A Framework for Reengineering Web Applications to Web Services

BOUCHIHA Djelloul [1], MALKI Mimoun [2], Mostefai Abd El Kader [3]

[1] EEDIS Laboratory, University of Sidi Bel Abbes 22000, Algeria
bou_dje@yahoo.fr
[2] EEDIS Laboratory, University of Sidi Bel Abbes 22000, Algeria
malki_m@univ-sba.dz
[3] EEDIS Laboratory, University of Sidi Bel Abbes 22000, Algeria
mostefai_aek@univ-sba.dz

Abstract. Web services technology and Service-Oriented Architectures (SOA) are rapidly developing and widely supported. However, it is fairly difficult for existing Web applications to expose functionality as services in a service-oriented architecture, because when Web applications were built, they served as monolithic systems. This paper describes a framework called WA2WS, which can be used for constructing Web Services from existing Web applications. This framework consists of two phases. First, an abstraction phase which consists in extracting UML conceptual schema from a Web application using domain ontology. Second, an implementation phase which consists in generating the JAVA code of Web service from the UML conceptual schema using mapping rules.

Keywords

Reengineering, Web Services, Service-Oriented Architectures (SOA), Web applications, Ontology, UML.

(Received September 15, 2007 / Accepted January 04, 2008)

1. Introduction

The World Wide Web is rapidly being adopted as the medium of collaboration among organizations. Web applications are today legacy systems, which constitute valuable assets to the organizations that own them. A Web application is an application delivered to users from a Web server over networks such as the Internet or an intranet. Web applications are popular due to the ubiquity of the Web browser as a client [7].

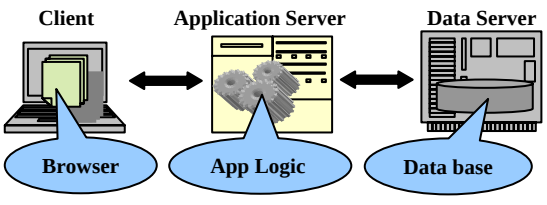

Figure 1: Three-tier model of Web application architecture.

A Web application is commonly structured as a three-tiered application as shown in Figure 1. In the most common form of the three-tier Web application model, the Web browser is the first tier, an engine using some dynamic Web content technology (such as CGI, PHP, JSP or ASP) is the middle tier, and a database is the third tier. The Web browser sends requests to the middle tier, which serves its client by making queries and updates against the database and by generating a user Interface (HTML responses) [7].

On the other hand, Web services technology is rapidly developing and widely supported. It consists of a set of related specifications that define how components should be specified (through the Web-Service Description Language – WSDL), how they should be

advertised so that they can be discovered and reused (through the Universal Description, Discovery, and Integration API – UDDI), and how they should be invoked at run time (through the Simple Object Access Protocol API – SOAP).

Web services are based on Service-Oriented Architectures (SOA), which is the keystone of service oriented computing. SOA includes some architectural components, such as service providers, service consumers and service repository. All the service usage, such as delivery, acquisition, consumption, composition and so on, is based on this architecture. SOA is an important paradigm that supports service management. It is an architecture evolution, and it affects the software life cycle from the service point of view. SOA is particularly applicable when multiple applications running on varied technologies and platforms have to communicate with each other [5].

This situation necessitates the development of automated reengineering methods for constructing Web services out of existing functionalities already offered through Web application of organizations today.

2. Related Work

Many approaches were proposed to revitalise Web applications in network environment with service-oriented technology:

Eleni and al. present a general method for constructing wrappers for Web-based applications, so that they exchange data with shared semantics such as defined in the XML domain model [2].

Yingtao and al. choose to reverse-engineer the presentation layer of the Web application, in order to extract from its behaviour a set of functionalities. The extracted functionalities can then be specified in terms of WSDL Web-service specifications, and they can be deployed through proxies accessing the original Web server and parsing its responses [10].

Jianzhi and al. propose a Grid services-oriented reengineering approach to create stateful resources from conventional HTML Web sites, which applies hierarchical cluster and wrapper techniques to extract and translate Web sites resources. It supports services identification and packaging and archives Web site evolution into Grid services environment by exploiting Web Service Resource Framework (WSRF) [6].

Hoang and al. propose a mechanism to wrap existing CGI-based Web sites in Web services. These services inherit all features from the sites while can be enriched with other Web service features like UDDI publishing, semantic describing, etc [4].

