Context-Aware Service Composition in Pervasive Computing Environments

Sonia Ben Mokhtar, Damien Fournier, Nikolaos Georgantas, and Valérie Issarny

INRIA Rocquencourt 78153 Le Chesnay, France
{Sonia.Ben_Mokhtar, Damien.Fournier, Nikolaos.Georgantas,
Valerie.Issarny}@inria.fr
http://www-rocq.inria.fr/arles/

Abstract. A major challenge in pervasive computing environments is to provide users with complex, context-sensitive applications, dynamically composed from networked services. In this paper, we present an approach to the dynamic, context-aware composition of services to perform user tasks, i.e., software applications abstractly described on the user's handheld device. Both networked services and user tasks are modeled as semantic Web services in OWL-S extended with context information. The distinctive feature of our solution is the ability to compose Web services that expose complex behaviors (conversations) to realize a user task that itself has a complex behavior. Furthermore, the context-related requirements of the task are met by aggregating the context-sensitive behaviors of the individual services.

1 Introduction

The user-centric view promoted by the pervasive computing paradigm advocates placing less demand on user attention [25]. Thus, pervasive computing applications need to be more autonomous and sensitive to context. In this domain, one of most challenging objectives to be achieved is to automatically enable software applications by dynamically composing services of the pervasive environment. In this direction, our work presented herein aims at enabling users to perform tasks (i.e., abstract user applications modeled as workflows) on the fly by composing various networked service capabilities provided in the pervasive environment. While service composition allows some degree of autonomy to be achieved, context-awareness allows user applications to be more sensitive to the environment's changes, leading to a higher level of autonomy and adaptation.

A number of research effort have been conducted in the area of context-aware systems. In particular, various models have been proposed to represent context information, among which attributes and values models, object oriented models and ontologies. Ontologies have proved to be the most suitable model for representing and reasoning on context information for the following reasons [5] : (i) ontologies enable knowledge sharing in open, dynamic systems; (ii) ontologies with well defined declarative semantics allow efficient reasoning on context information; and (iii) ontologies enable service interoperability and provide the means for networked services to collaborate in a non-ambiguous manner. The Ontology Web Language (OWL[1]) is a recent W3C

[1] OWL: Ontology Web Language. http://www.w3.org/TR/owl-ref/

N. Guelfi and A. Savidis (Eds.): RISE 2005, LNCS 3943, pp. 129–144, 2006.

recommendation for formally specifying ontologies. A number of context ontologies proposed in the literature are based on OWL.

Most existing approaches to context-aware systems propose either a user-centric or a service-centric view of context. User-centric approaches promote applications that move with the users and follow their preferences [20]. Service-centric approaches promote service adaptability to the context changes [12]. We propose a solution that combines both views. Indeed, both users-related and services-related contextual requirements are taken into account in our approach to service composition. This leads us to extend the Ontology Web Language for Services (OWL-S[2]) for modeling services-related context but also user tasks contextual requirements. Our OWL-S-based model for services and user tasks has various advantages. First, our model captures both services-related and user-related context enabling service composition to effectively be context-sensitive. Second, as our model is based on OWL-S, it is easily extensible. Third, our model can employ any existing context ontology to describe context information.

We present in this paper a solution for context-aware service composition based on workflow integration. More precisely, both networked services and user tasks have workflow descriptions in OWL-S enriched with context, and our aim is to integrate services' workflows to realize a target user task, further enabling context-awareness.

The remainder of this paper is structured as follows. The next section introduces definition of context and context awareness and presents effort on modeling context. In Section 3, we present our context model extending OWL-S for modeling context-aware services and tasks. Then, we present our approach to context-aware service discovery and composition in Section 4. Further in Section 5, we employ a scenario inspired from the networked home environment, as investigated in the IST Amigo project[3], that we adopt in our work to illustrate our solution. Finally, in Section 6 we review related research effort in the area of context-aware service composition and conclude with a summary of our contribution and future work.

2 Background

In this section, we present our adopted definition of context and context-awareness (§2.1). We also survey various efforts that have been undertaken for modeling context (§2.2).

2.1 Context-Awareness

Definitions of context and context-awareness are rather subjective in the literature. We adopt the generic definitions proposed by Dey et al. [6]:

*"**Context** is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."*

[2] OWL-S: Semantic Markup for Web Service. http://www.daml.org/services/owl-s

[3] Amigo: Ambient intelligence for the networked home environment. http://www.extra.research.philips.com/euprojects/amigo/

*"**Context-awareness** is a property of a system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."*

Different categories of context can be distinguished. For example, Schilit defines three categories of context [21]: (i) *Device context*, which is contextual information related to devices, such as available CPU, memory, reachable networks etc.; (ii) *User context*, including the user's profile and preferences, but also information about the user's applications; and (iii) *Physical context* such as location, weather, light etc. All these contextual information coming from various entities in the environment (e.g., sensors, applications, devices) are heterogeneous and have to be represented in a well defined shared model in order to be understandable by user applications.

