An Approach to Dynamic Discovery of Context-Sensitive Web Services

Victor G. da Silva, Carlos E. Cirilo, Antonio F. do Prado,
Wanderley L. de Souza
Computer Science Department
Federal University of São Carlos (UFSCar)
São Carlos, Brazil
Email:{victor.silva, carlos_cirilo, prado, desouza}@dc.ufscar.br

Vinícius Pereira
Institute of Mathematical Sciences and Computing
University of São Paulo (USP)
São Paulo, Brazil
Email: vpereira@icmc.usp.br

***Abstract*— With the Internet becoming increasingly present in people's lives and the growing availability of Web Services (WS), new challenges have emerged for Software Engineering regarding application development based on composition of highly reusable services within the so-called Service-Oriented Architecture (SOA). One such challenge refers to how to automatically accomplish WS discovery at runtime in order to compose personalized application features that meet particular user's requirements as his/her context of interaction changes. To tackle this issue, we propose in this paper an integrated approach that addresses dynamic WS discovery by combining the traditional WS technology stack with conceptions of Semantic Web on top of the UbiCon, a framework that supports context-sensitive behavior and supplies application with contextual information at runtime. The proposed approach counts on two main elements: (i) an algorithm that performs the dynamic discovery of context-sensitive WS; and (ii) a reusable software architecture that provides a skeleton which enables applying (i). The intention of this approach is to reduce development efforts and increase productivity as it encourages reusing pre-fabricated WS, as well as to improve software quality whereas applications are assembled as a set of services extensively tested and validated.**

***Keywords*-Web Services Discovery; Semantic UDDI; Context-Sensitivity; SOAP; RESTful**

I. Introduction

The Web has become one of the biggest media in the world, transforming the way in which people communicate and share content. Since its beginning, the Web has gone through some phases to fit it to recent technological possibilities and to attend new users' requirements. One can mention, for instance, the evolution from the traditional Web to Web 2.0 that is marked by greater interactivity, collaboration and communication among users [1]. This reality has driven to the need for applications that support large amounts of information to different users and devices across multiple contexts of use. This issue has attracted the interest of many researchers over the years. The academic community has proposed different solutions. Some of them, for instance, address frameworks (e.g., [2], [3], [4]) that support development of context-sensitive applications. Meanwhile, the software industry attempts to solve the problem of wide variety of access devices putting into practice the Service Oriented Architecture (SOA) [5] by means of the Web Services (WS) technology, so as to enable integration of heterogeneous systems, fostering greater interoperability.

SOA is widely used in the industry and has the WS as the main way for its realization. The model introduced by the WS technology has a well-defined and structured architecture. In spite of that, some issues still remain for SOA-based software development: One of the main challenges is to perform WS discovery to properly satisfy users' needs, especially at runtime when the context of interaction constantly changes. Semantic Web [6] arises to help out filling this gap, enabling integration among WS and Semantic languages. The Semantic Web is an extension of the Web, which allows computers and humans to work in cooperation, resulting in a better user experience.

It is noticeable that many systems technologies are integrated to provide users with better interactions. On this basis, we present in this paper an integrated approach to enable semantic annotation of WS for context-sensitive Web Services discovery. The approach combines the traditional WS-* stack [7] with Semantic Web technologies to provide developers with an architecture which enables application performing, at runtime, the dynamic discovery of WS in order to foster adaptable behavior based on the users' context of interaction. The framework UbiCon [4] has been applied on our proposal to furnish the contextual information needed to reach context-sensitivity.

The technologies used in the research are presented in the following sections. Section II presents the main concepts regarding the WS technology. Section III presents description languages of Web Services. Section IV broaches concerns about Universal Description, Discovery and Integration (UDDI) and WS discovery. Section V presents conceptions regarding Context and Context-Sensitive Systems. Section VI details our proposal. Finally, Section VIII presents final remarks and further work.

II. Web Services

WS have emerged as a technology that enables the realization of the Service Oriented Architecture (SOA), whose main purpose involves integrating heterogeneous systems and delivering applications and features as loosely-coupled services by means of open standards of the Internet (e.g., HyperText Markup Language – HTTP; eXtensible Markup Language – XML; etc). The World Wide Web Consortium (W3C) [8] defines a Web Service as a software designed to support interaction machine-to-machine over a network. WS are independent of the implementation language, which favors interoperability among disparate systems. WS use standard protocols for message exchange. They are identified by a Uniform Resource Identifier (URI), available in a standardized service descriptor, along with the signatures of its features and functions.

