Computer Science and Information Systems 13(3):779–807 DOI: 10.2298/CSIS160124026B

Measuring the understandability of WSDL specifications, Web Service Understanding Degree Approach and System

Mario Marcelo Berón[1], Hernán Bernardis[1], Enrique Alfredo Miranda[1], Daniel Edgardo Riesco[1], Maria João Varanda Pereira[2], and Pedro Rangel Henriques[3]

[1] Universidad Nacional de San Luis, Departamento de Informática, San Luis, Argentina
{mberon,hbernardis,eamiranda,driesco}@unsl.edu.ar
[2] Instituto Politécnico de Bragança, Dep. de Informática e Comunicações/Centro Algoritmi Bragança, Portugal
mjoao@ipb.pt
[3] Universidade do Minho, Dep. de Informática/Centro Algoritmi Campus de Gualtar, Braga, Portugal
prh@di.uminho.pt

Abstract. Web Services (WS) are fundamental software artifacts for building service oriented applications and they are usually reused by others. Therefore they must be analyzed and comprehended for maintenance tasks: identification of critical parts, bug fixing, adaptation and improvement. In this article, WSDLUD a method aimed at measuring a priori the understanding degree (UD) of WSDL (Web Service Description Language) descriptions is presented. In order to compute UD several criteria useful to measure the understanding's complexity of WSDL descriptions must be defined. These criteria are used by LSP (Logic Scoring of Preference), a multicriteria evaluation method, for producing a Global Preference value that indicates the satisfaction level of the WSDL description regarding the evaluation focus, in this case, the understanding degree. All the criteria information required by LSP is extracted from WSDL descriptions by using static analysis techniques and processed by specific algorithms which allow gathering semantic information. This process allows to obtain a priori information about the comprehension difficulty which proves our research hypotheses that states that it is possible to compute the understanding degree of a WSDL description.

Keywords: WSDL, Web Services Comprehension, LSP.

1. Introduction

Nowadays the Web Services (WS) are fundamental software artifacts for building service oriented applications. According to World Wide Web Consortium (W3C, for details see `http://www.w3.org/`), a WS is: *a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A WS supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols*. The organizations, increasingly, produce web services which are used by other organizations to produce new software systems aimed at solving business demands. Web services have associated a description which specifies the data types used, the operations provided, inputs and output, the technology

used to accomplish the communications between other high level and low level of software elements. These descriptions are published in the internet and the organizations can retrieve them and decide if some of those services are useful for building the software they need [43]. Web Services are software packages and therefore they must be comprehend for maintenance tasks (bug fixing, adaptation, evolution, etc.). The primary information source to accomplish this task is the respective WSDL (Web Service Description Language, `http://www.w3.org/TR/wsdl20/`) description. Although, there are several resources from which it is possible to collect information about the Web Service, the WSDL description is the first that the user employs for analysing its usefulness for his purposes. Furthermore, the web service descriptions are interesting because they provide a high level abstraction data which can be very useful to simplify the understanding of the web services. As said above, a standard language used to write web service descriptions is WSDL. This language is a dialect of XML with well defined rules to specify each component. Being a XML based language it is fastidious to read such a description, and therefore a tool is needed to assist the software engineer in this task. In this context, many tools can be found that are oriented to facilitate the inspection of WSDL descriptions, transform to a different WSDL version, compute several metrics, produce user-friendly visualizations, etc [25],[28],[35]. However, at the best of our knowledge, only a few are oriented to help their understanding. Taking this into consideration, in this article WSDLUD (Web Service Understanding Degree) is presented. WSDLUD is a metric aimed at providing, a priori, a measurement about the WSDL description understanding complexity. For calculating WSDLUD, Logic Scoring of Preference Method (LSP) [19],[56] is used. LSP is a multi-criteria evaluation method; it requires a Criteria Tree, an Aggregation Structure and a set of Elementary Criteria Functions to be defined [45]. Combining systematically such elements, this method produces a satisfaction level that indicates, in this case, the understanding degree of a WSDL description. In order to apply LSP and compute WSDLUD, the WSDL description must be statically analysed and all the information available must be retrieved. This information is submitted to different evaluation procedures in order to obtain satisfaction values (values in [0,1] or [0,100]). To perform these processes, the use of both compilation and natural language processing techniques are required. The first is used to retrieve formal elements from WSDL source code. The second is employed to gather semantic information from unstructured information sources.

A similar approach can be used to measure the understanding degree of the services published with Representational State Transfer (REST) specifications. Since these specifications are written in a XML based DSL, the same approach can be used to extract the information needed to compute the intermediate values (criteria) required by the multicriteria evaluation method. In this case, the criteria need to be defined because they must be oriented to measure the complexity of each REST specification part. Even considering the messages format, currently they are specified using JSON (Java Script Object Notation) or JSONP (Java Script Object Notation with Padding), so once again the same methods can be used to compute the understanding degree.

The article is organized as follow. Section 2 gives a justification about the reasons why WSDL is taken as research topic instead other technology. Section 3 describes the work tightly related with the research topics here presented. Section 4 proportionates an explanantion of the LSP method, which will be used for calculating WSDLUD. Section 5 presents the software framework used to implement WSDLUD and it describes the exe-

cution flow to symplify the understanding and to make the article self-contained. Furthermore, all the structures required to accomplished the evaluation are defined in sections 6 to 8. Section 9 presents the case studies where it is possible to observe the results obtained through the application of WSDLUD to some test cases available in several web sites[4]. Section 10 closes the paper with some conclusions and future work.

2. Why Web Service Description Language?

In the last years the REST (Representational State Transfer) is being often used for building WS- REST instead of WSDL [24,23,3]. Its popularity is due to: a) the methodology used to build WS is not so technical and formal like WSDL-SOAP; b) it has good performance in the net; c) it uses just only the HTTP protocol and a reduced set of actions based on CRUD (Create, Read, Update, Delete) operations of data bases. These characteristics turns REST an intuitive methodology, easy to use and learn by the software developers. Google, Facebook and Twitter have contributed to REST popularity by implementing their public APIs with this technology. This characteristic has generated discussions concerned with if WS-SOAP is still a good choice or if it must be left of using [46,27]. The discussion previously mentioned still doesn't have a clear answer. Generally, the choice depends on the WS context and the who will use the WS. On the one hand, REST is oriented to be used by the developers (a clear example are the API's of Google, Facebook, Twitter, etc), clearly they are favored by their less formal and intuitive characteristics. On the other hand, WS-SOAP is aimed at focusing to the business world and to be used by software agents which need a technical and formal focus. However, some enterprise like Yahoo and Flickr do not adopt a particular architecture and they offer their API's with both options. In this context, it is the developer who takes the last decision. From a more technical point of view, REST has a slight advantage of performance, nevertheless it is not enought to discard WS-SOAP [33,49]. Moreover, there are techniques for improving the SOAP performance which allow to reduce the gap with REST [57,33,49]. Furthermore, if security features is incorporated in REST specifications, by adding SSL certificates, the performance is notably degraded; this is not the case of SOAP. Actually, both architectures are often used. In this work WSDL-SOAP is taken as study focus for several reasons. In first place, it is an older architecture and there are many legacy systems which, depending on the context, need to be comprehended for migrating them to REST. Second, WS-SOAP allows to use different protocols, this peculiarity makes it a more complete option and therefore more complex to understand. Third, WSDL can be used for sending synchronous and asynchronous messages because it is possible to specify the transport protocol for each particular service. This kind of flexibility is fundamental when gateways are built for legacy systems with web-friendly protocols; REST does not meet these characteristics and it only can be used for sending synchronous messages by

[4] `https://code.google.com/p/dic/downloads/detail?name=GoogleSearch.wsdl` `http://queue.amazonaws.com/doc/2009-02-01/QueueService.wsdl` `http://www.webservicex.com/airport.asmx?wsdl` `http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL` and `www.webservicex.net/OFACSDN.asmx?WSDL`.

using http protocol. Fourth, WSDL-SOA was elaborated to make easier the machine-to-machine interoperability and not to facilitate the communication with the developers, for this reason to comprehend WSDL-SOA is important for purpose of maintenance, migration and evolution tasks. Anyway, REST must be considered as other research line, and can be taken as study focus in future works.

3. Related Work

The WSDL description analysis is based on static or behavioral information or a mix of both.

Considering the static information several interesting works can be found. The traditional approaches are oriented to compute metrics like numbers of ports, number of services and also provide visualizations easy to understand to compare and evaluate a set of program parameters [6], [55],[58] [59].

Innovative works can be found in the security context. Concerning metrics for organization's security they state that the easier to understand a WSDL description the easier will be to carry out fraudulent actions against the organization. On account of that, the authors compute the understanding level of WSDL description and if it is high they define approaches to reduce its readability [55] [39],[52],[1].

Simon and Rischbeck in [53] propose to annotate the WSDL descriptions in such way that the stakeholders are able to understand the business aspects. They hold that the WSDL descriptions include business aspects specified in some XML dialect which is not easy to understand by the stakeholders. For this reason the authors consider that inserting annotations user understandable will simplify the WSDL understanding. These annotations contain information about the business aspect for the user and technical for housekeeping operations.

Nandigam, Gudivada and El-Said explain WSExplorer [42], it is a tool for facilitating the understanding of WSDL descriptions allowing the access to all WSDL elements. This task is accomplished using traditional inspection techniques and user-friendly graphical interfaces.

