SEARCHING FOR SEMANTIC WEB SERVICES - A GOOGLE BASED APPROACH

SINUHE ARROYO[1], HAN SUNG-KOOK[2], DIETER FENSEL[1]

[1]DERI
Technikerstraße 21a,
6020, Innsbruck, Austria
{sinuhe.arroyo, dieter.fensel}@deri.org
http://www.deri.org

[2]Won Kwang University,
South Korea
www.wonkwang.ac.kr

Abstract. Semantic Web Service discovery and selection are a very time and resources consuming task. They require reasoning support for the matchmaking of the capabilities of services against user defined goals and constituent sub-goals, and for the mediation of the domain knowledge used to describe the different relevant aspects of services. This paper presents a performance study around the number of times the reasoner has to be used in nowadays initiatives. Such study lays the basis for an innovative approach inspired in the popular search engine Google, which tries to improve the performance of the whole process. The main idea is to carry the reasoning as an off-line task, storing the output for later reuse. It also elaborates on the idea of making service descriptions and goals available independently of registries or repositories, i.e. Web pages. Such idea permits to profit, extend and further reuse, well established concepts developed by popular search engines, thus assimilating service discovery and selection to any other type of search engine task.

Keywords: Semantic Web Services, WSMO, Google, Search engines.

1 Introduction

The combination of Semantic Web Technology and Web Services has been termed Semantic Web Services. Semantic Web Services count with the ability to change the Web into a common platform where agents (organizations, individuals, and software) communicate with each other to carry out various activities. In order to successfully use Semantic Web Services its relevant information needs to be described in a machine understandable and processable way. The semantic markup of services by means of ontologies facilitates the machine-processability and machine-understandability that added on top of Web Services permits to publish, discover, select, mediate, compose, execute, monitor, replace, compensate and audit services, for the benefit of some agent who seeks to fulfill some user-defined task conceptualized as a goal. The set of steps that put together the different relevant aspects for the use of Semantic Web Services has been termed, Semantic Web Service Usage Process. Such process will allow the development and execution of a value added services that will solve increasingly complex tasks by making available for discovery new composed Services. In order to successfully discover services, a detailed description of its relevant information, namely, service capability, interfaces, non-functional properties, goals and constituent sub-goals needs to be published. Current approaches to publication are based on services registries that store a partial description of the service, and goal repositories that store goals and constituent sub-goals. Once services are published, the discovery phase tries to match the capabilities of the different available services against the description of the goal that the end user aims to achieve. Finally, during selection and based on the non-functional properties of the service, the most appropriate ones, e.g. the most reliable and cost effective ones, among the discovered services are selected. A common feature to discovery and selection is that a reasoner engine is required in order to match goals and capabilities (discovery), and to mediate among domain specific terminologies (selection and discovery). Essentially the user goal is decomposed into constituent sub-goals which need to be individually matched against the capabilities of registered services, requiring the alignment of the terminology used to describe capabilities and sub-goals. During the selection phase mediation is also required in order to align the different vocabularies used to describe non-functional properties. It could be included as part of the goal-capability mediation in order to save resources. In general, the number of times that the reasoner needs to be used for goal-capability matching and mediation is really big, growing as the number of available services and vocabularies grows. In this paper an approach based on the popular search engine Google is presented, which tries to solve some of the performance limitations presented by discovery. It revolves around the idea of making the process in an off-line fashion, keeping a list of goals and sub-goals that include references to the services whose capabilities can satisfy them, minimizing the impact of the reasoning in the discovery and selection tasks. It also proposes to publish the description of services and goals independently of the registries and repositories respectively, making it available like any other Web resources, i.e. Web page. This approach allows the reuse of the concepts and ideas already developed for search engines, permitting to extend them and further reusing them in regard to Web Services, thus understanding the goal-capability matching like any other type of search. A performance study in regard the number of times the reasoner has to be used in the case that previously carried reasoning tasks are not stored is presented, setting the basis that allow elaborating and justifying the necessity of a new proposal.

