Bringing Context to Intentional Services for Service Discovery

Salma Najar
Centre de Recherche en Informatique-Université Paris1
90, rue de Tolbiac 75013 Paris - France
Salma.Najar@malix.univ-paris1.fr

Manuele Kirsch-Pinheiro, Carine Souveyet
Centre de Recherche en Informatique-Université Paris1
90, rue de Tolbiac 75013 Paris - France
Manuele.Kirsch-Pinheiro@univ-paris1.fr,
Carine.Souveyet@univ-paris1.fr

Abstract—**In service-orientation, the notion of service is studied from different point of views. On the one hand, several approaches have been proposing services that are able to adapt themselves according to the context in which they are used. On the other hand, some researches have been proposing to consider user intentions when proposing business services. We believe that these two views are complementary. An intention is only meaningful when considering the context in which it emerges. Conversely, context description is only meaningful when associated with a user intention. In order to take profit of both views, we propose to extend the Ontology Web Language for services description (OWL-S). We include on it both the specification of context associated with the service and the intention that characterize it. This extended description is experimented in a semantic registry that we built for service discovery purposes. Such registry considers a matching algorithm, which exploits the extended description. Then, we present experimental results of this matching algorithm that demonstrates the advantages one may have on using the proposed descriptor. Thus, we propose a new vision of service orientation taking into account the notion of intention and context. This new vision is based on the extended semantic descriptor, which is necessary in order to enhance transparency of the system by proposing to the user the most appropriate service.**

Keywords-OWL-S; SOA; Intentional Service; Context-Aware Service; Service Discovery

I. INTRODUCTION

Service-Oriented Architecture (SOA) is a computing paradigm lying on the notion of service. This notion is represented as fundamental element for developing software applications [25]. Besides, service stands to independent entities, with well-defined interfaces that can be invoked in a standard way. This does not require, from the user, knowledge about how the service actually performs its tasks [10].

SOA can be viewed through multiple lenses, from the IT perspective up to business leaders [37]. The notion of service is used on different abstraction levels. Technically, it refers to a large variety of technologies (Web Services, ESB [31], OSGI [24], etc.). On a business level, services are proposed as a way to respond to high-level user requirements.

One of the essential challenges in service orientation is *how to find a set of suitable service candidates with regard to a user request and needs*?

On the one hand, we can observe a tendency to context-awareness and adaptation on services. Several authors [15][34][35][36] have been proposing adaptable services to the context in which they are used. These services are usually called *context-aware services* [15]. Their importance is growing with the development of pervasive and mobile technologies. Context-aware services focus on service adaptation considering the circumstances in which it is requested. However, considerations such as why context is important and what is its impact to the user needs remain underestimated.

On the other hand, research has pointed out the importance of considering user requirements on service orientation. Several works [12][18][25][28] proposed to take into account user intentions when proposing business services. According to these works, a service is supposed to satisfy a given user intention.

However, even when considering high-level services, as business services, one should consider variability related to context. Several authors have been considering the influence of context information on business process [30][32]. This influence remains whenever such processes are implemented through business services. Such services still have to cope with the context in which they are called.

Therefore, we have two separated views of service orientation. First, we have an extremely technical view. It focuses on technical issues needed to execute and to adapt service in highly dynamic environments. In the opposite, we have a high level view. This view focuses on user requirements. The latter considers why a service is needed, without necessarily considering how it is executed, neither in which circumstances it is performed. More than the execution context, this high level view ignores the context in which user intentions emerge. Besides, technical view passes over user intentions behind service and observed context information.

We believe that these two views are complementary and should not be isolated from each other. Fully potential of service orientation will not be reached if we do not consider both points of view: *intention-based services* and *context-aware services*. A new vision of service orientation is necessary in order to enhance transparency of the system. In our opinion, an intention is only meaningful when considering it in a given context. Moreover, a context description is only meaningful when associated with a user intention.

Therefore, services should not only be realistic. They should also be described in sufficient detail to allow meaningful semantic discovery.

In order to explore such a complementary views, we should: (1) be able to represent user *intention* in order to be aware of the real use of a service; and (2) capture user *context* in order to choose the best strategy to reach user intention.

Thus, a closely relation can be observed between the concepts of intention, service and context. First, a user intention is defined as "*user requirement representing the intention that a user wants to be satisfied by a service without saying how to perform it*". Then, the context information is defined as "*any information that can be used to characterize the situation of an entity (a person, place, or object)* [7]". We believe that both concepts should be considered in service orientation. We advocate that the selection of the service satisfying user intention is valid only in a given context. For us, a context plays an important role influencing the manner to fulfill user intention and the execution of the service that satisfies this intention (Figure 1).

A user does not require a service because he is under a given context. He requires a service because he has an intention that a service can satisfy in this context. However, this intention is not a simple coincidence; it emerges because he is under a given context.

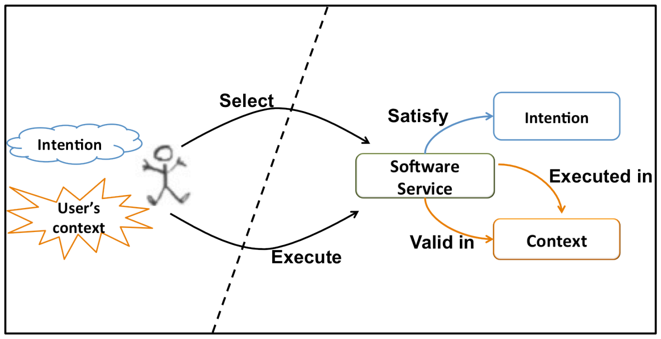


Figure 1. Context and Intention in Service Orientation

In a previous paper [21], we start exploring these ideas, by proposing a semantic description of services. This description encompasses the description of the intention service can satisfy and the context in which this intention is meaningful.

In the present paper, we go further on this semantic description. We propose a semantic description of services that fully describes service intentionality, contextual conditions and intentional composition of these services. We propose to enrich service registry by developing a complete semantic service descriptor based on our OWL-S extension. Then, we propose a service discovery process based on a matching algorithm guided by user and service context and intention. The matching algorithm is based on the implementation of our semantic service descriptor in order to find the most appropriate service according to the user request. This service discovery is implemented and evaluated in order to demonstrate the feasibility of our algorithm.

This paper is organized as follows: Section II presents an overview on related work. Section III presents a motivating scenario. The Section IV introduces the notion of intention and context as preliminary concepts. In Section V, we present our proposition of a semantic descriptor for intentional and context-aware services. Besides we present, in this section, the implementation of our enriched service discovery and the matching algorithm. In Section VI, we discuss our evaluation of the discovery process. And finally, we conclude in Section VII.

II. Related Work

A service can be seen as an independent and easily composed application that can be described, discovered and invoked by other applications and humans. In the last decade, the notion of service has evolved, from simple Web services to semantic Web services [17]. Indeed, we could observe an important tendency for semantically describing services, in order to handle potentially ambiguous service descriptions [17]. Such semantic description is based on richer representation languages, such as OWL-S [16]. OWL-S provides a comprehensive specification of a service.

From the one side, a semantic description is one of the building blocks of context-aware services. These context-aware services can be defined as services which description is associated with contextual properties. We can notice that, an important change has been performed on the way we work and on the way technology support us. We pass from a quite static model, in which people use to interact with business process only during their "work time" to "mobile worker" model [20]. With the evolution of mobile technologies, and notably smartphones, the static model does not fit anymore. Thus, Systems should now consider not only the tasks a user can (or must) perform, but also the context in which such user finds him when performing an action.

In this context, Taylor et al. [35] have considered enriching service with context information. Such works have considered using semantic Web technologies for describing context-aware services. These authors define context-aware services as services that are able to adapt themselves (their composition as well as the content they supply) according to the context in which they are used.

Next, several authors [34][36] have been proposing context-aware services, whose importance is growing with the development of pervasive technologies. An illustration of this phenomenon is given by [34], who propose improving service modeling, based on OWL-S, with context information (user information, service information and environment information). Suraci *et al.* [34] consider that user should be able to specify contextual requirements corresponding to the service he is looking for (availability, location, etc.). Furthermore, this user should be able to specify the context provided by the environment (wireless connection, etc.).

Other authors, such as [36], also advocate for representing context requirements when describing context-aware services. Toninelli *et al.* [36] consider that, in pervasive scenarios, users require context-aware services. These services are tailored to their needs, current position, execution environments, etc. Therefore, service modelling should be improved, including contextual information.

Moreover, Ben Mokhtar *et al*. [2] propose a context aware semantic matching of services based on ontologies. This is expressed in OWL-based languages for enriching service description. In order to support efficient, semantic and context-aware service discovery, they present EASY. From the one side, EASY provides a language for semantic specification of functional and non-functional service properties named EASY-L. From the other side, it provides EASY-M, a corresponding set of conformance relations. These authors [2] propose the use of ontologies in order to automatically and unambiguously discover such services.

Then, Xiao *et al.* [38] are interested on context-aware service and especially on the dynamicity of the environment. These authors propose a context modelling approach. This approach can dynamically handle various context types and values. They use ontologies to enhance the meaning of a user context values and automatically identify the relations among different context values. Based on the relations among context values, they discover and select the potential services that the user might need.

From the other side, several authors have considered a direct participation of the end user on service specification.

Brnsted et al. [4] illustrate this tendency by observing several approaches allowing end users to actively interact with service composition specification. However, these authors do not consider whether terminology used by these tools correspond to the user current vocabulary. The question that emerges here is the following: are these users technical people, who are familiarized with service-oriented technology? Or, are these users business actors who are totally unaware of technical considerations?

A different point of view is given by [4][12][18], which highlight the importance of considering user requirements on service orientation. According to them, a service is supposed to satisfy a given user intention, which becomes central to service definition.

For example, Web Service Modelling Ontology (WSMO) [44] provides a conceptual framework. This framework describes semantically the core element of semantic web services. It is well known by its intention-driven approach. This approach assumes that a user is looking for a service in order to satisfy a specific intention (goal). According to Roman et al. [29] an intention (goal) *describes aspects related to user desires with respect to the requested functionality*. Then, Keller et al. [6] present a mechanism for WSMO web service discovery. This mechanism is based on a matching process between the user goal and the web service capabilities. This information is represented as a set of objects referring to ontologies. The ontologies used in this service discovery mechanism capture general knowledge about a specific domain.

Moreover, WSMO is used in [43] in order to raise the business process management (BPM) from the IT level (Technical) to the user level (Business). In this project [43], the notion of intention is used in order to specify processes and tasks for which the most appropriate web services can be discovered dynamically.

Thus, in WSMO the user intention and the service capabilities are not formulated according to a specific template. As we mentioned, this information is only represented as a set of object. Therefore, they do not identify the real role that plays each object in the intention specification. Consequently, they do not exploit the semantic of verbs, targets and parameters that can represent an intention.

Besides, in WSMO we do not consider the contextual information that can influence the service execution. This element is not clearly defined in the service description. Thus, they neglect the influence that can have the context on the satisfaction of the user intention.