Jiang and his research group in [30] state that the WS comprehension is a difficult task. They mention that this peculiarity can be clearly observed in the WSDL automatically generated where the engineer has not the control about how the WSDL description was created. This characteristic difficults the comprehension because the generators have not taken into account, properties like: comprehensibility, semantic degree, reusability, etc. Therefore, they proposed to use model based on UML to comprehend and to compare WSDL descriptions. Taken as base these models it is possible to obtain information about of WS associated and the differences and similarities found between various WSDL descriptions of the same WS.

Others contributions are based on behavioral information. In this context several works concerned with measuring the WSDL description considering the complexity of the operations and messages involved can be found. The result obtained by the authors state that: the more complex the operations and messages are, the more complex will be to understand the WSDL description [16],[32],[37].

In the work accomplished by Ni and Fan [44], the ontologies are used to make possible the interoperability between organizations. In order to reach this goal the domain

specific ontologies are defined which specify a common vocabulary for the stakeholders who need to share information. These ontologies before mentioned are built from the WSDL description and business' process implemented with BPEL. The new aspect in this work is the using of ontologies to specify concepts used in the WSDL descriptions associated with the BPEL implementations, clearly this approach helps to understand the WSDL specficiation.

Segev and Toch in [51] present an approach that uses both static and behavioral information. In this work they extract some knowledge from WSDL descriptions for trying to understand them with the goal of defining a ranking to compose web services. The approach to extract information consists of three steps. The first one builds a basic vocabulary (Base Line) from WSDL descrition and some documentation associated by using inspection strategies. The second one uses TF/IDF a common strategy of information retrieval to extract keyword from a document corpus. Finally, the approach uses the web as knowledge source for contextualizing the information gathered.

In the work of Wang [60] a method to make easier the understanding of WSDL documents is proposed. The method extracts information from WSDL descriptions and this information is used to collect related web pages. This information is submitted to information retrieval techniques with the goal of enriching the WSDL description.

Fangfang Liu et. al. in [35] hold that, due to the quantity of Web Services available in the web, it is a problem to find a adequate Web Service for accomplishing the wished task. For this reason, the authors elaborate strategies to compare the similarity between WS from the semantic terms extracted of WSDL descriptions. This comparison is carried out by extracting the terms using identifier analysis techniques and, after that, a graph is built for each WS analyzed. These graphs are joined using the similar terms, the result of this operation is a bipartite graph. This graph is employed to compute the WS similarity through the metric based on distance between terms. The last task is achieved employing Web Search for detecting the similar terms when the terms are not found in a traditional dictionary.

Other kind of works found in the literature are related with experimental studies regarding the understandability of WSDL descriptions. The next paragraphs describes some of them.

In the work of Crasso et. al. [15] the authors describes the bad practices found in the WSDL descriptions which make difficult of understanding. The experimental studies accomplished demonstrated that the most common bad practices are: ambiguous names, lack of comments, inappropriate comments, port typed repeated, between others. For each problem, they describe possible solutions: separating the schema from the definition of the offered operations; removing repeated WSDL and XSD code; putting error information within fault messages and only conveying operation results within output ones; replacing WSDL element names with explanatory names if original names are cryptic; moving noncohesive operations from their original port type to separate port types; documenting the operations; etc.

Hu et. al in [29] present an experimental studies which demonstrate that the WSDL description containts implicit semantic information. These studies confirm that the 97% of this kind of operation can be found in the inputs and outputs of web services.

It is also possible to find works that use ad-hoc approaches. They are based on traditional object oriented metrics to measure quality attributes of WSDL descriptions [13],[14], [54].

WSDLUD method, defined in this article, is different from those found in the literature in several aspects. First, all formal elements of the WSDL description (types, messages, port types, bindings, services) are considered and for each one of them the understanding degree is measured.

Second, the WSDL description's understandability can be simplified if the information (provided by the identifiers and documentation) gives useful semantic information about the description's domain. For this reason, several metrics to measure the quality of the identifiers and documentation of the description, are defined and calculated.

Third, the value produced by WSDLUD is the combination of metrics (those mentioned before) which consider both quantitative and qualitative information. We used these metrics to measure WSDL descriptions and obtain a final value for each of them. This final value is computed by using a multi criteria method. This method is parameterizable allowing to reflect the engineer experience in the evaluation mechanism. Finally, as a side effect, the process used to compute WSDLUD can also be used for: i) To provide a ranking of WSDL descriptions understandability, ii) To build visualizations based in charts, and allow to analyse the results and to discover the possibilities to improve the WSDL description understanding.

To finish this section, it it important to notice that, at the best of our knowledge, a metric with the characteristics mentioned above was not yet described in the literature. So, we believe that the work here reported is a valid contribution for the area of Web Services' formal descriptions, in particular using WSDL notation, aiding on their improvement.

4. Logic Scoring Preference

Logic Scoring of Preference (LSP) is a multicriteria evaluation method based in the definition of: a criteria tree, elementary criteria functions and an aggregation structure.
LSP is useful to analyze, compare and select and the best alternative from a set of objects being graded and ranked. Multiple Criteria Decision Methods (MCDMs) are used to evaluate and make decisions regarding some problems that admit a finite number of solutions [47]. Nowadays there are a considerable number of MCDMs that are used in decision making in various topics. However, it was difficult to find, in the literature, systems that implement this kind of methods. The MCDMs most recently used and implemented are ELECTRE (ELimination Et Choix Traduisant la REalité) and PROMETHEE (Preference Ranking Organization METHod for Enrichment Evaluations). Both methods use a similar approach than LSP. ELECTRE was proposed by Bernard Roy in 1971 [48]. The tools that implement different versions of ELECTRE [40,2,41], generally have some drawbacks, for example: they employ traditional interaction strategies, they do not define a Domain Specific Language (DSL) to be used during the evaluation process (even when it would be very functional), they use complex fuzzy logic that user must deal with, etc. PROMETHEE was developed by Brans and further extended by Vincke and Brans [10]. PROMETHEE is quite simple in conception and application compared with the other MCDMs. Therefore, it is widely used in research and practical contexts. Two of the most used implementations of PROMETHEE are Decision LAB and PROCALC [9]. Never-

theless, both have similar drawbacks comparing to ELECTRE implementations. Other MCDM implementations such as AHP [50], MAUT [31], etc., were studied. However we could not find those implementations available for a deeper comparative analysis. In the case of LSP (Logic Scoring of Preference), there are tools based on this method which can be used to compute the the WSDL description understanding degree, some examples are: LSPmed [21], webQEM [45], ISEE [17], NESSy [38]. In the following subsections, all the LSP components will be explained.

4.1. Criteria Tree

The criteria tree has the characteristics that the objects under evaluation must have. With the goal of developing a complete criteria list, a hierarchical decomposition process is applied. At the end of this process a list of measurable attributes is obtained. In the first instance, the high level characteristics are defined. Then, they are decomposed in sub-characteristics and so on. This process is repeated until obtain the atomic attributes. The result of this task is a tree that describes the main characteristics that the objects under evaluation must meet.

4.2. Elementary Criteria

LSP requires the normalization of the measurable attributes. This normalization is necessary because: i) In several decision contexts the measurement units are different; ii) The values of different attributes may be incomparable.

The LSP attribute normalization is accomplished through the definition of Elementary Criterion Functions. An elementary criterion function maps a value taken by the performance variable in other contained in the interval [0,1] or [0,100]. This value represents the satisfaction level of the performance variable under observation. So, 0 represents a situation where the performance variable does not satisfy the requirements at all, and 1 (or 100) means that the requirement is totally satisfied. The elementary criteria can be classified as: *Absolute* or *Relative*.

An Absolute elementary criterion is used to determine the absolute preference of some attribute. A Relative elementary criterion is employed to establish the relative indicators of the tools under comparison.

Absolute elementary criteria can belong to different types, as defined below.

- Continuous Variable
  - **Multivariable:** The performance variable is computed by a function. This function receives parameters as its input and returns the value corresponding to the attribute under evaluation.
  - **Direct:** The performance variable has a value that is directly inserted by the evaluator.
- Discrete Variable
  - **Multilevel:** The performance variable can take one value from a set of discrete values. These values are established by the evaluator in the stage of *elementary preference definition*; they correspond to different preference levels. The engineer in the *evaluation* stage must choose a value from that set.

4.3. Aggregation Structure

The elemental preferences, that result from the application of the elementary criteria to the measurable attributes, must be aggregated in order to obtain the global preference. This global preference represents the satisfaction of all the requirements, by the object under evaluation.

In order to reach the global preference, some aggregation preference functions are used. These functions receive a set of elementary preferences and their corresponding weights as input. The weights represents the relative importance for each preference. The functions return aggregated preferences as their output. All the outputs are aggregated in the next level of the structure. This process is repeated until the global preference is reached. The aggregation function proposed by LSP is:

$$E = (w_1 e_1^r + w_2 e_2^r + ... + w_k e_k^r)^{\frac{1}{r}} \quad (1)$$

where:

$$-\infty \leq r \leq +\infty$$
$$0 \leq w_i \leq 1 \text{ and } i = 1..k$$
$$w_1 + ... + w_k = 1$$

E is a general instantiation scheme which produces a continuous spectrum of aggregation functions, depending on the value of *r*. Table 1 shows the most relevant values for *r*, taking into account the number, *n*, of function input values. For example, if the operator under consideration is D- and it receives three input values, then the value of *r* in the precedent formula is 2.19. To be clearer, *r* represents the conjunction-disjunction degree of each operator. We say that *r* generates several functions known as *Conjunctive Disjunctive Generalized functions* (CDG). These functions are the operators used to aggregate the elementary preferences. The formula employed to compute the values in table 1 is explained in [18].