The contents of this paper are organized as follows: Section 2 introduces the concept and basic ideas around Semantic Web Services. Section 3 sketches the main ideas behind the semantic usage of services, together with the requirements for describing them, putting it all together in relation publication, discovery and selection. Section 4 presents a performance study that motivates the rest of the work. Section 5 depicts the main components of the Google search engine. Section 6 draws a solution to improve the performance of the discovery and selection of services, inspired in the Google search engine and depicts its main building blocks. Finally, section 7 presents the conclusions and future work.

2 Semantic Web Services

The combination of machine processable semantics provided by the Semantic Web with current Web Service technologies has coined the term Semantic Web Services. Semantic Web Services offer the means to achieve a higher level of value-added services by adding dynamism to the task driven assembly of inter-organization business logics, thus making the Internet a global, common platform where agents (organizations, individuals, and software) communicate with each other to carry out various activities.

Ontologies enable the machine processable semantics that added on top of current Web Services realize the idea of the Semantic Web Services. Semantic Web Services are defined as "*Decoupled, semantically marked-up Web Services, with concrete execution semantics, that can be published, discovered, selected, composed, mediated and executed across the Web in a task driven way, carrying its interaction [1] following a choreographed or orchestrated approach*".

In order to fully realize the ideas behind the Web Service Usage process, different aspects of services need to be described in order to allow its interoperation. In the following section the most relevant aspects for publication, discovery and selection are sketched.

3 Semantic Usage of Services

The semantic markup of services allows the description their capabilities and interfaces for the benefit of some agent, who seeks to discover it, determine how to execute it and additionally may want to combine it with other services in order to produce some aggregated functionality.

The process of publishing, discovering and executing services carried with the aim of fulfilling some user-defined task, conceptualized as a goal, has been termed Semantic Web Service Usage Process. The aims of the service usage domain are wide, not being limited to just publish, discovery and execution. These core steps are complemented with selection, composition, mediation, monitoring, replacement, compensation and auditing, thus covering all the different aspects involved. The

3.1 Semantic Web Service Description Requirements

In order to allow the location of services suitable to accomplish a user defined task, different important aspects of services need to be described. Such description should be done, taking as basis the formalism and domain independence provided by ontologies.

Service publishers are in charge of describing the main aspects of services, and making that information available, so it can be further discovered and selected. Among the most relevant aspects that need to be described are counted [3]:

- **Capability:** Description of a Semantic Web Service by means of its functionality, based on pre-conditions, post-conditions, assumptions and effects.
- **Goal:** Specifies the objectives that a client may have when he consults a web service based on post-conditions and effects.
- **Interface:** Specification of how the functionality of the service can be achieved, by means of fulfilling its capability. It takes a twofold approach based on choreography, or how to communicate with the web service in order to consume its functionality, and orchestration, which defines how the overall functionality is achieved by the cooperation of more elementary service providers.
- **Non-functional properties**: Properties related to quality aspects of Web Service (QoS).

3.2 Publication

In order for successful discovery to occur, the publisher needs to facilitate a description of the relevant information of the service/s that is going to be made available, namely, service capability, interfaces, non-functional properties, goals and constituent sub-goals.

Current approaches to service publication are based on traditional UDDI registries semantically enhanced. Service registries make available a reduce amount of information about service descriptions, being the detailed depiction stored at the service provider. Such registries provide interfaces for both the service publisher and service requester, presenting a centralized approach that permits to find matching elements.

The same approach is followed in the case of goals, being also stored in goal repositories. Thus, once the description of a new service is made available, its corresponding goal and constituent sub-goals need to be made available in goal registries, thus facilitating discovery and enable its further reuse. Also, when services are composed, the corresponding new description and goals is made available in the service registries and goal repository respectively.

It is important to notice that the goals and sub-goals that a service makes available in the publication phase, which that can fulfill in principle, are the pre-conditions and effects of the capability of the service, renamed to post-conditions and effects.

3.3 Discovery

Discovery can be understood as the matchmaking task of the capabilities of services (pre-conditions and assumptions) against the goal (post-conditions and effects) that the end user aims to achieve. The idea is to locate a selection of services that by themselves or in combination with others –provide a solution to a sub-goal– allow accomplishing a particular goal. For this, the user goal is decomposed in sub-goals that are submitted as discovery queries. Matching should be as complete as possible, not being just restricted to simple string matching, but supporting subsumption and other techniques in order to carry more complex queries that return better matches. The discovery phase returns a set of candidate services whose functional characteristics can fulfill the goal. Due to the fact that there will be most likely thousands of different ontologies for a concrete domain, –publishers will describe the capabilities of services using specific domain knowledge and the goals provided to service requesters will be as well expressed by means of different vocabularies– the appropriate support for domain knowledge mediation among capabilities and goals must be available.

