Ontology Repositories

Jens Hartmann[1], Raúl Palma[2], and Asunción Gómez-Pérez[2]

[1] Center for Computing Technologies (TZI)
University of Bremen, Germany
e-Mail: jh@tzi.de
[2] Ontology Engineering Group, Laboratorio de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
e-Mail: {palma,asun}@fi.upm.es

Summary. The growing use and application of ontologies in the last years has led to an increased interest of researchers and practitioners in the development of ontologies, either from scratch or by reusing existing ones. Reusing existing ontologies instead of creating new ones from scratch has many benefits: It lowers the time and cost of development, avoids duplicate efforts, ensures interoperability, etc. In fact, ontology reuse is one of the key enablers for the realization of the Semantic Web. However, currently, ontologies are mostly developed from scratch, due to several reasons. First, ontologies are usually tailored to work for specific applications, restricting its potential reusability. Second, developers usually follow a monolithic approach when developing ontologies, usually covering different domains, hampering the reusability of relevant parts for other applications. Third, ontologies are rather difficult to find due to the lack of standards for documenting them and appropriate tools supporting intelligent ontology discovery and selection by end users. In this chapter, we define a generic ontology repository framework that enables the implementation of fully-fledged ontology repositories providing the technological support to the aforementioned issues. We distinguish between the ontology repository itself and the software to manage the repository, and describe their main aspects and services. Finally, we present two exemplary systems based on this framework.

1 Introduction

Knowledge reuse and access is one of the leading motivations for the Semantic Web. Driven by those intensions an increasing amount of ontologies can be found nowadays on the Web distributed among personal or institutional web pages. One of the key problems the ontology engineering community has to face at the moment is that most ontologies are built from scratch—rather than reusing existing ones—leading to high engineering efforts and costs. One of the main reasons is that most existing ontologies are build having a specific application scenario in mind, making them similar to custom software. This

leads to ontologies that are tailored to work with specific applications, but are not knowledge representation artifacts in the traditional sense. When designing these ontologies, engineers focus on expected behavior in the application rather than on reuse and interoperability with other ontologies. Another problem is that ontologies trying to cover domains in the knowledge representation sense are often too big to be reused efficiently. These ontologies try to capture the complete domain knowledge whilst ontology engineers normally only need to reuse certain parts for their ontology. Nevertheless, having modular ontologies is not enough to facilitate the reusability of ontologies if developers are not able to find them efficiently. We need an appropriate infrastructure that enables an intelligent ontology discovery and selection by end users.

Currently, initial collections of ontologies have been created during the last years, e.g. the DAML Library[3]. Apparently these resources are mostly created by hand and annotated manually. Many of these ontologies fail to follow consistent representation and storage conventions, so humans and machines are hampered by finding and reusing them. The process of identifying and accessing ontological resources, which can be summarized as *ontology retrieval*, is consequently affected by non-existing mechanisms and standards for storing and representing ontologies.

These circumstances point out the strong requirement for novel methods facilitating an efficient access and reuse of ontologies within a large scalable and reliable infrastructure, so called *ontology repositories*. The storage of knowledge encoded by ontologies can only be part of the solution. Crucial for ontology repositories is that additional knowledge about ontologies, so called *meta knowledge*, is managed together in such a repository.

We propose a Generic Ontology Repository Framework (GORF) including specific module support, tailored to exactly those requirements. We base our framework on experiences gained by realizing our ontology repository Onthology[4] and previous work on the ontology metadata vocabulary OMV[5]. After their deployment, it became evident that having a central place to find ontologies and ontology modules alone does not solve the problem of quality assurance or knowing which module is the most suited for a specific task. We therefore integrate an Open Rating System (ORS) into our repository to assist in the retrieval process and prove quality assurance through feedback from the community itself. To fill the initial void of modules, existing ontologies can be partitioned (depending on their knowledge representation formalism e.g. using the approach and tool mentioned in [6]) to create relevant modules.

In the following we discuss essential aspects of ontology repositories. To begin with, in section 2 we describe the historical development from data to ontology repositories. In section 3 we will describe the generic architecture

[3] c.f. http://www.daml.org/ontologies/
[4] c.f. `http://www.onthology.org/`
[5] c.f. `http://omv.ontoware.org/`

of an ontology repository and corresponding management systems. Core elements and services of an Ontology Repository are discussed in section 4. Further we illustrate management systems for ontology repositories in section 5 and exemplify a centralized versus a decentralized solution in section 6. We conclude in section 7 and comment on further steps.

2 From Data Repositories to Ontology Repositories

In this section we present an overview of how repositories have evolved throughout time from general purpose data repositories to specialized ontology repositories.

In literature exist many different meanings and definitions to what a data repository is, and in general to what a repository is. Hence we will first discuss what we understand by a data repository, instead of giving another definition. We consider a data repository as a collection of digital data that is available to one or more entities (e.g. users, systems) for a variety of purposes (e.g. learning, administrative processes, research, etc.) and that has the characteristics proposed by Heery R. and Anderson, S. [16]:

- content is deposited in a repository, whether by the content creator, owner or third party
- the repository architecture manages content as well as metadata
- the repository offers a minimum set of basic services e.g. put, get, search, access control
- the repository must be sustainable and trusted, well-supported and well-managed

The term data library is usually used in the literature to refer to subject specific datasets (e.g. climate data library, time series data library, geospatial data library, etc.). Moreover, a data library tends to house local data collections and provides access to them through various means. Thus, in general a data library usually provides access to the complete dataset instead of providing the basic services (e.g. search, put, get) a data repository offers.

