GES 2007 German e-Science

Available online at http://www.ges2007.de

Data Information Service based on Open Archives Initiative Protocols and Apache Lucene

Uwe Schindler[1], Benny Bräuer[2], Michael Diepenbroek[1]

[1] MARUM - University of Bremen, Bremen, Germany,
{uschindler,mdiepenbroek}@pangaea.de
[2] Alfred Wegener Institute for Polar and Marine Research, Bremerhaven, Germany,
benny.braeuer@awi.de

Abstract

We present a *generic portal system architecture* suitable for geoscientific data portals. The portals harvest data providers with *Open Archives Initiative (OAI)* protocols using XML based metadata formats like DIF or ISO-19139 format. Current implementations of OAI only support Dublin Core metadata. The new *Java based portal software* will support any XML format and makes them searchable through Apache Lucene without any other database software. The open architecture makes it possible to define searchable fields in several data formats by XPath allowing full text queries on all types of fields including numerical ranges. The metadata of all providers are stored in separate indices which makes it possible to combine them in several different portals. The web service interface allows to support custom front-ends for users and additional visualization in maps. The software will be made freely available through the Open-Source concept. A use case describes how the generic software is used in the *Collaborative Climate Community Data and Processing Grid (C3-Grid).*

1 Objectives

During the last decade there had been various initiatives and approaches in networking global data services. A main focus had been on earth sciences and related data. The lateley implemented *Group on Earth Observations (GEO)* conceived a plan for a sustained *Earth Observations System of Systems (GEOSS)* [4] which largely builds on concepts of Global Spatial Data Infrastructures (GSDI).

Correspondingly the *Infrastructure for Spatial Information in Europe (INSPIRE)* [12] is an EU directive in order to harmonize geodata in Europe. With INSPIRE the conditions of interoperable data management and improved data are created.

These two initatives clearly emphasize the need for portal frameworks to provide simple and transparent access to scientific data and metadata.

2 State of the Art

A *data portal* is an online available Internet site that typically provides simple and transparent access to distributed information resources. In the scientific context, it is perceived as broker between a multitude of data centers, information systems, institutions, organizations, and the scientific community. A data portal serves as information- and distributing system of scientific data for mutual benefit. Technically spoken, it is designed to use distributed applications, different numbers and types of middleware and hardware to provide services from a number of different sources.

Current data portals use two different solutions for metadata search:

- **Distributed search:** Every data provider has its own metadata catalogue with a (web service based) search interface to the portal. The portal receives the search request from the user and sends it instantly to all data providers. Problems with this interface are that all data providers need to have the same searching infrastructure (e.g. database software), thesauri and catalog service software (e.g. *OGC Catalog Services* [19]). Due to the distributed architecture the response time of the end-user search interface is inert. The slowest search provider dictates the overall time. An example of this architecture is *GeoPortal.BUND* [13].
- **Harvesting of distributed catalogues:** Every data provider has its own metadata catalogue but the search engine is centralized. The portal periodically harvests all metadata records into a central index and serves search requests from this index. Based on this layout all major web search engines work on (e.g. Google). The only disadvantage of this concept is reduced actuality: When metadata records are changed or deleted, the update cannot be seen immediately and users may fail with "404 Not Found" errors when accessing the data.

Due to the disadvantages of the first solution, the harvesting approach is the better solution in most cases. In earth sciences metadata is often distributed in XML based formats with content standards like ISO-19115 [20] or DIF [14]. The *Open Archives Initiative* developed interoperability standards that aim to facilitate the efficient dissemination of metadata. The *Open Archives Protocol for Metadata Harvesting (OAI-PMH)* [22] enables data providers to act as repositories that expose structured metadata. Portal providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP.

Current implementations of OAI-PMH harvesters are available in the open source community for different programming languages (Java, PHP, PERL,...) and infrastructures (harvesting to file system, to database,...). The problem with most implementations is the limitation of the metadata format to *Dublin Core* [9] because this format is mandatory for a basic OAI-PMH implementation and is the main standard in the library domain. Database structures in these software packages are designed for this simple metadata format and cannot be changed without hassle.

Based on the needs of these scientific communities we designed a generic portal system architecture suitable for (geo)scientific data portals without any constraints on the used metadata format.

