Context-oriented and Transaction-based Service Provisioning

Soraya Kouadri Mostéfaoui and Muhammad Younas

Department of Computing
Oxford Brookes University
United Kingdom
kouadris@brookes.ac.uk, m.younas@brookes.ac.uk

Abstract. This paper presents our approach for service provisioning in pervasive computing environments. The presented approach is based on the usage of context-aware features and transactions during the discovery and the deployment of composite services. Context ensures that the best service offers are selected to participate in a service composition. Transactions help in improving the reliability and efficiency of the composite services.
Keywords: service, context, service discovery, WSDL, transactions.

1 Background

The issues discussed in this paper are framed by the CAT[1] (Context-Aware Transaction for service deployment) conceptual framework developed at the Department of Computing, Oxford Brookes University. This project aims at developing a generic framework that combines service-oriented, context-aware computing using transactions in order to provide users with more adaptive and tailored services in highly pervasive environments.

In this paper we present the CWSDL[2] language which constitute our proposal for the enhancement of the actual W3C standard for service description, with context information. The processing of the proposed CWSDL as demonstrated in this paper takes place during the service discovery process which carries out the tasks of identifying suitable services and selecting the best offers on the current context. In [10], a multitude of context variables building up a context history were identified, from the user context (role, preferences, location, etc.), over the computing context (network connectivity, server load, etc.), to the temporal context. Thus in order to be specified, the CWSDL language needs to fulfill the following requirements:

– Define which part of the context, and which metrics to be used for services rating;

[1] The presented work is based on previous researches made by the two authors.
[2] CWSDL stands for Context-Based Web Service Description Langauge.

– Define how to compare the values obtained by service metrics and how to select the best services.

We will show how we choose to notate these information while doing this, we will remain completely compatible with the established W3C standard for service description (WSDL[3]). This will enable ordinary clients to take advantage of the service description without having to be aware of the additional information.

We also exploit the use of context in order to manage transactions in service compositions. Transactions help to increase the reliability and efficiency of the composed services, and to maintain consistency of the data sources during the concurrent execution of multiple requests and the failure of component systems or network communication [3].

In the remaining of this section we present basic definitions and highlight the main contributions of our approach.

1.1 Preliminaries

Web and mobile services : a Web service is an accessible application that other applications and humans can discover and invoke. Benatallah et al. suggest the following properties for a Web service [1]: (i) independent as much as possible from specific platforms and computing paradigms; (ii) mainly developed for inter-organizational situations; and (iii) easily composable so that developing complex adapters for the needs of composition is not required [8]. A mobile service is a Web service that can be executed in or triggered from a mobile platform.

Composite services : a composition approach connects Web/mobile services together in order to devise composite services. The connection of Web services implements a business logic, which depends on the application domain and control flow of the business case for which the composite service is being devised. Examples of business cases are various such as travel planning and journal-paper review. It is accepted the efficiency and reliability of a composite service strongly depend on the commitments, performance, and delivery capabilities of each of the component services.

Context : composed of con (with) and text, context refers to the meaning that can be inferred from an adjacent text. Dey defines context as any information that is relevant to the interactions between a user and an environment [6]. This information can be about the circumstances, objects, or conditions by which the user is surrounded. Many researchers have attempted defining context among them Schilit et al. who propose three categories of context [18]: (i) computing category such as network connectivity, communication cost, communication bandwidth, and nearby resources; (ii) user category such as profile, location,

[3] Web Service Definition Language, http://w3c.org/TR/wsdl.

nearby people, and even sometimes social situation; and (iii) physical category such as lighting, noise levels, traffic conditions, and temperature.

From a service-centric point of view context might be classified into internal and external categories. Internal context refers to all contextual information directly related to the service, such as for example, the number of instances of the services and its context function[4]. The external context defines all other information that might affect the provisioning of services, without explicitly being part of the service context, such as the user's location or preferences. Figure 1 illustrates this principle: in (a) a client requests a service and receives its response without being subject to any adaptation, in (b) the service is subject to adaptation using internal service context, and in (c) using external service context. External context, also called explicit context, is in general explicitly provided. The internal context, also called implicit context, is in general implicitly included during the provisioning of services [8].

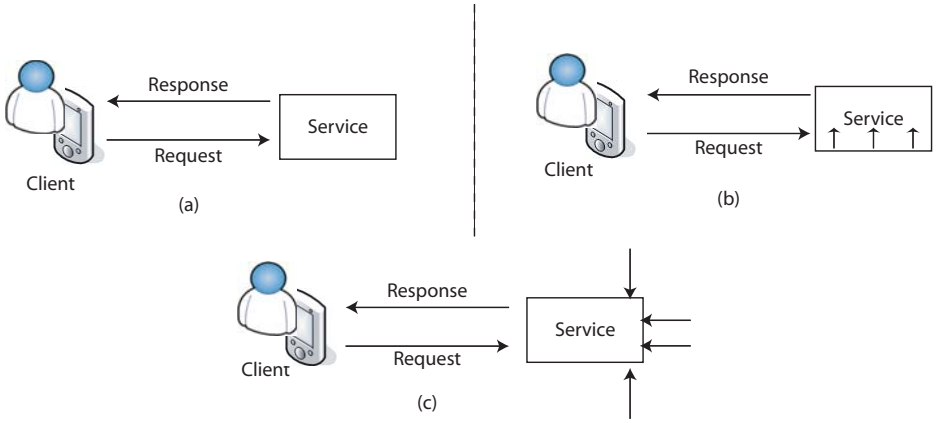

Fig. 1. Internal v.s external contexts

Transactions : a transaction is a unit of interaction that is treated in a coherent and reliable way independent of other transactions that must be either entirely completed or aborted. A transaction can be terminated in two ways: committed or aborted (cancelled). When a transaction is committed, all changes made within it are made durable. When a transaction is aborted, all changes made during the lifetime of the transaction are undone. Traditional transaction systems are typically referred to as ACID transactions. An ACID transaction has the following properties [7]:

[4] Context functions are presented in Section 3.1.

- Atomicity: the transaction completes successfully (commits), or if it fails (aborts), all of its effects are undone;
- Consistency: Transactions produce consistent results and preserve application-specific invariants;
- Isolation: Intermediate states produced while a transaction is executing are not visible to other transactions. Furthermore transactions appear to execute serially, even if they are actually executed concurrently. This is typically achieved by locking resources for the duration of the transaction so that they cannot be acquired in a conflicting manner by another transaction;
- Durability: The effects of a committed transaction are never lost (except by a catastrophic failure).

There exist various limitations of the ACID transactions within Web services, specially the atomicity and the isolation properties are very restricted. In order to deal with these issues various Extended Transaction Models (ETM) have been defined [7]. ETM relaxes the atomicity and isolation properties of ACID transactions and are therefore more appropriate for Web services. Our proposed model uses the ETM model for managing transactions.

1.2 Main Contributions of our Approach

The main contributions of the proposed approach are as follows:

- Ensures that the requirements and constraints on the services to participate in a composition process are taken into account;
- Enhance the service discovery process with context information this allows the discovery and selection of the best services (and/instances) in the current environment of the user;
- To maintain reliability of services and consistency of the data sources during the concurrent execution of multiple requests and the failure of component systems or network communication;
- To improve reliability (i.e., increase the success rate of composite services transactions).

