How Can Transactional Semantics Enhance the Commit Rate of Context-aware Service Composition in Advanced Pervasive Systems?

https://doi.org/10.3991/ijes.v9i4.25919

Widad Ettazi(✉)[1], Hatim Hafiddi[1,2], Mahmoud Nassar[1]
[1] ENSIAS, Mohammed V University of Rabat, Rabat, Morocco
[2] STRS Lab, INPT Rabat, Morocco
widad.ettazi@um5s.net.ma

Abstract—Context-aware composition of services exhibiting transactional properties poses several challenges. A major challenge is the transactional behavior of candidate services which is subject to perpetual change while the composition is running. Compositions of services displaying transactional properties must be dynamically adapted at run time to cope with context fluctuations. By dynamic adaptation, we refer to the ability to alter the composition behavior in response to changes affecting its execution. We focus on changes impacting the successful commit rate of transactional service composition. This has led us to explore the trail of a flexible homeomorphism between alternative behaviors. We propose a behavioral adaptation approach that adjusts the behavior of transactional compositions of services in a proactive and transparent manner. This strategy is based on the Profiled Task Class concept. A service composition generator has also been developed for the performance evaluation of components implementing the behavioral adaptation strategy in order to identify its impact on the commit rate of CATS compositions.

Keywords—context-awareness, transactional service, behavioral adaptation, profiled task class, adaptation mechanism, service composition

1 Introduction

Pervasive computing is characterized by three key characteristics that affect the context in which services and users evolve. First, it focuses on dynamic environments where available resources change continuously without prior knowledge of their availability, while in static environments the services provided to users are determined in advance. Second, it operates through ad hoc environments formed by mobile terminals connected via wireless networks and thus loses its interest in static environments where the infrastructures are fixed. Third, it is designed for the use of resource-limited devices. This shift from multi-resource static environments to limited-resource dynamic environments poses several challenges regarding the execution context of service and user, and hence impacts the execution cost and the successful commit rate of transac-

tional services. Furthermore, user requirements can change at any time, thus, altering the transactional properties of the services involved in the composition. The notion of context-awareness in the management of transactional services is not yet addressed. Let's consider, for example, a simple transactional service that books a room in a hotel. Current approaches will simply commit the operation if the required room is available, whether it is compensable or not. They do not take into consideration contextual information such as "a room should be reserved at a hotel that is located nearby the user's center of interests" and the user's transactional requirements such as "hotel reservation must be compensable".

Context-aware computing appeared since the 90s driven by the work of [1]. This term refers to systems capable of perceiving a set of conditions of use in order to adjust their behavior in terms of providing information and services. According to [2], the definitions ascribed to a context-aware system do not include all types of context-aware systems. Indeed, under these definitions, a system that simply collects the context in order to provide it to an application is not considered a context-aware system. Thus, the authors believe that "a system is context-aware if it uses context to provide relevant information and services to the user, where relevance depends on the task requested by the user". In context-aware environments, transactions must be able to adjust to systems that are not necessarily in a perfect environment, for example, that don't require a lock of their resources and do not care if transactions run for short periods of time or longer periods. These systems will operate in a flexible, dynamic environment, but less reliable and that presents contextual requirements (i.e., requirements and preferences expressed or implied by the user, connectivity, bandwidth, etc.) that hinder the transactions execution [3], [4], and [5].

Transactional properties are considered extra-functional properties of a service. In the field of Web Services [40, 41], these properties are generally described by the use of the WS-Policy language [6]. Other approaches such as [7] and [8] have proposed to extend the WSDL description to describe the transactional behavior of web services. Thus, in order to determine the transactional behavior of an elementary service, it is sufficient to analyze the operations exposed by this service. On the other hand, to determine the transactional behavior of composite services, preliminary studies have been reported in [9]. In summary, a composite service which includes transactional aspects must integrate certain characteristics that are specific to the fields of transactional processing and web services namely:

- The transactional properties of a composite service can change dynamically, in other words, the ACID model should not be intuitively adopted.
- The composition of the transactional service must take into account the nature of the participating services which are weakly coupled and have heterogeneous transactional properties.
- Users must have the ability to express and alter transactional requirements such as vitality, compensation and task substitution.

