

WS-NEXT, a Web Services Network Extractor Toolkit

Chantal Cherifi *¹, Yvan Rivierre †², Jean François Santucci *³

* Corsica University, France

¹ chantalbonner@gmail.com

³ santucci@univ-corse.fr

† Grenoble University, France

² rivierre@imag.fr

Abstract—In this article, a Web services network extractor toolkit, WS-NEXT (WS Network EXtractor Toolkit), is presented. WS-NEXT allows extraction of interaction and dependency WS networks. Networks can be extracted from syntactic and semantic WS descriptions. Such network structures can be analyzed using complex network tools. We provide examples of networks extracted from a publicly available WS collection. Additionally, we give some networks analysis results.

Keywords—Web Services Networks, Discovery, Composition, Complex Networks

I. INTRODUCTION

A Web service (WS) is a set of related functionalities that can be published and discovered in a WS registry and invoked for remote use. Those modular applications can be programmatically loosely coupled through the Web to form more complex ones. Two of the most popular problems in WS technology addressed by both industry and academia are discovery and composition. Discovery is the process of locating providers advertising WS that can satisfy a service request. Composition arises when several WS are needed to fulfill a request. The way those processes are achieved depends on how WS are described. For the syntactic one, discovery is performed against registries using keywords and compositions are manually defined before any submitted request. Semantic descriptions allow automatic discovery and composition processes. But finding the right WS to fulfill a given request is not an easy task. Indeed, the WS space is extremely volatile. The number of WS is continuously growing, and providers may change, relocate, or even remove them. In this context, we think that the WS space organization is a key for optimizing discovery and composition. The WS space can naturally be represented under the form of networks. Such kind of structures constitutes a convenient way to represent a collection of WS for visualization and analysis purposes. Moreover such precomputed structures can serve as a guide for WS discovery and composition.

Some work has already been done in these directions. Existing research concerns WS network models definition [1], WS network analysis [2, 3] and WS network representation for composition mining purposes [4–7]. In the literature we can observe that there is many ways to represent a WS collection as a network. Different aspects must be considered.

Among them, the first one is the WS description. In [2, 5] authors build their networks from syntactic WS while in [3, 4, 6, 7] they work with semantic WS. The second aspect is the network nodes definition. In [3–6] network nodes are WS. In [7] nodes are parameters. In [1] networks are defined using WS, operations or parameters as nodes. The third one is related to the meaning of the links between nodes. Using parameters as nodes we build dependency networks [7]. In this case the link represents a WS or an operation. When nodes are WS or operations we deal with interaction networks [1, 3–6]. The link represents the information flow between nodes. The fourth aspect is the mode of interaction. To draw links between nodes either all the information is provided or just part of it. We denote those two cases by full and partial invocation mode. In [1] networks are built with full and partial invocation mode. The last aspect is the matching, *i.e.*, how do we compare two nodes in the network. This depends on the WS description. In [2] equal and flexible matching are used for syntactic descriptions. The authors in [3,4] use equivalence and subsumption ontological concepts relationships for semantic descriptions. To summarize five variables can be used to modulate a network extraction: (1) the WS description that can be syntactic or semantic, (2) the granularity which describes the nodes entities, (3) the model representing the nature of the links, (4) the mode related to the available information to link the nodes, (5) the matching which depicts the similarity measure between parameters. We propose WS-NEXT, a WS network extractor toolkit allowing to build different types of WS networks from a collection of WS descriptions. The extracted networks format is compatible with major tools that can be used to analyze networks topological properties (Pajek, igraph, *etc.*). The resulting networks can also be used as pre-computed structures by WS discovery and composition algorithms.

This article is organized as follows. Background key elements are provided in Section II. Variables used to elaborate networks taxonomy are presented in Section III. In Section IV we introduce networks definitions. WS-NEXT architecture and implementation details are presented in Section V. Sample networks produced by WS-NEXT are given and analyzed in Section VI. Finally, conclusions are provided in Section VII.

