Anticipative CSW-Counting enhances the Discovery UI

Jens STUTTE[1] and Lyn GOLTZ[2]

[1]Planetek Italia s.r.l · Via Masaua 12 · I-70132 Bari
E-Mail: stutte@planetek.it
[2]lat/lon GmbH · Aennchenstraße 19 · D-53177 Bonn
E-Mail: goltz@lat-lon.de

Index

1. Abstract

In a Catalogue Search UI, the concept of "anticipative counting" of potential results for a search option helps the user to avoid time-consuming void-searches. It has been transposed from general-purpose use on the Discovery of the INSPIRE Geoportal and then integrated as new functionality into the portal's CSW metadata cache. In order to raise the re-use potential for the INSPIRE community and beyond, we decided to define an extension to the standard OGC CSW GetRecord request, rather than a proprietary ad-hoc interface. This implementation can be seen as a first shot on the way to become a recommendation of INSPIRE or even standardization through the OGC.

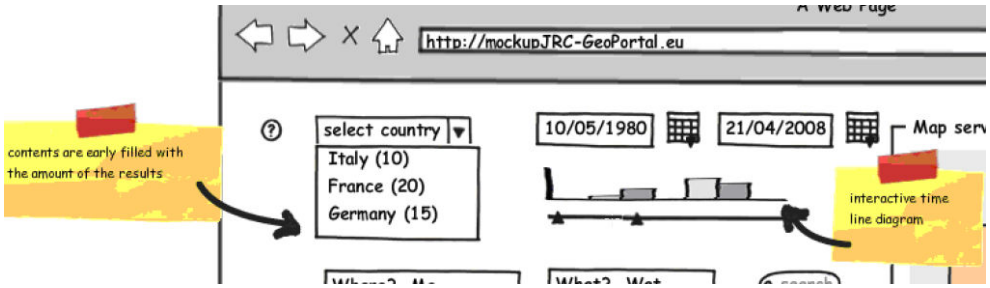
Starting from the jointly defined extension of the OGC interface protocol and a first joint proof of concept on SQL level in the database, the function has been implemented independently on portal UI side by Planetek Italia and on CSW server side by lat/lon. The choice to extend the CSW GetRecords request revealed positive not only for the re-use potential, but helped also to avoid misalignments both on UI and CSW side in the query generation and interpretation of the `<csw:Constraint>` tag in count and "normal" GetRecords queries and thus to ensure equal results.

The implementation proofed to support the user not only for the initially foreseen, simple Discovery use case, but added functionality also to other parts of the portal. In particular, the interactive timeline histogram allows the user to navigate easily the time dimension of the catalogue without risk for frustrating void-searches. The INSPIRE Geoportal UI benefits significantly from these improvements.

2. Context & the basic Idea

As part of the developments for the INSPIRE Geoportal at European Level[1], the implementing consortium has done already during proposal writing a survey of existing Geoportals and other generic websites, in order to seek for concepts, that have the potential to ease the use of a Geoportal significantly. One of those concepts is, to show the user on each search option he has an indication of how many items he can expect to satisfy this search criteria before clicking it, like most e-commerce sites do. At the time of this survey, only the French Geoportal had a similar functionality in its "Précisez votre recherche" box, but only for some selected criterias and only after having launched a first "blind" search.

At the time of proposal writing, however, the technical details of such an implementation of a counting function were not yet discussed and have been jointly elaborated during project lifetime by Planetek and lat/lon.[2] But already at that time, the potential of such a functionality to support different UI elements was understood and anticipated:



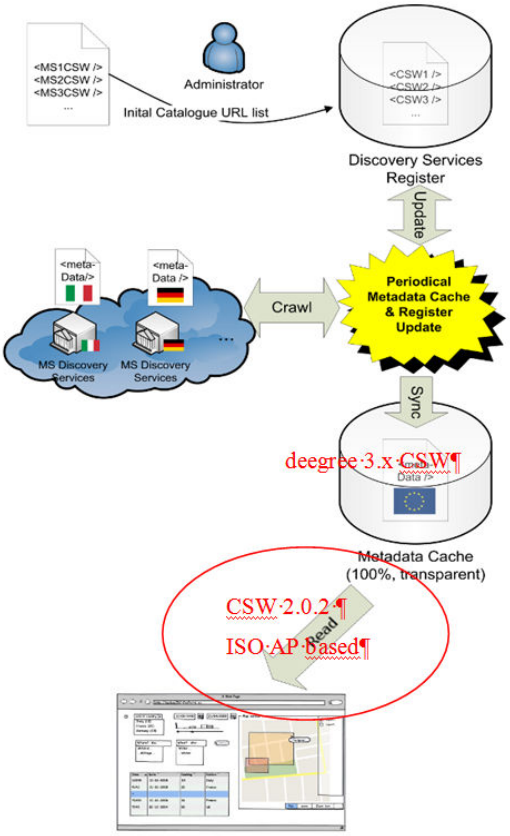
In fact, not only the selection of search criteria can be supported by an anticipative count, but the count itself gives valuable information to the user, and especially in case of dates this information can be presented in a very comprehensive form as an interactive histogram, that both displays the information for the currently selected time interval and allows the narrowing down of that interval by a simple click on the bars.

[1] We would like to thank JRC for the excellent cooperation throughout project lifetime and in particular Freddy Fierens, Ioannis Kanellopoulos and the entire involved staff from the Institute for Environment and Sustainability (IES). Thanks also for the kind permission to use some screenshots from the INSPIRE Geoportal and diagrams from the product documentation for this paper.

