Context-Aware Service Selection Using Graph Matching

M. Kirsch-Pinheiro[1], Yves Vanrompay[2], Y. Berbers[2]

[1] Centre de Recherche en Informatique, Université Paris 1 Panthéon-Sorbonne
90 rue de Tolbiac, 75013 Paris, France
[2] Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium

Manuele.Kirsch-Pinheiro@univ-paris1.fr, Yves.Vanrompay@cs.kuleuven.be,
Yolande.Berbers@cs.kuleuven.be

Abstract. The current evolution of Ubiquitous Computing and of Service-Oriented Computing is leading to the development of context-aware services. Context-aware services are services whose description is enriched with context information related to the service execution environment, adaptation capabilities, etc. This information is often used for discovery and adaptation purposes. However, context information is naturally dynamic and incomplete, which represents an important issue when comparing service description and user requirements. Actually, uncertainty of context information may lead to inexact matching between provided and required service capabilities, and consequently to the non-selection of services. In order to handle incomplete context information, we propose in this paper a graph-based algorithm for matching contextual service descriptions using similarity measures, allowing inexact matches. Service description and requirements are compared using two kinds of similarity measures: local measures, which compare individually required and provided properties (represented as graph nodes), and global measures, which take into account the context description as a whole, by comparing two graphs corresponding to two context descriptions.

1 Introduction

The term Ubiquitous Computing, introduced by Weiser [22], refers to the seamless integration of devices into users' everyday life [1]. This term represents an emerging trend towards environments composed by numerous computing devices that are frequently mobile or embedded and that are connected to a network infrastructure composed of a wired core and wireless edges [13]. In pervasive scenarios foreseen by Ubiquitous Computing, context awareness plays a central role. Context can be 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) [5]. Context-aware systems are able to adapt their operations to the current context, aiming at increasing usability and effectiveness by taking environmental context into account [1].

The dynamicity of pervasive environments encourages the adoption of a Service Oriented Architecture (SOA). Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications [15]. The key feature of SOA is that services are independent entities, with well-defined interfaces, that can be invoked in a standard way, without requiring the client to have knowledge about how the service actually performs its tasks [8]. Such loose coupling fits the requirements of high dynamic pervasive environments, in which entities are often mobile, entering and leaving the environment at any moment.

The adoption of SOA in pervasive environments is leading to the development of "context-aware" services. Context-awareness becomes a key feature necessary to provide adaptable services, for instance when selecting the best-suited service according to the relevant context information or when adapting the service during its execution according to context changes [6]. As pointed out by Maamar *et al.* [11], multiple aspects related to the users (level of expertise, location, etc.) and to the computer resources (on fixed and mobile devices), among others aspects, can be considered in the development of context-aware services. Thus, *context-aware services* can be defined as services which description is associated with contextual (notably non-functional) properties, *i.e.*, services whose description is enriched with context information indicating the situations to which the service is adapted to.

According to Suraci *et al.* [18], in order to provide context-aware services, one has to consider *context inputs*, *besides functional inputs, and outputs*, which may also depend on contextual information. Several authors, such as Suraci *et al.* [18], Tonielli *et al.* [21] and Ben Mokhatar *et al.* [2], propose to increase service description with context information. This information is normally used for adaptation purposes: for adapting service composition; for indicating an execution environment (device capabilities, user's location, etc.) to which the service is designed for; for indicating adaptation capabilities (mainly content adaptation) of the service, etc. This context information needs to be compared to the real user's or execution context before starting to use the service.

However, in ubiquitous environments, context information is naturally dynamic and incomplete. Dynamic context changes and incomplete context information may prevent perfect matches between required and provided properties, which may lead to the non-selection of one (or all) service(s). Service selection mechanisms have to cope with these issues: if some needed context information is missing, service selection still has to proceed and choose a corresponding service that best matches the current situation, even if context information is incomplete. In other words, when executing in pervasive environments, service matching mechanisms have to deal with the question: how to reduce problems related to mismatching between contextual conditions related to the execution of a service and current context information?

In order to overcome this issue, we propose in this paper a graph-based algorithm for matching context-aware services. The proposed service selection mechanism assumes that suitable services exist. This means our approach is employed only after the question whether suitable services are available has been answered positively. The proposed algorithm matches contextual non-functional descriptions of context-aware services using similarity measures, allowing inexact matches. Service description and the current context are interpreted as graphs, in which properties correspond to graph nodes and the edges represent the relations between these properties. Through this graph representation, service description and requirements are compared using two kinds of similarity measures: local measures, which compare individually required and provided properties (represented as graph nodes), and global measures, which take into account the context description as a whole, by comparing two graphs corresponding to two context descriptions. Moreover, we consider here only non-functional and context-related aspects of context-aware services. Even if functional aspects are the most relevant, once all services whose capabilities match functional requirements have been discovered, one has to select what service, among all the possible services, is the most suitable one, considering non-functional properties related to each service. Our graph-based service selection algorithm aims at selecting among available compatible services the most appropriate one considering the current context and taking into account the incompleteness of context information.