A. Definition

According to our need, we consider a WS as an interface. A WS interface is defined as a set of operations. An operation represents a specific functionality. It is characterized by one set of input parameters noted I_i , and one set of output parameters noted O_i . I_i is the required information in order to invoke a WS operation i . O_i is the provided information by the WS operation i . At the WS level, the set of input parameters of a WS k is $I_{wk} = \cup I_i$ and the set of output parameters $O_{wk} = \cup O_i$. Figure 1 represents a WS numbered 1 with two operations numbered 1 and 2, so that $I_1 = \{a, b\}$, $O_1 = \{d\}$, $I_2 = \{c\}$, $O_2 = \{e, f\}$, $I_{w1} = \{a, b, c\}$, and $O_{w1} = \{d, e, f\}$.

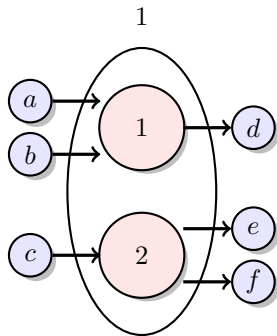


Fig. 1. Schematic representation of a WS

B. Description Languages

Production WS interfaces are mostly expressed with WSDL, a syntactic WS description language [8]. More recently, the research community followed the current semantic Web trend by introducing semantics in WS descriptions, in order to enrich them. Several initiatives for semantic description languages exist among which we can distinguish purely semantic descriptions such as OWL-S [9], from annotated WSDL descriptions such as WSDL-S [10] and SAWSDL [11].

C. Discovery and Composition

When one has a request r with input parameters I_r and desired output parameters O_r , one needs to find a WS k such that $I_r \supseteq I_{wk}$ and $O_r \subseteq O_{wk}$. Finding a WS k that can fulfill r alone is referred to as WS discovery. When it is impossible for a single WS to fully satisfy r , one needs to compose several WS $\{w_1, w_2, \dots, w_n\}$ such that for all $w_l \in \{w_1, w_2, \dots, w_n\}$, I_{w_l} is required at a particular stage in the composition and $(I_r \cup O_{w_1} \cup O_{w_2} \cup \dots \cup O_{w_n}) \supseteq O_r$. This problem is referred as WS composition.

In this section we give an accurate meaning of the previously identified WS network variables.

A. Description

The description variable represents the WS description type. Those two types are syntactic and semantic descriptions. Corresponding variables values are respectively noted *syntactic* and *semantic*. In a syntactic description, each parameter has a name and an XML type. In a semantic description, name and type are also generally specified, and an additional ontological concept is associated to the parameter. Ontological concepts are domain specific and consensual terms. They give parameters a contextual and precise meaning. For syntactic description the name of the parameter has to be considered, while for semantic description the ontological concept must be used.

B. Granularity

The granularity determines the nature of the nodes in a network. From coarser to finer, we consider WS, operations or parameters as node entities. We note the corresponding variables values as *service*, *operation* and *parameter*. In the following paragraphs we will have different definitions for WS and parameters. Unless stated, the definitions given for WS also apply to operations. Indeed an operation can be viewed as a WS with a single operation.

C. Model

The model expresses the type of relationship between nodes. This relationship depends on the granularity. Considering WS as nodes, a relationship between two WS corresponds to the information flow between them. It means that the first one is able to provide the information needed by the second one in order to invoke it. This model is called *interaction*. It is illustrated by Figure 2.