[2] It is worth to mention, that JRC did not set out any requirement for such functionality in the tender specifications, and also during implementation Planetek and lat/lon were independent in the establishing of the implementation details.

3. Implementation Architecture

The implementation was driven and constrained by some general architecture cornerstones of the INSPIRE Geoportal, be them intrinsically predefined by the tender requirements or explicitly set out by the architecture designed by the consortium.



One key point of the Geoportal architecture is the implementation of a "Metadata Cache", that contains metadata excerpts from all Member States' Discovery Services updated through periodic crawling and enriched with additional tags.[3] This metadata is made available through a portal internal CSW interface to the portal UI, enabling thus full CSW functionality, including spatial constraints. The local metadata cache also allows operations on the metadata database with very high performance compared to a distributed infrastructure and hence is a precondition for the implementation of the anticipative counting. The local metadata cache is a CSW with an ISO application profile implemented through a deegree 3.x SQL ISO metadata store[4]. The initial constraint of having a local cache thus led to the occasion to investigate on how to implement the counting functionality within existing OGC CSW protocols, in order to preserve the perspective of a potential use of this functionality also in different context than the INSPIRE Geoportal itself.[5]

[3] These enrichments regard multilanguage codes and additional tags from semantic keyword recognition.

[4] http://download.deegree.org/documentation/3.2.1/html/metadatastores.html#sql-iso-metadata-store

[5] In alternative we could have implemented this functionality through separate REST calls, that activate specific code operating directly on the Database, but this would have reduced the reuse potential significantly.

The package structure of the Geoportal is the following[6], highlighted are the mostly affected components:

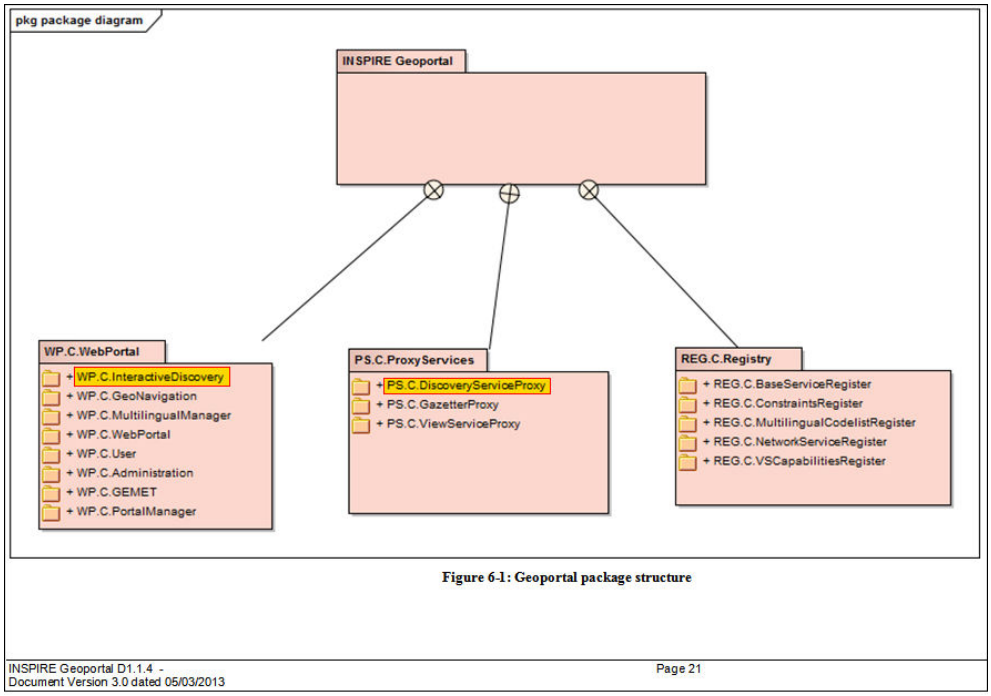


Figure 6-1: Geoportal package structure

In order to implement the anticipative count, three architectural levels had to be treated:

- The OGC interface protocol extension

- The portal and UI integration in WP.C.InteractiveDiscovery

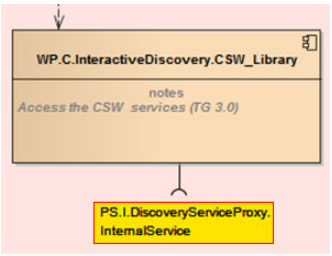
- The CSW internal and DB implementation in PS.C.DiscoveryServiceProxy

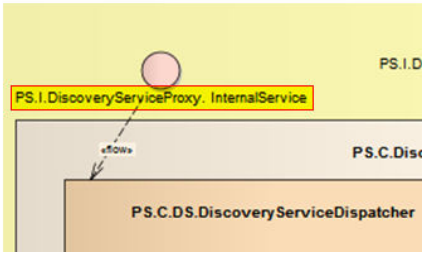
It soon emerged, that at all three levels we had to distinguish "normal" metadata properties (that is properties whose value is determined from a finite set of options) from date properties (namely Creation Date, Revision Date, Publication Date and Temporal Extent), which have potentially an infinite set of values and thus should be grouped by intervals, that have to be defined as input to the counting request.

[6] The UML diagrams are taken from pkt289-75-3.0_D1.1.4_INSPIRE_SW-HW_Architecture, a deliverable of the project not yet accessible publicly.

