WSPAB: A Tool for Automatic Classification & Selection of Web Services Using Formal Concept Analysis

Zeina Azmeh, Marianne Huchard, Chouki Tibermacine
LIRMM, CNRS and Univ. Montpellier II, 161 rue Ada
34392 Montpellier Cedex 5 - France
{*Zeina.Azmeh, Marianne.Huchard, Chouki.Tibermacine*}*@lirmm.fr*

Christelle Urtado, Sylvain Vauttier
LGI2P / Ecole des Mines d'Alès - Parc scientifique G. Besse
30035 Nîmes cedex - France
{*Christelle.Urtado, Sylvain.Vauttier*}*@ema.fr*

Abstract

The increased popularity of web services is accompanied with an increase in both provider and service number. This fairly large service number causes a deficiency in the selection of the most pertinent service, and makes it an effortful and time-consuming task. We propose the WSPAB (Web Service Personal Address Book) tool that aims at defining a complete solution for facilitating the task of finding the most pertinent web service. This includes two sub tasks, discovering and selecting. In this paper, we present the first part of the tool concerning the automation of the selection process, taking into consideration the quality of service (QoS) and user preferences. The WSPAB accomplishes the automatic selection of a service by filtering web services according to certain aspects of QoS and certain user requirements; then classifying these services using the formal concept analysis (FCA) approach, enabling users to easily select their needed service, identify its potential substitutes and keep trace of them either for future use, or to be shared with others.

1. Introduction

Web services are a key concept of service-oriented architecture (SOA). They provide various functionalities that can be accessed via the network. In the vision of SOA, the development of software systems can be performed by rapid, low-cost and easy composition of web services. SOA is defined as *"a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains"* [16]. It is built on the basis of three primary roles: service provider, service registry, and service requestor[1]. These primary roles define three primary operations: publishing, finding, and invoking (see figure 1). A provider publishes his services to a registry; a requestor finds a service published on a registry and binds to its provider to invoke it [11]. Finding the right web service to invoke includes: discovering a set of one or more potential web services, and selecting the most pertinent one to be invoked. The discovery is accomplished by contacting a service registry, and searching for a service that meets search criteria. This may return a set of matching web services. From this returned set, a service that best matches the expected requirements will be chosen.

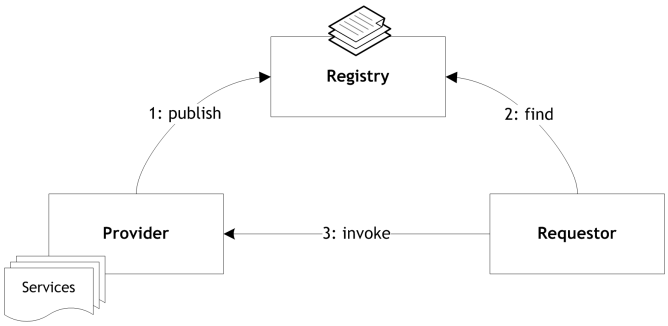

Figure 1. SOA primary roles and operations

The finding operation described above may become challenging because of certain issues like:

[1] Along the paper, we used the term user to refer to the service requestor role

- The absence of a unified public registry, thus the possible unawareness of some available registries, which makes the discovery somewhat, difficult to accomplish
- The possible large number of services in a returned result set makes the selection of the most pertinent service a hard, effortful and time-consuming task.

In this paper, we handle the problem of achieving the automatic selection of potentially pertinent web services out of a fairly large set of services, taking into consideration some quality of service (QoS) aspects with the validation of user preferences. The term quality of service indicates a service's functional and nonfunctional attributes, thus service's usability and utility. There are so many aspects regarding the quality of service such as service's availability, validity, accessibility, integrity, performance, reliability, etc. [15]. In this paper, we tried to address a few aspects like availability and validity, in addition to other aspects that reflect user preferences like service providers, free or paid services, number of operations, specific signatures, etc.

The potentially pertinent services are to be classified, in a certain way using a certain method of classification, in order to enable the user of choosing a service and identifying its potential substitutes clearly and easily.

Having a large set of web services may enable a better selection of a service, but will make manual selection more difficult, as mentioned previously. Thus, automatic selection would economize effort and time, and would give more opportunities of a better selection, especially when considering aspects of QoS and user preferences. Furthermore, providing the user with a few number of pertinent web services classified in a way that shows potential services with their potential substitutes will give the user a clearer organized view and thus will enable him to do a better selection.

The rest of the paper is organized as follows: A motivating example is demonstrated in Section 2. In Section 3, we are giving a quick definition of the adopted classification method. The WSPAB tool is described in Section 4 with the demonstration of its processing steps using a real web services example, and interpreting the resulting classification. In Section 5, we present the related work. In Section 6, we summarize our contributions and give our conclusion and perspectives.

