Brokering architecture: the ODIP prototype model views

Date	2018-05-27
Version	1.1
Authors	Enrico Boldrini, Fabrizio Papeschi, Mattia Santoro, Roberto Roncella, Massimiliano Olivieri, Paolo Mazzetti and Stefano Nativi
Organization	CNR - Institute of Atmospheric Pollution Research, Florence Division
Document type	Technical Specification
Status	Draft
Dissemination	Public
Abstract	Brokering architecture: the ODIP model views
Keywords	ODIP, Brokering services, architecture design
Contact info	enrico.boldrini@cnr.it
URN	http://essi-lab.eu/GI-suite/ODIP/BrokeringFramework

This document has been drafted in the context of ODIP. ODIP has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no [312492] and continued funding from the European Union's HORIZON 2020 Framework Programme for Research and Innovation under grant agreement no [654310].

Document Revisions Version	Date	Status	Author(s)	Description
1.0.0	2018-05-14	First version	Boldrini	First draft
1.1.0	2018-05-27		Nativi	Revision

SUMMARY

INTRODUCTION

Scope

The rationale of this document is to create a formal documentation of the Brokering architecture, proven successful in ODIP prototype 1+, as well as in several other real-world and prototype use cases, to serve as a best practice. Wider recognition and interest in the functionalities of this middleware (which is transparent by its nature) are expected as an outcome of this publication. ODIP Prototype 1+ (in the context of the Ocean Data Interoperability Platform Coordination and Support Action of the EU Research Infrastructures programme) aims at implementing interoperability between the following autonomous and distributed systems: SeaDataNet CDI, US NODC, and IMOS MCP. The prototype demonstrates data discovery and access services using a brokering middleware that is utilized by a couple of Web-portals:the global IODE-ODP and GEOSS portal.

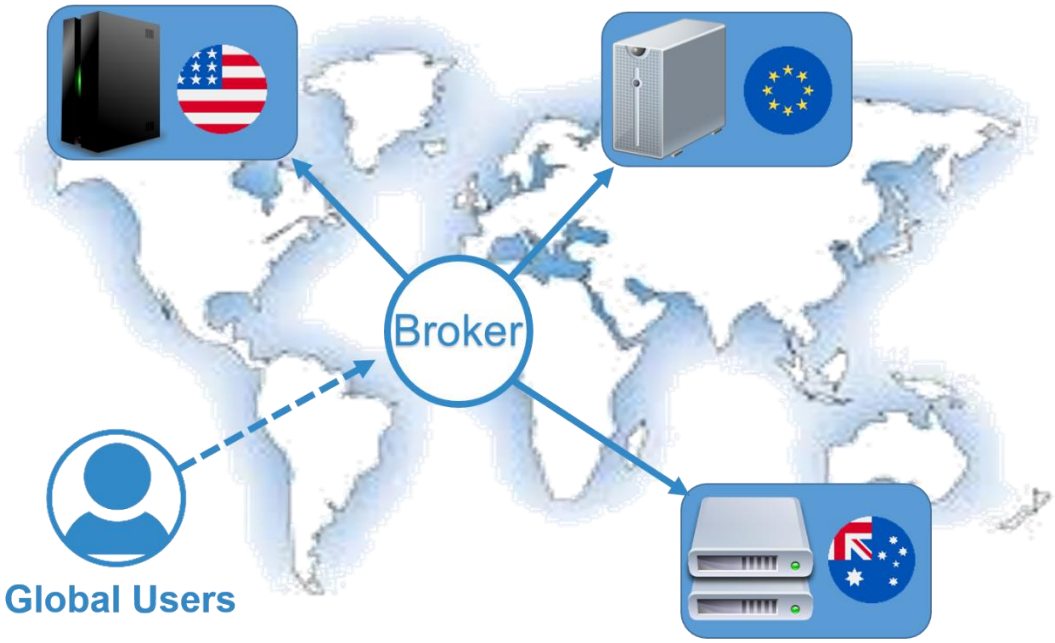

Figure 1 ODIP prototype 1+

Objectives

The objective of this technical specification is to formally describe the architecture of a multi-organizational brokering system, focusing on the ODIP broker system implemented for the ODIP prototype 1+.

This specification describes the adopted brokering platform according to the RM-ODP viewpoints framework formalism, in which different views are used to represent the whole system from the perspective of a related set of concerns.

Technical specification model

A viewpoints framework formalism, such as the ISO Reference Model of Open Distributed Processing (RM-ODP), is commonly used for formally describing complex distributed software systems according to simpler transversal viewpoints. In such formalism, different views are used to represent the whole system from the perspective of related set of concerns.

The *RM-ODP* is a well-known viewpoints framework formalism compliant with IEEE 1471 freely available as ISO standard [1]. Both RM-ODP and the UML4ODP [2] standards have been taken into account in drafting this document.

To the aim of describing the ODIP brokering framework the following viewpoints will be considered, each one composed by a set of related concerns:

- **Enterprise:** purpose, scope and policies governing the activities of the specified system within the organization of which it is a part;
- **Information:** the kinds of information handled by the system and constraints on the use and interpretation of that information;
- **Computational:** the functional decomposition of the system into a set of objects that interact at interfaces – enabling system distribution;
- **Engineering:** the infrastructure required to support system distribution;
- **Technology:** the choice of technology to support system distribution.

ENTERPRISE VIEW

The enterprise view is used to model the purpose, scope and policies governing ODIP brokering prototype. The specification tries to formalize the actors, requirements and objects extracting and refining them from the real-world prototype. This view describes both the high level interworking of ODIP broker prototype, and the high-level enterprise patterns for future similar real-world contexts.

ODIP Communities

The **ODIP broker** system **enterprise specification** represents the different Communities composing the ODIP prototype ecosystem. Each Community is characterized by its specific role, policy and enterprise objects). The recognized Communities are:

- **Regional marine communities**: one community for each ODIP participating organization (e.g. SeaDataNet, AODN IMOS, NODC, …), each community including at least actors with roles of point of contact, developers of data services, developers of semantics services, developers of community portals, community users
- **International data communities**: one community for each international program of interest (e.g. IODE, GEOSS), each community including at least actors with roles of point of contact, portal developers, portal users
- **ODIP Broker community**: it includes broker point of contact, administrators and developers
- **Steering community:** it includes ODIP prototype 1+ principal investigator, technical committee with members from the other communities

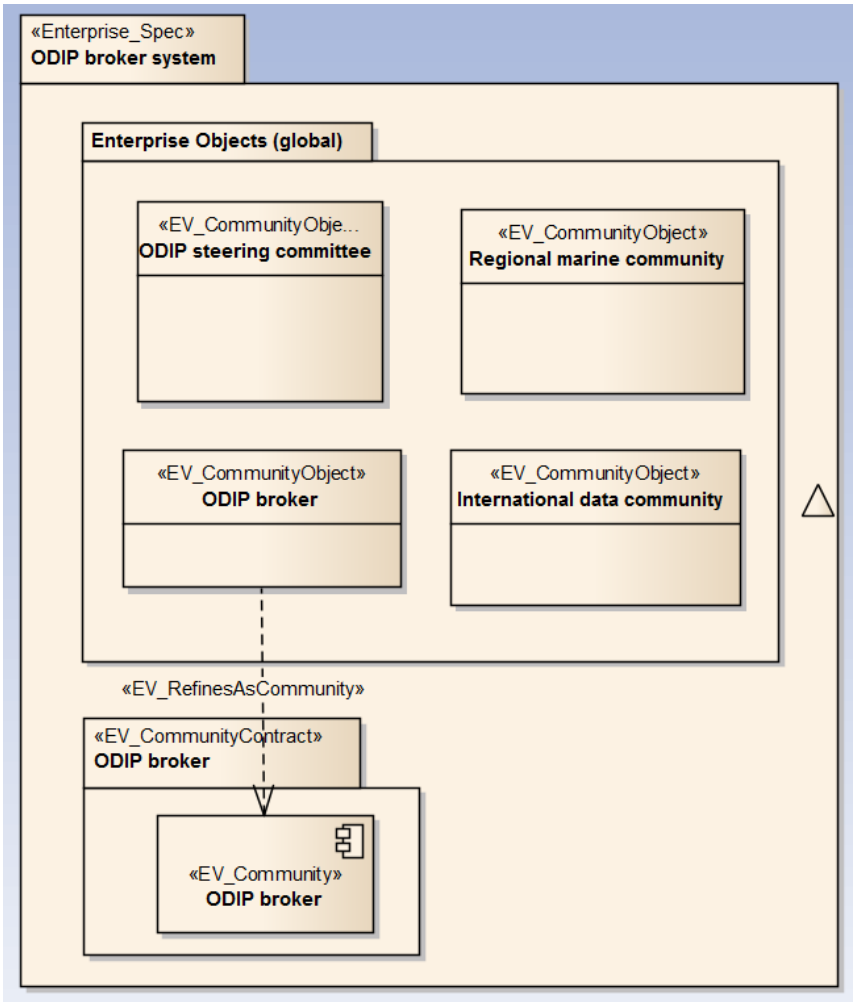

Figure 2 Enterprise specification of the ODIP prototype 1+

As depicted in Figure 2, different communities take part in the system, because of its distributed and multi-organizational nature, involving several heterogeneous systems..

Each community is characterized by its: stakeholder roles, enterprise objects, and policies. Besides, a Community has a whole community objective to be considered for the brokering success –in the specification, this is formalized by one class (stereotyped <<EV_Objective>>) which has a tagged value that express the community objective.

This document covers the ODIP broker Community and the actors from the other Communities that interact with it –see Figure 3.

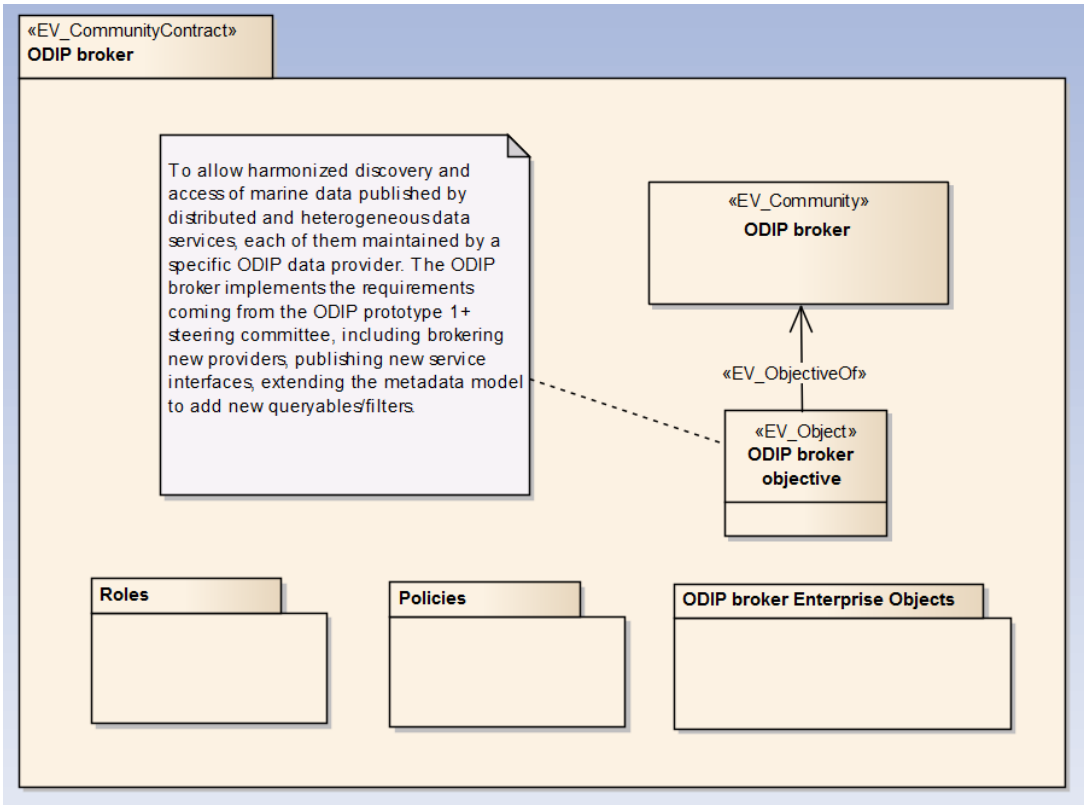

Figure 3 ODIP broker community

ODIP Broker Community

ODIP broker community general objective is to allow harmonized discovery and access of marine data published by distributed and heterogeneous data services, each of them maintained by a specific ODIP data provider.

The ODIP broker implements the requirements coming from the ODIP prototype 1+ steering committee, including brokering new data/semantics providers, publishing new service interfaces, extending the metadata model to add new queryables/filters and so on.

