSV, JSON, RDF).	Should have
REQ_IE_F_6	Users must be able to integrate data from various sources concurrently.	Should have
REQ_IE_F_7	Users must be able to choose whether to integrate data through query rewriting or materialization of sources.	Must have
REQ_IE_F_8	Users must be able to view a preliminary view of the resulting integration.	Could have
REQ_IE_F_9	Users must be able to choose to output their data in their desired format from a list of options.	Should have
REQ_IE_F_10	System should be able to recognise sets of dependencies that allow for more efficient data integration.	Could have
REQ_IE_F_11	Users must be able to execute queries over the integrated data with a standard query language (e.g., SQL or SPARQL).	Must have
REQ_IE_F_12	Users must be able to include a number of remote sources (either databases or APIs).	Must have

REQ_IE_F_13	System should operate with local sources if there is no internet connection available.	Should have
REQ_IE_F_14	System should compile a history of operations done by the user. In addition, it should allow the user to repeat these actions.	Could have
REQ_IE_F_15	System should work independently from other UPCAST plugins.	Should have
REQ_IE_NF_1	The plugin must be usable by people with just some domain specific knowledge.	Should have
REQ_IE_NF_2	The definition and creation of schema mappings must be intuitive.	Must have
REQ_IE_NF_3	Data integration should terminate in a reasonable amount of time.	Must have
REQ_IE_NF_4	System should provide guidance or suggestions when choosing between forward and backward chaining.	Could have
REQ_IE_NF_5	System should warn users when attempting to integrate data under constraints that are not guaranteed to terminate.	Could have
REQ_IE_NF_6	System should have the option to set a timeout for some of its processes (forward chaining in particular).	Should have

Table 10: Requirements for the Secure Data Delivery plugin.

ID	**Requirement**	**Priority**
REQ_SEC_F_1	Data processing must be executed in a secure sandbox environment	Must have
REQ_SEC_F_2	CIA requirements must be guaranteed for available datasets	Must have
REQ_SEC_F_3	Implementation of mechanisms to authenticate and verify the identity of users participating in the marketplace, such as through secure login systems or digital signatures.	Must have
REQ_SEC_F_4	Implementation of access control policies to ensure that users have appropriate access rights and permissions based on their roles within the marketplace.	Must have
REQ_SEC_F_5	Implementation of strong encryption algorithms to protect the confidentiality of data during transmission and storage.	Must have
REQ_SEC_F_6	Implementation of mechanisms to validate the integrity of data to ensure that it has not been tampered with or modified during transit or storage.	Must have
REQ_SEC_F_7	Utilisation of distributed ledger technologies (e.g., blockchain) to maintain an immutable record of	Must have

	transactions and data modifications, ensuring data integrity.	
REQ_SEC_F_8	Employ secure communication protocols (e.g., HTTPS, TLS) for data transmission between participants, protecting against interception or unauthorized access.	Must have
REQ_SEC_F_9	Allow data providers to specify access controls and permissions for their data, ensuring that only authorized entities can access and utilise it.	Must have
REQ_SEC_F_10	Utilisation of smart contracts to enforce agreements, terms and conditions	Must have
REQ_SEC_F_11	Maintenance of comprehensive audit logs of data transactions, access attempts, and modifications for monitoring and forensic analysis.	Must have
REQ_SEC_F_12	Ensuring compliance with relevant data protection regulations (e.g., GDPR) and industry standards.	Must have
REQ_SEC_NF_1	Scalability: The marketplace should be able to handle a growing number of participants, data transactions, and data volumes without compromising security.	Must have
REQ_SEC_NF_2	The marketplace should provide efficient and responsive operations, including data retrieval, data sharing, and authentication processes, to minimise delays and ensure a smooth user experience.	Must have
REQ_SEC_NF_3	Marketplace should implement fault tolerance mechanisms to handle failures in individual nodes or components.	Must have
REQ_SEC_NF_4	The marketplace should be available and accessible to users consistently, with minimal planned or unplanned downtime.	Must have
REQ_SEC_NF_5	Marketplace should provide compatibility and interoperability with different data formats, protocols, and standards to facilitate seamless integration with various data sources and consumers.	Must have
REQ_SEC_NF_6	The system should adhere to strict privacy and data protection regulations, such as GDPR or CCPA, to protect user data and ensure compliance.	Must have

Table 11: Requirements for the Monitoring plugin.

ID	**Requirement**	**Priority**
REQ_MON_F_1	The monitoring plugin must be able to collect data from different sources including access and use of datasets.	Must have
REQ_MON_F_2	The monitoring plugin must be able to store monitoring data for a configurable duration.	Must have

REQ_MON_F_3	The monitoring plugin must be able to use a JSON-based data model for the collected monitored data.	Must have
REQ_MON_F_4	The monitoring plugin must be able to provide visualisations of the collected data.	Must have
REQ_MON_F_5	The monitoring plugin should be able to integrate with external services that may further analyse the collected data.	Should have
REQ_MON_NF_1	The monitoring service must be fault tolerant.	Should have
REQ_MON_NF_2	The monitoring service must be configurable for the type and sources of collected data.	Must have
REQ_MON_NF_3	CIA requirements should be guaranteed for all monitoring data.	Must have

Table 12: Requirements for the Federated Machine Learning plugin.

ID	**Requirement**	**Priority**
REQ_FL_F_1	It must be able to aggregate data from multiple sources in a privacy-preserving manner.	Must have
REQ_FL_F_2	It must be able to train machine learning models on aggregated data.	Must have
REQ_FL_F_3	It must be able to distribute the training process across multiple devices.	Must have
REQ_FL_F_4	It must be able to coordinate the updates to the model parameters.	Must have
REQ_FL_F_5	It must be able to measure the accuracy and performance of the models.	Must have
REQ_FL_F_6	It must be able to compare the results of different models.	Must have
REQ_FL_F_7	It must be able to deploy machine learning models to production.	Must have
REQ_FL_F_8	It must be able to encrypt the data before it is aggregated or shared.	Must have
REQ_FL_F_9	It must use a secure encryption algorithm.	Must have
REQ_FL_F_10	It must keep the encryption keys safe.	Must have
REQ_FL_F_11	It must comply with all applicable privacy laws and regulations.	Must have
REQ_FL_F_12	It must be able to demonstrate that it is protecting the data in a secure and compliant manner.	Must have
REQ_FL_NF_1	It must be able to train and deploy machine learning models in a timely manner.	Must have

REQ_FL_NF_2	It must be able to scale to support a large number of users and devices.	Must have
REQ_FL_NF_3	It must be able to protect the data from unauthorised access, modification, or disclosure.	Must have
REQ_FL_NF_4	It must be interoperable with other federated learning frameworks and platforms.	Must have
REQ_FL_NF_5	It must be easy to maintain and update.	Must have
REQ_FL_NF_6	It must be transparent in its operations, so that users can understand how their data is being used.	Must have
REQ_FL_NF_7	It must protect the privacy of the data, so that users can be confident that their data is not being shared or used without their consent.	Must have

Table 13: System-wide requirements.

ID	Requirement	Priority
REQ_SYS_1	Technical interoperability with regards to support for different communication patterns, message routing and dispatching, etc. must be supported.	Must have
REQ_SYS_2	Semantic interoperability with regards to support for conformance to a common data model/vocabulary must be ensured.	Must have
REQ_SYS_3	Graphical User Interfaces (GUIs) must be provided for all plugins	Must have
REQ_SYS_4	Support for multilingualism should be supported by the plugins	Should have
REQ_SYS_F_5	System components should be developed in a modular, extensible and adaptable way.	Must have

4.5 System Information Model and Vocabularies

UPCAST Vocabulary is divided in 6 logical units or “Facets”, based on the concerns of the plugins most related to each facet. We reused classes from established vocabularies DCAT and IDS Information model, work from previous related EU projects smashHit and BPR4GDPR. In the following, we describe each facet. Blue squares and black arrows represent Classes and Properties reused from existing vocabularies, while pink squares and red arrows represent Classes and Properties created by UPCAST.

The documentation of the Vocabulary is available at https://eu-upcast.github.io/vocabulary/index-en.html.

4.5.1 Dataset Facet

This facet concerns the Resource Specification and Resource Discovery plugins. We reuse the core classes from the DCAT vocabulary: a Catalogue that hosts Resources and Datasets. We add two additional Catalogued resources: Artefacts, to model resources that are not datasets such as reports or visualizations; and Data Processing Task as a high-level class that allows the modelling of data processing that is not implemented in a containerised piece of Software. For processing tasks that are containerised, we reuse the DataApp class from the IDS Information Model. IDS Participants in a Data Space can be declared as Consumers and Providers of Resources.

In this facet (Figure 28) we introduce Data Processing Workflow as a subclass of prov:Activity (from the widely extended Prov-O vocabulary), linked to the Resources it uses and generates.

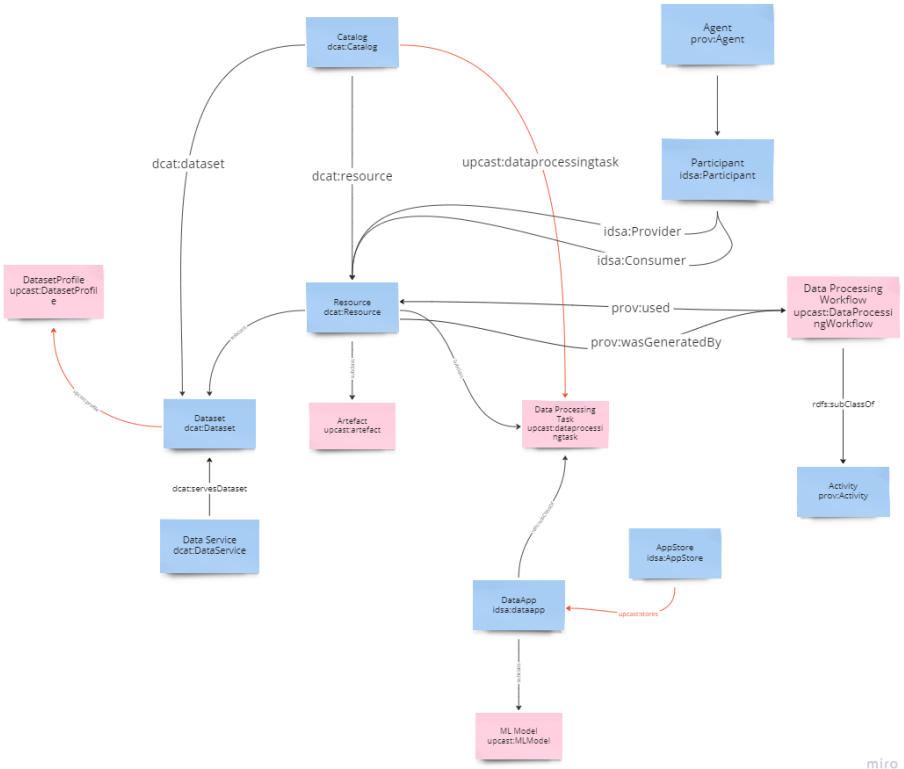

Figure 28: Dataset Facet.

4.5.2 Pricing Facet

The Pricing facet (Figure 29) concerns the Pricing and Valuation plugin. We create a Data Product class to model the packaging of one or more Distributions of a Dataset in a Data Product for sale. In addition to relevant properties and attributes in DCAT Dataset, such as Temporal Coverage and Description, we created attributes on a Data Product to hold the input and the output of the Pricing plugin. We will further research on how data products are being used in other contexts, and how other current EU projects in the Data Space topic handle this concept to assess if we need a further "version" of a Data Product, e.g., to support different prices for different delivery methods.

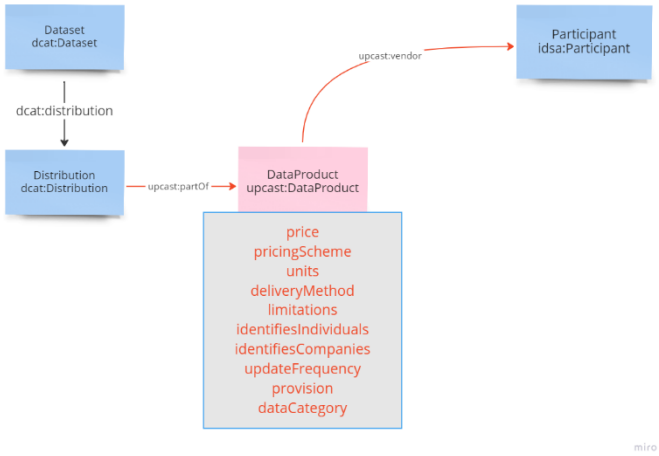

Figure 29: Pricing Facet.

4.5.3 Environmental Facet

We create a class "Environmental Profile" (Figure 30) to hold data about the estimation or measurement (depending on the permission to run the energy monitoring tool developed by CDR) of the environmental footprint of an execution and an "Execution Environment" class to hold hardware information of where a DataApp, a DataService or a Data Processing Workflow are run. This enables the comparison of the environmental profiles of the same Data App or Service in different Execution Environments.

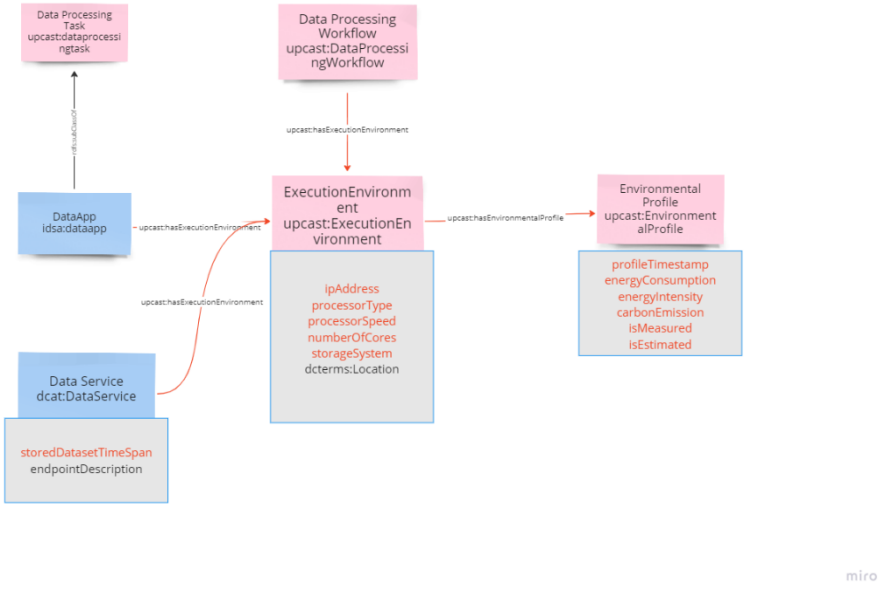

Figure 30: Environmental Facet.

4.5.4 Privacy and Usage Control Facet

This Facet is shown in (Figure 31) Here we reuse previous ontologies created by ICT-Abovo and developed during the BPR4GDPR project. Rules are implemented as

instances of the Permissions, Prohibitions and Obligations classes (sub-classes of the Rules class). The core of the rules are actions, implemented as Actions class instances. Actions consist of four elements, i.e., actor, operation, resource and organisation, that are either defined as concrete entities, by means of ConcreteEntities class instances, or abstract, implemented leveraging the concept of enhanced entities (EnhancedEntities class). The latter are enhanced in the sense that the ontology allows to define, apart from their semantic type, also the constraints upon attributes characterising the entity. For this, appropriate concepts are provided, including Expressions, the instances of which provide for modelling atomic constraints with a subject and a value, and Variables, that allow the specification of concepts in relation to other concepts (e.g., set the value of a resource's constraint to be relative to the semantic type of the actor), thereby maximising expressiveness.

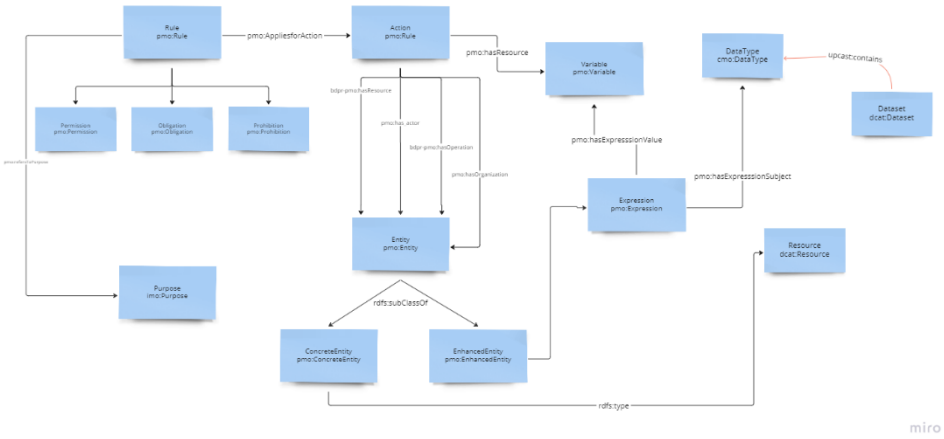

Figure 31: Privacy and Usage Control Facet.

4.5.5 Data Processing Workflow (DPW) Facet

A DPW is a graph where Tasks are the nodes, and the edges define the sequence of application, as well as overall data and control flow (Figure 32).

Edges carry InformationEntities that define the nature of data to be transferred from one task to the next. A wmo:InformationEntity may refer to a specific dataset, or describe transferred data in abstract terms, through a data type and potential additional constraints. An edge may also be characterised by flow conditions and constraints, further specifying and/or restricting the occurrence of implied transition.

A Task has one to several execution profiles. An ExecutionProfile describes an actor that applies an operation (implemented by a DataProcessingTask) on an Asset (often a Dataset), subject to some task conditions. wmo:ActorEntity instances denote the entities assigned with the execution of tasks. They may be defined at either concrete (e.g., a particular person or organisation) or abstract level (e.g., some role and possibly additional constraints). Asset entities represent the objects on which tasks are performed, meaning that they are accessed, processed or modified for the purpose of delivering the corresponding functionalities. They can similarly be defined in either concrete or abstract terms. In UPCAST they are typically expected to refer to (types of) datasets.

The wmo:OperationEntity describes the type of functionality executed at each workflow step. It is defined at an abstract level, through the corresponding operation type, and linked to its concrete materialisation through the upcast:implementedBy property.

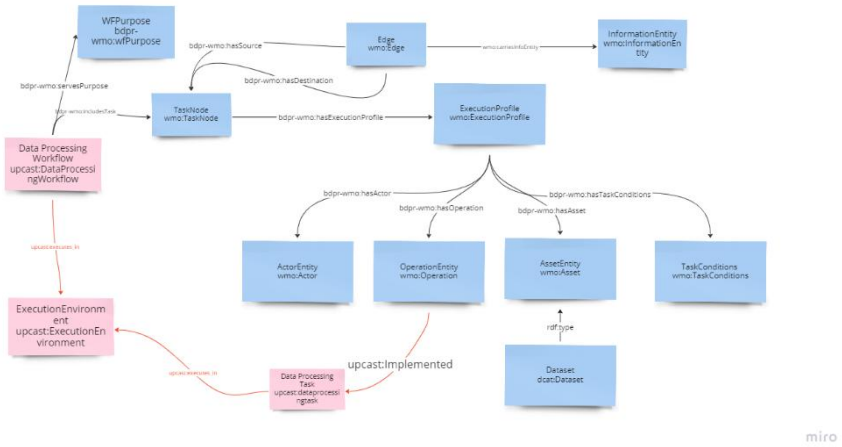

Figure 32: Data Processing Workflow Facet.

4.5.6 Contracting and Negotiation Facet

The Contracting and Negotiation Facet is shown in Figure 33. An instance of Data Processing Workflow describes the processing done to a set of Datasets with a set of Data Processing Tasks. A Contract formalises the state of the agreement of all participants in the DPW with that processing, from “Pending” to “Signed”, and describes the Medium and Jurisdiction (Location) of the Contract.

The description of the processing of a DPW represents the “clauses” of the contract regarding the processing. A party wishing to start a negotiation would propose the removal, update or addition of one or more of the Execution Profiles that comprise the DPW; the amendment is rechecked with respect to existing Rules. If it passes, it is then submitted to the approval of the other Participants.