2.2 Context Modeling

Context modeling defines how context data are structured and maintained which play a key role in supporting efficient context management. Context modeling mainly relies on two important features [29]. First, it is essential to adopt a flexible structure with explicit concepts and associations to model knowledge, define semantic relations and enable data sharing and reuse. Second, logic reasoning or inference mechanisms on raw data are necessary to deduce high-level contextual information from low-level context data (e.g., sensed data). Choosing an adequate context model is crucial to enable context-aware services. We survey below, proposed approaches to model context information.

The most simple way to model and maintain context information is to define a set of context attributes and associated values. The most known system using this kind of model, is the PARC's mobile computing environment, proposed by Schilit *et al.* in [22]. This is an easy solution to enable sharing of context information among applications. But, although it is fast and easy to set and update context modeled with such a data structure, context-awareness is only possible if applications use attributes with the given names. Moreover, such a model is not sufficient for describing concepts and associations between them, and for making abstraction of raw data. Markup languages are another way to model contextual knowledge. Flexibility and structural properties are strong advantages of this approach. In context-aware systems, markup languages are commonly used to describe profiles. For example, CC/PP (Composite Capabilities / Preferences Profile) is an RDF-based language designed for describing user preferences and hardware capabilities, well suited for mobile computing. A number of research effort have been made to adapt CC/PP to define context (e.g., [9]). Markup languages are simple, flexible, structured, and seem to be suitable for pervasive computing. But, data ambiguity must be solved by applications, complex relationships between data can not be defined, and a more formal definition of context can not be expressed. Another approach to model context, lies in using object-oriented concepts [26]. Object-oriented modeling allows defining a structured and scalable context model. Object implementation further uses aggregation mechanisms for abstraction, retrieving high-level concepts and solving data ambiguity and consistency. But, object-oriented models are often defined for a specific context domain. Sharing data and interoperability between different context applications may be difficult. Ranganathan *et al.* [19], and Seng W. Loke [11] have tried to model context with predicates and logic deduction. Modeling context using

first order logic allows higher-level context retrieval. However, this model lacks defining structures and relations between context information and does not resolve ambiguity between context data.

The last modeling approach that we consider in this section is the use of ontologies. Ontology-based models are very close to the requirements of context modeling. While the overall objective of using ontologies is to define common vocabularies, context modeling in pervasive computing uses ontology models to represent relevant information for users. Like many recent approaches, we choose to define context using ontologies built upon the Ontology Web Language (OWL) for the following reasons:

- Ontologies allow defining concepts, entities, properties and also relationships between concepts.
- Since OWL is based on RDF Schema, OWL ontologies can be validated with tools as Jena[4] or OWLP[5].
- OWL is dynamic and flexible, allowing context data to be easily added, deleted, and updated with programming interfaces for OWL (e.g., Jena, OWL API[6]).
- Many tools have been developed for reasoning on OWL ontologies to deduce abstract concepts (e.g., Racer[7], Jena).
- Knowledge sharing can be achieved between heterogeneous context sources.

Among recent approaches to ontology-based context modeling, SOUPA [5] and CONON [29] define two-level ontologies for context modeling. A core ontology defines generic concepts that are usually modeled in context such as platform or users, while more specific ontologies introduce concepts for a particular application domain such as characteristics of users and devices, and location (vertical extensions), or the definition of intelligent environments such as home or office (horizontal extensions). SOUPA also addresses security and privacy with the protection of users' personal information, and proposes a set of policies to restrict data access. CONON focuses on characterizing user situation with a set of user-defined first order logic rules. More generic, Preuveneers *et al.* in [17] define a simple context ontology which is easily extensible and allows the description of semantic Web services. Interoperability is also studied in [3], which proposes an approach for mapping concepts between different ontologies.

3 Modeling Context-Aware Services and Tasks

Our objective is to allow a user to perform a task any where and at any time, on the fly, without any previous knowledge about the services available in the environment. Moreover, accounting for both the user's actual context and the services' contextual requirements, but also ensuring a QoS that fulfills the requirements specified by the user. In our approach, we consider as QoS all context information related to resource consumption such as the device context (§2.1). In this paper, we focus on managing the remaining context information, such as user context and physical context, while QoS management has been described in [2].

