Project No. 249024

NETMAR

Open service network for marine environmental data

| Instrument: *Please tick* | CA | ✓ **STREP** | IP | NOE |
|---|---|---|---|---|

**ICT - Information and Communication Technologies Theme**

**D7.5.2 Dissemination material and workshop reports**

Reference: D7.5.2_Dissemination_Material_and_Workshop_Reports_r1_20121130

Due date of deliverable (as in Annex 1): M0 + 33
Actual submission date: 30 November 2012

Start date of project: 1 February 2010 Duration: 3 years

Nansen Environmental and Remote Sensing Center

Revision 1

| **Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

European Commission
Information Society and Media

| | |
|---|---|
| | **NETMAR**<br>Open service network for marine environmental data<br>Project Reference: 249024<br>Contract Type: Collaborative Project<br>Start/End Date: 01/03/2010 - 31/01/2013<br>Duration: 36 months |
| | Coordinator: Prof. Stein Sandven<br>Nansen Environmental and Remote Sensing Center<br>Thormøhlensgate 47, Bergen, Norway<br>Tel.: +47 55 20 58 00<br>Fax. +47 55 20 58 01<br>E-mail: stein.sandven@nersc.no |

**Acknowledgements**

The work described in this report has been partially funded by the European Commission under the Seventh Framework Programme, Theme ICT 2009.6.4 ICT for environmental services and climate change adaptation.

**Consortium**

The NETMAR Consortium is comprised of:

- Nansen Environmental and Remote Sensing Center (NERSC), Norway (coordinator).
  Project Coordinator: Prof. Stein Sandven (stein.sandven@nersc.no)
  Deputy Coordinator: Dr. Torill Hamre (torill.hamre@nersc.no)
  Quality Control Manager: Mr. Lasse H. Pettersson (lasse.pettersson@nersc.no)
- British Oceanographic Data Centre (BODC), National Environment Research Council, United Kingdom
  Contact: Dr. Roy Lowry (rkl@bodc.ac.uk)
- Centre de documentation de recherche et d'expérimentations sur les pollutions accidentelles des eaux (Cedre), France.
  Contact: Mr. François Parthiot (Francois.Parthiot@cedre.fr)
- Coastal and Marine Resources Centre (CMRC), University College Cork, National University of Ireland, Cork, Ireland.
  Contact: Mr. Declan Dunne (d.dunne@ucc.ie)
- Plymouth Marine Laboratory (PML), United Kingdom.
  Contact: Mr. Steve Groom (sbg@pml.ac.uk)
- Institut français de recherche pour l'exploitation de la mer (Ifremer), France.
  Contact: Mr. Mickael Treguer (mickael.treguer@ifremer.fr)
- Norwegian Meteorological Institute (METNO), Norway.
  Contact: Mr. Øystein Torget (oysteint@met.no)

**Author(s)**

- Torill Hamre (torill.hamre@nersc.no)
- Alan Leadbetter (alead@bodc.ac.uk)
- Anthony Patterson (a.patterson@ucc.ie)
- Jorge Mendes de Jesus (jmdj@pml.ac.uk)
- Yassine Lassoued (Y.Lassoued@ucc.ie)
- Pete Walker (petwa@pml.ac.uk)
- Mike Grant (mggr@pml.ac.uk)

**Document approval**

- Document status: Revision 1
- WP leader approval: 2012-11-30
- Quality Manager approval: 2012-11-30
- Coordinator approval: 2012-11-30

# Revision History

| Issue | Date | Change records | Author(s) |
|---|---|---|---|
| Draft | 2012-10-16 | Outline of content. | T. Hamre |
| Draft | 2012-10-24 | Inserted description for the EuroICAN workshop. | D. Dunne |
| Draft | 2012-10-30 | Inserted description and slides for presentation at SeaDataNet Plenary Workshop. | A. Leadbetter |
| Draft | 2012-11-08 | Inserted description for the GEOSS AIP-5 Kickoff Workshop, and forthcoming workshop in Ocean Science Community and user demonstrations. | P. Walker |
| Draft | 2012-11-09 | Incorporated description of PEGASO workshop. | Y. Lassoued |
| Draft | 2012-11-29 | Incorporated slides from partners. | T. Hamre |
| Draft | 2012-11-30 | Inserted description of standardisation activities. Completed descriptions and executive summary. | D. Dunne, Y. Lassoued, T. Hamre |
| 1 | 2012-11-30 | First version of the report approved by the coordinator. | T. Hamre |

# Executive Summary

NETMAR aims to develop a pilot European Marine Information System (EUMIS) for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. It will be a user-configurable system offering flexible service discovery, access and chaining facilities using Open Geospatial Consortium (OGC), Open-source Project for a Network Data Access Protocol (OPeNDAP) and World Wide Web Consortium (W3C) standards. It will use a semantic framework coupled with ontologies for identifying and accessing distributed data, such as near-real time, forecast and historical data. EUMIS will also enable further processing of such data to generate composite products and statistics suitable for decision-making in diverse marine application domains.

This document describes dissemination activities carried out in the period May-October 2012, and some planned activities in the coming months:

- GEOSS AIP-5 Kick-Off Workshop, 3-4 May 2012
- PEGASO Hands-on Workshop, 23-36 October 2012
- SeaDataNet2 Plenary Meeting, 19-21 September 2012
- ICAN Semantics Cookbook (constituted of four parts: Understanding semantics, Understanding metadata, Establishing a CSW metadata catalogue with GeoNetwork opensource, Connecting your Atlas to the ICWA prototype)
- Semantic data delivery and processing services cookbook
- EuroICAN Workshop, 26 November 2012
- ICAN Conference call demonstration (December 2012)
- Arctic ROOS Annual Meeting, 6-7 November 2012
- Technical workshop on the future of data dissemination in the ocean science community, February 2013
- User testing and demonstrations at PML

Slides prepared for these events are included in appendices.

The semantic framework specification was submitted to the Chair of the GEOSS Data Sharing WG in mid August. The semantic framework specification is included in appendices. The submission of the Architecture specification activity to the GEOSS Best Practices Wiki is in progress. The submission of the architecture specification is scheduled as part of the NETMAR work plan to be finalised at the end of the project.

## Contents

# 1 Introduction

## *1.1 Background*

NETMAR aims to develop a pilot European Marine Information System (EUMIS) for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. It will be a user-configurable system offering flexible service discovery, access and chaining facilities using Open Geospatial Consortium (OGC), Open-source Project for a Network Data Access Protocol (OPeNDAP) and World Wide Web Consortium (W3C) standards. It will use a semantic framework coupled with ontologies for identifying and accessing distributed data, such as near-real time, forecast and historical data. EUMIS will also enable further processing of such data to generate composite products and statistics suitable for decision-making in diverse marine application domains. Figure 1-1 illustrates how observations, derived parameters and predictions are retrieved from a distributed service network through standard protocols, and delivered through the EUMIS portal using ontologies and semantic frameworks to select suitable products and where new products can be generated dynamically using chained processing services.

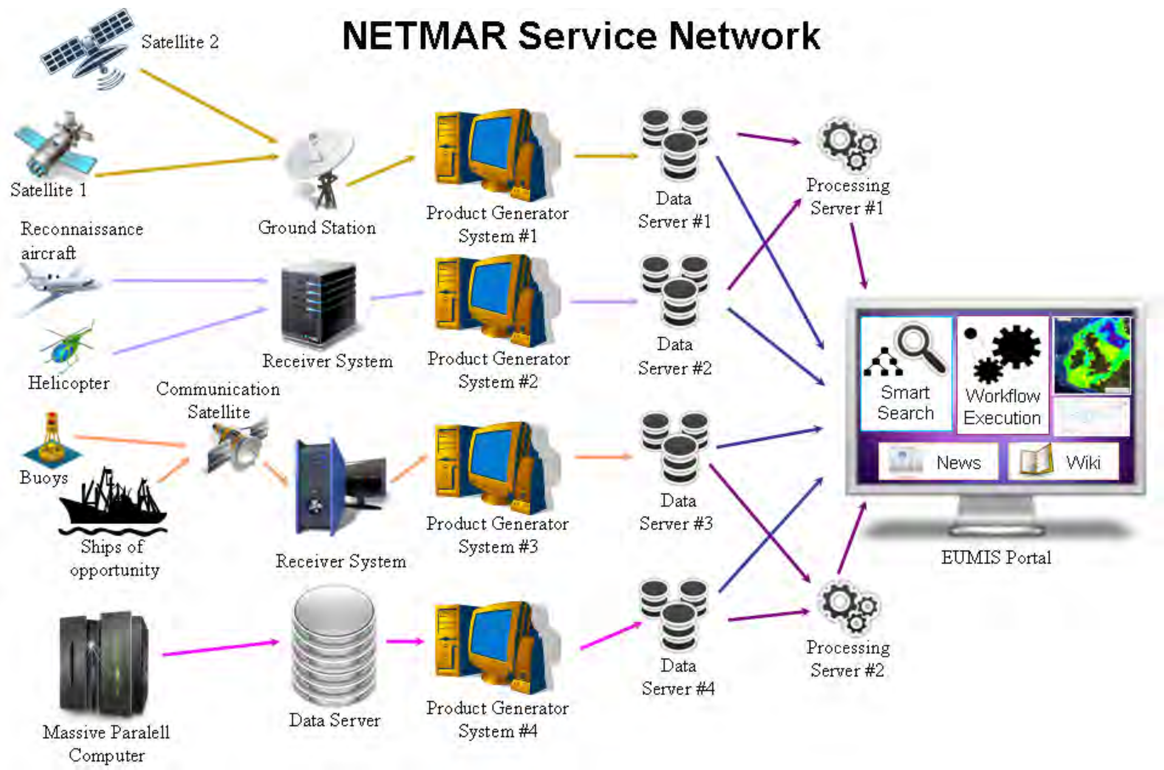



*Figure 1-1 The NETMAR service network.*

Four pilots have been defined as testbeds for the developed EUMIS components and the underlying semantic resources:

- Pilot 1: Arctic Sea Ice and Met-ocean Observing System
- Pilot 2: Oil spill drift forecast and shoreline cleanup assessment services in France
- Pilot 3: Ocean colour – Marine Ecosystem, Research and Monitoring
- Pilot 4: International Coastal Atlas Network (ICAN) for coastal zone management

## *1.2 Objective of this report*

This report describes the user workshops organised by the project in the period May-October 2012, as well as the workshops planned for the remainder of the project (November 2012 – January 2013). The report also describes the standardisation activities for the semantic framework specification carried out in the period May-October 2012.

## *1.3 Terminology*

A **vocabulary** can be either a list of terms or a list of terms and some text providing a definition of the term. A vocabulary ensures that terms are used, and spelt, consistently. A vocabulary can be extended in its power by providing definitions of concepts.

A **thesauri** expand the knowledge contained within a vocabulary by adding information about the relationships between the terms of the vocabulary. These relationships fall broadly into three categories:

- Synonyms – the current term is synonymous with a given, different term. e.g. “dogs” is synonymous with “canines”.
- Broader relations – the current term has a more specific definition than a given different term. e.g. “dogs” has a broader relationship to “pets”
- Narrower relations – the current term has a less specific definition than a given different term. e.g. “dogs” has a narrower relationship to “terriers”

In a more complex thesaurus, the concepts at the top of the hierarchy of broader and narrower relations may be stated explicitly, rather than being inferred by software agents. This provides the simplest form of a formal **ontology**.

A **portal** is a web site that collects input from a number of sources, and presents it in a uniform manner to the user. The portal content is perceived to come from the same source – the portal – while it typically is a combination of content from several sources, or an extract of selected content from a single external source (such as a news feed).

A **portlet** is portal component that can be deployed in a portal. A portlet can provide many types of functionality, among others, retrieve data from external sources, process and analyse data, present retrieved data on a geographic map. A portlet can also communicate with other portlets running in the same portal.

## *1.4 Organisation of this report*

Section 2 describes the presentation of NETMAR results at conferences/workshops and online during the period May-October 2012. Section 3 describes planned presentations at upcoming venues. Section 4 describes the standardisation activities for the semantic framework specification to the GEOSS Best Practice Wiki. Slides from the presentations and the submitted semantic framework specification are included in the appendices.

# 2 NETMAR workshops and presentations May-October 2012

## *2.1 GEOSS AIP-5 Kickoff Workshop, 3-4 May 2012*

PML attended the GEOSS AIP-5 Kickoff Workshop on May 3-4 2012 at UNEP International Environment House #1, Geneva.

During the semantic working group initial meeting it was proposed that the NETMAR project could be used as an example of good semantic integration of data, services and clients. It was also suggested that any developments made with the semantic working group should be in step with the semantic framework laid out within NETMAR. The project members of NETMAR were also given as potential contacts for the working group.

## *2.2 PEGASO Hands-on Workshop, 23-26 October 2012*

PEGASO (February 2010 to January 2014) is an EU FP7 project focusing on integrated coastal zone management (ICZM). Yassine Lassoued from CMRC attended the PEGASO Hands-on Workshop organised by the Flemish Marine Institute (VLIZ), and hosted by the International Oceanographic Commission (IOC) from 23rd to 26th October 2012, in Ostend, Belgium.

Yassine Lassoued presented the ICAN concept and the developments carried on as part of the NETMAR projects. A demonstration of the ICAN mediator and semantic framework was given, which received very positive and encouraging feedback. The PEGASO workshop was an opportunity for sharing our achievements in the fields of semantics and ontologies with the PEGASO community, and was a very good example of cross-fertilisation between FP projects.

## *2.3 SeaDataNet2 Plenary Meeting, 19-21 September 2012*

SeaDataNet is a Pan-Eueopean Infrastructure for Ocean & Marine Data Management. Adam Leadbetter, BODC, presented NETMAR at the SeaDataNet2 Plenary Meeting, Rhodes, Greece. 19 - 21 September 2012. There were 83 participants from Europe, America and Africa (full attendance list at http://www.seadatanet.org/Events/Plenary-meetings/First-annual-meeting/List-of-participants). The slides from this presentation are in the Appendices.

## *2.4 NETMAR Semantics Cookbook*

The revised NETMAR semantics cookbook [D7.9.2] was published on the public NETMAR web site on 7 August 2012. It was specifically targeted at the International Coastal Atlas Network (ICAN) and consists of four parts:

- Understanding Semantics
- Understanding Metadata
- Establishing a CSW metadata catalogue with GeoNetwork opensource
- Connecting your Atlas to the ICWA prototype

The cookbook as also announced in the ICAN Newsletter [IN12] in September 2012 on the ICAN web site (http://ican.science.oregonstate.edu/).

## 2.5 NETMAR semantic data delivery and processing service cookbook

A new cookbook on semantic data delivery and processing services [D7.8] was announced on the NETMAR public web site on 7 August 2012. The cookbook is primarily aimed at service providers/implementers who would also like to add semantic metadata to their services, and aims to provide guidance in the form of example XML and code snippets. It requires an understanding of the OGC WxS standards and XML. For those developing processing services familiarity with Python would also be useful.

# 3 Forthcoming user workshops

## 3.1 EuroICAN workshop 26th November 2012

A half-day workshop of European and African participants of ICAN will take place on Monday 26th November 2012 in IODE, Ostend, Belgium. A key objective of this workshop, organised to coincide with Littoral 2012, is to present and collect feedback from the ICAN and wider coastal community on the latest version of the NETMAR/ICAN sematic interoperability prototype and the NETMAR/ICAN cookbooks. It is also planned to be demonstrate WPS services and the NETMAR Service Chaining Editor at the workshop. A valuable resource in this respect is the GEOSS WPS tutorial [JGG+12] and the PyWPS tool [PyWPS], both developed with considerable effort from the NETMAR project. A draft agenda for the EuroICAN workshop is placed in the Appendices, as are the slides used to present the catalogue search and ontology extensions developed in the NETMAR project.

## 3.2 ICAN conference call demonstration (December 2012)

On Tuesday October 2nd 2012 the ICAN technical team convened a web conference call. During this conference call a live demo illustrating the current status of the NETMAR/ICAN prototype was presented. The final version of the NETMAR/ICAN prototype will be deployed online in late October 2012. While this version will be accessible online for the ICAN community to test, it was also decided to convene another web conference call in December 2012. This will enable a structured demonstration of the final version NETMAR/ICAN prototype, affording an opportunity for the ICAN group to give interactive feedback.

## 3.3 Arctic ROOS annual meeting 6-7 November 2012

During the Annual Arctic ROOS (Regional Ocean Observing System) Meeting, 6-7 November 2012, a short session is allocated for presentation of members' relevant projects and activities. During this session, NERSC presented selected results from the NETMAR project, focusing on the EUMIS pilot and the ice products developed for the Pilot 1 (Arctic Sea Ice Monitoring and Forecasting). The slides from this presentation are included in the appendices.

## 3.4 Technical workshop on the future of data dissemination in the Ocean Science community, February 2013

PML is planning a workshop aimed at leaders in development of the systems used by Ocean Scientists (and other data users) to find and work with increasing large data sets. The workshop will take place in a central location to enable the largest attendance. Areas of relevance to NETMAR will be

- Semantics (both usage and discovery). How can the NETMAR tools form the basis for further development?
- Web Processing Services. Are web accessible toolkits a useful way of dealing with the increasing complexity of data? How can the chaining of these services help scientists manage their data and develop new products?

Other projects and developers will be encouraged to attend. For instance, the EarthServer project will be presenting developments in the Web Coverage Processing Service (WCPS) and the handling of large data.

## 3.5 User testing and demonstrations at PML

PML has planned a user testing and demonstration session for members of the Western Channel Observatory at the end of November (subject to availability). The workshop will focus on how service chaining and the EUMIS portal can make it easier for WCO members to extract and compare data from the multiple (EO, in situ and model) data sources available. This will be combined with a user test session feeding in to D1.3.2

PML will also host a remote demonstration and discussion session with members of ChloroGIN based in the University of Cape Town. It is likely that there will be physical demonstrations in 2013, when meetings or the technical workshop allow combination of the demonstrations with other necessary international travel.

# 4 Standardisation activities

CMRC was in contact with a contact for the GEOSS Best Practice Wiki (BPW) during spring-summer 2012 to establish which area of the BPW the semantic framework specification should be submitted. The Chair of the GEOSS Data Sharing WG suggested the GEOSS Components and Services Registry and the BPW's semantic area. The latter wad found to be the best suited for the semantic framework specification, which was submitted to the GEOSS contact in mid August. Further contact was later made with the GEOSS AIP-5 team in October 2012, and the specification submitted to several members of the AIP-5 team. The submitted specification is included in the appendices.

The submission of the Architecture specification activity to the GEOSS Best Practices Wiki is under progress. The holding page http://wiki.ieee-earth.org/Best_Practices/GEOSS_Transverse_Areas/Data_and_Architecture/Service_Portal has been created. The submission is scheduled as part of the NETMAR work plan to be finalised at the end of the project.

# 5 References

[D7.8] D7.8 Semantic Data Delivery and Processing Services Cookbook, 2012. Available at http://netmar.nersc.no/sites/netmar.nersc.no/files/D7.8_Semantic-Data-and-Processing-Service-Cookbook_r1_20120730_0.pdf.

[D7.9.2] D7.9.2 ICAN semantic interoperability pilot cookbooks, 2012. Available at http://netmar.nersc.no/sites/netmar.nersc.no/files/D7.9.2_ICAN_semantic_cookbooks_r2_20120731_0.pdf.

[IN12] ICAN Newsletter, Volume 1, Number 2, September 2012. Available at http://dusk.geo.orst.edu/ICAN_EEA/ICAN_Newsletter_September_2012.pdf

[PyWPS] PyWPS Wiki, http://wiki.rsg.pml.ac.uk/pywps/Main_Page. (accessed 29 November 2012)

[JGG+12] de Jesus, Jorge, Simone Gianfranceschi, Benoît Gschwind, Andrea Marongiu, Lionel Menard and Sneha Rao, 2012, Web Processing Service Tutorial for GEOSS Providers. Available from http://wiki.ieee-earth.org/Documents/GEOSS_Tutorials/GEOSS_Provider_Tutorials/Web_Processing_Service_Tutorial_for_GEOSS_Providers.

# Appendices

## *Appendix A. EuroICAN meeting 26 November 2012 – Advance Program*

### International Coastal Atlas Network (ICAN)

**Meeting of European and African participants**

***Draft Agenda***

***Monday 26th November 2012 (2:00pm – 5:30pm)***

***Venue: North Sea Room, IODE, Ostend, Belgium***

A key objective of this workshop, organised to coincide with Littoral 2012, is to present and collect feedback from the ICAN and wider coastal community on the latest version of the NETMAR/ICAN prototype interoperability platform (funded under FP7), which facilitates the connection of multiple atlases and the search and viewing of metadata and data.

You will find out about the prototype functionality and will have the opportunity to provide input to assist development of the final version. You will also learn how to connect your own atlas to the platform and what resources are available to assist you in this task. Moreover, the utility of Web Processing Services and service chaining, used in other NETMAR application areas, will be demonstrated.

This meeting also provides the opportunity to launch officially the IODE ICAN Pilot Project in the lead-up to its establishment as a full IODE Project which will be proposed to the 22nd session of the IOC Committee on International Oceanographic Data and Information Exchange (IODE-XXII) in March 2013. We will also explore stories on the use of coastal and marine atlases, based on the experiences of the African and Caribbean Marine Atlases. The workshop provides an opportunity to review and develop the ICAN draft work plan to be presented at IODE-XXII, as well as preliminary discussions on the themes to be the focus of the next workshop of the full ICAN community in June 2013 (to be held in Victoria, Canada).

Attendance at the event is free and participants from outside the current ICAN community are most welcome, especially those currently using, or contemplating creation of, coastal or marine atlases.

Please contact Ned Dwyer (n.dwyer@ucc.ie) before **October 26th 2012** to reserve your place at the workshop.

Draft Agenda over page

**Meeting Conveners:**
Ned Dwyer, Coastal and Marine Research Centre, University College Cork, Ireland
Roger Longhorn, EUCC & IODE

**Agenda**

**Welcome and Objectives of Meeting**

**The NETMAR/ICAN Interoperability prototype**

Presentation of system features and functionality including:

- Ontology browsing
- Distributed metadata search and view
- Data visualisation

Comment and Feedback on the system

**Demonstration of utility of Web Processing Services and service chaining used in NETMAR**

**Coffee Break**

**Launch of ICAN IODE Pilot Project**

**Latest Developments in the African Marine Atlas and Caribbean Marine Atlas**

**Discussion of draft ICAN work plan including:**

- ICAN strategic plan
- The future of the ICAN interoperability prototype
- Communication plan
- Training Activities
- Technical Advice to AMA and CMA projects
- Guide on interacting with Users
- Technical cookbooks/guides

**Key themes for ICAN-6, Victoria, Canada**

**Closing Remarks**

----------------------------------------------------------------------------------------------------

Further Information will be posted on the ICAN web site in due course

ICAN: http://www.icoastalatlas.net

## *Appendix B. Final program and slides from the EuroICAN workshop, 26 November 2012*

The final program and the slides presented at the workshop are enclosed.

# International Coastal Atlas Network (ICAN)

## Meeting of European and African participants

***Agenda***

***Monday 26th November 2012 (2:00pm – 5:30pm)***

***Venue: North Sea Room, IODE, Ostend, Belgium***

A key objective of this workshop, organised to coincide with Littoral 2012, is to present and collect feedback from the ICAN and wider coastal community on the latest version of the NETMAR/ICAN prototype interoperability platform (funded under FP7), which facilitates the connection of multiple atlases and the search and viewing of metadata and data.

You will find out about the prototype functionality and will have the opportunity to provide input to assist development of the final version. You will also learn how to connect your own atlas to the platform and what resources are available to assist you in this task. Moreover, the utility of Web Processing Services and service chaining, used in other NETMAR application areas, will be demonstrated.

This meeting also provides the opportunity to launch officially the IODE ICAN Pilot Project in the lead-up to its establishment as a full IODE Project which will be proposed to the 22nd session of the IOC Committee on International Oceanographic Data and Information Exchange (IODE-XXII) in March 2013. We will also explore stories on the use of coastal and marine atlases, based on the experiences of the African and Caribbean Marine Atlases. The workshop provides an opportunity to review and develop the ICAN draft work plan to be presented at IODE-XXII, as well as preliminary discussions on the themes to be the focus of the next workshop of the full ICAN community in June 2013 (to be held in Victoria, Canada).

Agenda over page

November 2012

**Meeting Conveners:**
Ned Dwyer, Coastal and Marine Research Centre, University College Cork, Ireland
Roger Longhorn, EUCC

## Agenda

| | |
|---|---|
| **2:00 – 2:10 pm** | **Welcome and Objectives of Meeting (*Ned Dwyer*)** |
| **2:10 – 2:50 pm** | **The NETMAR/ICAN Interoperability prototype (*Declan Dunne, Adam Leadbetter*)**<br><br>Presentation of system features and functionality including:<br>• Ontology browsing<br>• Distributed metadata search and view<br>• Data visualisation<br><br>Comment and Feedback on the system |
| **3:00 – 3: 30** | **Demonstration of utility of Web Processing Services and service chaining used in NETMAR (*Declan Dunne*)** |
| **3:30 – 4:00pm** | **Coffee Break** |
| **4:00 – 4:15 pm** | **Launch of ICAN IODE Pilot Project (*Roger Longhorn*)** |
| **4:15 – 4: 30 pm** | **Latest Developments in the African Marine Atlas and Caribbean Marine Atlas (*Greg Reed*)** |
| **4:30 – 4:40 pm** | **A TopoBathy Database for Mozambique (*Charles de Jongh)*** |
| **4:40 pm – 5:15 pm** | **Discussion of draft ICAN work plan (*Roger Longhorn*) including:**<br>• ICAN strategic plan<br>• The future of the ICAN interoperability prototype<br>• Communication plan<br>• Training Activities<br>• Technical Advice to AMA and CMA projects<br>• Guide on interacting with Users<br>• Technical cookbooks/guides |
| **5:15 – 5:25 pm** | **Key themes for ICAN-6, Victoria, Canada (*Ned Dwyer*)** |
| **5:25 – 5:30 pm** | **Closing Remarks** (*Ned Dwyer*) |

---------------------------------------------------------------------------------------------------

ICAN: http://www.icoastalatlas.net

November 2012

# International Coastal Web Atlas (ICWA) prototype Version 3

Presenter: Declan Dunne

*d.dunne@ucc.ie*

Author: Yassine Lassoued

*y.lassoued@ucc.ie*

Coastal and Marine Research Centre

# Outline

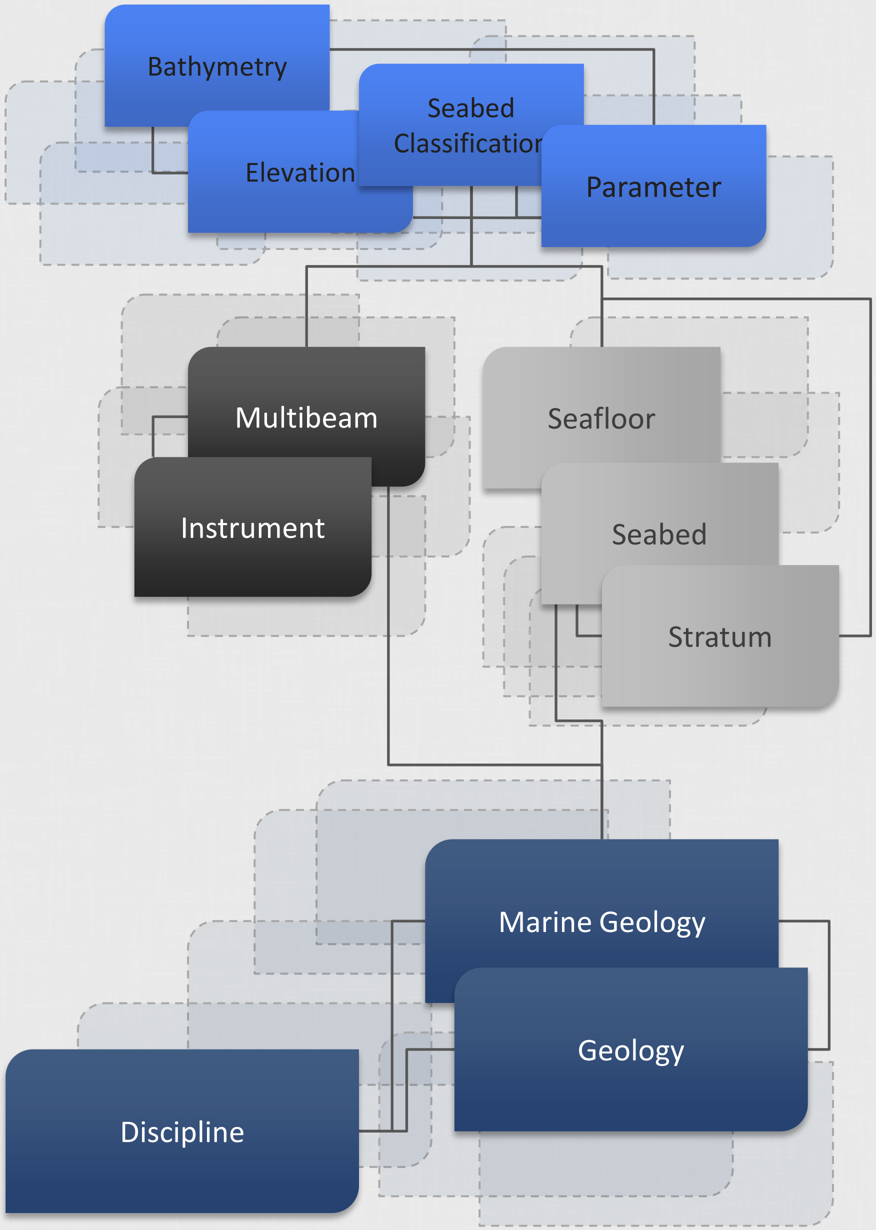
- Terminology
- Problem
- Approach
- Improvements
- Demonstration
- Connecting Atlases
- Current Work

**Interoperability**

Make distributed heterogeneous information systems (web services, databases, etc.) communicate

## Semantics

Meaning of “information” (data, metadata, etc.): term definitions, semantic relationships, etc.

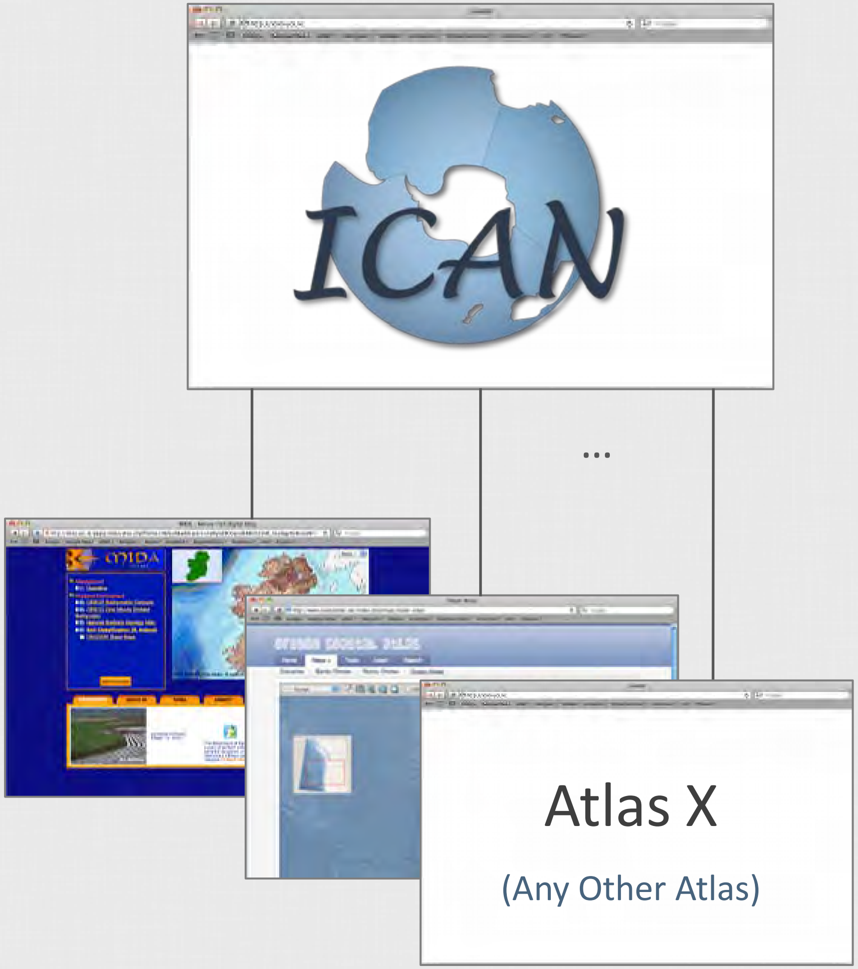


## Coastal Web Atlas (CWA)

Web application for the delivery of coastal resources, including: maps, geospatial data, metadata, thematic information

## CWA Semantic Interoperability

Providing seamless access to distributed, and semantically heterogeneous coastal web atlases



# Problem

- Heterogeneity:
  - Syntactic (data formats, query languages)
  - Structural (data schemas)
  - Semantic (meaning of data values)

Example: Metadata

- Different metadata standards (ISO-19115 vs. FGDC)
- Different vocabularies: 'Seabed' vs. 'Seafloor'
  'Coastline' vs. 'Shoreline'
  French, Spanish, English…

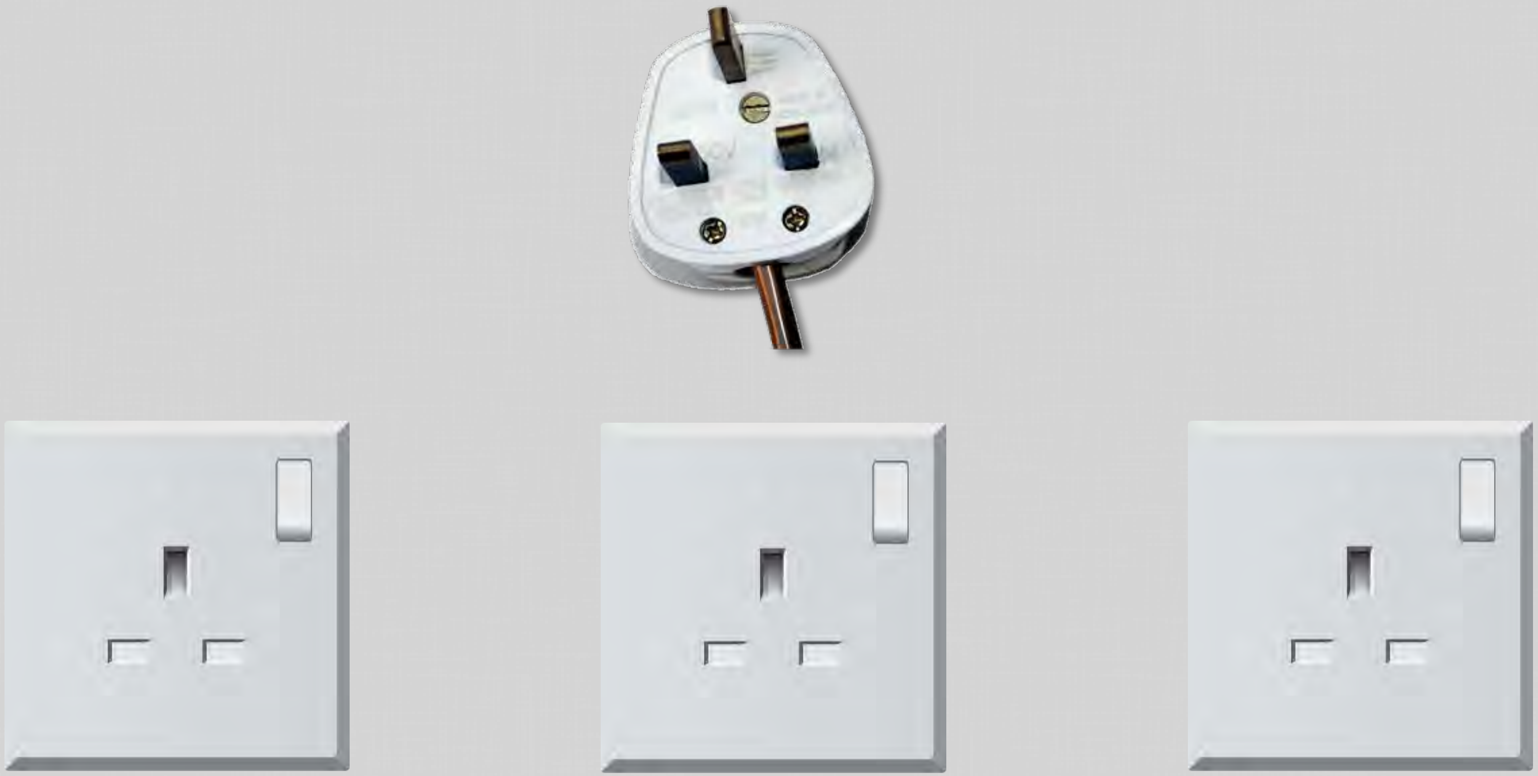
# Problem

*“Coastline”* *“Shoreline”* *“Ligne de côte”*

# Problem

# Approaches

## Approach 1: Standardisation



# Approaches

## Approach 2: Mediation (Adaptor)

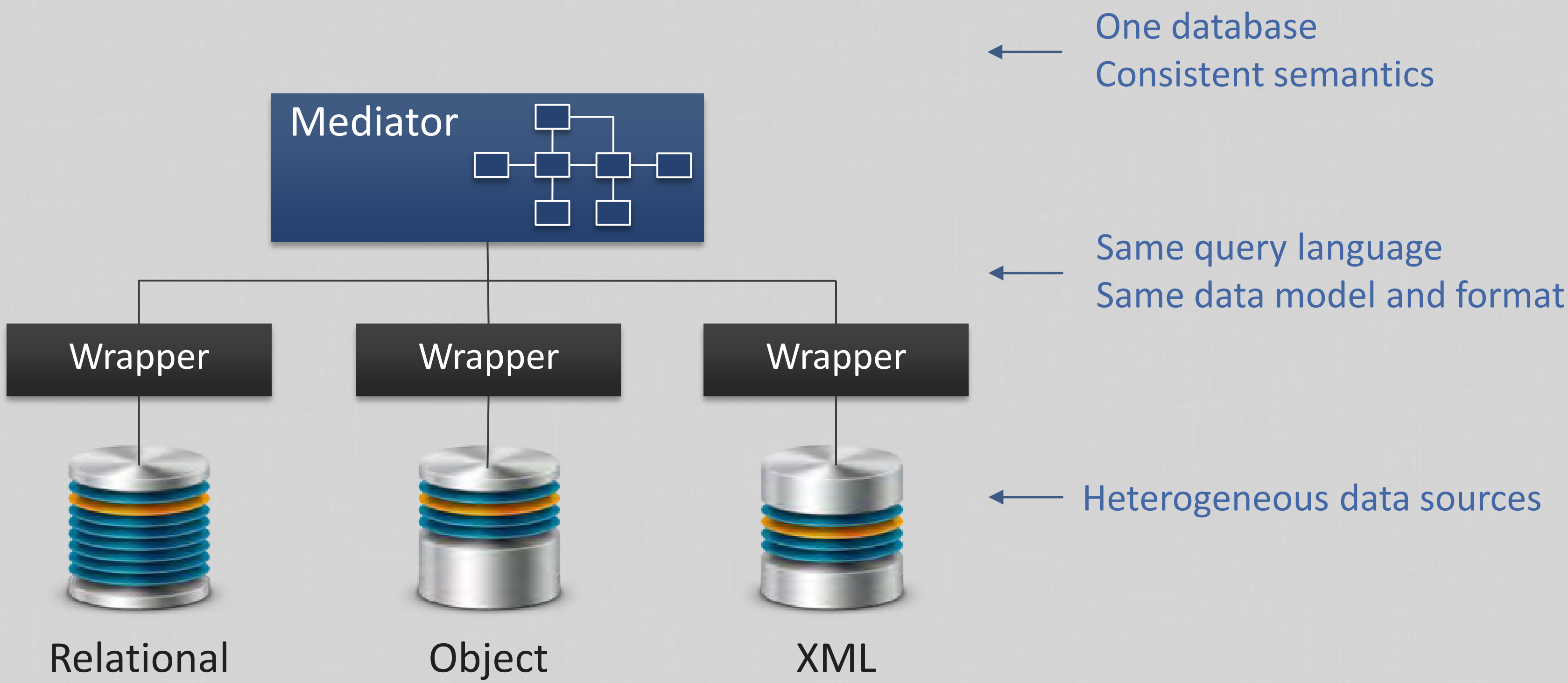
# Standardisation

- Standardise access interfaces and data formats
  - Implement OGC Web Services
  - Use metadata standards
    - ISO-19115, ISO-19139, ISO-19119, Dublin Core

