PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 28/05

Swell, SwellOnt and SwellQL - a Software Engineering Environment for Searching Semantic Web Services

João Felipe Santos Condack

Daniel Schwabe

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 28/05 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena August, 2005

Swell, SwellOnt and SwellQL - a Software Engineering Environment for Searching Semantic Web Services*

João Felipe Santos Condack, Daniel Schwabe

condack@rdc.puc-rio.br, dschwabe@inf.puc-rio.br

Abstract. In this report, we describe the Swell environment. It is a software engineering tool for searching semantic web services. In order to accomplish this, we developed a specialized ontology and a query language, named SwellOnt and SwellQL respectively. The environment is available as an extension of Eclipse.

Keywords: Web services; semantic web; components; tool; software engineering; ontology; web services description.

Resumo. Nesta monografia é descrito o ambiente Swell. Trata-se de uma ferramenta de engenharia de software para busca de web services semânticos. Para a sua realização, foi desenvolvida uma ontologia especializada e uma linguagem de consulta, chamadas SwellOnt e SwellQL respectivamente. O ambiente está disponível como uma extensão da plataforma Eclipse.

Palavras-chave: Web services; web semântica; componentes; ferramenta; engenharia de software; ontologia; descrição de web services.

* This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil

In charge for publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Introduction

The internet originally was a communication channel used for data exchange, functioning as a text-based content dissemination vehicle. With the advent of e-commerce and the Web's unfolding, it became a systems deployment platform. Web services have been used more recently as a way to provide functionality in a "componentized' way. According to [9] one of web services' cornerstone is the on-the-fly software development through reusable and lowly coupled components. Our main motivation is to answer the following question: How to explore those new opportunities enabled by these evolutions of both Software Engineering and the Internet?

We focus our attention to a specific question. According to [6] one of the service integration challenges in open systems is the discovery of such services. Moreover, the authors also state that a critical factor, among many to be overcome, is exactly the description of services. Finally he forcefully points out the following questions: "How systems will operate since there are multiple formalisms (and it is pretty sure to happen)? How can those multiple formalisms be composed?".

This motivates us to explore the development of web services descriptions that are evolution tolerant and universal enough to encompass existing formalisms. Furthermore, one of the most important contributions of this approach is its use in searching for services. With this in mind, we have developed the Swell ontology and its accompanying environment.

The Swell environment helps Web Based Systems development. Its cornerstone is the semantic selection of web-services. The environment is a tool whose goal is to help in the application design and development tasks, aiming to reach a high degree of reuse. This is achieved through the support for the description, search and selection of web services for application composition. The Swell environment was conceived as a framework with support for the evolution of web components technologies. It provides hot spots that allow for adaptation of new web services descriptions and reflecting of these changes in the search engine.

The remainder of the work is organized as follows. Section 2 briefly describes the theoretical basis from this paper; Section 3 explains the SwellOnt ontology; Section 4 presents SwellQL query language; next, in Section 5, implementation issues are shown; and finally Section 6 briefly compares existing work and draws some conclusions.

2 Developing Component Based Web Applications

This section outlines the relevant concepts to this work, namely, Web Based Systems (WBSs); Component Based Development; Web Services and Semantic Web; and a Development Process. It is a background section for what is new and interesting about the Swell approach described in subsequent sections.

2.1 Based Systems (WBSs)

According to [40] the WWW is a project that joins hypertext and information recovery techniques in order to easy the creation of global information systems. To do so it uses HTTP - Hypertext Transfer Protocol - that allows browsers to request and receive information.

Technologies like CGI, scripts, plug-ins and application servers represented what was web usage evolution.

In this paper we deal with another evolutionary step. "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format [41]. Particularly we will be talking about semantically enriched Web Services.

2.2 Component Based Development

The maturation of the WWW has contributed for a wider perception of distributed computation, which is one of the factors that led organizations to construct its applications using third party components [5]. Therefore Software Engineering faces a scenario where component based systems find a fertile ground in the new web technologies. Web services have become an alternative which is capable to bind both concepts: component oriented development and the WWW as a deployment platform. It seems clear that components technology finds a new form of implementation as web services, as implied in [39].

Some problems in component based development are solved by the web services architecture, such as:

- The ability for consumers to post requests for components [12].
- The ability for the component developer to publish correct information about components such as available interfaces and services [12].
- Supporting distributed and parallel development, due to independence among developers;
- Using best component available, given the possibility to describe and subsequently to search for a specific service.

One problem raised by using the web services architecture is that integration tests cannot be carried out (at least straightforwardly) due to the late biding of components from independent sources.

2.3 Web Services and Semantic Web

Regarding the web (also) as a repository of services, it is reasonable to expect that, with its expansion, it will be necessary to compose and reuse these services in increasingly more precise and efficient ways.