ODIP broker processes

Processes specify the behavior in terms of (partially ordered) sets of steps, and are related to achieving some particular sub objective within the community.

The processes of the ODIP broker community are expressed by a set of activities stereotyped as <<EV_Process>> that have the component that expresses it as their context, as shown in Figure 4.

The source for these requirements are the ODIP II deliverable D3.1 [3] and they include:

- **Harmonized discovery**: seamlessly query heterogeneous sources of marine information and obtain results in a common data model

- **Semantically enhanced discovery**: obtain enhanced query results by expanding user query terms from regional marine vocabularies recurring to a semantics service (i.e. Rosetta Stone web service). Required vocabularies to support, at least:
    - SDN:P02
    - SDN:EDMO
    - AODN:parameter
    - AODN:platform
    - NERC:P01
    - NODC:datatype
    - NODC:platform
- **Filters/Facets discovery**: faceted search consists in presenting the actual values documenting a specific metadata element in a set of resources, to the aim of having the user select one of the values to act as a result set filter
- **Paging**: to browse big results sets page by page
- **Ranking metrics**: to have results order by importance (definition of importance based as a customizable formula, dependent on query matching score and quality of results score)
- **Broker new data source**: to add a regional data service as an additional source of discovery and access by means of the ODIP broker. Required sources at least:
    - AODN IMOS (CSW/ISO-MCP service protocol)
    - US NODC (CSW/ISO-NODC service protocol)
    - SeaDataNet (ISO-CDI service protocol)
- **Add discovery/access service interface**: to publish an additional standard discovery or access service by the ODIP broker. Required at least the ones needed to connect with:
    - GEOSS portal
    - IODE ODP portal
- **Broker ontology provider**: to access semantics capabilities of a given semantics service (e.g. Rosetta Stone semantics capabilities)
- **Harmonized access**: seamlessly access heterogeneous marine data sources to download data in common standard formats and having it transformed by means of simple transformations (e.g. data format conversion, crs reprojection, interpolation, subsetting)
- **Add simple transformation**: to add a specific simple transformation (e.g. EPSG:4326 to EPSG:3857)
- **Add queryable metadata element**: to make a specific metadata element to act as a queryable (to use it for discovery). Required to be supported at least:
    - Instruments (how)
    - Platforms (how)
    - Data originators (who)
    - Bounding box (where)
    - Time interval (when)
- **Add filter metadata element**: to make a specific metadata element to act as a filter for the faceted search. Required to be supported at least:
    - Instruments (how)
    - Platforms (how)

- Data originators (who)
- Bounding box (where)
- Time interval (when)

- **Connect through interoperability API**: to make available an Application Programming Interface to make it easy for a web portal developer to connect to ODIP broker functionalities

From this list of processes, other important system requirements emerge.

- **Flexibility**: to be able to support existing and emerging standard
- **Modularity**: to easily support extensions through additional modules (opposite to a *monolithic* architecture)
- **Rich and extensible data model**: to accommodate most common information (and also community specific and additional information)

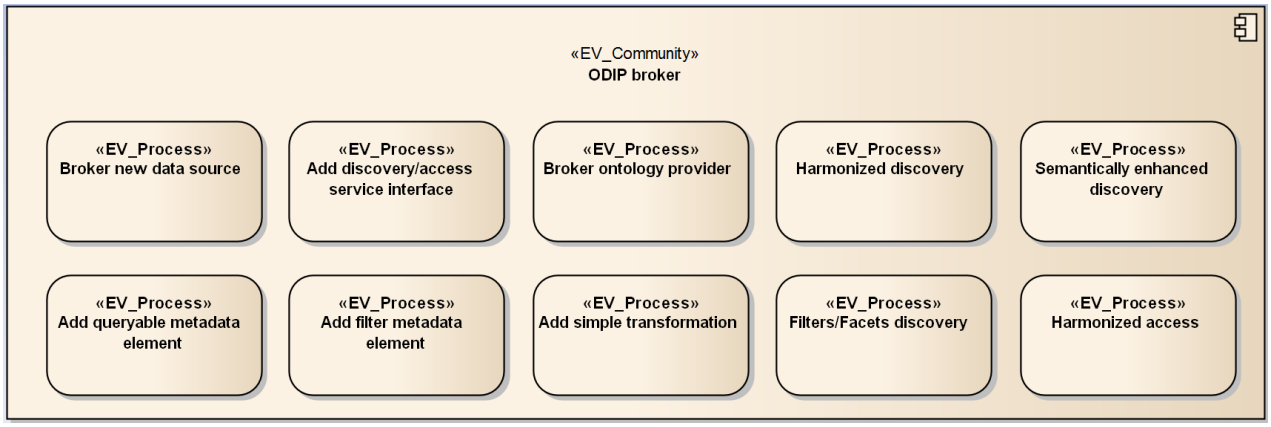

Figure 4 ODIP broker processes

Each of these activities it is associated to an Activity Diagram expressing the process steps and identifying the different roles (either as actor or as artefact roles). For instance, the basic process to **Broker a new data source** is detailed in Figure 5.

In the **Broker a new data source** process, the behavior of the **ODIP broker** role is defined by the actions in the activityPartition for the **ODIP broker** role. While, the complete behavior of the **ODIP broker** role is the composition of its behaviors in all of the processes in which it is involved.

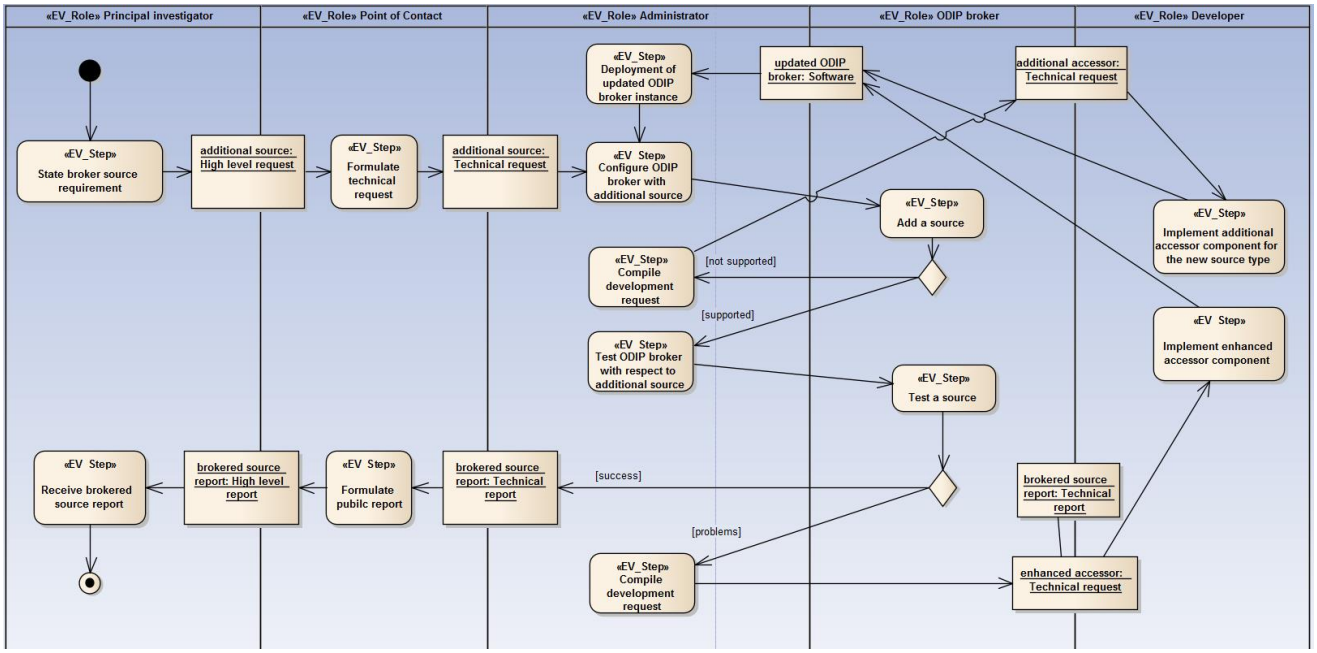

Figure 5 Broker a new data source process

The process starts with the ODIP **Principal investigator** performing the step **State broker source requirement.** This step is commonly triggered by previous steps occurred in the other community activities, such as:

- directly taking as an input a requirement expressed in ODIP deliverable D3.1 [3]
- regional data provider had requested to be part of ODIP broker prototype, by applying with a data service to be brokered; the ODIP steering committee had to accept the request and as a consequence the PI formulates the broker new source requirement.

This first step implies that a **High level request** (enterprise object) has come into existence, and this fact is modelled by an artefact of **High level request** expressed as an objectFlow, named in the model **additional source** which has type **High level request**.

The **Point of contact** (a role filled by a member of the ODIP broker community), next performs the step **Formulate technical request**, which references, as an artefact, the enterprise object **Technical request** (named **additional source**), which is a translation, in the technical language spoken by the ODIP broker community, of the first request.

The remainder of Figure 5 is largely self-explanatory and is not detailed further in textual form, except for the general idea: the **Administrator** steps include re-deployment of software updates coming from the **Developer(s)**, configuration and testing of the **ODIP broker**, compiling development requests for enhancement of the **ODIP broker** system.

The **Point of contact** role acts as a gateway between the ODIP broker community and the others, translating incoming requests and out coming reports from a technical language to a high level

language. A single actor can as well play both the **Point of Contact** and the **Administrator** roles in the real-world case.

The **Developer** steps with regards to this process include implementing additional/enhanced accessor components.

ODIP broker roles

Figure 6 shows ODIP broker roles within the package that contains the specification of the community, associated to a realization link to the component that expresses the community. The behavior identified by a role is expressed by the set of behavioral features of the class that expresses the role. Example given, it follows a (partial) list of behavioral feature for the identified roles in the **ODIP broker** community:

- **Point of contact**
  - **receives requests:** the ODIP prototype 1+ steering committee can formulate requests to the ODIP broker point of contact, e.g.:
    - brokering of new data/semantics providers
    - enhancement/fix of the current version of ODIP broker implementation can be requested as well

    The point of contact can accept (providing an estimated time for completion) or refuse a given request, based on technological implications
  - **delegates brokering request:** incoming requests are delegated to the broker administrator for implementation
- **Administrator**
  - **deploys broker:** the administrator is responsible to deploy an instance of the ODIP broker system on a given infrastructure (e.g. to a cloud infrastructure)
  - **configures broker:** the administrator is responsible to configure the broker to implement an incoming request
  - **tests a configuration:** the administrator conducts tests to assure that the brokering has been successfully completed as expected
  - **delegates development request:** the administrator can delegate a request involving software development to developers
  - **provides support:** the administrator provides support to ODIP broker users
- **ODIP broker (system)**
  - **brokers source:** configure an accessor instance for brokering a specific source
  - **publishes interface:** publish the specific service interface
- **Developer(s)**
  - **implements enhancement/fix:** the developer is in charge of developing new ODIP broker software components or to patch existing ones

Some of the roles can of course be interpreted by the same actor (e.g. point of contact and administrator).

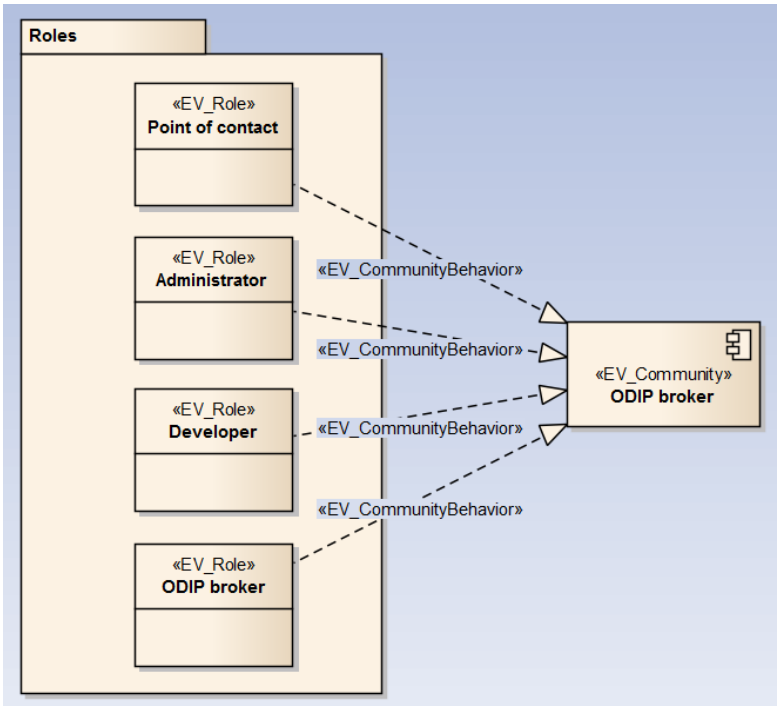

Figure 6 ODIP broker community roles

It is useful to highlight the more important roles existing in the other communities and interacting with the **ODIP broker** system or with the roles of its community.