Around the middle 1990s the term digital library (previously also known as electronic library or virtual library) was first made popular by the NSF/DARPA/NASA Digital Libraries Initiative. According to [1] a digital library is a managed collection of information, with associated services, where the information is stored in digital formats and accessible over a network. The information stored can be very diverse and used by many different users. In general a digital library is considered similar to a traditional library (i.e. it used by users to find information that others have created, and use it for study, reference, or entertainment) but it takes advantage of the new technologies to deliver the information to users.

Data warehouses [17] became popular during the late 1980s and early 1990s. The purpose of a data warehouse is to perform analysis of the stored data for management's decision making. Data is entered into this repository

periodically, usually in an append-only manner. A data repository however, does not necessarily have that analysis functionality provided by a data warehouse.

Similarly to data repository, it is also possible to find many different meaning and definitions to what is a knowledge base. Yet, in general, a knowledge base is a central repository of knowledge artifacts. Usually a knowledge base may use an ontology to formally represent its content and its classification scheme, but it may also include unstructured or unformalized information expressed in natural language or procedural code. Also, in contrast to a data repository, usually the purpose of the knowledge base is to allow automated deductive reasoning over the stored knowledge (i.e. decide how to act by running formal reasoning procedures over the base of knowledge).

It is not surprising that some years ago, the ontology and semantic web community became interested in using repositories to hold semantic content (e.g. ontologies). Within the last years, ontologies have seen an enormous development and application in many domains, especially in the context of the semantic web. Academia and industry are developing and using ontologies to provide new technologies and support daily operations. Therefore, currently there exists a large amount of ontologies developed by many different parties which makes necessary the means to share and reuse them.

Initial efforts to collect the base of existing ontologies proposed the creation of library systems (i.e. known as Ontology library systems) that offered various functions for managing, adapting and standardizing groups of ontologies [8]. These systems defined an important environment in grouping and reorganizing ontologies for further re-use, integration, maintenance, mapping and versioning. They defined an evaluation model based on the functionality the library system provided. Examples of library systems are: WebOnto, Ontolingua, DAML Ontology Library System, SchemaWeb, etc.

Currently, efforts are put in the creation of ontology repositories. An ontology repository is similar to what Ding et al defined as an ontology library system [8], but they also have some differences. In the remaining of this chapter we will propose an widely-accepted definition of these terms.

3 Generic Ontology Repository Framework

Ontology reuse is still rarely encountered today. This is partly due to the problem of finding suitable ontologies to reuse, and the way most ontologies are created, namely without reusability in mind. Also, most of the established ontologies containing domain knowledge are simply too big to be easily reused, and no quality information is available on web ontologies.

We argue that ontology engineers can adopt from software engineers a way how ontologies could be designed, namely modular. This way, small, reusable components (ontology modules) are produced during creation. To manage and provide access to ontologies we propose an *Generic Ontology Repository*

Framework GORF with specific module and rating support. So not only ontologies or modules can be found in a single place, one can also see reviews about their quality or their usefulness in different scenarios. This way ontology engineers have a one-stop-shop for reusable knowledge artifacts.

The term *ontology repository* can be seen as evolved term coming from the classical understanding of data repositories [17]. In the remaining we rely on the following understanding of an ontology repository and corresponding management systems.

Definition 1 (Ontology Repository & Management System). *An Ontology Repository (OR) is a* **structured collection** *of ontologies (schema and instances), modules and additional meta knowledge by using an Ontology Metadata Vocabulary. References and relations between ontologies and their modules build the* **semantic model** *of an ontology repository. Access to resources is realized through semantically-enabled interfaces applicable for humans and machines. Therefore a repository provides a formal* **query language**.

Software to manage an ontology repository is known as Ontology Repository Management System (ORMS). An ORMS is a system to store, organize, modify and extract knowledge from an Ontology Repository.

The main driving motivation creating ontology repositories is to support knowledge access and reuse for humans and machines. Hence ontology repositories on the one hand act as a storage facility and on the other provide access to knowledge through defined interfaces and policies. To achieve these goals, comprehensive facets must be considered by an ontology repository when handling ontologies. In general, these facets can be separated into access-related and storage-related aspects. A general requirement is that ontology repositories can support the entire ontology lifetime, i.e. ranging from the ontology engineering process to the desired application within specialized tools or tasks. Additionally, long term knowledge conservation is one of the crucial ontology repository tasks.

On a technical level, practical realizations of ontology repositories might differ in their concrete implementation. In contrast to that, relevant components or services on a conceptual level are reusable among different technical solutions. Consequentially, we now present a conceptual framework for ontology repositories. Based on different ontology repository implementations and realizations in the past [14], we identified a set of relevant components and services which are embedded into a scalable and reliable framework.

To be more specific the Generic Ontology Repository Framework (GORF) extends conceptually the SEAL (SEmantic portAL) [15] framework and preliminary work on ontology repositories, as described in [14]. As a result, the remaining framework GORF facilitates semantic-driven access and reuse of ontologies, thereby maintaining the vision of the semantic web. Furthermore, the framework remains scalable and can be distributed and interconnected