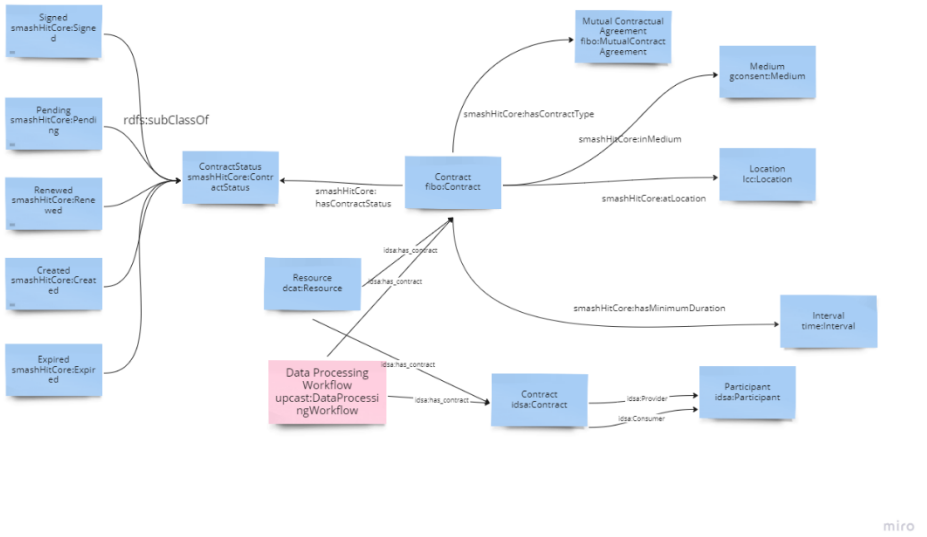

Figure 33: Contracting and Negotiation Facet.

4.6 System Decomposition Model

The UPCAST infrastructure comprises a number of interoperating plugins (components) that collectively support the UPCAST MVP functionalities as shown in Figure 34. The dependencies between the plugins are shown in this figure as well.

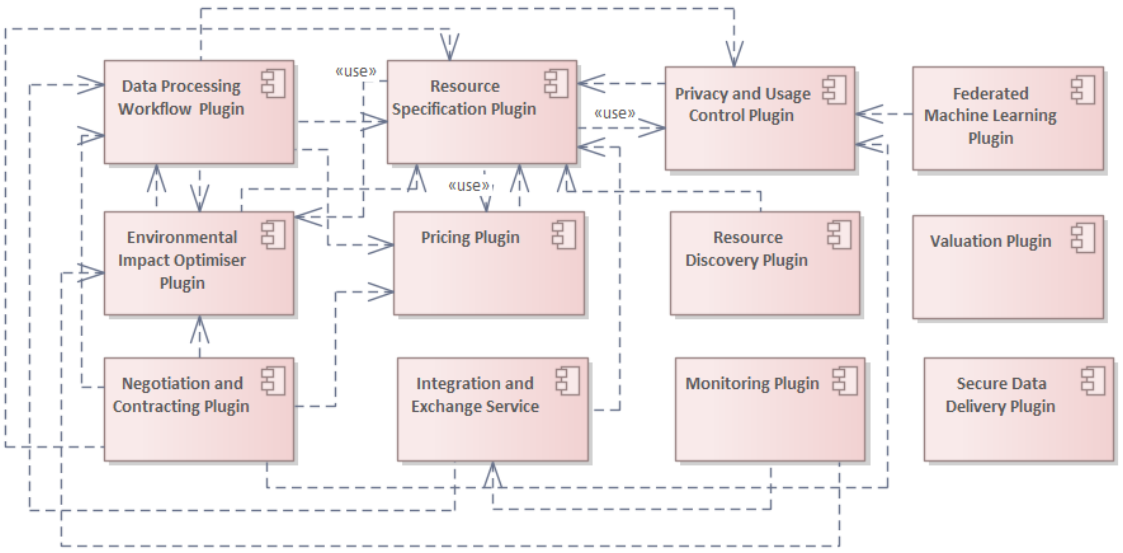

Figure 34: System decomposition model.

4.7 System Collaboration Model

This section describes the system collaboration model. It models the interaction of UPCAST plugins and involved stakeholders regarding the processing of datasets (dataset workflow) as described in the user journeys in Chapter 3. This model represents the most common scenario and uses as many plugins as possible. The collaboration

model for other types of resources (operations & artefacts) is similar, but also simpler, as some activities described below apply only to datasets (e.g., data integration).

Figure 35 illustrates the overall system collaboration model for the dataset workflow, where the upper swimlane represents the workflow for Resource Provider and the lower swimlane represents the workflow for Resource Consumer. The detailed collaboration models for the main activities in Figure 35 are modelled in Figure 36 to Figure 43.

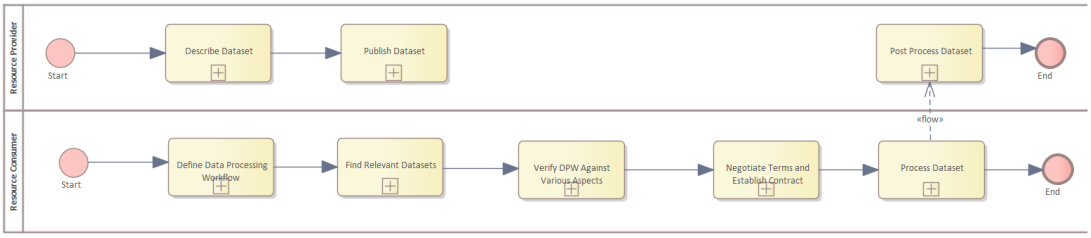

Figure 35: Overall system collaboration model for dataset workflow.

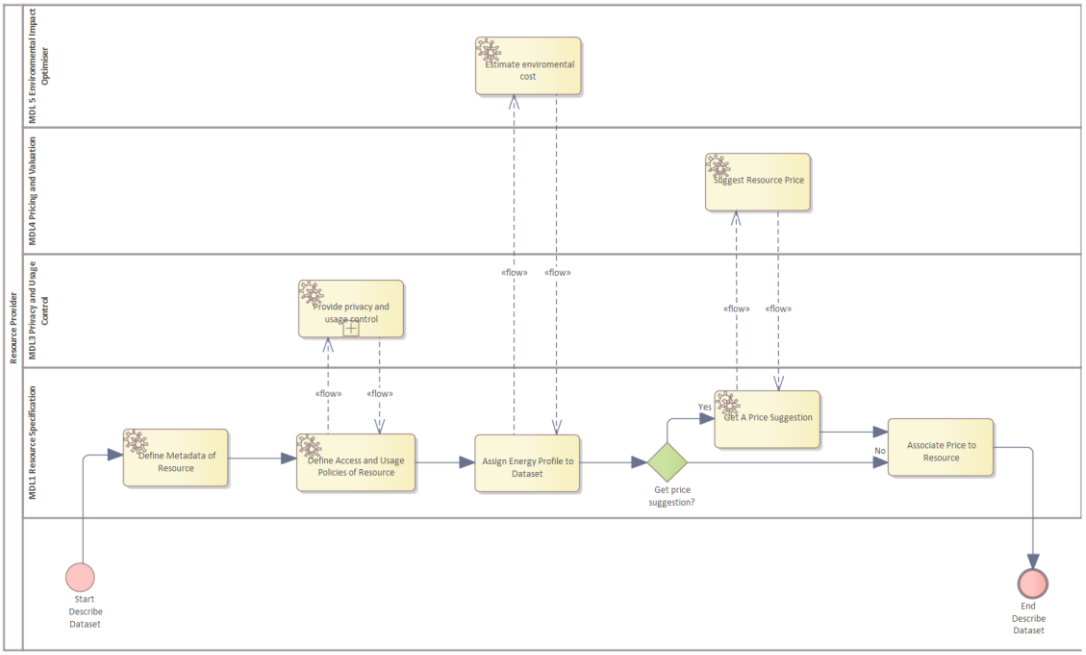

Figure 36: Collaboration model for the "Describe Dataset" process.

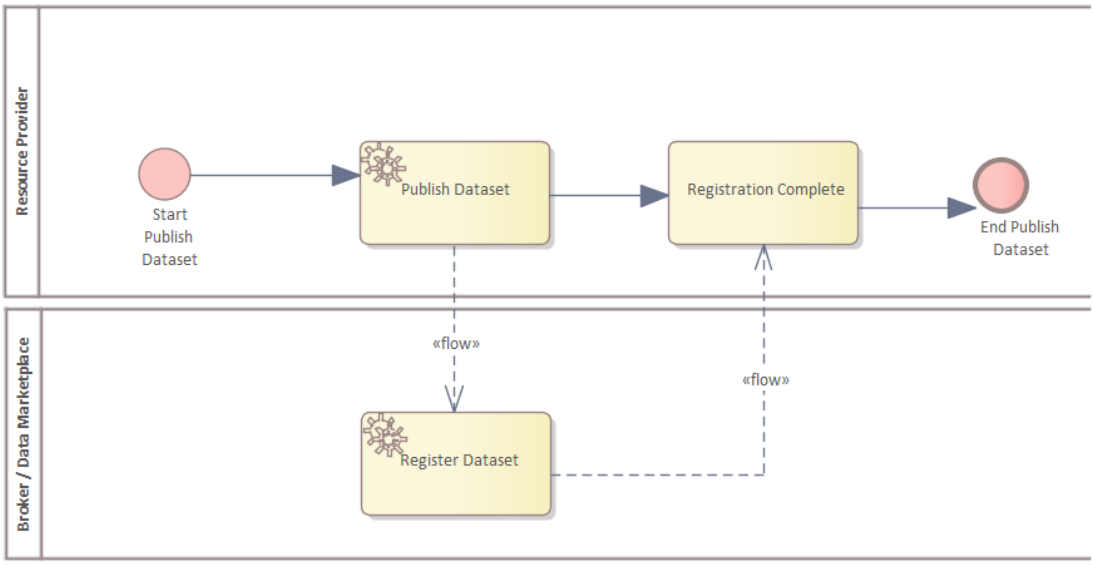

Figure 37: Collaboration model for the "Publish Dataset" process.

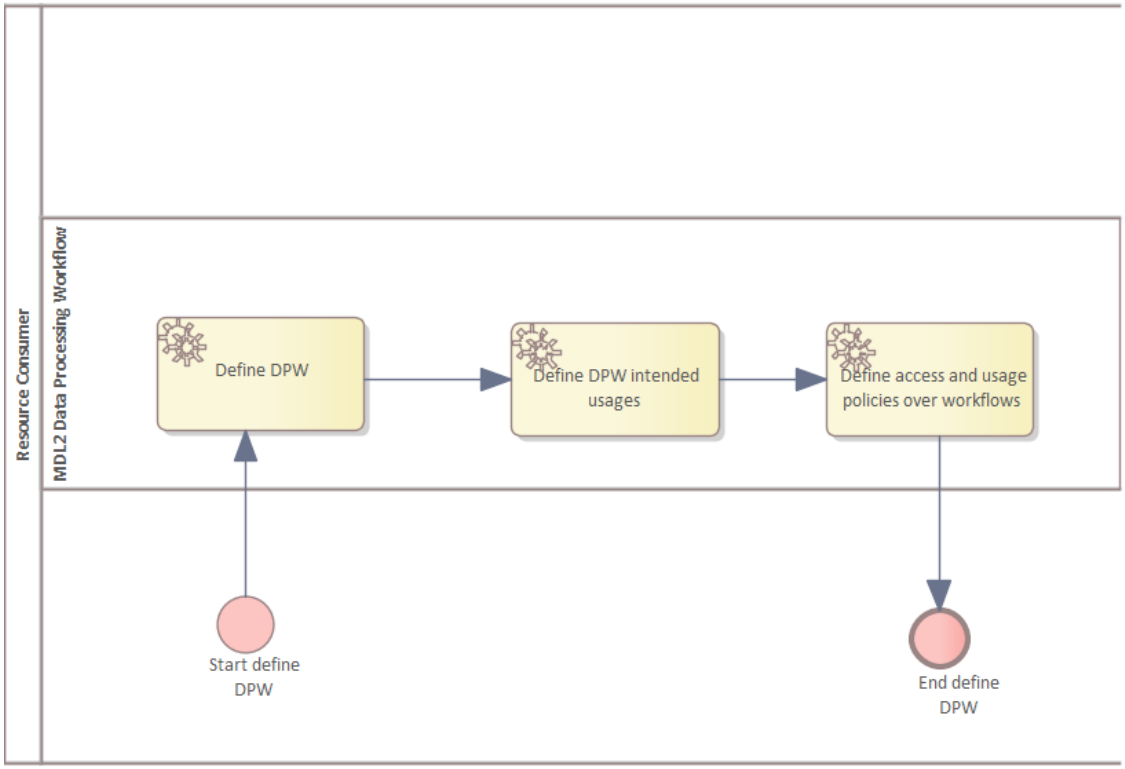

Figure 38: Collaboration model for the "Define Data Processing Workflow" process.

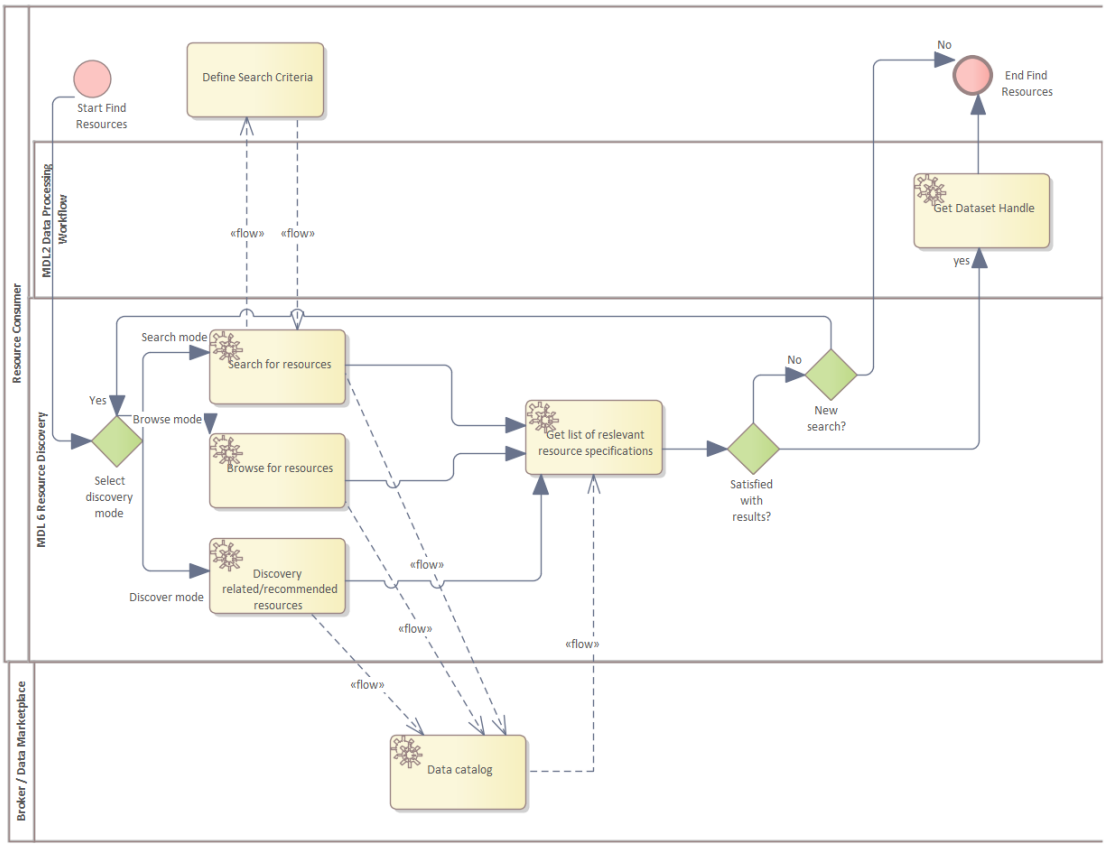

Figure 39: Collaboration model for the "Find Relevant Datasets" process.

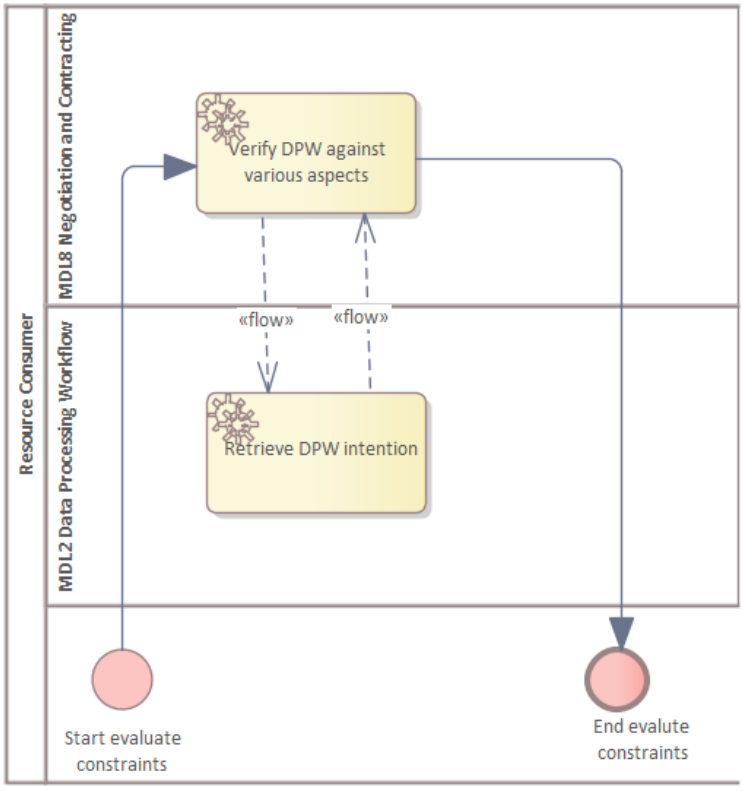

Figure 40: Collaboration model for the "Verify DPW against various aspects" process.

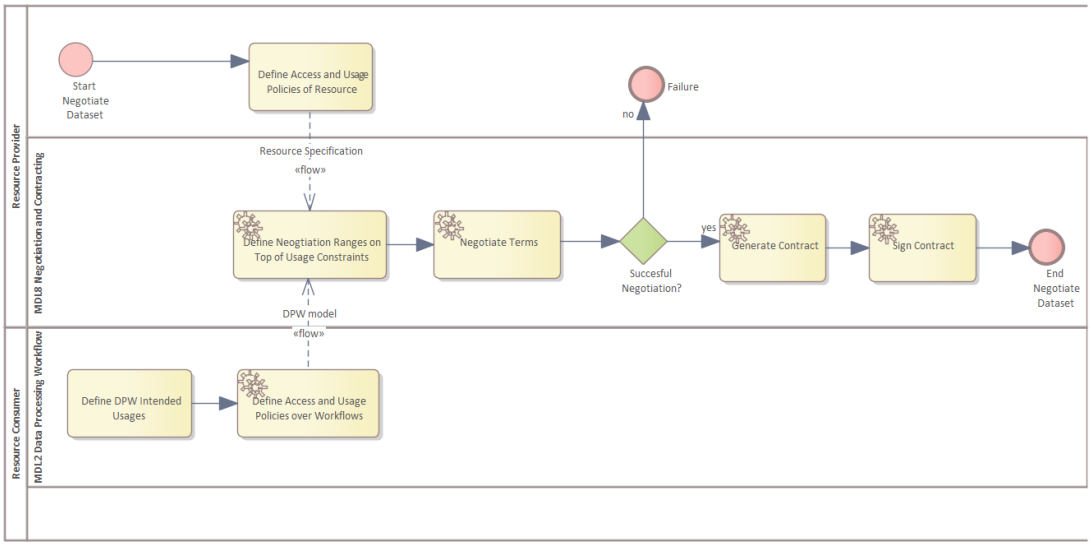

Figure 41: Collaboration model for the "Negotiate Terms and Establish Contract" process.