From the **Steering committee** Community, the following roles are identified (along with a partial list of associated behavioral features):

- **Principal investigator**
  - **formulates high level requests:** the PI submits high level requests (e.g. broker a new source – from DOW) to the **Point of contact** in the **ODIP broker** community.
  - **receives high level reports:** the PI receive high level reports from the **Point of contact** in the **ODIP broker** community, as a result of previously submitted request

From the **Regional marine** Community, the following roles are identified (along with a partial list of associated behavioral features):

- **Data service** (system)
  - **receives harvest requests:** the ODIP broker submits harvesting request to inventory data provider services, in order to retrieve all the available metadata content in order to optimize subsequent user searches.
  - **receives discovery requests:** the ODIP broker submits discovery request to real time data provider services, in order to retrieve all the available metadata matching user searches at real time.
- **Semantics service** (system)

    - **receives semantics queries:** the ODIP broker submits semantics queries to semantics services (e.g. Rosetta Stone), in order to retrieve semantics related terms from a in input term selected by the user.
- **Community app** (system)
    - **submits discovery web requests:** a community app submits discovery web requests using protocols implemented by the community data services. The ODIP broker can transparently reply as well to such requests, as long as it is implementing the required service interface.
- **Community end user**
    - **searches through a community app**: a marine community end user is a person who discovers marine resources using a community application (and community defined vocabularies). He/she is not necessarily aware of the existence of the ODIP broker.

From the **International data** Community, the following roles are identified (along with a partial list of associated behavioral features):

- **Web portal** (system)
    - **submits discovery web requests:** the portal submits a discovery web request (determined by user interacting with the portal UI) to a discovery web service interface published by the **ODIP broker**.
- **Web portal end user**
    - **searches through the international portal**: an international portal end user discovers marine (but not only) resources from international data sources using an international portal. He/she is not necessarily aware of the existence of the ODIP broker.

INFORMATION VIEW

The information viewpoint deals with the system information modelling. This information specification defines the semantics of information and the semantics of information processing in the ODIP brokering prototype, independently from: its implementation, the computational process, and the nature of the distributed architecture used.

Figure 7 shows the full information specification of the brokering system, consisting of several interconnected packages that contain sets of information objects. The information packages are:

- **Core**: common objects handled by the brokering system, such as: **Resource, Resources Collection, Source, RelationType**
- **Query & View**: needed to realize discovery and views, such as: **User**, and **Query** (including **Constraint**), **Request, View**
- **ResultSet**: results of discovery, such as: **ResultSet, CountSet, ElementValueFrequency**
- **Metadata**: describing resources, such as: **MetadataElement** (**Core, Augmented, Extended), OriginalMetadata, Identifiers**
- **Semantics**: semantics related objects, such as: **Ontology**
- **Service**: to model different geo information services, such as: **Service (Access, Discovery, Processing)**
- **O&M**: to model concepts from Observation and Measurement model, such as: **FOI, Observation, Sensor**
- **Dataset**: to handle information needed to realize access, such as: **Dataset, Encoding, Thumbnail, Variable**
- **BP**: to handle information for the execution of business processes, such as: **BP, Workflow, EnvironmentalModel**
- **Document**: to handle descriptive resources

Some of these packages contains information objects needed to implement specific enterprise requirements that are beyond the ODIP experimentation (e.g. Business Processes management, Documents management packages).

For the scope of this document, this view focuses on the semantics (i.e. main information aspects) characterizing the elements identified in the enterprise view (see the Enterprise view):

- Metadata elements schema
- Data elements schema

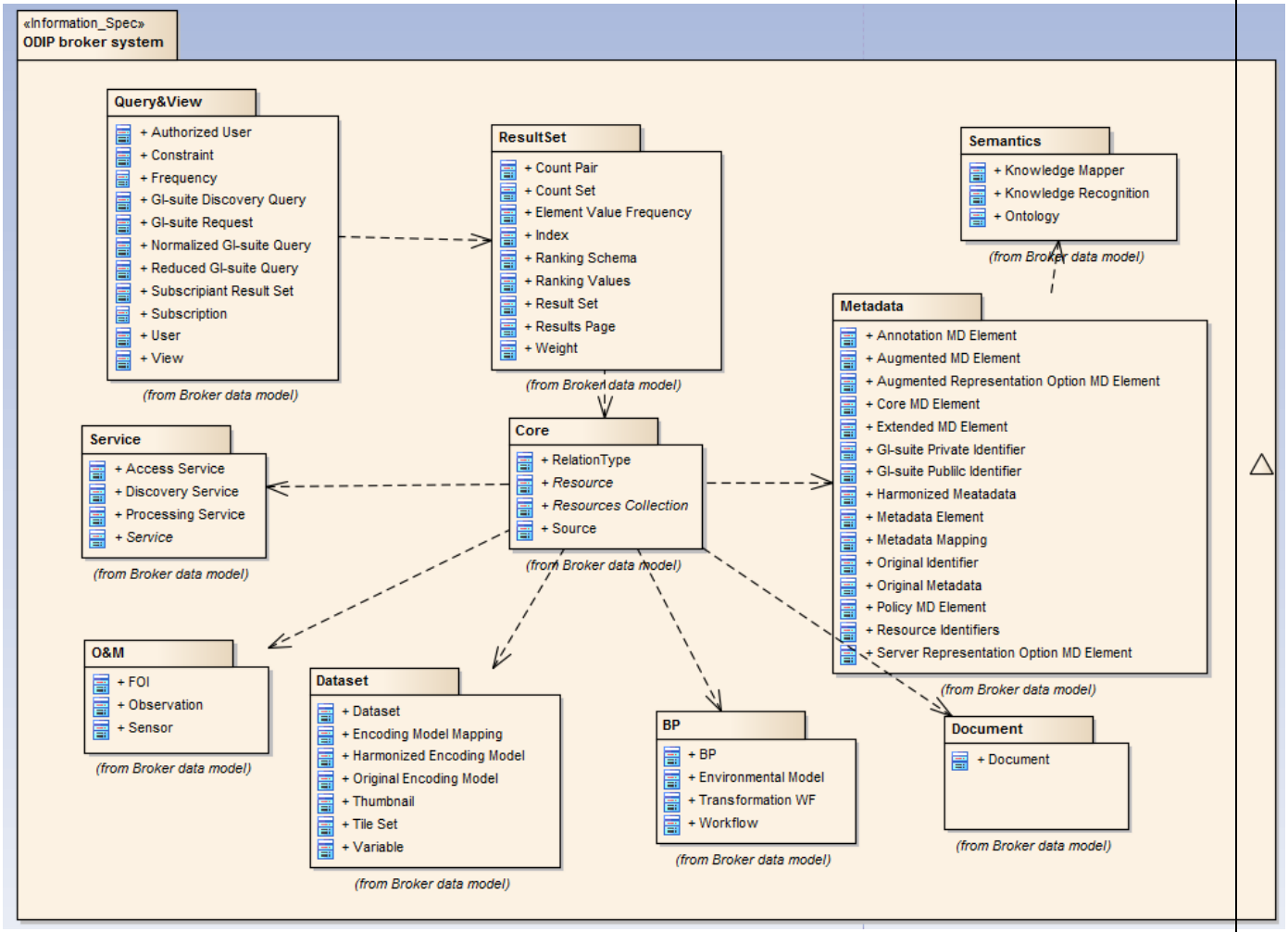

Figure 7 Structure of the information viewpoint specification of the full brokering system

Actions have been identified from processes expressed in the Enterprise View and using the related information objects (Figure 8 provides an excerpt of these).

For this viewpoint, the information actions are expressed using a package that expresses the invariant schema that specifies the action types supported by the information objects of the system.

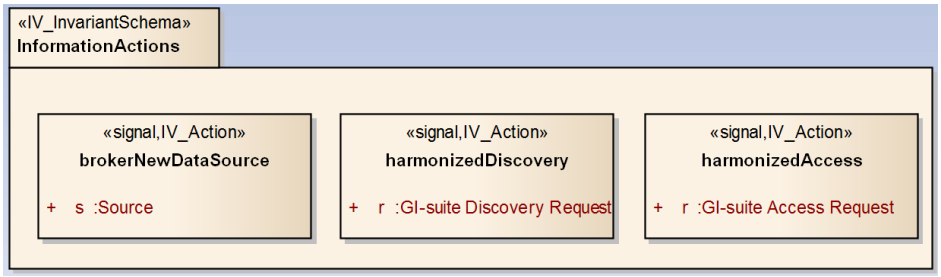

Figure 8 ODIP broker actions (excerpt)

Metadata schema

The metadata view concerns the information objects related to the discovery use case (harmonized discovery of metadata records from heterogeneous sources).

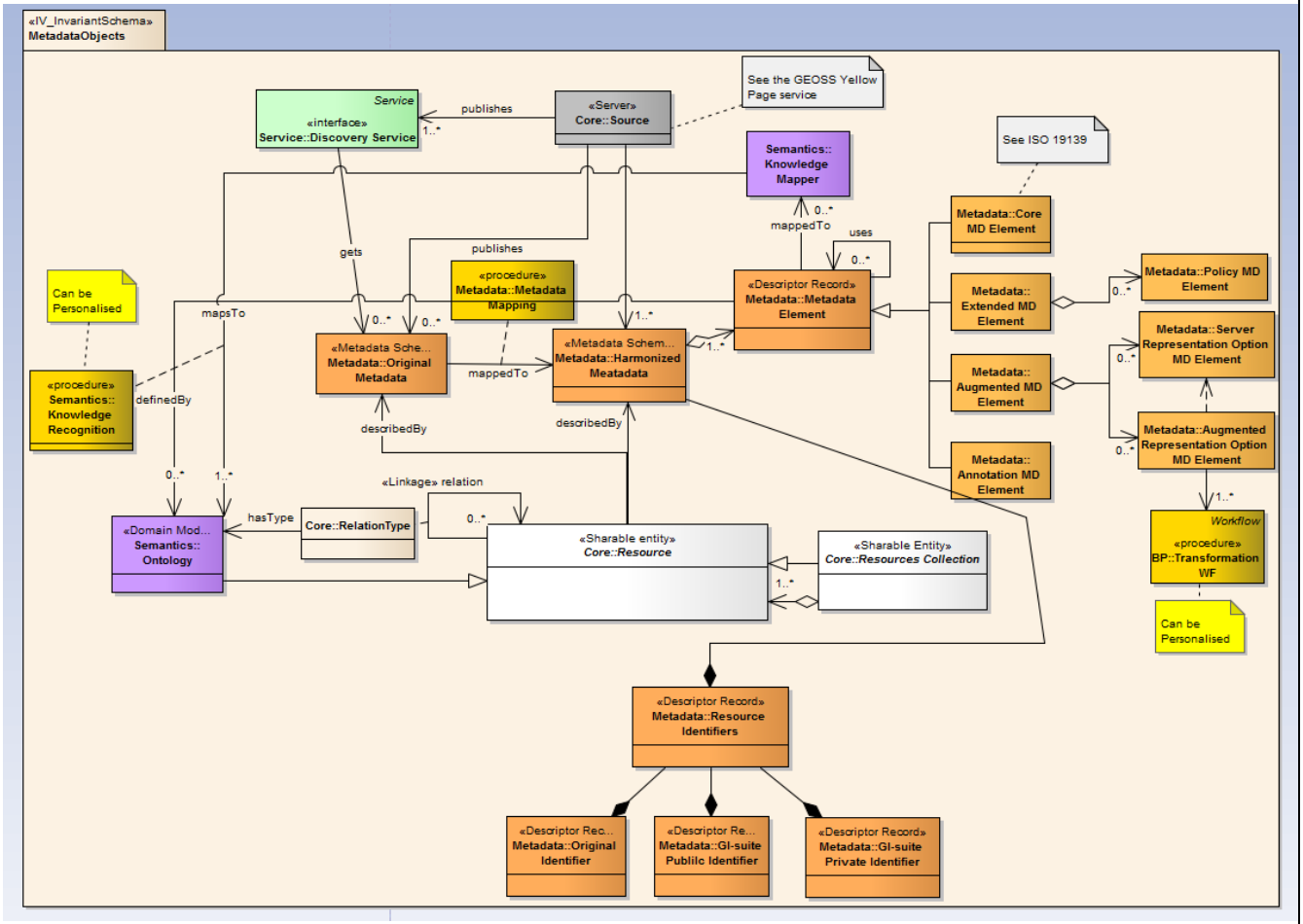

Figure 9 Metadata information view

The **Source** information object describes an organization (according to GEOSS Yellow Pages content model) publishing one or more **Discovery Services**.