Matching requires reasoning support both for goal capability resolution, as well as domain knowledge mediation, that it is not provided as part of the registry functionality.

3.4 Selection

Once the list of matching services has been retrieved, a selection process is to be carried based on the non-functional properties of the service, with the aim picking the most appropriate ones, e.g. the most reliable and cost effective one. A common feature of the services that reach this phase is that they can be used as a solution to a sub-goal. In case no appropriate services are found, or the ones found don not provide a solution to each one of the sub-goals, the service requester should inform the end user, who should either refine the description of the discovery query, in case the partial solutions are not sufficient, or proceed with a group of services. Heterogeneity still remains in this part of the usage process. Services are provided by different vendors with different characteristics, and probably for a different domain, thus requiring mediation facilities, to make their non-functional properties understandable to the service requester.

4 Performance study

The approach introduced in section 3 for publication and discovery presents an important limitation in terms of performance when it comes to carry the reasoning process required for the matchmaking of the user goal and the capability of the service. Matching is carried by a reasoning engine for each one of the sub-goals. In an environment with a couple hundreds of services restricted to a concrete application domain where services are distributed over few registries, the approach could be feasible. Nevertheless, in a real setting where possible hundreds of thousands of services are available to solve a part of the goal, distributed over a number of registries, the reasoning becomes an intensive time and resources consuming task, same for network and computational power. Prior to the goal-capability matching, in some cases, mediation should be applied, to translate among domain specific terminologies. Due to the fact that ontology mediation also relies heavily on reasoning, the whole process becomes even more resource and time consuming.

The reasoning required to align the meaning of non-functional properties is understood to be carried as part of the goal-capability matching, only in case the service satisfies partially or completely the goal.

Let's imagine a setting with *n* service registries, with *c* denoting the number of capabilities of services in the each one of the registries. The total number of terminologies used to describe the capabilities of the different available services throughout the registries is denoted by *ts*. The goal can be decompose into *g* sub-goals expressed using a concrete domain terminology.

In the best case all the capabilities available in the registries exactly match the un-decomposed goal. In case no ontology mediation is required, all the capabilities will be expressed using the same terminology as the goal, it will be necessary to carry the reasoning (*n* * *c*) times, this is the number of registries times the number of capabilities found in each one of them. In the average case mediation will be required *ts/2* times, thus requiring reasoning (*n* * *c* * *ts/2*) to align terminologies, and (*n* * *c* * *ts/2*) + (*n* * *c*) times in total. In the worst case mediation will be required (*n* * *c* * *ts*) times for mediation and *(n* * *c* * *ts*) + (*n* * *c*) times over the whole discovery process.

In the average case, the goal will be decomposed into *g/2* sub-goals requiring each one of them to be individually matched against each one of the capabilities of the services registered. Then the reasoning will have to be carried (*n* * *g/2* * *c*). Same as before in case they all use the same terminology no mediation has to be put in place being this the best possible scenario. In the average case, mediation will be required *ts/2* times, thus requiring reasoning (*n* * *g/2* * *c* * *ts/2*) times, and a total of (*n* * *g/2* * *c* * *ts/2)* + (*n* * *g/2* * *c*) times counting alignment and matching. In the worst case mediation will be required (*n* * *g/2* * *c* * *ts*) times, and the whole process will require (*n* * *g/2* * *c* * *ts*) + (*n* * *g/2* * *c*) times the support provided by the reasoning engine.

In the worst case the goal will be decomposed into its most primitive components requiring to carry the reasoning (*n* * *g** *c*) times. Again if the language is the always the same no mediation is required. In the average case, mediation will be required *ts/2* times, thus needing reasoning support (*n* * *g* * *c* * *ts/2*) times, and a total of *(n* * *g* * *c* * *ts/2*) + (*n* * *g** *c*) times. In the worst case, mediation will be required (*n* * *g* * *c* * *ts*) times, and thus the total process will involve (*n* * *g* * *c* * *ts*) + (*n* * *g** *c*) calls to the reasoner.