2. A Motivating Example

Along our research, we carried out our experiments on real web services. We obtained these services throughout Seekda.com, which is a global web service search engine based on focused crawling, allowing access to publicly available web services [12]. It indexes and monitors 27K public web services from over 6K providers [2]. In order to understand more the problem of manual selection of web services, we are going to demonstrate an example of a web service that performs currency conversion. We will first obtain a set of web services, by querying seekda.com using the keywords "currency + converter". This returns a result set of 75 web services[2].

We observed that in order to select a certain service, users follow certain criteria during their search. These criteria are on two levels: functional and nonfunctional (reflects the QoS concept). Functional means that a service provides (a) certain operation(s) that meet(s) the required functionality, and nonfunctional means that a service has certain attributes like availability, performance, free or paid, from specific providers, etc. To verify the functional level, a user must verify each service interface described by WSDL, in order to discover what a service offers.

WSDL, the Web Service Description Language is the standard for defining web service interfaces. It describes two different aspects of a service: its signature (name, operations with their parameters), and its binding and deployment details (protocol and location) [16]. By analyzing the WSDL file, a user can verify the existence of a required operation, thus, can determine the service pertinency. After this, a user checks the nonfunctional aspects (QoS) of a service, whether it is available or not, its response time, etc. And also attributes like free services, provided by specific providers, etc. For a fairly large set of web services, this manual QoS filtration and selection tasks requires time and effort. The WSPAB automates these tasks, and offers the user a reduced set of services that meet the required functionality and validate a certain level of QoS. Later on, we will demonstrate the same example, but automated using the WSPAB tool.

3. The Adopted Classification Method

As a classification method, we adopt Formal Concept Analysis [7], a mathematical framework which has already been successfully applied to variants of the practical problem of similarity discovery and abstraction emergence in a wide range of domains including software engineering [20], machine learning [14], data mining [19] or linguistics [18].

The classification we build is based on the partially ordered structure known as *Galois lattice* or *concept lattice* [22] which is induced by a context K, composed of a binary relation R over a pair of sets O (*objects*) and A (*attributes*) (Figure. 2). A formal concept C is a pair of corresponding sets (E, I) such that:
$E = \{ e \in O | \forall i \in I, (e, i) \in R\}$
is called *extent* (covered objects),
$I = \{ i \in A | \forall e \in E, (e, i) \in R\}$
is called *intent* (shared features).

[2]The search result set is obtained on 12th of June 2008

For example, $(\{1,2\},\{b,c\})$ is a formal concept because objects 1 and 2 exactly share attributes b and c (and vice-versa). On the opposite, $(\{2\},\{b,c\})$ is not a formal concept.

Furthermore, the set of all formal concepts $\mathcal{C}$ constitutes a lattice $\mathcal{L}$ when provided with the following specialization order based on intent / extent inclusion:

$$(E_1, I_1) \leq_{\mathcal{L}} (E_2, I_2) \Leftrightarrow E_1 \subseteq E_2 \text{ (or equivalently } I_2 \subseteq I_1).$$

Figure 3 shows the Hasse diagram of $\leq_{\mathcal{L}}$.

	a	b	c	d	e	f	g	h
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

Figure 2. Binary relation of K = (O, A, R) where O = {1,2,3,4,5,6} and A = {a,b,c,d,e,f,g,h}.

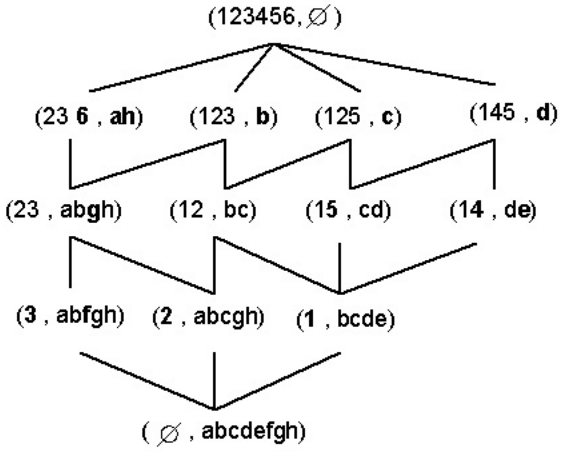

Figure 3. Formal concept lattice $\mathcal{L}$.

We are using the formal concept analysis method, in order to classify web services. We accomplish this classification by defining a binary relation between services and operation signatures. Thus, the objects set represents the services set, and the attributes set represents the operation signatures set. Later on, we can find further details about this classification, with an illustrating example.