Through **Discovery Service** it is possible to obtain a set of **Original Metadata** objects, documenting data provider **Resources**, according to the original metadata model implemented by a specific discovery service type. The currently brokered ODIP discovery services implements the following original metadata schemata:

- **SeaDataNet:** CDI profile [4] [5] of ISO 19115
- **AODN IMOS**: MCP profile [6] of ISO 19115
- **US NODC**: NODC profile of ISO 19115

Other common metadata information models implemented by other discovery services are:

- **Dublin core**;

- **DIF**;
- **Atom**;
- **RSS**;
- **O&M**.

All these metadata schemata are in general composed by a set of metadata elements, defined and encoded by metadata technical specifications.

The information content held by **Original Metadata** is mapped through a **Metadata Mapping** procedure to a **Harmonized Metadata** object, holding a translated description of the original record according to the Broker internal metadata model. Each different original metadata record is mapped to a correspondent **Harmonized Metadata** object, making possible a harmonized discovery.

The **Harmonized Metadata** object is further composed by a set of **Metadata Element** that can be used to describe the metadata object element by element. They can have different types:

- **Core MD Element**: Based on the ISO 19115 comprehensive profile metadata model, composed by more than 300 metadata elements described by the ISO standard [7].
- **Extended MD Element**: An extension point to hold custom information elements defined by a particular community.
- **Augmented MD Element**: These are elements created or updated as a result of a batch augmenter procedure (e.g. a batch procedure exists to test the access of remote datasets and updates the metadata object with the information provided by the access test results)
- **Annotation MD Element**

Semantics information

Specific metadata elements can be documented with terms from community based controlled vocabularies. As an example, let's consider the measured attribute (or parameter) *sea level*, a common measured attribute well known across the ODIP communities, but differently documented by each of them.

The following figures Figure 10, Figure 11 and Figure 12 show how the different communities document the same measured attribute element. In general terms, each community use different metadata elements to store this information. The information content is the term for *sea level* as expressed by a specific community controlled vocabulary.

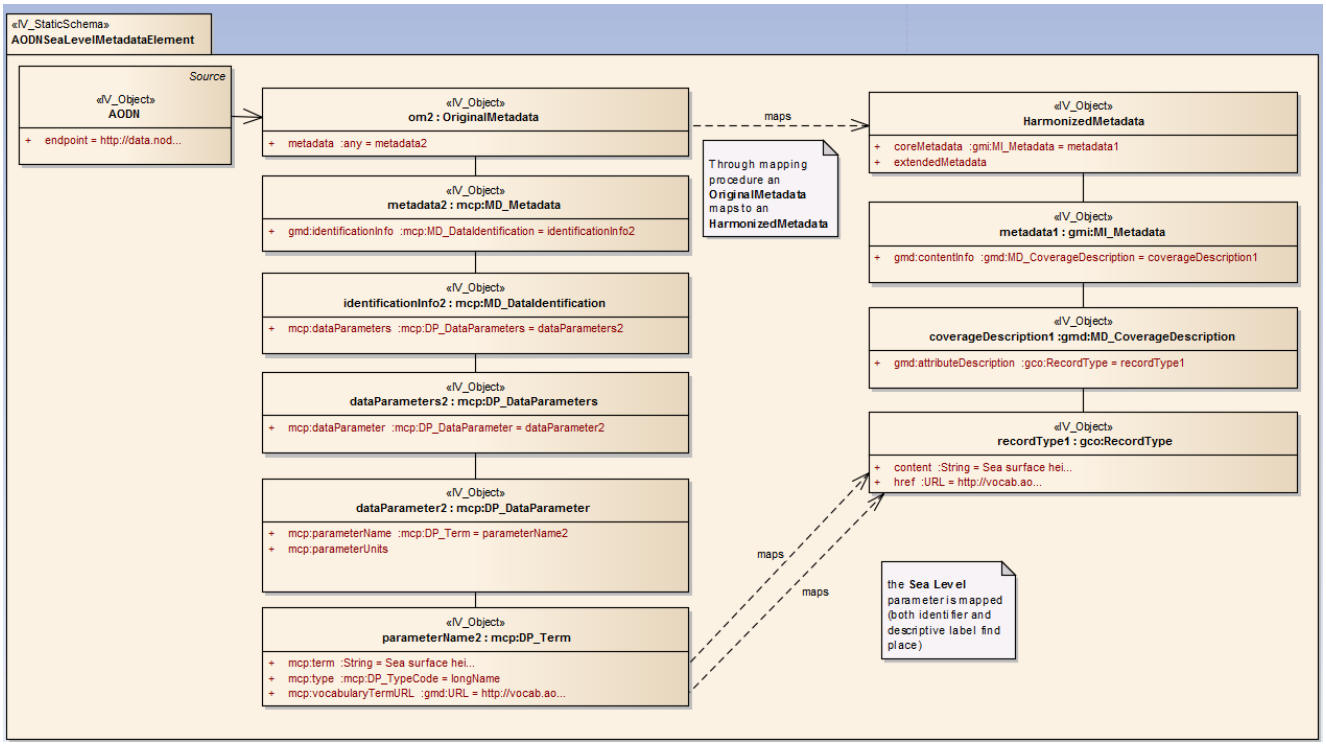

Figure 10 Documentation of Sea Level measured attribute in AODN community and its correspondent metadata harmonization by the broker

AODN community has drafted ISO-19115 community extensions (extended elements) to accommodate the semantics for the measured attribute.

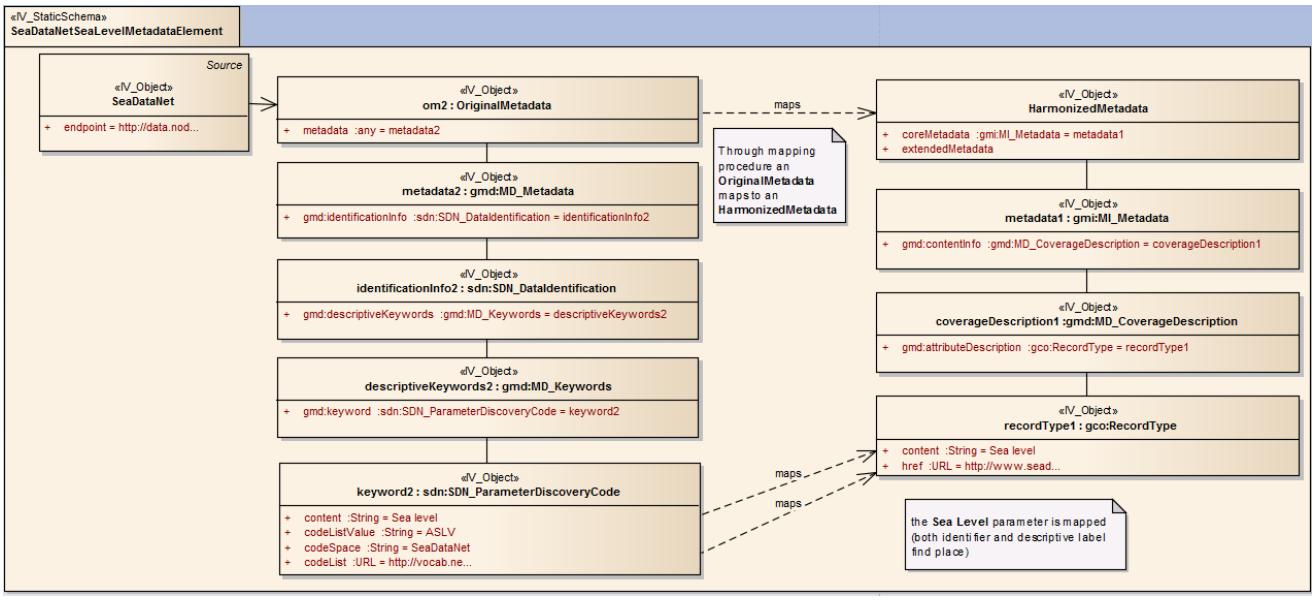

Figure 11 Documentation of Sea Level measured attribute in SeaDataNet community and its correspondent metadata harmonization by the broker

The SeaDataNet community has drafted ISO-19115 community extensions (extended codelists) to accommodate the semantics for the measured attribute.

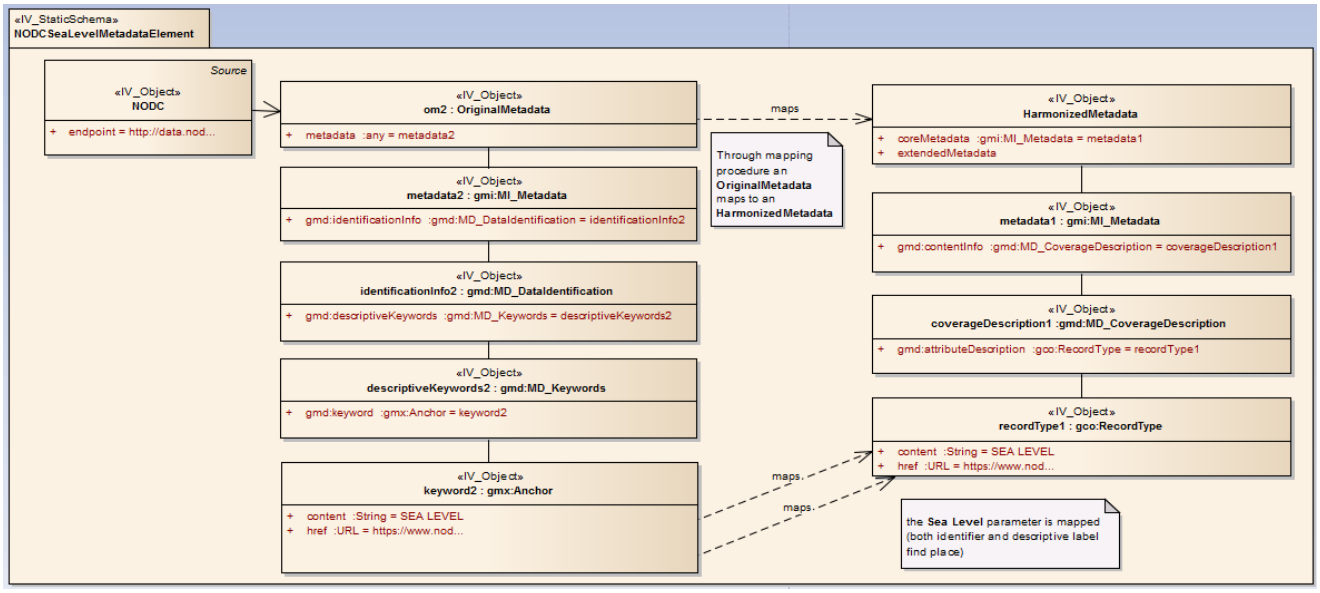

Figure 12 Documentation of Sea Level measured attribute in NODC community and its correspondent metadata harmonization by the broker

The NODC community has used the web content mechanism of web anchors, defined in ISO 19139 to accommodate the semantics for the measured attribute.

So, just for the *measured attribute* information element we have found three different ways of encoding its content (even if using the same general framework of ISO 19115). Each of them has its reasons to exist and is perfectly valid inside each community. It's responsibility of the broker to extract the *measured attribute* information from the different **OriginalMetadata** instances and map them to an harmonized metadata element in the **HarmonizedMetadata** document. The broker in this case uses the *attributeDescription* element from the comprehensive profile of ISO 19115, containing both a descriptive label and an identifier.

The identifiers are particularly important, as they are the primary means of identifying concepts in a semantics service, such as the ODIP Rosetta Stone. Rosetta Stone provides an ontology linking the concepts from the ODIP vocabularies, documenting them with the same identifiers that are present in the **HarmonizedMetadata** class, making thus possible for the broker to semantically augment the searches.

Figure 13 shows an example, for the measured attribute *sea level*, of the concepts present in Rosetta Stone service along with their relations. Rosetta Stone implements this way a common ontology for ODIP, linking terms from vocabularies of the different ODIP communities through common relations, such as owl:sameAs, skos:narrower and skos:broader.

