STUDIA INFORMATICA 2009
Volume 30 Number 2A (83)

Piotr PIOTROWSKI
Politechnika Gdańska, Katedra Inżynierii Oprogramowania

THE INTERNET AS A KNOWLEDGE SOURCE IN KNOWLEDGE VIEWS[1]

Summary. This paper presents how the Knowledge Views concept can be applied to gather information from various sources, reason over such information and present the knowledge to the user application in a form required by that application. Using Knowledge Views an application can "see" the Internet as a single knowledge source.

Keywords: knowledge base, ontology

INTERNET JAKO ŹRÓDŁO WIEDZY W WIDOKACH NA BAZĘ WIEDZY

Streszczenie. Artykuł ten przedstawia koncepcję widoków na bazę wiedzy, jak może ona zostać wykorzystana w celu zbierania informacji z różnych źródeł, wnioskowania nad tymi informacjami i udostępnienia ich w postaci wymaganej przez aplikację. Korzystając z widoków na bazę wiedzy, aplikacja może „widzieć" Internet jako pojedyncze źródło wiedzy.

Słowa kluczowe: baza wiedzy, ontologia

1. Introduction

Before going into detail some key terms need to be clarified, since they can be understood differently. First of all what is a knowledge source? The term knowledge has no single definition. For example it is not uncommon to encounter knowledge bases, while browsing

[1]This work has been funded by Polish Ministry of Research and Higher Education, project No. N N516 4115 33.

the Internet that are actually repositories of documents on some particular subject. For example a product knowledge base on some commercial website like Intel's. In this case knowledge is considered the overall information present in a collection of books, articles, leaflets, etc. On the other hand, the Semantic Web circles correlate the term knowledge with reasoning. Let's consider the following definition of a knowledge base: a knowledge base is a repository of facts that has the capacity of providing new facts derived from the ones already stored according to some rules of inference. This definition, however, is a bit too general. For example a query defining a database view can be considered an inference rule and the view itself provides new facts derived from the existing ones. To amend this problem an addition stating that the inferred facts should be on par with the source ones and therefore take part in further reasoning process to produce additional conclusions. Following the database comparison, it would mean the view could be recursive adding new rows until there is nothing new to add. To obtain a knowledge source definition from the knowledge base one it is enough to substitute the "repository of facts" expression with "provider of facts". In contrast to a base a source does not have to be able to store facts. The source can be read only. More over the provided facts do not have to be physically stored, but might be as well calculated on the fly. These definitions are quite general. This work, however, focuses only on knowledge bases based on Description Logic with the addition of rules, thus covering SWRL [9] expressivity.

2. The Knowledge Views concept

What are the Knowledge Views? The idea started as a way to allow data-based components to communicate with knowledge-based ones [5, 12]. According to this idea data-based components would interact with a knowledge source via an additional layer (Data View) which provides an SQL interface. This concept, however, has been extended to include an object interface (Object View) [13], since today's programmers mainly use object oriented programming languages. There are some object interfaces to some knowledge bases in existence, for example Jastor [16] or Elmo [11], however they are usually dedicated to a particular knowledge base vendor and can not be used with any other. The proposed solution [13], tries to be general enough, to be able to have any knowledge source as a back-end.

The work on two kinds of views led to the extraction of some common framework (Knowledge View) which facilitates the addition of virtually any new interface. It also allows different kinds of sources: knowledge base-like (for example Jena) or a database, enhancing them semantically if necessary. To sum up the Knowledge Views are a piece of software that

enables an application to see a knowledge source or a part of it in a way required by this application. It might be compared to database views combined with some mapping library like JPA [4], which performs object-relational mapping.

2.1. The Knowledge Views architecture

The development of the Knowledge Views concept, as well as implementation, led to the adoption of a layered architecture (see Fig. 1). The architecture has been designed in a way allowing the individual components to be used separately or in different configurations if necessary.

