A COMPOSITE MATCHING TECHNIQUE FOR SEMANTIC BASED WEB SERVICE DISCOVERY

Nalaka Gooneratne [1], Zahir Tari [1] & Gregory Craske [1]

[1] School of Computer Science and Information Technology,
RMIT University

ABSTRACT: Current web service composition techniques compose web services based on their functionality. They do not consider user constraints. We propose a semantic based composite matching technique that composes web services according to three types of constraints. Local constraints the apply to individual web services, global constraints that apply to multiple web services, and workflow constraints that apply to the web services of a composition because of the data flows that exist between them. We also propose a workflow model that is used to specify the workflow of the composition and the three types of constraints. The techniques that match local and workflow constraints use the proposed condition matching technique. The global constraints are matched using a three dimensional structure that models the nodes of the workflow, web services that map to the nodes, and the scopes of the conditions of the web services in its dimensions. Using simulation experiments the proposed technique is compared against a syntactical technique and an existing composition technique.

INTRODUCTION

Web services are autonomous and modular applications that are accessed over the Internet. Web service discovery is the process in which web services are located without prior knowledge. When locating a web service a user request is matched to a web service description.

Peer-to-peer matching techniques are used when a single web service description is matched against a single user request. There are two types of peer-to-peer matching techniques. They are syntactic matching techniques and semantic matching techniques. Syntactic matching techniques perform the matching based on data types and keywords [ETW04, WS03]. Semantic descriptions describe the functionality of a web service using ontological terms [ETW04, SKWL99]. These descriptions are also used by users to specify user requests. Semantic based matching techniques perform the matching by relating the semantic descriptions of a web service and a user request using ontological relationships [ETW04, LH03, SKWL99].

With the increasing utilisation of web-based applications, many users make requests that require a composition of web services to satisfy. Furthermore, users may specify constraints that apply to individual as well as multiple web services of the composition. A composition of web services is generated according to a business process that is specified by the user. The process-oriented nature that has to be adopted when composing web services implies that the composition of web services must be generated according to a workflow, which specifies the control flow of the processes. When web services are composed according to a workflow the data flows between the web services of the composition have to be matched. This applies a constraint on the web services that can be selected for the composition and we refer to this type of constraint as a workflow constraint. The constraints that are applied to individual web services of a composition are referred to as local constraints, and constraints that apply to multiple web services are referred to as global constraints. Local constraints and workflow constraints can be matched using peer-to-peer matching techniques, as they only require one-to-one matching. However, global constraints have to be matched using a technique that supports one-to-many matching.

Peer-to-peer matching techniques are not sufficient when a composition of web services has to be matched according to the local, global, and workflow constraints. None of the current composition techniques compose web services according to the above constraints [MBE03, ZBD$^+$03, WPS$^+$03]. We propose a workflow model for representing the business process of a composition along with the three types of constraints. The proposed composite matching technique composes web services according to the local, global, and workflow constraints specified in the workflow model and it is a semantic based matching technique. The technique also includes its own condition matching technique. We propose a three dimensional structure for matching global constraints. In its dimensions the structure models the nodes of the workflow, web services that map to the nodes, and the scopes of the conditions of the web services.

BACKGROUND

The proposed workflow model semantically represents the functionality that is requested by the user along with the constraints. The semantic representation of the workflow model is based on an existing semantic based web services description framework. The above framework is described in this section. The composite matching technique relates the semantic representations of the workflow model to semantic descriptions of web services using ontological relationships. The ontological structures that are used to represent the above relationships are also described in this section. This section also describes the conditions that are used to represent the constraints of the user and the web services.

PILLAR

PILLAR [ETW04] is a framework that is used to produce semantic descriptions of web services. Unlike other web service description frameworks, PILLAR captures the functional aspects of a web service through a goal model (G^+). G^+ captures the goal of a web service through a domain operation that is defined in the meta-ontology, which is described in the next section. Scenarios are used to specify the sequence of operations that can be be performed in order to achieve the goal. The context in which a goal is achieved is specified as a set of context conditions, of the G^+ model.