In Section 2 we present the pervasive computing challenges and the motivations of the approach and a simple yet realistic usage scenario, aiming at motivating the role of context and transactions in real life pervasive example. Our model for the enhancement of the current services description standard and how transactions are used to support context-aware service provisioning are presented in Sections 3 and 4. Section 5 is dedicated to the presentations of the experimental evaluations conducted to validate the presented concepts. Section 6 presents some of the existing related work. Finally Section 7 concludes this paper and highlights future directions.

2 Pervasive Computing Challenges and Motivations of the Approach

The general pervasive computing model in a mobile configuration comprises two distinct sets of entities: users with their pervasive devices and fixed hosts. A fixed host can communicate with pervasive devices within its radio coverage area called wireless cell. A mobile client can communicate with a fixed host/server via a fixed host over a wireless channel. From an operation perspective, users expect to be provided with services through pervasive communication networks. Services are to be made available while considering the following aspects: devices, propagation techniques and security. Some of the requirements that the pervasive computing model has to satisfy are summarized below:

- Service availability requirement: it illustrates the need for a user to have an uninterrupted and secure access to services on the pervasive network;
- Network survivability requirement: it illustrates the need to maintain the pervasive communication network "alive" despite the potential failures;
- Information security requirement: it illustrates the importance of providing reliable and unaltered information;
- Additional requirements exist. The increasing reliance and growth in pervasive services impose three requirements (availability, scalability, and cost efficiency) on the offered services.

Despite the multiple opportunities that pervasive computing offers especially to those who are on the move, different obstacles still hinder the expansion of this model. For instance, pervasive devices are still limited by their battery power, and current technologies are meant to be used for situations with permanent and reliable communication infrastructure. In order to optimize service provisioning in pervasive environments, several important issues need to be dealt with [13] [8]:

- Handling disconnections during service execution: in a pervasive scenario, disconnections may be frequent due to the lack of coverage areas or devices changing location. As a result, device disconnection is critical to service execution success;
- Context-sensitive service deployment: in addition to criteria such as monetary cost and time of response, service deployment should consider locations of users and capabilities of computing resources on which services will operate. Locations and capabilities need to be assessed before service deployment.

Traditional usage of pervasive computing devices has revolved around merely using these devices to make and answer phone calls and short text messages. The increasing power and versatility of these devices are starting to enable implementing complex workflow-like scenarios comprising service composition [14]. In the following, a simple yet realistic usage scenario related to trip planning is presented aiming at motivating the role of using context and transactions during service[5] provisioning.

[5] In the rest of this paper the terms service will be used in place of Web/Mobile service.

2.1 Running Scenario

A composite service may aggregate multiple services to allow a user to watch sporting events for the upcoming London Olympic Games 2012. The user may wish to book a flight to its location, reserve overnight accommodation and buy tickets for the sporting events. These services need to be discovered, selected and connected to each other. Furthermore, there may exist various alternatives for each of these services, such as services for different hotels, bed and breakfast, etc. Using context might help in offering the best services and the best service combinations (compositions).

This simple scenario yields insight into the multiple challenges that contextual service discovery and composition in a pervasive environment face, including: how is context related to services? How to manage non-deterministic transactions which may span the boundaries of autonomous services? While answering these questions constitute the actual challenge of the research in adaptive services. In the following we present our contributions, which represent a step towards a full realization of the context-aware services vision using transactions.

3 Enhancing Service Discovery with Context-Aware Features

In this section, we describe the main steps of our context-aware service discovery approach. It is in the shed of the above cited pervasive computing challenges that we have developed our context-aware service discovery based on an enhanced Web Service description language.

3.1 CWSDL- Adding Context to the Web Service Description Language

In our approach, and in order to introduce context in the service descriptions, we have provided a two layer-based description. Each service published in a service registry is defined by its profile description and its proper description. The service profile is viewed as an abstract entity in the provisioning system and refers to a category of services. We have decided to follow a WSDL-like syntax in order to describe our services[6]. This choice was driven by the motivation of facilitating future integrations with other Web Services standards, like SOAP and UDDI.

Service profile description Used by service registries in order to classify and regroup services by categories, a profile description contains all information useful to describe the functional characteristics of its services. A service can have

[6] This explains the abbreviation CWSDL, the prefix C stands for context.

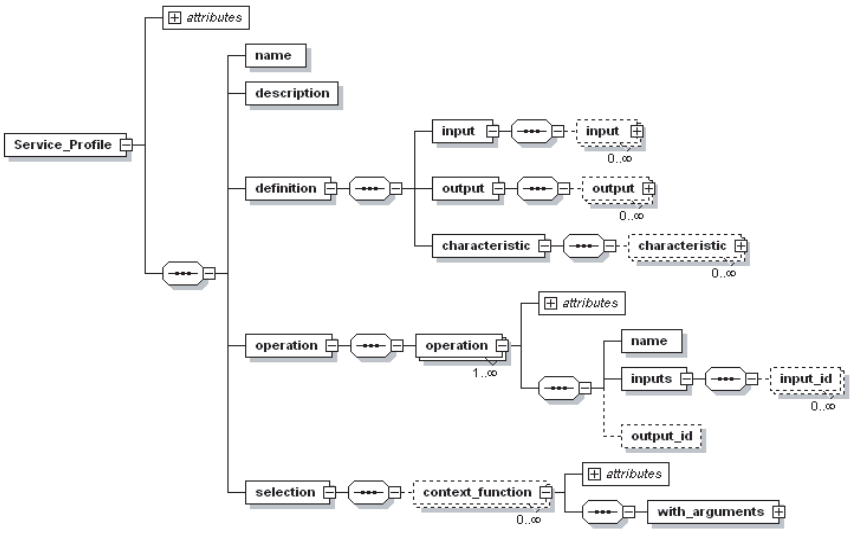

Fig. 2. Service profile description

one or many profile descriptions, for example an SMS service can have the profile of a mobile-display service, as well as wake-up service.

As depicted in Figure 2, the service profile description contains the following elements:

- name: refers to the name of the category of services described by this profile;
- description: gives general information on the service profile;
- definition: gives a description of input and output parameters common to all services that can be classified under this profile;
- operation: describes how the service operates;
- selection: according to the context sensitivity of the corresponding category of services, the selection element defines a set of context functions that are used in order to select the best service offer, along with a set of elector elements that define how to rate the corresponding context functions.

Service description The service description is an instantiation of the profile description. When advertising a service, the provider first defines what the profile of the offered service is, and then attaches its different constraints, and defines the bindings. Different providers might publish the same services with different constraints and different bindings. This is expressed in the service definition and bindings elements (as shown in Figure 3). Constraints might be set on inputs as well as on outputs, and on the number of allowed instances of the service.

An illustrative example is given in Figure 4 in order to illustrate the concepts of service profiles, descriptions and instances, we take two simple types of services, SMS and Wake-up. Before publishing its services a provider should know in advance under which profile to classify it[7].

