Developing Service Oriented Enterprise by Composing Web Services Based on Context

Sodki Chaari[1], Khouloud Boukadi[2], Chokri Ben Amar[3], Frédérique Biennier[1] and Joël Favrel[1]

(1) INSA of Lyon, France, (2) EMSE of Saint-Etienne, France, (3) ENI of Sfax, Tunisia

Summary
Service Oriented Enterprise (SOE) is a dynamic and temporary collaboration between autonomous Web services. Web services are selected and composed to form an inter-enterprise business process. Web services selection is based usually on functional criteria. However, in the case of SOE there is a need to consider more pragmatic criteria such as context parameters. In the present work we propose a Framework for SOE creation which enables the selection of Web services based on contextual parameters. We extend the WS-Policy standard to describe the different categories of context parameters we consider. In order to enhance Web service selection and to improve the manageability of a high number of Web services, we use the community concept in order to gather specific domain Web services. The proposed Framework uses abstract templates called Goal templates to insure an abstract description of required Web services. Goal templates gather both functional and contextual parameters used to select suitable services.

Key words:

Web service, Service Oriented Enterprise, Context, enterprise collaboration.

1. Introduction

The Web has grown from a simple repository and sharing of information to a big platform for service provision. Web services are gradually taking root following the convergence of business and government efforts to making the Web the primary medium of interactions[10, 28]. In addition, the maturity of XML based Web service standards, such as WSDL and SOAP, are driving the great adoption of Web services technologies [16, 30]. This trend is motivating a paradigm shift in enterprise structure: from the traditional single entity to a collaboration of Web services. Web services provided from several autonomous enterprises in different locations will be identified, selected and composed together to achieve tasks, provide information, transact business, and take action on behalf of users.

We refer to these inter-organizational composed services as a Service Oriented Enterprises (SOE). Authors in [14] define the SOE as a dynamic and temporary collaboration between several autonomous Web services which provide collectively a value added service to a final users which can be individuals or enterprises. The SOE is created on demand, by discovering the "right" component Web services that meet the inter-organizational business goals and tailored to satisfy the customers' needs. We refer to this process as Web service composition problem, which requires both inter- and intra-organizational business processes be composed in a coordinated manner.

The XML standards languages such as WSDL, UDDI, RDF [1], DAML-S [33], and OWL-S [25], facilitate the Web service selection and their composition as a Service Oriented Enterprise. However, all these standards allow the discovery and composition of Web services based on syntactic and semantic description [2]. We believe that, based on these two descriptions we are not enabled to achieve a fully automated Web Service composition. In fact, we need to consider more pragmatic rules to make a decision whether a specific Web service can be used as participant in the service oriented enterprise. These pragmatic rules include for example the geographical and temporal constraints imposed by the Web service, the quality of service, cost, payment methods, etc. These pragmatic parameters characterize contexts on which Web services and the fully Service Oriented Enterprise evolve.

We attempt in this present work to enhance web service selection and composition in a SOE scenario with a particular focus on contextual parameters. We present a context-based Web services community framework for service based enterprise collaboration. The final objective of this proposal is to look into the role of context and community in framing the automatic creation of the Service Oriented Enterprise. The main contribution in this paper is summarized in the following points:

i. We extend the WS-Policy framework by adding new components to enable a structured presentation and an easy processing of context policies.

Manuscript received June 5, 2008.
Manuscript revised June 20, 2008.

ii. In order to enhance Web service selection and to improve the manageability of a high number of Web services, we develop a set of communities by gathering related Web services. Every community exposes a set of generic methods related to the Web service domain as well as a set of generic context parameters which are relevant to this community.
iii. We introduce the concept of Goal Template used as an abstract description of an activity in the collaborative business process (the SOE process). It will be considered as a mean to structure the request of Web service regarding to particular activity in the SOE process. This will contain the requirements (functional and context) which must be verified by the selected Web service.

The remainder of this paper is organized as follows. Section 2 suggests a motivating example in order to underline the usefulness of context parameters in the composition process. In section 3, we detail the context categories we propose. Section 4 addresses the problem of context parameters representation and introduces the WS-Policy standard as support to such representation. The Framework for the SOE establishing is exposed in section 5. Next, section 6 gives a non exhaustive overview of related works. Then, some implementation features are presented in section 7. Finally a conclusion and future work are presented.

2. Motivating example

To illustrate the problem, let's consider the case of a stuffed Toy Manufacturer (TM) who receives an order from a large supermarket wishes to order a large quantity of stuffed toys. Providing stuffed toys consists in sewing woolen felt, stuffing polyester fiber filling, adding fabric and buttons according to individual toy designs. Once the order is received, the TM proceeds to establish a collaborative business process in order to fill the stuffed toy order. The collaborative process consists in providing raw material and patterns to the sewing partner. Once the raw materials are received, the aforementioned sewing partner begins manufacturing stuffed animals. In parallel, the TM must procure a supplier for the toy boxes in which the stuffed toys are to be placed. Finally, a firm is selected to pick up the toys from the TM warehouse and deliver them to a specific location.

Assuming that all participants in this value chain have Service Oriented Architecture based processes. Interconnecting the TM, sewing partner, box supplier and deliverer processes induce the composition of corresponding services and consequently the establishment of a Service Oriented Enterprise. A selection phase is required to find suitable services responding to differing requirements. However, finding services that match only the functional requirements of the stuffed toy manufacturer is generally considered as insufficient to guarantee the reliability of the retrieved services. In fact, some contextual information may be as relevant as the required functionality itself. They should be taken into account when selecting Web services. Contextual information can be considered as a decisive criterion which a consumer must evaluate to select a specific service among several equivalent functionalities. For example, considering two sewing services from two providers offering similar capabilities in terms of stuffed toy sewing, the TM would prefer the one having the shorter execution time or the offering better prices. In addition, these constraints can be further defined:

- The TM may prefer that the supplier of boxes be a local enterprise which in some cases can be more expensive than one farther away but offers the advantage of shorted delivery time and without extra travel and perhaps customs fees.
- The TM has some cost constraints where the toys should not exceed 10 euro per item.
- The stuffed toy must be delivered within 10 days to meet a sales deadline for the supermarket.

Consequently, we need to consider more constraints related to the context governing the establishment of the collaborative business processes. Contextual information will be used to make decisions of the adequacy of service with regard to the requirements.

3. Adopting context in the Service Oriented Enterprise

The concept of context has been studied in several fields for quite sometime and there are a number of different definitions and uses of the term context. Context appears in many disciplines as meta-information to characterize the specific situation of an entity, to describe a group of conceptual entities, and to partition a knowledge base into manageable sets or as a logical construct to facilitate reasoning services [5]. Our definition of context follows that of Dey [18] who says that a *context* is "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves*".

The goal of the context-aware applications is to acquire and utilize information about the context to provide services that are appropriate to the particular people, place, time, events, etc. There are several definitions of context aware system. Simply put, *a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task* [23].

Context includes whether implicit input and explicit input. For example, the user related context can be deduced in an implicit way by the service provider such as in pervasive environment using physical or software sensors. Explicit context is determined explicitly by entities involved by the context. Authors in [6] depict that a variety of categorizations of context have also been proposed. In fact, there are certain types of context which are, in practice, more used than others. These contexts' categories are ***location***, ***identity***, ***time*** and ***activity*** used for characterising the situation of a particular entity. Despite the various attempts to develop categorization for contexts, there is no generic context categorization. Relevant information differs from a domain to another and depends on the effective use of this information [29].

