Web Semantics: Science, Services and Agents on the World Wide Web 9 (2011) 16–28

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Humans, semantic services and similarity: A user study of semantic Web services matching and composition

Eran Toch [a,*], Iris Reinhartz-Berger [b], Dov Dori [c]

[a] Carnegie Mellon University, ISR, 5000 Forbes, Pittsburgh, PA 15213, USA
[b] University of Haifa, Haifa, Israel
[c] Technion - Israel Institute of Technology, Haifa, Israel

ARTICLE INFO

Article history:
Received 25 November 2009
Received in revised form 31 July 2010
Accepted 18 October 2010
Available online 10 November 2010

Keywords:
Semantic Web services
User study
Similarity
Approximation

ABSTRACT

Inferring similarity between Web services is a fundamental construct for service matching and composition. However, there is little evidence of how humans perceive similarity between services, a crucial knowledge for designing usable and practical service matching and composition algorithms. In this study we have experimented with 127 users to define and evaluate a model for service similarity in the context of semantic Web services. Our findings show that humans take a complex and sophisticated approach towards service similarity, which is more fine-grained than suggested by theoretical models of service similarity, such as logic-based approaches. We define a similarity model, based on our empirical findings and prove that the similarity model, expressed by a distance metric, is complete and that it closely predicts humans' perceptions of service similarity. Finally, we describe an application of a Web service search engine that implements our model.

1. Introduction

Service orientation is an emerging software engineering paradigm that emphasizes the reuse of existing and distributed software services. Two promising technologies in service orientation are service matching, which facilitates discovery of services on the Web [1–3], and service composition, which aims to assemble services into new applications [2–8]. These technologies, henceforth collectively referred to as service retrieval, promise to help users find and reuse new services, providing an agile and trustworthy environment for executing services and creating new applications.

A central aspect of service matching and composition is finding a good notion of similarity between services. The way in which similarity is defined is crucial to determining how services match a query and how they can be composed. Good similarity measures exist in many fields, including information retrieval, databases and image recognition, but they are far less developed in the field of semantic service retrieval. Particularly, to the best of our knowledge, there are no studies that examined how human subjects perceive service similarity in the context of semantic service retrieval.

As a motivating example, which runs throughout the paper, consider the following query: *Find a service that accepts an address as input and returns the closest hospital.* Even if the service repository includes only services that accept a longitude/latitude input parameter and return a hospital, software engineers would build a transformation that maps the longitude/latitude parameter to the address parameter. However, a large class of service retrieval algorithms, namely logic-based approaches, would fail to approximate an address with a longitude/latitude parameter unless they are conveniently related through a subsumption relation. Works such as OWLS-UDDI [9], Inter-OWL-S [4], OWLS-MX (Logic) [10], and SAM [8], are based on this underlying assumption. While logic-based matching is theoretically sound, its notion of similarity differs from the way humans perceive service similarity. This gap may reduce the usability of service matching, resulting in applications that are unintuitive to users and developers.

In this work we study how human developers perceive service similarity. A questionnaire administered to 127 information systems engineering students was used to assess the similarity of queries and service compositions by providing quantitative and qualitative feedback. Our findings show that human subjects perceive similarity in a more fine-grained way than predicted by logic-based service matching approaches. Humans' notion of similarity is based on broader constructs and on a higher level of approximation than the level predicted by logic-based service matching. Similarity is based on a broader set of semantic relations (e.g., object properties) and behavioral aspects (i.e., the structure of the composition). At the same time, human subjects exhibited a

* Corresponding author. Tel.: +1 412 499 3726.
E-mail addresses: eran@cs.cmu.edu (E. Toch), iris@mis.haifa.ac.il (I. Reinhartz-Berger), dori@ie.technion.ac.il (D. Dori).

1570-8268/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2010.10.002

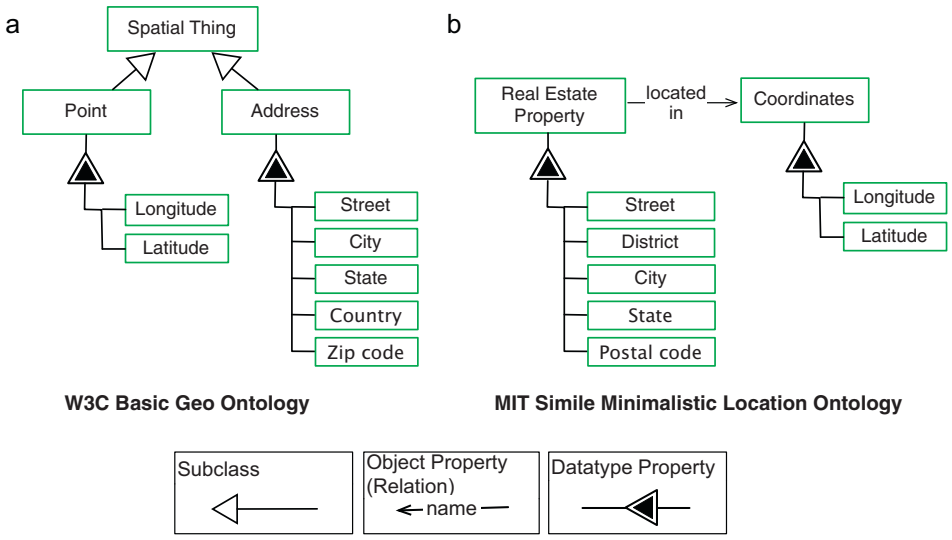

Fig. 1. Location specification using two different ontologies. Logic-based approaches would return the correct result when the service is specified using ontology (a), but would fail to do so when the service is specified using ontology (b).

softer and approximate notion of similarity, which was based on the evaluation of the service reusability considering the task at hand.

We used the quantitative findings to design an approximate service similarity mechanism, in which we integrated conceptual and behavioral approximations to capture people's notion of similarity. The model is based on a straightforward idea of estimating the number and depth of changes required to adapt the retrieved service to the query. Similarity is hence measured as the edit distance between the original service and the estimation. Quite surprisingly perhaps, we found this method to be sufficient for describing semantic and behavioral aspects of similarity via a model that is both formal and intuitive. We prove two important properties of the proposed similarity measure: (1) the similarity is based on a distance metric that can simplify service retrieval applications, and (2) the similarity measure is complete, i.e., it covers all the possible ontology-based compositions.

In summary, the contributions of this work are the following.

- Describing human perceptions of semantic service similarity,
- providing a complete formal metric for service similarity that is meaningful and intuitive to human developers, and
- combining semantic and behavioral approximation in a single frame of reference.

The rest of the paper is structured as follows. Section 2 reviews current approaches to service matching. Section 3 defines our theoretical framework of similarity, while Section 4 describes the similarity patterns. Section 5 elaborates on the empirical evaluation of our approach. Section 6 describes an implementation that is based on findings of this research. Section 7, which concludes the paper, refers to future research directions.

2. Current approaches to service retrieval

In this section, we review the literature pertaining to our study and two related approaches in service retrieval: logic-based approaches and hybrid approaches.

2.1. Semantic Web services

The underlying model of our research is semantic Web services, which aim to resolve the heterogeneity and lack of semantics in Web service specifications [11]. Languages such as OWL-S [12] describe services unambiguously by providing meta-data descriptions for Web services, including mapping service properties (e.g., input and output parameters) to common concepts. The concepts are defined in ontologies [13] on the Semantic Web [14] using the Web Ontology Language (OWL) [15]. Web ontologies serve as the key mechanism to globally define and reference common understanding in a Web-based distributed environment.

Semantic Web services in general and OWL-S specification in particular are used throughout this study, as they provide a clear formal model for service description, and there exists a considerable body of work for reference and comparison. While interesting results were available for retrieval of non-Semantic Web services [16], they lack formal semantics, making them insufficient for automated service composition. Therefore, we limit the scope of this article to Semantic Web services.

2.2. Logic-based approaches

Logic-based approaches in service retrieval are based on reasoning over the semantic descriptions of Web services to match queries and service properties. Service matchmakers, such as OWLS-UDDI [9], Inter-OWL-S [4], Bae et al. [2], and SAM [8], match service advertisements with queries by inferring the relations between their underlying concepts classes [17].

Zaremski and Wing defined a widely-used classification of matching degrees in logic-based software component matching [18]. We assume a set of concepts related to a given query $C_Q = \{C_1, C_2, \ldots, C_n\}$, and a set of properties related to a given advertised service, $C_S = \{C'_1, C'_2, \ldots, C'_m\}$. The categories include four levels of matching:

- **exact**, which represents perfect semantic identity of the query properties (e.g., input) with the service advertisement properties, denoted as $C_Q \equiv C_S$,
- **plugin**, in which properties of the service advertisement are fully contained in the concepts of the query, such that the query has at least one concept which is more general than a corresponding property of the service ($C_S \sqsubseteq C_Q$),
- **subsumes**, in which concepts of the query are fully contained in the properties of the service advertisement, such that some of the service properties may not fully answer the query in the sense of set-theory ($C_S \sqsupseteq C_Q$), and

- **disjoint**, in which none of the properties of the service can be related to any concept of the query ($C_S \sqcap C_Q = \varnothing$).

