A Methodology for *e*-Service Substitutability in a Virtual District Environment

Valeria De Antonellis[1], Michele Melchiori[1], Barbara Pernici[2], and Pierluigi Plebani[2]

[1] *Università di Brescia*
Via Branze, 38 – 25123 Brescia, Italy
{deantone,melchior}@ing.unibs.it
[2] *Politecnico di Milano*
Piazza Leonardo da Vinci, 32 – 20133 Milano, Italy
{pernici,plebani}@elet.polimi.it

Abstract. A virtual district is defined as a consortium of independent member enterprises which operate in an integrated and organic way to exploit business opportunities. Such enterprises cooperate to achieve common goals following agreed upon *cooperative processes*.
Considering a cooperative process as a set of activities performed by *e*-Services provided by the different enterprises, this work studies the compatibility among *e*-Services to support the dynamic substitution of failed or modified *e*-Services.
The methodology takes into account both syntactic and semantic analysis, using a domain ontology to match the different terminology used by the different *e*-Service providers.

Keywords: *e*-Service, dynamic substitution, compatibility, virtual districts

1 Introduction

The last few years have witnessed several efforts for the development of service oriented environments with the purpose of supporting the retrieval, composition, and execution of e-Services over the Internet. Research work [1,2,3] has recently focused on methods and tools for presenting an abstract view of internal processes to hide internal details of process execution inside organizations and for service composition [4]. Several research issues are still to be solved concerning service composition using web services [5].

Within the Italian VISPO (Virtual district Internet-based Service PlatfOrm) Project, started in 2001, we aim at developing an environment for cooperative processes based on e-Services composition mechanisms. In particular, the goal is to support the dynamic composition of e-Services considering not only the design of complex e-Services starting from simple e-Services, but also the dynamic substitution of failed or modified e-Services.

This work proposes a methodology to support the dynamic substitution of available *e*-Services in a virtual district. Such methodology is composed by four

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 552–567, 2003.

phases devoted to the clustering of *e*-Services in compatibility classes for possible substitution.

The paper is organized as follows: Section 2 briefly presents the VISPO project, Section 3 the actors and tools involved in our approach. Section 4 presents the main steps that compose our methodology, followed by Section 5 where a running example used for further illustration of our approach is introduced. Finally, Sections 6, 7, and 8 illustrate in detail the methodological phases.

2 VISPO Project

A virtual district is defined as a consortium of independent member enterprises which operate in an integrated and organic way to exploit business opportunities. From a technological standpoint the information systems of such enterprises have to communicate following a process called *cooperative process*.

Within the Italian VISPO project we aim at developing a flexible environment to support the design of service-based cooperative processes and the composition of *e*-Services, by considering not only the design of complex *e*-Services starting from simple ones, but also the dynamic substitution of failed or modified *e*-Services. The structure of the cooperative processes and the modalities of interaction are designed according to the nature of interactions in a virtual district, ranging from completely hierarchical structures to open markets [6].

In VISPO the following models are defined:

- e-*Service Model* (target of this work): characteristics of *e*-Services are specified in their syntactic and semantic aspects;
- *Orchestration Model*: interactions between *e*-Services and their coordination are specified;
- *Business Transaction Model*: the orchestration is defined on the basis of predefined business transaction patterns, which specify the characteristics of interactions between organizations and provided *e*-Services.

In the *e*-Service Model, according to [7], we consider an *e*-Service as a black box that exports a set of methods (operations) to be invoked in a precise order. In particular, an *e*-Service can be defined in terms of:

- **Interface**: specifying the provided operations and exchanged information.
- **Behavior**: specifying the effects in terms of pre- and post-conditions and the order in which the provided operations have to be invoked.
- **Quality of service**: defining *e*-Service features like availability, performance level, and cost.

In this paper we will consider the models for representing interface and behavior of *e*-Services in terms of pre- and post-conditions, which will be presented in subsection 6.1 and 6.2 respectively.

Several languages are available to describe *e*-Services. WSDL [8] is the de facto standard language to describe a Web Service interface. BPEL4WS [9] can

be used to describe the behavior of the *e*-Services and the conversation among them. As for the quality, WSOL [10] is currently available. Furthermore, DAML-S [11] can be used to create a service ontology, also specifying the pre- and post-condition, whereas the UDDI Registry can be used to store references to existing *e*-Services.

Once the *e*-Services that compose a cooperative process are identified, one or more of them could be substituted after being selected, due to one of the following events:

- during execution, the *e*-Service fails or cannot be reached;
- a better service, with respect to quality of service parameters, could be provided by a new *e*-Service;
- a new version of a selected *e*-Service, offered by the same provider is available.

