Design of Web services filtering and clustering system

Witold Abramowicz, Konstanty Haniewicz, Monika Kaczmarek, Dominik Zyskowski
Poznan University of Economics, Department of Information Systems,
Al. Niepodległości 10, 60-967 Poznań, Poland
{w.abramowicz, k.haniewicz, m.kaczmarek, d.zyskowski} @kie.ae.poznan.pl

Abstract

The need for filtering of services results from the ever growing number of available Web services that may be used to compose various business applications. Some of the available Web services offer similar functionalities, thus the need to differentiate between them occurs. The Semantic Web services filtering process is therefore based not only on the ontological description of functional aspects of services (i.e. what a service does), but also on a description of non-functional ones (i.e. how it performs its functionality). Within the filtering process, both functional and non-functional aspects of a service expressed using ontology are confronted with the preferences a user specified and the description of a composite application the service may become a part of. One of the weaknesses of the described filtering process is lack of high efficiency and its complexity as processing ontological descriptions and reasoning on them is time-consuming. In order to speed-up the filtering process, clustering techniques narrowing down the set of potential services to be considered by the filtering mechanism may be applied. In this article the architecture for Semantic Web services filtering and clustering system is briefly discussed.

***Keywords:* Semantic Web services, filtering, clustering, architecture**

1. Introduction

Service Oriented Architecture (SOA) systems may be implemented using Web services technology, which allows for easy creation of reusable components. These components serve as building blocks to create composite structures i.e. business applications that should have high level of quality and consist of the best-of-breed (i.e. the best available) components. However, the market of services is not static and the number and properties of services are changing constantly. There were over 20.000 publicly available services in 2005 [1], whereas about 1200 in 2004 [24]. According to the latest research of Al-Masri and Mahmoud the number of publicly available Web services between October 2006 and October 2007 increased by 131% [36]. With the augmented appearance of service substitutes, a need emerges not only to identify the functionally relevant services but also to distinguish the best-fitting ones to be used within the composition. Once relevant components are identified, e.g. better in terms of non-functional aspects than the already used ones, the replacement of components in the application may follow.

In order to efficiently perform the process described above, the need for service selection, discovery and filtering arises. Due to the overwhelming number of Web services, which will exceed human cognitive capabilities, automation of these processes is strongly recommended. It may be achieved by using semantics and Semantic Web technologies [2] - in the consequence by the exploitation of Semantic Web services (SWS) paradigm. However, although using semantic allow for automation, most of the processes based on the ontology are time and resource consuming.

In this article, which is related to our ICIW 2007 publication [38] we propose architecture and algorithms for filtering and clustering to support identification of relevant services and selection of the best-fitting Semantic Web services to be used by a business system using external Web services. The proposed system is an extension of the F-WebS project [4] and may be used also in the context of Semantic Web services e-marketplaces [37].

The structure of the article is as follows. First, in the section 2 the related work is discussed. Then, we present a motivating scenario that will justify the application of service filtering system. In the next section the basic definitions relevant to the concept of service filtering and clustering are presented. In the following section the architecture of the implemented system is shown. Finally, the future work and conclusions are given.

2. Related work

Semantic Web services and their applications are one of the most popular research topics these days. Some researchers focus on creating the adequate semantic description of a Web service that would make this idea possible – e.g. OWL-S [3], WSMO [5], WSDL-S [6] or SAWSDL [28] while others concentrate more on mechanisms and algorithms used within the Semantic Web services description based interactions [7]. Many of the publications on service interactions tend to put more emphasis on certain aspects of reasoning [8, 9] rather than on focusing on current constraints and foreseeable evolvement of service interactions.

The ultimate challenge in the SWS world is still an issue of expressive description, reasoning mechanisms and their efficiency [14, 15]. Dealing with the ontologized description of a service implies the necessity to use the appropriate reasoning engines. Researches in AI and knowledge representation emphasize the fact that a choice between expressiveness of the notation and efficiency has to be made (due to feasibility of the task). Taking this issue into account most of the initiatives in the SWS field decides to use description expressed in the terms DL [12]. Nevertheless, the efficiency of the performed processes is still an open problem.

There are many publications describing the architecture of Semantic Web services systems performing various interactions among others also discovery and matchmaking of services in various domains [9, 10, 11, 12, and 13]. What is more, there is tremendous research effort in several EU-funded projects (some still ongoing) that deal with Semantic Web services and their applications in business context (e.g. ASG [12], METEOR-S [13], SUPER [25]). To our best knowledge there is none among them using the algorithms described in this paper.

The filtering of Web services and then Semantic Web services was first proposed by Abramowicz et al. [4]. It takes advantage of the achievements in SWS description and discovery area [9, 16, 17, 18 and 19] as semantic-based Web services filtering uses a variant of matching algorithms similar to ones used in Semantic Web services discovery process.