Another works such as [12] and [28] propose a service oriented architecture based on an intentional perspective. Such architecture proposes the notion of *intentional service*. This represents a service focusing on the intention that allows satisfying rather than the functionality it performs. Therefore, the introduction of intentional services is an alternative for bridging the gap between low level, technical software-service descriptions and high level, strategic expressions of business needs for services.

Then, Aljoumaa *et al.* [1] propose an approach for building the Intentional Services Model (ISM) proposed by [12][28]. These authors [1] present an ontological based solution to help user discovering and formulating his needs. They propose a mechanism for matching user needs formulated in business terms as intentions with the intentions of services published in an extended registry.

Moreover, Mirbel e*t al.* [18] also adopt ontology and semantic web technologies for proposing intention-based service discovery mechanisms. They propose a semantic approach guided by the user intentions. In this approach, user requests are expressed using semantic Web technologies.

Then, Olson *et al.* [23] believe that by using intentions, services can be described on any arbitrary and useful level of abstraction. According to these authors, through an intention refinement algorithm, intentions can be used not only for describing services, but also for improving the performance of service discovery.

In addition, Baresi et al [2] propose an innovative intention-based approach to represent requirements and adaptation capabilities for service composition.

None of these works considers the notion of context, contrary to Bonino et al. [4]. These authors [4] propose an intention-based dynamic service discovery and composition framework that uses context information. Nevertheless, context information is used only for filtering the input of the user request.

All these works represent two different views of service orientation: (i) one view proposing a context-aware based approach. This view focuses on the adaptation of services according to the context information; and (ii) a second view focusing on an intention-based approach, proposing high level services. This view focuses on user intentions. The first view focuses on service discovery and composition on a highly dynamic environment. It does not consider why service is needed. More, it focuses especially on the context on which a service is valid or can be executed rather than the real use of the service and the purpose of the user.

The latter considers this question without considering the context in which this need emerges. The user requests a service with a specific intention. Although, this intention is more significant when considered in a specific context that can influence its satisfaction.

Questions such as "*why a service is useful in a given context*?" or "*in which circumstances a service need raises*?" remain unexplored. Thusn in order to explore both views; we have first to represent them in a semantic way. Thus, we propose a semantic descriptor of services that encompasses notions of context and intention. This description will enrich the service discovery and will improve the selection of the most appropriate service.

III. MOTIVATING SCENARIO

Bob works as a commercial in a company. He is responsible for preparing customer proposals. His company offered him different manners to prepare his proposals. When he is in the company, he has a direct access to the enterprise resources planning (ERP). He uses the service *prepare proposal* that allow him to write the proposal and send it to the customer via an e-mail. When he is outside, he needs first of all, to make a VPN connection that will allow him to access the ERP. Then, he has to write the proposal and finally send it via fax or e-mail to the customer. In this situation, Bob needs to know how to prepare his proposal depending on his context (if he needs a VPN connection, if he has an Internet connection, if he has a fax next to him, etc.). The information system provides several implementations that Bob needs to know. Such technical details seem too complicate for Bob, who would prefer just a service to prepare his proposal. Actually, Bob needs a transparent access to the service he is looking for, without any technical details concerning which implementation is available in a given context. In order to handle this problem, we propose to describe and to search for him services based on the intention they are supposed to satisfy, which is easier to understand for Bob than technical details about available implementations.

IV. PRELIMINARY CONCEPTS: INTENTION & CONTEXT

Before presenting details about our proposition, some concepts should be introduced, essentially the notion of *context* and *intention*. In this paper, we exploit the close relation between these two concepts. This is in order to enrich the service description and enhance the service discovery mechanism.

In the next part, we will introduce the notion of intention, define intentional services and present the intentional composition.

A. *Intentional service at the glance*

The term intention has several different meanings. According to [11], an intention is an "optative" statement expressing a state that is expected to be reached or maintained. The intention represents the goal that we want to achieve without saying how to perform it [11]. Bonino *et al.* [4] defines an intention as a goal to be achieved by performing a process presented as a sequence of intentions and strategies to the target intention. Even if they differ, all these definitions let us consider an intention as a *user requirement representing the intention that a user wants to be satisfied by a service without saying how to perform it* [22].

This intention represents the user request when he is looking for a service satisfying his needs. Aljoumaa et al. [1], present a mechanism, based on ontologies, that guide user when he formulates his intention. They present a methodology that help user to discover his needs and formulate it consequently.

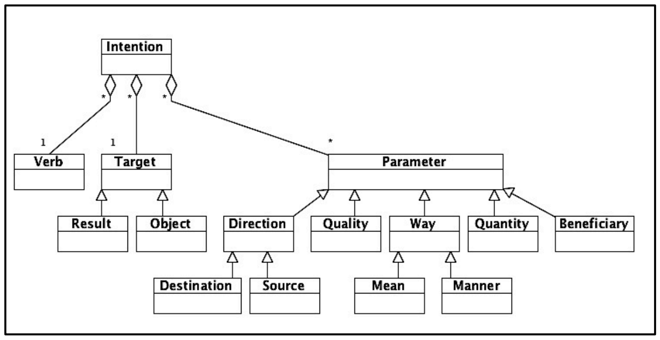


Figure 2. Intention template based on [28]

Thus, to ensure a powerful intention matching, we formulate the intention according to a specific template [12][28]. This template is defined based on linguistic approach [26]. This approach is inspired by the Fillmore case grammar [9] and its extensions by Dick [8]. It represents user and service requirements. In this template, an intention is expressed by a *verb*, a *target* and a set of optional *parameters,* as illustrated in Figure 2. The verb and target are mandatory, while the other parameters are optional and play specific roles with respect to the verb.

First, the *verb* exposes the action allowing the realization of the intention. Then, the *target* represents either the *object* existing before the achievement of the intention, or the *result* resulting from the intention satisfaction. The parameters are useful to clarify the intention and to express additional informational such as: direction, ways, quality, etc. The *direction* parameter characterizes the *source* and *destination* of the entities. From the one side, the *destination* identifies the location of the entities produced by the intention satisfaction. From the other side, the *source* identifies the initial location of the entities. In addition, the intention template represents the *ways* parameter. This parameter refers to the instrument of the intention satisfaction. It represents the *mean* and the *manner*. The *mean* describes the entity that serves as an instrument to achieve the intention, while the *manner* identifies an approach in which the intention can be satisfied. Finally, the *quality* parameter defines a property that must be reached or maintained [20].

In addition to intention template proposed on [12][28], we also consider the sense of a verb. The intention formalism is based on the verb as an element that expresses the actions, the states, the activities, etc.

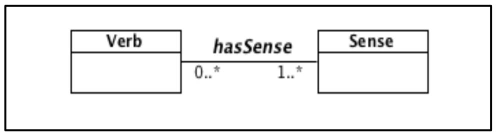


Figure 3. Sense of the verb

The same verb can have different senses depending on his use. For every intention verb, we attribute a set of senses, as illustrated in Figure 3. These senses indicate the meanings of this verb. For example the verb "*reserve*" has different senses such as: "*give or assign a resource to a particular person or cause*", "*arrange for and reserve (something for someone else) in advance*", etc.

1) Intentional services definition

As we mentioned above, an intentional service is represented as a service captured at a high-level, in business comprehensible terms. This service is described by the intention it can satisfy, i.e., according to an intentional perspective. A model of this intentional service is presented in [12][28]. These authors [12][28] present an intentional service model (ISM) that associate to each service an intention it can satisfy. ISM is composed of 4 facets, represented in Figure 4, namely the *service interface*, the *service behaviour*, the *service composition* and the Q*oS*.

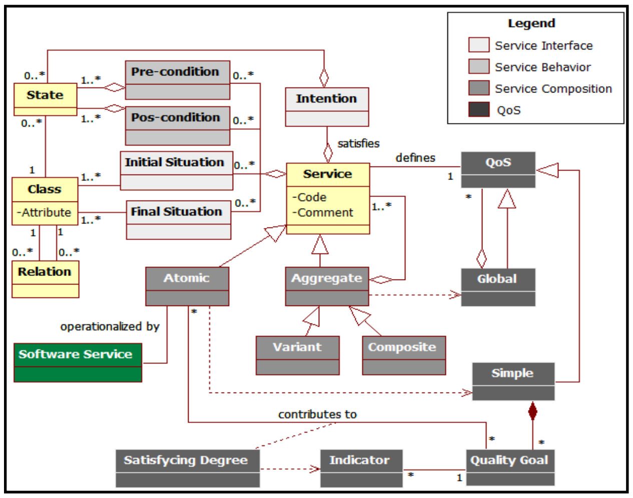


Figure 4. Intentional Service Model (ISM) [28]

First, the *service interface* represents the service that permits the fulfilment of an intention. This is based on an initial situation and terminating in a final situation. Then, the *service behaviour* specifies the pre and post conditions. The pre condition represents the sets of initial states required by the service for the intention achievement. The post condition represents the set of final states resulting from intention achievement. Next, the *service composition* represents the possibility of composing more complex intentions by combining lower abstraction level intentions. Next section gives more precisions about service composition. Finally, the *QoS* introduces the non-functional dimension of service. It represents the quality requirements associated with intentional services.

2) Intentional services composition

The intentional service model emphasises variability on the satisfaction of its corresponding intention. It allows the variability through the service composition. In the ISM model, an intentional service can be *aggregate* or *atomic*. First, aggregate services represent high-level intentions. These intentions can be decomposed in lower level one, helping business people to better express their strategic/tactical intentions.

Intentional composition admits two kind of aggregate services: a *composite* and a *variant*. While composite services reflect the precedence or succession relationship between two intentions, variant service correspond to the different manner to achieve an intention. This needs for variability is justified by the need to introduce flexibility in intention achievement [12][28].

According to [28], atomic services are related to operationalized intentions and can be fulfilled by SOA functional services. Atomic intentional services are then operationalized by software services. In contrast, aggregate services have high-level intentions that need to be decomposed in lower level ones till atomic intentional services are found.

Nevertheless, we advocate that this vision does not consider the evolution of service technology. This evolution can stand now for small pieces of software. This software encapsulates reusable functionalities, as well as for large legacy systems, whose complex process are hidden by technologies such as Web Services or ESB [31].

By considering that only atomic services can be operationalized by software service, ISOA architecture limits the reuse of such legacy systems under an intentional approach. Actually, legacy systems often encompass complex services. These systems subsume the satisfaction of multiple intentions or an intensive variability on their satisfaction. Moreover, such systems can be compared to aggregate intentions, but they cannot be assimilated to simple atomic intentions.

In this paper, we extend the vision originally proposed by [12][28]. We consider that both atomic and aggregate intentional services can be operationalized by software service, which can be also atomic or composite. As a consequence, both technical and intentional compositions are possible independently, allowing more powerful constructions. Besides, contrarily to [28], we do not consider that intentional service should be seen as a separate entity from technical service. Such separation leads to poor technical descriptions that are semantically incomplete, since they do not include an intentional description. We propose in this paper a full semantic descriptor, which considers service as a single entity with multiple dimensions: intentional, technical and contextual dimensions.