4.4. The evaluation process

The evaluation process is carried out defining the values of all performance variables for each tool under evaluation. In this way, for each system, a global preference will be computed and this value is used to elaborate the ranking. Figure 1 shows a representation of the LSP Evaluation Method.

The global preference is obtained from the computation (represented in figure 1 by L($E_1..En$)) of all the elementary preferences.

And these elementary preferences are the result of applying the elementary criteria to the performance variables. Finally, the elementary criteria can be computed because the engineer provides the required values.

5. WSDLUD

In order to compute WSDLUD several tasks must be done. The central one, crucial to obtain the final degree, is the LSP evaluation process. To accomplish that task, three steps

Operation Name	Symbol	r			
		n=2	n=3	n=4	n=5
Disjunction	D	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Strong Cuasi Disjunction	D+	9.52	11.09	12.28	13.16
Cuasi Disjunction	DA	3.83	4.45	4.82	5.09
Weak Cuasi Disjunction	D-	2.02	2.19	2.30	2.38
Arithmetic Media	A	1.00	1.00	1.00	1.00
Weak Cuasi Conjunction	C-	0.26	0.20	0.17	0.16
Cuasi Conjunction	CA	-0.72	-0.73	-0.71	-0.67
Strong Cuasi Conjunction	C+	-3.51	-3.51	-2.18	-2.61
Conjuction	C	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Table 1. Values of r corresponding to each CDG

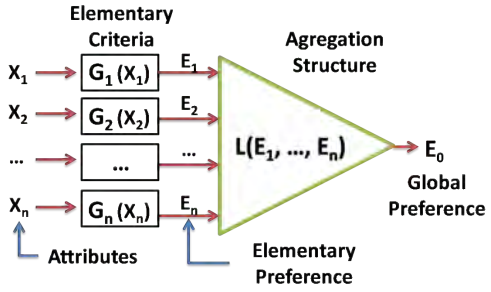

Fig. 1. LSP Method Representation

must be done: building the criteria tree, performing the aggregation, and running the algorithms that compute the elementary preferences (EP) and the one that constructs the ranking. However, to be possible to execute the LSP Evaluation, some information must be retrieved from the WSDL descriptions source code. The data so far extracted is then processed by specific algorithms to obtain quantitative and qualitative metrics. The first group is concerned with the evaluation of size metrics (with precise or rigorous definitions) such as *Ports Number*, *Services Number*, *Binding Number*, etc. The second group deals with more subjective definitions concerned with quality measurements that depend on the problem domain; the identifiers and documentation gathered are processed resorting to domain specific dictionaries which allow to assign a mean to each information part. Fig. 2 shows the architecture of a software framework developed to compute WSDLUD according to the approach here proposed. As can be seen in the sketch, the proposed tool has three modules, or architectural blocks: *WSDL Processor*, *Information Extractor*, and *LSP Evaluator*.

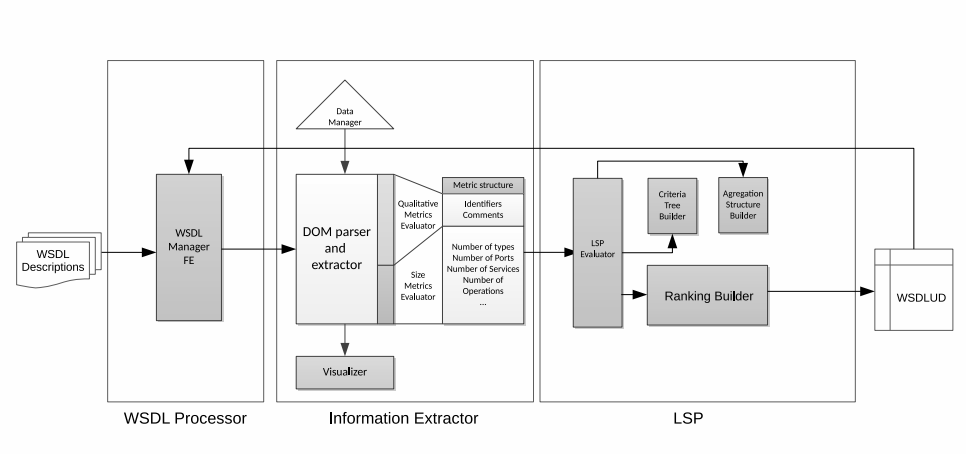

Fig. 2. Logical View of WSDL Tool.

The module *WSDL Processor* is composed of:

WSDL Manager FE: this component implements the front-end needed to analyze WSDL descriptions. It allows to select one or more WSDL description/s and submitted them to the analysis process. Futhermore, it is possible to add, delete, inspect WSDL descriptions, etc.

The module *Information Extractor* has the following components:

Data Manager: this component allows to use efficiently domain specific dictionaries and other knowledge bases that are necessary to compute some of the qualitative metrics.

DOM Parser and Extractor: this component uses a DOM parser to analyze the input (the WS specifications given are written in a specific XML dialect), and it builds internal data structures to store the extracted data in order to enable the metrics evaluation.

Qualitative Metrics Evaluator: this component implements traversals to the internal structure where comments and identifiers are stored and processes them in order to extract meaning and evaluate numeric information about their quality.

Size Metrics Evaluator: this component implements traversals to the internal structures in order to compute the value of size metrics, evaluating the *operation number*, *parameter number*, *service number*, etc.

Visualizer: this component is used to produce the visualizations relevant for making easier the intermediate or final information understanding. At present this module builds visualizations based on charts to depict size metrics like: *Types Number*, *Ports Number*, *Services Number*, etc [7].

The module *LSP* implements the LSP operation flow. Futhermore, it allows to retrive the evaluation state, save results, among other administrative operations. It has three components:

LSP Evaluator: it is the evaluation coordinator, this component is responsible to access the internal data structures, produced by the previous module, in order to get the concrete criteria values necessary to rank the quality of the input description. Complementary, this component instantiates the aggregation structure defined with the concrete criteria parameters obtained in the previous step.

Criteria Tree Builder: on one hand, it provides the facilities to visually build a criteria tree. The engineer can insert, delete and modify nodes which represent characteristics to consider in the evaluation. Furthermore, it is possible to define hierarchical relations between the characteristics and subcharacterstics until obtaing the preferences be measurable.

Aggregation Structure Builder: like *Criteria Tree*, the Aggregation Structure must be built. In order to simplify this task, this component implements methods aimed at insert nodes i.e. logical operators and conect them through arcs which indicates how the evaluation process must be accomplished.

Ranking Builder: this component carries out the evaluation process. For this task, the component provides methods to validate the data inserted in the aggregation structure and traverse it to produce a value which indicates the satisfaction level that in this project is the WSDL description understandability degree.

The logical view defined has been described it shows the static aspects of the software used to implement WSDLUD. However, to get a complete comprehension about how WSDLUD is calculated, a dynamic view is presented in Fig. 3. The Sequence Diagram in Fig. 3 allows to observe the operation flow between the components described above to compute a metric related to the identifiers (in this case a elementary preference).

First, the identifier is retrieved by using the method *retrieveIdentifier()*. Once the identifier is available, the split algorithm is applied; in this case *samurai()* splitter is the used algorithm [11],[22], [12]. Samurai returns a set of words (that compound the identifier under analysis) which are searched in the domain specific dictionary (*findInDictionary()*) with the goal to verify if it is a valid word or not. As next step, the words are expanded using the *Classic Expansion Algorithm* (*classicExpansionAlgorithm()*) [5],[4], [26],[34]. This algorithm needs to access the dictionaries to verify if the expanded words obtained have a mean (an word is considered to be meaningful if it exists in the dictionary). As a

final step, the criterion concerned with the current evaluation is gathered (*retrieveCriterion()*) and the elementary preference is computed (*computePreference()*). This value is saved in the aggregation structure for proceeding to the final evaluation. This last task is carried out when all the elementary preferences for all the criteria are computed. In this case, the *LSP Evaluator* receive the message *evaluate()* from *WSDLManager*, and it in turn sends the message *computeRanking()* to the *Ranking Builder*. As next step, the *Ranking Builder* retrieves the aggregation structure with the elementary preferences using the method *aggregationStructure()* and return as result the global preference (see Fig. 4).

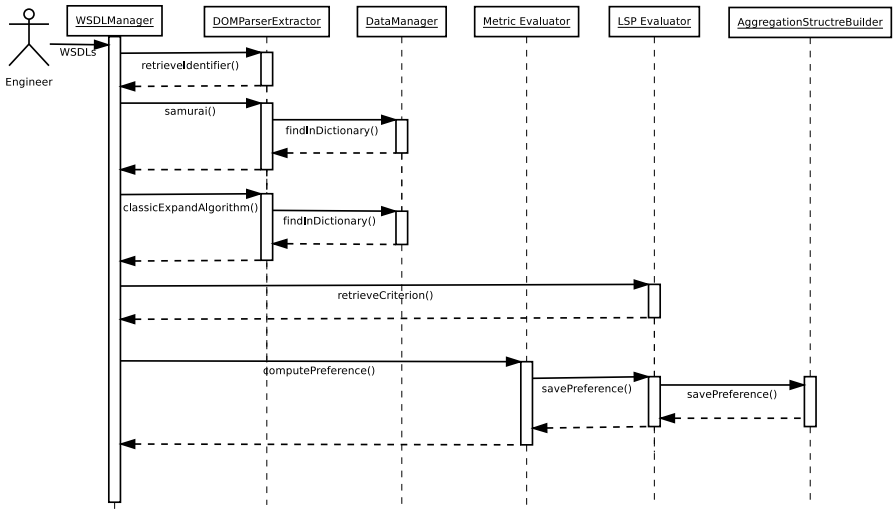

Fig. 3. Dynamic Logical View of WSDL Tool - Identifier metrics.