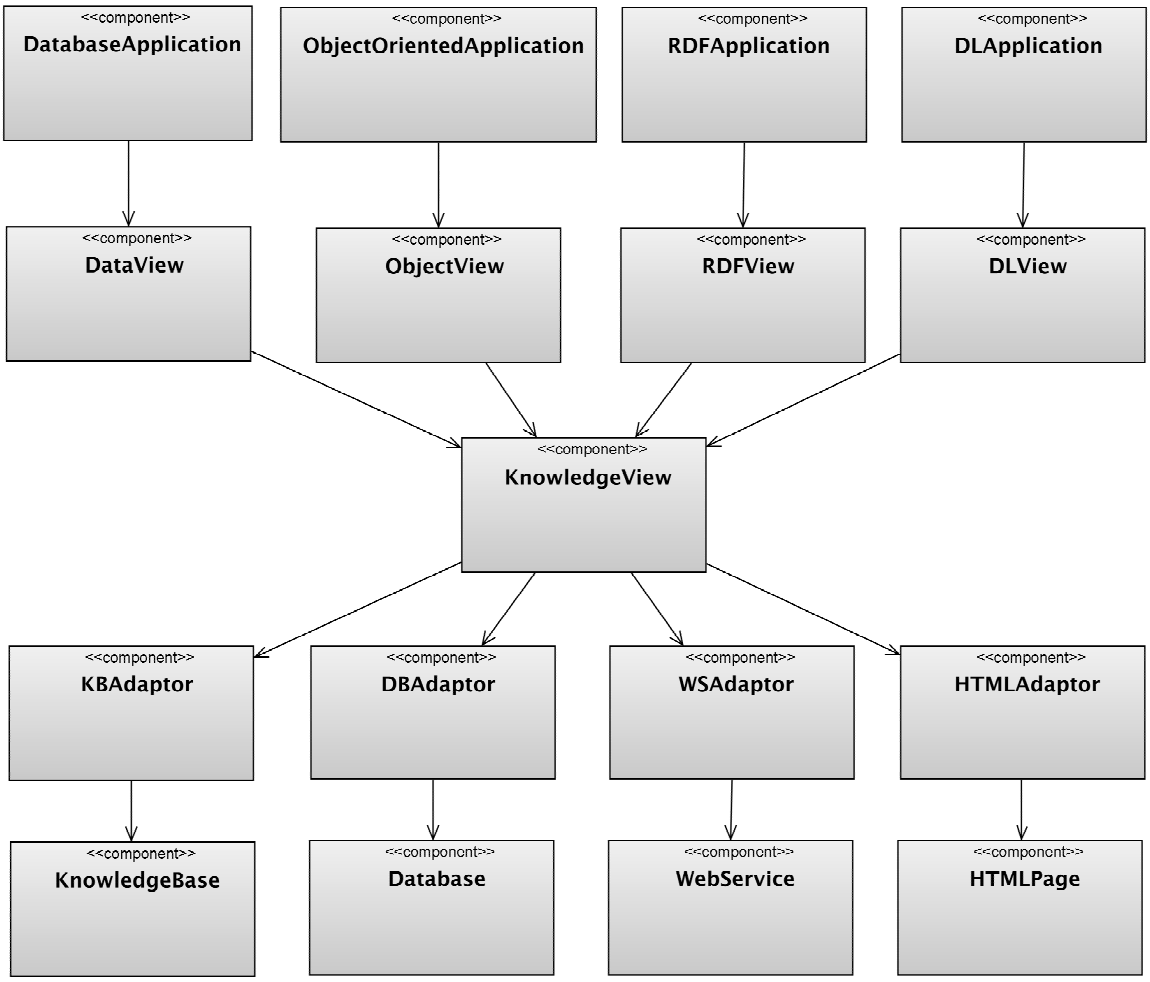

Fig. 1. The Knowledge Views architecture
Rys. 1. Architektura widoków na bazę wiedzy

2.1.1. Information sources

The bottom layer in this architecture consists of various information sources:

- knowledge bases,

- databases,
- web services,
- HTML pages,
- …

Information sources can be divided into two groups. First group consists of sources designed for query answering, like knowledge bases or databases. The other group does not have such capabilities, therefore retrieving arbitrary information can be a lengthy process. Documents stored as HTML pages are an example of such sources. To enhance the search performance for such sources some kind of indexing is required.

2.1.2. Adaptors

The second layer are adaptors. Since different information sources are accessed in a different manner, there is a need for a unified interface that is used by the upper layer. The interface consists mainly of the following methods:

- getConcepts,
- getRoles,
- getAttributes,
- getConceptInstances,
- getRoleInstances,
- getAttributeInstances.

The first three can be considered helper methods which can be used to retrieve the concepts, roles and attributes that the source supports. The rest are the core methods that provide the facts. These methods take a concept, role or attribute as the first argument and an instance of a concept, role or attribute as the second argument. The second argument can be null to denote a wild card or if the instance is a pair either of the elements can be null. This follows the query by example concept. Apart from the listed methods the adaptors can implement optional methods for fact insertion and deletion. In case of the sources that do not provide any query language, the adaptor may provide some indexing capabilities to enhance performance. This, however, is transparent to the upper layer.