3 Generic Metadata Portal Software Package

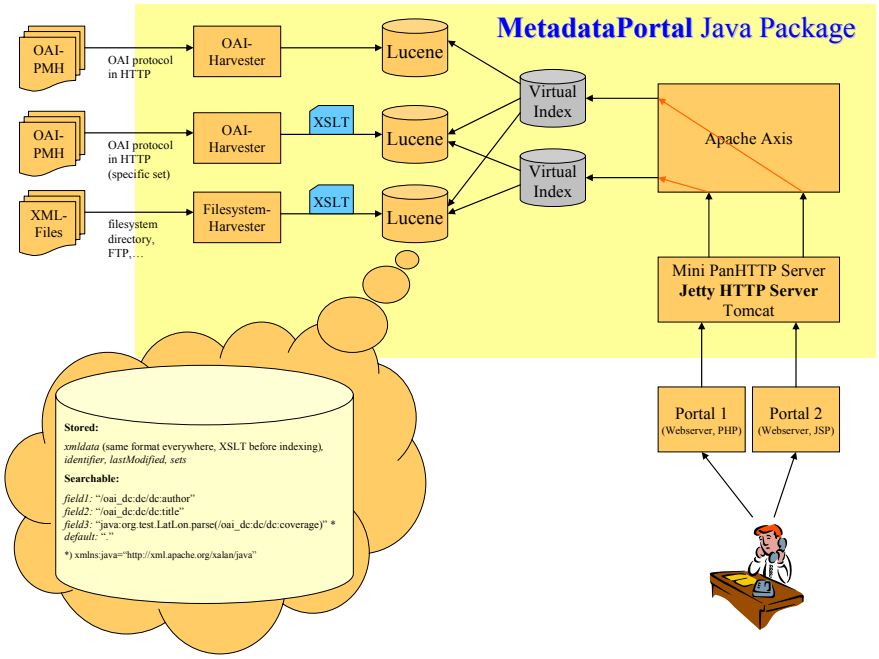

Figure 1: Overview on the generic metadata portal software

The new Java based portal software supports any XML format that can be harvested from OAI-PMH Repositories, file systems, or OGC Catalog Services (in preparation) and makes them searchable through *Apache Lucene* [16] without any other database software.

The portal software will be made freely available through the open source concept when the code base has proven its usability and the design of the programming API for portal implementers is stable. Configuration of metadata formats, data providers and searchable fields is done by a XML based configuration file (see figure 1).

3.1 Introduction to Apache Lucene

For the following sections we give some general information about the techniques and features of Apache Lucene – a high-performance, full-featured text search engine library written entirely in Java that is suitable for nearly any application requiring full-text search, especially cross-platform. Some of the features are:

- **Ranked searching:** best results returned first.
- **Many powerful query types:** phrase queries, wildcard queries, proximity queries, exact phrase queries, range queries for date/time and number values.
- **Fielded searching:** All fields are searchable as a whole or each field separately.
- **Boolean operators:** Any combination between search terms (AND, OR, NOT).
- **Sorting by any field.**
- **Multiple index searching with merged results.**
- **Simultaneous searching and updates.**

The fundamental concepts in Lucene are *index, document, field* and *term*. An *index* contains a sequence of *documents*. A *document* is a sequence of *fields*. A *field* is a named sequence of *terms*. A *term* is a text string. The same string in two different fields is considered a different term. Thus terms are represented as a pair of strings, the first naming the field, and the second naming text within the field (see top right of figure 2).

The index stores statistics about terms in order to make term-based search more efficient. Lucene's index falls into the family of indexes known as an *inverted index*. This is because it can list, for a term, the documents that contain it. This is the inverse of the natural relationship, in which documents list terms.

3.2 Harvester and Index Builder

The portal software harvests all metadata into the Lucene index directly without the need to store them separately. The event-based abstract harvester class is universally designed to support a lot of different harvesting solutions. It is responsible for collecting new or updated metadata XML files from different sources. For each new or updated document it creates a *Document Object Model (DOM) tree* [2] and notifies the index builder that analyzes the tree in a different thread and updates the index.

Additionally the harvester class allows for an on-the-fly validation by schema and transformation by XSLT [7] from any metadata format into the index specific one provided that the content standards are compatible. This helps if one of the data providers uses another metadata format.

The reference implementation is a high-performance OAI-PMH harvester. We went for an own implementation because available open-source harvesters mostly support only Dublin Core metadata which is very simple. In contrast, metadata from science is often complex (e.g. ISO 19115). As OAI-PMH embeds

all documents in a "big" XML file that should be parsed sequentially, available DOM-based harvesters often fail with "out of memory" problems. We use the *Simple API for XML (SAX)* [5] to parse the OAI response using *Jakarta Digester* [1] as frontend. Jakarta Digester was extended to support on-the-fly switching to build a DOM tree when coming to the metadata component and switching back when going further with OAI-PMH protocol. This makes it possible to sequentially harvest the whole XML file and build multiple separate DOM trees that are sent to the index builder. To also support simple setups our package contains a very simple harvester for XML files from local file systems.