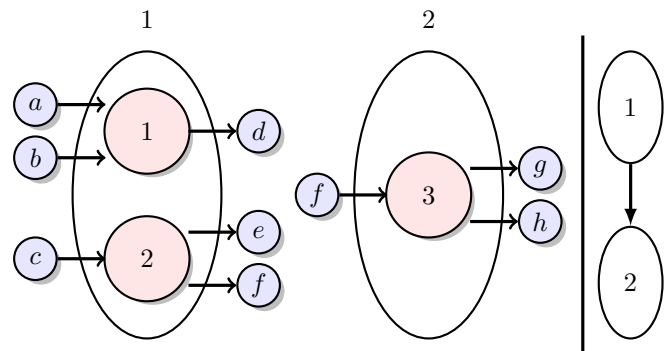


Fig. 2. Example of WS interaction

Considering parameters as nodes, if one is an input parameter of a WS and the other an output parameter of the same WS, there exists a dependency relationship between them. Indeed, the production of the second parameter depends on the provision of the first one through the invocation of the WS. This model is noted *dependency* and is illustrated by Figure 3.

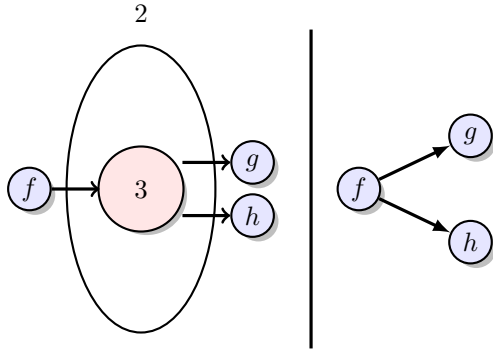


Fig. 3. Example of parameters dependency

Dependency and interaction models are different ways to materialize WS composition.

D. Mode

The mode represents the amount of information used to relate two entities. Two cases must be considered. Either all the information is provided or only part of this information exists. Considering the interaction model, if a WS can provide all the parameters values needed to invoke another one, we will denote this case as *full* interaction mode. Figure 2 is an example of the full interaction mode. If a WS cannot provide all the input parameters required by a second one, this mode is denoted by *partial*. Such a case is illustrated by Figure 4.

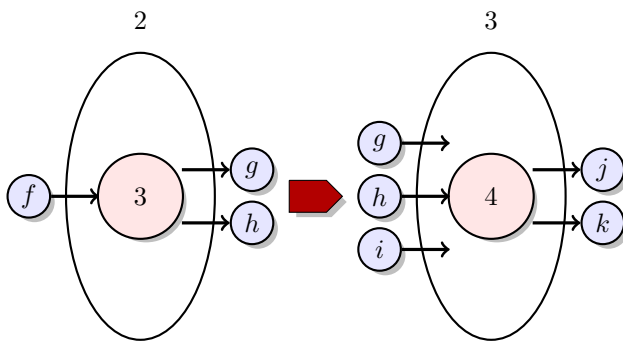


Fig. 4. Example of WS partial interaction

E. Matching

The matching variable describes the similarity between parameters. It is computed differently for syntactic and semantic descriptions. For syntactic descriptions, matching consists of comparing two WS parameters name using similarity functions. We distinguish two cases. The first case considers two parameters as similar if their names are exactly the same string. It is called *equal*. The second case considers two parameters as similar if their name presents a certain level of similarity. It is called *flexible*. The level is computed by string similarity metrics. Different similarity metrics can be used. Classical ones such as Jaro, Levenshtein, Jaro-Winckler and a smoothed metric based on Levenshtein distance between filtered strings have been implemented. These metrics are denoted as *Jaro*, *Winckler*, *Levenshtein* and *Smoothed*. For semantic descriptions, matching consists in comparing ontological concepts associated to parameters. This is done by classical operators (exact, plugin and subsume). Exact corresponds to a perfect matching, *i.e.*, both concepts belong to the same ontology and are exactly identical. Plugin means the concept associated to the first parameter is strictly more specific than the other one. Subsume represents the fact the first concept is strictly more general than the second one. We add a fourth operator called *fitin* which allows drawing a link when there is simultaneously plugin and exact similarities between two nodes. This operator leads to a more flexible semantic interaction representation. The matching variables values are denoted by *exact*, *plugin*, *subsume* and *fitin*.