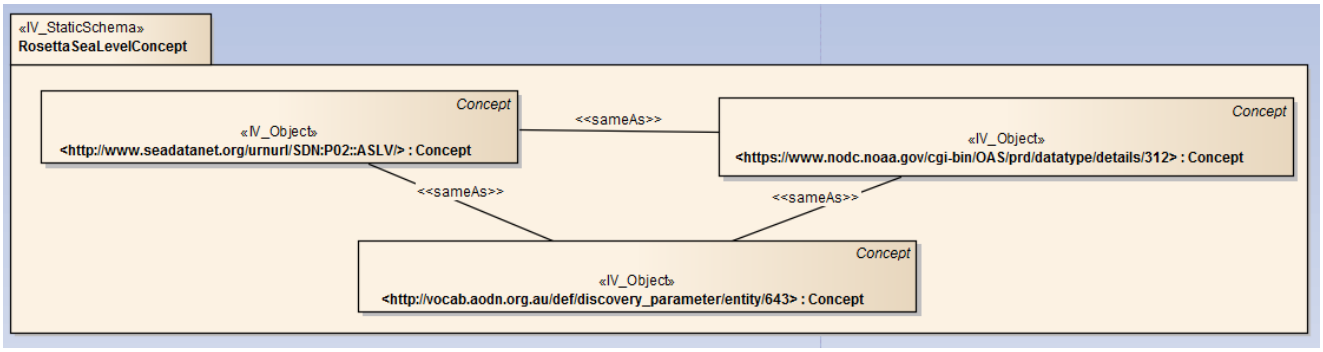

Figure 13 Rosetta Stone ontology (excerpt)

Data schema

The data view concerns the information objects related to the access use case (harmonized access of data from heterogeneous sources).

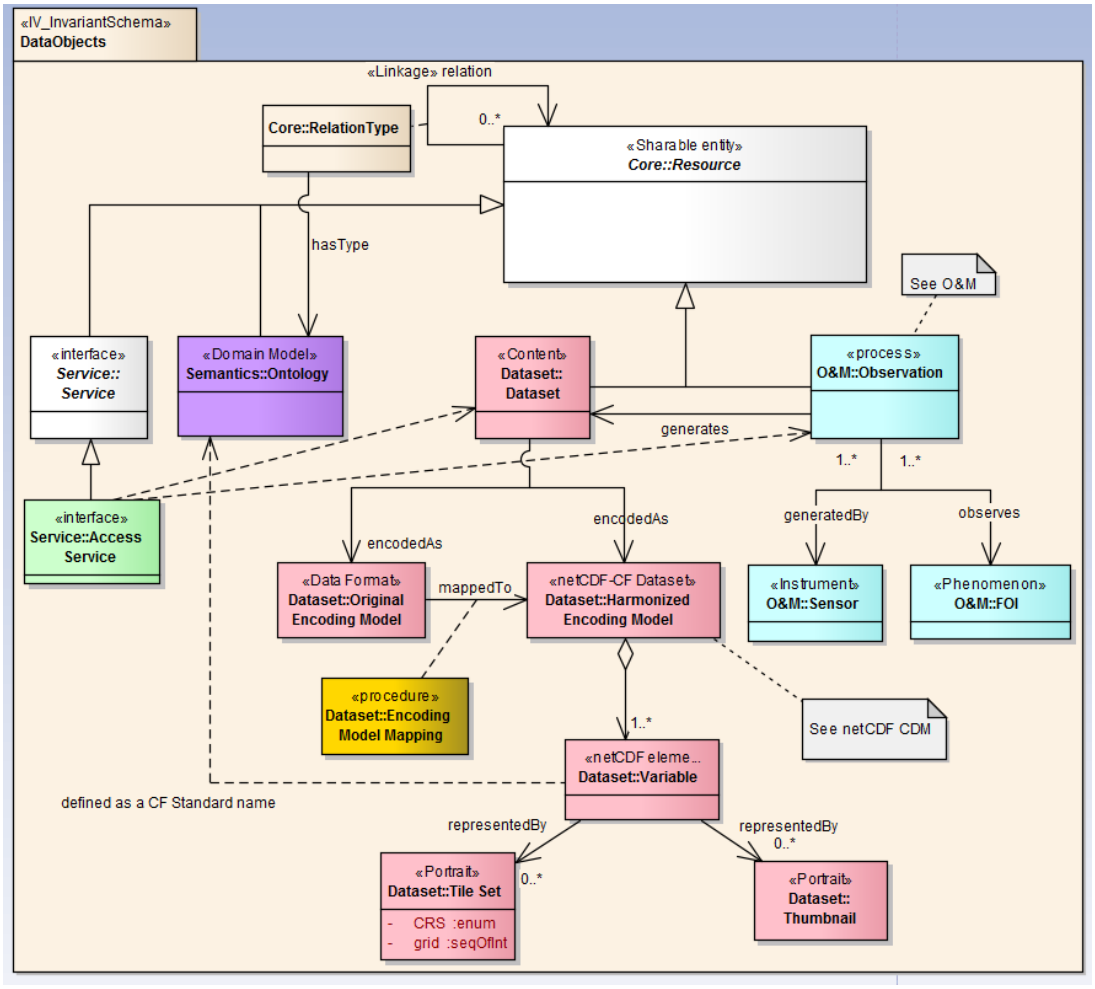

Figure 14 Data information view

An **Access Service** can publish **Datasets** and **Observations**, which are subtype of the **Resource** object. In general, an **Observation** generates a **Dataset** with a link back to its **Observation.** **Observation** is generated by a **Sensor** and observes a specific **Feature Of Interest**.

Also in the access case, datasets are encoded according to **Original Encoding Models**, that are in general different, according to the access service implementation (e.g. O&M, GeoTIFF, CSV, …). An **Encoding Model Mapping** procedure is used by the broker to map the different encodings to an **Harmonized Encoding Model** object, based on NetCDF-CF data model [8], hence composed by a set of **Variables** (possibly represented by **Tile Set** and **Thumbnails** when a graphical overview can be generated). Harmonizing all the heterogeneous data information content to NetCDF-CF makes it also easier to subsequently apply simple transformations on the harmonized data model (such as subset, CRS reprojection, etc.).

COMPUTATIONAL VIEW

The computational viewpoint deals with functional the decomposition of the ODIP prototype system in distribution transparent terms. This computational specification defines units of function as computational objects (expressed as components), and the interactions among those computational objects, without considering their distribution over networks and nodes.

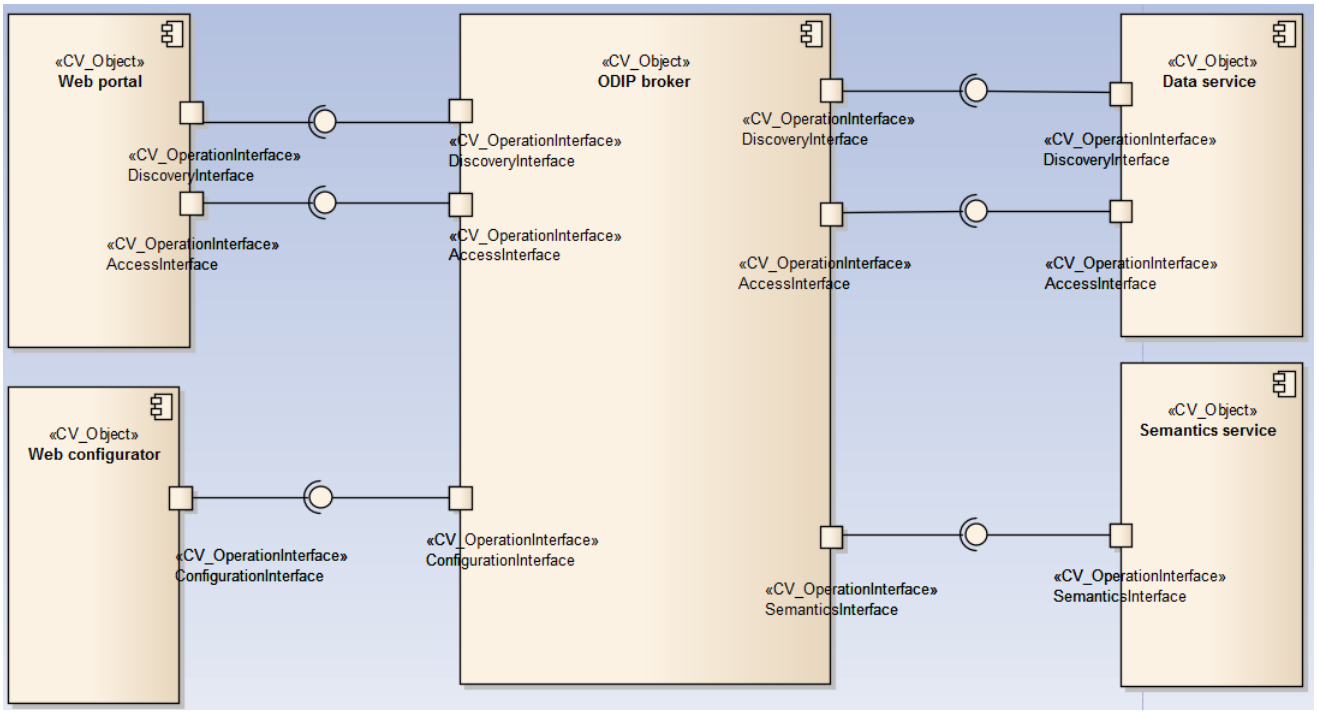

Figure 15 Component diagram highlighting macro components of the ODIP broker system

The component diagram in Figure 15 shows the macro components of the ODIP broker system. These are:

- **ODIP broker**: the main component
- **Web portal**: a community web portal, interacting with the ODIP broker through discovery and access interfaces
- **Data service**: a community data service, accessed by the ODIP broker through discovery and access interfaces
- **Semantics service**: a semantics service, accessed by the ODIP broker through semantics interface
- **Web configurator**: used to configure the ODIP broker system through requests to its configuration interface

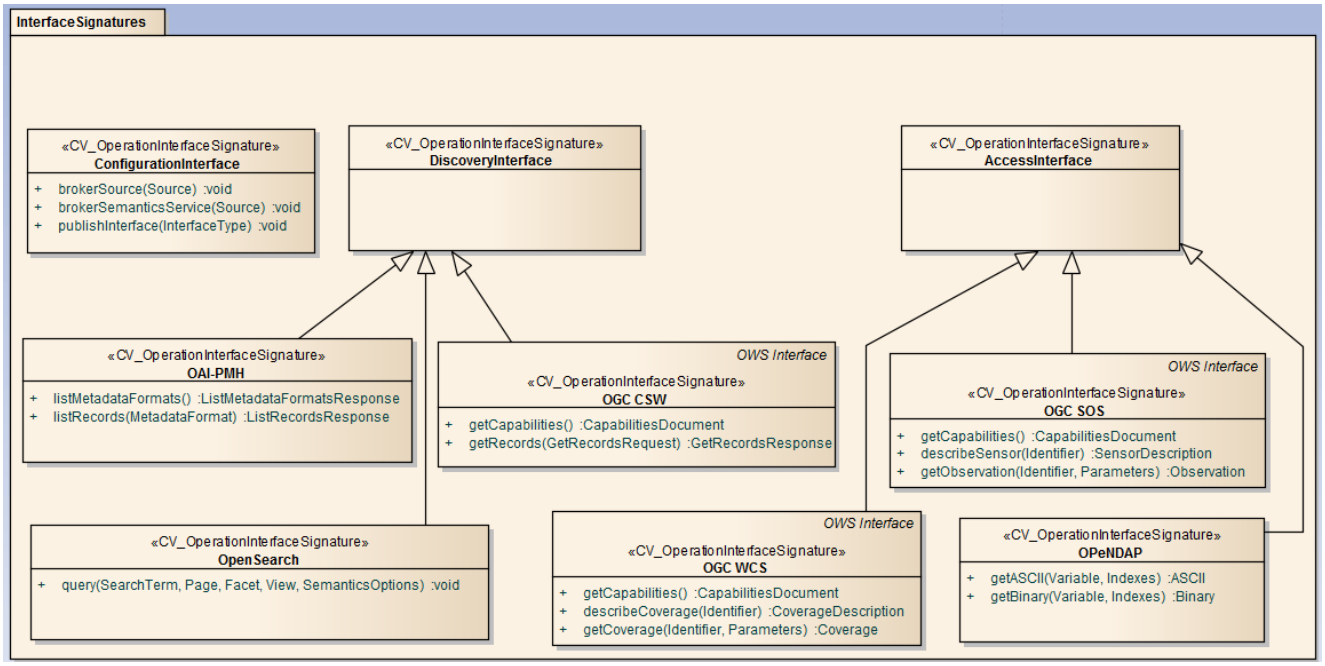

Figure 16 Interaction signatures (excerpt)

Object interfaces are expressed as component ports, the components interact with each other at computational interfaces (port instances). Each port is of a particular type and implements or uses several interfaces (which express the corresponding interface signatures shown in Figure 16). They are all operation interface signatures.