4. The Web Service Personal Address Book

4.1. General Overview

The Web Service Personal Address Book is a tool that aims at providing an environment for a simplified dealing with web services. It consists of two parts: discovery and selection. The discovery part means that users can search, select, organize and share their web services. They will form a community of web service users, in which they can work with each other towards building a collaborative address book of proper recommended web services. And the selection part means that web services are well filtered and classified, to enable users of performing an easy proper selection of the most pertinent service, which meets their needs and validates certain aspects of QoS, and having the ability of finding the potential service substitute, in case of a service failure or discontinuity.

4.2. WSPAB-Related Terminology

The WSPAB uses certain concepts during its processing steps. These concepts are defined as follows:

- The **filtration criterion** is a collection of nonfunctional quality attributes: mandatory and optional. Mandatory attributes indicate the service usability. They include attributes as availability, validity and performance of a service. Optional attributes present user preferences. They include attributes as free or paid services, the maximum number of service operations, documented or undocumented services, from certain providers, etc.
- A **pertinent service** is a service that validates the filtration criterion, thus, validates both mandatory and optional nonfunctional quality attributes and provides the needed functionality.
- A **service substitute** is a service that specializes another service, in other words, it offers an extended functionality and can replace the other service. It will be explained further on when interpreting the lattice that results from the formal concept analysis approach.

4.3. WSPAB in Action

In figure 4, we can see an overview of the WSPAB in action. We can divide the execution into the following successive steps:

- Analysis of web services set
- Filtration of web services

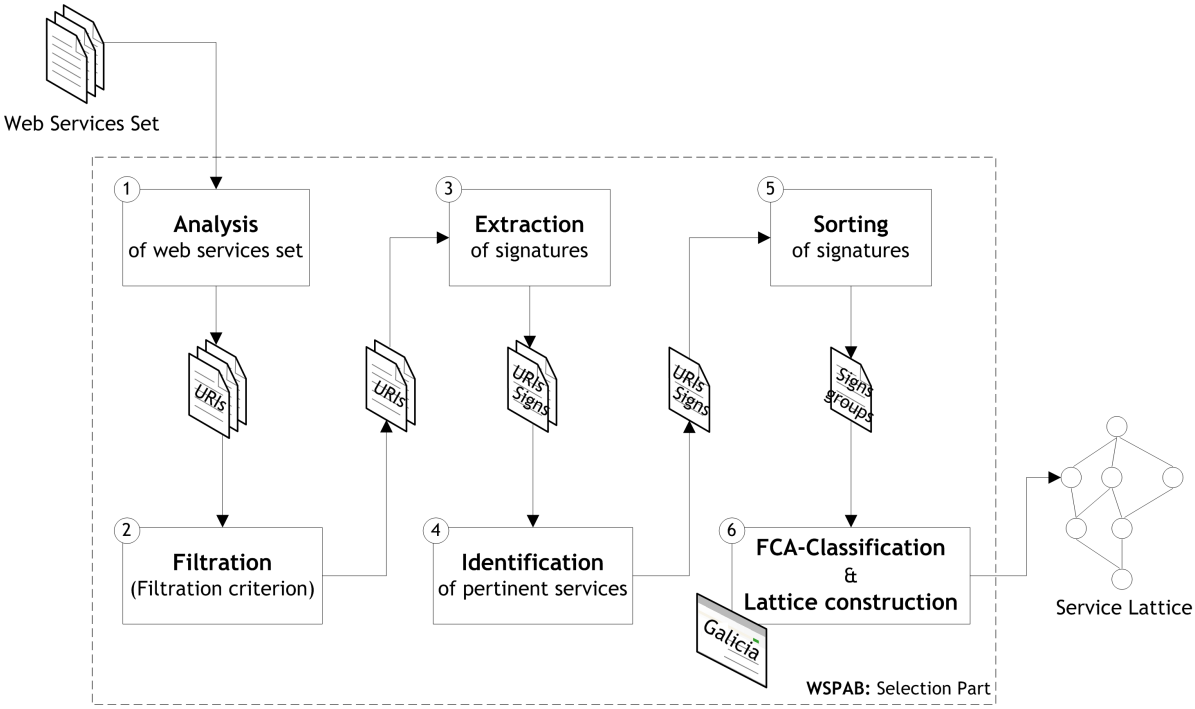

Figure 4. WSPAB in action

- Extraction of operation signatures
- Identification of pertinent services
- Sorting of signatures
- FCA-Classification and service lattice construction