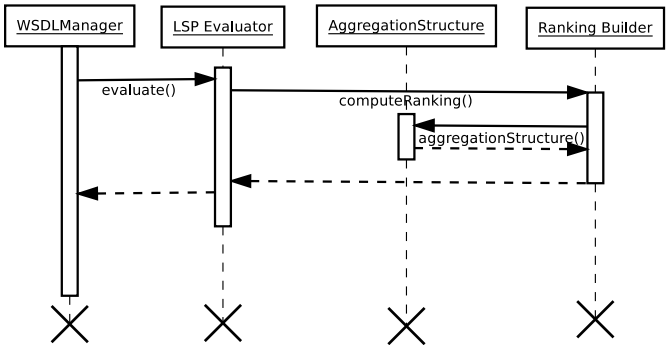

Fig. 4. Dynamic Logical View of WSDL Tool - Global Preference.

6. WSDL Description Criteria Tree

The criteria tree of a WSDL description is composed by the following characteristics[4]: i) Type Understanding Degree, ii) Message Understanding Degree, iii) Port Type Understanding Degree, iv) Binding Understanding Degree and v) Service Understanding Degree. Each characteristic has associated a sub criteria tree which takes into consideration the proper properties of the evaluated element. In the next paragraphs, the Criteria Tree for these characteristics will be explained.

Type Understanding Degree. This characteristic is composed by the following attributes: *Number of Primitive Types*, *Number of Complex Types*, *Documentation Quality*, *Type Name Quality* and *Number of Fields*. Clearly, a primitive type (a primitive type is a type provided by the language), for example: text, integer, real, boolean, etc. will be easier to understand than a complex type (a complex type is a type defined by the user). A primitive type can be deduced from its identifier and the explanations provided by the language manual. A complex type is more difficult of perceiving because it is composed by several identifiers, which are susceptible to do many analysis and the explanations exposed in the language manual are not enough. In this context, if the documentation provided is bad or null, the comprehension will be even more difficult.

Message Understanding Degree. This characteristic can be evaluated taking into consideration the following attributes: *Message Documentation Quality*, *Message Name Quality* and *Part Understanding Degree*. Concerning the first two elements, it is possible to say that they will provide relevant information when some semantic information can be extracted. The sub-characteristic named *Part Understanding Degree* is considered atomic.

Port Type Understanding Degree. This characteristic has the following attributes: *Port Type Name Quality*, *Port Type Documentation Quality* and *Operation Understanding Degree*. The first two are important because they provide semantic information when they are well defined. The third also is relevant because it is composed of several elements like name, documentation, parameters, etc. (see figure 10 for more details about the disaggregation of this subcharacteristic [8]). It is unnecessary to say that these elements provide interesting information.

Binding Understanding Degree. This characteristic is composed by the following attributes: *Binding Name Quality*, *Binding Documentation Quality*, *Binding Type Understanding Degree* and *Binding Operation Complexity*. Once more, the name quality and the documentation quality are important characteristics to measure using the attributes: *Binding Name Quality* and *Binding Documentation Quality*. The others two attributes are already defined in others characteristics. *Binding Type Understanding Degree* is defined in *Type Understanding Degree* and *Binding Operation Complexity* is defined in *Port Type Understanding Degree*. For this reason, during evaluation process we re-use the values obtained in previous computation.

[4] These characteristics were extracted from a WSDL specification provided by W3C.

Service Understanding Degree. A service is made available by a WSDL description. A service has a name and documentation and it is composed by ports. For analyzing the *Service Understanding Degree* it is necessary to measure *Service Name Quality*, *Service Documentation Quality* and *Service Port Understanding Degree* in a Service context.

7. Aggregation Structure

As LSP method states [56], the satisfaction values that result from the application of the Elementary Criteria Functions to the measurable attributes, must be aggregated in order to obtain the Global Preference. This Global Preference represents the satisfaction of the object under evaluation. As could be seen in section 6, we propose a Criteria Tree for each WSDL element (type, message, port, etc.). For each of these Criteria Trees, we developed a specific Aggregation Structure. To illustrate the approach and to save space, in Fig. 5 and Fig. 6, we only show the Aggregation Structure for the characteristics *Message Understanding Degree* and *Port Type Understanding Degree*.

For the first, we used a partial absorption LSP function (compound by operator A (arithmetic mean) and SQU (square mean) — all the LSP operators are better explained in [20]— to aggregate Message Documentation Quality and Message Name Quality. This kind of asymentric compound operators are used when some input values could be zero (non-mandatory input). It is necessary because in many cases, messages do not have a good documentation (sometimes do not have at all). A medium conjunctive operator (CA) is used to compute the Message Understanding Degree Global Preference. This kind of operator is employed when the input requirements are mandatory. Thus if one of the input values is zero, the operation result will be zero. The weights are used to express the relative importance of input preference. As message documentation and name provides more significant semantic information, its weight is 70%, as opposed to *Part Understanding Degree* which provides less semantic information (its weight is 30%).

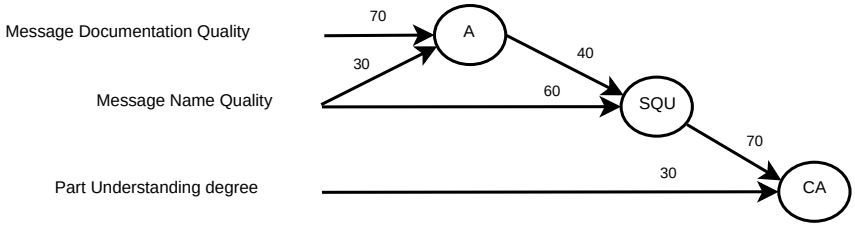

Fig. 5. *Message Understanding Degree* Aggregation Structure.

The second characteristic follows a similar approach, nevertheless the weights are changed and the satisfaction value of *Operation Understanding Degree* is computed using an aggretation structure (see Fig. 10 for more details about the aggregarion structure of this criterion). Because the *Port Type Name Quality* is an identifier and *Port Type Documentation Quality* is a documentation, they have the same characteristics as the equivalent message attributes.

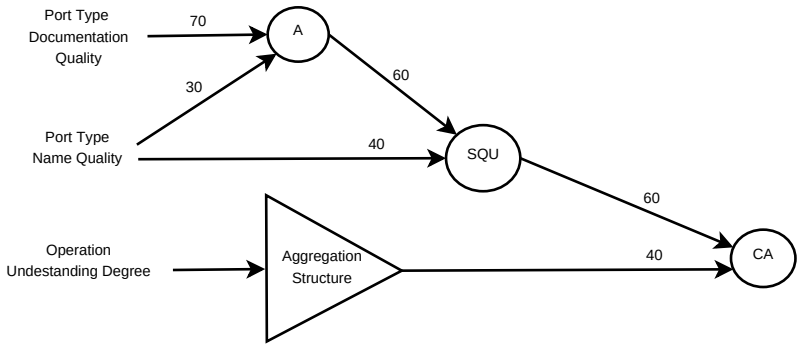

Fig. 6. *Port Type Understanding Degree* Aggregation Structure.

The weights assigned to edges related with the identifier and documentation follow the same distribution of the *Message Understanding Degree*, however the input edges to the CA operator indicates that the operation name and its documentation have the more importance level that *Operation Understanding Degree*. Nevertheless, the operations provide more information than the message parts because it has more elements (the parts are considered atomic elements). As a result of the previous observations, the port type operations have much information which is useful to comprehend the port type.

To have a deeper Port Type understanding, the operations provided by the port, must be comprehended. In order to estimate the operations understanding difficulty, an average of the satisfaction level of *Operation Understanding Degree* attribute can be computed. This task is carried out using the aggregation structure shown in Fig. 7. As it is possible to observe, the aggregation structure follows the same pattern for the attributes *Operation Documentation Quality* and *Operation Name Quality*. However the operator used to aggregate *Parameter Complexity* whith the other attributes is C-. It is because this operator does not produce a zero value when one of its inputs is zero. Moreover, the weights of its inputs indicates that the informal information is more important than information provided by *Parameter Complexity*.

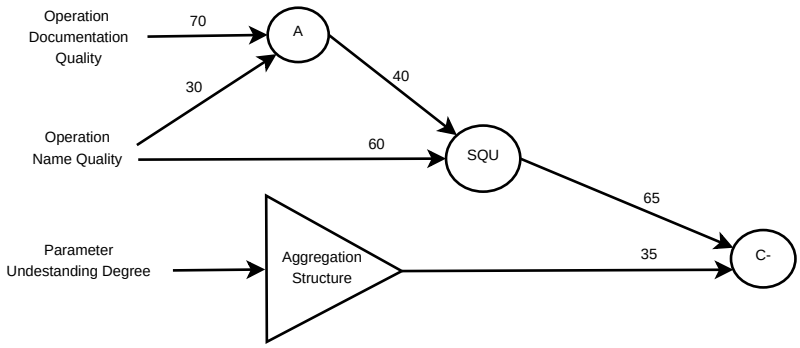

Fig. 7. *Operation Understanding Degree* Aggregation Structure.