[4] Jena: A Semantic Web Framework for Java. http://jena.sourceforge.net/

[5] OWLP: http://www-db.research.bell-labs.com/user/pfps/owlp/

[6] OWL API: sourceforge.net/projects/owlapi

[7] Racer: http://www.sts.tu-harburg.de/ r.f.moeller/racer/

3.1 OWL-S Based Context Model for Pervasive Services

Semantic Web services results from the combination of the Semantic Web and Web Services. A number of research effort have been conducted in order to bring semantics to Web services [16, 7, 24]. These effort aim at the semantic specification of Web Services towards automating Web services discovery, invocation, composition and execution monitoring. In this area, the Ontology Web Language for Services (OWL-S) is the most complete effort for describing semantic Web services. Besides describing high-level capabilities of services, OWL-S allows the description of services' behaviors using conversations. OWL-S defines Web services capabilities in three parts: the Service Profile, the Process Model and the Service Grounding. The Service Profile gives a high-level description of a service and its provider. It is generally used for service publication and discovery. The Process Model describes the service's behavior as a process. This description contains the specification of a set of sub-processes coordinated by a set of control constructs. These control constructs are: Sequence, Split, Split + Join, Choice, Unordered, If-Then-Else, Repeat-While, and Repeat-Until. The sub-processes can be either composite or atomic. Composite processes are decomposable into other atomic or composite processes, while atomic ones correspond to WSDL[8] operations. The Service Grounding specifies the information necessary for service invocation, such as underlying communication protocols, message formats, serialization, transport and addressing information. The Service Grounding defines mapping rules to link OWL-S atomic processes to WSDL operations.

In our approach, networked services are described in OWL-S extended with context information. This information decomposes to: (i) high-level context attributes; and (ii) contextual preconditions and effects. A high-level context attribute makes a service aware of some context information, such as location or physical conditions. Then, the service may use this context information to provide the user with appropriate context-sensitive responses. Contextual preconditions are conditions to be fulfilled for the valid execution of a service, while contextual effects result from this execution and affect the current context. For example the operation Turn_On_The_Light of a Light_Management_Service requires as preconditions that there is some one in the room, the light level is low, and nobody is sleeping; this operation has as effect the lighting of the corresponding room.

OWL-S allows the description of non-functional properties of services by extending the Service Profile ontology. Thus we extend the Service Profile to allow service providers to specify context attributes that characterize a service (see Figure 1). In addition to this, as our composition approach involves services' atomic processes (operations) [1], further context information is needed at the atomic process level. This context information is provided in the form of contextual preconditions and effects. Thus, we propose to extend OWL-S to allow atomic processes to provide contextual preconditions and effects. As shown in Figure 1, contextual preconditions and effects are subclasses of the Condition and Expression classes defined in the OWL-S specification. These classes allow the description of logical expressions in the form of literals

[8] WSDL: Web Services Description Language. http://www.w3.org/TR/wsdl

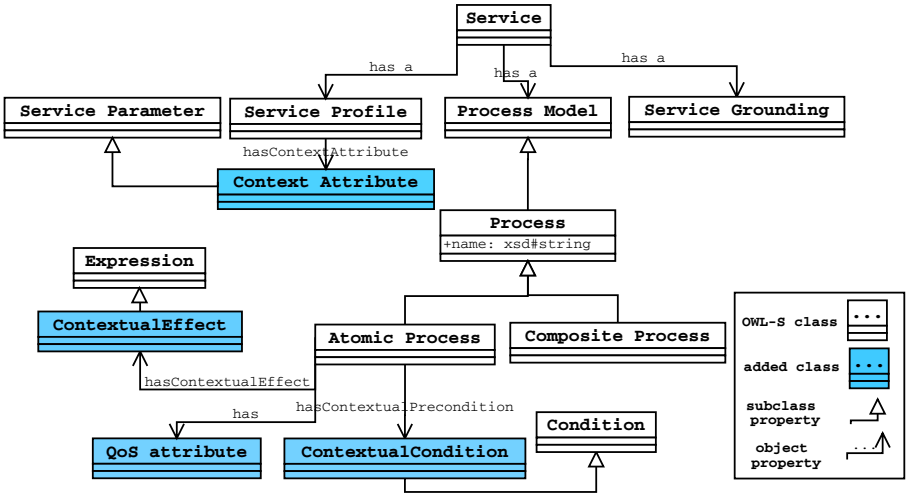

Fig. 1. Context-Aware Service Description

encoded in XML by using existing logic languages such as SWRL[9], KIF[10] and PDDL[11]. Another extension that we have introduced to OWL-S is the specification of QoS attributes at the atomic process level. This allows specifying, among others, resource consumption due to the invocation of a service operation. For example, based on execution histories a service can publish for each operation the latency induced by the invocation on this operation. This information can be further exploited to compose services in a way that fulfills the QoS requirements specified by the user task [2].

3.2 OWL-S Based Context Model for User Tasks

As for networked services, the description of the user task is given in the form of an abstract OWL-S process. Furthermore, we extend the OWL-S process description in order to allow the specification of non-functional requirements, such as QoS requirements and contextual conditions (see Figure 2). QoS requirements correspond to global requirements that have to be fulfilled by the resulting service composition (e.g., $latency < 2sec$, $availability > 80\%$). On the other hand, contextual conditions can be either global or local. A global contextual requirement has to be fulfilled by the resulting service composition. For example, a global contextual requirement can be: "the commission of a travel reservation composite service < 10 euros" or "the distance to be covered by the user wishing to use a printer and a scanner < 300 m". These global requirements have to be checked during the service composition, by aggregating contextual information provided by the individual services. Local contextual requirements

[9] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.daml.org/2003/11/swrl/

[10] KIF: Knowledge Interchange Format. http://logic.stanford.edu/kif/kif.html

[11] PDDL: Planning Domain Definition Language. http://planning.cis.strath.ac.uk/competition/pddl.html