The idea of Web services clustering is not a novel one. First attempts were made based on the WSDL service description [20]. However, the effective and precise SWS clustering is still an ongoing research topic. The majority of researchers have left illusions of any reasonable results based on adoption of standard methods derived from the information retrieval field. At the moment, the only feasible solutions base on the employment of semantics and creation of similarity measures that take advantage of the underlying ontologies [20, 29]. It also seems that the most important issue associated with Web services clustering is the similarity measure, which has to fully map the relationships between various, differently formulated however similar Semantic Web services.

3. Motivating scenario

One of the reasons to build system according to the SOA paradigm and use the Web services technology is to easily and rapidly compose applications out of available services. Even though, the current state of publicly available Web services is far from the envisioned one, a user has an access to a variety of simple services that may be used to create a piece of software of real utility to business users.

The aim of this example is to sketch up a real world situation where not only a practical Web services based application is created but also it is maintained with the support of Web services filtering and clustering system.

The domain of application has been selected based on the following criteria:

- current availability of services,
- possibility of application,
- perks from application,
- relative ease of services description.

In our opinion the best domain for the practical illustration of our research is a financial area which has two following advantages: existing variety of services, wide spectrum of potential applications also for non-enterprise end-users.

An exemplary application built out of services is designed to manage personal finance, having an access to bank accounts, and authority to transfer money among accounts, buy or sell. This ideal example is based on general assumptions that the application can represent its user having all his rights. Discussion of soundness of this statement is beyond the scope of this article.

Table 1 enumerates necessary services (to be specific - service types, not the exact Web services) which have to be encapsulated to form desired application.

The mentioned elements may form an application that invests any superfluous money on bank accounts in one of the possible ventures. For example, by analyzing the exchange course between any pair of currencies, a user can decide (basing on a suggestion made by the discussed application) to play arbitration games by exchanging money from one currency to other. The user can choose alternatively to make a deposit in a bank that has a higher interest rate that the one that possesses additional funds. Examples can be easily multiplied.

Table 1. Financial services

Service	Functionality description
financial situation	service should enlist all the liquid assets and on this basis one may undertake decision
money transfer	an application to be useful has to be able to transfer money from one account to other, thus the need of transfer functionality
exchange course	exchange course informs user of the ratio between currencies
trend	trend functionality should return tendency of some input data. E.g. introducing average price of some commodity, the functionality should return whether there is steady rise, fall, stagnation or some seasonal fluctuation
risk	risk service in the simplest mode should describe the deviation of prices of any commodity throughout some time
invest-ment situation	investment situation should list the actual state of all investments made. It should be based on their history.
invest-ment history	investment history apart from delivering data to the investment state should provide some manner of report creation for introspective reasons
interest monitor	interest monitor should be able to provide maximum information of interest rates of different commodities, such as stock options, raw materials, precious metals etc.
transfer cost	informs about financial viability of transaction (whether there should be a gain that is lesser than the cost).

The sheer power of the proposed solution lays in the constant monitoring of elements and providing suggestions for any possible tweaks and exchanges in the orchestrated workflow. Imagine that new kind of Web service that provides more accurate approximation of trend or risk evaluation should appear. The system will filter out any Web service that can be an upgrade for any of the components used in the application taking into account both functional and non-functional parameters that were defined by the user in his profile.

This stage is crucial for the system as not only users may enhance their application but also they may be sure that the elements they use are the best ones fitting their needs and preferences.

4. Web services filtering and clustering

This section is divided into three subparts. The first one provides general information on the process of both clustering and filtering. The second describes various considerations of the clustering task. Finally, the third subsection presents details of the filtering process.

4.1 General information

In general, information filtering can be described as specifying which objects from a given stream are relevant to a given profile. According to [21], a profile is a representation of regular information interests that may change slowly over time, as conditions, goals and knowledge change. This representation is used in filtering systems to provide users with information with the highest relevance.

In the Fig. 1 the process flow in the Semantic Web services filtering and clustering system defined based on the general architecture of the filtering systems [21] is presented. However, the filtering process has been enhanced with few additional activities aiming at increasing the effectiveness of the system (i.e. clustering).

In this model three main functional sections can be distinguished: service description creation (A), profile creation (B), and filtering and refinement process (C) (here comes into play clustering algorithm which not only saves execution time but also improves refinement of stored data). Each section consists of several subprocesses.