According to [2] the Semantic Web is the evolution of today's web, providing semantic meta-data to currently available resources. Such meta-data is typically specified through ontologies.

Ontologies cam be specified in languages such as OWL [44]. More specifically, there is an ongoing standard proposal for semantically describing Web Services, OWL-S [7]. The Swell environment can provide the developer with more interesting search mechanisms by exploring this semantic enrichment attached to a service description.

According to [22] the semantic vision of web services is based on the description of its capacities and content in non ambiguous ways, interpretable by machines. That is what we aim to achieve with the SwellOnt ontology.

2.4 Development Processes Supported by Swell

The Swell environment is oriented towards supporting component oriented development processes. To help understanding, we outline one such process, but it should be clear the other processes could equally be supported.

In our experiment, the basic phases involved in this development process are as follows: Requirements Gathering, Selection of Components and System Composition; the selection is further subdivided into Description, Search and Choice. This process follows existing proposals in other areas, considering that:

- Given that web services can be implemented following COTS ("Component Off-the-Shelf") concepts, it makes sense that systems development processes with COTS can be applied to web service development, as discussed in [4][3][13][17].
- Developing systems using components leads the development process to emphasize the selection of parts rather than constructing new code. In extreme cases, this composition could be made almost automatically, as proposed by [19][20][25][33][36]. In this work, however, we partake the view that, at least for some time, the development process will still involve human intervention. This difference is evidenced in the focus of the development process: while we focus the search of services, the above-mentioned authors focus on service composition.
- For the of service description aspect, several proposals exist, such as [7][9][6][26][42]. Such variety suggests the use of a more abstract description approach, which allows accommodating any of these (and future) proposals. Consequently, we defined an ontology for service characteristics description called SwellOnt, presented next.

3 The SwellOnt Ontology

To accomplish the search activity, first it is necessary to define how searched elements will be described. There are several kinds of web services descriptions: OWL-S, WSMO [31], METEOR-S [26] and WSML [42] are just a few. The possibilities vary not only at the same abstraction level, due to the different approaches, but also at various diverse abstract levels. OWL-S and WSMO can be considered descriptions at the same level of semantic enrichment, whereas WSML is lower level than both since it deals with the description in a level closer to the implementation.

As mentioned, the approach taken in SwellOnt is to specify a generic and search oriented service description ontology, which can be mapped onto any of the existing approaches. Next we will detail its classes and also present an example instance.

In SwellOnt, service is described as a set of characteristics, which can be either functional or non-functional. Each is further specialized, as shown in Figure 1. This ontology, based on both OWL-S and WSMO, extends both by borrowing concepts from studies in components area [5][45], more specifically in COTs [1][17].

Next we describe each class in SwellOnt in more detail.

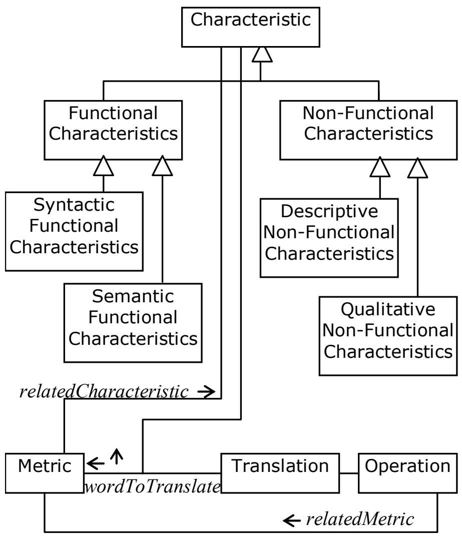

Characteristic
Functional Characteristics
Non-Functional Characteristics
Syntactic Functional Characteristics
Semantic Functional Characteristics
Descriptive Non-Functional Characteristics
Qualitative Non-Functional Characteristics
relatedCharacteristic
Metric
wordToTranslate
Translation
Operation
relatedMetric

Figure 1 SwellOnt classes and their relations

3.1 Characteristic

It is an abstract class that will generate a structure of characteristics aimed at services description. A synonym for "Characteristic" would be "Requirement", however since we are interested in a description connotation, we believe that "Characteristic" is a better term.

3.2 Functional Characteristic

Defines **what** services must do considering its systemic nature. It is the characteristic associated with service's core activity and are divided in two groups: Syntactic and Semantic.

The syntactic functional characteristics characterize the service through the information that is consumed and generated during service execution.

Table 1 Instances of Syntactic Functional Characteristics

Functional Characteristics	
Syntactic Functional Characteristics	
Deal with information change process	
Input	Information that is consumed by the service
Output	Information that is produced by the service

According [18] the characteristics above are represented in OWL-S by "inputs" and "outputs". However in WSMO they are equivalent to "pre-conditions" and "post-conditions" respectively.