4. OGC Interface Protocol Extension

The interface that had to be extended was identified in the architecture as follows:





As the implementation of PS.I.DiscoveryServiceProxy.InternalService was initially meant to be specific to the INSPIRE Geoportal at European Level, we did not foresee to do substantial changes to the existing CSW 2.0.2 (ISO AP)[7], but seeked a way to express additional functionality without breaking the existing interface definition. Thus the here proposed solution cannot yet be seen as a standardization attempt, but besides the INSPIRE Geoportal specific implementation it may help as a functional proof of concept ready to be recommended within INSPIRE and for further standardisation.

The main interface idea was hence, to use a normal `<csw:GetRecords>` query with all its possibilities for the constraints definition and add a function and a new output schema.

```
<csw:GetRecords
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dct="http://purl.org/dc/terms/"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:gmd="http://www.isotc211.org/2005/gmd"
    xmlns:gco="http://www.isotc211.org/2005/gco"
    xmlns:ows="http://www.opengis.net/ows"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:geid="http://www.geoentrance.eu/InteractiveDiscovery/1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0"
    xmlns:fct="http://www.deegree.org/InteractiveDiscovery/FunctionQuery"
    xmlns:ranking="http://www.geoentrance.eu/Ranking/1.0"
    service="CSW" version="2.0.2" resultType="results"
    outputSchema="http://www.deegree.org/InteractiveDiscovery/HitOccurences">
    <fct:Query typeNames="gmd:MD_Metadata">
        <csw:Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:And>
                    <ogc:Or>
                        <ogc:And>
                            <ogc:PropertyIsGreaterThanOrEqualTo>
                                <ogc:PropertyName>apiso:CreationDate</ogc:PropertyName>
                                <ogc:Literal>1900-01-01T00:00:00</ogc:Literal>
                            </ogc:PropertyIsGreaterThanOrEqualTo>
                            <ogc:PropertyIsLessThanOrEqualTo>
                                <ogc:PropertyName>apiso:CreationDate</ogc:PropertyName>
                                <ogc:Literal>2019-12-31T23:59:59</ogc:Literal>
                            </ogc:PropertyIsLessThanOrEqualTo>
                        </ogc:And>
```

[7] http://www.opengeospatial.org/standards/specifications/catalog

Because the usual `<ows:Query>` element does not allow `<ogc:Function>` children, it was necessary to extend the `<csw:AbstractQuery>`[8] element as follows:

```
<xsd:element name="Query" type="fct:QueryType" id="Query" substitutionGroup="csw:AbstractQuery" />
<xsd:complexType name="QueryType" id="QueryType">
    <xsd:complexContent>
      <xsd:extension base="csw:AbstractQueryType">
        <xsd:sequence>
          <xsd:element ref="ogc:Function" minOccurs="1" maxOccurs="unbounded" />
          <xsd:element ref="csw:Constraint" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="typeNames" type="csw:TypeNameListType" use="required" />
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

This allows to replace the `<ows:Query>` element with the `<fct:Query>` element containing the functions for counting.[9]

Then two functions have been defined, `getOccurencesByProperties` and `getOccurencesByDateIntervals`. These functions use the standard syntax of `<ogc:Function>` elements without any extensions.

4.1. `getOccurencesByProperties` Function

The definition of this function is very simple: besides the function name itself, only 1-n property names can be given, for which the function shall retrieve the occurrences:

```
                            </ogc:PropertyIsGreaterThanOrEqualTo>
                            <ogc:PropertyIsLessThanOrEqualTo>
                                <ogc:PropertyName>apiso:TempExtent_end</ogc:PropertyName>
                                <ogc:Literal>2019-12-31T23:59:59</ogc:Literal>
                            </ogc:PropertyIsLessThanOrEqualTo>
                        </ogc:And>
                    </ogc:Or>
                    <ogc:BBOX>
                        <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                        <gml:Envelope>
                            <gml:lowerCorner>-27 34.5</gml:lowerCorner>
                            <gml:upperCorner>42 73</gml:upperCorner>
                        </gml:Envelope>
                    </ogc:BBOX>
                </ogc:And>
            </ogc:Filter>
        </csw:Constraint>
        <ogc:Function name="getOccurencesByProperties">
            <ogc:PropertyName>geid:Theme</ogc:PropertyName>
        </ogc:Function>
    </fct:Query>
</csw:GetRecords>
```

In the above example, the INSPIRE Themes are queried. As you can see, the function is put right after the standard elements of a query, in particular all queryable properties advertised

[8] Please refer to http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd.

[9] The entire schema can be downloaded here: http://schemas.lat-lon.de/jrcportal/csw-count-request.xml

by the CSW in its capabilities can be used to narrow down the search. The function responds to the GetRecords request with a custom output schema. An example response:[10]

```
<csw:GetRecordsResponse
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd http://www.deegree.org/InteractiveDiscovery/
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
  <csw:SearchStatus timestamp="2013-04-11T10:58:02.729+02:00"></csw:SearchStatus>
  <csw:SearchResults
    recordSchema="http://www.deegree.org/InteractiveDiscovery/HitOccurences"
    numberOfRecordsMatched="289"
    numberOfRecordsReturned="0"
    nextRecord="1"
    expires="2013-04-11T10:58:02.773+02:00">
    <occ:OccurencesRecord xmlns:occ="http://www.deegree.org/InteractiveDiscovery/HitOccurences">
      <occ:OccurrencesByProperties>
        <occ:PropertyOccurences>
          <ogc:PropertyName xmlns:ogc="http://www.opengis.net/ogc" xmlns:geid="http://www.geoentrance.eu/InteractiveDiscovery/1.0">geid:Theme</ogc:PropertyName>
          <occ:PropertyOccurence count="57">
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">http://inspire-registry.jrc.ec.europa.eu/registers/FCD/items/=17</ogc:Literal>
          </occ:PropertyOccurence>
          <occ:PropertyOccurence count="56">
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">http://inspire-registry.jrc.ec.europa.eu/registers/FCD/items/=5</ogc:Literal>
          </occ:PropertyOccurence>