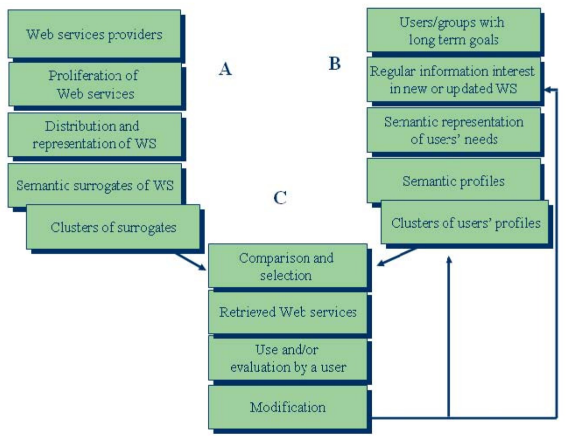
Web services providers
Proliferation of Web services
Distribution and representation of WS
Semantic surrogates of WS
Clusters of surrogates
A
B
Users/groups with long term goals
Regular information interest in new or updated WS
Semantic representation of users' needs
Semantic profiles
Clusters of users' profiles
C
Comparison and selection
Retrieved Web services
Use and/or evaluation by a user
Modification

Figure 1. Semantic Web Services Filtering and Clustering

The filtering begins with clients of the filtering system having relatively stable and long-term goals (to consist of the best-of-breed components). Attaining such objectives is connected with acquiring information about new or updated services on the e-marketplace. This entails a demand for information, which is subjected to continuous changes mainly due to natural change of users' goals, conditions (e.g. change in components that are part of a composite application) or changes on the e-marketplace itself. The users' or applications' information needs can be represented in the form of profiles placed in the filtering system. In our system, the profiles are represented by enhanced OWL-S description [4] (OWL-S with extended list of non-functional properties, so that larger number of quality aspects would be taken into account in the filtering process). Additionally, user preferences assigned to the specific component of a business application are also included in his profile (e.g. the credit card payment service needs to be secure).

Simultaneously, service providers attempt to distribute their services, so that users become aware of them. Such services are represented as semantic artefacts that are amenable to computer processing. In order to make service descriptions and user profiles comparable, both are reduced to sets of attributes that map objects to a common representation. That is why as the service description within our system again enhanced OWL-S was used.

Moreover, in order to increase the effectiveness of the filtering process, the clustering analysis on profiles and services is performed. Main goal of the clustering algorithm is to shorten the duration of the matchmaking between the user's profile and the profiles of services stored in the repository. The efficiency paradigm of this task limits the range of feasible algorithms to those which can update their clusters within the online transaction and make use of medioids as the common denominator for the whole cluster.

4.2 Clustering details

Before we delve into the matter of medioids, their definition and selection has to be introduced into general set of assumptions and representations used for the need of the clustering task.

By enhancing Web services with semantic annotation a new set of additional information is introduced. This information set is represented by already mentioned domain ontology and should be used within the clustering process.

Traditional approach to representation of entities for clustering needs [32] postulates a vector-like representation of every service. Naive implementation would follow this idea by mapping ontology to a vector of data where every consecutive field would denote either absence or presence of some trait taken from the abovementioned domain ontology and checked with a Web service in scrutiny.

In addition to fact that this approach seems to be an excess in terms of meaningfulness (one has to bear in mind that ontologies grow and evolve, one vector cannot suffice all Web services without assumption of changelessness [33]), one should also consider the sheer amount of data that has to be loaded into computer's memory for the sake of computation.

Every Web service description takes into account four most important aspects: inputs, outputs, preconditions and effects (the same assumption is used within the filtering phase that is discussed in more detail later on in this section). Due to the fact that preconditions and effects are hard to define without considering every usage scenario, practice has dictated to annotate only inputs and outputs.

When these two are taken into consideration, a Web service can be described as pair of two vectors, first for all input parameters and second one for output parameters.

$$ws = (i, o)$$
$$i - vector of inputs$$
$$o - vector of outputs$$

Formula 1 - Web service description as a pair of input and output vectors

If a Web service is to be perceived as an abstraction for function, vector of outputs can be replaced by a single value. Nevertheless, in a general usage scenario there are two vectors which size depends on the buoyant environment as any domain ontology which is used for description of every parameter is prone to change.

The key element of clustering task is a representation of distances among all Web services stored in a system. However, a general distance function to be used for Web services clustering is hard to define due to varying number of parameters and interactions occurring among services. Here, one has to consider a usage of distance matrix, where a distance among services is presented as a pair of two values where, in accord to representation chosen before, first value represents distance between input parameters of two Web services and the other distance of output parameters.

$$\begin{bmatrix} (dws_{1,1}i, dws_{1,1}o) & \cdots & (dws_{1,n}i, dws_{1,n}o) \\ \vdots & \ddots & \vdots \\ (dws_{n,1}i, dws_{n,1}o) & \cdots & (dws_{n,n}i, dws_{n,n}o) \end{bmatrix}$$

Formula 2 - Distance matrix for semantically annotated Web services

Where $dws_{x,y}i$ denotes distance between Web service's x inputs and Web service's inputs y and $dws_{x,y}o$ between output vectors.

