Specifying Web Service Compositions on the Basis of Natural Language Requests

Alessio Bosca[1], Giuseppe Valetto[2], Roberta Maglione[2], and Fulvio Corno[1]

[1] Politecnico di Torino, Torino, Italy
{alessio.bosca, fulvio.corno}@polito.it
[2] Telecom Italia Lab, Torino, Italy
{roberta.maglione, giuseppe.valetto}@tilab.com

Abstract. The introduction of the Semantic Web techniques in Service-oriented Architectures enables explicit representation and reasoning about semantically rich descriptions of service operations. Those techniques hold promise for the automated discovery, selection, composition and binding of services. This paper describes an approach to derive formal specifications of Web Service compositions on the basis of the interpretation of informal user requests expressed in (controlled) Natural Language. Our approach leverages the semantic and ontological description of a portfolio of known service operations (called Semantic Service Catalog).

1 Introduction

The recent introduction of Semantic Web [1] ideas and results in the field of service-oriented computing has originated a vision of *Semantic Web Services* [6, 7], founded on machine understandability of the nature of operations made available as Web Services. The linguistic and ontological means for representing the properties and the capabilities of Web Services, and thus enhancing the ability to reason about the tasks they perform, seem particularly appealing for the support, based on operation semantics, of highly dynamic service selection and composition, which is an important goal of the Web Service paradigm. A major outstanding challenge to reach that goal is how to map the requirements describing a complex, composite service-oriented application (sometimes called a Value-Added Service, or VAS) to a multiplicity of simple, atomic Web Service operations, as well as to an overall service logic that coordinates their interactions.

We present an approach for the automatic generation of a high-level VAS specification (*Abstract Composition* in the remainder) on user demand, that is, starting from informal user requests. Our approach leverages semantic information about the operations exposed by a portfolio of Web Services and targets simple requests that can be expressed in (restricted) Natural Language, covering a range of workflows that can be modeled according to a set of modular *logic templates*. The Abstract Composition generated from the interpretation of a user request can be translated into an executable flow, and maps the user needs and intentions – as inferred from the original request - to known Web Service operations that can satisfy them, in a "task-oriented" way [5].

We employ OWL-S annotations to provide a formal representation of service and operation semantics, as well as a classification of the Web Services in the portfolio.

2 Approach Overview

Our approach for the specification of service compositions at run-time has two starting points: the user request, which is processed and interpreted on the fly, to elicit functional requirements as well as a high-level view of the composition logic implied in the request; and a repertoire of well-known services that are described by rich semantic meta-data. Those two elements are, respectively, the *Request Interpreter* and the *Service Catalog,* displayed in Figure 1 together with other elements of our prototype.

This technique follows from two major assumptions: user requests are relatively simple and concise, in structure and terminology, to be expressed with a controlled subset of natural language; furthermore, a common ontological vocabulary can be established, and is consistently applied to all entries in the Service Catalog. While the latter assumption is unfeasible in a context, in which Web Services over the Internet at large and owned by multiple parties should be summoned in response to the user request, it seems reasonable and manageable in the context of a limited set of Web Services that are kept under the control of a single entity, like in the case of a provider or operator that offers value-added services to its customer base.

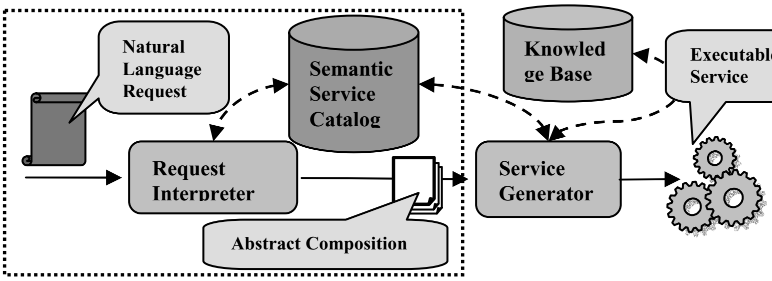

Fig. 1. System overview

Besides being used for the annotation of Web Services included in the Catalog, we exploit OWL-S also in the Request Interpreter, to support NLP techniques and our approach also includes mechanisms to transform that abstract service specification into a concrete one: the *Service Generator*. Although, functionally speaking, the role of the Service Generator – as shown in figure 1 – is simply to translate Abstract Compositions into a notation that can be executed over a service-oriented runtime of choice, its task is multifold and its structure complex, therefore a complete discussion of our solution, detailing its internal architecture, mechanisms and algorithms is not feasible here due to space limitations, and is outside the scope of this paper.

3 A Semantic Service Catalog in OWL-S

OWL-S is a framework to describe services from several perspectives: more precisely it characterizes services through a set of sub-ontologies. We have recognized the need for models and algorithms to select services on the basis of semantic annotations stored not only in their profile (as proposed for example in [2]), but also in their IO-PEs (see [3, 4]). We also propose to exploit IOPEs as a means to drive composition.