IV. NETWORK DEFINITIONS

Dependency and Interaction network can be used to represent the WS space. Interaction networks can use either operations or WS as nodes. Dependency networks nodes are parameters. Independently of the granularity two network models can be defined.

A. Dependency Network

We define a dependency network as a directed graph whose nodes correspond to depending parameters and links indicate the head parameter depends on the tail parameter (for example as illustrated by Figure 3, g depends on f) [12]. In the context of dependency networks, each WS k is formally defined as a triplet (I_{wk}, O_{wk}, K_{wk}) , where K_{wk} denotes the set of dependencies defined by k . We consider each output parameter depends on each input parameter. To build such a network, we first create one node for each parameter present in the whole collection. Then, links are created by considering each WS separately: a link is added between each one of its input parameters and each one of its output parameters. Additionally, one parameter may be used by several WS, either as an input or an output. Consequently we have to decide if two parameters are similar. This is the role of the matching functions described in Subsection III-E. In the case of syntactic dependency network, *equal* matching is applied. For a semantic description *exact* matching is applicable.

B. Interaction Network

We define an interaction network as a directed graph whose nodes correspond to interacting WS and links indicate the possibility for the tail WS to act on the head WS [13]. To represent a collection of WS descriptions under the form of an interaction network of WS, we first define a node to represent each WS in the collection. Then, a link is drawn from a WS 1 towards another WS 2 if and only if for each input parameter in I_{w_2} , a similar output parameter exists in O_{w_1} . In other words, the link exists if and only if WS 1 can provide the information requested to invoke WS 2. In the interaction network, a link between two WS therefore represents the possibility to compose them. The matching functions described in Subsection III-E are used to determine the similarity between two parameters.

V. WS-NEXT

Parameters dependency and WS composability can be depicted by networks. Those assumptions were the basis in developing WS-NEXT. Hereafter we present the functionalities and the architecture of WS-NEXT as well as examples of its application.

A. Functionalities

WS-NEXT allows extracting networks from a collection of WS descriptions files, according to the models previously defined. Networks are extracted choosing a *Profile*, also called set of *Traits* (a trait corresponding to a network variable). Figure 5 gives an extract of the WS networks taxonomy. Actual networks are represented by a tree starting from the root, going through each trait and ending by an underlying leaf. A profile example is Full Interaction of Syntactic Operation with Equal Matching. At its present version, WS-NEXT is able to extract eighteen full interaction networks, eighteen partial interaction networks and two dependency networks (called interaction in WS-NEXT) from WSDL or SAWSDL description files.

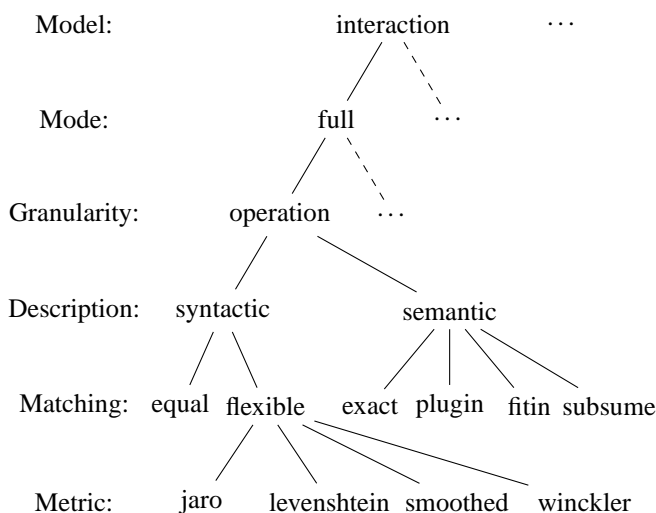


Fig. 5. Extract of WS networks taxonomy

WS-NEXT can also achieve some secondary tasks, such as counting occurrences of parameters names, doing some statistics on flexible matching and serializing collections.