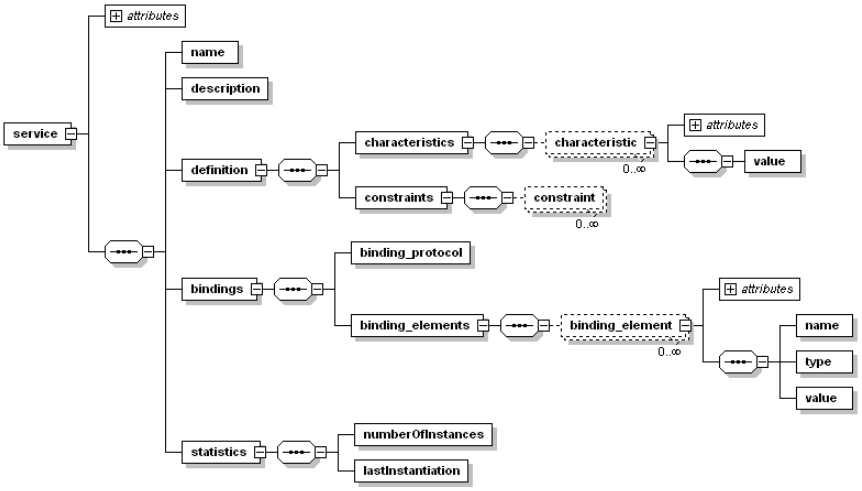

Fig. 3. Adding context to the Web Service description

In this example there are three services provided by three different providers P1, P2 and P3. The SMS service provided by P3 can be classified both under SMS and wake-up service profile. P1 offers two implementations of its SMS service attaching different constraints on each one. The first implementation limits the number of input characters to 150, while the other one limits the maximum number of input characters to only 50 characters.

Context functions The context function represents the sensitivity of the category of services to contextual information. Unlike the other parts of the description, the value of this function is not known in advance, it is calculated at run time when the service is instantiated, in order to help with better service selection. The value of the context function changes over time. A context function can be an aggregation of two or more other context functions, simple examples of such a context function may be:

[7] While automatic classification of services, might be performed using semantic descriptions, this falls out of the scope of this paper. However in the conclusions we present how we envision a simple attempt towards this goal.

– For a weather forecast service, a context function may determine which service has the best response time in the current context;
– For a student cafeteria service inside a campus, a context function based on the available number of seats, determines which cafeteria is least loaded;
– For a restaurant service, a composite context function based on user's location and the number of people in the restaurant, determines which is the nearest least loaded restaurant;
– For a printer service, a composite context function might determine first what is the set of active printers in the time the service is requested, then which printers are least loaded (i.e. better printer service in terms of response time in the current context) [10] [9], and finally the nearest one among those selected;
– etc.

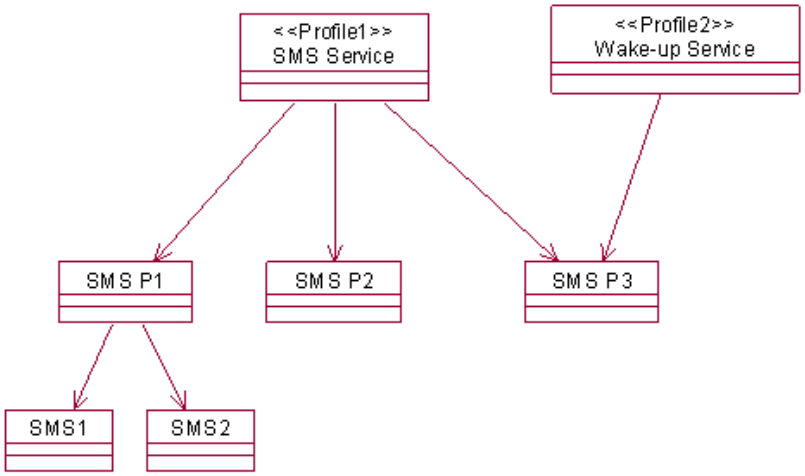

Fig. 4. Illustrative example

An indicator attached to a context function indicates if the function is critical or not. For example a critical context function called "activity" might check the activity of a given device. In case this latter is out of use it is useless to pass it through the other context functions such as for instance, context functions that check the quality of display of the device. As soon as a service fails to satisfy one of the critical context functions, it is removed from the list of candidate services.

For the same service, one or more critical context functions can be associated. From a user centric point of view a critical function can be seen as required and a non-critical function as optional; this feature is used only to augment the quality of the offered services.

Along with each context function, metrics are associated in order to express how to rate the result of the function after evaluations, and how to rank the services. This is expressed inside an elect element integrated with the context function argument. For example, for a certain type of service an elect element might operate by selecting the maximum of the calculated values and for another type of services another rating based on the minimum might be used, depending on the semantics and the operations to be performed by the service.

Also depending on the type of the service and its duties, different weights might be assigned to the context functions. This is expressed by the use of a service Table of Priorities (S-ToP) attached to the context function in the service profile description. For example, for a movie preview service, it might be more important to check first if the service has a good response time and then test its resolution. While for a printer service it might be worth to check the nearest printer, and then perform a check on its speed or resolution.

Furthermore, depending on the users' interests and the type of services, so called User's Tables of Priorities (U-ToP) are also used. This allows expressing the importance users would like to put on the quality of their services. For example a user might put more importance on the quality of the displayed movie and less on the response time.

In our proposed system we have placed a higher priority on users' priorities that service's priorities. This means that if for the same service we have different S-ToP and U-ToP, it is the U-ToP which is taken into account while discovering and selecting services, and the S-ToP is ignored.

Moreover, the same user might have different U-ToP tables according to the domain in which services will be deployed. For example while in an urgent situation in a health care environment, it is more important to place the concern on response time of the services. And once at home for the same service the user might place different weights on the U-ToP putting less concern on the response time and more on the quality or other services' attributes.

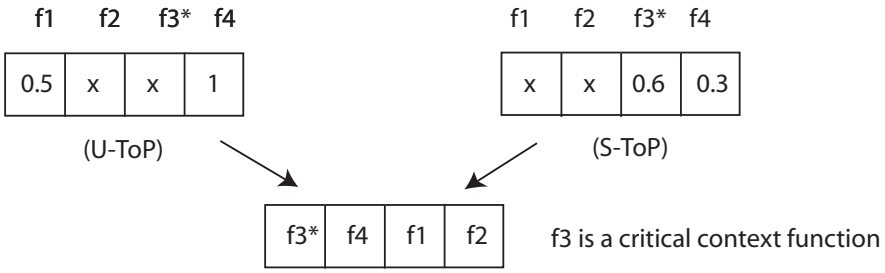

Fig. 5. Example of context functions ranking

Figure 5 shows how to rank the context functions according to their types (critical, non-critical) and the U-ToP and S-ToP tables. In this example a user

and the service provider express different weights for the same service. The user indicates in its U-ToP table that $f4$ is the most important function to evaluate when discovering the service, $f1$ is less important, and finally, she does not express any wishes concerning the remaining context functions. The provider of the service expresses its weights as follows: $f3$ is more important than $f4$, and does not express priorities concerning $f1$ and $f2$.