In the first step, the set of web services is analyzed so to extract the URIs (endpoints) of web services. These URIs are stored in a new set, and are passed to the second step, the filtration of web services. In this step, all of the web services specified by their URIs are filtered according to the filtration criterion. The filtration criterion, as indicated before, contains two kinds of nonfunctional quality attributes; mandatory and optional. Each web service is tested against these attributes, so that, services which do not validate them are discarded. The filtration checks the mandatory attributes first, then if validated, it checks the optional attributes. The filtration step also generates a set of web service URIs, but a reduced set that contains only services that are assured to be pure (validate the filtration criterion). In the step of operation signatures extraction, the service URIs listed in the last produced set are used to obtain the service descriptions represented as WSDL files. Each WSDL file is parsed, and operation signatures are extracted. An operation signature is the combination of the operation name, its input parameters, and its output. A new set is generated from this step. It contains for each service its URI, its name and its operation signatures. This set is used in the following step, the pertinent service identification step. In this step, a comparison is carried out between the service signatures and a user given signature of a potential pertinent operation. For now, we will be comparing the signatures according to the number of input parameters. But with the intention of making the comparisons more precise and accurate later on. After the comparison of the signatures with the user's provided signature, we obtain a reduced set of best matching web services. In the next step, the signatures are sorted according to the number of input parameters in each operation. This gives us groups of operation signatures, and for each group, a new signature will be generated to represent the group. These newly generated signatures are then passed to the final step, the lattice construction step, and are used to build the binary table of the FCA, to define the binary relation between the signatures and the services. Then finally, out of this table, the service lattice is constructed with the help of a tool for formal concept analysis, called Galicia (Galois Lattice Interactive Constructor) [21] [1]. The Galicia tool offers many lattice construction algorithms. It analyzes the binary relation between the services and the operations, construct the corresponding lattice and returns a graphical layout of the constructed lattice, showing the resulting concepts with the relations between them.

In the following Section, the WSPAB processing steps are explained using the example of the currency conversion

web service, which was presented previously.

4.4. Example: Currency Converter

In Section 2, we demonstrated an example of the manual selection of a currency conversion web service. Next, we will use the same example to show the automatic selection of the WSPAB tool. We are assuming a user with the following filtration criterion:

- Mandatory nonfunctional attributes: available and valid
- Optional nonfunctional attributes: free, with no more than 6 operations

And with a required functionality represented by the signatures convert(with three input parameters: fromCurrency, toCurrency, amount) or convert(with two input parameters: fromCurrency, toCurrency).

We will demonstrate the automated classification and selection step by step as follows:

The result set returned by Seekda.com is distributed on many HTML pages. Thus, the analysis step in this case represents an HTML parser. Each HTML page is parsed, and web services endpoints (URIs) are extracted and saved to a new set. This set (in our example) contains 75 URIs, and is passed to the filtration step. In this step, each URI from the set is checked, so to determine the validity of mandatory nonfunctional attributes then the validity of optional nonfunctional attributes.

These nonfunctional attributes as already specified above are: available, valid, free services with maximum 6 operations. They are checked as follows:

- The "availability" attribute is verified by checking whether the endpoint URI exists or not.
- The "validity" attribute is verified at the WSDL parsing level (signature extraction step), in which the WSDL structure is verified to be valid or not.
- The "free" attribute is verified by two checks, provider name checking and certain keywords checking. The provider name is checked against a modifiable predefined list of provider names who provide only paid services. The keywords checking is done at the WSDL parsing level (signature extraction step), by checking the existence of certain keywords like: license, password, userId, etc. These keywords are also listed in a predefined modifiable list.
- The number of operations will be checked in the signature extraction step.
- Seekda returned result set may contain some repeated services (i.e. services provided by the same provider and offering exactly the same operations). These services are checked and eliminated in the filtration step and also in the signature extraction step.

In the next step, the signature extraction step, services that are invalid, paid, repeated, or having an operation number higher than the user specified number (which is 6 operations) are discarded. This is done by parsing every service WSDL, in order to extract the service signatures. This step generates a new reduced set that contains for each service: its URI, its name, and its operations signatures (operation name with input and output parameter types). These services signatures are used in the identification of pertinent services, in which, a comparison is carried out between a user's given signature of a required operation and each operation signature of a service. Services with matching operation signatures are stored in a final set, others are discarded. This final set is illustrated in figure 5, it contains only 5 services[3].

```
Service1
http://www.webservicex.com/CurrencyConvertor.asmx?wsdl
Name: CurrencyConvertor
Operations: ConversionRate(string,string)→double

Service2
http://www.mobile88.com/epayment/curconvert.asmx?wsdl
Name: CurConvert
Operations: GetCurrecySign(string)→string
            ConvertCurrency(string,string,string)→double

Service3
http://www.petermeinl.de/CurrencyConverter/CurrencyConverter.asmx?wsdl
Name: CurrencyConverter
Operations: GetConversionRate(string,string)→double
            GetConversionRateList()→string

Service4
http://www.alraimedia.com/CurrencyConvertor.asmx?wsdl
Name: CurrencyConvertor
Operations: Convert(string,string)→double

Service5
http://ws.houseofdev.com/currencyrates.asmx?wsdl
Name: CurrencyRates
Operations: getConversion(string,string,double)→double
            getCurrenciesList()→string
            getCurrenciesListWithDescription()→string
            getRate(string)→double
```

Figure 5. List of pertinent web services

In the final step, the operation signatures of all services are sorted according to the number of parameters, and are assembled under a unified signature of the form:
op (types of input parameters)→type of output parameter,
as shown in figure 6.