Of utmost importance is to emphasize once more that we deal with varied entities. Varied to the point where one has to compare sets of information describing different number of parameters.

Thus, a need for heuristic functions to bring different Web services to a common denominator in terms of inputs and outputs occurs.

In the general case, the distance is calculated with use of reasoner, yet it can be delegated to other software if one would decide to ignore other than hierarchical relationships in domain ontology. It is a common practice due to difficulties with evaluation of the impact that these other types of relationships impose to the problem domain – in some cases they introduce ambiguity in others they are irrelevant.

The complexity of distance computation is of O(n2) class due to the need of cross-examination of every parameter of a first service against every parameter of a second one. The same situation appears within the first stage of general filtering phase. It is to be remembered that this procedure is performed for each pair of services in the system and that the distance matrix is not symmetric due to the nature of relationships and their meaning for distance computation (simple inversion of values is not a valid value of inverse distance measure).

For the sake of discussion, consider following assumptions as to the values representing relations among concepts:

- when two parameters in question annotated with concepts from a domain ontology are subsuming one another a default value of 0.75 is used,
- default value for subsuming concepts is being modified depending on number of levels between them, when an ontology is treated as a taxonomy – i.e. how much more general one concept is from the other (default 0.75 is reserved for case of simple derivation – no other concepts lay on the path from the more general one to the less general one),
- when dealing with inverted subsuming concepts a default value of 0.25 is introduced that can be modified in analogical manner to the above-described one,
- when two concepts match a value of 1 is used,
- when there is no relation between parameters 0 is used to denote the state.

When one is presented with three different semantically annotated Web services i.e. ws_1, ws_2 and ws_3 which for the presentation's sake have the same number of input and output parameters, one can observe how distance is represented in general.

$$dws_{1,2} = ((1, 0.75, 0), 0.5)$$
$$dws_{1,3} = ((0.25, 0.5, 0.25), 0.25)$$

Formula 3 - An example of distances between Web services

Where $dws_{x,y}$ denotes distance between a Web service x and a Web service y.

As mentioned before, the situation in the example is rather simple one. Nevertheless, it can be used to demonstrate that average which could be easily applied to answer which Web service is closer to the first one is not applicable when the assumption of equal number of parameters is removed.

A first step to forging out of good heuristic function which would serve us in distance measurement is to answer a question of variable parameter number in Web services to be compared.

It is known that when a Web service has less input parameters in comparison to another one the situation is far from being optimal for the algorithm. We cannot assume that a Web service is worse (in terms of effectiveness) due to a fact that it does not use of all information in our possession. This statement is derived from observation of extra parameters that are treated as default values thus bearing no interest to the user in terms of his preferences. The possible composition of parameters (inputs or outputs) has not been proved to give satisfactory results in general use cases [34].

We can make an assumption that a parameter described by a more general concept is more desirable than a parameter described by less general one when we deal with input parameters and in reverse manner when we deal with output parameters (the more specific, the better).

Lack of relation between parameters is to be penalized in a manner similar to the situation when the number of parameters is different for both Web services.

When we employ a reasoner to answer whether two concepts are in relation (one subsumes the other one or they are identical) or no relation is present and the assumption of taking into account hierarchical relations holds we face a problem of meaningful information loss. Pray consider two concepts that are siblings (thus share a common parent). We can speculate that they can be in strong relation despite the fact that reasoner returns no relation value. To enable algorithm not to miss this kind of data it has been enhanced by a routine that check which concepts are instantiated and which serve only as classification ones.

If one is to consider a concept of currency that has only two subconcepts, euro and dollar concepts it is easy to weed out all input and output parameters that are of these types. Highly probable is that no service would employ a concept of currency but its specialization, either euro or dollar. Thus, we gain knowledge that the two are in strong relation and are possibly interchangeable.

All these is gathered and presented along with example of computation of distance measures among Web services in every cluster represented by medioids in further part of the section. Nevertheless, one can easily see that all presented steps were introduced to show how original distance matrix is to be transformed into a one that can be used by one of the well described clustering algorithms ($dws_{x,y}$ – transformed distance measure between two web services).

$$\begin{bmatrix} dws_{1,1} & \cdots & dws_{1,n} \\ \vdots & \ddots & \vdots \\ dws_{n,1} & \cdots & dws_{n,n} \end{bmatrix}$$

Formula 4 – Transformed distance matrix for semantically annotated Web services

We have proposed to represent the medioid as a most general Web service's profile i.e. the profile which has the factor of generality of the highest rank (the factor of generality is the weighted average of inputs, outputs and the existence of necessary non-functional parameters). This approach has been used by [29]. The issue with this factor arises from the fact that some Web services can take more inputs as others yet provide the same functionality (as depicted above in the section). It is easy to depict such an example. Let us assume that in the domain ontology the notion of amount of money is defined as an actual amount and the currency which applies to it. One Web service may take only one input with the stated amount of money already with the currency denominator, other may take two separate inputs one for the number, another for the currency. Controlling the domain ontology gives the ability to state that the inputs from the second Web service are encompassed by the one from the first. Therefore for such a simple case in which we have the two mentioned services put together in the cluster the first Web service is chosen for medioid. In current implementation the output has greater weight in the generality factor as obtaining what we want neglecting all what we do not need, has a greater value to the potential user.