As can be derive from the results presented, current approach to discovery is not feasible. Even though registries could include replication facilities that minimize the network overload, and grouping of services by capability and domain knowledge used to describe them, thus restricting up to some extent the number of services whose capabilities should be considered for reasoning, still the whole process would be really heavy and resource consuming.

A workaround that could alleviate this problem revolves around the idea of storing previously carried reasoning tasks and performing the goal-capability reasoning as an off-line task. Besides, it is interesting to consider the idea of not storing service description and goals in registries and repositories respectively, but make them available just like any other Web resource, i.e. Web page. Thus the semantic description would be published and discovered from the provider site extending and further reusing the concepts already developed by popular search engines such as Google or Yahoo. In this sense the goal-capability matching task is understood as any other type of search, which profits from well established concepts.

Section 6 presents a proposal that tries to alleviate these problems by combining the concepts and architecture of Google introduced in section 5, with Semantic Web Services.

5 How Google works

In this section the main building blocks of the Google search engine are introduced as a basis to inspire the approach presented in section 6.

The Google search engine is made of three different parts namely [2]:

- **Googlebot:** It can be understood as a crawler, event though it does not crawl the web. Its functionality is divided into three different steps. First it sends requests to Web servers to get a concrete web page, then downloads the page and finally hands it to the indexer. Once a page has been downloaded the links found on it are added to a queue in order to be further requested. Due to the different nature of Web pages available, and in order to keep indexes up to date, Googlebot must determine how often a web page must be re-indexed, a newspaper content changes more dynamically than a personal Web page. The Googlebot makes use of a number of computers requesting and fetching pages in parallel.
- **Indexer:** The content of the pages downloaded by the Googlebot is stored in the index database where it is sorted alphabetically by terms including references to documents where the term appears.
- **Query processor:** User queries are forwarded to the index servers, where are matched against the terms indexed retrieving the reference of the pages that include the term. Then, the query is submitted to the document server which retrieves the stored documents presenting the search results to the user. Google uses PageRank to determine the relevance of documents with respect to a concrete query. It takes under consideration the popularity of the page or the proximity of search terms to one another in the page among others. It uses a voting policy in which every link from one page to another is considered a vote that increase the relevance of the voted page. The relevance of a vote is also determined by the importance of the voter, weighing more the votes of well-know pages than the vote of less popular ones.

6 A Google based approach

Current approaches to service discovery present a serious limitation in terms of efficiency when it comes to reasoning as introduced in section 4. This section presents an approach inspired in the popular search engine Google, which tries to minimize the impact of the reasoning in the discov-

ery task by making the process in an off-line fashion and keeping a list of goals and sub-goals that include references to the services whose capabilities can satisfy them.
The concepts elaborated here revolve around the idea of having the description of services and goals accessible at the provider side and independently of registries and repositories, just like any other Web resources, as already presented.
The main building blocks of the service search engine are depicted in the following lines.

- **WSbot:** It performs the same function as the Googlebot but in this case for Web Services. It crawls the Web searching for Web Services. It sends requests to servers in order to get the description and reference to the service, passing it to the Goal-Indexer.
- **Goal-Capability Reasoner:** It is responsible for the reasoning support required to match the goal or sub-goal with the service(s) that can satisfy it.
- **Domain Reasoner:** It takes care of the alignment of the different terminologies used to describe goals and the capabilities of services. Its functioning is required before the Goal-Capability Inference engine can carry on its work. By having two separate reasoners performance can be improved.
- **Query processor:** It translates the end user discovery query to concrete application domain knowledge and decomposes it into sub-goals. Each one of the sub-goals is handed to the Goal Indexer that matches them against available goals, retrieving the list of references to services that fulfill the goal and presenting the search result to the user. Another approach is to present to the user a set of goals for reuse among the ones stored in the Goal-WS database for a concrete application domain, being then the task of the translation engine and search engine in general, much simpler and efficient. Since there will probably be a number of services satisfying the same goal, the selection of the most suitable ones is achieved based on the non-functional properties of the service, expressed by the end-user in the discovery query. The popularity of the service should be taken under consideration to rank results. The proposed approach is based on the calculation of the number of times the service is selected as a solution to the goal.
- **Goal Indexer:** The goals and sub-goals translated by the Query processor, and also the ones ex-