Logic-based methods have two main drawbacks: limited approximation and the subsumption assumption. Approximate results depend on whether the concepts are related through some set-hierarchy relations. This is not always the case in many realistic scenarios, and as a result, logic-based methods are prone to low recall. Fig. 1 exemplifies this problem using two ontologies that express geographical entities: ontology (a) depicts the WC3 Basic-Geo ontology,[1] in which a Spatial Thing concept represents a superclass for both longitude/latitude pair and an address. Ontology (b) depicts the MIT Simile project ontology,[2] in which the exact same data structure is modeled by having the longitude/latitude as an object property (a relation) of a real-estate concept (an identical concept of address). While the two ontologies express the same intention, logic-based methods would be able to retrieve services that have Address parameters to queries that need a longitude/latitude parameter using ontology (a), but would not be able to infer similarity based on ontology (b).

The second drawback of logic-based approaches is rooted at the subsumption assumption, which dictates that subset concepts are acceptable as substitutes of the parent concept. This may result in unconditional acceptance of inaccurate results. For example, given a query that requires a Spatial Thing concept, a logic-based matchmaker would return a service that exhibits either a Point concept or an Address concept as perfect matches (see ontology (a) in Fig. 1). However, there is a difference between the ontological aspects and the engineering aspects of the matching in this case. A human programmer would still need to convert the service from Spatial Thing to an Address, and might reject the notion of perfect match in this case.

The similarity measure suggested in this study is designed to address these two drawbacks of logic-based approaches, namely the limited approximation and the subsumption assumption. It extends the approximation scope to handle several ontology modeling approaches. Our similarity measure also provides a gradual approach towards service similarity, ranking results according to the distance edit, i.e., the amount of work required to adapt them to a query. Our edit distance measure for calculating semantic and behavioral differences between an advertised and a requested service can also explain matchmaking results to users.

Our work is also related to non-monotonic service similarity of Noia et al. [19], who found it to be more intuitive than text-based vector space model retrieval using an experimental evaluation of three test cases and 30 people. Our study differs from [19] in two important aspects. First, we use a different methodology: rather than developing a theoretical similarity model and then evaluating it, we first found out empirically users' similarity preferences and then modeled them. Second, we study the similarity between complex services, which include behavioral properties and general relations between concepts.

2.3. Hybrid approaches

The rigidness of logic-based methods led to the development of hybrid matchmakers [17]. This approach augments logic-based methods with other matching methods, especially from the information retrieval (IR) field. Works such as LARKS by Sycara et al. [20], OWLS-MX by Klusch et al. [1], iSPARQL by Kiefer and Bernstein [21] and Stroulia and Wang [22] introduce methods that combine text-based similarity, type signature matching, and logic-based matching of concept classes. Experimental evaluations done by Mikhaiel and Stroulia [23] and by Klusch et al. [24] show that hybrid approaches outperform logic-based approaches, and that using a multitude of methods improves the performance of the matchers.

Our work complements the hybrid approaches; the edit-distance similarity measure can be employed as a sub-method in a hybrid matchmaker, along with text-based and other matching methods. Our method can be used as a fine-grained addition to logic-based matching, providing a similarity measure which can be explained to the user via ontological concepts. From the practical standpoint of service matching, it is highly likely that embedding our method within a hybrid approach would provide more precise matches than our method alone for two reasons. First, mounting evidence has shown that the combination of multiple matching methods outperforms any single method. Second, our method takes into account semantic relations as expressed in an ontology, which may limit the recall and precision if the ontology is incomplete or insufficient.

Our work also differs from other efforts in service matching in its motivation and methodology. Our main motivation is to understand people's preferences in service similarity rather than improve service retrieval using existing measures for precision and recall. Knowledge about users' similarity perceptions can be used to enhance existing hybrid approaches by guiding benchmarks and test cases for Semantic Web services matching such as OWLS-TC [24].

3. Approximate service similarity

In this section we describe a schema for measuring similarity between Web services. We first introduce the basic definitions used throughout the paper. We then formalize a general model for measuring the affinity between two service compositions by estimating the edit distance between two compositions, i.e., the number of changes required to bridge the gap between the compositions.

3.1. Service retrieval components

The basic components of service retrieval are *operations*. An operation is a specification of an atomic function provided by the service that performs a task which is not further divided. Operations are defined using a set of properties, which specify meta-data about the functionality of the operation. In this study we use only two types of properties: inputs and outputs.[3] We formally define an operation as follows:

Definition 1 *(Operation, OP).* An operation is a tuple $OP \equiv \langle URL, Props, label \rangle$, such that

- *URL* is a uniform resource locator—a text string that uniquely identifies the operation.
- *Props* is a set of properties, $p_1, p_2, \ldots, p_n$. Each property is taken from a type domain: $p_i \in \text{Input} \cup \text{Output}$
- $label : Props \rightarrow \mathcal{O}$ is a labeling relation that associates a property from the set *Props* with a concept class taken from an ontology $\mathcal{O}$.

We base our definition of ontology on OWL [15], the most common ontology language. We assume that we have a single ontology

[1] WC3 Basic Geo Vocabulary, http://www.w3.org/2003/01/geo/.

[2] MIT Simile Project Location Ontology, http://simile.mit.edu/2005/05/ontologies/location.

[3] For the sake of simplicity, we consider OWL-S effects as outputs and OWL-S preconditions as inputs.

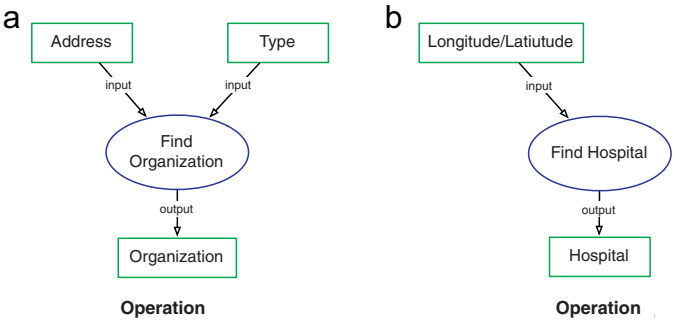

Fig. 2. Two operations which can possibly answer the query *"Find a service that accepts an address and returns the closest hospital"*. The "closest to" ontological relation is not displayed.

$\mathcal{O}$, which denotes a unification of all the ontologies in the service retrieval framework.[4]

A service is defined as an identifiable set of operations with no mandatory information about the execution order of the operations. The behavior of a service is described using a *composition*. In our model, compositions are used to describe both queries and the results of the service retrieval processes related to these queries.

Definition 2 *(Composition, Com).* A composition is a directed, acyclic and weakly connected graph $Com \equiv \langle \mathcal{OP}, Flows \rangle$, where:

- $\mathcal{OP} = \{OP_1, OP_2, \ldots, OP_n\}$ is a set of operations.
- $Flows \subseteq \mathcal{OP} \times \mathcal{OP}$ is a set of directed data flows between operations.

A composition provides a limited description about the behavior of a set of operations by specifying precedence between the operations. This view is inherently limited, as we do not specify information about such items as the conditions in which some precedence takes place, the data that is passed from one operation to another, and state changes.

3.2. Composition affinity

In this section we define a general model for analyzing and measuring the affinity between any two compositions. Our model for similarity is based on counting the number of changes required to modify one composition to another, much like the way edit distance is used in strings. Given two compositions, Com_1 and Com_2, we say that Com_1 is similar to Com_2 if Com_1 is augmented with a finite set of operations which imitate the functionality of Com_2. The augmenting operations are called **virtual operations**, as they reflect desired functionality, which does not necessarily exist.[5]

To illustrate how the similarity measure is applied, consider our running example depicted in Fig. 2 (the underlying ontology is displayed in Fig. 3). The two operations, *a* and *b*, differ in their input and output parameters:

- Operation *a*: *Find Organization* receives an *address* and returns an *Organization*, e.g., a company or a non-profit organization.
- Operation *b*: *Find Hospital* receives a *longitude/latitude* and returns a *Hospital*.

[4] We assume that $\mathcal{O}$ is a single ontology, which is a combination of the original ontologies used to annotate the Semantic Web services. More background of ontology merging and aligning can be found in Euzenat and Shvaiko [25].

[5] This definition holds for operations as well, as they can be considered compositions that include a single operation.

None of the two operations suits perfectly the query expressed in the caption of Fig. 2. Operation *a* receives an Address concept, which is common to both the query and the operation, but returns an Organization. Furthermore, it has an extra input parameter. Therefore, by adding an operation that transforms an Organization to a Hospital, Operation *a* can be reused. Similarly, Operation *b* can be used by adding an operation that transforms a longitude/latitude concept to an Address concept. The certainty that such transformations are feasible depends on the properties of the ontological relations between the two concepts. A "rational" human engineer would choose the operation that requires the least amount of adaptation operations to implement the service defined in the query. This is the intuition that guides our similarity measure.

Similarity is measured by calculating the certainty of the inferred virtual operations. The certainty is derived from either the structure of the ontology or the composition. Consider the ontology model in Fig. 3. As there is a 1:1 relation between *Address* and *longitude/latitude*, it is likely that a mapping operation from longitude/latitude to Address can be constructed. However, the relation between Agent and Address is 1:m, lowering the probability of a successful transformation between these concepts. In the following section we define a schema for evaluating the constructability of virtual operations. We analyze the situations in which virtual operations can be created and how the impact on similarity can be assessed.

The basic tool for evaluating similarity is a *constructor*, a function that maps two comparable compositions to a set of virtual operations.