There is a well-known architectural model for WS that is based on the interaction of three main components: *Registry*, *Provider* and *Client*. Registry is a repository of WS in which service Providers can post descriptions of their services; Clients, in turn, can query the Registry in order to find out the descriptions of the services they are looking for. The Providers are the ones who furnish the services, which are materialized as Web Services and achievable by some Provider software agent. Clients perform requests to the WS, which involves getting a description of a service from the Registry and then invoking the Web Service functionalities by means of a client software agent for messaging and communication with the Provider agent.

The interaction between these components occurs through some operations: *Publish*, *Find* and *Bind*. When the Provider offers WS, providing a description that is published in the Registry, a *Publish* operation is performed. *Find* operation occurs when a Registry is used by the Client to discover and to retrieve information about services of interest stored in published service descriptors. Finally, the *Bind* operation happens when the Client uses the service description to bind with the Provider and interact with the implementation of the Web Service. WS can be based on SOAP or REST, as explained in the following subsections.

A. Web Services SOAP

The Simple Object Access Protocol (SOAP) [9] is a set of conventions defined by the W3C for exchanging messages that are transmitted and negotiated over a network on the top of the HTTP protocol. Information exchange between the client and the service provider is based on XML format and happens over decentralized and distributed environments in Remote Call Procedures (RPC) style. The SOAP packages the messages to be exchanged in a standard envelope structure, presenting itself as a simplified and lightweight mechanism for exchanging structured information.

By using SOAP, both the application server and the client must be able to interpret the messages structure. This requires the developer to implement appropriate software agents that communicate and understand the protocol's details, resulting in greater efforts to evolve the system. For this reason, using SOAP as a standard technology for WS development has become a deprecated practice. On the other hand, even with recent technology achievements on the WS development, SOAP still has its usage and contribution in software industry and shall be considered in service discovery.

B. Web Services RESTful

RESTful WS are based on the Representational State Transfer (REST) architecture, defined by Roy Fielding [10]. This kind of Web Service has similar characteristics to the SOAP WS, but RESTful ones are lighter and easier to access [11], [12]. A RESTFul Web Service focuses on the service's resources, rather than on its functionalities, and are naturally transferred by the HTTP through its methods GET, PUT, POST, and DELETE. RESTFul WS are gaining momentum, both from the research community and companies [13], which are adopting REST because of its easiness and simplicity of publication, invocation and maintenance [14], [15], [16]. Deploying RESTful WS is quite similar to deploying a dynamic website [11].

Despite of its widespread architectural model, the traditional Client-Provider-Registry architecture for WS is falling into disuse. Client businesses are making off-line agreements and no longer need to query the service *Registry*, because they know in advance the WS' location and how to access them. There are, still, on-line documentation, as well as websites of developers that publish how to access their services, which avoids the use of descriptors and mitigates the need of conventional service registries – which does not adhere to the traditional WS architectural model [11], [12], [17]. This new "model" hinders the dynamic discovery of WS. Since this article deals with service discovery, both SOAP as RESTful must be able to be understood and processed computationally, enabling dynamic discovery. For this end, semantic annotations are used, allowing contextual processing of WS.

III. Description Languages Syntactic and Semantic

The Web Services architectural model provides a means of communication between their organizations, exchanging information through messages and descriptors Web Services. The descriptors describe, syntactically, how to access the service and what the service provides. The W3C has standardized the format descriptor, naming the Web Service Description Language (WSDL) to describe web services in a structured way. According to the W3C [18] WSDL defines an XML grammar and a model for describing network services as collections of communication endpoints capable of exchanging messages. A description of Web Service is a document by which the provider communicates the specifications for the client to invoke the Web Service, thereby defining how the interaction between them should be, how and where to access, and what the input data and output are.

Still thinking of the architectural model of Web services, service discovery happens through a search for keywords in the *Registry*, preventing the discovery of content. Faced with this problem, the proposal is to use semantics to represent the content of Web Services, enabling the dynamic discovery of services.