The semantic functional characteristics deal with environment state change due to the execution of the system.

Table 2 Instances of Semantic Functional Characteristics

Functional Characteristics	
Semantic Functional Characteristics	
Deal with state change reflected in the environment	
Precondition	State of the environment before the service execution
Post condition	State of the environment after the service execution

Still in accordance with [18], the state change is expressed in OWL-S using preconditions and effects. WSMO describes this trough "assumptions" and "effects" respectively. It is important to notice that a service can have 0..N occurrences of any of the four functional characteristics.

3.3 Non-Functional Characteristic

They are characteristics that describe meta-information about the service or are associated with how the service must behave. The Non-Functional Characteristics can be either Descriptive or Qualitative.

Table 3 Instances of Descriptive Non-Functional Characteristics

Non-Functional Characteristics	
Descriptive Non-Functional Characteristics	
Describe service meta-information.	
Title	Name by which the service is known. In OWL-S it is equivalent to "serviceName"
Creator	Entity responsible for the creation of the service. In OWL-S it can be expressed by the element "contactInformation"
Subject	Keywords or phrases that can be associated to the service
Description	Summary of the service. In OWL-S it is equivalent to "textDescription"
Publisher	Entity responsible for the service's availability. In OWL-S it can be expressed by the element "contactInformation"
Contributor	Service's contributors. In OWL-S it can be expressed by the element "contactInformation"
Date	Date associated with some event of the service lifecycle, as for example, its creation or publication
Type	Identification of some element in a categorization schema to which the service belongs. In OWL-S it is equivalent to "serviceCategory"
Source	Reference to the source code location or origin of the service
Language	Language in which the service operates
Relation	Reference to other related services
Coverage	Scope of the system, can include information regarding to geographic localization, time or jurisprudence among others
Version	Service's version. In OWL-S it is equivalent to "versionInfo"

Qualitative Non-Functional Characteristics describe how the service must behave in terms of the offered quality. To describe how the service must behave during its execution, we collected the literature on the subject [18][21][28][35][38][31][34], generating the following list of qualitative non-functional characteristics:

Table 4 Instances of Qualitative-Non Functional Characteristics

Non Functional Characteristics	
Qualitative Non Functional Characteristics	
Scalability	Service's ability of being capable to take care of an increasing band of solicitations
Capacity	Limit of Requests that the service can take care of simultaneously
Performance	Measures the speed of fulfilling of the service
Reliability	Service's ability in executing the specified functions
Availability	It is related to the probability of the service to be ready u-

	pon request
Robustness/ Flexibility	It is the capacity of the service to operate correctly when receiving unexpected inputs (e.g.. invalid, incomplete or conflicting data)
Exception handling	Evaluates how the system answers in unexpected situations. It can be categorized in three types: brutal, graceful or absent
Accuracy	It is the capacity to answer requests correctly. It is inversely related to the volume of produced errors
Integrity	It is the capacity associated with a service to prevent non authorized modifications to the code or the data. It guarantees the coherence of the data and states before and after the execution of solicitations
Regulatory	Associates to the service an alignment level according to some regulation
Supported Standard	It indicates the accepted standards for the service and how this relation happens
Stability	Frequency of changes in the service
Guaranteed Messaging	It describes how and if the service guarantees the delivery, order and persistence of the messages
Cost	Measure of the financial cost related to the service execution.
Completeness	It is related with the difference between the volume of functionalities effectively implemented and of functionalities requested
Security	It is about a characteristic that involves others things among: Authentication Authorization Confidentiality Accountability Traceability and Auditability Data encryption Non-Repudiation

It is pertinent to observe that some of the qualitative non-functional characteristics are part of the WSMO definition of semantic web services already. Those are: scalability, performance, reliability, robustness, accuracy, integrity, cost, authentication, authorization, confidentiality, traceability and encryption.

3.4 Metric

Metrics provides a measure of comparison of services over some characteristic. In the case of qualitative non-functional characteristics they are strongly tied on the service quality. Besides *name* and *description* a metric also has a related characteristic, for e.g.:

- Time-To-Repair (TTR) is a metric that measures Availability, a qualitative non-functional characteristic;
- Error Rate is metric that measures Accuracy, also a qualitative non-functional characteristic;

- NAICS Code [NAICS] (North American Industry Classification System) is another metric that can measure the descriptive non-functional characteristic named Type.

3.5 Translation

Translations define mappings between ontologies with different levels of semantic enrichment, SwellOnt being the richest one. It is also used to associate operations on characteristics (or metrics) with lower level executable statements.