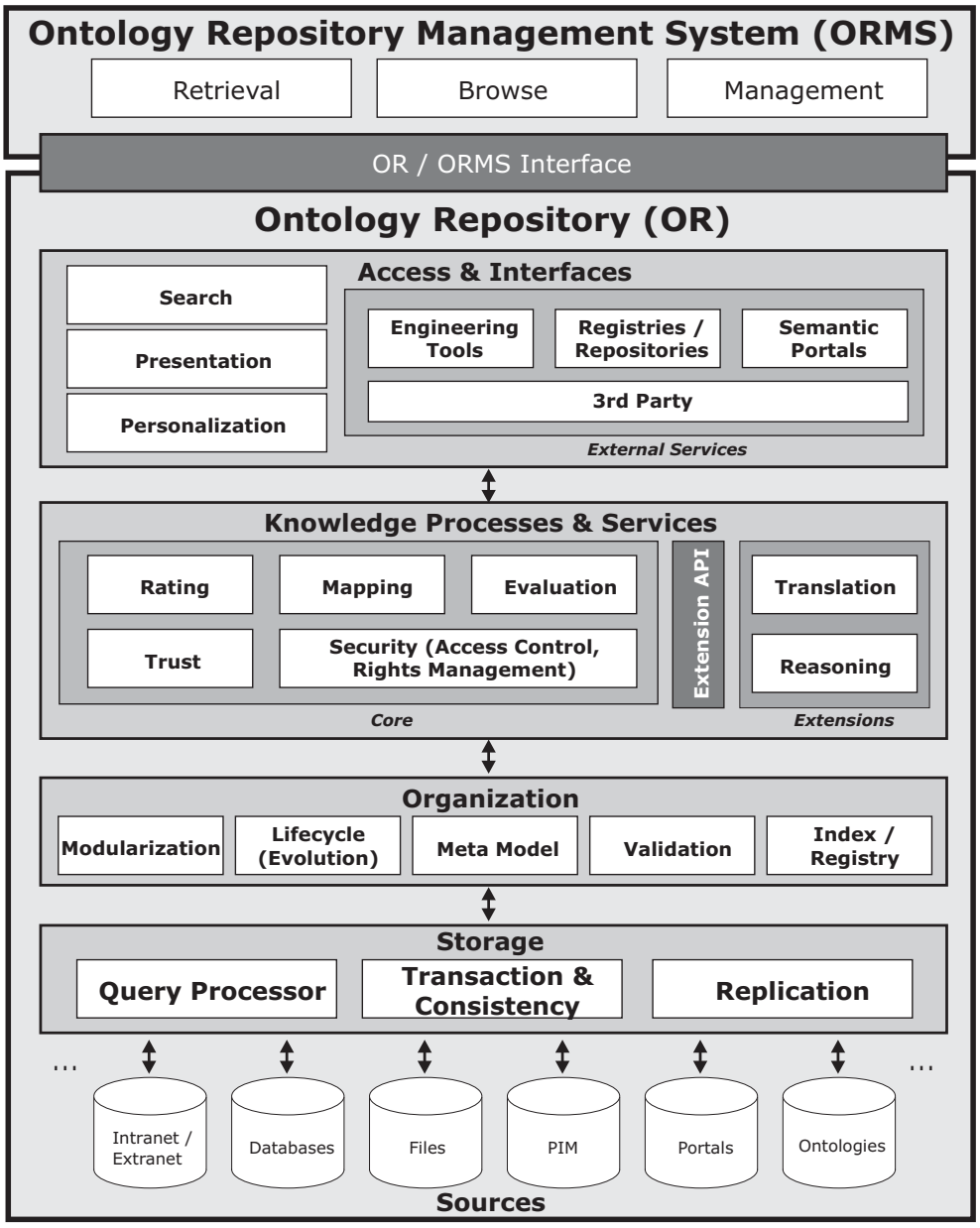

Fig. 1. The GORF Architecture

to other repositories—like the one used in KMI's Watson[6]—using e.g. web services. We assume that there will be technically heterogeneous solutions for

[6] c.f. `http://watson.kmi.open.ac.uk/`

semantic applications and especially for ontology repositories on the WWW. To ensure an easy and efficient knowledge exchange between applications, knowledge workers, and repositories, GORF acts as a framework identifying required components and services on a conceptual level, as shown in Fig. 1.

GORF distinguishes between the ontology repository itself and software to manage the repository. The latter one is called Ontology Repository Management System (ORMS). We claim that such knowledge intensive applications are also build using semantic technologies, as shown in [12]. In the following we briefly discuss the main aspects and services of an Ontology Repository and an ORMS.

4 Ontology Repositories

The framework for ontology repositories includes five conceptual layers. These layers can be seen as knowledge workflows, from the bottom to the top layer. In the following, we discuss each layer briefly.

4.1 Knowledge Access

Maintaining the vision of the Semantic Web [3], the architecture provides interfaces for humans and machines. Presenting knowledge to users involves a sophisticated visualization of knowledge for users with different interests and experiences. Thus, the framework must provide adaptable views on the stored knowledge, as shown in [15]. Although an ontology represents a commonly shared conceptualization of a domain, users typically have their own personal views and may request different visualizations. Therefore, an ontology repository should provide personalization services, which can become key success factors.

To sum up, the following aspects show up as elementary access functions.

- **Presentation & Visualization**: The access layer generates flexible graphical user interfaces for users in different formats, e.g. HTML output. Underlying templates define where, how, and when the framework presents knowledge to the user. Furthermore, the framework is able to dynamically generate ontology browsing interfaces and navigation bars,
  So, the presented framework for ontology repositories must provide several views of the stored knowledge. The presentation layer generates graphical user interfaces for users, e.g. by producing HTML output. Underlying templates define where, how, and when the framework presents knowledge to the user. Portals based on SEAL dynamically generate the ontologys browsing interface and navigation bar. To support users interacting with the portal, we developed a context-sensitive help system that provides useful tips and explanations based on the current context.
- **Searching & Querying a repository**: Our framework offers several search and query functionalities to the user. These include both standard full-text search forms and complex query forms, such as allowing a query for specific concepts or using simple query logic for a set of queries.

- **Personalization**: Although an ontology represents a commonly shared conceptualization of a domain, users typically have their own personal views and may request different visualizations. So, a semantic portal, especially a community one, should provide personalization services, which can become key enablers for successful repositories.

Besides functionalities for accessing knowledge and rendering its presentation, the access component defines the interface used by the ORMS to manage the OR.

4.2 Knowledge Processes & Services

Crucial to ontology repositories are processes and services for handling the stored knowledge within the repository.