Definition 3 *(Constructor).* A constructor is a function:

$$\Psi : Com \times Com \rightarrow VOP_1, VOP_2, \ldots, VOP_n$$

characterized by two properties:

- $type : \Psi \rightarrow \{\text{Set-hierarchy, Relation, Behavioral}\}$ assigns each constructor an affinity pattern type.
- $\mu : \Psi \rightarrow \mathcal{R}$ is a cost function that assigns a single cost value to each specific construction.

The cost function, μ, is calculated differently for each type of constructor, as described in the following section. The similarity definition (Definition 8) uses μ as the basic building blocks in comparing several compositions retrieved for a query, preferring those with the minimal cost.

4. Affinity patterns

We classify the possible constructors of virtual operations into three semantic affinity patterns, which reflect possible inferences over the ontology and the structure of the composition. Two constructors are semantic, i.e., they are based on semantic properties of the underlying ontology that defines the operation's parameters. The third constructor is behavioral, as it is based on the structure of the composition graph. In this section, the three affinity patterns are defined and explained. We then define two properties of these patterns: completeness and similarity.

4.1. Semantic affinity patterns

The main challenge in designing the semantic affinity patterns is to provide a single frame of reference which is adaptable to various types of ontological relations, and specifically the following relations:

- Set relations between concepts classes, such as OWL sub-class, intersection, and union.

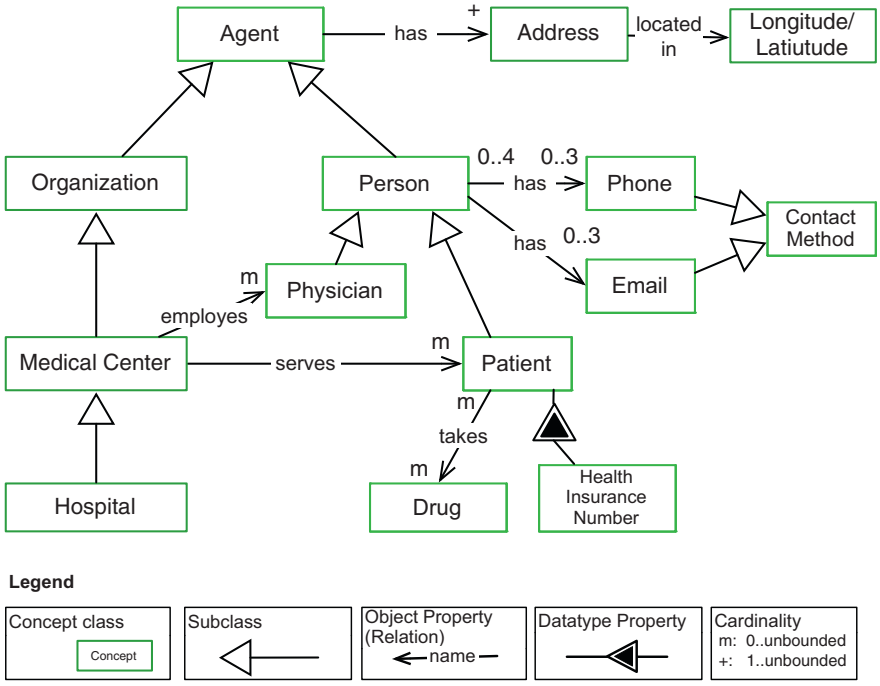

Fig. 3. An example of a healthcare ontology, including concepts, subclass relations, object properties, and datatype relations.

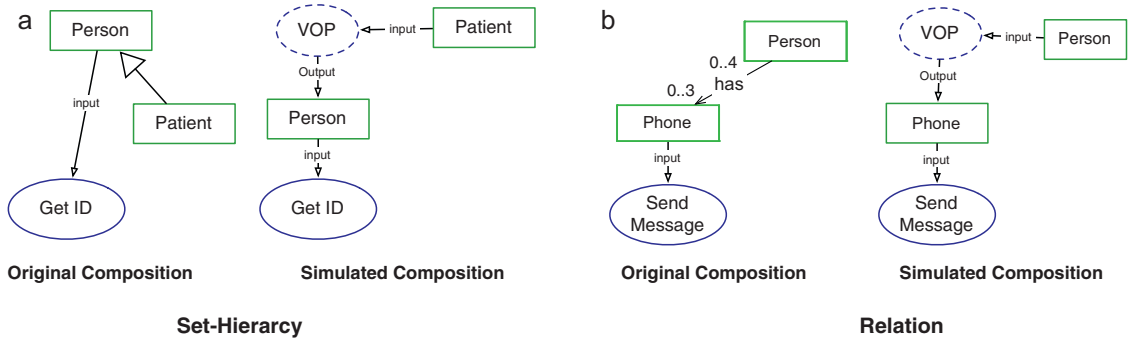

Fig. 4. Examples of applying the set-hierarchy and relation pattern.

- Property relations, as in the OWL object properties and datatype properties.

Semantic constructors are handled by substituting an operation parameter concept with another concept. The construction cost is as function of the semantic relation between the concepts. In what follows, we assume two given operations, OP_1 and OP_2, each with a single parameter of the same type (e.g., input or output), C_1 and C_2, respectively. The constructor is denoted by $\Psi(OP_1, OP_2)$. Section 4.1.3, in which combining semantic patterns is discusses, includes cases in which the operations have more parameters. The construction cost is based on the cardinality of the relational similarity [26] between concepts C_1 and C_2. The relation cardinality represents the information content of the relation. As the number of instances of the relation grows, so does the uncertainty of finding an adequate assignment of these instances. When the cardinality of *Rel* is infinite, finding a probable assignment becomes impossible.

4.1.1. Set hierarchy pattern

The set hierarchy pattern is concerned with affinity between operations whose parameters are related by a set relation of subclass, union, intersection, or restriction. As depicted in Fig. 4(a), an operation whose input concept is *Person* can be simulated by an operation whose input is the concept *Patient*, which is a subclass of *Person*. We define two concepts, C_k and C_l. With no loss of generality, let us refer to C_l as the input concept of the original composition. We denote $P(C_i)$ as the set of data-type and object properties that concept C_i exhibits. We define the constructor cost for a single concept as follows.

Constructor 1 (Set hierarchy pattern). The construction cost for two operations OP_l with C_l as a parameter and OP_k with C_k as a parameter of the same property (e.g., input or output), is defined as:

$$\mu(\Psi_{set}(OP_l, OP_k)) = \frac{|P(C_l) \cup P(C_k)|}{|P(C_l) \cap P(C_k)|}$$

□

For example, consider the two concepts, *Person* and *Patient*, defined in Fig. 3, where *Patient* is a sub-class of *Person*. The property sets of the two concept classes are:

P(Person) = {Email,Phone,Address}
P(Patient) = {Health Insurance Number,Takes Drug,Email,Address,Phone}

The construction cost function is calculated as the ratio between the union and the intersection of the property sets:

$$\mu(\Psi_{set}(OP_1, OP_2)) = \frac{5}{3}$$

To better understand how the construction cost is calculated, consider the construction cost of a slightly different operation, for which the *Agent* and the *Patient* concepts are parameters. The size of the intersection between the properties of the two concepts is 1 (based on the property Has Address), while the size of the union is 5 (based on the properties Has Address, Has Phone, Has Email, Takes Drug, and Health Insurance Number). Therefore, the construction cost would be 5. This example shows how the construction cost reflects the semantic distance between concepts. As the semantic difference between *Agent* and *Patient* is greater than the semantic difference between *Person* and *Patient*, constructing a composition between operations with these concepts would require a greater number of modifications, as reflected by the construction cost.

The set hierarchy pattern finds the cardinality of the relation between the two concepts based on the proportion of common properties between the concepts. The motivation behind this design is to cover all the types of OWL hierarchical set relations, such as subclass, union, and intersection. Furthermore, this pattern relies on theoretical foundations, including Jaccard similarity coefficient [27], similarity in semantic networks by Rada et al. [28], feature-based similarity in description logic by Borgida et al. [29], and general cognitive theories about similarity by Tversky [30].

This pattern has some limitations. First, concepts that do not contain object or datatype properties will not come out as being similar even if set hierarchy relations are specified. Second, the overall number of properties influences the construction cost. For concepts at lower levels of the class hierarchy and which inherit many properties, the construction cost is expected to be higher than that of concepts that inherit only few properties.

4.1.2. Relation pattern

In this study, we investigated how users substitute operations based on general relations. For example, in Fig. 4(b), the original composition exhibit a *Phone* concept as input. The relation pattern describes the construction that substitutes the composition with one in which the input concept is a *Person*.

Relations can be interpreted in various ways. An overview on relations in description logic was done by Kü sters and Bordiga for [31]. In this study, we refer to relations which are known in OWL as bidirectional (i.e., non-functional) object properties. We assume that relations are parameterized by mandatory cardinality, e.g., two classes that have min n and max m object properties. For example, in Fig. 3, a medical center has an unlimited number of patients, and a patient has a single medical center. We also assume that relations cannot have restrictions over the type of instances belonging to the relation.

Let C_s and C_d be two arbitrary concepts that are related through relation $Rel(C_s, C_d)$. As object properties are binary, using two concepts is sufficient for defining the pattern and calculating the cardinality of the relation. The virtual operation maps properties of source concept to properties of the related destination concept. We use the colon mark (:) to denote the relation between an instance (on the left-hand side) and a concept class (on the right-hand side).

