Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche

researchportal.unamur.be

UNIVERSITÉ DE NAMUR

RESEARCH OUTPUTS / RÉSULTATS DE RECHERCHE

Towards Automated Alignment of Web Services to Requirements

Verlaine, Bertrand; Dubois, Yves; Jureta, Ivan; Faulkner, Stephane

Published in:
Proceedings of the First International Workshop on the Web and Requirements Engineering in collaboration with the 18th IEEE International Requirements Engineering Conference (RE'10), Sydney, Australia

Publication date:
2010

Document Version
Early version, also known as pre-print

[Link to publication]

Towards Automated Alignment of Web Services to Requirements

Bertrand Verlaine*, Yves Dubois*, Ivan J. Jureta* and Stéphane Faulkner*

*PReCISE Research Center

University of Namur

{bertrand.verlaine,yves.dubois,ivan.jureta,stephane.faulkner}@fundp.ac.be

Abstract—The engineering of a web service-oriented system requires the specification of the functions that the various Web Services (WSs) should provide, before WSs are either built or selected. Being written in a service description language, the service specification instantiates concepts different than those of interest during the requirement engineering (RE): the former speaks in terms of, e.g., operations and bindings, while the latter manipulates, e.g., goals and domain assumptions. It is, however, clear that functions expected of WSs will be relevant to the stakeholders if and only if they satisfy the stakeholders' requirements. There is therefore a gap between the two specifications which must be bridged in order to ensure that the WS system is adequate w.r.t. stakeholders' requirements. This paper proposes mappings between the RE concepts and those of the WS Description Language (WSDL) and WS Level Agreement (WSLA). A working prototype is presented that implements the mappings and allows automated translation of the instances of RE concepts into instances of WSDL and WSLA concepts. The mappings and the prototype facilitate the engineering of WS systems, as fragments of WS descriptions can be generated from requirements.

Index Terms—Requirements Engineering for Service-oriented Computing

I. INTRODUCTION

Engineering and managing the operation of increasingly complex information systems is a key challenge in computing (e.g., [1], [2]). It is now widely acknowledged that degrees of automation needed in response cannot be achieved without distributed, interoperable, and modular information systems. Among the various, often overlapping approaches to building such systems, service-orientation stands out in terms of its reliance on the World Wide Web infrastructure, availability of standards for describing and enabling interaction between services, attention to interoperability and uptake in industry.

A *service* is a self-describing and self-contained modular application designed to execute a well-delimited task, and that can be described, published, located, and invoked over a network [2], [3]. A *Web Service* (WS) is a service that can be invoked over the World Wide Web. WSs are offered by service providers that ensure service implementations, advertise service descriptions, and provide related technical and business support.

The engineering of WS-oriented systems involves many issues treated in the literature — among them, infrastructure for services (e.g., [4]–[6]), descriptions of services' interfaces, capabilities, behaviors, and qualities (e.g., [7]–[10]), service discovery (e.g., [11]), service composition (e.g., [12]–[15]), and ontologies and ontology languages (e.g., [10], [16]–[20]).

1) Problem: The engineering of a WS-oriented system cannot be successful if the services intervening in it cannot satisfy the requirements of the systems' stakeholders. Requirements Engineering (RE) for such systems is a promising area of inquiry that already attacked some of the key issues. One pressing concern which has received less attention and is the focus of this paper is *how to bridge the gap between a specification of requirements and* WS *descriptions?* A description of a WS specifies the functions that the WS can or should provide. It is on the basis of such a specification that WSs are developed or sought among available ones. Specialized languages have been designed for the description of WSs using concepts of, e.g., operation and binding, tailored to the WS description. On the other hand, requirements that these services ought to satisfy are classified according to ontologies tailored to RE, which rely on concepts such as goal, task, and domain assumption. While it is clear that the functions expected of WSs will be relevant to the system if and only if they satisfy the stakeholders' requirements, the differences in the conceptualizations that underlie WS descriptions and RE specifications make it unclear how exactly to translate a requirements specification into WS descriptions, hence the gap.

2) Contributions: This paper is a first step towards addressing the gap between RE specifications and WS descriptions by mapping the concepts of the core ontology for RE [21] to the concepts of the World Wide Web Consortium's WS Description Language (WSDL) [22] and the WS Level Agreement (WSLA) [23] formalism. Two contributions are made. Firstly, the mappings between the two requirements representations are presented both informally and in the Distributed Description Logic formalism, and the rationale for the mappings is discussed. Once the mappings are available and a requirements specification is given, it is possible to facilitate the writing of WS descriptions in WSLA/WSDL by translating fragments of the requirements specification into fragments of WSLA/WSDL descriptions. The second contribution is the working prototype tool that implements the mappings, allowing thereby the translation of the instances of requirements concepts into instances of WSLA/WSDL concepts. The mappings and the prototype facilitate the engineering of WS systems, as fragments of service descriptions can be generated from requirements.

3) Organization: The ontology for RE, and WSDL and WSLA are presented first informally (§II). Then, the formalization of

the taxonomies is presented followed by the mapping between them (§III). This mapping allows us to build a tool which is able to automate the specification of requirements into technical documents (§IV). Finally, we briefly relate comparable discussions (§V) before drawing up conclusions followed by some interesting directions for future work (§VI).

II. BASELINE

To bridge the gap between requirements and WS specifications, we use a requirements ontology and we build a service taxonomy. By *requirements specification*, we mean the specification of the requirements problem and of its alternative solutions. The core ontology for requirements (CORE) [21], [24] was adopted as it carries concepts playing a central role in the definition of the requirements problem and of its solutions. At the service level, we use the WSDL and WSLA formalisms. We need two languages at the service level because of the inability of either of them alone to cover the scope of CORE.

A. The Core Ontology for Requirements

The root concept of the taxonomy of CORE is Communicated information, specialized as follows:

1) Goal, specialized on: Functional goal, Quality constraint, Softgoal;
2) Plan;
3) Domain assumption, specialized on: Functional domain assumption Quality domain assumption, Soft domain assumption;
4) Evaluation, specialized on: Individual evaluation, Comparative evaluation.