This paper is organized as follow: Section 2 presents an overview on related work. Section 3 introduces our approach of service selection. Section 4 presents the proposed matching algorithm and similarity measures. We conclude in Section 5.

2 Related work

A growing interest in context-aware services can be observed in the literature. For instance, several European projects are focusing on Service-Oriented Computing [16], and context-awareness appears as a crosscutting issue for these works. According to Tonielli *et al.* [21], in pervasive scenarios, users require context-aware services that are tailored to their needs, current position, execution environments, etc. According to Suraci *et al.* [18] user and service entities have requirements on context information they need in order to work properly. A user may have requirements on context of the service he is looking for (availability, location…) and on the context provided by the environment (wireless connection…). A service can require the user to provide specific context information (location, terminal capabilities…) and the environment to provide context information too (network QoS…).

The support for context-aware services depends on an improved semantic modeling of services by using ontologies that support formal description and reasoning [8]. Such a semantic modeling may contribute not only to handle problems related to service interoperability, but also in order to take into account different aspects of the environment in which the service is executed. Indeed, authors, such as Zarras *et al.* [24], advocate that semantic matching is essential for pervasive systems.

In the literature, several works, such as Ben Mokhatar *et al* [2], propose the semantic modeling and matching of services based on ontologies often expressed in OWL-based languages for enriching service description. These authors [2] propose the use of ontologies (in OWL-S) for the semantic description of functional and non-functional features of services in order to automatically and unambiguously discover such services. Klusch *et al.* [9] propose a service matching algorithm which combines reasoning based on subsumption and similarity measures for comparing inputs and outputs of service description and user request. Reiff-Marganic *et al.* [18] propose a method for automatic selection of services based on non-functional properties and context. However, inexact matching caused by incomplete or uncertain context information is not taken into account.

Other authors such as Suraci *et al.* [18] and Yau & Liu [23] propose to improve service modeling with context information. Suraci *et al.* [18] propose a semantic modeling of services in which service profile description in OWL-S is enriched with a “context” element pointing to this required context information. Yau & Liu [23] propose to enrich service description with specific external pre- (and post-) conditions expressed in the OWL-S service description denoting contextual conditions for using a given service.

Tonielli *et al.* [21] propose a framework for personalized semantic-based service discovery. This framework aims at integrating semantic data representation and match-making support with context management and context-based service filtering. In such framework, services, users and devices are modeled through a set of profiles. describing capabilities and requirements of the corresponding service. The integration is then performed in a middleware using a matching algorithm based mainly on subsumption reasoning.

The majority of research cited above concentrates the semantic matching on solving ambiguity problems related to service inputs and outputs. Such works focus mainly on functional aspects, using semantic descriptions to enrich input and output description of services. Most works related to context-aware services, as those cited above, do not consider the natural uncertainty of context information. Context information is naturally dynamic and uncertain: it may contain errors, be out-of-date or even incomplete. Uncertainty in context information is traditionally handled by appropriate models, such as Chalmers *et al.* [4], who represent context values by intervals or sets of symbolic values. In these models, incompleteness of context information is seldom considered. However, semantic matching of context-aware services should take this into account. When considering context-aware services, matching algorithms have to consider the fact that some context information can be simply missing. Such incomplete information may lead to an inexact match between service description and requirements related to the user’s current context.

In this paper, we focus particularly on this issue: how to deal with incompleteness of context information when selecting context-aware services. We propose a graph-based approach, in which service descriptions and requests are interpreted as graphs whose nodes and overall structure are compared by using similarity measures. The use of similarity measures in Computer Science is not new, as testifies the work of Liao *et al.* [10]. However, unlike Liao *et al.* [10], our work does not focus on proposing such measures. Our focus is to handle incompleteness of context information on service selection by using similarity measures. Such measures, in our case and unlike those proposed by Klush *et al.* [9], focus on non-functional and context-related aspects of context-aware services, and not on functional input and output of such services. In this sense, our approach is similar to the one proposed by Bottaro *et al.* [3], who propose ranking services according to context models evaluating the interests of a service in a composition. However, contrary to these authors, we are not particularly focusing on service composition, but on service selection in general.

3 Graph-based service selection

3.1 Proposal overview

The graph-based service selection approach proposed in this paper is part of a larger initiative, the MUSIC Project. The MUSIC Project [14] is a focused initiative aiming at the development of context-aware self-adapting applications. The main target is to support both the development and run-time management of software systems that are capable of being adapted to highly dynamic user and execution context, and to maintain a high level of usefulness across context changes. MUSIC adopts a component-based architecture, on which modeling languages allow the specification of context dependencies and adaptation capabilities. Such adaptation capabilities are based on the specification, at design time, of multiple variations (implementations) for each component. The selection of the most appropriate variation is performed by the MUSIC middleware, during run-time execution, based on the context dependencies associated with each variant and based on the current execution context.

In addition to MUSIC components, the MUSIC project aims at exploiting SOA by allowing MUSIC applications to use external services (*i.e.* services that are executing on non-MUSIC nodes). When considering those external services, we are interested in exploiting variability and non-functional properties of context-aware services in a similar way we consider for native MUSIC components. In other words, we consider that several service implementations can supply the same functional capabilities (with a similar syntax), but with different non-functional context-related properties.