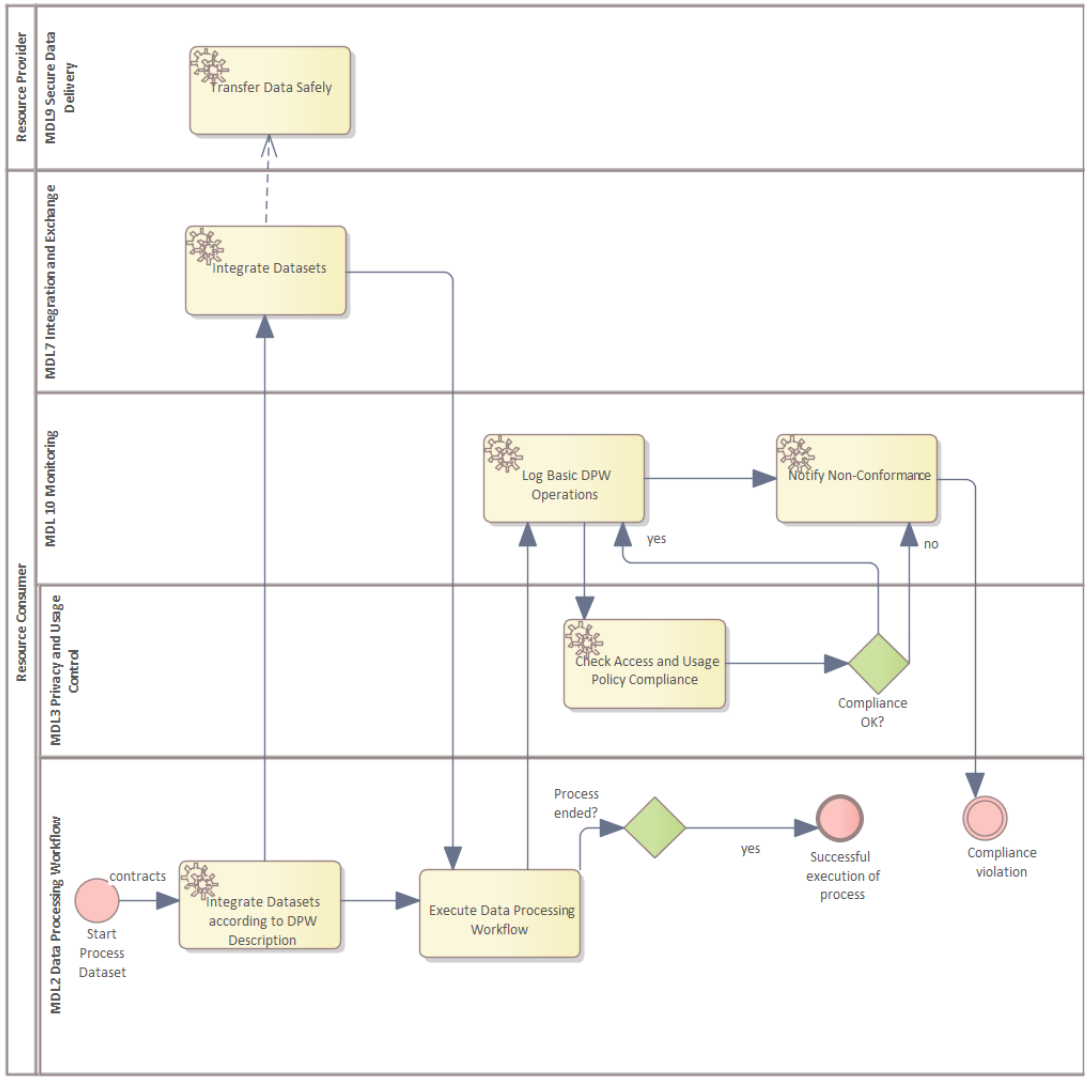

Figure 42: Collaboration model for the "Process Dataset" process.

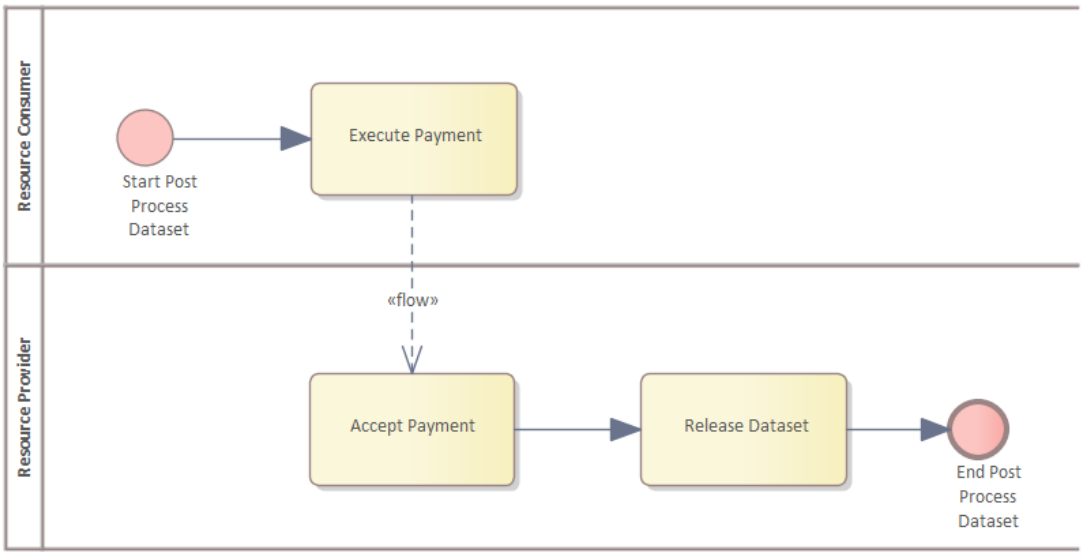

Figure 43: Collaboration model for the "Post Process Dataset" process.

4.8 Component and Interface Specification Model

This section specifies the UPCAST plugin interfaces (i.e., APIs) in terms of methods and their input and output. These APIs can be implemented using different technologies, for example, REST API is the most common API on the Web. The APIs can be communicated synchronously using direct method calls or asynchronously using a data bus for easy integration. The API methods can be converted to messages/events to facilitate the use of a data bus. Therefore, this section defines the UPCAST plugin interface specification in the following tables with a description of method, input, output, topic and message type. The message type can be "Direct" (using method calls) or "Broadcast" (using message queues).

4.8.1 Resource Specification

Table 14: Interface specification for the Resource Specification plugin.

Method	Input	Output	Topic	Message type	description (comments)
CreateResource	Type[Dataset/DataProcessingOperation]	URI of created resource	Resource Spec	Direct	
SetMetadataValues	URI of Resource, Dict of Attribute/Value	OK	Resource Spec	Direct	Metadata Values replaced by input
SetPricingMetadata	URI of Resource, Dict of Pricing Metadata	OK	Resource Spec	Direct	Pricing metadata values replaced by input

SetPrivacyMetadata	URI of Resource, Dict of Privacy Metadata	OK	Resource Spec	Direct	(Privacy metadata replaced by input)
ListResources	[All,Dataset, Data Processing Operation]	List of Resources	Resource Spec	Direct	
getResource	URI	Resource	Resource Spec	Direct	
importVocabulary	Path to vocabulary file	OK	Resource Spec	Direct	

4.8.2 Data Processing Workflow

Table 15: Interface specification for the Data Processing Workflow plugin.

Method	Input	Output	Topic	Message type	description (comments)
defineDPW	Graphical representation of DPW (as designed by the user through the GoodFlows[8] UI)	Ontological DPW model (.owl)	DPWDefinition	Direct	Consumer provides specifications of a DPW along with intended usage
defineDPWpolicies	DPW identifier, access and usage control rules	ResourceDescription	ResourceSpec	Broadcast	The user defines access and usage control constraints at the level of the whole DPW, in case the latter is planned to be advertised as a Resource to be used in other DPWs
StartExecution	DPW		WorkflowProcessing	Direct	Commencement of processing
endExecution	DPW		WorkflowProcessing	Direct	Termination of processing

[8] GoodFlows was developed in the context of BPR4GDPR project and offers a process planning environment -- https://www.ict-abovo.gr/goodflows/

accessResource	DPW		WorkflowProcessing	Direct	Execution requests access to a dataset
releaseResource	DPW		WorkflowProcessing	Direct	Execution releases resource
filter	DPW		WorkflowProcessing	Direct	Execution filters a dataset according to certain criteria
project	DPW		WorkflowProcessing	Direct	Execution projects attributes of a dataset
map	DPW		WorkflowProcessing	Direct	Execution maps a function to a dataset
join	DPW		WorkflowProcessing	Direct	Execution joins a dataset with another
union	DPW		WorkflowProcessing	Direct	Execution unions the dataset with another

4.8.3 Privacy and Usage Control

Table 16: Interface specification for the Privacy and usage Control plugin.

Method	Input	Output	Topic	Message type	description (comments)
defineRPRules	RP constraints as provided upon Resource Specification	PUC rules	PUCrules Specification	Direct	RP' constraints reflected in the respective licences are translated into the underlying semantic policy language in order for the negotiation to take place
defineRCrules	RC rules defined through the PUC UI	PUC rules	PUCrules Specification	Direct	RC's access and usage control rules (i.e., organisation-specific and prescribed by the applicable regulations) defined as PUC rules
identifyConflicts	RP constraints, DPW constraints, RC internal rules	Identified conflicts	ConflictIdentification	Direct	RC access and usage intentions along with organisation-specific and legislation-related rules are checked against RP access and usage constraints and possible conflicts are identified
evaluateAndTransformAccessAndUsageRequest	RP constraints, DPW constraints,	Authorisation decision/ transform	PolicyDecision	Direct	Evaluation and/or transformation of request for access/usage

	RC internal rules	ed request			

4.8.4 Pricing

Table 17: Interface specification for the Pricing plugin.

Method	Input	Output	Topic	Message type	description (comments)
GetSimilarProducts	Resource description	Data products found in commercial DMs	pricing	Direct	The plugin searches in the internal, pre-populated cross-Data-Marketplace database and suggests data products sold in commercial data marketplaces that are close to the description provided by the user.
GetPriceEstimation	Resource (Dataset Metadata)	Price Range	pricing	Direct	The provider will request the price of the dataset adhering to the resource specification vocabulary and this module will calculate the estimated price based on static or dynamic methods
GetXAIMetrics	Resource (Dataset Metadata), Model version (s)	Explainability Metrics	pricing	Direct	This module will generate metrics that explain the factors contributing to the estimated price(s) of the dataset

Note: The XAI Metrics may be included as a part of one single method combined with the price estimation method, depending on the plugin design

4.8.5 Valuation

Table 18: Interface specification for the Valuation plugin.

Method	Input	Output	Topic	Message type	description (comments)
GetEligibleDatasets	Task (model, function, valuation, valuation method, …)	Id of eligible datasets / data providers	Valuation	Direct	The plugin returns the number and id of data providers that are eligible for a certain task or valuation method
GetValue	Task (model, function, valuation, valuation method, …)	Value of the data according to the criteria provider by the	Valuation	Direct	The plugin executes a function or trains a model using data from different providers and returns a number representing this value

	Id of target Dataset / data provider	user (real number)			
GetRelativeValue	Task (model, function, valuation, valuation method, …) Valuation model (Shapley, LOO, etc,) Id of target, Datasets or data providers	Relative value of datasets / data from data providers according to the criteria provider by the user (real number	Valuation	Direct	The plugin returns a relative valuation of different datasets or data from different data providers in the system that are eligible for a particular task.
GetValue Explanations	Id of target valuation exercise	Explanation of values obtained (str, values, etc.)	Valuation	Direct	The plugin returns an explanation for the values calculated in a previous value calculation exercise.[9]

4.8.6 Environmental Impact Optimiser

Table 19: Interface specification for the Environmental Impact Optimiser plugin.

Method	Input	Output	Topic	Message type	description (comments)
GetEnergyProfile	Resource (Dataset metadata), Hardware information, operations	Profiled dataset energy consumption	Env_optimiser	Direct	The provider will request the estimated energy profile of the dataset
GetPowerConsumption	Infrastructure access, operations	Actual power consumption	Env_optimiser	Direct	This method will monitor the operations and processes applied to the dataset
GetXAIMetrics	Energy profile, Resource (Dataset metadata)	Explainability Metrics	Env_optimiser	Direct	This module will generate metrics that explain the factors contributing to the energy profile of the dataset

[9] This is strongly context-dependent. A particular method will be delivered for the use case in health and fitness.

4.8.7 Resource Discovery

Table 20: Interface specification for the Resource Discovery plugin.

Method	Input	Output	Topic	Message Type	description (comments)
searchFacet	Search Facet Object	List of Resources		Direct	Get a list of resources based on a query generated from a facet
searchText	Search Text	List of Resources		Direct	Get a list of resources based on a query generated from a textual explanation
getRelatedDatasets	Resource Metadata	List of Resources		Direct	Get a list of resources based on a query generated from a single resource
startDiscovery	Resource Metadata	OK	discovery	Broadcast	Start the discovery process for a new resource
updateDiscovery	Resource Metadata	OK	discovery	Broadcast	Update the related content about a resource from the knowledge graph
cleanResource	Resource Metadata	OK	discovery	Broadcast	Delete the related content about a resource from the discovery knowledge graph

4.8.8 Negotiation and Contracting

Table 21: Interface specification for the Negotiation and Contracting plugin.

Method	Input	Output	Topic	Message type	description (comments)
submitDatasetRequest	Dataset identifier, RC metadata	OK	Negotiation	Direct	Once RP and RC have been matched, RC submits a request for the matching resource to the RP; RP gets informed that their dataset is included in the DPW specified by the RC
editRPNegotiationParameters	Dataset description	Dataset description	Negotiation	Direct	RP defines the negotiation range for each statement in the resource specification
editRCNegotiationParameters	DPW	DPW	Negotiation	Direct	RC may fine-tune the DPW specification in order to reflect their own negotiation ranges

verifyDPW	DPW, RP constraint, RC rules	Identified conflicts, DPW alternatives	Negotiation	Direct	DPW gets verified, resulting in identification of conflicts and suggestions for conflict resolution reflected in valid DPW alternatives.

4.8.9 Integration and Exchange

Table 22: Interface specification for the Integration and Exchange plugin.

Method	Input	Output	Topic	Message type	description (comments)
submitSourceSpec	Specifications of a source	OK	Integration and Exchange	Direct	Consumer provides specifications of a source, which can then be used to define a mapping from source to target.
rewriteQuery	Source, one or more mappings, and global schema	Access to integrated source	Integration and Exchange	Broadcast	Broadcast perhaps to any remote machines that can run the integration. Otherwise, direct message if run locally
materialiseViews	Source, one or more mappings, and global schema	Access to integrated source	Integration and Exchange	Broadcast	Same as above. Perhaps duplicate and we can give the choice of forward or backward chaining as input. Of course, mappings are different for each case.
requestSource	Identifier for a source	Access to source	Privacy	Direct	Consumer requests access to a source (that they should have access to) so it can be used in integration.

4.8.10 Secure Data Delivery

The detailed design and implementation of this plugin will be provided when the new project partner responsible for this plugin joins the consortium.

4.8.11 Monitoring

Table 23: Interface specification for the Monitoring plugin.

Method	Input	Output	Topic	Message type	description (comments)
configMonitoring	Monitoring Data		Processing Monitoring	Direct	Configuration of monitoring plugin
startMonitoring	Monitoring Data		Processing Monitoring	Direct	Commencement of monitoring
endMonitoring	Monitoring Data		Processing Monitoring	Direct	Termination of monitoring
sendMonitoringData	Monitoring Data		Processing Monitoring	Direct	Execution sends monitoring data

4.8.12 Federated Machine Learning

Table 24: Interface specification for the Federated Machine Learning plugin.

Method	Input	Output	Topic	Message type	description (comments)
initializeFederatedLearning				Direct	Initializes the federated machine learning process
registerDevice	deviceID (txt; a unique identifier for the device), deviceType (txt; the type of the device, e.g., mobile, IoT)			Direct	Registers a new device to participate in federated learning
setTrainingParameters	trainingParams (Dict; A dictionary containing training parameters)			Direct	Sets training parameters such as learning rate, epochs, and batch size for the federated learning process
sendModeltoDevice	deviceID (txt), model (Model; the machine learning model to be trained locally)			Direct	Sends a machine learning model to a registered device for local training
trainLocalModel	deviceID (txt), trainingData	trainedModel (Model;		Direct	Trains the locally provided machine

	(local training data)	the locally trained machine learning model)			learning model on the device using local data
aggregateLocalModels				Direct	Aggregates the locally trained models from registered devices to create a global federated model
evaluateGlobalModel	globalModel (Model), evaluationData (Data)	evaluationMetrics (Dict; metrics assessing the performance of the global federated model		Direct	Evaluates the performance of the global federated model on evaluation data.
updateGlobalModel	globalModel (Model; the updated global federated machine learning model)			Direct	Updates the global federated model with the latest aggregated version
stopFederatedLearning				Direct	Stops the federated machine learning process
getFederatedLearningStatus		status (str), Current status (e.g., 'Idle', 'Training', 'Completed')		Direct	Retrieves the current status of the federated learning process.
sendMessage	deviceID, messageType, messageData (The data contained in the message)				Sends a message from a device to the central system

4.9 UPCAST Architecture

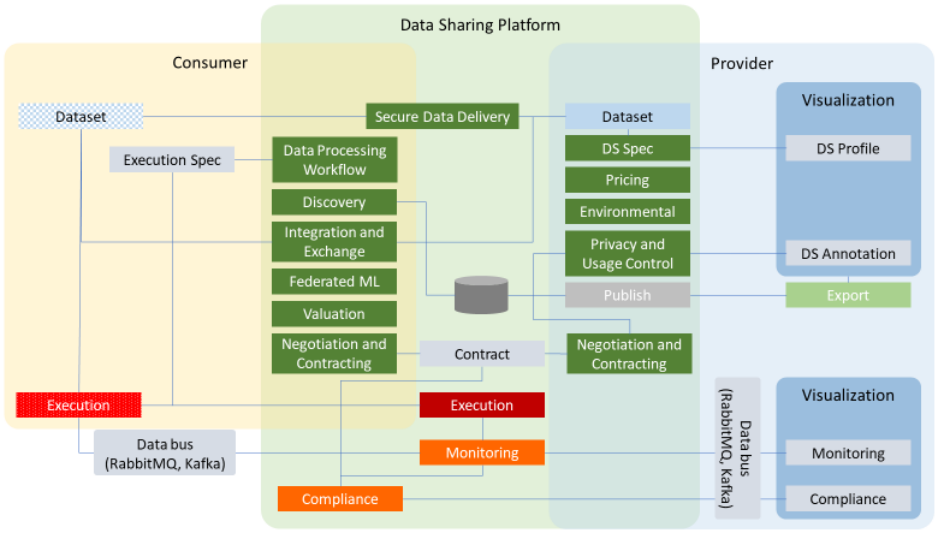

Figure 44: UPCAST technical architecture.

Figure 44 depicts the overall architecture of UPCAST. The architecture is the result of the analysis of user requirements, the available technologies, and technical approach of UPCAST as shown in the DoA.

The three major roles that are shown in the architecture are:

- Data Consumer[10]: the entity that makes use of a dataset through processing it.
- Data Provider: the entity that makes a dataset available to other entities for processing.
- Data Sharing Platform[11]: The entity that hosts the UPCAST plugins and facilitates the interactions between providers and consumers. Datasets are also hosted by the Data Sharing Platform.

The UPCAST architecture shows that providers and consumers interact through the Data Sharing Platform. The overlapping areas between the Data Sharing Platform and the provider and the Data Sharing Platform and the consumer signify that both providers and consumers are identified in the Data Sharing Platform and are authenticated into it before any interactions take place.

UPCAST plugins are shown in green or orange and are deployed in the Data Sharing Platform. Some plugins are addressed to the provider, others to the consumer, while the Negotiation and Contracting plugin is meant to be used by both providers and consumers to reach consensus on the use of the dataset that is expressed through a contract.

[10] UPCAST supports resources which can be datasets, data operations or artefacts. In this section, we focus on datasets, thus address Data Consumers and Data Providers instead of Resource Providers and Resource Consumers. All plugins handle datasets while some of the plugins are also applicable for other types of resources.
[11] A data marketplace is an example of a Data Sharing Platform.

The provider uses a number of plugins to create a specification of a dataset (Resource Specification plugin), annotate it with pricing information (Pricing plugin), environmental information (Environmental Impact Optimiser plugin) and specifications of its privacy, access and usage policies (Privacy and Usage Control plugin). This set of specifications express the provider's requirements for the intended use of the dataset that they make available. The dataset specification and annotation are visualised by the provider through a visualization engine. An example of such visualization engine is Mira, a digital platform that supports the functions of digital twins. In UPCAST, the dataset is a data asset, while Mira is used to specify and implement its digital twin. The specification and annotations of the dataset twin are exported and further published to the Data Sharing Platform by the provider.