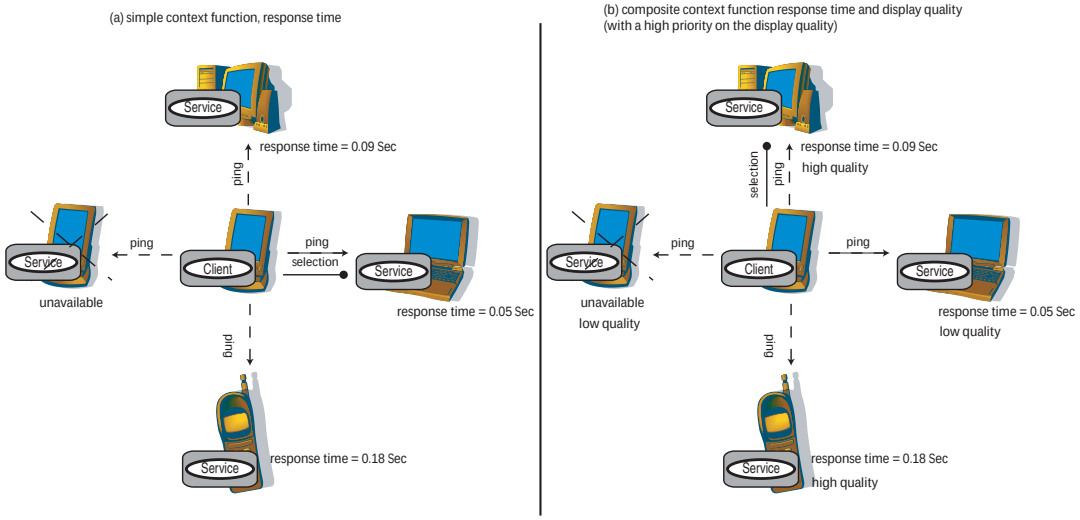

Fig. 6. Illustration of the service selection principle

According to this service type it is indicated that $f3$ is a critical function (annotated with an asterisk "*"). This means that, this function should be evaluated before the others. If none of the candidate services passes this function the discovery process is then aborted (there exist no services that satisfy the user's query in the current context). The order of context functions will be then, $f3$ (it is the critical context function of this types of services), $f4$ and $f1$ as per the user's weights and $f2$ is the remaining function.

In Figure 6 illustrations of the selection of the best services after evaluation of the context functions are shown. In case (a), the requested service is sensitive to only one context function "response time". Thus the best service in terms of response time is chosen (0.05 Sec is the best response time in the actual context). In (b) the requested service is sensitive to two context functions. Namely, the "response time" and the "quality of display". Additionally, the user expresses a higher priority on the "quality of display" function. Thus the best service in terms of quality of display is selected even if it has not the best response time.

In Figure 7 we illustrate roughly, how to map the context function principle, into the standard WSDL syntax. As shown CWSDL follows the same principle

as WSDL. CWSDL adds in the Definitions part the definitions of the context functions and the elect elements respectively. In the Operations part it introduces the operation of the context functions. Finally it binds to the standard WSDL Bindings.

Furthermore Figure 9 shows how the proposed context function integrates to the actual Web Services standards.

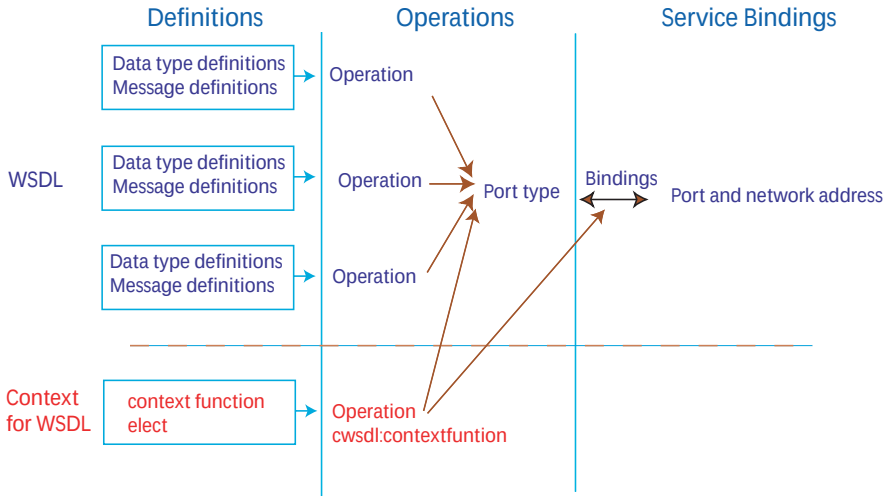

Fig. 7. Integration of the context function into the WSDL standard

4 Deploying Service Composition Using Transactions

In this section we describe the process of transaction-based service composition. We start first by presenting the *STM* (Service Transaction Model) and then present the operation of the discovery and composition of services using CWSDL and *STM* concepts.

4.1 The Service Transaction Model

The proposed model is referred to as the Service Transaction Model (*STM*). In *STM* a transaction is defined as the execution of a composite service which can be divided into well defined units (or component service transactions) that provide correctness and data consistency of services.

An *STM* service transaction is hierarchical and is based on the Extended Transaction Model (to be described later). In STM, a transaction corresponding to the composite service is called *root transaction*. A root transaction can be

decomposed into a set of *sub-transactions* corresponding to the participating (basic) component services of a composite service, (for instance, a composite service for the London Olympic Games can be represented as a root transaction, while its component services such as flight and hotel can be represented as sub-transactions). It is worth noticing here that a sub-transaction can also contain other sub-transactions. In Figure. 8 we show a generalised structure of the root transaction together with its sub-transactions.

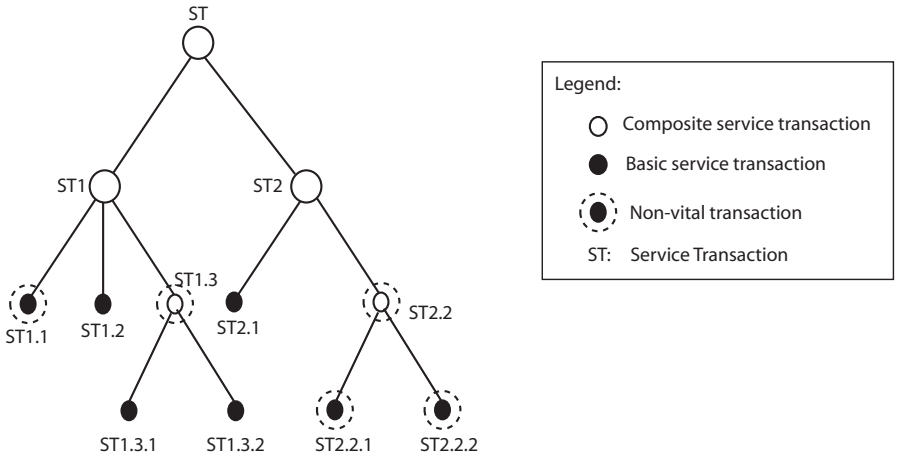

Fig. 8. Tree-based model of service transactions

Each of the sub-transactions of the root transaction can either represent basic service transaction or an other composite service transaction. The basic[8] service transaction corresponds to a transaction which can not be decomposed into other sub-transactions. A basic transaction is represented by a leaf on the transaction tree. A composite service transaction is a transaction that can recursively be decomposed into sub-transactions and correspond to non-leaf transaction on the tree. The root transaction correspond to the combination of its sub-transactions using one or more of the following execution modes:

- Sequential execution of the component sub-transactions;
- Parallel execution of the component sub-transactions;
- Dynamic creation of sub-transactions;
- Definition and usage of a set of alternative sub-transactions according to the composite service and the current context.