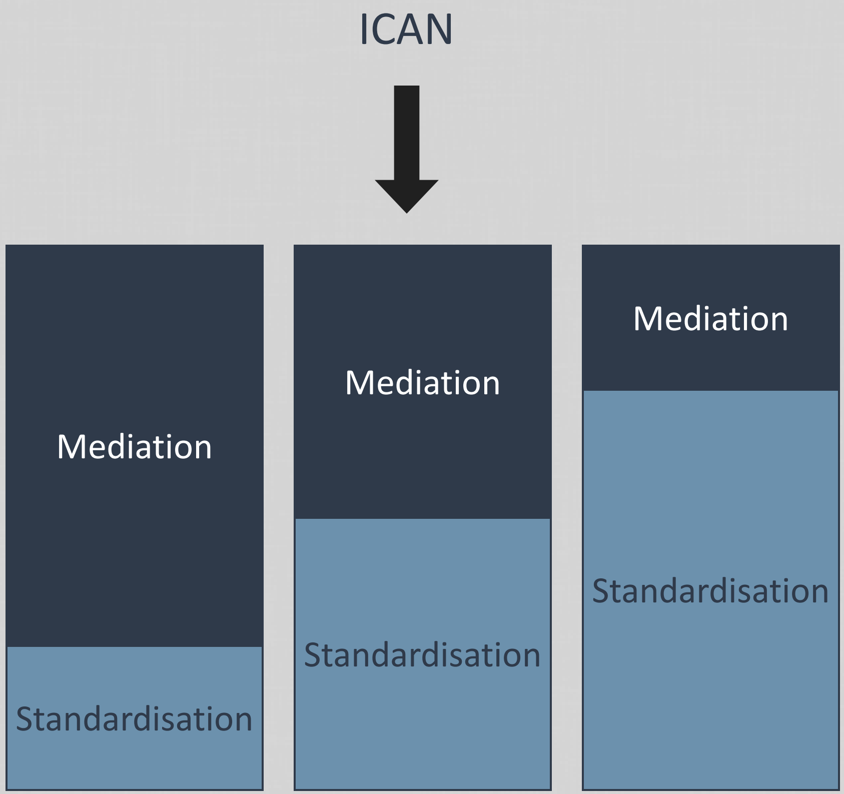
# Standardisation

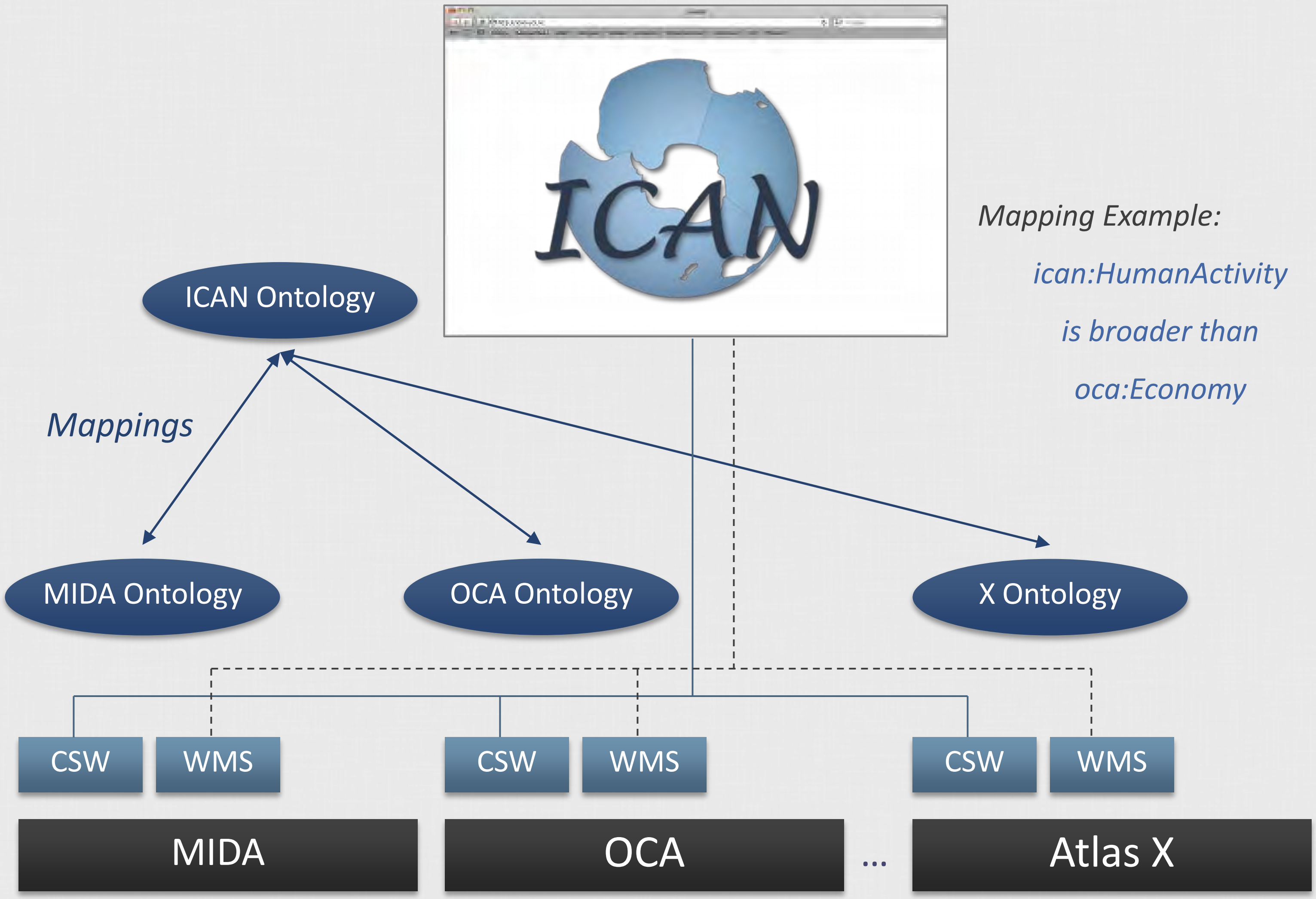
- Open Geospatial Consortium (OGC) Web Services
  - OGC specification
  - Interface allowing requests for geographic "*resources*" across the Web using platform-independent calls
  - Common OGC services:
    - Catalogue Service for the Web (CSW)
    - Web Feature Service (WFS)
    - Web Coverage Service (WCS)
    - Web Map Service (WMS)

# Mediation


Mediator
Wrapper
Wrapper
Wrapper
Relational
Object
XML
One database
Consistent semantics
Same query language
Same data model and format
Heterogeneous data sources

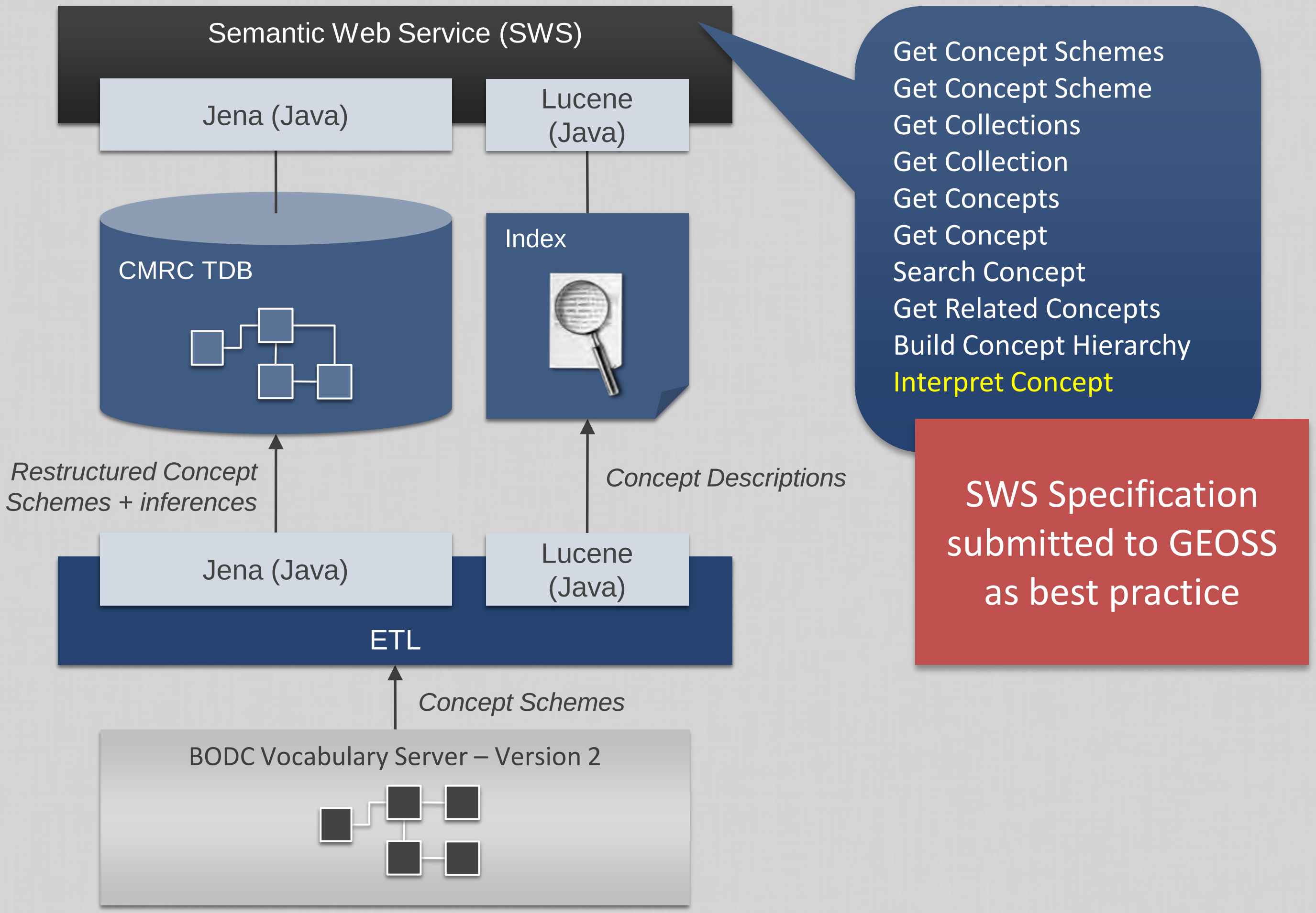
# Standardisation vs. Mediation



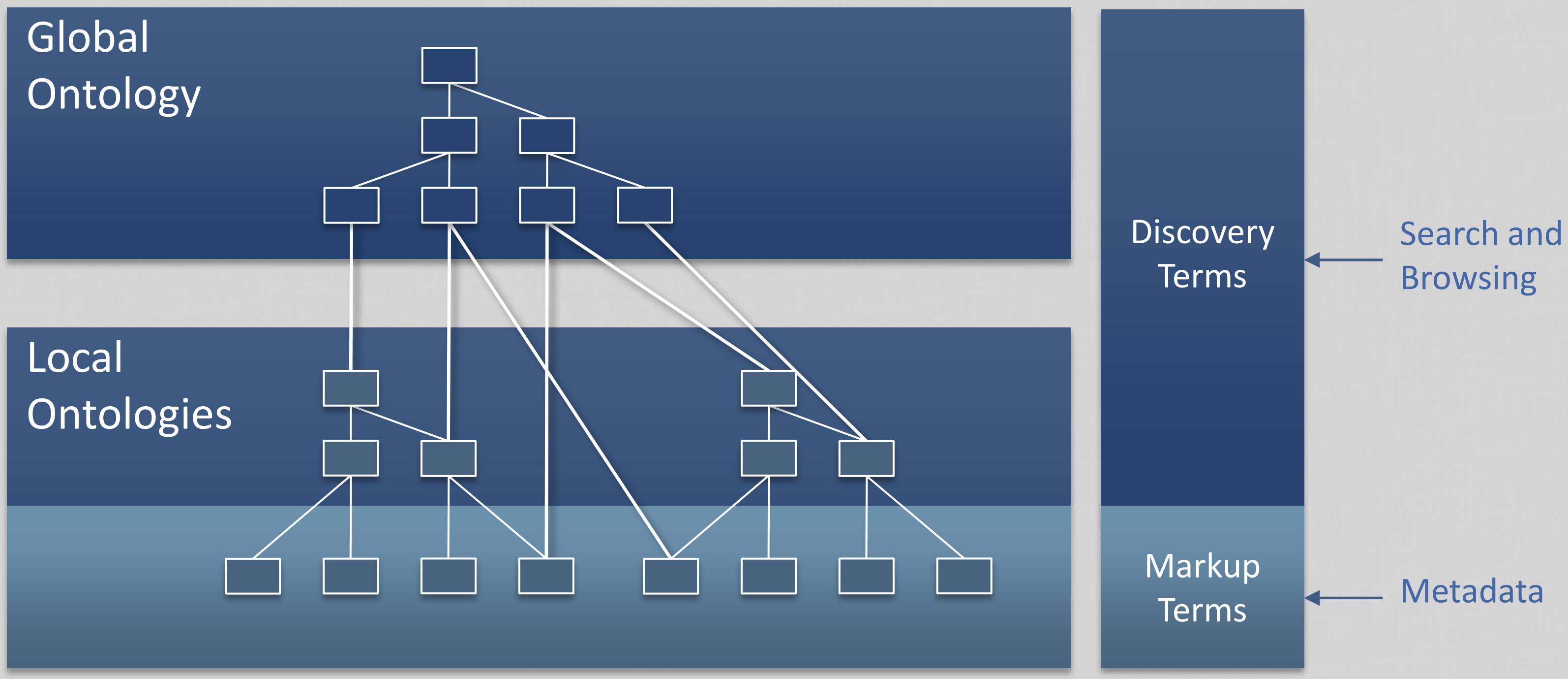


ICAN
Mapping Example:
ican:HumanActivity
is broader than
oca:Economy
ICAN Ontology
Mappings
MIDA Ontology
OCA Ontology
X Ontology
CSW
WMS
CSW
WMS
CSW
WMS
MIDA
OCA
...
Atlas X

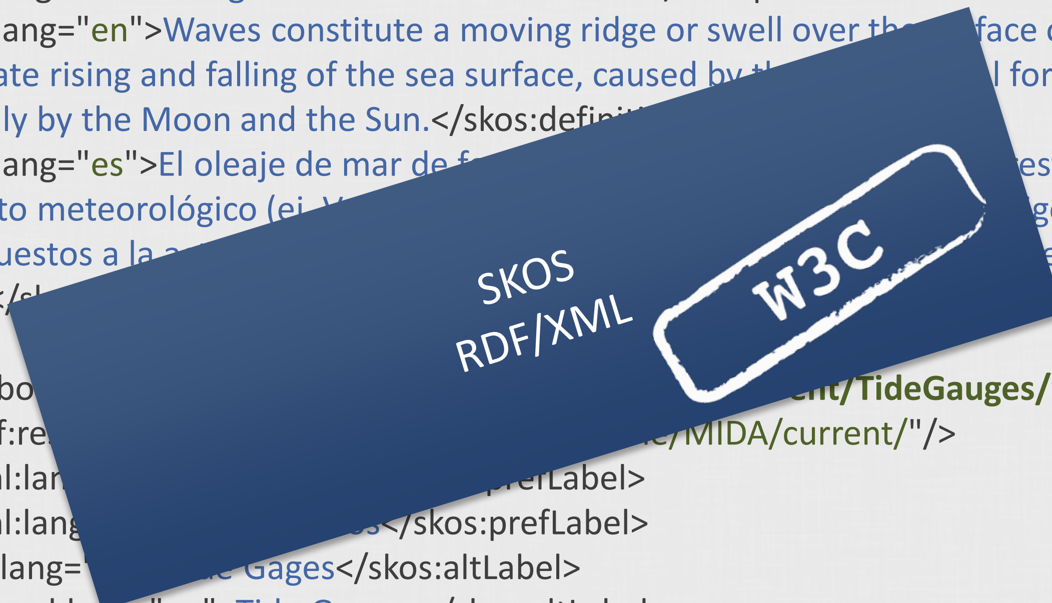
# Semantic Framework



# Ontology Structure



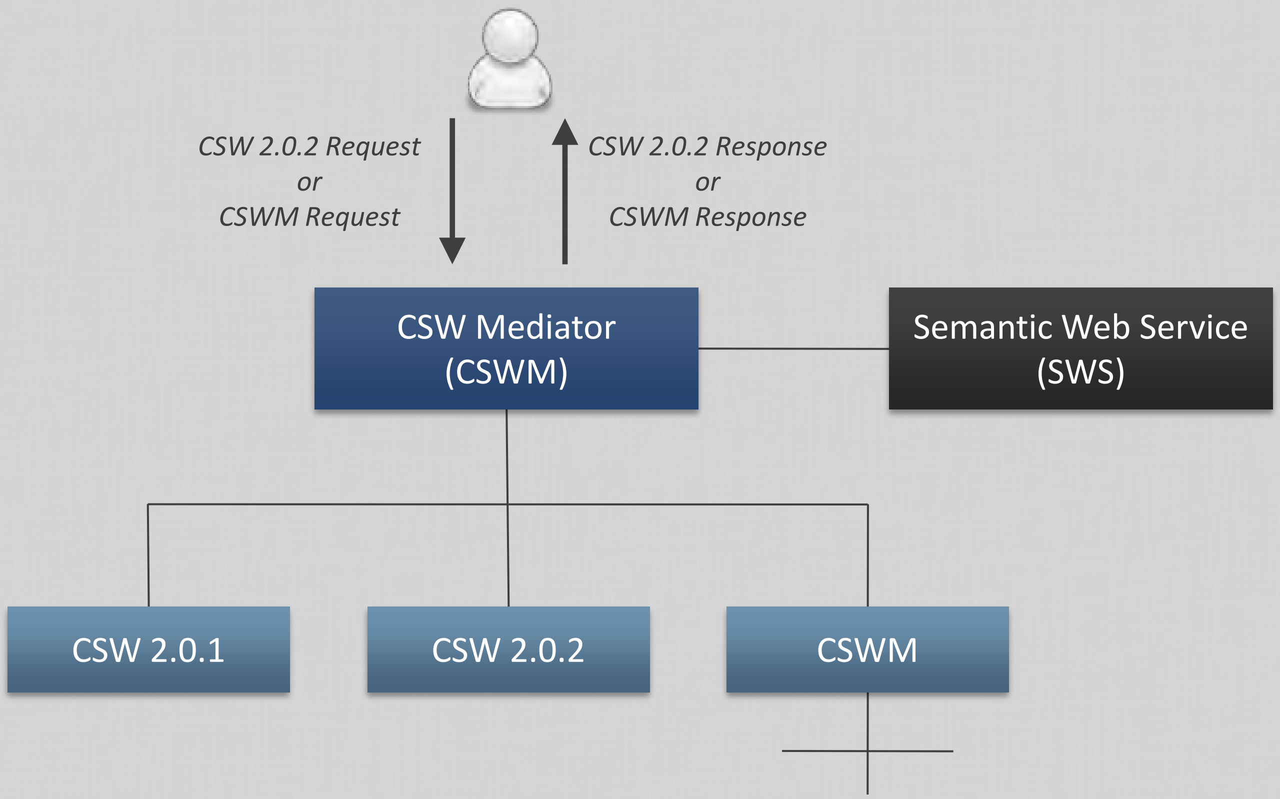
```xml
<rdf:RDF>
 <skos:Concept rdf:about="http://vocab.nerc.ac.uk/collection/A04/current/WavesAndTides/">
  <skos:inScheme rdf:resource="http://vocab.nerc.ac.uk/scheme/MIDA/current/"/>
  <skos:prefLabel xml:lang="en">Waves and Tides</skos:prefLabel>
  <skos:prefLabel xml:lang="es">Olas generadas en mar de fondo</skos:prefLabel>
  <skos:definition xml:lang="en">Waves constitute a moving ridge or swell over th… face of the sea or a lake.
   Tides are the alternate rising and falling of the sea surface, caused by t… l forces acting on the Earth's
   fluid surface primarily by the Moon and the Sun.</skos:defin…
  <skos:definition xml:lang="es">El oleaje de mar de f… estas, generadas en mar
   abierto por un evento meteorológico (ej. V… gen. Los patrones de
   oleaje están superpuestos a la a… era significativa la
   medida de las olas.</s…
  <skos:narrower>
   <skos:Concept rdf:abo… nt/TideGauges/">
    <skos:inScheme rdf:re… /MIDA/current/"/>
    <skos:prefLabel xml:lan… prefLabel>
    <skos:prefLabel xml:lang… s</skos:prefLabel>
    <skos:altLabel xml:lang="… Gages</skos:altLabel>
    <skos:hiddenLabel xml:lang="en">Tide Guages</skos:altLabel>
    <skos:definition xml:lang="en">A measuring instrument used to measure the level (and extremes) of tidal
     movement of sea levels at a point on the Earths surface.</skos:definition>
    <skos:definition xml:lang="es">Instrumento de medición utilizado para medir el nivel medio (y los extremos) del
     movimiento de las mareas en un punto sobre la superficie de la tierra.</skos:definition>
   </skos:Concept>
  </skos:narrower>
  <!-- More related terms -->
 </skos:Concept>
 <!--More concepts-->
</rdf:RDF>
```



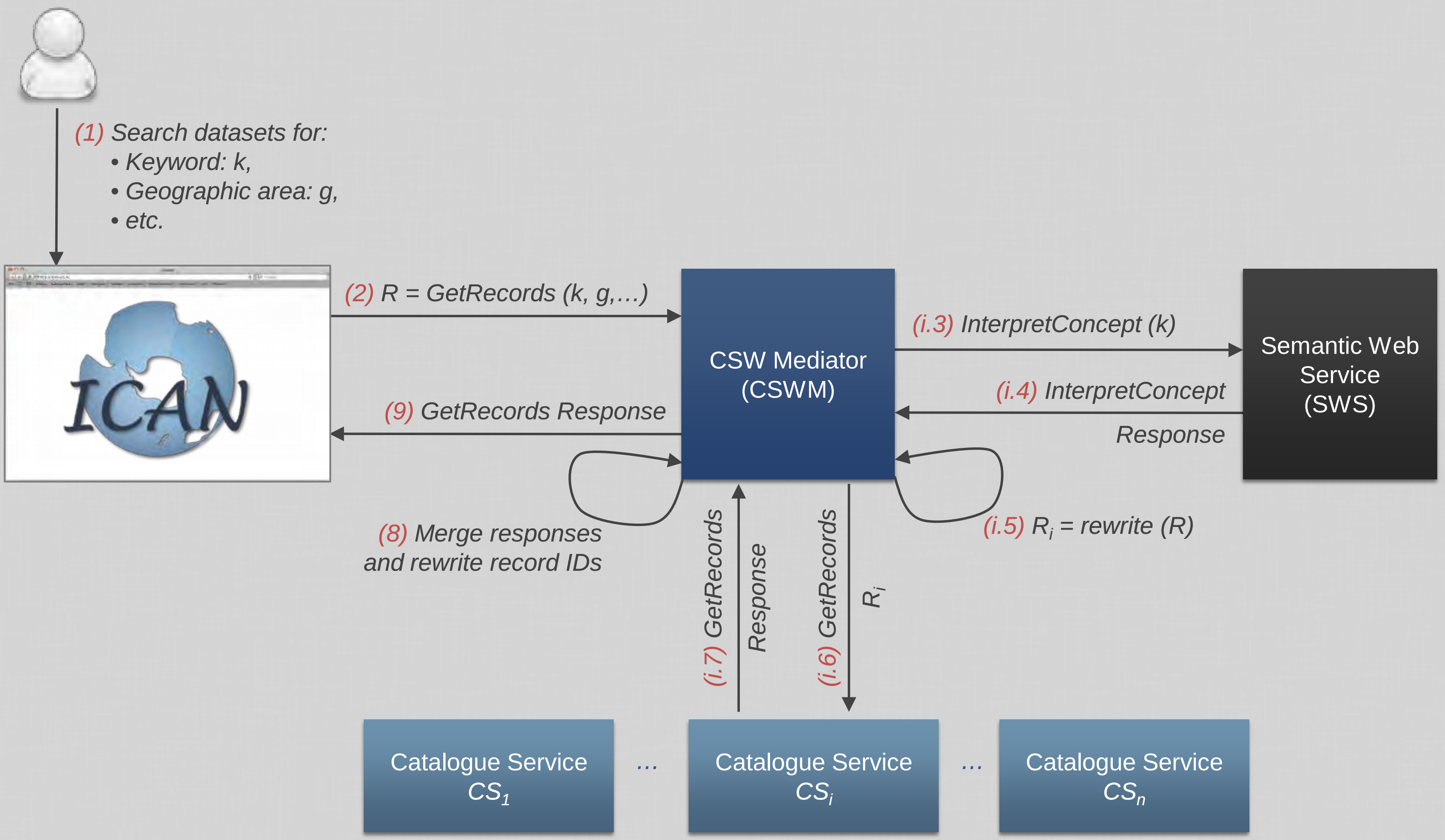
# Semantic Annotations

```
<gmd:MD_Metadata>

  ...

  <!--A list of keywords from the local thesaurus-->
  <gmd:MD_Keywords>
    <!--One keyword-->
    <gmd:keyword>
      <gmx:Anchor
       xlink:href=”http://vocab.nerc.ac.uk/collection/A04/current/Shipwrecks/”>
       Shipwrecks
      <gmx:Anchor>
    </gmd:keyword>
    <!--You may include as many keywords as you wish-->
    ...
  </gmd:MD_Keywords>

  ...

</gmd:MD_Metadata>
```

# CSW Mediation Architecture



# CSW Mediation Work Flow



# Query Rewriting

- Rewrite a user's request into requests supported by local catalogues
  - Translate query format
    - E.g., CSWM to CSW 2.0.2, CSW 2.0.2 to CSW 2.0.1, etc.
  - Translate term semantics

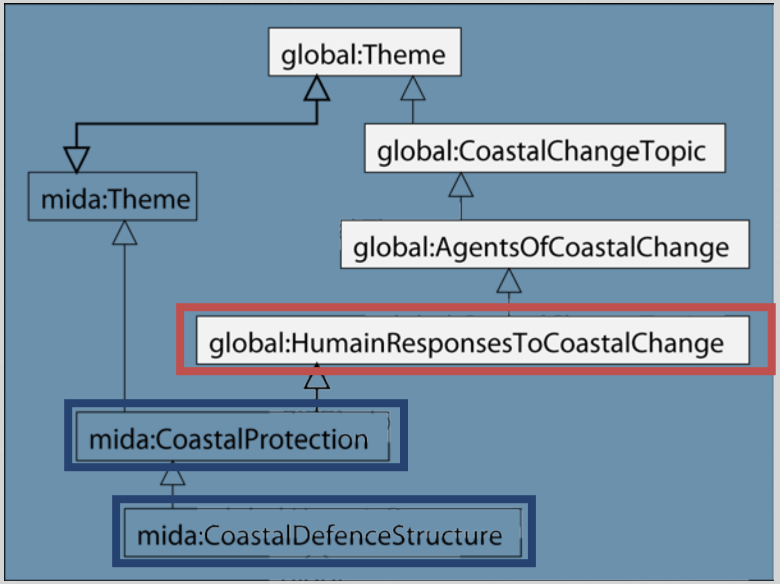
```
http://ican2.ucc.ie/icansrv/Explorer?
request=GetRecords&service=CSW&version=2.0.2
&resultType=results
&namespace=csw:http://www.opengis.net/cat/csw
&maxRecords=1000
&elementSetName=summary
&constraint=
<?xml version="1.0" encoding="UTF-8"?>
<Filter xmlns=http://www.opengis.net/ogc xmlns:gml=http://www.opengis.net/gml
   xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <And>
        <PropertyIsLike wildCard="%" singleChar="_" escape="\">
            <PropertyName>keyword</PropertyName>
            <Literal>HumanResponsesToCoastalChange%</Literal>
        </PropertyIsLike>
        <BBOX>
            <PropertyName>/csw:Record/ows:BoundingBox</PropertyName>
            <gml:Envelope
              srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:lowerCorner>-180 -90</gml:lowerCorner>
                <gml:upperCorner>180 90</gml:upperCorner>
            </gml:Envelope>
        </BBOX>
    </And>
</Filter>
&constraintLanguage=FILTER
&constraint_language_version=1.1.0
```

```
http://ican2.ucc.ie/icansrv/Explorer?
request=GetRecords&service=CSW&version=2.0.2
&resultType=results
&namespace=csw:http://www.opengis.net/cat/csw
&maxRecords=1000
&elementSetName=summary
&constraint=
<?xml version="1.0" encoding="UTF-8"?>
<Filter xmlns=http://www.opengis.net/ogc xmlns:gml=http://www.opengis.net/gml
   xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <And>
        <PropertyIsLike wildCard="%" singleChar="_" escape="\">
            <PropertyName>keyword</PropertyName>
            <Literal>HumanResponsesToCoastalChange%</Literal>
        </PropertyIsLike>
        <BBOX>
            <PropertyName>/csw:Record/ows:BoundingBox</PropertyName>
            <gml:Envelope
              srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:lowerCorner>-180 -90</gml:lowerCorner>
                <gml:upperCorner>180 90</gml:upperCorner>
            </gml:Envelope>
        </BBOX>
    </And>
</Filter>
&constraintLanguage=FILTER
&constraint_language_version=1.1.0
```

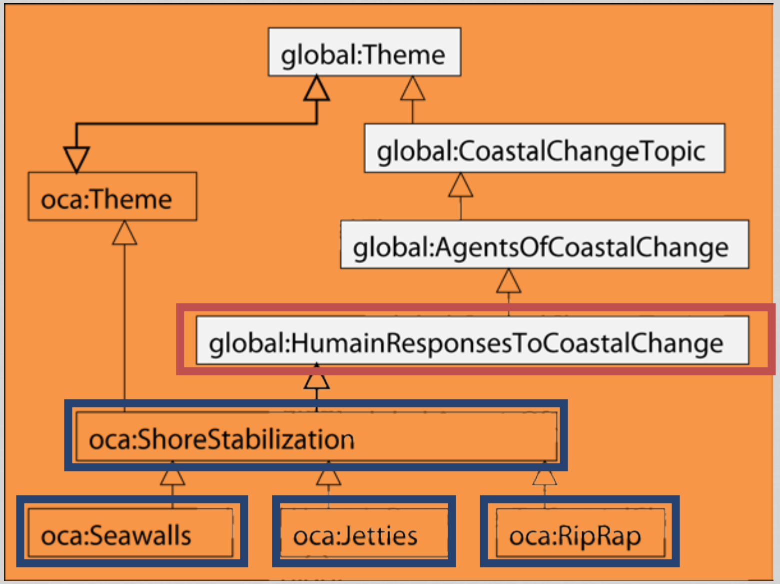
# Term Translation

- CSW Mediator uses the semantic web service to translate (interpret) global terms into local terms

MIDA Mappings




OCA Mappings



Global

```
<PropertyIsLike wildCard="%“ singleChar="_" escape="\">
    <PropertyName>keyword</PropertyName>
    <Literal>HumanResponsesToCoastalChange%</Literal>
</PropertyIsLike>
```

MIDA

```
<Or>
    <PropertyIsLike wildCard="%“ singleChar="_"
      escape="\">
        <PropertyName>keyword</PropertyName>
        <Literal>CoastalProtection%</Literal>
    </PropertyIsLike>
    <PropertyIsLike wildCard="%“ singleChar="_"
      escape="\">
        <PropertyName>keyword</PropertyName>
        <Literal>CoastalDefenceStructure%</Literal>
    </PropertyIsLike>
</Or>
```

# Improvements

- Standard ontology model: SKOS
- Multilingual ontologies
  - MIDA: English, Spanish
  - ICAN: English, Spanish, French, Norwegian, etc.
  - Smart multilingual search
- New ontology mappings:
  - MIDA - INSPIRE
  - MIDA - OCA
- New graphical user interface
  - Map viewer under development
- Improved performance:
  - Node requests processed in parallel
- Improved robustness and fixed bugs
- CSW Mediator has CSW 2.0.2 and CSWM 1.0 interfaces
- CSW Mediator supports CSW 2.0.1 and 2.0.2, and CSWM 1.0 nodes
- Improved code structure

Demo

# HTTP://ICAN2.UCC.IE/ATLAS

# Connecting Atlases

- Metadata delivered through CSW 2.0.2 (or 2.0.1)
- Metadata may use a controlled vocabulary
  - If so, controlled vocabulary (SKOS) needs to be stored in the NERC Vocabulary Server
  - You may want to reuse MIDA, and OCA vocabularies and extend them with new terms
- Metadata should point to WMS links

# Connecting Atlases

- Cookbooks
  - Understanding Semantics
  - Understanding Metadata
  - Establishing a CSW metadata catalogue with GeoNetwork
  - Connecting your Atlas to the ICWA prototype

- Cookbooks can be downloaded from:

  http://netmar.nersc.no/sites/netmar.nersc.no/files/D7.9.2_ICAN_semantic_cookbooks_r2_20120731_0.pdf

# Current Work

- Map viewer
- Connect more atlases
- Launch

# Thanks

Presenter: Declan Dunne

*d.dunne@ucc.ie*

Author: Yassine Lassoued

*y.lassoued@ucc.ie*

Coastal and Marine Research Centre

PML | Plymouth Marine Laboratory

Marine Matters

# NETMAR
# Service Chaining Editor

Presenter: Declan Dunne
Coastal and Marine Research Centre
*d.dunne@ucc.ie*

Authors: Pete Walker, *et al.*
Plymouth Marine Laboratory
*petwa@pml.ac.uk*

NETMAR

SEVENTH FRAMEWORK PROGRAMME



## Introduction

- What is a WPS?
- What is a Service Chain (or workflow)?
- How does the NETMAR Service Chaining Editor work?
- Demo
- Summary

## What is a WPS?

- The OGC Web Processing Service (WPS) provides a standard for implementing geospatial processing as a Web service.
- The WPS standard defines how a client can request the execution of a process, and how the output from the process is handled.
- WPS defines three operations:
  - **GetCapabilities** (What processes are available)
  - **DescribeProcess** (What the process does. What its inputs and outputs are)
  - **Execute** (Runs the process and returns results)



## What does a WPS do?

- WPS can describe any calculation (i.e. process) including all of its inputs and outputs, and trigger its execution as a Web service.
- Data can be local or on network.
- Designed to work with spatially referenced data but can be used with any kind of data.
- The specific processes defined by the owner of that server.
- WPS makes it possible to publish, find, and bind to processes in a standardized and thus interoperable fashion.

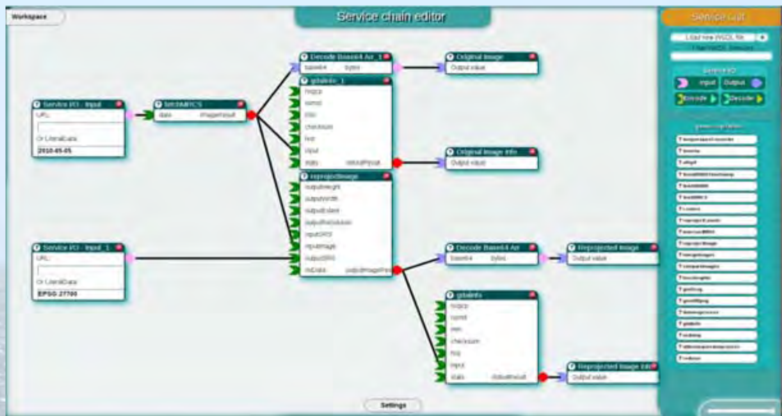
## What is a Service Chain?

- AKA (Scientific) Workfow.
- A way of chaining simple modular processes together to form more complex systems.
- Very popular in bio-informatics but becoming more widely used.
- Often have nice GUIs to allow a workflow to be created by drag and drop.
- One example is Taverna, http://www.taverna.org.uk, (on which the NETMAR Service Chaining is built).



## How does the NETMAR Service Chaining Editor work?

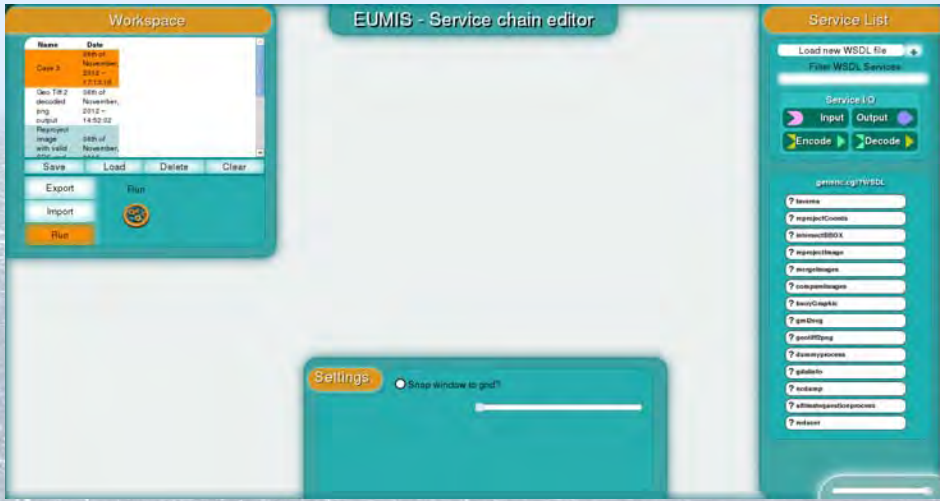
- Fully web based system.
- Service Chaining Editor runs in web browser.




- Service Execution Engine runs on server and executes the workflows created by the editor.
- Workflows are held in XML and can be saved and emailed as required.
- Execution Output is held on the server and retrieved via a web access.