The graph-based service selection approach proposed here contributes to the service selection mechanism used by the MUSIC Middleware for selecting the most suitable service among discovered and compatible services. Using this approach, the MUSIC Middleware compares context-aware service descriptions and current execution context in order to select most suitable service, considering current situation. The proposed service selection mechanism assumes that suitable services exist. It is part of a two-step process in which the first step selects all services whose functional properties match the functional requirements that are needed. This means our approach (the second step), dealing with non-functional requirements, is employed only after suitable services are discovered. So the proposal premises is the following: if there are several discovered services able to satisfy a request formulated by a user, one has to select the service that suits best the current execution context. Such service selection should take into account the fact that context information is naturally dynamic and incomplete.

We focus our approach on non-functional context-related aspects of service description. Indeed, we do not investigate functional aspects (inputs and outputs) of a service, but only non-functional contextual conditions related to the execution environment of a service. We consider that functional aspects of a service have the priority, since mismatching on service input or output may have negative (even disastrous) effects on the running application. Incompleteness on service input or output entries (missing input or output) can lead to severe exceptions (or errors), which may affect correctness and execution flow on both service and calling application. Thus, we decide to focus on non-functional aspects of context-aware services, assuming a selection process for meeting functional requirements already took place.

We consider that each context-aware service describes a set of “contextual” conditions (non-functional properties) describing context elements needed for using it appropriately (in the best conditions). For instance, considering a content sharing service (*e.g.* a photo sharing), several variations of this service can be proposed using different implementations (*e.g.* implementations focusing a given user profile, a particular location, etc.). These contextual conditions refer potentially to any observed context element and they can be expressed using the MUSIC context model [17].

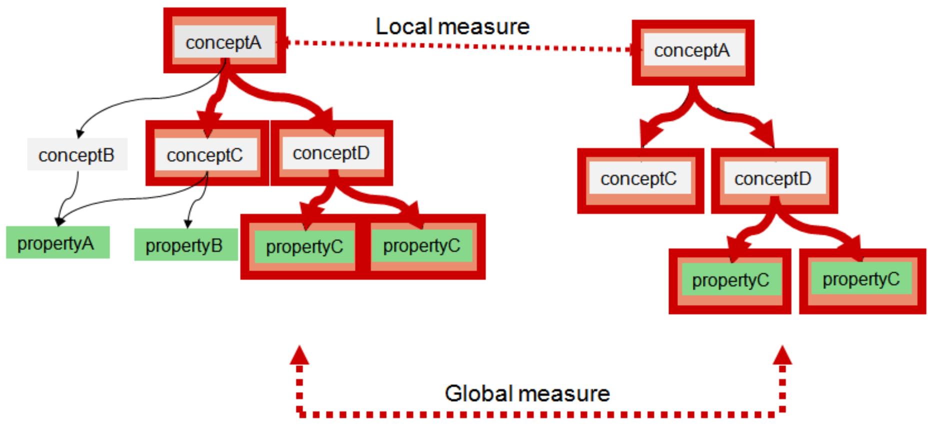


Fig. 1. Local and global measures comparing two graphs.

In order to perform service selection based on a “contextual” matching, service descriptions are enriched with non-functional context-aware properties related to the execution environment most suited for the service. Such requirements are included in the service profile description, using OWL-S. Such contextual description is analyzed as a graph, in which objects represent concepts and properties and edges represent the relations connecting such concepts. The same analysis is performed on the description of the current execution context, which is represented based on an OWL-ontology, and which acts like a “request” (requested execution environment) for the service. This allows us to compare both based on similarity measures between graphs. The proposed service selection algorithm then ranks the available services, indicating to the MUSIC middleware (our user) the services that best match the current context.

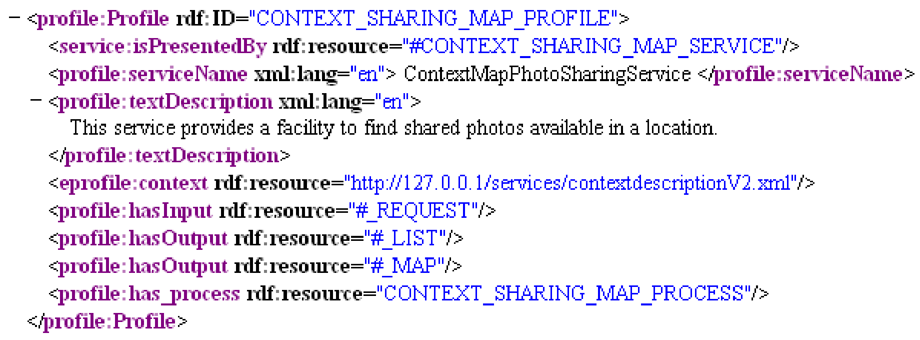
In order to compare the graphs built using service description and current context description, we propose local and global similarity measures. Local measures compare two nodes individually, considering only the concept it represents and its properties. Global measures take into account the graph as a whole, evaluating, for instance, the proportion of similar elements in both graphs. By using such measures, our approach allows dealing with incomplete context information and inexact matching between conditions expressed in the service description and current context description, since missing information on the latter will not block the analysis and the ranking of the former. This means that the selection looks for the service that matches the best the contextual conditions, but is not necessarily a perfect match. Fig. 1 illustrates these measures. It shows a local measure comparing two individual concepts labeled *conceptA*, and a global measure comparing the graphs formed by these concepts (highlighted in Fig.1). Moreover, this approach assumes that several measures can be considered in order to evaluate local values. These local measures are associated to particular context scopes defined in the MUSIC ontology, taking into account the semantic aspects represented in the ontology.