These unified signatures are then used in the FCA binary table with the final obtained set of services as shown in table 2, in order to build the service lattice using the Galicia tool.

[3]Service2 has an operation called GetCurrecySign(string), there is an error (service provider error) in the operation name: 'Currecy' instead of 'Currency'

```
0 parameters: op0()→string
 S3:  GetConversionRateList()→string
 S5:  getCurrenciesList()→string
 S5:  getCurrenciesListWithDescription()→string

1 parameters: op1(string)→string||double
 S2:  GetCurrecySign(string)→string
 S5:  getRate(string)→double

2 parameters: op2(string,string)→double
 S1:  ConversionRate(string,string)→double
 S3:  GetConversionRate(string,string)→double
 S4:  Convert(string,string)→double

3 Parameters: op3(string,string,string||double)→double
 S2:  ConvertCurrency(string,string,string)→double
 S5:  getConversion(string,string,double)→double
```

Figure 6. The signature sorting step

The resulting lattice is shown in figure 7. The interpretation of the lattice is given in the following Section.

In table 1, we can find a summary of the outputs after each processing step, describing the resulting set with the obtained number of services.

Table 1. Summary of obtained service sets

	In Set	Out Set
Analysis Step	HTML pages	75 URIs
Filtration & Signature Extraction Steps	30 repeated, 20 unavailable or invalid, 28 paid	17 URIs, service names, operation signatures
Pertinent Services Identification Step	17	5 URIs, service names, operation signatures

4.5. Service Lattice Interpretation & Use

The service lattice in figure 7 reflects the relation between services and signatures in table 2. It is built by the Galicia tool as mentioned previously. It contains information that can be interpreted using a set of rules that are common in all concept lattices. These rules are:

- The concepts are represented using the intent (I), extent (E), reduced I, and reduced E sets. The reduced sets are deduced from the complete sets as follows:
  - A service that appears in the reduced E of a concept is inherited by all the concepts that are above it, and disappears from their reduced E.
  - A signature that appears in the reduced I of a concept is inherited by all the concepts that are below, and disappears from their reduced I.
- When the reduced E is not empty, the concept represents exactly the service(s) that is/are in reduced E with its/their signatures in the I set.
- When the reduced E is empty, this signifies that the concept represents a new service specification (an abstract service) that does not exist in the services set.

From these rules, we can obtain the following information from our service lattice:

- It has three concepts presented by the nodes 5, 2, 3 and are respectively:
  ({Service5, Service2}, {op1(string)→string‖double, op3(string,string,string‖double)→double})

  ({Service5, Service3}, {op0()→string})

  ({Service3, Service4, Service1}, {op2(string,string)→double})

- Service4 and Service1 are equivalent because they are in the same reduced E set, and they do offer the same unique operation represented by the signature op2(string,string)→double
- Service3 can substitute Service1 and Service4, because it is a specialization of these services. This means that Service3 offers the same operation that Service1 and Service4 offer, which has the signature op2(string,string)→double, and also offers another operation that these two services do not offer, which has the signature op0()→string.
- Service5 is a substitute of Service2, because it is a specialization of it, for similar reasons.
- The top of our lattice (node 1) does not present an interesting concept, because there are no common operations in all the services.
- The bottom of our lattice (node 7) does not present an interesting concept either, because none of the services group all the operations listed in I.

The specialization relations between the services are better shown in figure 8, in which we can clearly notice the possible substitute of each service, and the abstract service deduced from the lattice.

In our approach, we group the operations depending on the input parameter number of each operation, i.e. we consider that the similarity factor between the operations is the number of input parameters. Then we generate signatures that represent these groups of operations, and we use them to classify the services.