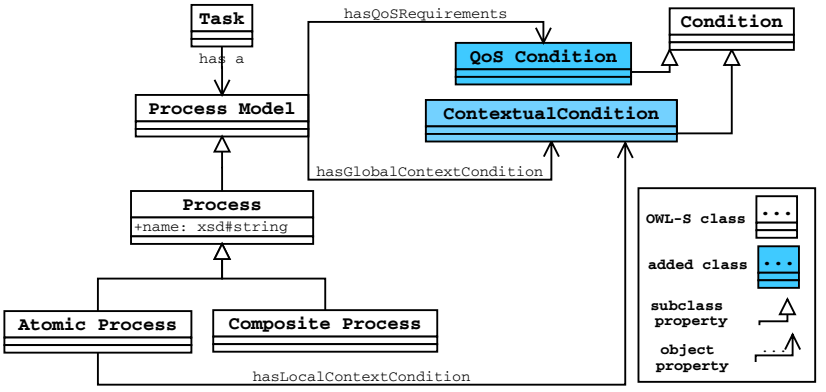

Fig. 2. User task description

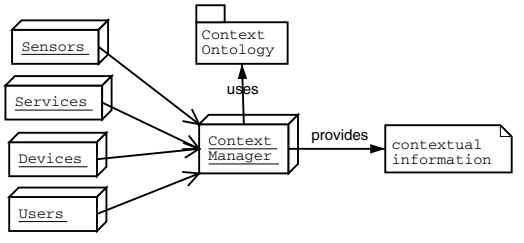

Fig. 3. Context manager service

are associated to a part of the task's description (some of the task's atomic processes) and have to be fulfilled by services' operations. For example, if the user task involves looking for a movie theater, the operation that will be selected will have to take into account the user's actual location and return to the user the nearest movie theater. Thus, the service providing this operation should have as a context attribute the physical distance awareness. Other examples of local contextual requirements can be the selection of the least loaded printer, or the selection of the road with least traffic. The main difference between user task descriptions and service descriptions is that in contrast to the atomic processes involved in the services processes, those involved in user task processes are not bound to any service, since services to be invoked are dynamically discovered. Thus the OWL-S Grounding corresponding to a task's process is generated at runtime from the Groundings of the composed services.

Context management associated with context-sensitive services relies on a context manager, which provides on demand contextual information in terms of a specific ontology as depicted in Figure 3. We do not enforce the use of a specific context ontology, any existing OWL-based context ontology can be used (e.g., [18, 5, 29, 17]). What we define is a model for services and tasks enabling context-aware service composition. This model can easily be used in conjunction with any OWL context ontology. In Figure 3, the context manager collects information from different entities that affects context (e.g., sensors, users, devices, etc.). Then, it uses a context ontology to provide

contextual information formatted in this ontology. The context manager has two modes. A normal running mode in which it returns the actual context, and a simulation mode in which it returns a simulated context. The latter mode is used during the composition process in which we need to know what will be the new context if a specific operation is performed. Further information about the use of the simulation mode is given in Section 4.2. The context manager can be a central entity that aggregates the heterogeneous information provided by various entities in the environment [5], but it can also be distributed [14]. A distributed context manager can be a set of collaborating intelligent agents that exchange context knowledge fragments to synthesize context information.

3.3 Modeling OWL-S Processes as Finite State Automata

Towards dynamic composition of services, we introduce formal modeling of OWL-S processes as finite state automata. Other approaches for formalizing Web services conversations and composition have been proposed in the literature based on Petri nets [27], process algebras [10] or finite state machines [8]. Our objective in using finite state automata is to translate the problem of integrating complex processes to an automata analysis problem. Figure 4 describes the mapping rules that we have defined for translating an OWL-S Process Model to a finite state automaton. In this model, automata symbols correspond to the OWL-S atomic processes (the services operations) involved in the OWL-S process. The initial state corresponds to the root OWL-S composite process, and a transition between two states is performed when an atomic process is executed. Each process involved in the OWL-S Process Model, either

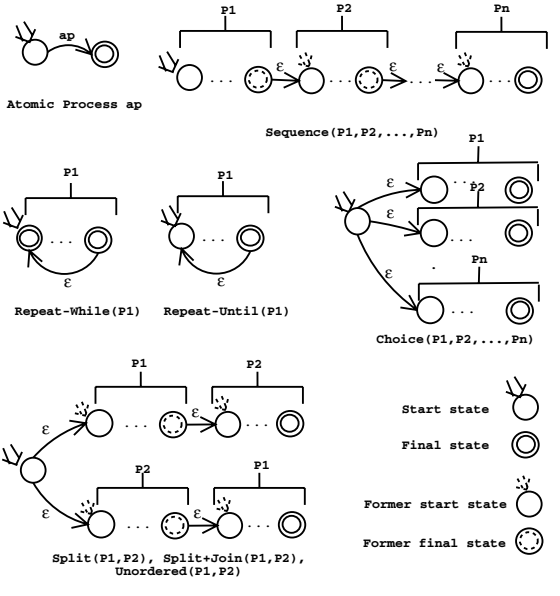

Fig. 4. Modeling OWL-S processes as finite state automata

atomic or composite, is mapped to an automaton and linked together with the other automata in order to build the OWL-S process automaton. This is achieved by following the OWL-S process description and the mapping rules shown in Figure 4. Further details about modeling OWL-S processes as automata can be found in [1].