Finally, the *Parameter Complexity* is also computed as average of understanding complexity of each parameter, which is calculated using the aggregation structure shown in Fig 8.

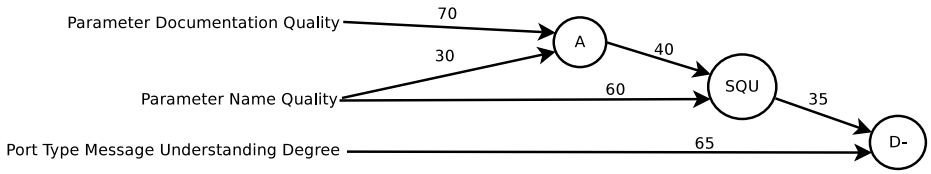

Fig. 8. *Parameter Complexity* Aggregation Structure.

Like the previous examples, the differences are centered in the aggregations of *Port Type Message Understanding Degree* with the others attributes and with the weights assigned. In this case, the operator used is D- (weak disjunction) which models interchangeability. The minus sign declares that the result obtained by D- is near to the aritmethic average (A). This peculiarity imposes restrictions to the total interchangeability. The weights of D- operators affirm that the information provided by the message is more relevant than the parameter qualitative information (Documentation and Name). It is because the messages have many elements which contribute to produce more semantic information.

For finishing this section, it is interesting to remark that the attributes' value related with names and documentations are calculated by the functions *nameQuality* and *documentationQuality* (both functions are specified in section 8).

The next step is the extraction of the information needed to compute the elementary criteria which allow to carry out the final evaluation. This topic is explained in next section.

8. Information Extraction Techniques and Elementary Criteria Functions

The information extraction techniques and the Elementary Criteria Functions are the most important features for the evaluation process that will be described. The former allows to obtain the information and perform all the analysis to get each attribute value for the Criteria Tree. The latter maps each of these in a satisfaction level, i.e., a value in the interval [0,1] (or [0,100]). This value represents the satisfaction degree of the attribute for the object under evaluation according to the sensibility and experience of the authors.

8.1. Information Extraction Techniques.

The approach used to extract information from a WSDL description combines compilation techniques, natural language processing algorithms and strategies to compute indicators [11]. As was described at the beginning of section 5, the first is implemented using DOM (Domain Object Model) a parser for XML language which explicitly builds an internal representation of the analysed XML source code. Several traversals are applied

through this internal representation for gathering the desired information. The identifiers and the documentation are extracted by using compilation techniques. In order to retrieve semantic information the *Information Extractor* [4] is used. Remember that this software is aimed at applying algorithms to divide, expand and find a meaning for the identifiers of a program. Finally, with the goal to provide a measure about of the understanding degree of a WSDL description, LSP Evaluator [38] was used. For attributes like *Type Name Quality*, *Message Name Quality* or *Binding Name Quality* we use identifier analysis techniques. The purpose of this analysis is to discover the relation between the names and the concepts of the problem domain. The name quality is higher when its related words are meaningful. The result of the techniques is a percentage which indicates the satisfaction level for a particular name quality.

For attributes like *Type Documentation Quality*, *Message Documentation Quality*, *Binding Documentation Quality*, etc., we use documentation analysis techniques. This kind of attributes has as main goal to measure the usefulness level of the information provided by the element's documentation (the *Information Extractor* also is used to carry out this task). *Type Name Quality* and *Message Name Quality* require similar analysis to their names that will be divided into words. Each word must be searched in the dictionaries to verify if it has a meaning, in this case the word is computed as valid and contributes to evaluate the identifier quality. The more meaninful words are found the higher the quality of the identifer. Function *nameQuality* shows how the identifier quality is computed.

Function nameQuality

```
input : name a word that represents a name.
input : d a domain specific dictionary
output: Satisfaction Level, a percentage that indicates
        the name quality.
Data: wordSet a set of words.
Data: wd a word extracted from a name.
Data: w a word.
Data: wordsWithMeans an integer variable which counts
      the number of words extracted from name which have
      meaning.
wordSet←samurai(name)- extractStopWords(wordSet);
wordsWithMean←0;
foreach w in wordSet do
    wd←classicExpansionAlgorithm(w);
    if hasMean(wd,d) then
        wordsWithMean←wordsWithMeans + 1;
end
return (wordsWithMeans / |wordSet|);
```

A similar process is carried out for the documentation, in this case it is divided in several words. Each word is searched in the dictionary and if it is found, then contributes

to produce a relevant documentation. The function *documentationQuality* shows how the process previously describe is accomplished.

Function documentationQuality

```
input : doc a string which represents a documentation.
input : d a domain specific dictionary.
Result: A percentage that indicates the documentation
        quality.
Data: wordSet a set of words.
Data: w a word.
Data: sl a real variable the number of word with
      meaning.
/* the function tokenizer divides the documentation in
words. The character used to divide is the blank space
*/;
wordSet← tokenizer(documentation);
wordSet← wordSet - extractStopWord(wordSet);
satisfactionLevel←0;
foreach w in wordSet do
  sl← sl + nameQuality(w,d);
end
;
return sl/|wordSet|;
```

8.2. Elementary Criteria Functions.

In this evaluation process, the majority of Elementary Criterion Function are direct mappings, since most of the attributes values are computed by extraction techniques. They take as input the strings to be analysed and return a percentage value that could directly be mapped to a satisfaction value. It is the case of parameters in the context of operations, to understand the parameter intention it is necessary take in consideration the following characteristcis: *parameter name*, *parameter documentation* and a *message* transmitted. The first characteristic (*name*) can be analyzed by using the function *nameQuality*. *Parameter Documentation*, like the documentation associated to other components, can be calculated by using the function *documentationQuality*. Regarding to messages, the *Message Understanding Degree* can be used to measure the Port Type Message Understanding complexity. Algorithm 1 shows the process to compute the parameters complexity.

After the analysis of all the operation's characteristics it is possible to propose the Algorithm 2 to estimate the *Operation Complexity*.

Algorithm 1: Parameter Complexity Satisfaction Level

input : *p* it is a list of operation's parameters to be evaluated. Each element of *p* is a triple (*n*,*d*,*m*) where *n* is the parameter name, *d* is the parameter documentation and *m* is the message associated to *p*.
input : *d* a domain specific dictionary.
Result: A percentage that indicates Parameter Complexity Satisfaction Level.
Data: *sl* a real variable. It holds a partial satisfaction level.
Data: *pn* a real variable. It holds the parameter name quality value.
Data: *pd* a real variable. It holds the parameter documentation quality value.
Data: *mud* a real variable. It holds the message understanding degree value.
Data: *temp1,temp2* real variables used to storage intermediate values.

```
sl←0;
foreach par in p do
    pn←nameQuality(getParameterName(par),d);
    pd←docQuality(getDocParameter(par),d);
    mud← messageUnderstandingDegree(getMessage(par),d);
    temp1← A(pd,70,pn,30);
    temp2← SQU(temp1,40,pn,60);
    sl← sl+D-(temp2,35,mud,65);
end
return sl/length(p);
```

9. Evaluation, experimental results

This section presents the evaluation of five WSDL descriptions using LSP and the structures defined in section 5 [36]. All descriptions analyzed and assessed define to web services frequently used by real world information systems:

i) *Google Web APIs*[8], provides operations to do Google searchs, ii) *Create Queue (Amazon)*[9], offers a reliable, highly scalable hosted queue for storing messages as they travel between computers, iii) *Airport*[10], provides useful information of all world airports (e.g. airport codes, names, countries, countries code, latitude, longitude, etc.) iv)

[8] https://code.google.com/p/dic/downloads/detail?name=GoogleSearch.wsdl
[9] http://queue.amazonaws.com/doc/2009-02-01/QueueService.wsdl
[10] http://www.webservicex.com/airport.asmx?wsdl

Algorithm 2: Satisfaction Level of Operation Complexity

```
input : ol a list of Port Type Operations. Each element
        of ol contains all the operation components,
        i.e: name, documentation, parameters.
input : d a domain specific dictionary.
output: A percentage that indicates the satisfaction
        level of the criterion Operation Complexity.
Data: sl a real variable.
Data: on a real variable. It holds the operation name
      quality value.
Data: od a real variable. It holds the operation
      documentation quality value.
Data: pc a real variable. It holds the parameter
      complexity value.
sl←0;
foreach op in ol do
    od← documentationQuality(getDocumentation(op),d);
    on←nameQuality(getName(op),d);
    /*parCompexitySatisfactionLevel is an abbreviation
    of parameterComplexitySatisfactionLevel */
    pc←parCompexitySatisfactionLevel(getParameters(op),d);

    sl← sl+C-(SQU(A(od,70,on 30),40,on,60),65, pc,35);
end
return sl/length(p);
```

Global Weather [11], gets weather report for all major cities around the world, and v) *OFAC* (`http://www.webservicex.net/OFACSDN.asmx?WSDL`) aids banks in meeting the requirements of the US Treasury Department's Office of Foreign Asset Control (OFAC).

In order to explain in detail how the valuation process operates up to reach the global Understanding Degree (UD), a special case of is developed (specifically Create Queue of Amazon). In particular, the Understanding Degree for the characteristic *Port Type*. Figure 10 exhibits the entire Criteria Tree and the Aggregation Structure for *Port Type UD*.