In STM a transaction may have different types, such as compensatable, vital and non-vital.

[8] The words basic service transaction and component service transaction will be used interchangeably in the remaining of the paper.

Compensatable and replaceable transactions : a transaction is compensatable if its effects can be semantically undone by executing a compensating service transaction. It is replaceable if there is an associated alternative service transaction. In our approach alternative services correspond to the services found during the discovery process but that have not been selected (i.e services that do not have the best context function).

Vital service transaction *vs.* non-vital service transaction : a vital service transaction corresponds to the compulsory participation of all the component services to the execution process. Because it is expected that the component services will be spread across the network, the reliability of the execution process of each component service affects the whole reliability of the composite service. A non-vital service transaction does not necessarily involve all the component services. Some component services can be skipped during execution due to various reasons such as possibility of substitution or non-availability.

The failure of a vital service transaction determines immediately the failure of the parent transaction. The failure of a non-vital transaction does not have a direct effect on the execution of the parent transaction. If a non-vital transaction has aborted it is replaced by an alternative transaction (the service which has failed is replaced by an alternative service).

In our proposed model a transaction is characterized by Extended Transaction Model properties. These are described as follows:

Semantic atomicity : is a weaker requirement than the classical atomicity. Whereas an atomic transaction is guaranteed to complete successfully or not at all. Semantic atomicity allows the unilateral commit of component transactions irrespective of the commitment of their (parent) services transaction, with the constraint that information sources must remain consistent after the execution of transaction;

Consistency : means that the data should be consistent during the execution of the service transaction. Such consistency can be insured using the services semantics;

Durability : requires that effects of a committed transaction must be made permanent in the respective data sources, even in the case of failures.

Contingency : is the ability to commit in spite of failures or service unavailability. Contingency is achieved by associating alternative service transactions with the basic service transactions of a composite one. As described earlier, composite service can be composed of various component services which are associated with many alternative services. For instance, various alternative services are available for flight and accommodation.

4.2 Operation of the Discovery and Composition

In this section we describe how we perform service discovery and selection taking into account context information using services described in CWSDL.

We have adopted a two-tier service discovery approach, in which the discovery of services might be performed into two distinct domains, respectively local or global. The local domain corresponds to the domain in which the user issues the request and the global domain corresponds to the wide-area domain such as Internet. Service registries might be hosted on dedicated servers or on users' devices. The adoption of a two-tier service discovery aims at reducing: i) the number of interactions and ii) the number of remote data exchanges between domains. This will speed up the response rate for the users and lessen the burden on the services. The local domain services are closest to the user, this means that when a user issues a request, they are more probable to be discovered. When a user changes its location it might switch to other domains.

Service Discovery: when a user requests a service it might be necessary to compose complex services out of the registered basic services. The problem of splitting the request into sub-services is complex and belongs to the domain of planning in Artificial Intelligence, which is outside the scope of our present work. The current design of our system uses UCM's social laws[9] [20] to decompose complex service requests into basic services and to determine the process model of execution.

After identifying the primitive services to be used for the composite service, a discovery process is launched for each of these primitives. For this purpose we have developed a two-phase algorithm. In the first phase the algorithm tries to discover the basic services against the ones registered on the user's vicinity (using the location attribute of the user). It first matches the user's query as well as the user's device capabilities into the nearby service registry and progressively increases its search radius [10] [9] [5] to discover all the different services necessary for the query solving.

Then, it has to figure out other end requirements, i.e., device capabilities anticipated by service implementations. The first step of the algorithm produces a set of candidate services. In the second phase, the algorithm iterates through the resulting set, evaluates the context functions and finally picks up the best offers according to the current context and constraints. In other words, services are incrementally filtered [10] [9] according to the evaluation of the context functions.

The adopted algorithm is shown below. This algorithm is first applied into the user's local domain. In case no services are found into the local domain, a switch is made to the global domain and a global service discovery is performed following the same steps (starting from step:4).

1. Retrieve the user's location;

[9] UCM stands from Ubiquitous Coordination Model.

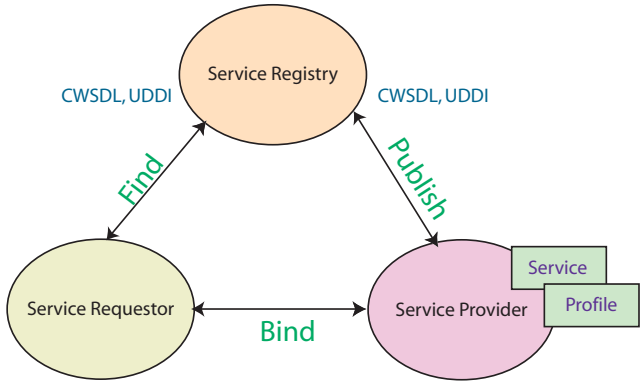

Fig. 9. Integration of CWSDL to the Web Services standards

2. According to the user's location, determine the user's local domain;
3. Find the nearest services registries in the local domain with a search radius;
4. Perform a keyword-based search on the semantic service descriptions within the nearest service registries;
5. If no services are found, increase the search radius and goto step:4;
6. According to the service profile, determine the critical functions;
7. Evaluate the critical functions;
8. Filter the services according to the resulting evaluations;
9. If the user issues a U-ToP table along with its request;
10. Then, according to U-ToP and for each type of the context functions evaluate the function, elect the best service offers and cache the remaining services (alternative services);
11. Else, according to S-ToP and for each type of the context functions evaluate the function and elect the best service offers and cache the remaining services (alternative services);

Deployment of the Composite Services Using Transactions: once the component services are identified and their alternative services cached, the Service Transaction Coordinator (STC) is launched. The STC consists of several Component Transaction Coordinators (CTC). STC is deployed at a fixed node whereas $CTCs$ can be deployed at fixed as well as mobile nodes. $CTCs$ are lightweight such that they can be deployed at the resource scarce pervasive devices. The STC and each CTC maintain log files in order to record the required information about the execution of services transaction.