The proposed methodology aims at supporting such substitution cases during the execution time of a cooperative process and without affecting the process behavior.

3 The Approach

Since two *e*-Services can be substituted each other only if the two are compatible, our substitutability approach is based on the idea that the *e*-Service specifications have to be classified in order to track the relationships among them in terms of compatibility. In such a way, given an *e*-Service and using this relationships we can, in a semi-automated way, search for the compatible *e*-Service

Figure 1 shows the reference VISPO architecture of the proposed approach. Two main sets of *e*-Services are considered:

- `District specific e-Services`: oriented to the cooperative processes in the district and provided by the organizations belonging to the virtual district.
- `General purpose e-Services`: realized to be used in several environments and published in UDDI Registry [12].

In particular, we perform *e*-Service identification on the basis of semantic relationships. Such distinction derives from the difficulty to collect both the syntactic and semantic aspects of the *e*-Service not directly managed inside the districts.

For the district specific *e*-Services an enhanced version of UDDI, called VISPO Registry and described later in this work, is mainly used to maintain *e*-Service specifications. Semantic knowledge about services and information contents is organized in the the Domain Service Ontology and Domain Knowledge Ontology.

Two main actors are involved during the design and execution time of the cooperative process: the `Application programmer` and the `Domain Expert`.

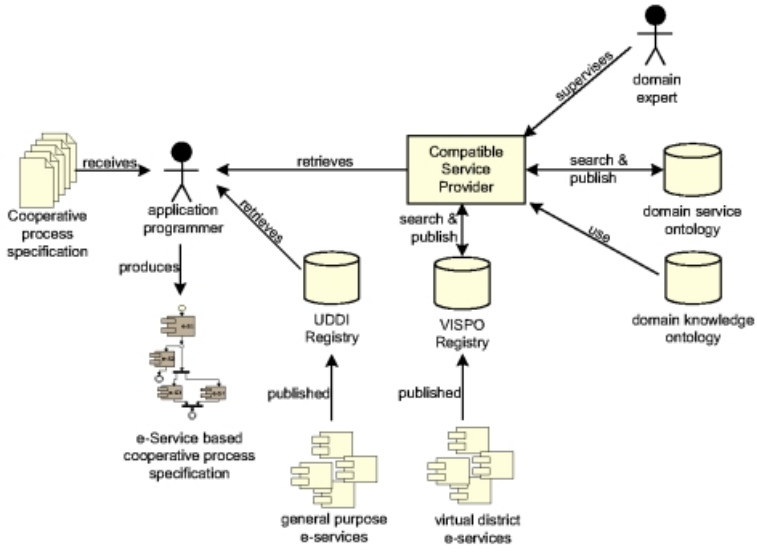

Fig. 1. Reference VISPO architecture

Given a cooperative process specification, the Application Programmer composes the available *e*-Services, in order to satisfy given requirements for a cooperative process. Considering the *e*-Services as black boxes, two different behaviors are to be considered for the cooperative process:

- *Non-observable* behavior: related to the behavior of the composing *e*-Services. Since an *e*-Service is considered as a black box, we cannot modify its internal structure, but only use the provided methods, following its specifications.
- *Observable* behavior: related to the existing cooperation among the organizations which participate in the process.

The objective is to be able to substitute an *e*-Service, when necessary, without affecting the observable behavior of the cooperative process [13].

For this purpose, the Application Programmer is supported by the Compatible Service Provider (CSP). The CSP is a module of the VISPO architecture which, under the supervision of the Domain Expert:

- Allows to publish *e*-Services maintaining the needed classification of *e*-Service specifications in the VISPO Registry and in the Domain Service Ontology.
- Given an *e*-Service specification, allows to retrieve a compatible *e*-Service.

Moreover, the CSP refers to a `Domain Knowledge Ontology` to discover relationships, as for example homonymy or hyperonymy, between terms used in the organizations belonging to the district.

In our approach, two *e*-Services can substitute each other only if they are compatible. In such a way, given an *e*-Service we can, in a semi-automated way, search the compatible *e*-Services.

Once identified the elements needed to discover and store the compatibility relationships among *e*-Services are identified, we can define the *compatibility class* as a set where all the belonging *e*-Services can be substituted with each other. All the identified services are related to an abstract description that represents all the members of the class: this abstract description is called the *abstract service* whereas the members are the *concrete services*.

When problems related to substitutability arise (e.g. a service fails), the failed concrete service can be substituted with another one belonging to the same class according to *mapping information*, a set of rules which have to be satisfied when we substitute one *e*-Service with another.