- **Rating**: GORF will support an Topic-Specific Trust Open Rating System (TS-ORS) that can provide means to ensure the quality of ontologies and modules in the repository. Open Rating Systems (ORS) [10] have become increasingly popular over the last years. Nowadays, businesses have realized their value in customer satisfaction and information, and a variety of websites employ them. Examples are Epinions[7], where products can be reviewed, Slashdot[8], where articles and news can be reviewed, Amazon[9] (in the user review section), and iTunes[10] where users can review music. The general idea of ORS is to give everyone a voice, not only the so-called experts. One nice aspect about not excluding anyone is that ORS scale even when the rate of new content is growing steadily. Normally, in ORS, a meta-rating approach is used. This means that reviews are rated useful or not by other users. Based on these meta-reviews, a Web of Trust can be computed [11] and reviews and products can be ranked. One of the problems of the original algorithms and models [10, 11] proposed for ORS was that they were inflexible in the way objects could be reviewed (only complete objects) and how trust could be expressed (only globally). This is problematic because most people are only experts in certain fields and not in all. Another problem is that content in the system could only be rated as a whole, not specific properties separately. When applying ORS to review complicated content like ontologies [22], this is not sufficient. Users have to be able to review only those parts of the content they understand or have expertise on reviewing. To solve this problem, Lewen et al introduced topic-specific trust in ORS [19] and extended the underlying ORS model, making it a TS-ORS.
- **Mapping**: Mapping—frequently also called alignment—of ontologies is a core task to achieve interoperability and therefore a foundational service of

[7] c.f. `www.epinions.com`
[8] c.f. `www.slashdot.org`
[9] c.f. `www.amazon.com`
[10] c.f. `www.apple.com/itunes`

an ontology repository. Because most ontologies reflect a subject-oriented view of the world, different people will model knowledge differently. Being able to link these different representations is important for the success of the Semantic Web. Thus, an ontology repository needs to support mapping mechanisms.

- **Evaluation**: In contrast to ontology rating that is a subjective assessing of an ontology, ontology evaluation can be seen as an assessment of the quality and the adequacy of an ontology or parts of it regarding a specific aim, goal or context. So far, several methods for evaluating ontologies have been proposed. An overview can be found in [29]. Selected evaluation strategies can be implemented in an *evaluation component* and applied in a large repository.
- **Trust**: Trust is an ongoing and currently not fully solved research problem in the area of the Semantic Web. However, we see trust management as an important functionality for managing knowledge in ontology repositories. The ORS partly addresses this issue in GORF.
- **Security**: Due to intellectual property rights, commercial licenses, patents or copyrights, not all knowledge artifacts may be accessible by the public. Therefore clear access control and right management functionalities are required. While knowledge access might be restricted, meta knowledge like OMV [13] remains accessible and processable. As a result, commercially used knowledge artifacts can be identified in a repository while the access is secured by specialized services like payment systems.

Additional processes or services might be added or attached through an *extension component*. For example, reasoning or validation services might be included here.

4.3 Knowledge Organization

The developed framework GORF can handle massive amounts of knowledge stored in a repository. Additionally, the technology allows for having multiple portals as access points on top of one ontology repository. Using the knowledge representation mechanism, we developed several continuative knowledge-organization methods to provide fast and effective access to knowledge.

- **Modularization**: The introduced framework aims to use ontology modules as key elements to be stored in a repository—alongside existing ontologies—for the reasons motivated beforehand. The core idea behind ontology modularization is the identification of reusable knowledge artifacts which are adaptable to different tasks and remain domain-independent. In contrast to an ontology, which aims at providing a domain-specific conceptualization in one construct for a set of tasks, a module represents a shared, domain-independent conceptualization which is adaptive to and intended for re-occurring tasks and applications. In general, ontology modules are comparable to software libraries in the software engineering domain.

A key pre-condition for modularization is an expressive modeling language for ontologies, e.g. [5]. Currently first approaches can be found in literature for the modularization of ontologies and connecting these modules [30]. These approaches are, however, mostly driven by the underlying logic—except the ones working on structural information, like graph-based modularization approaches (for an overview we refer to [7]).
The main principle of modularization is described in an abstract way whereby we distinguish the process of modularization on existing ontologies and modularization whilst an ontology is being created. The first task, modularization on existing ontologies, can be considered as an ontology re-engineering task and the latter one represents an ontology engineering task. We assume that in most cases the effort required to identify useful modules in large and possibly unknown ontologies is too expensive for manual modularization. So automatic or at least semi-automatic mechanisms are required. First steps in this direction can be found in [6], where the ontology engineering environment itself provides means to extract modules out of existing ontologies, keeping the logical entailments intact. So the main challenge in extracting modules from existing ontologies is to identify modules that are best suited for reuse and maintenance issues.
When new ontologies are created, they can be designed with a module-based approach in mind. Software engineers are used to program following the object-oriented modeling paradigm [26], meaning they encapsulate required behavior in smaller blocks according to functionality. If ontology engineers would follow the same principle, the task of ontology engineering could become less cumbersome and less costly, due to an increase in reusable knowledge components, namely ontology modules.
At a certain point ontology modules are likely to require connections to other modules or ontologies to provide the functionality required for the given application. Techniques for linking ontologies and modules range from simple use of the *owl:imports* statement to fully fledged linking mechanisms [4, 2, 9]. These techniques differ in the assumptions they make regarding the source and target ontologies or modules. Some approaches require that the local domains and terminologies are disjoint [18], others require the use of special semantics. For an evaluation of their properties and usefulness in a distributed scenario as well as a more detailed introduction of their internal workings, see [30].
Which formalism the ontology engineer uses at the end to connect ontologies or modules is dependent on requirements of the specific application [20]. For our purposes it is just important to note that different formalisms exist, and modular ontologies can be created using different modules.