A basic idea in CORE is that requirements are communicated by the stakeholders to the requirements engineer, so that the latter classifies requirements based on how and what was communicated and how. The Communicated information concept is a catchall one, the instances of which are propositions communicated by the stakeholders. Once an instance of that concept is available, the question to ask is what mode was that proposition communicated in. The notion of mode (or modus in linguistics) reflects the idea that we can distinguish between the content of a communication and the intentional state it was communicated in, whereby different kinds of mode correspond to different intentional states of the stakeholder. If the stakeholder tells the engineer that she believes that some condition holds in the operating environment of the system-to-be, then the proposition stating the condition is an instance of the Domain assumption concept. If she instead desires that the condition be made to hold by the system-to-be, then the proposition is an instance of the Goal concept. In case an intention to perform particular actions is conveyed, which may then be delegated to the system-to-be, the engineer classifies the propositions describing these actions as instances of the Plan concept. Since stakeholders can also indicate that they prefer some goals to be satisfied than others, or that some of them must be satisfied, while others are optional, CORE includes the concept of Evaluation the instances of which convey evaluations arising out of emotions.

CORE distinguishes three kinds of goals. Functional goal refers to a desired condition the satisfaction of which is verifiable and is binary, i.e., it is either satisfied or not. quality constraints define desired values of non-binary measurable properties of the system-to-be (e.g., how many seconds it takes to answer a query). As functional goals and quality constraints are not necessarily known at the very start of the RE process, the Softgoal concept is instantiated to capture requirements that refer to vague properties of the system-to-be (e.g., a "fast" IS). Same specialization applies to the Domain assumption concept, which has its functional variant (which refers to binary properties of the system-to-be and/or its environment), its quality variant, Quality domain assumption, and its soft variant, Soft domain assumption. Finally, Evaluation can qualify individual requirements through Individual evaluation) or compares goals, domain assumptions, and/or plans trough Comparative evaluation.

B. The Web Service Taxonomy

IBM's WSLA [23] intends to specify contracts called Service Level Agreements (SLAs). The topic of the contracts are constraints on Quality of Service (QoS) properties of WSs. While WSLA focuses on the quality of WSs, WSDL[1] [22], the second formalism used, is commonly used to specify the functional characteristics of a WS.

The WSLA concepts are *Party*, *Service definition*, *Metric* and *Obligations*[2]. The WSDL concepts are *Operation*, *Binding* and *Service*. We retain the following four of these seven concepts.

1) *Metric* identifies an observable property of a WS when the WS is in use, and indicates the measurement directive for that property, i.e., it specifies how that property can be accessed [23], [25].
2) The *Obligations* concept defines the guaranteed QoS level of the WS identified in the *service definition* as well as constraints imposed on the metrics and triggered actions [23], [25]. *Obligations* is specialized on:
   a) *Service level objective* which defines the different assurances regarding the observable characteristics of the WS, and
   b) *Action guarantee* which groups promises of the signatory parties and/or of third parties concerning the achievement of an action when a determined precondition occurs.
3) *Operation* defines the interaction between the service and the other parties involved in the interaction, as a sequence of input and output messages [22], [26].
4) *Binding* specifies concrete message format and transmission protocol details concerning the WS use [26].

Party, *Service definition*, and *Service* are not retained as targets of our mappings, for the following reasons:

---

[1]WSDL allows fault management by the specification of fault conditions and repair actions, which certainly is relevant given that WS oriented systems are often distributed and given potential Web server breakdowns. We leave out this aspect of WSDL in this paper, leaving it for future work.
[2]An WSLA or an WSDL concept is depicted as *Concept* and an instance of one of those concepts is depicted as *instance*.

- Instances of *Party* identify the WS provider, the WS consumer and third parties, which may be stakeholders expressing requirements w.r.t. the service use. As the definition of the requirements problem abstracts from these identifiers, we do not carry information on what stakeholder gave which requirement to the level of the services.
- A *Service definition* instance is not directly evaluated by the WS consumer. It links a document which describes the functional characteristics of the WS. As we use WSDL, the WS consumer (i.e., a stakeholder) can directly evaluate the functional characteristics through the WSDL document.
- *Service* is not relevant in the present discussion, as the actual Web location of the WS is unimportant because of the intrinsic nature of the Internet. Only its presence or absence is crucial. If important for the consumer, the possible unresponsiveness of the WS could be evaluated through other concepts (e.g., an *obligations*).

III. FORMALIZATION AND MAPPING OF CORE AND WSDL/WSDL

In order to bridge CORE and WSLA/WSDL, we use the description logic $\mathcal{SIN}$ [27] first to rewrite each taxonomy. This rewriting allows us to connect WSDL to WSLA (to get what we refer to as WSLA/WSDL), and then CORE to WSLA/WSDL (see §III-C).

A. Formalization

1) CORE in description logic: Table I is based on the definitions and axioms of the CORE ontology given. Line 1 defines the root concept of the ontology. Requirements expressed during RE are classified into the four main classes of CORE, i.e., Goal, Plan, Domain assumption and Evaluation, and finally in the leaves of CORE, i.e., Quality constraint, Soft domain assumption, Comparative evaluation, and so on. Detailed informal definitions of the CORE concepts are not repeated here. Unchanged softgoals and soft domain assumptions cannot be propagated to the level of service descriptions: they need to be replaced by more precise requirements. Just as, say, imprecise goals are refined, so are softgoals and soft domain assumptions approximated [21], [24], whereby their approximation involves the identification of quality constraints and quality domain assumptions, while comparative evaluations may indicate how alternative quality constraints or quality domain assumptions may rate in terms of relative desirability. Lines 10 and 13 reflect this in the ontology.