The identified interfaces (from the requirement captured by the enterprise and information specifications) are:

- **Configuration interface**: containing operations to configure the ODIP broker to the required scenario
- **Discovery interface**: containing operations to discover resources. This abstract interface is specialized by many different discovery interfaces. Amongst them, example given:
    - **OAI-PMH** [9]: The Open Archives Initiative Protocol for Metadata Harvesting (referred to as the OAI-PMH in the remainder of this document) provides an application-independent interoperability framework based on metadata harvesting. An implementation of OAI-PMH must support representing metadata in Dublin Core, but may also support additional representations.
    - **OGC CSW** [10]: Catalogue services support the ability to publish and search collections of descriptive information (metadata) for data, services, and related information objects. Metadata in catalogues represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software. Catalogue services are required to support the discovery and binding to registered information resources within an information community.
    - **OpenSearch** [11]: Search clients can use OpenSearch description documents to learn about the public interface of a search engine. These description documents contain

parameterized URL templates that indicate how the search client should make search requests. Search engines can use the OpenSearch response elements to add search metadata to results in a variety of content formats.

- **Access interface** containing operations to access resources. This abstract interface is specialized by many different access interfaces. Amongst them, example given:
    - **OPeNDAP** [12]: the Open-source Project for a Network Data Access Protocol (OPeNDAP) Data Access Protocol (DAP), a data transmission protocol designed specifically for science data. The protocol relies on the widely used and stable Hypertext Transfer Protocol (HTTP) and Multipurpose Internet Mail Extensions (MIME) standards, and provides data types to accommodate gridded data, relational data, and time series, as well as allowing users to define their own data types.
    - **OGC SOS** [13]: The SOS standard is applicable to use cases in which sensor data needs to be managed in an interoperable way. This standard defines a Web service interface which allows querying observations, sensor metadata, as well as representations of observed features.
    - **OGC WCS** [14]: a Web Coverage Service (WCS) offers multi-dimensional coverage data for access over the Internet.

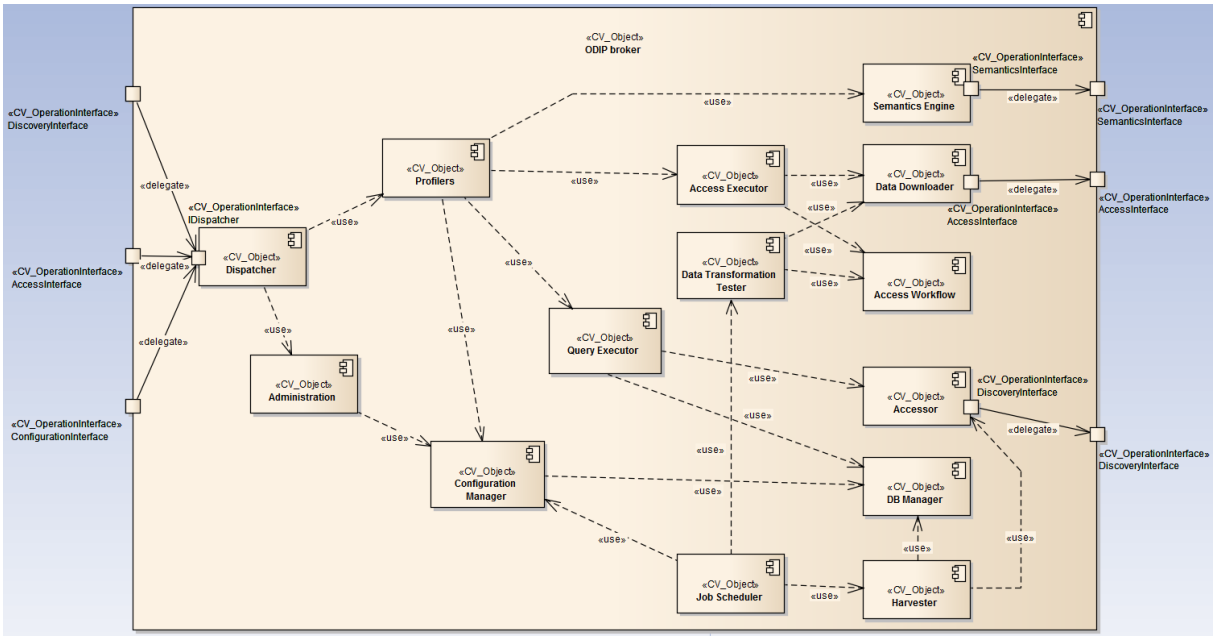

Figure 17 Internal structure of the ODIP broker computational object

Figure 17 shows the main subcomponents of ODIP broker, along with their interactions. They are:

- **Dispatcher**: in charge of dispatching incoming requests to either the Administration component or to one of the available Profiler. The Dispatcher uses a path-based strategy to select the component to forward the request to.
- **Profiler**: publishes GI-suite functionalities according to a specific service interface (e.g. OGC CSW, OGC WCS, OPeNDAP, etc.). In order to execute the incoming request, the Profiler must

perform a set of actions in a given order. The set of actions and their order depend on the specific incoming request and are defined by Function Handlers.

- **Query executor**: executes discovery of the resources matching the user queries (both count and retrieval) from the sources that are available in the source configuration document and depending on user authorizations.
- **Accessor**: in charge of communicating with remote heterogeneous services, downloading the original metadata records and transforming the **Original metadata** to **Harmonized metadata**.
- **Semantics engine**: in charge of communicating with remote semantics services, in order to execute semantics queries (e.g. to retrieve related terms in an ontology).
- **Access executor**: in charge of executing access requests, orchestrating the **Data Downloader**and the **Access Workflow**.
- **Data downloader**: in charge of retrieving data from the provider access service.
- **Access workflow**: in charge of executing simple transformations (such as format conversion, CRS reprojection, subset, interpolation) to transform the downloaded data according to the user access request
- **Administration**: in charge of managing ODIP broker system options, delegating incoming requests to the configuration manager
- **Configuration manager**: in charge of reading/writing the configuration to the **DB**; periodically synchronize local configuration with the remote one (on DB); fire an event when an updated configuration is found on the **DB**
- **Job Scheduler**: in charge of scheduling and launching ODIP broker recurrent jobs (e.g. harvesting, access tests). The component defines the interface for scheduling and launching jobs in GI-suite; provides mediation functionalities to use external dedicated scheduling libraries
- **Data Transformation Tester**: verifies that all required access transformation workflow outputs are reachable from at least one **RemoteDataDescriptor**.
- **Harvester**: implements harvesting functionality which means collect metadata from a **Source** and store them into the **DB**.
- **DB Manager**: provides complete interaction with a database. The interaction is done through a set of published interfaces, which provide an abstraction layer on the underlying database implementation.

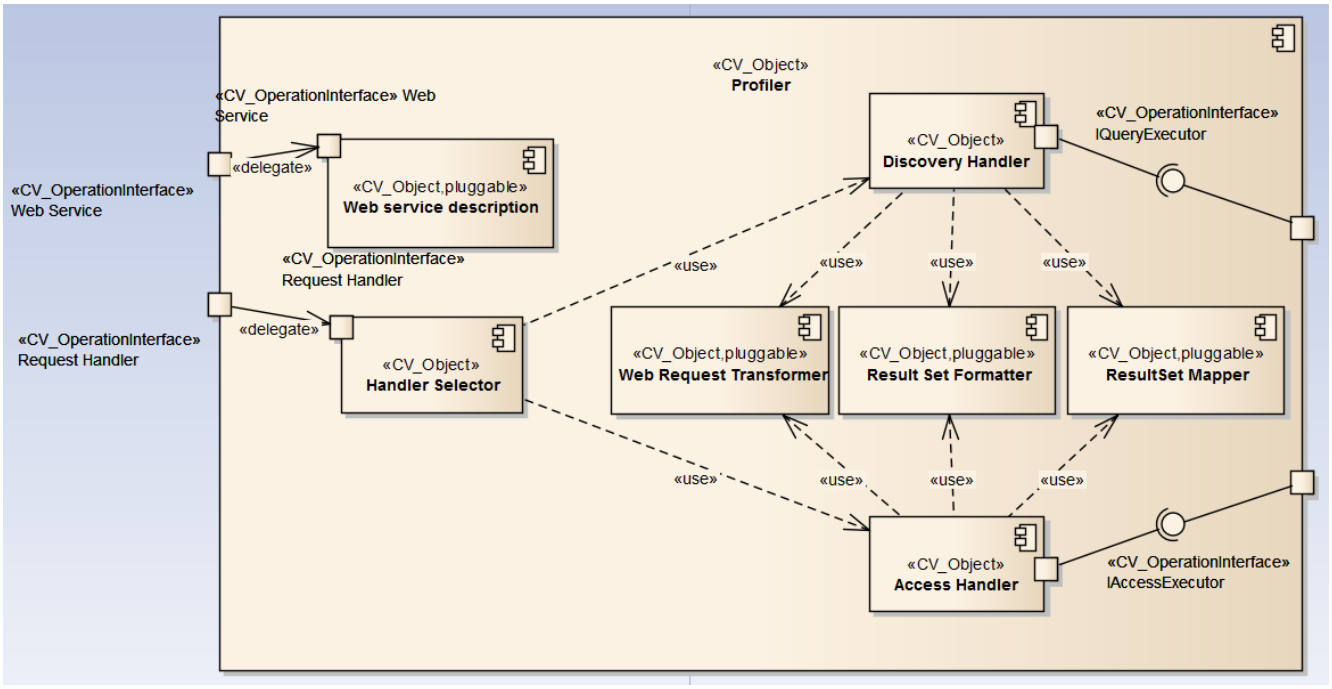

Figure 18 Internal structure of the Profiler component, showing pluggable sub components

The sub components of the **Profiler** component are here reported as an example to appreciate in full details the flexibility and modularity capabilities enabled by ODIP broker <<pluggable>> components.

Two main function handlers exist, responsible for the two main categories of available functionalities: the **Discovery Handler** and the **Access Handler**. Each of them is composed by a configurable set of <<pluggable>> components, together concurring to determine their actual behaviour. E.g.

- **Web Request Transformer**: in charge of validating and transforming a Web Request to an internal harmonized Request
- **Result Set Mapper**: in charge of creating a one to one mapping of the **Harmonized metadata** in the **Result Set** according to the metadata model required by the service interface specification
- **Result Set Formatter**: in charge of join and format the mapped **Result Set** in order to be presented to the client, according to the service interface specification

The <<pluggable>> stereotype indicates that is possible to easily extends ODIP broker, by simply making available an additional implementation of such a <<pluggable>> component to the system (e.g. to easily create additional components that are able to publish new interface types or brokering new source types).

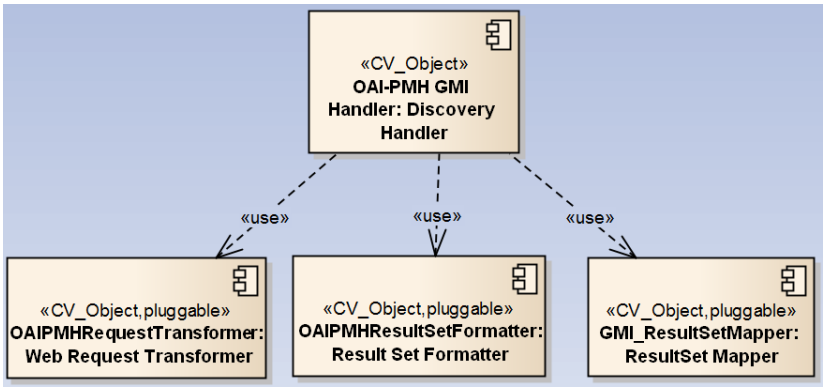

Figure 19 Component instances of an instantiated Discovery Handler: the OAI-PMH GMI Handler

Figure 19 shows an example of instantiated components, concurring to "construct" the composed component **OAI-PMH GMI Handler** that is a component able to publish the OAI-PMH service interface supporting ISO 19115-2 as metadata format. The instantiated components are in this case:

- **OAIPMHRequestTransformer**: in charge of transforming web requests valid according to the OAI-PMH protocol to the internal harmonized **Query request**.
- **OAIPMHResultSetFormatter**: in charge of creating a valid (according to OAI-PMH) list records web response structure to be filled with mapped records
- **GMI_ResultSetMapper**: in charge of mapping from the **Harmonized metadata** model to the ISO 19115-2 metadata model.