Robert and al. propose an integration approach, which consists in exploiting Web application interface, and converting HTML responses documents to XML documents. Wrapper technology is used for extracting appropriate information from HTML documents and translating this information to XML documents, which can be treated later automatically [8].

Michiaki and al. propose a framework called H2W, which can be used for constructing Web Service wrappers from existing multi-paged Web applications. H2W's contribution is mainly for service extraction, rather than for the widely studied problem of data extraction [7].

The described approaches above can be classified according to two criterions: either by the analysed element in input (interface or source code), or by the generated element at output (Wrapper, new Web service or other). With the first criterion [2], [6], [7], [8] and [10] analyze the interface, i.e., analyse HTML responses documents of HTTP requests and not the source code of the Web application. However, [4] analyses the source code (CGI queries) of the Web application. With the second criterion which is the generated element at output [2], [4], [6], [7] and [8] generate a Wrapper to wrap the Web application as a Web service. Whereas,

[10] generates WSDL specification, which can be exploited to use the Web application as a Web service.

Web applications need to undergo a sequence of preliminary activities to evolve toward Web services. In our work, these activities may be conceived as the cascade of two phases: an abstraction phase centered around a preliminary reverse-engineering activity. Followed by an implementation phase, i.e., a sequence of forward engineering steps leading to the new Web service.

In fact, we need a preliminary conceptualization phase; during this phase we build an abstract conceptual schema, a high level representation encompassing the existing Web application abstraction using domain ontology. We recover an UML conceptual schema. The recovered schema is now being used to create a new Web service offering the same functionalities as the Web application; implementing this Web service is the task of the forward engineering phase.

3. Proposed framework

There are two essential challenges for reengineering Web applications towards Web services. First is extracting the logical data for the machines from data decorated with HTML for human readers. Second is extracting a noninteractive service for machines from interactive services scattered over multiple Web pages for humans [7].

In this paper we propose a framework called WA2WS for constructing Web Services from existing Web applications. We regard our framework as mainly for data extraction, because many of the Web applications around us are data-intensive, where the main purpose of the application is to present a large amount of data to their users [9]. Our goal is to migrate an existing Web application to new Web Service by combining a reverse-engineering approach first and a forward engineering approach after (Figure 2).

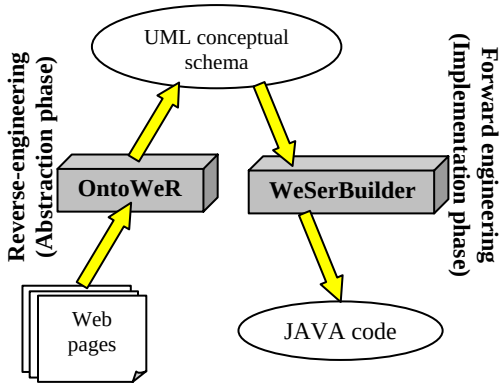

Figure 2: WA2WS framework.

OntoWeR: is a software supporting an Ontology based Web Reverse-engineering approach, covering the abstraction phase by recovering a presentation schema using domain ontology. The presentation schema is stored through UML language.

WeSerBuilder (Web Service Builder) is a CASE (Computer Aided Software Engineering) tool covering the implementation phase (Web service forward engineering) by generating a JAVA code from the UML conceptual schema using mapping rules (between conceptual level and logical level).

4. Abstraction phase

Bouchiha and al. propose a new approach for reverse-engineering Web applications. The approach aims to generate an UML conceptual schema modeling the Web application. The major contribution of this approach is the use of ontology[1] in the abstraction process [1].

The intuition underlying this approach is: an UML conceptual schema is hidden under the user interface of a Web application. This interface exposes HTML forms to their users' browsers, possibly enhanced with client-side scripts in different languages. The user has to appropriately interpret the semantics of the information required by the form and to fill it out correctly. Then, the server application responds with another HTML

[1] Ontology is an explicit specification of a conceptualization [3].

document containing usually tables and lists that the user can interpret as an answer to his original request.

Ontology based Web Reverse-engineering approach consists of three successive phases (Figure 3): First is the extraction of useful information from HTML pages. Second phase is the analysis of the extracted information using domain ontology. Last phase is the generation of corresponding UML conceptual schema.