2) WSDL and WSLA in description logic: Lower two parts of Table I are based on publications about the WSLA formalism [23], [25] and on the W3C recommendations concerning WSDL 2.0 [22], [26], [28]. Line 17 (WSLA) states the use of the WSLA specification as a proposal or an agreement. The latter is the primary purpose of WSLA. A proposal could be suggested either by a WS consumer or a WS provider. Requirements concerning non-functional WS properties are specified via WSLA. COMMITMENT, used in Lines 18, 21 and 41, refers to a promise to achieve (conditionally or not) a predetermined task. SLA PARAMETERs are observable characteristics used to evaluate the QOS of the WS as well as their measurement process (Lines 34 and 37). Line 36 uses Distributed Description Logic (DDL) [29] in order to bridge WSLA with WSDL.

Line 44 (WSDL) has the same purpose as Line 17 of the WSLA taxonomy. Line 54 covers the *Operation* concept: by ordering the messages exchanged between the WS provider and the WS consumer, it organizes the data flow. Though this data flow, the service provided by the WS is structured. It enables to know what is the function of the service provided.

B. Bridging the WSLA/WSDL concepts and the four main CORE classes

The first step in the mapping building is to classify the WSLA and the WSDL concepts into one of the four main classes of the CORE ontology, i.e., Goal, Plan, Domain assumption and Evaluation. The methodology is as follows. First, we check if the WS consumer can perform a specific speech act — corresponding to a specific CORE concept — about instances of the studied WSLA or WSDL concept. Then, we verify if the WSLA or the WSDL specification allow the representation of what the requirement conveys. Otherwise, some requirements could be lost during the mapping.

Table II, grounded on the definitions of the CORE concepts (see below) and of the WSLA/WSDL concepts (see §II-B), illustrates this classification; explanations follow it.

The Goal concept captures conditions not yet satisfied that the service consumer desires to see become true in the future [21]. Goal is mapped with the four WSLA/WSDL concepts. The consumer can express her desire about the presence or absence of a particular observable property, i.e., a *metric*, which can be included in the future electronic agreement. The WS consumer can also express her desire (i) to set the value of a *service level objective* to a specific number, (ii) and/or that a party involved in the future agreement achieves a particular action specified via an *action guarantee*. Those two kinds of desires can be specified in an WSLA proposal as *obligations*. Concerning the *Operation* and *Binding* concepts, the service consumer can respectively indicate her desire about a precise pattern of exchanged messages with particular input and output, and/or her desire about a particular message format or transmission protocol. These two requirements can be specified inside an *operation* — where the important pieces of information for the WS consumer are the first output of data sent and the final input of data received — or a *binding*.

A plan catches intentions that the service consumer intends to perform, conditionally or not. This concept is also mapped with all WSLA/WSDL concepts. The WS consumer can express her intention to perform the measurements of QOS properties via a *metric* and then deliver the results to other parties. The WS consumer can aim at performing an *action guarantee*, instance of *Obligation*. The WS consumer can also promise to send predetermined messages which are specified inside an *operation*, or to use particular message formats and/or communication protocols which can be specified through a *binding*.