Similarly, the consumer uses a number of plugins to specify the dataset they need for processing, and the intended processing of it. The Data Processing Workflow plugin is used to specify the algorithms that will be applied to the dataset, and it expresses the consumer's requirements for the processing of the dataset. The Data Processing Workflow is used to produce the processing specification for the dataset (a DPW model), which will be further converted to Execution Specification. Once the processing requirements have been expressed by the consumer, the Resource Discovery plugin is used to search for a dataset in the Data Sharing Platform based on consumer search criteria.

The Negotiation and Contracting plugin is used by both the provider and the consumer for negotiating access and usage parameters that are not covered by the provider's specifications of the dataset and the consumer requirements of the dataset. The successful outcome of the negotiation is the formation of a contract that will be respected during processing of the dataset.

Once a contract between the consumer and the provider of the dataset is in place, processing of the dataset may commence. Execution may take place within the Data Sharing Platform, if the Data Sharing Platform supports such functions or, as is the case of most UPCAST pilots, in the consumer's space. If execution takes place in the consumer site, a replica of the dataset must be securely transferred to the consumer using the Secure Data Delivery plugin. The execution engine[12] that is deployed in either the Data Sharing Platform or the consumer site executes the Execution Specification that results from the Data Processing Workflow modelling.

The UPCAST Monitoring plugin has two modules (the Monitoring module and the Compliance module) as shown in the two orange boxes. Execution of the dataset produces monitoring data that are fed to the Monitoring module. If execution takes place in the Data Sharing Platform the monitoring data are fed directly to the Monitoring module, otherwise they are fed to it through standard messaging technologies like RabbitMQ or Apache Kafka. The Compliance module assures that processing of the dataset proceeds according to the contract. It, therefore, receives the monitoring data

[12] The execution engine is outside of UPCAST scope and is to be provided by the consumer or the Data Sharing Platform. It can be any engine that can executes the processing specification, for example, an execution engine that can run bash scripts if the processing specification can be converted to bash scripts.

that are produced by the datasets being processed and the contract specifications and produces alerts in case any violations are detected.

The monitoring data and the compliance data may also be visualized by a platform like Mira for the provider to have a precise view of the progress of the processing of their dataset and possible violations of the agreed contract with the consumer. Monitoring data and data coming from the Compliance module are shipped to the visualization (for example Mira) through a data bus.

The architecture shown in Figure 44 will be used for the developments of UPCAST. An alternative architecture as shown in Figure 45 has also been assessed by UPCAST. The architecture of Figure 45 separates the provider, the consumer and the Data Sharing platform. Interactions between the provider and consumer take place directly between them, while the Data Sharing platform takes the role of a broker that facilitates the advertisement of dataset and searching for them. The architecture of Figure 45 introduces a new component, the UPCAST Engine, that is used to deploy and host plugins to the provider and the consumer. The UPCAST Engine may replicate some of the functionalities provided by the Data Sharing platform. Moreover, a Trusted Third Party (TTP) is introduced to guarantee the safe processing of the dataset. The functions of the TTP can be provided by the Data Sharing platform itself.

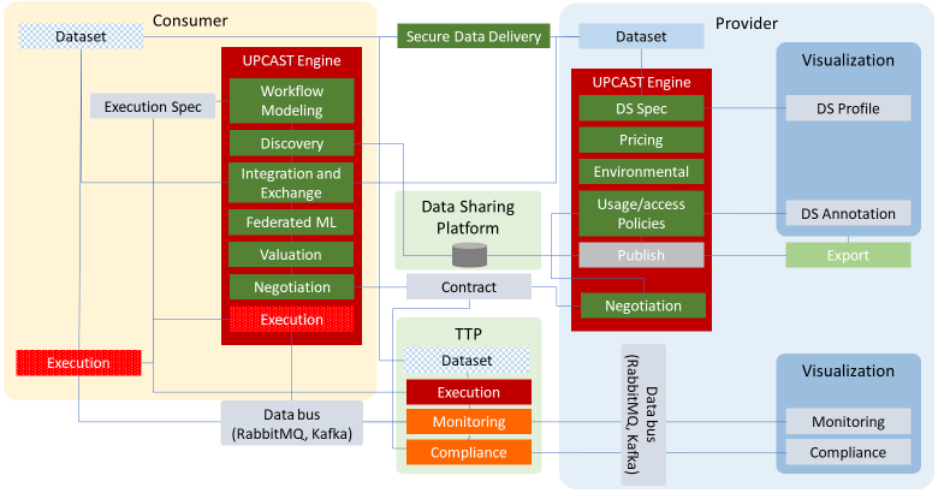

Figure 45: UPCAST alternative technical architecture.

UPCAST will use the architecture of Figure 44 for its developments as this architecture complies with the project's DoA that calls for the development of plugins for data marketplaces instead of building the UPCAST Engine of Figure 45 that replicates the functions of a typical Data Sharing Platform.

5 PILOT DESIGN AND FUNCTIONALITY – BIOMEDICAL AND GENOMIC DATA SHARING

This pilot focuses on biomedical and genomic data sharing between NHRF and their research partners. NHRF uses bioinformatics tools to analyse in-house generated genomic data and explore molecular and clinical data from well-established cancer-associated data repositories.

5.1 Roles and Stakeholders

As shown in Figure 46, NHRF acts as both resource/data consumer (utilising data from research partners and data repositories in biomedical analysis) and resource/data provider (sharing analysis results with others).

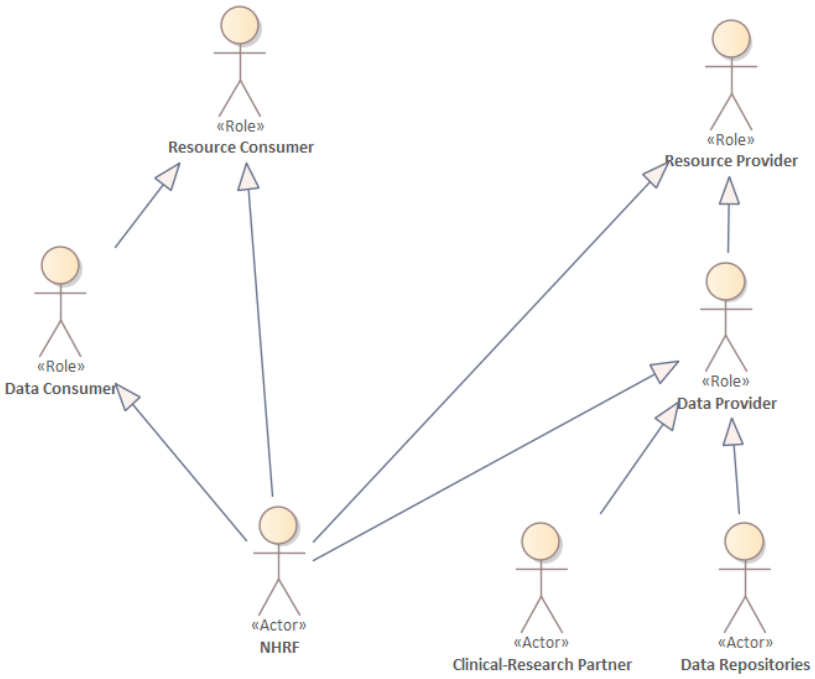

Figure 46: Roles and Stakeholders for the Biomedical pilot.

5.2 Reference Use Cases

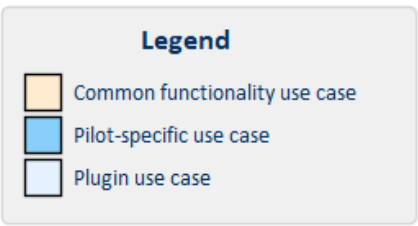

Figure 47: Legend of the use case colours used for the pilot reference use cases.

Figure 48 presents the top-level use cases for the Biomedical and Genomic Data Sharing pilot. These top-level use cases are detailed in Figure 49, Figure 50 and Figure 51. These diagrams reuse the plugin use cases whenever possible where plugin use cases are illustrated with the UPCAST Plugin name in namespace (parenthesis). The use cases are distinguished in different colours as depicted in Figure 47. These conventions are followed in the description of all pilot reference use cases in Chapter 5 – Chapter 9.

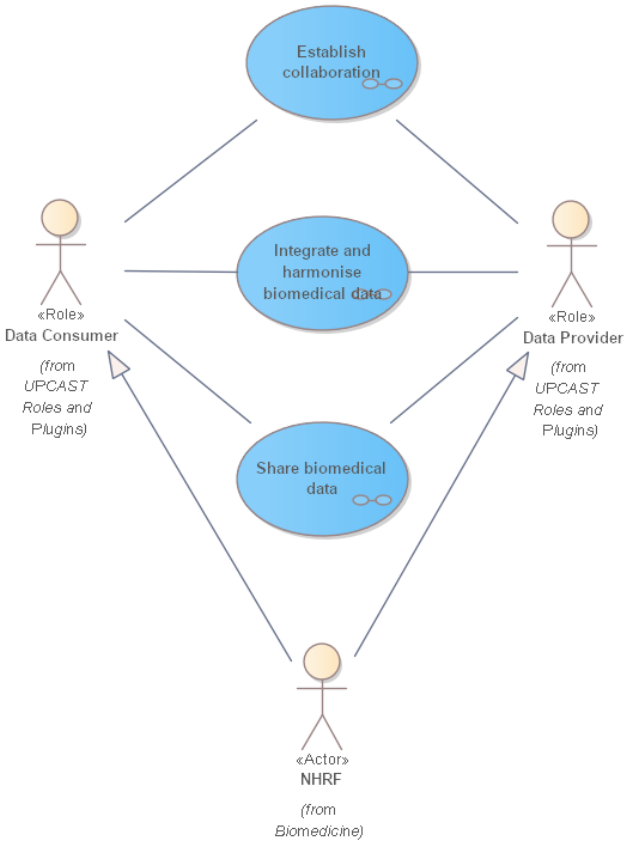

Figure 48. Top-level use case model for the Biomedical pilot.

For the <<Establish collaboration>> use case (Figure 49): This use case enables different stakeholders to establish contractual agreements with NHRF for collaboration, including definition of specific clauses, obligations and timelines and assurance of the compliance of legal and ethical requirements. The UPCAST system will ensure that formal declarations exist that patient consents are in place before contracts can be established. It also includes a review and approval process for all involved parties to reach agreements and establish contracts.

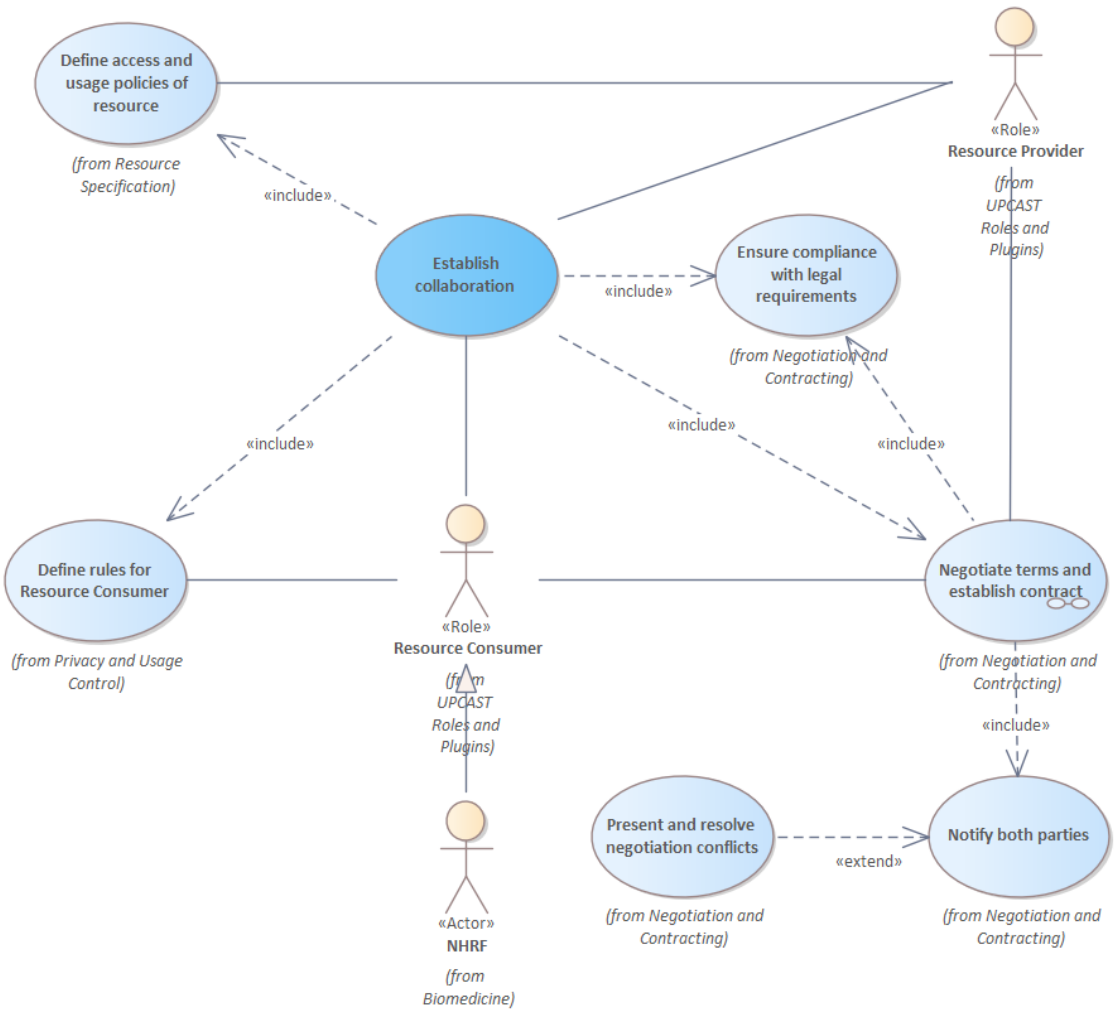

Figure 49. Establishing collaboration between NHRF and external resource providers.

For <<Integrate and harmonise biomedical data>> (Figure 50): This use case integrates and harmonises biomedical data from different sources (data repositories). It allows to define and execute data processing workflow that combine and analyse data effectively for genomics research.

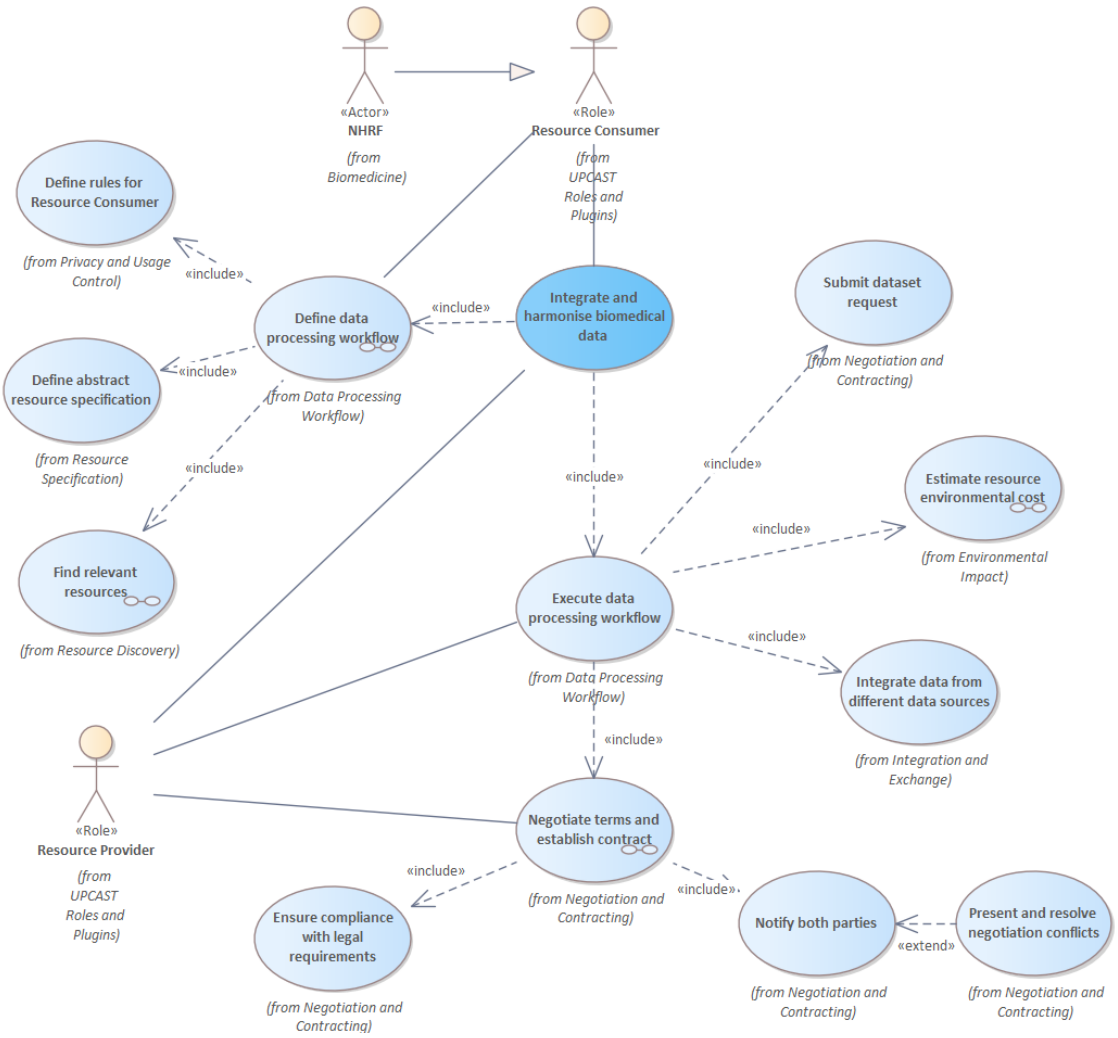

Figure 50. Integrate and harmonise biomedical data.

For the <<Share biomedical data>> use case (Figure 51): This use case offers a secure data sharing framework for NHRF to share genomic data with data consumers. It also facilitates the commercialisation of NHRF proprietary or curated genomic datasets.

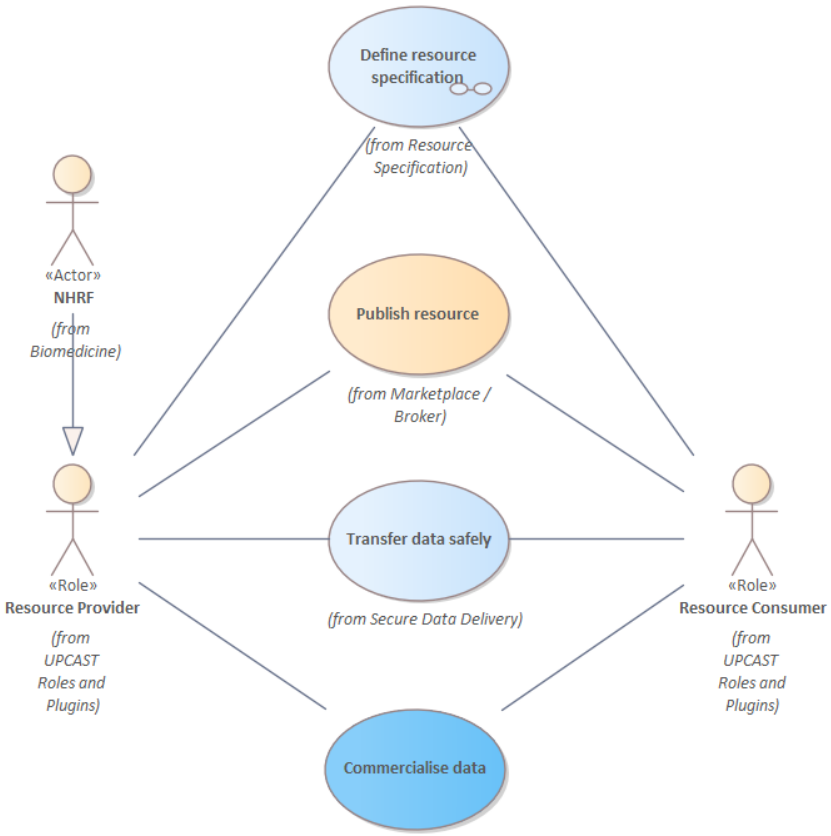

Figure 51. Sharing biomedical data.