The translation will allow queries on the characteristics to be generated at execution time by the tool. When a query using SwellOnt characteristics or metrics needs to be performed this task is done by the environment translating a template statement with place holders. For example to translate a query containing…

```
Input=http://www.daml.org/services/owls/1.0/Concepts.owl#Airport
```

…a Translation instance should have

```
wordToTranslate: Input
operation: equality ( = )
executableQueryLanguage: RQL
templateSentence: select Y from {X} @P1 {Y}, {X} @P2 . @P3 {W} where  @P1 =
service:presentedBy and @P2 = profile:hasInput and @P3 = process:parameterType
and W = <!-- PLACE HOLDER--> using namespace service =
http://www.daml.org/services/owl-s/1.0/Service.owl# , profile =
http://www.daml.org/services/owl-s/1.0/Profile.owl# , process =
http://www.daml.org/services/owl-s/1.0/Process.owl#
```

In order to add new underlying implementation languages in Swell it is just a matter of instantiating new elements of Translation class, similar to the example above.

3.6 Operation

This class defines comparison operations that are accepted for an instance of Metric. For example: The metric "Error Rate" accepts integer comparison operations such as "<", ">", "=". Another metric "Compatibility" could have values such as "specialization", generalization" and "equivalence", evaluated over a class hierarchy. Associating the symbols "<", ">", "=" to this operations, "<" would represent specialization, ">" generalization and "=" class equivalence.

4 The SwellQL Query Language

To facilitate searching for services described in SwellOnt, a specific language was defined, called SwellQL. It helps the discovery of services whose characteristics have been described using the SwellOnt ontology.

The main advantage of SwellQL is that, since it is specialized, queries are more concisely written, and can range over services described in several formalisms. In general terms queries in SwellQL resemble a function call where the parameters are instances of SwellOnt classes.

Next we will describe the signature of an operation called "findSevices" that is defined in SwellQL. This operation does the search for services, and is the only operation in the language..

A portion of the syntax of SwellQL is given by the BNF specification shown next:

Table 5 Portion of SwellQL's BNF

```
query               ::= "findSevices " paramList

paramList           ::= param
                      | {"(" paramList ")" logic_op "(" param ")"}

logic_op            ::= "and" | "or" | "minus"

param               ::= functional_param comparison_op uri
                      | numeric_nonfunctional_param comparison_op number
                      | integer_nonfunctional_param comparison_op integer
                      | temporal_nonfunctional_param comparison_op time
                      | literal_nonfunctional_param "like" literal
                      | "ExceptionHandling" comparison_op eh_value
functional_param
                    ::= "Input" | "Output" | "Precondition" | "Postcondition"
```

It is important to note that SwellQL language expresses selections only; it does not have operation for insertions, exclusions or updates. Moreover, the selection is theoretically restricted to only one kind of object, that is, a service. This, restriction is consistent with the objective of SwellQL, namely, the search for services. By now we are not considering new extensions but in the future it could be rethought.

Some examples of statements using SwellQL:

- `findServices (Input= http://www.daml.org/services/owls/1.0/Concepts.owl#Airport)`
- `findServices (Creator like "John Doe" or  Publisher like "John Doe" or Contributor like "John Doe")`
- `findServices (MaxConcurrentRequests > 10)`

5 The Swell Environment

The Swell environment was built as a framework-like fashion, providing hot spots so that the environment can adapt to new web services descriptions, and reflecting this adaptation in the search mechanism. All this concern is consonant with the web services architecture requirements [43]) that are being studied by the W3C.

In order to create a software engineering environment we used the Eclipse platform [8] which is an extensible IDE with a flexible open source framework, written in Java. Through this framework it is possible to add, to exclude or to modify extensions called plug-ins. With such extensions, it is possible to modify Eclipse itself to build Swell environment functions. Another technology used was the Jena library [15] allowing the manipulation of data described in RDF. Also Junit [16], Sesame [32], Protégé [27] and JavaCC [14] were used.

Figure 2 presents an overall vision of the Swell environment.