The index builder extracts the contents from the DOM tree and passes four types of objects to the full text search engine Lucene which performs the subsequent indexing. The types of objects are:

- **A list of user-defined fields with their contents.** The open architecture makes it possible to define all searchable fields in several data formats by XPath [8]. This allows not only full text queries, even numerical or date ranges are retrievable on specific parts of the metadata. In Lucene, fields may be *stored*, in which case their text is stored in the index literally, in a non-inverted manner for later retrieval as part of the document. Fields that are inverted are called *indexed*. A field may be both *stored and indexed*. The text of a field may be *tokenized* into terms to be indexed, or the text of a field may be used literally as a term to be indexed. Most fields are tokenized, but it is useful for identifier and numeric fields to be indexed literally. The implementation of searchable (inverted) numerical fields (even dates are numerical values) inside a full text index is described in section 3.4. XPath definition and properties of the fields are done by the configuration file.
- **A tokenized *default* field covering the whole document for a Google like search.**
- **The full string-serialized DOM tree as a *compressed stored* field.** It comprises the central metadata inventory to be used for e.g. the display of metadata details. An additional relational database for storage of this information is not needed. In contrast, usual combinations of e.g. MySQL and built in full text search engine (FTS) are limited in functionality and performance, in particular for larger databases.
- **Control information for each record:** a timestamp, the record identifier (for later updates), and – when using OAI-PMH – the set information *(stored and indexed)*.

The metadata of all different harvested data providers are stored internally as separate indexes (with exactly equal structure) giving the administrator the possibility to manage them separately and allowing for flexible combinations into "virtual" indexes for searching.

3.3 Search Interface

We added a portal-specific Java API and a corresponding web service to the API of Lucene which allows for a full featured and flexible usage of the search

engine. It abstracts the underlying Lucene API and combines it with the portal configuration file. You can formulate queries as described in section 3.1 using the defined fields. It uses optimized range queries (section 3.4) for queries on numerical and date/time values. Search results are returned in the metadata XML format and/or the stored fields.

Based on the API the programmer can even display the results in maps (e.g. using *UMN mapserver* or *Google Earth*). Sample implementations for different metadata formats will be provided together with the portal package.

The World Data Center for Marine and Environmental Sciences (WDC-MARE) with its data library PANGAEA® [10] has implemented several data portals for EU projects (e.g. EUR-OCEANS, CARBOOCEAN,... – a current list of portals can be found at http://wiki.pangaea.de/wiki/Portal) to disseminate and publish data and metadata with a front-end software written in PHP [3] using the web service interface. Search speed for any query type is excellent (query on $\approx$ 500,000 metadata records $< 0.05\,\mathrm{s}$ incl. displaying of first 10 results on a state-of-the-art Opteron machine running Linux). The later described C3-Grid DIS/WFIS components (see section 4) use the native Java API.

3.4 Optimized Range Queries

As Lucene is a full-text search engine and not a database it cannot handle numerical ranges (e.g. field value is inside user defined bounds, even dates are numerical values). So it expands a range to a big "OR" query consisting of all terms between the boundaries (this is called query rewriting and is used for wildcard queries, too). When the index contains lot of documents with distinct numerical values and the range boundaries are far-off, this "OR" list is extremely long. Older versions of Lucene (current is version 2.1) had a limitation to a maximum number of "OR" terms and threw an exception. Current versions use a bitmap for these type of queries, nevertheless, all terms between the range boundaries must be discovered.

We developed an extension to Lucene that stores the numerical values in a special string-encoded format with different precisions (all numerical values like doubles, longs, and timestamps are converted to 8 byte *long long words* and stored with precisions from one byte to the full 8 bytes). A range is then divided recursively into multiple intervals: The center of the range is searched only with the lowest possible precision, the boundaries are matched more exact. This reduces the number of terms dramatically (in the lowest precision of 1-byte the index only contains a maximum of 256 distinct values). Overall, a range could consist of a theoretical maximum of

$$\underbrace{7 \times 256}_{\substack{\text{boundaries split into} \\ \text{7 different precisions}}} \times \underbrace{2}_{\text{lower and upper part}} + \underbrace{256}_{\text{center with lowest precision}} = 3840$$

distinct terms (when there is a term for every distinct value of an 8-byte-number in the index and the range covers all of them). Practically, we have seen up

to 300 terms in most cases (index with 500,000 documents and a homogeneous dispersion of values). A detailed description of the algorithm is given in [21].