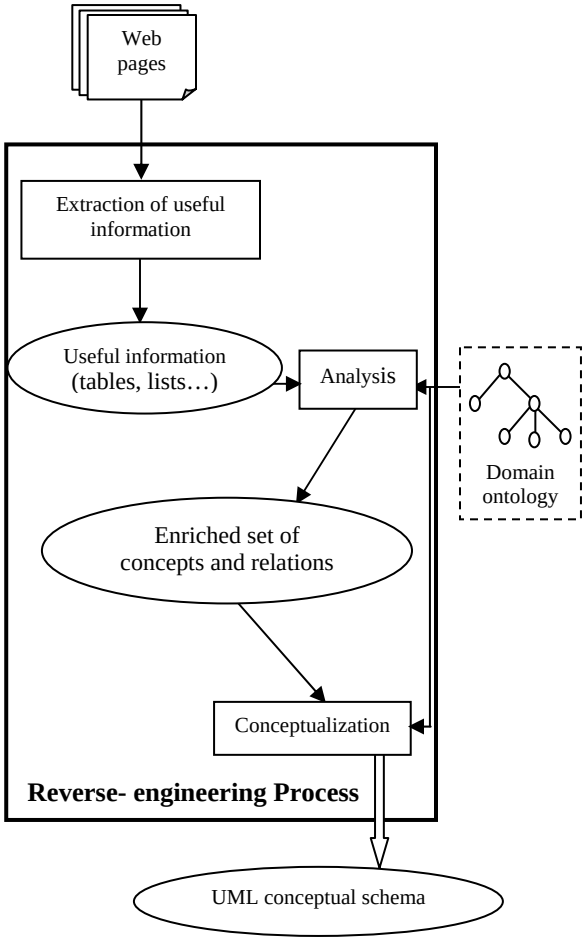

Figure 3: Ontology based Web Application Reverse-engineering Process [1].

4.1. Extraction of useful information

This phase starts with filtering HTML pages, followed by the extraction of DOM[2] and finally the extraction of useful information from DOM. Filtering consists in browsing the source code of HTML pages, eliminate useless tags, and preserve useful ones as <form>, <table>, <td>, <tr>, <ul>, <li>, etc. The result of this stage is a set of cleaned HTML pages. Cleaned HTML pages will be presented in DOM logical format in order to facilitate its manipulation. From this DOM logical format, we can now extract useful information hidden in tables, lists, forms etc.

4.2. Analysis

The analysis phase consists of three stages. First is a Morphological analysis applied on the fields of tables and the elements of lists, extracted from HTML pages. We obtain a set of terms which can be identified later as concepts of the ontology. Second stage is the calculation of semantic distance which aims to quantify how much two concepts are alike. Last stage is the inference which consists in inferring new concepts and relations before generating the UML conceptual schema. The inference starts with the deduction of new concepts and relations to obtain a set of concepts and relations with which we can form a set of groups. Each group represents a connected graph[3]. Before generating the final conceptual schema, the resulting groups from the previous stage must be unified in a single group.

4.3. Conceptualization

From the enriched set of concepts and relations, we can now construct an UML conceptual schema as follow: each concept and relation will be presented respectively by an UML class and relation. The relation expressed by the term part-of will be presented as an UML aggregation relation. The subsumption relations will be translated as a generalization relationship. Multiplicities of the relations are also extracted from the domain ontology to be presented in the UML conceptual schema.

[2] DOM: Document Object Model is an API which consists in decomposing the HTML or XML document content in a tree structure of nodes (each element of the document is a node).

[3] A graph is connected if and only if there is a path between any pair of vertex in the graph.

4.4. OntoWeR tool

This approach is supported by a tool named OntoWeR. This tool is composed of four subsystems (Figure 4):

– Acquisition Module: allows the acquisition of HTML pages, as well as the domain ontology.

– Extractor: allows extracting useful information from the acquired HTML pages.