2.1.3. Knowledge View

The Knowledge View component is the core of the system. It consists of several tools. The tools are designed to:

- gather facts from several sources,
- select and modify facts from the source using a rule based query language,
- semantically enhance the set of facts.

The first function provides means to join several sources into one logical source. The resulting source can be therefore distributed and heterogeneous. The second function helps selecting only the required facts from the source. It is also possible to alter the facts if they do not meet the requirements of a particular application, for example temperature is given in Fahrenheit scale while Kelvin scale is required. This is done using a rule based query language with some built-in predicates like in SWRL [9]. The selection and manipulation of facts can be compared to database views where rows can be selected and manipulated with a SQL query.

The last function provides capabilities to add some semantics to the raw set of facts. That is add a DL terminology optionally accompanied by rules. The terminology, however, is not provided as a file. Such a solution would be troublesome, because of the many ways to encode a terminology. Instead the terminology is provided as an object implementing a method that can answer whether one concept subsumes the other. The object has also some supplementary methods. The object's implementation embeds some inference engine that performs terminological reasoning. Such a solution hides both the terminology format as well as the reasoner implementation. Having a TBox together with an ABox and optionally a set of rules, one can perform reasoning over the set of facts. This approach is similar in concept to Knowledge Layer [7], however it is more general both in terms of the information sources as well as the DL reasoner used.

2.1.4. Model mapping

On top of the Knowledge View layer there is a model mapping layer. This layer can be compared to JAXB [10] or JPA [4]. Its role is to map ontological entities to entities of the appropriate model. More over each model is accompanied by a dedicated query language.

This layer is designed to ease the usage of knowledge sources. It automates some actions that need to be performed in most applications that use external information sources. However, care has to be taken, since various model types may differ greatly. This is called model mismatch. The mismatch may reveal itself in differences in data types, but differences in modeling conventions may be much more problematic. The same domain may be modeled differently by a relational model, object model and ontological model. This is because each model type has been devised to solve different class of problems. Relational model used in databases supports fast information retrieval. Object model, as opposed to relational one was designed with dynamic systems in mind, that is systems whose entities have state that can change over time. These entities can perform actions that are influenced by the state of the entity. In ontological model the focus is on logic and inference. As a result in an application each model is used in a different layer (see Fig. 2) and only some information is passed between those models so that each of them could fulfil its purpose.

In the proposed solution the model mappings themselves are simple and the model mismatch is handled using Knowledge View that has model manipulation capabilities. The user knowing the model of the knowledge source can adjust it using Knowledge View to his needs, then the resulting ontological model is mapped to a different kind of model depending on the paradigm chosen for the application.

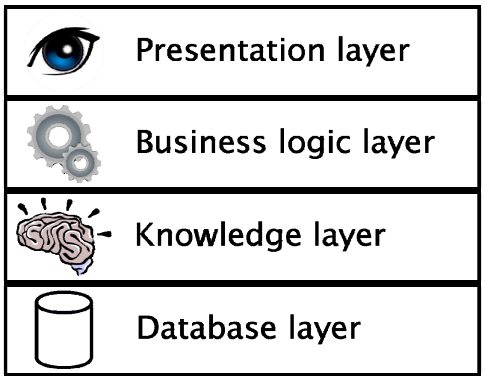

Fig. 2. Sample layered application
Rys. 2. Przykładowa aplikacja warstwowa

2.1.4.1. Data View

The Data View maps the ontological model to relational one. Thanks to this view the application can use a knowledge source as a database issuing SQL queries. The mapping complies to the following rules:

- concepts are mapped to entity sets,
- roles are mapped to relationships,
- attributes are mapped to attributes,
- the subsumes relationship is mapped to IS_A relationship.

2.1.4.2. Object View

The Object View maps the ontological model to object one. This view represents the ontology individual as an object. The wide usage of object oriented programming languages makes this view especially useful. This view also provides query capabilities through OQL [3]. The sample implementation of the Object View concept supports only a small portion of OQL which is a proof of concept rather than a production ready solution. The ontology model is mapped to object one according to the following rules:

- concepts are mapped to classes,
- roles are mapped to associations,
- attributes are mapped to attributes,
- the subsumes relationship is mapped to inheritance.