		CORE ontology	
1 :	COMMUNICATED INFORMATION	$\equiv$	GOAL $\sqcup$ PLAN $\sqcup$ DOMAIN ASSUMPTION $\sqcup$ EVALUATION
2 :	$\bot$	$\sqsubseteq$	GOAL $\sqcap$ PLAN $\sqcap$ DOMAIN ASSUMPTION $\sqcap$ EVALUATION
3 :	**refine**	$\equiv$	**refined-by**$^-$
4 :	**refined-by**	$\equiv$	**refine**$^-$
5 :	$\top$	$\sqsubseteq$	$\forall$ **refine**.COMMUNICATED INFORMATION
6 :	$\forall$ **refine**.GOAL	$\equiv$	FUNCTIONAL GOAL $\sqcup$ QUALITY CONSTRAINT $\sqcup$ SOFTGOAL
7 :	$\bot$	$\sqsubseteq$	FUNCTIONAL GOAL $\sqcap$ QUALITY CONSTRAINT $\sqcap$ SOFTGOAL
8 :	**approximate**	$\equiv$	**approximated-by**$^-$
9 :	**approximated-by**	$\equiv$	**approximate**$^-$
10:	SOFTGOAL	$\sqsubseteq$	$\exists$ **approximate**.QUALITY CONSTRAINT
11:	$\forall$ **refine**.DOMAIN ASSUMPTION	$\equiv$	FUNCTIONAL DOMAIN ASSUMPTION $\sqcup$ QUALITY DOMAIN ASSUMPTION $\sqcup$ SOFT DOMAIN ASSUMPTION
12:	$\bot$	$\sqsubseteq$	FUNCTIONAL DOMAIN ASSUMPTION $\sqcap$ QUALITY DOMAIN ASSUMPTION $\sqcap$ SOFT DOMAIN ASSUMPTION
13:	SOFT DOMAIN ASSUMPTION	$\sqsubseteq$	$\exists$ **approximate**.QUALITY DOMAIN ASSUMPTION
14:	$\forall$ **refine**.EVALUATION	$\equiv$	COMPARATIVE EVALUATION $\sqcup$ INDIVIDUAL EVALUATION
15:	$\bot$	$\sqsubseteq$	COMPARATIVE EVALUATION $\sqcap$ INDIVIDUAL EVALUATION
		Taxonomy for WSLA	
16:	WSLA DOCUMENT	$\equiv$	PARTY $\sqcap$ SERVICE DEFINITION $\sqcap$ METRIC $\sqcap$ OBLIGATIONS
17:	WSLA DOCUMENT	$\equiv$	WSLA PROPOSAL $\sqcup$ WSLA AGREEMENT
18:	WSLA PROPOSAL	$\equiv$	$\exists$ **proposed-by**.(QoS LEVEL $\sqcap$ COMMITMENT)
19:	**propose**	$\equiv$	**proposed-by**$^-$
20:	**proposed-by**	$\equiv$	**propose**$^-$
21:	WSLA AGREEMENT	$\equiv$	QoS LEVEL $\sqcap$ COMMITMENT $\sqcap$ $\forall$ **agreed-by**.WS CONSUMER $\sqcap$ $\forall$ **agreed-by**.WS PROVIDER
22:	**agree**	$\equiv$	**agreed-by**$^-$
23:	**agreed-by**	$\equiv$	**agree**$^-$
24:	$\top$	$\equiv$	$\forall$ **proposed-by**.SIGNATORY PARTY $\sqcup$ $\forall$ **agreed-by**.SIGNATORY PARTY
25:	PARTY	$\equiv$	SIGNATORY PARTY $\sqcup$ THIRD PARTY
26:	PARTY	$\equiv$	$\forall$ **involved-in**.WS USE
27:	**involve**	$\equiv$	**involved-in**$^-$
28:	**involved-in**	$\equiv$	**involve**$^-$
29:	SIGNATORY PARTY	$\equiv$	WS CONSUMER $\sqcup$ WS PROVIDER
30:	THIRD PARTY	$\equiv$	$\neg$SIGNATORY PARTY $\sqcap \forall$ **provide**.METRIC
31:	**provide**	$\equiv$	**provided-by**$^-$
32:	**provided-by**	$\equiv$	**provide**$^-$
33:	SERVICE DEFINITION	$\equiv$	SERVICE OBJECT $\sqcap$ OPERATION
34:	SERVICE OBJECT	$\equiv$	SLA PARAMETER $\sqcap$ METRIC
35:	OPERATION	$\sqsupseteq$	SERVICE OBJECT
36:	*WSLA*:OPERATION	$\overset{\sqsubseteq}{\longrightarrow}$	*WSDL*:OPERATION
37:	METRIC	$\equiv$	$\forall$ **measure**.SLA PARAMETER
38:	**measure**	$\equiv$	**measured-by**$^-$
39:	**measured-by**	$\equiv$	**measure**$^-$
40:	OBLIGATIONS	$\equiv$	SERVICE LEVEL OBJECTIVE $\sqcup$ ACTION GUARANTEE
41:	SERVICE LEVEL OBJECTIVE	$\sqsubseteq$	COMMITMENT
42:	ACTION GUARANTEE	$\sqsubseteq$	PROMISE $\sqcap$ ACTION
		Taxonomy for WSDL	
43:	DESCRIPTION	$\equiv$	MESSAGE TYPES $\sqcap$ INTERFACE $\sqcap$ BINDING $\sqcap$ SERVICE
44:	DESCRIPTION	$\equiv$	WSDL PROPOSAL $\sqcup$ WSDL AGREEMENT
45:	WSDL PROPOSAL	$\equiv$	$\exists$ **proposed-by**.(OPERATION $\sqcap$ BINDING)
46:	**propose**	$\equiv$	**proposed-by**$^-$
47:	**proposed-by**	$\equiv$	**propose**$^-$
48:	WSDL AGREEMENT	$\equiv$	OPERATION $\sqcap$ BINDING $\sqcap$ $\forall$ **agreed-by**.WS CONSUMER $\sqcap$ $\forall$ **agreed-by**.WS PROVIDER
49:	**agree**	$\equiv$	**agreed-by**$^-$
50:	**agreed-by**	$\equiv$	**agree**$^-$
51:	$\top$	$\equiv$	$\forall$ **proposed-by**.WS ACTOR $\sqcup$ $\forall$ **agreed-by**.WS ACTOR
52:	WS ACTOR	$\equiv$	WS PROVIDER $\sqcup$ WS CONSUMER
53:	INTERFACE	$\sqsupseteq$	OPERATION
54:	OPERATION	$\sqsubseteq$	≥ 2 **order**.MESSAGE
55:	**order**	$\equiv$	**ordered-by**$^-$
56:	**ordered-by**	$\equiv$	**order**$^-$
57:	BINDING	$\equiv$	MESSAGE FORMAT $\sqcap$ COMMUNICATION PROTOCOL
58:	SERVICE	$\equiv$	WEB SERVICE ENDPOINT

A domain assumption indicates that its content is believed true by the service consumer, or that its content is made true by the service consumer's speech act. Domain assumption is only mapped with *Metric*: a WS consumer can express her representation of the description of an observable parameter that she believes true regardless of the actual state of affairs. She also has the capacity to structure and to organize herself the measurements of some observable parameters. On the other hand, Domain assumption is not mapped with *Obligations*, *Operation* and *Binding* respectively because (i) *action*

	WSLA concept		WSDL concept	
	Metric	*Obligations*	*Operation*	*Binding*
Goal	✔	✔	✔	✔
Plan	✔	✔	✔	✔
Domain assumption	✔	✗	✗	✗
Evaluation	✗	✔	✗	✗

guarantees can only be promised or desired by a party and *service level objectives* result from a negotiation so that a WS consumer is not expected to have beliefs about them, and she cannot make them true alone, (ii) it seems inappropriate to assume that a WS consumer would believe in particular messages sent by the WS provider without any information about them neither about the (future) WS provider and she cannot make the messages exchange pattern true alone, and (iii) a WS consumer dealing with the communication protocol or the message format is expected to have some basic knowledge about those kinds of technologies, and she cannot make them true alone; otherwise, he is expected not to worry about the way messages are formatted and sent.

An evaluation captures the preference, or the appraisal, of the WS consumer about a single condition, or between conditions that may hold. During the RE process, a WS consumer can express appraisals or preferences of/between goals, domain assumptions and plans which represent the conditions evaluated. Unfortunately, only appraisals and preferences about *obligations* can be specified through the WSLA/WSDL languages. The use of a monetary measurement tool allows the WS consumer to express his emotions and feelings in comparison with *service level objectives*. An *action guarantee* can be tied to the respect of one or more determined *service level objective(s)*. Through those *action guarantees*, *service level objectives* can be linked to financial penalties and rewards. A positive compensation reflects his favor toward a *service level objective*; a negative one reflects his disfavor. If the reward (penalty) of two *service level objectives* are different, then the WS consumer expresses a preference for one of them.

Some gaps in the WSLA/WSDL specifications in comparison with CORE have been highlighted when studying the Evaluation concept: some evaluations could be lost at the lower level of requirements representation. Because of the scope of this paper, we let the discussion of this issue for future work.