4 The Methodology

In order to provide to the CSP the needed information to create and use the compatibility classes, using the data sources and tools, four main steps compose our methodology from *e*-Services publication to their invocation:

- *Publication*: the service provider publishes its *e*-Service in a repository. In case of district specific *e*-Services the VISPO Registry is used, instead, in case of general purpose *e*-Services any available UDDI Registry can be used.
- *Classification*: under supervision of the Domain Expert control, a compatibility analysis is performed on *e*-Services stored in the VISPO Registry in order to construct a Domain Service Ontology that classifies *e*-Services according to semantic relationships. The classification is is not performed on general purpose *e*-Services classification because is not feasible to collect syntactic and semantic information of all the *e*-Service published in all the available UDDI Registries.
- *Retrieval*: for district specific *e*-Services we analyze the relationship with the other *e*-Services described into the Domain Service Ontology defined in the previous step. For general purpose *e*-Services, we look for a similar *e*-Service querying the existing UDDI Registry and using a subset of the techniques adopted for the classification of the other kind of *e*-Services.
- *Substitution*: using the mapping information, and under the supervision of the Domain Expert, we substitute the failed service with another one belonging to the same class.

5 A Running Example

The activity diagram in Figure 2 represents a sofa manifacturing process schema, where the `fabric cutting`, `frame creation`, and `sewing and stapling` phases represent the strategic activities that mark the quality of the sofa.

The rest of the activities can be performed by services provided by external organizations, where the procurement service can be used to acquire the basic material and the selected delivery service will have to be able to transport huge elements.

Let us assume that inside the virtual district several *e*-Service providers exist. For example:

- The organizations A and B provide a procurement service.
- The organizations A, C, and D provide a warehousing service.
- The organization E provides a delivery service.

Nearby, a set of general purpose *e*-Services can be used in order to support activities like the procurement of water, or a cleaning service. In our example, the organization F provides a cleaning service that can be used for cleaning of the sofa after assembling and for the enterprise office cleaning.

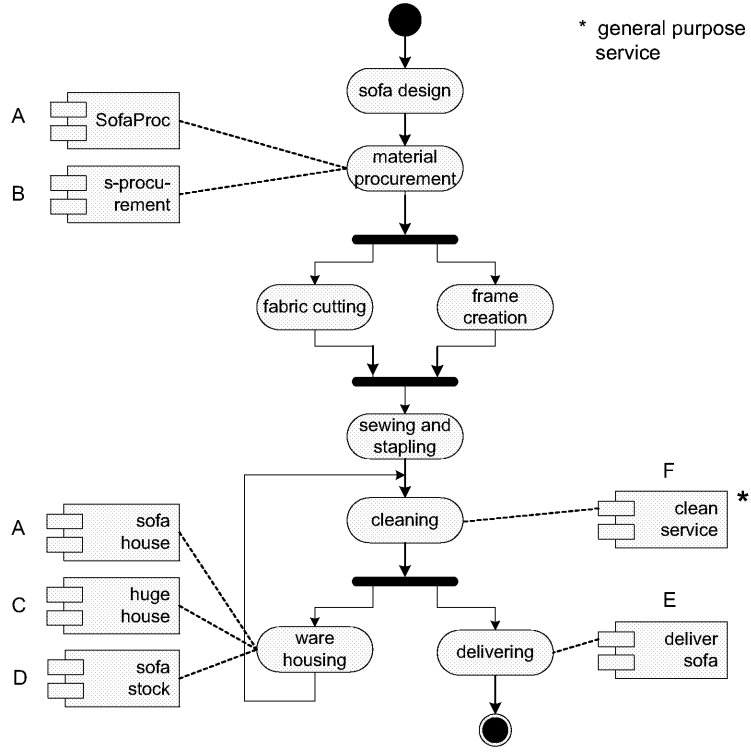

Fig. 2. Activity diagram of the sofa realization process and relationships with available *e*-Services

6 Publication

In order to perform compatibility analysis, we describe an *e*-Service in terms of its interface and behavior with pre- and post-condition. The descriptions are stored in the VISPO Registry.

6.1 *e*-Service Interface

For interface specification, we use WSDL to represent provided operations and corresponding input and output messages. Figure 3(a) illustrates a portion of the WSDL file that represents the `DeliverSofa` *e*-Service provided by the organization E.