The Web service paradigm enforces the vision of customized and context-aware service provisioning. In fact, Web services are conceived as an encapsulated module presenting a set of functionality that can be easily reused, so long as the inputs, outputs, and messaging protocol are conformant with the descriptions that present. However, when we begin to look beyond simple examples, we find that the situation become more complex especially in a Service Oriented Enterprise scenario [24]. To support automated discovery and selection of world-changing services, for example, service descriptions must be unambiguous about what situations will guarantee successful service uses, and what new situations will result from those uses. For some categories of services, service behaviour may vary with time, location, user history, pre-existing contractual commitments, and so forth; descriptions of such distinctions can quickly become complex [23, 24]. In addition, when we look in practical case, many other aspects of services must be taken into account which exceeds the simple services' description. As example, Service orchestrator must take into account ay provider reputation, and recommendations from third parties. In addition, service compositions, to be effective, must consider several resource constraints and interrelationships between service providers. Finally, recovery from failure must be based in a complex set of constraints including QoS constraints, user preferences, policies, etc. Indeed, when considering the full range of service-related activities, it becomes clear that dealing with context is a major challenge, requiring greater expressiveness and reasoning capabilities than are supported by the current widely accepted building blocks of the Web services stack [24].

3.1 Categorizing context in the Service Oriented Enterprise

The categorization of context parameters is important for the development of context-aware Web service environments. Despite the various attempts to develop a taxonomy for context parameters, there is no generic NFP categorization. Relevant information varies from one domain to another and depends on the effective use of this information. We propose a categorization of context parameters in Web service environments. One or more non contextual properties are associated to a Web service. Each property belongs to a certain category which can be either QoS related or business related.

3.1.1 QoS properties category

QoS related properties represent a very important aspect of non-functional characteristics for a Web service. The international quality standard ISO 9000 describes quality as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [19]. QoS encompasses several quality parameters which characterize the behavior of a Web service when delivering its functionalities [11, 15]. We consider three main categories of QoS depending on the behavior of the Web service they characterize:

- Execution: includes the performance parameters which characterize the interaction with the Web service. We consider the following properties [32]:
  - Response Time: Time elapsed from the submission of a request to the time the response is received.
  - Availability: It represents the percentage of time that a service is operating. Large values mean high availability. Small values indicate low availability.
  - Accessibility: It represents the degree that a Web service is able of serving a request.
- Security: Related to the ability of a given Web service to provide suitable security mechanisms. We consider the following parameters:
  - Encryption: the ability of a Web service to support the encryption of messages (received and sent).
  - Authentication: the capacity of a Web service to offer suitable mechanisms dealing with the identification of the invoking party (i.e., service consumer) and allow operation invocation.
  - Access control: Whether the Web service provides access control facilities to restrict the invocation of operation and the access to information to authorized parties.

- Privacy: Includes the ability of the Web service in protecting privacy of the submitted information. we present the following parameters:
  - Privacy policy: indicates whether the Web service presents a privacy policy.
  - Information sharing: indicates whether the Web services shares the collected information with a third parties.
  - Information disclosure: indicates if the Web service asks for an explicit permission to reveal information to third parties.

3.1.2 Business properties category

Business properties are considered as part of context properties related to a Web service. Like QoS properties, they are relevant for differentiating Web services having the same functional characteristics. We consider two main categories of business properties: (i) the environmental properties and (ii) the strategic properties. Environmental properties include location and temporal properties. Strategic properties include the following properties:

- Cost: It represents money that a consumer of a Web service must pay in order to use the Web service, i.e., the cost of operations invocation.
- Reputation: Measures the reputation of the Web services based on user feedbacks [43]. Users are prompted to rate Web services on a scale after the end of a querying session. The reputation corresponds to the average of collected ratings.
- Organization arrangement: Includes preferences and history (ongoing partnerships)
- Payment method: It represents the payment methods accepted by a Web service, i.e. transfer bank, Visa card…
- Monitoring: required for a number of purposes, including performance tuning, status checking, debugging, and troubleshooting [26]. The *monitoring context* offers mechanisms that check the status of a service invocation. It also includes elements for inquiring about the "health" of a service in real time by detecting signs of failure.

3.2 Describing Context categories by ontology

We use an ontology to represent the above context categorization (see Figure 1). Ontology can be defined as a formal explicit definition of a shared conceptualization [21] and has many benefits like knowledge sharing, logic inference, knowledge reuse. From the point of view of Web services and their discovery in distributed registries, ontology of context properties will function as universal dictionaries so that the mechanism and registries share the same interpretation of the terms contained in the WS description.

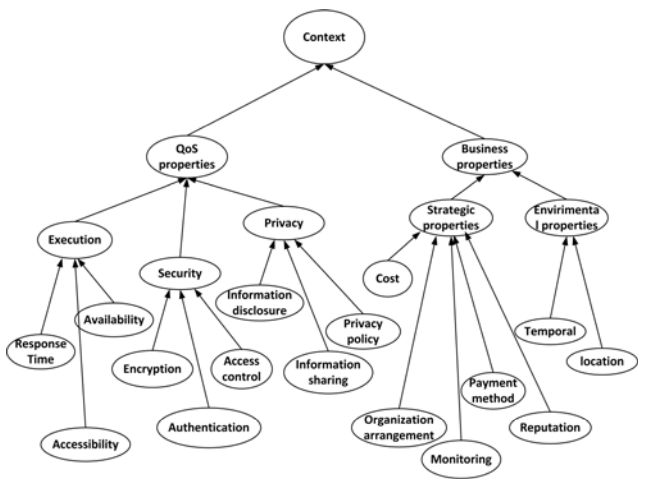


Fig. 1 Ontology for context categories.

4. Using policies to express contexts

Modelling context is a crucial issue that needs to be addressed to assist context-aware applications. In our case, we refer by context modelling the language which will be used to define both Web service and enterprise collaboration contexts. Since, there is a diversity of contextual information, we find several context modelling languages such as *ConteXtML* [34], *contextual schemas* [39], *CxBR* (context-based reasoning) [20], and *CxG* (contextual graphs) [7]. These languages provide the means for defining context in specific application domains such as pervasive and mobile computing. All these representations have strengths and weaknesses. As stated in [9], lack of generality is the most frequent drawback: usually, each representation is suited for only a specific type of application and expresses a narrow vision of the context. Consequently, they present little or no support for defining context in Web service based collaboration scenarios.