C. Mappings between CORE and WSLA/WSDL

Table III uses DDL to formalize the mapping between CORE and WSLA/WSDL. In the mappings, concepts are prefixed by the name of the taxonomy they belong to. The sign $\overset{\equiv}{\longrightarrow}$ means that the mapping is complete: each instance of the corresponding CORE concept can be translated in the WSLA and/or WSDL concepts. The sign $\overset{\sqsupseteq}{\longrightarrow}$ indicates that an evaluation can be lost because the scope of CORE is larger than the scope of WSLA/WSDL (see §III-B). We refine the mapping by comparing

the definition of the subclasses of the four main CORE concepts with the WSLA/WSDL concepts.

Table II indicates that Goal is bridged to all WSLA/WSDL concepts. Lines 59 and 60 from Table III specialize it.

Line 59: Functional goal is linked to *Metric*, *Action guarantee* and *Operation*. A *metric* specifies how the measurement of a QoS property is achieved. The WS consumer's desire concerning its presence or absence is not the representation of a quality. An *action guarantee* or an *operation* are the representation of a process to perform but not of a quality.

Line 60: Quality constraint is linked to *Service level objective* and *Binding*. Seeing that the observable parameters are described into a *metric*, the *Service level objective*'s quality space is common to the parties. The description of the communication protocol and the message format is a quality of the message structure and of the communication process. Its quality space is shared among the parties which can easily notice the use of one or another protocol/language.

Line 61 does not add any information compared with Table II because Plan has not subclasses in the CORE ontology.

Line 62: For the same reason as the refinement of the Goal concept, i.e., a *metric* is not the representation of a quality, Functional domain assumption is mapped to Metric.

Lines 63 and 64 refine the mapping between an evaluation and an *obligations*. The WS consumer could relate a *service level objective* to a reward and/or a penalty. This is made via *action guarantees* that the WS consumer pays or receives to/from the WS provider. She can also evaluate two or more SLOs if she gives different rewards/penalties to each of them.

There is no mapping link between the Quality domain assumption concept and an WSLA/WSDL concept. Since "[...] domain assumptions concern what is true [in the future IS and its environment]" [24], we expected to have only a few mapping links for this class. Our application domain — the WS use process and its environment — is specific because many characteristics are negotiable. The few non-negotiable elements mainly concern the *unreliable* network infrastructure.

IV. A TOOL BASED ON THE PROPOSED MAPPING: STR@WS

A. Technologies used

Our tool[3], named STR@WS for "Specifications Transcribed from Requirements" in a WS environment (hence the @WS in the name), is developed with the language Java O.O. We also use the JAXB API[4] which allows us to translate XML document into Java object as well as marshalling, unmarshalling and validating XML documents based on XSD or DTD documents.

B. The STR@WS components

STR@WS is compounded of four components:

1) **RequirementEditor** allows a WS consumer to add and remove requirements about a WS he is looking for.
2) **Translator** bridges the requirements expressed by the WS consumer with the WSLA/WSDL concepts based on

[3]The reader can request the tool by contacting one of the first two authors.
[4]https://jaxb.dev.java.net/

59:	*CORE*:FUNCTIONAL GOAL $\xrightarrow{\equiv}$	*WSLA*:METRIC ⊔ *WSLA*:ACTION GUARANTEE ⊔ *WSDL*:OPERATION
60:	*CORE*:QUALITY CONSTRAINT $\xrightarrow{\equiv}$	*WSLA*:SERVICE LEVEL OBJECTIVE ⊔ *WSDL*:BINDING
61:	*CORE*:PLAN $\xrightarrow{\equiv}$	*WSLA*:METRIC ⊔ *WSLA*:ACTION GUARANTEE ⊔ *WSDL*:OPERATION⊔ *WSDL*:BINDING
62:	*CORE*:FUNCTIONAL DOMAIN ASSUMPTION $\xrightarrow{\equiv}$	*WSLA*:METRIC
63:	*CORE*:INDIVIDUAL EVALUATION $\xrightarrow{\sqsupseteq}$	*WSLA*:OBLIGATIONS
64:	*CORE*:COMPARATIVE EVALUATION $\xrightarrow{\sqsupseteq}$	*WSLA*:OBLIGATIONS

the mapping between CORE and the WSLA/WSDL specifications. Regarding one-to-many relations (Lines 59, 60 and 61), we categorize the requirements expressed based on a syntactic matching between a knowledge base and the requirements content. Each word of the latter is compared with the items contained in the knowledge base corresponding to the possible concept (e.g., words of a quality constraint is compared with items corresponding to *Service level objective* and to *Binding*). The WS consumer can add items to the knowledge base. The knowledge base has already been fed with terms for each concept definition (e.g., we use [30] to add terms related to *Metric*).

3) **OpenFile** enables to open a specification file, i.e., an WSLA or an WSDL document, or a requirements file which has been saved with STR@WS.

4) **SaveFile** enables to save specification files or requirements files.

C. The use of STR@WS through an scenario

A entrepreneur owing an express transport company would like to optimize the routes of his trucks. Orders and clients data are centralized in his IS where the routes of each truck are calculated depending on urgent/deleted orders, truck breakdowns, delays, etc. He has equipped all trucks with a navigation system based on both the GPS and the UMTS technologies. He would like that his IS sends the data needed in real time to the trucks when the previous job is ending. To avoid waste of time, the devise can directly find the way with the coordinates (longitude and latitude) of the client. However, his IS only stores the clients' postal addresses.

The entrepreneur looks for a WS providing the coordinates (longitude and latitude) when it receives a postal address. He demands an answer within 600ms and preferably within 400ms. The service must be available 24/7 with maximum downtime of 10min. He agrees to pay $0.01 per use when a minimum QoS is satisfied. Otherwise, the WS use is free.

Fig. 1.A gives an insight into the tool menu.