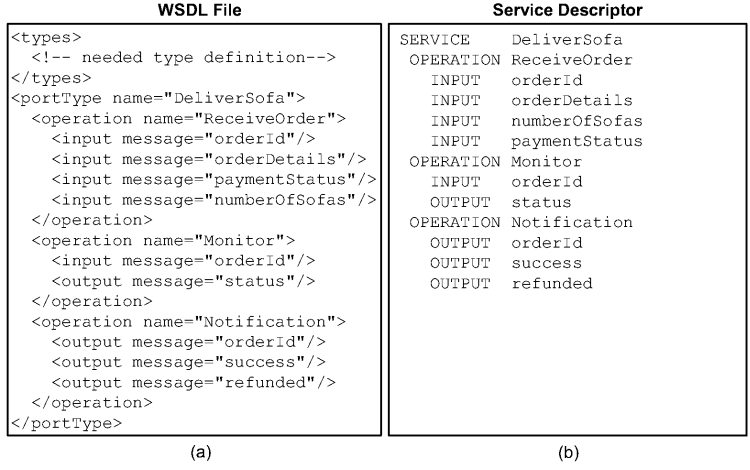

Fig. 3. WSDL example specification part of the *deliver sofa* e-Service

To facilitate compatibility analysis a *service descriptor* is added to the previous representation. Approaches based on the use of descriptors are widely studied in the field of reusable software components [14] for discovering components in a library that match with given requirements. Following a similar perspective, we describe and analyze *e*-Services with respect to the input/output information entities they exchange during the cooperative process execution and to the operations that they are able to perform. In particular, a service descriptor provides a summary, structured representation of the features of an *e*-Service that are relevant for a compatibility based analysis.
A descriptor is formally described as a set of triplets:

$$\langle \text{ operation } (OP), \text{ input entities } (IN), \text{ output entities } (OUT) \rangle$$

to provide the following information about an *e*-Service:

- a set OP of the operations that the *e*-Service can perform;
- a set IN of the input information entities;
- a set OUT of the output information entities.

Descriptors can be automatically extracted from the WSDL file, obtaining for each `portType` the structure depicted in Figure 3(b). This figure presents a possible descriptor of a `DeliverSofa` *e*-Service.

6.2 *e*-Service Behavior with Pre- and Post-conditions

Specification of *e*-Service behavior is obtained through a pre- and post-condition pair that gives a characterization of the *e*-Service semantics and in particular, provides complementary information with respect to the *e*-Service descriptor.

Pre- and post-conditions are logical statements on the input/output parameters of services and constitute a well known means to specify formally the behavior of a piece of software [13]. In our case the pre-condition is the statement to be verified before the execution of the service, whereas the post-condition must be satisfied after the service execution.

As a simple example, for the `DeliverSofa` service it can be required, as a pre-condition, that at least a sofa is available and that the payment for the delivery is valid before initiating its execution. In this case, the pre-condition can be written as: $(numberOfSofas \geq 1 \wedge paymentStatus = OK)$. The post-condition may state the successful delivery or that in the case of failed delivery a refund is required: $(success = YES \wedge refund = NO) \vee success = NO \wedge refund = YES)$.

Since pre- and post-conditions may not always available or specified for a given service, our approach allows analysis of service compatibility also when the descriptor is the only information available on *e*-Services.

According to the service description proposed in DAML-S, pre- and post-conditions can be associated both to the whole service and to the single operations of the service. Currently, we define the service compatibility on the basis of single operation pre- and post-conditions. Future work will extend the model to take into account also the possible orders in which operations of a service can be performed, to obtain more information for the compatibility analysis.

6.3 The VISPO Registry

Since the publication of the general purpose *e*-Services can be performed using a typical UDDI Registry [15], in this section we consider only the district specific *e*-Services.

To store *e*-Service specifications and descriptors, an augmented version of UDDI Registry called VISPO Registry is introduced in the VISPO Project. Figure 4 represents the internal structure of the VISPO Registry where, beside the WSDL references, there is a database that stores the descriptors.

As suggested in [16] the specification of an *e*-Service can be composed by two different WSDL documents linked to the respective `tModel`:

- *WSDL Service Interface Document* which presents only the `type`, `message`, and `portType` elements which define the interface of the *e*-Service without any reference about the protocol used for the invocation.
- *WSDL Service Implementation Document* which specializes the `portType`, through the `binding` element, specifying the protocol used.

In order to cluster *e*-Services in compatibility classes, a WSDL file that represents the abstract service is introduced. Under the supervision of the `Domain Expert` any UDDI entry which represents the *e*-Service can be linked, through the `tModel`, to the abstract service specification.