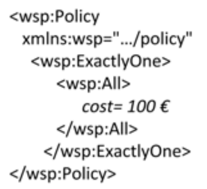
In this work we model the different types of context based on policy. Relation between context and policies is depicted in the definition below:

Definition 1 A Context Cx is a pair (Cx-name, P) where Cx-name corresponds to the context name and P is the policy related to the given context Cx. Let P-set= {P_1, P_2, …, P_n} denotes the set of policies and WSCx= {Cx_1, Cx_2,…, Cx_n} the set of context properties related to a particular Web service, we express the mapping between Web service' contexts and policies with the mapping function CP: WS-Cx$\rightarrow$P-set which gives the policy related to a given context. It can be expressed as $CP(Cx_i) = P_j$ where $Cx_i \in WSCx$ and $P_j \in P$

In the last definition, every context has a corresponding name and will be described with a set of policies. Consequently, to express context we need to express at first policies. To this end, a policy definition language is required. The selection of this language is guided by some requirements that need to be satisfied as reported in [17] expressiveness to support the wide range of policy requirements arising in the system being managed , simplicity to ease the policy definition tasks for people with various levels of expertise, enforceability to ensure a mapping of policy specification into concrete policies for various platforms, scalability to guarantee adequate performance and analyzability to allow reasoning about and over policies. Additional requirements for policy definition languages are also given in [38]. The approach presented in this work uses the WS-Policy Framework to express context policies.

4.1 The basic WS-Policy framework

WS-Policy framework provides a grammar for representing web services' properties based on XML. A policy is defined as a collection of alternatives which is, itself, defined as a collection of assertions. An assertion is used to represent a requirement, capability or a behavior of a Web service [40]. Assertions specify characteristics which are critical for selecting and using the Web services, for instance contextual properties. An assertion can include an arbitrary number of child assertions and attributes. To facilitate interoperability, WS-Policy defines a normal form for policy expressions which is a straightforward XML Infoset representation of a policy, enumerating each of its alternatives that in turn enumerate each of their assertions [3]. A WS-Policy normal form contains the following tags: The "Policy" tag which is used to start and finish a policy expression. The "ExactlyOne" tag which contains a set of alternatives and finally the "All" tag which includes all the assertions in an alternative. The cost property of a Web service can be represented using the WS-Policy as follow (Figure 2):



```
<wsp:Policy
  xmlns:wsp=".../policy"
    <wsp:ExactlyOne>
        <wsp:All>
            cost= 100 €
        </wsp:All>
      </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 2 Example of the extended WS-Policy expressing a cost policy.

However, such representation using the original specification of WS-Policy presents several difficulties especially in the processing of the policies. Consequently the matching and intersections processes between two policies will be difficult. To overcome this shortage, we introduce some new components in the WS-Policy [13] as presented in next section.

4.2 Main components of extended WS-Policy

WS-Policy was written with some flexibility in such a way that assertions are domain independent. Hence, according to the domain, the form and specification of the assertion contents can be defined. In order to address the context-based policy, we suggest extending WS-Policy specifications with a super set of elements to enable an automatic parsing and reasoning over policies. For this reason, we propose using an ontology-based model to specify these elements as illustrated in Figure 3. We introduce concepts as well as relationships among them to represent context policies. Below, we describe the various elements which constitute the extended policy:

- *Policy*: represents the root WS-Policy element and indicates a policy expression, which is an XML representation of a policy;
- *Name*, *Id*: identify a given policy within the WS-Policy;
- *PolicyCategory*: each policy belongs to a specific policy category corresponding to a particular context attribute;
- *PolicyReference*: enables assertion reusing across policies. Then, the content of a policy may be included into another policy;
- *Service*: describes details of the service for which the policy has been specified. A capability policy includes this element in order to specify some details corresponding to the service type to which the policy is applied;
- *PolicyOperators*: the *ExactlyOne* and *All* elements are WS-Policy policy operators. The *ExactlyOne* operator is a collection of policy alternatives and the *All* operator gathers policy assertions into alternatives;

The context policy assertion is composed of the following attributes:

- AssertiontType: indicates whether the assertion is a capability or a requirement. This Type enhances the match-making process. In fact, in normal cases the matching process considers all assertions as capabilities and requirements and takes one-to-one matching. This complicates matching and makes the process time consuming. The AssertionType seeks to facilitate the matching process;
- *MatchingType*: indicates the type of reasoning that can be used regarding the assertion. There are two types:

- The first type is "xsd", denoting any data type supported by the XML schema such as string and float. In this case our reasoning about this assertion compatibility must take into account the manipulation of such data.
- The second type is "ont," indicating that a semantic reasoning can be used when matching this assertion. When defining assertion, the policy manager, based on its domain knowledge, can assess if reasoning using transformation rules can be applied to the assertions. For example, an administrator defines two policy assertions concerning the "Execution Time" and "Network Time." But, he or she knows that the sum of these two QoS attribute values can produce another QoS attribute: the "Response Time" attribute. The administrator indicates that semantic reasoning can be applied by setting attribute of the *MatchingType* to the "ont" value in the assertion definition;

▪ *ServiceOperation*: an assertion is related to a particular Web service operation. In some cases the assertion may be applied to a particular attribute in the service operation;

▪ *Expression*: guarantees an easy parsing and facilitates automatic comprehension and handling of assertions. This element contains sub-elements: the *Parameter* sub-element represents the context parameter which will be expressed by the assertion, such as the cost; the *Value* sub-element is related to the supported assertions; the *Unit* sub-element corresponding to a measure unit of metrics related to the context attribute; and the *Operator* sub-element is used in the assertion expression to represent relationship between context attributes and values.

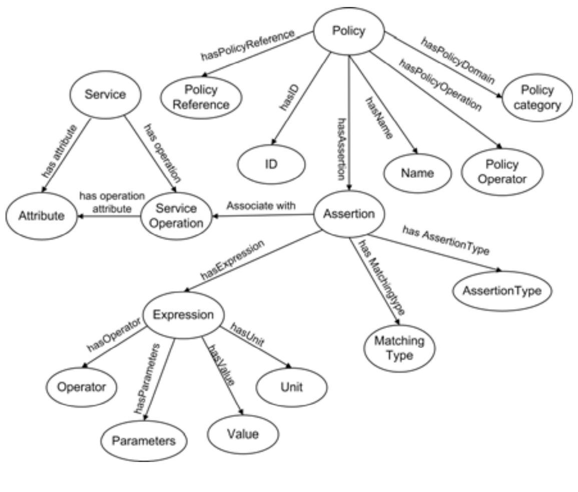
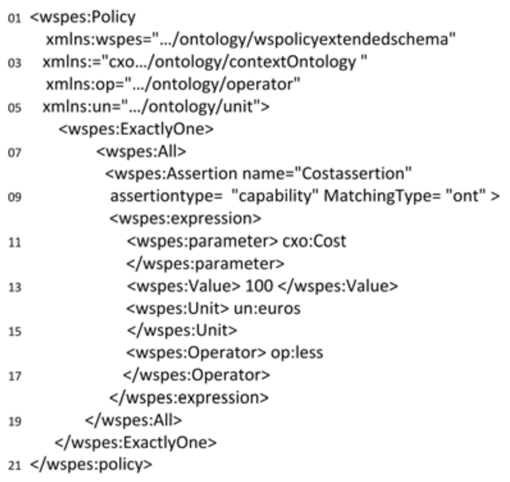


Fig. 3 WS-Policy ontology.

The example illustrated in Figure 4 defines a context policy based on the extended WS-Policy. This example addresses the cost property. Lines (02-05) show name-spaces: *wspes*, *cxo*, *op*, *un* which correspond respectively to the name-space URIs of the web service extended policy, the context categories Ontology, the Unit and Operator Ontologies. The line (08) indicates the name of the cost, (*CostAssertion*). Its expression is defined between lines (11) and (18) which indicate that the cost must be less than 100 euros.

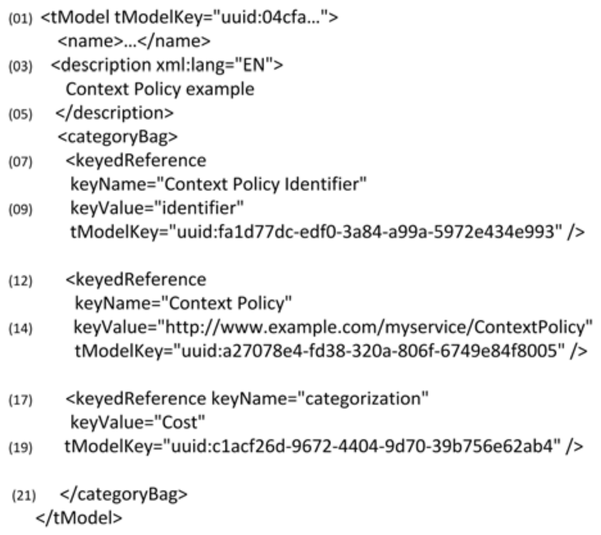