B. Architecture

Architecture of WS-NEXT and networks extraction steps are illustrated by Figure 6. The main parts of this tool are described below.

WS-NEXT Parser processes WS descriptions files one after each other. It detects duplicates and do not parses them twice. Meaningless empty or generically named parameters are discarded. For semantic descriptions, corresponding ontologies must be available while parsing files. The parsing step results in internal objects which represent WS of the collection.

WS-NEXT Network Extractor takes as input objects (that internally represent the collection), and the profiles selected by the user, to extract appropriate networks.

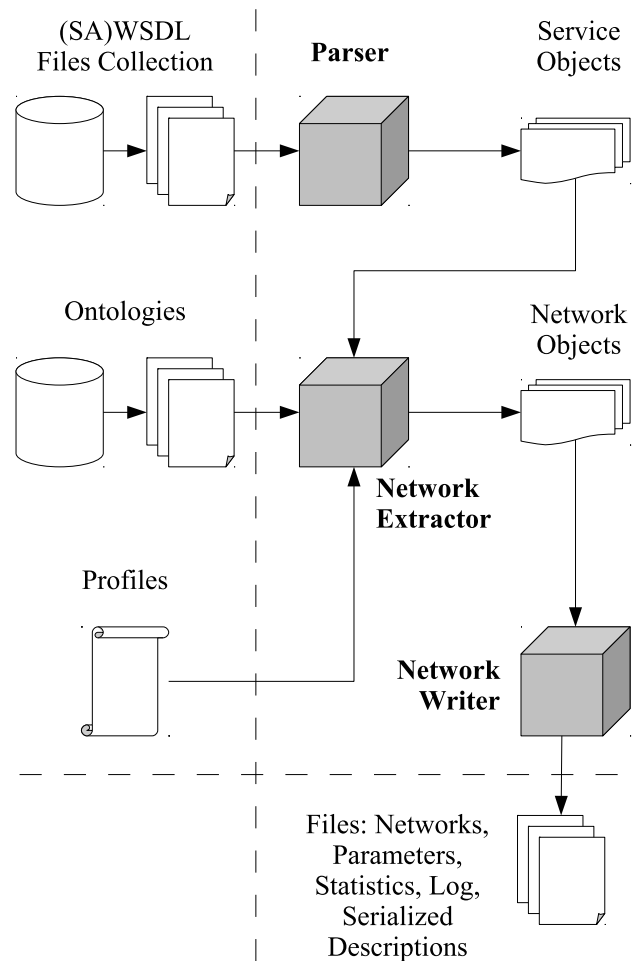


Fig. 6. WS-NEXT architecture

WS-NEXT Network Writer write networks and the internal represented collection on files, in a specified output directory, along with log and some statistics files. Networks files contain metadata about the source collection and extraction profile. Log files contain eventual errors and warnings pointed out at any step of the whole process.

C. Implementation

As shown in Figure 7, WS-NEXT provides a graphical user interface. It allows specifying the collection to parse, the target directory for extracted networks, information (name and description language) about the collection and the network extraction profiles to be used. WS-NEXT can also run without GUI by adding arguments on the command-line, in order to be called by scripts.

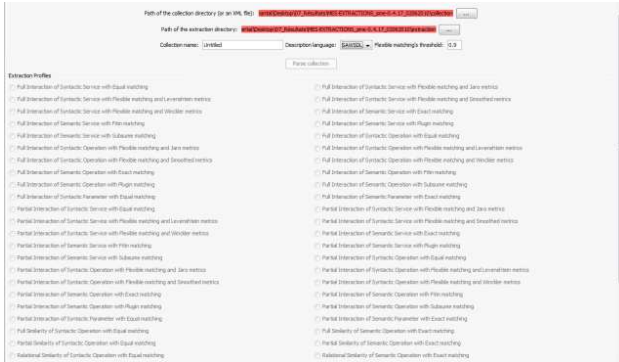


Fig. 7. WS-NEXT graphical user interface

WS-NEXT is coded in Java. It is cross-platform as it only relies on Java Runtime Environment 6. Current implementation of WS-NEXT only supports WSDL and SAWSDL as input files format and Pajek.net as output. It is easily extendable to any format since Java interfaces are available for both input/output layers.