3.2 Describing context-aware services

Service descriptions are expressed in OWL-S. According to Suraci *et al.* [18], "*for describing the semantics of services, the latest research in service-oriented computing recommends the use of the Web Ontology Language for Services (OWL-S).*" These authors consider that, even if OWL-S is tailored for Web Services, it is rich and general enough to describe any service. We consider to enrich this description with context information describing the execution context for which the service is best suited. For instance, let us consider a mobile content sharing platform that enables users to browse, search for, and share multimedia content scattered on such devices in different situations, such as conferences, shopping malls, football stadiums, etc. This scenario foreseen by the MUSIC Project is called Instant Social [7] and it proposes to explore cooperating multi-user applications hosted on mobile devices carried by users. In this scenario, several content sharing services can be available on the platform. Each service can indicate contextual conditions in which it runs appropriately. For example, a given photo sharing service can be particularly designed considering client devices with high screen resolution and memory capacities, a second implementation of the same service can be designed considering a particular location (a conference hall or a stadium), or a particular user profile (*e.g.* adult users).

Such contextual information can be considered as part of the service description, since it indicates situations to which the service is better suited. A service description in OWL-S includes three main parts [12]: (i) service profile; (ii) service model; and (iii) service grounding. The service profile corresponds roughly to the service description. The service model specifies the process executed by the service. The service grounding indicates how the service can be accessed (like an API).

Thus, similarly to Suraci *et al.* [18], we propose to enrich the service profile with a "*context*" element pointing to context description related to the service. This description should be included in an external file (indicated in the "*context*" element) and not directly in the OWL-S description. Context information is dynamic and cannot be statically stored on the service profile. On the one hand, context properties related to the execution of a service can evolve and vary according to the service execution environment itself. For instance, the load of the device executing a service may affect the service and consequently the context properties related to it. On the other hand, the service profile is supposed to be a static description of the service in the sense that it is not supposed to change in short intervals of time (as context information does). An external file describing contextual non-functional requirements and properties related to a service allows the service supplier to easily update such context information related to the service without modifying the service description itself. Fig. 2 presents an example of service profile including the "*context*" element. This example illustrates the extended profile of a photo sharing service, like those foreseen in MUSIC project scenario. This service returns, for a given request on input, a list of interesting photos and a map locating them. As stated before, such a service may have different implementations, considering particular contexts. The one related to this particular implementation is given in Fig. 3.



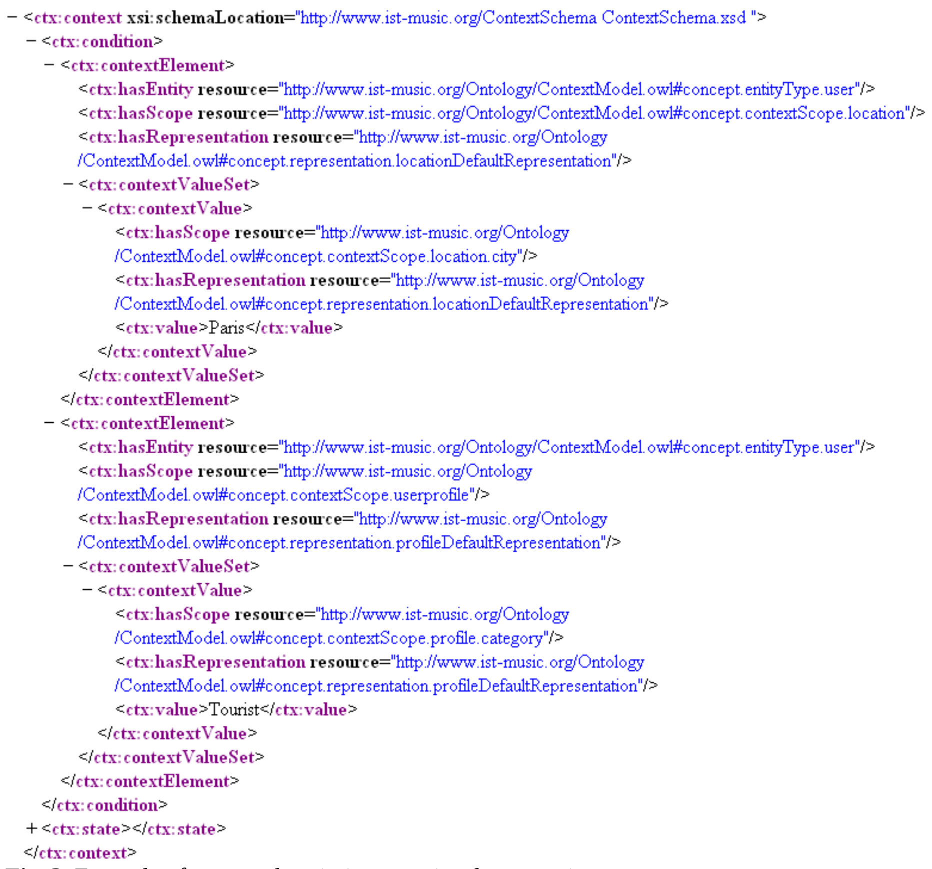
```
- <profile:Profile rdf:ID="CONTEXT_SHARING_MAP_PROFILE">
    <service:isPresentedBy rdf:resource="#CONTEXT_SHARING_MAP_SERVICE"/>
    <profile:serviceName xml:lang="en"> ContextMapPhotoSharingService </profile:serviceName>
  - <profile:textDescription xml:lang="en">
      This service provides a facility to find shared photos available in a location.
    </profile:textDescription>
    <eprofile:context rdf:resource="http://127.0.0.1/services/contextdescriptionV2.xml"/>
    <profile:hasInput rdf:resource="#_REQUEST"/>
    <profile:hasOutput rdf:resource="#_LIST"/>
    <profile:hasOutput rdf:resource="#_MAP"/>
    <profile:has_process rdf:resource="CONTEXT_SHARING_MAP_PROCESS"/>
  </profile:Profile>
```