```
01 <wspes:Policy
     xmlns:wspes="…/ontology/wspolicyextendedschema"
03   xmlns:="cxo…/ontology/contextOntology "
     xmlns:op="…/ontology/operator"
05   xmlns:un="…/ontology/unit">
       <wspes:ExactlyOne>
07           <wspes:All>
               <wspes:Assertion name="Costassertion"
09              assertiontype=  "capability" MatchingType= "ont" >
                <wspes:expression>
11                 <wspes:parameter> cxo:Cost
                   </wspes:parameter>
13                 <wspes:Value> 100 </wspes:Value>
                   <wspes:Unit> un:euros
15                 </wspes:Unit>
                   <wspes:Operator> op:less
17                </wspes:Operator>
                </wspes:expression>
19          </wspes:All>
       </wspes:ExactlyOne>
21 </wspes:policy>
```

Fig. 4 A context-based policy example.

4.3 Publishing context WS-Policy

In order to be used in the selection process, context parameters specified as WS-Policy must be attached to Web service and published in a registry. In our work, we use the UDDI registry for publishing Web services and their attached WS-Policies. For this purpose we adopt the same mechanisms introduced in the WS-PolicyAttachment to attach policy expressions to UDDI [42]. We register context policies of a given Web service as distinct tModels. tModel is a UDDI concept introduced for defining technical fingerprints (or specifications) of Web services. As illustrated in Figure 5, the *tModelKey* attribute (line 1) in the tModel tag represents a unique identifier generated by UDDI to refer to the tModel. This identifier also represents the context policy identifier presented by the tModel. The first *keyedReference* (lines 7-10) corresponds to the service identifier on which context constraint will be attached. The second *keyedReference* (lines 12-15) represents the URL of XML file corresponding to the context policy. The third *keyedReference* represents the category for the context policy (lines 17-19). Its value acts like an index which refers to the context policy attribute in UDDI.



```
<tModel tModelKey="uuid:04cfa...">
  <name>...</name>
 <description xml:lang="EN">
   Context Policy example
  </description>
  <categoryBag>
   <keyedReference
    keyName="Context Policy Identifier"
    keyValue="identifier"
    tModelKey="uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993" />

   <keyedReference
    keyName="Context Policy"
    keyValue="http://www.example.com/myservice/ContextPolicy"
    tModelKey="uuid:a27078e4-fd38-320a-806f-6749e84f8005" />

   <keyedReference keyName="categorization"
    keyValue="Cost"
   tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />

  </categoryBag>
</tModel>
```

Fig. 5 Publishing context policy in the UDDI tModel.

5. Model for Service Oriented Enterprise creation

In this section, we present our model for Web service based enterprise collaboration. We introduce a key concept in our proposal which is the idea of Web service community. Web service community supports the dynamic discovery and selection of Web services which will participate in the collaborative business process.

5.1 Community support for Web service discovery

A *community* is a "container" which clusters Web services based on a specific area of interest. All Web services which belong to a given community share the same area of interest. Communities provide descriptions of desired services without referring to any actual service [36].

Communities are defined by *community providers* (see Figure 6). For example, they can be a domain specific consortium such as a group of organizations which contributes in a particular marketplace or simply an administrator.

A community *C* is formally defined as a tuple

C= < *Id-Community*,*Cg*, *WSS*, *GM*, *GCx*>, where:

- *Id-Community* is the identifier of the community;
- *Cg* represents the category of the community. Examples of categories include Delivery, Payment, etc;
- *G-Method* is a set of Generic Methods proposed by the community C. Generic Methods are "abstract" methods that summarize the major functions offered by a community. Community providers define generic methods based on their expertise on the corresponding business domain. The term "abstract" means that no implementation is provided for generic method. Community providers present only an interface description for every generic method. This interface could subsequently be used and implemented by a concrete Web service.
- *WSS* is the set of Web services members of the corresponding community. Being members of a community, Web service promise that they will support one or several of the community's generic methods.
- *GCx* is the set of generic context elements which the community is sensitive to. Like generic methods, generic context parameters are defined by community provider. Examples of context elements include QoS, location, price, etc.

Our approach for creating communities is based on using functional ontologies that capture the relevant methods, their inputs and outputs in a domain. In addition, contextual parameters are added to a community based also on context parameter ontology defined previously.

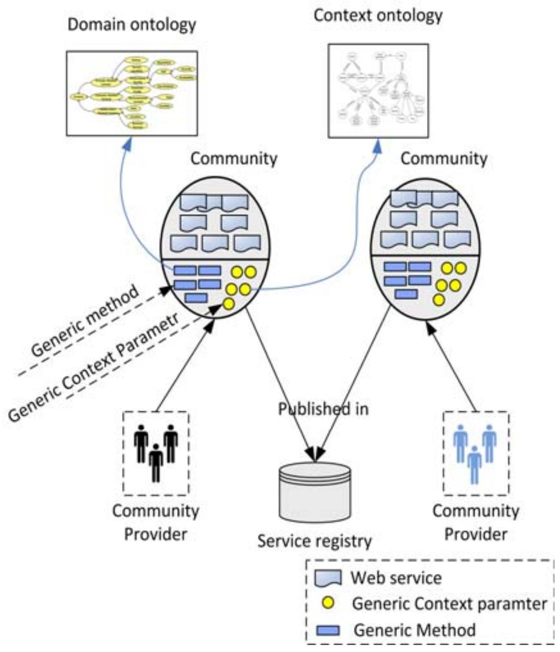


Fig. 6 Community for Web service.

5.2 Collaborative business process Construction

The business model, presented here, is based on the concept of collaborative process specification also referred to as Abstract Process. An abstract process represents a Web service based collaboration process whose control and data flow are defined, but the concrete Web services are not selected until a later time. The advantage of this approach is that complexities in control and data flow can be collected using a manual approach and Web service discovery and selection can be automated using discovery mechanisms based on functional and contextual parameters.

We identify two steps for dynamic collaborative process construction: (i) collaborative business process schema creation and (ii) Web service discovery that we detail heretofore.

5.2.1 Collaborative Process Schema

A Collaborative Process Schema (CPS) is composed of two basic sections: (i) Goal Template which captures the capabilities of the requested service and (ii) description of the flow with special type of activities called synchronization Activities. Figure 7 presents an example of a collaborative process schema.

Definition 2 (Collaborative Process Schema). A Collaborative Process Schema is a tuple

$CPS = < GTs, C, L >$ where:

GTs is the set of goal templates, C is the set of connectors, $L \in (GTs \cup C) \times (GTs \cup C)$ is the set of control links which describes the connections among goal templates and defines the flow structure, which is the sequencing of goal templates within a schema.

Definition 3 (Goal Template). A Goal Template (*GT*) is defined as a pattern used to identify suitable candidate Web services which can be involved in the collaboration process. A goal template represents the requirements of a service requestor and is formally defined as:

GT= <*Id-Template, Id-Community, IGM, ICx*> where:

- *Id-Template* is the identifier of the Goal Template,
- *Id-Community* represents the community identifier in which the goal template is related.
- *IGM* represents the set of instantiated general methods where $IGM \subset GM$.
- *ICx* corresponds to the set of instantiated general context parameters where $ICx \subset GCx$.

The concept of goal template is introduced to help in the Web service discovery process. This describes abstract Web services in specific domain.

Our approach for creating Goal Templates is based on using communities which detail the relevant generic methods, their inputs and outputs in a specific Web service sector. Generic methods inherited from the community are instantiated with relevant parameters related to the collaboration process request. Besides, a Goal Template defines also the set of context parameters which must be verified by the candidate Web services.

A key issue that affected our definition of Goal Templates is the notion of treating Web services as a set of indivisible units of functionalities and contextual parameters. Consequently, to discover the requirements of a given Goal Template, we must find a Web service which provides suitable methods with appropriate set of contextual parameters.

Definition 4 (Synchronization Activity). A synchronization activity can be:

- And-Join activity: a point in the process where two or more parallel executing activities converge into a single common thread of control.
- Or-join activity: a point within the process where two or more alternative branches converge to a single common activity as the next step in the process.
- And-fork activity: point within the process where a single thread of control splits into two or more parallel activities.
- OR-fork activity**:** a point within the process where a single thread of control makes a decision upon which branch to take when encountering multiple alternative process branches.
- Loop activity: a cycle in the process involving the repetitive execution of one (or more) service(s) until a condition is met.

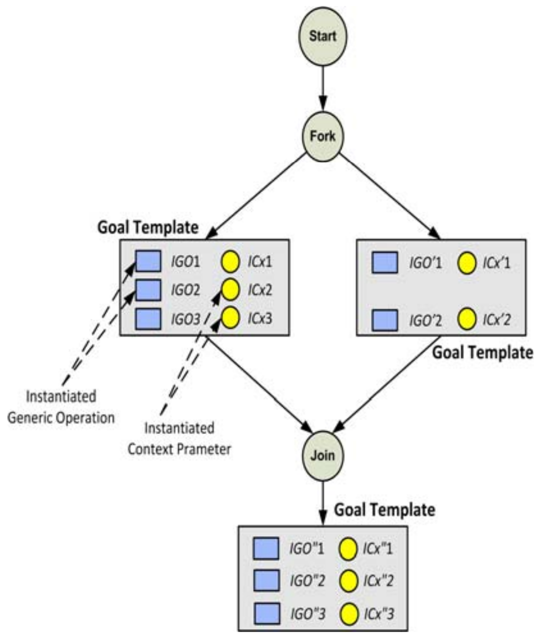


Fig. 7 Collaborative process schema based on template.

5.2.2 Web service discovery

Web Service discovery process is based on the requirements defined using the Goal Templates to find candidate services for this cooperative process.

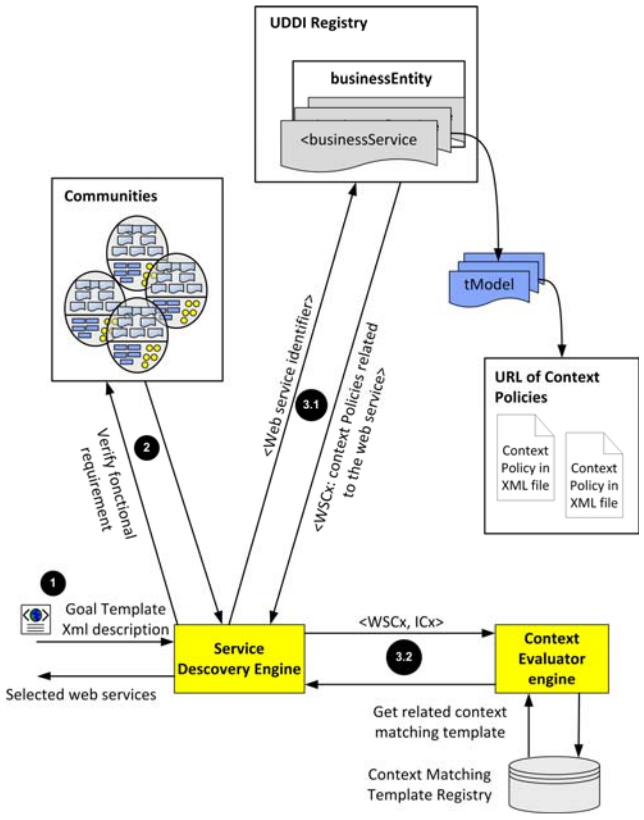

UDDI Registry
businessEntity
<businessService
Communities
tModel
URL of Context Policies
Context Policy in XML file
Context Policy in XML file
<Web service identifier>
3.1
<WSCx: context Policies related to the web service>
2
Verify fonctional requirement
1
Goal Template Xml description
Service Discovery Engine
Selected web services
<WSCx, ICx>
3.2
Context Evaluator engine
Get related context matching template
Context Matching Template Registry

Fig. 8 Discovery framework overview.

The cornerstone of our framework for Web service discovery (see figure 8) is the Service Discovery Engine (SDE). The SDE takes each Goal Template defined in the Abstract Process creation phase and tries to project the requirements fixed in the Goal Template (GT) in the actual Web service set.

SDE offers two operations. The first is the *findWebService* operation. This takes as input the XML file describing the Goal Template, then returns a list of candidate Web services from the Web service registry (UDDI) fitting the GT description. The second is the *updateCommunityList* operation used by community providers at the community creation time in order to update SDE's list of available communities. Each entry in *updateCommunityList* contains the service community ID (*Id-Community*) and the community category.

The discovery process is a 3-step process:

Step1**:** (Identify Web service community). When receiving the XML description of the Goal Template, the SDE determines the category of the community related to the template.

Step2**:** (Identify Web services fitting functional constraints). Based on the instantiated generic methods described by the template, the SDE searches in this community the Web services which match these functional requirements. This step leads to a primary list of Web services.

Step3**:** (Match Goal Template contextual constraints with Web service context parameters). This step consists in verifying contextual constraints expressed by the template. It is composed of 2 sub-steps:

- ***Sub-step3.1****:* SDE gets the set of context policies related to every Web service from the primary list identified in the previous step 2. For this purpose it sends the service identifier to the UDDI registry which checks all existing tModels and returns the set of context policies related to this Web service.
- ***Sub-step3.2*****:** This sub-step consists on matching instantiated generic context parameters with the extracted context policies of each Web services. The SDE sends these two sets of context constraints to a Context Evaluator Engine (CEE). Determining if two contexts are equivalent relies on Context Matching Template (CMT). The CEE has its own set of CMT. Based on the category of the context category, it triggers the suitable CMT stored in the CMT base. The syntax of context matching template is given below:

```
Context Matching Template CMT-name
  Context category category name
  Constraints source policy, member policy
  Action matchContextPolicies (source policy,
                                member policy)
```

The context matching template represents an equivalency criterion related to a specific context category. The CMT provides mechanism to compare correspondence between two context policies. The Action field in the CMT corresponds to a matching function called *matchContextPolicies* which specifications are related to context policy category and to the community in which the Web service belong. For example, in some cases, this function uses a simple syntactic comparison of context constraints and in other cases it uses transformation rules as mentioned in [41] before comparing context constraints.

This step returns a list of Web services which match the Goal Template instantiated generic context.

6. Related work

Web service composition is a very active research and development areas. In this section, an overview of major techniques related to our approach is suggested. We will introduce some relevant works in dynamic Web service composition techniques. Depicted also are important work in context-aware Web service.

6.1 Service composition approaches

Many researchers have worked on Web service composition using disparate approaches, which vary from manually composing services using either GUI based tools or AI planning based techniques to automatically compose the service. Next, we survey some relevant works in the in e-business collaboration field.

Medjahed et al. [27, 28] use semantic Web technologies for the automatic composition of Web services which are semantically described in terms of non-functional properties such as their purpose, category and quality. A "composability" model is defined, for comparing syntactic and semantic features of services, verifying if they can be composed. The client request is expressed in the Composite Service Specification Language, and delineates the sequence of desired operations that the composite service should perform. Thus, given the client request and a set of available atomic services, they develop a technique for semi-automatic composition. Finally, the composite service is translated into an orchestration language that is based on a mediated architecture.

Orriëns et al. [31] address the issue of service composition, where services are atomically represented as Web components. As a composite service is constituted of a set of (possibly composite) services, also a Web component can be constituted of other Web components, manually "glued" together by the so-called composition logics. The authors have also developed the language SCPL (Service Composition Planning Language), for representing a composite service, by specifying relations among Web components, in terms of the execution order (either sequential or concurrent) of Web components within composition, or dependencies among input and output parameters.

Thakkar et al. [37] suggest a dynamic AI approach for Web services composition based on mediator. The composition algorithm assimilates as input the set of available services modelled as data-sources, and a user query, expressed in terms of inputs and requested outputs provided by the user. The output is an integration plan that can be executed by a streaming, highly parallel execution engine called Theseus. Moreover, a mediator system as described, accepts a user query and returns a URL of the new dynamically composed Web services that can answer a class of user queries similar to the original user query.

Several dynamic service composition systems have been proposed and implemented, such as SELF-SERV, e-Flow and so on and so forth. In [4], authors present a framework SELF-SERV to compose declarative Web services. The resulting services can be executed in a decentralized manner within a dynamic environment. The service selection approach uses a local selection strategy. Although the service selection is optimal locally, they may not satisfy the overall constraint. E-Flow is an e-business service composite system designed by the HP lab [12]. It is a platform for specifying, enacting, and monitoring composite Web services. Composite Web services are modelled as business processes, enacted by a service process engine. Although this supports Web service management and deployment, however Web services providers must register, and there is not a fully supported Web services environment.

The works being presented previously focused on simple cases of Web service composition which can't be compared to an inter-enterprise collaboration. Indeed the majority of work present methodologies for an automatic Web service composition which do not take into account the complexity of the collaborative process in terms of the control flow. To overcome this shortage our approach is based on a manual collaborative process modelling (abstract process) and offers the suitable mechanisms to transform the abstract process into a concrete composition of Web services. Finally, the previous works have taken into account only the functional parameters of Web services in the selection process. Nevertheless, pure functional descriptions of Web Services are insufficient to develop valuable Service Oriented Enterprise. Consequently, we have proposed an approach which combines functional and non-functional parameters (contextual parameters) in the composition process.

6.2 Context-aware Web service

The need to integrate context-aware in Web service field has been underlined in different studies, and recently several context-aware approaches have been proposed to enhance Web service discovery and composition mechanisms.

Martin [24] discussed different kinds of knowledge that can be considered contextual with respect to Web services tasks and challenges. In addition, he demonstrated how several projects had made contributions to the understanding of the role of context. Also, he gave the main directions that are needed in Web service research to effectuate more efficient handling of the wide range of contextual knowledge that may be incorporated in future Web services-based systems.

Broens et al. [8] discuss the shortcomings of existing methodologies for service discovery, and propose a interesting approach to overcome some of them. They consider the available contextual information about a particular user or service provider (e.g. user location or service opening times). In addition, they use ontologies to semantically express user queries, service descriptions, and contextual information. Nevertheless, the proposed approach takes into account only Web service discovery, and does not consider contextual information necessary for Web services composition.

Sattanathan et al. [35] use three types of context (I/W/C-context): (i) I-context refers to a Web service instance context, (ii) W-context is the Web service context that is defined by means of I-contexts, and (iii) C-context is the composite service and is defined by the respective W-contexts. The authors focus on the context reconciliation among different Web services. The importance of a language for context specification and management was also stressed, for example OWL-C language.

Maamar et al [22] offer a context-based multi-type approach for Web services composition. Context is used as a support for the development of adaptable and flexible Web services, and policies are used to specify the behaviour of Web services which will participate in the composite service. The binding among Web services occurs at four levels: component level to deal with Web services' definitions and capabilities, composite level to address how Web services are discovered and combined, semantic level to handle the semantic heterogeneity that can arise among Web services, and resource level to focus on the performance of Web services. Due to the composition complexity, the approach uses three types of policies: *engagement* assesses a Web service participation in a given composition, *mediation* deals with the semantic heterogeneity, and *deployment* manages the interactions between component Web services and computing resources.

Despite the interesting work of [22, 35], these authors focus only on run-time context which includes information related to the execution of composite Web services and their participants (e.g., number of current service instances and their status). In our work, we consider context parameters for selection (composition) phase. In addition, context categories presented in all these works are domain based and deal only with a special kind of application. Collaboration scenarios need to be investigated more in the categorization of context. To this end, we have introduced a new context categorization suitable for the SOE environment

7. Implementation

To illustrate the feasibility of the proposed architecture, we have implemented a prototype: SOE Creation prototype (*SOEC*) enables the discovery of Web services which supports a Collaborative Business Process based on an abstract description.

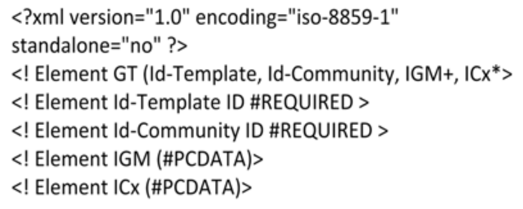
7.1 Prototype presentation

The developed prototype offers a tool set to represent the collaborative business process schema as a collection of Goal Templates, control nodes and edges. It enables also the managing different domain and context ontologies in order to add descriptions about generic methods and generic context parameters to Goal Templates. SOEC forms and deploys Web service communities.

The SOEC contains 4 modules as illustrated in the figure 10:

1. Collaborative business process schema module: provides a set of tool for modeling the collaborative business process as a combination of goal templates and control flow. Based on this module, CBP designer (initiator) can assign a goal template to a particular community. Plus, the CBP designer can affect the generic methods and generic context parameters by instantiating them with concrete values based on users preferences and constraints.

2. XML generator module: contains two functions. First, it generates XML files containing Goal templates' descriptions and overall collaborative business process schema description (see figure 9) a DTD description for the XML file). Second, this module develops the final collaborative business process description by switching the goal template tags in the schema file with the actual selected Web service. This XML file can be an entry for a process execution engine.