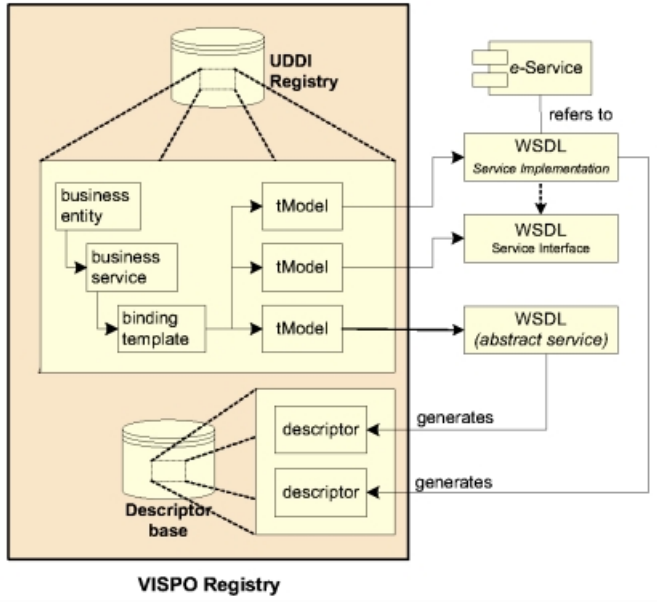

Fig. 4. VISPO Registry architecture

7 Classification

In this section we describe the process of *e*-Service classification and service ontology construction in VISPO, that is based on the analysis of compatibility between services. As noted above, the classification is required to permit the evaluation of *e*-Services compatibility and to facilitate their dynamic substitution in the cooperative process.

To the purposes of describing, maintaining and allowing the access to the classified *e*-Services, we introduce the Domain Service Ontology that organizes *e*-Services according to semantic relationships. We also propose an approach to construct the Domain Service Ontology in a disciplined way, based on the the

available information about the *e*-Services enriched with knowledge provided by the Domain Expert, and to represent the ontology according to DAML-S model.

We think this aspect of our approach is particular relevant since there are several models for service ontologies but there is a little work on how to actually construct them.

In our framework, the Domain Expert designs the Domain Service Ontology on the basis of service descriptions composed of: (i) the service descriptor, that we assume available for each *e*-Service, (ii) the documentation associated with each *e*-Service, and (iii) further characterization of *e*-Services, provided by the Domain Expert, as pre- and post-conditions.

The *e*-Services are organized in the ontology according to three semantic relationships: `equivalent-to`, `is-a` and `similar-to`. These relationships are exploited to permit the browsing of ontology and to support the search for a compatible *e*-Service of a given abstract service. In fact, if we consider two *e*-Services described and semantically related in the ontology, and we need to substitute the first one with the second one in a cooperative process, then the existence of specific semantic relationships between the *e*-Services gives us information about their degree of compatibility.

7.1 Construction of the Domain Service Ontology

Behavior-Based Analysis and Classification

`Equivalent-to` and `is-a` relationships are established between pairs of *e*-Services on the basis of analysis of their pre- and post-conditions and of their descriptors. Description of this analysis is based on the ideas and the terminology of [13] for the software component specification and matching, but introducing some modification and adaptation to make concepts suitable to our context.

First of all, we define "matching properties" between operations. Given two *e*-Services S_i and S_j, we say that operation o_{ik} of S_i has an *OP-exact-match* with o_{jh} of S_j, if: (i) pre-conditions (resp. post-conditions) of o_{ik} and o_{jh} are logically equivalent, (ii) o_{ik} and o_{jh} have the same name and the same parameters. That is, the two operations are equivalent.

Exact match is a strict requirement, so a weaker relation is introduced. Operation o_{ik} of S_i has an *OP-partial-match* with o_{jh} of S_j, if: (i) pre-condition of o_{ik} implies pre-condition of o_{jh}, (ii) post-condition of o_{jh} implies post-condition of o_{ik}, (iii) o_{ik} and o_{jh} have the same name and the same parameters. That is, o_{ik} can be substituted with o_{jh}. Here the rationale is that o_{jh} can substitute o_{ik} in a process since, before the operation execution, satisfaction of o_{ik} pre-condition implies the satisfaction of o_{jh} pre-condition. After o_{jh} executed, the o_{jh} post-condition is true and therefore o_{ik} post-condition is satisfied, as expected by the process logic.

We now can extend the definitions of the exact match and the partial match properties to *e*-Service.

A service S_i has *SERV-exact-match* with S_j (S_i *SERV-EM* S_j) if: (i) they have identical descriptors (that is, they provide the same set of operations with the same parameters), (ii) each operation o_{ik} in S_i has an *OP-exact-match* with

an operation o_{jh} in S_j and viceversa. The meaning is that there is a strong indication that service S_i can substitute S_j and viceversa.

Furthermore, we say that S_i has *SERV-partial-match* with S_j (S_i *SERV-PM* S_j), if: (i) the descriptor of S_j includes the descriptor of S_i (or equivalently, the interface of S_j provides at least the operations described in the interface of S_i), (ii) for each operation o_{ik} in S_i, there is a corresponding operation o_{jh} in S_j, such that o_{ik} has OP-exact-match/OP-partial-match with o_{jh}. That is, for each operation o_{ik}, there is corresponding operation o_{jh} that can replace o_{ik} and this conditions gives an indication that S_i can be substituted by S_j.