The following are the subsections that address the technologies used for syntactic and semantic description.

A. WADL and WSDL Descriptors

The first version of WSDL emerged in mid-2001 and allowed to only describe SOAP-based Web Services. Soon after came the REST-based Web services, RESTful Web Services appointed, using much of the characteristics of the HTTP protocol for messaging between *Provider* and *Client*, which provided an updated version of WSDL to also support the description of Web RESTful Services, called WSDL 2.0.

In addition to the WSDL, which is the descriptor of Web Services standard, there is the WADL proposed by Hadley, to provide a description that is can be processed by machine, and RESTFul resources that is based on HTTP [13]. A large number of web-based companies (Google, Yahoo, Amazon and others) have used RESTful [13] to provide access to its internal data, but use documentation with web sites, instead of using the WADL. WADL is different because it provides an interface in XML, describing an application and not a service, mapping the concepts that form the RESTful paradigm [1].

Even though WADL describes easier and more efficient RESTful Web Services, WSDL describes both SOAP-based Web Services as RESTful, which facilitates the discovery and invocation of both service types, justifying the use of standard WSDL. Finally, it is worth mentioning that a drawback in dynamic discovery of Web Services is that only descriptors include syntactic description of their services, thus needing a semantic description.

B. Ontologies and OWL-S

Ontologies have a very important role in automated discovery and composition of Web Services. They are the ones that describe and give knowledge to the computer that can process and understand the content of Web Services, thus being able to discover, compose and invoke the available services automatically. The most used definition of ontology in the literature on services semantic web is: Ontology is a collection of Web Services that share the same domain of interest and describe how Web Services can be described and accessed [19].

The semantic description is essential for efficient discovery of Web Services, and occurs through ontologies, which are built from a markup language. Research has shown that the ontology OWL-S [20], which extended the Web Ontology Language (OWL) adding semantics, is more efficient in the context of semantic description to Web Services. OWL-S is an ontology for Web Services that lets you discover, compose, monitor and invoke services with self-degree of automation. It is composed of 3 elements: *Service Profile* - announces and discovers services, *Service Model* - describes service operations, *Service Grounding* - provides details of communication protocols and message format.

Joining Web Services with semantic language have Semantic Web Services. The semantic services deal the limitations in current Web Services through the improvement of the description of services, defining a semantic layer, in order to achieve automatic processes of discovery, composition, execution and monitoring [21]. To solve this problem scientists have used four attributes described by the semantic class Service Profile of OWL-S: *Inputs*, *Outputs*, *Preconditions* and *Effect* (IOPE) [22], [23], [24], [25]. Certainly there are functions that are unique to OWL-S, and the main one is the mapping between OWL-S specifications and syntactic specifications of WSDL, which allows the integration of ontologies with Web Services. This mapping is one descriptor to one ontology. Furthermore, the *Service Profile* settings matter are defined in other Semantic Service Profiles and other ontologies, thereby facilitating reuse and avoiding ambiguities [22].

Finally, we have Semantic Web Services that can be searched by content and not just keywords, increasing the power of discovery Web Services in Registry.

IV. UDDI

Web Services should stay available in order to be accessed by any client application present on the Web, using a static or dynamic way. The static mode to access a Web Service is characterized when the service address is provided to the client application to locate and access the service, i.e., the client application needs to know the Web Service address to access and invoke it. The service can also be discovered, characterizing the dynamic mode of discovery a Web Service. Universal Description Discovery and Integration (UDDI) is a platform-independent structure built for dynamically describe and find published services, which are contained in the architectural model of Web Services. According to OASIS [26], UDDI specifications, define a *Service Registry* for Web Services. An UDDI Registry manages information related to service providers, service implementations and their data. *Providers* can use UDDI to advertise their services and *Clients* can use it to find services, which meet their requirements. In this way, UDDI refers to a repository of Web Services provided by companies and their descriptors.