          <!-- ... -->

          <occ:PropertyOccurence count="19">
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">http://inspire-registry.jrc.ec.europa.eu/registers/FCD/items/=13</ogc:Literal>
          </occ:PropertyOccurence>
          <occ:PropertyOccurence count="3">
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">http://inspire-registry.jrc.ec.europa.eu/registers/FCD/items/=19</ogc:Literal>
          </occ:PropertyOccurence>
          <occ:PropertyOccurence count="3">
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">http://inspire-registry.jrc.ec.europa.eu/registers/FCD/items/=16</ogc:Literal>
          </occ:PropertyOccurence>
        </occ:PropertyOccurences>
      </occ:OccurrencesByProperties>
    </occ:OccurencesRecord>
  </csw:SearchResults>
</csw:GetRecordsResponse>
```

Each `<occ:PropertyOccurences>` block starts with an `<ogc:PropertyName xmlns:ogc="…" xmlns:geid="…">geid:Theme</ogc:PropertyName>` tag, that carries the name of the property for which follow the results.
The results are a list of aggregations, such that for each value present in the DB for the property the number of metadata records is reported, that have this value:

```
<occ:PropertyOccurence count="56">
      <ogc:Literal xmlns:ogc="…">…=5</ogc:Literal>[11]
</occ:PropertyOccurence>.
```

The count attribute contains the number of results for this specific value.

At interface level, it is thus not predefined, what is the expected set of (possible) values for a given property, nor currently this set can be restricted in input. This kind of simple interface has then the advantage to return only results for existing values in the DB (no `count="0"` will ever occur) - and the disadvantage to return results for **every** value within the result set defined by the `<csw:Constraint>` tag, which might be very much (or which might contain some clutter with respect to the expected result set), depending on the nature of the property.

[10] http://schemas.lat-lon.de/jrcportal/csw-count-response.xml

[11] The INSPIRE themes are coded keywords in the Geoportal's CSW, the URL "`http://inspire-registry.jrc.ec.europa.eu/registers/FCD/items/`" indicates the namespace and the number the id in the thesaurus. The entire string should be seen in this context as the property value.

4.2. `getOccurencesByDateIntervals` Function

Date properties cannot be handled by the simple interface described above. They contain values from an almost infinite value set and should be grouped by intervals given in input to the function. Furthermore we have more than one date property (Creation Date, Revision Date, Publication Date and Temporal Extent), for which INSPIRE sets only the rule that one of those properties must be set, without being explicit, which. This made it necessary, to give the user the possibility to search for records that have (at least) one of the given date properties within a search interval. Furthermore the Geoportal's UI offers the possibility, to display the occurrences of records for a set of intervals, and in order to reduce the overhead of http calls from the UI to the CSW, the getOccurencesByDateIntervals function has been studied (and implemented) mainly for this specific use case. But the interface definition should be sufficiently flexible in order to support also different use cases. Here an example call:

```
<ogc:Function name="getOccurencesByDateIntervals">
    <ogc:PropertyName>apiso:CreationDate</ogc:PropertyName>
    <ogc:PropertyName>apiso:RevisionDate</ogc:PropertyName>
    <ogc:PropertyName>apiso:PublicationDate</ogc:PropertyName>
    <ogc:PropertyName>apiso:TempExtent</ogc:PropertyName>
    <ogc:Literal>CLOSED_DATE_INTERVAL_SERIES</ogc:Literal>
    <ogc:Literal>1900-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1910-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1920-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1930-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1940-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1950-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1960-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1970-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1980-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>1990-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>2000-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>2010-01-01T00:00:00</ogc:Literal>
    <ogc:Literal>2019-12-31T23:59:59</ogc:Literal>
</ogc:Function>
```

In this example, all four INSPIRE date properties are queried (which means, that they are put into an OR relation for the query). Then the type of interval is defined (currently only `CLOSED_DATE_INTERVAL_SERIES` has been defined and implemented) and it follows a list of thirteen date values, that implicitly define twelve intervals.

The result reads like this:

```
<csw:GetRecordsResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.ne
  <csw:SearchStatus timestamp="2013-04-11T12:11:50.139+02:00"></csw:SearchStatus>
  <csw:SearchResults recordSchema="http://www.deegree.org/InteractiveDiscovery/HitOccurences"
    numberOfRecordsMatched="289" numberOfRecordsReturned="0" nextRecord="1" expires="2013-04-11T12:11:50.350+02:00">
    <occ:OccurencesRecord xmlns:occ="http://www.deegree.org/InteractiveDiscovery/HitOccurences">
      <occ:OccurrencesByDateIntervals>
        <ogc:PropertyName xmlns:ogc="http://www.opengis.net/ogc" xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0">apiso:CreationDate</ogc:PropertyName>
        <ogc:PropertyName xmlns:ogc="http://www.opengis.net/ogc" xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0">apiso:RevisionDate</ogc:PropertyName>
        <ogc:PropertyName xmlns:ogc="http://www.opengis.net/ogc" xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0">apiso:PublicationDate</ogc:PropertyName>
        <ogc:PropertyName xmlns:ogc="http://www.opengis.net/ogc" xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0">apiso:TempExtent</ogc:PropertyName>
        <occ:DateIntervalOccurences count="0">
          <occ:LowerBoundary>
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">1900-01-01T00:00:00</ogc:Literal>
          </occ:LowerBoundary>
          <occ:UpperBoundary>
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">1910-01-01T00:00:00</ogc:Literal>
          </occ:UpperBoundary>
        </occ:DateIntervalOccurences>