Meta-ontology

An ontology is used to describe the terms that are used in a domain and to describe the relationships that exist between them. The meta-ontology proposed by Elgedawy et al. [ETW04] separates the terms used in an ontology as concepts, operations, roles, and rules. Concepts are the entities in a domain such as airline ticket, hotel, and flight. Operations are the transactions that occur in a domain such as reserve_airline_ticket(). Roles are the actors of a domain such as travel agent and tourist. Rules are the conditions that apply to a specific domain, such as a tourist must have a valid passport if he or she wants to travel overseas. Each term in the meta-ontology consists of a set of attributes that define its scope.

Ontologies that represent terms in a hierarchical manner, assume that the hierarchical relationships between the terms or the distance between the terms will be used to assess the semantic similarity between them [DMD+03]. This technique is not accurate in all situations. For example the term *car* cannot be substituted by the term *vehicle* in every situation, even though they could exist close to each other in an ontology that is defined using a hierarchical or subsumption structure. Elgedawy et al. [ETW04] propose a meta-ontology structure that enables the description of relationships between terms, as in a hierarchical or subsumption based ontology. It represents the relationships between terms along with the contexts in which they are valid. It uses substitution graphs and transformation graphs to represent the relationships that exist between terms.

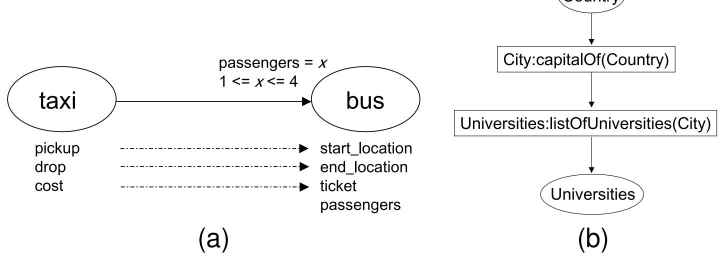


Figure 1: Substitution graph and transformation graph

A substitution graph represents the relationship between two terms of the meta-ontology. The context in which a relationship is valid is specified through a substitution condition. A substitution graph also specifies how the attributes of one term should be mapped to the attributes of the other term when the substitution takes place. Figure 1a is an example of a substitution graph in which the term *bus* is substituted by the term *taxi*. The substitution is shown with the solid lined arrow and the attribute mappings are shown with the dashed lined arrows. The condition above the solid lined arrow specifies the substitution condition. A transformation graph represents a relationship between two concepts of the meta-ontology using operations. Figure 1b is a transformation graph which specifies the relationship between a country and a list of universities in its capital.

Basic condition

A condition is an expression that consists of two operands and a comparative operator and evaluates to either true or false. An operand can either be a term and an attribute defined in the meta-ontology, or a value. A basic condition is a condition in which one operand is a value. *A term of an operand is referred to as a scope as it limits the attributes that can be used in the operand according to the meta-ontology*. Hence, a basic condition consists of a scope, an attribute, a comparative operator, and a value. An example of a basic condition is *credit.limit*<\$500, where *credit* is the scope, *limit* is the attribute, $<$ is the comparative operator, and \$500 is the value.

Context condition

Each web service has a separate context in which it achieves its goal. In the PILLAR framework the context in which a web service achieves its goal is represented with context conditions. Context conditions are the collection of pre-conditions and post-conditions that apply to the web service. For example, if a web service that validates credit cards only validates travel related purchases then it would have a goal *validate_credit_card()* and a context condition *purchase.type = travel*.

WORKFLOW MODEL

The proposed workflow model represents the business process of a composition and incorporates a constraint model. A workflow is defined as a directed graph that consists of nodes and edges. The proposed model contains a schema for representing a workflow along with the constraints that can be applied to a composition of web services. The model also defines a schema for representing a web service (an instance of a node) and a composition of web services (a workflow instance).

Web service : A web service is uniquely identified through an *id*. The functionality of a web service is represented through a goal, which is specified as an operation within the meta-ontology. The context in which the goal is achieved by a web service is specified through a set of context conditions. We define that a web service consists of an *id*, a goal, and a set of context conditions.

Local constraint : Each node of a workflow specifies a functionality that has to be achieved by a web service that maps to the node. This functionality is represented through a goal. The context in which the goal of a node is to be achieved is specified through a set of context conditions. The set of context conditions are referred to as a set of local constraints as they only apply to web services that map to a single node.

If a node and a web service have the same goal and both goals are achieved within the same context, then the web service maps to that particular node. Any web service that maps to a particular node is an instance of that node.