- **Lifecycle**: The support for evolution of knowledge, in particular for ontologies, is a major requirement for ontology repositories. In contrast to static content, ontologies are changing and demand mechanisms for updating and evolving knowledge over time. [27] defines an ontology evolution

process model based on a requirement analysis. The approach relies on a declarative specification of change requests and evolution strategies. The approach has been practically realized within the KAON [21] ontology engineering framework. Such evolution mechanisms can be included into GORF through so called *lifecycle components*.

- **Meta Model**: Ontologies are commonly used as a shared means of communication between computers and between humans and computers. Ergo ontologies should be represented, described, exchanged, shared and accessed based on open standards such as the W3C standardized web ontology language OWL. However, most ontologies today exist without any additional information about authorship, domain of interest and other metadata about ontologies. Therefore, searching and identifying existing ontologies which are potentially reusable because they are for example applied in similar domains, used within similar applications, or have similar properties, is a rather hard and tedious task.
  We argue that meta knowledge (seen as meta model) in the sense of machine processable information for the Web[11] helps to improve accessibility and reuse ontologies. Further, it can provide other useful resource information to support maintenance. We claim that metadata does not only help when it is applied (or attached) to documents, but also to ontologies. As a consequence, ontologies which are annotated by metadata require an infrastructure including metadata support—like the registry component in the GORF. Metadata simply consisting of attribute-value pairs is not sufficient for efficient knowledge access and reuse. Therefore first metadata vocabularies for ontologies have been developed [25, 13] and successfully applied in numerous applications [14]. In a more language centered way an approach relating UML with ontologies has been developed [5].
- **Validation**: Integrating different knowledge sources requires capable validation services. Validation components mainly analyze and validate the syntax of ontologies.
- **Registry & Indices**: Knowledge artifacts may exist as entities, modules, schemes, or ontologies as a whole. Those artifacts are indexed by a registry component (OMR -Ontology Metadata Registry).
  The OMR provides services for storage, cataloging, discovery, management, and retrieval of ontology metadata definitions. The OMR provides the means to support advanced semantic searches of ontologies based on their characteristics. In general the OMR can be a GORF component or an independent system.
  Analogically to the Dublin Core Metadata Initiative's (DCMI) Metadata Registry[12], the OMR is designed to promote the discovery and reuse of existing ontologies. It provides users, and applications, with an authori-

[11] c.f. `http://www.w3.org/Metadata/`
[12] http://dublincore.org/dcregistry/

tative source of information about the characteristics of ontologies, thus simplifying the discovery process.
Registry services and indices accelerate access to a repository, especially for search. Generally, indices are useful for concepts, relations, and full-text search capability. Our facilities offer repository administrators the possibility to freely define further indices.

Summarizing, the knowledge organization layer provides efficient methods for handling and organizing knowledge in repositories.

4.4 Knowledge Storage

To support the envisioned large, scalable application scenario of GORF, we use a highly scalable storage mechanism. Distributed repositories are set up in a cluster for handling several requests. Therefore the storage layer includes components for querying, transactions and replication of knowledge within such repositories.

- **Query Processor**: The query processing component handles queries for single knowledge artifacts in a repository. As query language we prefer standardized languages like the well-known SPARQL[13] query language.
- **Transaction & Consistency**: Performing and handling access to knowledge simultaneously requires sophisticated mechanisms preventing inconsistencies. Thus, transaction and consistency components analyze and check queries against the underlying knowledge storages.
- **Replication**: Being designed for scenarios with a high number of users and queries, GORF provides adaptable knowledge replication mechanisms. Those mechanisms replicate knowledge storages and additionally distribute them among pre-defined spaces.

The knowledge storage layer provides mechanisms for handling and accessing distributed knowledge sources, which are described below.

4.5 Knowledge Sources

The presented approach provides a sophisticated framework for integrating knowledge from different sources like *files, data bases, ontologies* or other *semantic portals*. The framework is capable of using existing sources along with their attached infrastructure. Therefore, our framework can rest atop existing technologies and act as a kind of semantic layer for these technologies to use the developed integration mechanisms.

Knowledge sources are typically distributed and heterogeneous, and tend to change during semantic interrelation, aggravating the task of integrating information into one common knowledge repository. The layer comprises two modules. The generic knowledge integration module shares and integrates

[13] c.f. `http://www.w3.org/TR/rdf-sparql-query/`

knowledge from previously unknown sources. The interconnected-integration module handles sources that are closely interconnected technically and semantically. This module mainly integrates content such as other portals and semantic metadata.

5 Ontology Repository Management Systems

An *Ontology Repository Management System (ORMS)* is a semantically-enabled software to *store, organize, modify* and *extract* knowledge from an Ontology Repository. Two systems, namely Oyster[14] and ONTHOLOGY[15] [14] are already available to the end user.

In general, the main tasks of an ORMS are providing access to knowledge resources, supporting retrieval and allocating sufficient management mechanisms.

Retrieval

Ontology retrieval for humans and machines is a key functionality of an ontology repository management system. The retrieval component provides mechanisms to manage search and discovery functions of an ontology repository. For example consider the allocation of indices or the provision of metadata.

Browsing and Navigation

Semantically-driven navigation through knowledge stocks enables users to identify new and potentially useful knowledge artifacts within a repository. The navigation through repositories can be guided by specialized ontologies for semantic navigation, as introduced in [15]. The *browsing and navigation* component therefore allows mainly the selection of such navigation ontologies. Based on a usage analysis [28], a repository manager is able to evolve deployed navigation ontologies.

Management

An ontology repository is administrated through a management component which contains all administrative functionalities required to store, organize and maintain the knowledge within a repository. In general, manageable components in a repository provide interfaces to the *management component*. The main task hereby is to collect all manageable functionalities and to enable a standardized and centralized access to all relevant administrative functionalities. The entire business logic is implemented within each repository component itself. Thus, the management component itself does not contain real business logic. As a result, components within GORF are easily interchangeable and the whole framework remains flexible and scalable.