        <!-- ... -->

        <occ:DateIntervalOccurences count="206">
          <occ:LowerBoundary>
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">2010-01-01T00:00:00</ogc:Literal>
          </occ:LowerBoundary>
          <occ:UpperBoundary>
            <ogc:Literal xmlns:ogc="http://www.opengis.net/ogc">2019-12-31T23:59:59</ogc:Literal>
          </occ:UpperBoundary>
        </occ:DateIntervalOccurences>
      </occ:OccurrencesByDateIntervals>
    </occ:OccurencesRecord>
  </csw:SearchResults>
</csw:GetRecordsResponse>
```

First the four <ogc:PropertyName> are repeated. Then for each date interval a block of <occ:DateIntervalOccurences count="206"> reports the number of occurrences and repeats the date interval, the count is related to. This allows to parse and understand the result without memory of the request. As you can see in the above example, for this kind of requests count="0" are reported, as well.

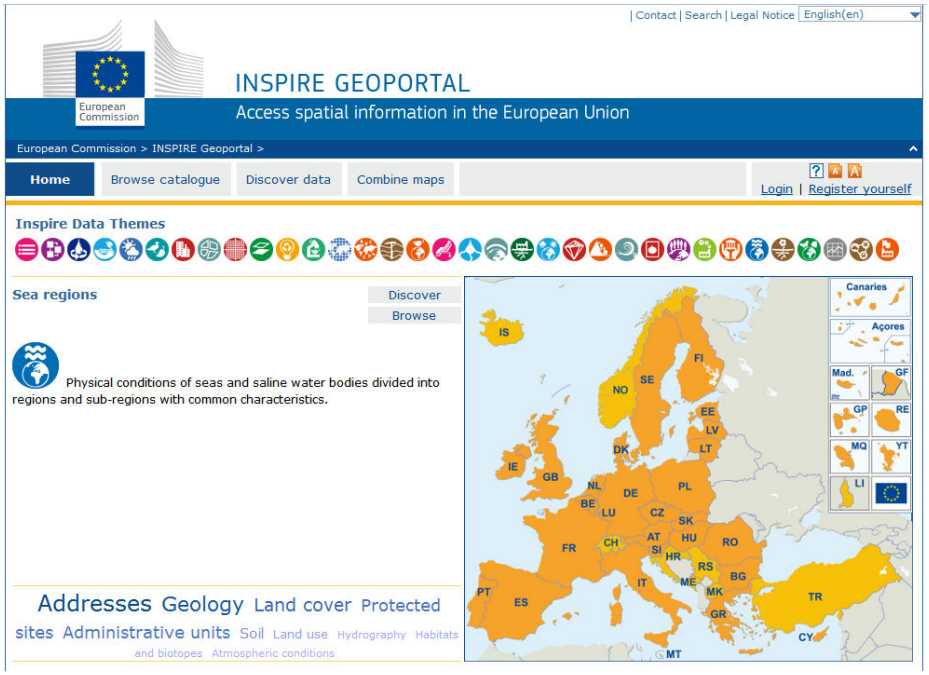
4.3. Count Interface definition outlook

It is clear, that the above interface has not yet been standardized to a point, that would allow the integration into the CSW specifications, but it has for sure the potential to serve as valuable input and proof of concept for such a standardization attempt. Furthermore only those parts of the interface have been thought from end-to-end, that are really used by the current implementation, such that a more general approach would probably lead to some adjustments also of the already implemented parts.

One important point is for example the set of properties, for which getOccurences-ByProperties can be called. The current implementation uses implicit knowledge about this set (that means: it is hardcoded in the source code both on UI and on CSW side), that a more general approach should define either as a fix set in the standards or as a new section in a getCapabilities response. It is clear, that for all INSPIRE related use cases, the currently implemented set of properties will be most likely sufficient, such that this "implicit knowledge" could be moved also to some INSPIRE Technical Guideline as a recommendation (like "…a Discovery Service, that implements the counting interface, shall support the following properties for counting…").

Another point of improvement in case of a real standardisation might be the definition of the functions itself. As we already added a schema extension for the <fct:Query> tag, one could think about a specific function definition, that avoids ambiguity in the use of the functions. But as long as these functions are seen as implementation specific, we prefer the flexibility of the chosen approach.

5. UI Integration of anticipative Counting

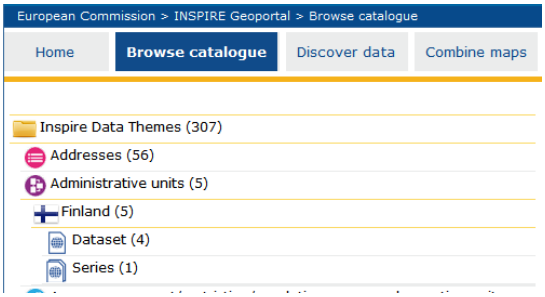


The UI of the INSPIRE Geoportal has been designed to support inexperienced users as well as skilled users in order to help them to easily discover and view the INSPIRE datasets they are interested in. Please refer to the handbook[12] for a complete overview of the functions.

The anticipative counting had been initially thought of as a help for the use of filters in the Interactive Discovery. But it turned out, that also other parts of the portal could make use of counting functions:

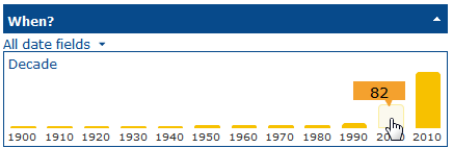
The map on the home page offers direct access to the Discovery with a predefined filter for the country the user clicked on. A mouse-over makes appear not only the name of the country, but also a hint, how many metadata records are in the catalogue for this country. Besides the immediate help in taking the decision, if it is worth to click, the number itself might give an information to a very specific kind of user.

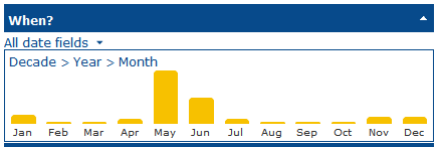
[12] pkt289-117-1.0_D2.2_User_Handbook_en.doc, at the time of writing not yet accessible online. An overlook over the Discovery functionality is given also here: http://inspire.jrc.ec.europa.eu/events/conferences/inspire_2012/presentations/137.pdf, a video from this speech can be found here: http://www.youtube.com/watch?feature=player_embedded&v=8W5RHuiFCmE#t=0s



Also the Browse Catalogue page includes a numerical information on the number of records for INSPIRE Themes, countries and resource types. Again, also these numbers can be seen as information as such.

The Interactive Discover page contains a histogram, that is initially based on decades and shows the distribution of INSPIRE data during time:





This histogram is interactive, that is a click on one of the bars restricts the search interval to the given extent. This allows for a very intuitive and fast browsing "through the time", as empty search results are efficiently avoided.

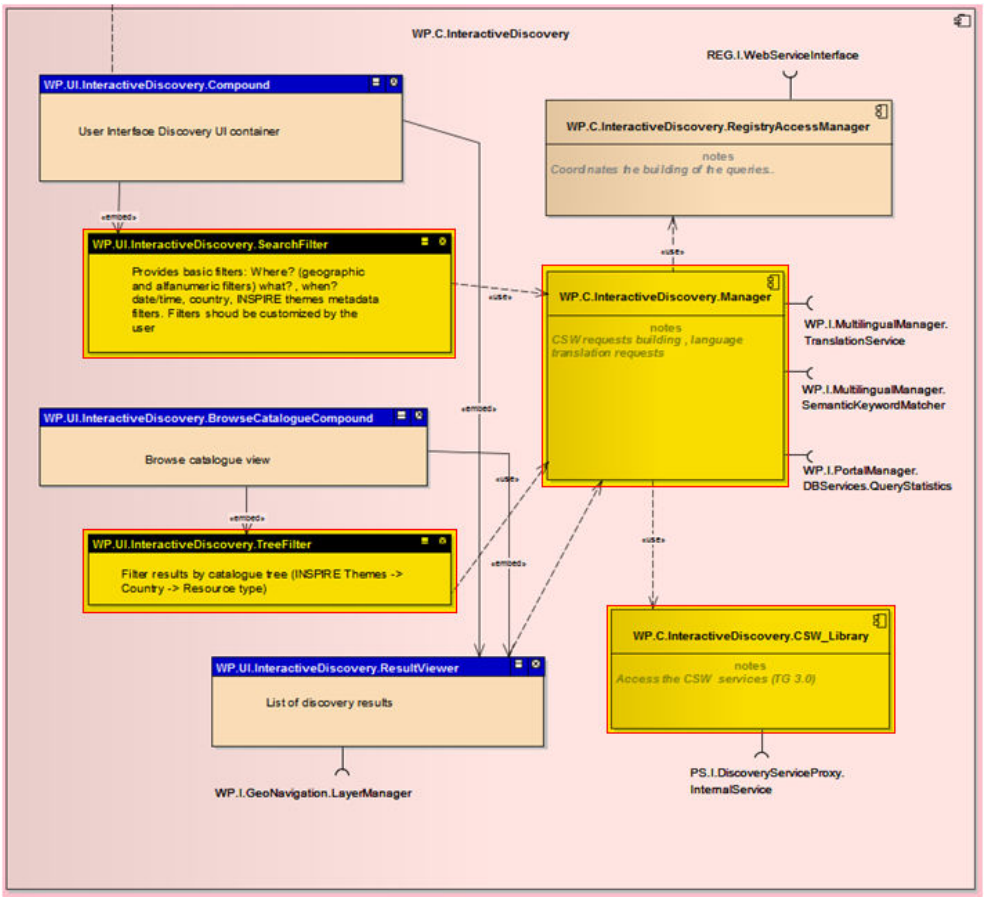
Last but not least the anticipative counting has been used obviously for the population of search filters in the Discover Data page.

5.1. Implementation notes

From the technical point of view, the integration into the UI was straightforward, as the libraries for the generation and execution of CSW XML queries had to be made, anyway. In particular, the usage of the same constraint filter XML definition as for normal catalogue queries minimized the risk of misalignments between catalogue and count results.

Some additional code had to be created for the interpretation and display of the count results XML. Here the precaution during interface definition to include all needed information into the result XML and to avoid the need for any memory of the query issued during result display eased this task significantly.

The affected components in the architecture are highlighted in the following UML diagram:

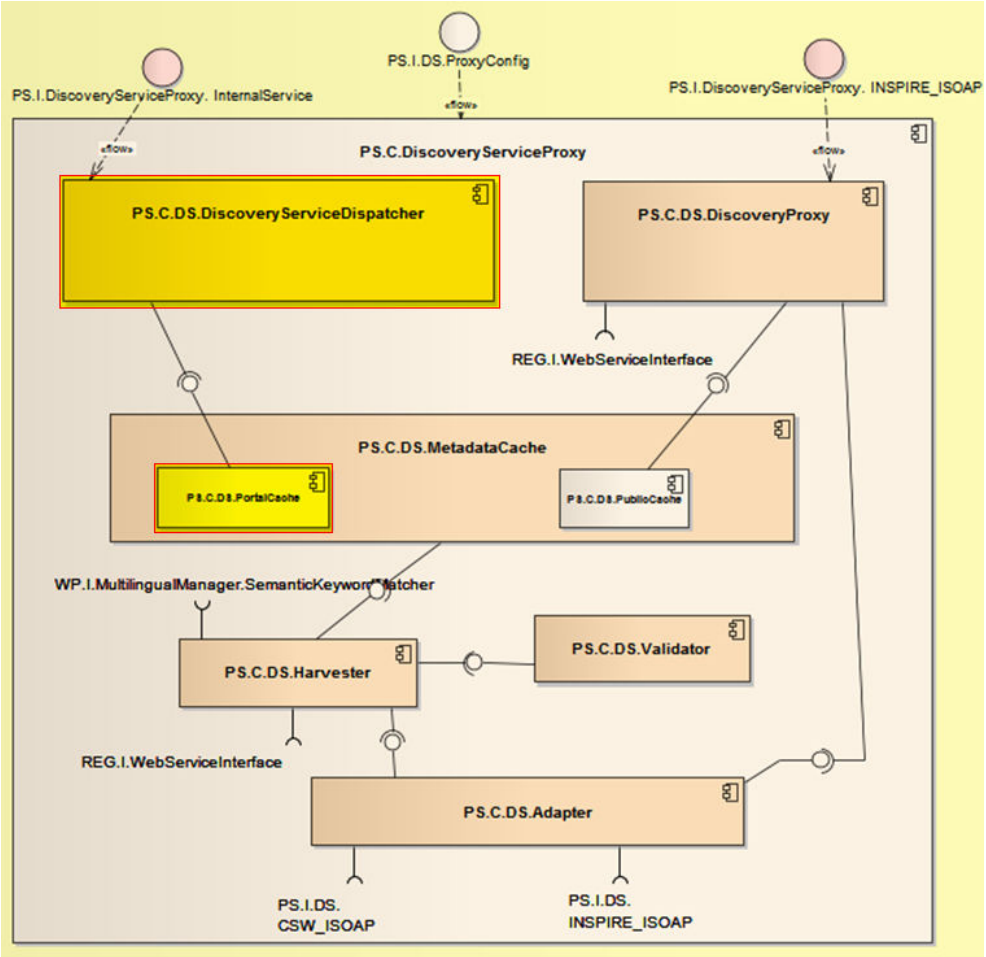


As the rest of the Geoportal UI components, the implementation in the web UI is based on Javascript and XML (AJAX) and posed no particular problems.

Only the number of http calls to the count interface remains a possible field of improvement: The interface definition of `getOccurencesByProperties` already foresees the possibility, to ask the counts for more than one property, but the current implementation does not make use of this feature in order to simplify implementation. The background of this decision was, that for different filters different `<csw:Constraint>` clauses may apply, due to the fact, that a select box filter, that has already a value set, should not consider its underlying property in the `<csw:Constraint>` clause in order to populate the count values of the other options correctly. Still some aggregation for equal constraints could be done, but this has not yet been implemented.