VI. USE CASE OF WS-NEXT

In this section, results of network extraction of dependency and interaction networks are presented. A succinct analysis is provided. Both networks have been extracted from the SAWSDL-TC1 [14] WS descriptions collection. This collection provides 894 semantic WS descriptions written in SAWSDL, containing one interface, one operation per interface and 2136 parameters instances. They are distributed over 7 thematic domains (education, medical care, food, travel, communication, economy and weapon). It originates in the OWLS-TC2.2 collection, which contains a part of real-world WS descriptions retrieved from public IBM UDDI registries, and semi-automatically transformed from WSDL to OWL-S.

A. Network extraction

Figure 8 represents an extracted network. Isolated nodes have been eliminated.

Its profile is Full Interaction of Semantic Operation with Exact Matching. This network has a giant component. This globally reflects a large number of potential compositions.

A second example of a network extracted by WS-NEXT is presented in Figure 9. It is a dependency network where isolated nodes are not represented. Its profile is Partial Interaction of Semantic Parameter with Exact Matching. This network has the same structure than the previous one. This reflects that both models retain the same information.

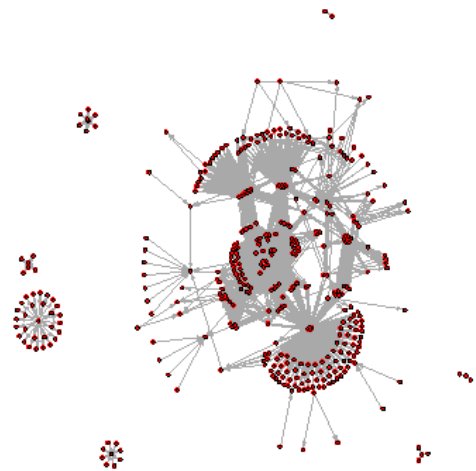


Fig. 8. A trimmed interaction network

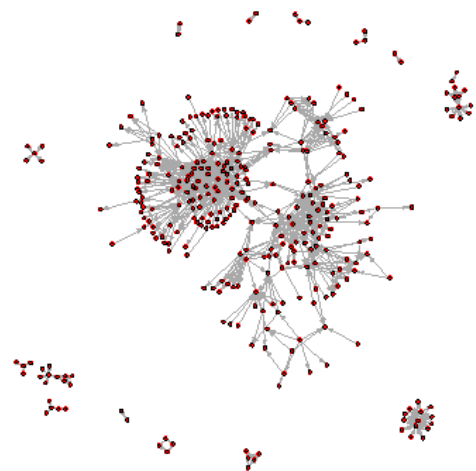


Fig. 9. A trimmed dependency network

B. Network Analysis

In this section, we present a use case to demonstrate the utility of the extracted networks through a network analysis. Our aim is to get an idea of the WS composability. From the SAWSDL-TC1 collection of publicly available WS descriptions, we extract 4 different WS interaction networks (1 syntactic and 3 semantic). We then take advantage of tools from the complex networks field to analyze them and determine their properties. The results show that all WS interaction networks exhibit some of the typical characteristics observed in real-world networks, such as short average distance between nodes, presence of a giant component and community structure. These properties illustrate a highly connected space.

We additionally perform a comparative analysis of the syntactic and semantic approaches used to describe WS. The results show that using semantic WS descriptions should improve the composition process. Indeed, we observe the syntactic giant component was slightly larger, which might be due to the presence of false positives, *i.e.*, operations irrelevantly connected. Although semantic giant components contain less links, their interconnection structure is more efficient, leading to a smaller average distance between operations (in terms of composition) and a smaller diameter (maximal composition size). We can conclude that the introduction of semantics in WS description allows a more accurate representation of their potential interactions, and should consequently result in a more efficient search for composition processes, at least for the considered collection. This observation is of uppermost importance as we are witnessing a rapid development in semantic-related web technologies allowing the development of semantic WS.