In our example above, in the case of Service5 and Service2, we can observe the following: Service2 offers the operation: getCurrecySign(string)→string. This operation takes as an input a string representing a country

Table 2. Binary Relation: Services $\times$ Signatures

	$op_0()\rightarrow string$	$op_1(string)\rightarrow string\|\|double$	$op_2(string,string)\rightarrow double$	$op_3(string, string, string\|\|double)\rightarrow double$
Service1 CurrencyConverter Webservicex.com			x	
Service2 CurConvert mobile88.com		x		x
Service3 CurrencyConverter petermeinl.de	x		x	
Service4 CurrencyConvertor alraimedia.com			x	
Service5 CurrencyRates houseofdev.com	x	x		x

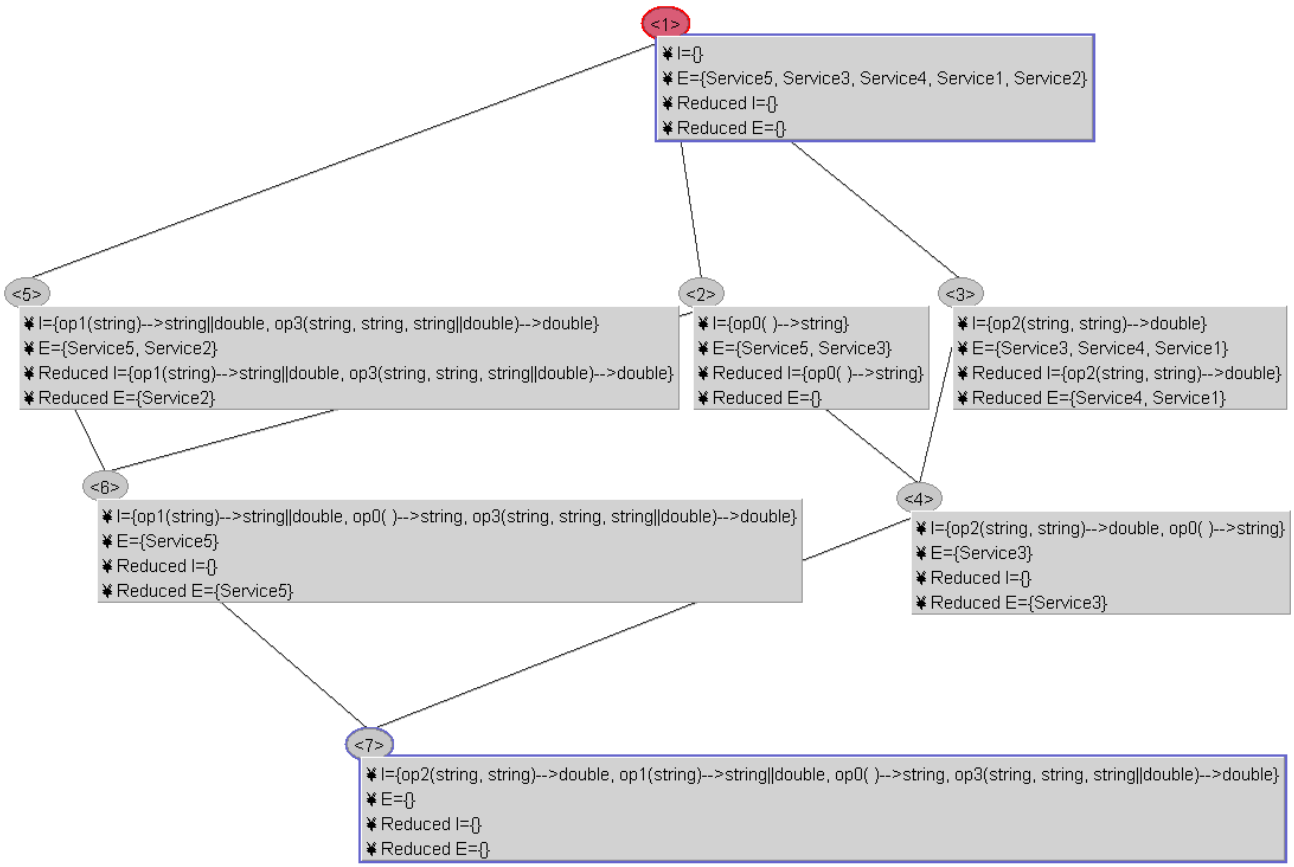

Figure 7. Galicia constructed service lattice

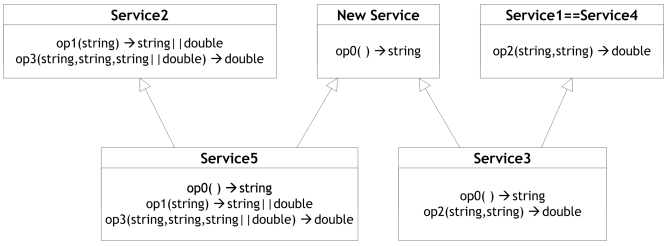

Figure 8. Services and specialization relations

name, and returns as an output the currency sign used in this country. While Service5 offers the operation: getRate(string)→double. This operation takes as an input a string representing a currency sign, and returns as an output the currency rate between the input currency sign and the US dollar. We grouped both of these two operations under the same signature, op1(string)→double, while they provide completely different functionalities. And we have also deduced that Service5 is a substitute of Service2, depending on the resulting lattice. While in fact, Service5 cannot provide the getCurrcySign(string)→string that Service2 offers. This limitation is to be met in a further work, by using other similarity measures, such as using a thesaurus to calculate the similarity between operation names, and similarities between parameter names together with their types. Defining an accurate similarity measure will lead us towards optimized results, together with the service lattice structure, which gives us a clear organized view of the pertinent services with their potential substitutes. This enable us of a better and easier selection of the needed service.