B. *Context information description*

Context information corresponds to a very wide notion. As we mentioned earlier, it is usually defined as any information that can be used to characterize the situation of an entity (a person, place, or object considered as relevant to the interaction between a user and an application) [7]. The notion of context is central to context-aware services that use it for adaptation purposes. Context information can stand for a plethora of information, from user location, device resources [27], up to user agenda and other high level information [13]. Nevertheless, in order to perform such adaptation processes, context should be modelled appropriately. The way context information is used depends on what it is observed and how it is represented. The context-adaptation capabilities depend on the context model [19].

Thus different kinds of formalism for context representation have been proposed. Nevertheless, an important tendency can be observed on most recent works: the use of ontology for context modelling [19]. According to [19], different reasons motivate the use of ontologies, among them their capability of enabling knowledge sharing in a non-ambiguous manner and its reasoning possibilities. This tendency follows the evolution of context-aware services, which adhere, in their majority, to a semantic description of such services. In this paper, we also adhere to this tendency, adopting an ontology-based context modelling based on [27].

Reichle *et al.* [27] define context information based on three main concepts: 1) the ***entity*** referring to the element to which the context information refers; 2) the ***scope*** identifying the exact attribute of the selected entity that it characterizes; and 3) the ***representation*** used to specify the internal representation used to encode context information in data-structures.

According to this context model, we directly associate the scope that we observe with the entity that the context element refers to. This let us consider that, in order to have the value for a given scope, we have to observe his corresponding entity. However, this represents an ambiguity since some scopes are not directly related to a precise entity. For example, if we want to represent the humidity around a given user, this information can not be captured by observing the *user* but rather the *environment*.

Therefore, in order to make this context model more meaningful, we believe that we must separate clearly the notion of *entity* that we want to represent from the *property* that we want to observe.

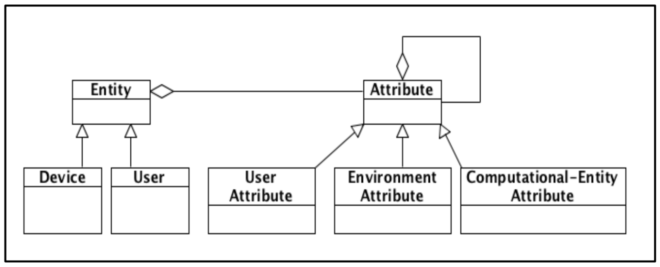


Figure 5. Context Model

The Figure 5 illustrates our proposed context model. Each context information is identified by two important concepts, the *entity* and the *attribute*. The distinction between these two concepts is adopted in order to not mix up the *entity* to which the context information refers to (e.g., user, device, etc.) with the *attribute* that characterize the property that we want to observe. The attribute represents a piece of context information about the environment (location, time...), a user (profile, role...) or a computational entity (resource, network....).

Moreover, this context model is based on a multi-level ontology representing knowledge and describing context information (Figure 6). It provides flexible extensibility to add specific concepts in different domains. All these domains share common concepts that can be represented using a general context model, but they differ in some specific details.

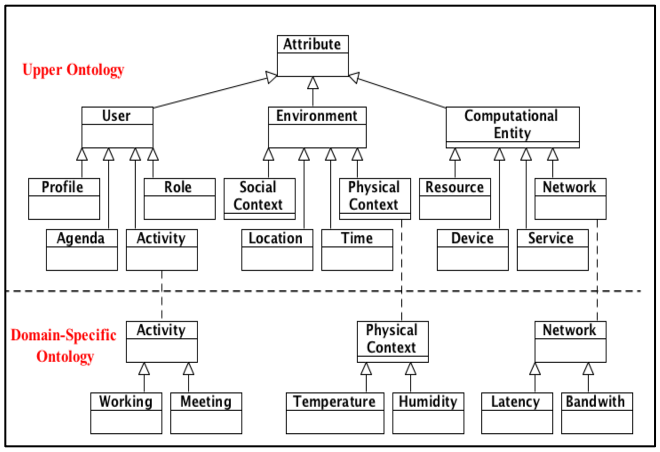


Figure 6. Multi-Level Context Ontology

According to [19], this context model presents context according to three main categories: (i) *environment context* representing contextual information about user location, time, social context, etc.; (ii) *user context* that represents user profile, agenda, Role, activity, etc.; (iii) *computational entity context* including contextual information related resource, network, etc.

This two-level ontology consists in an upper level, defining general context information (e.g., profile, activity, location, network, etc.), and a lower level, with more specific context information (temperature, latency, etc.). Therefore this separation enhances the reuse of general context information and provides flexibility to add domain-specific knowledge.

Besides, in our opinion, context information does not have all the same importance. It can differ from a user to another according to their preferences. Consequently, we propose to associate with context attributes the notion of '*weight*'. Our purpose is to clarify the importance of a context attribute according to the domain and to the user preferences. The profile context model, presented in the Figure 7, consider this by allowing to each entity to have a profile specifying this weight.

A *profile* represents the user preferences regarding context information. These user preferences are represented as a *profile* assigned to each *entity*. It allows the definition of a *weight* that the profile owner allocates to each context attribute. This weight, whose value is between 0 and 1, represents the importance of an attribute to a given entity.

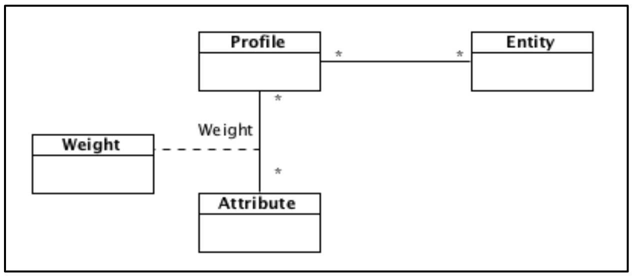


Figure 7. Profile Context Model

The purpose here is to highlight the real importance of a context attribute according to user preferences. The importance of the attribute is proportional to its weight. It decreases if the value affected decreases, and it grows if this value grows. The *weight* can then be used for matching purposes, and notably during the matching between the user current context and service context conditions. By proposing this profile model, we intend to promote context attributes that are seen as most relevant ones for a given user. For example, by considering this profile, a context attribute having a lower weight (i.e., that is not particularly interesting for the user) will be less influent for calculating the context matching score, than another attribute with higher weight. Even if this context attribute participates in the matching process, the weight assigned to it will decrease its importance, and consequently the context score will be calculated according to user preferences.

Therefore, a user (or even a system administrator) may define, for an entity, a set of profiles representing his preferences. Through this notion of profile, it is possible to enhance this selection of the most appropriate service that can interest the user.

The next section describes how all dimensions can coexist in the proposed service semantic descriptor.

V. PROPOSITION: PUTTING EVERYTHING TOGETHER: CONTEXT-AWARE INTENTIONAL SERVICES

The latest research in service oriented computing recommends the use of the OWL-S for semantically describe services [34]. Even if OWL-S is tailored for Web services, it is rich and general enough to describe any service [34]. OWL-S [16] defines web service capabilities in three parts representing interrelated sub-ontologies named service profile, process model and grounding. The *service profile* expresses what the service does. It gives a high-level description of a service, for purposes of advertising, constructing service requests and matchmaking. The *process model* answers to the question: how is it used? It represents the service behaviour as a process and describes how it works. Finally, the *grounding* maps the constructs of the process model onto detailed specification of message formats, protocols and so forth (often WSDL).

The OWL-S represents a flexible and extensible language, as demonstrated by works such as [14][34]. Similar to these works, we propose to extend service description in OWL-S by including information concerning both context and intention that characterize a service.

A. Describing context-aware intentional service in OWL-S

In this section, we present our extension of OWL-S. This extension includes: (i) intentional information about services; and (ii) contextual information about services conditions of execution.

In the following part, we present the intentional extension of OWL-S.

1) Describing service intentions

According to an intentional perspective, a user requires a service because he has an intention that the service is supposed to satisfy. Hence, the importance of considering user intentions emerges on service orientation. Such new dimension is central to service definition.

Thus, we propose to enrich OWL-S service description with the intention associated to it. We extend OWL-S, including on it the intention that a service can satisfy. This is done by adding a new sub-ontology, which describes the intentional information of the service.

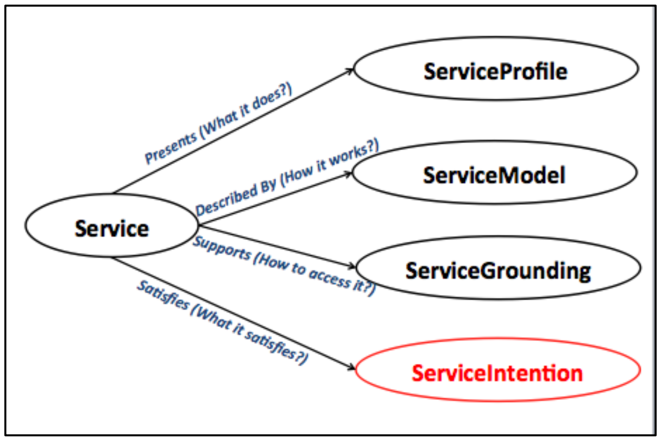
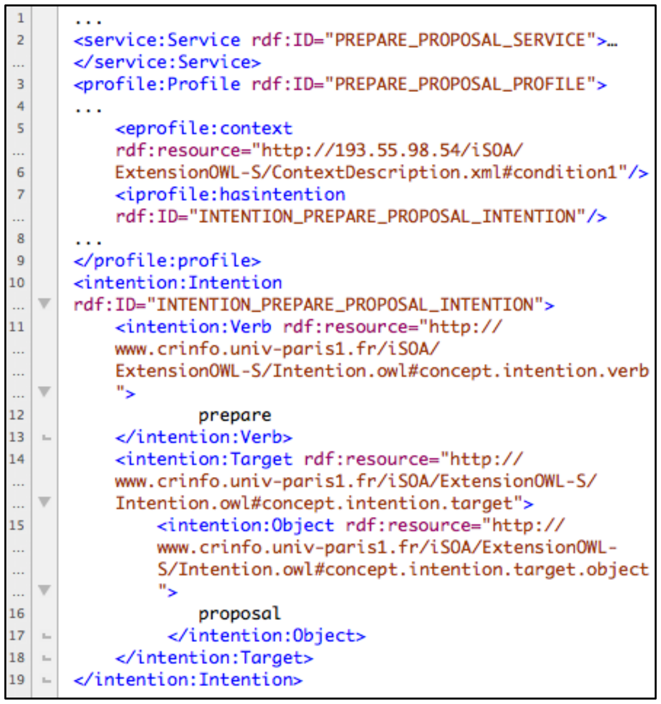


Figure 8. Service intention Description in OWL-S

The Figure 8 illustrates our intentional extension of owl-s. The proposed property *satisfies* is a property of *Service*. The class *ServiceIntention* is the respective range of this property. Each instance of *Service* will *satisfy* a *ServiceIntention* description. The *ServiceIntention* provides the information needed to discover the appropriate service in order to satisfy a specific *intention*. The service intention presents "what the service satisfies", in a way that is suitable to determine whether the service fulfills user intention. This part of the service description presents the principal intention of the service. This intention is formulated according to a specific template [28].