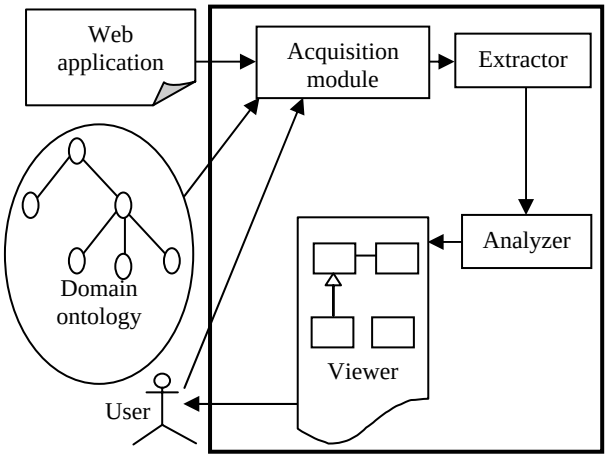

Figure 4: OntoWeR Architecture.

– Analyzer: executes operations, which aim to generate an UML conceptual schema.

– Viewer: allows viewing the resulting conceptual schema.

4.5. Example

To see that this approach is efficient, feasible and reliable, a detailed example has been presented in [1]. For their experimentation they chose an ontology for a Semantic Web of tourism[4]. It describes the tourism domain. The Web site on which they perform their experiments is http://www.hm-usa.com/. It is a Web site for tourism in the United States of America. The UML conceptual schema obtained after applying their approach on the chosen site is presented in Figure 5.

[4] http://protege.stanford.edu/plugins/owl/owl-library/travel.owl

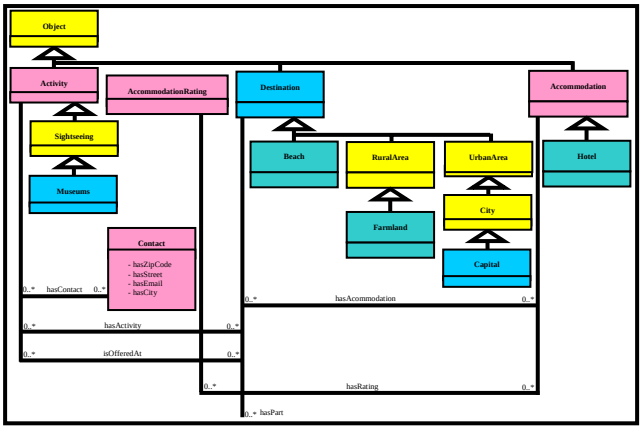

Figure 5: UML Conceptual schema of the chosen site.

5. Implementation phase

This phase consists of two successive steps (Figure 6): first step is mapping to DLM, which consists in applying mapping rules to have the corresponding data logical model (DLM). Second step is the generation of JAVA source code, which consists in generating the source code of the new Web service.

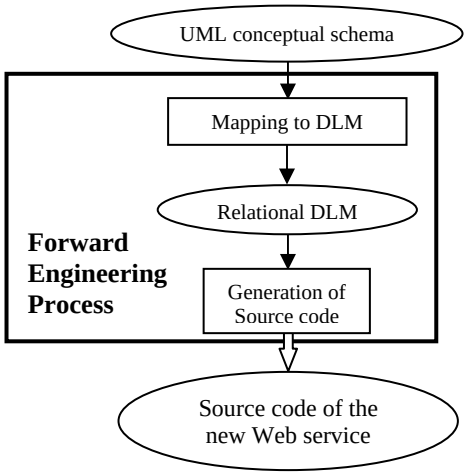

Figure 6: Forward engineering Process.

5.1. Mapping to DLM

For representing data at a logical level, we have chosen the relational model, because it is simple and easy to be manipulated. Mapping rules, which are applied to have a relational DLM from UML conceptual schema, are as follows:

- Each class becomes a relation. Attributes of the class become attributes of the relation. The class identifier[5] becomes a primary key of the relation.
- Each 1-1 association is translated by including the primary key of one of the two relations as a foreign key into the other relation.
- Each 1-N association is translated by including the primary key of the relation, of which the maximal multiplicity is *, as a foreign key in the other relation.
- Each M-N association is translated by creating a new relation, of which the primary key is the concatenation of the two primary keys of participating relations. Attributes of the association class are inserted into this new relation, if it is necessary.
- For the generalization relationship, the superclass and subclass are each one represented by a relation. The two relations share the same primary key. Discriminator[6] must appear as an attribute in the relation corresponding to the superclass.
- Aggregations follow the same rules as associations.

5.2. Generation of JAVA code source

For each relation of DLM, we must create many methods corresponding to the operations of manipulation of the data base table described by this relation. The most important operations are: Listing, insertion, suppression, updating and research.