This dramatically improves the performance of Lucene on range queries. Performance is no longer dependent on the index size and number of distinct values because there is an upper limit not related to any of these properties.

4 Use Case: Adoption of Framework in C3-Grid

The Collaborative Climate Community Data and Processing Grid (*C3-Grid* [6]) proposes to link distributed data archives in several German institutions for earth system sciences and to build up an infrastructure for scientists which provides tools for effective discovery, transfer, and processing of scientific data with modern grid technologies.

4.1 Problem Definition

At the beginning of the project planning "data discovery" faced two problems – first we had to harmonize different data standards used by project partners within C3-Grid. We solved this issue by using the ISO-19115 specification (with ISO-19139 metadata schema), but this led to the second problem, a more technical one. Which architecture is flexible and intelligent enough to:

- collect the metadata files from the data providers (e.g. via OAI-PMH)?
- store them in a "central index"?
- provide a fast, generic access to this data for the users of C3-Grid?

4.2 Problem Solution

We arrived at the decision to use the Data Information System introduced in this paper because it is up to the mark:

- **Collecting:** The data providers of C3-Grid have agreed on serving their metadata via OAI-PMH. Therefore, in the majority of cases they use *DLESE jOAI software* [11].
- **Data Store:** Harvesting of the distributed data resources leads to storage of the metadata in a central index. With this method it is possible to create copies of the index at one or more sites (partner institutions) fast and easy. We will use this for fall back solutions, load balancing, test systems, etc.
- **Access:** Because the system is based on the Apache Lucene project, it inherits its vantages and features, like high-performance index querying (see section 3.1).

4.3 Data Information Service

Figure 2 shows the integration of the Data Information Service (DIS) in the architecture of C3-Grid, a classical 3-tier model. Tier-1 (portal) is the GUI for users and allows access to the DIS and its contained data. Tier-2 is the DIS itself, it comprises web server frontend, index and harvesting backend. The

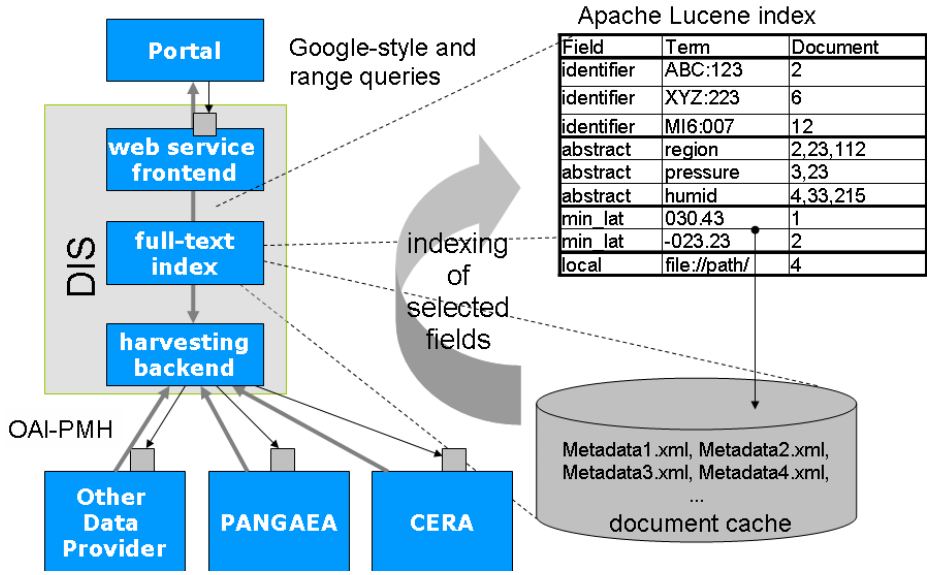

Figure 2: C3 DIS Architecture *(by T. Langhammer, ZIB, Germany)*

latter allows the access to Tier-3, the actual data archives which are provided by C3-Grid partners.

A graphical user interface for e.g. queries and presentation of data is supplied by the *GridSphere Portal Framework* [15]. It is pure Java and fully compliant to Java Specification Request 168 (Portlets) [17], so we have implemented the interface to the search service (see section 3.3) directly (resp. used its API). A detour over the web service is not necessary here. On the portal frontend user can search for datasets by full text, variable names, date/time constraints, or a bounding box and start jobs on selected data sets. As shown in figure 3 the term "ipcc" was entered in the Google-like search form (a field was not set, so it is a global search within the index entries). The result page is generated with the field values from the retrieved records.