The existence of a *SERV-partial-match* among two *e*-Services S_i and S_j gives an indication that we can substitute *e*-Service S_i with *e*-Service S_j but possibly with an adaptation effort because of the not exact match between their corresponding operations. The contribution given to the ontology construction by the behavior-based analysis is the following.

If a *SERV-exact-match* among S_i and S_j is established, a relationship S_i `equivalent-to` S_j is added to the Domain Service Ontology to denote their equivalence according to behavior and interface.

If a *SERV-partial-match* among S_i and S_j is established and there is no *service exact match*, a relationship S_j `is-a` S_i is added to the Domain Service Ontology to denote can substitute *e*-Service S_i with service S_j according to their interface and behavior.

Interface Similarity-Based Analysis and Classification

We describe now a different analysis, called *Interface similarity-based analysis*, that permits to establish further semantic relationships among *e*-Services and that is based on descriptors, and therefore on interface information only. This kind of analysis is particular suitable in the case where pre- and post-conditions are not available at all or are missing for some of the *e*-Services.

In the *Interface similarity-based analysis* starting from the descriptors, *e*-Services are classified according to similarity criteria with respect to information entities and performed operations.

In particular, the *e*-Service classification is based on the computation of the following similarity coefficients performed by the ARTEMIS tool environment [17].

- *Entity-based similarity coefficient.* The Entity-based similarity coefficient of two *e*-Services S_i and S_j, denoted by *ESim*(S_i, S_j), is evaluated by comparing the input/output information entities in their corresponding descriptors.
  In particular, names of input and output entities are compared to evaluate their degree of affinity $A()$ (with $A() \in [0, 1]$). The affinity A() between names is computed exploiting a thesaurus of weighted terminological relationships (e.g., synonymy, hyperonymy). To cover the terminology used in the descriptors two different alternatives are possible in ARTEMIS: to use a pre-existing, domain independent basic ontology, such as WordNet, or to use an hybrid ontology that is a thesaurus containing both terminological relationships extracted from WordNet and terminological relationships supplied by the domain expert.

Two names n and n' of entities have affinity if there exists at least one path of terminological relationships in the thesaurus between n and n' and the strength of path is greater or equal to a given threshold.
The higher the number of pairs of entities, one from the first service and one from the second, with affinity, the higher the value of $ESim$ for the considered *e*-Services.

- *Functionality-based similarity coefficient.* The Functionality-based similarity coefficient of two *e*-Services S_i and S_j, denoted by $FSim(S_i, S_j)$, is evaluated by comparing the operations in their corresponding descriptors. Also in this case, the comparison is based on the affinity $A()$ function.
  Two operations are similar if their names, their input information entities and output information entities have affinity in the thesaurus. The similarity value of two operations is obtained by summing up the affinity values of their corresponding elements in the descriptors.
  The value of $FSim$ coefficient is such that the higher the number of pairs of operations, one from the first service and one from the second, with similarity, the higher its value for the considered *e*-Services.

Finally, a global similarity coefficient $GSim$ for each pair of services S_i and S_j is evaluated by taking a weighed sum of $ESim(S_i, S_j)$ and $FSim(S_i, S_j)$, that is a measure of their level of overall similarity.

The result of similarity-based classification are clusters of similar *e*-Services with defined similarity relationships. A hierarchical clustering algorithm [18] is used to determine clusters based on the strength of global similarity established among *e*-Services. In particular, similarity thresholds can be properly set and experimented in the ARTEMIS tool environment to provide different levels of compatibility under different perspectives.

The contribution given to the ontology construction by the Interface similarity-based is the following.

If S_i and S_j are in the same cluster according to the similarity-based classification, and there is no SERV-exact-match or SERV-partial-match between them, the S_i `similar-to` S_j relationship maintained in the cluster is referred in the ontology.

A particular case of `is-a` relationship is given by two *e*-Services which have in common the same WSDL Service Interface Document. In such situation the *e*-Services will have an high $GSim$ value because they probably differ only for the protocol adopted during the transmission.