When comparing the three semantic networks, a clear distinction appears between the loose matching functions, plugin and subsume, and the exact one. Loose matching functions lead to networks with even smaller diameters and average distances, corresponding to a larger proportion of links between the domains, which in turns result in a weaker community structure. This highlights the importance of the selected matching function. More detailed results concerning this study can be found in [13]. Another use case concerning the analysis of dependency networks can be found in [12].

VII. CONCLUSION

A WS network extractor is presented with one use case for the extracted networks in the network analysis application domain. The WS-NEXT development is inspired by the fact WS composition can naturally be represented on the form of networks. Additionally powerful tools are available for complex networks analysis. Extracted networks can be used to give a snapshot of the underlying WS space. Another application domain of uppermost importance is to use those networks as pre-computed structures during the discovery, and composition processes. Nevertheless, further research is needed to enrich WS-NEXT with the possibility of extracting a global semantic network including several semantic matching measures with weighting links for interaction degrees. A more advanced work would be to add a Web digger along with the existing collection digger.

REFERENCES

- [1] S.-C. Oh, "Effective web service composition in diverse and large-scale services networks," Ph.D. dissertation, Pennsylvania State University, USA, 2006.
- [2] H. Kil, S. chan Oh, and D. Lee, "On the topological landscape of semantic web services matchmaking," in *SMR*, vol. 178, 2006.
- [3] J. Gekas and M. Fasli, *Employing Graph Network Analysis for Web Service Composition*. IGI Global, 2008.
- [4] H. N. Talantikite, D. Aïssani, and N. Boudjlida, "Semantic annotations for web services discovery and composition," *Computer Standards & Interfaces*, vol. 31, no. 6, pp. 1108–1117, 2009.
- [5] J. Liu and L. Chao, "Web services as a graph and its application for service discovery," in *GCC*. IEEE Computer Society, 2006, pp. 293–300.
- [6] M. M. Shiao, J. O. Fladmark, and B. Thiell, "An incremental graph-based approach to automatic service composition," in *IEEE SCC*, vol. 1. IEEE Computer Society, 2008, pp. 397–404.
- [7] S. V. Hashemian and F. Mavaddat, "A graph-based approach to web services composition," in *SAINT*. IEEE Computer Society, 2005, pp. 183–189.
- [8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web services description language (wsdl) 1.1," W3C, Tech. Rep., 2001. [Online]. Available: <http://www.w3.org/TR/wsdl>
- [9] M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "Owl-s: Semantic markup for web services," W3C, Tech. Rep., 2004. [Online]. Available: <http://www.w3.org/Submission/OWL-S/>
- [10] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. T. Schmidt, A. Sheth, and K. Verma, "Web service semantics - wsdl-s," W3C, Tech. Rep., 2005. [Online]. Available: <http://www.w3.org/Submission/WSDL-S/>
- [11] J. Farrell and H. Lausen, "SawSDL: Semantic annotations for wsdl and xml schema," W3C, Tech. Rep., 2007. [Online]. Available: <http://www.w3.org/TR/sawSDL/>
- [12] C. Cherifi, V. Labatut, and J. F. Santucci, "Web services dependency networks analysis," in *ICNMI*, 2010, pp. 115–120.
- [13] C. Cherifi, V. Labatut, and J. F. Santucci, "Benefits of semantics on web service composition from a complex network perspective," in *NDT (2)*, vol. 88. Springer, 2010, pp. 80–90.
- [14] M. Klusch and P. Kapahnke, "SawSDL-tc1 - sawSDL service retrieval test collection," 2008. [Online]. Available: <http://projects.semwebcentral.org/projects/sawSDL-tc/>