- The listing method allows displaying the content of the data base table corresponding to the relation. This method has not any parameter.
- The insertion method allows adding a new element in the data base table. It has as parameters the attributes of the relation.
- The suppression method allows deleting an element from the data base table. It has as parameter the element code to be deleted.
- The updating method allows modifying the values of an element of the data base table. It has as parameters the element code to be modified, plus the new values to be saved.
- The research method allows finding an element in the data base table. It has as parameter the element code to be found.

Finally, all methods, which allow the manipulation of the entire data base, will be integrated in one Web service.

5.3. WebSerBuilder CASE

The implementation phase is supported by a CASE tool named WeSerBuilder. This CASE tool is composed of three subsystems (Figure 7):

- Acquisition Module: allows the acquisition of UML conceptual schema.
- Mapping engine: allows the generation of the source code after applying the mapping rules.

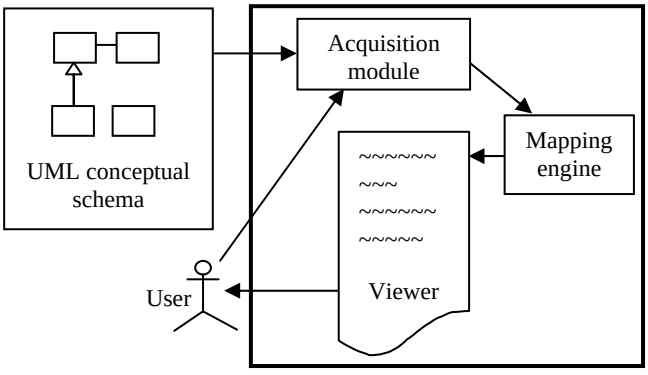

Figure 7: WeSerBuilder Architecture.

- Viewer: allows viewing the resulting DLM and JAVA code source.

[5] Identifier is the chosen attribute that uniquely identifies an object from other objects.
[6] Discriminator is an attribute of enumeration type indicating that the object property became abstract by a particular generalization relationship.

5.4. Example

For a stock management system, we have the following information: A customer is characterized by a customer code, first name, last name, birth date and Address. He orders products at a given date and with a given amount. A product is characterized by a code product and a unit price. A product can be either a paper or a pen. A paper is characterized by its width and its height. A pen is characterized by its color. Each product is ordered from only one supplier (but the supplier can provide many products). A supplier is characterized by a supplier code and name.

The UML conceptual schema describing such system can be represented as follow:

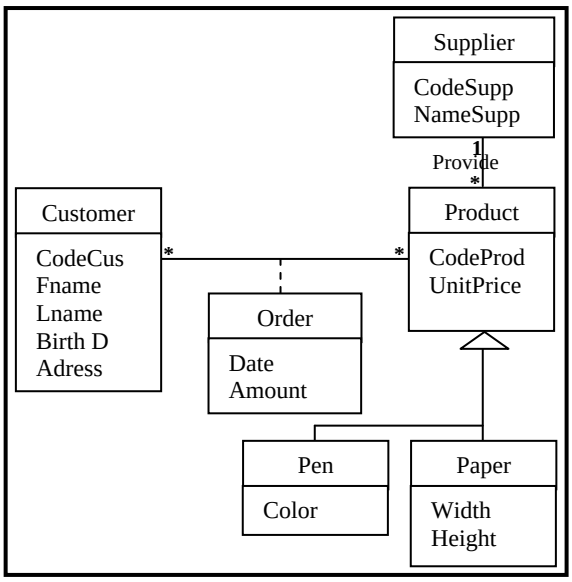

Figure 8: The UML conceptual schema of a stock management system.

When WeSerBuilder tool receives in input the UML conceptual schema given above, it applies the mapping rules to have the following data logical model.

CUSTOMER (CodeCus, Fname, Lname, Birth D, Adress)
PRODUCT (CodeProd, Type, UnitPrice, CodeSupp)
PAPER(CodeProd, Width, Height)
PEN(CodeProd, Color)
SUPPLIER (CodeSupp, NameSupp)
ORDER (CodeCus, CodeProd, Date, Amount)

We can see that each class is represented by a relation. The ORDER association class is also represented by a relation, with adding the customer code and the supplier code as attributes in the ORDER relation. The 1-N association between supplier and product is translated by adding the supplier code to the PRODUCT relation. We add the Type discriminator to the PRODUCT relation. We add also the code of PRODUCT superclass to PAPER and PEN relations.