Dustdar et al. [19] asserts that although UDDI Registry and other UDDI based-models have been implemented, they are not widely used, and for the service dynamic discovery they do not meet the requirements yet. The UDDI-based discovery is performed using keywords. This search includes Web Services with low relevance and ambiguity. To solve this problem there are researches related to dynamic discovery through semantics based on ontologies. The discovery provides semantic search through content and not only by keywords, facilitating the understanding by the computer [20]. The web service discovery depends on the service descriptor, in this case the WSDL. WSDL complements the UDDI standard by providing an uniform way for describing the interface and the connection between *Provider* and *Client*. Since the UDDI specifications do not define a semantic processing for Web Services, UDDI should be extended in order to understand the service semantic context, allowing the dynamic discovery of semantic services within the UDDI.

V. Sensitivity to Context

With the explicit presence of the Internet in daily life, the use of applications that contain a large amount of information is growing rapidly, as well as the need of users to do complex tasks and to process information in a short time, thus creating a challenge to computational systems. The challenge aims to reduce the need of explicit interaction of the user with the system to get what it want [27]. Faced with this problem, researchers have created new systems approach, which aims to understand the context in which the user is in order to assist in the needed actions. These systems are called Context-Sensitive Systems [27] or Context Aware Systems [2]. This article refers to the term Context-Sensitive Systems because it reflects better the semantic of a system which detects context changes, adapts itself and reacts to these changes [3].

A. Context

The context is the key to filter the available information and turn them into relevant information. In the literature there are several definitions of context. According to Vieira [27], Context is everything that involves a situation at a given moment, allowing in the identification of what is and is not relevant to interpret and understand the situation. Santos [3] also makes a distinction between: context and contextual elements, arguing that Contextual Element (CE) is any data, information or knowledge that allows to characterize an entity into domain. As defined in [2], context is any information that can be used to characterize the situation of an entity (e.g., person, place, object, user application). Context involves information that refer to various aspects associated with the operation of the application, such as user, device access, environment, network, among others [28].

Using these ideas, in computer systems Context is an instrument to support communication between systems and their users. From understanding the context the system can, in many circumstances, change its sequence of actions, style of interactions and type of information provided to users in order to adapt to the current needs. Systems using the context to guide actions and behaviors are called Context-Sensitive Systems or Context-Aware Systems [3].

B. Context-Sensitive Systems

Context-Sensitive Systems (SSC) or Context-Aware Systems are computer systems that use context to provide more relevant services or information supporting user tasks, where these tasks are context dependent [2], [29]. There is a big difference between traditional applications and context-sensitive applications. Traditional applications act considering only requests and information provided explicitly by users. Regard to the context-sensitive applications, they mainly differ in the amount of input and output data. The input data can be the explicit information provided by the user, the information stored in knowledge bases contextual, inferred through reasoning, and also those perceived from environment monitoring [3].

To better illustrate the functioning of context-sensitive applications, it is considered, for instance, the case of an application to help medical appointments, which takes into account the patient preferences and context. The patient registers its data with preferences in a database shared by hospitals and clinics. When the patient searches for a doctor to schedule an appointment, the application acts helping him in his actions. The patient can inform their illness or medical specialty required to filter the search for clinics and centers. The system is located next to the appropriate clinical patient, considering its location and preferences, and search for doctors with the specialties required, considering the doctor's schedule to make an appointment. The system sends alert messages for both the patient and the specialist if the query is scheduled successfully, or in the occurrence of a fault.

C. UbiCon

There are many challenges to develop a context-sensitive system. Researches have been developed for the purpose of generating specific tools and methods to support the treatment of these challenges, such as toolkits [2] and frameworks [3], [4]. These approaches aim to meet basic functionality for the management of contextual information so that applications can make use of services, simplifying the development of such systems. The basic features are listed by Vieira [3], setting them into 3 categories:

- Specification Context: handles the context of requirements gathering and modeling the contextual information needed for the application.
- Management Context: indicates how the context will be handled by the system, in terms of the following tasks: acquisition, storage, processing and dissemination of contextual elements.

- Use Context: defines how context influences the behavior of the system and how it will be used effectively.

Considering that any information can be considered context, the information obtained should be relevant. Santos [3] defines a framework called Contextual Elements Management Through Incremental Modeling and Knowledge Acquisition (CEManTIKA) to capture, manage and disseminate computational context, which is used as the base architecture for context extraction in computational work. Faced with the challenges and motivations for building a context-sensitive application, we developed a framework in the same context of research, called Ubiquitous Context Framework (UbiCon) [4]. The UbiCon encapsulates the manipulation tasks context and provides functionalities divided into 4 modules: acquisition, processing, dissemination and adaptation of content, based on context architecture proposed by Santos [3]. Thus arises a motivation to continue this line of research, reusing the results.