4 Context-Aware Dynamic Service Composition

Our approach to context-aware service composition is performed in two steps. First, context-aware service discovery provides a set of services that are candidate to the composition (§4.1). Second, starting from the automata descriptions of the selected services and user task, context-aware process integration provides a set of composition schemes that conform to the task's behavior further meeting all the context requirements (§4.2).

4.1 Context-Aware Service Discovery

In this section, we present our semantic- and context-aware service discovery, which aims at selecting a set of services providing atomic processes that are semantically equivalent in terms of provided capabilities to the atomic processes involved in the user task. Additionally, during the discovery of services, the contextual requirements of the task are matched against the context attributes of services. This discovery algorithm compares semantically the atomic processes of the user task with those of the networked services.

Semantic-aware service discovery is based on the matching algorithm proposed by Paolucci *et al.* in [15]. This algorithm is used to match a requested service with a set of advertised ones. The requested service has a set of provided inputs in_{Req}, and a set of expected outputs out_{Req}, whereas each advertised service has a set of expected inputs in_{Adv} and a set of provided outputs out_{Adv}. This matching algorithm defines four levels of matching between two concepts in an ontology, a requested concept C_{Req} and an advertised concept C_{Adv}.

- *Exact*: if $C_{Req} = C_{Adv}$ or C_{Req} is a direct subclass of C_{Adv}
- *Plug in*: if C_{Adv} subsumes[12] C_{Req}, in other words, C_{Adv} could be used in the place of C_{Req}
- *Subsumes*: if C_{Req} subsumes C_{Adv}, in this case, the service does not completely fulfill the request. Thus, another service may be needed to satisfy the rest of the expected data.
- *Fail*: failure occurs when no subsumption relation between the advertised concept and the requested one is identified.

We have introduced a number of modifications to adapt the above algorithm to our composition approach:

- We match atomic processes rather than high level service capabilities.
- The previous algorithm recognizes an *exact* match between two concepts C_{Req} and C_{Adv} if $C_{Req} = C_{Adv}$ or C_{Req} is a direct subclass of C_{Adv}. For the latter case, our algorithm recognizes a *plug in* match.

[12] Subsumption means the fact to incorporate something under a more general category.

- We are not interested in *subsumes* matches as we consider that this degree of match cannot guarantee that the required functionality will be provided by the advertised service. Moreover, as we match operations we do not want to split them between two or more services.
- We do not define priorities between matching inputs and outputs. Thus, we consider that a match between an advertised atomic process and a requested atomic process is recognized when all outputs of the request are matched against all outputs of the advertisement; and all the inputs of the advertisement are matched against all the inputs of the request.

In order to optimize semantic service discovery, we assume that services are classified according to their context-attributes. This classification is done off-line (not during the composition process) by the service discovery protocol each time a new service appears in the network. This classification allows minimizing the number of semantic matches performed during the composition. Specifically, we propose to classify services' descriptions by using the context attributes specified in the service descriptions (§3.1). For example, location-aware services, i.e., services that provide information according to the user's location, will be put together. Thus, when looking for an operation involved in the user task, which has a contextual requirement such as physical distance awareness, we will only perform semantic matching within the above group of services. Once a set of services that provide semantically equivalent operations with those of the user task, further meeting the contextual requirements of the task, is selected, the next step is to compose those services in a way that meets the task's conversation structure. This service composition involves the integration of the selected services' processes, as described in the following section.

4.2 Context-Aware Process Integration

Our context-aware process integration algorithm aims at integrating the processes of services, selected by semantic service discovery, to reconstruct the process of the target user task. To perform such an integration, we employ the finite state automata model that we have defined earlier. Thus, we consider the automaton representing the process of the user task and the automata representing the processes of the selected services. In a first stage, we connect the automata of selected services to form a global automaton. This global automaton contains a new start state, empty transitions that connect this state with the start states of all selected automata, and other empty transitions that connect final states of each selected automata with the new start state. This allows the use of the selected services more than once in the same service composition. The next stage of our process integration algorithm is to parse each state of the task's automaton starting with the start state and following the automaton transitions. Simultaneously, a parsing of the global automaton is carried out in order to find for each state of the task's automaton an equivalent state of the global automaton. An equivalence is detected between a task's automaton state and a global automaton state when for each incoming operation[13] of the former, there is at least one semantically equivalent incoming operation of the latter. We recall that equivalence relationship between operations is

[13] Incoming operations are the set of symbols attached to a state's following transitions.

a semantic equivalence that have already been checked during the semantic discovery. Each state of the task's automaton is parsed just once.