```
<?xml version="1.0" encoding="iso-8859-1"
standalone="no" ?>
<! Element GT (Id-Template, Id-Community, IGM+, ICx*>
<! Element Id-Template ID #REQUIRED >
<! Element Id-Community ID #REQUIRED >
<! Element IGM (#PCDATA)>
<! Element ICx (#PCDATA)>
```

Fig. 9 XML DTD representing the Goal Template.

3. Service discovery engine module: discovers and selects as described in previous section, Web services fitting the Goal Template requirements.

4. Ontology and community management module: creates a community and assigns generic methods and

generic context parameters to the community based in both domain ontology and context ontology.

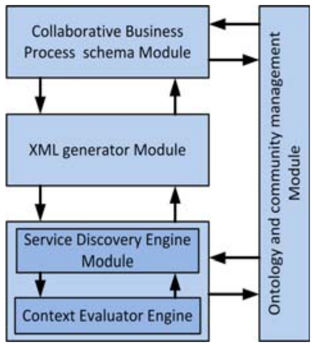
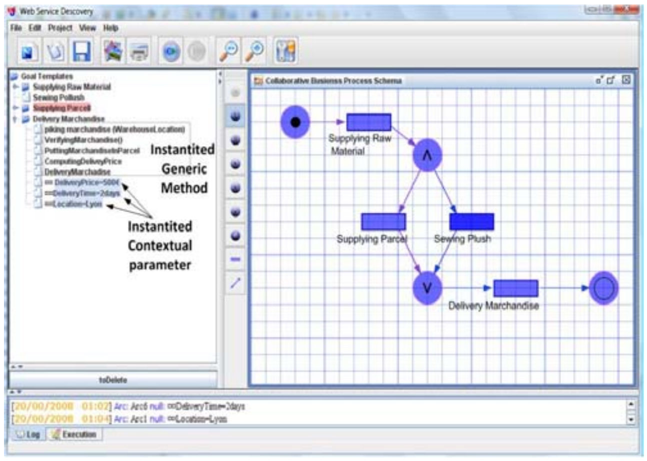

Collaborative Business Process schema Module
XML generator Module
Service Discovery Engine Module
Context Evaluator Engine
Ontology and community management Module

Fig. 10 Prototype's modules.

These modules are implemented using Eclipse IDE. Apache Axis is used to deploy and invoke Web services as well as Apache Tomcat server to host Web services. We use jUDDI, an open source Java implementation of UDDI, as a service registry.

The next figures (Figure 11 and Figure 12) depict a screen shot of the implemented prototype.



[20/00/2008 01:02] Arc: Arc6 null: ==DeliveryTime=2days
[20/00/2008 01:04] Arc: Arc1 null: ==Location=Lyon

Fig. 11 Screen shot of *SOEC* prototype.

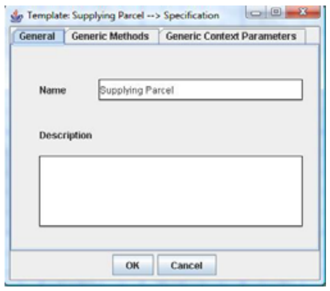

Template: Supplying Parcel --> Specification
General
Generic Methods
Generic Context Parameters
Name
Supplying Parcel
Description
OK
Cancel

Fig. 12 Screen shot of the Template configuration interface.

7.2 Prototype evaluation

We have tested the developed prototype based on two criteria: the usability of the prototype and its performance. The tests were executed on a computational environment characterized by the configuration shown in the next Table (Table 1).

Table 1: Computational environment characteristics

Operating System	Windows Vista business
Java Virtual machine	Sun Java Run Time Environment 1.5.0_06
Screen size	15.4 inch

For the first criteria we have asked some users to model several Collaborative Processes having different behaviors and different Template number. It turned out that even novice users can easily conceive the process schema of a SOE and configure the different Templates just by reading the help provided by the prototype.

For the second criteria we have tested the prototype for the modeling of different Collaborative Processes with different number of Templates. We conclude that modeling the behavior of the Collaborative Processes becomes complex from 20 Templates process given the limited size of the used computer screen. It becomes very complex part of 40 Templates process. The next figure (Figure 13) exposes the evolution of the time consuming when the number of Template becomes important.

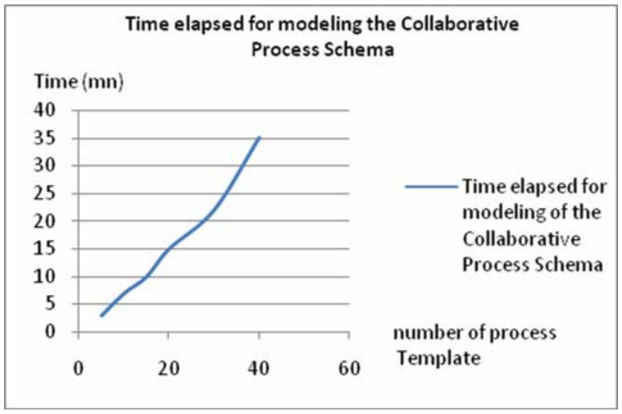


Fig. 13 Time consuming for the Collaborative Process modeling.

8. Conclusion and future work

Service Oriented Enterprise results from a composition of several Web services as an inter-enterprise Collaborative Process. Relevant mechanisms for web service selection and composition are consequently a key point in establishing such collaborative process. Nevertheless, Web service selection mechanisms are based essentially on functional parameters. This leads on some deficiency on the selection phase. In fact more pragmatic criterions are required in such collaborative process. In this work we have presented a framework for discovering and selecting Web service in case of Collaborative Business Process based on contextual parameters. Using context parameters for describing and selecting Web services enhances the compatibility of selected services with the user requirements. For this end, our framework is based on four key elements: context categories (suitable to collaboration scenario), WS-Policy (for context parameters representation), Web service community (to manage Web service registry space) and goal template (to enable an easy specification of collaboration request and selection criteria). Goal templates are used in an abstract specification of the Collaborative Business Process. It gathers functional and contextual parameters for describing a target Web service.

As a future work we envisage to add some semantic annotations to the Goal Template in generic method side and contextual parameter side. Adding such annotation empowers Template descriptions and consequently enhances the selection process. We envisage also clustering context parameters in different measurement scales (nominal, ordinal, interval and ratio) in order to compute exactly the dissimilarity between two context descriptions.

References

[1] Resource Description Framework (RDF), Published online at http://www.w3.org/RDF/.

[2] A. Alamri, M. E. and, and A. E. Saddik, "Classification of the state-of-the-art dynamic web services composition techniques," *International Journal for Web and Grid Services*, vol. 2, pp. 148–166, 2006.

[3] Bajaj et. al, "Web Services Policy Framework (WS-Policy) published on line at: http://www-128.ibm.com/developerworks/library/specification/ws-polfram/," 2006.

[4] B. Benatallah, Q. Z. Sheng, and M. Dumas, "The Self-Serv environment for Web services composition," *IEEE Internet Computing*, vol. 7, pp. 40-84, 2003.

[5] D. Benslimane, A. Arara, G. Falquet, Z. Maamar, P. Thiran, and F. Gargouri, "Contextual Ontologies: Motivations, Challenges, and Solutions," in *Fourth Biennial International Conference on Advances in Information Systems*, Izmir, Turkey, 2006, pp. 168 - 176.

[6] N. A. Bradley and M. D. Dunlop, "Toward a Multidisciplinary Model of Context to Support Context-Aware Computing," *Human-Computer Interaction*, vol. 20, pp. 403-446, 2005.

[7] P. Brezillon, "Context-based modeling of operators' Practices by Contextual Graphs," in *14th Mini Euro Conference in Human Centered Processes*, 2003.