5.3 Business to Systems Mapping Model

Figure 52 shows how the top-level use cases of the pilot are implemented by the UPCAST plugins.

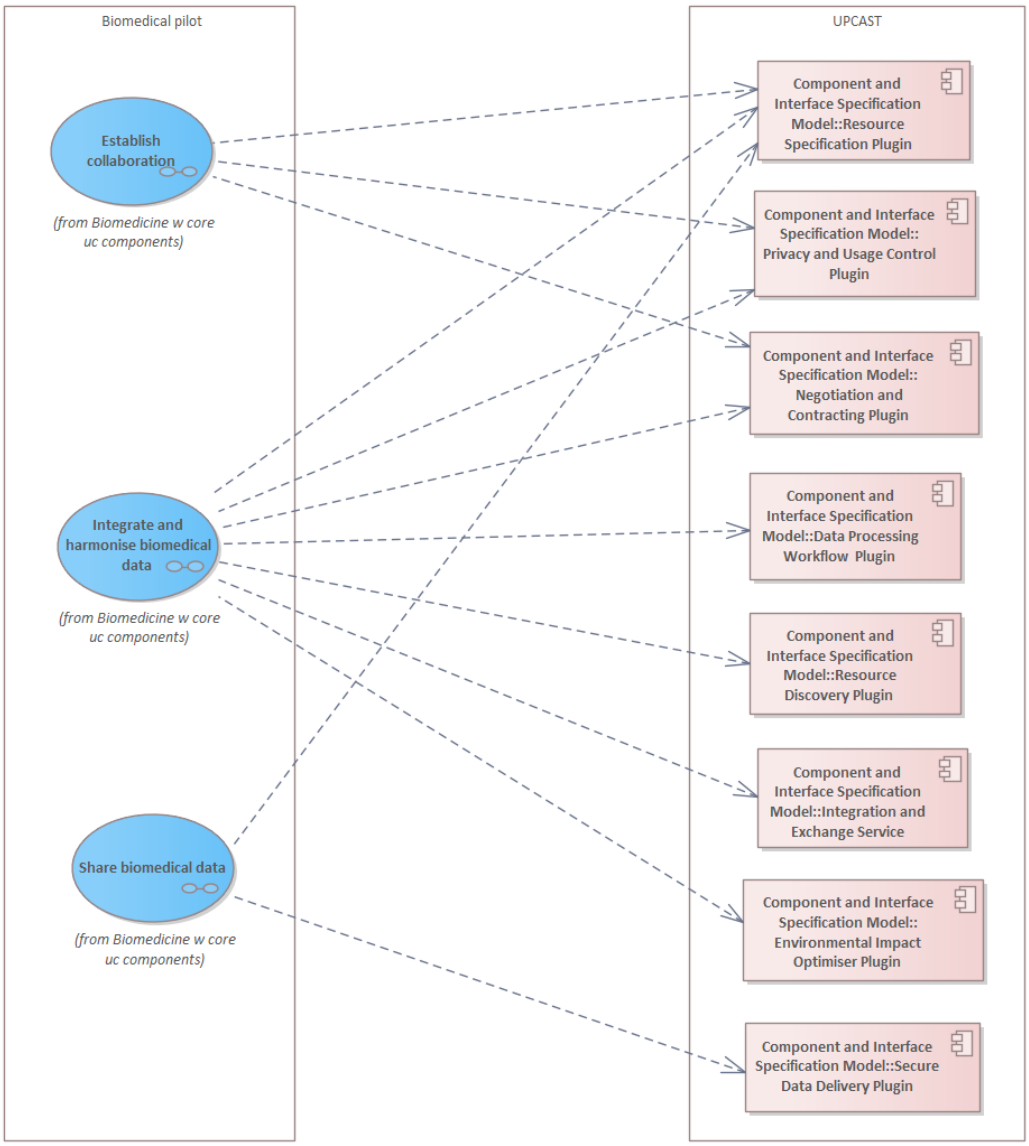

Figure 52: Business to Systems Mapping Model for the Biomedical pilot.

5.4 Pilot Implementation Plan

NHRF researchers are defining Data Processing Workflows, constituting implementations of algorithms by means of well-structured data processing scenarios. The underlying tasks are typically executed by one or more software tools and applications devised for scientific data processing. The latter can be math tools (e.g., R) or Python scripts, but, in most cases, they refer to specialised bioinformatics software, such as raw sequence data quality controllers, NGS aligners, NGS RNAseq and WES analysers, annotators, functional analysers, etc. Execution takes place either at NHRF's own infrastructure or processing resources provided by research and academic clouds.

The fundamental fuel of bioinformatics workflows is data. To this end, NHRF possesses own datasets, and seeks for new datasets to be a leverage to NHRF research. There are several sources, repositories and marketplaces for genomic data. Following UPCAST, some of them are assumed to adopt UPCAST, thereby offering respective functionality. For the rest, referred to as “legacy marketplaces”, it is assumed that discovered datasets

are imported “as is” (essentially comprising own datasets) or that are subject to some UPCAST functionality (e.g., pricing).

The selected deployment scenario for the Biomedical pilot is depicted in Figure 53. According to this, all UPCAST functionality is offered through an UPCAST-enabled marketplace by means of plugins, whereas the execution of the Data Processing Workflows themselves will take place on NHRF infrastructure (including other shared research and academic facilities).

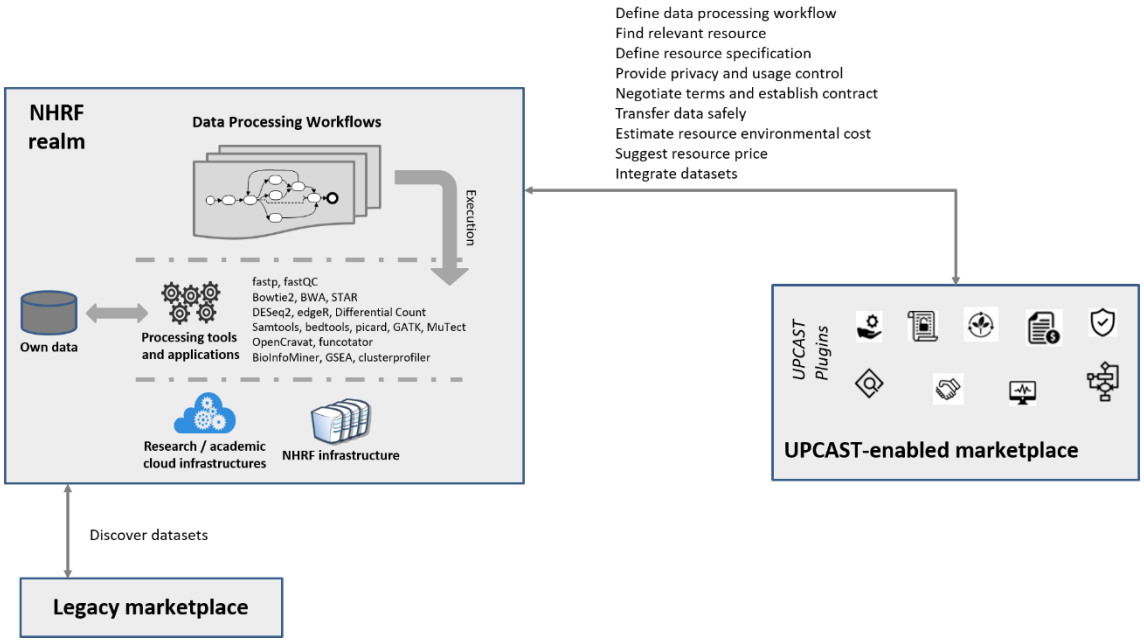

Figure 53: Deployment scenario for the Biomedical pilot.

6 PILOT DESIGN AND FUNCTIONALITY – PUBLIC ADMINISTRATION

In this pilot, the Major Development Agency Thessaloniki (MDAT) and Open Knowledge Foundation Greece (OKF Greece) will use the UPCAST plugins for integration and exchange of all data related to the data driven environmental policy making of the Metropolitan Area of Thessaloniki.

6.1 Roles and Stakeholders

As shown in Figure 54, MDAT-OKF acts as both data provider and data consumer in this pilot.

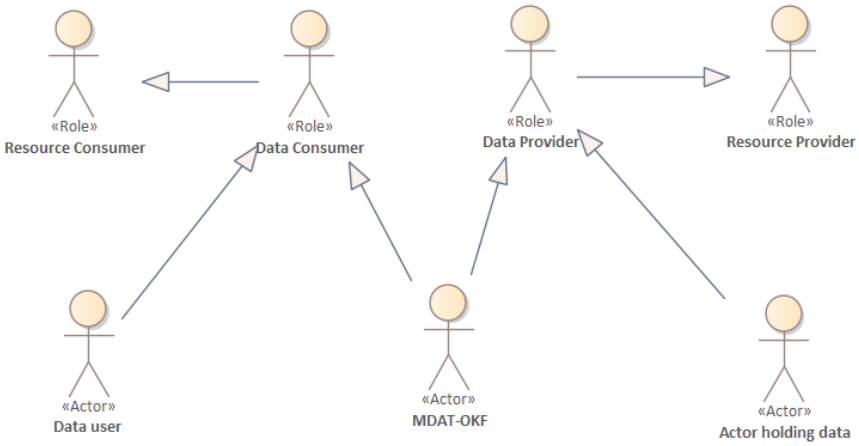

Figure 54: Roles and stakeholders for the Public Administration pilot.

6.2 Reference Use Cases

Figure 55 shows the top-level use cases for the Public Administration pilot.

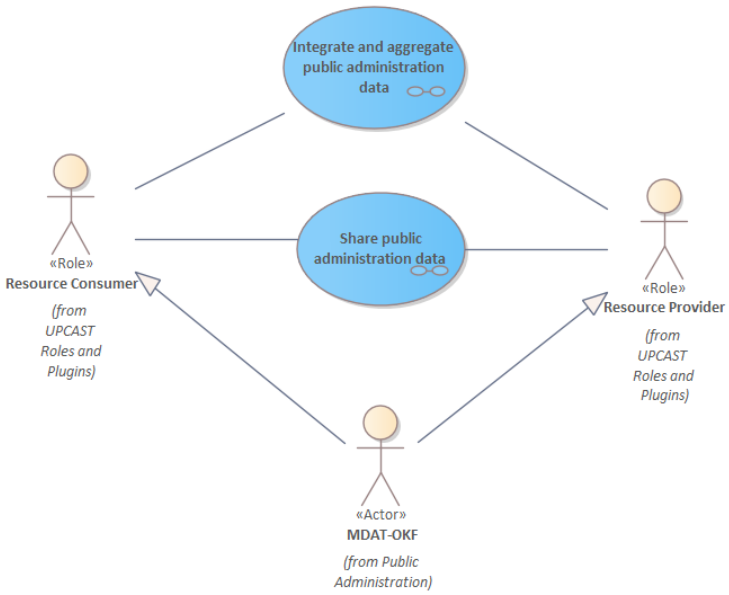

Figure 55. Top-level use case for the Public Administration pilot.

For <<Integrate and aggregate public administration data>> use case (Figure 56), MDAT-OKF acts as a resource consumer, which integrates and aggregates the data based on a defined data processing workflow. Using UPCAST plugins, MDAT-OKF can negotiate with data providers in an automated way and at the same time respect data providers' privacy conditions and requirements.

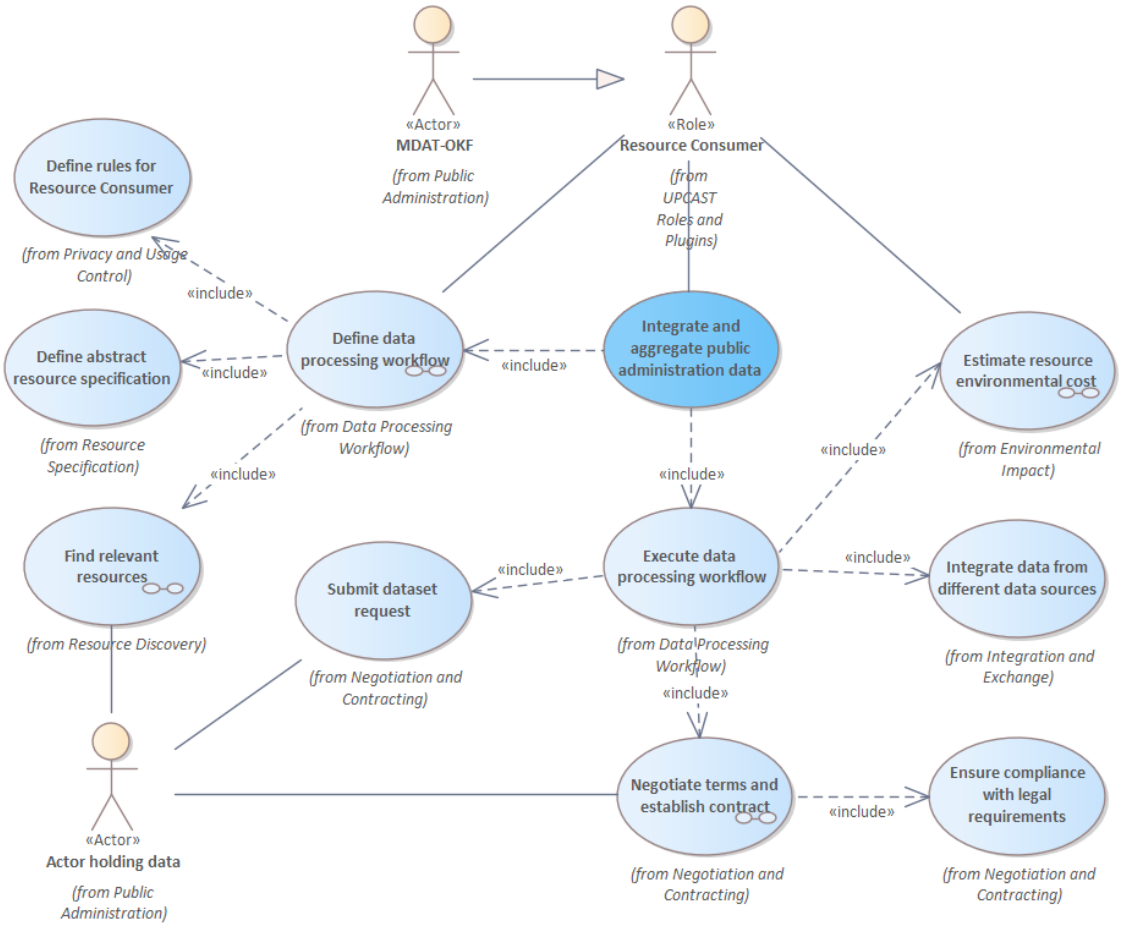

Figure 56. Integrate and aggregate public administration data.

For <<Share public administration data>> use case (Figure 57): MDAT-OKF publishes datasets specified using domain-specific vocabularies and ontologies and the datasets will be transferred securely to data users.

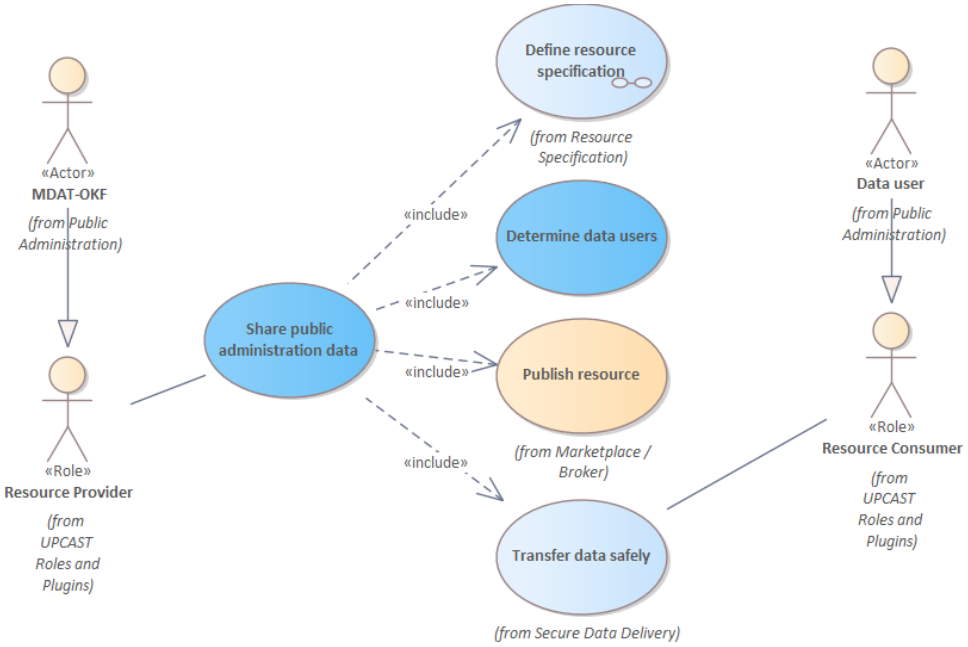

Figure 57. Share public administration data.

6.3 Business to Systems Mapping Model

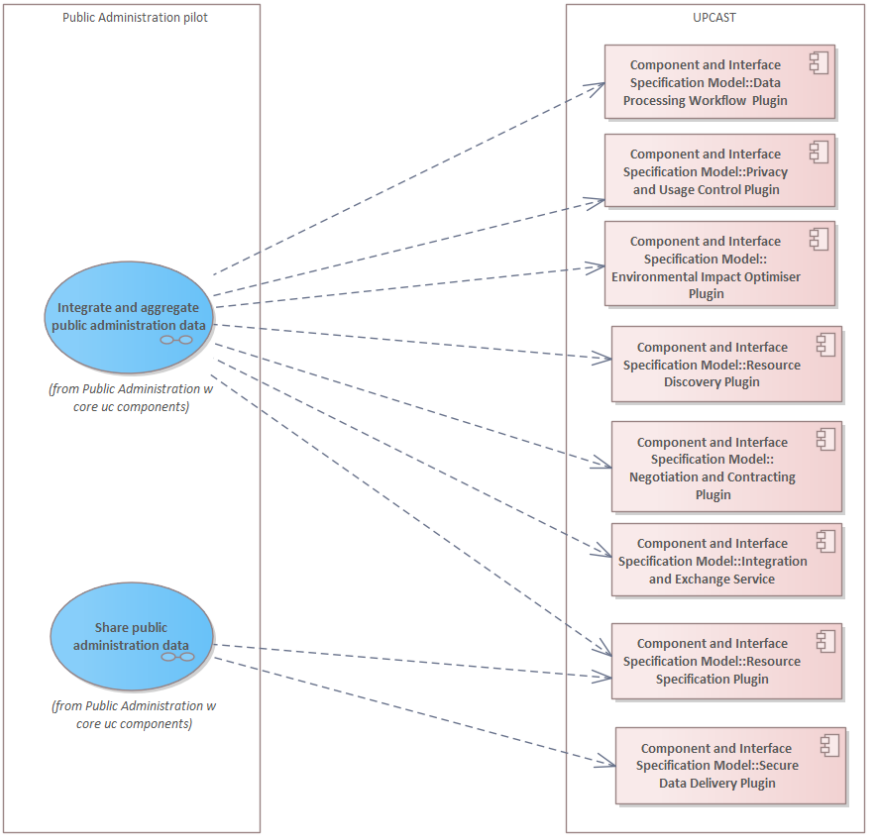

Figure 58: Business to Systems Mapping Model for the Public Administration pilot.

Figure 58 shows how the top-level use cases of the pilot are implemented by the UPCAST plugins.

6.4 Pilot Implementation Plan

In this pilot, the data marketplace operates on top of an open data repository (DKAN[13]) and is under the ownership and management of MDAT.

MDAT will install DKAN in its own infrastructure. To benefit from the UPCAST services, a selection of the UPCAST plugins will be installed as Docker containers in the same infrastructure. The functionality provided by the UPCAST plugins will be available to DKAN through Drupal modules that will be developed by OKF Greece.

More specifically, MDAT will undertake the task of mapping the environmental data market in the metropolitan area of Thessaloniki and creating resource specifications for relevant data.