The arising questions of possible multiple inputs out of which some do fit into the pattern and others do not, is generally well handled as the stored Web services provide only one functionality (one Web service does one thing, thus resembles a function in programming language). Furthermore, if one is to decide whether the Web service fits into a cluster by examining whether the output suits the not neglected inputs thus satisfying the goal of algorithm.

When medioids are chosen, one has to decide on the number of clusters in the repository. If we are to consider only the most general concepts (as F-WebS uses OWL, most general concepts are those derived directly from owl:Thing) we have to put certain amount of trust into domain ontology architect's skills in the matter of granularity choice of the main concepts.

Alternatively, one can use expert's approach to amend possible shortcomings of granularity induced by the ontology architect and come up with a number that is more desirable.

Natural algorithm for clustering when medioids are present is Partitioning around medioids [35]. There are four main steps in the algorithm:

- Initialization. Setting desired number of clusters (k). Domain is not covered by any of clusters. We initialise k medioids.
- Algorithm checks for the closest element to one of k medioids.
- Test for stop criterion is performed. Is the whole domain covered by the target number of clusters? If the answer is affirmative, algorithm ends its work, else it goes on.
- Medioids are updated with freshly found closest elements and thus a need for their recalculation arises. When finished, the algorithm returns to the second step.

Equipped with all the necessary information it is time to review an example. Let us consider the situation with three Web services:

- ws_1 has four input parameters: begin, end, stock-exchange and country. It has one output parameter of euro type.
- ws_2 has two input parameters: time-period and market, it has one output parameter of dollar type
- ws_3 has one input parameter of country type and one output parameter of timestamp type.

Services are described with the following example domain ontology.

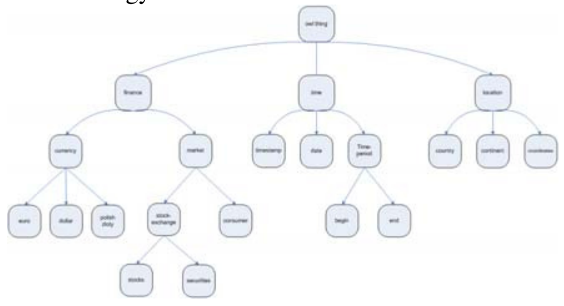

Figure 2. Example domain ontology

First phase takes into account number of parameters (whether an over or underflow is present) and it pairs best fitting parameters along with evaluation of their numerical relation. The phase is shown in Table 2.

Table 2. First phase of distance measure transformation

	parametres					flow	
	input				output	-	+
ws₁	begin	end	stock-exchange	country	**euro**	2	0
ws₂	time-period		marke		**dolar**		
result	*0,25*		*0,25*		*0,4*		
ws₁	begin	end	stock-exchange	country	**euro**	3	0
ws₃				country	**timestamp**		
result				*1*	*0*		
ws₂	time-period	market			**dolar**	0	2
ws₁	begin	end	stock-exchange	country	**euro**		
result	*0,75*	*0,75*			*0,4*		
ws₂	time-period	market			**dolar**	1	0
ws₃	country				**timestamp**		
result	0				0		
ws₃	country				**timestamp**	0	3
ws₁	begin	end	stock-exchange	country	**euro**		
result	*1*				*0*		
ws₃	country				**timestamp**	0	1
ws₂	time-period	market			**dolar**		
result	*0*				*0*		

First part of the table 1 informs us of which services were analyzed. Second part gives information of paired parameters, their relation (notice use of 0.4 for parameters that would have no relation in standard reasoning yet were classified as instantiations of general concept) and underflows (column -) and overflows (column +) in the number of parameters.

The second phase is started with computation of distance of input parameters using following formula:

$$d_i = \mu(1 - 0.1l - 0.05e)$$

Formula 5 – Distance measure for input parametres

Where μ stands for relations average in the analyzed parameters, l for underflow of parameters and e for overflow of parameters.

For our example results are presented in table 3.

Table 3. Input distances matrix

services	*ws_1*	*ws_2*	*ws_3*
ws_1	1	0.2	0.35
ws_2	0.675	1	0
ws_3	0.85	0	1

Due to the fact that every service has only one output calculations of output distance are obvious and presented in the table 1.