Global constraint : A global constraint applies a constraint to a tuple of web services. Given a tuple of web services $<S_x,...,S_y>$, which map to a tuple of nodes $<N_x,...,N_y>$, if a global constraint applies to the tuple of web services $<S_x,...,S_y>$, we refer to the tuple of nodes $<N_x,...,N_y>$ as the set of nodes affected by the global constraint. The global constraint contains a condition that has to be matched by a tuple of context conditions that belongs to the tuple of web services $<S_x,...,S_y>$. We refer to the condition of a global constraint as a global condition. A context condition can either be a pre-condition or a post condition. Therefore, each global constraint contains a variable, which specifies whether its global condition is a pre-condition, post-condition or both. If the global condition is a pre-condition it is only matched against the pre-conditions of a tuple node instance and if it is a post-condition it is matched against the post-conditions of a tuple of node instances.

In order to compare the tuple of context conditions against the global condition the tuple of context conditions have to be aggregated to one condition. The way in which the tuple of context conditions are aggregated to one condition is specified through an aggregating function. The aggregating function takes a tuple of context conditions that have the same scopes and attributes and returns a single aggregated condition. Hence, each global constraint contains a global condition, a variable that specifies the type of the global condition, a set of affected nodes, and an aggregating function.

Workflow constraint : A workflow constraint is a constraint that applies to a tuple of node instances of a workflow because the data flows that exist between the node instances have to be matched. For example let us assume that N_x is the source node and N_y is the destination node of an edge, S_x is a web service that maps to the node N_x and S_y is a web service that maps to the node N_y. The edge between N_x and N_y implies a constraint that the pre-conditions of S_y have to be matched by the post-conditions of S_x. The above constraint is defined as a workflow constraint. As the workflow constraints apply on the data flows that exist between node instances, the workflow constraints cannot be represented explicitly in the workflow schema. Hence, each node of the workflow contains a reference to the set of preceding nodes. This enables us to derive the data flows that need to be matched when the workflow matching is performed.

Node : A node is a graph construct that is used to create a workflow. Each node specifies a functionality that has to be achieved through a web service of the composition. The functionality is specified through a goal and a set of local constraints. Each node of a workflow is uniquely identified through an *id*. A node contains a set of preceding nodes in order to represent the workflow constraints. We define that a node of a workflow consists of an *id*, a goal, a set of local constraints, and a set of preceding nodes.

Workflow : A workflow is a directed graph that consists of nodes, edges, local constraints, global constraints, and workflow constraints. With our notion of representing workflow constraints we implicitly represent the edges of a workflow in the nodes. Hence, we represent a workflow as a set of nodes and a set of global constraints.

Composition of web services (Workflow instance) : An instance of a workflow is a single tuple from the set of node instances of the workflow. An instance of a workflow that conforms to the local, global, and workflow constraints is a valid workflow instance.

COMPOSITE MATCHING TECHNIQUE

The proposed composite matching technique is used to obtain valid instances of a workflow. The workflow and the web services are represented using the model that was defined in the previous section. The proposed composite matching technique is a semantic based matching technique, and utilises the structures of the meta-ontology to semantically relate conditions.

Semantic based condition matching

The proposed condition matching technique is used to semantically match two conditions. We define that two conditions are semantically matched if the scopes, the attributes, the comparative operators, and the values are matched in the following way. A scope of a condition has a set of scopes that it's related to according to the substitution and transformation graphs in a meta-ontology. The scopes of two conditions semantically match if the above sets of the two scopes contain a common scope. Hence, the scopes of the two conditions would get substituted to the common scope. When the substitution takes place, the attributes of the two conditions get mapped to attributes of the common scope. The attributes of the two conditions match if they are mapped to the same attribute. We define that an attribute of a condition is testable against the attribute of another condition if both attributes match. The comparative operator and the value of a condition are used to specify the range of values that conform to the condition. The comparative operators and the values of the two conditions match if the ranges of values that conform to the conditions have at least one common value.