Fig. 2. Example of service profile including the property "context".

Fig. 3 presents an example of a context description related to the service in Fig. 2. This description follows the MUSIC Context Model described in Reichle *et al.* [17]. The MUSIC context modeling approach identifies three basic layers of abstraction that correspond to the three main phases of context management: the conceptual layer, the exchange layer and the functional layer.

The conceptual layer enables the representation of context information in terms of *context elements*. The *context elements* provide context information about *context entities* (the concrete subjects the context data refers to: a user, a device, etc.) belonging to specific *context scopes*. Such context scopes are intended as semantic concepts belonging to a specific ontology described in OWL. Moreover, the ontology is used to describe relationships between entities, *e.g.* a user has a brother. The exchange layer focuses on the interoperability between devices. Context data in this layer is represented in XML and is used for communication between nodes. The functional layer refers to the implementation of the context model internally to the different nodes.

The description illustrated in Fig. 3 belongs to the exchange level, since it is used for information exchange among different nodes. Thus, context information in Fig. 3 is described in XML by context elements, which refer to a given entity and scope, and a set of context values, which also refer to a given scope. It is worth noting that Fig. 3 supplies two separate context descriptions: (*i*) a first description (under the element "*condition*") supplying the conditions under which this service adapts the best (*i.e.* the contextual situation in which it is most appropriate to call this service); and (*ii*) a second description referring to the current state of the service execution context (under which conditions this service is running on the service supplier). Thus, through the condition element in Fig. 3, the service supplier indicates that the content supplied by this service implementation (whose profile is represented in Fig. 2) is proper to tourist users (who are familiar with the city they are visiting) and that this service disposes of a detailed database for the city of Paris, which makes it better adapted to being used when in this location. The next section describes how the proposed graph-based matching algorithm considers and handles these descriptions.



```
– <ctx:context xsi:schemaLocation="http://www.ist-music.org/ContextSchema ContextSchema.xsd ">
  – <ctx:condition>
    – <ctx:contextElement>
        <ctx:hasEntity resource="http://www.ist-music.org/Ontology/ContextModel.owl#concept.entityType.user"/>
        <ctx:hasScope resource="http://www.ist-music.org/Ontology/ContextModel.owl#concept.contextScope.location"/>
        <ctx:hasRepresentation resource="http://www.ist-music.org/Ontology
        /ContextModel.owl#concept.representation.locationDefaultRepresentation"/>
      – <ctx:contextValueSet>
        – <ctx:contextValue>
            <ctx:hasScope resource="http://www.ist-music.org/Ontology
            /ContextModel.owl#concept.contextScope.location.city"/>
            <ctx:hasRepresentation resource="http://www.ist-music.org/Ontology
            /ContextModel.owl#concept.representation.locationDefaultRepresentation"/>
            <ctx:value>Paris</ctx:value>
          </ctx:contextValue>
        </ctx:contextValueSet>
      </ctx:contextElement>
    – <ctx:contextElement>
        <ctx:hasEntity resource="http://www.ist-music.org/Ontology/ContextModel.owl#concept.entityType.user"/>
        <ctx:hasScope resource="http://www.ist-music.org/Ontology
        /ContextModel.owl#concept.contextScope.userprofile"/>
        <ctx:hasRepresentation resource="http://www.ist-music.org/Ontology
        /ContextModel.owl#concept.representation.profileDefaultRepresentation"/>
      – <ctx:contextValueSet>
        – <ctx:contextValue>
            <ctx:hasScope resource="http://www.ist-music.org/Ontology
            /ContextModel.owl#concept.contextScope.profile.category"/>
            <ctx:hasRepresentation resource="http://www.ist-music.org/Ontology
            /ContextModel.owl#concept.representation.profileDefaultRepresentation"/>
            <ctx:value>Tourist</ctx:value>
          </ctx:contextValue>
        </ctx:contextValueSet>
      </ctx:contextElement>
    </ctx:condition>
  + <ctx:state></ctx:state>
  </ctx:context>
```

Fig. 3. Example of context description associated to a service.