4.4 Workflow Information Service

In a later stage of C3-Grid also possible workflows at various computer centers will be described by metadata and made available in so called *Workflow Information Service* (WFIS). Likewise, for DIS, the workflow providers generate a set of workflow information containing a pattern of the offered workflow and the preconditions for the metadata which can be used within the selected workflow(s). This set (in XML format) will also be available via OAI-PMH and searchable via the generic portal system.

Users of the C3-Grid portal choose a workflow to work with, and the WFIS will return all necessary information. The preconditions are based on a DIS-

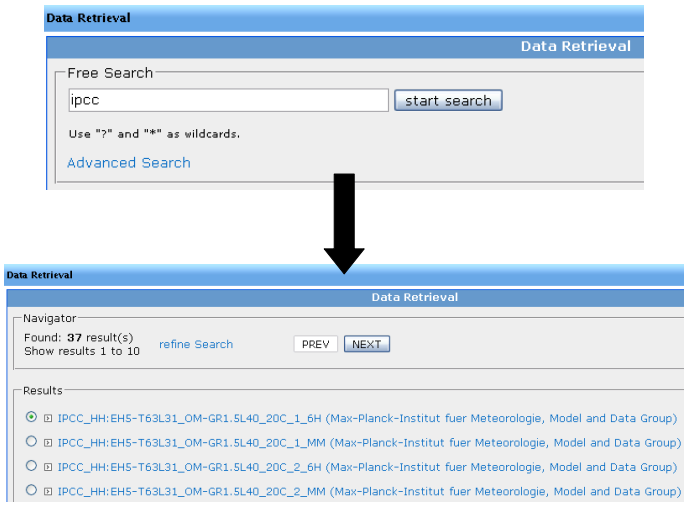

Figure 3: A simple query and its result

query (all information are available via Lucene) which returns only those datasets the selected workflow can handle. The workflow pattern specifies how to generate a web-form for the user where he can insert parameters etc. for this workflow. This translation could be made with XSLT, Python, or any other popular script languages.

C3-WFIS is still in the planning phase. Figure 4 shows a possible integration in our architecture.

5 Conclusions

The new generic portal software helps providing a fast and easy access to scientific data based on standardized, XML metadata formats. Usage of the Apache Lucene full text search engine – in contrast to classical relational data bases – conforms to the current user experience.

Due to the open architecture and usage of open protocols it is open to the library world, too. It helps for wide interoperability between data bases, data grids, and classical publications in the context of the *Open Access Initiative* and beyond.

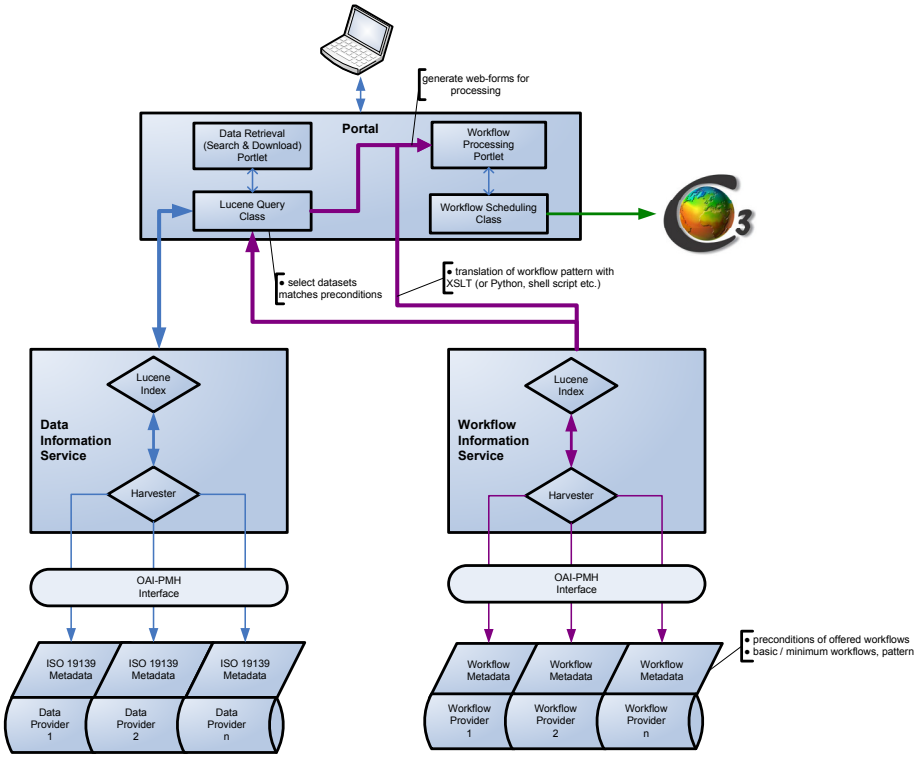

Figure 4: Overview on the integration of the metadata portal software into DIS and WFIS