All operators in Aggregation Structure must be resolved in order of precedence, thus, in the first phase, all Elementary Criterion Functions must be computed. In the particular case of Port Type, all measurable attributes (represented by Criteria Tree leaves) are extracted by i) documentation and identifier analysis techniques or ii) are values previously computed (for example, *Message UD* for *Parameter UD*). The first level of operations are the calculations corresponding to *Parameter UD*. Once computed this value, the process estimates Operation UD and finally, all these values allows to obtain *Port Type UD*. When there are more than one element (for example 2 or more Parameters, Operations or Ports), the process obtains the average for all UD values.

[11] `http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL`

In the particular case of *Amazon Create Queue*, it has two Port Types: *QueueService-PortType* and *MessageQueuePortType*.

The port Type *QueueServicePortType* contains two operations: *CreateQueue* and *List-Queues*; all composed by two parameters: *input* and *output*. This port type is exposed in Fig. 9.

```
<wsdl:portType name="QueueServicePortType">
<wsdl:operation name="CreateQueue">
<wsdl:documentation>
The CreateQueue action creates a new queue, or returns
the URL of an existing one.
When you request CreateQueue, you provide a name for
the queue. To successfully create a new queue, you must
provide a name that is unique within the scope of your own
queues. If you provide the name of an existing queue, a new
queue isn't created and an error isn't returned. Instead, the
request succeeds and the queue URL for the existing queue is
returned.
Exception: if you provide a value for DefaultVisibilityTimeout
that is different from the value for the existing queue, you
receive an error.
</wsdl:documentation>
<wsdl:input message="tns:CreateQueueRequestMsg" wsa:Action=
"urn:CreateQueue"/>
<wsdl:output message="tns:CreateQueueResponseMsg" wsa:Action=
"urn:CreateQueue:Response"/>
</wsdl:operation>

<wsdl:operation name="ListQueues">
<wsdl:documentation>
The ListQueues action returns a list of your queues.
</wsdl:documentation>
<wsdl:input message="tns:ListQueuesRequestMsg"
wsa:Action="urn:ListQueues"/>
<wsdl:output message="tns:ListQueuesResponseMsg"
wsa:Action="urn:ListQueues:Response"/>
</wsdl:operation>
</wsdl:portType>
```

Fig. 9. Port Type *QueueServicePortType*

For example, the parameter *input* in operation *CreateQueue*, has a message *tns:Create-QueueRequestMsg*. This message obtained 82,3438 for *Message UD* (already computed).

This parameter does not have a name or documentation, thus *Parameter Name Quality* and *Parameter Documentation Quality* values are 0. The UD computation for this parameter (i.e, the partial preference computed by operator D-) is accomplished using the formula:

$$\left[(0,35) * (0)^{2,018} + (0,65) * (82,3438)^{2,018}\right]^{\frac{1}{2,018}}$$

which results in 66,5154.

The parameter *output* UD are computed likewise *input* UD employing the formula:

$$\left[(0,35) * (0)^{2,018} + (0,65) * (73,17)^{2,018}\right]^{\frac{1}{2,018}}$$

obtaining a UD value of 59,105.

Then, the process calculates the average value of both of them (in this case, the average value is 62,8102).

Next phase, *Operation UD* is calculated for *CreateQueue*. Aggregation Structure takes as input *Parameter UD* (average for all operation's parameters), *Operation Name Quality* and *Operation Documentation Quality*. The function *nameQuality* throws an elementary preference of 100 for the *Operation Name Quality*. Also, for *Operation Documentation Quality* the elementary preference is 100.

All previous values are computed by the operator C- and the result is 85,5193. This port type has a second operation named *ListQueues*. As this operation obtains exactly the same values for *Operation Name Quality*, *Operation Documentation Quality* and *Parameter Understanding Degree* the UD value for this operation is exactly as 'CreateQueue'. Likewise Parameter UD, all operations UD values must be averaged and it represents the Operation UD for a particular port type. For QueueServicePortType, the Operation UD is 85,5193 (both operations have the same UD).

Port Type Name Quality is 100 and *Port Type Documentation Quality* is 100. Operator CA returns the preference for this Port Type. As *Amazon* WSDL has two port types, their corresponding UD values must be averaged to obtain the whole *Port Type UD* for this WSDL. "QueueServicePortType" has an UD value of 81,9748 and "MessageQueuePortType" obtains an UD value of 82,2829. The average for these values is 82,12885, and represents the whole Port Type UD for this WSDL.

Previous paragraphs exhibit the operation of evaluation process, considering *Port Type UD*. Next paragraphs shows global Understanding Degree values obtained for all case studies.

Table 2 shows the global understanding degree for each WSDL description. Each *Global Preference* was computed aggregating all the characteristic preferences with the logical operator *CA* (this function simulates simultaneity) and the weight equally distributed among the characteristics (20% for each one). The choice of this operator is due to the fact that all WSDL components (type, message, port type, etc.) must be understandable. If one of these is incomprehensible, the whole WSDL will be difficult to understand.

As can be seen in Table 2, almost all WSDL are very similar taking into account understanding degree, except for *OFAC* WSDL description. This is because that description has numerous identifiers with acronyms which decreases the satisfaction levels.

Weather and *Airport* define each type using a few primitive and complex types. Furthermore they specify explicit and unambiguos identifiers. On the other hand, *Google* uses a number of primitive and complex types that exceed the established thresholds. The

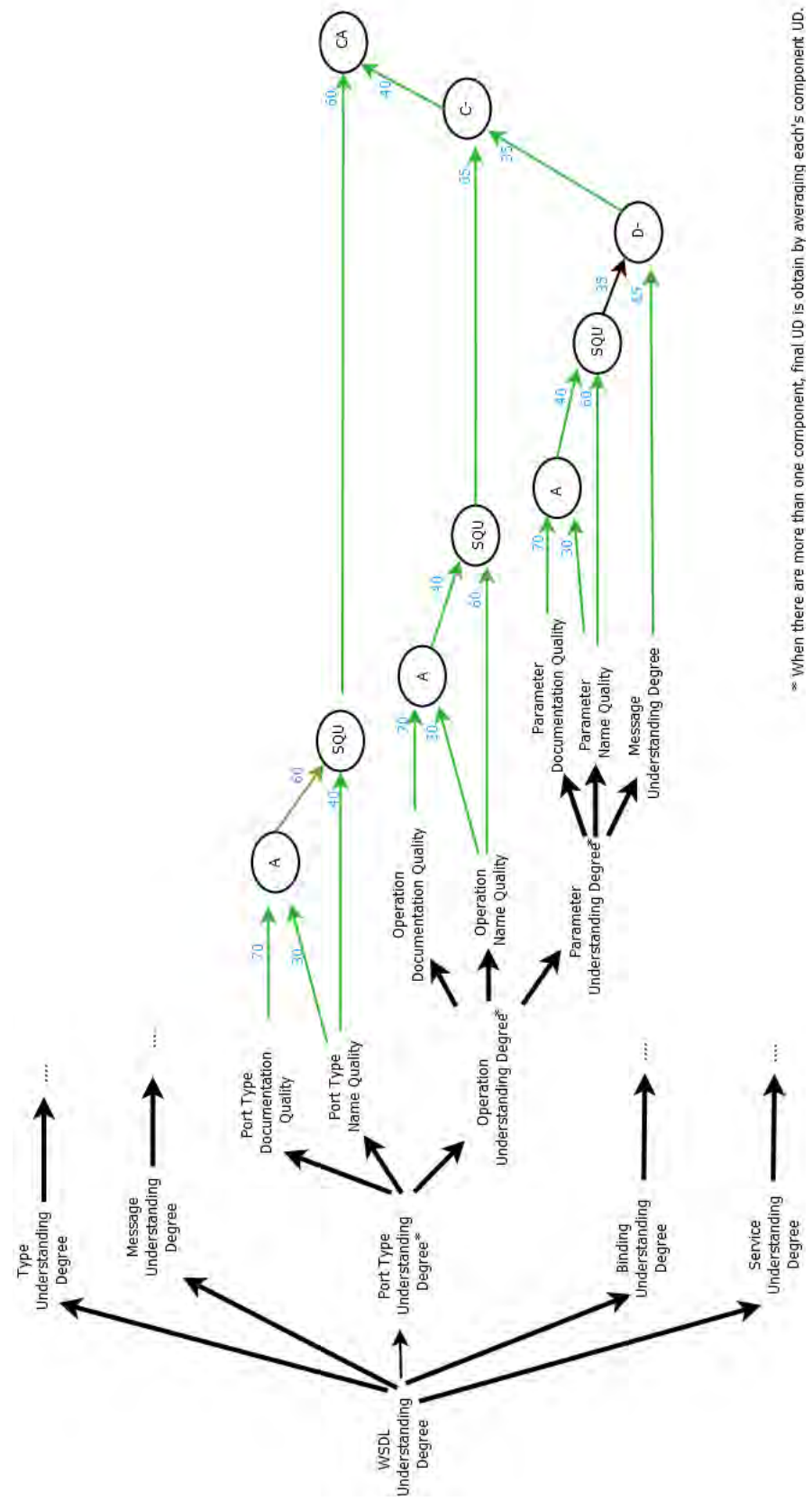

Fig. 10. Entire Criteria Tree and Aggregation Structure for *Port Type understanding Degree*.