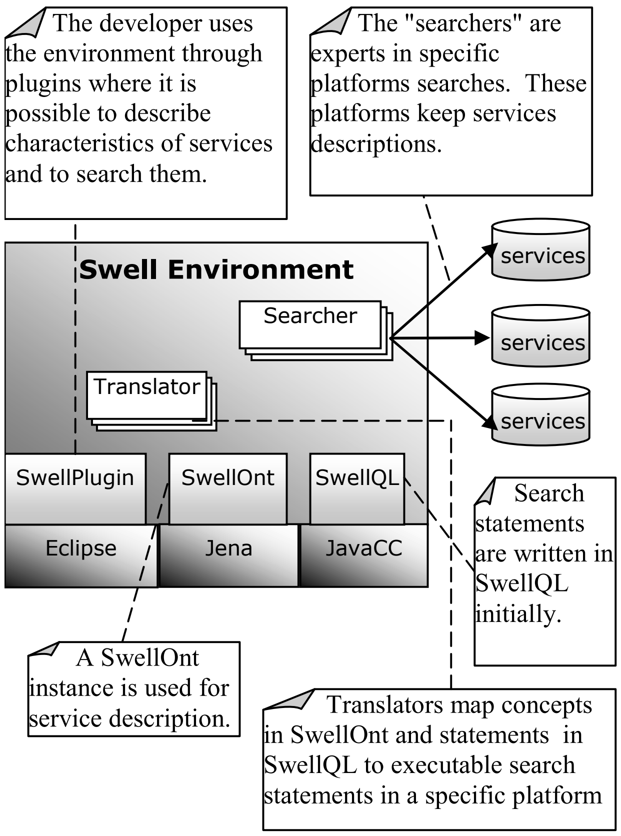

The developer uses the environment through plugins where it is possible to describe characteristics of services and to search them.
The "searchers" are experts in specific platforms searches. These platforms keep services descriptions.
Swell Environment
Searcher
Translator
services
services
services
SwellPlugin
SwellOnt
SwellQL
Eclipse
Jena
JavaCC
Search statements are written in SwellQL initially.
A SwellOnt instance is used for service description.
Translators map concepts in SwellOnt and statements in SwellQL to executable search statements in a specific platform

Figure 2 Software architecture of the Swell Environment

5.1 Implementing SwellOnt

The SwellOnt ontology implementation was made in two stages. First the description presented in the section 3 was edited in Protégé [27] , and then classes were written in Java to represent and manage ontology terms.

Using Protégé we generated:

- an RDFS description of the classes;
- an RDF instance of SwellOnt, the one described in tables 1-4. In particular, the instance was used by the environment to generate Java objects corresponding to the ontology resources.

Since Swell is data-driven, it is flexible in two different ways:

1) Any change in the RDF specification of the ontology instance is reflected in Swell's interface, without changes in code or recompilations. In particular, it becomes very easy to change possible search parameters.

2) Translation objects, generated from the RDF instance, contain statement templates, with placeholders; written in some executable query language. The placeholders are substituted by real values when necessary, allowing a flexible search scheme, built at execution time. This also allows changing the translation strategy without requiring changes in the environment itself.

5.2 Implementing SwellQL

To implement SwellQL language the JavaCC tool was used. . We developed a grammar specification from the BNF, and the information presented in section 4 . A file containing this specification was processed by JJTree program, creating a new specification and classes for tree processing. This new specification was given as input to the JavaCC, which in turn, finished the parser construction.

6 Conclusions

The approach shown in this paper has several advantages, which we discuss next. At the end, we show future research directions being followed.

6.1 SwellOnt Ontology

As previously discussed, there are various distinct ways to describe web services. Given this range of possibilities, the SwellOnt ontology takes a more abstract description approach, allowing the generalization of descriptions in any of the existing proposals.

In that way, the proposed solution was the creation of a description ontology generic enough to take care of service description. Thus the SwellOnt, a web service search oriented description ontology was born. Moreover, the ontology also served to make the environment flexible: it defines interface issues and tool behavior in execution time.

Another SwellOnt ontology application, not explored in this paper, is in web services mediation. When two services described in different languages need to interact, they should do it through mediators. The SwellOnt ontology can be used as a common semantic reference into which each service description can be mapped. It worth to observe that WSMF framework [9] foresees the use of mediators for some tasks. It is not expected from developers to use SwellOnt so mediators are needed to do necessary ontology translations. One sort of mediators are the environment Translator's.

Another important point is the definition of an extensible non-functional characteristics taxonomy, with associated metrics, which allows specifying (and querying) Quality of Service parameters.

6.2 OWL-S Extension for Non-Functional Characteristics

The OWL-S description ontology does not contain a satisfactory range of non functional characteristics, which can be overcome through extensions mechanisms available in OWL-S itself. We have defined an extension for OWL-S with the non functional characteristics described in the SwellOnt instance.

The extension was written, as provided in OWL-S, using RDF sentences and OWL vocabulary for class definition. The classes created in the extension were chosen in order to be also a superset of the non functional characteristics contained in WSMO.

6.3 SwellQL Query Language

The SwellQL language aims to facilitate formulating queries for searching web services. This is done through SwellOnt oriented statements.