This description differs from the last service intention description presented in [20]. One can notice that the service intention description presented in this present paper is defined in a separate sub-ontology. It is related to the service description instead of describing it as a service parameter in the service profile description. This change is due to several reasons: (i) since the service intention, in our proposition, represents an important aspect of service definition. It will affect the service discovery process, and is more meaningful and clear to describe it in a separate block; (ii) the evaluation performed on both descriptions demonstrated that the analysis of a service description with separately intention description on a sub-ontology is more efficient than the proposal presented in [20]; (iii) the analysis of a service description by a matching algorithm that ignores intention description is easiest: if this description is separated from the rest (in other words, extended services remain compatible with old registry).



```
1   ...
2   <service:Service rdf:ID="PREPARE_PROPOSAL_SERVICE">…
    </service:Service>
3   <profile:Profile rdf:ID="PREPARE_PROPOSAL_PROFILE">
4   ...
5       <eprofile:context
        rdf:resource="http://193.55.98.54/iSOA/
6       ExtensionOWL-S/ContextDescription.xml#condition1"/>
7       <iprofile:hasintention
        rdf:ID="INTENTION_PREPARE_PROPOSAL_INTENTION"/>
8   ...
9   </profile:profile>
10  <intention:Intention
    rdf:ID="INTENTION_PREPARE_PROPOSAL_INTENTION">
11      <intention:Verb rdf:resource="http://
        www.crinfo.univ-paris1.fr/iSOA/
        ExtensionOWL-S/Intention.owl#concept.intention.verb
        ">
12              prepare
13      </intention:Verb>
14      <intention:Target rdf:resource="http://
        www.crinfo.univ-paris1.fr/iSOA/ExtensionOWL-S/
        Intention.owl#concept.intention.target">
15          <intention:Object rdf:resource="http://
            www.crinfo.univ-paris1.fr/iSOA/ExtensionOWL-
            S/Intention.owl#concept.intention.target.object
            ">
16              proposal
17           </intention:Object>
18      </intention:Target>
19  </intention:Intention>
```

Figure 9. Example of describing service intention in OWL-S

In the Figure 9, a service is associated with the intention "prepare a proposal" (line 7, <*iprofile:hasintention*>). This intention *(<intention:Intention>*) is then described according the template "verb, target, parameters" (see lines 10-19), using the extended OWL-S elements. In this example, the verb *(<intention:Verb>*) is "prepare" and the target (<*intention:Target*>) represents the object "proposal" (<*intention:Object*>).

In the next section, we will present the extension of OWL-S including service contextual information.

2) *Describing contextual information*

An intention that a user wants to satisfy emerges in a given context. In our opinion, it has a closely relation between the notion of context and intention. This relation should be exploited. Thus, the user intention becomes less important and less significant if we did not take it with its context of use. According to this, we propose to extend the service profile. Our purpose is to allow service provider to define context information that characterize an intentional service.

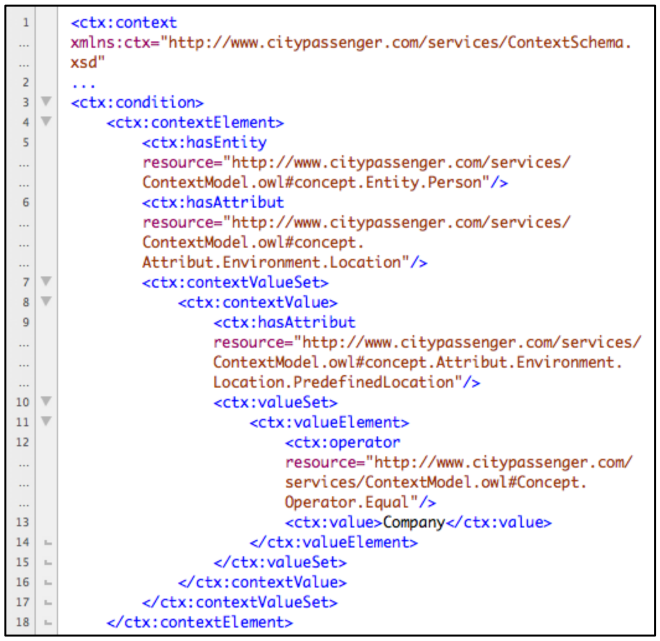
For instance, let us consider the intention *prepare proposal* (described in the Section III). For this intention different implementations are available enabling users to search, prepare and send a proposal to the customer in different situations. This service can be particularly executed considering client *location*, type of the used *device* and type of the *network*.

A first implementation can be proposed considering the user is in the company (location). He writes his proposal from his personal computer (device) and sends it via fax to the customer. This user is connected via the Ethernet of the company (network).

A second implementation of the same service can be executed when the user is outside (location). He accesses, via his smart phone (device) with a 3G connexion (network), to a specific application allowing him to write a proposal and then send it via mail to the customer.

Each one of these implementations can be associated with a different context description. By considering such a description and the user current context, it is possible to select the most appropriate implementation in a transparent way for the user.

For example, the Figure 10 and Figures 11 illustrates an example of a context conditions description that can be associated to the first implementation of the *prepare proposal* intentional service.

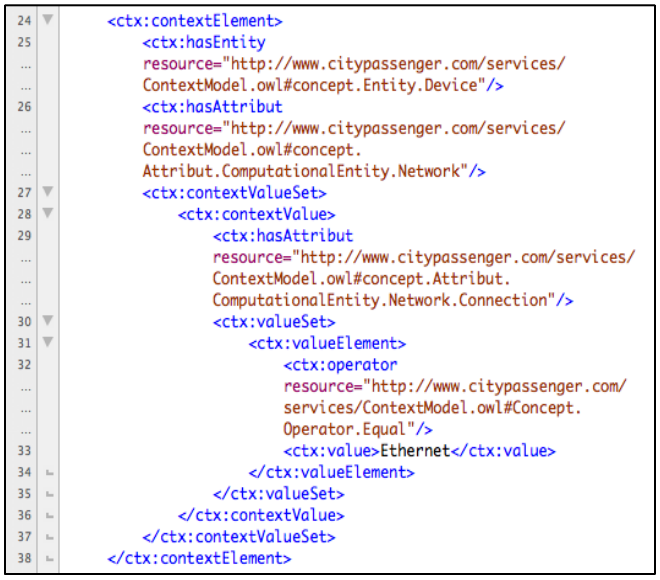


```
<ctx:context
xmlns:ctx="http://www.citypassenger.com/services/ContextSchema.
xsd"
...
<ctx:condition>
    <ctx:contextElement>
        <ctx:hasEntity
        resource="http://www.citypassenger.com/services/
        ContextModel.owl#concept.Entity.Person"/>
        <ctx:hasAttribut
        resource="http://www.citypassenger.com/services/
        ContextModel.owl#concept.
        Attribut.Environment.Location"/>
        <ctx:contextValueSet>
            <ctx:contextValue>
                <ctx:hasAttribut
                resource="http://www.citypassenger.com/services/
                ContextModel.owl#concept.Attribut.Environment.
                Location.PredefinedLocation"/>
                <ctx:valueSet>
                    <ctx:valueElement>
                        <ctx:operator
                        resource="http://www.citypassenger.com/
                        services/ContextModel.owl#Concept.
                        Operator.Equal"/>
                        <ctx:value>Company</ctx:value>
                    </ctx:valueElement>
                </ctx:valueSet>
            </ctx:contextValue>
        </ctx:contextValueSet>
    </ctx:contextElement>
```

Figure 10. User context Description: Condition 1

First, the Figure 10 represents the condition that the "location" of the user is the "company". The *user* represents the *entity* to which the context refers (<ctx:hasEntity>). The *location* represents attribute that characterize the observed property of the context (<ctx:hasAttribute>). And the *company* represents the observed value of the attribute (<ctx:value>).

Next, the Figure 11 represents another context condition. This illustrates that the "network" (attribute) of the "device" (entity) is "Ethernet" (value).



```
<ctx:contextElement>
    <ctx:hasEntity
    resource="http://www.citypassenger.com/services/
    ContextModel.owl#concept.Entity.Device"/>
    <ctx:hasAttribut
    resource="http://www.citypassenger.com/services/
    ContextModel.owl#concept.
    Attribut.ComputationalEntity.Network"/>
    <ctx:contextValueSet>
        <ctx:contextValue>
            <ctx:hasAttribut
            resource="http://www.citypassenger.com/services/
            ContextModel.owl#concept.Attribut.
            ComputationalEntity.Network.Connection"/>
            <ctx:valueSet>
                <ctx:valueElement>
                    <ctx:operator
                    resource="http://www.citypassenger.com/
                    services/ContextModel.owl#Concept.
                    Operator.Equal"/>
                    <ctx:value>Ethernet</ctx:value>
                </ctx:valueElement>
            </ctx:valueSet>
        </ctx:contextValue>
    </ctx:contextValueSet>
</ctx:contextElement>
```

Figure 11. User context Description: condition 2

Thus, and according to [20], contextual information can then be considered as part of the service description, since it indicates situations to which the service is better suited. However according to [14], context information cannot be statically stored on the service profile due to its dynamic nature. Context properties related to service execution can evolve, whereas service profile is supposed to be a static description of the service.

Thus, in order to handle dynamic context information on static service description, we adopt the approach [14]. This approach enriches OWL-S service profile with a *context* attribute, which represents a URL pointing to context description file. Since context information is dynamic, we opt to describe context element in an external file. Thus, this will allow service provider to easily update such context information related to the service description itself. The context description of a service describes, from the one side, the situation status of the requested service (environment in which the service is executed), and from the other side, the contextual conditions (requirements) to execute the service. Both information can be used for service discovery purposes. In the next section, we will describe briefly the composition of intention.

3) *Composing intentions*

Intention and context attributes described above intent to expose both aspects of a service notably for discovery purpose. Thanks to the OWL-S extension we propose, a service can be discovered either by intention it can satisfy, or by the context associated with this intention. In addition to these aspects, a third aspect should be exposed: the service variability. Such variability is expressed, in the intentional perspective, by the composition of intentional services. This indicates the decomposition of the service intention on lower level intentions. Thus, according to [20], (i) the technical composition of a service, described in OWL-S process model, represents software components. These components are combined to supply service operations; (ii) the intentional composition represents not only lower level intentions necessary to satisfy service intention, but also different possibilities the service offers for satisfying this intention [20].

The technical composition supplies technical elements necessary for service execution. Then, the intentional composition provides an understanding, from final user point of view, of the service and the diverse forms of satisfying service intention. Thus, we propose to extend OWL-S process model by including the specification of an intentional service process, as illustrated in Figure 12.