## NETMAR Service Chaining Editor

- The SCE is made up of an editing canvas on which the work flow is created and three control windows
    - Workspace
    - Service List
    - Settings




- A workflow is created by dragging services from the Seervice List onto the canvas and joining the inputs and outputs

PML | Plymouth Marine Laboratory

## NETMAR Service Chaining Editor Demo

# Summary

- WPS allows us to execute processes running on web based servers
- Service Chaining allows us to combine these processes into more complex programs
- The NETMAR Service Chaining Editor lets us do this from the comfort of our web browser

# ICAN & Web 3.0

Adam Leadbetter
*alead@bodc.ac.uk*
British Oceanographic Data Centre

## Overview

- A little Web history
- Online Controlled Vocabularies...
  - ... and some of their uses
- ICAN use case...
  - ... and its implementation

## A little Web history

- Web 1.0 - The read-only web

## A little Web history

- Web 1.0 - The read-only web
- Web 2.0 - Interaction, collaboration, social
    - Wiki sites
    - Twitter, Facebook
    - Docs / Drive

# A little Web history

- Web 1.0 - The read-only web
- Web 2.0 - Interaction, collaboration, social
  - Wiki sites
  - Twitter, Facebook
  - Docs / Drive
- Web 3.0 - The semantic web
  - Unstructured web of documents...
  - ... becomes a structured web of data

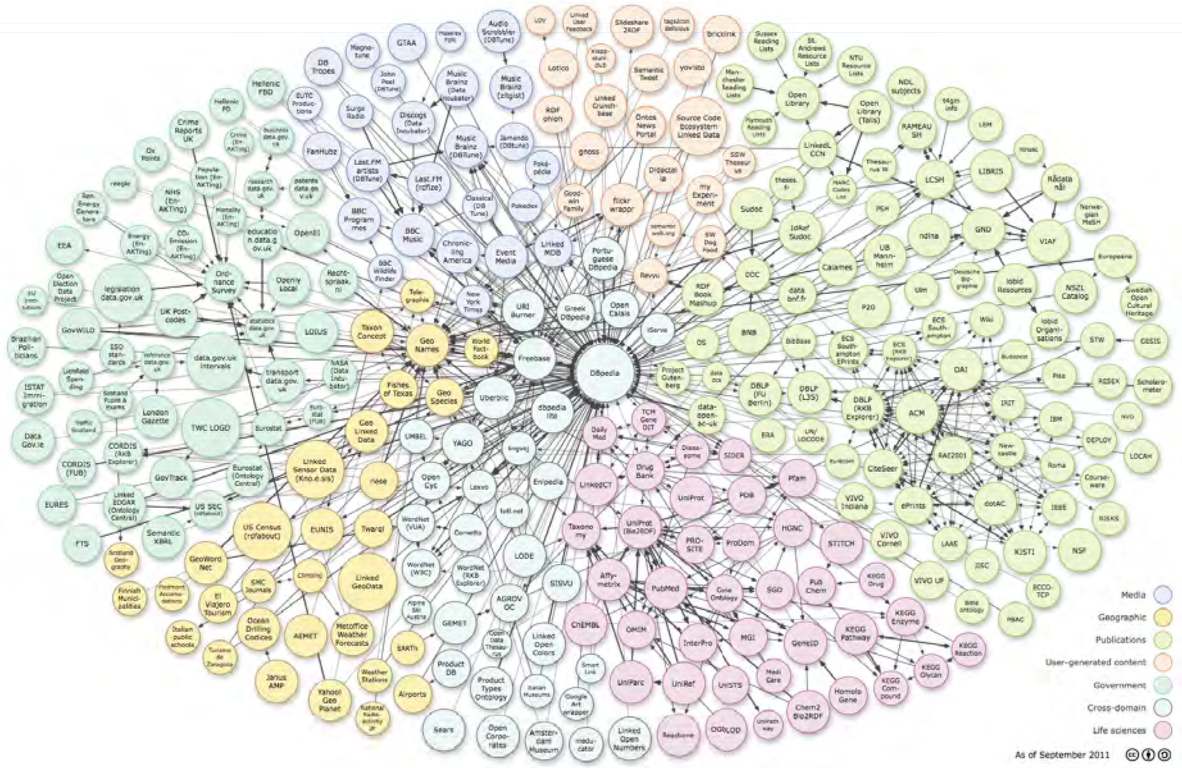
ICAN BODC British Oceanographic Data Centre NATURAL ENVIRONMENT RESEARCH COUNCIL NETMAR

# A little Web history

- Web 1.0 - The read-only web
- Web 2.0 - Interaction, collaborati
  - Wiki sites
  - Twitter, Facebook
  - Docs / Drive
- Web 3.0 - The semantic web
  - Unstructured web of documents...
  - ... becomes a structured web of data
    - Sir Tim Berners-Lee, W3C, Weaving the Web (1999)

ICAN BODC British Oceanographic Data Centre NATURAL ENVIRONMENT RESEARCH COUNCIL NETMAR

# The future of the web?




# Online controlled vocabularies

- What are they?
  - A collection of concepts for populating a given metadata field

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Online controlled vocabularies

- What are they?
  - A collection of concepts for populating a given metadata field
  - Ensure consistent spellings & syntax

ICAN
British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL
NETMAR


# Online controlled vocabularies

- What are they?
  - A collection of concepts for populating a given metadata field
  - Ensure consistent spellings & syntax

  - Prevent metadata misunderstandings


ICAN
British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL
NETMAR

## Online controlled vocabularies

- What are they?
  - A collection of concepts for populating a given metadata field
  - Ensure consistent spellings & syntax
  - Prevent metadata misunderstandings
  - Maintain a static relationship between metadata fields and the real world

## Online controlled vocabularies

- What are they?
  - A collection of concepts for populating a given metadata field
  - Ensure consistent spellings & syntax
  - Prevent metadata misunderstandings
  - Maintain a static relationship between metadata fields and the real world
  - Concepts from different controlled vocabularies may be connected using simple mapping relationships

# Online controlled vocabularies

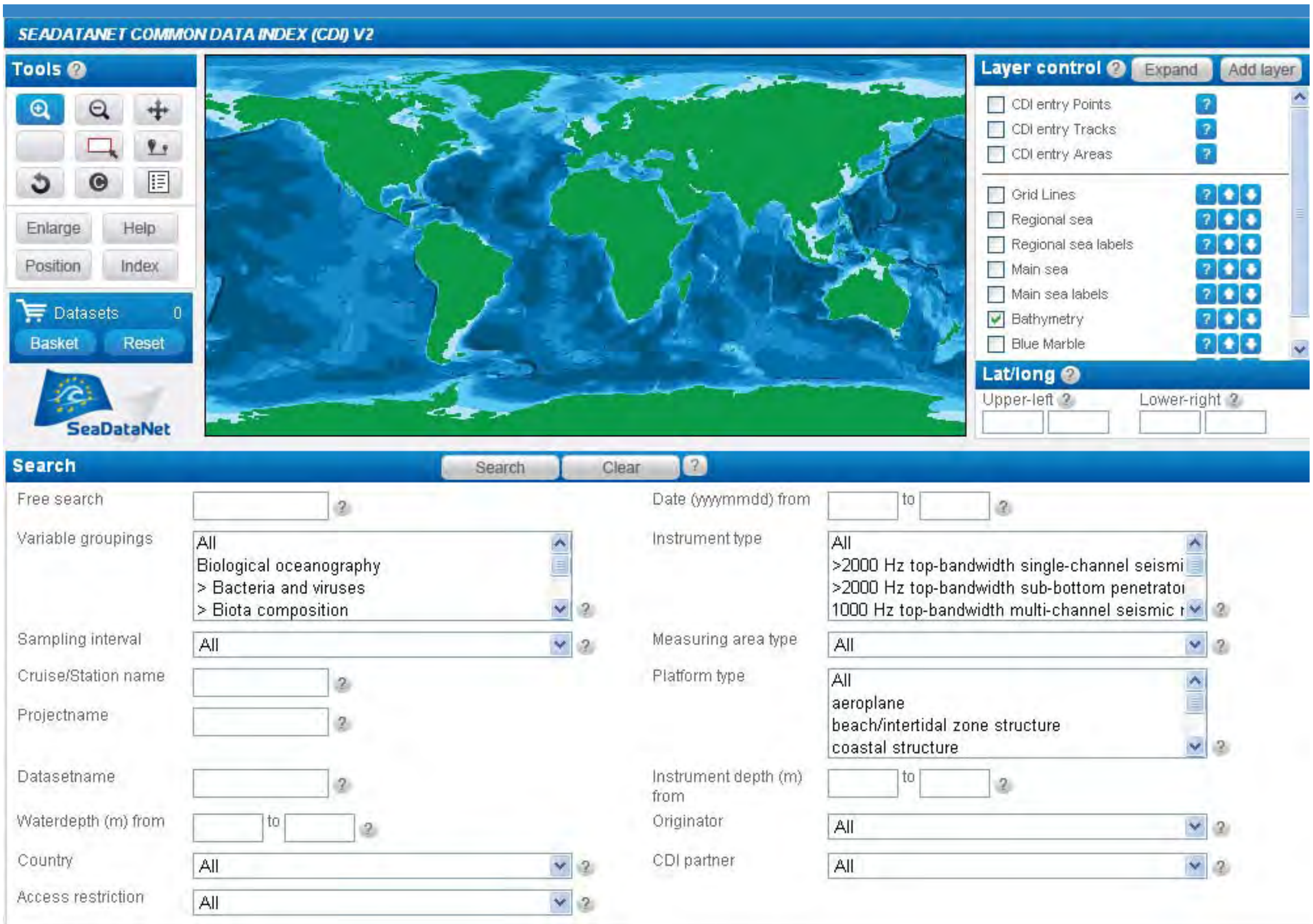
- What are they?
  - A collection of concepts for populating a given metadata field
  - Ensure consistent spellings & syntax
  - Prevent metadata misunderstandings
  - Maintain a static relationship between metadata fields and the real world
  - Concepts from different controlled vocabularies may be connected using simple mapping relationships
  - Web accessible - you can browse to them

ICAN BODC British Oceanographic Data Centre NATURAL ENVIRONMENT RESEARCH COUNCIL NETMAR

# Online controlled vocabularies

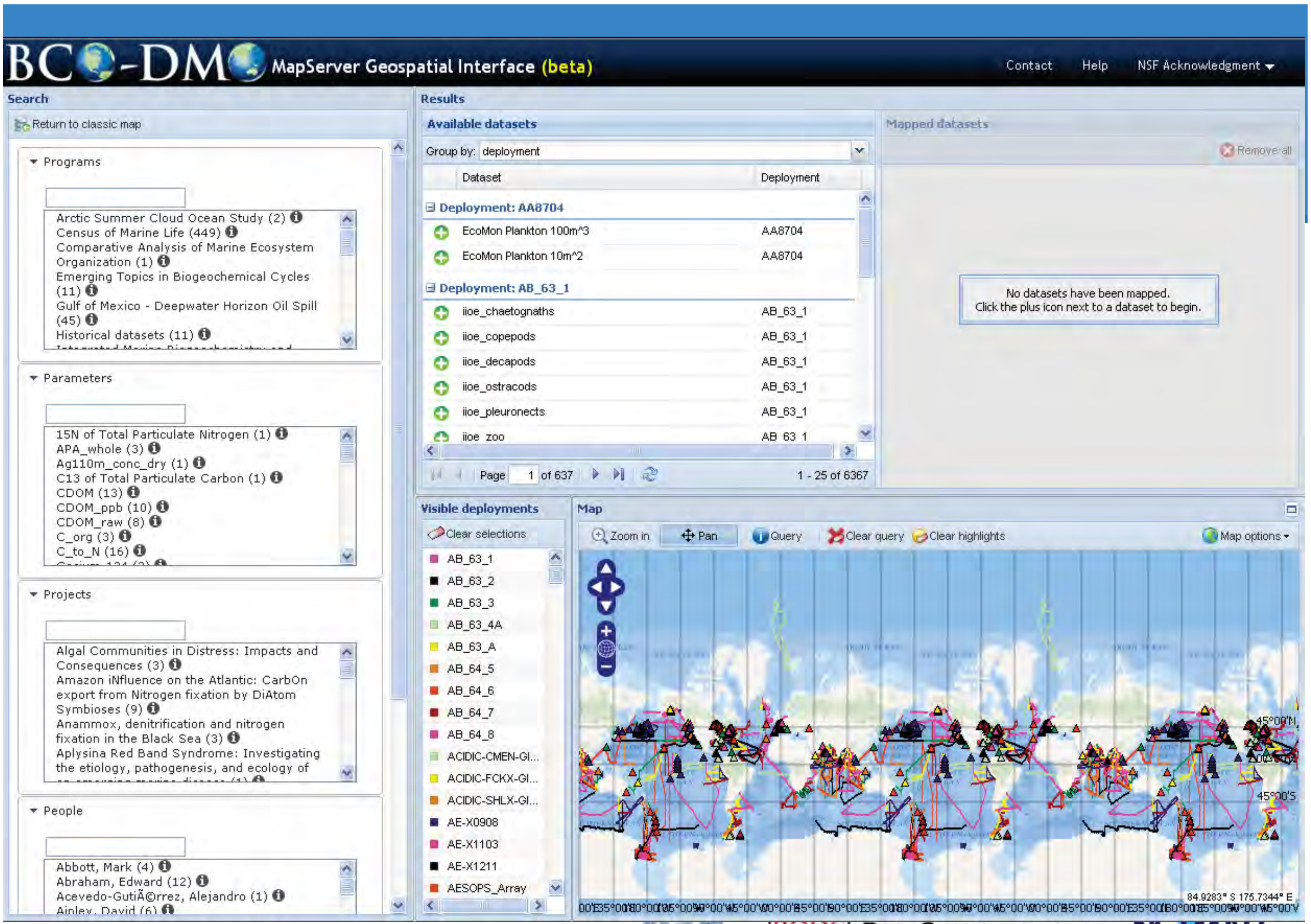
- What are their uses?
  - SeaDataNet

ICAN BODC British Oceanographic Data Centre NATURAL ENVIRONMENT RESEARCH COUNCIL NETMAR




# Online controlled vocabularies

- What are their uses?
  - SeaDataNet
  - BCO-DMO




# Online controlled vocabularies

- What are their uses?
  - SeaDataNet
  - BCO-DMO
  - Climate and Forecast

ICAN
BODC British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL
NETMAR

# Online controlled vocabularies

- What are their uses?
    - SeaDataNet
    - BCO-DMO
    - Climate and Forecast
        - Standard Name:
          "surface_upward_mass_flux_of_carbon_dioxide_expressed_as_carbon_due_to_emission_from_crop_harvesting"
        - EEA pollutant: *"Carbon dioxide"*
        - GEMET source: *"harvest"*

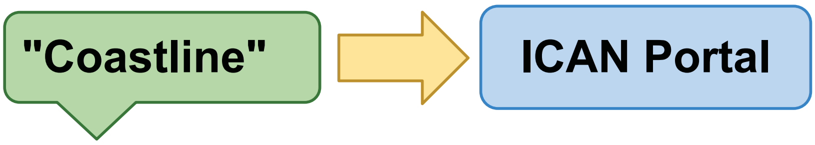
ICAN British Oceanographic Data Centre NATURAL ENVIRONMENT RESEARCH COUNCIL NETMAR

# ICAN "Semantic" Use Case

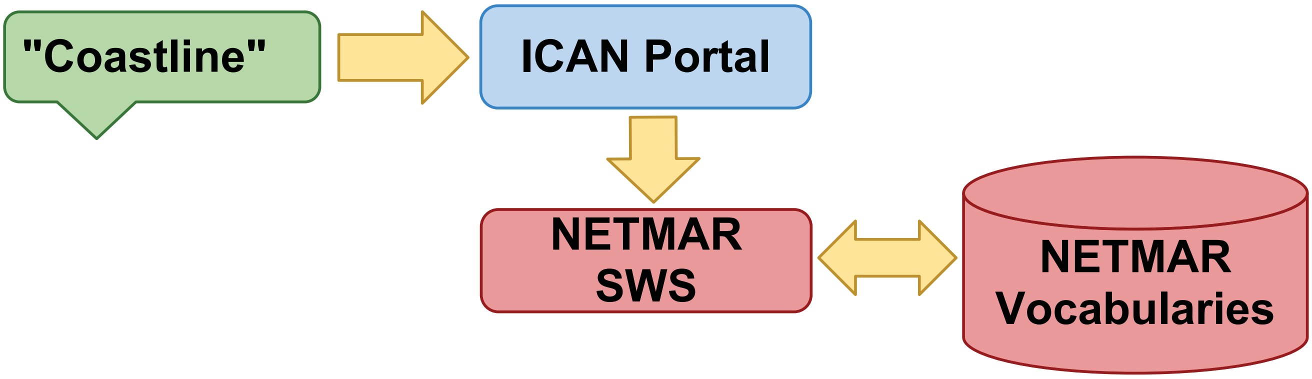
**"Coastline"**

ICAN British Oceanographic Data Centre NATURAL ENVIRONMENT RESEARCH COUNCIL NETMAR

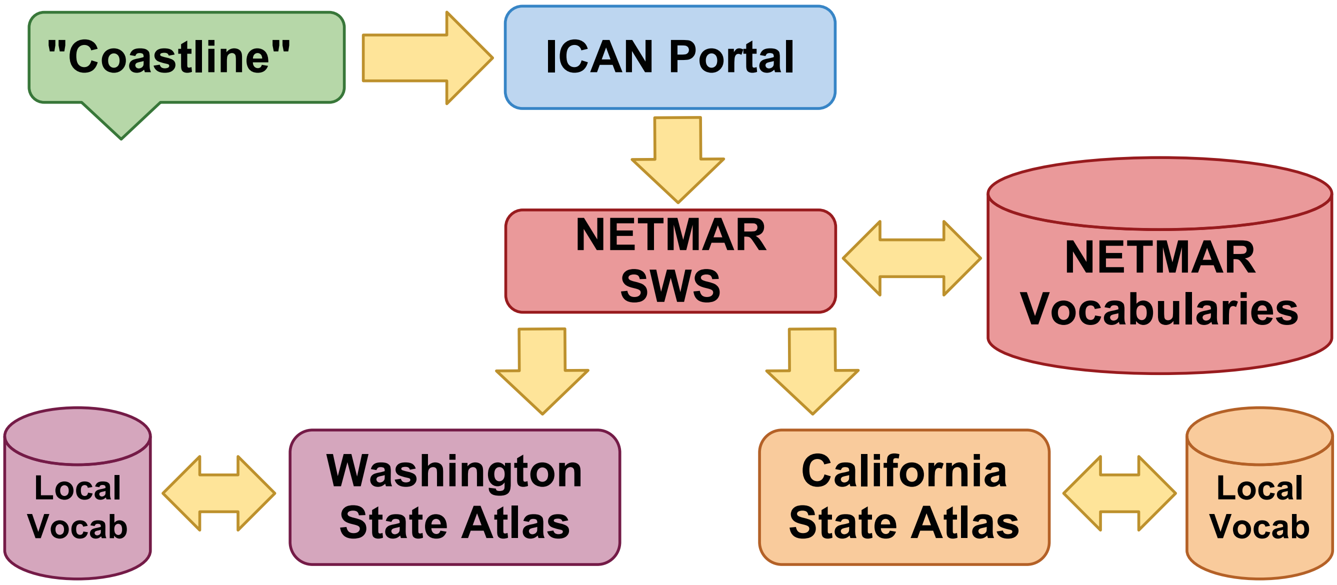
# ICAN "Semantic" Use Case




# ICAN "Semantic" Use Case

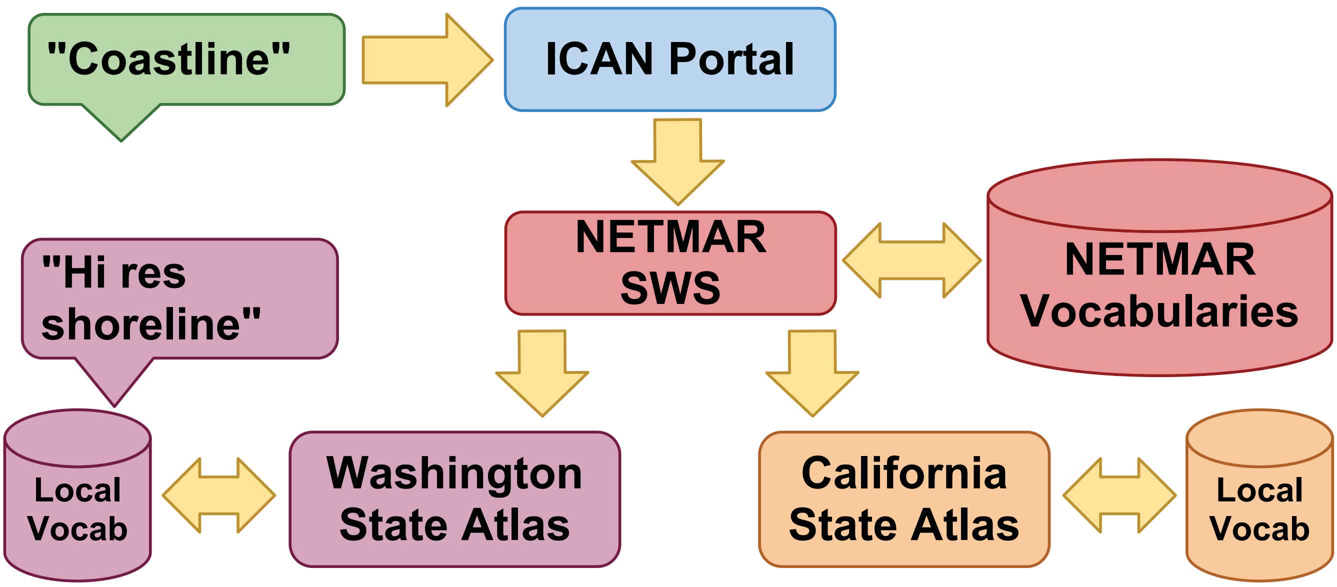


# ICAN "Semantic" Use Case




British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# ICAN "Semantic" Use Case




British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Implementation

- Standards based approach
  - W3C & OGC
    - W3C for vocabulary standards
    - OGC for catalogue services

# Implementation

- Standards based approach
  - W3C & OGC
    - W3C for vocabulary standards
    - OGC for catalogue services
- Implemented on NETMAR technology
  - Semantic mediator
    - Links catalogue services...
    - ... with distributed definitions

## International Coastal Atlas Network Coastal Erosion Discovery Thesaurus

Thesaurus of discovery terms that may be mapped to terms used to mark up atlas component data or metadata.

**Created by:** International Coastal Atlas Network

**Published by:** Natural Environment Research Council

Agents of Coastal Change(+)

**Definition:** Actions and processes that change the nature or location of the boundary between land and sea.

**Definition:** Non disponible

**Definition:** Ikke tilgjengelig

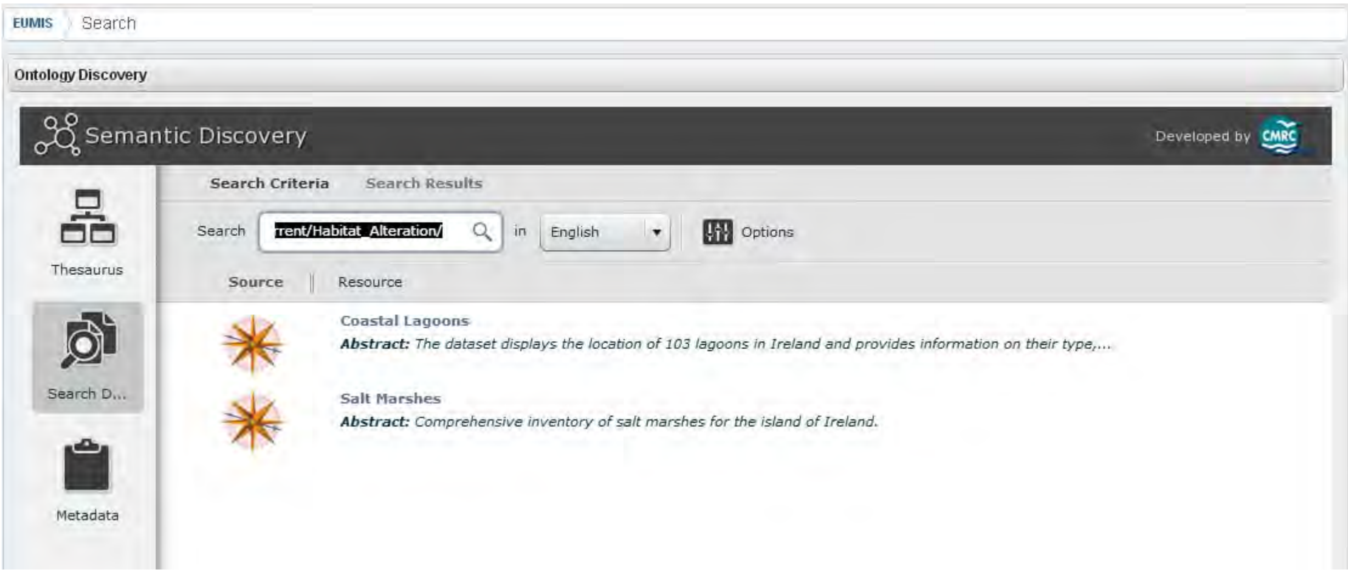
**Definition:** não disponível

**Definition:** no disponible

Human Activity(+)
Natural Processes(+)
Sediment Budget(+)

Effects of Coastal Change(+)

# Implementation



# Implementation

- Currently at demonstrator phase
  - Connected nodes are:
    - Oregon
    - MIDA (Ireland)

# Implementation

- Currently at demonstrator phase
  - Connected nodes are:
    - Oregon
    - MIDA (Ireland)
- How to join in?
  - Cookbooks published through the ICAN website

## *Appendix C. Slides from the SeaDataNet Plenary Meeting, 19-21 September 2012*

The slides presented at the meeting are enclosed.

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Common Vocabularies for SeaDataNet

Adam Leadbetter (alead@bodc.ac.uk) & Roy Lowry (rkl@bodc.ac.uk)
British Oceanographic Data Centre

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Why controlled vocabularies?

- A collection of concepts for populating a given metadata field

- Controlled vocabularies
  - Ensure consistent spellings
  - Ensure consistent syntax

- Well-managed controlled vocabularies
  - Prevent metadata misunderstandings
  - Maintain a static relationship between metadata fields and the real world

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Why controlled vocabularies?

- Concepts from different controlled vocabularies may be connected using simple mapping relationships:

Bacillariophycaea same-as diatoms
IPTS68 temperature related ITS90 temperature
Nutrients in rivers related nitrate in water bodies
Salinity is-narrower-than physical oceanography
Physical oceanography is-broader-than salinity

- The results may termed thesauri

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Where to find the controlled vocabularies?

- NERC Vocabulary Server
  http://vocab.nerc.ac.uk/
  Vocabulary names (Collections) e.g. P011 -> P01
  Code names remain the same: e.g. PSALCU01

  Changes to be incorporated to SDN all at once

  Web service based access:

  ReSTful & SOAP
  See http://vocab.nerc.ac.uk/vocab2.wsdl

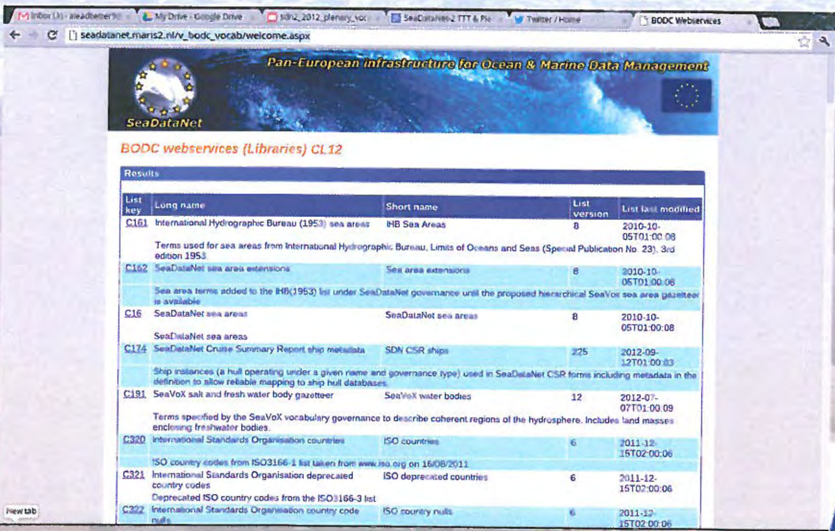
SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Where to find the controlled vocabularies?

## - Maris

- http://seadatanet.maris2.nl/v_bodc_vocab/welcome.aspx




SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Using the common vocabularies

| Method | ReSTful | SOAP |
|---|---|---|
| GetCollections | Y | Y |
| GetConceptCollection | Y | Y |
| GetConcept | Y | Y |
| GetSchemes | Y | Y |
| GetConceptSchemes | Y | Y |
| GetRelatedConcepts | | Y |
| GetTopConcepts | | Y |
| SearchVocab | | Y |
| VerifyConcept | Y | Y |

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Using the common vocabularies

- Full documentation online
  - At BODC website & SDN Extranet
    http://www.bodc.ac.uk/products/web_services/vocab/
- New content may be requested
  - May require acceptance by "content governance"
    - e.g. SeaVoX or ICES
- (O)250,000 API calls from (O)500 users in 12 months

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Using the common vocabularies

- Included within NEMO software (not yet released)
  - Will be included within Mikado

SeaDataNet

## Vocabulary users - Beyond Europe

- BCO-DMO, Woods Hole Oceanographic Institute
- Instituto de Investigaciones Marinas y Costeras, Colombia
- Climate and Forecast (CF) Community - Worldwide

- Links with Integrated Marine Observing System, Australia and NSF's R2R project



British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Vocabulary developments

- NERC Vocabulary Server v2.0 is ~ 1 year old

  Serves latest version of W3C standards

  Additional functionality
    - Grouping of terms in hierarchies
    - Multiple languages
    - Mapping to external vocabularies

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Vocabulary developments

**SeaDataNet Device Thesaurus**

A thesarus comprising categorisations of devices (sensors, instrument packages and sample collectors) developed by SeaDataNet mapped to a catalogue of specific devices maintained by SeaVoX.

**Created by:** SeaDataNet

**Published by:** Natural Environment Research Council

**SeaDataNet sample collector categories(+)**
**SeaDataNet sample processor categories(+)**

**Definition:** Categories used in the SeaDataNet project to classify devices that alter the nature of a sample prior or during property determination.

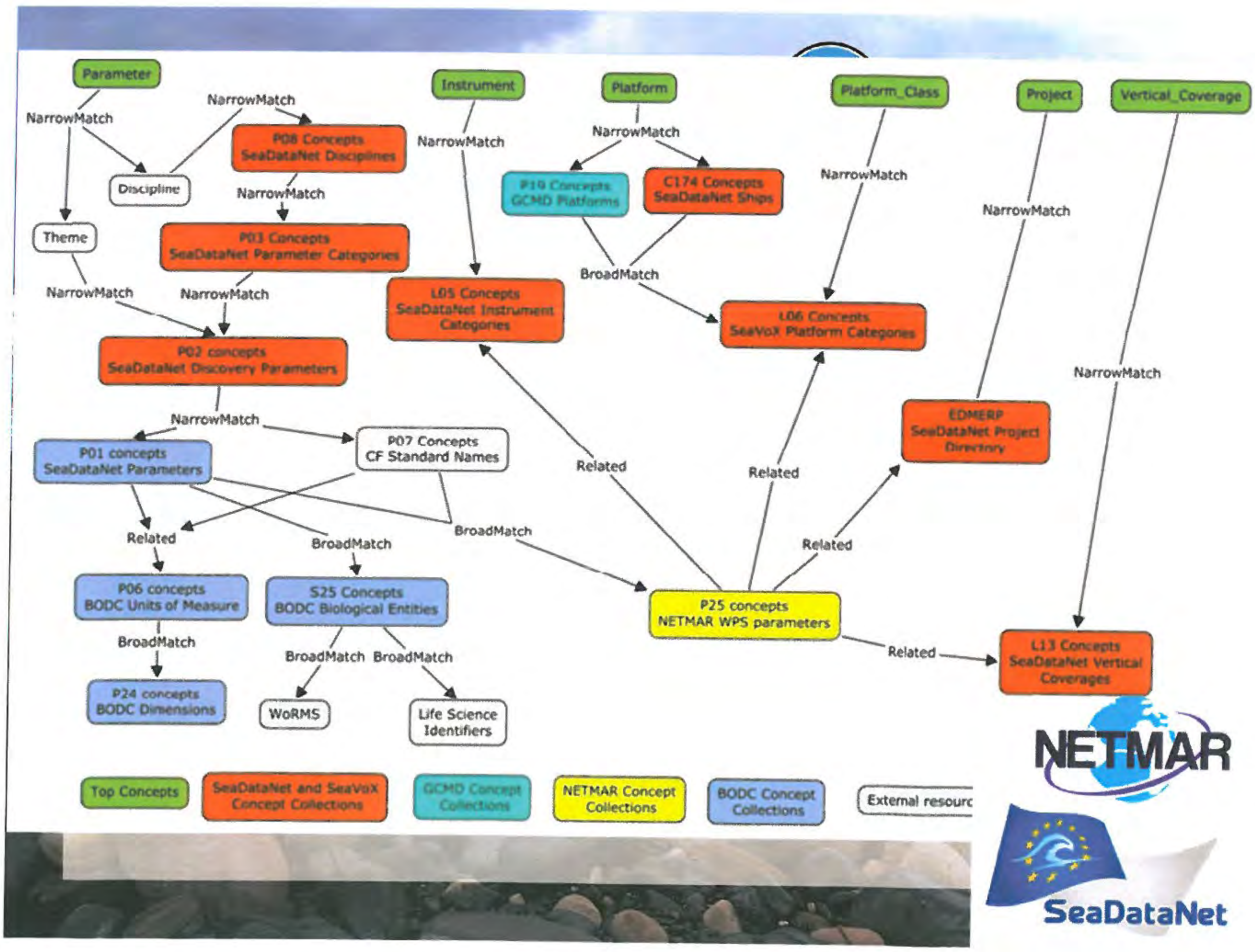
**Alternative concept labels:** sample processors

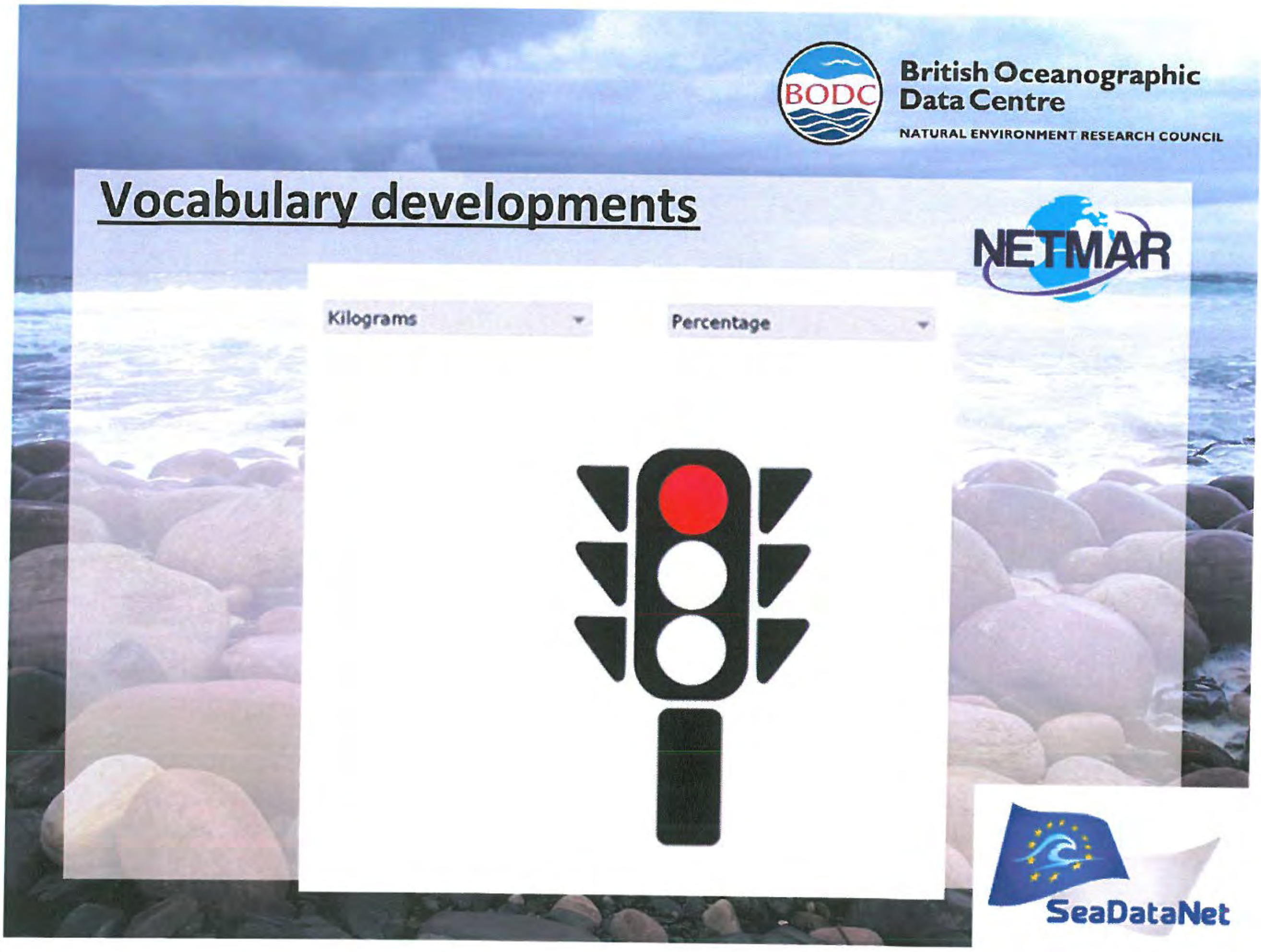
**shipboard incubators(+)**
**in-situ incubators(+)**
**centrifuges(+)**
**sieves and filters(+)**
**benthic incubation chambers(+)**

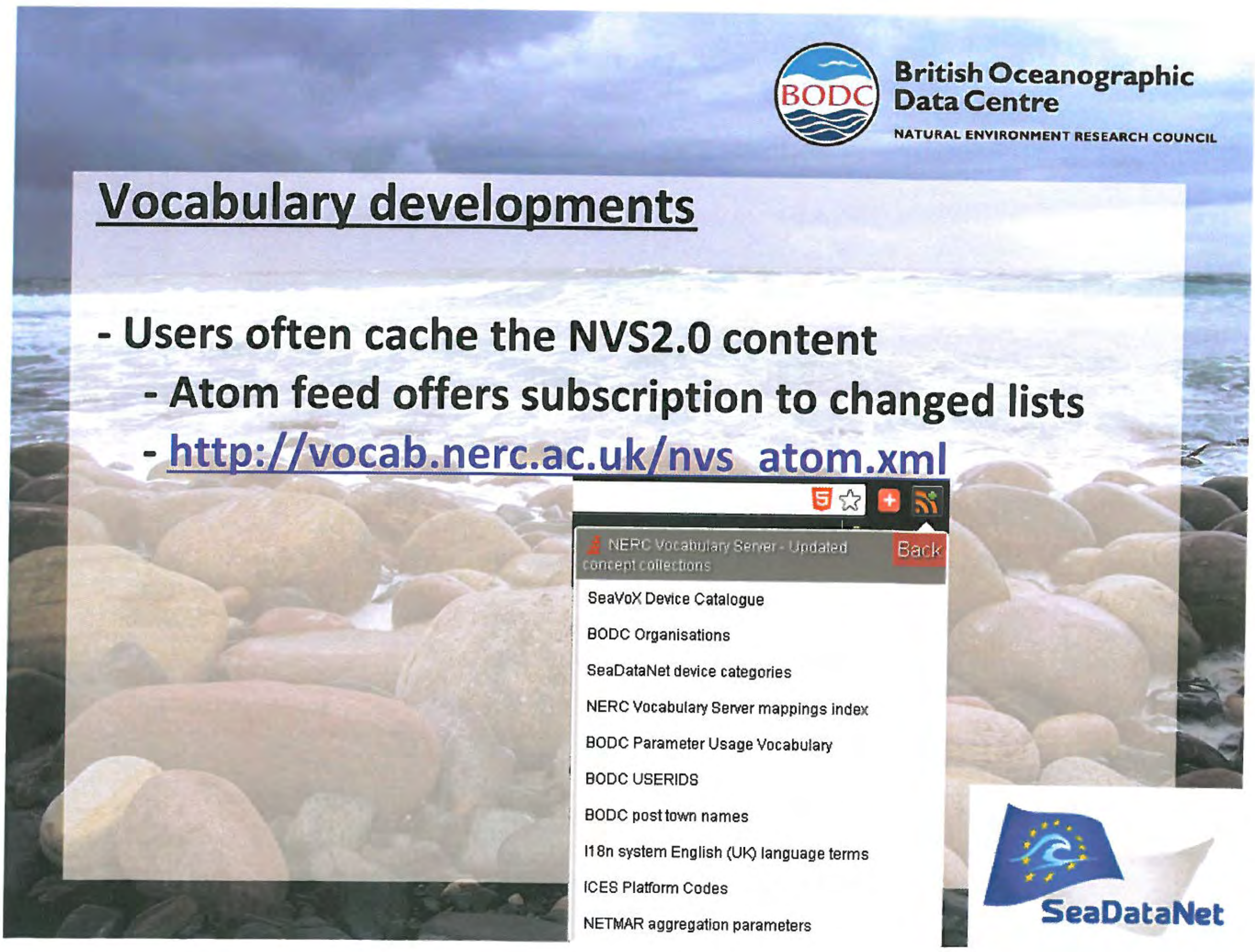
**SeaDataNet sample measurer categories(+)**
**SeaDataNet in-situ sensor and instrument package categories(+)**
**SeaDataNet remote sensor categories(+)**

NATURAL ENVIRONMENT RESEARCH COUNCIL

SeaDataNet




British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL
BODC
Vocabulary developments
NETMAR
Kilograms
Percentage
SeaDataNet


British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL
BODC
Vocabulary developments
- Users often cache the NVS2.0 content
- Atom feed offers subscription to changed lists
- http://vocab.nerc.ac.uk/nvs_atom.xml
NERC Vocabulary Server - Updated concept collections
Back
SeaVoX Device Catalogue
BODC Organisations
SeaDataNet device categories
NERC Vocabulary Server mappings index
BODC Parameter Usage Vocabulary
BODC USERIDS
BODC post town names
I18n system English (UK) language terms
ICES Platform Codes
NETMAR aggregation parameters
SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Vocabulary developments

- SPARQL endpoint
  - Deploying soon
  - Allowing querying of the knowledge store
  - ReSTful access to all API methods

SeaDataNet

British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL

## Vocabulary developments

- ISO 19139
  - ISO XML records use CodeLists – not W3C
  - ISO views of SDN controlled vocabs created
    - http://vocab.nerc.ac.uk/isoCodelists/sdnCodelists/cdicsrCodeList.xml
  - GMX CodeLists imported as controlled vocabs
  - ISO views created
    - http://vocab.nerc.ac.uk/isoCodelists/sdnCodelists/gmxCodeLists.xml
  - ISO catalogue of EDMED records
    - http://vocab.nerc.ac.uk/isoCodelists/sdnCodelists/edmedCodeList.xml

SeaDataNet


BODC
British Oceanographic Data Centre
NATURAL ENVIRONMENT RESEARCH COUNCIL
SeaDataNet

## *Appendix D. Slides from the PEGASO workshop, 23-26 October 2012*

The slides presented at the workshop are enclosed.