For example let us match the condition *taxi.cost*<\$20 against the condition *bus.ticket*>\$10 using the substitution graph depicted in Figure 1. The set of scopes the scope *bus* is related to is <*bus*, *taxi*>. The set of scopes the scope *taxi* is related to is <*taxi*>. Both scopes semantically match as the both sets contain the common scope *taxi*. As the common scope is *taxi*, the scope *bus* is substituted by the scope *taxi*. When this substitution takes place the attribute *ticket* is mapped to the attribute *cost*. Therefore, the attributes *ticket* and *cost* match as both attributes map to the common attribute *cost*. The range of values <20 and the range of values >10 match as both ranges have common values.

Local constraint matching

For a web service to map to a particular node, the functionality of the node and the web service has to be the same. For the functionality of a web service to match the functionality specified in a node, the goal and the context conditions of a web service have to match the goal and the local constraints of a node. In the proposed technique, the local constraints of a node are matched by the context conditions of a web service if each local constraint is matched by some context condition. A context condition is matched to a local constraint using the proposed semantic based condition matching technique.

Global constraint matching

The proposed technique semantically matches tuples of node instances that conform to all the global constraints. The technique finds tuples of node instances that satisfy each global constraint individually. Thereafter, the tuples of node instances that are obtained for each global constraint are joined together to obtain tuples of node instances that conform to all the global constraints. The tuples obtained are instances of a partial workflow, as all the nodes of a workflow may not be affect by global constraints.

A tuple of node instances conform to a global constraint, if a tuple of context conditions derived from the tuple of node instances match the condition of the global constraint. When matching a tuple of context conditions against a global condition, the scopes of the context conditions and the scope of the global condition have to be semantically matched. The scopes of a tuple of context conditions and the scope of a global condition match if their scopes can be related to a common scope. We refer to the above scope matching as scope levelling.

We propose a three dimensional structure that is used to implement scope levelling called a scope levelling (SL) cube. An SL cube is generated for each global constraint of a workflow. When generating an SL cube it is assumed that each node that is affected by the global constraint has a set of instances that map to them and that each instance conforms to the local constraints. The set of nodes affected by the global constraint are modelled in the first dimension of the SL cube. The second dimension models the instances of the nodes that are modelled in the first dimension. The third dimension models the scopes and the related scopes of the context conditions of the node instances that are modelled in the second dimension.

For example let us take a global constraints ${}_g Cnt_x$, which has a condition $Cnd({}_g Cnt_x)$. The global constraint ${}_g Cnt_x$ affects the nodes N_1, N_2, and N_3. The instances that map to each node, and the scopes and the related scopes of the context conditions of the instances are given in Table 1. The scope and the set of related scopes of the global condition $Cnd({}_g Cnt_x)$ are the scopes $<Scp_1, Scp_3>$.

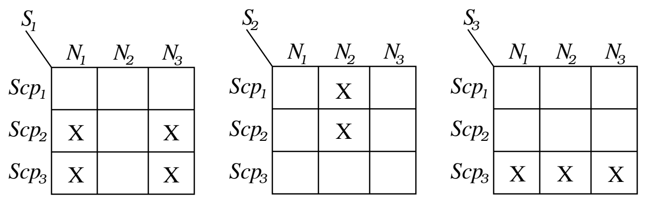
Node	Instances
N_1	$<S_1,S_3>$
N_2	$<S_1>$
N_3	$<S_1,S_2,S_3>$

Instance	Scopes
S_1	$<Scp_2,Scp_3>$
S_2	$<Scp_1,Scp_2>$
S_3	$<Scp_3>$

Table 1: Details of example SL cube

Figure 2 is the SL cube generated for the above example. The X in a particular slot denotes a list of context conditions.

In an SL cube if a tuple of context conditions do not level with the global condition at a particular scope,



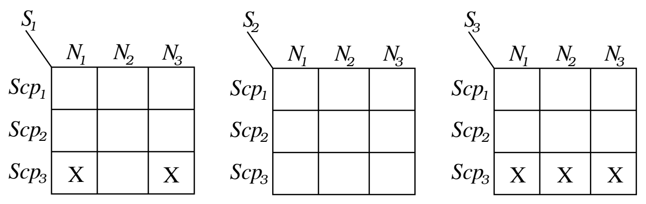
S_1	N_1	N_2	N_3
Scp_1			
Scp_2	X		X
Scp_3	X		X

S_2	N_1	N_2	N_3
Scp_1		X	
Scp_2		X	
Scp_3			

S_3	N_1	N_2	N_3
Scp_1			
Scp_2			
Scp_3	X	X	X

Figure 2: Sliced view of an SL cube

then the entries related to that particular scope are removed from the SL cube. We refer to the above process as levelling. In the above example the entries in the SL cube for the scope *Scp*$_1$ are eliminated as none of the instances of the node *N*$_1$ contain context conditions of which the scopes are related to the scope *Scp*$_1$. The entries related to the scope *Scp*$_2$ are eliminated as the scope of the global condition *Cnd*$({}_g Cnt_x)$ is not related to the scope *Scp*$_2$. Figure 3 displays a sliced view of the SL cube in Figure 2 after it is levelled.