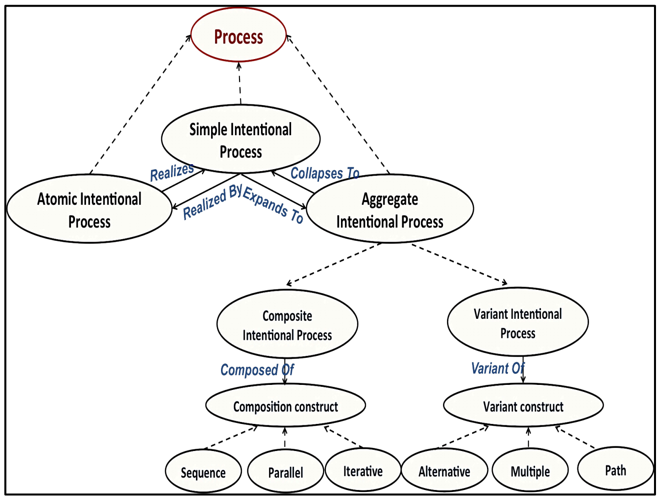


Figure 12. Composing intentions in OWL-S process Model

The Figure 12 presents the extension we propose for the process model. This extension considers two kinds of process: the atomic intentional process and the aggregate intentional process. It considers also a simple intentional process, which is used to provide an abstracted view that can be atomic or aggregate. A simple intentional process is realized by an atomic intentional process and expands into an aggregate intentional process. An aggregate intentional process can be either a composite intentional process or a variant intentional process.

First, the composite intentional processes reflect the precedence/succession relationship between their intentions. Such relationships are specified using composition constructs such as *Sequence*, *Parallel* and *Iterative*. The composition represents a *sequence* in which there is a sequential order between component processes, or a *parallel* in which components can run in parallel. The *iterative* construct is used when the satisfaction of an intention may require iterative execution of a given set of actions.

Then, the variability is represented by the variant intentional process, which uses constructs such as *multiple*, *alternative* and *path.* The *multiple* construct offers a non-exclusive choice in the realization of the intention. It groups multiple simple processes, among them, at least one will be chosen. The *alternative* construct represents a process with an alternative choice. It regroups several simple processes that are mutually exclusive. Then, it builds a new process of the same level of abstraction but of higher granularity. And finally, the *path* construct offers a choice in how to achieve the intention of the aggregate process. It offers composite processes that are mutually exclusive.

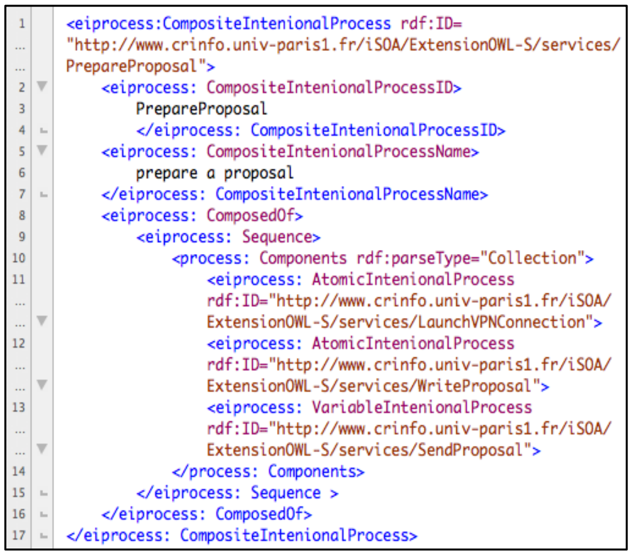

```
<eiprocess:CompositeIntenionalProcess rdf:ID=
"http://www.crinfo.univ-paris1.fr/iSOA/ExtensionOWL-S/services/
PrepareProposal">
    <eiprocess: CompositeIntenionalProcessID>
        PrepareProposal
        </eiprocess: CompositeIntenionalProcessID>
    <eiprocess: CompositeIntenionalProcessName>
        prepare a proposal
    </eiprocess: CompositeIntenionalProcessName>
    <eiprocess: ComposedOf>
        <eiprocess: Sequence>
            <process: Components rdf:parseType="Collection">
                <eiprocess: AtomicIntenionalProcess
                rdf:ID="http://www.crinfo.univ-paris1.fr/iSOA/
                ExtensionOWL-S/services/LaunchVPNConnection">
                <eiprocess: AtomicIntenionalProcess
                rdf:ID="http://www.crinfo.univ-paris1.fr/iSOA/
                ExtensionOWL-S/services/WriteProposal">
                <eiprocess: VariableIntenionalProcess
                rdf:ID="http://www.crinfo.univ-paris1.fr/iSOA/
                ExtensionOWL-S/services/SendProposal">
            </process: Components>
        </eiprocess: Sequence >
    </eiprocess: ComposedOf>
</eiprocess: CompositeIntenionalProcess>
```

Figure 13. Example of OWL-S Intentional composition: Composite Intentional Process

For instance, let us consider the example of the service (described in the Section III) satisfying the intention *prepare proposal*. This service is offered by Bob ERP. It allows him to write a proposal and send it to the customer.

This service is described as a composite service (<eiprocess:CompositeIntentionalProcess> in Figure 13). It represents a *sequence* (<eiprocess:Sequence>) between the atomic intentions (<eiprocess:AtomicIntentionalProcess>) *launch VPN connection* and *write proposal*, and the variant intention (<eiprocess:VariableIntentionalProcess>) *send proposal,* as illustrated in Figure 13.

This latest variant represents a path (<eiprocess:Path>) between the atomic intentions *send proposal by mail* and *send proposal by fax* , as illustrated in Figure 14. From a technical point of view, this service is composed by multiple ERP functionalities, described in OWL-S process model. Such description is beyond the scope of this paper, since no modification has been proposed for technical composition on OWL-S process model.

In our vision, an aggregate intentional service, which is composed of other intentional services, can be associated with a software service. This software service can be also composite of other technical services. This extends the vision of [1][12][28] that consider that only atomic intentional service can be operationalized by a software service.

According to them, aggregate intentional service need to be decomposed till an atomic intentional service is found. These authors do not take into account, for example, the software encapsulating reusable functionalities.

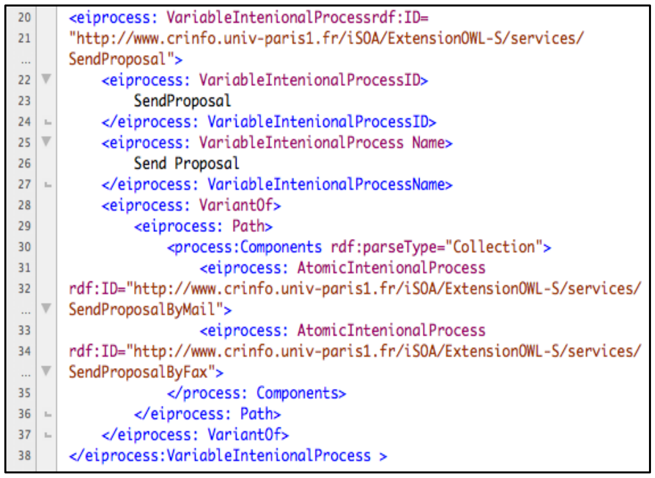

```
<eiprocess: VariableIntenionalProcessrdf:ID=
"http://www.crinfo.univ-paris1.fr/iSOA/ExtensionOWL-S/services/
SendProposal">
    <eiprocess: VariableIntenionalProcessID>
        SendProposal
    </eiprocess: VariableIntenionalProcessID>
    <eiprocess: VariableIntenionalProcess Name>
        Send Proposal
    </eiprocess: VariableIntenionalProcessName>
    <eiprocess: VariantOf>
        <eiprocess: Path>
            <process:Components rdf:parseType="Collection">
                <eiprocess: AtomicIntenionalProcess
rdf:ID="http://www.crinfo.univ-paris1.fr/iSOA/ExtensionOWL-S/services/
SendProposalByMail">
                <eiprocess: AtomicIntenionalProcess
rdf:ID="http://www.crinfo.univ-paris1.fr/iSOA/ExtensionOWL-S/services/
SendProposalByFax">
            </process: Components>
        </eiprocess: Path>
    </eiprocess: VariantOf>
</eiprocess:VariableIntenionalProcess >
```

Figure 14. Example of OWL-S Intentional composition: Variant Intentional Process

Thanks to the OWL-S extension proposed here, we enable the description of intentional composition, from final user point of view. This extension exposes the variability representing different manners to satisfy user intentions. The intentional composition description allows a service discovery guided by intention, presented at a high level.

4) *Describing Service Resource*

A service, with an intentional description, can be seen as an *intentional service*. Each intentional service acts as a fragment of process implemented by the software service. It handles input information in order to satisfy its corresponding intention and resulting in some output information. Input and output of an intentional service describes, respectively, an initial and a final situation. These situations are expressed as set of states over resources handled by the service. Such initial and final situations are important for intentional composition. This is because they are supposed to guide the satisfaction of high-level intention associated with the aggregate service.

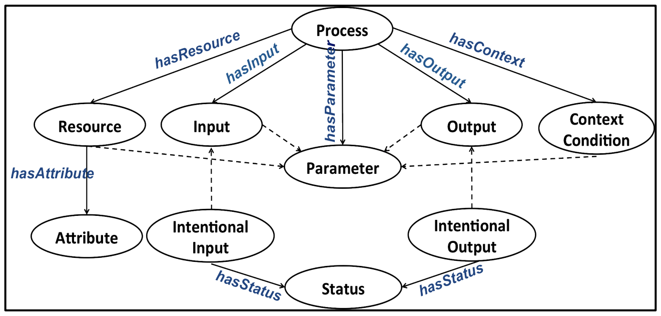
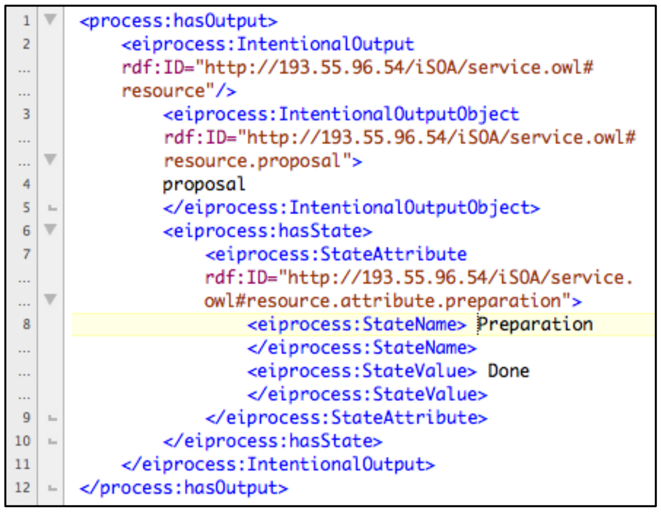


Figure 15. Resource description in OWL-S process model

According to this, we introduced the notion of resource on OWL-S, as presented in Figure 15. A *resource* represents a class of objects, with its corresponding attributes, that are manipulated by an intentional service. For instance, a service implementing the intention "*prepare proposal*" will manipulate a "*proposal*" resource, with a "*preparation state*" attribute. Then, when the resource is used as intentional input or output parameter, a *state* can be assigned to the resource. The element "*state*" allows then attaching values to each resource attribute.