VI. Proposal

This proposal can be best viewed considering his major contributions to computing. They are divided into steps, explained below.

A. Architecture

The proposed architecture is shown in Figure 1. This architectural approach proposes the reuse and extension of the architectural model of Web Services, adding knowledge and context sensitivity, obtained by UbiCon framework, and Web Services search for content using ontologies that are constructed from the interpretation and validation of descriptors service. Thus, the architectural model of Web services is extended for the context, compared with the contents of Web services described using semantic, and they can be interpreted automatically by the computer, achieving the object of discovering dynamic context sensitive.

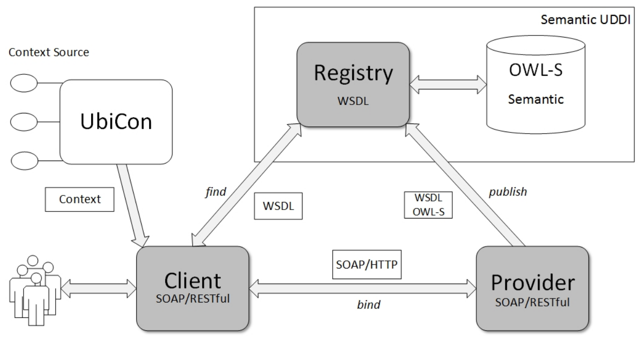

Figure 1. Architectural Model for Dynamic Discovery of Context-Sensitive Web Services.

B. Extending the framework UbiCon

Considering that the service discovery task is dynamic and it varies according to various relevant factors to obtain the contextual information required to be performed, the service discovery must be an adequate automated task. In this way, the UbiCon framework will be extended to retrieve contextual information that will guide the discovery process of services, getting a higher volume of contextual information about the entities associated with the operations of the discovery process of services. Therefore the discovery mechanism may consume contextual information obtained by the modules Acquisition, Processing and Dissemination UbiCon, and perform the discovery of services which meet the contextual variations observed. To allow the automation in discovery tasks, the contextual information involved in the execution of the application (e.g., user profiles, device profiles, network access and SLA) should be available to facilitate reasoning and inference about the same runtime.

One of the biggest challenges in Ubiquitous Computing, cited by Gimson et al. [30], is a description of the delivery context, which is defined as a set of attributes that characterize the capabilities of the access device, the user preferences and other aspects related to the delivery of Web content. To address these problems and in order to achieve the proposed goal, this paper proposes the use of ontologies for describing the characteristics of the delivery context, focusing on the discovery of services. Ontologies allows to express concepts and relationships of domain and turn them computable. This work will build ontologies to represent knowledge about the different profiles and services.

The specification of the ontologies that characterize the Profiles and Services have bases on previous work in building the Extend Internet Content Adaptation Framework (EICAF) [22]. The EICAF is a content adaptation framework for Web applications based on Web services and ontologies. In EICAF, the contextual information about the domain entities considered (e.g., device, user, SLA, network, content) are available through profiles specified in OWL, being easily reused in OWL-S. These ontologies will be refined and extended in this work to meet the requirements of dynamic discovery of services and the context of the application.

C. Discovery of Context-Sensitive Web Services

The Discovery of Context-Sensitive Web Services is the main purpose of the proposed work. It happens when the client wishes to invoke some kind of service that meets their situation or context in which it appears. The *Listener Client*, which is waiting and watching if there have been changes in the context, *searches* the service through the *Semantic Registry*, passing as a parameter the *Context* obtained through of the framework UbiCon. The search occurs through the *Service Profile* class, which extends the *Service* class of ontology OWL-S, that has a high degree of specification [22]. In the *Service Profile* there are features (in the form of representation of functions that provides the service) that allows the service discovery, which are: *Inputs*, *Outputs*, *Preconditions* and *Effects* (IOPE). Through these representations, the context information and services are compared, returning the descriptions of services that corresponds the sent context, realizing the discovery of services dynamically.