Now, WeSerBuilder generates the source code of Web service which allows the manipulation of a data base named Stock and represented by the previous DLM. Next, we present an extract of the generated code source.

```
import java.io.*;
import java.sql.*;
public class Stock {
  public Stock()  {  }
  private static Connection dbCon;
  static String Tab[][]=new String[10][2];
  static Connection connect() throws
ClassNotFoundException, SQLException {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    dbCon =
DriverManager.getConnection("jdbc:odbc:Stock","","");
    return dbCon;
  }
  static void close() throws SQLException {
    dbCon.close();
  }
  public static int Inserer(int CodeSupp, String
NameSupp) throws SQLException,
ClassNotFoundException {
    dbCon=connect(); int rs = 0;
    try {
              Statement s = dbCon.createStatement();
              rs= s.executeUpdate("INSERT INTO
Supplier (CodeSupp, NameSupp) VALUES (" +
CodeSupp +", '" + NameSupp +"')");
    }catch(SQLException e) {}
           close();  return rs;  }
…
```

This part of the source code contains: java API declaration, class constructor, declaration of the used variables, connection and closing functions of the data base and a function which allows the insertion of one supplier.

6. Conclusion and future work

Reengineering is the process of analyzing a subject system to identify the system components and their interrelationships, and to create representations of the system in an improved or a new form. The work we have described in this paper focuses clearly on the latter goal. The basic idea underlying our work is to reverse-engineering the existing data driven Web application to create representations at a higher level of abstraction, then we apply mapping rules to create a new Web service with the same functionalities of the initial Web application. In this paper, we proposed a fundamental framework, called WA2WS, for Web service construction from existing Web applications by combining a reverse-engineering approach first and a forward engineering approach after.

The strong point of the reverse-engineering process is that it relies on a very rich semantic reference which is the domain ontology. However, it focuses on removing presentation design and recovering structural information from only HTML interface of the existing Web Application. As future work, we have to investigate also the code source to have not only static abstraction but also dynamic one. The forward engineering process must be enhanced by new rules for implementing dynamic aspect.

The work reported in this paper is clearly work in progress, but we believe that the results of our initial experimentation are quite promising, and we continue to develop and evaluate this process.

References

[1] Bouchiha Dj., Malki M., Benslimane S-M. *Ontology Based Web Application Reverse-Engineering Approach*. INFOCOMP (Journal of Computer Science) VOLUME 6-N. Pages: 37-46. 1-MARCH 2007.

[2] Eleni S., Judi T., Gina S. *Constructing XML-speaking Wrappers for WEB Applications: Towards an Interoperating WEB*. In the Proceedings of the 7 th Working Conference on Reverse Engineering, Brisbane, Queensland, Australia, 59-68, IEEE Computer Society Press. Page: 23-25 November 2000.

[3] Gruber T., *A translation approach to portable ontology specification*s. Knowledge Acquisition 5(2), pages 199-220, 1993.

[4] Hoang P-H., Takahiro K., Tetsuo H. *Web service gateway - A step forward to e-business*. In Proceedings of the 2004 IEEE International Conference on Web Services (ICWS”04). 2004.

[5] Jen-Yao C., Kwei-Jay L., Richard G-M. *Web Services Computing: Advancing Software Interoperability*. IEEE Computer, pp. 35-37, 2003.

[6] Jianzhi L., Hongji Y. *Towards Evolving Web Sites into Grid Services Environment*. Proceedings of the Seventh IEEE International Symposium on Web Site Evolution (WSE’05). 2005.

[7] Michiaki T., Kenichi T. *Decomposition and Abstraction of Web Applications for Web Service Extraction and Composition*. In Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006). Chicago, Illinois, USA. pp.859-868. September, 2006.

[8] Robert B., Georg G., Marcus H., Wolfgang S. *Interactively adding web service interfaces to existing web applications*. Symposium on Applications and the Internet (SAINT'04) p. 74. 2004.

[9] Stefano C., Piero F., Maristella M. *Conceptual modeling of data-intensive Web applications*. IEEE Internet Computing, 6(4):20–30, 2002.

[10] Yingtao J., Eleni S. T*owards Reengineering Web Sites to Web-services Providers*. In Proceedings of the Eighth European Conference on Software Maintenance and Reengineering (CSMR’04), pp. 296-305. 2004.