The Figure 16 illustrates the resource "*proposal*" used as intentional output with a state defined by the attribute "*preparation state*" with the value "*done*".



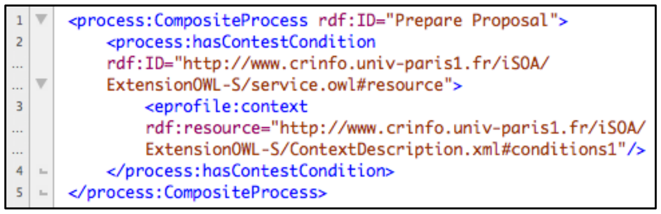
```
<process:hasOutput>
    <eiprocess:IntentionalOutput
    rdf:ID="http://193.55.96.54/iSOA/service.owl#
    resource"/>
        <eiprocess:IntentionalOutputObject
        rdf:ID="http://193.55.96.54/iSOA/service.owl#
        resource.proposal">
        proposal
        </eiprocess:IntentionalOutputObject>
        <eiprocess:hasState>
            <eiprocess:StateAttribute
            rdf:ID="http://193.55.96.54/iSOA/service.
            owl#resource.attribute.preparation">
                <eiprocess:StateName> Preparation
                </eiprocess:StateName>
                <eiprocess:StateValue> Done
                </eiprocess:StateValue>
            </eiprocess:StateAttribute>
        </eiprocess:hasState>
    </eiprocess:IntentionalOutput>
</process:hasOutput>
```

Figure 16. Intentional output in OWL-S process model

Besides, we believe that variability on intention achievement may depend on external factors. These factors concern context information. Each variant may have context conditions in which it is most appropriate to use it. For each variant, we attribute a context conditions description. This context description represents in which circumstances it is most appropriate to use it.

Thus, in order to consider context influence on intentional variants, we propose including context information also on the process variability description. We associate contextual conditions to each process variant described at the intentional level. This context description will enhance the variability dynamic of intentional process.

Thus, such extension can help to choose the variant according to context conditions. Thus, we extend OWL-S process model by including on it a contextual condition through the element "*context*" (<eprofile:context> in Figure 17). Similar to context element associated with service profile, this element points out to an external file containing context description (see Section V.A.2), referring to context conditions that apply to a given variant.



```
<process:CompositeProcess rdf:ID="Prepare Proposal">
    <process:hasContestCondition
    rdf:ID="http://www.crinfo.univ-paris1.fr/iSOA/
    ExtensionOWL-S/service.owl#resource">
        <eprofile:context
        rdf:resource="http://www.crinfo.univ-paris1.fr/iSOA/
        ExtensionOWL-S/ContextDescription.xml#conditions1"/>
    </process:hasContestCondition>
</process:CompositeProcess>
```

Figure 17. Context Condition on OWL-S process model

For example, the he Figure 17 illustrates this OWL-S process model extension through a contextual condition pointing out the context description file.

B. *Context-Aware Intentional Service registry: Implementation*

In this section, we will present the implementation of our semantic service descriptor. Then, we will introduce our service discovery.

1) *Overview*

In this paper, we present a semantic enriched service descriptor. In order to demonstrate feasibility, we implemented a semantic service registry. This takes into account an enriched service descriptor based on the extension of OWL-S described in this paper. This description provides comprehensive specifications of a service. This specification is based on the intention it satisfies and the context conditions in which it is valid and executed. This extended service description is then tested and used by a context-aware intentional service discovery process (detailed in the Section V.B.3). The purpose is to find the most appropriate service according to a given request.

The Figure 18 shows the architecture of our enriched registry application [33]. The interface *ServiceManager* represents the entry point to the application. It offers a set of methods allowing ontology management and service discovery and selection. The implementation of this interface holds two references of the *PersistenceManager* and the *SearchEngine* interfaces. Both implementations use the Strategy Pattern in order to provide a flexible change of the strategy. Then, this can facilitate the addition of new persistence or/and matchmaker implementations. To load the right strategy, the application uses a properties file in which it is stated the strategy class to use.

The *SerachEngine* uses a *MatcherFacade* interface that acts as a façade between the SearchEngine and the API to operate service descriptions. The *PersistenceFacade* interface acts as a façade between the PersistenceManager and the database to access service and ontology descriptions.

Thus, our proposed semantic service registry can be divided into two core parts. The first one is the *persistence* package. It handles ontologies and service descriptions. The second one is the *search engine* package. It is in charge of searching an appropriate service for a given request, based on the extended service description.

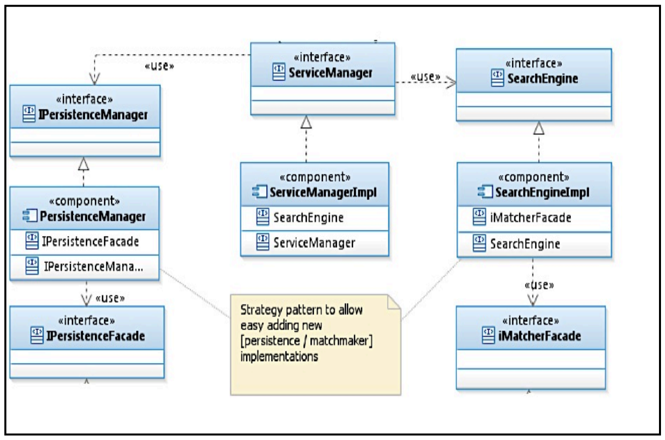

«interface» ServiceManager
«use»
«interface» SearchEngine
«interface» IPersistenceManager
«component» PersistenceManager
IPersistenceFacade
IPersistenceMana...
«component» ServiceManagerImpl
SearchEngine
ServiceManager
«component» SearchEngineImpl
iMatcherFacade
SearchEngine
«interface» IPersistenceFacade
Strategy pattern to allow easy adding new [persistence / matchmaker] implementations
«interface» iMatcherFacade

Figure 18. Semantic Service Registry Architecture

In the next sections, we explain the different used ontologies and models. They are implemented in order to create the enriched semantic service registry and to implement the extended OWL-S descriptor proposed here. Then, we explain how this extended service description is used by the service discovery manager. And how it can find the most appropriate service that fits user request in his current context.

2) Implemented models for a semantic service descriptor

The OWL-S description is based on a set of ontologies. This ontologies supplies service providers with a core set of constructs. It describes the properties and the capabilities of their services. In order to enhance this description and implement the semantic descriptor proposed in this paper, we extend OWL-S ontologies: (i) *Service.owl* (ontology providing the service profile) and (ii) *Profile.owl* (ontology providing a service grounding). Then, we add a new ontology, called *Intention.owl.* This ontology contains elements and concepts necessary to describe the intention that a service is able to satisfy. Then a new ontology named *ExtendedService.owl* brings together service and intention ontologies on this new service descriptor. The *ExtendedProfile.owl* defines the structure of the elements describing the service profile and the intention this service satisfies. In this description, we add an ObjectProperty *has_Intention* and the context information by adding a new DataProperty *context* (pointing to the context description file).

Based on the OWL-S API Mindswap [41], we develop an OWL-S extension API in Java. This extension implements the service description according to his intention and context. In order to evaluate the proposed implementation, the service retrieval test collection OWLS-TC2 [42] is used. Although, OWLS-TC contains only basic service descriptions based on OWL-S. It does not have any information about the service intention and context conditions. We preferred this test collection because it provides a large number of services from several domains, test queries and relevant ontologies. The collection is intended to support the evaluation of the performance of OWL-S service matching algorithms. OWLS-TC [42] provides 576 semantic Web services written in OWL-S 1.0 and OWL-S 1.1 from 7 domains (education, medical care, food, travel, communication, economy, weapon).

Based on this collection, we have implemented our presented semantic service descriptor. We use it in order to extend OWLS-TCS service descriptions with intentional and contextual description. Besides, we develop an interface called OWL-S extended API. This interface allows us to load from the OWLS-TC a service description and enrich it with contextual and intentional.

Thus, all this models and service descriptions are then used for a context-aware intentional service discovery process. The purpose is to search the most interesting service according to user request and context. This is detailed in the next section.

3) Context-Aware Intentional Service Discovery

With the variability and the diversity of services that are potentially available to the user in a pervasive environment, we propose a mechanism for services discovery. This mechanism take into account the user context and intention. It is based on the presented semantic descriptor of services and on a matching algorithm that we detail below. The purpose of presenting this algorithm here is to illustrate potential application of the semantic description we propose for service discovery.

The service discovery process is launched when the user sends his request representing the intention he wants to be satisfied. Once the request submitted, we load the collected user current context file. The Service Discovery Matcher loads all the semantic description of the available services (described using our proposed extension of OWL-S). Then, launches the matching process on all the available services.

The matching is a two-step process, illustrated on Figure 19. First of all, we match the user intention with the intention that the service satisfies. Second, all service context conditions are matched with the user current context (step 1.2). Finally, we calculate the degree of match between the user request and the provided service (the sum of intention and context degree of matching). Next, we add the service with its obtained score in a list (step 1.3). These steps are done for all the available services. Then, from the resulted list, we select the service having the highest score (step 2 on Figure 18).

The most important steps in this matching process are thus, the *intention matching* and the *context matching* (step 1.1 and 1.2 on Figure 19).

The intention matching process compares semantically the *verb* and the *target* of the user intention with those from service intention. From the one side, we compare the user intention verb with the service intention verb. This semantic comparison is based on a verb ontology. This verb ontology contains a set of verbs, relations between them (synonym, hyponym, hypernym) and their meanings in a specific domain. From the other side, we compare semantically the user intention target with the service intention target based on domain-specific ontologies. This domain-specific ontology represents a set of possible targets in a specific domain.

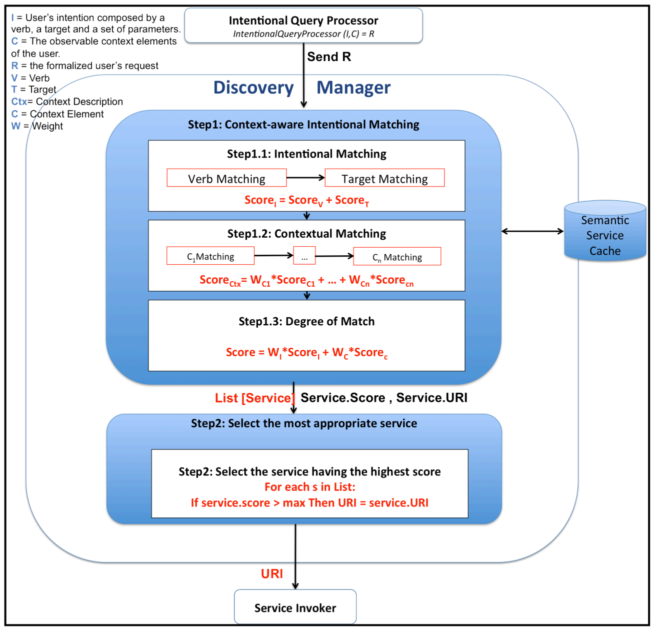

I = User's intention composed by a verb, a target and a set of parameters.
C = The observable context elements of the user.
R = the formalized user's request
V = Verb
T = Target
Ctx= Context Description
C = Context Element
W = Weight
Intentional Query Processor
IntentionalQueryProcessor (I,C) = R
Send R
Discovery Manager
Step1: Context-aware Intentional Matching
Step1.1: Intentional Matching
Verb Matching
Target Matching
$Score_I = Score_V + Score_T$
Step1.2: Contextual Matching
C_1Matching
...
C_n Matching
$Score_{Ctx} = W_{C1}*Score_{C1} + ... + W_{Cn}*Score_{cn}$
Step1.3: Degree of Match
$Score = W_I*Score_I + W_C*Score_c$
Semantic Service Cache
List [Service] Service.Score , Service.URI
Step2: Select the most appropriate service
Step2: Select the service having the highest score
For each s in List:
If service.score > max Then URI = service.URI
URI
Service Invoker

Figure 19. The context-aware intentional service discovery

Then, the context matching process, based on the context model, matches the different context element values. These values represent the context conditions of the service with those from the user current context.