D. Method and Evaluation

The work is being carried out iteratively and incrementally applying an evolutionary process model [31] for the implementation of the activities identified in the previous subsections. In this sense, it follows that for each iteration of the activities, an increase of work is produced. Each increment is made up of parts of the artifacts resulting from each activity iterated until all activities are carried out completely. The first activity was developed and iterated according to the architecture proposed in this paper. In the following iterations, new artifacts are created, tested, refined and evolved until a full version of the project, in order to validate the proposal.

The validation and monitoring will be done using case studies and experiments developed at the end of each interaction, advocating the hypothesis of the work aimed at discovering dynamic context-sensitive services.

VII. Related Works

Tegegne et al. [32] present an architecture/framework for health systems in countries with poor infrastructure. The article covers health systems that manage patient registration, claiming that these systems are the biggest problem in the health area. The article presents a solution, still immature, with SOA, SOC and Web Services technologies to solve these problems. The authors do not specify what type of technology it is used, but they present an approach very similar to the one proposed in this paper. The part of the context that manages patient information and services, makes a junction between these informations, thus, it is discovering services. Presents a specific service repository that can not be reused.

John et al. [33] define in his work a standard RDF with Microformats aiding semantics for RESTful WS. It describes services using annotations in HTML and work in specific cases, but in generic cases it does not work.

Ferreira Filho et al. [34] create an ontology to aggregate with RESTful WS. Their approach uses an extension of OWL-S Grounding and WADL descriptors. The work does not address SOAP services and does not use the standard WSDL.

In this work, RESTFul and SOAP services are discovered based on the descriptions WSDL2 and OWL-S. The UbiCon framework is used for context management. The repository present in this article can be inserted legacy services, this enables the reuse.

VIII. Final Remarks and Future Works

This paper proposes a new approach to discover Web services, using technologies that enable the use of context and semantics which support searching by content and context-sensitive. This proposal allows the reuse of other sources of finding web services, taking as an input the descriptions of the services, which will be converted to OWL-S, adding semantics and allowing the search for content within the registry. The article discusses problems and challenges of the area, approaching a solution and contributing to the state of the art.

Future work aims to improve the semantics of Web services with the addition of WordNet [35], improving the effectiveness of search by content. Also creating a validator and converter WSDL to OWL-S, so that the service provider can publish their services without the need to publish its ontology.

At the time of publication happens converting the WSDL to standard OWL-S proposed by Martin et al. [20]. Another possible work is to get the context of social networks, increasing the sensitivity of context.

References

[1] O. F. Ferreira Filho and M. A. G. V. Ferreira, “Semantic web wervices: a restful approach,” *Proceedings of the IADIS International Conference on WWW/Internet*, pp. 169–180, 2009.

[2] A. K. Dey, “Providing architectural support for building context-aware applications,” Ph.D. dissertation, Georgia Institute of Technology, 2000.

[3] V. V. Dos Santos, “Cemantika: A domain-independent framework for designing context-sensitive systems,” Ph.D. dissertation, Universidade Federal de Pernambuco, 2008.

[4] C. E. Cirilo, A. F. Do Prado, W. L. De Souza, and L. A. M. Zaina, “A hybrid approach for adapting web graphical user interfaces to multiple devices using information retrieved from context,” in *DMS 2010 - Proceedings of the 16th International Conference on Distributed Multimedia Systems*, 2010, pp. 168–173.

[5] T. Erl, *Soa: principles of service design*. Prentice Hall Upper Saddle River, 2008, vol. 1.

[6] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, “The semantic web,” *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[7] F. Curbera, F. Leymann, T. Storey, D. Ferguson, and S. Weerawarana, *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR Englewood Cliffs, 2005.

[8] H. Haas and A. Brown. (2004) Web services glossary. [Online]. Available: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[9] Y. L. Nilo Mitra. (2007) Soap version 1.2 part 0: Primer (second edition). [Online]. Available: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

[10] R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” in *Proceedings - International Conference on Software Engineering*, 2000, pp. 407–416.

[11] C. Pautasso, O. Zimmermann, and F. Leymann, “Restful web services vs. ”big” web services: Making the right architectural decision,” in *Proceeding of the 17th International Conference on World Wide Web 2008, WWW’08*, 2008, pp. 805–814.