tracted from the capability of the services found by the WSbot, are handed to the Goal Indexer who is responsible for storing them in the Goal-WS database in case they do not already exit. The Goal-WS database is indexed by goal, having each one attached a set of references to Web Services that can fulfill it. Every time the WSbot gets a new service, its capability is matched against the stored goals and sub-goals, adding a reference to the Web Service in the Goal-WS database in case the capability and the goal are compatible. This process takes place off-line. In case there is no matching among the capability of service and exiting goals in the repository, a new entry is added for each one of goal/sub-goals of the capability. A similar procedure is applied for the goal and sub-goals of the discovery query. In case the goal or constituent sub-goals are found in the Goal-WS database the list of services is handed to the query processor who is responsible to present it to the user, as already stated. In case the goal or one of the sub-goals does not exist in the Goal-WS database, the Goal Indexer will add a new entry for each one of them, and will try to find matching services in the WS database. This matching process is also carried off-line, thus returning the Goal indexer an empty list of services for the corresponding sub-goal of the query. The goal-capability matching relies on the Domain reasoner to align terminologies and the Goal-Capability reasoner to carry on the matching. In order to make the search process more efficient goals could be grouped by domain.

The architecture presented in this section tries to provide a more scalable an efficient solution to the discovery of Web Services, by reducing and trying to perform as off-line as possible the reasoning required. The underlying concept revolves around publishing the descriptions of the services and the goals they satisfy independently of registries or repositories, just like in the case of any other Web resource, being the task of a dedicated search engine to find services, and to match them against goals. Nevertheless, this approach could as well be applied to services whose descriptions are stored in service registries.

7 Conclusions and future work

In this paper a new approach for the discovery and selection of Semantic Web Services based on the architecture of the popular search engine Google was presented. It tried to improve the performance of both processes, overcoming current limitations of the reasoning process required for the goal-capability matchmaking, and the mediation required for translating among the different domain specific terminologies, used to describe the relevant aspects of services.

The paper presented a performance study based on the number of times that the reasoning process needs to be carried in a real setting, from which the inefficiency and unfeasibility of current approach is derived. It took under consideration the best, average and worst case. Further, the paper proposed a workaround to alleviate the problem, by reducing the number of times the reasoner is required. The driving concept revolves around storing previously carried reasoning tasks, and performing the reasoning required for the goal-capability and mediation as an off-line task. This approach is complemented with the idea of making the description of services and goals available independently of registries or repositories, like any other Web page. Such initiative allows profiting and reusing the concepts and ideas developed in popular search engines, thus assimilating discovery and selection to regular search tasks. Finally an architecture inspired in the popular search engine Google was presented, which depicted the main building blocks and the relation among them.

In regard to future work is worth to mention that based on the ideas elaborated in this paper, a new proposal for a EU funded project will be submitted, with the aim of implementing the approach presented. Still there is a lot of work to be done in order to improve the performance and efficiency of the usage process of services. An important open point revolves around improving the reasoning involved in the remaining parts of the usage process. A solution could possibly follow a similar approach, in terms of storing the results of already carried reasoning, in these cases mainly required for mediation.

Acknowledgements

The work is funded by the European Commission under the projects DIP (IST – 2004-507483), Knowledge Web (IST – 2004-507482), Ontoweb (IST – 2000-29034), SEKT (IST – 2004-50826, and SWWS (IST 2001-37134); by Science Foundation Ireland under the DERI-Lion project and by the Austrian government under the Co-Operate program.

References

1. S. Arroyo, R. Lara, J. M. Gómez, D. Berka, Y. Ding,, D- Fensel, D.: *Semantic aspects of Web Services, in Practical Handbook of Internet Computing*. Munindar P. Singh, editor. Chapman Hall and CRC Press, Baton Rouge, 2004.
2. Google Inc. How Google works. http://www.googleguide.com/google_works.html, 2003.
3. D. Roman, H. Lausen, H. and U. Keller (eds): "*Web Service Modeling Ontology*". WSMO Working Draft v0.3. http://www.wsmo.org/2004/d2/v1.0/, 2004 .