This service discovery mechanism taking into account the notion of context and intention is well detailed in [22].

In order to evaluate our proposition, we first implement our semantic descriptor. Then, we propose to users an interface that allows them to load from the OWLS-TC a service description and enrich it with contextual and intentional information.

In our architecture, the *ServiceManager* module (Figure 18) represents the entry point for our discovery process. It supplies search methods for client applications. Two different search methods are offered. The first one proposes only the best ranked service. The second one proposes a list of all suitable services with their matching scores. This method is interesting because it allows the requester to observe the score of the different services and then decide whether the service really fits the request. Furthermore, the complete list allows the requester to make his own choice of which service is the best for him [33]. For example, if there is more than one service with a top score, the requester could examine each of them and decide by himself which one he wants to use. The *searchService*, on the other hand, is a simple method, that returns only the top service without any score. This method is interesting whether the client is not interested by interacting with a list and just needs a simple and fast result.

The *searchService* is based on a *Matchmaker* interface representing a discovery service matching algorithm. In our implementation, the matchmaker is easily replaceable in order to support multiple discovery processes. Thus, to select the matchmaker that should be launched with the application, we have to set up the properties file. This can be done by adding the name of the needed matcmaker.

Two main implementations of matchmaker have been proposed. First, we implement a basic matching algorithm that we called the *BasicMatchmaker*. This matchmaker is based on the input and output information. According to the user input and output, the *BasicMatchmaker* will be in charge of searching and selecting the best service [33]. Then, we implement our proposed context-aware intentional service matching algorithm (Figure 19) using OWL-S extended API, Jena [40] and the reasoner Pellet [39]. ***Jena*** [40] is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. We mainly use this framework for persistence purposes: reading, writing and verifying ontologies. Second, *Mindswap OWL-S API* [41] provides a Java API for programmatic access to read, execute and write OWL-S service descriptions. The extension of this API (OWL-S extended API) is used to operate on the service descriptions, such as getting the intention and the context of each service in order to calculate its match score. Finally, *Pellet* [39] is an OWL reasoner for Java. We adopt Pellet mainly due to its good performance and widely usage.

The implemented context-aware intentional service algorithm is called the *ContextIntentionMatchmaker*. It calculates a score based only on the user and service intention and context. This matchmaker is based on two classes *ContextMatching* and *Intentionmatching*, which are in charge of calculating respectively the context and intention scores. By separating score computation, it is possible to easily disable one of these classes in order to evaluate separately the impact of context or intention matching on the core.

The *ContextIntentionMatchmaker* demonstrates how the search method works and how a more sophisticated matchmaker can be implemented and used in the application.

In order to evaluate the validity of our context-aware service discovery algorithm and the impact of the enriched service descriptor, we present the results experiments and the evaluation of our proposition in the next section.

VI. EVALUATION

As mentioned earlier in this paper, we generate a semantic repository. This repository contains a set of extended service descriptions based on the extended OWLS-TC2 [42]. We choose the Travel domain for the test. It represents about 200 service descriptions. We enrich those descriptions by intentional and contextual information related to each service. Then, the evaluation has been performed under an Intel Core 2 Duo 2,26 GHz CPU with 2Gb of main memory.

The purpose of our experiments is to evaluate the validity of our algorithm and the feasibility of our extended service descriptor. Our objective is to evaluate 1) whether the processing time is reasonable: *scalability;* 2) whether the algorithm effectively select the most appropriate services: *result quality*.

A. Scalability

We measure the scalability of our service discovery algorithm with respect to the number of services and the capabilities of the laptop, by measuring the average processing time.

We implement a bash script that runs our implemented service discovery algorithms. This script returns the response time in second. We launch it several times and then calculate the average response time.

The result of the service discovery performance is shown in Figure 20. We evaluate the response time of three types of service discovery: (i) input/output service discovery (actually the BasicMatchFacade) (IO); (ii) intentional service discovery (implemented by the Context Intention MatchMaker with the context matching score disabled) (I) and (iii) context-aware intentional service discovery (IC). The average response time is measured with different quantity of services (10, 30, 60, 100 and 200 files).

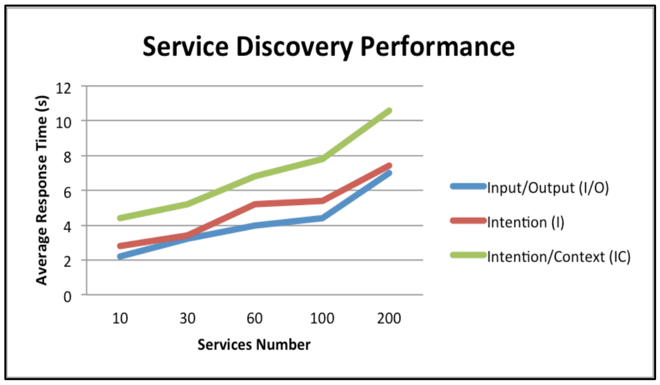


Figure 20. The evaluation of the response time of different service discovery

For example, for 60 services, the (IO) takes 4 s to select the desired services, while (I) takes 5,2 s and (IC) takes 6,8 s. The graph clearly illustrates that the context-aware intentional service discovery algorithm (IC) takes longer to process services. However, this difference on execution time does not represent a serious time difference from a user perspective (from 7s, for the faster algorithm, up to 10,6 s for the slower for 200 services). Actually, we notice that the service discovery based on input and output goes faster for selecting services. But, the response time of our algorithm still represents a reasonable processing time for selecting the most appropriate service.

As the next section demonstrates, our algorithm offers better results than input/output algorithm, since we proceed to service selection also according to user context and intention.

B. Result Quality

In order to measure the quality of the result, we cover the two most useful evaluations: *precision* and *recall*. These two metrics are defined in terms of a set of retrieved services and a set of relevant services. The *precision* represents the proportion of retrieved services that accurately matches user intention in a given context. It is calculated according to the formula (1). The *recall* represents the proportion of relevant services that are retrieved. It is calculated according to the formula (2).

$$Precision = \frac{|\{relevant\ services\} \cap \{retrived\ services\}|}{|\{retrieved\ services\}|} \quad (1)$$

$$Recall = \frac{|\{relevant\ services\} \cap \{retrived\ services\}|}{|\{relevant\ services\}|} \quad (2)$$

Thus, to evaluate this two metrics, we formulate five user requests. These requests represent the user intention in a given context, as illustrated in Table I. These requests are relatives to the travel domain. They are formulated and inspired from the available service descriptions.

We choose to represent some requests that can be accurately matched with available services (Exact). Besides, we add some requests that can have a good matching score with the available services (Not Exact).

These requests are formulated, as mentioned above, in order to validate the correctness of our IC algorithm. Moreover, our purpose is to verify if it really returns what it is used to return.

For evaluation purposes, we adopt a scenario from the *travel* domain corresponding to the service set used for testing. In this scenario, the user wants to practice a sport during his holiday. He is looking for surfing or hiking sport. Thus, he searches a destination where he can practice such sports. Then he wants to reserves a hotel or a BedAndbreakfast room for the period. Based on this scenario, we propose five requests with different context elements in order to evaluate the result quality of our implemented context aware intentional service discovery.

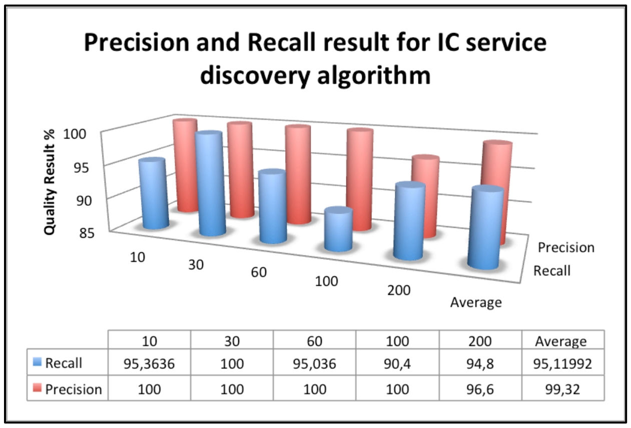
TABLE I. USER REQUEST WITH CURRENT CONTEXT

Intention	Context
Reserve Hotel	- Age >=18
Reserve BedAndBreakfast	- Age >= 18 - Season = Summer
Locate Sport Destination	- Age >= 18 - Season = Summer - City = Germany
Search Surfing Destination	- Age >= 18 - Season = Summer - Surfing Level = Beginner - Weather = not disturbed
Search Hiking	Age >= 18 - Hiking Level = Confirmed - Weather = not disturbed - Health = Good

Through the experiments, we could observe that the precision and recall is most important when considering the user intention and service context in the service discovery.

The result in Figure 21 shows that we obtain about 99 % of precision and about 95 % of recall for the 5 randomly chosen requests. These results are then compared to those obtained by IO service discovery algorithm illustrated on Figure 22. This figure shows that we could obtain an interesting recall 95,2% but a lower precision, which is about 50%.

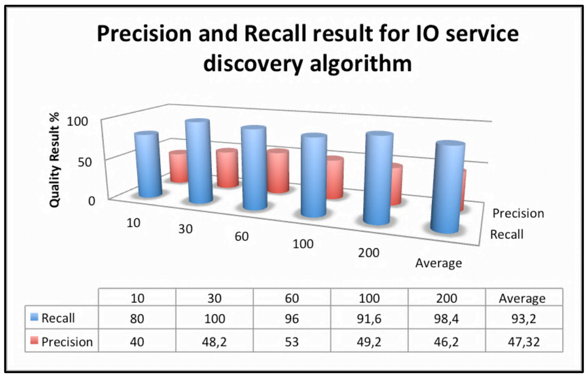
From the one side, the 95% of recall obtained by the IO algorithm is circumstantial since this algorithm is not able to select a service adapted neither to user context nor to his intention. In fact, the IO service discovery algorithm can only return all the service related to the request with a high rate of "false-positive" (indicated by precision).