S_1	N_1	N_2	N_3
Scp_1			
Scp_2			
Scp_3	X		X

S_2	N_1	N_2	N_3
Scp_1			
Scp_2			
Scp_3			

S_3	N_1	N_2	N_3
Scp_1			
Scp_2			
Scp_3	X	X	X

Figure 3: Levelled SL cube

Through an SL cube, we obtain a tuple of context conditions of which the scopes level with the scope of the global condition. We then match the attributes of the tuple of context conditions against the attribute of a global condition. The attributes of a tuple of context conditions match the attribute of a global condition, if the attribute of each condition of the tuple of context conditions is testable against the attribute of the global condition. Once the attributes are matched the tuple of context conditions are aggregated into an aggregated condition using the aggregating function that is defined in the global constraint. Then the aggregated condition is matched to the global condition using the defined condition matching technique.

Hence, a tuple of context conditions match the condition of a global constraint if the following can be achieved.

1. The scopes of the context conditions and the scope of the global condition can be levelled.
2. The attributes of the context conditions are testable against the attribute of the global condition.
3. The global condition is matched by the aggregated condition (The aggregated condition is the condition returned when the tuple of context conditions are aggregated using the aggregating function of the global constraint).

If a tuple of node instances contain a tuple of context conditions that conform to the above rules, then the tuple of node instances conform to the global constraint. Sets of such tuples are found for every global constraint of the workflow. Thereafter, the sets are joined to obtain tuples of node instances that conform to all the global constraints. The above tuples are partial workflow instances, as the global constraints may not affect all the nodes of a workflow.

Workflow constraint matching

The proposed technique is used to match an instance of the workflow that conforms to the local, global, and workflow constraints. This technique requires each node of the workflow to have instances that conform to the local constraints mapped onto them, and the tuples of node instances that conform to all the global constraints. An instance of a workflow is generated with a tuple of node instances. Each node instance in the tuple has to conform to the workflow constraints. An instance of a node conforms to the workflow constraints if its set of pre-conditions is matched by a set of post-conditions of the preceding node instances. A set of pre-conditions is matched by a set of post-conditions, if each pre-condition of the set of pre-conditions is matched by some post-condition of the set of post-conditions, according to the proposed semantic based condition matching technique. Such a workflow instance is a valid workflow instance if it overlaps with at least one tuple of node instances that conforms to all the global constraints.

EVALUATION

An implementation of the devised composite matching technique (CMT) was compared against implementations of two other techniques. The first was a syntactical technique that matches conditions by using the scopes as keywords. The second was a technique that only considers workflow constraints, which was similar to the technique proposed by Medjahed et al. [MBE03] (Medjahed). The techniques were compared for effectiveness (precision) and the efficiency (time taken). Currently, there is no standard dataset that is used for evaluating semantic based web service discovery techniques. Hence, a random dataset was generated to perform the experiments.

Figure 4 depicts the results obtained for the experiments that were performed by increaseing the number of web services. Figure 5 depicts the results obtained for the experiments that were performed by increasing the number of nodes.