Constructor 2 (relation pattern). The cost function of the constructor is derived from the cardinalities of the relation as follows.

$$\mu(\Psi_{rel}(OP_1, OP_2)) = \\ |\{x : C_d \mid (y, x) \in Rel(C_s, C_d)\}| \times |\{y : C_s \mid (y, x) \in Rel(C_s, C_d)\}| \quad \square$$

The possible assignment of instances in the relation is the Cartesian product of the assignments on both directions of the relation. Therefore, the relation pattern exemplified in Fig. 4(b), which C_s = *Person*, C_d = *Phone* and $Rel(C_s, C_d)$ = *has* has the cost function:

$$\mu(\Psi_{rel}(OP_1, OP_2)) = 3 \cdot 4 = 12$$

The cardinality reflects the uncertainty of finding a specific instance pair among all pairs linked by the relation. In the case of the relation between *Person* and *Email*, where every email has a single Person, finding a specific assignment of the relation relies only on the number of emails a person has (up to 3 in our example). Therefore the construction cost is 3, much less than the construction cost of finding an assignment of phone to person. Unlimited maximal cardinality (e.g., 0 to n, where n is unbounded), yields an infinite cost, since the average cardinality is infinite as well.

4.1.3. Combining semantic patterns

The possibly complex structure of ontologies and operations requires special handling of cases with complex ontological structures or multi-parameter operations. In this section we define two additional constructors: (1) the Min constructor, which handles complex ontological structures, and (2) the multiple constructor, which handles multi-parameter operations.

Concepts can be related indirectly, through other concepts and with multiple types of relations. Consider, for example, the relations between *Organization* and *Person* in the ontology depicted in Fig. 3. As the two concepts are not directly related, comparing two operations which are based on these concepts is not feasible with the patterns introduced so far. Furthermore, some concepts are comparable even if they are related by a combination of different ontological relations, such as *Address* and *Organization*.

In order to compare operations based on two concept classes that are not directly related, the algorithm iterates over the possible paths between these concepts in the ontology and selects the path that minimizes the overall construction cost. The *Min* constructor, defined below, reflects the minimal amount of work necessary for adapting a composition to a query.

Definition 4 *(Min Constructor).* Let $path = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ be a sequence of constructors, starting at the source concept class C_s and ending at the destination concept class C_d. We define the cost function of the combined constructor as the sum of the constructor costs along the minimal path (*minpath*), i.e., the path that satisfies the following equation:

$$\mu(\Psi_{min}) = \mu(minpath) = \min_{path_i} \sum_{\Psi_j \in path_i} \mu(\Psi_j)$$

The *Min* constructor finds the shortest weighted path, where the cost is the sum of weights on the edges. For example, evaluating a construction for two operations, OP_1 with parameter *Person* and OP_2 with parameter *Contact Method* in the ontology depicted in Fig. 3, could be done in two ways:

1. Using the *Phone* concept, with cost $\mu = 12$.
2. Using the *Email* concept, with cost $\mu = 3$.

Therefore, the cost of the construction is: $\mu(\Psi_{min}(OP_1, OP_2)) = 3$

The second constructor is the multiple constructor, which handles the common case in which an operation includes more than one parameter. To evaluate the similarity of the whole operation, we define a cost function over the combination of the constructors handling the parameters of the operation. As reusing the existing operation requires adding virtual operations that would transform all necessary parameters, we define the combined cost as the sum of the costs of all the constructors. We define an *orphan parameter* as a parameter which exists in one operation but does not have a counterpart finite construction cost parameter in the other operation.

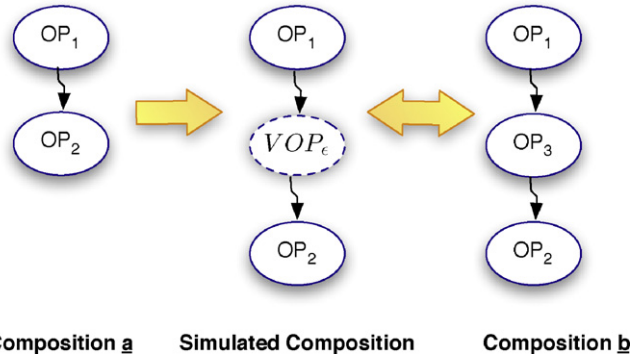

Fig. 5. An example of applying the behavioral pattern: composition *a* is matched with composition *b* by simulating *b* with *a*. If such a simulation exists, then the similarity is defined by its construction cost.

To comply with the definition below of the behavioral pattern, we set the price of an orphan parameter to 1. The multiple constructor is defined as follows:

Definition 5 *(Multiple constructor).* Let $Con_{OP_k} = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ be a set of constructors handling multiple parameters $1\ldots, m$ of operation OP_k. Let n_p be the number of orphan parameters (parameters which appear in one of the operations and do not have a counterpart in the second operation). We define the cost of the multiple construction as the sum of the costs of Con_{OP_k}:

$$\mu(\Psi_{multiple}(OP_k)) = n_{(p)} + \sum_{\Psi_i \in Con_{OP_k}} \mu(\Psi_i)$$

Consider the two operations in Fig. 2 (page 2). The construction cost between Address and longitude/latitude is 1, and the construction cost between Hospital and Organization is 3. The parameter named "Type" is orphan, as it does not have a counterpart. Therefore, the overall construction cost is 1 + 3 + 1 = 5.

4.2. Behavioral pattern

The behavioral pattern is used to compare compositions rather than single operations. It evaluates inexactness stemming from the structure of the composition graph rather than from the semantic properties of the operations. For example, the behavioral pattern evaluates the similarity of compositions that share some of the operations but not all of them.

Fig. 5 depicts two compositions, where composition *b* has one excessive operation, OP_3, compared with composition *a*. The simulated composition is defined as the intersection of the compositions. It contains the operations shared by all the compositions. In our example, the shared operations are OP_1 and OP_2. An empty transition operation, denoted VOP_ϵ and defined below, replaces operations that originally connected existing operations and were removed by the construction. For example, OP_3 connected OP_1 and OP_2 in composition *b*. VOP_ϵ serves as a channel for delivering information between operations without affecting their interface.

We define the virtual operation, VOP_ϵ, using its constructor, Ψ_ϵ:

Definition 6. (Empty Transition Virtual Operation Constructor - Ψ_ϵ)

$\Psi_\epsilon(OP_i, OP_j) = \{OP_i, OP_j, VOP_\epsilon\}$
s.t.
$VOP.in = OP_i.out \wedge VOP.out = OP_j.in$

The cost of VOP_ϵ is arbitrarily set to 1. Therefore, the cost of the construction is the *graph edit distance* between the two original compositions and the simulated composition. The graph edit distance is defined as the number of node and edge deletions or insertions necessary to transform one graph into another [32]. It is a simple measure of graph similarity, similar to edit distance on strings. If the graphs are identical, $edit(Com_1, Com_2) = 0$. If the graphs are disjoint, i.e., they do not contain any common subgraph, then $edit(Com_1, Com_2) = |Com_1| + |Com_2|$. In Fig. 5, the edit distance is 1, as one operation was removed through the construction. Finally, we define the constructor cost for two compositions:

Definition 7. Composition construction cost Given a complex construction process $\hat{\Psi}(Com_1, Com_2)$, which consists of a set of lower-level constructors, $\Psi_1, \Psi_2, \ldots, \Psi_n$, the overall construction cost is:

$$\mu(\hat{\Psi}(Com_1, Com_2)) = \sum_{i=1..n} \mu(\Psi_i)$$

4.3. Construction cost properties

We now prove that the construction cost μ is a distance metric. Proving this characteristic has several advantages. First, learning algorithms such as K-means, nearest-neighbors classifiers and kernel algorithms (e.g., SVM) require a distance metric. Proving that the cost is a distance metric is useful for various applications that rely on learning algorithms, including indexing by service clustering and automatic categorization of services. Second, some valuable distance metric properties of the construction cost are inherited, including insensitivity to the order in which constructors are applied to a composition.

Theorem 1. *The construction cost function is a distance metric.*

Proof. The construction cost μ satisfies the four properties of a distance metric as follows:

1. Non-negativity: μ is a sum of relation set cardinality, which is always non-negative. Therefore $\mu(\Psi) \geq 0$ for every Ψ.
2. Identity: Applying a constructor to the same composition will yield an empty set of virtual operations. As the composition construction cost function is a sum of the virtual operations, $\mu(\Psi(Com_i, Com_i)) = 0$.
3. Symmetry: By the definition of the constructors, they are insensitive to order, such that the following holds for each type of constructor: $\Psi(Com_1, Com_2) = \Psi(Com_2, Com_1)$. Thus, their costs are equal too.[6]
4. Triangle inequality: $\mu(\Psi(x, z)) \leq \mu(\Psi(x, y)) + \mu(\Psi(y, z))$ Let us assume that there exist three compositions, x, y and z, such that $\mu(\Psi(x, z)) > \mu(\Psi(x, y)) + \mu(\Psi(y, z))$. According to Definition 4, Ψ_{min} is the construction that yields the minimal cost. We can construct $\Psi' = \Psi_{min}(x, y) \cup \Psi_{min}(y, z)$, with $\mu(\Psi') \geq \mu(\Psi(x, z))$. Because this construction is feasible, it would be chosen as the minimal cost construction, such that, $\Psi' = \Psi_{min}(x, z)$, and $\mu(\Psi_{min}(x, z)) = \mu(\Psi')$. Therefore, $\mu(\Psi(x, z)) \leq \mu(\Psi(x, y)) + \mu(\Psi(y, z))$. □

4.4. Similarity definition

In this section, we define a measure of similarity, which is based on the construction cost of affinity patterns. The construc-

[6] Note that the similarity function contains different weights to superclass and subclass relations, which violates similarity. However, the proof given here relates solely to the construction cost. In practice, when implementing the similarity function, weights are applied after the construction cost is calculated.