# International Coastal Atlas Network

Yassine Lassoued

*y.lassoued@ucc.ie*

*Coastal and Marine Research Centre*

# Outline

- Introduction
- ICAN
- Problem
- Terminology
- Solution
- Thesauri
- Query Rewriting
- Demonstration
- Current Work

## Introduction

- Demonstrate how geospatial data standards and semantic web technologies can be used to:
  - Help (heterogeneous) geographic information systems interoperate within an SDI
    - Facilitate resource sharing
  - Improve data discovery



## ICAN

- International Coastal Atlas Network
- Network of scientists, organisations and institutions with interest in the coastal and marine domain
- Objective
  - Build and strengthen atlas networks
  - Develop an internationally-enabled coastal Web atlas (ICWA)

# ICAN

- More than 30 members from more than 12 countries:
  - Coastal and Marine Resources Centre
  - Oregon State University
  - European Environmental Agency
  - Marine Metadata Interoperability
  - International Oceanographic Data and Information Exchange
  - Marine Institute
  - Etc.



# Problem

- Interoperability of distributed autonomous and heterogeneous coastal Web atlases (CWA)

…

# Problem

- Coastal Web Atlas (CWA):
  - Web application for the delivery of coastal resources such as:
    - Geographic datasets
    - Maps
    - Metadata
    - Thematic (educational) information



# Problem

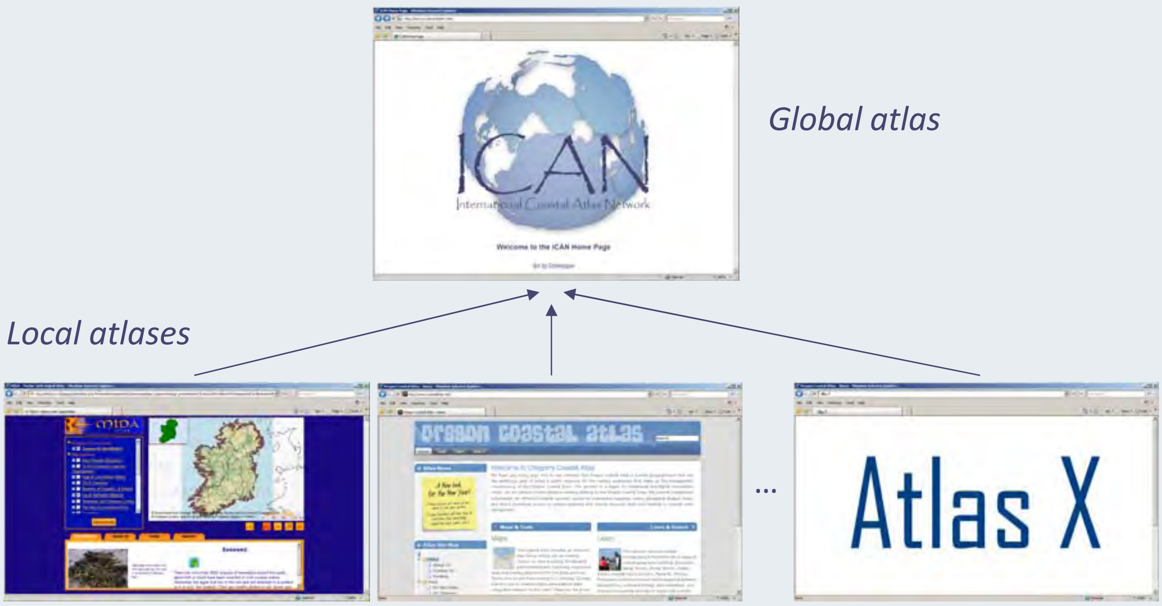
- Heterogeneity:
  - Syntactic (data formats, query languages)
  - Structural (data schemas)
  - Semantic (meaning of data values)

Example: Metadata

- Different metadata standards (ISO-19115 vs. FGDC)
- Different vocabularies: 'Seabed' vs. 'Seafloor'
  'Coastline' vs. 'Shoreline'
  French, Spanish, English…

# Approach

- Connect individual coastal atlases to an integrated global atlas






# Terminology

- OGC Web Service:
  - OGC specification
  - Interface allowing requests for geographic "*resources*" across the Web using platform-independent calls
  - Common OGC services:
    - Catalogue Service for the Web (CSW)
    - Web Feature Service (WFS)
    - Web Coverage Service (WCS)
    - Web Map Service (WMS)

# Terminology

- OGC Web Service:
  - Catalogue Service for the Web (CSW)
    - Allows requests for metadata across the Web
    - E.g. GeoNetwork implements CSW

| Request | Response |
|---|---|
| `Get Capabilities` | *Metadata about the types / operations the CSW supports* |
| `Get Records` | *Metadata records available, with possibility of filtering (bounding box, spatial, temporal, keywords search, etc.)* |
| `Get Record By ID` | *Record with the specified ID* |



# Terminology

- OGC Web Service:
  - Web Map Service (WMS) → Maps
    - Allows requests for maps across the Web
    - E.g. UMN MapServer and GeoServer implement WMS

| Request | Response |
|---|---|
| `Get Capabilities` | *Metadata about the types / operations the WMS supports* |
| `Get Map` | *Map of the requested data* |
| `Get Feature Info` | *Thematic information about a particular point within a map* |

# Terminology

- Ontologies:
  - A Knowledge Organisation System (KOS)
  - Define concepts (classes and objects)
  - Define relationships between concepts
  - Define inference rules
  - Example:
    - John *is a* Person
    - Pete *is a* Person
    - Pete *is father of* John
    - **If** (X *is father* of Y & Y *is father of* Z)
      **then** X is grand-father of Z



Terminology

- Thesauri:
  - Define concepts (terms)
  - Define relationships between concepts
    - Hierarchy
    - Synonymy
    - Relatedness
  - Example:
    - Bathymetry *is narrower than* Elevation
    - Thermometer *is related to* Temperature
    - Coastline *is synonym to* Shoreline

# Terminology

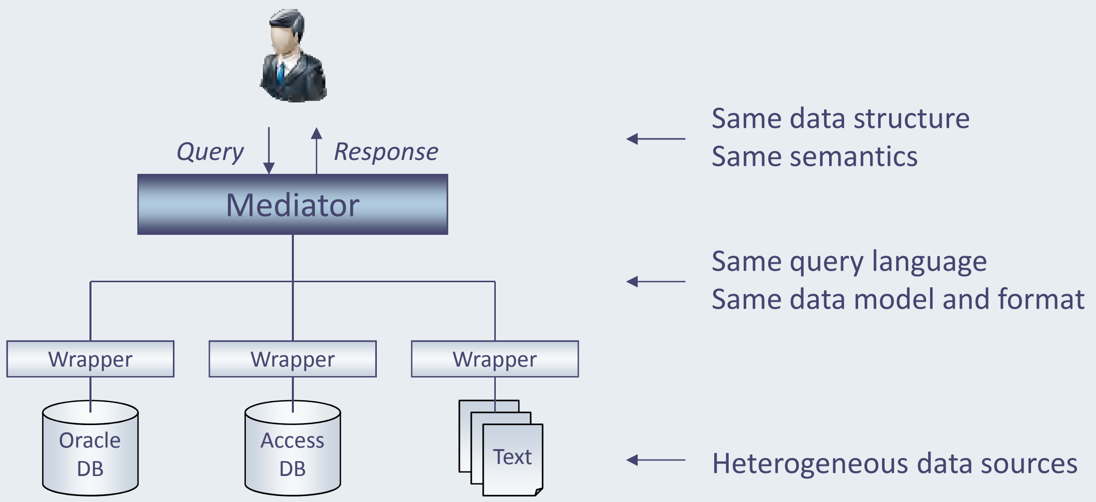
- Thesauri:
  - SKOS: Simple Knowledge Organization System
    - W3C recommendation
    - Data model for defining thesauri

```
<skos:Copncept rdf:ID="http://geodi.ucc.ie/ont/20110429/geoscience.owl#TideGauge">
        <skos:prefLabel xml:lang="en">Tide Gage</skos:prefLabel>
        <skos:prefLabel xml:lang="en">Tide Gage</skos:prefLabel>
        <skos:hiddenLabel xml:lang="en">Tide Guage</skos:hiddenLabel>
        <skos:prefLabel xml:lang="fr">Marégraphe</skos:prefLabel>
        <skos:prefLabel xml:lang="es">Mareógrafos</skos:prefLabel>

        <skos:definition xml:lang="en">
                A gauge used to measure extremes or the present level of tidal movement
        </skos:definition>

        <skos:related rdf:resource="http://geodi.ucc.ie/ont/20110429/geoscience.owl#tides"/>
</skos:Concept>
```



# Terminology

- Mediation:
  - A virtual data integration approach
  - Allows transparent access and integration of autonomous distributed heterogeneous data sources



# Solution

- To achieve interoperability:
  1. Standardisation:
     - Standardise access interfaces and resource formats
       - Implement OGC Web Services
         - » Catalogue Service for the Web (CSW)
         - » Web Feature Service (WFS)
         - » Web Coverage Service (WCS)
         - » Web Map Service (WMS)
       - Use ISO metadata standards
         - » ISO-19115 & ISO-19139

     → Standardise web querying and delivery formats

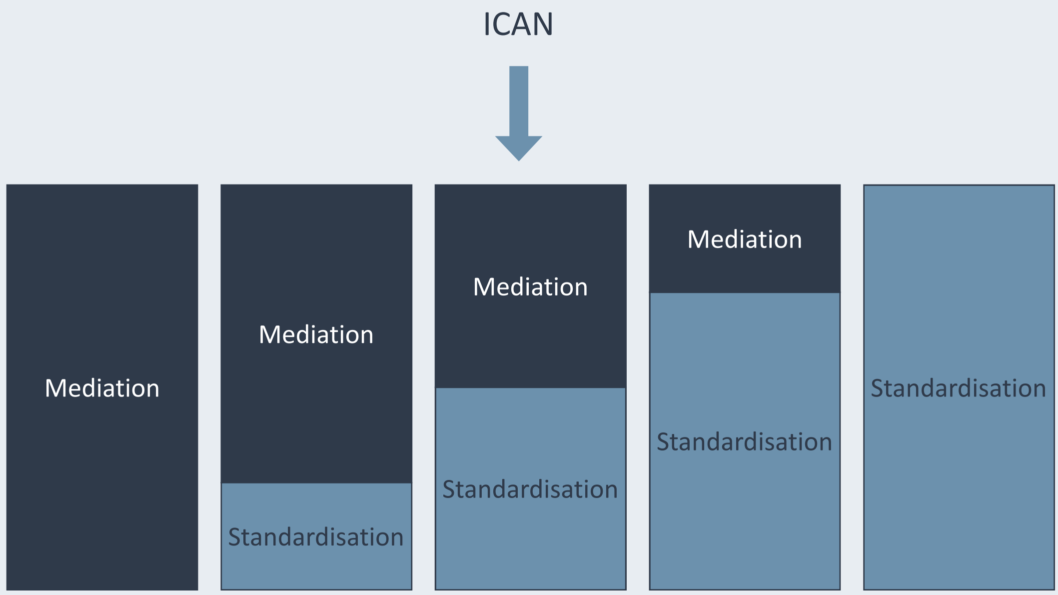


# Solution

- To achieve interoperability:
  2. Mediation:
     - Allow local atlases to use their own vocabularies (ontologies)
     - Use a common ontology for the global atlas: global ontology
     - Provide mappings (translations) between local ontologies and the global ontology

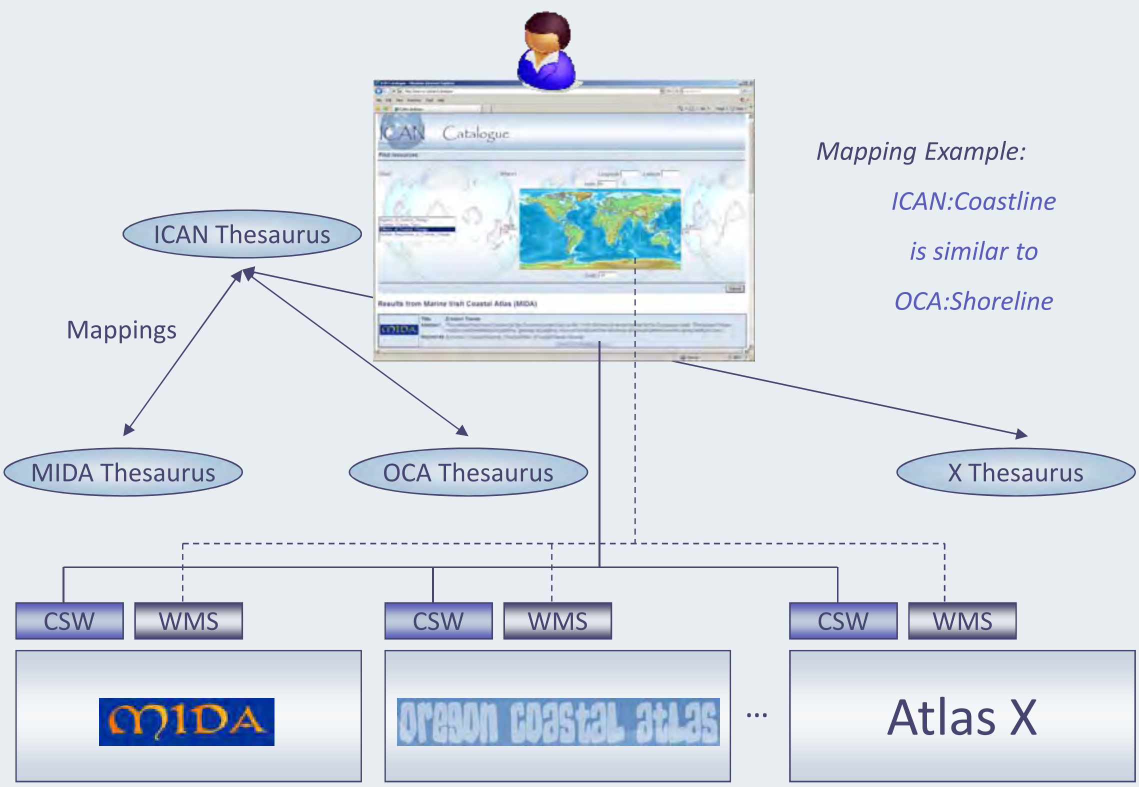
# Solution

## Harmonisation vs. Mediation

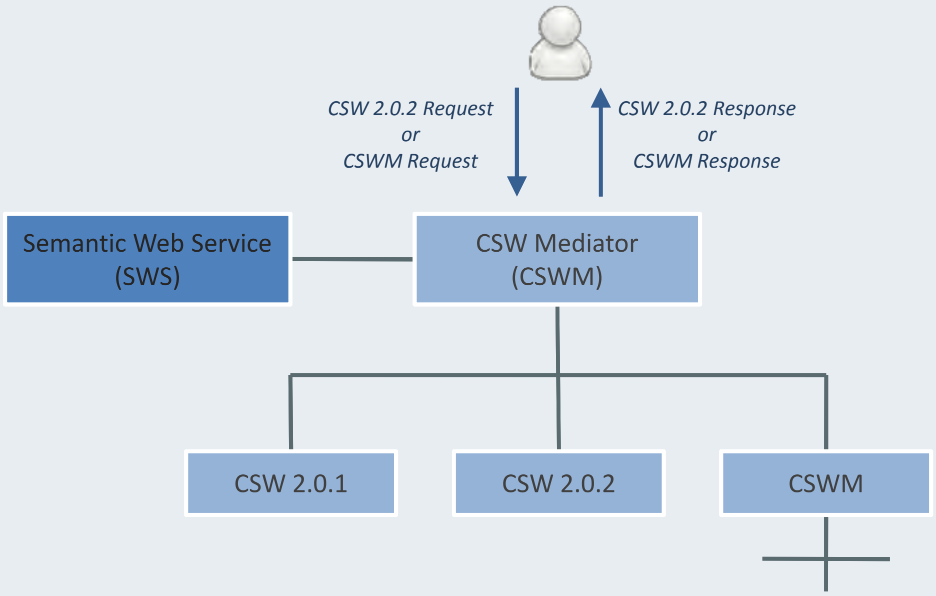





# Solution



# Solution






# Thesauri

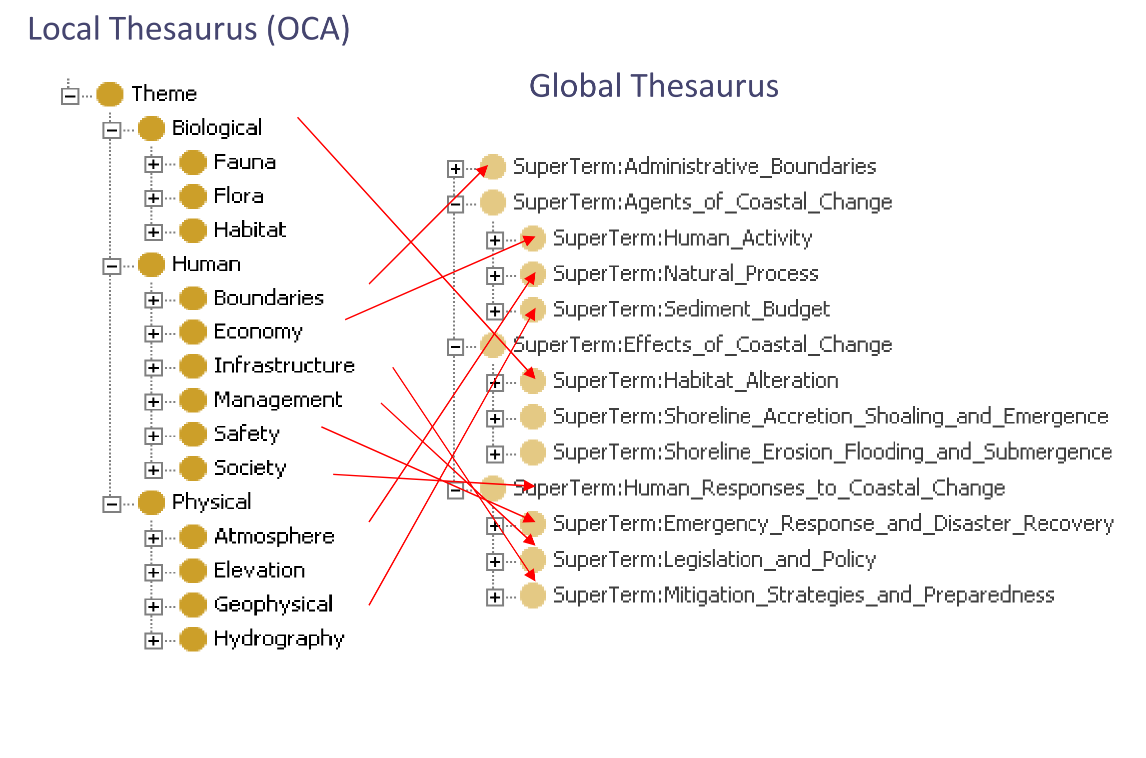
## Local Thesaurus (OCA)

- Theme
  - Biological
    - Fauna
    - Flora
    - Habitat
  - Human
    - Boundaries
    - Economy
    - Infrastructure
    - Management
    - Safety
    - Society
  - Physical
    - Atmosphere
    - Elevation
    - Geophysical
    - Hydrography

## Global Thesaurus

- SuperTerm:Administrative_Boundaries
- SuperTerm:Agents_of_Coastal_Change
  - SuperTerm:Human_Activity
  - SuperTerm:Natural_Process
  - SuperTerm:Sediment_Budget
- SuperTerm:Effects_of_Coastal_Change
  - SuperTerm:Habitat_Alteration
  - SuperTerm:Shoreline_Accretion_Shoaling_and_Emergence
  - SuperTerm:Shoreline_Erosion_Flooding_and_Submergence
- SuperTerm:Human_Responses_to_Coastal_Change
  - SuperTerm:Emergency_Response_and_Disaster_Recovery
  - SuperTerm:Legislation_and_Policy
  - SuperTerm:Mitigation_Strategies_and_Preparedness

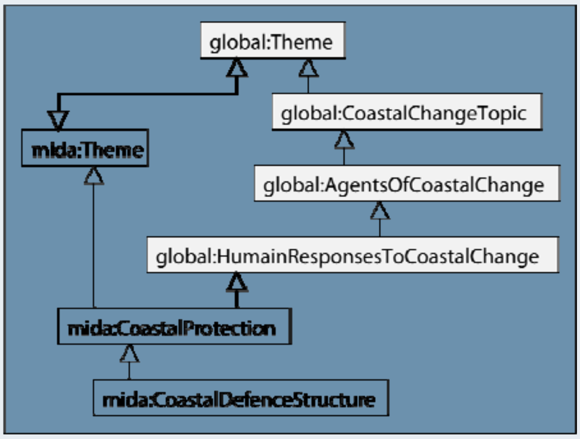
# Thesauri



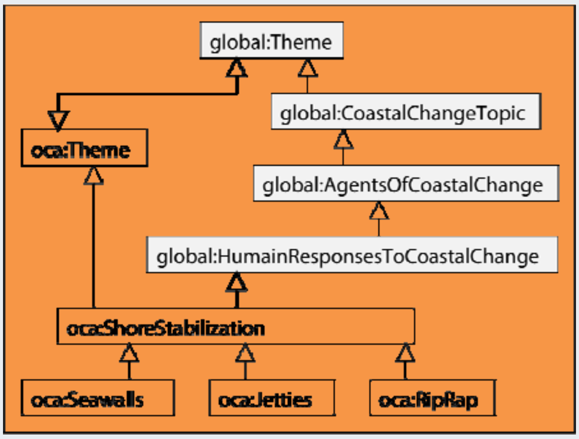



# Thesauri

MIDA Mappings




OCA Mappings



# Query Rewriting

- Rewrite user's request into requests supported by local CSWs



# Query Rewriting

```
http://ican.ucc.ie/srv/en/csw?request=GetRecords&service=CSW&version=2.0.1
&resultType=results&namespace=csw:http://www.opengis.net/cat/csw&maxRecords=1000
&elementSetName=summary
&constraint=
<?xml version="1.0" encoding="UTF-8"?>
<Filter xmlns=http://www.opengis.net/ogc xmlns:gml=http://www.opengis.net/gml
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
  <And>
    <PropertyIsLike wildCard="%“ singleChar="_" escape="\">
      <PropertyName>keyword</PropertyName>
      <Literal>HumanResponsesToCoastalChange%</Literal>
    </PropertyIsLike>
    <BBOX>
      <PropertyName>/csw:Record/ows:BoundingBox</PropertyName>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
        <gml:lowerCorner>-180 -90</gml:lowerCorner>
        <gml:upperCorner>180 90</gml:upperCorner>
      </gml:Envelope>
    </BBOX>
  </And>
</Filter>
&constraintLanguage=FILTER
&constraint_language_version=1.1.0
```

# Query Rewriting

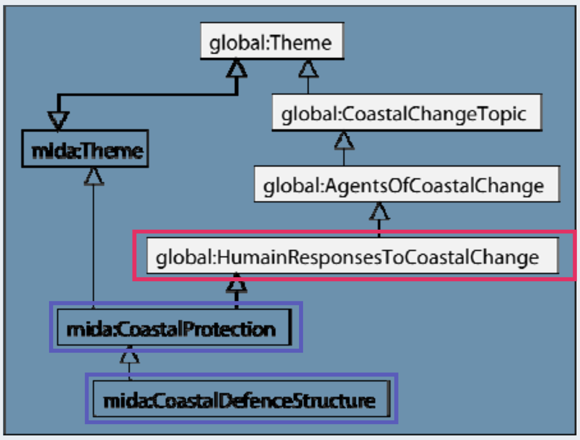
```
http://ican.ucc.ie/srv/en/csw?request=GetRecords&service=CSW&version=2.0.1
&resultType=results&namespace=csw:http://www.opengis.net/cat/csw&maxRecords=1000
&elementSetName=summary
&constraint=
<?xml version="1.0" encoding="UTF-8"?>
<Filter xmlns=http://www.opengis.net/ogc xmlns:gml=http://www.opengis.net/gml
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <And>
        <PropertyIsLike wildCard="%“ singleChar="_" escape="\">
            <PropertyName>keyword</PropertyName>
            <Literal>HumanResponsesToCoastalChange%</Literal>
        </PropertyIsLike>
        <BBOX>
            <PropertyName>/csw:Record/ows:BoundingBox</PropertyName>
            <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:lowerCorner>-180 -90</gml:lowerCorner>
                <gml:upperCorner>180 90</gml:upperCorner>
            </gml:Envelope>
        </BBOX>
    </And>
</Filter>
&constraintLanguage=FILTER
&constraint_language_version=1.1.0
```



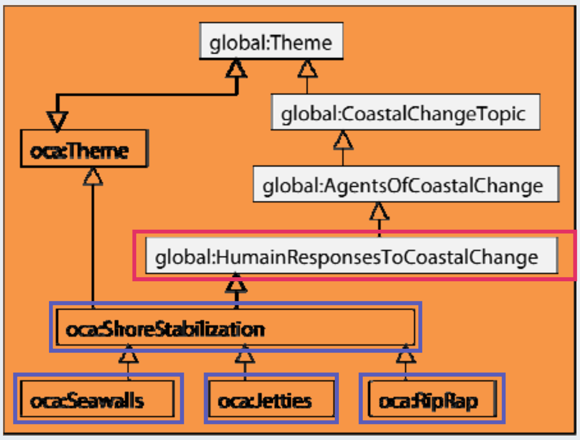
# Query Rewriting

Mediator uses inference engine to translate global terms into local terms

MIDA Mappings




OCA Mappings



# Query Rewriting

Global

<PropertyIsLike wildCard="%“ singleChar="_" escape="\">
<PropertyName>**keyword**</PropertyName>
<Literal>HumanResponsesToCoastalChange%</Literal>
</PropertyIsLike>

↓

MIDA

<Or>
<PropertyIsLike wildCard="%“ singleChar="_" escape="\">
<PropertyName>**keyword**</PropertyName>
<Literal>CoastalProtection%</Literal>
</PropertyIsLike>
<PropertyIsLike wildCard="%“ singleChar="_" escape="\">
<PropertyName>**keyword**</PropertyName>
<Literal>CoastalDefenceStructure%</Literal>
</PropertyIsLike>
</Or>