6. CSW internal and DB Implementation

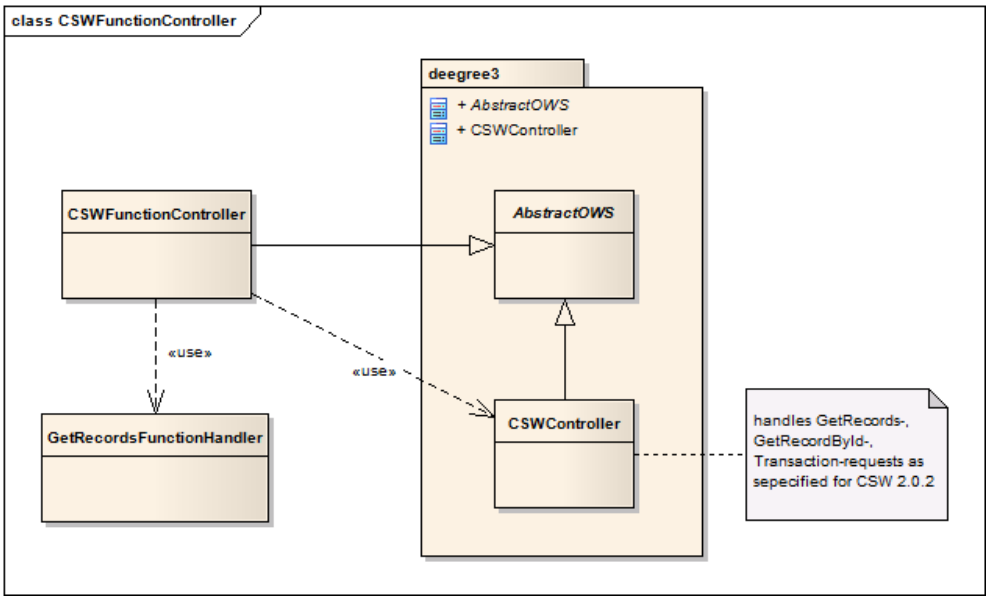
In the INSPIRE Geoportal architecture, the following components are involved on proxy side:



Though based on the deegree 3.x trunk, the INSPIRE Geoportal needed some extensions to the codebase of deegree, that have not(yet) been integrated to the main trunk. In some cases this has functional reasons (they are and will always be too specific), in other cases this is due to a very "vertical" implementation, such that only a small and specific set of potential use cases has been tested. This is the case for the anticipative counting.

The implementation fits well into the deegree 3.x service concept and architecture and implements the abstract class AbstractOWS. With this the implementation is closely connected to deegree 3.x. Furthermore the GetRecords-, GetRecordById- and Transaction-Requests, specified in the CSW 2.0.2 specification, are forwarded to the deegree 3.x CSWController. This class also handles the standard implementation of the CSW 2.0.2.

GetRecords-request containing a function that invokes CSWFunctionController which delegates the handling to the GetRecordsFunctionHandler.



It then turned out, that while the DB structure used by the deegree CSW is well suited for direct queries, it is not optimized for the aggregated counting of property values. In order to not impact on the standard code base of the CSW, the DB structure has been extended by an additional view, that reorganizes the properties' values in a suitable way.

As the concrete DB implementation has been done on Oracle 11g[13], the "Materialized View"[14] feature has been used to do this. The definition of this materialized view does also implicitly define the set of properties, that counting is enabled for, as only values for those properties are included.

This materialized view contains very few fields, basically it is a mapping between metadata record id, property name and property value. The resulting SQL query is based on the following simple query:

[13] This was an explicit requirement from the commission.

[14] See also http://docs.oracle.com/cd/B28359_01/server.111/b28326/repmview.htm. A materialized view is defined as a normal view through a more or less complex SQL statement, with the only difference, that its definition implies a physical copy of the underlying data into a new table that is structured exactly as the view. Such materialized views can be updated on-demand or automatically. As the Geoportal writes only in very few specific moments into the catalogue (during periodic updates of the cache), an on-demand strategy has been chosen in order to reduce the impact on DB performance. In other RDBMS, materialized views are either built-in or can be simulated (as for example for PostgrSQL, see http://tech.jonathangardner.net/wiki/PostgreSQL/Materialized_Views)