Table 1
Affinity pattern weights. The weights were determined empirically such that higher weight indicates a higher level of similarity between compared objects.

Pattern	Weight (ω)
Set-Hierarchy (subclass)	3.2
Set-Hierarchy (superclass)	4
Relation	2.5
Behavioral	3.8

tion cost is a measure of dissimilarity, such that a high cost reflects a great difference between compositions. In many practical scenarios, however, users find more intuitive measures of similarity than measures of dissimilarity. For example, in Web search engines, high ranked items are expected to be more similar to the query than low ranked items. Thus, we wish to define similarity with two goals in mind. The first goal is that the similarity definition reflects theoretical principles in perceptions of similarity, e.g., information-theoretic methods.

The second goal is that the similarity reflects our empirical findings, described in detail in Section 5. We have found that while the construction cost is linear with the number of changes required in the construction process, this is not how users perceive similarity. Therefore, we have designed a similarity measure that reflects this phenomenon using a mathematical model, which is both simple to analyze and intuitive for users and application developers.

Formally, similarity is defined as a function that accepts two compositions, Com_1 and Com_2, and returns a number in $\mathbb{R}$:

Definition 8 *(Similarity).* The similarity between two compositions, Com_1 and Com_2, is defined as follows:

$$sim(Com_1, Com_2) = \lambda - \sum_{\Psi_i} \log_{\omega_i}(1 + \mu(\Psi_i))$$

The similarity function assigns the constant λ to identical compositions, which have a construction cost of $\mu = 0$, and a decreasing value to compositions with incresing construction costs. λ is used to parameterize the similarity value in a way that is intuitive to users. For example, to define the top similarity score to 5, as we did when analyzing the experimental results, we set $\lambda = 5$.

The weight parameter, $\omega : type \rightarrow \mathcal{R}$, is used to parameterize the cost function according to the type of the pattern. The weight parameter is the outcome of analyzing our empirical results, which revealed different gradients of the similarity function for different patterns. The weights were computed by performing linear regression analysis over the empirical results presented in Section 5.2 such that the similarity is transformed into a linear function. Table 1 shows the weight values of the different affinity patterns. Based on our empirical results, we split the weight of the Set Hierarchy pattern into a superclass case and a subclass case, as specified in Table 1.

Our definition of similarity is inspired by information-theoretic methods developed by Lin [33], Resnik [34], and Hau et al. [35]. In two separate studies, Lin and Resnik suggested associating a probability p with concepts in an ontology subclass hierarchy to denote the likelihood of encountering an instance of a concept class C. If $C_1 \sqsubseteq C_2$ (i.e., C_1 is a sub-class of C_2) then $p(C_1) < p(C_2)$. The information content of a concept C is then defined as a function over the probability of its instance likelihood. Hau et al. extended this notion to semantic Web service matching by defining the information carried by each concept as its set of properties and comparing the sets [35]. We further extend the notion of information-theoretic similarity in two ways: First, based on the composition structure, we extend semantic similarity to include behavioral similarity. Second, we extend the possible set of semantic relations for similarity with object properties.

Table 2
Definitions of context classes, each representing groups of concepts with a particular relation to the anchor concept, $\check{C}$.

Context class of $\check{C}$	Definition
Equivalents	$\{C_i \in \mathcal{O} \mid C_i = \check{C}\}$
Types	$\{C_i \in \mathcal{O} \mid C_i : \check{C}\}$
Subclasses	$\{C_i \in \mathcal{O} \mid C_i \sqsubseteq \check{C}\}$
Superclasses	$\{C_i \in \mathcal{O} \mid \check{C} \sqsubseteq C_i\}$
Intersections	$\{C_i \in \mathcal{O} \mid \check{C} \sqsubseteq C_i \sqcap C_j\}$
Unions	$\{C_i \in \mathcal{O} \mid \check{C} \sqsubseteq C_i \sqcup C_j\}$
Datatype properties	$\{P_i \in \mathcal{O} \mid P_i \in P(\check{C})\}$
Object properties	$\{C_i \in \mathcal{O} \mid \exists R, R(\check{C}) = C_i\}$
Composed	$\{C_i \in \mathcal{O} \mid \exists C_1, C_2, \ldots, C_n, C_1 \in Class(C_2) \ldots C_n \in Class(\check{C})\}$
Unrelated	$\{C_i \in \mathcal{O} \mid C_i \notin Class(\check{C})\}$

4.5. Completeness

In this section, we prove the completeness of the similarity measure. We show that the set of affinity patterns covers all possible relations between compositions, such that the set of affinity patterns returns a similarity value for any arbitrary composition. We start by proving that the set of semantic patterns is complete, i.e., that we do not need any other pattern to compare any two operations based on their parameter semantics. We show how any two concepts can be compared by our method. We do so by demonstrating that all the types of semantic relations within an ontology are covered by one of the semantic affinity patterns or a combination of them. We then prove that the behavioral pattern is complete. Finally, we show that graph edit distance, which is equivalent to our behavioral pattern method, is computable on any arbitrary compositions.

Theorem 2. *The set of affinity patterns is complete.*

Proof. For any given concept, $\check{C}$, we define a set of *context classes*, each of which defines a subset of concepts in $\mathcal{O}$, according to their relation to the concept. Table 2 contains the formal semantics of each of the relations within our definition of the ontology. Each concept in the ontology is classified to *one* of the context classes. For example, all the concepts equivalent to the concept $\check{C}$ are classified into the Equivalents context class. We now prove that the set of semantic affinity patterns satisfies all the types of relations between any two concepts.

All the classes except the Composed class in Table 2 cover all the axioms of the ontology. Theorizing on the ontology as a graph [15], these classes define all the direct relations between two concepts. A concept which has an indirect relation to the anchor concept is defined through the Composed class as a concept which is related to the anchor concept through a set of concepts $C_1, C_2, \ldots, C_n$. The context class Unrelated defines all the concepts which are not related through any other context class (including indirectly, through the composed class).

Each one of the context classes is covered by an affinity pattern as follows:

1. The set-hierarchy pattern covers all the cases of equivalents, subclasses, superclasses, intersections and unions.
2. The relation pattern covers all the cases of object and datatype properties (referred to in this paper as properties and relations, respectively).
3. Pattern composition covers all the cases of the composed pattern, where concepts are not related through a single relation. As the construction cost is a distance metric, the composition pattern returns the maximal similarity value.
4. Unrelated concepts are not similar, and thus cannot be bridged by a construction process.

Therefore, the set of semantic affinity patterns is complete. A proof regarding the completeness of the behavioral pattern is provided in Appendix A. □

5. Evaluating affinity patterns

We have evaluated the affinity patterns empirically, with the objective of predicting the way humans would benefit from retrieval systems that utilize the patterns we defined. To this end, we designed and administered a questionnaire, in which subjects were asked to assess the similarity of given models.

This study was inspired by the seminal work "Features of Similarity" by Tversky [30], which suggested a general model of cognitive similarity assessment. The study of Budanitsky and Hirst [36], which compared WordNet [37] similarity measures, and the study of Bernstein et al. [38], which examined an ontology for service description, have shown that in a setting of ontology-based knowledge systems, human judgment provides the best assessment of the quality of a measure for affinity between concepts.

5.1. Method

The evaluation was done via a Web-based survey, in which participants were asked to assess the relation between a set of query and composition pairs. The pairs were assessed both quantitatively, by providing a grade for the similarity, and qualitatively, by providing a detailed explanation for various aspects of the similarity. The survey consisted of three stages. At the first stage, participants supplied demographic information, including their age, years of study, and education. At the second stage, participants provided feedback on the similarity of query and composition pairs. At the final stage, participants reviewed their initial assessment and could modify it.

5.1.1. Research population

The research population included 127 participants, of whom 15% were studying towards their masters or doctoral degrees, while 85% were in their 5th semester of studying towards their bachelor degrees. Of the participants, 70% were students in the Faculty of Industrial Engineering and Management at the Technion, Israel Institute of Technology, while the other 30% were students of the Technion_(tm)s Faculty of Computer Science. All participants were students in a course teaching analysis and specification of information systems. Filling in the questionnaire was defined as a bonus task for the course, crediting the participants with 3% of the final course grade. The grade was based on the level of details supplied as answers to all the required questions.

All the participants had at least one basic and one advance course in software engineering, and they also took at least one course in systems analysis. Hence, our research population is a satisfactory proxy to software engineers and systems analysts, who are the potential users of service composition. One of the characteristics of experienced software engineers and system analysts is their ability to intuitively evaluate semantic and structural approximations. For our participants, who were novice users, these skills were less developed.

Table 3
Experiment test cases, according to the pattern they were assigned to explore and ontology domain

Pattern	E-commerce	HR	Travel
Set-hierarchy	4	4	
Relation	5	5	4
Behavioral	3		2

5.1.2. Similarity assessment

Participants were presented with a sequence of 12 query/composition pairs. The pairs were randomly selected and ordered from a set of 30 test cases from three domains: e-commerce, Human Resources-HR, and travel. The concepts used in each test case were based on a single OWL ontology. The ontology was not presented to the participants in order to evaluate intuitive usage of the system. We adjusted the number of pairs assigned to each user according to a reasonable expected burden, as participants were asked to provide detailed qualitative feedback on each pair. Table 3 describes the exact distribution of the test cases, domains and patterns.