There are a number of other languages for the semantic web. RQL is a query language based on the graphs underlying model of RDF. It allows to search for instances of RDF and also in the underlying RDFS model. The same occurs with other languages, all of them are based on some model, for example:

- DAML Query Language (DQL)[10],
- OWL Query Language (OWL-QL) [10],
- Sesame RDF Query Language (SeRQL) [32],

- RDQL [15]

The SwellQL is also based on a model, characterized by the SwellOnt, whose advantage is to be platform independent and oriented towards service search. By virtue of its specialization, SwellQL allows much more concise queries than general purpose languages such as RQL or RDQL. By way of comparison, below we show similar search queries written in SwellQL and RQL.

The purpose of this first pair is to find services that have, as an input, a particular OWL class or any subclass of it:

```
SwellQL: findServices (input < = http://www.daml.org/services/owl-
s/1.0/Concepts.owl#Airport)


RQL: select Y
from  {X} @P1 {Y}, {X} @P2 {Z}, {Z} @P3 {W}
where  @P1 = service:presentedBy and @P2 = profile:hasInput and @P3 =
process:parameterType and W IN subClassOf ( http://www.daml.org/services/owl-
s/1.0/Concepts.owl#Airport ) using namespace
  service = http://www.daml.org/services/owl-s/1.0/Service.owl# ,
  profile = http://www.daml.org/services/owl-s/1.0/Profile.owl# ,
  process = http://www.daml.org/services/owl-s/1.0/Process.owl#
```

Next, the second pair, searches for services which handle more than ten concurrent requests:

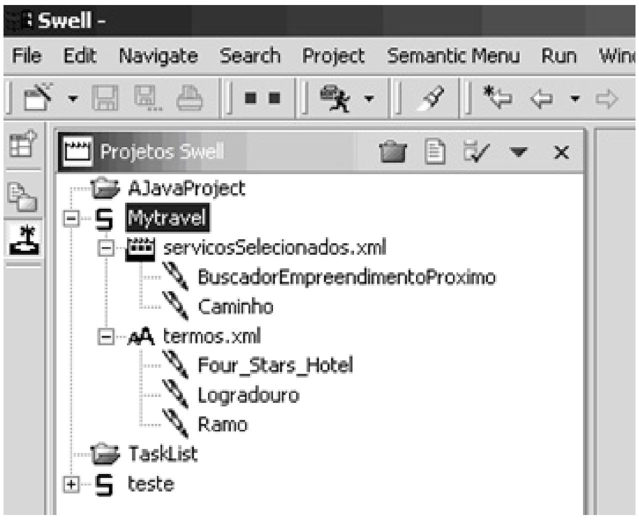
```
SwellQL: findServices (MaxConcurrentRequests > 10)


RQL: select Y , @P3, W
from {X} @P1 {Y}, {X} @P2 {Z :$Z}, {Z} @P3 {W}
where  @P1 = service:presentedBy and @P2 = profile:serviceParameter and $Z =
nonFuncPar:MaxConcurrentRequests
using namespace
  service = http://www.daml.org/services/owl-s/1.0/Service.owl# ,
  profile = http://www.daml.org/services/owl-s/1.0/Profile.owl# ,
  process = http://www.daml.org/services/owl-s/1.0/Process.owl# ,
  nonFuncPar = http://www.tecweb.inf.puc-
rio.br/swell/1.0/NonFunctionalParameters.owl#
```

6.4 Swell Environment Prototype

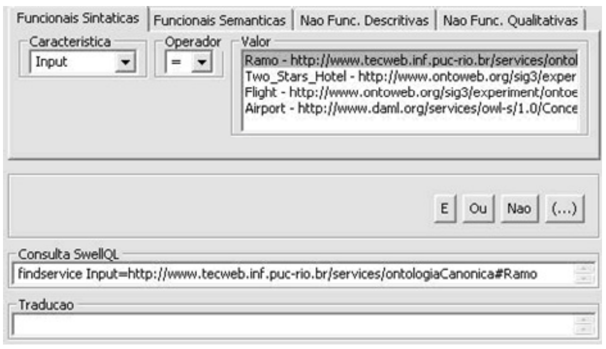
A first version of Swell environment was developed using the Java language. To create a software engineering environment we used Eclipse platform. Using its plug-in architecture, we extended it to perform Swell related functions.