References

1. The Apache Jakarta Project. **Jakarta Commons Digester.** http://jakarta.apache.org/commons/digester/
2. Apparao, V, Byrne, S, Champion, M, Isaacs, S, Jacobs, I, Le Hors, A, Nicol, G, Robie, J, Sutor, R, Wilson, C, Wood, L, 1998. **Document Object Model (DOM).** W3C Recommendation, http://www.w3.org/DOM/
3. Arntzen, T, Bakken, S, Caraveo, S, Gutmans, A, Lerdorf, R, Ruby, S, Schumann, S, Suraski, Z, Winstead, J, Zmievski, A. **PHP – Hypertext Preprocessor.** http://www.php.net
4. Battrick, B, 2005. **Global Earth Observation System of Systems (GEOSS) 10-Year Implementation Plan Reference Document.** ESA Publications Division, Noordwijk, The Netherlands
5. Brownell, D, 2002. **SAX2.** O'Reilly – SAX project homepage online: http://www.saxproject.org/
6. C3-Grid. **The Collaborative Climate Community Data and Processing Grid.** http://www.c3grid.de
7. Clark, J, 1999. **XSL Transformations (XSLT) – Version 1.0.** W3C Recommendation, http://www.w3.org/TR/xslt

8. Clark, J, DeRose, S, 1999. **XML Path Language (XPath) – Version 1.0.** W3C Recommendation, http://www.w3.org/TR/xpath
9. DCMI Usage Board, 2006. **DCMI Metadata Terms.** http://dublincore.org/documents/dcmi-terms/
10. Diepenbroek, M, Grobe, H, Reinke, M, Schindler, U, Schlitzer, R, Sieger, R, Wefer, G, 2002. **PANGAEA - an Information System for Environmental Sciences.** Computer & Geosciences, 28, 1201-1210, doi:10.1016/S0098-3004(02)00039-0
11. Digital Library for Earth System Education. **The DLESE jOAI software.** http://www.dlese.org/dds/services/joai_software.jsp
12. The European Parliament, 2007. **DIRECTIVE 2007/.../EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of establishing an Infrastructure for Spatial Information in the European Community (INSPIRE).** http://www.ec-gis.org/inspire/
13. Geschäfts- und Koordinierungsstelle des Interministeriellen Ausschusses für Geoinformationswesen (IMAGI) im Bundesamt für Kartographie und Geodäsie. **GeoPortal.BUND.** Bundesamt für Kartographie und Geodäsie, Frankfurt am Main, http://www.geoportal.bund.de/
14. Global Change Master Directory. **Directory Interchange Format (DIF) Writer's Guide.** National Aeronautics and Space Administration, http://gcmd.nasa.gov/User/difguide/
15. GridSphere. **The GridSphere Portal Framework.** http://www.gridsphere.org
16. Hatcher, E, Gospodnetic, O, 2004. **Lucene in Action.** Manning Publications – Apache Lucene online: http://lucene.apache.org/java/docs/
17. Hepper, S, 2006. **Java Specification Request 286: Portlet Specification 2.0.** http://jcp.org/en/jsr/detail?id=286
18. Kresse, W, Fadaie, K, 2004. **ISO Standards for Geographic Information.** Springer, Heidelberg.
19. Open GIS Consortium, 1999. **The OpenGIS™ Abstract Specification – Topic 13: Catalog Services**
20. Østensen, OM. ISO/TC211, 2003. **Geographic information – Metadata.** International Organization for Standardization, ISO 19115
21. Schindler, U, Diepenbroek, M, 2007. **Generic Toolbox for Metadata Portals.** Computer & Geosciences, *in prep.*
22. Van de Sompel, H, Nelson, ML, Lagoze, C, Warner, S, 2004. **Resource Harvesting within the OAI-PMH Framework.** D-Lib Magazine, December 2004, http://www.dlib.org/dlib/december04/vandesompel/12vandesompel.html