Fig. 1.B illustrates how the requirements can be expressed by the WS consumer with STR@WS. In a textual and natural way, the WS consumer expresses his requirements concerning the WS he is looking for. We assume that the WS consumer cannot classify the requirement: *Free service if the* QoS *is not satisfied* of the transport company's case. This requirement is so classified in a "raw category". It is then possibly classified in the right WSLA/WSDL concept thanks to the knowledge

base center where it is compared to each WSLA/WSDL concept. Without convincing result, the requirement is classified in a "unknown" category.

Fig. 1.C shows the result of the matching between the requirements expressed in the CORE ontology and the main concepts of the WSLA and WSDL specifications[5]. It could be used by a system-to-be able to compare and select WSs.

Fig. 1.D shows an extract of an WSLA proposal corresponding to the requirement "Answer within 400ms" linked to $<SLO>Answer\ within\ 400ms</SLO>$.

V. RELATED WORK

The use of textual requirements communicated by the WS consumer has been tackled several times in the literature.

Two tools [31], [32] and an innovative method [33] have been proposed in order to ease the WS discovery process. Based on textual requirements, WSs matching the WS consumer needs are suggested. However, these works exclusively focus on functional requirements and the requirements are expressed without any RE structure. That makes the discovery task more demanding in methods for extracting accurate information.

Rolland et al. [34] introduce a model for Intentional Service Modelling (ISM): WS providers have to describe their WSs in an intentional way and WS consumers use an intentional matching mechanism to select potential WSs. This model requires new technologies for publishing, browsing and discovering services in comparison to the most widespread ones, i.e., UDDI and ebXML registries. The QoS characteristics of WSs are not considered in the discussion.

Regarding the solutions of semantic matching between the WS descriptions and the needs of the WS consumer, related work is often built on technical languages and specifications. For instance, [35], [36] and [37] respectively use USQL (Universal Service Query Language), DAML-S and BPOL (Business Process Outsourcing Language). The handling of those technologies requires thorough knowledge of each of them. Works on semantic matching often concentrate on the WS provider side, e.g., [38]–[41]. In order to have a complete approach of the problem, we first need a user-friendly solution that eases the requirements elicitation task at the WS consumer side.

The work of Zachos et al. [42] shares some similarities with ours. They create a tool which is able to discover WSs based

---

[5]The meaning of the tags used to show the output is as following: $<METRIC/>$ for *metrics*, $<AG/>$ for *action guarantees*, $<OP/>$ for *operations*, $<SLO/>$ for *service level objectives*, $<BIND/>$ for *bindings*, $<OBLIG/>$ for *obligations* and $<UNKW/>$ for *unlinked requirements*.

A Str@ωs

Menu Help

Open XML Requirements File
Open XML WS Specification File

Save Requirements to XML File
Save WS Specification to XML File

Transform to XML WS Specification File
Options

Quit

Parameters
Clear
WS Transform   Clear All Requirements

B Str@ωs

Menu Help

Category	Parameters	
Functional Goal	receive coordinates	Clear
Plan	Provide postal address	Clear
Quality Constraint	Answer within 600ms	Clear
Quality Constraint	Answer within 400ms	Clear
Quality Constraint	Available all the time (24 hours a day, 7 days a week)	Clear
Quality Constraint	Maximum downtime of 10min	Clear
Plan	Pay per use of $0.01 when the QoS is satisfied	Clear
Comparative Evaluation	Prefer an answer within 400ms to 600ms	Clear
RAW	Free service if the QoS is not satisfied	Clear

Add Another Specification   WS Transform   Clear All Specifications

C Str@ωs

```
<OP>receive coordinates</OP>
<OP>Provide postal address</OP>
<AG>Pay per use of $0.01 when the QoS is satisfied</AG>
<SLO>Answer within 600ms</SLO>
<SLO>Answer within 400ms</SLO>
<SLO>Available all the time (24 hours a day, 7 days a week)</SLO>
<SLO>Maximum downtime of 10min</SLO>
<OBLIG>Prefer an answer within 400ms to 600ms</OGLIB>
<AG>Free service if the QoS is not satisfied</AG>
```

D

```xml
<SLA>
  <Parties> ... </Parties>
  <ServiceDefinition>
    <Operation>
      <SLAParameter name="ResponseTime" type="long" unit="milliseconds">
        <Metric>ResponseTimeComputing</Metric>
      </SLAParameter>
      ....
    </Operation>
  </ServiceDefinition>
  <Obligations>
    <ServiceLevelObjective name="ResponseTime" serviceObject="WSDLSOAPGetQuote">
      <Obliged>provider</Obliged>
      <Validity>
        <Start>xsd:dateTime</Start>
        <End>xsd:dateTime</End>
      </Validity>
      <Expression>
        <Predicate xsi:type="wsla:Less">
          <SLAParameter>ResponseTime</SLAParameter>
          <Value>400</Value>
        </Predicate>
      </Expression>
      <EvaluationEvent>NewValue</EvaluationEvent>
    </ServiceLevelObjective>
    .....
  </Obligations>
</SLA>
```
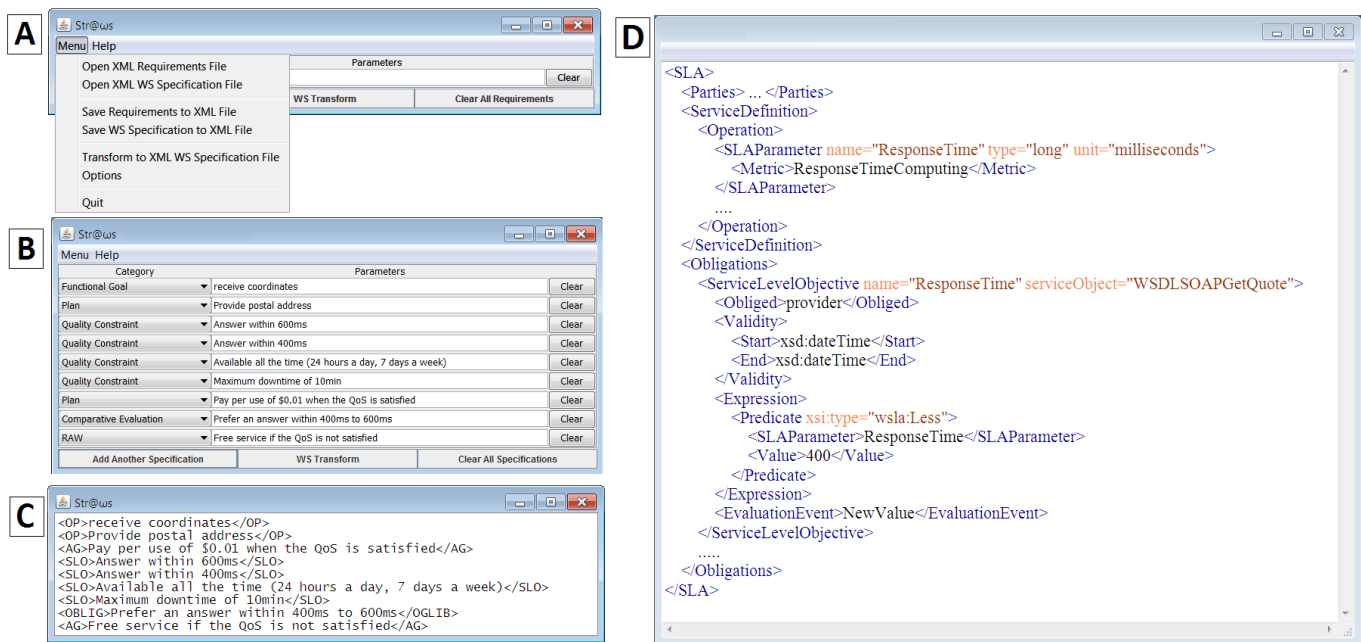
Fig. 1.   Illustrations of STR@WS