5. Related Work and Discussion

So far, most of the work on web services registries is focused on their architectural styles (centralized, federated, peer-to-peer) and the standardization of APIs to publish and to search for web services descriptions [5]. Registries usually provide simple data models with limited capabilities for structuring their contents. UDDI and WSDA registries for instance [9][8] are designed essentially for the indexation and the discovery of web services. Their entries do not contain any detailed description of the web services but point to external descriptions (generally WSDL) maintained by the web service providers. So, the web service classification provided by those registries consist in the use of keywords that enable to associate web services with business categories. On the contrary, ebXML registries [10] contain extensive descriptions of web services, stored thanks to a complex, extensible data model. This data model enables the definition of multiple classifications, such as business categories or packages of logically related web services. However, classification is handled manually by the web service providers, sending explicit classification information in their web service registration requests. This leads to poorly structured contents and erroneous web service retrievals. To address these issues, more recent work have studied more automatic classification schemes.

In the similarity search field, Dong et al. present a search engine called Woogle which is able to make similarity search between web services [4]. This similarity search combines multiple techniques (similarity primitives) to compare web services and their operations. These primitives are applied to different levels of a web service description: input/output parameter names, operation description and web service description. The authors introduce some algorithms to cluster terms in these descriptions into concepts. They use then some information retrieval metrics (such as TF/IDF) to measure the similarity between concepts.

The work presented in the paper cited above is complementary to ours. The techniques presented are purely semantic while our work is purely syntactic (signature-based). The goal of similarity search is to find among all available service operations, which ones are semantically substitutable or composable (the output of one operation is similar to the input of another). The solution that we propose in this paper provides complementary functionalities. Indeed, while Dong's solution returns flat result sets, WSPAB provides, starting from these flat sets, reduced graphs (hierarchical structures) of concepts.

In [17], Peng et al. present a method to classify and retrieve web services in a repository using formal concept analysis. The authors propose to build a concept lattice starting from a context where objects are web services in a UDDI repository and attributes represent the operations in these services. In this approach, some optimizations are made in order to reduce the size of operation set used for building the lattice. Similarity search techniques are used to compare operation descriptions and input/output messages' data types, which are extracted from WSDL files. Similar operations are merged and only one operation among these ones is put in the context table. References to the other similar (merged) operations are stored in an external data structure to easily find them during web service retrieval. The authors propose an algorithm for retrieving web services. This algorithm parses the lattice in breadth-first manner from top to bottom to find the concepts (services/operations) that match queries defined by users. These queries are first abstracted as formal sequence of operations to be handled then by this algorithm.

The approach presented in this paper resembles our approach; there are however some differences. First, in our approach the lattice is built starting from the (filtered) result set returned by a web service search engine. In the paper

cited above, the lattice is defined for a whole repository of web services. We are convinced that, even with size optimization of operation sets, building a concept lattice for large repositories is a time-consuming task. The obtained lattices are very large and their parsing slows greatly service retrieval. Their experiments have been performed on medium-size repositories (about 2000 web services). In our approach, experiments have been conducted on large web service sources with more than 25000 services. The concept lattice built for the whole services present in this web service source is very large and its parsing will be inevitably not very efficient.

Second, in our work we give a practical view of retrieving web services. Users of WSPAB can use their favorite search engine where they can make lookups of services by simple keywords. Optimization is made on the returned result set as explained in the previous sections. The classification using concept lattices is performed afterwards on a small list of web services. In Peng's work, there is no discussion on how users define their queries. They suppose that these queries can be abstracted as formal expressions representing a web service composed of a sequence of operations. These formal expressions are used in their tool to parse the lattice and retrieve the services that better match the query.

There exists in the literature some other works similar to Peng's one. In [13], FCA is used for component classification based on keywords. Besides the lattice construction, the author investigates incremental specification of queries for refining the search space. The proposal is applied to a large Unix system call documentation. In [3], three configurations of FCA are proposed for analyzing web services and facilitating their selection. In the first configuration a lattice is built on services described by keywords extracted from the documentation through a sophisticated process based on morphological analysis and word frequency metric. In the second configuration, a lattice emerges from the description of operations of a given service by their input and output parameters. In the third configuration, the lattice indicates how complex types (in signatures) are composed of simple types. Their case studies use downloads from UDDI registries. We propose a different FCA configuration, where the lattice is designed for operational purpose: services are described by their operations (including input/output parameters) and effective candidates for substitution can be found and adapted rapidly. The tool is conceived as a companion to search engines (where keyword-based search is proposed).