We used OWLS-TC, a benchmark for semantic service retrieval [24], to construct the test-cases. Test cases were designed using well known domain ontologies from OWLS-TC [1]. We looked for test cases that included sufficient semantic and behavioral variance. Each set of test cases included several compositions, which were identical except for an unrelated variable: a single change one of the semantic or behavioral properties of the composition with regard to the query. For example, in Fig. 6, test-case 2 includes an ISBN input rather than a Title input. The composition that was identical to the query was called the "baseline" composition. This way we were able to control all the elements of the compositions except for the independent variable and could evaluate how similarity is

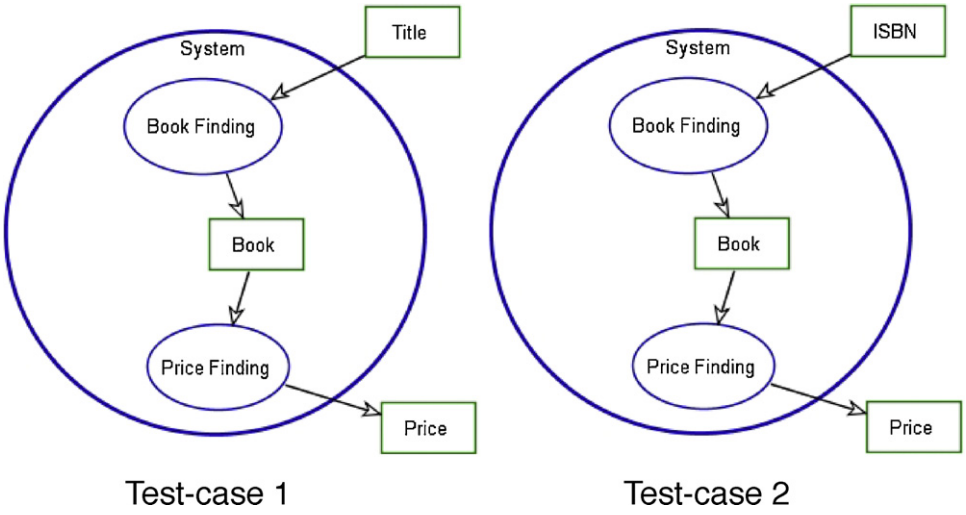

Fig. 6. An example of two test cases on the same query: "Find the price of a book according to the book's title".

perceived. Test-case 1 in Fig. 6 is a baseline test-case. All the test-cases were assessed by two fellow researchers, which were not the authors of this article.

Fig. 6 depicts a sample of a query and two compositions. Queries were expressed using a short textual description to simulate usable scenarios in service composition by end-users [39]. Compositions were expressed by Object-Process Diagrams, the visual formalism of OPM (Object-Process Methodology) [40] to define the compositions in an exact yet usable and readable format.

Test case grading was based on a Likert scale of 1 to 5 for each one of the following parameters:

- *Usefulness*: The extent to which the model can be used to implement the query.
- *Completeness*: The extent to which the model meets all the query's requirements.
- *Exactness*: The extent to which the model contains the right amount of elements, i.e., that the model contains neither excessive nor missing elements.

The first parameter, usefulness, measures the correspondence between the query and the composition in the context of *reuse*. The two other parameters measure the correspondence in a more general context. Exactness and completeness overlap, as an exact matching is also a complete matching. However, we wanted to gain a more subtle distinction between a situation in which the composition contains excessive elements and a situation in which the composition does not contain all the necessary elements. The completeness parameter distinguishes between these two cases. Participants were asked to provide one open-ended explanation to describe their grading and another open-ended explanation for describing the changes they would make to the composition to make it more similar to the query.

Each set of compositions met the following criteria:

- At least one composition answers the given query perfectly. We denote this composition as the *baseline* composition.
- At least one composition has a minor difference with respect to the query.
- At least one composition has a major difference with respect to the query.

5.2. Results

The results are organized in two categories. First, we looked at general characteristics of service similarity, such as how various parameters, e.g., usefulness and exactness. Second, we evaluated each affinity pattern. We measured the statistical significance of the results by comparing the scores for different test cases to the baseline composition. An unpaired t-test shows that the score sets for all the test-cases were statistically significant ($p < 0.03$). Naturally, the scores for the baseline composition are not maximal, since when it comes to human judgment, there were different user perceptions of the baseline test cases. We attribute some of the variance to the fact that queries were represented in text, while results were visually represented using diagrams. However, for each of the test cases, the users gave the top scores to the baseline, and the score variance for the baseline case was significantly lower compared with the other test cases.

The similarity measure is modeled by approximating the mathematical model according to the empirical results. We evaluated several similarity models using the empirical results by looking at the minimum of the average difference between the predictin of the candidate model and the experimental results. The final model is compared with the users' 1-5 Likert scale by parameterizing the similarity definition (Definition 8), such that the maximal similarity prediction constant, λ, was set to 5. This parameterization allows model predictions to be compared directly with the empirical results. Linear regression was used to find the pattern weights presented in Table 1 (predictions and results were transformed to a linear function to compute the regression).

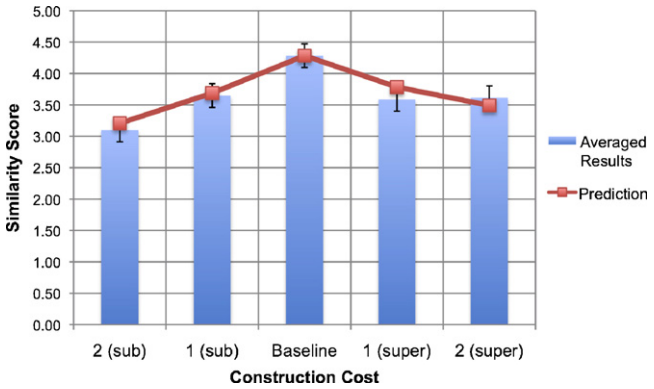

Fig. 7. Set-hierarchy pattern similarity assessment. The *X*-axis represents the cost construction, based on the set hierarchy distance. The *Y*-axis represents the similarity. The bars are the averaged results over all the test cases, and the line represents the prediction of our model.

5.2.1. Parameter grading

The correlation between the three parameters, completeness, exactness and usefulness, is significant (Pearson correlation of 0.93). Participants tended to give higher values to usefulness, which scored an average of 3.2 vs. 3.03 of exactness and 3.07 of completeness. Nevertheless, no statistical significance was found between the different criteria. The textual feedback provided some insight into an explanation of this phenomenon. When describing their grading, the participants tended to base their feedback on the existing aspects of the composition. They thoroughly described the differences between the composition and the query in terms of missing or excessive elements, different process orders, and different interfaces. In describing improvements, participants were creative, specifying new operations and data structures, and relating them to the existing composition using relations, generalized concepts, and specific concepts.

5.2.2. Set hierarchy pattern

The analysis of the set hierarchy relations yields a clear pattern, shown in Fig. 7. Each column in the graph stands for the average score (of all parameters) given by participants for all the test cases related to the set hierarchy pattern. Each *X*-axis value stands for a given set hierarchy distance. The results are ordered according to their subset distance, which is their location in the class hierarchy. For example, column 2 (sub) indicates test-cases in which the variable concept is a subclass (more specific) by two hierarchy levels than the baseline.

The average score by the hierarchical distance form a bell pattern, where the highest score for completeness, exactness and usefulness, is received for the baseline query/composition pairs. The difference between more general results becomes less significant after the initial difference from the baseline, so that the average score of the 1 (superclass) set and the 2 (superclass) set are almost identical. Our results show that similarity between concepts and their superclass concepts is larger than between concepts and their subclass. We model these results by setting a larger weight to superclasses ($\omega = 4$) than to subclasses ($\omega = 3.2$).

5.2.3. Relation pattern

The relation pattern evaluation is presented in Fig. 8. Each column represents an average score for a given relation cardinality

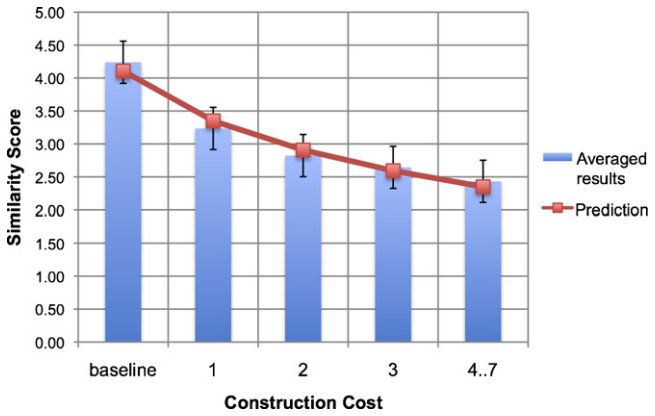

Fig. 8. Relation pattern similarity assessment. The *X*-axis represents the cost-constructions, based on the set relation cardinality. The *Y*-axis represents the similarity. The bars are the averaged results over all the test cases and the line represents the prediction of our model.

value, averaging all relevant relation-pattern results for all users, all test-cases and all parameters. The baseline column represents the results for concepts that correspond directly to the query concepts. Columns with higher construction cost represent concepts that are related to the query concepts with higher cardinality. For example, a query that contains a book concept and a composition that contains an ISBN concept are related by 1:1 relation, which is displayed as a construction cost of 1. The last column, marked by the '4.7" label, represents a set of several composition/query pairs for which multiple cardinalities were presented.