[14] c.f. `http://oyster.ontoware.org/`
[15] c.f. `http://www.onthology.org/`

To sum up, an Ontology Repository Management System (ORMS) is a powerful tool to manage ontology repositories, even several distributed ones together. This way, established workflows and processes can be easily interchanged among other repositories reducing maintenance efforts and increasing the usability of such repositories.

6 Centralized vs. decentralized Systems

We now present exemplary running systems based on GORF. In detail, we present two complementary applications, namely the decentralized P2P system Oyster and the centralized ontology portal Onthology. In general, the two tools differ in their usage perspective and are appropriate for different tasks. However, as we will see, only the combined application of both tools will offer users the full potential of ontology management.

6.1 Centralized Systems

Ensuring a scalable and reliable access to ontologies, optimization techniques are required. One well-known approach is a hybrid storage mechanism from the data warehouse area which materializes content to provide faster access. We present the conceptual design of a centralized ontology portal and its implementation, so-called Onthology standing for "anthology of ontologies".

Scope

Centralized systems allow to reflect long-term community processes in which some ontologies become well accepted for a domain or community and others become less important. Such well accepted ontologies and in particular their metadata need to be stored in a central metadata portal which can be accessed easily by a large number of users whereby the management procedures are well defined. Hence, a main goal of a centralized metadata portal is to act as large evidence storage of metadata resp. their related ontologies to facilitate access, reuse and sharing as required for the Semantic Web.

Actors

We identified several different user roles for Onthology: The *visitor* is an anonymous user, he is allowed to browse the public content of the portal. A visitor can become a *user* by completing an application form on the website. In order to avoid unnecessary administrative work, a user is added automatically to the membership database. Users can customize their portal, e.g. the content of their start-page or their bookmarks. If a user wants to submit metadata to the portal, this submission has to be reviewed before it is published. Onthology establishes a *review process* in order to ensure a certain level of quality. *Reviewers* check the new submissions before they are published. The *technical administrator* is responsible for any other task mainly the maintenance of the portal.

Functionalities

Functionalities of ONTHOLOGY can be separated into two groups based on the usage. Indeed, *basic functionalities* which are provided to every user who accesses the portal and *sophisticated functionalities* for reviewers and administrators. The main operations a user can perform on the repository are (i) *Search*, (ii) *Submit* and (iii) *Export*.

The search and export can be performed by any visitor without being registered to the repository. Since providing new metadata is based on a certain community confidence, a visitor has to register at the portal to be able to submit data.

Architecture

ONTHOLOGY consists of an ontology repository and an ORMS. Exemplary, Sesame[16] or KAON[17] can be used as back-end metadata storage solution for an ontology-based representation. Furthermore, *access* and in particular the *management* of the repository must be guaranteed, too. Therefore, ONTHOLOGY is based on the proposed framework GORF. It supports queries to multiple sources, but beyond that also intensive use of the schema information itself to allow for automatic generation of navigational views such as navigation hierarchies that appear as `has- part`-trees or `has- subtopic` trees in the ontology. In addition to that mixed ontology and content-based presentation is supported. Further information can be found at [13, 25].

6.2 Decentralized Systems

In this section we describe the distributed ontology registry (Oyster).

Oyster[23] is a Peer-to-Peer application that exploits semantic web techniques in order to provide a solution for exchanging and re-using. In order to achieve this goal, Oyster implements the proposal for a metadata standard OMV[25] as the way to describe ontologies.

Oyster Design

The Oyster system[18] was designed using a service-oriented approach, and it provides a well defined API. Accessing the registry functionalities can be done using directly the API within any application, invoking the web service provided or using the included java-based GUI as a client for the distributed registry. As part of the design, Oyster identifies an ontology metadata entry by the URI of the ontology it describes, therefore two ontology metadata entries are considered the same when the URI of both ontologies are the same. However, due to the distributed nature and potentially large size of the

[16] `http://www.openrdf.org/`

[17] `http://kaon.semanticweb.org/`

[18] For a complete information and for downloading Oyster system we refer the reader to `http://ontoware.org/projects/Oyster/`

Peer-to-Peer network, two ontology metadata entries might refer to the same ontology but have different URI, in which case they are considered duplicates.

In Oyster, ontologies are used extensively in order to provide its main functions described in the following:

Creating and importing metadata: Oyster enables users to create metadata about ontologies manually, as well as to import ontology files and to automatically extract the ontology metadata available, letting the user fill in missing values. The ontology metadata entries are aligned and formally represented according to two ontologies: (1) the OMV ontology, (2) a topic hierarchy (i.e. the DMOZ topic hierarchy), which describes specific categories of subjects to define the domain of the ontology.

Formulating queries: A user can search for ontologies using simple keyword searches, or using more advanced, semantic searches. Here, queries are formulated in terms of these two ontologies. This means queries can refer to fields like name, acronym, ontology language, etc. or queries may refer to specific topic terms.

Routing queries: Users may query a single specific peer (e.g. their own computer, because they can have many ontologies stored locally and finding the right one for a specific task can be time consuming, or users may want to query another peer in particular because this peer is a known big provider of information), or a specific set of peers (e.g. all the members of a specific organization), or the entire network of peers (e.g. when users have no idea where to search), in which case queries are routed automatically in the network.

Processing results: Finally, results matching a query are presented in a result list. The answer of a query might be very large, and contain many duplicates due to the distributed nature and potentially large size of the P2P network. Such duplicates might not be exact copies because of the semi structured nature of the metadata, so the ontologies are used again to measure the semantic similarity between different answers and to remove apparent duplicates. As proposed by the ontology metadata standard, all the different realizations of an ontology (ontology documents) can be grouped by the same ontology base to give a more organized view of the results.