Data providers encompass a diverse group, including public organisations, non-governmental organisations (NGOs), civil associations, private companies, and even individuals who collect sensor measurements for personal purposes.

Data consumers can belong to the same categories mentioned above, and they may also include other entities interested in staying informed about international activities related to data-driven environmental policymaking. They can share and compare practices, data, and indicators with other European cities. Consumers will design data processing workflows to discover pertinent datasets within the providers' repositories using the Resource Discovery plugin.

To ensure privacy and usage control, the Privacy and Usage Control plugin will be employed. The plugin will ensure that environmental datasets do not contain any personal information and are fully anonymised. Additionally, the resulting datasets may require cleaning, integration, and aggregation as part of the designed data processing workflow.

Public entities, NGOs, civil associations, and private companies providing environmental data may have specific requirements regarding how their data is used. To address these concerns, the Negotiation and Contracting plugin will be utilised to facilitate agreements and meet these requirements.

[13] https://github.com/GetDKAN/dkan

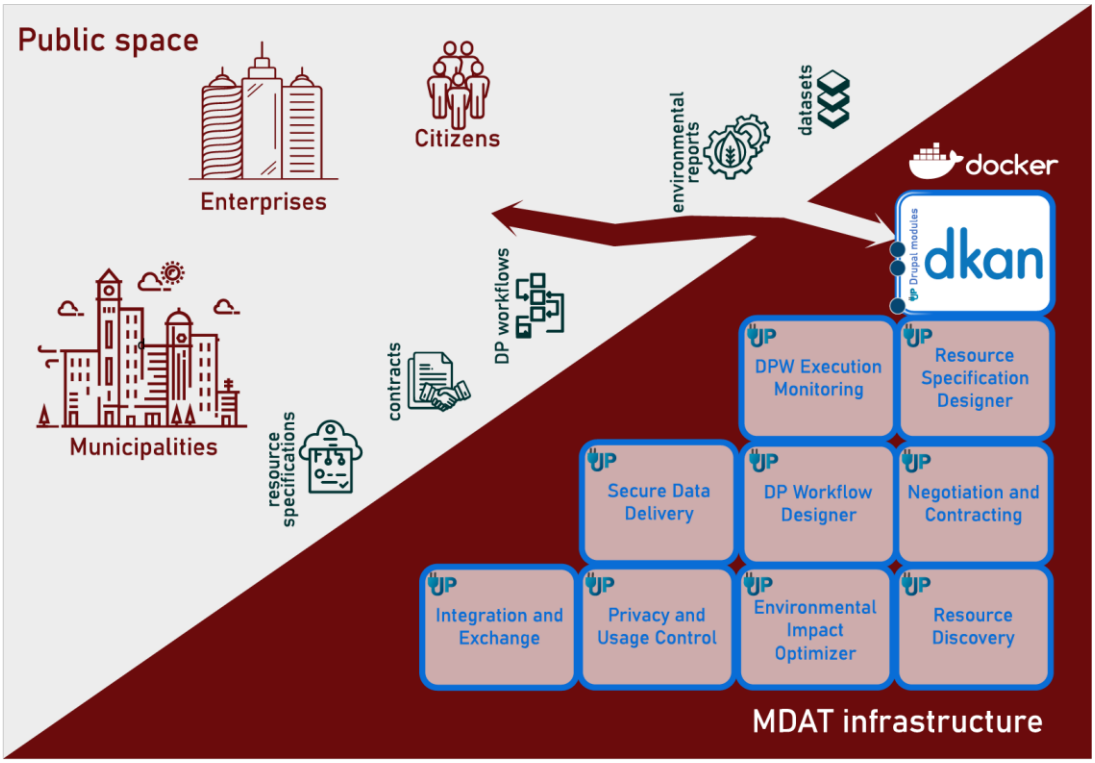

Figure 59: Deployment model for the Public Administration pilot.

7 PILOT DESIGN AND FUNCTIONALITY – HEALTH AND FITNESS

This pilot focuses on the valuation, sharing and trading of data streams related to health and fitness data. Such data is collected from wearables and IoT-enabled fitness equipment during physical activities. Nissatech is a company that operates fitness clubs and provides a technical system for real-time monitoring for fitness based on wearables and IoT environment.

7.1 Roles and Stakeholders

The main roles and stakeholders of this pilot are shown in Figure 60. Nissatech and trainees are providers of the health and fitness data. Service providers (e.g., gym) and product vendors (e.g., vendors of fitness equipment and supplements) are data consumers. This pilot also aims for an efficient and secure monetisation of such health and fitness data, and data trader is an important role for data monetisation. Data trader is an actor that sells data to realise data monetisation. Data trader can be a Data Provider or a Data Intermediary as defined by IDSA RAM 4.0. In line with the basic roles defined in IDSA RAM 4.0, a Data Creator is an actor that produces data and a Data Owner is an actor that executes control over data.

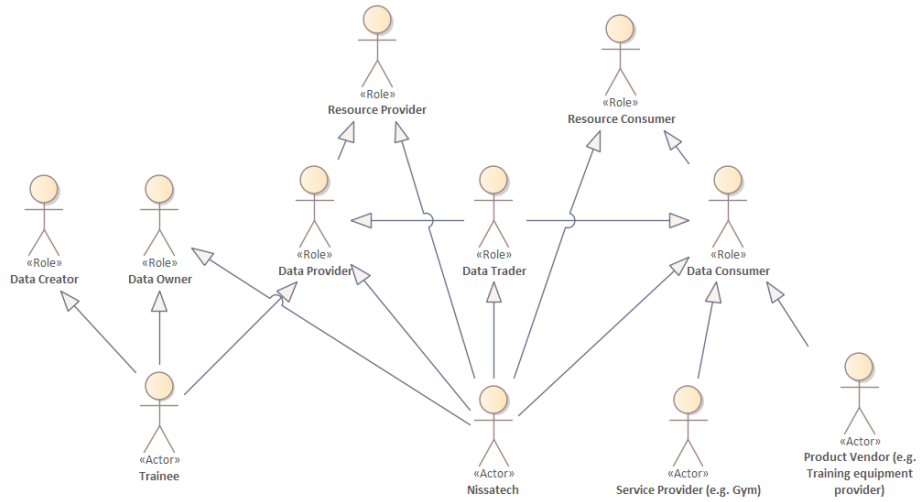

Figure 60: Roles and Stakeholders for the Health and Fitness pilot.

7.2 Reference Use Cases

Figure 61 shows the top-level use cases for the Health and Fitness pilot. The details for the use cases <<Establish collaboration with trainees>>, <<Bundle and prepare datasets>> and <<Trade data>> are shown in Figure 62, Figure 63 and Figure 64 respectively. For data monetisation, data providers (trainees) need to understand how the data is "valuable" for potential data consumers and how the data price is formed. For this purpose, the UPCAST Valuation plugin is used to valuate data contribution from trainees to a data product (<<Suggest dataset valuation>> use case) so that the data owners can decide if they want to do the monetisation and for how much as well as be motivated to generate as much as possible such data.

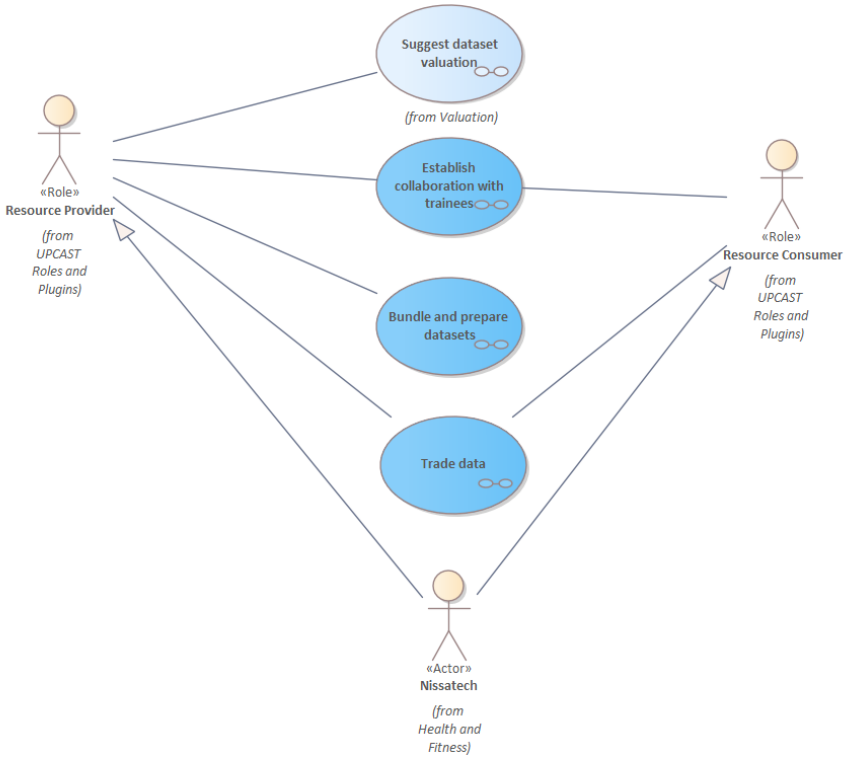

Figure 61: Top-level use case model for the Health and Fitness pilot.

For <<Establish collaboration with trainees>> use case (Figure 62): Nissatech will establish collaboration with trainees to collect and use their data. Data usage constraints and policies will be defined and negotiated to establish contracts. This process will ensure compliance with legal requirements, e.g., consent is given by the trainees when they share data, and the data shared is compliant with GDPR.

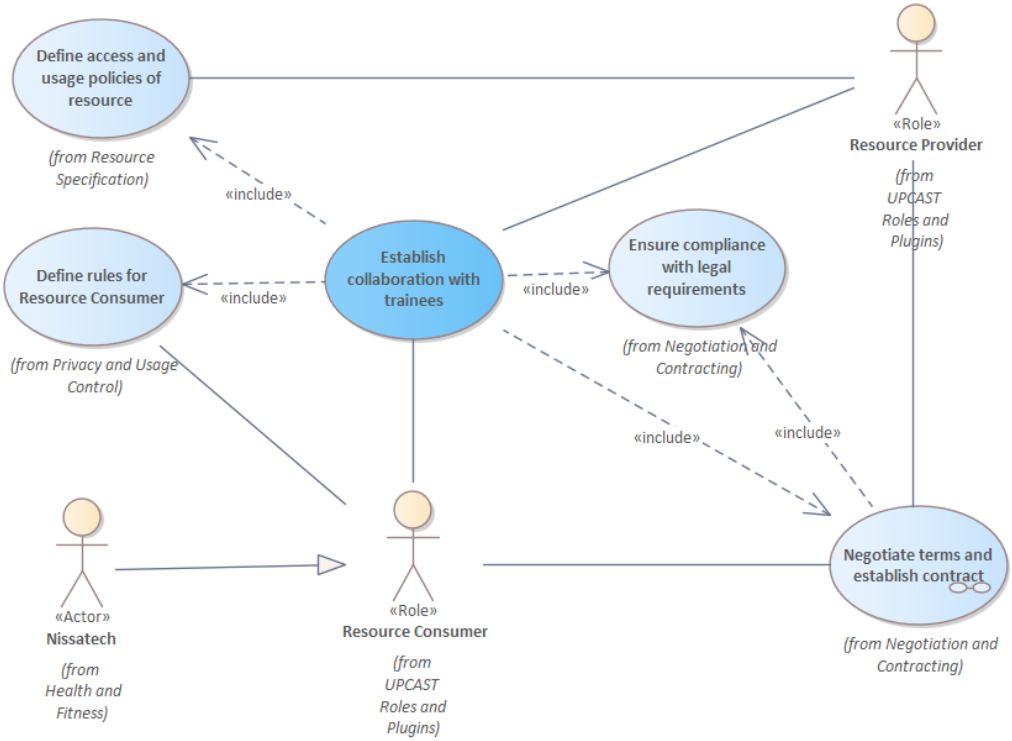

Figure 62: Establish collaboration with trainees.

For <<Bundle and prepare datasets>> use case (Figure 63): Nissatech will make some bundles of data to make a better offering. The new dataset will be described, and the environmental impact and the price of the dataset will be estimated.

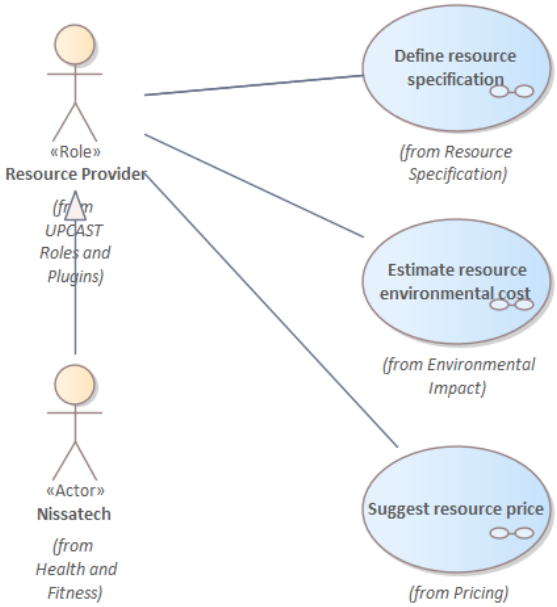

Figure 63: Bundle and prepare datasets.

For <<Trade data>> use case (Figure 64): the dataset to be traded will be published and monetised and transferred securely to the data consumer.

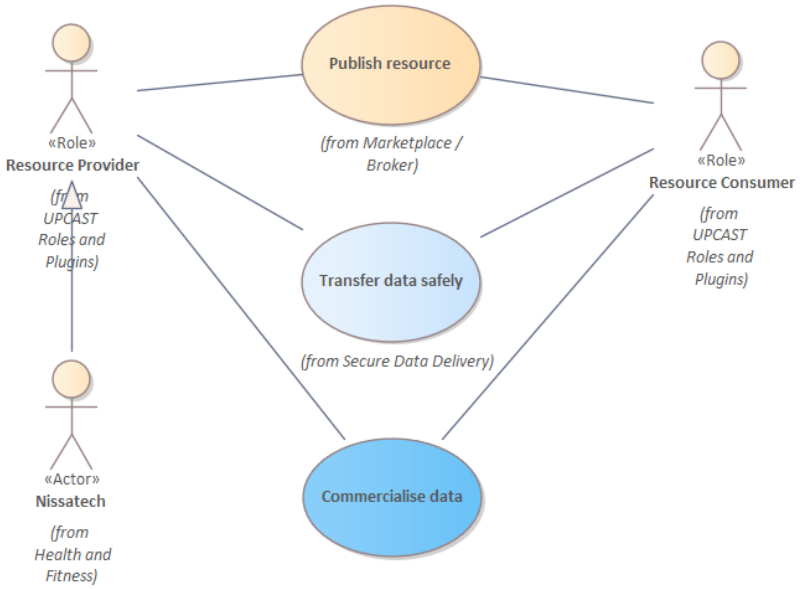

Figure 64: Trade data.

7.3 Business to Systems Mapping Model

Figure 65 shows how the top-level use cases of the pilot are implemented by the UPCAST plugins.

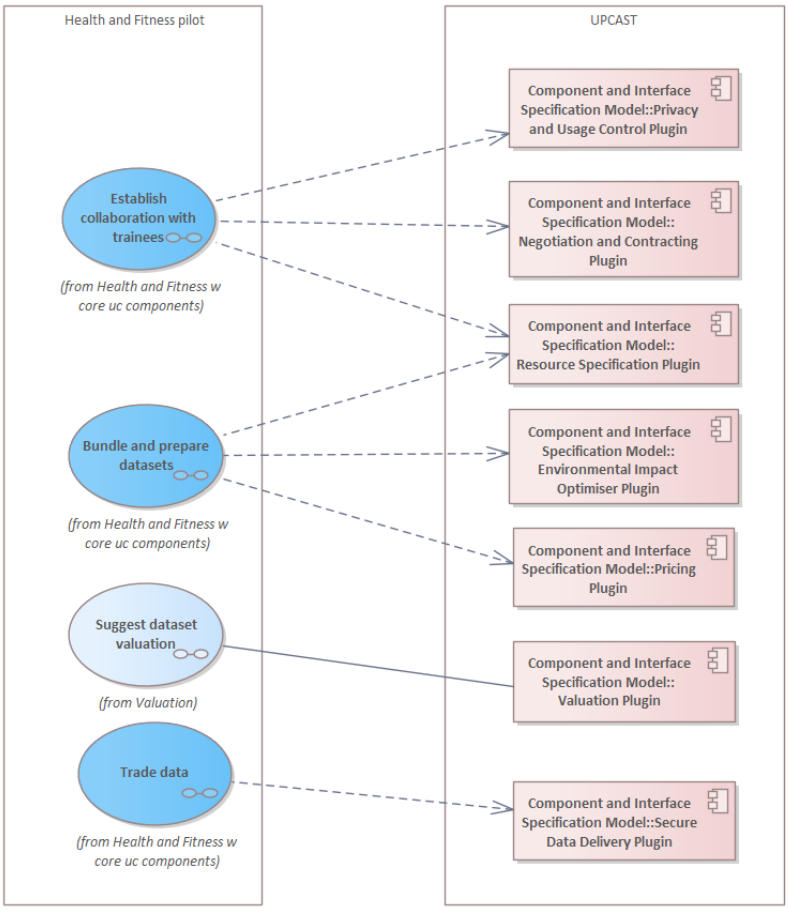

Figure 65: Business to Systems Mapping Model for the Health and Fitness pilot.

7.4 Pilot Implementation Plan

Figure 66 illustrates the deployment model for the Health and Fitness pilot. In this pilot data providers and data consumers activate UPCAST-enabled plugins.

In Nissatech's infrastructure, data is stored both on AWS servers for provider usage (overview of their own data – fitness data) and at a dedicated server after anonymisation for trading from where plugins are being activated.

Data providers have access only to their own data, and data consumers have access to data samples (before purchase).

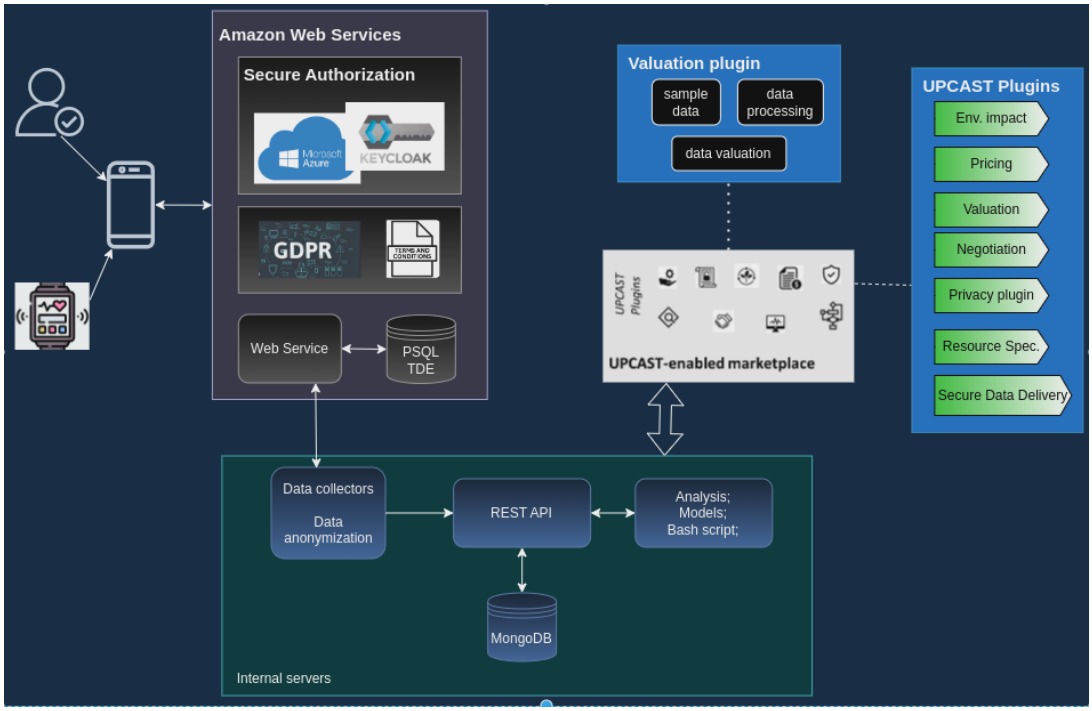

Figure 66: Deployment model for the Health and Fitness pilot.