on requirements expressed by the user in natural language. The requirements elicitation process depends on use-case analysis. Requirements related to the use-cases are then added in the system, UCaRE, which follows the VOLERE requirements shell. The scope of our work is more restricted than theirs: we focus exclusively on the mapping between the requirements of the WS consumer and WSLA/WSDL. Our contribution lies in the use of CORE, which covers main classes of requirements, compared to use-cases. Moreover, we formalize the mapping between the requirements, which could be expressed in natural language, and their specifications. First, it will allow to keep the track of requirements when a WS is selected. If the system-to-be selecting WSs cannot replace a defective WS, it is able to identify too demanding requirements by comparing the characteristics of the best fitted WS and the consumer requirements contained in the service request. Secondly, it enables to directly analyze the consequences of requirements changes in comparison with the (composite) WS chosen. This is very significant for requirements monitoring in an SOA, as already noted in [43]. With regards to works related to RE monitoring in a service-oriented environment [43], [44], proposed methods to elicit requirements are based on RE techniques. Our contribution could be complementary to those works in order to improve the RE.

VI. CONCLUSION

The dynamic environment of the service oriented computing raises new issues. Authors often work with technical specifications as the requirement of the WS consumer. Adding a clear link between a core ontology for requirements and the WSLA/WSDL specifications allows (i) to move closer to automated creation of WSDL and WSLA documents based on requirements, (ii) to help the WS composition system to identify easily non-suitable requirements asked by the WS consumer, (iii) to know which requirements are no longer satisfied when a WS provider fails to comply with the agreement and (iv) to know precisely which part of an WSLA and/or WSDL document must be modified when the WS consumer changes some of his requirements. Creating and keeping this link in the IS is permitted by the developed mapping between the two levels of requirements representation. The original idea is to base the high level representation from an ontology for RE and translate it to WS descriptions.

This work paves the way for an abstract mapping without any references to precise service specifications. For that, a technological independent ontology concerning the technical characteristics (functional and non-functional) of a WS is needed. Concerning the RE side, a RE methodology must be created or adapted to the service oriented paradigm in order to capture the requirements WS consumers. It could be grounded on Techne [45].

This paper does not cover the difference between hard and soft SLOs. WS consumers often express their minimal requirements regarding the non-functional characteristics of the WS as well as additional (soft) SLOs increasing their satisfaction. It also avoid the issue of requirements concerning orchestration and choreography. Before tackling this question, RE for single WS should be done more suitably.

Taking into account the gaps (see §III-B) between the two levels of requirements representation is also a future task. This can be done within a wider IS composed of our tool as well as other computational modules enabling the composition of WSs based on the WSLA/WSDL specifications.

REFERENCES

[1] D. L. Tennenhouse, "Proactive computing," *Commun. ACM*, vol. 43, no. 5, pp. 43–50, 2000.

[2] M. P. Papazoglou and D. Georgakopoulos, "Introduction," *Commun. ACM*, vol. 46, no. 10, pp. 24–28, 2003.

[3] S. A. McIlraith and D. L. Martin, "Bringing semantics to web services," *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 90–93, 2003.

[4] F. Casati, M.-C. Shan, and D. Georgakopoulos, "Guest editorial," *VLDB Journal*, vol. 10, no. 1, pp. 1–1, 2001.

[5] W3C, "Simple Object Access Protocol (SOAP)," 2003. [Online]. Available: http://www.w3.org/TR/soap

[6] OASIS, "Universal Description, Discovery, and Integration (UDDI)," 2003. [Online]. Available: http://www.uddi.org

[7] E. Christensen, F. Curbera, G. Meredith, and S.Weerawarana, "Web Services Description Language (WSDL 1.1)," 2001. [Online]. Available: http://www.w3.org/TR/wsdl

[8] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara, *OWL-S: Semantic Markup for Web Services 1.1*. DAML Services Coalition, 2004.

[9] D. Berardi, M. Gruninger, R. Hull, and S. McIlraith, "Towards a First-Order Ontology for Semantic Web Services," in *Proceedings of the W3C Workshop on Constraints and Capabilities for Web Services*, 2005.