```
select IDX_NAME, IDX_VALUE, count(IDX_VALUE) AS NUM
from count_mview
where IDX_NAME='THEME'
group by IDX_NAME, IDX_VALUE
order by IDX_NAME, IDX_VALUE;
```

which aggregates the number of occurrences of all values in the DB, but in order to be filtered by the OGC constraint from the original query, it has to be put in join with the main CSW table (through the metadata record id) in order to add the same filters as for normal catalogue queries[15]. This way counting has more or less the same DB query performance as a normal getRecords request.

Overall, the CSW implementation demonstrates very well, how to implement such a functionality in a performing way and without interfering too much with the existing CSW codebase, though there is room for improvement and generalization with respect to the Geoportal specific use cases.

[15] The showed SQL is thus simplified on purpose for this paper in order to demonstrate the principle.

7. Conclusions

"Anticipative Counting" is a powerful instrument for the enhancement of Discovery user interfaces. The main advantages for the user are:

- A help to avoid time-consuming "void searches" with an empty result set

- A substantial improvement of the navigation through the time-dimension of the catalogue

- A numerical feedback on the content of the catalogue

The base is a count functionality, that has to be tightly integrated into the catalogue server itself, in order to guarantee identical result numbers for identical constraints. An ad-hoc extension of the OGC interface and the use of the same OGC filter constraints eased this task both on UI and on CSW side.

The performance of this functionality is very important, as, once present, it can be used in many contexts. The Discover Data page issues around ten CSW count requests each time the search constraints change (instead of one single normal getRecords). A more flexible interface definition and implementation could help in the future to reduce this number significantly, in order to avoid http and XML parsing overhead, but it turned out to be much more important, to organize the DB structure in a way that allows for such performance.

Overall, the effort we needed to implement "Anticipative Counting" is well worth its usefulness, and we see a good chance, to adopt the here discussed counting interface (or a derivate of it) also in other catalogue portal implementations, in particular where these are INSPIRE related, but not only.

REFERENCES

Stutte, J., Seiler M. and. Fierens F. (2011). *EC's INSPIRE Geoportal towards operational Status*, 7th Information Brochure INSPIRE GMES, Ed. M Schilcher, TU München, ISBN 978-3-935049-73-0

Stutte, J., Seiler M and Fitzke J. (2012). *INSPIRE Geoportal Catalogue capabilities beyond standards add value to searches*, INSPIRE Conference 2012, Istanbul

IOC Task Force for Network Services (2011): *Technical Guidance for INSPIRE Discovery Services v3.1*

OGC 07-045, CSW ISO AP, *OGC™ Catalogue Services Specification 2.0.2* - ISO Metadata Application Profile for CSW 2.0, version 1.0.0 (2007)