The final step is to combine transformed inputs and outputs to come up with transformed distance matrix. This is achieved by applying the formula 6:

$$d_{ws} = (0.45d_i + 0.55d_o) - h$$

Formula 6 – Final transformation of distance measures

Where di is a distance measure of inputs, do is a distance measure of outputs and h is penalty applied when the absolute difference between is greater than 0.4 and is greater than 60% of value expressed by first part of formula 6.

Final distance matrix is presented in table 4.

Table 4. Final distance matrix

services	*ws_1*	*ws_2*	*ws_3*
ws_1	1	0.31	0.1575
ws_2	0.52375	1	0
ws_3	0.135	0	1

4.3 Filtering

The filtering algorithm works in two stages. The first stage, ontology-based filtering aims at detection of service substitutes. It is quite similar to the typical matchmaking process. There are a few algorithms that match functionalities of provided and requested services [7]. Some of them are divided into stages, while some do everything in one step. We decided to take advantage of the method proposed in [19]. For more details regarding the filtering process see [4].

Analyzed elements of OWL-S service description are following: inputs, outputs, and service category. The algorithm starts whenever new service appears on the market. It checks whether the functionality of the service is relevant to any user profile. In order to shorten the time it is compared not to each service separately but to the medioid representing every cluster of services. If the new service turns out to be relevant to the given medioid, it is then compared to each and every service from the cluster in question. The four levels of matching between two properties/parameters were distinguished

- Equivalence - concepts have the same meaning;
- Subconcept - one concept is a subconcept of the other concept;
- Unclassified - one of two concepts is not classified;
- No relation - in other cases

Functions that determine levels of match use the ontology reasoner Pellet [30].

The final result is an aggregation of results from all partial comparisons. The new service is relevant to the profile when the final result of ontology-based filtering is higher than the threshold defined in the system. In other cases new services are turned down and not passed to further analysis.

The second stage of procedure is the constraint-based filtering. Its objective is to identify the best service from the set of relevant services according to the user preferences. Some propositions to compute utility function over the given parameters can be found in [22, 23], it is also possible to compute a distance measure, but it does not take preferences into account. That is why we have decided to take advantage of a multiple criteria analysis (MCA). Using this method services can be compared according to their characteristics, e.g. price, response time, accessibility. To each characteristic a weight is assigned, reflecting an application's preferences [4]. The exact method used to compare phenomena is computation of the synthetic indicator. For example, if two services are given, together with some statistics concerning their characteristics e.g. response time, reliability etc., it may occur that one of the services performs better according to reliability, while the other is more accurate and less expensive. Additionally, one service is paid by credit card and another by wire transfer. The synthetic indicator allows for comparison of such services, given the vector of user preferences. For details concerning this stage see [27]. The highest value is chosen as the indicator of the best service. If the best service is the incoming service then the user is notified.

5. Architecture of the system

The architecture of the system described in the previous section should consist of at least few components connected according to the SOA paradigm. The conceptual architecture model of Filtering and clustering system is presented in the fig. 3.

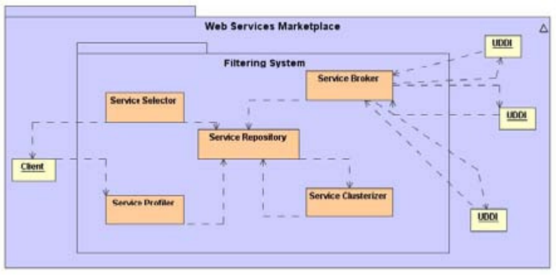

Figure 3. The conceptual model of the Filtering and Clustering System

In order to perform the filtering of newly appearing services, the system needs to store service profiles in the repository. To be able to interact with service providers and collect the information about new services the system should have the broker functionality (active search for new services, being passive source on information for the filtering system, performing all the necessary interactions with providers in order to create system-processable service descriptions and acquiring information about service quality parameters). The other important component of the system is the filterer. Its task is to perform the analysis of QoS constraints and semantic matching between two, potentially similar, Web services. Service Clusterizer creates groups of similar profiles. The last important element of the filtering system is a Service Profiler. This component is used by clients to create profiles of composite applications that use Web services.

The idea behind the prototyping was to use as many open source tools to increase the popularity of the solution. The prototype of implemented system was written in Java with use of XPath and MySQL as the database server. In its earliest version the system had Web-browser interface, as this way of communication is extremely user-friendly (implemented in Java servlets technology).

The appropriate experiment on the scenario described in the section 3 was conducted. The results were promising. The usage of the clustering algorithm has no impact on the precision and relevance of the system but it speeds up the process of processing the single service. As it works independently of the service discovery and filtering functionality, it may work in the background continuously updating the created clusters taking into account new services in the profiles. A brief summary of all system components is given in following subsections.

5.1 Service Repository

The service repository has two main functionalities: it stores service profiles created by the system clients and stores the information about all services on the market. Initially we considered the division of these two functions into two components. But some factors convinced us to keep everything in the one repository

- the service registered by a user can be also a new service on the market,
- clustering performed on the bigger sample of services gives better results.