4 Graph-based matching

4.1 From description to graphs

The first step for performing the graph-based matching is to analyze the context description associated with the available service. Based on the context description presented above, we propose a graph-based approach for ranking and selecting services. In this approach, non-functional context-related properties of the services represented in the context description file described previously are interpreted as a graph. In this graph, nodes represent the context elements indicated in this description, and the edges represent the relations that can exist between these elements. The same interpretation is used when analyzing the current execution context. The MUSIC middleware is responsible for service selection and for collecting and managing context information related to the user. It keeps this information in context elements expressing their current values. These context elements are seen as graph nodes, whereas relations between such elements are seen as graph edges. Thus, a graph G is defined as follow:

- $G = < N, E >$ where:
  - $N = \{ C_{Ei} \}_{i>0}$: set of context elements C_{Ei} ;
  - $E = \{ < C_{Ei}, C_{Ej} > \}$: set of relations between context elements C_{Ei} and C_{Ej}.

Thus, comparing two graphs representing two different context descriptions corresponds, with regard to the MUSIC Context Model, to comparing two sets of context elements and their relations.

4.2 Matching algorithm

Once all available services have been analyzed and their corresponding graphs are created, the matching based algorithm may proceed. The goal of this matching algorithm is to rank the available services based on their contextual non-functional properties. It compares the graph generated by each proposed service to the graph created based on the current execution context information. This matching starts by comparing nodes from both graphs (from the context description of the service and from the current context) individually, using local similarity measures. Based on the results of these measures, the matching algorithm compares the graphs globally, using global similarity measures that also consider the edges connecting the nodes. The results of such global measures are used to rank the services corresponding to the compared graphs. Next sections present both local and global similarity measures.

4.2.1 Comparing graph nodes: local similarity measures

When comparing two nodes from two graphs defined in Section 4.1, we are comparing two *context elements* representing context information about a given entity and referring a given scope. By considering these elements individually, we focus on how similar their context values are. In order to perform this comparison, we consider local similarity measures $Sim_l (C_{Ei}, C_{Ej})$ that compares two context elements C_{Ei} and C_{Ej} locally (i.*e.* without considering their position in the corresponding graphs). This measure can be defined as follows:

- $Sim_l (C_{Ei}, C_{Ej}) =$ x, where $x \in \mathbb{R}, x \in [0, 1]$

Ideally, the similarity measure $Sim_l (C_{Ei}, C_{Ej})$ depends on the context scope. If the context elements being compared do not belong to compatible context scopes, their similarity is by definition zero. For example, we cannot compare context elements referring to the user's age or preferences with context elements referring to the user's location because both elements belong to context scopes that are incompatible. Similarity measure $Sim_l (C_{Ei}, C_{Ej})$ has to consider the representation associated with the context elements. In the MUSIC context modeling approach each context element is associated with a corresponding representation. For instance, considering location information, this can be represented using geographical coordinates like latitude and longitude (*e.g. 48°49'38" N, 2°21'02" E*), as well as using a representative name (*e.g.*

Paris, France). Each measure *Sim_l (C_{Ei}, C_{Ej})* is proposed considering a given set of possible representations, which it may handle. Only context elements that correspond to the context scope and representation supported by the giving measure can be compared using it. The MUSIC middleware keeps then a library with all knows similarity measures *Sim_l (C_{Ei}, C_{Ej})*. Before comparing two nodes, it looks for the appropriate measure in its library.

Once the appropriate similarity measure *Sim_l (C_{Ei}, C_{Ej})* is chosen, the matching starts by taking each node in the graph corresponding to the context description of the service and comparing it to the nodes with a compatible scope and representation from the graph corresponding to the current execution context. For each node, it keeps tracks of the best-ranked node, in order to use this value in the global similarity measures (Section 4.2.2). Thus, being $G_{Sk} = < N_{Sk}, E_{Sk} >$ the graph corresponding to the service S_k and $G_C = < N_C, E_C >$ the graph corresponding to the current context, we compare each node C_{Ei} from G_{Sk} to all nodes C'_{Ei} in G_C for which *C_{Ei}.scope* and *C'_{Ei}.scope* and *C_{Ei}.representation* and *C'_{Ei}.representation* are compatible, keeping in memory the best-ranked C'_{Ei}. For example, considering the graph generated by the context description in Fig. 3, the node referring to the user's profile is compared to all nodes having the same scope (*user profile*) in the graph corresponding to current user's context.