Description of the ontology. *e*-Service descriptions and semantic relationships in Domain Service Ontology are formally represented in the DAML-S, that provides a model and a representation language for Web service ontologies. The formal specification of the ontology permits, for example, the access of ontology by DAML-S-enabled agents and the consistency checking. In particular, the DAML-S language is a markup language that extends DAML+OIL [19] with a set of classes and properties for describing services. In particular, a service is described according to different perspectives: (i) a declarative perspective, called

`Service Profile`, for giving information on the service provider, on what functionalities the service provides, and on quality rating information; (ii) an operational perspective, called `Process Model`, where the service is described in terms of operations, parameters, pre- and post-conditions and flow control; (iii) a `Service Grounding` perspective that specifies details, such as protocols and message formats, addressing, etc., on how to access a concrete service.

To the purpose of giving DAML-S representation of our Service Ontology, we notice that descriptions of *e*-Services, in particular descriptors and pre- and post conditions, fit directly in the `Process Model` in DAML-S. Further information can be provided by the Domain Expert to cover also the `Service Profile` and the `Service Grounding` of the service to the purpose of giving a more complete service characterization. The semantic relationships can be easily represented: the `is-a` are naturally mapped into the `subClassOf` construct of DAML-S, while the `equivalent-to` and `similar-to` are not basically included in DAML-S and we need to introduce them explicitly in DAML-S representations of our ontology.

7.2 Definition of Mapping Information

Often a perfect and transparent substitution of a *e*-Service with another *e*-Service is practically impossible; in fact, different *e*-Services use different technologies and the exchanged messages are different in syntax and semantics. So, besides the construction of the compatibility class, it is necessary to define the information, here called *mapping information*, necessary to actually substitute the service with a compatible one.

Mapping information is used to define the transformations needed to pass from the abstract service to the concrete service.

For each operation defined in the abstract service we still use ARTEMIS to identify a corresponding operation into the concrete service. This identification is based on the degree of affinity between the names used in the WSDL descriptions.

It is important to note after this analysis, it can occur that a compatible concrete service provides an operation not requested by the abstract service, or that a compatible concrete service does not provide an operation requested by the abstract service. Whereas in the former case the concrete service can be chosen even if there is an operation that will be never invoked, in the latter one even if the previous analysis does not discover any compatibility, the *e*-Service has to be discarded because we assume that the minimal substitutable element is the *e*-Service and not the operation.

Given two operations with name affinity, the correspondences between their parameters are analyzed. Moreover, for each matching pair of parameters, their data types are matched.

All the information about the correspondence between the operations, parameters and data types constitutes the mapping information used by the VISPO platform for wrapping the *e*-Service after the invocation.

8 Retrieval and Substitution

The relationships defined in the Service Ontology and in the VISPO Registry described above allow to determine the compatibility degree only for district specific *e*-Services and to individuate the compatibility class of an abstract service. The compatibility and substitutability among general purpose *e*-Service is established only on the basis of similarity-based analysis.

Substitutability among district specific e-*Services.* This analysis is strongly based on the Domain Service Ontology. According to the three relationship described in the Service Ontology, different efforts in substitutions are required:

- *equivalent-to* or *is-a* relationships: the matching is exact, so the substitution can be performed automatically but should be validated by the Domain Expert. Each of the methods defined for the substituted *e*-Service has a corresponding method in the substituting *e*-Service. Given a service described in the ontology the compatibility class corresponds to all the services that are equivalent to, or that are descendant in the is-a hierarchy, of the considered service.
- *similar-to* relationship: here the domain expert holds a central role in order to substitute the two compatible *e*-Service. Definition of mapping information is required to facilitate the wrapping of the substituting *e*-Service. The compatibility class of a service is formed in this case of all the service that are related to the given one by *similar-to* relationship.

Substitutability among general purpose e-*Services.* For this set of *e*-Services the a-priori knowledge about their syntactic and semantic aspects is less than the specific district *e*-Services. For this reason the compatibility evaluation can be performed during the substitution time.

After a typical retrieval in the available UDDI Registries, a possible approach can use a subset of the technics described above. In particular, once a possible compatible *e*-Service is found, the descriptors for both, the failed and candidate *e*-Service, are automatically generated. Using the Interface similarity-based approach discussed in the Section 7, we can also identify the semantic affinity and the mapping information.

In the VISPO architecture when substitution problems arise, the Compatible Service Provider, given an abstract service, queries the VISPO Registry and, based on the available compatibility classes, creates a list of compatible services ordered by their compatibility degrees. In such a list, besides the name of the *e*-Services, the mapping information are present.

A second module, called *Invocation Module*, gives in input such list. The Invocation Module selects the best available *e*-Service in the list and wraps it using the mapping information in order to invoke it in the cooperative process. In future research work, such selection will take into consideration also quality of service characteristics for each *e*-Service.

9 Conclusion

In the present work we propose methods and techniques to study the compatibility of services in order to support their dynamic composition inside a virtual district. The *e*-Service interface is analyzed and the behavior is studied focusing on its representation provided by pre-conditions and effects.