In addition to checking for each state the equivalence between incoming operations, management of contextual information is performed. This management is composed of two parts. First, contextual preconditions and effects of service operations have to be taken into account, and second, global task's contextual requirements have to be checked. The first part implies the verification of the contextual preconditions of the selected operations. Furthermore, each time an operation is selected, its contextual effects have to be taken into account as a new parameter of the user's actual context. Thus, a simulation of the user's context based on the assumed contextual effects of the selected operations added to the current context, have to be done each time an operation is selected. This simulation is performed by the context manager (§3.2). Note that we assume that the user's context does not change during the composition process. This assumption is valid because we consider that the composition duration (in milliseconds [2]), is very short compared to a user activity that affects the context, such as moving to the button switch and turning on the light (may take a few seconds). The second part implies that each time an operation is selected, some context attributes from the above simulated context have to be compared to the task's global requirements.

During the composition process, various paths in the global automaton that represent intermediate composition schemes, are investigated. Some of these paths will be rejected during the composition while some others will be kept. A path can be rejected for one of the following reasons:

1. Starting from the actual state of the path, the task's following symbols cannot be reached in the global automaton;
2. The simulated context, i.e., the contextual effects of the path's operations added to the current context, does not fulfill the contextual preconditions of the incoming operations;
3. Some attributes of the simulated context do not meet the global contextual requirements of the user task.

The remaining paths, represent the possible composition schemes that can be performed to realize the user task, further meeting both the task's contextual requirements, and the contextual precondition and effects of the involved services.

The proposed process integration algorithm gives a set of sub-automata from the global automaton that behave like the task's automaton. The last step is then to select arbitrarily one of these automata as they all behave as the user task. Using the sub-automaton that has been selected, an executable description of the user task that includes references to existing environment's services is generated, and sent to an execution engine that executes this description by invoking the appropriate service operations.

5 Scenario

We present a simple example that illustrates how our context-aware composition algorithm can be used in a networked home environment. This example is inspired from one of the Amigo scenarios.

"...Robert, (Maria's and Jerry's son) is waiting for his best friend to play video games. Robert's friend arrives bringing his new portable DVD player. He proposes to watch a film rather than playing games, and asks Robert if he prefers to watch one of the films brought on his device or if he rather prefers to watch any films from his home databases. In order to use his friend's DVD player, Robert has asked the system to consider this device as a guest device and to authorize it to use the home's services. This player is quite complex as it takes into consideration some user's contextual and access rights information. The former is used to display the video streams according to the user's physical environment and preferences (for example by adapting the lighting and the sound volume or displaying the film on other devices more pleasant for the user, such as a plasma or a home cinema), while the latter is used to check whether the user is authorized to visualize a specific stream (for example some violent films may be unsuitable for children)..."

Robert's friend DVD player contains a Video Application that uses Web ontologies to describe its offered and required capabilities. The conversation that is published by this application is depicted in Figure 5 (left higher corner). This conversation is described as an OWL-S process that contains concrete offered operations (in white) and abstract required operations (in gray) that have to be bound to environment's operations. On the other hand, Robert's home environment contains a number of services among which a Digital Resource Database Service a Context Manager Service and a Location-Aware Display Service; all publish OWL-S conversations as shown in the same figure (on the right higher, left lower and right lower corners respectively). In this case, the context manager is a networked home service, which provides different kinds of context information, among which the user's location (using a network of cameras or radio receivers and a transmitter embedded on the user's device).

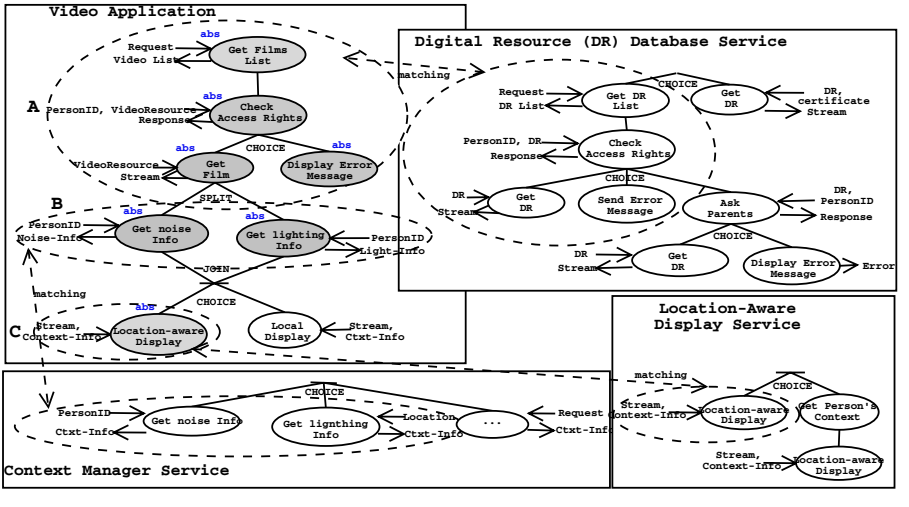

Fig. 5. Example of the context-aware process integration

At execution time, the DVD player will discover the missing abstract conversation fragments included in its description. The semantic discovery step will allow the selection of the three previous services as they contain operations that match the operations of the video application. Furthermore, the location-aware context condition specified in the operation Location-aware Display of the user task will lead to the selection of the display service which provides the location-awareness context attribute. The second step of the composition is the process integration step. In this step our algorithm attempts to reconstruct the abstract conversation of the Video Application by using the conversations of the selected services. The selected fragments after matching are shown in Figure 5. In this step, the fragment A of the task's conversation will be matched against a part of the DR Database service, while the fragments B and C will be matched against parts of the Context Manager Service and the Display Service respectively.