Table 2. Partial and global evaluation of WSDL

High-Level Characteristic	Google	Weather	Amazon	Airport	OFAC
Types U. D.	60,2665	71,5131	68,8148	72,2303	40,5846
Messages U. D.	69,1173	83,3624	79,753	77,4924	58,8801
Port Types U. D.	75,7194	81,4166	82,1289	81,8902	45,3519
Bindings U. D.	75,5258	79,3457	82,2505	79,5241	42,755
Services U. D.	78,9946	79,6724	89,4138	79,7011	42,0794
Final Scores	71,5594	77,0112	80,1496	78,0884	45,4495

majority of messages's parts of *Weather* WSDL uses primitive types and this fact rise its *Messages Understanding Degree* satisfaction value.

In general, *Amazon* WSLD presents more documentation than others in different parts, like messages, types, port types and services. This makes this WSDL the most understandable of the case study.

From another point of view, this set of metrics was proposed to measure each component individually inside a WSDL. In this sense, we could compare, for example, all elements of a kind that a WSDL contains (e.g. types, messages or services), in order to analyze it individually. This could be useful for maintainability or re-structuring purposes. In this context, we measured the quality of three different messages of the Create Queue (Amazon) WSDL description; the results are shown in Table 3.

Table 3. Messages analysis for *Create Queue (Amazon)* WSDL description.

Sub-characteristic	SMR	RPR	DeleteMessageResponse
M. Doc. Quality	0	0	0
M. Name Quality	100	100	100
M. Parts U. D.	60,9759	93,6933	73,1726
Final Scores	73,17	83,5379	77,6729

As can be seen, *RemovePermissionRequest* (RPR) message is the most understandable of these three messages and *SendMessageResponse* (SMR) the worst. This is basically due to *Message Part Understanding Degree* satisfaction values.

This is a comparative analyse that allows to identify the most critical parts of the description. If we want to analyse the results individually we would say that a score below 50% represents a candidate description for improvement.

10. Conclusion and Future Work

In this article WSDLUD, a metric to measure the understanding degree of WSDL description, was defined. In order to compute WSDLUD other metrics were also specified. These metrics have as main goal to provide an estimation about the understanding degree of each description part. Each part is associated with an importance level specified by the engineer. Both values (understanding degree and importance level) are used by LSP (a multi criteria evaluation method) to produce a global value which represents the desired WSDL description understanding degree which proves our research hypotheses stated in the abstract.

We believe that our approach is novel because it makes possible to analyse each part of a particular WSDL description as well as the global understanding degree. Yet more important, all the engineer's experience can be included in the evaluation process in order to get more significant results. All the detailed information provided by our system can be used to identify the most critical parts of the description and the chances for quality improvement. In some cases, the description can be simplified or made more readable. But, in other cases, the complexity of the description is full dependent on the domain complexity and there is not chance for improvement.

As future work we intend to: i) Improve the documentation quality evaluation in order to cope with the possibility of identifying problem domain concepts and relations among the text; ii)Improve the Criteria Tree (CT) and Aggregation Structure (AS); iii) Extend the work presented in this paper to WSDL 2.0, this work is near to be finished and the results obtained will be presented in a future article; iv) Apply a similar analysis to study business processes specified with BPEL (Business Process Execution Language).

References

1. Abid, K., Abid, A., Ansari, M.: A better approach for conceptual readability of wsdl. In: Multimedia Big Data (BigMM), 2015 IEEE International Conference on. pp. 260–263 (April 2015)
2. Aguezzoul, A., Rabenasolo, B., Jolly-Desodt, A.M.: Multicriteria Decision Aid Tool for Third-Party Logistics Providers' Selection. In: Service Systems and Service Management, 2006 International Conference on. vol. 2, pp. 912–916 (2006)
3. Allamaraju, S.: RESTful Web Services Cookbook. O'Reilly (2010)
4. Azcurra, J., Berón, M., Montenjano, G., Farnese, A., Henriques, P., M.Pereira.: AId: Uma Ferramenta para Análise de Identificadores de Programas Java. In: Congreso Nacional de Ingeniería Informática/Sistemas de Información, 2014. pp. 880–892 (Nov 2014)
5. Azcurra, J., Beron, M., Montejano, G.: Análisis de Identificadores para Abstraer conceptos del Dominio del Problema. Universidad Nacional de San Luis (2015)
6. Bernardis, H., Beron, M., Riesco, D., Henriques, P.R.: Extracción de información y cálculo de métricas en WSDL 1.1 y 2.0. In: Congreso Nacional de Ingeniería Informática/Sistemas de Información. pp. 963–974 (Nov 2014)
7. Beron, M., Montejano, G., Riesco, D., Henriques, P., Debnath, N.: Sip: A simple tool for inspecting and evaluating wsdl specifications. In: Information Technology: New Generations (ITNG), 2013 Tenth International Conference on. pp. 14–19 (April 2013)
8. Beron, M., Henriques, P.R., Riesco, D., Pereira, M.J.V.: On the Comprehension of WSBPEL Programs. Tech. rep., Universidad Nacional de San Luis - Universidade do Minho (2015)
9. Brans, J.P., Mareschal, B.: PROMETHEE methods. In: Multiple criteria decision analysis: state of the art surveys, pp. 163–186. Springer (2005)

10. Brans, J., Vincke, P., Mareschal, B.: How to select and how to rank projects: The Promethee method. European Journal of Operational Research 24(2), 228–238 (1986), ¡ce:title¿Mathematical Programming Multiple Criteria Decision Making¡/ce:title¿
11. Carvalho, N.R.: An Ontology Toolkit for Problem Domain Concept Loction in Program Comprehension. Ph.D. thesis, Escola de Engenharia, Universidade do Minho (2014)
12. Carvalho, N.R., Almeida, J.J., Henriques, P.R., Pereira, M.J.V.: From source code identifiers to natural language terms. Journal of Systems and Software 100, 117–128 (2015), http://dx.doi.org/10.1016/j.jss.2014.10.013
13. Coscia, L.O., Crasso, M., Mateos, C., Zunino, A.: Estimating Web Service interface quality through conventional object-oriented metrics. CLEI Electronic Journal 16(1) (2013), http://www.clei.org/cleiej/paper.php?id=258
14. Coscia, L.O., Mateos, C., Crasso, M., Zunino, A.: Refactoring code-first Web Services for early avoiding WSDL anti-patterns: Approach and comprehensive assessment. Sci. Comput. Program. 89, 374–407 (2014), http://dx.doi.org/10.1016/j.scico.2014.03.015
15. Crasso, M., Rodriguez, J.M., Zunino, A., Campo, M.: Revising wsdl documents: Why and how. IEEE Internet Computing 14(5), 48–56 (2010)
16. Deng, L., Zhou, A., Guo, H., Han, P.: Investigating, modeling and evaluating the interface complexity of web services. In: Communications, Computing and Control Applications (CCCA), 2011 International Conference on. pp. 1–6 (March 2011)
17. Dujmović, J., Kadaster, M.: A technique and tool for software evaluation. Evolution 374, 246 (2002)
18. Dujmovic, J., Elnicki, R., of Florida, U., of Standards, U.S.N.B.: A DMS Cost/benefit Decision Model: Mathematical Models for Data Management System Evaluation, Comparison and Selection (part 1 of Second Deliverable). National Bureau of Standards (1981)
19. Dujmovic, J.: Continuous Preference Logic for System Evaluation. IEEE Transactions on Fuzzy Systems 15(6), 1082–1099 (Dec 2007)
20. Dujmovic, J.: Characteristic forms of generalized conjunction/disjunction. In: Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on. pp. 1075–1080. IEEE (2008)
21. Dujmović, J.J., Ralph, J.W., Dorfman, L.J.: Evaluation of Disease Severity and Patient Disability Using the LSP Method. In: Proceedings of the 12th Information Processing and Management of Uncertainty international conference (IPMU 2008). pp. 1398–1405 (2008)
22. Enslen, E., Hill, E., Pollock, L., Vijay-Shanker, K.: Mining source code to automatically split identifiers for software analysis. In: Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on. pp. 71–80. IEEE (2009)
23. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. 2(2), 115–150 (may 2002), http://doi.acm.org/10.1145/514183.514185
24. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis (2000), aAI9980887
25. Gold, N., Bennett, K.: Program comprehension for web services. In: Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on. pp. 151–160 (June 2004)
26. Guerrouj, L., Di Penta, M., Antoniol, G., Guéhéneuc, Y.G.: Tidier: an identifier splitting approach using speech recognition techniques. Journal of Software Maintenance and Evolution: Research and Practice (2011)
27. Guinard, D., Ion, I., Mayer, S.: Mobile and Ubiquitous Systems: Computing, Networking, and Services: 8th International ICST Conference, MobiQuitous 2011,Copenhagen, Denmark, December 6-9, 2011, Revised Selected Papers, chap. In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers' Perspective, pp. 326–337. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-30973-1_32