Future work will extend the current approach considering the quality of service aspects and also the behavior in terms of execution status. Automatic or semi-automatic adaptation of *e*-Service to the requirements of a given process will be studied, extending the mapping information approach. *e*-Service substitutability in mobile information systems will be studied, with the goal of dynamically selecting the best available *e*-Services at any moment during process execution.

Acknowledgments. The authors thank Enrico Mussi for his thesis work on the realization of the Compatible Service Provider.

This work is supported by Italian MIUR-MURST under the VISPO and MAIS contracts.

Pierluigi Plebani is supported by a grant from Fondazione Silvio Tronchetti Provera.

References

1. Casati, F., Georgakopoulos, D., Shan, M.C., eds.: Technologies for E-Services, Second International Workshop, TES 2001, Rome, Italy, September 14–15, 2001, Proceedings. Volume 2193 of Lecture Notes in Computer Science. Springer (2001)
2. Buchmann, A.P., Casati, F., Fiege, L., Hsu, M.C., Shan, M.C., eds.: Technologies for E-Services, Third International Workshop, TES 2002, Hong Kong, China, August 23–24, 2002, Proceedings. Volume 2444 of Lecture Notes in Computer Science. Springer (2002)
3. Bussler, C., Hull, R., McIlraith, S.A., Orlowska, M.E., Pernici, B., Yang, J., eds.: Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27–28, 2002, Revised Papers. Volume 2512 of Lecture Notes in Computer Science. Springer (2002)
4. Yang, J., Heuvel, W., Papazoglou, M.: Tackling the Challenges of Service Composition in e-Marketplaces. In: 12th International Workshop on Research Issues on Data Engineering (RIDE-2EC 2002), San Jose, CA, USA, February 25–26, 2002, Proocedings. (2002)
5. Hansen, M., Madnick, S., Siegel, M.: Process Aggregation using Web Services. In: Technologies for E-Services, Third International Workshop, TES 2002, Hong Kong, China, August 23–24, 2002, Proceedings. (2002)
6. Colombo, E., Francalanci, C., Pernici, B.: Modeling Coordination and Control in Cross-organizational Workflows. In: On the Move to Meaningful Internet Systems, 2002 – DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002 Irvine, California, USA, October 30–November 1, 2002, Proceedings. Volume 2519 of Lecture Notes in Computer Science. Springer (2002)

7. Mohan, C.: Dynamic E-business: Trends in Web Services. In: Technologies for E-Services, Third International Workshop, TES 2002, Hong Kong, China, August 23–24, 2002, Proceedings. (2002)
8. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. `http://www.w3.org/TR/wsdl` (2001)
9. Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., Weerawarana, S.: Business Process Execution Language for Web Services, version 1.0. `http://www.ibm.com/developerworks/library/ws-bpel/` (2002)
10. Tosic, V., Patel, K., Pagurek, B.: WSOL - Web Service Offerings Language. In: Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27–28, 2002, Revised Papers. (2002)
11. DAML Service Coalition: DAML-S: Semantic Markup For Web Services. `http://www.daml.org/services/daml-s/0.7/daml-s.html` (2002)
12. ARIBA, IBM, Microsoft: Uddi Executive White Paper. `http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf` (2001)
13. Zaremski, A.M., Wing, J.M.: Specification matching of software components. In: Proceedings of the 3rd ACM SIGSOFT symposium on Foundations of software engineering. ACM Press (1995) 6–17
14. Damiani, E., Fugini, M.G.: Fuzzy Identification of Distributed Components. In: Computational Intelligence, Theory and Applications, International Conference, 5th Fuzzy Days, Dortmund, Germany, April 28-30, 1997, Proceedings. Volume 1226 of Lecture Notes in Computer Science. Springer (1997)
15. McKee, B., Ehnebuske, D., Rogers, D.: Uddi Version 2.0 API Specification. `http://www.uddi.org/pubs/ProgrammersAPI-V2.00-Open-20010608.pdf` (2001)
16. Brittenham, P., Cubera, F., Ehnebuske, D., Graham, S.: Understanding WSDL in a UDDI Registry. `http://www-106.ibm.com/developerworks/webservices/library/ws-wsdl/` (2002)
17. Castano, S., de Antonellis, V.: A Schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases. In: International Database Engineering and Applications Symposium, IDEAS 1999, 2–4 August, 1999, Montreal, Canada, Proceedings, Montreal, Canada (1999)
18. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, N.J. (1988)
19. van Harmelen, F., Patel-Schneider, P.F., Ian Horrocks, E.: Reference description of the DAML+OIL ontology markup language. `http://www.daml.org/2001/03/reference (2001)`