The interaction diagram in Figure 20 shows the interactions between the described components during execution of an OAI-PMH List Records request for GMI metadata records:

- The *OAI-PMH Web Request* is received by the Broker OAI-PMH service interface. The first component to manage it is the **Dispatcher**, passing the request to be served to the **Profiler** component able to manage it.
- The **Handler Selector** is a **Profiler** sub component able to delegate to a configured **Handler** (in this case the **Discovery Handler**, with **GMI Result set Mapper** and **OAI-PMH Formatter**).
- The **Discovery Handler** uses the **Web Request Transformer** to translate from the *OAI-PMH Web Request* to a discovery request expressed as an internal *Discovery Message.*
- The *Discovery Message* is (possibly) expanded through an invocation to the **Semantics Engine**, which in turns contacts a semantics service such as Rosetta Stone to resolve search terms (if any) according to the user required semantics relation (if any).
- The augmented *Discovery Message* is passed to the **Query Executor**, in charge of executing the query and return matching resources from both the DB and the remote sources.
- Each returned *Resource* (in particular each *Harmonized Metadata* object) is mapped to a GMI metadata document in this example;
- The mapped records are collected and then formatted by the **Result Set Formatter** according to OAI-PMH response schema.
- The response is sent back to the client application.

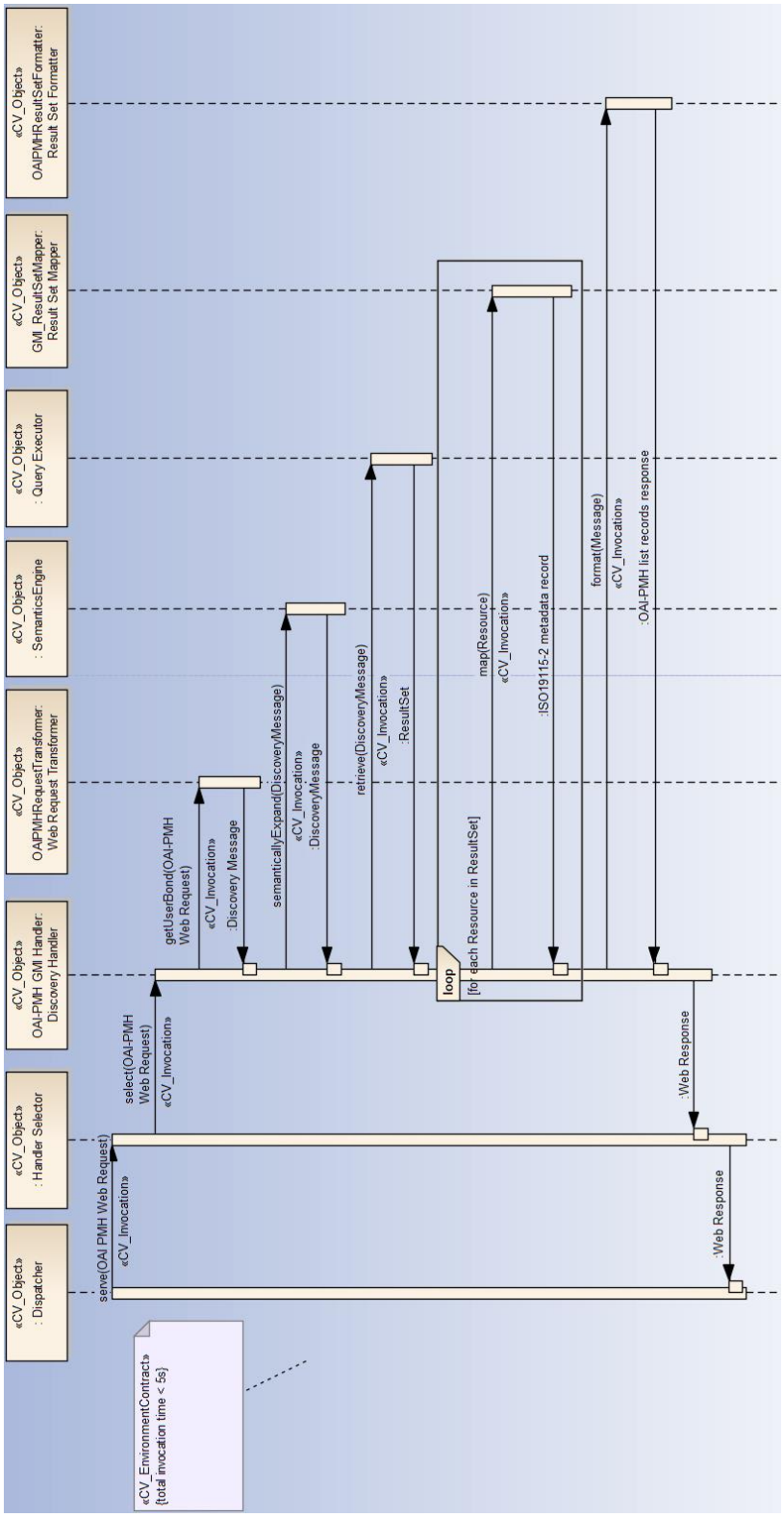

Figure 20 Interaction diagram focusing on sub components involved during an execution of an OAI-PMH List Records request

Interoperability API components

To simplify the development of third-party applications and clients that make use of the broker services, an high level client-side Open APIs (Application Program Interface) was designed and developed in JavaScript. The interoperability API is published online [15]; such an interface is going to be submitted to OGC for standardization, with the support of GEO. The main objects around which the API is developed are shown in Figure 21. These are:

- **Broker**: The API entry point. This object provides the ability to create a node connected to an existing broker server instance. **Broker** is a composed **GI-node** and it's the root of the hierarchical structure defined by the brokered sources. The main **Broker** operation allows to discover any **GI-node** regardless of their level in the hierarchy.
- **ConsumerDefinedView**: this object enables the same operations allowed by the **Broker**, but limiting results on a specified subset of resources of interest, selected by a set of predefined discovery constraints.
- **GI-node**: A **GI-node** is a Geo Information node representing an abstract geospatial resource, a "dataset" or a "service", available as result of a query to the **Broker**.
  GI-node properties such as title and abstract are described by a **Report**. A particular **Report** property attribute is type, which defines which kind of interactions are available with the node. When a node represents a "service" such as WMS, WCS, etc, the report has an additional service property.
- **Source**: This kind of **GI-node** represents a source brokered by a **Broker** instance. **Broker sources** can be retrieved with the getSources() method. Since the **Broker sources** are first

level nodes (the **Broker** direct "children"), they can also be retrieved as result of the first call of the expand method.

- **Paginator**: This object is provided as result of a discover or expand/expandNext operation and has several information such as the number of the returned nodes (the size of the result set) and the number of pages with which the result set is split.
- **Page**: A result set page of GI-nodes

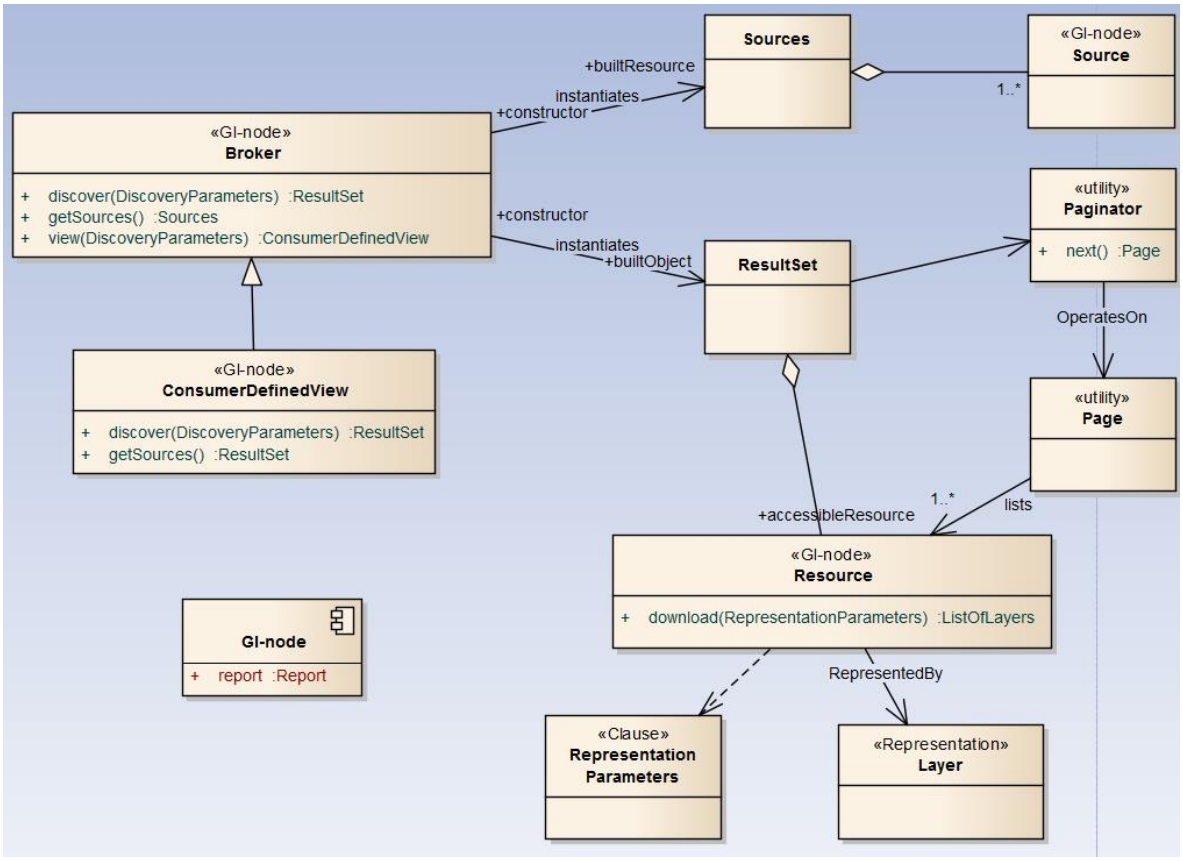

Figure 21 Class diagram of the interoperability API showing main objects and operations

ENGINEERING VIEW

The ODIP broker system is based on a three tier architectural style consisting of the following nodes –see As Figure 22 :

- **ClientTier**: an user of a SeaDataNet community app will use a desktop or notebook PC, which serves as **ClientTier**; a portal such as the ODP Portal is installed on a server node serving as **ClientTier**
- **BrokerTier**: requests from **ClientTier** nodes are sent to a middle tier server node, which serves as the **BrokerTier**. The ODIP broker middleware service is executing on the ODIP broker ECS cluster node.
- **ServerTier**: functional requests are sent from the **BrokerTier** to other server nodes, which serves as **ServerTier**. Data and semantics services are executing on these server nodes, such as IMOS AODN CSW/ISO-MCP services and Rosetta Stone semantics service.

For a system-of-systems development, the three tiers brokering architecture has many advantages with respect to the traditional two tiers Client-Server archetype (depicted in Figure 23 Figure 23). The critical interoperability issue can be summarized as the problem of allowing M different applications to interact with N different data sources: an MxN complexity problem.

By an architectural point-of-view, System-of-systems can be implemented in a pure two-tier (client-server) environment. The M clients can interact with N servers simply, because only one type of interaction is defined by a common model –aka the federated model. The MxN complexity is solved at client/server level requiring both of them to be compliant with the federated model. On the other hand, brokered architectures introduce a middle-tier, between clients and servers, reducing the MxN potential interactions (each client interacting with each server) to M+N (each client and each server only need to interact with the brokers). The middleware is in charge of mediating between heterogeneous clients and server, leaving them autonomous –i.e. they do not have to implement the federated model.

The deployment diagram, depicted in Figure 24 provides further details on the deployment of the ODIP broker on an Amazon Web Services (AWS) ECS cluster. Two virtual machines are dedicated to the deployment. Each one hosts an ODIP broker service. Each service is composed by an auto-scaling set of containers, deployed as Basic Engineering Objects (BEO), providing identical brokering services (because instantiated by identical container images). A BEO **Application Load Balancer** distributes incoming requests amongst the available **ODIP broker containers**. Auto-scaling is regulated by upscaling and downscaling rules, triggered by request execution times. A health check mechanism is in place to remove containers eventually starting to exhibit a malfunctioning behavior. The container based architecture along with the Amazon services enable portability, reproducibility and production level Quality Of Service (QOS) requirements in terms of availability, reliability and performance.