2.1.4.3. RDF View

RDF [2] is one of the knowledge representation forms. It is recommended by W3C, therefore it is very popular in Semantic Web circles. The RDF View gives the user the

possibility to treat the knowledge source as an RDF repository supporting SPARQL [14] as a query language. The way of encoding DL ontology in RDF form is defined by W3C.

2.1.4.4. DL View

The Knowledge View is designed in conformance to Description Logic, therefore could be itself considered a DL View. However, the interface provided by the Knowledge View is very simple. On the other hand available reasoners, like Pellet [15], provide DIG [1] interface. Therefore the DL View role would be not to provide any mapping, but to provide query capabilities through DIG interface.

2.1.4.5. More?

It possible to add even more interfaces or mappings on top o the Knowledge View layer if needed. The architecture has been designed with extensibility in mind: new information source as well as interfaces can be easily added using the existing framework.

2.1.5. Application layer

The proposed solution provides the application programmer with the means to easily integrate the power of reasoning into his application by using knowledge sources through a familiar interface like SQL, OQL or SPARQL depending on the application. This way the programmer can focus on writing business logic rather then dealing with data access problems and learning new interfaces.

2.2. Rationale

The choice of the approach, where first only the facts are manipulated and only later a terminology is added, has been made for several reasons. First of all, the Internet is full of information sources, while there are comparatively few knowledge sources. More over a knowledge source is also an information source. Secondarily it is much easier to operate on raw facts. Lastly if a knowledge base is to play a similar role in software engineering as databases nowadays, then terminologies should be created for particular narrow applications rather then trying to encompass some general concepts as upper ontologies like GFO [8] do. When a database application is built to solve some particular problem, the database schema is created that supports fast retrieval of the relevant data. Anything else is ignored and is not allowed to be stored in the database. Similarly this approach assumes that ontologies are created on demand. This means that having a particular problem and a set of source facts, an ontology is built from which facts that are required for the problem to be solved can be inferred. Anything else is expendable and usually only unnecessarily complicate the solution.

Let's assume a knowledge base is to be used as a subsystem of a greater system. In such a system every subsystem has its own internal model. The models do not have to correspond

with each other. However there has to be some kind of mapping between them, to allow information exchange. This might be compared to two people speaking to each other. They exchange information, but both have different ontologies in their minds and therefore can interpret the same facts differently and even draw different conclusions from them. In such a case only the facts are exchanged not the terminology.

3. The Internet as a knowledge source

The Internet is a vast source of information. It would be beneficial to take advantage of it. However there are some problems. First of all the Internet is not a single source but rather a collection of many sources. Therefore there is a need for gathering information from various sources. The sources do not have a single interface and the information can be encoded in a variety of ways. Some have query languages, while other require the application to search it through. Information in some sources can be contradictory. The list of problems is quite long. Some of the problems like heterogeneity can be dealt with and are dealt with in Knowledge Views software. However some problems, especially those concerning false information and deciding what is important and what can be discarded, are hard even for humans and still require human supervision when it comes to computer programs.

The Knowledge Views concept and implementation tries to provide the application programmer with some tools supporting him in using the Internet as a knowledge source. The main goals are for the software to be easy to use and to require as little additional training of the contemporary software engineer as possible.

4. Summary

The Knowledge Views concept combines several ideas into a single framework. It has been developed looking at the problem from the software engineer point of view. It focuses on generalization and simplicity to be useful in a wide range of applications, and easily adapted to new requirements. The initial versions of the Data View and the Object View have already been used as parts of the KMS system [6] which was developed for the PIPS (Personalised Information Platform for Life and Heath Services [17]) project funded by the European Commission under the Framework 6 call. The purpose of the KMS system was to manage knowledge used to support decision making concerning health care. The Data and Object Views were the intermediate layers between a knowledge base and a web portal. The

current development of both the concept and the software is influenced by the experience from this project.