6. Conclusion & Perspectives

In this paper, we presented the WSPAB tool for the automatic classification and selection of web services from an online web service repository. This tool processes by multiple successive steps. It first analyzes the content of the service repository, querying it to find a first set of candidate services. Secondly, it filters this service set according to functional and nonfunctional criteria, considering user preferences. It then extracts the operation signatures of the services from the resulting set in order to further filter them according to this syntactic information. Finally, the set of remaining services is classified into a service lattice using Formal Concept Analysis. The obtained lattice can be used to identify both the service that best adapts to the user's needs and its possible substitutes when needed. WSPAB conforms to the Web 2.0 philosophy as organized and trusted web service address books enhances sharing and collaborations among web service user communities.

The main originality of our work lies in the fact that we use syntactic information on web services where most existing approaches use manually set quality criteria. This enables our solution to be fully automatic even if some of the information provided on each service indexed in the repository is incorrect or incomplete.

Perspectives for this work are numerous. We plan to refine the process proposed in this paper using a thesaurus to better identify matching services that do not have the exact searched name. We also intent to extend the proposed tool in order to deal with semantic web services. At last, we plan to work on web service discovery, as a preliminary phase of the process described in this paper, in order not to depend on some service repository.

References

[1] Galois lattice interactive constructor - http://galicia.sourceforge.net/.

[2] Seekda.com, a web services search engine: www.seekda.com.

[3] L. Aversano, M. Bruno, G. Canfora, M. Di Penta, and D. Distante. Using concept lattices to support service selection. *International Journal of Web Services Research*, 3(4):32–51, 2006.

[4] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB'04)*, pages 372–383. VLDB Endowment, 2004.

[5] S. Dustdar and M. Treiber. A view based analysis on web service registries. *Distributed and Parallel Databases*, 18:147–171, 2005.

[6] B. Ganter, G. Stumme, and R. Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*. Springer, 2005.

[7] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1997.

[8] W. Hoschek. The web service discovery architecture. In I. C. S. Press, editor, *Int'l. IEEE/ACM Supercomputing Conference (SC 2002)*, 2002.

[9] http://uddi.xml.org/. *Universal Description, Discovery and Integration (UDDI) v3.0.2*, Fev 2005.

[10] http://www.oasis-open.org/. *ebXML Registry Services Specification (RS) v3.0*, May 2005.

[11] N. M. Josuttis. *SOA In Practice, The Art of Distributed System Design*. O'REILLY, August 2007.

[12] H. Lausen and T. Haselwanter. Finding web services. *In Proc. of the 1st European Semantic Technology Conference (ESTC), Vienna, Austria*, 2007.

[13] C. Lindig. Concept-based component retrieval. In J. Köhler, F. Giunchiglia, C. Green, and C. Walther, editors, *Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs*, pages 21–25, 1995.

[14] M. Liquiere and J. Sallantin. Structural machine learning with galois lattice and graphs. In J. W. Shavlik, editor, *ICML*, pages 305–313. Morgan Kaufmann, 1998.

[15] A. Mani and A. Nagarajan. Understanding quality of service for web services, improving the performance of your web services - IBM. Technical report, http://www.ibm.com/developerworks/library/ws-quality.html, January 2002.

[16] M. P. Papazoglou. *Web Services: Principles and Technology*. Prentice Hall, 2008.

[17] D. Peng, S. Huang, X. Wang, and A. Zhou. Management and retrieval of web services based on formal concept analysis. In *Proceedings of the The Fifth International Conference on Computer and Information Technology (CIT'05)*, pages 269–275. IEEE Computer Society, 2005.

[18] U. Priss. Linguistic applications of formal concept analysis. In Ganter et al. [6], pages 149–160.

[19] G. Stumme. Efficient data mining based on formal concept analysis. In A. Hameurlain, R. Cicchetti, and R. Traunmüller, editors, *DEXA*, volume 2453 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.

[20] T. Tilley, R. Cole, P. B. 0002, and P. W. Eklund. A survey of formal concept analysis support for software engineering activities. In Ganter et al. [6], pages 250–271.

[21] P. Valtchev, D. Grosser, C. Roume, and M. R. Hacene. GALICIA: an open platform for lattices. In A. d. M. B. Ganter, editor, *Using Conceptual Structures: Contributions to ICCS'03*, pages 241–254, Aachen (DE), 2003. Shaker Verlag.

[22] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. *Ordered Sets*, 83:445–470, Sept. 1982.