[10] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, and S. Tabet, "Semantic Web Services Framework (SWSF)," 2005. [Online]. Available: http://www.w3.org/Submission/SWSF/

[11] B. Benatallah, M.-S. Hacid, A. Léger, C. Rey, and F. Toumani, "On automating web services discovery," *VLDB Journal*, vol. 14, no. 1, pp. 84–96, 2005.

[12] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web services," in *Proceedings of the International Conference on the World Wide Web (WWW 2002)*, 2002, pp. 77–88.

[13] S. A. McIlraith and T. C. Son, "Adapting Golog for Composition of Semantic Web Services," in *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR'02)*, 2002, pp. 482–496.

[14] B. Medjahed, A. Bougettaya, and A. K. Elmagarmid, "Composing web services on the Semantic Web," *VLDB Journal*, vol. 12, no. 4, pp. 333–351, 2003.

[15] I. J. Jureta, S. Faulkner, Y. Achbany, and M. Saerens, "Dynamic Web Service Composition within a Service-Oriented Architecture," in *Proceedings of the International Conference on Web Services (ICWS'07)*, 2007, pp. 304–311.

[16] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services." *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[17] I. Horrocks, "DAML+OIL: a Description Logic for the Semantic Web," *IEEE Data Engineering Bulletin*, vol. 25, no. 1, pp. 4–9, 2002.

[18] G. Antoniou and F. van Harmelen, "Web Ontology Language: OWL." in *Handbook on Ontologies*, ser. International Handbooks on Information Systems, S. Staab and R. Studer, Eds. Springer, 2004, pp. 67–92.

[19] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuiness, P. F. Patel-Schneider, and L. A. Stein, *OWL Web Ontology Language*. World Wide Web Consortium, 2004.

[20] S. Staab and R. Studer, Eds., *Handbook on Ontologies*, ser. International Handbooks on Information Systems. Springer, 2004.

[21] I. J. Jureta, J. Mylopoulos, and S. Faulkner, "A Core Ontology for Requirements," *Applied Ontology*, vol. 4, no. 3-4, pp. 169–244, 2009.

[22] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," World Wide Web Consortium, W3C Recommendation, June 2007. [Online]. Available: http://www.w3.org/TR/wsdl20/

[23] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification," IBM Corporation, Tech. Rep., 2003. [Online]. Available: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[24] I. Jureta, J. Mylopoulos, and S. Faulkner, "Revisiting the Core Ontology and Problem in Requirements Engineering," in *16th IEEE International Requirements Engineering Conference (RE 2008)*, 2008, pp. 71–80.

[25] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *Journal of Network and Systems Management*, vol. 11, no. 1, 2003.

[26] D. Booth and C. K. Liu, "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer," World Wide Web Consortium, W3C Recommendation, June 2007. [Online]. Available: http://www.w3.org/TR/wsdl20-primer/

[27] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[28] R. Chinnici, H. Haas, A. A. Lewis, J.-J. Moreau, D. Orchard, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts," World Wide Web Consortium, W3C Recommendation, June 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626

[29] A. Borgida and L. Serafini, "Distributed Description Logics: Assimilating Information from Peer Sources," *Journal on Data Semantics*, vol. 1, pp. 153–184, 2003.

[30] A. Paschke and E. Schnappinger-Gerull, "A Categorization Scheme for SLA Metrics," in *Service Oriented Electronic Commerce*, vol. 80, 2006, pp. 25–40.

[31] Y. Hao, Y. Zhang, and J. Cao, "WSXplorer: Searching for Desired Web Services," in *The 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)*, 2007, pp. 173–187.

[32] X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, 2004, pp. 372–383.

[33] Y. Hao, J. Cao, and Y. Zhang, "Efficient IR-Style Search over Web Services," in *The 21st International Conference on Advanced Information Systems Engineering (CAiSE 2009)*, 2009, pp. 305–318.

[34] C. Rolland, R. S. Kaabi, and N. Kraïem, "On ISOA: Intentional Services Oriented Architecture," in *The 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)*, 2007, pp. 158–172.

[35] L. Baresi, M. Miraz, and P. Plebani, "A Flexible and Semantic-Aware Publication Infrastructure for Web Services," in *The 20th International Conference on Advanced Information Systems Engineering (CAiSE 2008)*, 2008, pp. 435–449.

[36] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara, "Semantic Matching of Web Services Capabilities," in *The First International Semantic Web Conference*, 2002, pp. 333–347.

[37] L.-J. Zhang and B. Li, "Requirements Driven Dynamic Services Composition for Web Services and Grid Solutions," *Journal of Grid Computing*, vol. 2, no. 2, pp. 121–140, 2004.

[38] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web services," *Journal of Information Technology and Management*, vol. 6, no. 1, pp. 17–35, 2005.

[39] T. Pilioura, G.-D. Kapos, and A. Tsalgatidou, "PYRAMID-S: A Scalable Infrastructure for Semantic Web Service Publication and Discovery," in *14th International Workshop on Research Issues in Data Engineering: Web Services for e-Commerce and e-Government Applications*, 2004, pp. 15–22.

[40] T. Pilioura and A. Tsalgatidou, "Unified publication and discovery of semantic Web services," *ACM Transactions on the Web*, vol. 3, no. 3, pp. 1–44, 2009.

[41] F. Colasuonno, S. Coppi, A. Ragone, and L. L. Scorcia, "jUDDI+: A SemanticWeb Services Registry enabling Semantic Discovery and Composition," in *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, 2006, p. 69.

[42] K. Zachos, N. A. M. Maiden, X. Zhu, and S. Jones, "Discovering Web Services to Specify More Complete System Requirements," in *The 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)*, 2007, pp. 142–157.

[43] W. N. Robinson, "Monitoring Web Service Requirements," in *11th IEEE International Conference on Requirements Engineering (RE 2003)*, 2003, pp. 65–74.

[44] K. Mahbub and G. Spanoudakis, "A Framework for Requirements Monitoring of Service Based Systems," in *Second International Conference on Service-Oriented Computing (ICSOC 2004)*, 2004, pp. 84–93.

[45] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling," in *18th IEEE International Conference on Requirements Engineering (RE 2010)*, 2010, to be published.