28. Haidar, A., Abdallah, A.: Abstractions of web services. In: Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on. pp. 182–191 (June 2009)
29. Hu, X., Feng, Z., Chen, S.: Analyzing distribution of implicit semantic information in web services. In: IEEE 37th Annual Computer Software and Applications Conference, COMPSAC Workshops 2013, Kyoto, Japan, July 22-26, 2013. pp. 415–420 (2013), http://dx.doi.org/10.1109/COMPSACW.2013.122
30. Jiang, J., Lipponen, J., Selonen, P., Systa, T.: Uml-level analysis and comparison of web service descriptions. In: Ninth European Conference on Software Maintenance and Reengineering. pp. 236–240 (March 2005)
31. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge University Press (1993)
32. Kumar, R., Indraveni, K., Goel, A.K.: Automation of detection of security vulnerabilities in Web Services using dynamic analysis. In: 9th Int. Conf. on Internet Technology and Secured Transactions (ICITST). pp. 334–336 (Dec 2014)
33. Kumari, S., Rath, S.K.: Performance comparison of soap and rest based web services for enterprise application integration. In: Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on. pp. 1656–1660 (Aug 2015)
34. Lawrie, D., Feild, H., Binkley, D.: Extracting meaning from abbreviated identifiers. In: Seventh IEEEInternational Working Conference on Source Code Analysis and Manipulation. p. 213–222 (Sept 2007)
35. Liu, F., Shi, Y., Yu, J., Wang, T., Wu, J.: Measuring similarity of web services based on wsdl. In: Web Services (ICWS), 2010 IEEE International Conference on. pp. 155–162 (July 2010)
36. Liu, L., Sun, T., Fang, W., Liu, N.: Usability evaluation of the subway train dispatching system. In: Information Science and Technology (ICIST), 2011 International Conference on. pp. 1123–1128 (March 2011)
37. Lu, X., Lin, J., Zou, Y., Peng, J., Liu, X., Zha, L.: Investigating, modeling, and ranking interface complexity of web services on the world wide web. In: Services (SERVICES-1), 2010 6th World Congress on. pp. 375–382 (July 2010)
38. Miranda, E., Berón, M., Montejano, G., Pereira, M.J.V., Henriques, P.R.: NESSy: a New Evaluator for Software Development Tools. In: 2nd Symposium on Languages, Applications and Technologies, SLATE 2013, June 20-21, 2013 - Porto, Portugal. pp. 21–37 (2013), http://dx.doi.org/10.4230/OASIcs.SLATE.2013.21
39. Mirtalebi, A., Khayyambashi, M.: Enhancing security of web service against wsdl threats. In: Emergency Management and Management Sciences (ICEMMS), 2011 2nd IEEE International Conference on. pp. 920–923 (Aug 2011)
40. Montazer, G.A., Saremi, H.Q., Ramezani, M.: Design a new mixed expert decision aiding system using fuzzy electre iii method for vendor selection. Expert Syst. Appl. 36(8), 10837–10847 (Oct 2009), http://dx.doi.org/10.1016/j.eswa.2009.01.019
41. Mousseau, V., Slowinski, R., Zielniewicz, P.: A user-oriented implementation of the ELECTRE-TRI method integrating preference elicitation support. Comput. Oper. Res. 27(7-8), 757–777 (Jun 2000), http://dx.doi.org/10.1016/S0305-0548(99)00117-3
42. Nandigam, J., Gudivada, V., El-Said, M.: Teaching web services using wsexplorer. 2013 IEEE Frontiers in Education Conference (FIE) 0, S3H–20–S3H–25 (2007)
43. Newcomer, E.: Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley Professional (2002)
44. Ni, Y., Fan, Y.: Ontology based cross-domain enterprises integration and interoperability. Services Part II, IEEE Congress on 0, 133–140 (2008)
45. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. IEEE MultiMedia, 2002 09(4), 20–29 (2002)
46. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. "big"' web services: Making the right architectural decision. In: Proceedings of the 17th International Conference

on World Wide Web. pp. 805–814. WWW '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1367497.1367606
47. Romero, C.: Teoría de la Decisión Multicriterio: Conceptos, Técnicas y Aplicaciones. Alianza Editorial: Madrid. (1993)
48. Roy, B.: Problems and methods with multiple objective functions. Mathematical Programming 1, 239–266 (1971), http://dx.doi.org/10.1007/BF01584088
49. S. Ahuja, K. Umapathy, Z.P.: Comparing performance of web service interaction styles: Soap vs. rest. In: Conference on Information Systems Applied Research, 2012 Proceedings of the (2012)
50. Saaty, T.: How to make a decision: The analytic hierarchy process. European Journal of Operational Research 48(1), 9–26 (Sep 1990)
51. Segev, A., Toch, E.: Context-based matching and ranking of web services for composition. IEEE Transactions on Services Computing 2(3), 210–222 (2009)
52. Shahgholi, N., Mohsenzadeh, M., Seyyedi, M., Qorani, S.: A new soa security framework defending web services against wsdl attacks. In: Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on. pp. 1259–1262 (Oct 2011)
53. Simon, A., Rischbeck, T.: Service contract template. In: IEEE SCC. p. 511. IEEE Computer Society (2006), http://dblp.uni-trier.de/db/conf/IEEEscc/scc2006.html#SimonR06
54. Sneed, H.: Measuring web service interfaces. In: Web Systems Evolution (WSE), 2010 12th IEEE International Symposium on. pp. 111–115 (Sept 2010)
55. Sripairojthikoon, P., Senivongse, T.: Concept-based readability measurement and adjustment for web services descriptions. In: 16th Int. Conf. on Advanced Communication Technology (ICACT). pp. 378–388 (Feb 2014)
56. Su, S., Dujmovic, J., Batory, D.S., Navathe, S.B., Elnicki, R.: A Cost-benefit Decision Model: Analysis, Comparison and Selection of Data Management. ACM Trans. Database Syst. 12(3), 472–520 (Sep 1987), http://doi.acm.org/10.1145/27629.33403
57. Tekli, J., Damiani, E., Chbeir, R., Gianini, G.: Soap processing performance and enhancement. vol. 5, pp. 387–403 (Third 2012)
58. Tibermacine, O., Tibermacine, C., Cherif, F.: A Practical Approach to the Measurement of Similarity between WSDL-basedWeb Services. RNTI: Revue des Nouvelles Technologies de l'Information Special Issue CAL 2013(RNTI-L-7), 03–18 (2014)
59. Wang, L., Xu, L., Yu, J., Xue, Y., Zhang, G., Luo, X.: Search of web service based on association rule. In: Cognitive Informatics Cognitive Computing (ICCI*CC), 2015 IEEE 14th International Conference on. pp. 262–266 (July 2015)
60. Wang, L., Liu, F., Zhang, L., Li, G., Xie, B.: Enriching descriptions for public web services using information captured from related web pages on the internet. In: The Fifth IEEE International Symposium on Service-Oriented System Engineering, SOSE 2010, June 4-5, 2010, Nanjing, China. pp. 141–150 (2010), http://dx.doi.org/10.1109/SOSE.2010.28

Mario Berón obtained a PhD in Computer Science from National University of San Luis (UNSL/Argentina) also recognized by Minho University (Portugal). He is Adjunct Professor and a researcher at UNSL. He is a professor of postgraduate course in Master of Software Engineering and a member of the Software Engineering group at UNSL. His research interests are in the área of Program Comprehension, Reverse Engineering, Programming Languages, Security Informatics and Embedded Systems, among others.

Hernán Bernardis obtained his bachelor degree in Computer Science from National University of San Luis (UNSL). He is teacher assistant and researcher at this institution.

He is finishing a Master on Software Engineering also at UNSL where he is a member of the Software Engineering group. His research interests include Reverse Engineering, Program Comprehension, Software Engineering and Programming Languages.

Enrique A. Miranda obtained his bachelor degree in Computer Science from National University of San Luis (UNSL). He is teacher assistant and researcher at this institution. He pursues a PhD in Informatics Engineering with a PhD scholarship from National Scientific and Technical Research Council (CONCIET). He is member of the Software Engineering group at UNSL. His research interests include Reverse Engineering, Program Comprehension, Software Engineering and Programming Languages.

Daniel Riesco has a PhD from University of Vigo (Spain) and a Master from Polytechnic University of Madrid (Spain). He is Associate Professor at National University of San Luis (Argentina) of Informatics Engineering. He is director of a research project with 15 members and co-director of the Software Engineering Group. He has more than 100 referred research articles in international journals, congress and workshops from IEEE, ACM, Springer, among others.

Maria João Tinoco Varanda Pereira has a PhD and a Master on Computer Science and Software Engineering from Minho University (Portugal). She is Adjunt Professor at Polytechnic Institute of Bragança (also in Portugal) and member of Processing Language Group at Minho University. Her research work and teaching activity is deeply related with formal specification of languages, automatic construction of compilers and other language-based tools, conception and implementation of visual domain specific languages, visualization and animation of programs and program comprehension. She has 19 journal articles and 56 international conference papers.

Pedro Rangel Henriques has a PhD on Computer Science and is an Associate Professor at Minho University. His main research areas are Language Specification and Processing (methods & tools): Grammar development and Parsing algorithms, Attribute Grammars and Attribute Evaluation, Attributed Tree Transformation, Code Generation and Optimization, Language Processor Generators (compiler compilers), Visual Language parsing and translation, Program Visualizers and Animators; Programming Languages and Paradigms (imperative, declarative, OO); Program Comprehension (models, approaches, and tools); Document Specification and Processing (markup languages), Knowledge Representation and Discovery (data-mining and text-mining). Pedro Henriques has an extensive experiencie in teaching and researching. He teaches several disciplines, at Minho University, related with Language Processing, Programming Paradigms and Programming Methods. He has been a leader of several research projects, in this context, and has supervised many master and phd thesis; he has produced more than 100 referred articles in international journals, books, congress and workshops.

Received: January 25, 2016; Accepted: July 31, 2016.