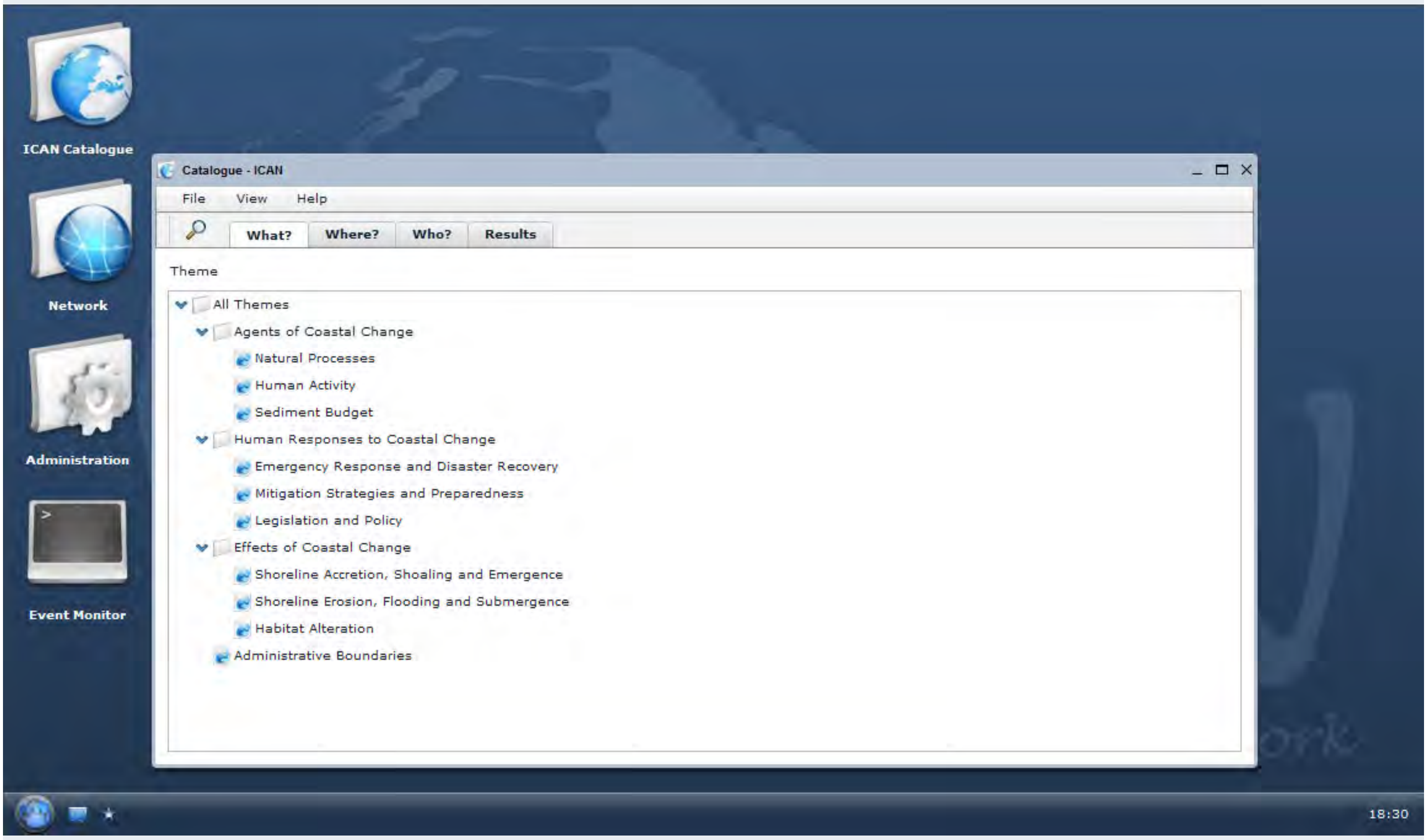
# Demonstration

# Demonstration



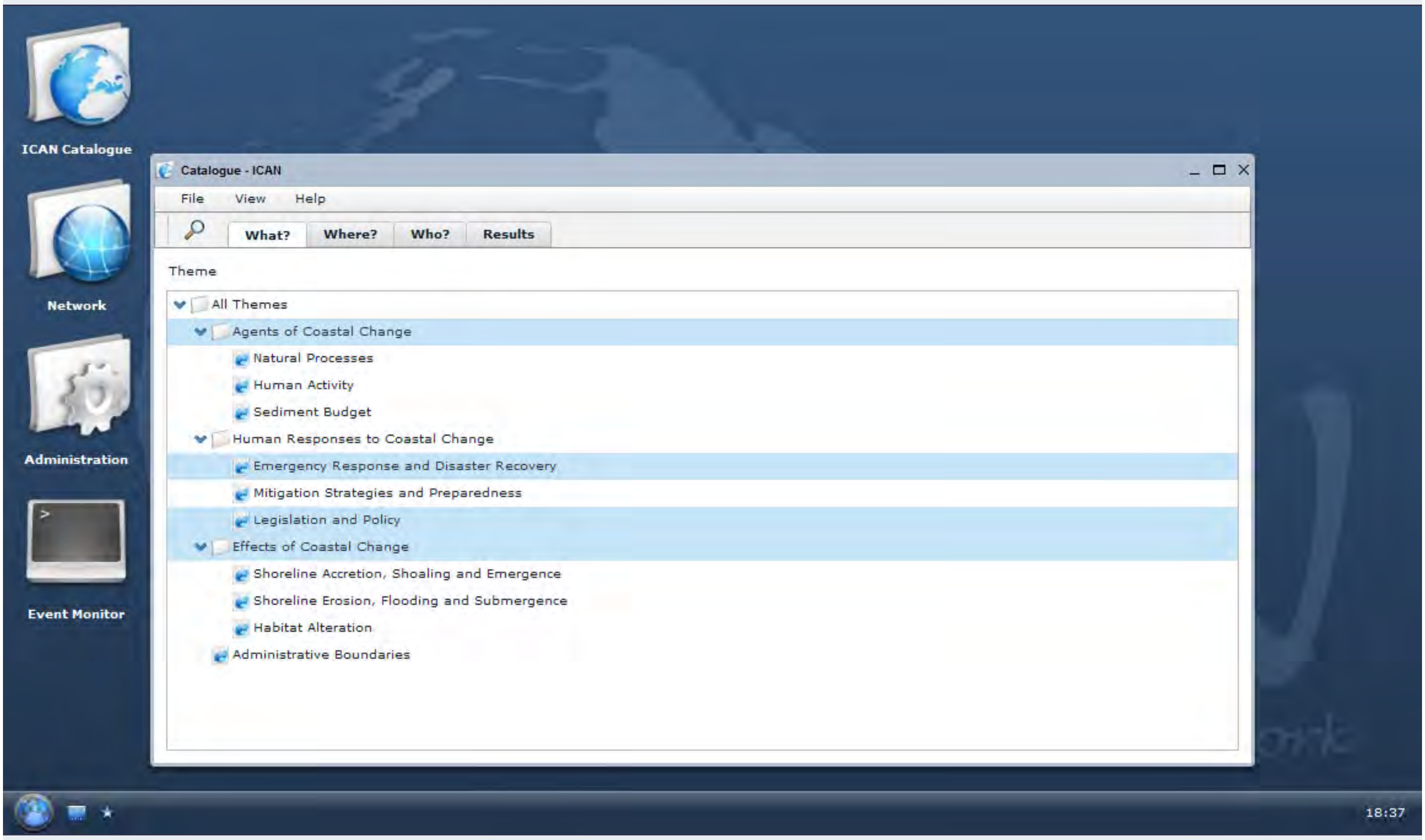

# Demonstration

# Demonstration

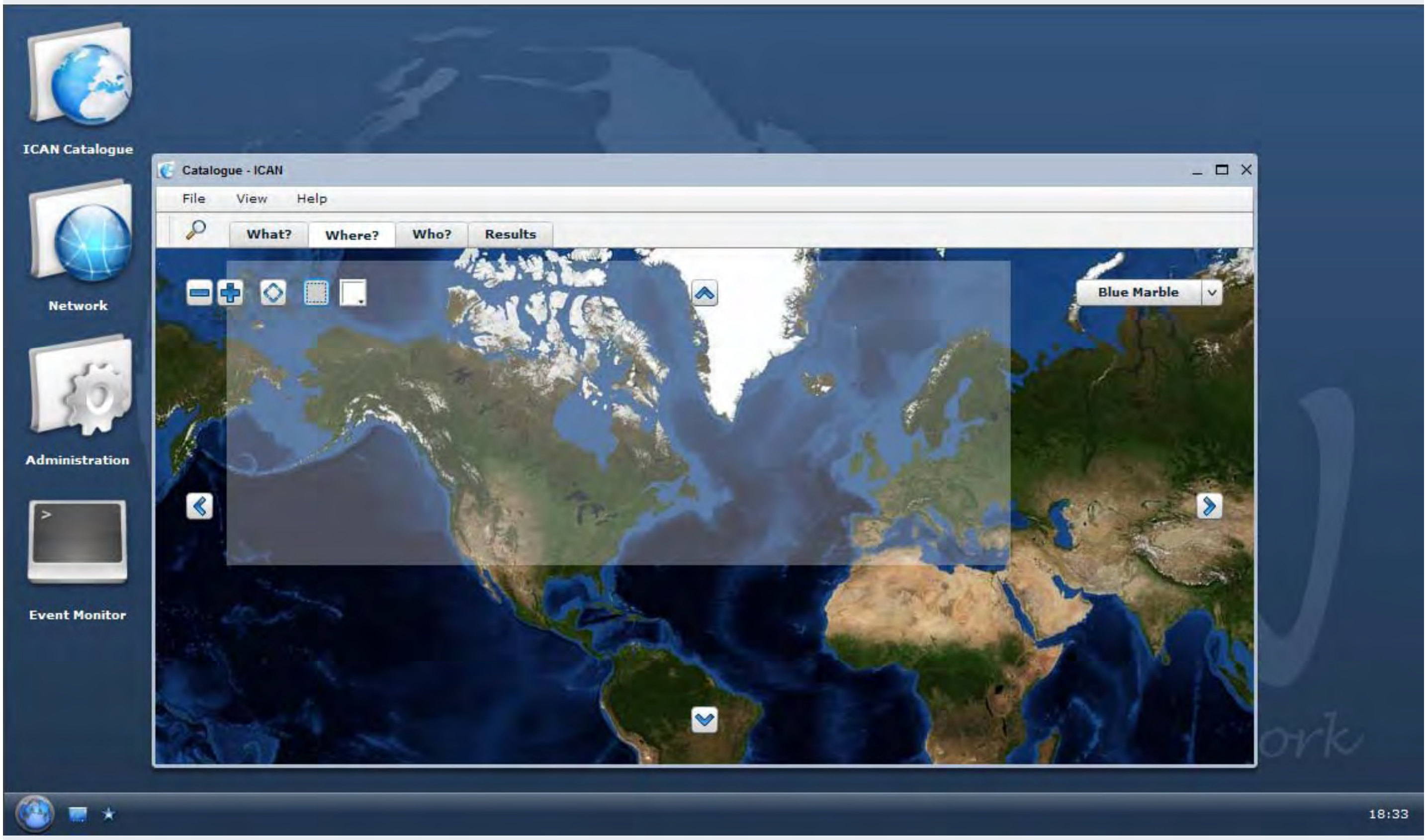





# Demonstration

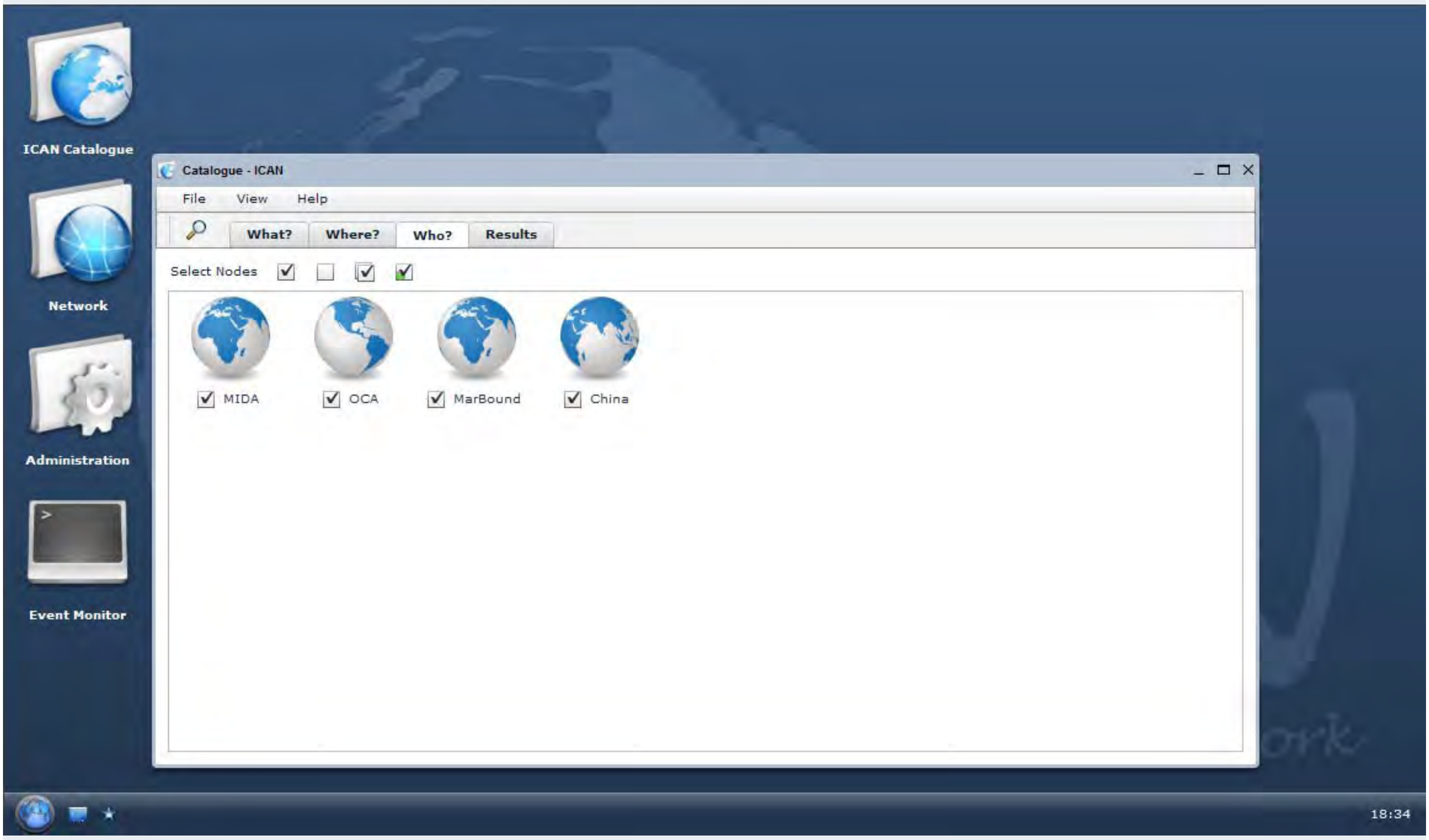


# Demonstration

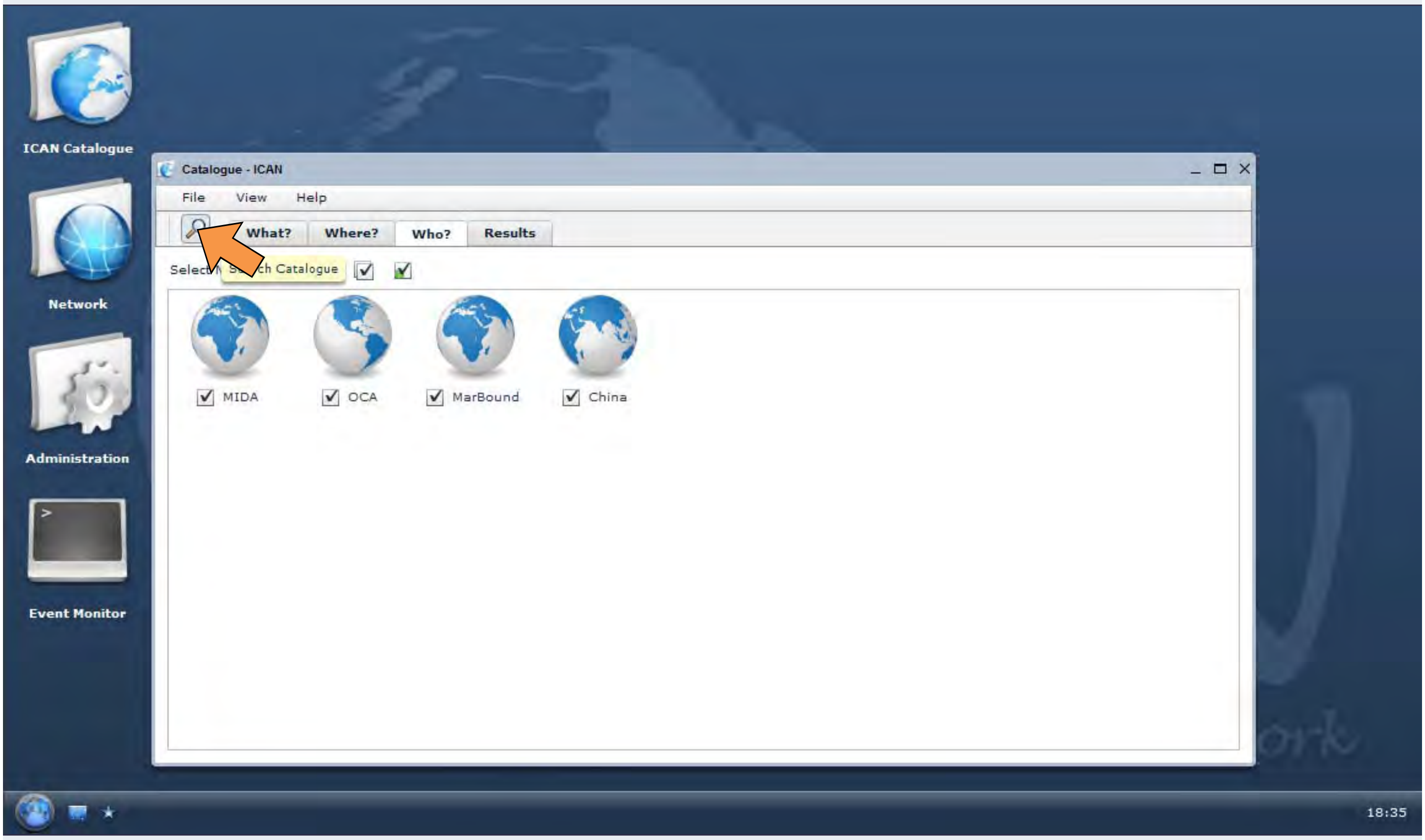





# Demonstration

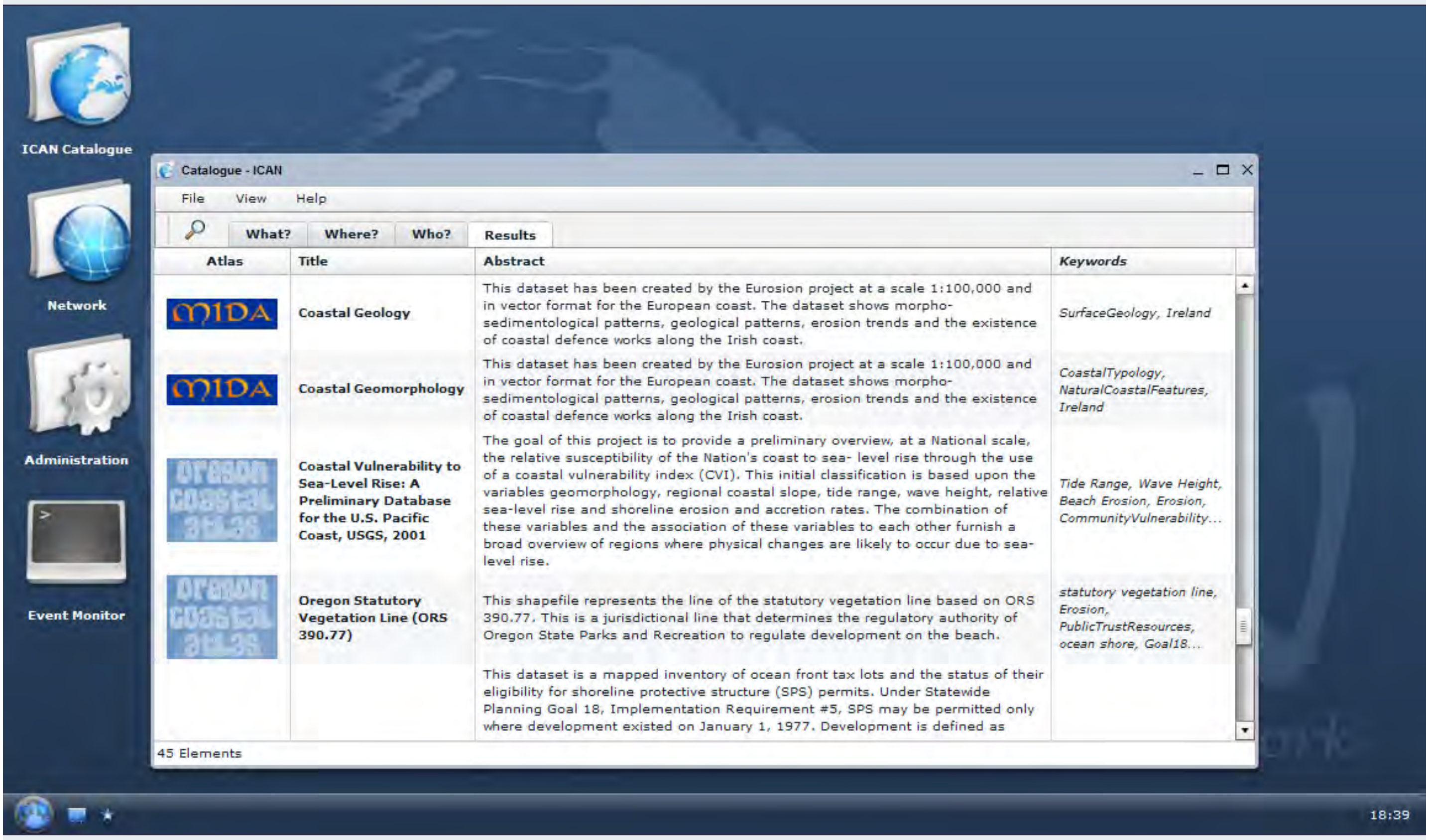


# Demonstration

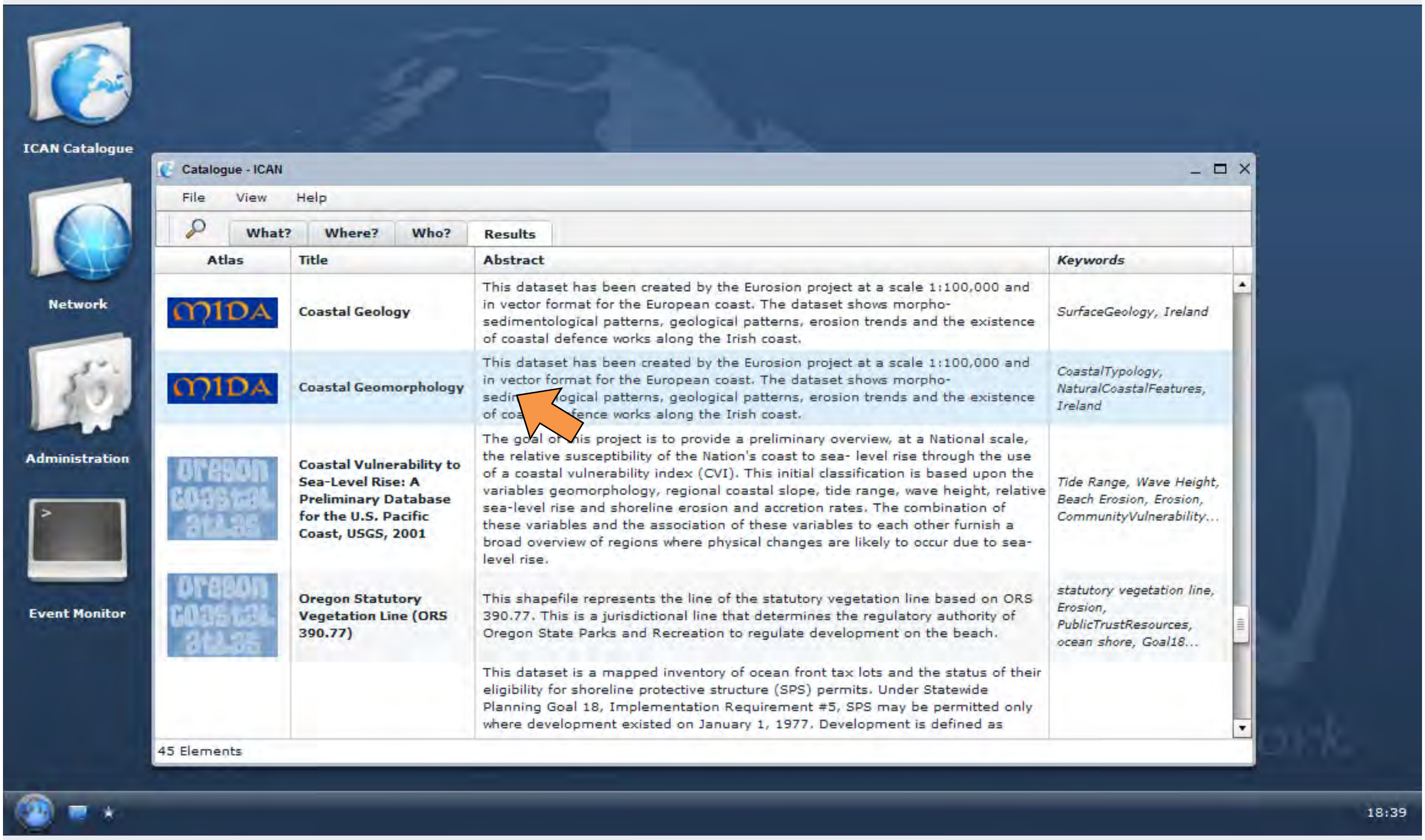





# Demonstration

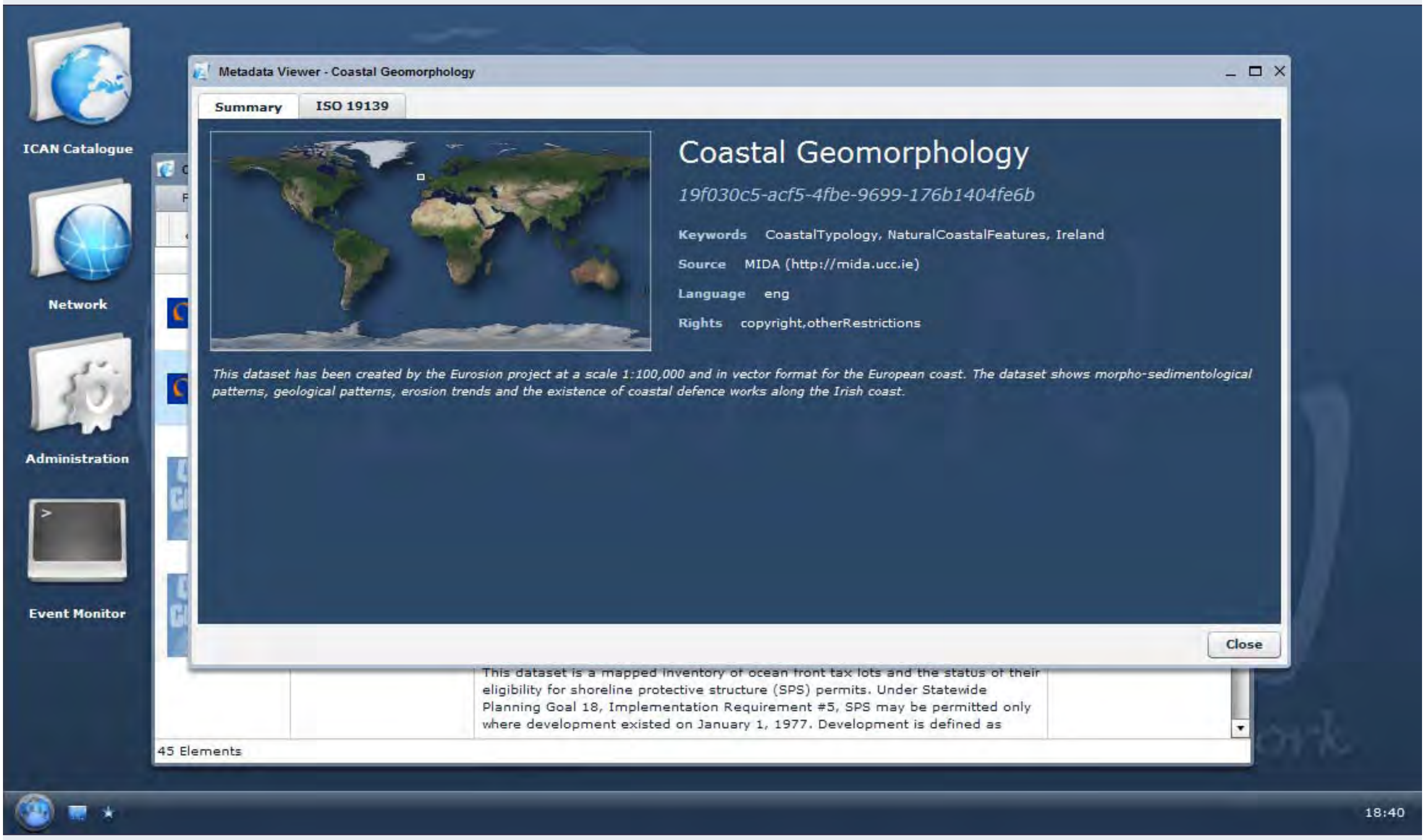


# Demonstration

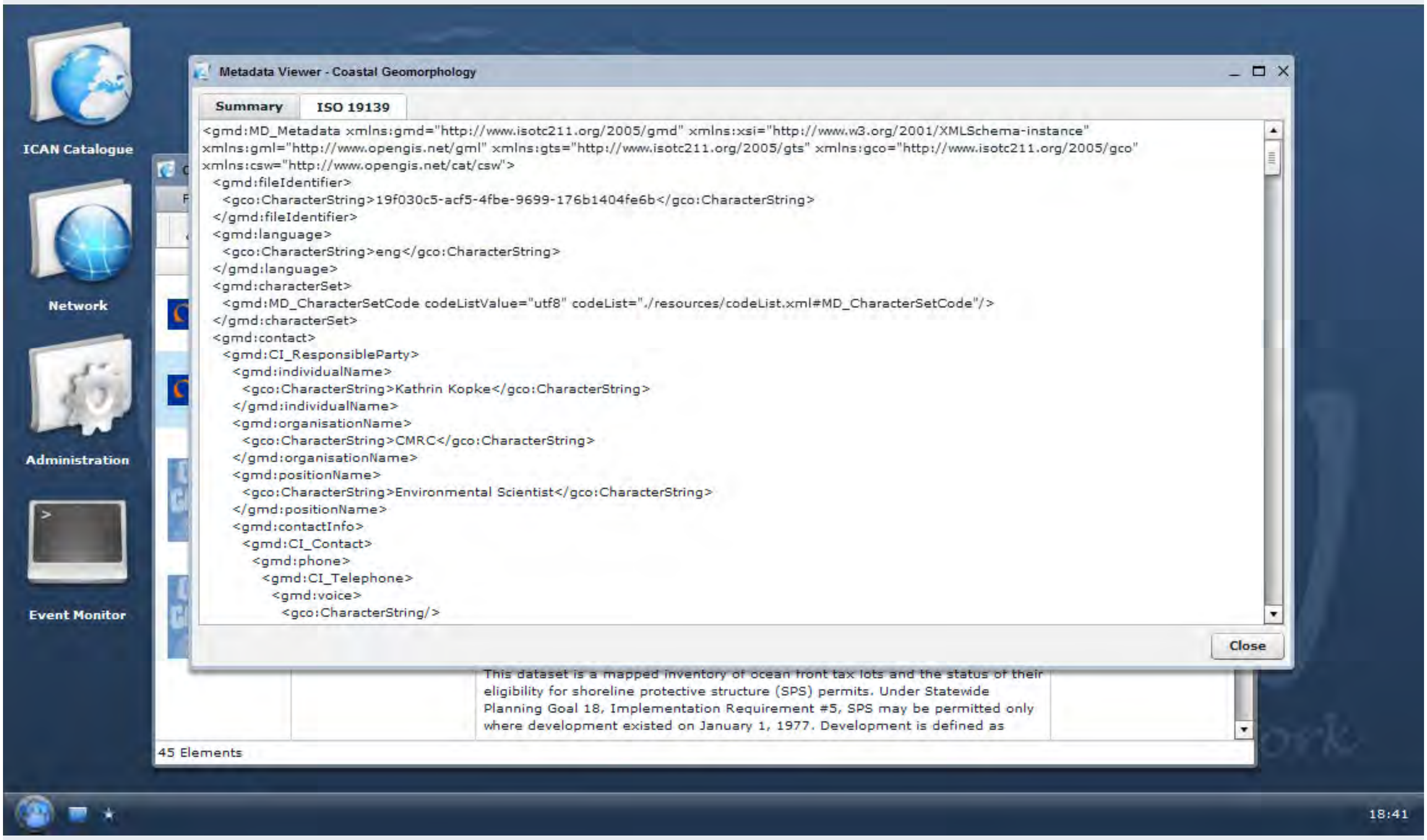





# Demonstration



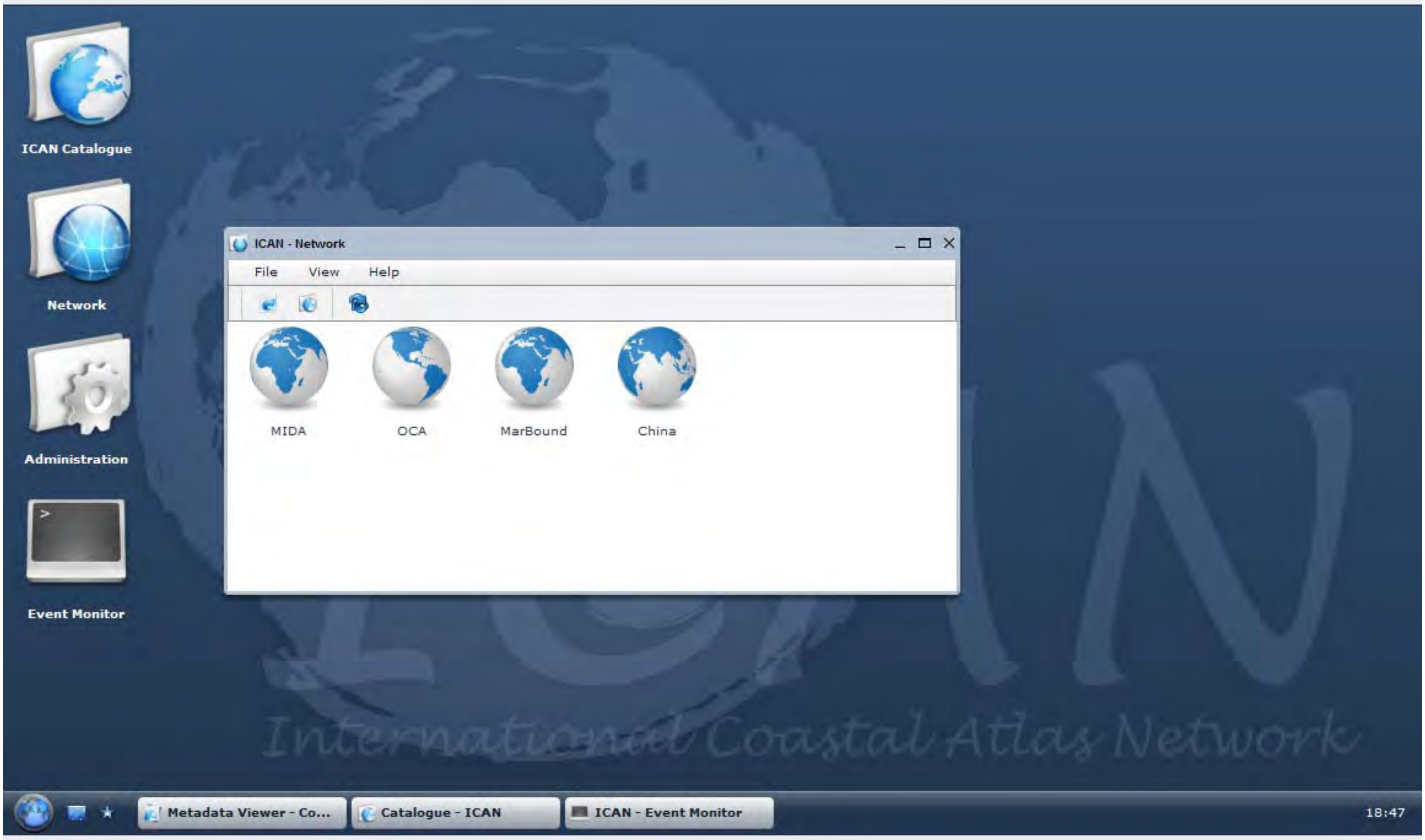
# Demonstration





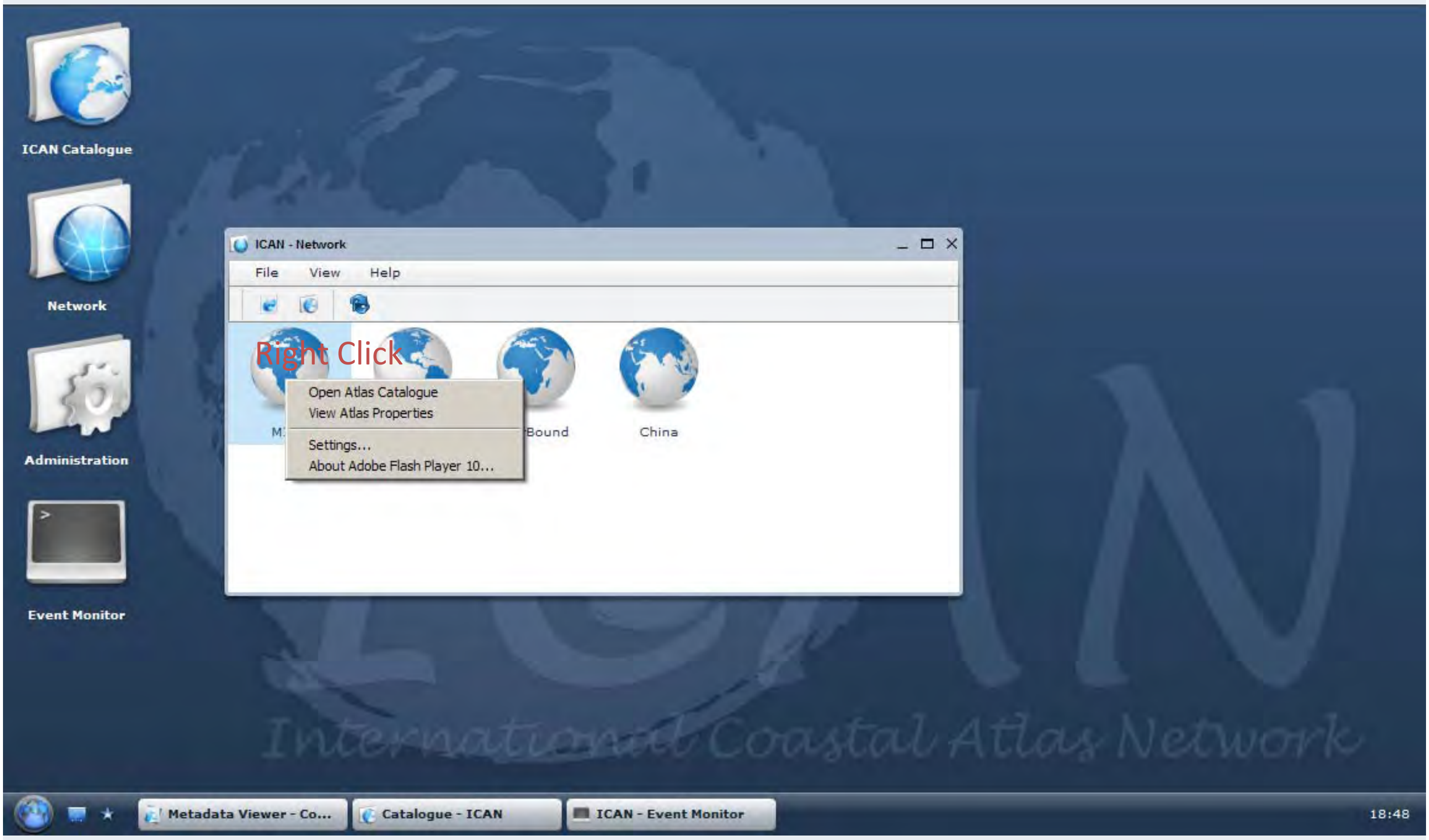

# Demonstration

# Demonstration





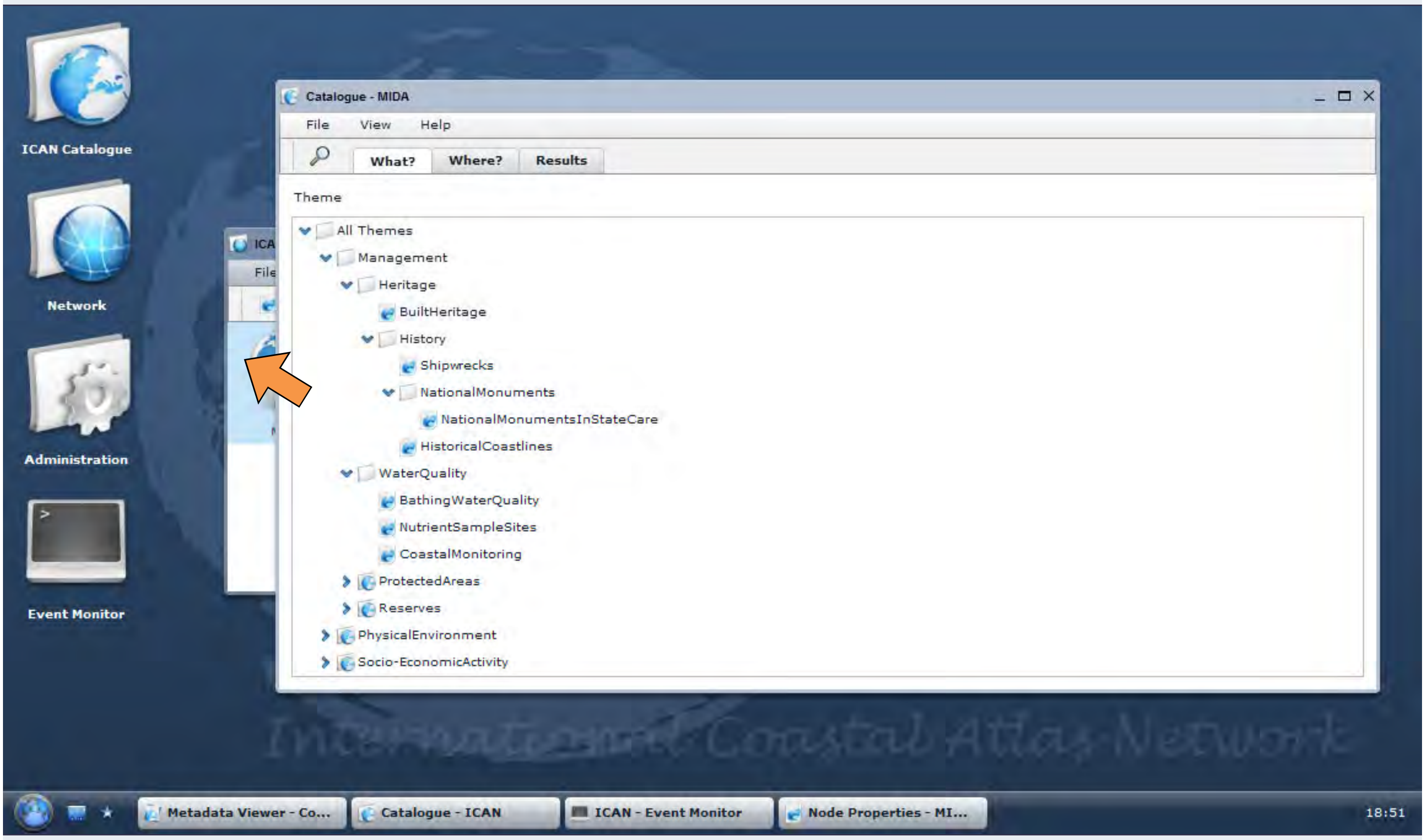

# Demonstration



# Demonstration




# Demonstration



# Current Work

- Version 3 of the ICWA is being developed as part of the EU FP7 NETMAR project
  - To be launched soon
- New graphical interface
- Improved performance
- Improved management of thesauri
- Support for CSW 2.0.2, WMS 1.3.1, and SKOS
  - Metadata to reference WMS links in delivery information
- Multilingual thesauri
- Smart search
- Support for inter-thesaurus mappings
- Mapping interface
- Towards an operation system…

## *Appendix E. Slides from the Arctic ROOS Annual Meeting, 6-7 November 2012*

The slides presented at the meeting are enclosed.

**EUMIS** - an open portal framework for interoperable marine environmental services

T. Hamre[1], S. Sandven[1], A. Leadbetter[2], V. Gouriou[3], D. Dunne[4], M. Grant[5], M. Treguer[6], and Ø. Torget[7]

[1]NERSC, [2]BODC, [3]CEDRE, [4]CMRC, [5]PML, [6]Ifremer, [7]METNO

*Arctic ROOS Annual Meeting – Sopot, Poland – 6-7 November 2012*

# Outline

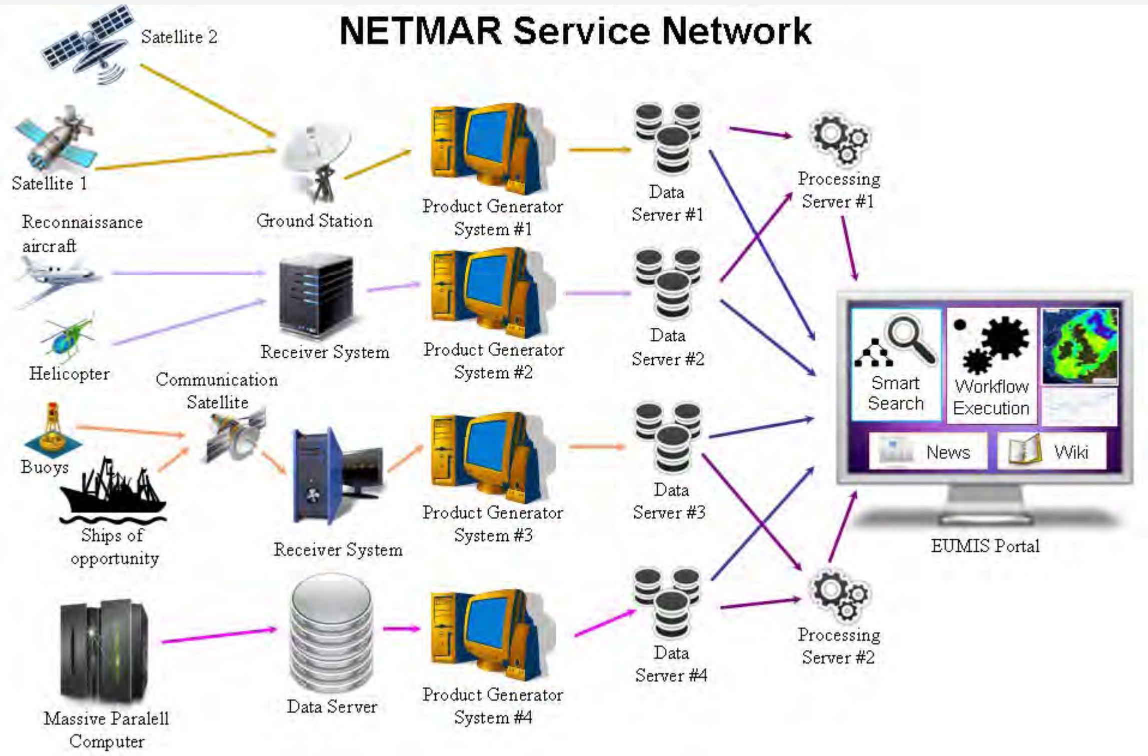
- Objectives and concepts
- Pilots
- Ontologies and semantic framework
- EUMIS portal and components
  - GIS Viewer
  - Discovery Client
  - Service Chaining Editor
- Conclusion

# Objectives and concepts

- NETMAR aims to develop a ***pilot European Marine Information System (EUMIS)*** for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. It will be a user-configurable system offering ***flexible service discovery, access and chaining facilities*** using OGC, OPeNDAP and W3C standards. It will use a ***semantic framework coupled with ontologies*** for identifying and accessing distributed data, such as near-real time, forecast and historical data. EUMIS will also enable further processing of such data to generate ***composite products and statistics*** suitable for decision-making in different marine application domains.