In order to enable the selection of service operations that satisfy some user requests or needs our modeling approach promotes the description of Effects in terms of the computing task that is performed by each atomic operation exposed by each service in our Catalog. To this end, we have implemented an *ad hoc ontology* called **Effects** (see bottom of Fig. 2). Additionally, we focus on I/O parameters semantics referring to a set of concepts collected in another *ad hoc ontology* called **IOtypes**. In order to reason on inputs and outputs for the automatic, semantic-based composition of operations, we extended the OWL-S model with a couple of bi-directional properties that allow us to link processes to their I/O parameters and parameter to processes that can produce or consume them (see Fig. 2).

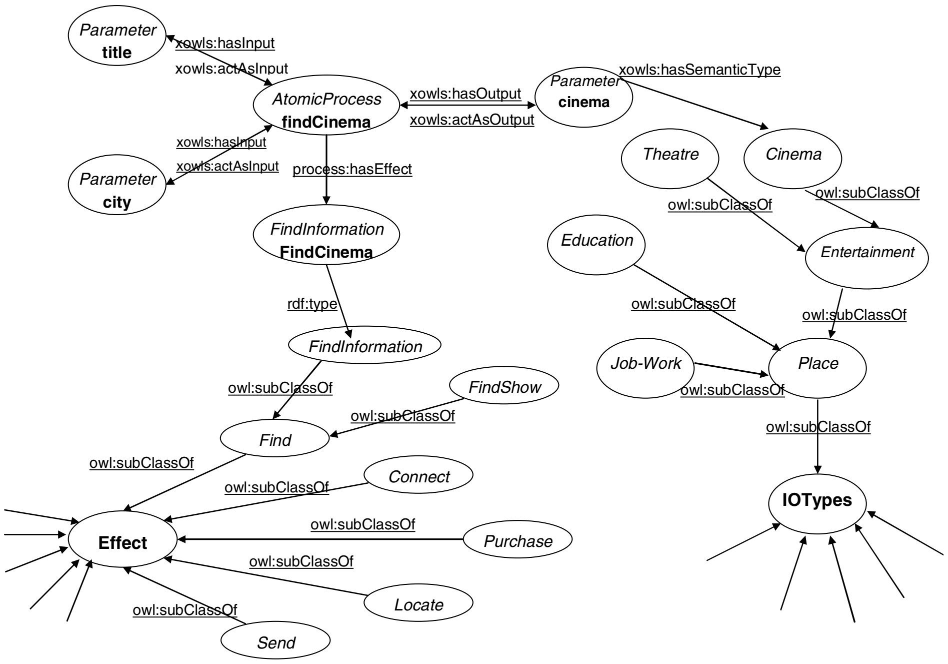

Fig. 2. Atomic Process: findCinema

4 Request Interpreter

Starting from a user request expressed in Natural Language (NL) the Request Interpreter is in charge of decomposing the sentence in order to isolate expressions that can be semantically associated to Effects listed in the Semantic Service Catalog, and hence mapped onto specific service functionality provided by atomic operations. At that end, the Request Interpreter translates the NL request into an Abstract Composition document, that is, a *formal* VAS specification. That specification includes a lattice of logic templates, describing how they relate to and can be composed with each other into the global VAS flow; moreover, it includes a list of Effects, which act as generic, semantic placeholders for operations that must be invoked along that flow.

Rather than completely parsing the sentence and interpreting each single fragment, we chose a simpler algorithmic approach that in a first step decomposes the request into fragments according to a proper logic template (if then, while do, sequence..). Then it leverages the dictionary in order to search for lexical patterns within the fragments and consequently infer the user's intention and eligible parameters.

4.1 An Instrument for Request Interpretation: The SSC Dictionary

As stated before, the dictionary contains lexical elements related to some entities within OWL-S ontology and includes pure lexical resources (as lists of verbs or preposition grouped by their role or meaning), as well as more complex ones, related to the sentence structure and its verbal governance (the *Sentence Constructions List* and the *Recognizer Catalog*).

The *Sentence Constructions List* is the main resource within the dictionary and it models the distinct expressive ways through which it is possible to request a service identified by a given *Effect* concept (see Fig. 2 for the relation between *AtomicProcesses* and *Effects*). For each *Effect* present in the SSC a set of thematic keywords and a list of eligible constructions are reported, each construction specifying a group of verbs and a set of parameters.

The *Recognizer Catalog* is the other key resource within the dictionary and models the different information the system should be able to recognize as potential parameters. It focuses on the different *IOTypes* present in the SSC and specifies for each of them a set of features which enable the isolation and recognition of a given *IOType* within a free text. Such features both concern how the data appears and which value it holds; the recognition process in fact relies on data format, on the presence of a keyword or on the candidate parameter's occurrence within a given list.

4.2 The Request Interpretation Process

This section details the various phases of the interpretation process and describes how it exploits the lexical resources within the dictionary.

The first operative step consists in recognizing the logic flow behind the request and coupling it to one of the logic templates supported in the system (if then, if then else, while do, sequence). A set of parsers properly tailored to the aforementioned templates process the request by trying to validate it against their own sentence model and if it matches, extract the distinct sentence blocks tagging them as conditions or actions. The result of this phase thus consists in the identification of the logical template and of the distinct clauses.