[8] T. Broens, S. Pokraev, M. v. Sinderen, J. Koolwaaij, and P. D. Costa, "Context-aware, ontology-based, service discovery," in *Ambient Intellegence*. vol. 3295, 2004, pp. 72-83.

[9] O. Bucur, P. Beaune, and O. Boissier, "What Is Context and How Can an Agent Learn to Find and Use it When Making Decisions?," in *international workshop of central and eastern europe on multi agent systems*, Budapest, Hungary, 2005, pp. 112-121.

[10] C. Bussler, "The Role of Semantic Web Technology in Enterprise Application Integration," *Data Engineering Bulletin*, vol. 26, pp. 62-68, 2003.

[11] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, pp. 281-308, 2004.

[12] F. Casati and M.-C. Shan, "Dynamic and adaptive composition of e-services," *Information Systems*, vol. 26, pp. 143-163, 2001.

[13] S. Chaari, Y. Badr, and F. Biennier, "Enhancing web service selection by QoS-based ontology and WS-policy," in *2008 ACM Symposium on Applied Computing*, Brasil, 2008, pp. 2426-2431.

[14] S. Chaari, F. Biennier, C. Benamar, and J. Favrel, "Towards Service Oriented Enterprise," in *Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management* Helsenki, 2006, pp. 920-925.

[15] M. Conti, M. Kumar, S. K. Das, and B. A. Shirazi, "Quality of service issues in Internet web services," *IEEE Transactions on Computers*, vol. 51, pp. 593-594., 2002.

[16] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, pp. 86-93, 2002.

[17] N. Damianou, N. Dulay, E. lupu, and M. Solman, "the Ponder Policy Specification Language," in *International Workshop, Policy 2001*, Bristol, UK, 2001, pp. 18-38.

[18] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction*, vol. 16, pp. 97-166, 2001.

[19] R. L. Glass, "Defining Quality Intuitively," *IEEE Software*, vol. 15, pp. 103-104, 1998.

[20] A. J. Gonzales and R. Ahlers, "Context-based representation of intelligent behavior in training simulations," *International Transactions of the Society for Computer Simulation*, pp. 153-166, 1999.

[21] T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knwledge Acquisition*, vol. 5, pp. 199-220, 1993.

[22] Z. Maamar, D. B. , P. Thiran, C. Ghedira, S. Dustdar, and S. Sattanathan, "Towards a context-based multi-type policy approach for Web services composition," *Data & Knowledge Engineering*, doi:10.1016/j.datak.2006.08.007, 2006.

[23] Z. Maamar, D. Benslimane, P. Thiran, C. Ghedira, S. Dustdar, and S. Sattanathan, "Towards a context-based multi-type policy approach for Web services composition," *Data & Knowledge Engineering*, 2006.

[24] D. Martin, "Putting Web Services in Context," in International Workshop on Context for Web Services in conjunction with the 5th International and Interdisciplinary Conference on Modeling and Using Context, 2005, pp. 3-16.

[25] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara,

"Bringing Semantics to Web Services: The OWL-S Approach," in *SWSWPC*, 2004.

[26] B. Medjahed and Y. Atif, "Context-based matching forWeb service composition," *Distributed and Parallel Databases archive*, vol. 21, pp. 5-37, 2007.

[27] B. Medjahed and A. Bouguettaya, "A Dynamic Foundational Architecture for Semantic Web Services," *Distributed and Parallel Databases*, vol. 17, pp. 179–206, 2005.

[28] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid, "ComposingWeb services on the Semantic Web," *Very Large Data Base*, vol. 12, pp. 333-351, 2003.

[29] S. K. Most´efaoui and G. K. Most´efaoui, "Towards A Contextualisation of Service Discovery and Composition for Pervasive Environments," in *the Workshop on Web-services and Agent-based Engineering*, 2003.

[30] E. Newcomer, *Understanding Web service XML, WSDL, SOAP et UDDI*: Addison-Wesley Professional, 2002.

[31] B. Orriëns, J. Yang, and M. P. Papazoglou, "Service Component: a mechanism for web service composition reuse and specialization," *Journal of Integrated Design and Process Science*, vol. 8, pp. 13-28, 2004.

[32] M. Ouzzani, "Efficient Delivery of Web Services," *PHD thesis*, vol. 15, 2004.

[33] M. Paolucci, A. Ankolekar, N. Srinivasan, and K. Sycara, "The DAML-S Virtual Machine," in *the Second International Semantic Web Conference (ISWC2003)*, 2003, pp. 335-350.

[34] N. Ryan, "ConteXtML: Exchanging contextual information between a Mobile Client and the FieldNote Server,"http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html.

[35] S. Sattanathan, N. C. Narendra, and Z. Maamar, "Ontologies for Specifying and Reconciling Contexts of Web Services," *Electronic Notes in Theoretical Computer Science*, vol. 146, pp. 43-57, 24 January 2006 2006.

[36] S. Sattanathan, P. Thiran, Z. Maamar, and D. Benslimane, "Engineering Communities of Web Services," in *International Conference on Information Integration and Web Based Applications & Services (iiWAS)* Timisoara, Romania, 2007, pp. 57-66.

[37] S. Thakkar, J. L. Ambite, and C. A. Knoblock, "A Data Integration Approach to Automatically Composing and Optimizing Web Services," in *International Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[38] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok, "Semantic Web languages for policy representation and Reasoning: A comparison of KAOS, Rei and Ponder," in *International Semantic Web Conference*, Florida, USA, 2003, pp. 419-437.

[39] R. M. Turner, "Context-mediated behavior for intelligent agents," *Human-Computer studies*, vol. 48, pp. 307-330, 1998.

[40] K. Verma, R. Akkiraju, and R. Goodwin, "Semantic Matching of Web Service Policies," in *Second International Workshop on Semantic and Dynamic Web Processes (SDWP 2005)*, 2005.

[41] K. Verma, R. Akkiraju, and R. Goodwin, "Semantic Matching of Web Service Policies," in *in Second International Workshop on Semantic and Dynamic Web Processes (SDWP 2005)*, 2005.

[42] W3C, " Web Services Policy Attachment, available at http://www.w3.org/Submission/WS-PolicyAttachment," 2006.

[43] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS aware Middleware for Web Services composition," *IEEE Transactions on software engineering*, vol. 30, pp. 311-327, 2004.