8 PILOT DESIGN AND FUNCTIONALITY – DIGITAL MARKETING 1

In this pilot, JOT is offering market-related data to data consumers based on a new data-as-a service business model. Data consumers can decide which data is needed and how the data and insights should be delivered.

8.1 Roles and Stakeholders

As shown in Figure 67, JOT acts as both Data Provider and Service Provider to external Data Consumers (Resource Consumer).

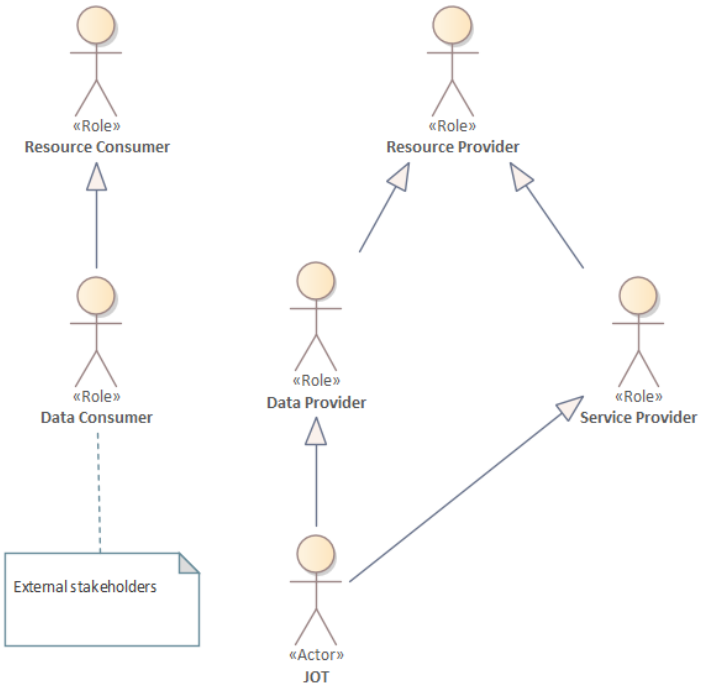

Figure 67: Roles and stakeholders for the Digital Marketing JOT pilot.

8.2 Reference Use Cases

The main use cases of this pilot are presented in Figure 68:

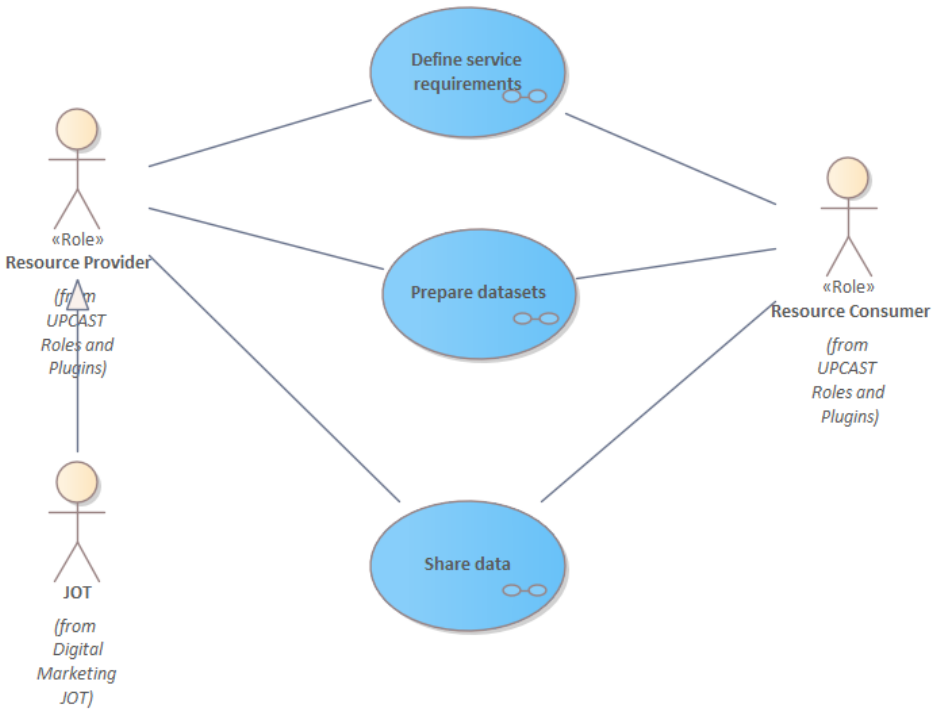

Figure 68. Top-level use case model for the Digital Marketing JOT pilot.

For <<Define service requirements>> (see Figure 69): The data consumer defines the type of service requested and expected price, and provides information about dataset needs and expected insights so that it can be possible to get value from the data and adjust the service to the budget. JOT will be able to define the service providing the datasets, including access and usage policies and rules for the data consumers. JOT may negotiate terms with data consumer and establish contract in this process.

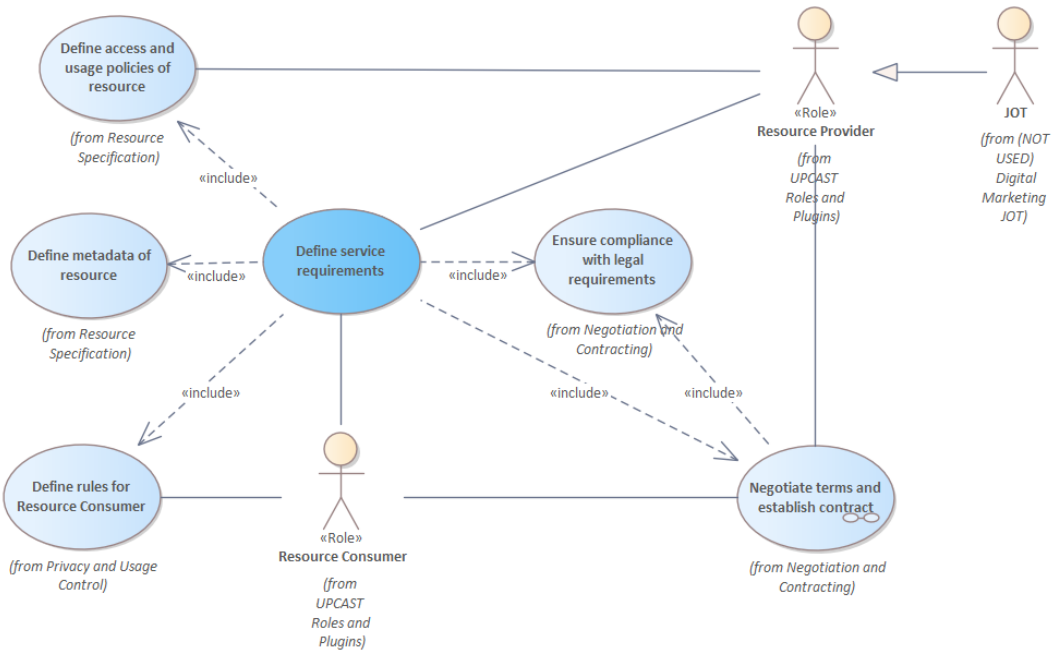

Figure 69. Define service requirements with Resource Consumer.

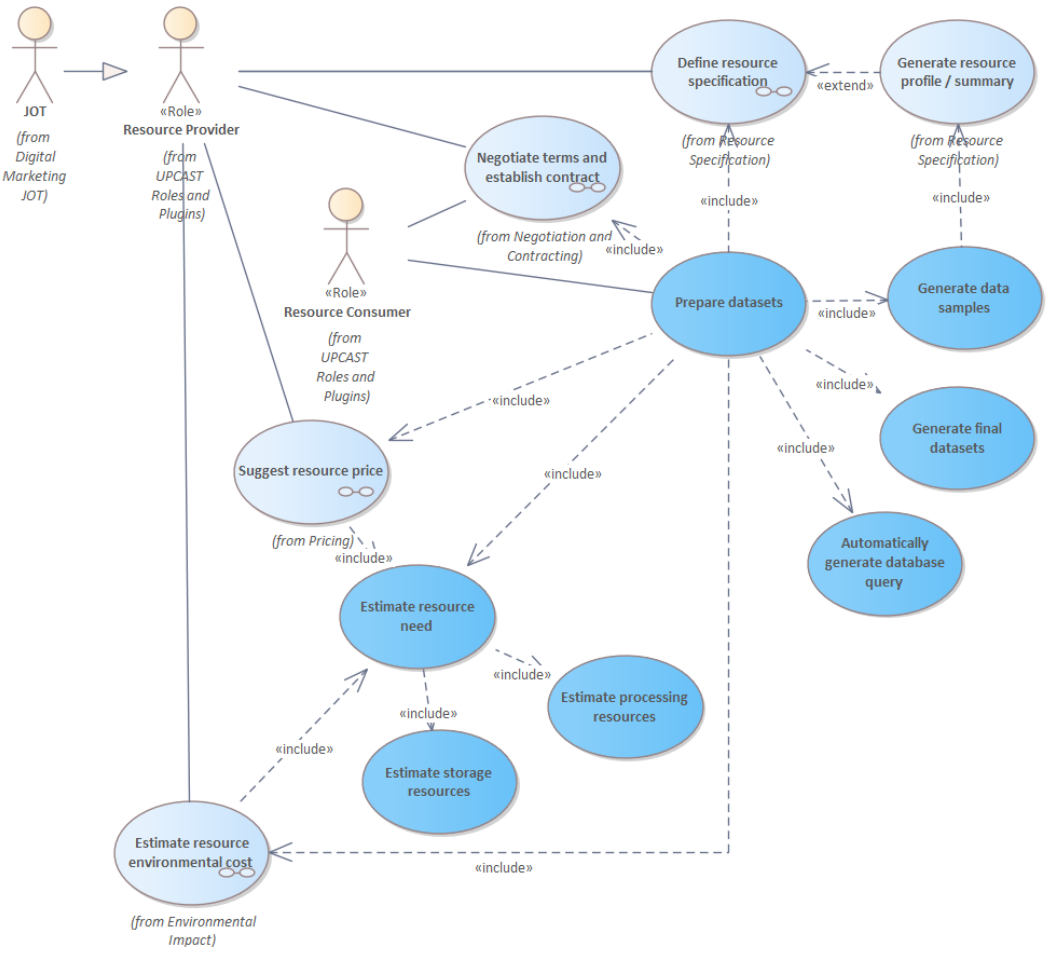

Figure 70. Prepare datasets.

For <<Prepare datasets>> (see Figure 70): JOT will generate data samples, which the Data Consumer can check and confirm the attributes and formats. The Data Consumer will define a service request based on a pre-defined set of filters and features. JOT will generate the complete dataset as requested by the Data Consumer, estimate the needed resources (processing and storage) and the customised price for the service requested, define and negotiate the terms of the contract, and sign the contract with the Data Consumer. JOT will also automatically generate database query based on the Data Consumer needs so that the dataset requests can be managed with no/minor manual actions.

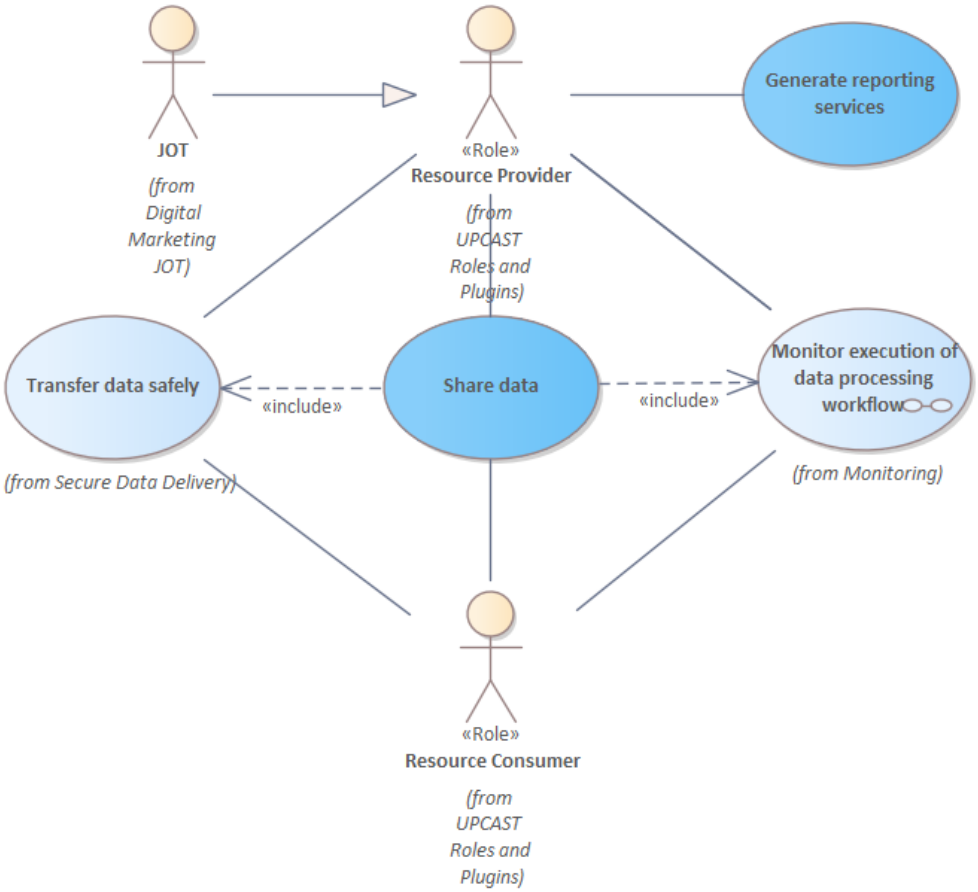

Figure 71. Share data with Resource Consumer.

For <<Share data>> (see Figure 71): This use case implements different services, including sharing data with Data Consumer using interactive and updatable dashboard and report generation based on selected KPIs.

8.3 Business to Systems Mapping Model

Figure 72 shows how the top-level use cases of the pilot are implemented by the UPCAST plugins.

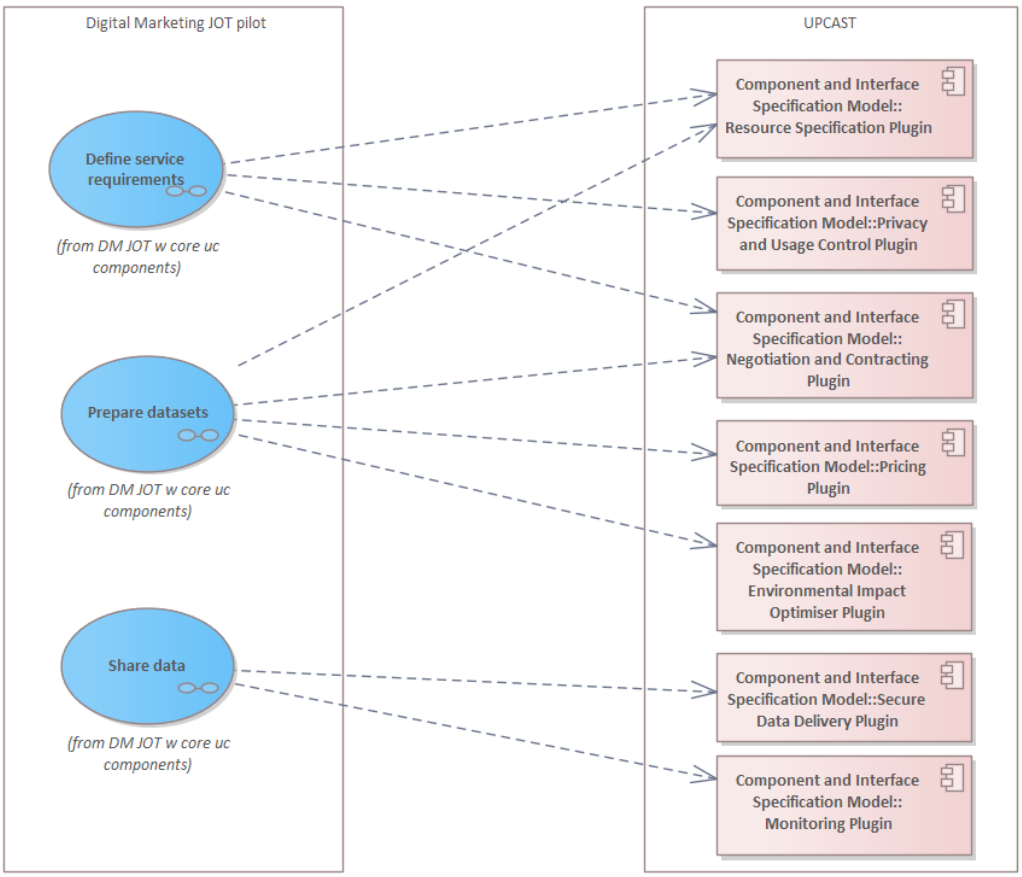

Figure 72: Business to Systems Mapping Model for the Digital Marketing JOT pilot.

8.4 Pilot Implementation Plan

Figure 73 shows the general architecture for the TO-BE scenario for this pilot and the main steps of the service flow in this TO-BE scenario. Figure 74 illustrates the technical integration of the UPCAST plugins in the service flow. The JOT marketplace is populated by the digital marketing performance data to identify user interests in any domain or business verticals. The data consumer will have access to the data catalogue as well as some examples in the web service, where it is also possible to define and request the proper datasets. This process will then launch the needed plugins via dedicated APIs for contract generation, workflow, resource specifications and so on.

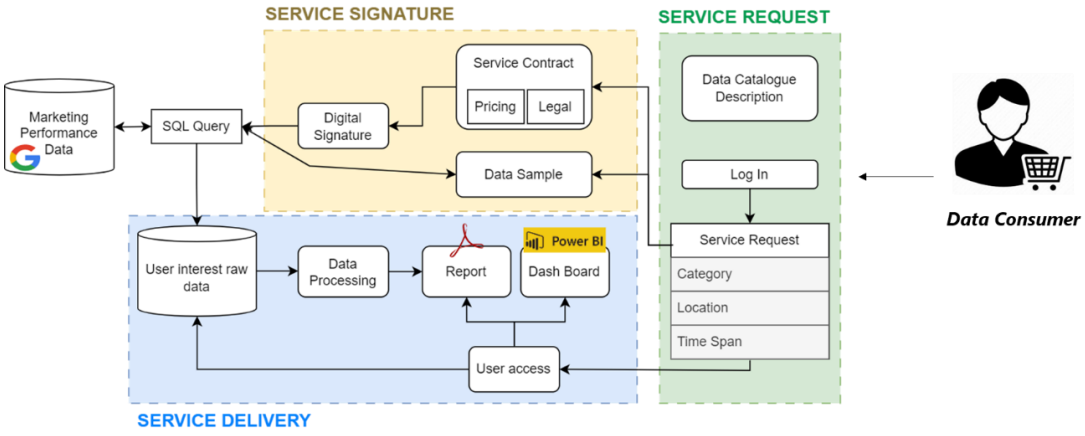

Figure 73: Generic architecture for the TO-BE scenario (Figure 10 from D1.1).