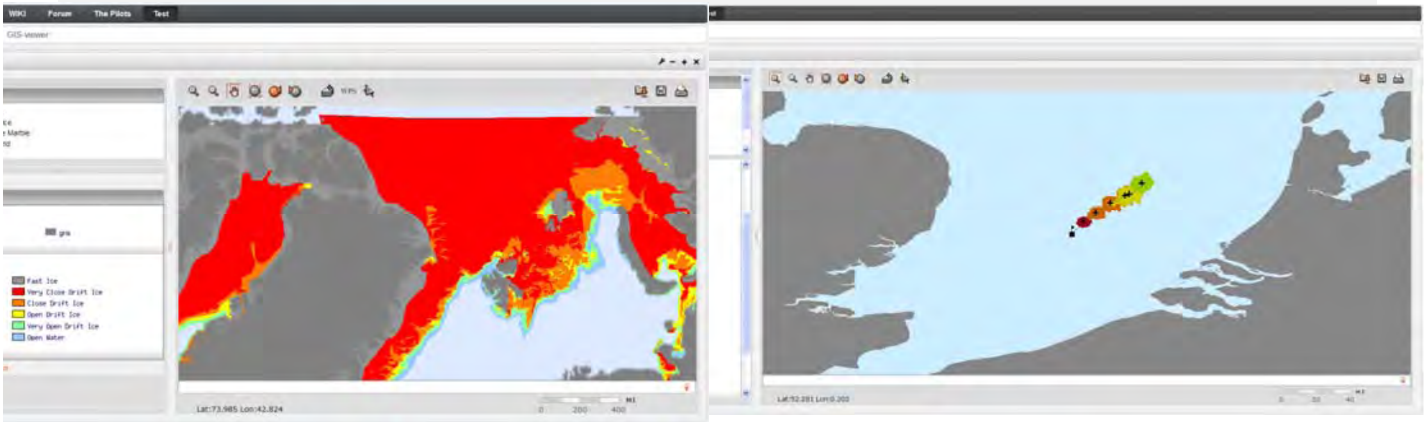


# Objectives and concepts

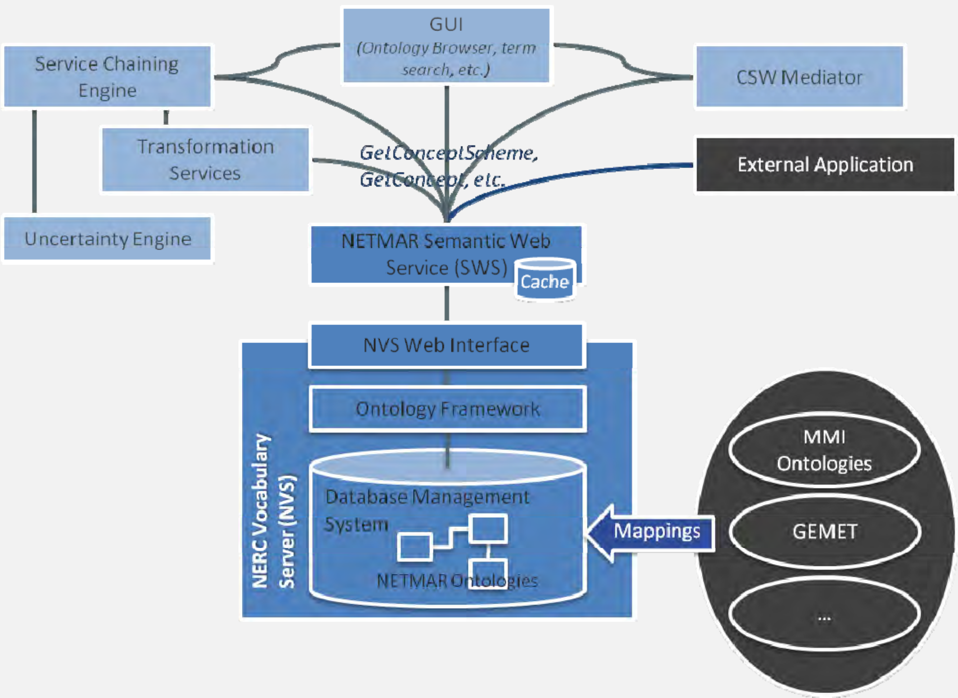


# Pilots

- Pilots in NETMAR
  1. Arctic Sea Ice monitoring and forecasting
  2. Oil spill forecasting and shoreline cleanup
  3. Ecosystem monitoring and modelling
  4. ICAN (International Coastal Atlas Network)

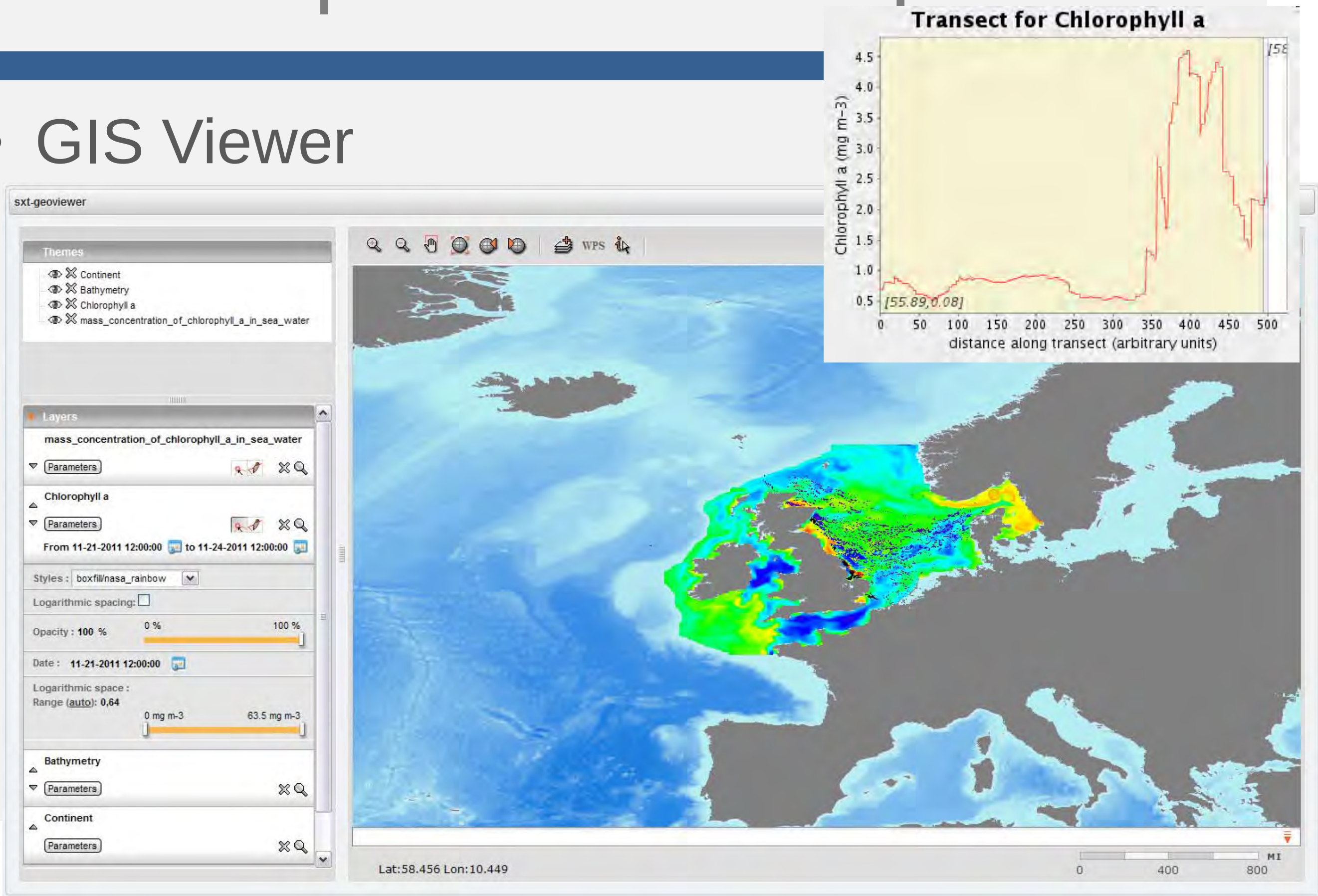


# Ontologies and semantic framework



# EUMIS portal and components

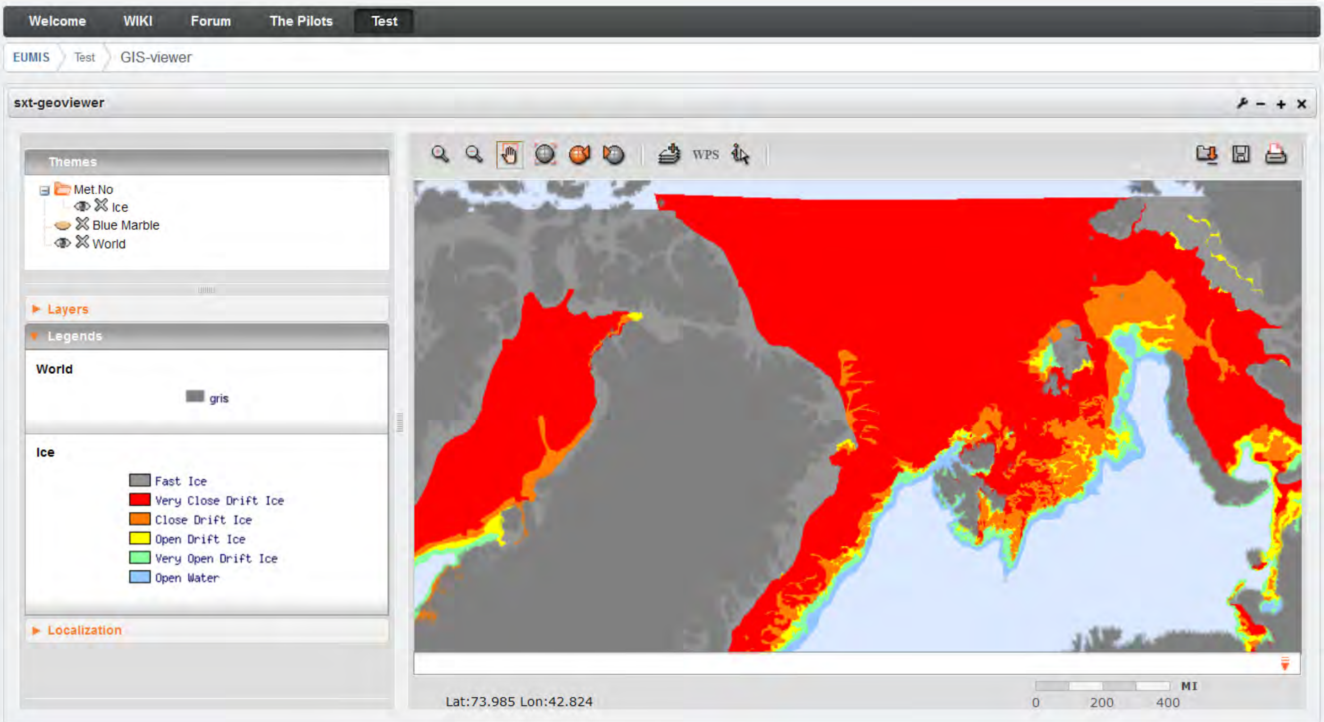
- GIS Viewer




6-7

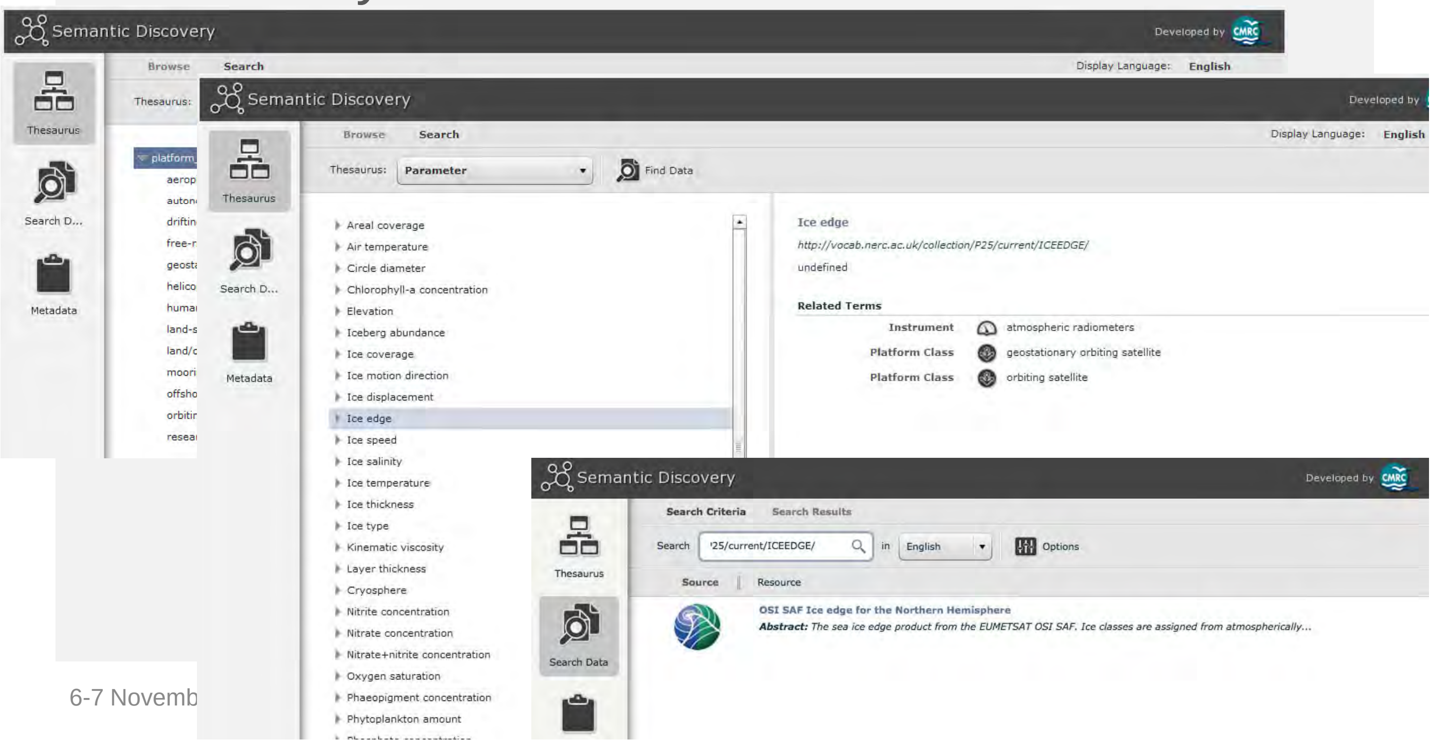
# EUMIS portal and components

- GIS Viewer



# EUMIS portal and components

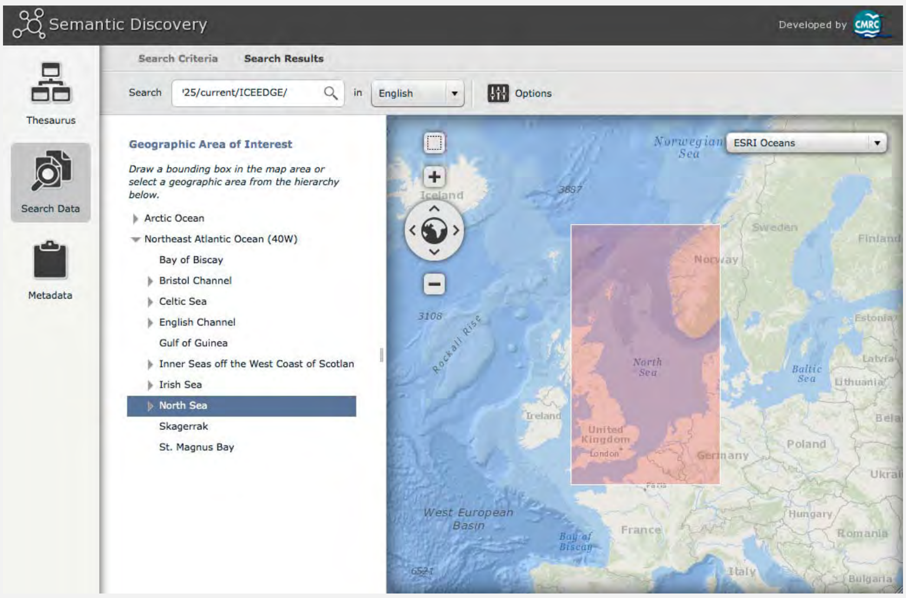
- Discovery Client





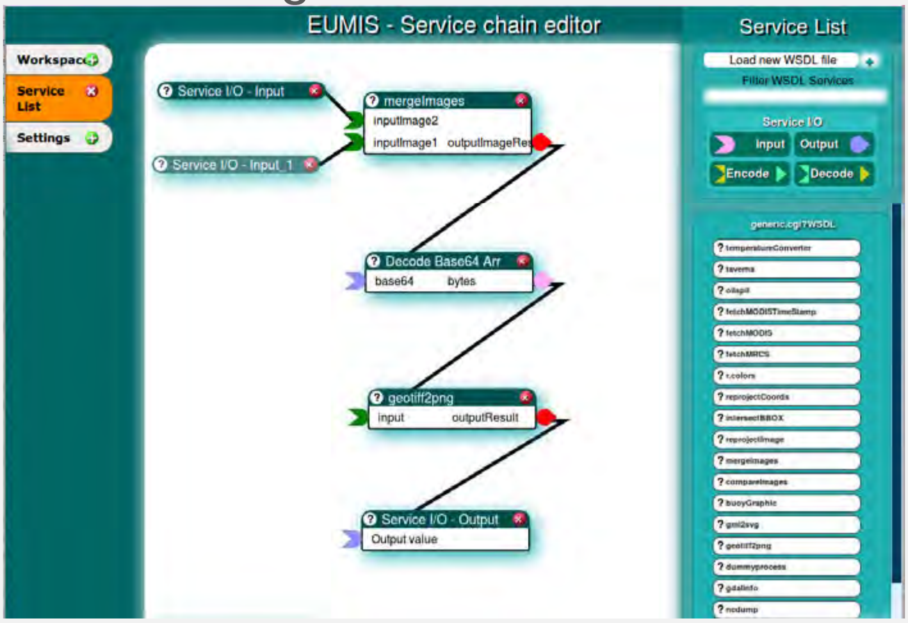

# EUMIS portal and components

- Discovery Client



# EUMIS portal and components

- Service Chaining Editor






# Conclusion

- We have implemented the EUMIS portal with a set of components
  – GIS Viewer
  – Discovery Client
  – Service Chaining Editor
  – Wiki, Forum, RSS feeds

  using multiple programming languages, and deployed them within the Liferay platform.
- The second version of EUMIS is now available for testing. Your feedback is welcome!

# Thank you!

NETMAR web site: http://netmar.nersc.no

EUMIS portal: http://eumis.nersc.no/

Contact Torill Hamre (torill.hamre@nersc.no)

## *Appendix F. Semantic framework specification, 13 August 2012*

The semantic framework specification submitted to the GEOSS Best Practices Wiki contact on 13 August 2012, is enclosed.

# Semantic Framework Specification

## Version 2.0

*2012.08.13*

***Draft***

**Content developed under the EU FP7 NETMAR project**
Open Service Network for Marine Environmental Data (NETMAR)
EU FP7 249024

| | |
|---|---|
| Semantic Framework Specification | Version: 2.0 |
| GEOSS Best Practice Document | 2012.08.13 |


**Author(s)**

**Yassine Lassoued**

Coastal and Marine Research Centre (CMRC)
University College Cork
y.lassoued@ucc.ie


**Document approval**

Document status:
Quality Manager approval:


**Acknowledgements**

The work described in this document has been partially funded by the European Commission under the Seventh Framework Programme, Theme ICT 2009.6.4 ICT for environmental services and climate change adaptation.

## Revision History

| Issue | Date | Editor/Author | Change Records |
|---|---|---|---|
| Draft | 2012.08.13 | Yassine Lassoued | Initial Draft |
| | | | |
| | | | |
| | | | |
| | | | |

## Executive Summary

This document provides a formal specification for the semantic framework designed and implemented as part of the NETMAR project for discovering, accessing and chaining (marine) environmental web services and defines a set of practical semantics use cases. The semantic framework specification defines the interface, called ***Semantic Web Service (SWS)***, for accessing and querying semantic resources using HTTP as the distributed computing platform. Via the SWS, a web user or service can use and combine semantic knowledge.

The semantic framework specification defines a set of practical use cases for semantic knowledge, which are:

1. Ontology Browsing: graphically navigate an ontology or a thesaurus in order to understand the meaning of the concepts (ideas represented by terms) contained therein and to find out how these relate to each other (related, narrower or broader concepts, etc.).
2. Product Discovery: improve the pertinence of the search results of catalogue services, by exploiting the semantic relationships between terms (narrower, related, same as, etc.), and/or trying to interpret the meaning of a user free text keyword according to a given thesaurus.
3. Interoperability: Facilitate the interoperability of two or more information systems (e.g., catalogue services, etc.) using heterogeneous data structures and semantics.
4. Service Chaining: Use semantic knowledge to ensure that the inputs and outputs of each component of a service chain are "semantically compatible."

The SWS interface specification builds on existing work and tries to cover the types of operations required by most common use cases and supported by existing vocabulary services (NVS, GEMET, SISSvoc, MMI).

The semantic web service specified in this document supports the following operations:

1. ***GetCapabilities***
   Retrieves service metadata, including supported operations, response formats, available concept schemes, and their supported languages.
2. ***GetConceptSchemes***
   Lists available concept schemes with their annotations (labels, definitions, etc.).
3. ***GetConceptScheme***
   Returns a concept scheme definition given its URI. The response includes the concept scheme's annotations.
4. ***SearchConceptScheme***
   Returns the definition(s) of one or more concept scheme(s) matching a specified free-text keyword.
5. ***GetConceptSchemeContent***
   Returns the content of a given concept scheme (identified by its URI), including its collections and concepts.
6. ***GetCollections***
   Lists available concept collections with their annotations. Collections may be filtered by one or more concept schemes.
7. ***GetCollection***
   Returns a collection definition identified by its URI. The response includes the collection's annotations.
8. ***SearchCollection***
   Returns the definition(s) of one or more collection(s) matching a specified free-text keyword.
9. ***GetCollectionContent***

Returns the content of a given collection (identified by its URI), including member collections and concepts.

10. ***GetConcepts***
    Returns the definitions of the concepts belonging to a specified concept scheme and/or collection.
11. ***GetConcept***
    Returns a concept definition given its URI. The response includes the concept's annotations.
12. ***SearchConcept***
    A search operation that returns the concepts that textually match a given keyword.
13. ***GetRelatedConcepts***
    Returns the concepts related to one or many given concept(s) using one or many given SKOS relationship(s) (e.g., skos:narrower, skos:broader, skos:related, etc.), both from direct assertions and by entailment.
14. ***GetExplicitTopConcepts***
    Returns the concepts that have explicitly been asserted as top concepts of a specified concept scheme.
15. ***GetImplicitTopConcepts***
    Returns the top-level concepts of a specified concept scheme.
16. ***GetConceptHierarchy***
    This operation is suitable for small thesauri, and is useful for ontology browsers. It returns the hierarchy of the concepts within a given concept scheme and/or collection.
17. ***InterpretKeyword***
    Returns the concepts that semantically match a given keyword, within a specified concept scheme and or collection.
18. ***CheckRelation***
    Checks whether two specified concepts are related via a specified SKOS relationship.

The SWS specification aims to pave the road to the specification of a standard semantic web service for spatial data infrastructures.

## Contents

# 1 Introduction

## *1.1 Context*

The content of this document has been developed under the EU FP7 NETMAR[1] project (EU FP7 249024). NETMAR aims to develop a pilot European Marine Information System (EUMIS) for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. EUMIS is a user-configurable system offering flexible service discovery, access and chaining facilities using OGC, OPeNDAP and W3C standards. It uses a semantic framework (SF) coupled with ontologies for identifying and accessing distributed data, such as near-real time, forecast and historical data. EUMIS also enables further processing of such data to generate composite products and statistics suitable for decision-making in diverse marine application domains.

## *1.2 Problem Addressed*

Several projects and organisations are developing and maintaining large controlled vocabularies and ontologies, and web services for accessing and querying these valuable and complex resources. Despite the momentum gained by semantic technologies in the scientific and spatial data infrastructure (SDI) communities, two fundamental problems still need to be addressed:

1. The use of the developed vocabularies and ontologies and the exploitation of their full power remain very limited. Despite the advances achieved by the IT and the semantic web communities in semantic web technologies and applications, the use of ontologies in the scientific and SDI communities is often limited to metadata or data semantic annotation and thesaurus browsing. There is an obvious lack of practical ontology use cases, applications, and semantically enabled information systems in SDIs; and ontologies often remain used as mere dictionaries or controlled vocabularies.
2. Several semantic web services currently exist, e.g., the NERC Vocabulary Server[2] (NVS), the General Multilingual Thesaurus[3] (GEMET), the SISS Vocabulary Server[4] (SISSvoc), the Marine Metadata Interoperability (MMI) Semantic Framework[5], etc. They all aim at providing a high-level web interface for interrogating SKOS thesauri. But the lack of a standard specification for this type of services has led to disparate models and web interfaces, which makes the integration of this type of services, in order to build integrated cross-domain semantic knowledge, a challenge. A standard or common web interface specification is needed in order to facilitate the interoperability of semantic web services.

The work described in this document aims to help address the problems outlined above by defining a set of practical use cases for semantic knowledge, and by paving the road to the specification of a standard semantic web service for SDIs.

## *1.3 Objective of this Document*

This document provides a formal specification for the semantic framework designed and implemented as part of the NETMAR project for discovering, accessing and chaining (marine) environmental web services and defines a set of practical semantics use cases. The semantic framework specification defines the interface, called ***Semantic Web Service (SWS)***, for accessing and

[1] http://www.netmar-project.eu/

[2] http://vocab.nerc.ac.uk/

[3] https://svn.eionet.europa.eu/projects/Zope/wiki/GEMETWebServiceAPI

[4] https://www.seegrid.csiro.au/wiki/Siss/SISSvoc30Specification

[5] https://marinemetadata.org/semanticframework

querying semantic resources using HTTP as the distributed computing platform. Via the SWS, a web user or service can use and combine semantic knowledge.

The SWS interface specification builds on existing work and tries to cover the types of operations required by most common use cases (ontology browsing, data and service discovery, interoperability, and service chaining) and supported by existing vocabulary services (NVS, GEMET, SISSvoc, MMI).

## 1.4 Scope

The NETMAR semantic framework has been tested in four real-world case studies[6]:

1. Arctic Sea Ice and Met-ocean Observing System,
2. Near real time monitoring and forecasting of oil spill,
3. Ocean Colour - Marine Ecosystem, Research and Monitoring,
4. International Coastal Atlas Network (ICAN) for coastal zone management.

This said, the specification aims to respond to a variety of generic and practical use cases, such as data and metadata semantic annotation, data and service discovery, and service chaining. It is, therefore, of wider relevance to spatial and non-spatial data infrastructures and information systems.

The NETMAR semantic framework has been designed to support the Simple Knowledge Organization System (SKOS) model. The selection of SKOS as a data model was based on the simplicity of this standard while offering enough expressiveness to respond to the needs of the most common use cases (product discovery, interoperability, service chaining, etc.).

## 1.5 Terminology

This document uses several key terms that are defined as follows.

***Semantic Framework***

The key concept in this report is "semantic framework". In reality, there is no formal or common definition for such a concept.

The term "***semantic framework***" (SF) as used in this report means: a collection of classes, libraries, application programming interfaces (APIs), or applications that can be used to build semantics-aware information systems that integrate, manage, handle or deliver semantic knowledge related to the information system's data and services.

***Operation***

Specification of a transformation or query that an object may be called to execute [Pe02]

***Interface***

A named set of operations that characterise the behaviour of an entity [Pe02]

***Service***

A distinct part of the functionality that is provided by an entity through interfaces [Pe02]

***Service Instance***

An actual implementation of a service; service instance is often interchangeable with server

***Client***

[6] http://www.netmar-project.eu/content/pilots

A software component that can invoke an operation from a server

***Request***
An invocation by a client of an operation

***Response***
The result of an operation returned from a server to a client

***Capabilities XML***
Service-level metadata describing the operations and content available at a service instance

***Recommendations***
For recommendations, the following terms are used.

**Rule**
Rules SHALL be followed to ensure compatibility and/or conformance with standards, directives or the project objectives. A rule is characterised by the use of the words SHALL and SHALL NOT.

**Recommendation**
Recommendations consist of advice to implementers that will affect the usability of the final module (here the NETMAR semantic framework). A recommendation is characterised by the use of the words SHOULD and SHOULD NOT.

**Permission**
Permissions clarify areas of the specification that are not specifically prohibited. Permissions reassure the reader that a certain approach is acceptable and will cause no problem. Permissions are characterised by the use of the word MAY.

**SHALL**
"SHALL" is a keyword indicating a mandatory requirement. Designers SHALL implement such mandatory requirements in order to ensure conformance with the project objectives. This word is usually associated with a rule.

**SHOULD**
"SHOULD" is a keyword indicating flexibility of choice with a strongly preferred implementation. This word is usually associated with a recommendation.

**MAY**
"MAY" is a keyword indicating flexibility of choice with no implied preference. This word is usually associated with permissions.

## *1.6 Organisation of this Document*

This document is organised as follows…

# 2 Vocabularies and Semantics Use Cases

In order to specify the semantic framework functionality, it is crucial to understand its intended usage and application beforehand. The NETMAR project defined a set of practical and generic use cases (applications) for ontologies that aim to improve the pertinence of an SDI. Each use case typically requires a set of recurrent semantic operations. This specification tries to cover the common operations and capabilities required by the identified use cases. However, more use case may be defined, which may require additional operations.

## *2.1 Use Case 1 (UC-1): Ontology Browsing*

Ontology browsing is the ability to graphically navigate an ontology or a thesaurus in order to understand the meaning of the concepts (ideas represented by terms) contained therein and to find out how these relate to each other (related, narrower or broader concepts, etc.). Ontology browsing is useful in web atlases as a way of providing educational information about a given domain (domain knowledge). It is also commonly used in product discovery interfaces (c.f., UC-2, section 2.2) as a way to find products by topic (e.g., multi-faceted product browsing). Existing semantic web services usually provide graphical ontology browsers.

Typically, an ontology browser needs to find out:

1. What thesauri are delivered by a given semantic service;
2. What terms or collections of terms are contained within a given thesaurus or term collection;
3. What terms are related (same as, narrower, broader, etc.) to a given term;
4. What thesauri, term collections, or terms match a given free text keyword.

Additionally, an ontology browser may request the semantic service to:

5. Build a graph structure, most commonly hierarchy, of the terms of a given thesaurus or term collection.

## *2.2 Use Case 2 (UC-2): Product Discovery*

Ontologies may be used by product (e.g., data, services, etc.) discovery services (e.g., catalogue services) as a means to improve the pertinence of their search results, by exploiting the semantic relationships between terms (narrower, related, same as, etc.), and/or trying to interpret the meaning of a user free text keyword according to a given thesaurus. For instance, if you search for datasets matching the term “seabed”, you would be able to get those tagged with the keyword “seafloor” (synonym), or if you search for “CTD” (i.e., Conductivity, Temperature, Depth), you would be able to get “Sea Surface Salinity” datasets.

Typically, a semantically enabled product discovery service needs to find out:

1. What terms are related to a given term (same as, narrower, related, etc.);
2. What terms match a given free text keyword.

## *2.3 Use Case 3 (UC-3): Interoperability*

Database and service interoperability is another common use case of ontologies and semantic knowledge. Typically, in this area, ontologies are used as a mapping mechanism between

A. Two data structures/schemas (structural interoperability);
B. The values of similar properties (attributes) in different databases, using different representations (semantic interoperability).

The former occurs when two information systems, with structurally heterogeneous backends, need to interoperate with each other or with a third party system, e.g., mediator, broker, extract transform and load (ETL) tool, etc. The latter supposes that structural interoperability has already been achieved and that actual data values need to be mapped or translated from one model, classification scheme, or terminology, to another. This is the typical case of distributed catalogue services using different vocabularies, possibly from different domains or in different natural languages, for product metadata values, e.g., descriptive keywords, units of measure, parameter names, organisation names, etc.

Structural and semantic interoperability, in the general case, may require complex ontology models and mappings, able to capture database primary and foreign keys, or how to convert attribute values from one model into another (e.g., degrees Celsius to Fahrenheit, etc.). However, in several practical cases (e.g., catalogue service interoperability), a simple semantic model relying on basic semantic relationships, such as broader, narrower, and related, may be sufficient to perfectly interoperate two semantically or structurally heterogeneous systems. In such a case, the interoperable agent (ETL, mediator, etc.) typically requires information such as:

1. What terms are related (same as, narrower, broader, etc.) to a given term; this is useful to translate one attribute value from one model into another for example;
2. What terms best match a keyword; this is useful for an ETL or a schema matcher to find out which table or attribute from a destination database schema matches a table or attribute from a data source schema.

## *2.4 Use Case 4 (UC-4): Service Chaining*

A service chain is defined in ISO 19119 as “a sequence of services where, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action” [ISO05]. From a user perspective, service chaining is the linking together of standardised data and processing services into a workflow to produce results that are not predefined by the service providers. The defined workflows will then be passed to a workflow engine for execution.

Typically, a processing service requires input data with given structure and semantics and outputs data also with given structure and semantics. While chaining services, one must make sure that the data output by a service and fed into another conform to the required structure and semantics of the latter.

It is possible to use semantic knowledge to ensure that the inputs and outputs of each component of a service chain are “semantically compatible.” This can be achieved by using ontologies as a way to represent the semantics of data and service input and output parameters. The service-chaining engine needs then to check whether an input dataset parameter is “semantically compatible” with a processing service input parameter (in terms of the parameter type, dimension, unit of measure, etc.). In this way, correct connections between components can be enforced (e.g., do not send chlorophyll data to a component that only knows how to process sea surface temperature).

Checking the semantic compatibility of two parameters in most practical cases is a matter of checking whether two concepts are the same, or whether a concept is narrower/broader than the other, or whether both concepts share a common broader concept. Therefore, it is commonly possible to express parameter compatibility as a combination of one or more of the following operations:

1. What concepts (parameters, dimensions, units, etc.) are narrower or broader than a given concept (parameter, dimension, unit, etc.);
2. Is a given concept (parameter, dimension, unit, etc.) narrower than another given concept (parameter, dimension, unit, etc.)?

## 2.5 Note

The use cases defined above have been implemented for the NETMAR case studies listed in section 1.4. More use cases may be defined for other case studies, and may require extra types of operations.

# 3 Semantic Data Model

The semantic web service specified in this document is primarily intended to respond to most common/recurring queries over SKOS-based ontologies.

In this chapter, we introduce basic SKOS constructs. As SKOS is available in OWL (Web Ontology Language [DS04]), we firstly define a few useful OWL constructs, such as classes and properties.

This chapter does not intend to provide a full SKOS documentation. Rather it defines basic notions required to understand the remaining of the document. For further details about SKOS, readers are referred to the SKOS reference [MB09].

## 3.1 Terminology

### 3.1.1 *Resources*

A resource is an abstract term that refers to any of the notions defined in the subsequent subsections. We refer to resources using the notation <namespace>:<resourceName>, where <namespace> refers to the namespace of the vocabulary or XML schema in which the resource is defined, and <resourceName> refers to the local name of the resource. For instance, owl:Class refers to the construct "Class" defined in the OWL language.

In this chapter, in addition to SKOS and OWL, we use the following vocabularies:

- RDF, the Resource Description Framework [MM04], a language for representing information about resources in the World Wide Web;
- RDF-S, i.e., RDF Schema [BG04], which extends RDF by adding more modelling primitives such as classes, inheritance, domains, etc.

Below is the list of namespaces used in this chapter.

```
rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
owl:  http://www.w3.org/2002/07/owl#
skos: http://www.w3.org/2004/02/skos/core#
```

### 3.1.2 *Classes & Instances (Individuals)*

The term "*class*", as used in this document, is interchangeable with "*OWL class*" (owl:Class) [DS04]. In OWL, classes provide an abstraction mechanism for grouping resources that share similar characteristics. For instance, Instrument or Parameter can be regarded as classes. A resource belonging to a given class is called *instance* of that class or also *individual*. For example, Thermometer is an instance of Instrument.

A class may be a *subclass* of another, in which case the latter would be called *super-class* of the former. Inheritance relationships (sub-class and super-class) are transitive. In OWL, all classes are direct or indirect sub-classes of the OWL Thing class (owl:Thing).

### 3.1.3 *Properties*

RDF properties (rdf:Property) provide a mechanism for creating relationships amongst resources or between resources and data values. OWL refines RDF properties into four types of properties:

- Object properties (owl:ObjectProperty), which link individuals to individuals; for instance an Instrument ***measures*** a Parameter; therefore *measures* may be considered as an object property;
- Data type properties (owl:DatatypeProperty), which link individuals to data values (also called atomic values or literals); for instance: a Place may have a *northernmost latitude* which can be defined as a data type property;
- Annotation properties (owl:annotationProperty), which link resources (classes, individuals, properties, etc.) to data values or resources; for instance the RDFS label property (skos:label) and the SKOS preferred label property (skos:prefLabel) are annotation properties;
- Ontology properties (owl:OntologyProperty), which link ontologies to resources or data values; for instance owl:backwardCompatibleWith (backward compatible with) is an ontology property.

According to RDFS, a property may be a sub-property of another (see examples below from SKOS). OWL provides constructs for specifying mathematical properties of properties, such as transitivity, symmetry, inverse property, etc.

### 3.1.4 *SKOS Concept Schemes*

In SKOS, vocabularies MAY be organised into concept schemes (skos:ConceptScheme). A concept scheme can be regarded as a knowledge organisation system, a thesaurus or a classification scheme.

In SKOS, concept schemes SHALL be implemented as instances of the SKOS *ConceptScheme* class (skos:ConceptScheme). Below is an example of a concept scheme definition.

```
<skos:ConceptScheme rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3">
  <skos:definition xml:lang="en">
    Terms used to classify SeaDataNet Agreed Parameter Groups to provide
    topic/theme level terms in a hierarchical parameter discovery interface
  </skos:definition>
  <skos:prefLabel xml:lang="en">
    SeaDataNet Parameter Disciplines
  </skos:prefLabel>
  <!--...-->
</skos:ConceptScheme>
```

### 3.1.5 *SKOS Concepts*

A concept is defined in the SKOS reference [MB09] as “an idea or notion; a unit of thought.” SKOS concepts are instances of the SKOS Concept class (skos:Concept). As we are only supporting OWL DL (and consequently OWL Lite) [MH04], concepts SHALL NOT be classes. Nevertheless they MAY be organised internally into OWL classes. For instance, one MAY define a class *Place* for containing place concepts, and a sub-class of that named *ICESDivision for ICES divisions*.

Concepts SHALL be linked to concept schemes using the skos:inScheme object property as shown in the example below.

```
<skos:Concept rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3/DS10">
  <skos:inScheme rdf:resource="http://vocab.ndg.nerc.ac.uk/list/P081/3"/>
  <skos:prefLabel xml:lang="en">Environment</skos:prefLabel>
  <skos:definition xml:lang="en">
    The domain documenting the activities of man that have an effect on the
    Earth System
  </skos:definition>
</skos:Concept>
```

IF a concept is a top-level concept of a given concept scheme then it SHOULD be defined as such using the skos:topConceptOf object property (the inverse of which is skos:hasTopConcept). A concept is a top concept of a concept scheme if it has no broader concept within the concept scheme in question[7].

```
<skos:ConceptScheme rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3">
  <!–Concept scheme definition...-->
  <skos:hasTopConcept>
    <skos:Concept rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3/DS10">
      <!--...-->
    </skos:Concept>
  </skos:hasTopConcept>
</skos:ConceptScheme>
```

Please note that skos:topConceptOf is a sub-property of skos:inScheme (c.f. Figure 2.2). Therefore, if a concept is defined as a top concept of a given concept scheme then there is no need to define it as belonging to it.

```
skos:inScheme
 |
 +- skos:topConceptOf <--> (inverse of skos:hasTopConcept)
```

**Figure 3.1.** Concept-Concept Scheme Relationships

### 3.1.6 *SKOS Collections*

Another useful notion in SKOS is that of collections (skos:Collection). SKOS collections are groups of concepts that share something in common and which can be labelled (skos:Collection) or ordered (skos:OrderedCollection); c.f., Figure 2.7.

```
skos:Collection [A meaningful collection of concepts]
 |
 +- skos:OrderedCollection [An ordered collection of concepts, where both the grouping and the
                            ordering are meaningful]
```

**Figure 3.2.** SKOS Collections

SKOS collections are useful in general as a way to define simple classification schemes within a thesaurus (concept scheme) or as a way to order concepts. For instance, concepts representing remote sensing instruments may be grouped into one collection. A collection may contain concepts and/or other collections.

Collection memberships are expressed using the skos:member and skos:memberList properties. The former is used to ascertain that a collection contains a concept or another collection. The latter is used to list the elements of an ordered collection in an rdf:list [MM04].

Below is an example that defines two collections: *RemoteSensingInstruments* and *ActiveRemoteSensingInstruments*. The latter contains two concepts, MultibeamEchosounder and

[7] Please note that, as per the SKOS reference, this is not a strict rule in SKOS.

SingleBeamEchosounder, and is a member of the former. For the sake of readability, we do not use URIs as identifiers in the subsequent examples.

```
<skos:Collection rdf:ID="RemoteSensingInstruments">
  <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
  <!--Collection definition here: labels, definition, etc.-->
  <skos:member>
    <skos:Collection rdf:ID="ActiveRemoteSensingInstruments">
      <!--Collection definition here: labels, definition, etc.-->
      <skos:member>
        <skos:Concept rdf:ID="MultieamEchosounder">
          <!--Concept definition here: labels, definition, etc.-->
        </skos:Concept>
      </skos:member>
      <skos:member>
        <skos:Concept rdf:ID="SingleBeamEchosounder">
          <!--Concept definition here: labels, definition, etc.-->
        </skos:Concept>
      </skos:member>
    </skos:Collection>
  </skos:member>
</skos:Collection>
```

The example below defines an ordered collection, called MarineStrata, which consists of four concepts: SubSeabed, Seabed, WaterColumn, and WaterSurface, ordered vertically from the lowest to the highest.

```
<skos:OrderedCollection rdf:ID="MarineStrata">
  <skos:prefLabel xml:lang="en">Marine Strata</skos:prefLabel>
  <!--Collection definition here: labels, definition, etc.-->
  <skos:memberList>
    <rdf:List>
      <rdf:first>
        <skos:Concept rdf:ID="SubSeabed"/>
      </rdf:first>
      <rdf:rest>
        <rdf:List>
          <rdf:first>
            <skos:Concept rdf:ID="Seabed"/>
          </rdf:first>
          <rdf:rest>
            <rdf:List>
              <rdf:first>
                <skos:Concept rdf:ID="WaterColumn"/>
              </rdf:first>
              <rdf:rest>
                <rdf:List>
                  <rdf:first>
                    <skos:Concept rdf:ID="WaterColumn"/>
                  </rdf:first>
                </rdf:List>
              </rdf:rest>
            </rdf:List>
          </rdf:rest>
        </rdf:List>
      </rdf:rest>
    </df:List>
  </skos:memberList>
</skos:OrderedCollection>
```

### 3.1.7 *SKOS Annotations*

SKOS defines a set of annotation properties for annotating resources (classes, properties, concepts, etc.). SKOS annotation properties are divided into two groups: lexical labels and documentation properties (or note properties).