Service repository is a database. It also triggers Service Clusterizer when new service appears.

5.2 Service Broker

The broker plays the role of intermediary between providers and the system itself. Broker can actively seek new services or be just passive receiver of notifications from UDDIs. When new WSDL file comes to the system it is automatically converted to OWL-S format. Afterwards, the broker asks provider for giving the information about non-functional properties' values of the service. When it is done, the service description (profile) is stored in the repository. Broker should give an interface that helps providers complete descriptions of their services. This functionality may be also provided as a stand-alone application that converts WSDL services to OWL-S format enhanced with parameters required by the filtering.

5.3 Service Profiler

The Service Profiler's goal is to help client express his needs. A client, through specialized interface defines the properties of Web services of which his application consists. The OWL-S descriptions of every atomic service are stored in the repository. Moreover, a client can define desired values of QoS parameters. Additionally, it is possible to put weights on these parameters, because for example, one client prefers cheap, but less reliable service, whereas other one is able to pay more for more reliable Web service. Every OWL-S file has accompanying vector of preferences. Altogether, they create a profile of an ideal atomic service. Such a profile is later clusterized. This profile is also matched against new services registered in the system.

5.4 Service Clusterizer

This component handles the task of clustering of atomic services stored in the repository. It is worth noting that the criterion of the clustering process is the functionality of a service. Thus, services of different providers can be grouped in one cluster. The granularity of clusters has several levels. The highest level relates to service category, lower ones are created according to the level of semantic equivalence between services in the same cluster. Effects of the clustering process are later taken into account during the filtering. New services are compared only to corresponding cluster. In the effect the number of comparisons is dramatically lower, because for example new payment service is not semantically matched to weather service.

5.5 Service Selector

Service selection is performed in two stages. The first phase, called the ontology-based filtering, is responsible for semantic matching of services functionalities. In the next phase, the non-functional properties are analyzed. When the overall level of match between the new service and the service in profile exceeds threshold value the client gets the notification that new, better service was filtered by the system.

6. Summary and future work

The presented architecture of the Semantic Web services filtering and clustering system may solve some of the problems of the SOA paradigm. The system consists of several components dealing with one aspect of the task. The elements are chained in a workflow that reflexes the step-by-step solution. The results of the filtering process as presented in Abramowicz et al [4] are promising but not as precise as one could wish for, mainly due to the ontology related problems. In our opinion, one of the main problems is to define a precise ontology and service profiles for Web services description so the right services could be matched and filtered according to the requirements and user's preferences. We do not expect that clustering of Web services will be a remedy for all the problems connected with the efficiency of semantic matching algorithms. However, the limitation of the set of compared services can save a lot of time, as reasoning on ontologies is undoubtedly time-consuming process. Our next step is to show the results proving the usability of the clustering in the filtering process.

Additionally, one of our goals is to improve the semantic matching effectiveness by better description of preconditions and effects. Well-prepared financial ontology would be a great tool to achieve this goal. We also plan to extend the list of non-functional features that are taken into account during the constraint-based filtering stage. Works driving at creation of upper and lower ontology of non-functional properties are in progress.

Another question is how the proposed approach deals with composite services, that is, services that are composed by other services that can be discovered and replaced dynamically during the runtime. Orchestration problems can arise during the execution of such composite services when new potential services arise during a discovery process. This is however, the aim of the further research.

7. References

[1] Bachlechner, D., Siorpaes, K., Fensel, D. and Toma, I. Web service Discovery – A Reality Check, DERI Technical Report, January 2006

[2] Berners-Lee, T., Handler, J., Lassila, O., The Semantic Web, Scientific American, May 2001

[3] OWL-S, http://www.daml.org/services/owl-s/

[4] Abramowicz W., Godlewska A., Gwizdała J., Kaczmarek M., and Zyskowski D. Application-oriented Web services Filtering, in Proceedings of the International Conference on Next Generation Web services Practices (NWeSP 2005), pages 63-68, IEEE August 2005

[5] WSMO, http://www.wsmo.org/

[6] WSDL-S, http://lsdis.cs.uga.edu/projects/WSDL-S/wsdl-s.pdf

[7] Paolucci, M., Kawamura, T, Payne, T., Sycara, K., Semantic Matching of Web services Capabilites, In Proceedings of the 1st ISWC, 2002

[8] Gonzalez-Castillo, J., Trastour, D., Bartolini, C., Description logics for matchmaking of services, In KI Workshop on Applications of Description Logics, 2001;

[9] Li, L., Horrocks, I., A software framework for matchmaking based on semantic web technology, In Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary, May 2003.