After this parsing procedure, we assume that different propositions have been identified and that each sentence block is constituted by only one clause with a principal verb and a set of objects. The following steps (2, 3 in Fig. 3) consist then in the interpretation of any individual action or condition retrieved in the precedent phase.

The presence within the clause of a thematic keyword (recorded in the dictionary) provides hints about the user's intention and focuses the algorithm's attention on a set of services, considered as potential solutions and therefore inserted into a list of eligible *Effects*. By taking in exam the advices contained in the *Sentence Constructions List,* the system guesses the information to look for and if a suitable verbal form is found, a search is triggered over the sentence block for values that fit the parameters reported in the dictionary. A proper recognizer is thus tuned according both to the functional features reported in the parameters' description (as the introductive prepositions) and to the semantic ones reported in the correspondent *IOType* element of the dictionary, and it accordingly tries to identify a block of text as eligible information.

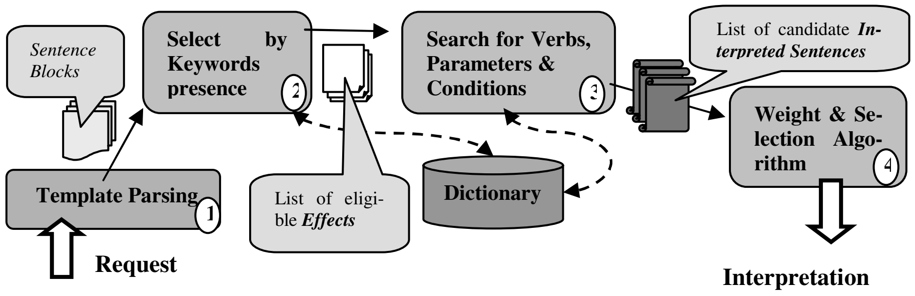
Sentence Blocks
Select by Keywords presence 2
Search for Verbs, Parameters & Conditions 3
List of candidate Interpreted Sentences
Weight & Selection Algorithm 4
Template Parsing 1
List of eligible Effects
Dictionary
Request
Interpretation

Fig. 3. Request Interpretation

If the functional information about where we expect to find the parameter and its "appearance" (the format of the data) are both verified as well as the semantic countercheck (the presence of a proper keyword in the same sentence chunk or the occurrence of the data in a list of known values), then a strong "found" is triggered, otherwise, a weak one. Once all eligible parameter values are found, the textual fragments identified as weak founds are evaluated in order to be promoted or rejected.

The analysis of each sentence block generates an *Interpreted Sentence* reporting the *Effect* id, the verbal form found, the list of the recognized parameters and the conditional expression, if present. These *Interpreted Sentences* are gathered into a list of candidate solutions and processed by a selection algorithm that constitutes the final step (phase 4) of the Request Interpretation. The algorithm works under the hypothesis that an interpretation holding more information should constitute a better solution, thus it simply assigns a score to the *Interpreted Sentences* for each element found (verbs, parameters, conditions) and then selects the ones with the highest rank.

The structural information concerning the logical flow of the request, retrieved from the template parsing (phase 1), and the distinct *Interpreted Sentences*, obtained in the followings (phases 2-4), are then unified into an *AbstractComposition* document.

5 Conclusions

We presented an approach that allows the specification of Web Services Compositions starting from user requests expressed in Natural Language. This paper shows how, under the assumptions stated in Section 2, it is possible to establish a synergy between the semantic service descriptions and the interpretation of user requests through a common ontology and consistent vocabulary. We have presently deployed a prototypal version of the system, provided with a semantic Service Catalog in OWL-S, comprising several tenths entries, and with a limited set of logic templates able to capture a range of simple workflow constructs. Our preliminary experiments with the system are encouraging, since they already enable to express and synthesize significant service compositions on demand.

We are currently working to expand the Service Catalog with a wealth of information, communication and e-commerce services in order to constitute a wider source of information, thus increasing the stress and the overall noise in the recognizing and selection procedures. At the same time, we are developing a test set of user requests that focus on our SSC servicing scope, in order to establish a validation resource for proving and tuning the algorithm.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American*, 2001, 284(5): 34–43.
2. D. Mandell, and S. McIlraith, "Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation", *Proceedings of the Second International Semantic Web Conference*, 2003.
3. K. Sivashanmugam, K. Verna, A. Sheth and J. Miller, Adding Semantics to Web Services Standards, *Proceedings of the International Conference on Web Services*, 2003.
4. E. Sirin, B. Parsia, and J. Hendler, Composition-driven filtering and selection of semantic web services, *AAAI Spring Symposium on Semantic Web Services*, 2004.
5. Y. Ye and G. Fisher. Supporting Reuse by Delivering Task-Relevant and Personalized Information, *Proceedings of the 24th International Conference on Software Engineering*, 2002.
6. DAML-S, "Semantic Wes Services", http://www.daml.org/services/.
7. WSMO, "Web Services Modeling Ontology", http://www.wsmo.org/.