Swell, as we saw, is a tool to assist in the WBS development process aiming to reach high degree of reuse. The prototype developed also provides hot spots allowing adaptation for new descriptions of web services and the reflection of these changes in the search mechanism. Below there are some snapshots from the prototype:



AJavaProject
Mytravel
servicosSelecionados.xml
BuscadorEmpreendimentoProximo
Caminho
termos.xml
Four_Stars_Hotel
Logradouro
Ramo
TaskList
teste

Figure 3 Semantic Web Service Project View



Caracteristica: Input
Operador: =
Valor:
Ramo - http://www.tecweb.inf.puc-rio.br/services/ontol
Two_Stars_Hotel - http://www.ontoweb.org/sig3/exper
Flight - http://www.ontoweb.org/sig3/experiment/ontoe
Airport - http://www.daml.org/services/owl-s/1.0/Conce

Consulta SwellQL:
```
findservice Input=http://www.tecweb.inf.puc-rio.br/services/ontologiaCanonica#Ramo
```
Traducao

Figure 4 Search Query Configuration Panel

6.5 Future Work

Specifically with regard to services composition, it would be interesting to handle it in order to facilitate service development. This composition could be made under two views - static, with the fixed definition of components and construction of adapters; or dynamic, during execution time through mediators. The environment could suggest compositions, or a search result it could also be an adequate composition, in style similar to the Web Service Composer [36].

Another interesting possibility is to transform the prototype into a framework and to document how to build its extensions. So a developer would be able to create their own translators and adapt the tool to other services descriptions. A hot spot that could be explored would be the creation of a searcher capable of carrying researches in UDDI repositories.

Another possibility is to allow the user to set weights in search parameters. With this the tool would be able to order the query results in more interesting way to the user. This idea was inspired in quality evaluation framework from [24] and in the research about semantics and similarity concept calculation proposed by [29] and [30] respectively.

One of the prototype problems is characteristic measurement: there is no unit of measure when configuring search parameters. The literature we read does not have this concern in regard to service description also. A solution would be adding unity data in instances of Metric and than create mediators to handle the conversion between different units.

Another issue is by the time we developed Swell there was not enough resources of semantic web services descriptions so we were not able not perform truly experimental tests. Those tests would help to answer how better the approach proposed really facilitates discovery of web services.

References

[1] ALVES, C.; CASTRO, J. CRE: A Systematic Method for COTS Selection. In XV Brazilian Symposium on Software Engineering (Rio de Janeiro, Brazil, Out, 2001). http://www.cs.ucl.ac.uk/staff/C.Alves/SBES'01_Alves.pdf.

[2] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web, (2001). http://www.scientificamerican.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[3] BRAUN, C.L. A lifecycle process for the effective reuse of commercial off-the-shelf (COTS) software. Proceedings of the 1999 symposium on Software reusability (Los Angeles, California, United States, 1999) 29-36.

[4] BRITTENHAM, P. Web Services Development Concepts (WSDC 1.0). IBM Software Group (2001). http://www-306.ibm.com/software/solutions/webservices/pdf/WSDC.pdf.

[5] BROWN, A.W. Component-Based Software Engineering - Selected Papers From theSoftware Engineering Institute. IEEE Computer Society Press (California, 1996)

[6] CONSTANTINESCU, I.; WILLMOTT, S.; FALTINGS, B. Abstract Behavior Representations for Service Integration. Agentcities: Challenges in Open Agent Environments (Springer-Verlag 2003), p 25-31.

[7] DAML-S Coalition. DAML Services - Releases of DAML-S / OWL-S. http://www.daml.org/services/owl-s/.

[8] ECLIPSE Eclipse.org Main Page. http://www.eclipse.org.

[9] FENSEL, D.; BUSSLER, C. The Web Service Modeling Framework WSMF. http://informatik.uibk.ac.at/users/c70385/wese/wsmf.paper.pdf.

[10] FIKES, R. (Ed); HAYES, P.(Ed); HORROCKS, I(Ed). OWL-QL – A Language for Deductive Query Answering on the Semantic Web. Knowledge Systems Laboratory (Stanford University, Stanford, CA, 2003).

[11] FIKES, R. (Ed); HAYES, P.(Ed); HORROCKS, I.(Ed) DAML Query Language (DQL) - Abstract Specification. Knowledge Systems Laboratory, 2002.

[12] HEINEMAN, G.T.; COUNCILL W.T. (Ed). Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley. 2001.

[13] IRIBARNE, L., VALLECILO, A., ALVES, C., CASTRO, C. A Non-functional Approach for COTS Components Trading. In IV Workshop on Requirements Engineering (Buenos Aires, Argentina. 2001) 124-138.

[14] JAVACC. javacc: JavaCC Home. https://javacc.dev.java.net/

[15] JENA. Hewlett-Packard Development Company Jena 2. http://www.hpl.hp.com/semweb/jena.htm.

[16] JUNIT. Junit testing framework. http://junit.sourceforge.net/

[17] KONTIO, J. A case study in applying a systematic method for COTS selection. Proceedings of the 18th international conference on Software engineering (May, 1996) 201-209.

[18] LARA, R.; ROMAN, D.; POLLERES, A. D4.1v01 Conceptual Comparison WSMO/OWL-S. DERI. Working Draft (2004). http://www.wsmo.org/2004/d4/d4.2/v01/.