[10] Bussler, Ch., Maedche, A., Fensel, D., A Conceptual Architecture for Semantic Web Enabled Web services, ACM Special Interest Group on Management of Data: Volume 31, Number 4, Dec 2002

[11] Deng, S., Wu, Z., Li, Y., ASCEND: a framework for automatic service composition and execution in dynamic environment, in Proceedings of International Conference Systems, Man and Cybernetics, 2004, pages 3457-3461

[12] ASG http://www.asg-platform.org

[13] METEOR-S: Semantic Web services and Processes, lsdis.cs.uga.edu/proj/meteor/SWP.htm

[14] Lara, R., Laursen, H., Arroyo, S., de Bruijn, J., Fensel, D., Semantic Web services: Description Requirements and Current Technologies. In International Workshop on Electronic Commerce, Agents, and Semantic Web services, September 2003

[15] Abramowicz W., Godlewska, A., Gwizdała, J., Jakubowski, T., Kaczmarek, M., Kowalkiewicz, M., Zyskowski, D., A survey of QoS computation for Web Services Profiling, In the Proceedings of ISCA 18th International Conference on Computer Applications in Industry and Engineering 2005, Honolulu, USA

[16] Verma, K., Sivashanmugam, K., Sheth, A.. Patil, A., Oundhakar, S. and Miller, J., METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services. Inf. Tech. and Management, 6(1):17–39, 2005.

[17] Lynch, D., Keeney, J., Lewis, D., O'Sullivan, K., “A Proactive Approach to Semantically-Driven Service Discovery”, in the Proceedings of 2nd Workshop on Innovations in Web Infrastructure, May 2006, Edinburgh

[18] Srinivasan, N., Paolucci, M., Sycara, K..., Semantic Web Service Discovery in the OWL-S IDE, HICSS, p. 109b, Proceedings of the 39th HICSS'06, Track 6, 2006

[19] Jaeger, M., Tang, S., Ranked Matching for Service Descriptions using DAML-S. 2004

[20] Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J., Similarity search for web services. In Proc. of VLDB, 2004
[21] Belkin, N. J., Croft, W.B., Information filtering and information retrieval: two sides of the same coin?, Communications of the ACM, 35(12):29-37,1992
[22] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q., Quality driven Web Services Composition, in Proceedings of the 12th international WWW conference, Budapest, Hungary, May 2003,
[23] Liu, Y., Ngu, A.H.H., Zeng, L., QoS computation and Policing in Dynamic Web Service Selection, in Proceedings of the 13th international WWW conference, New York, USA, ACM Press, May 2004
[24] Myeon Kim, S., Catalin Rosu, M., A Survey of Public Web Services, WWW 2004, May 17–22, 2004, New York
[25] SUPER: Semantics Used for Process management within and between EnteRprises, http://www.ip-super.org
[26] Aslam J.A., Pelekhov E., Rus D., The Star Clustering Algorithm for Information Organization Grouping Multidimensional Data, Recent Advances in Clustering, Springer-Verlag, Berlin, 2006
[27] Abramowicz W., Haniewicz K., Kaczmarek M., Zyskowski D., Filtering of Semantic Web Servics with F-WebS System, The Semantic Web: ASWC 2006 Workshops Proceedings, p. 317-324
[28] Semantic annotations for WSDL http://www.w3.org/TR/sawsdl/#Intro
[29] Taush, B., d'Amato, C., Staab, S., Fanizzi, N., Efficient Service Matchmaking using Tree-Structured Clustering, 5th ISWC 2006
Athens, GA, USA, November 5-9, 2006
[30] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y., Pellet: A practical OWL-DL reasoner, Journal of Web Semantics, 2006.
[31] Jaeger, M., Tang, S., Ranked Matching for Service Descriptions using DAML-S. 2004
[32] Sebastiani F.: Machine Learning in Automated Text Categorization, ACM Computing Surveys, 2002
[33] Noy N. F., Klein M.: Ontology evolution: Not the same as schema evolution, Technical Report SMI-2002-0926, Stanford Medical Informatics, 2002
[34] Srivastava B., Koehler J.: Web Service Composition Current Solutions and Open Problems, ICAPS 2003
[35] Kaufman L., Rousseeuw P.: Finding groups in data: an introduction to cluster analysis, New York: John Wiley and Sons, 1990.
[36] Al-Masri, E., Mahmoud, Q.H.,Investigating Web Services on the World Wide Web, WWW 2008.
[37] Abramowicz W., Haniewicz K., Kaczmarek M. and Zyskowski D. E-Marketplace for Semantic Web Services, Service-Oriented Computing – ICSOC 2008: 6th International Conference, Sydney, Australia, December 1-5, 2008.
[38] Abramowicz W., Haniewicz K., Kaczmarek M., Zyskowski D., Architecture for Web Services Filtering and Clustering, International Conference on Internet and Web Applications and Services (ICIW'07)