The results show a decline in the average scores as the cardinality grows. The baseline yields the highest similarity values, while higher cardinality yields lower similarity. The negative slope of the curve becomes moderate as the cardinality grows, with good correspondence to the similarity prediction. We did not find a relation between the direction of the relation and similarity. Based on the qualitative analysis of the written explanation written by the participants, we concluded that participants estimated similarity according to the probability of relating two instances of concepts, regardless of the relation direction.

5.2.4. Behavioral affinity pattern

Fig. 9 presents the results for the behavioral similarity pattern. The *X*-axis represents the graph edit distance between the composition and the query. The baseline column represents compositions that are functionally equivalent to the query. The 1 set represents a single edit, whereas the 2 set represents two edits. Test cases with

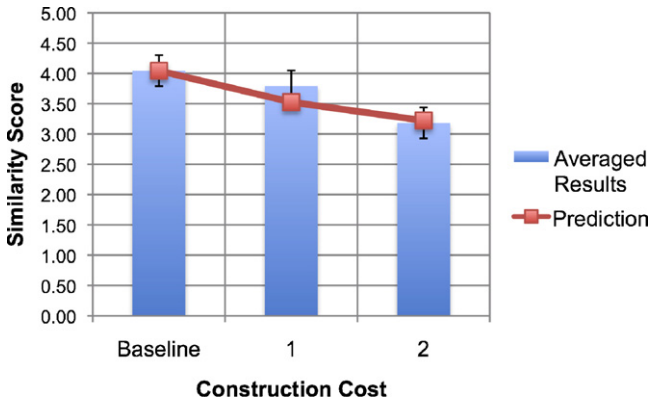

Fig. 9. Behavioral pattern similarity assessment. The *X*-axis represents the graph edit distance between the composition and the query, and the *Y*-axis represents the similarity.

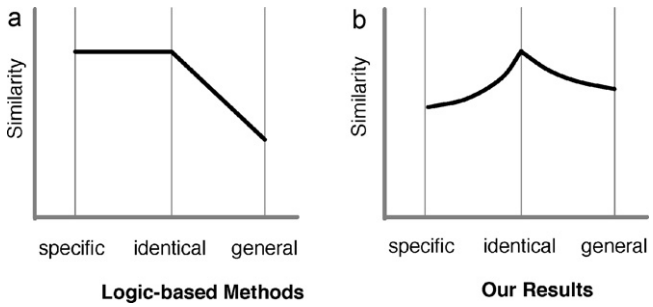

Fig. 10. Comparing logic-based similarity with our definition of similarity.

a larger number of edits were not included in the study, as they were hardly usable and self-explanatory. The *Y*-axis represents the similarity score over all the test cases, all the parameters, and all the participants.

Behavioral similarity provided a higher level of approximation in comparison to semantic approximation. We explain this result by the relative simplicity of implementing changes to operation order where operations are considered "black boxes", with no assumption on their state.

5.2.5. Discussion

The results raise several questions regarding approaches to service similarity. For example, logic-based results assign equal importance to plugin (more specific) and to exact (baseline) results [17], because more specific results follow the axioms of the general results. According to our empirical results, human participants perceive specific results as different from exact results. This is depicted in Fig. 10, where the *Y*-axis represents some abstract similarity and the *X*-axis represents the specification dimension. The difference starts with more specific results, through the baseline results (identical to the query), and ending with the more general results. Moreover, we discovered that human participants perceive a "softer" notion of similarity than that defined by logic-based methods [9,24]. This observation is apparent also from evaluating the contexts in which humans judge similarities. In the context of reuse, participants relaxed their similarity definitions and were more forgiving towards inexact results.

Another significant result refers to the relation pattern. Human participants perceive property relations as valid means for approximation. The results show that there is a relation between service similarity and the cardinality of the relation. As far as we know, this is the first time such a relation is reported.

Our results have several implications for designing service retrieval applications. First, the results can be used to create better evaluation benchmarks, which take into account people's perceptions with regard to the precision and recall of retrieval algorithms. Service retrieval similarity is perceived like the analogous notion in the field of information retrieval [25,34]. We believe that this result demonstrates how information retrieval techniques, such as latent semantic indexing, become relevant to service retrieval. Soft similarities measures can also be used to create new types of application for semantic Web service retrieval, including recommender systems, service filtering, and service classification.

Finally, the results provide hints regarding the type of service retrieval applications users might find usable and intuitive. Current service retrieval approaches are based on the intention of being used in a fully automated environment, which prompts crisp notions of similarity. However, the soft notion of similarity found in our study suggests using service retrieval in user-facing applications, including search engines, ranking compositions, and mobile application markets, such as Apple's App Store and Google's Android Market.

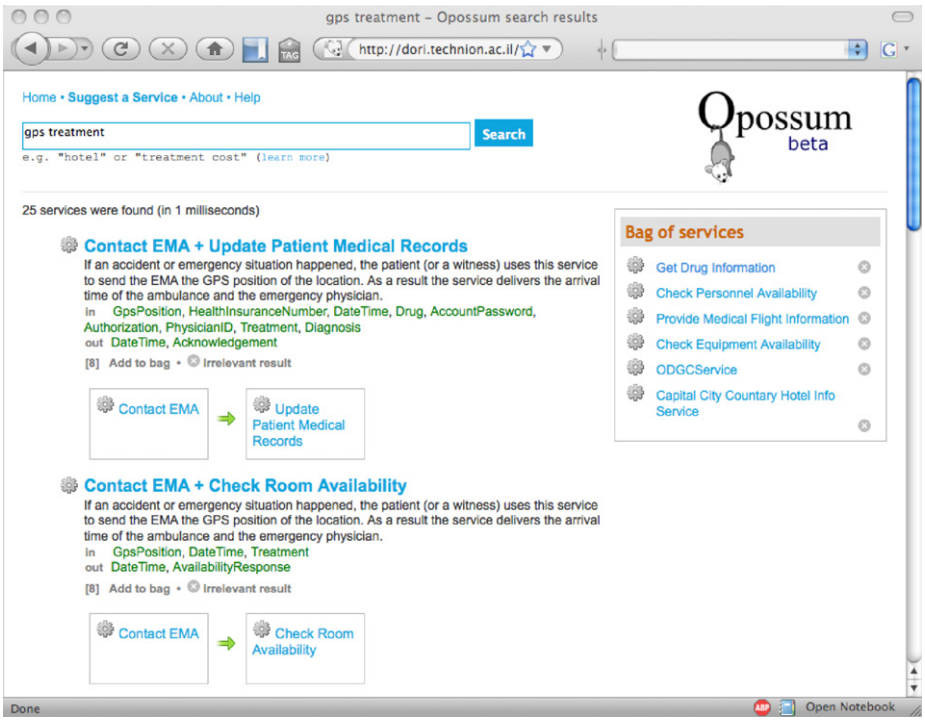

Fig. 11. Compositions ranked and retrieved by the OPOSSUM search engine. The search engine was updated with the semantic similarity patterns defined in this study.

6. Implementation

To provide a proof-of-concept of our model, we implemented parts of the similarity measure defined in this study in an existing Web-based search engine for Web services named OPOSSUM (Object-PrOcedure-SemanticS Unified Matching).[7] OPOSSUM crawls the Web for WSDL and OWL-S descriptions, making them retrievable using simple text queries [41]. OPOSSUM supports the patterns that were analyzed in this study, except for the behavioral pattern.[8]

As Fig. 11 demonstrates, users interacting with OPOSSUM are presented with a GUI that provides them with experience similar to a common search engine. Queries for service retrieval and composition are entered using a query language with which users describe service properties and their order of execution. Results include single operations as well as compositions, and are ranked according to the similarity measure.

The OPOSSUM search engine contains several additional techniques beyond those described in this study, including setting a threshold for similarity retrieval and an internal index for composition representation, a query parser, and a query optimizer. While query response times and time complexity were beyond the scope of this paper, OPOSSUM has sub-linear query response time due to an indexing mechanism.[9] The system is based on MySQL 5.0 as a database server, Apache Tomcat as a Web application server, and the Java programming language.

7. Conclusions

This study provides an empirical and theoretic basis for re-evaluating similarity measures for semantic Web services. As far as we know, this is the first study in which paradigms of ontology-based service retrieval were examined with human participants on this scale. The results enabled us to identify three affinity patterns that capture the essence of similarity between service compositions:

1. Set hierarchy pattern.
2. Relation pattern.
3. Behavioral affinity pattern.

We have proved that the list of affinity patterns is complete, under the assumption that services are fully described using a restricted ontology language. While the first pattern formally defines a notion of similarity already discussed in the semantic Web service composition literature, the two other patterns define new notions of similarity.

We have shown that humans take a more fine-grained approach when assessing similarity than predicted by logic-based approaches. This difference may provide some explanation of the slow adoption rate of automatic service composition paradigms [17]. Our similarity measures exhibit some desired properties, such as explainability, as each similarity-based decision can be analyzed and explained to the user, based on the underlying ontology.

We are currently extending the model suggested in this paper in several directions. First, we work on relaxing some of the restrictions of this study, such as supporting only a single ontology. Second, we work on ranking compositions according to usability and preciseness. Finally, empirical analysis of similarity measures for service retrieval give rise to several interesting research questions that still remain open:

- Investigating how ontology modeling patterns affect service similarity.