[12] W. Jiang, D. Lee, and S. Hu, “Large-scale longitudinal analysis of soap-based and restful web services,” in *Proceedings - 2012 IEEE 19th International Conference on Web Services, ICWS 2012*, 2012, pp. 218–225.

[13] M. Hadley. (2009) Web application description language. [Online]. Available: http://www.w3.org/Submission/wadl

[14] S. Vinoski, “Serendipitous reuse,” *IEEE Internet Computing*, vol. 12, no. 1, pp. 84–87, 2008.

[15] F. Belqasmi, J. Singh, S. Y. Bani Melhem, and R. H. Glitho, “Soap-based vs. restful web services: A case study for multimedia conferencing,” *IEEE Internet Computing*, vol. 16, no. 4, pp. 54–63, 2012.

[16] A. Rodriguez. (2008) Restful web services: The basics. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-restful

[17] F. Belqasmi, R. Glitho, and C. Fu, “Restful web services for service provisioning in next-generation networks: A survey,” *IEEE Communications Magazine*, vol. 49, no. 12, pp. 66–73, 2011.

[18] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana *et al.*, “Web services description language (wsdl) 1.1,” 2001.

[19] S. Dustdar and W. Schreiner, “A survey on web services composition,” *International Journal of Web and Grid Services*, vol. 1, no. 1, pp. 1–30, 2005.

[20] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne *et al.*, “Owl-s: Semantic markup for web services,” *W3C member submission*, vol. 22, pp. 2007–04, 2004.

[21] G. Antoniou and F. Van Harmelen, *A semantic web primer*. MIT press, 2004.

[22] M. Forte, W. L. de Souza, and A. F. do Prado, “Using ontologies and web services for content adaptation in ubiquitous computing,” *Journal of Systems and Software*, vol. 81, no. 3, pp. 368–381, 2008.

[23] U. Bellur and H. Vadodaria, “Web service ranking using semantic profile information,” in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 872–879.

[24] P. R. Reddy and A. Damodaram, “Web services discovery based on semantic similarity clustering,” in *2012 CSI 6th International Conference on Software Engineering, CONSEG 2012*, 2012.

[25] P. B. Santos, L. K. Wives, and J. P. M. De Oliveira, “An improved approach for measuring similarity among semantic web services,” in *WEBIST 2012 - Proceedings of the 8th International Conference on Web Information Systems and Technologies*, 2012, pp. 83–88.

[26] OASIS. (2002) Oasis uddi specificaion tc. [Online]. Available: https://www.oasis-open.org/committees/uddi-spec/faq.php

[27] V. Vieira, P. Tedesco, and A. C. Salgado, “Designing context-sensitive systems: An integrated approach,” *Expert Systems with Applications*, vol. 38, no. 2, pp. 1119–1138, 2011.

[28] A. K. Dey, “Understanding and using context,” *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[29] V. Vieira, L. R. Caldas, and A. C. Salgado, “Towards an ubiquitous and context sensitive public transportation system,” in *Proceedings - 4th International Conference on Ubi-Media Computing, U-Media 2011*, 2011, pp. 174–179.

[30] R. Gimson, R. Lewis, and S. Sathish, “Delivery context overview for device independence,” *W3C Working Group Note*, vol. 20, 2006.

[31] R. S. Pressman, *Engenharia de software*, 6th ed. São Paulo: McGraw-Hill, 2006.

[32] T. Tegegne, B. Kanagwa, and T. van der Weide, “ehealth service discovery framework for a low infrastructure context,” in *Computer Technology and Development (ICCTD), 2010 2nd International Conference on*. IEEE, 2010, pp. 606–610.

[33] D. John and M. Rajasree, “A framework for the description, discovery and composition of restful semantic web services,” in *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*. ACM, 2012, pp. 88–93.

[34] O. F. Ferreira Filho and M. A. G. V. Ferreira, “Semantic web services: a restful approach,” in *Proceedings of the IADIS International Conference on WWW/Internet*, 2009, pp. 169–180.

[35] G. A. Miller. (2013) About wordnet - wordnet. [Online]. Available: http://wordnet.princeton.edu