Oyster Architecture

The high-level design of the architecture of a single Oyster node in the Peer-to-Peer system is shown in Figure 2. In the following, we discuss the individual components of the system architecture.

The *Local Repository* of a node contains the metadata about ontologies that it provides to the network. It supports query formulation and processing and provides the information for peer selection. In Oyster, the Local Repository is based on KAON2 and it supports SPARQL as its query language.

The *Knowledge Integrator* component is responsible for the extraction and integration of knowledge sources (i.e. ontologies) into the Local Repository. Oyster supports automatic extraction of metadata for OWL, DAML+OIL,

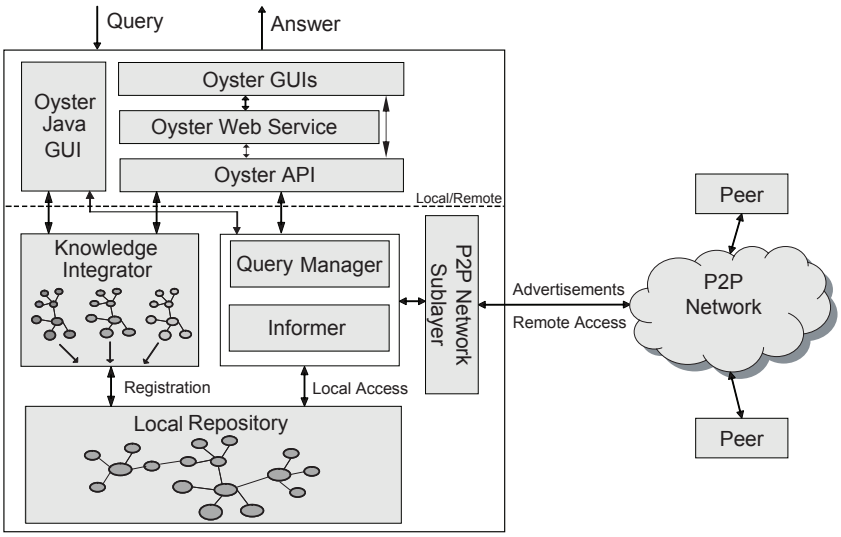

Fig. 2. Overview of Oyster Architecture

and RDF-S ontology languages. This component is also in charge of how duplicate query results are detected and merged.

The *Query Manager* is the component responsible for the coordination of the process of distributing queries. It receives queries from the user interface, API or from other peers. Either way it tries to answer the query or distribute it further according to the content of the query. The decision to which peers a query should be sent is based on the scope of the query (i.e. a specific set of peers or entire network) and optionally on the knowledge about the expertise of other peers.

The *Informer* component is in charge of proactively advertising the available knowledge of a Peer in the Peer-to-Peer network and to discover peers along with their expertise. This is realized by sending advertisements about the expertise of a peer. In Oyster, these expertise descriptions contain a set of topics (i.e. ontology domains) that the peer is an expert in. Peers may accept these advertisements, thus creating a semantic link to the other peer. These semantic links form a semantic topology, which is the basis for intelligent query routing.

The *Peer-to-Peer network sub-layer* is the component responsible for the network communication between peers. It provides communication services for the data exchange with remote nodes, i.e. to propagate advertisement messages and to realize the access to remote repositories. In Oyster, we rely on an RMI-based implementation, however, other communication protocols would be possible as well.

The API, WS and GUI components provide alternative ways for accessing Oyster functionalities. The API defines a set of methods that expose all the

registry functionalities, while the web service encapsulates the API exposing only a reduced subset. The GUIs provide ready-to-use clients that access the registry via the WS or the API. However, Oyster also include a former java-based GUI that access directly the registry.

Additional registry functionalities can be provided by engineering components. Some of these components are described in [24].

6.3 Discussion

Both presented applications are covering a variety of different tasks. Indeed, for a user who wants to store metadata individually similar to managing his personal favorite song list, a repository is required to which a user has full access and can perform any operation (e.g. create, edit or delete metadata) without any consequences to other users. Exemplary, users from academia or industry might use a personal repository for a task-dependent investigation, or ontology engineers might use it during their ontology development process to capture information about different ontology versions. We argue, that a decentralized system is the technique of choice, since it allows the maximum of individuality while it still ensures exchange with other users.

Centralized systems allow to reflect long-term community processes in which some ontologies become well accepted for a domain or community and others become less important. Such well accepted ontologies and in particular their metadata need to be stored in a central metadata portal which can be accessed easily by a large number of users whereby the management procedures are well defined. Obviously, personal repositories are quite limited from this perspective. Actually, the Oyster system and Onthology are not necessarily two completely separated repositories. Indeed, they are interconnected and they exchange metadata between each other. We are currently supporting the access of metadata stored in Onthology from any Oyster peer.

The benefit of connecting both systems lies mainly in the simple use of existing ontology metadata information within Oyster. So, while users are applying or even developing their own ontologies they can manage their own metadata along with other existing metadata in one application (in Oyster). If some metadata entries from Oyster have reached a certain confidence, an import into Onthology can be performed easily. In combination, both systems ensure efficient and effective ontology metadata management for various use cases.

7 Conclusions

Ontology repositories will be a crucial cornerstone facilitating efficient knowledge access and reuse especially in the context of the Semantic Web. We have presented our Generic Ontology Repository Framework GORF including rating and module support. We expect that there will be a shift in ontology engineering towards developing ontologies in a modular way. We are optimistic that then the critical mass of ontology modules in our repository can

be reached, and ontology engineers will start reusing them and providing new ones.

Already existing realizations like ONTHOLOGY and Oyster illustrate the benefits of such systems. We assume that ontology repositories will play an important role in realizing the Semantic Web vision.