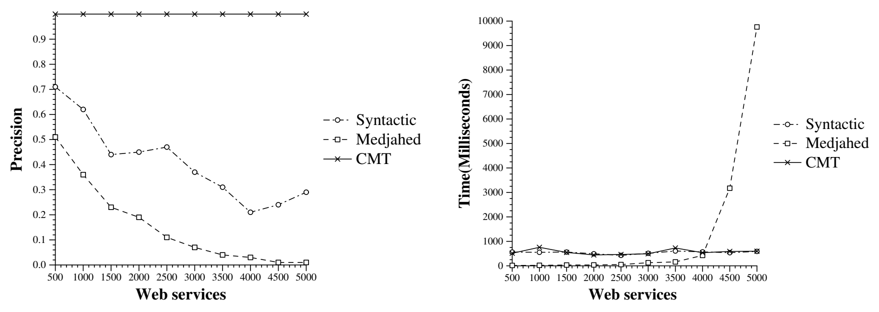


Figure 4: Comparison based on the number of web services

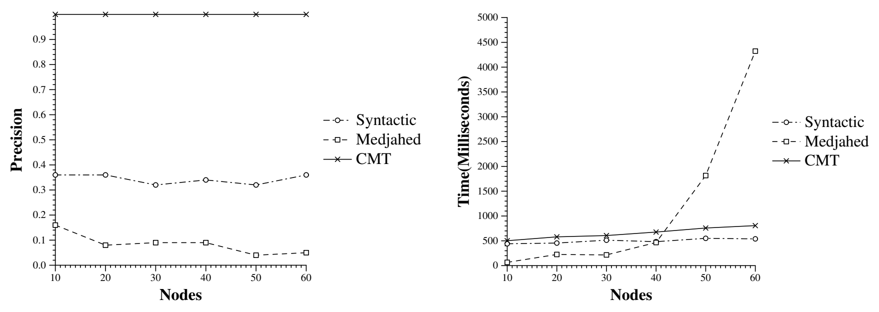


Figure 5: Comparison based on the number of nodes in a workflow

From the experiments that were conducted Medjahed was the least effective in terms of precision, as it matches instances of the workflow that only conform to the workflow constraints (not to the local and global constraints). The syntactical technique is more effective than Medjahed as it considers all three types of constraints. However, it is not as effective as CMT as it only considers the scopes and does not consider the other elements of the conditions. This results in inaccurate conditions being matched by the syntactical technique.

The syntactical technique is the most efficient, as it does not consider the relationships that exist between the scopes through the substitution graphs and the transformation graphs. Also it only matches the scopes of the conditions. Even though Medjahed only matches workflow constraints, CMT is much more efficient when the number of nodes and web service increase. CMT filters out the web services that do not conform to the local and global constraints. This reduces the number of web services that have to be considered for the workflow constraint matching. Medjahed has to consider all the web services when workflow constraints are matched.

CONCLUSION

The work described in this paper proposes an approach for matching a composition of web services according to a workflow. Unlike other composition techniques that exist, it matches a composition of web services according to the user constraints. Three types of constraints that apply to a composition of web services are considered by the composite matching technique; namely local, global, and workflow constraints. We propose a workflow model for representing a composition. Unlike models that are used by other composition techniques, the proposed workflow model represents the control flow of the composition along with the three types of constraints. Simulation results indicate that the devised technique achieves a higher precision when web services have to be composed according to user constraints.

ACKNOWLEDGEMENTS

All the work done in this paper is fully supported by the ARC (Australian Research Council), under the ARC Linkage project scheme no. 0347217.

REFERENCES

[DMD+03] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the Semantic Web. *The VLDB Journal*, 12(4):303–319, 2003.

[ETW04] Islam Elgedawy, Zahir Tari, and Michael Winikoff. Exact functional context matching for web services. *To be published In Proceedings of International Conference on Service Oriented Computing (ICSOC)*, 2004.

[LH03] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the twelfth international conference on World Wide Web*, pages 331–339. ACM Press, 2003.

[MBE03] Brahim Medjahed, Athman Bouguettaya, and Ahmed K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 12(4):333–351, 2003.

[SKWL99] Katia Sycara, Matthias Klusch, Seth Widoff, and Jianguo Lu. Dynamic Service Matchmaking Among Agents in Open Information Environments. *SIGMOD Rec.*, 28(1):47–53, 1999.

[WPS+03] Dan Wu, Bijan Parsia, Evren Sirin, James Hendler, and Dana Nau. Automating DAML-S web services composition using SHOP2. In *Proceedings of the 2nd International Semantic Web Conference*, volume 2870 / 2003, pages 195–210. Springer-Verlag Heidelberg, 2003.

[WS03] Yiqiao Wang and Eleni Stroulia. Flexible interface matching for web-service discovery. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering(WISE '03)*, December 2003.

[ZBD+03] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *Proceedings of the 12th International Conference on World Wide Web*, pages 411–421. ACM Press, 2003.