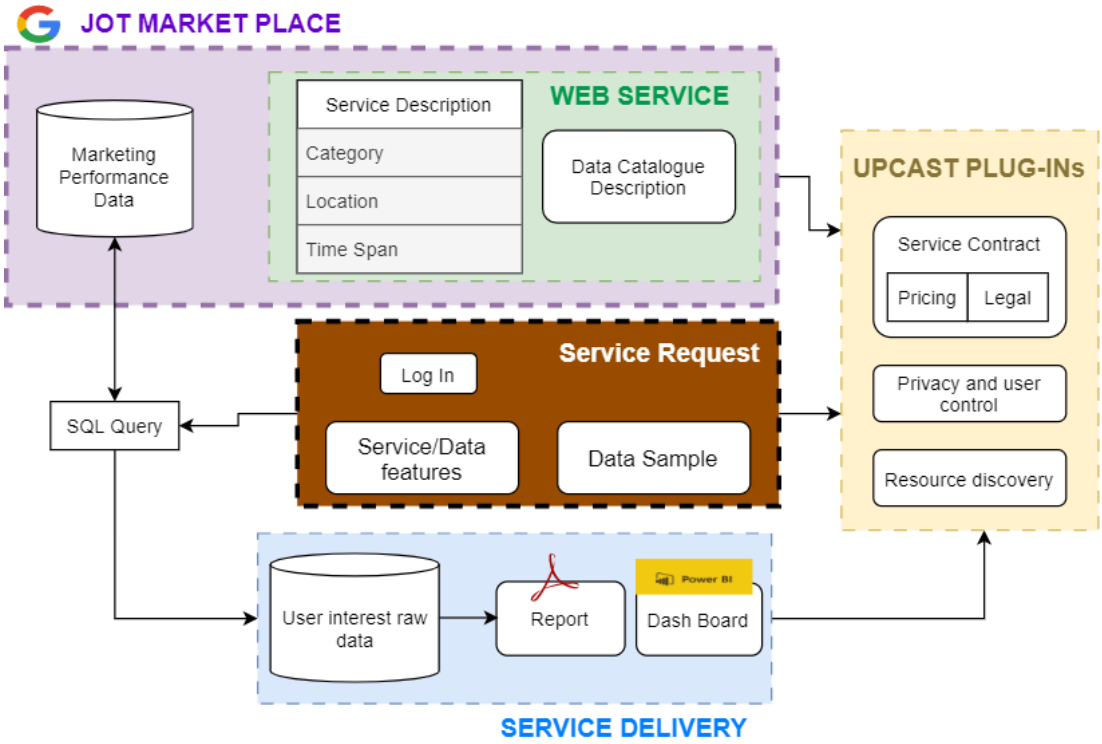

Figure 74: Deployment model for the Digital Marketing JOT pilot.

9 PILOT DESIGN AND FUNCTIONALITY – DIGITAL MARKETING 2

This pilot focuses on data sharing between Cactus (a technology company specialising in digital marketing and web development) and its clients. Cactus utilises client data to provide optimal digital marketing tools tailored to individual clients and develop marketing strategies aligned with the client's overall business goals. Cactus also shares data (e.g., competitive intelligence reports) with competitors/partners to help them identify areas for improvement.

9.1 Roles and Stakeholders

As shown in Figure 75, Cactus acts as both Resource Consumer (obtaining data from clients and competitors/partners) and Resource Provider (sharing data with clients and competitors).

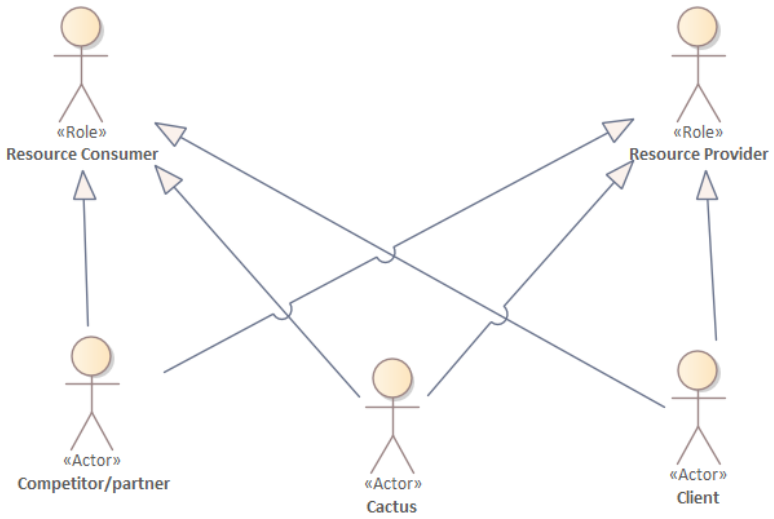

Figure 75: Roles and stakeholders for the Digital Marketing Cactus pilot.

9.2 Reference Use Cases

The top-level use case model for this pilot is shown in Figure 76.

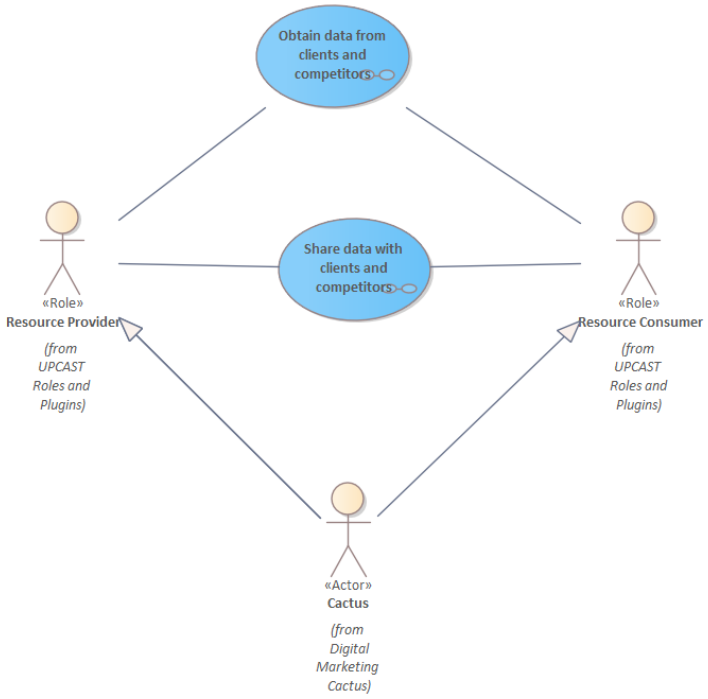

Figure 76. Top-level use case model for the Digital Marketing Cactus pilot.

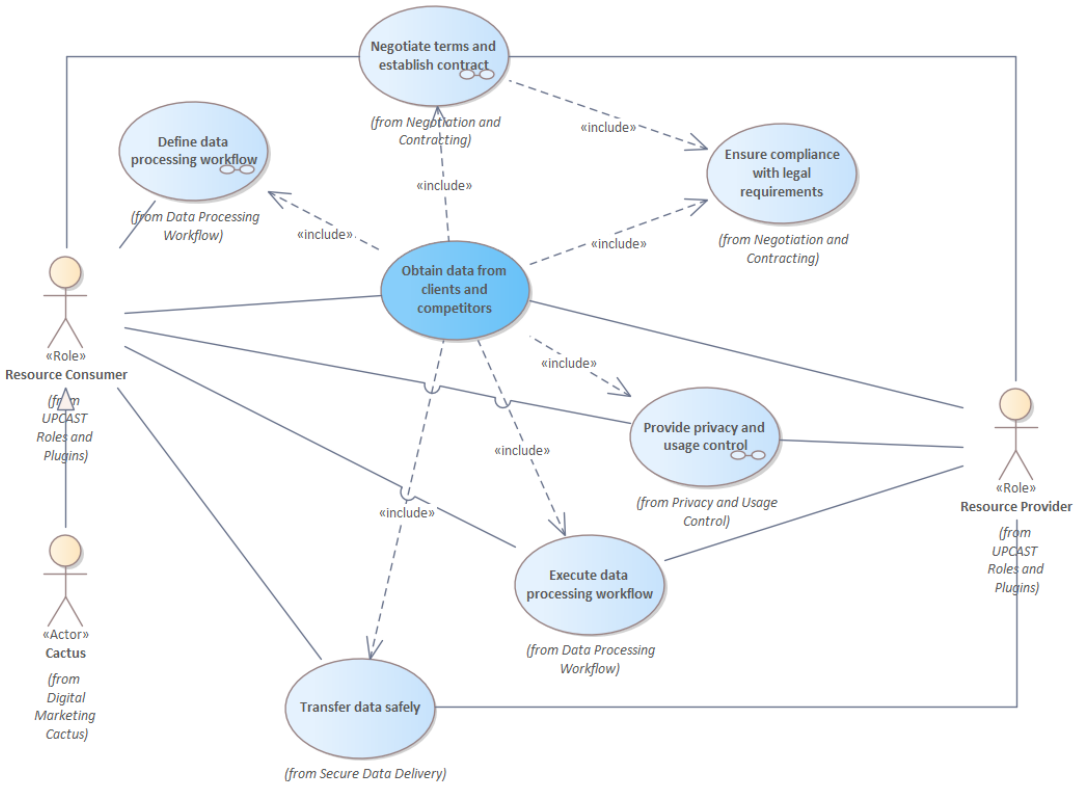

Figure 77. Obtain data from clients and competitors.

For <<Obtain data from clients and competitors>> (Figure 77): Cactus will negotiate terms with potential clients and obtain account information to gain access to the data from clients and competitors. Cactus will define data processing workflows for the digital marketing tools based on the data, negotiate the terms and establish the

contracts. This process will also define the usage and access control and ensure the compliance with legal requirements. The defined data processing workflows will be executed, and secure data transfer will be implemented to share clients/competitors data with Cactus.

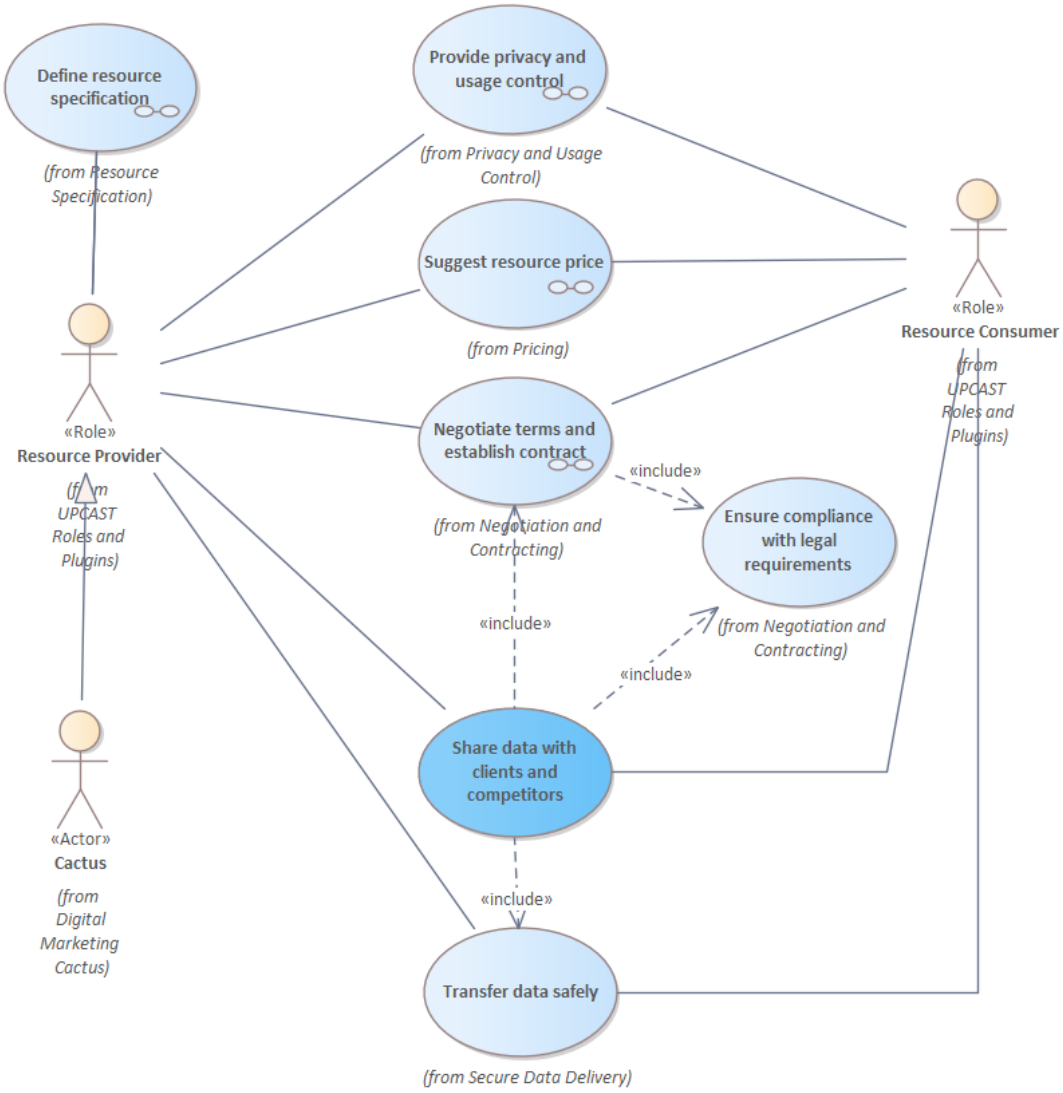

Figure 78. Share data with clients and competitors.

For <<Share data with clients and competitors>> (Figure 78): Cactus is the Resource Provider (offering data to clients and competitors). The sharing of data includes specification of the dataset to be shared, estimation of the dataset price, access and usage control, legal assessment, negotiation of contracts, and secure data transfer.

9.3 Business to Systems Mapping Model

Figure 79 shows how the top-level use cases of the pilot are implemented by the UPCAST plugins.

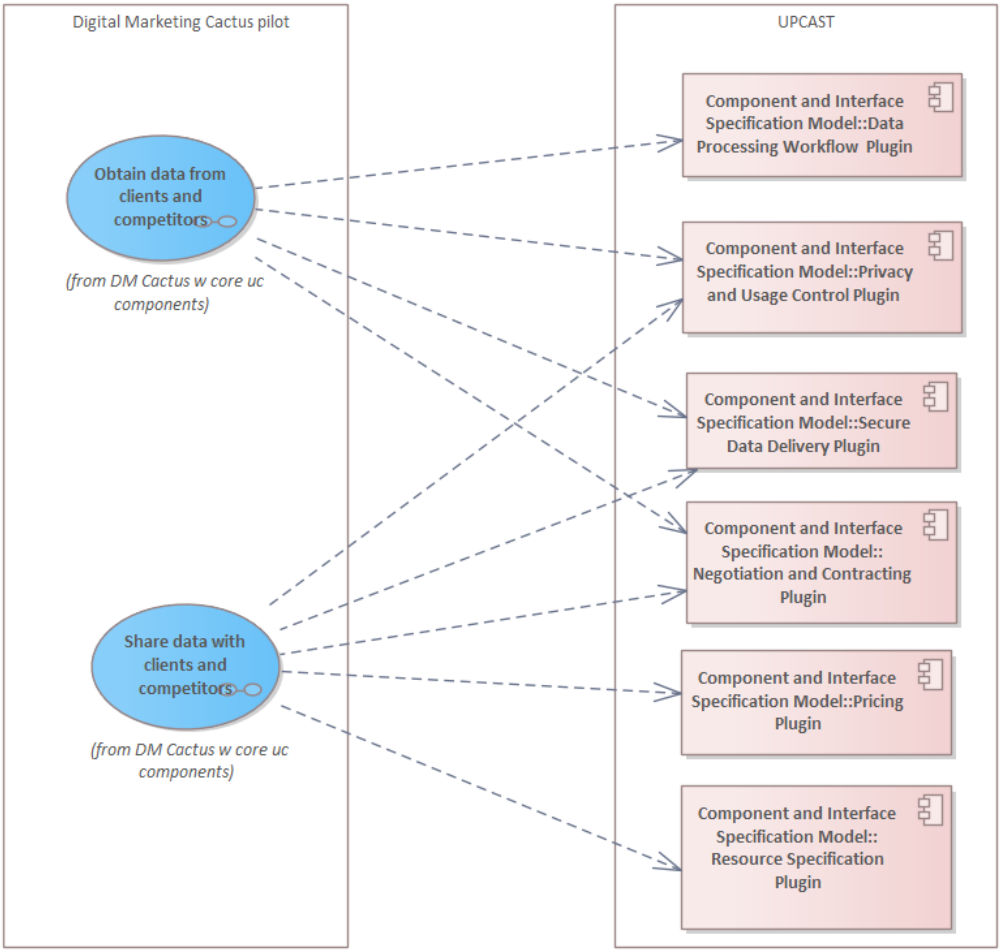

Figure 79: Business to Systems Mapping Model for the Digital Marketing Cactus pilot.

9.4 Pilot Implementation Plan

Cactus obtains data from clients, primarily from Google and Meta Cloud, creates digital marketing tools based on data processing workflows. Cactus offers an online platform with user-friendly interfaces to share data with clients, which allows the clients to control the actions performed within the Cactus platform. They also sell data to competitors for performance improvement. Figure 80 shows the deployment plan for the pilot. The UPCAST plugins that will be utilized in the pilot will be deployed in an UPCAST-enabled marketplace and provide the needed functionality for the pilot.

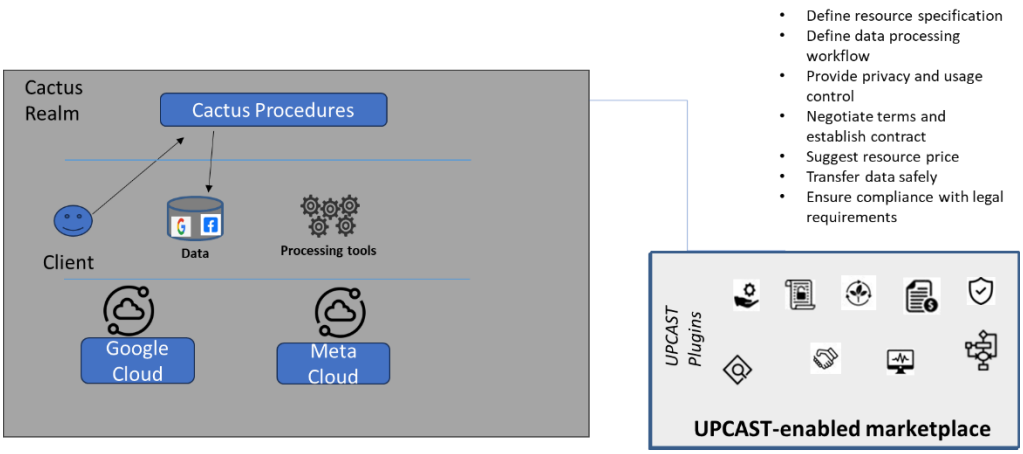

Figure 80: Deployment model for the Digital Marketing Cactus pilot.

10 CONCLUSION AND FUTURE WORK

This report presents the main features of the UPCAST MVP, the initial conceptual and technical architecture for the MVP development, the elaborated pilot design and functionality for demonstration of the MVP, and the initial input related to the vocabulary and data model to be used in UPCAST MVP and pilots. The ARCADE framework is used to design and document the initial UPCAST architecture and pilot design. This provides a technical specification and guideline for the implementation of the MVP and pilot use cases in the following project period.

Compared to D1.1, the following changes have been made in the organisation of plugins in this version of UPCAST architecture:

- A new Federated Machine Learning plugin has been defined to provide Federated Machine Learning functionality.
- The original Valuation and Pricing plugin has been separated into Pricing plugin and Valuation plugin as they cover two different functionalities.
- The original Safety and Security plugin has been changed to Secure Data Delivery to better reflect its intended functionality.
- The Legal Assessment horizontal service will consist of a human related part (guidelines and policies to check) and a software-assisted automated part. The automated part is considered as a function to be performed by the Negotiation and Contracting plugin to ensure compliance with legal requirements.

This deliverable is an initial architecture specification and an initial input to the vocabulary and data model. Some technical details (such as the detailed methods defined for component interfaces) need to be clarified and further elaborated. The following future work is planned to finalise the architecture, vocabularies and pilot demonstration:

- Secure Data Delivery plugin: Only generic functionality is defined. Details on further design and interface will be provided together with the new project partner who will take over the responsibilities of this plugin.
- Federated Machine Learning plugin: Future work is needed to find out how this plugin will be used and demonstrated in pilots.
- UPCAST technical architecture: UPCAST will implement the architecture depicted in Figure 44 where data provider and consumers interact with marketplace (or a data sharing platform in general), and UPCAST plugins are deployed on the marketplace. Feedback will be gathered from pilot demonstration to improve the initial architecture.
- Vocabulary and data model: Each plugin team will lead further iterations of their corresponding data model facet hand in hand with the technical development of the plugin. Technical coordinator will ensure alignment in monthly technical meetings. We will liaise with Coordination and Support Action to evaluate if Data Models currently being developed by other ongoing projects are useful for ours.

11 ACRONYMS

Table 25: Acronyms.

Acronyms List	
DoA	Description of Action
DPW	Data Processing Workflow
DSL	Domain-Specific Language
EIO	Environmental Impact Optimiser
MVP	Minimum Viable Product
PDP	Policy Decision Point
PUC	Privacy and Usage Control
RC	Resource Consumer
RP	Resource Provider
UI	User Interface
XAI	Explainable AI