When the required component services are identified, the application starts invoking a Service Transaction (ST) through the STC. STC coordinates the execution of service transactions. It communicates various messages with $CTCs$ during the execution of the ST. The UML statechart diagram in Figure. 10

describes the working mechanism of the proposed protocol. For the sake of simplicity, STC is shown to communicate with one CTC. Transaction execution is accomplished through the following steps:

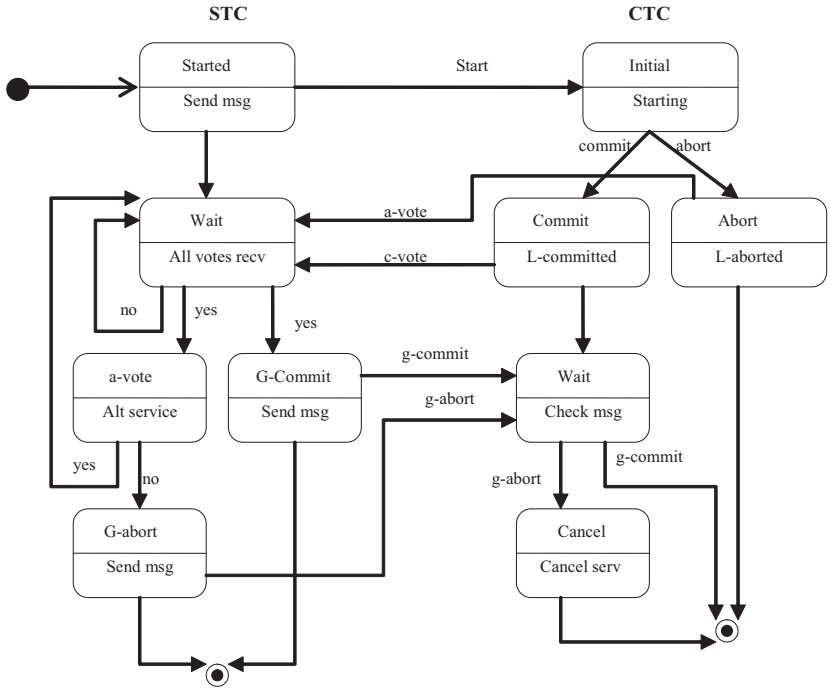

Fig. 10. Statechart of the proposed transactional model

1. A new service transaction, ST, is assigned to a STC, which records the start of ST in a log file, and sends a "*start*" message to $CTCi$ to initiate component service transaction, *csti*. ST then enters into a *wait* state, awaiting messages from $CTCs$ concerning completion of *csti*;
2. $CTCi$ records the beginning of a *csti* in the log file. After processing *csti*, CTC sends a commit or abort vote to STC. If the latter, CTC records the abort of *csti*, sends an abort vote to ST, and declares *csti* to be local-aborted (L-aborted). Otherwise, CTC forcibly writes a commit decision, sends a commit vote to ST and awaits $ST's$ decision;
3. ST receives $CTCs'$ votes. If all votes register a commit decision, ST globally commits (G-commit), by forcibly writing the commit decision and sending

global commit messages to all $CTCc$. It then terminates and starts the processing of a new transaction;
4. If any aborted $csti$ is replaceable (having alternative services cached during the discovery process), STC initiates the alternative component service transaction and awaits the $CTC's$ decision regarding commit or abort of the alternative component service. If $csti$ is not replaceable (no cached alternative services) STC globally aborts, by forcibly writing the abort decision, sending global abort messages to all $CTCs$ and terminating the ST;
5. After receiving a global commit decision (G-commit), CTC simply writes the global commit of $csti$ and changes $csti's$ status from locally-committed to globally-committed. If $csti$ is locally-committed and CTC receives an abort message, then CTC must execute the compensating transaction for $csti$ (cancelling service). This is logged by CTC by simply writing the "*cancel*" decision and then marking the end of $csti$.

5 Proof-of-Concept

This section describes the evaluation of the proposed approach in terms of service transaction reliability, in other words how the use of alternative services (determined by the use of the context functions defined in the CWSDL) increases the commit chances of a composite service transaction. First we devise an evaluation criterion which is based on probability theory. We then present experimental results which are based on this criterion. We define the following types of probabilities for the service transaction, ST, and its component services transactions:

- Actual commit probability (ACP): this refers to the commit probability of ST where no alternative service transactions are involved, i.e., ST commit is based on the commit of actual basic service transactions;
- Actual abort probability (AAP): this refers to the abort probability of ST where no alternative service transactions are involved, i.e., ST aborts based on the abort of actual component service transactions;
- Total commit probability (TCP): this refers to the commit probability of ST where alternative service transactions are involved;
- Total abort probability (TAP): this refers to the abort probability of ST where alternative service transactions are involved;
- Existential probability $P(AST)$: this refers to the probability of the existence of one or more context functions with a service transaction;
- $CP(AST)$ represents the commit probability of alternative service transactions;
- $AP(AST)$ represents the abort probability of alternative service transactions.

Based on the above, we devise a set of expressions to calculate the total commit probability of a ST by taking into account different existential probabilities of service transactions. We consider multiple alternative service transactions.

That is, service transactions can have multiple alternative transactions AST1 and AST2. For example, if the accommodation is not available in the hotel, then it can be booked at bed-and-breakfast. The commit probabilities (CP) and abort probabilities (AP) of alternative service transaction, $AST1$, is calculated as follows (as shown in Figure 11):

$CP(AST1) = CP * AP * P(AST) - - - > (1)$

$AP(AST1) = AP * AP * P(AST) - - - > (2)$

Based on (1) and (2), the commit and abort probabilities of AST2 is calculated as:

$CP(AST2) = AP(AST1) * CP - - - > (3)$

$AP(AST2) = AP(AST1) * AP - - - > (4)$

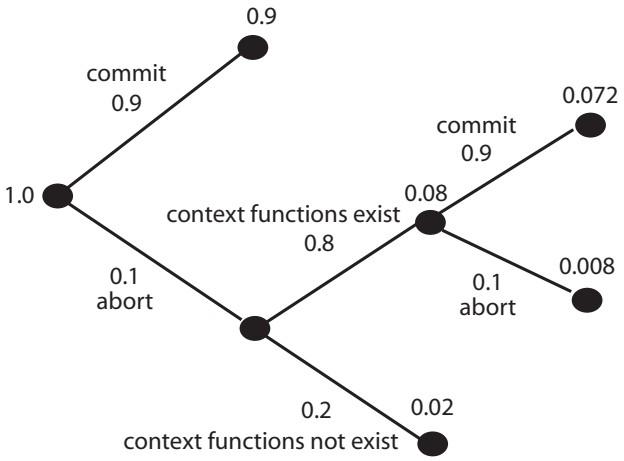

Fig. 11. Calculation of the Total Commit Probability

After calculating the commit and abort probabilities of $AST1$ and $AST2$, we are in a position to calculate the total abort probability, TAP, and total commit probability, TCP, of MST. The total commit and abort probabilities of MST with respect to $AST1$ are calculated as follows:

$TAP1 = 1 - CP + CP(AST1) - - - > (5)$

$TCP1 = 1 - TAP1 - - - > (6)$

The total commit and abort probabilities of the composite service transactions with respect to AST2 are calculated as follows:

$TAP2 = 1 - CP + CP(AST1) + CP(AST2) - - - > (7)$

$TCP2 = 1 - TAP2 - - - > (8)$

5.1 Experimental Results

Using the above expressions $(1) - (8)$, we conduct various experiments to evaluate the reliability of the proposed approach, using existential probability of

0.4 - showing that there exist 40 percent of services with context functions (i.e, potential alternative services transactions) in ST. In each case, different values of the actual commit/abort probabilities are also used in conjunction with an existential probability. Various situations are considered. For example, the worst case scenario where the abort probability of ST is very high, and also the best-case scenario where the commit chances of ST are high. To model the best-case scenario, the actual commit probability is kept high showing that abort chances of ST are low. The commit probability is then changed to different levels. Table 1 shows the total commit/abort probabilities of ST. The total commit probabilities calculated in Table 1 are graphically represented in Figure 12[10]. The graph clearly indicates that in the proposed approach if there exist multiple alternative service transactions (such as $AST2$) then the commit probability of ST is further increased. However the selection of alternative services is not trivial. In our proposal context functions are used to greatly facilitate the selection of alternative service transactions. To the best of our knowledge non of the existing research works have used context and transactions in order to support the provisioning of services.

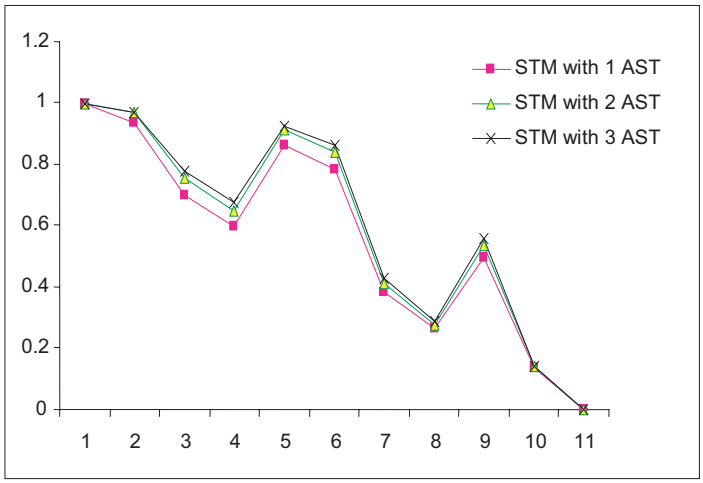

Fig. 12. Evaluation of the Proposed Approach

[10] In the graph we use three alternative services. The X axis represents the total commit rate and the Y axis represents the total abort rate.

Table 1. Commit and Abort Probabilities (Equations 1-8)

ACP	*AAP*	*P(AST)*	***CP(AST1)***	*AP(AST1)*	*CP(AST2)*	*AP(AST2)*	*TAP1*	*TAP2*	*TCP1*	*TCP2*
1.0	0.0	0.4	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000
0.9	0.1	0.4	0.036	0.004	0.032	0.003	0.064	0.031	0.936	0.968
0.6	0.4	0.4	0.096	0.064	0.057	0.038	0.304	0.246	0.696	0.753
0.5	0.5	0.4	0.100	0.100	0.050	0.050	0.400	0.350	0.600	0.650
0.8	0.2	0.4	0.064	0.016	0.051	0.012	0.136	0.084	0.864	0.915
0.7	0.3	0.4	0.084	0.036	0.058	0.025	0.216	0.157	0.784	0.842
0.3	0.7	0.4	0.084	0.196	0.025	0.058	0.616	0.590	0.384	0.409
0.2	0.8	0.4	0.064	0.256	0.012	0.051	0.736	0.723	0.264	0.276
0.4	0.6	0.4	0.096	0.144	0.038	0.057	0.504	0.465	0.496	0.534
0.1	0.9	0.4	0.036	0.324	0.003	0.032	0.864	0.860	0.136	0.139
0.0	1.0	0.4	0.000	0.400	0.000	0.000	1.000	1.000	0.000	0.000

6 Related Work

There are several research initiatives in the field of services composition [17]. However, to our knowledge, none of these initiatives have attempted including context in the composition process of services, or to use transactions in order to improve the reliability of the composed services. Besides the traditional selection criteria that are used in a similar process (e.g., execution cost and execution time), we have shown that the use of context functions has a major effect on the services in general and their composition in particular. For example launching a new discovery process when a service fails instead of using an alternative service definitely delays the development of a composite service. In the rest of this section, we highlight some of the works that have supported our thoughts and inspired us shape our context-oriented and transactions-based approach for services composition.

The first attempts towards a composition language were the IBM Web Services Flow Language (WSFL) [11] and the BEA Systems Web service Choreography Interface (WSCI). Later on comes the Business Process Execution Language (BPEL) [22] as an attempt to combine these languages using the Microsofts XLANG [21]. In the following, we briefly present the most significant proposals of service composition models. eFlow is one the systems that adopts workflows [4]. In eFlow composite services are represented as process schemas consisting of basic services and are modeled by an execution graph. In [19], a UML-based composition approach was suggested. It aims at forming UML models to be automatically converted into specifications of compositions. In [23], a Web component-based approach towards service composition, which includes composition planning, specification, implementation, and execution, was proposed. The use of Web components was backed by the concepts of reuse, specialization, and extension. Indeed, a Web component packages together elementary or complex services and presents their interfaces and operations in a consistent and uniform manner in the form of class definitions.

Work on transaction management in web services generally follows ACID and extended transaction models. M. Papazoglou [16] reports on web services transactions and proposes a Business Transaction Framework (BTF) for web services. This work outlines the requirements and characteristics of business transactions. It also analyzes other transaction related initiatives such as the Business Process Execution Language (BPEL) for Web services, Web Services Transactions (WS-Transactions), Web Services Coordination (WS-Coordination), etc. WS-Transactions [3] and WS-Coordination [2] have been developed by software vendors such as IBM, Microsoft, and BEA Systems. These approaches aim to define frameworks for providing transactional coordination for clients of services offered by multiple autonomous businesses that are based on Web services technologies. They also provide support for long-running business activities in addition to the short-lived (atomic) transactions. M. Little et al. [12] give a comparative analysis of web services transaction protocols [24]. The analysis pertains to the OASIS Business Transaction Protocol (BTP) and WS-Transactions. It identifies the similarities, differences, strengths and weaknesses of these protocols. However, this analysis does not cover the performance aspects of web services protocols, nor does it considers composite web services transactions. T. Mikalsen et al. [15] describes a framework called WSTx (Web Services Transactions) for web services, and introduces the concept of transactional attitudes. This approach requires web service clients to declare their transactional requirements and web service providers to declare their individual transactional capabilities and semantics.

7 Conclusion and Future Directions

The use of context in the provisioning of services ensures that the requirements of and constraints on the services to participate in a composition process are taken into account. While current services composition approaches rely on different selection criteria such as execution cost, execution time and reliability [25], it is deemed appropriate to include context of services in the discovery and composition, particularly when a reactive composition of services is adopted.