4.2.2 Comparing graphs: global measures

The main goal of global similarity measures is to compare overall composition of two graphs, taking into account both nodes and edges composing each graph. We define such measures as follow:

- $Sim_g (G_{Sk} , G_C) = x, x \in \mathbb{R}$, where
  - G_{Sk} corresponds to the graph determined by the context description of the service;
  - G_C corresponds to the graph determined based on the current execution context.

Several global measures $Sim_g (G_{Sk} , G_C)$ are possible for comparing two graphs. These measures can be based on different well-know algorithms such as subgraphing matching or graph isomorphism. The most important aspect for us is that the global similarity measure $Sim_g (G_{Sk} , G_C)$ must support incompleteness of context information represented in these graphs. This means that the $Sim_g (G_{Sk} , G_C)$ should not stop processing if some context information is missing. For instance, if the context description of a service refers to a given context element for which there is no corresponding element with a compatible context scope in the current context description, the similarity measure $Sim_g (G_{Sk} , G_C)$ should continue the processing, arriving in a valuable result that takes into account this fact.

In the MUSIC middleware, we consider a single yet powerful similarity measure $Sim_g (G_{Sk} , G_C)$ defined based on the proportion of nodes and edges belonging to the context description of the service that have a similar correspondence in the current context description. For this, the similarity measure considers the results obtained by the local similarity measures. For each pair $<C_{Ei} , C'_{Ei}>$, with $C'_{Ei} \in G_{Sk}$ and $C'_{Ei} \in G_C$ and C'_{Ei} being the node of G_C with the greatest value for $d(C_{Ei} , C'_{Ei})$, the proposed measure $Sim_g (G_{Sk} , G_C)$ analyses the similarity among the edges connecting these nodes to their neighbors. The similarity between two edges is calculated based on the similarity of their corresponding labels (or weights), if the edges are labeled, and the similarity between the objects forming the edges. Similarly to the local measures, we consider in the global measure only the greatest value obtained when comparing each edge connecting a node C_{Ei}. Then, we sum up both nodes and edges best similarities measures and make the proportion taking into account the total number of nodes and edges in graph defined by the context description of the service. Fig. 4 shows the definition of the measure Sim_g (G_{Sk}, G_C).

It is worth noting that, since the maximum value for Sim_l (a,b) is *1* (cf. Section 4.2.1), if the graph G_{Sk} is a subgraph of G_C, for each node and edge, we will have a corresponding node or edge for which the local similarity measure is 1. Thus, by considering the proportion of the greatest values obtained for all individual nodes and edges in the total size of the graph, this measure considers implicitly that some nodes or edges may have no similar element ($max(Sim_l\ (a,b))=0$). This eventuality leads to a reduction in the value of the global similarity measure Sim_g (G_{Sk}, G_C), but it does not prevent a valuable result. Even if the compared graphs have no element in common ($max(Sim_l\ (a,b))=0$ for all $a \in G_{Sk}$ and $b \in G_C$), the measure Sim_g (G_{Sk}, G_C) still returns a value that can be used to rank the service. For instance, when considering the photo sharing service represented Fig. 2, the measure Sim_g (G_{Sk}, G_C) gives a valuable result ($x \geq 0$) even if the current user's context does not possess any context element referring to the location scope (user's device has no GPS or any location sensor available). This resulting value is then used to rank this particular implementation of photo sharing service. Incompleteness of context information is dealt with in this way.

Considering that:

$$\text{if } \mathbf{G_{S_k}} = \langle \mathbf{N}, \mathbf{E} \rangle \text{, where } \begin{array}{c} \mathbf{N} = \{\mathbf{C_{E_i}}\}_{1 \leq i \leq n} \text{ and} \\ \mathbf{E} = \{\langle \mathbf{C_{E_i}}, \mathbf{C_{E_j}} \rangle_{\mathbf{k}}\}_{0 \leq k \leq m} \end{array} \quad then\ |\boldsymbol{G_{S_k}}| = \boldsymbol{n} + \boldsymbol{m}$$

And considering two edges E_i and E_j that:

$$Sim_l(E_i, E_j) = \frac{Sim_l(l_i, l_j) + \sum_1^p Sim_l\left(C_{E_i}, C_{E_j}\right)}{(p+1)}, where \begin{array}{l} l_i\ and\ l_j are\ the\ edges\ labels, and \\ C_{E_i}\ and\ C_{E_j} are\ edges\ extremeties \end{array}$$

$Thus, Sim_g\left(G_{S_k}, G_C\right)\ can\ be\ defined\ as:$

$$Sim_g\left(G_{S_k}, G_C\right) = \frac{\sum max\left(Sim_l(C_{E_i}, C'_{E_i})\right) + \sum max\left(Sim_l(E_j, E'_j)\right)}{|G_{S_k}|}$$

Fig. 4. Definition of the global similarity measure *Sim*$_g$.

5 Conclusions

In this paper, we present a graph-based approach for service selecting in ubiquitous computing. The main goal of this approach is to select the most adapted service for the current situation. We compare contextual non-functional properties of context-aware services to the current execution context in which they are called. Our approach considers particularly the natural incompleteness of context information when selecting a context-aware service among all available services. For this, our approach is based on a graph-based analysis of both current context situation and context description associated with the service. This analysis is the basis for a set of similarity measures that compare graphs representing these descriptions. Such measures allow us to compare graphs that represent context information by considering scope and incompleteness of such information.