Acknowledgement. Research reported in this chapter was partially supported by two European Projects: The Network of Excellence KnowledgeWeb (FP6-507482) and the NeOn Project (FP6-027595).

References

1. W. Y. Arms. *Digital Libraries*. The MIT Press, 2001.
2. J. Bao, D. Caragea, and V. Honavar. Towards collaborative environments for ontology construction and sharing. In *International Symposium on Collaborative Technologies and Systems (CTS 2006)*, pages 99–108. IEEE Press, 2006.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American Magazine*, 284(5):34–43, 2001.
4. A. Borgida and L. Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In *OTM Federated Conference CoopIS/DOA/ODBASE*, pages 36–53, 2002.
5. S. Brockmans, R. M. Colomb, E. F. Kendall, E. Wallace, C. Welty, G. T. Xie, and P. Haase. A Model Driven Approach for Building OWL DL and OWL Full Ontologies. In I. C. et al., editor, *The Semantic Web - ISWC 2006: 5th International Semantic Web Conference*, volume 4273 of *LNCS*, pages 187–200, Athens, Ga, USA, NOV 2006. Springer.
6. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the Right Amount: Extracting Modules from Ontologies. In *Proc. of the Sixteenth International World Wide Web Conference (WWW 2007)*, 2007.
7. M. d'Aquin, M. Sabou, and E. Motta. Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components. In *1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC*, 2006.
8. Y. Ding and D. Fensel. Ontology library systems: The key to successful ontology reuse, 2001.
9. B. C. Grau, B. Parsia, and E. Sirin. Working with multiple ontologies on the semantic web. In *International Semantic Web Conference*, pages 620–634, 2004.
10. R. Guha. Open Rating Systems. Technical report, Stanford University, CA, USA, 2003.
11. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *Proc. of the Thirteenth International World Wide Web Conference*, pages 403–412, New York, NY, MAY 2004. ACM Press.
12. J. Hartmann. *Ontology-based Modeling and Realization of Knowledge Management Systems*. PhD thesis, University of Karlsruhe (TH), Institute AIFB, D-76128 Karlsruhe, 2007.
13. J. Hartmann, R. Palma, Y. Sure, P. Haase, and M. C. Suárez-Figueroa. OMV – Ontology Metadata Vocabulary. In C. Welty, editor, *ISWC 2005 Workshop on Ontology Patterns for the Semantic Web*, NOV 2005.

14. J. Hartmann, R. Palma, Y. Sure, M. C. Suárez-Figueroa, P. Haase, A. Gómez-Pérez, and R. Studer. Ontology metadata vocabulary and applications. In *International Conference on Ontologies, Databases and Applications of Semantics. In Workshop on Web Semantics (SWWS)*, LNCS 3762, pages 906–915. Springer, OCT 2005.
15. J. Hartmann and Y. Sure. An Infrastructure for Scalable, Reliable Semantic Portals. *IEEE Intelligent Systems*, 19(3):58–65, MAY 2004.
16. R. Heery and S. Anderson. Digital repositories review, February 2005.
17. W. H. Inmon and W. H. Inmon. *Building the Data Warehouse, 3rd Edition*. John Wiley & Sons, Inc., New York, NY, USA, 2002.
18. O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of description logics. In *Description Logics Workshop, CEUR-WS Vol 81*, 2003.
19. H. Lewen, K. Supekar, N. F. Noy, and M. A. Musen. Topic-Specific Trust and Open Rating Systems: An Approach for Ontology Evaluation. In *Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web (EON2006) at the 15th International World Wide Web Conference (WWW 2006)*, Edinburgh, UK, MAY 2006.
20. F. Loebe. Requirements for logical modules. In *1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC*, 2006.
21. A. Maedche, B. Motik, and L. Stojanovic. Managing Multiple and Distributed Ontologies in the Semantic Web. *VLDB Journal*, 12(4):286–302, 2003.
22. N. F. Noy, R. Guha, and M. A. Musen. User ratings of ontologies: Who will rate the raters? In *Proc. of the AAAI 2005 Spring Symposium on Knowledge Collection from Volunteer Contributors*, Stanford, CA, 2005.
23. R. Palma and P. Haase. Oyster - sharing and re-using ontologies in a peer-to-peer community. In *International Semantic Web Conference*, pages 1059–1062, 2005.
24. R. Palma, P. Haase, Y. Wang, and M. d'Aquin. D1.3.1 Propagation models and strategies. Technical report, Universidad Politécnica de Madrid, NOV 2007.
25. R. Palma, J. Hartmann, and P. Haase. OMV - Ontology Metadata Vocabulary for the Semantic Web. Technical report, Universidad Politécnica de Madrid, University of Karlsruhe, 2008. v2.4. Available at `http://omv.ontoware.org/`.
26. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modeling and design*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1991.
27. L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, Universität Karlsruhe (TH), Universität Karlsruhe (TH), Institut AIFB, D-76128 Karlsruhe, 2004.
28. N. Stojanovic, J. Hartmann, and J. Gonzalez. The OntoManager - a system for usage-based ontology management. In *Proceedings of FGML Workshop. Special Interest Group of German Information Society (FGML - Fachgruppe Maschinelles Lernen der GI e.V.)*, 2003.
29. D. Vrandecic and Y. Sure. How to design better ontology metrics. In W. May and M. Kifer, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, Innsbruck, Austria, JUN 2007. Springer. to appear.
30. Y. Wang, J. Bao, P. Haase, and G. Qi. Evaluating Formalisms for Modular Ontologies in Distributed Information Systems. In M. Marchiori and J. Z. Pan, editors, *Proceedings of The First International Conference on Web Reasoning and Rule Systems (RR2007)*, LNCS 4524, pages 178–182, Innsbruck, Austria, JUN 2007. Springer.

Index