#### 3.1.7.1 Lexical Labels

Lexical labels, the values of which are UNICODE characters in a given natural language, are used to associate various types of labels with concepts or resources in general, such as preferred label or alternative labels, etc.

```
rdfs:label
 |
 +- skos:prefLabel
 |
 +- skos:altLabel
 |
 +- skos:hiddenLabel
```

**Figure 3.3.** SKOS Lexical Labels

As shown in Figure 2.4, all SKOS lexical labels are sub-properties of the RDFS label annotation property (rdfs:label). Preferred and alternative labels (skos:prefLabel and skos:altLabel) are human-readable representations of concepts or resources. A hidden label MAY be a popular misspelled label, which may be useful for improving vocabulary search. For instance, one may define "Tide Guage" (which is a very common misspelling of "Tide Gauge") as a hidden label for the concept "Tide Gauge." This would help users find the term "tide gauge" event when they misspell it as "tide guage".

Below is an example of a valid concept definition using the three SKOS lexical labels (preferred, alternative and hidden labels).

```
<skos:Concept rdf:ID="TideGauge">
  <skos:prefLabel xml:lang="en">Tide Gauge</skos:prefLabel>
  <skos:altLabel xml:lang="en">Tide Gage</skos:altLabel>
  <skos:hiddenLabel xml:lang="en">Tide Guage</skos:hiddenLabel>
  <!--...-->
</skos:Concept>
```

Preferred, alternative and hidden labels in a given language SHALL be all mutually exclusive. Therefore, the same term SHALL NOT be used simultaneously as alternative label and preferred label (or as preferred label and hidden label, or as alternative label and hidden label) within one given language.

Every concept SHOULD at least have a label (preferred label) per language, even if the label is the same as the concept ID (c.f. example above). And, as per the SKOS specification [MB09], a resource SHALL NOT have more than one preferred label (skos:prefLabel value) per language.

Finally, as all SKOS lexical labels are sub-properties of the RDFS label, ontology administrators SHOULD avoid defining RDFS labels for the ontology resources. Rather they SHALL use the SKOS lexical labels, which are more specific. RDFS labels can be inferred from SKOS labels.

#### 3.1.7.2 Documentation Properties

SKOS defines a set of documentation properties (also called note properties) designed to provide information about SKOS concepts. The top level SKOS documentation property is skos:note which may be used directly, or as a super-property for more specific note types. The SKOS documentation properties are illustrated in Figure 2.5. The main of them is skos:definition which provides a statement or formal explanation of the meaning of a concept. Definitions are very useful as they explain to users what the concepts within a given ontology actually mean, thus allowing for the disambiguation of terms. Therefore, every concept SHALL have at least one definition per supported language.

Please note that there is no restriction on the nature of the information provided by SKOS documentation properties. This can be plain text, hypertext, image, or even an object (individual). Nevertheless, in this version of the semantic framework, we only support plain text literals.

```
skos:note [A general note, for any purpose]
 |
 +- skos:changeNote [A note about a modification to a concept]
 |
 +- skos:definition [A statement or formal explanation of the meaning of a concept]
 |
 +- skos:editorialNote [A note for an editor, translator or maintainer of the vocabulary]
 |
 +- skos:example [An example of the use of a concept]
 |
 +- skos:historyNote [A note about the past state/use/meaning of a concept]
 |
 +- skos:scopeNote [A note that helps to clarify the meaning and/or the use of a concept]
```

**Figure 3.4.** SKOS Documentation Properties

### 3.1.8 *SKOS Semantic Relations*

SKOS provides a set of object properties, known as semantic relations, useful to link concepts **within a concept scheme**. All SKOS semantic relations are sub-properties of skos:semanticRelation. Figure 2.6 shows the SKOS semantic relations, which are represented in black (red ones correspond to mapping properties, c.f., subsection 2.1.8).

```
skos:semanticRelation [Links a concept to a concept related by meaning]
 |
 +- skos:broaderTransitive [Transitive super-property of skos:broader]
 |    |
 |    +- skos:broader [Relates a concept to a concept that is more general in meaning]
 |         |
 |         +- skos:broadMatch [See below]
 |
 +- skos:narrowerTransitive [Transitive super-property of skos:narrower]
 |    |
 |    +- skos:narrower [Relates a concept to a concept that is more specific in meaning]
 |         |
 |         +- skos:narrowMatch [See below]
 |
 +- skos:related [A statement or formal explanation of the meaning of a concept]
 |    |
 |    +- skos:relatedMatch [See below]
 |
 +- skos:mappingRelation [Relates two concepts coming, by convention, from different schemes]
      |
      +- skos:broadMatch [Hierarchical mapping between two concepts in different schemes]
      |
      +- skos:narrowMatch [Hierarchical mapping between two concepts in different schemes]
      |
      +- skos:relatedMatch [Associative mapping between two concepts in different schemes]
      |
      +- skos:closeMatch [Links two concepts that are similar enough to be used interchangeably]
           |
           +- skos:exactMatch [Links two concepts with a high degree of similarity]
```

**Figure 2.6.** SKOS Semantic Relations

By convention, those semantic relations written in italic SHALL NOT be used directly to make assertions. Rather they SHOULD be used to draw inferences about the hierarchical structure of relationships or their transitivity.

For instance, assume you have two concepts *Geology* and *MarineGeology*, the former being broader then the latter. Then, you SHOULD NOT make the assertion

MarineGeology skos:broaderTransitive Geology,

or the assertion

MarineGeology skos:semanticRelation Geology.

Rather you SHOULD make the following assertion (from which the above assertions can be inferred if requested by a user)

MarineGeology skos:broader Geology.

### 3.1.9 *SKOS Mapping Properties*

SKOS provides a set of mapping properties intended for linking concepts **from different concept schemes**. The SKOS mapping properties are all sub-properties of skos:mappingRelation. SKOS mapping properties are represented in red in Figure 2.6.

When mappings concepts from different schemes one SHOULD use SKOS mapping properties rather than SKOS semantic relations.

## *3.2 Multilinguality*

As per the best practices and guidelines for multilinguality defined by the European Environment Information and Observation Network (EIONET)[8]:

1. Language codes SHALL follow the ISO 639-1 international standard;
2. In order to be able to display Latin, Greek, Armenian, Georgian and Cyrillic on the same page, we SHALL use a character set that contains all these alphabets. Therefore, we SHALL use UTF-8 as the character code for content transmitted to the web browser (or semantic framework clients in general).

In RDF and OWL, labels and definitions of resources SHALL declare their languages using the xml:lang attribute. An example of a SKOS multilingual concept is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
...
<skos:Concept rdf:ID=”Geology”>
  <skos:prefLabel xml:lang=”en”>Geology</skos:prefLabel>
  <skos:prefLabel xml:lang=”fr”>Géologie</skos:prefLabel>
  <skos:definition xml:lang=”en”>The scientific study of the origin, history,
structure, and composition of the earth</skos:definition>
  <skos:definition xml:lang=”fr”>Domaine scientifique qui étudie l'origine de la
Terre, son histoire, sa forme, les matériaux qui la composent et les processus
qui ont agi sur elle ou qui agissent encore</skos:definition>
  ...
</skos:Concept>
```

[8] http://www.eionet.europa.eu/

As per the SKOS specification [MB09], a resource SHALL NOT have more than one preferred label (skos:prefLabel value) per language tag.

### *3.3 General OWL and SKOS Rules and Recommendations*

When working with SKOS, one must bear in mind that concepts, collections, and concept schemes SHALL be mutually disjoint. Therefore, a concept cannot be a concept scheme or a collection, and a collection cannot be a concept scheme. In the same way, you SHALL NOT use SKOS semantic relations or mapping properties to link a collection to a concept (or vice versa) or a concept scheme to a collection or a concept, and so on. In fact, SKOS semantic relations and mapping properties SHALL only link concepts to concepts.

More rules and recommendations related to SKOS can be found in the SKOS reference [MB09].

The semantic knowledge internal to the semantic web service MAY benefit from the rich capabilities and expressiveness of OWL. For instance it MAY organise concepts internally in a hierarchy of OWL classes and MAY use inference rules, class restrictions, OWL and OWL 2 constructs, etc. However this SHOULD be transparent to the SWS users. Data delivered by the SWS SHOULD be SKOS-compliant.

### *3.4 Semantic Resources Identification*

Semantic resources SHOULD be identified using their URIs.

It is very common that the URI of a semantic resource within a concept scheme or ontology is of the form: <namespace>#<resource local name>, where <namespace> is an ontology or concept scheme URI. However, this is not a general rule. For instance, the NERC Vocabulary Server (NVS)[9] uses URIs of the form:

- http://vocab.ndg.nerc.ac.uk/list/<listReference>/<listVersion> for lists (i.e., concept schemes),
- http://vocab.ndg.nerc.ac.uk/term/<listReference>/<listVersion>/<entryReference> for entry terms (i.e., concepts).

[9] http://www.bodc.ac.uk/products/web_services/vocab/

# 4 Semantic Framework - Overview

The typical architecture of the proposed semantic framework is illustrated in Figure 4.1. As shown in this diagram, the SWS uses semantic knowledge that may be stored and managed in a variety of ways, e.g., triple store, database, ontology files (RDF, OWL, etc.), external web service, etc.

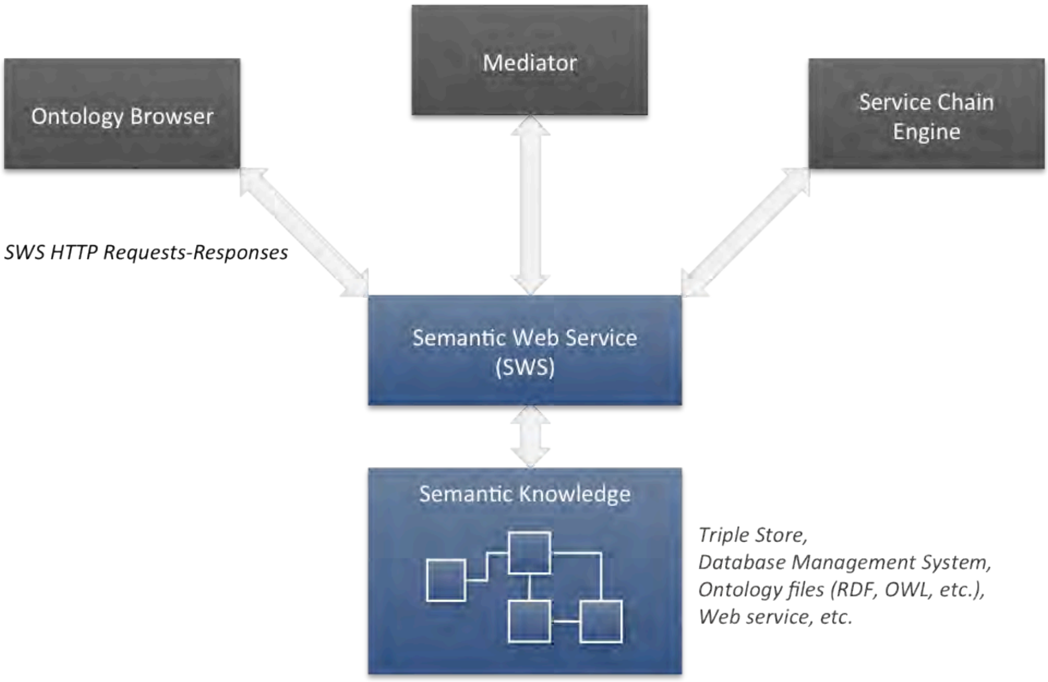



**Figure 4.1.** Typical Semantic Framework Architecture

The SWS is made accessible on the web to "semantically-enabled" applications, such as ontology browsers, semantic mediators, or service chain engines. These interact with the SWS using SWS requests and responses over HTTP.

## *4.1 SWS Operations*

The SWS aims to provide a web interface for querying SKOS thesauri through a standardised set of high-level and easy-to-use operations required by most common semantically-enabled clients such as data and metadata mediators, service chaining engines, vocabulary browsers and data and service discovery interfaces. The SWS supports the following operations.

19. ***GetCapabilities***
    Retrieves service metadata, including supported operations, response formats, available concept schemes, and their supported languages.

20. ***GetConceptSchemes***
    Lists available concept schemes with their annotations (labels, definitions, etc.).

21. ***GetConceptScheme***
    Returns a concept scheme definition given its URI. The response includes the concept scheme's annotations.

22. ***SearchConceptScheme***
    Returns the definition(s) of one or more concept scheme(s) matching a specified free-text keyword.

23. ***GetConceptSchemeContent***
    Returns the content of a given concept scheme (identified by its URI), including its collections and concepts.

24. ***GetCollections***
    Lists available concept collections with their annotations. Collections may be filtered by one or more concept schemes.

25. ***GetCollection***
    Returns a collection definition identified by its URI. The response includes the collection's annotations.

26. ***SearchCollection***
    Returns the definition(s) of one or more collection(s) matching a specified free-text keyword.

27. ***GetCollectionContent***
    Returns the content of a given collection (identified by its URI), including member collections and concepts.

28. ***GetConcepts***
    Returns the definitions of the concepts belonging to a specified concept scheme and/or collection.

29. ***GetConcept***
    Returns a concept definition given its URI. The response includes the concept's annotations.

30. ***SearchConcept***
    A search operation that returns the concepts that textually match a given keyword.

31. ***GetRelatedConcepts***
    Returns the concepts related to one or many given concept(s) using one or many given SKOS relationship(s) (e.g., skos:narrower, skos:broader, skos:related, etc.), both from direct assertions and by entailment.

32. ***GetExplicitTopConcepts***
    Returns the concepts that have explicitly been asserted as top concepts of a specified concept scheme.

33. ***GetImplicitTopConcepts***
    Returns the top-level concepts of a specified concept scheme.

34. ***GetConceptHierarchy***
    This operation is suitable for small thesauri, and is useful for ontology browsers. It returns the hierarchy of the concepts within a given concept scheme and/or collection.

35. ***InterpretKeyword***

Returns the concepts that semantically match a given keyword, within a specified concept scheme and or collection.

36. ***CheckRelation***
Checks whether two specified concepts are related via a specified SKOS relationship.

## 4.2 SWS Client-Service Interaction

The interaction between an SWS client and a server relies on a series of SWS operation calls and responses. In this section we provide two examples of SWS client-service interactions: vocabulary (or ontology) browsing, and catalogue service mediation.

### 4.2.1 *Vocabulary Browsing*

The protocol diagram of Figure 4.2 outlines the typical protocol to be followed in order to process semantic web service requests involved in vocabulary browsing.

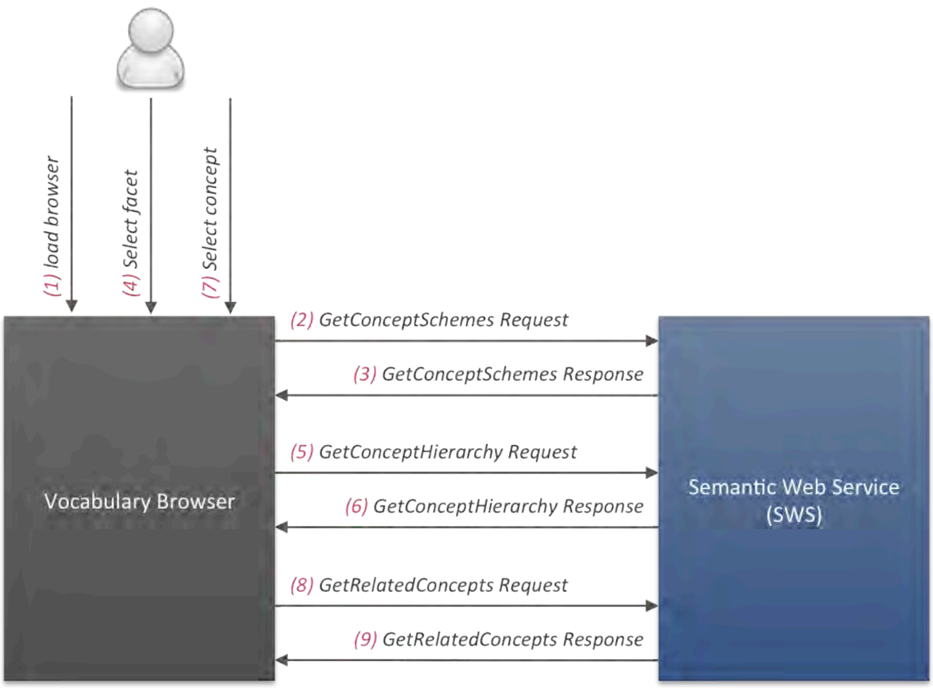



**Figure 4.2.** Protocol Diagram for an SWS Client-Service Interaction involved in Vocabulary Browsing

This protocol diagram above involves 3 actors:

- The user of the system,
- The vocabulary browser,
- The semantic web service.

The typical component interactions illustrated in this figure are explained below.

- (1) The user loads the vocabulary browser (VB).
- (2) While loading, the VB submits a *GetConceptSchemes* request to the SWS in order to get all information (definitions, labels, etc.) related to the thesauri (concept schemes).

- (3) The SWS returns the list of concept schemes together with their definitions and labels, which are used then displayed by the browser.
- (4) The user selects the concept scheme of interest.
- (5) The VB sends a *GetConceptHierarchy* request to the SWS with the aim to retrieve a hierarchy of the concepts contained within the concept scheme selected by the user.
- (6) The SWS returns the concept hierarchy for the concept scheme selected by the user and the VB displays it.
- (7) The user selects a concept of interest from the concept hierarchy.
- (8) In order to display the list of concepts related to the selected concept, the VB sends a *GetRelatedConcepts* request to the SWS.
- (9) The SWS responds back and the OB displays the related concepts

The interactions can continue from step 4 (user selects a different concept scheme, or selects a related concept from a different concept schemes) or step 7 (use selects a related concept from the current concept scheme).

### 4.2.2 *Catalogue Service Mediation*

The protocol diagram of Figure 4.3 outlines the typical protocol to be followed in order to process semantic web service requests involved in the mediation of OGC catalogue services (CSW) [NW05].

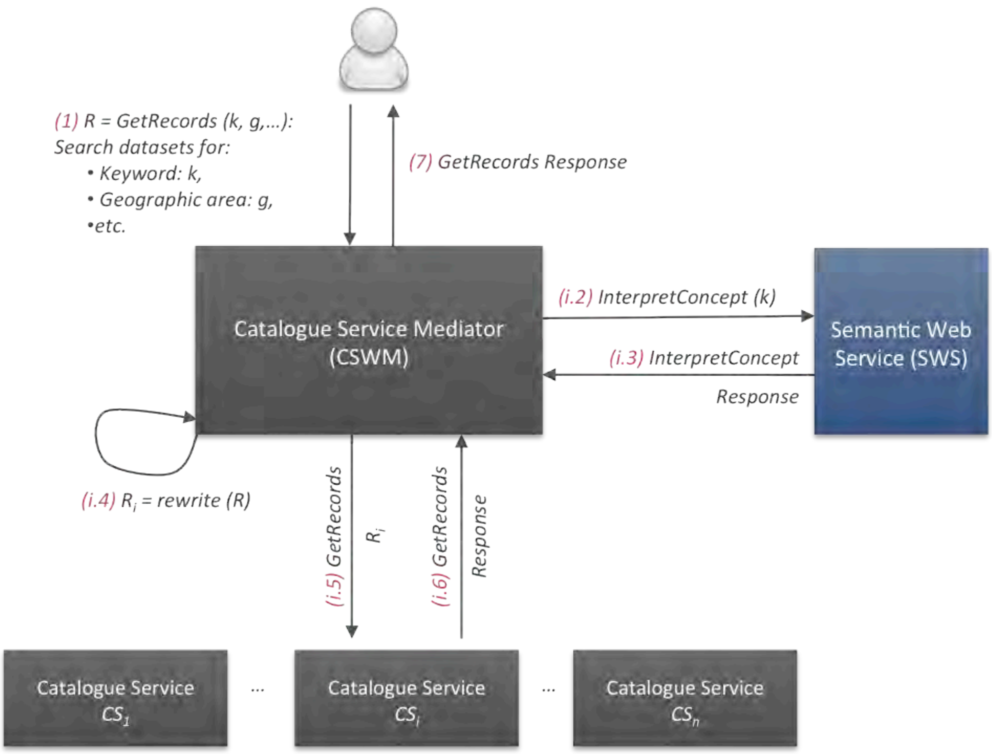



**Figure 4.3.** Protocol Diagram for an SWS Client-Service Interaction involved in Catalogue Service Mediation

This protocol diagram above involves 4 actors:

- The user of the system,
- The catalogue service mediator (CSWM),
- The semantic web service,
- A set of distributed catalogue services (CSW).

The typical component interactions illustrated in this figure are explained below.

- (1) The user sends a GetRecords request to the catalogue service mediator to search datasets for a given keyword *k* and a geographic extent *g*.
- The CSWM parses the user query *R*, and for each catalogue service $CS_i$ to be involved in the query (by default all the catalogue nodes are involved unless otherwise specified in the user request R) it does the following (in parallel):
    - (i.2) The CSWM extracts the keyword of interest *k* and submits an InterpretKeyword request to the SWS with *k* as the keyword parameter value.
    - (i.3) The SWS responds to the CSW by sending the list of concepts semantically covered by the user keyword *k* (i.e., best matches of the keyword and their narrower concepts).
    - (i.4) The CSWM, now, has all the concepts covered by the user's keyword. It rewrites the user's request into a request supported by catalogue node $CS_i$ and translates the original keyword using the concepts obtained in step (i.4). Let $R_i$ denote the so-obtained query.
    - (i.5) The CSWM submits the rewritten query $R_i$ to catalogue node $CS_i$.
    - (i.6) Catalogue node $CS_i$ returns a response for the CSWM's request $R_i$.
- (7) The CSWM mediator collects all the catalogue node responses, wrap them in a CSWM GetRecords request and sends them back to the user.

# 5 Basic Service Requirements

This chapter specifies basic service requirements for the semantic framework, version 2.0.

## *5.1 Version Numbering*

This section specifies the way the semantic web service versioning should be dealt with. Please note that versioning here is relative to the SWS only and not to the vocabularies within. A versioning mechanism for the SWS will be very useful, as the SWS specification will evolve.

### 5.1.1 *Version Number Format*

The version number format for the semantic web service contains two positive integers, separated by a decimal point, in the form “x.y”, e.g., “2.0”.

### 5.1.2 *Version Changes*

The semantic resource server software version number shall be changed with each revision. The number shall increase monotonically and shall comprise no more than two integers separated by decimal points, with the first integer being the most significant. There may be gaps in the numerical sequence. Some numbers may denote experimental or interim versions. Service instances and their clients need not support all defined versions, but must obey the negotiation rules below.

### 5.1.3 *Version Number Negotiation*

Version number negotiation SHALL occur as follows:

1. If the server implements the requested version number, the server must send that version.
2. If the client request is for an unknown version greater than the lowest version that the server understands, the server SHALL send the highest version less than the requested version.
3. If the client request is for a version lower than any of those known to the server, then the server SHALL send the lowest version it knows.
4. If the client does not understand the new version number sent by the server, it may either cease communicating with the server or send a new request with a new version number that the client does understand, but which is less than that sent by the server (if the server had responded with a lower version).
5. If the server had responded with a higher version (because the request was for a version lower than any known to the server), and the client does not understand the proposed higher version, then the client may send a new request with a version number higher than that sent by the server.

## *5.2 General HTTP Request Rules*

### 5.2.1 *HTTP GET*

The service SHALL respond to HTTP GET KVP requests. These MAY come from client applications or web browsers.

#### 5.2.2 *HTTP POST*

The service SHALL accept requests encoded either as XML requests compliant with the XML schemas defined herein, or SOAP messages.

#### 5.2.3 *HTTPS*

HTTPS is not required for the basic semantic resources; however any editing functionality for the semantic resource SHALL be secured using HTTPS (see section 6.2 below).

### *5.3 Request Encoding*

The service SHALL accept requests encoded either as XML requests or SOAP requests.

This document defines two methods of encoding HTTP requests. The first uses XML as the encoding language, and is intended for use with the HTTP POST method. The second encoding uses keyword-value pairs (KVP) to encode the various parameters of a request and is intended for use with HTTP GET.

### *5.4 Response Encoding*

This document mandates the use of XML as an encoding for the semantic web service responses. In addition, semantic resources described within the responses SHALL be encoded in RDF/XML [BM04] and use the SKOS vocabulary where appropriate. The use of other response formats (e.g., JSON) or ontology languages (e.g., TURTLE) is optional.

### *5.5 Namespaces*

Standard namespaces SHALL be used where appropriate, i.e. SKOS and RDFS for the delivery of semantic resources.

Below is the list of namespaces to be considered and which are used in this document.

```
xml:        http://www.w3.org/XML/1998/namespace
xs:         http://www.w3.org/2001/XMLSchema#
dct:        http://purl.org/dc/terms/
rdf:        http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:       http://www.w3.org/2000/01/rdf-schema#
owl:        http://www.w3.org/2002/07/owl#
skos:       http://www.w3.org/2004/02/skos/core#
sparql:     sparql xmlns="http://www.w3.org/2005/sparql-results#
soap:       http://www.w3.org/2003/05/soap-envelope
sws:        http://cmrc.ucc.ie/sws/2.0
```

### *5.6 Simple Object Access Protocol (SOAP)*

A SOAP interface to the SWS MAY be provided. The following XML fragment illustrates a skeleton SOAP message.

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap=http://www.w3.org/2003/05/soap-envelope
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Header>
    <!--...-->
  </soap:Header>
  <soap:Body>
    <!--.....>
    <soap:Fault>
      <!--...-->
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

When a client interacts with the SWS using SOAP, it SHALL embed the SWS XML request in the <soap:Body> element in the request message.

The SWS SHALL then respond by generating a SOAP message where the response to the client's request is the content of the <soap:Body> element.

IF an exception is encountered while processing a SWS request encoded in a SOAP envelope, THEN the SWS SHALL generate a SOAP response message where the content of the <soap:Body> element is a <soap:Fault> element. Below is the skeleton XML of such a response.

```
<soap:Body>
  <soap:Fault>
    <soap:faultcode>soap:Server</soap:faultode>
    <soap:faultstring>A server exception was encountered.</soap:faultstring>
    <soap:faultactor><!--URL of SWS server--></soap:faultactor>
    <soap:detail>
      <sws:ExceptionReport>
        <!--...-->
      </sws:ExpetionReport>
    </soap:detail>
  </soap:Fault>
</soap:Body>
```

## *5.7 Exception Reporting*

Upon encountering an exception, the service SHALL generate an XML document stating that an exception has occurred and providing an intelligible traceback report of the error for debugging purposes.

The format of the XML error responses SHALL validate against the XML schema for SWS exceptions available at http://netmar.ucc.ie/xsd/sws/2.0/exceptions.xsd.

An HTML documentation for the SWS exceptions XML schema is available at the following URL: http://netmar.ucc.ie/xsd/2.0/doc/exceptions.html.

Exceptions are reported using the <ExceptionReport> element, which SHALL contain one or more exceptions, expressed using the <Exception> element. Exception messages SHALL be contained within the <ExceptionMessage> element, within an <Exception> element.

Each type of exception SHALL have a unique code, specified using the code attribute of the <Exception> element. The exception codes for this version of SWS (2.0) are defined in table 3.1.

**Table 3.1.** SWS Exception Codes

| Exception code | Description |
|---|---|
| *InternalError* | Internal error to the SWS |
| *InvalidParameterValue* | The parameter value is not valid |
| *InvalidRequest* | The request message is either invalid or is not well-formed |
| *MissingParameter* | A parameter is missing |
| *NoApplicableCode* | There is no applicable code to this exception |
| *NotImplemented* | The (abstract) operation has not been implemented |
| *NotSupported* | A service option or capability is not supported |
| *NullResourceValue* | A requested resource has a null value |
| *NullValue* | Null value exception: a required parameter or variable is null |
| *ResourceNotFound* | The requested resource does not exist or could not be found |
| *ResourceTypeMismatch* | The requested resource does not have the required type |
| *UnknownError* | The error type is unknown |

In addition to a code, an exception SHOULD have a locator that identifies the origin of the exception (e.g. the invalid or missing parameter or value, etc.). The locator is specified using the locator attribute of the <Exception> element.

The XML fragment below illustrates an exception report generated by the SWS due to an unknown concept scheme specified in a *GetConceptScheme* operation (c.f., subsection 5.3.3).

```
<sws:ExceptionReport xmlns:sws="http://cmrc.ucc.ie/sws/2.0"
 xmlns:xml=http://www.w3.org/XML/1998/namespace xml:lang="en" version="1.0">
  <sws:Exception exceptionCode="ResourceNotFound"
   locator="http://netmar.ucc.ie/ont/20110901/geoscience.owl#BadCS">
    <sws:ExceptionText>
      Resource "http://geodi.ucc.ie/ont/20110901/geoscience.owl#BadCS": No such
      concept scheme or resource.
    </sws:ExceptionText>
  </sws:Exception>
</sws:ExceptionReport>
```

# 6 Functional Requirements

This chapter specifies the operations to be supported by the NETMAR semantic web service, version 2.0. Table 6.1 defines the SWS 2.0 operations and specifies their use: mandatory (M) or optional (O). SWS operations are split into two functional interfaces: discovery interface (M), and semantic interface (M). The former provides the operations required for requesting service metadata. The latter supports the most common semantic operations required by SWS clients.

**Table 6.1.** SWS Operations

| Discovery Interface | | | |
|---|---|---|---|
| **1** | *GetCapabilities* | Retrieves service metadata, including supported operations, response formats, available concept schemes, and their supported languages. | M |
| Semantic Interface | | | |
| **2** | *GetConceptSchemes* | Lists available concept schemes with their annotations (labels, definitions, etc.). | M |
| **3** | *GetConceptScheme* | Returns a concept scheme definition given its URI. | M |
| **4** | *SearchConceptScheme* | Returns the definition(s) of one or more concept scheme(s) matching a specified free-text keyword. | M |
| **5** | *GetConceptSchemeContent* | Returns the content of a specified concept scheme identified by its URI, including its collections and concepts. | O |
| **6** | *GetCollections* | Lists available concept collections with their annotations. Collections may be filtered by one or more concept schemes. | O |
| **7** | *GetCollection* | Returns a collection definition identified by its URI. | O |
| **8** | *SearchCollection* | Returns the definition(s) of one or more collection(s) matching a specified free-text keyword. | O |
| **9** | *GetCollectionContent* | Returns the content of a given collection identified by its URI, including member collections and concepts. | O |
| **10** | *GetConcepts* | Returns the definitions of the concepts belonging to a specified concept scheme and/or collection. | M |
| **11** | *GetConcept* | Returns a concept definition given its URI. The response includes the concept's annotations. | M |
| **12** | *SearchConcept* | A smart search operation that returns the concepts matching a given keyword. | M |
| **13** | *GetRelatedConcepts* | Returns the concepts related to one or many given concept(s) using one or many given SKOS relationship(s) both from direct assertions and by entailment. | M |
| **14** | *GetExplicitTopConcepts* | Returns the concepts that have explicitly been asserted as top concepts of a specified concept scheme. | O |
| **15** | *GetImplicitTopConcepts* | Returns the concepts of one concept scheme that have no broader concepts within the latter. | M |
| **16** | *GetConceptHierarchy* | Returns the hierarchy of the concepts within a given concept scheme and/or collection. | O |
| **17** | *InterpretKeyword* | Returns the concepts that semantically match a given keyword, within a specified concept scheme and or collection. | O |
| **18** | *CheckRelation* | Checks whether two specified concepts are related via a specified SKOS relationship. | O |

This chapter specifies the operations listed in Table 6.1, and defines their request and response formats. Request formats are provided in both KVP and XML. Fragments of XML schemas are used to describe the XML structures of the SWS requests. The full XML schema for the SWS requests is available at http://netmar.ucc.ie/xsd/sws/2.0/requests.xsd. An HTML documentation for this XML schema is available at the following URL: http://netmar.ucc.ie/xsd/2.0/doc/requests.html.

## *6.1 Common SWS Request Parameters*

All SWS operation requests, except for GetCapabilities, SHALL include the following four parameters.

- Service type, this should be "SWS" for the semantic web service;
- Semantic web service version;
- Request name (e.g., GetConceptSchemes, GetConcept, etc.);
- Response format (e.g., XML, JSON, etc.)

In addition to these four common parameters, the semantic interface operations 2 to 17 share the following two parameters.

- The response language parameter, which specifies the language to be used for the lexical labels and documentation properties in the server response. If the language parameter is omitted, the server shall retrieve lexical labels and documentation in all available languages.
- The element set parameter, which specifies the amount of information (detail level) to be returned by the SWS. The element set parameter has the following five possible values, called *element set names*.
    - "***abstract***": only the resource URI SHALL be returned.
    - "***brief***": abstract information and preferred label SHALL be returned.
    - "***summary***": brief information, associated concept schemes (if applicable), and definitions SHALL be returned.
    - "***full***": summary information and alternate and hidden labels SHALL be returned.
    - "***extended***": full information and any other information (e.g., mappings) SHALL be returned.

The response detail levels for the element set names above are defined using the RDF graph templates specified in Table 6.2.

**Table 6.2.** RDF Graph Template for each Element Set Name

| Element Set Name | RDF Graph Template | Comment |
|---|---|---|
| *abstract* | ?resource rdf:type ?type . | |
| *brief* | ?resource rdf:type ?type ;<br>skos:prefLabel ?pl . | |
| *summary* | ?resource rdf:type ?type ;<br>skos:prefLabel ?pl ;<br>skos:definition ?def ;<br>skos:inScheme ?cs . | skos:inScheme only applicable to concepts and collections |
| *full* | ?resource rdf:type ?type ;<br>skos:prefLabel ?pl ;<br>skos:definition ?def ;<br>skos:inScheme ?cs ;<br>skos:altLabel ?al ;<br>skos:hiddenLabel ?hl . | skos:inScheme only applicable to concepts and collections |
| *extended* | ?resource ?predicate ?object . | |

### 6.1.1 *KVP Encoding of the Common Parameters*

In KVP encoding, the above common request parameters are encoded as specified in Table 6.3. Note that the parameter names in all KVP encodings SHALL be handled in a **case insensitive** manner.

**Table 6.3.** Common SWS Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *service* | 1 | Type of service requested<br>***Possible values:*** "SWS" | service=SWS |
| *version* | * | Service version accepted by the client<br>***Possible values:*** "2.0", "1.0" | version=2.0 |
| *request* | 1 | Operation requested by the client<br>***Possible values:*** see Table 6.1 | request=GetConcept |
| *acceptFormat* | 0..1 | Response format expected by the client<br>***Possible values:*** "text/xml", "application/json" | AcceptFormat=text/xml |
| *responseLanguage* | 0..1 | Response language<br>***Type:*** ISO 639-1 two-letter language code | responseLanguage=en |
| *elementSet* | 0..1 | Level of resource details returned by the SWS<br>***Possible values and meanings:***<br>• "abstract": *only URI SHALL be returned*<br>• "brief": *abstract information and preferred label SHALL be returned*<br>• "summary": *brief information, associated concept schemes (if applicable), and definitions SHALL be returned*<br>• "full": *summary information and alternate labels SHALL be returned*<br>• "extended": *full information and hidden labels SHALL be returned* | elementSet=full |

### 6.1.2 *XML Encoding of the Common Parameters*

In XML encoding, all SWS operation request elements, except for *GetCapabilities and CheckRelation*, extend the abstract sws:SWSRequestType which is defined by the following XML schema fragment. For the full definition of the sws:SWSRequestType type and all the SWS XML requests, see the XML schema at http://netmar.ucc.ie/xsd/sws/2.0/requests.xsd.

```xml
<!--SWSRequestType is an abstract base type for all SWS requests-->
<xs:complexType name="SWSRequestType" abstract="true">
  <xs:sequence>
    <!--Server versions accepted-->
    <!--When omitted, server shall return latest supported version.-->
    <xs:element name="AcceptVersions" type="sws:SWSVersionListType" minOccurs="0"/>

    <!--Response format accepted-->
    <!--When omitted, server shall return service metadata document using-->
    <!--the MIME type "text/xml". If the specified format is not supported,-->
    <!--the server shall raise a NotSupported exception.-->
    <xs:element name="AcceptFormat" type="sws:ResponseFormatType" minOccurs="0"/>

    <!--Element set name, optional. Default is "full"-->
    <xs:element name="ElementSet" type="sws:ElementSetNameType"/>
  </xs:sequence>

  <!--Service type, default is SWS (Semantic Web Service)-->
  <xs:attribute name="service" type="sws:ServiceType" use="required"/>

  <!--Language used for lexical labels and documentation, optional.-->
  <!--If omitted, server shall retrieve lexical labels and documentation-->
  <!--in all available languages.-->
  <!--If specified language is not supported, server shall return empty-->
  <!--lexical labels and documentation.-->
  <xs:attribute name="responseLanguage"" type="sws:LanguageType" use="optional"/>
</xs:complexType>
```

The sws:AcceptVersionsType, sws:AcceptFormatsType, and sws:ServiceType, and sws:ElementSetNameType types are defined in the SWS requests XML schemas.

## 6.2 *GetCapabilities Operation*

The semantic web service SHALL have the ability to describe its capabilities by returning service metadata in response to a *GetCapabilities* request. The GetCapabilities operation is more completely specified in Table 6.3. Its parameters are specified in Table 6.4.

**Table 6.4.** Definition of the *GetCapabilities* Operation

| | |
|---|---|
| **Definition** | Allows clients to retrieve service metadata |
| **Receives** | Optional capabilities document section names |
| **Returns** | Service capabilities document including the sections specified in the request |
| **Exceptions** | InvalidParameterValue, InternalError |

### 6.2.1 *KVP Encoding*

The *GetCapabilities* request uses the *section* parameter that specifies the capabilities document sections requested by the client  (c.f., Table 6.5).

**Table 6.5.** *GetCapabilities* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *section* | 0..4 | Capabilities document section requested<br>***Possible values:***<br>• “ServerIdentification”<br>• “ServiceProvider”<br>• “OperationsMetadata”<br>• “SupportedConceptSchemeList” | section=ServiceProvider<br>&section=OperationsMetadata |

### 6.2.2 *XML Encoding*

The <sws:GetCapabilities> element is used to request a capabilities document from a semantic web service; it is defined by the following XML Schema fragment.

```
<!--GetCapabilities request-->
<xs:element name="GetCapabilities" type="sws:GetCapabilitiesType"/>

<!--GetCapabilitiesType-->
<xs:complexType name="GetCapabilitiesType">
  <xs:sequence>
    <!--Capabilities sections-->
    <!--When omitted or not supported by server, server shall return-->
    <!--complete service metadata (Capabilities) document.-->
    <xs:element name="Sections" type="sws:CapabilitiesSectionListType"
      minOccurs="0"/>
  </xs:sequence>
  <!--Service type, default is SWS (Semantic Web Service)-->
  <xs:attribute name="service" type="sws:ServiceType" use="required"/>
</xs:complexType>
```

The optional Sections element of type sws:CapabilitiesSectionListType specifies the capabilities sections requested by the client (c.f., next subsection).

### 6.2.3 *Response*

The root element of the response to a *GetCapabilities* request is the <sws:GetCapabilitiesResponse> element, which is partially defined in the following XML Schema fragment.