[7] OPOSSUM was not used for the user study, and was implemented mainly to provide a proof-of-concept of the theoretical model.

[8] The code of OPOSSUM is distributed under open-source license, and can be downloaded from http://projects.semwebcentral.org/projects/opossum/.

[9] OPOSSUM received the first prize for query response time in the Semantic Service Selection contest (http://www-ags.dfki.uni-sb.de/klusch/s3). At the same contest, OPOSSUM received the 5th place in the average precision results.

- Analyzing how usage scenarios (e.g., ad hoc vs. design-time service composition) and user type (i.e., engineers vs. end-users) affect the perception of similarity.

Acknowledgments

The authors gratefully acknowledge the contribution of the Israel Ministry of Science for the Eshkol grant.

Appendix A. Behavioral pattern completeness

Lemma 1. *The behavioral pattern can compare any two arbitrary graphs.*

Proof. We will prove this property by presenting an algorithm that compares any two arbitrary graphs. We then show that the algorithm is identical to the cost construction estimation on compositions.

Let G_a and G_b be two directed graphs of type $G = \langle V, E, l(V) \rangle$, where V is a finite set of nodes, $E \subseteq V \times V$ is a set of edges, and $l(V)$ is a function that assigns unique labels to vertices. Let us compare the graphs using the following algorithm:

1. Compute the set of common vertices: $V_a \cap V_b$. The set is always computable as V is finite and vertices can be uniquely identified.
2. Compute the set of common edges: $E_a \cap E_b$. The set is always computable as every edge can be uniquely identified by two vertices.
3. Compute the edit distance: $d = |V_a| + |V_b| - |V_a \cap V_b| + |E_a| + |E_b| - |E_a \cap E_b|$.

As the composition nodes (operations) are directed graphs, which are uniquely identifiable using their URL, graph, edit distance is computable on any arbitrary compositions. As the construction cost on composition is the graph edit-distance, we conclude that the behavioral pattern is complete for any arbitrary compositions. □

References

[1] M. Klusch, B. Fries, K.P. Sycara, Owls-mx: a hybrid semantic web service matchmaker for owl-s services, J. Web Semant. 7 (2) (2009) 121–133.
[2] J. Bae, L. Liu, J. Caverlee, W.B. Rouse, Process mining, discovery and integration using distance measures, in: 2006 IEEE International Conference on Web Services (ICWS 2006), 18–22 September 2006, Chicago, IL, USA, 2006, pp. 479–488.
[3] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, C. Pedrinaci, Irs-iii: a broker-based approach to semantic web services, J. Web Semant. 6 (2) (2008) 109–132.
[4] E. Sirin, J. Hendler, B. Parsia, Semi-automatic composition of web services using semantic descriptions, in: Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003, April 2003, [online], available: http://www.mindswap.org/papers/composition.pdf.
[5] D. Wu, E. Sirin, J. Hendler, D. Nau, B. Parsia, Automatic web services composition using shop2, in: Twelfth World Wide Web Conference, 2003, [online], available: http://citeseer.ist.psu.edu/671310.html.
[6] L. Zeng, B. Benatallah, A.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, Qos-aware middleware for web services composition, IEEE Trans. Softw. Eng. 30 (5) (2004) 311–327.
[7] N. Venkatasubramanian, C.L. Talcott, G. Agha, A formal model for reasoning about adaptive qos-enabled middleware, ACM Trans. Softw. Eng. Methodol. 13 (1) (2004) 86–147.
[8] A. Brogi, S. Corfini, R. Popescu, Semantics-based compositionoriented discovery of web services, ACM Trans. Internet Technol. 8 (4) (2008) 1–39.
[9] M. Paolucci, T. Kawamura, T.R. Payne, K.P. Sycara, Semantic matching of web services capabilities, in: International Semantic Web Conference, 2002, pp. 333–347.
[10] M. Klusch, A. Gerber, Evaluation of service composition planning with fowls-xplan, in: WI-IATW '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, Washington, DC, USA, 2006, pp. 117–120.
[11] M. Sabou, J. Pan, Towards semantically enhanced web service repositories, J. Web Semant. 5 (2) (2007) 142–150.
[12] A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng, Daml-s: semantic markup for web services, in: Proceedings of the International Semantic Web Workshop (SWWS), July 13, 2001, pp. 411–430, [online], available: http://www.daml.ri.cmu.edu/site/pubs/daml-s.pdf.
[13] T. Gruber, Towards principles for the design of ontologies used for knowledge sharing, in: N. Guarino, R. Poli (Eds.), Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer, 1993.
[14] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001) 34–43.
[15] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, OWL web ontology language reference, http://www.w3.org/TR/owl-ref/, W3C, W3C Candidate Recommendation, (2004).
[16] X. Dong, A.Y. Halevy, J. Madhavan, E. Nemes, J. Zhang, Simlarity search for web services, in: VLDB, 2004, pp. 372–383.
[17] M. Klusch, Semantic service coordination, in: H.S.M. Schumacher, H. Helin (Eds.), CASCOM—Intelligent Service Coordination in the Semantic Web, Birkhaeuser Verlag, Springer, 2008, ch. 4.
[18] A. Zaremski, J. Wing, Specification matching of software components, ACM Trans. Softw. Eng. Methodol. 6 (4) (1997) 333–369.
[19] T.D. Noia, E.D. Sciascio, F.M. Donini, Semantic matchmaking as non-monotonic reasoning: a description logic approach, J. Artif. Intell. Res. (JAIR) 29 (2007) 269–307.
[20] K. Sycara, S. Wido, M. Klusch, J. Lu, Larks: dynamic matchmaking among heterogeneous software agents in cyberspace, Autonomous Agents Multi-Agent Syst. 5 (2) (2002) 173–203.
[21] C. Kiefer, A. Bernstein, The creation and evaluation of isparql strategies for matchmaking, in: 5th European Semantic Web Conference (ESWC2008), June 2008, pp. 463–477, [online], available: http://data.semanticweb.org/conference/eswc/2008/paper/133.
[22] E. Stroulia, Y. Wang, Structural and semantic matching for assessing web-service similarity, Int. J. Cooperative Inf. Syst. 14 (4) (2005) 407–438.
[23] R. Mikhaiel, E. Stroulia, Examining usage protocols for service discovery, in: ICSOC, 2006, pp. 496–502.
[24] M. Klusch, B. Fries, M. Khalid, K. Sycara, Owls-mx: hybrid semantic web service retrieval, in: Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web, AAAI Press, 2005.
[25] J. Euzenat, P. Shvaiko, Ontology Matching, Springer-Verlag, 2007.
[26] P.D. Turney, Similarity of semantic relations, Comput. Linguist. 32 (2006) 379, [online] available: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0608100.
[27] L. Lee, Measures of distributional similarity, in: Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, 1999, pp. 25–32.
[28] R. Rada, H. Mili, E. Bicknell, M. Blettner, Development and application of a metric on semantic nets, IEEE Trans. Syst. Man Cybern. 19 (1) (1989) 17–30, [online], available: http://dx.doi.org/10.1109/21.24528.
[29] A. Borgida, T. Walsh, H. Hirsh, Towards measuring similarity in description logics, in: Proceedings of the 2005 International Workshop on Description Logics (DL2005), July 26–28, 2005, Edinburgh, Scotland, UK, ser. CEUR Workshop Proceedings, I. Horrocks, U. Sattler, F. Wolter, Eds., vol. 147. CEUR-WS. org, 2005, [online], available: http://www.ceur-ws.org/Vol-147/25-BorgidaEtAl.pdf.
[30] A. Tversky, Features of similarity, Psychol. Rev. 84 (4) (1977) 327–352.
[31] R. Küsters, A. Borgida, What's in an attribute? consequences for the least common subsumer, J. Artif. Intell. Res. 14 (1) (2001) 167–203.
[32] H. Bunke, On a relation between graph edit distance and maximum common subgraph, Pattern Recogn. Lett. 18 (9) (1997) 689–694.
[33] D. Lin, An information-theoretic definition of similarity, in: 15th International Conference on Machine Learning, USA, 1998.
[34] P. Resnik, Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language, J. Artif. Intell. Res. 11 (1999) 95–130, [online], available: http://www.citeseer.ist.psu.edu/resnik99semantic.html.
[35] J. Hau, W. Lee, J. Darlington, A semantic similarity measure for semantic web services, in: Web Service Semantics Workshop 2005 at WWW2005, 2005.
[36] A. Budanitsky, G. Hirst, Semantic distance in wordnet: an experimental, application-oriented evaluation of five measures, in: Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh PA, June 2001, pp. 29–34.
[37] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database (Language, Speech, and Communication), The MIT Press, May 1998, [online], available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09–20amp;path=ASIN/026206197X.
[38] A. Bernstein, E. Kaufmann, C. Burki, M. Klein, How similar is it? towards personalized similarity measures in ontologies, in: 7. Internationale Tagung Wirtschaftsinformatik, February, 2005.
[39] J. Lima, K.-H. Lee, Constructing composite web services from natural language requests, J. Web Semant., 2009.
[40] D. Dori, Object-Process Methodology—A Holistic Systems Paradigm, Springer-Verlag, 2002.
[41] E. Toch, A. Gal, I. Reinhartz-Berger, D. Dori, A semantic approach to approximate service retrieval, ACM Trans. Int. Technol. 8 (1) (2007) 2.