BIBLIOGRAPHY

1. Bechhofer S.: The DIG Description Logic Interface: DIG/1.1. Proceedings of DL 2003 Workshop, 2003.
2. Beckett D.: RDF/XML Syntax Specification. W3C, 2004.
3. Cattell R. G. G., Barry D. K., Berler M., Eastman J., Jordan D., Russell C., Schadow O., Stanienda T., Velez F.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann, 2000.
4. DeMichiel L., Keith M.: Java Persistence API. JSR 220: Enterprise JavaBeans, Version 3.0. Sun Microsystems, 2006.
5. Goczyła K., Grabowska T., Waloszek W., Zawadzki M.: Designing world closures for knowledge-based system engineering. Software engineering: evolution and emerging technologies, IOS Press, 2005, p. 271÷282.
6. Goczyła K., Waloszek W., Zawadzka T., Zawadzki M.: Inference mechanisms for knowledge management system in e-health environment. Software engineering: evolution and emerging technologies, IOS Press, 2005, p. 418÷423.
7. Goczyła K., Zawadzka T., Zawadzki M.: Wnioskowanie z danych zapisanych w zewnętrznych źródłach w systemie zarządzania wiedzą. Bazy danych. Nowe technologie - Architektura, metody formalne i zaawansowana analiza danych, Wydawnictwo Komunikacji i Łączności, 2007, p. 283÷293.
8. Herre H., Heller B., Burek P., Hoehndorf R., Loebe F., Michalek H.: General Formal Ontology (GFO): A Foundational Ontology Integrating Objects and Processes. Part I: Basic Principles. Research Group Ontologies in Medicine, University of Leipzig, 2007.
9. Horrocks I., Patel-Schneider P. F., Boley H., Tabet S., Grosof B., Dean M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C, 2004.
10. Kawaguchi K., Vajjhala S., Fialli J.: The Java Architecture for XML Binding (JAXB) 2.1. Sun Microsystems, 2006.
11. Leigh J.: Elmo User Guide. http://www.openrdf.org/doc/elmo/1.0-beta2/user-guide/.
12. Piotrowski P.: Implementacja widoków danych na bazę wiedzy. TPD 2007: II Krajowa Konferencja Naukowa Technologie przetwarzania danych, Wydawnictwo Politechniki Poznańskiej, 2007, p. 174÷185.

13. Piotrowski P.: Object Views - metoda mapowania obiektowo-ontologicznego. Bazy Danych. Rozwój metod i technologii – Bezpieczeństwo, wybrane technologie i zastosowania, Wydawnictwa Komunikacji i Łączności, 2008, p. 207÷216.
14. Prud'hommeaux E., Seaborne A.: SPARQL Query Language for RDF. W3C, 2007.
15. Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics Vol. 5, No. 2, Elsevier 2007.
16. Szekely B., Betz J.: Jastor – Typesafe, Ontology Driven RDF Access from Java. http://jastor.sourceforge.net/.
17. Personalised information platform for life and heath services. http://www.pips.eu.org/.

Recenzent: Dr inż. Małgorzata Bach

Wpłynęło do Redakcji 31 stycznia 2009 r.

Omówienie

Widoki na bazę wiedzy stanowią warstwę pomiędzy źródłem wiedzy a aplikacją z niego korzystającą. Zadaniem tej warstwy jest zebranie informacji z jednego lub wielu źródeł, wzbogacenie tych informacji o terminologię i ewentualnie reguły, aby można było wyciągać wnioski z zebranych faktów, a następnie przekazanie ich do aplikacji w postaci przez nią wymaganej. Dzięki widokom na bazę wiedzy aplikacja może korzystać zarówno z bazy wiedzy, jak i z bazy danych przez interfejs SQL albo też z obiektowej bazy danych, mając do dyspozycji OQL.

Rozwijając ten pomysł, starano się patrzeć na problem z punktu widzenia programisty, a nie inżyniera wiedzy, aby ułatwić korzystanie z baz wiedzy programistom, którzy mogą mieć małą wiedzę na temat Semantic Web. Rozwój pomysłu wraz z implementacją doprowadziły do stworzenia warstwowej architektury (patrz rys. 1). Architekturę tę można rozszerzać, dzięki czemu można dodać do niej kolejne komponenty, realizujące dostęp do nowych źródeł informacji oraz dostarczające nowych interfejsów dla aplikacji.

Stworzony zestaw narzędzi umożliwia spojrzenie na Internet jak na źródło wiedzy. Internet dostarcza wiele niezależnych źródeł informacji, lecz korzystając ze stworzonych narzędzi możliwe jest połączenie kilku źródeł w jedno, wybranie tylko tych informacji, które są istotne, wzbogacenie semantyczne stworzonego źródła, a następnie dostarczenie zebranych oraz wywnioskowanych faktów aplikacji w dogodny dla niej sposób.

Address

Piotr PIOTROWSKI: Politechnika Gdańska, Katedra Inżynierii Oprogramowania,
ul. Narutowicza 11/12, 80-233 Gdańsk, Polska, piopio@eti.pg.gda.pl.