[19] LI, L.; HORROCKS, I. A Software Framework For Matchmaking Based on Semantic Web Technology. WWW2003 (Budapest, Hungary, May, 2003) 331-339.

[20] MANDELL, D.J.; MCILRAITH, S.A. A Bottom-Up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. In The Proceedings of the Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW '03)(Budapest, 2003). http://www.daml.org/services/pubs/www2003sam-djm-workshop.pdf.

[21] MANI, A.; NAGARAJAN, A. Understanding quality of service for Web services. Desenvolvido por IBM (2002). http://www-106.ibm.com/developerworks/library/ws-quality.html.

[22] MCILRAITH, S.; MARTIN, D. Bringing Semantics to Web Services. IEEE Intelligent Systems (jan./fev 2003), number 18, 90-93.

[23] NAICS. North American Industry Classification System. http://www.census.gov/epcd/www/naics.html

[24] OLSINA, L. Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web. Tese de doutorado (UNLP, Argentina, 2000).

[25] PAOLUCCI, M.; SYCARA, K.; KAWAMURA, T. Delivering Semantic Web Services. WWW2003(Budapest, Hungary, Mai, 2003).

[26] PATIL, A.; OUNDHAKAR, S.; SHETH, A.; VERNA, K. METEOR-S Web Service Annotation Framework. WWW 2004 (New York, USA, May, 2004) 553-562.

[27] PROTÉGÉ. The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/.

[28] RAN, S. A Model for Web Services Discovery With QoS. ACM SIGecom Exchanges Volume 4 , Issue 1, 1-10. 2003.

[29] ROCHA, C.; SCHWABE, D.; DE ARAGÃO, M. P. A Hybrid Approach for Searching in the Semantic Web. WWW 2004 (New York, NY USA, May, 2004) 374-383.

[30] ROCHA, C.; SCHWABE, D.; DE ARAGÃO, M. P. Integrating Semantic Concept Similarity in Model-Based Web Applications. LA-Web 2004 (Ribeirão Preto, SP Brazil, Oct,2004)

[31] ROMAN D.(Ed); LAUSEN, H.(Ed); KELLER, U.(Ed) D2v03. Web Service Modeling Ontology - Standard (WSMO - Standard). DERI. http://www.wsmo.org/2004/d2/v0.3/20040329/.

[32] SESAME. openRDF.org: News. http://www.openrdf.org/.

[33] SHESHAGIRI, M.; DESJARDINS, M.; FININ, T. A Planner for Composing Services Described in DAML-S. In AAMAS Workshop on Web Services and Agent-Based Engineering (2003). http://www.agentus.com/WSABE2003/program/sheshagiri.pdf.

[34] SHETH, A.; CARDOSO, J.; MILLER,J.; KOCHUT, K.; KANG, M. QoS for Service-oriented Middleware. The 6th World Multiconference on Systemics, Cybernetics and Informatics, Proceedings Vol. 8, (Orlando, FL, July 14-18, 2002) 528-534.

[35] SINGHERA, Z.U. Extended Web Services Framework to Meet Non-Functional Requirements. In Proceedings of the 2004 International Symposium on Applications and the Internet Workshops (SAINTW'04) 334-341.

[36] SIRIN, E. Web Service Composer. Mindswap - Maryland Information and Network Dynamics Lab Semantic Web Agents Project. http://www.mindswap.org/~evren/composer/ (2004).

[37] SIRIN, E.; HENDLER, J.; PARSIA, P. Semi-automatic Composition of Web Services using Semantic Descriptions. In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003 (Angers, France, April, 2003).

[38] SUMRA, R.; ARULAZI, D. Quality of Service for Web Services—Demystification, Limitations, and Best Practices. http://www.developer.com/services/article.php/2027911.

[39] SZYPERSKI, C. Components and Web Services. http://www.sdmagazine.com/documents/s=825/sdm0108c/.

[40] W3Ca W3C Home Page. World Wide Web Consortium. http://www.w3.org/.

[41] W3Cb Web Services Glossary. World Wide Web Consortium. http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/.

[42] W3Cg Web Services Description Language (WSDL) 1.1. World Wide Web Consortium (2001). http://www.w3.org/TR/wsdl/.

[43] W3Ch. Web Services Architecture Requirements. World Wide Web Consortium. http://www.w3.org/TR/wsa-reqs/#csfsreqs.

[44] W3Ci. Web Ontology Language (OWL). World Wide Web Consortium. http://www.w3c.org/2004/OWL/

[45] YANG, J. Web service componentization. Communications of the ACM, Volume 46, Number 10, 35-40. 2003.