In this paper we have first presented our approach for the enhancement of the current Web Services standard for service description, with context-aware features. CWSDL is generic in the sense that it is a quasi standard for the actual model of Web Services and complementary to the existing standards. The use of CWSDL allows to use dynamic and run-time related information to be included in the selection of the best offers of services. Moreover, the use of context is suitable for tracing the execution of services during exception handling. With our approach it is possible to know at any time what happened and what is happening with a service and all its respective instances.

In the second part of the paper we have shown how to deploy the service compositions using a transactional model. Our transactional model is based on the ETM which relaxes the atomicity and isolation properties of the classical ACID transactions. The use of transactions to deploy composite services helps

to improve the reliability and efficiency of the composite services. Indeed, as shown in the evaluations presented in Section 5 the use of transactions together with context increases the success rate of the composite services.

Our future work covers three research thrusts. First the integration of a standardized ontology for CWSDL. Indeed, in the presented CWSDL model, context functions are associated to a class of services, and system knows in advance how to classify the services. However due to the the wide variety of available services in the future pervasive computing environments, it is of top priority that all the system components agree on a common understanding about the used context functions. This will ensure that the context functions will operate on semantically conformant values. Adaptability to new context functions needs also to be explored. The model should be able to take into account unknown classes of services, and thus unknown context functions. It should discover the new classes of services and new functions, and finally incorporate them into the model in a completely transparent way. This may be possible by the the integration of an ontology into the model. Indeed, as a next step in the development of CWSDL, we are investigating how context functions might be expressed by means of an OWL ontology, and to which extent this might enhance the adaptability of the deployed composite services. It is also important to deal with false rating in CWSDL. Then trust mechanisms should be added to the model in order to insure that the results of the elect functions are not erroneous

Second the adaptation of the context function's priorities. In the actual CWSDL design, we have proposed two types of weight tables, in order to express the weights users and services' providers put on the context functions. Namely, U-ToP (User Table of Priorities) and S-ToP (Service Table of Priorities). As a matter of fact, our discovery method puts more importance on the users priorities, and thus follows the priorities expressed in the U-ToP tables, before processing S-ToP priorities. However, we foresee situations in which it may be more interesting to take into account the service providers' priorities. In order to reflect this in the CWSDL design, introduction of learning algorithms, or other adaptation techniques might enhance the adaptability of services. Indeed, depending on the contextual situation the model would be able to choose between taking into account users' or provider's priorities, or combining both.

Finally, the application of the proposed model into real life pervasive scenarios is under development.

References

1. B. Benatallah, Q.Z. Sheng, and M. Dumas. The Self-Serv Environment for Web Services Composition. *IEEE Internet Computing*, 7(1), January-February 2003.
2. F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, and S. Thatte. Web Services Coordination (WS-coordination), 2002.
3. F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, and S. Thatte. Web Services Transaction (ws-transaction), http://www-106.ibm.com/developerworks/library/ws-transpec/, 2002.

4. F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and Dynamic Service Composition in eFlow. In *Proceedings of the Twelfth International Conference on Advanced Information Systems Engineering (CAiSE'2000)*, Stockholm, Sweden, June 2000.
5. D. Chakraborty, F. Perich, A. Joshi, T.W. Finin, and Y. Yesha. A Reactive Service Composition Architecture for Pervasive Computing Environments. In *Proceedings of the IFIP TC6/WG6.8 Working Conference on Personal Wireless Communications (PWC'2002)*, Singapore, October 2002.
6. A. K. Dey, G. D. Abowd, and D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction, Special Issue on Context-Aware Computing*, 16(4), 2001.
7. A.K Elmagarmid. Database Transaction Models for Advanced Applications. *Morgan Kaufman*, 1992.
8. S. Kouadri. Supporting Context-Aware Services in Pervasive Environments. Ph.D Thesis Number 1495, Department of Informatics, University of Fribourg, Fribourg, Switzerland, November 2005.
9. S. Kouadri and B. Hirsbrunner. A Context-Based Services Discovery and Composition Framework for Wireless Environments. In *Proceedings of the 2003 IASTED International Conference on Wireless and Optical Networks (WOC'2003)*, Banff, Alberta, Canada, July 2003.
10. S. Kouadri and B. Hirsbrunner. Towards a Context-Based Service Composition Framework. In *Proceedings of the International Conference on Web Services (ICWS'2003), CSREA Press*, Las Vegas, Nevada, USA, June 2003.
11. F. Leymann. Web Services Flow Language (WSFL 1.0). Technical report, IBM Software Group, May 2001.
12. Little. M and Freund. A. A Comparison of Web Services Transaction Protocols: A Comparative Analysis of WS-C/WS-Tx and OASIS BTP, http://www-128.ibm.com/developerworks/webservices/library/ws-comproto, 2002.
13. Z. Maamar, Q. Z. Sheng, and B. Benatallah. On Composite Web Services Provisioning in an Environment of Fixed and Mobile Computing Resources. *Information Technology and Management Journal, Special Issue on Workflow and E-Business, Kluwer Academic Publishers*, 5(3-4), July-October 2004.
14. Q. H. Mahmoud and Z. Maamar. Challenges and Possible Solutions in Wireless Application Design. *Cutter IT Journal*, June 2005.
15. T. Mikalsen, S. Tai, and I. Rouvellou. Transactional Attitudes: Reliable Composition of Autonomous Web Services. In *Proceedings of the Workshop on Dependable Middleware-Based Systems (WDMS'2002) at the Dependable Systems and Network Conference (DSN'2002)*, Bethesda, MD, USA, 2002.
16. M. Papazoglou. Web Services and Business Transactions. *World Wide Web*, 6(1), 2003.
17. M. Papazoglou and D. Georgakopoulos. Introduction to the Special Issue on Service-Oriented Computing . *Communications of the ACM*, 46(10), 2003.
18. B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'2004)*, Santa Cruz, California, USA, December 1994.
19. D. Skogan, R. Gronom, and I. Solheim. Web Service Composition in UML. In *Proceedings of the IEEE International Conference on Enterprise Distributed Object Computing, (EDOC'2004)*, Monterey, California, USA, September 2004.
20. A. Tafat-Bouzid, M. Courant, and B. Hirsbrunner. Implicit Environment-based Coordination in Pervasive Computing. In *Proceedings of the ACM Symposium on Applied Computing, (SAC'2005)*, Santa Fe, New Mexico, USA, March 2005.

21. S. Thatte. XLANG: Web Services for Business Process Design, http://www.xml.com/pub/r/1153, June 2001.
22. P. Wohed, W. Van-der Aalst, M. Dumas, and A. Ter-Hofstede. *Analysis of Web Services Composition Languages: The Case of BPEL4WS, Web Application Modeling and Development*. LNCS 2813, Springer-Verlag Berlin Heidelberg, 2003.
23. J. Yang and M.P. Papazoglou. Service Components for Managing the Life-Cycle of Service Compositions. *Information Systems, Elsevier*, 29(2), April 2004.
24. M. Younas and K.M. Chao. A Tentative Commit Protocol for Composite Web Services. *International Journal of Computer and System Sciences, Elsevier Science*, 26(5), May Forthcoming.
25. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality Driven Web Services Composition. In *Proceedings of the Twelfth International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, May 2003.