```
<!--GetCapabilitiesResponse-->
<xs:element name="GetCapabilitiesResponse" type="sws:GetCapabilitiesResponseType"/>

<!--GetCapabilitiesResponseType-->
<xs:complexType name="GetCapabilitiesResponseType">
  <xs:sequence>

    <!--Server identification information section.-->
    <!--Contains service type and service type version.-->
    <xs:element ref="sws:ServiceIdentification" minOccurs="0"/>

    <!--Service provider section-->
    <!--Conatins provider name, provider site, and contact information-->
    <xs:element ref="sws:ServiceProvider" minOccurs="0"/>

    <!--Supported operations section-->
    <!--Contains information about the operations supported by the SWS.-->
    <xs:element ref="sws:OperationsMetadata" minOccurs="0"/>

    <!--Concept schemes section-->
    <!--List of concept schemes delivered by the SWS-->
    <xs:element name="SupportedConceptSchemes" type="sws:ConceptSchemeListType"
      minOccurs="0"/>

    <!--Service version ,required-->
    <xs:attribute name="version" type="sws:SWSVersionType" use="required"/>
  </xs:sequence>
</xs:complexType>
```

The capabilities response document contains the following sections:

1. **Server Identification Section –** Provides information about the SWS itself;
2. **Service Provider Section –** Provides metadata about the organisation operating the semantic web service;
3. **Operations Metadata Section –** Provides the list of SWS operations implemented by the SWS instance;
4. **Supported Concept Schemes Section –** Provides the list of concept schemes delivered by the SWS, including their SKOS lexical labels and annotations.

## *6.3 GetConceptSchemes Operation*

The mandatory GetConceptSchemes operation is used to retrieve information about the available concept schemes. This operation is more completely specified in Table 6.6.

**Table 6.6.** Definition of the *GetConceptSchemes* Operation

| | |
|---|---|
| **Definition** | Allows clients to describe the available concept schemes delivered by the SWS |
| **Receives** | Optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | List of available concept schemes with their annotation properties, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, InternalError |

### 6.3.1 *KVP Encoding*

The *GetConceptSchemes* operation does not require more parameters than the ones defined in Table 6.3. An example of a *GetConceptSchemes* request encoded in KVP is shown blow.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptSchemes
&elementSet=full
&responseLanguage=en
```

### 6.3.2 *XML Encoding*

The following XML Schema fragment defines the XML encoding of the *GetConceptSchemes* operation.

```
<!--GetConceptSchemes-->
<xs:element name="GetConceptSchemes" type="sws:GetConceptSchemesType"/>

<!--GetConceptSchemesType-->
<xs:complexType name="GetConceptSchemesType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType"/>
  </xs:complexContent>
</xs:complexType>
```

### 6.3.3 *Response*

The *GetConceptSchemes* response consists is an RDF document containing the list of concepts schemes delivered by the SWS.

The XML below shows a fragment of the *GetConceptSchemes* response associated with the request example above. For sake of readability, namespace declarations and concept schemes URI namaespaces have been omitted.

```
<rdf:RDF>
  <skos:ConceptScheme rdf:about="Themes">
    <skos:prefLabel xml:lang="en">Themes</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining theme keywords
    </skos:definition>
  </skos:ConceptScheme>
  <skos:ConceptScheme rdf:about="Parameters">
    <skos:prefLabel xml:lang="en">Parameters</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining parameter keywords
    </skos:definition>
  </skos:ConceptScheme>
  <!--Other concept schemes-->
</rdf:RDF>
```

## 6.4 *GetConceptScheme Operation*

The *GetConceptScheme* operation, which is fully defined in Table 6.7, allows a client to retrieve information about a specified concept scheme identified by its URI.

**Table 6.7.** Definition of the *GetConceptScheme* Operation

| | |
|---|---|
| **Definition** | Allows clients to describe a concept scheme specified by its URI |
| **Receives** | The URI of a concept scheme of interest, and optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | Definition of the concept scheme specified in the request, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.4.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptScheme* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.8).

**Table 6.8.** *GetConceptScheme* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | 1 | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

An example of a *GetConceptScheme* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptScheme
&elementSet=full
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

### 6.4.2 *XML Encoding*

The <sws:GetConceptScheme> element is used to call the *GetConceptScheme* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConceptScheme request-->
<xs:element name="GetConceptScheme" type="sws:GetConceptSchemeType"/>

<!--GetConceptSchemeType-->
<xs:complexType name="GetConceptSchemeType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.4.3 *Response*

The *GetConceptScheme* response is an RDF document describing the requested concept scheme. The XML below shows a fragment of the *GetConceptScheme* response associated with the request example above.

```
<rdf:RDF>
  <skos:ConceptScheme rdf:about="Themes">
    <skos:prefLabel xml:lang="en">Themes</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining theme keywords
    </skos:definition>
  </skos:ConceptScheme>
</rdf:RDF>
```

## *6.5 SearchConceptScheme Operation*

The optional *SearchConceptScheme* operation, which is fully defined in Table 6.9, allows a client to retrieve information about one or more concept scheme(s) matching a free-text keyword.

**Table 6.9.** Definition of the *SearchConceptScheme* Operation

| | |
|---|---|
| **Definition** | Allows clients to search one or more concept schemes by keyword. |
| **Receives** | A free-text keyword, and optionally, the keyword's language, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | Definition of the concept schemes matching the specified keyword in the specified language, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered. |
| **Exceptions** | InvalidParameterValue, MissingParameter, InternalError |

### 6.5.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *SearchConceptScheme* request uses the *keyword* parameter and its language (*keywordLanguage*), c.f., Table 6.8.

**Table 6.10.** *SearchConceptScheme* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *keyword* | 1 | Search keyword<br>***Type:*** free text | keyword=Theme |
| *keywordLanguag e* | 0..1 | Keyword language<br>***Type:*** ISO 639-1 two-letter language code | keywordLanguage=en |

An example of a *SearchConceptScheme* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=SearchConceptScheme
&elementSet=full
&responseLanguage=en
&keyword=Theme
&keywordLanguage=en
```

### 6.5.2 *XML Encoding*

The <sws:SearchConceptScheme> element is used to call the *SearchConceptScheme* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--SearchConceptScheme request-->
<xs:element name="SearchConceptScheme" type="sws:SearchConceptSchemeType"/>

<!--SearchConceptSchemeType-->
<xs:complexType name="SearchConceptSchemeType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, mandatory-->
        <xs:element name="Keyword" type="sws:AnnotationType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:AnnotationType is a data type that contains a free-text annotation (keyword) and a language attribute (sws:language).

### 6.5.3 *Response*

The *SearchConceptScheme* response consists of an RDF document describing the concept scheme(s) matching the keyword specified in the request. The XML below shows a fragment of the *SearchConceptScheme* response associated with the request example above.

```
<rdf:RDF>
  <skos:ConceptScheme rdf:about="Themes">
    <skos:prefLabel xml:lang="en">Themes</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining theme keywords
    </skos:definition>
  </skos:ConceptScheme>
</rdf:RDF>
```

## 6.6 *GetConceptSchemeContent Operation*

The optional *GetConceptSchemeContent* operation, which is fully defined in Table 6.11, allows a client to retrieve the content of a specified concept scheme identified by its URI, including its collections and concepts.

**Table 6.11.** Definition of the *GetConcetpSchemeContent* Operation

| | |
|---|---|
| **Definition** | Allows clients to retrieve the content (collections and concepts) of a specified concept scheme. |
| **Receives** | The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | List of collections and concepts belonging to the specified concept scheme, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.6.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptSchemeContent* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.12).

**Table 6.12.** *GetConceptSchemeContent* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | 1 | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

An example of a *GetConceptSchemeContent* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptSchemeContent
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
```

### 6.6.2 *XML Encoding*

The <sws:GetConceptSchemeContent> element is used to call the *GetConceptSchemeContent* operation of an SWS. It is defined by the following XML Schema fragment.

```xml
<!--GetConceptSchemeContent request-->
<xs:element name="GetConceptSchemeContent" type="sws:GetConceptSchemeContentType"/>

<!--GetConceptSchemeContentType-->
<xs:complexType name="GetConceptSchemeContentType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.6.3 *Response*

The *GetConceptSchemeContent* response is an RDF document listing and describing the concepts and collections belonging to the requested concept scheme. The XML below shows a fragment of the *GetConceptSchemeContent* response associated with the request example above.

```xml
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
  </skos: Collection>
  <skos:Collection rdf:about="InSituLaboratoryInstruments">
    <skos:prefLabel xml:lang="en">In Situ/Laboratory
      Instruments</skos:prefLabel>
  </skos:Collection>
  <!--More collections-->
  <skos:Concept rdf:about="SidescanSonar">
    <skos:prefLabel xml:lang="en">Sidescan Sonar</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

## 6.7 *GetCollections Operation*

The optional *GetCollections* operation, which is fully defined in Table 6.13, allows a client to retrieve the list of collections available, with the possibility of filtering by concept schemes.

**Table 6.13.** Definition of the *GetConcetpSchemeContent* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the list of available collections, possibly within one or more specified concept schemes. |
| **Receives** | Optionally, the URIs of the concept schemes of interest and, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | List of collections belonging to the specified concept schemes, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.7.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptSchemeContent* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme of interest (c.f., Table 6.14).

**Table 6.14.** *GetCollections* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | * | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments |

An example of a *GetCollections* request is provided below.

| <Service URL>? |
|---|
| service=SWS |
| &version=2.0 |
| &request=GetCollections |
| &elementSet=brief |
| &responseLanguage=en |
| &conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments |

### 6.7.2 *XML Encoding*

The <sws:GetCollections> element is used to call the *GetCollections* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetCollections request-->
<xs:element name="GetCollections" type="sws:GetCollectionsType"/>

<!--GetCollectionsType-->
<xs:complexType name="GetCollectionsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.7.3 *Response*

The *GetCollections* response is an RDF document describing the available collections belonging to the specified concept scheme. The XML below shows a fragment of the *GetCollections* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
  </skos: Collection>
  <skos:Collection rdf:about="InSituLaboratoryInstruments">
    <skos:prefLabel xml:lang="en">In Situ/Laboratory
      Instruments</skos:prefLabel>
  </skos:Collection>
  <!--More collections-->
</rdf:RDF>
```

## 6.8 *GetCollection Operation*

The *GetCollection* operation, which is fully defined in Table 6.15, allows a client to retrieve information about a specified concept collection identified by its URI.

**Table 6.15.** Definition of the *GetCollection* Operation

| | |
|---|---|
| **Definition** | Allows clients to describe a collection specified by its URI |
| **Receives** | The URI of a concept collection of interest, and optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | Definition of the concept collection specified in the request, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.8.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetCollection* request uses the *collection* parameter that specifies the URI of the concept collection requested by the client (c.f., Table 6.16).

**Table 6.16.** *GetCollection* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *collection* | 1 | URI of the concept collection requested<br>***Type:*** URI | collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments |

An example of a *GetCollection* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetCollection
&elementSet=full
&responseLanguage=en
&collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments
```

### 6.8.2 *XML Encoding*

The <sws:GetCollection> element is used to call the *GetCollection* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetCollection request-->
<xs:element name="GetCollection" type="sws:GetCollectionType"/>

<!--GetCollectionType-->
<xs:complexType name="GetCollectionType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="Collection" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.8.3 *Response*

The *GetCollection* response is an RDF document describing the requested collection. The XML below shows a fragment of the *GetCollection* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
    <skos:definition xml:lang="en">
      Instruments that are deployed on or from a platform and that allow the
      collection of data about the earth's land or water areas
    </skos:definition>
  </skos:Collection>
</rdf:RDF>
```

## *6.9 SearchCollection Operation*

The optional *SearchCollection* operation, which is fully defined in Table 6.17, allows a client to retrieve information about one or more collection(s) matching a free-text keyword.

**Table 6.17.** Definition of the *SearchCollection* Operation

| | |
|---|---|
| **Definition** | Allows a client to search one or more concept collections by keyword. |
| **Receives** | A free-text keyword, and optionally, the keyword's language, the URIs of the concept schemes of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language. |
| **Returns** | Definition of the collections matching the specified keyword in the specified language, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError. |

### 6.9.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *SearchCollection* request uses the *keyword* parameter, its language (*keywordLanguage*), and the conceptScheme parametr which specifies the URI of a concept scheme of interest, c.f., Table 6.18.

**Table 6.18.** *SearchCollection* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *keyword* | 1 | Search keyword<br>***Type:*** free text | keyword=remote sensing |
| *keywordLanguage* | 0..1 | Keyword language<br>***Type:*** ISO 639-1 two-letter language code | keywordLanguage=en |
| *conceptScheme* | * | URI of a concept scheme of interest<br>***Type:*** URI | conceptScheme= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments |

An example of a *SearchCollection* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=SearchCollection
&elementSet=full
&responseLanguage=en
&keyword=remote sensing
&keywordLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
```

### 6.9.2 *XML Encoding*

The <sws:SearchCollection> element is used to call the *SearchCollection* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--SearchCollection request-->
<xs:element name="SearchCollection" type="sws:SearchCollectionType"/>

<!--SearchCollectionType-->
<xs:complexType name="SearchCollectionType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>

        <!--URI of the requested concept schemes-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

### 6.9.3 *Response*

The *SearchCollection* response is an RDF document describing the collection(s) matching the keyword specified in the request and belonging to the concept schemes of interest. The XML below shows a fragment of the *SearchCollection* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
    <skos:definition xml:lang="en">
      Instruments that are deployed on or from a platform and that allow the
      collection of data about the earth's land or water areas
    </skos:definition>
  </skos:Collection>
</rdf:RDF>
```

## 6.10 GetCollectionContent Operation

The optional *GetCollectionContent* operation, which is fully defined in Table 6.19, allows a client to retrieve the content of a specified collection identified by its URI, including its member collections and concepts. The content may be filtered by concept schemes.

**Table 6.19.** Definition of the *GetCollectionContent* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the content (member collections and concepts) of a specified collection. |
| **Receives** | The URI of the collection of interest and, optionally, the URIs of the concept schemes of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | List of collections and concepts belonging to the specified collection and concept schemes, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.10.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetCollectionContent* request uses the collection parameter that specifies the URI of the requested collection, and the *conceptScheme* parameter that specifies the URI of a concept scheme to which the returned resources must belong (c.f., Table 6.20).

**Table 6.20.** *GetCollectionContent* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *collection* | 1 | URI of the collection requested<br>***Type:*** URI | collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments |
| *conceptScheme* | * | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments |

An example of a *GetCollectionContent* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetCollectionContent
&elementSet=brief
```

```
&responseLanguage=en
&collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
```

### 6.10.2 *XML Encoding*

The <sws:GetCollectionContent> element is used to call the *GetCollectionContent* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetCollectionContent request-->
<xs:element name="GetCollectionContent" type="sws:GetCollectionContentType"/>

<!--GetCollectionContentType-->
<xs:complexType name="GetCollectionContentType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested collection, mandatory-->
        <xs:element name="Collection" type="xs:anyURI"/>

        <!--URIs of the requested concept schemes-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.10.3 *Response*

The *GetCollectionContent* response is an RDF document listing and describing the concepts and collections belonging to the requested concept scheme. The XML below shows a fragment of the *GetCollectionContent* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="ActiveRemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">
      Active Remote Sensing Instruments
    </skos:prefLabel>
  </skos: Collection>
  <skos:Collection rdf:about="PassiveRemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">
      Passive Remote Sensing Instruments
    </skos:prefLabel>
  </skos:Collection>
  <!--More collections-->
</rdf:RDF>
```

## *6.11 GetConcepts Operation*

The mandatory *GetConcepts* operation, which is fully defined in Table 6.21, retrieves all the concepts of one concept scheme with possible filtering by collections.

**Table 6.21.** Definition of the *GetCollectionContent* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the concepts of a specified concept scheme with possible filtering by collections. |
| **Receives** | The URI of the concept scheme of interest, and optionally, the URIs of the collections of |

| | |
|---|---|
| | interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language. |
| **Returns** | List of concepts belonging to the specified concept scheme and collections, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.11.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConcepts* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme of interest, the collection parameter that specifies the URI of the collections to which the returned concepts must belong (c.f., Table 6.20).

**Table 6.22.** *GetConcepts* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | 1 | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments |
| *collection* | * | URI of a collection requested<br>***Type:*** URI | collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments |

An example of a *GetConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConcepts
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
&collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#ActiveRemoteSensingInstruments
```

### 6.11.2 *XML Encoding*

The <sws:GetConcepts> element is used to call the *GetConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```xml
<!--GetConcepts request-->
<xs:element name="GetConcepts" type="sws:GetConceptsType"/>

<!--GetConceptsType-->
<xs:complexType name="GetConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>

        <!--URIs of the collections of interest-->
        <xs:element name="Collection" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.11.3 *Response*

The *GetConcepts* response is an RDF document listing and describing the concepts belonging to the requested concept scheme and collections (if specified). The XML below shows a fragment of the *GetConcepts* response associated with the request example above.

```xml
<rdf:RDF>
  <skos:Concept rdf:about="SidescanSonar">
    <skos:prefLabel xml:lang="en">Sidescan Sonar</skos:prefLabel>
  </skos:Concept>
  <skos:Concept rdf:about="LiDAR">
    <skos:prefLabel xml:lang="en">LiDAR</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

## *6.12 GetConcept Operation*

The *GetConcept* operation, which is fully defined in Table 6.23, allows a client to retrieve information about a specified concept collection identified by its URI.

**Table 6.23.** Definition of the *GetConcept* Operation

| | |
|---|---|
| **Definition** | Allows clients to describe a Concept specified by its URI |
| **Receives** | The URI of a concept of interest, and optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | Definition of the concept specified in the request, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.12.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConcept* request uses the *concept* parameter that specifies the URI of the concept requested by the client (c.f., Table 6.24).

**Table 6.24.** *GetConcept* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *concept* | 1 | URI of the concept requested<br>***Type:*** URI | concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology |

An example of a *GetConcept* request is provided below.

| |
|---|
| <Service URL>? |
| service=SWS |
| &version=2.0 |
| &request=GetCollection |
| &elementSet=full |
| &responseLanguage=en |
| &concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology |

### 6.12.2 *XML Encoding*

The <sws:GetConcept> element is used to call the *GetConcept* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConcept request-->
<xs:element name="GetConcept" type="sws:GetConceptType"/>

<!--GetConceptType-->
<xs:complexType name="GetConceptType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept, mandatory-->
        <xs:element name="Concept" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.12.3 *Response*

The *GetConcept* response is an RDF document describing the requested collection. The XML below shows a fragment of the *GetConcept* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
    <skos:inScheme rdf:about="Themes"/>
    <skos:inScheme rdf:about="Disciplines"/>
    <skos:inScheme rdf:about="INSPIREThemes_Annex2"/>
    <skos:definition xml:lang="en">
      Geology characterised according to composition and structure.
      Includes bedrock, aquifers and geomorphology.
    </skos:definition>
  </skos:Concept>
</rdf:RDF>
```

## 6.13 *SearchConcept Operation*

The *SearchConcept* operation, which is fully defined in Table 6.25, allows a client to retrieve information about one or more concept(s) matching a free-text keyword.

**Table 6.25.** Definition of the *SearchConcept* Operation

| | |
|---|---|
| **Definition** | Allows a client to search one or more concept collections by keyword. |
| **Receives** | A free-text keyword, and optionally, the keyword's language, the URI of a concept scheme of interest, the URIs of the collections of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language. |
| **Returns** | Definition of the concepts matching the specified keyword in the specified language, and belonging to the specified concept scheme and/or collections, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError. |

### 6.13.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *SearchConcept* request uses the *keyword* parameter, and optionally its language (*keywordLanguage*), the conceptScheme parameter that specifies the URI of the concept scheme of interest, and the collection parameter that specifies the URI of a collection of interest, c.f., Table 6.26.

**Table 6.26.** *SearchConcept* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *keyword* | 1 | Search keyword<br>***Type:*** free text | keyword=geology |
| *keywordLanguage* | 0..1 | keyword language<br>***Type:*** ISO 639-1 two-letter language code | keywordLanguage=en |
| *conceptScheme* | 0..1 | URI of a concept scheme of interest<br>***Type:*** URI | conceptScheme= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |
| *collection* | * | URI of a collection of interest<br>***Type:*** URI | collection= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

An example of a *SearchConcept* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=SearchConcept
&elementSet=full
&responseLanguage=en
&keyword=geo
&keywordLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Parameters
```

### 6.13.2 *XML Encoding*

The <sws:SearchConcept> element is used to call the *SearchConcept* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--SearchConcept request-->
<xs:element name="SearchConcept" type="sws:SearchConceptType"/>

<!--SearchConceptType-->
<xs:complexType name="SearchConceptType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>

        <!--URI of the requested concept scheme-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"/>

        <!--URI of the requested concept schemes-->
        <xs:element name="Collection" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

### 6.13.3 *Response*

The *SearchConcept* response is an RDF document describing the concept(s) matching the keyword specified in the request and belonging to the concept scheme and/or collections of interest. The XML below shows a fragment of the *SearchConcept* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="GeotechnicalPropertyOfSeabedSamples">
    <skos:prefLabel xml:lang="en">
      Geotechnical Property of Seabed Samples
    </skos:prefLabel>
    <skos:inScheme rdf:about="Parameters"/>
    <skos:definition xml:lang="en">
      Parameter pertaining to the geotechnical analysis of seabed samples
    </skos:definition>
  </skos:Concept>
</rdf:RDF>
```

## *6.14 GetRelatedConcepts Operation*

The mandatory *GetRelatedConcepts* operation, which is fully defined in Table 6.27, retrieves all the concepts related to one or more specified concepts, identified by their URIs, by one or more SKOS relationships. Concepts may be filtered by concept scheme and/or collections.

**Table 6.27.** Definition of the *GetRelatedConcepts* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the concepts related to one or more concept schemes. |
| **Receives** | A list of concepts, and optionally, a list of SKOS relationship URIs, the URIs of the concepts schemes and collections of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language. |
| **Returns** | Definitions of the concepts related to the specified ones using the specified relationships and belonging to the specified concept scheme and collections, in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError. |

### 6.14.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetRelatedConcepts* request uses the parameters defined in Table 6.28.

**Table 6.28** *GetRelatedConcepts* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *concept* | 1..* | URI of the concept of interest | concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology |
| *relationship* | * | URI or local name of a SKOS relationship | relationship=http://www.w3.org/2004/02/skos/core#narrowerTransitive |
| *conceptScheme* | * | URI of a concept scheme<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Disciplines |
| *collection* | * | URI of a collection<br>***Type:*** URI | collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments |

An example of a *GetRelatedConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetRelatedConcepts
&elementSet=full
&responseLanguage=en
&concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
&relationship=http://www.w3.org/2004/02/skos/core#narrowerTransitive
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

### 6.14.2 *XML Encoding*

The <sws:GetRelatedConcepts> element is used to call the *GetRelatedConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetRelatedConcepts-->
<xs:element name="GetRelatedConcepts" type="sws:GetRelatedConceptsType"/>

<!--GetRelatedConceptsType-->
<xs:complexType name="GetRelatedConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept, mandatory-->
        <xs:element name="Concept" type="xs:anyURI" maxOccurs="unbounded"/>

        <!--SKOS Relationship, optional, if omitted than the SKOS relationship-->
        <!--skos:semanticRelation SHALL be considered.-->
        <xs:element ref="sws:SKOSRelationship" minOccurs="0"
          maxOccurs="unbounded"/>

        <!--URIs of target concept schemes; optional. If omitted then the-->
        <!--related concepts search is performed in all concept schemes.-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>

        <!--URIs of target collections; optional. If omitted then the-->
        <!--related concepts search is performed in all collections.-->
        <xs:element name="Collection" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.14.3 *Response*

The *GetRelatedConcepts* response is an RDF document listing the concepts related to the requested concepts using the specified relationships and belonging to the specified concept schemes and or collections. The XML below shows a fragment of the *GetRelatedConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="AcousticSeabedClassification">
    <skos:prefLabel xml:lang="en">
      Acoustic Seabed Classification
    </skos:prefLabel>
    <skos:inScheme rdf:about="Themes"/>
    <skos:definition xml:lang="en">
      The classification of seabed based on acoustic properties. This can
      be divided into two main categories: seabed surface classification and
      seabed sub-surface classification (sub bottom profiling).
    </skos:definition>
  </skos:Concept>
  <skos:Concept rdf:about="FaciesInterpretation">
    <skos:prefLabel xml:lang="en">Facies interpretation</skos:prefLabel>
    <skos:inScheme rdf:about="Themes"/>
    <skos:definition xml:lang="en">
      The characterisation of a rock or series of rocks reflecting their
      appearance, composition, and conditions of formation
    </skos:definition>
  </skos:Concept>
  <!--Other related concepts-->
</rdf:RDF>
```

## 6.15 *GetExplicitTopConcepts Operation*

The optional *GetExplicitTopConcepts* operation, which is fully defined in Table 6.29, retrieves all the concepts asserted as being the top concepts of a specified concept scheme identified by its URI.

**Table 6.29.** Definition of the *GetExplicitTopConcepts* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the top concepts of a specified concept scheme. |
| **Receives** | The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | List of concepts asserted as top concepts of the specified concept scheme, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.15.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetExplicitTopConcepts* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.30).

**Table 6.30.** *GetExplicitTopConcepts* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | 1 | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

An example of a *GetExplicitTopConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetExplicitTopConcepts
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

### 6.15.2 *XML Encoding*

The <sws:GetExplicitTopConcepts> element is used to call the *GetExplicitTopConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetExplicitTopConcepts request-->
<xs:element name="GetExplicitTopConcepts" type="sws:GetExplicitTopConceptsType"/>

<!--GetExplicitTopConceptsType-->
<xs:complexType name="GetExplicitTopConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.15.3 *Response*

The *GetExplicitTopConcepts* response is an RDF document listing and describing the explicitly asserted top concepts of the specified concept scheme. The XML below shows a fragment of the *GetExplicitTopConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
  </skos: Concept>
  <skos:Concept rdf:about="Elevation">
    <skos:prefLabel xml:lang="en">Elevation</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

## *6.16 GetImplicitTopConcepts Operation*

The *GetImplicitTopConcepts* operation, which is fully defined in Table 6.31, retrieves all the concepts belonging to a concept scheme and having no broader concepts within that same concept scheme. This includes the explicit top concepts.

**Table 6.31.** Definition of the *GetImplicitTopConcepts* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the top concepts of a specified concept scheme. |
| **Receives** | The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | List of concepts belonging to the specified concept scheme and having no broader concepts within that concept scheme, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.16.1 *KVP Encoding*

Like the GetExplicitTopConcepts operation, the *GetImplicitTopConcepts* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 3.32).

**Table 6.32.** *GetImplicitTopConcepts* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | 1 | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

An example of a *GetImplicitTopConcepts* request is provided below.

| |
|---|
| <Service URL>? |
| service=SWS |
| &version=2.0 |
| &request=GetImplicitTopConcepts |
| &elementSet=brief |
| &responseLanguage=en |
| &conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

### 6.16.2 *XML Encoding*

The <sws:GetImplicitTopConcepts> element is used to call the *GetImplicitTopConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetImplicitTopConcepts request-->
<xs:element name="GetImplicitTopConcepts" type="sws:GetImplicitTopConceptsType"/>

<!--GetImplicitTopConceptsType-->
<xs:complexType name="GetImplicitTopConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.16.3 *Response*

The *GetImplicitTopConcepts* response is an RDF document listing and describing the implicit top concepts of the specified concept scheme. The XML below shows a fragment of the *GetImplicitTopConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
  </skos: Concept>
  <skos:Concept rdf:about="Elevation">
    <skos:prefLabel xml:lang="en">Elevation</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

## 6.17 *GetConceptHierarchy Operation*

The *GetConceptHierarchy* operation, which is fully defined in Table 6.33, retrieves all the concepts of a specified concept scheme, identified by its URI, organised in a hierarchy (nested structure) according to the SKOS broader and narrower relationships. This operation is suitable for small concept schemes (hundreds of concepts) and is useful for ontology browsers.

**Table 6.33.** Definition of the *GetConceptHierarchy* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the hierarchy of the concepts of a specified concept scheme. |
| **Receives** | The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language |
| **Returns** | The concepts of the specified concept scheme, organised in a hierarchy (nested structure) according to the SKOS broader and narrower relationships, with their annotation properties in the specified language, encoded in RDF/XML. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.17.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptHierarchy* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.34).

**Table 6.34.** *GetConceptHierarchy* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *conceptScheme* | 1 | URI of the concept scheme requested<br>***Type:*** URI | conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes |

An example of a *GetConceptHierarchy* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptHierarchy
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

### 6.17.2 *XML Encoding*

The <sws:GetConceptHierarchy> element is used to call the *GetConceptHierarchy* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConceptHierarchy request-->
<xs:element name="GetConceptHierarchy" type="sws:GetConceptHierarchyType"/>

<!--GetConceptHierarchyType-->
<xs:complexType name="GetConceptHierarchyType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 6.17.3 *Response*

The *GetConceptHierarchy* response is an RDF document containing the hierarchy of the concepts belonging to the requested concept scheme. The XML below shows a fragment of the *GetConceptHierarchy* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="NaturalRiskZones">
    <skos:prefLabel xml:lang="en">Natural risk zones</skos:prefLabel>
  </skos:Concept>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
    <skos:narrower>
      <skos:Concept rdf:about="MarineGeology">
        <skos:prefLabel xml:lang="en">Marine geology</skos:prefLabel>
        <skos:narrower>
          <skos:Concept rdf:about="MarineGeophysics">
            <skos:prefLabel xml:lang="en">Marine Geophysics</skos:prefLabel>
            <skos:narrower>
              <skos:Concept rdf:about="MarineGravityField">
                <skos:prefLabel xml:lang="en">
                  Marine Gravity Field
                </skos:prefLabel>
              </skos:Concept>
            </skos:narrower>
            <!--Other concepts narrower than "MarineGeophysics"-->
          </skos:Concept>
        </skos:narrower>
        <!--Other concepts narrower than "MarineGeology"-->
      </skos:Concept>
    </skos:narrower>
    <!--Other concepts narrower than "Geology"-->
  </skos:Concept>
  <!--Other top concepts-->
</rdf:RDF>
```

## 6.18 *InterpretKeyword Operation*

The *InterpretKeyword* operation, which is fully defined in Table 6.35, allows a client to retrieve the concept(s) that semantically match to a free-text keyword.

**Table 6.35.** Definition of the *InterpretKeyword* Operation

| | |
|---|---|
| **Definition** | Allows a client to retrieve the concepts that semantically match a free-text keyword in a given language or a concept URI. |
| **Receives** | A free-text keyword (which may be a concept URI), and optionally, the keyword's |

| | |
|---|---|
| | language, the URI of a concept scheme of interest, the URIs of the collections of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language. |
| **Returns** | Definition of the concepts matching the specified keyword in the specified language, and belonging to the specified concept scheme and/or collections, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError. |

### 6.18.1 *KVP Encoding*

In addition to the common SWS request parameters (defined in Table 6.3), the *InterpretKeyword* request uses the parameters defined in Table 6.36.

**Table 6.36.** *InterpretKeyword* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *keyword* | 1 | Search keyword<br>***Type:*** free text | keyword=marine geophysics |
| *keywordLanguage* | 0..1 | keyword language<br>***Type:*** ISO 639-1 two-letter language code | keywordLanguage=en |
| *conceptScheme* | 0..1 | URI of a concept scheme of interest<br>***Type:*** URI | conceptScheme= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Parameters |

An example of an *InterpretKeywod* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=InterpretKeyword
&elementSet=brief
&responseLanguage=en
&keyword=marine geophysics
&keywordLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Parameters
```

### 6.18.2 *XML Encoding*

The <sws:InterpretKeyword> element is used to call the *InterpretKeyword* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--InterpretKeyword request-->
<xs:element name="InterpretKeyword" type="sws:InterpretKeywordType"/>

<!--InterpretKeywordType-->
<xs:complexType name="InterpretKeywordType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, or concept URI, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>

        <!--URI of the requested concept scheme-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

### 6.18.3 *Response*

The *InterpretKeyword* response is an RDF document listing the concepts related to the specified keyword and belonging to the specified concept scheme. The XML below shows a fragment of the *InterpretKeyword* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="MarineGravityField">
    <skos:prefLabel xml:lang="en">Marine Gravity Field</skos:prefLabel>
  </skos:Concept>
  <skos:Concept
   rdf:about="MarineMagnetics">
    <skos:prefLabel xml:lang="en">Marine Magnetics</skos:prefLabel>
  </skos:Concept>
  <!--Other concepts-->
</rdf:RDF>
```

## *6.19 CheckRelation Operation*

This optional *CheckRelation* operation, which is fully defined in Table 6.37, checks whether two specified concepts are related via a specified SKOS relationship.

**Table 6.37.** Definition of the *CheckRelation* Operation

| | |
|---|---|
| **Definition** | Checks whether two specified concepts are related via a specified SKOS relationship. |
| **Receives** | The URI of an origin concept scheme (subject), that of a target concept (object), and the URI or short name of a SKOS relationship (predicate). |
| **Returns** | A SPARQL response encapsulating a Boolean value: true if the concepts are related, false else. |
| **Exceptions** | InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError |

### 6.19.1 *KVP Encoding*

In addition to the common SWS request parameters, the *GetConceptHierarchy* request uses the *subject* parameter that specifies the URI of the origin concept, the *predicate* parameter that specifies the URI or local name of the SKOS relationship, and the *object* parameter that specifies the URI of the target concept (c.f., Table 6.38).

**Table 6.38.** *CheckRelation* Request Parameters

| Parameter | Cardinality | Definition | Example |
|---|---|---|---|
| *subject* | 1 | URI of the origin concept (subject)<br>***Type:*** URI | subject=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology |
| *predicate* | 1 | URI or local name of a SKOS relationship (predicate) | predicate=http://www.w3.org/2004/02/skos/core#narrowerTransitive |
| *object* | 1 | URI of the target concept (object)<br>***Type:*** URI | object=http://netmar.ucc.ie/ont/20120801/geoscience.owl#MarineGeophysics |

An example of a *CheckRelation* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=CheckRelation
&subject=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
&predicate=http://www.w3.org/2004/02/skos/core#narrowerTransitive
&object=http://netmar.ucc.ie/ont/20120801/geoscience.owl#MarineGeophysics
```

### 6.19.2 *XML Encoding*

The <sws:CheckRelation> element is used to call the *CheckRelation* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--CheckRelation-->
<xs:element name="CheckRelation" type="sws:CheckRelationType"/>

<!--CheckRelationType-->
<xs:complexType name="CheckRelationType">
  <xs:sequence>
    <!--Server versions accepted-->
    <!--When omitted, server shall return latest supported version.-->
    <xs:element name="AcceptVersions" type="sws:SWSVersionListType" minOccurs="0"/>

    <!--Response format accepted-->
    <!--If omitted, server shall return service metadata document using-->
    <!--the MIME type "text/xml". If the specified format is not supported,-->
    <!--the server shall raise a NotSupported exception.-->
    <xs:element name="AcceptFormat" type="sws:ResponseFormatType" minOccurs="0"/>


    <!--URI of the origin concept (subject), mandatory-->
    <xs:element name="Subject" type="xs:anyURI"/>

     <!--SKOS Relationship (predicate), mandatory-->
    <xs:element name="Predicate"  type="sws:SKOSRelationshipType"/>

    <!--URI of the target concept (object), mandatory-->
    <xs:element name="Subject" type="xs:anyURI"/>
  </xs:sequence>

  <!--Service type, default is SWS (Semantic Web Service)-->
  <xs:attribute name="service" type="sws:ServiceType" use="required"/>
</xs:complexType>
```

### 6.19.3 *Response*

The *CheckRelation* response is a SPARQL response document encapsulating a Boolean value stating whether the specified concepts are related via the specified SKOS relationship. The XML below shows a fragment of the *CheckRelation* response associated with the request example above.

```
<sparql:sparql>
  <sparql:results>
    <sparql:boolean>true</sparql:boolean>
  </sparql:results>
</sparql:sparql>
```

# 7 References

[BG04] Brickley, D., and Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. 10 February 2004. Available online at: http://www.w3.org/TR/rdf-schema/.

[BM04] Beckett, D., and McBride, B.: RDF/XML Syntax Specification (Revised). W3C Recommendation. 10 February 2004. Available online at http://www.w3.org/TR/REC-rdf-syntax/.

[Co11] Cox, S.: Observations and Measurements – XML Implementation, Version 2.0. OGC Implementation. 22 March 2011. URL: http://www.opengeospatial.org/standards/om.

[CW11] Cornford, D., and Williams, M.: UncertML Best Practice Proposal. The Uncertainty Enabled Model Web, UncertWeb. March 2011.

[Do11] Domenico, B.: OGC Network Common Data Form (NetCDF) Core Encoding Standrad version 1.0. Candidate OpenGIS Encoding Standard. 05 April 2011.

[DS04] Dean, M., and Schreiber, G.: OWL Web Ontology Language: Reference. W3C Recommendation. 10 February 2004. Available online at: http://www.w3.org/TR/owl-ref/.

[HLW11] T. Hamre, Y. Lassoued, P. Walker, J. de Jesus, and M. Treguer. NETMAR Deliverable D6.2 – First Version of EUMIS Subsystems. May 2011.

[ISO03] ISO/TC211: ISO 19115:2003 – Geographic Information – Metadata. International Organization for Standardization, 2003.

[ISO05] ISO, 2005. ISO 19119:2005 Geographic information – Services.

[ISO06] ISO/TC211: ISO/PRF TS 19139 – Geographic information – Metadata – XML schema implementation. International Organization for Standardization, 2006.

[JBCW10] Jones, R., Bastin, L., Cornford, D., and Williams, M.: Handling and communicating uncertainty in chained geospatial Web Services. Spatial Accuracy, Leicester, UK July 2010.

[LBTR04] Lake, R., Burggraf, D. S., Trninic, M., and Rae, L.: Geography Mark-Up Language (GML) – Foundation for the Geo-Web. John Wiley. 2004.

[LLW10] Lassoued, Y., Leadbetter, A., and Walker, P.: Semantic Framework Strategy. November 2010.

[MM04] Manola, F., and Miller, E.: RDF Primer. W3C Recommendation. 10 February 2004. Available online at: http://www.w3.org/TR/rdf-primer/.

[MB09] Miles, A., and Bechhofer, S.: SKOS Simple Knowledge Organization System: Reference. W3C Recommendation. 18 August 2009. Available online at: http://www.w3.org/TR/2009/REC-skos-reference-20090818/.

[MH04] McGuinness, D., and Harmelen, F.: OWL Web Ontology Language: Overview. W3C Recommendation. 10 February 2004. URL: http://www.w3.org/TR/owl-features/.

[MGH09] Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation. 27 April 2009. Available online at: http://www.w3.org/TR/owl2-profiles/.

[NW05] Nebert, D., and Whiteside, A.: OGC Catalogue Services Specification. Open Geospatial Consortium Inc. 2005.

[Pe02] Percivall, George, ed.: OpenGIS Document 02-112, The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3, 2002.

[PS08] Prud'hommeaux, E., and Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation. 15 January 2008. URL: http://www.w3.org/TR/rdf-sparql-query/.

[RDF04] RDF Working Group: Resources Description Framework, World Wide Web Consortium (W3C), 10 February 2004. Available at: http://www.w3.org/RDF/.

[Sc07] Schut, P.: OpenGIS Web Processing Service, Version 1.0.0. OpenGIS Standard. 08 June 2007. URL: http://www.opengeospatial.org/standards/wps.

[SGPP10] Stasch, C., Gerharz, L., Pross, B., and Pebesma, E.: Report on Integration Requierements in UncertWeb. The Uncertainty Enabled Model Web, UncertWeb. November 2010.

[Ta82] Taylor, J. R.: An Introduction to Error Analysis. University Science Books, Mill Valley CA, 1982.

[WE08] Whiteside, A. and Evans, J.D., Web Coverage Service (WCS) Implementation Standard (Version 1.1). Open Geospatial Consortium Inc., March 2008.

# 8 Acronyms

| | |
|---|---|
| API | Application Programming Interface |
| BODC | British Oceanographic Data Centre |
| CF | Climate and Forecast |
| CMRC | Coastal and Marine Resources Centre |
| CSW | Catalogue Service for the Web |
| EO | Earth Observation |
| EUMIS | European Marine Information System |
| GML | Geography Markup Language |
| INTAMAP | Interoperability and Automated Mapping |
| ISO | International Organization for Standardization |
| MODEIS | Moderate Resolution Imaging Spectroradiometer |
| NERC | Natural Environmental Research Council |
| NetCDF | Network Common Data Form |
| NETMAR | Open service Network for Marine Environmental Data |
| NVS | NERC Vocabulary Server |
| O&M | Observations and Measurements |
| OGC | Open Geospatial Consortium |
| OWL | Web Ontology Language |
| PSGP | Projected Sequential Gaussian Processes |
| RDBMS | Relational Database Management System |
| RDF | Resource Description Framework |
| RDFS | (Also RDF-S) RDF Schema |
| ReST | (Also REST) Representational State Transfer |
| SKOS | Simple Knowledge Organisation System |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SPARQL | SPARQL Protocol and RDF Query Language *(Recursive acronym)* |
| SWS | Semantic Web Service |
| URI | Uniform Resource Identifier |
| URN | Uniform Resource Name |
| WCS | Web Coverage Service |
| WPS | Web Processing Service |
| XML | eXtensible Markup Language |