In addition to the composition of services' conversations, the fulfillment of the contextual preconditions of some operations have to be checked. For example, the operation Get Film may have as precondition that there is enough free disk on the user's device. On the other hand the operation Location-aware Display may have as precondition that nobody is sleeping in the current user's location.

6 Conclusion

Compared to existing work in the field of context-aware service composition, our composition approach is more flexible. Indeed, most existing approaches, such as [14, 28, 23] propose simple composition solutions based on planning techniques. Planning is an automated service composition approach, generally performed by a planner, that starts from a set of user's required outputs and provided inputs and performs a combinatorial service chaining based on signatural compatibility, i.e., compatibility between exchanged inputs/outputs. Then, the planner gives a set of plans (service compositions) that meets the user's request. In this area, Vukolvic *et al.* propose a syntactic model to represent context attributes [28]. The composition approach proposed in that work is based on HTN planning using the SHOP2 planner. In this approach, a description of a composition is given in BPEL4WS. Within this description, predefined actions are specified according to static context conditions. Then, this description is translated into SHOP2 goals, and the planner gives a plan meeting these goals. Finally, the returned plan is translated to BPEL4WS and sent to a BPEL execution engine. Mostefaoui *et al.* use an RDF-based model for describing context information [14]. The composition approach proposed in that work considers a complex service request that is decomposed into basic services. These services are discovered in the environment using context information and constraints specified in the service description such as "free memory $>$ 128 KB". While composing the discovered services, additional constraints extracted from the user preferences are taken into account, such as "commission $<$ 500 euros" for a travel reservation composite service. Sheshagiri *et al.* propose an approach for managing context information, in which context information sources are represented as semantic Web services [23]. This enables context information to be automatically discovered and accessed. In that approach context knowledge is represented in OWL, and

services are represented as OWL-S atomic processes. A backward-chaining approach is then used to build a plan corresponding to a service composition. While planning allows an automated service composition, we argue that it presents the following limitations: (i) the resulting service compositions are based on service signatural compatibility; however, this does not guarantee that the resulting composition will really meet the user's intended semantics; and (ii) since it is a combinatorial solution, planning is a costly computational approach, and we argue that it is not well suited for resource-constrained devices.

Coming from the software engineering community, complex composition approaches have been proposed in the literature [4, 13]. Compared to these approaches our composition approach deals with a higher level of flexibility. More precisely, the distinctive feature of our solution is the ability to compose Web services that expose complex behaviors (i.e., workflows) to realize a user task that itself has a complex behavior. Existing approaches generally assume that either the services or the task have a simple behavior (i.e., services are described as a list of provided operations, and/or tasks are described as a single or a set of required operations), thus leading to simple integration solutions. In our case, we assume complex behaviors for both services and tasks described as OWL-S processes, and we propose a matching algorithm that attempts to integrate the services' processes to realize the user task.

In the last few years, context-awareness and dynamic composition of services have been very active fields of research. However, there is a little work combining both efforts. Researchers working on context-aware systems generally focus on modeling context knowledge and defining software architectures for the perception and aggregation of context information from heterogeneous sources in the environments. Thus, most service composition approaches proposed in this area are simple composition approaches, generally based on planning techniques. On the other hand, researchers from the software engineering domain have been very active in the field of dynamic service composition, and very interesting approaches have been proposed in this area. However, most of these approaches poorly deal with context-awareness.

Based on a flexible service composition approach, our solution deals at the same time with context information, combining both user-centric and service-centric views. We are developing a prototype for evaluating our approach to context- and QoS-aware service composition. Currently, the prototype contains only an implementation of our QoS- aware service composition approach. The preliminary results show that the runtime overhead of our algorithm is reasonable, and further, that QoS-awareness improves its performance [2]. Finally, our ongoing research effort aims at introducing context-awareness to our prototype and at evaluating the efficiency and performance of our solution to context-awre service composition.

References

1. Sonia Ben Mokhtar, Nikolaos Georgantas, and Valerie Issarny. Ad hoc composition of user tasks in pervasive computing environments. In *Proceedings of the 4th Workshop on Software Composition (SC 2005). Edinburgh, UK, April 2005. LNCS 3628.*
2. Sonia Ben Mokhtar, Jinshan Liu, Nikolaos Georgantas, and Valerie Issarny. QoS-aware dynamic service composition in ambient intelligence environments. Submitted for publication.

3. Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-owl: Contextualizing ontologies. In *International Semantic Web Conference*, pages 164–179, 2003.
4. A. Brogi, S. Corfini, and R. Popescu. Composition-oriented service discovery. In *Proceedings of the 4th Workshop on Software Composition (SC 2005). Edinburgh. UK. LNCS 3628.*
5. Harry Chen, Filip Perich, Timothy W. Finin, and Anupam Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *MobiQuitous*, pages 258–267. IEEE Computer Society, 2004.
6. A. Dey, D. Salber, and G. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In *Human-ComputerInteraction*, volume 16, pages 97–166, 2001.
7. Andreas Eberhart. Ad-hoc of invocation semantic web services. In *IEEE International Conference on Web Services (San Diego, California ICWS 2004)*, pages 116–124, June 2004.
8. Howard Foster, Sebastian Uchitel, Jeff Magee, and Jeff Kramer. Model-based verification of web service compositions. In *IEEE International Conference on Automated Software Engineering*, 2003.
9. Jadwiga Indulska, Ricky Robinson, Andry Rakotonirainy, and Karen Henricksen. Experiences in using cc/pp in context-aware systems. In Ming-Syan Chen, Panos K. Chrysanthis, Morris Sloman, and Arkady B. Zaslavsky, editors, *Mobile Data Management*, volume 2574 of *Lecture Notes in Computer Science*, pages 247–261. Springer, 2003.
10. M. Koshkina and F. van Breugel. Verification of business processes for web services. Technical report, York University, 2003.
11. Seng Wai Loke. Logic programming for context-aware pervasive computing: Language support, characterizing situations, and integration with the web. In *Web Intelligence*, pages 44–50. IEEE Computer Society, 2004.
12. Zakaria Maamar, Soraya Kouadri, and Hamdi Yahyaoui. A web services composition approach based on software agents and context. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1619–1623, New York, NY, USA, 2004. ACM Press.
13. Shalil Majithia, David W. Walker, and W. A. Gray. A framework for automated service composition in service-oriented architecture. In *1st European Semantic Web Symposium*, 2004.
14. Soraya Kouadri Mostéfaoui, Amine Tafat-Bouzid, and Béat Hirsbrunner. Using context information for service discovery and composition. In *iiWAS*, 2003.
15. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of Web services capabilities. *Lecture Notes in Computer Science*, 2342:333–347, 2002.
16. Joachim Peer. Bringing together Semantic Web and Web services. *The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy, 2002.*, 2342:279, 2002.
17. Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koenraad De Bosschere. Towards an extensible context ontology for ambient intelligence. In Panos Markopoulos, Berry Eggen, Emile H. L. Aarts, and James L. Crowley, editors, *EUSAI*, volume 3295 of *Lecture Notes in Computer Science*, pages 148–159. Springer, 2004.
18. Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koenraad De Bosschere. Towards an extensible context ontology for ambient intelligence. In *EUSAI*, pages 148–159, 2004.
19. Anand Ranganathan, Roy H. Campbell, Arathi Ravi, and Anupama Mahajan. Conchat: A context-aware chat program. *IEEE Pervasive Computing*, 1(3):51–57, 2002.

20. Manuel Roman and Roy H. Campbell. A user-centric, resource-aware, context-sensitive, multi-device application framework for ubiquitous computing environments. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, 2002.
21. B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, 1994.
22. Bill N. Schilit, Norman Adams, Rich Gold, Michael M. Tso, and Roy Want. The parctab mobile computing system. In *Workshop on Workstation Operating Systems*, pages 34–39, 1993.
23. Mithun Sheshagiri, Norman Sadeh, and Fabien Gandon. Using semantic web services for context-aware mobile applications. In *MobiSys 2004 Workshop on Context Awareness*.
24. Kaarthik Sivashanmugam, Kunal Verma, Amit P. Sheth, and John A. Miller. Adding semantics to web services standards. In *Proceedings of the International Conference on Web Services, ICWS '03, 2003, Las Vegas, Nevada, USA*, pages 395–401, June 2003.
25. João Pedro Sousa and David Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture*, pages 29–43. Kluwer, B.V., 2002.
26. Maria Strimpakou, Ioanna Roussaki, Carsten Pils, Michael Angermann, Patrick Robertson, and Miltiades E. Anagnostou. Context modelling and management in ambient-aware pervasive environments. In Thomas Strang and Claudia Linnhoff-Popien, editors, *LoCA*, volume 3479 of *Lecture Notes in Computer Science*, pages 2–15. Springer, 2005.
27. W.M.P. van der Aalst and A.H.M. ter Hofstede. Yawl: Yet another workflow language. *Accepted for publication in Information Systems*, 2004.
28. Maja Vukovic and Peter Robinson. Adaptive, planning-based, web service composition for context awareness. In *International Conference on Pervasive Computing, Vienna, April 2004*.
29. Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *PerCom Workshops*, pages 18–22, 2004.