	10	30	60	100	200	Average
🟦 Recall	95,3636	100	95,036	90,4	94,8	95,11992
🟥 Precision	100	100	100	100	96,6	99,32

Figure 21. The result quality of the context-aware intentional service discovery

The comparison between the Figure 21 and the Figure 22 illustrates that our proposed IC service discovery algorithm presents a more interesting result and a high level quality of results. This good result is due to 1) the use of an intention that describes the user real need; and 2) the use of context that makes the service selection most appropriate to the user (by selecting only the services that are valid and that can be executed in the user current context).



	10	30	60	100	200	Average
🟦 Recall	80	100	96	91,6	98,4	93,2
🟥 Precision	40	48,2	53	49,2	46,2	47,32

Figure 22. The result quality of the input/output service discovery

Thus, these results demonstrate that the IC algorithm is able to find all or almost services that can fulfills user intention in a given context, with the lowest rate of "false-positive"

VII. Conclusion and Future Works

In this paper, we considered context-aware and intention-based service orientation as complementary approaches that should not be isolated from each other. We explain our belief that an intention is only meaningful when considering it in a given context. Moreover, we believe that a context description is only meaningful when associated with other intention. We proposed, consequently, to enrich OWL-S service description. This extension includes the description of the intentions a service can satisfy. It includes also the context in which this intention is meaningful, context in which service is (or can be) executed.

From the one side, we enriched service description with knowledge about intentions and composition of intentions that are meaningful for final users, who request the service. From the other side, we enriched this service description with context information necessary for adapting such service.

By proposing such a semantic descriptor of service, we enable the expression of services that can adapt themselves to context of use and that represent a formulated user requirements. The service discovery process will exploit this extended description in order to enhance the satisfaction of the user request. By exposing both aspects of a service, we could develop a context-aware intentional service registry. From the one side we implement different models and ontologies needed by our proposed semantic service descriptor. From the other side, we implement a context aware intentional service discovery that illustrates how the semantic descriptor we propose can be exploited for discovery purposes.

The evaluation of our implementation demonstrates that our extension of the service description (by adding the context and intention information) makes the description more meaningful and the service discovery more precise and appropriate to the user needs.

In order to progress on this sense, our next step is to improve the implementation and analysis the results of our proposed semantic descriptor and context-aware intentional service discovery mechanism. Then, we expect to evaluate our service discovery mechanism in a more interesting real world scenario. Besides, these experiments will be tested on a more important number of services.

Based on these results, our efforts will be then focused particularly on the service prediction. Given the large amount of existing services and user needs, our purpose is to help users. We opt to propose them personalized services, without their demand, that can interest them according to their history and current context.

References

[1] K. Aljoumaa, S. Assar, and C. Souveyet, "Reformulating User's Queries for Intentional Services Discovery Using an Ontology-Based Approach", 4th IFIP Int. Conf on New Technologies, Mobility and Security (NTMS), Paris, France, pp. 1-4, 2011

[2] L. Baresi and L. Pasquale, "Adaptive Goals for Self-Adaptive Service Compositions," IEEE Int Conf on Web Services (ICWS), pp. 353-360, 2010

[3] S. Ben Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers, "EASY: Efficient semAntic Service discoveryY in pervasive computing environments with QoS and context support", Journal of Systems and Software 81(5), pp. 785-808, 2008

[4] L.O. Bonino da Silva Santos, G. Guizzardi, L.F. Pires, and M. Van Sinderen, "From User Intentions to Service Discovery and Composition," Proceeding ER'09 Proceedings of the ER 2009 Workshops (CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS) on Advances in Conceptual Modeling – Challenging Perspectives, pp. 265-274, 2009

[5] J. Brnsted, K. Hansen, and M. Ingstrup, "Service Composition Issues in Pervasive Computing," IEEE Pervasive Computing, vol. 9(1), pp. 62 -70, 2010

[6] U. Keller, R. Lara, A. Polleres, I. Toma, M. Kifer, D. Fensel, "D5.1v0.1 WSM Web Service Discovery", available in http://www.wsmo.org/ 2004/d5/d5.1/v0.1/20041112, November 2004

[7] A. Dey, "Understanding and using context," Journal Personal and Ubiquitous Computing, vol 5(1), pp. 4-7, 2001

[8] S.C. Dik, "The theory of functional grammar," Foris publications, Dodrecht, Nederthlands, 1989

[9] C.J. Fillemore, "The case for case, in Universals in linguistic theory," Holt, Rinehat and Winstonc inc, E.Bach/R.T.Harms (eds), 1968

[10] V. Issarny, M. Caporuscio, and N. Georgantas, "A Perspective on the Future of Middleware-based Software Engineering," In: Briand, L. and Wolf, A. (Eds.), Future of Software Engineering 2007 (FOSE), ICSE (Conf on Software Engineering), IEEE-CS Press, 2007

[11] M. Jackson, "Software Requirements and Specifications: A lexicon of practice, principles and prejudices," Addison Wesley Press, 256, 1995

[12] R.S. Kaabi and C. Souveyet, "Capturing intentional services with business process maps," 1st IEEE International Conference on Research Challenges in Information Science (RCIS), pp. 309-318, 2007

[13] M. Kirsch-Pinheiro, J. Gensel, and H. Martin, "Representing Context for an Adaptative Awareness Mechanism," G.-J. de Vreede; L.A. Guerrero, G.M.Raventos (Eds.), LNCS 3198 - X Workshop on Groupware (CRIWG 2004), 2004, pp. 339-348

[14] M. Kirsch-Pinheiro, Y. Vanrompay, and Y. Berbers, "Context-aware service selection using graph matching," 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'08), ECOWS. CEUR Workshop proceedings, vol. 411, 2008

[15] Z. Maamar, D. Benslimane, and N.C. Narendra, "What can context do for web services?," Communication of the ACM, vol. 49(12), 2006, pp. 98-103

[16] D. Martin, M. Paolucci, S. Mcllraith, 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," Cardoso, J. & Sheth, A. (Eds.), SWSWPC 2004, LNCS 3387, Springer, 2004, pp. 26-42

[17] S.A. Mcllraith, T.C. Son, and H. Zeng, "Semantic Web Services," IEEE Intelligent Systems, vol. 16, pp. 46-53, 2001.

[18] I. Mirbel and P. Crescenzo, "From end-user's requirements to Web services retrieval: a semantic and intention-driven approach," J.-H. Morin, J. Ralyte, M. Snene, "Exploring service science", First International Conference, IESS 2010, LNBIP 53, Springer, pp. 30-44, 2010

[19] S. Najar, O. Saidani, M. Kirsch-Pinheiro, C. Souveyet, and S. Nurcan, "Semantic representation of context models: a framework for analyzing and understanding," J. M. Gomez-Perez, P. Haase, M. Tilly, and P. Warren (Eds),1st Workshop on Context, information and ontologies (CIAO 09), European Semantic Web Conference (ESWC), ACM, pp. 1-10, 2009

[20] S. Najar, M. Kirsch-Pinheiro, and C. Souveyet, "The influence of context on intentional service," 5th Int. IEEE Workshop on Requirements Engineerings for Services (REFS'11) - IEEE Conference on Computers, Software, and Applications (COMPSAC), Munich, Germany, pp. 470–475, 2011

[21] S. Najar, M. Kirsch-Pinheiro, and C. Souveyet, "Bringing context to intentional services," 3rd Int. conf on Advanced Service Computing, Service Computation'11, Rome, Italy, pp. 118-123, 2011

[22] S. Najar, M. Kirsch-Pinheiro, C. Souveyet, L.A. Steffenel, "Service Discovery Mechanisms for an Intentional Pervasive Information System", Proceedings of 19th IEEE International Conference on Web Services (ICWS 2012), Honolulu, Hawaii, 2012, pp. 24-29

[23] T. Olsson, M.Y. Chong, B. Bjurling, and B. Ohlman, "Goal Refinement for Automated Service Discovery", 3rd Int. Conf on Advanced Service Computing, Service Computation'11, Rome, Italy, pp. 46-51, 2011

[24] OSGi Alliance, http://www.osgi.org/ : January, 2011

[25] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: A Research Roadmap," Int. J. Cooperative Inf. Syst. vol 17 n° 2, 2008, pp. 223-255

[26] N. Prat, "Goal formalisation and classification for requirements engineering," In Proc. of the 3rd International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'97). E.Dubois, A.L.Opdahl, K.Pohl. (eds), Presses Universitaires de Namur, 1997

[27] R. Reichle, M. Wagner, M. Khan, K. Geihs, L. Lorenzo, M. Valla, C. Fra, N. Paspallis, and G.A. Papadopoulos, "A Comprehensive Context Modeling Framework for Pervasive Computing Systems," In 8th IFIP Conf on Distributed Applications and Interoperable Systems (DAIS), Springer, 2008

[28] C. Rolland, M. Kirsch-Pinheiro, C. Souveyet, "An Intentional Approach to Service Engineering," IEEE Transactions on Service Computing, vol. 3(4), 2010, pp. 292-305

[29] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, D. Fensel, "Web Service Modeling Ontology", Applied Ontology, vol. 1(1), 2005, pp. 77- 106

[30] M. Rosemann, J. Recker, and C. Flender, "Contextualization of Business Processes," Int. J. Business Process Integration and Management, vol. 1(1/2/3), 2007

[31] W. Roshen, "SOA-Based enterprise integration: a step-by-step guide to services-based application integration," McGraw Hill, 2009

[32] O. Saidani and S. Nurcan, "Towards Context Aware Business Process Modeling," 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07), CAiSE'07, 2007

[33] S. Schulthess, "Construction of a registry for searching web service," Master Thesis, EFREI, Engineering School Paris, 2011

[34] V. Suraci, S. Mignanti, and A. Aiuto, "Context-aware Semantic Service Discovery," 16th IST Mobile and Wireless Communications Summit, pp. 1-5, 2007

[35] N. Taylor, P. Robertson, B. Farshchian, K. Doolin, I. Roussaki, L. Marshall, R. Mullins, S. Druesedow, and K. Dolinar, "Pervasive Computing in Daidalos," Pervasive Computing, vol. 10(1), 2011, pp. 74 -81

[36] A. Toninelli, A. Corradi, and R. Montanari, "Semantic-based discovery to support mobile context-aware service access," Computer Communications, vol.31(5), 2008, pp. 935-949

[37] R. Welke, R. Hirschheim, and A. Schwarz, "Service-oriented architecture maturity," IEEE Computer, vol. 44(2), pp. 61-67, 2011.

[38] H. Xiao, Y. Zou, J. Ng, and L. Nigul, "An Approach for Context-aware Service Discovery and Recommendation", IEEE Int. Conf on Web Services (ICWS), Miami, pp. 163-170, 2010

[39] http://www.mindswap.org/2003/pellet/ : March, 2011

[40] http://sourceforge.net/projects/jena/files/ : March, 2011

[41] http://www.mindswap.org/2004/owl-s/api/ : January, 2011

[42] http://semwebcentral.org/projects/owls-tc/ : February, 2012

[43] http://www.ip-super.org/ : June, 2012

[44] http://www.wsmo.org : June, 2012