Currently, we are testing the proposed approach with the MUSIC middleware in order to evaluate its performance in ubiquitous environments. We also intend to compare our results with other libraries of similarity measure such as SimPack [20].

Acknowledgements. The authors would like to thank their partners in the MUSIC-IST project and acknowledge the partial financial support given to this research by the European Union (6th Framework Programme, contract number 35166).

References

[1] Baldauf, M.; Dustdar, S. & Rosenberg, F., A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, n.4, 2007, 263–277

[2] Ben Mokhtar, S.; Kaul, A.; Georgantas, N. & Issarny, V., Efficient Semantic Service Discovery in Pervasive Computing Environments, Proceedings of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06), 2006

[3] Bottaro, A.; Gerodolle, A. & Lalanda, P., Pervasive service composition in the home network, 21st International IEEE Conference on Advanced Information Networking and Applications (AINDA'2007), 2007

[4] Chalmers, D.; Dulay, N. & Sloman, M., Towards Reasoning About Context in the Presence of Uncertainty, 1st international workshop on advanced context modelling, reasoning and management, Nottingham, UK, September 2004

[5] Dey, A., Understanding and using context. Personal and Ubiquitous Computing, vol. 5 n. 1, 2001, 4-7.

[6] Eikerling, H.-J.; Mazzoleni, P.; Plaza, P.; Yankelevich, D. & Wallet, T., Services and mobility: the PLASTIC answer to the Beyond 3G challenge. White Paper, PLASTIC Project, December 2007. http://www-c.inria.fr/plastic/dissemination/plastic/dissemination

[7] Fraga, L.; Hallsteinsen S. & Scholz, U., InstantSocial – Implementing a Distributed Mobile Multi-user Application with Adaptation Middleware, Communications of the EASST, vol. 11. http://eceasst.cs.tu-berlin.de/index.php/eceasst/issue/view/18.

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

[9] Klusch, M.; Fries, B. & Sycara, K., Automated semantic web service discovery with OWLS-MX, Proceedings of the 5th International joint conference on Autonomous agents and multiagent systems (AAMAS '06), ACM, 2006, 915-922

[10] Liao, T.W.; Zhang, Z. & Mount, C.R., Applied Artificial Intelligence, Volume 12, Number 4, 1 June 1998 , Taylor & Francis, 267-288

[11] Maamar, Z.; Benslimane, D. & Narendra, N. C., What can context do for web services?, Communication of the ACM, vol. 49, n° 12, Dec. 2006, 98-103

[12] Martin, D. (Ed.), OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004, http://www.w3.org/Submission/OWL-S

[13] Moran, T. & Dourish, P., Introduction to this special issue on context-aware computing. Human-Computer Interaction, vol. 16, n. 2-3, 2001, 87–95

[14] MUSIC Consortium, Self-Adapting Applications for Mobile Users in Ubiquitous Computing Environments (MUSIC), Website: http://www.ist-music.eu/

[15] Papazoglou, M. P. & Georgakopoulos, D. Service-Oriented Computing, Communication of ACM, vol. 46, n. 10, Oct. 2003, 24-28

[16] Patouni, E.; Alonistioti, N. & Polychronopoulos, C. (eds.), Service Adaptation over Heterogeneous Infrastructures, White Paper, 2008. http://www.opuce.tid.es/Publications.htm

[17] Reichle, R.; Wagner, M.; Khan, M.U.; Geihs, K.; Lorenzo, L.; Valla, M.; Fra, C.; Paspallis, N. & Papadopoulos G.A. A Comprehensive Context Modeling Framework for Pervasive Computing Systems, 8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS), 4-6 June, 2008, Oslo, Norway, Springer.

[18] Reiff-Marganiec, S., Qing Yu, H., Tilly, M., Service Selection based on Non-Functional Properties, NFPSLA-SOC07 Workshop at The 5th International Conference on Service Oriented Computing (ICSOC2007), Sept. 17 2007, Vienna, Austria.

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

[20] SimPack Project Page, http://www.ifi.uzh.ch/ddis/research/semweb/simpack/

[21] Toninelli, A.; Corradi, A. & Montanari, R., Semantic-based discovery to support mobile context-aware service access, Computer Communications, vol. 31, n° 5, March 2008, 935-949.

[22] Weiser, M. The computer for the 21st century, Scientific American, vol. 66, 1991.

[23] Yau, S. S. & Liu, J., Incorporating situation awareness in service specifications, In: Lee, S.; Brinkschulte, U.; Thuraisingham, B. & Pettit, R. (Eds.), 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 2006), 2006, 287-294

[24] Zarras, A.; Fredj, M.; Georgantas, N. & Issarny, V., Engineering reconfigurable distributed software systems: issues arising for pervasive computing, In: Butler, M.; Jones, C.; Romanovsky, A. & Troubitsyna, E. (Eds.), LNCS 4157, Rigorous Development of Complex Fault-Tolerant Systems, Spring, 2006, 364-386