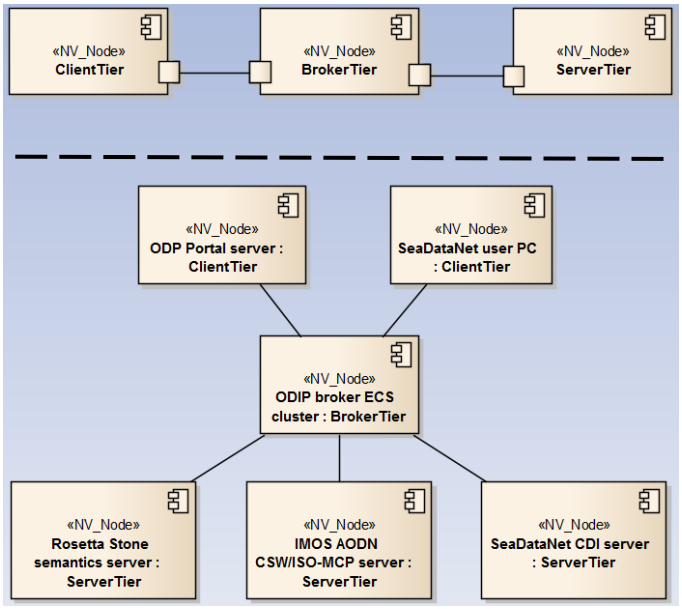

Figure 22 Node configuration for the brokering system architecture (client and server nodes are connected through the broker: M+N connections)

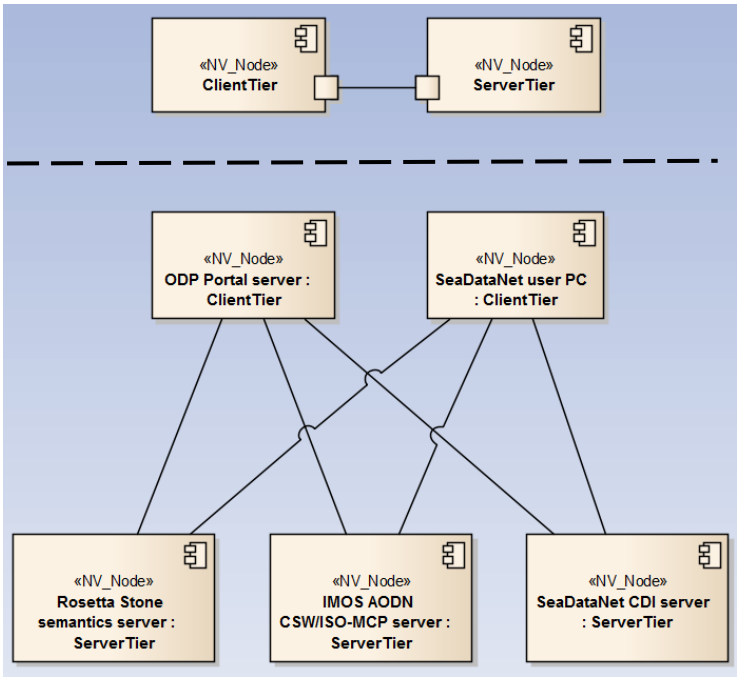

Figure 23 Node configuration for a client server architecture (each client node connects to each server node: M*N connections)

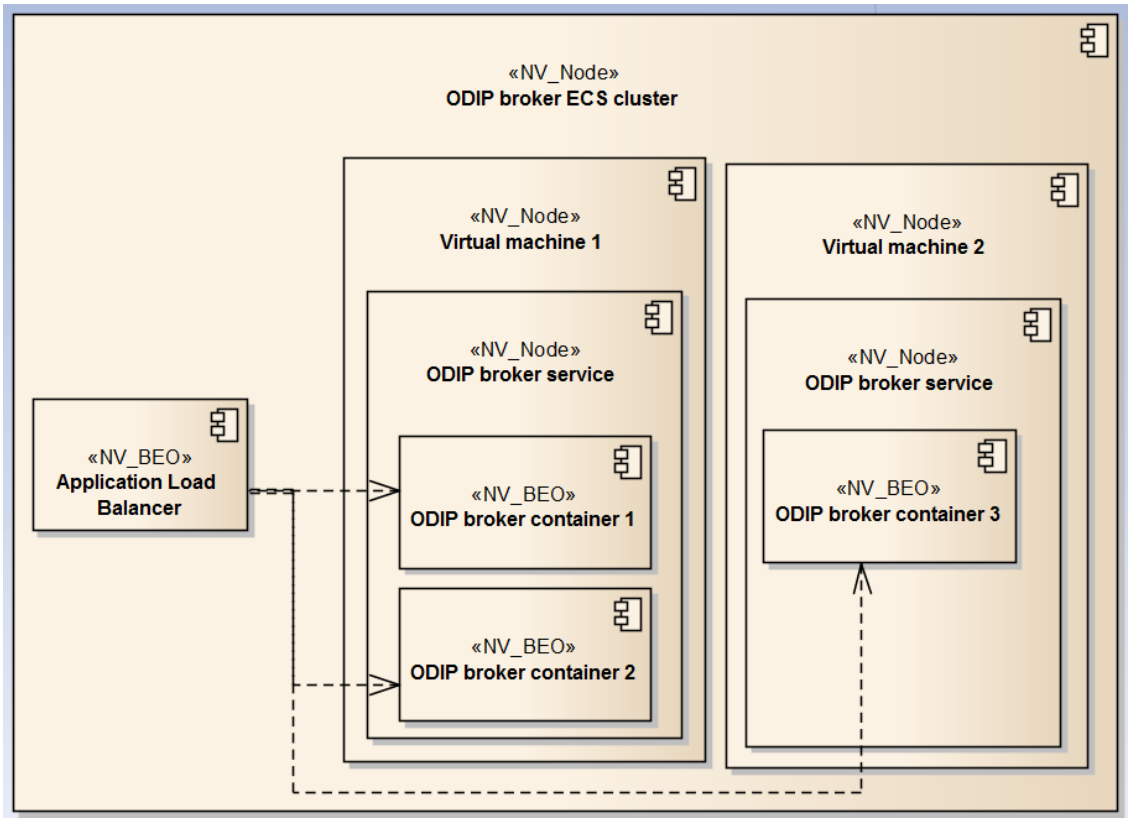

Figure 24 Details of ODIP broker AWS ECS cluster deployment

TECHNOLOGICAL VIEW

The **ODIP broker** is a Java based software framework supporting a multiplatform deployment.

Technology implemented and tested in the realization of GEOSS Discovery and Access Broker (**GEO-DAB**) [16] has contributed to realize the ODIP brokering framework.

For the **AWS ECS** [17] deployment in ODIP the following technology stack has been adopted: **Docker** API version 1.3.5 to build the container image (based on **Debian** Jessie, Java **OpenJDK** version 1.8, **Apache Tomcat** servlet container version 8).

Java **ServiceLoader** mechanism has been used to provide pluggable capability to specific component types.

JAX-WS specification has been implemented to publish the broker web service interfaces, and realized through **Apache CXF** Web services framework.

MarkLogic Server [18] has been adopted as a XML database for local cache of metadata content enabling optimized searches.

The interoperability API has been realized as a **JavaScript** library, adopting an object-oriented style paradigm. Web portal developers can easily import it in their web project to connect to the ODIP broker.

The **Atlassian JIRA** [19] issue tracking system is used to document and manage technical requests (as mentioned in the Enterprise view section), as well as to document and manage code releases. It provides bug tracking, issue tracking, and project management functions.

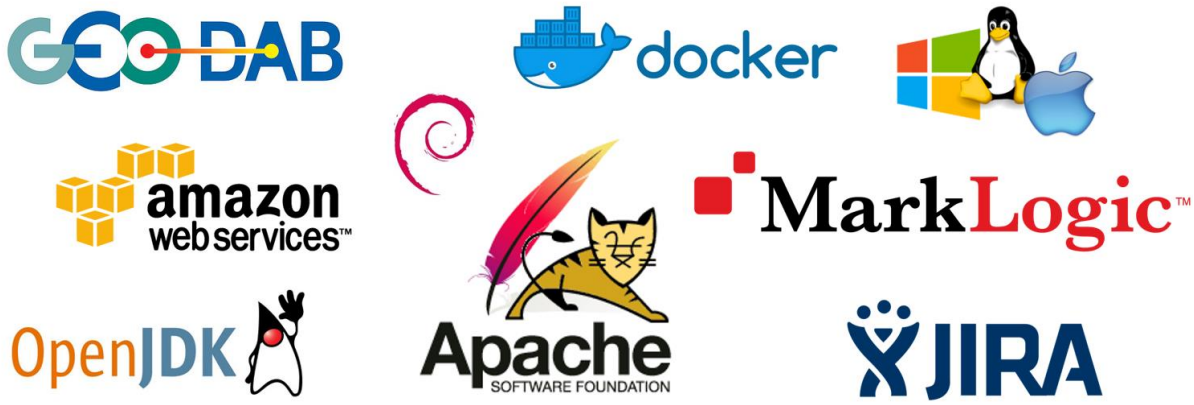

Figure 25 Main technologies powering up the ODIP broker prototype

DISCUSSION

To adopt a formal reference model has proven to be useful to provide a documentation of the ODIP broker system, highlighting both the approach and technical features:

- **Enterprise view**: The actors and requirements of the ODIP broker have been highlighted, although focusing on the ODIP broker (system) community, it would be useful as a future work to describe in more details as well the other communities that interact with it, especially considering specific policies and rules that might emerge in real-world cases in the international context.
- **Information view**: the information managed by the broker has been described according to this important view. Heterogeneous information coming from remote distributed services needs to be harmonized and managed by the broker.
- **Computational view**: components implementing the functional requirements are presented by this view. The main components are shown, focusing on the possibility to customize and extend the ODIP broker in the future leveraging pluggable components.
- **Engineering view**: the actual deployment of the ODIP broker system has been described, focusing on the Cloud deployment to assure QOS requirements.
- **Technological view**: technologies used to actually implement the ODIP broker system have been presented

CONTACT POINTS

For further information, please contact:

Enrico Boldrini (enrico.boldrini@cnr.it)

Stefano Nativi (stefano.nativi@cnr.it)

REFERENCES

[1] ISO/IEC, «Information technology — Open Distributed Processing — Reference model: Overview,» 15 12 1998. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c020696_ISO_IEC_10746-1_1998(E).zip.

[2] ITU-T, «Information technology - Open distributed processing - Use of UML for ODP system specifications,» 10 2014. [Online]. Available: http://www.lcc.uma.es/%7Eav/download/UML4ODP_IS_V2.pdf.

[3] D. M. A. Schaap, «Deliverable D3.1: Definition of ODIP Prototypes 1,» 2016.

[4] E. Boldrini e S. Nativi, «SeaDataNet metadata profile of ISO 19115,» 2017. [Online]. Available: https://www.seadatanet.org/content/download/1855/11028/file/CDI-profile-V10.0.1.pdf?version=3.

[5] E. Boldrini, S. Nativi e D. M. Schaap, «INSPIRE compliant international standards for the SeaDataNet marine metadata,» in *IMDIS 2013*, Lucca, 2013.

[6] Australian Ocean Data Centre, «Marine Community Profile Manual v2.0,» 2006. [Online]. Available: http://mcp-profile-docs.readthedocs.io/en/stable/.

[7] ISO, «ISO 19115 Geographic information - Metadata,» ISO, Geneva, 2003.

[8] B. Domenico e S. Nativi, «CF-netCDF Data Model extension specification,» OGC, 2012.

[9] The OAI Executive, «The Open Archives Initiative Protocol for Metadata Harvesting,» 2002. [Online]. Available: http://www.openarchives.org/OAI/openarchivesprotocol.html.

[10] OGC, «Catalogue Service,» [Online]. Available: http://www.opengeospatial.org/standards/cat.

[11] C. DeWitt, «OpenSearch 1.1 Draft 6,» [Online]. Available: https://github.com/dewitt/opensearch/blob/master/opensearch-1-1-draft-6.md.

[12] OPeNDAP , «OPeNDAP - Advanced Software for Remote Data Retrieval,» [Online]. Available: https://www.opendap.org/.

[13] OGC, «Sensor Observation Service,» [Online]. Available: http://www.opengeospatial.org/standards/sos.

[14] OGC, «Web Coverage Service,» [Online]. Available: http://www.opengeospatial.org/standards/wcs.

[15] «DAB JavaScriptAPI,» CNR-IIA, 2018. [Online]. Available: http://api.eurogeoss-broker.eu/docs/index.html.

[16] Group on Earth Observations, «GEO Discovery and Access Broker,» [Online]. Available: http://www.geodab.net/.

[17] Amazon, «Amazon Elastic Container Service,» [Online]. Available: https://aws.amazon.com/ecs/.

[18] MarkLogic, «MarkLogic Server homepage,» [Online]. Available: https://www.marklogic.com/.

[19] Atlassian, «Atlassian Jira,» [Online]. Available: https://jira.atlassian.com/.