A study of the various works carried out in this field shows that most of the proposed solutions:

- Do not exploit the re-execution property of participating services which improves the commit rate of compositions.
- Do not offer adaptation techniques to the dynamic change in users' transactional requirements.
- Do not support dynamic replacement of participating services.

CATS (Context-aware Transactional Service) composition in pervasive environments typically involves dynamic execution contexts, service unavailability, and varying user requirements. Therefore, composition techniques of this type of services should be designed to be proactive. Indeed, context-aware computing envisages satisfying user tasks on the fly, thus the time available for the selection and composition of services is limited compared to the complexity of processing requests. In addition, the context-aware composition of services exhibiting transactional properties poses several challenges. A major challenge is the transactional behavior of candidate services which is subject to perpetual change while the composition is running. Service composition strategies must also take into account the transactional properties of services (i.e. ACID model) by adjusting them in relation to the execution context.

Service compositions need to be adapted at run time based on context fluctuations. From this perspective, a first approach consists in replacing the failing services. However, selecting services while composition is running can delay and even interrupt the process. To cope with this problem, the compositions of alternative services must be selected simultaneously when constructing the initial composition that fits the user's task. Existing service selection algorithms focus on selecting a single service composition. In this paper, we aim to select several compositions of alternative services, which allow dynamic binding of services at runtime.

In this paper, we proposed a novel approach for context-driven transactional service composition based on transactional semantics of service description and context information. The proposed approach is based on behavioral adaptation strategy which is focused on (i) changes having an impact on the commit rate of the transactional composition of services, in particular, (ii) changes in the transactional needs of users, (iii) changes in the services context due to the dynamicity of pervasive environments (e.g., mobility of users, availability of services, etc.) and (iv) alteration of the transactional properties of the services participating in the composition. We used different adaptation mechanisms to leverage the behavioral adaptation strategy by introducing the profiled task class concept.

This article is organized as follows. The section II will be devoted to review some basic concepts and related work. In section III, we present the transactional features of composite services and introduce the most prominent adaptation mechanisms. Section IV details the proposed behavioral adaptation strategy and exhibits CATS specification and the profiled task class concept. The section V presents encouraging experimental results demonstrating our proposition. Finally, we conclude in the section VI.

2 Literature review

The works presented in this section represents existing approaches for reliable execution of context-aware composite services, particularly composite services exhibiting transactional properties. Due to the absence of an accepted test bench for the adaptation of composite services, and the complexity and specificities of the implementation of each approach, it is difficult to perform a reliable comparative analysis quantitatively. We largely limit the comparison to a qualitative analysis of the different approaches.

Authors in [10] proposed a transactional model called FENECIA which includes forward recovery by re-executing and replacing the service, backward recovery by compensating and the concept of vital and non-vital services. If a vital service fails and it cannot be re-executed or has no alternate services, the execution of the composite service is interrupted. In contrast, running a composite service can be successful even if non-vital services have failed. Another work in [11] proposed another framework called FACTS to ensure the adaptability of composite services. It is a hybrid approach that combines exception handling and transactional properties. When a failure occurs at run time, FACTS first uses exception handling strategies to try to fix it. If the failure cannot be defined, it brings back the composite service to a consistent state using compensation mechanisms. The authors in [12] developed a new model for context-aware transactions in the context of mobile services. This model provides a relaxed set of transaction correctness criteria called SACReD (Semantic Atomicity, Consistency, Resiliency, Durability) and a supporting protocol. Unlike ACID criteria, SACReD does not impose an isolation policy, thus allowing transactions to be partially committed. The resiliency property allows alternative services to be executed when a service fails or does not meet the required context. In [13], the authors proposed the TQoS approach for the selection and composition of services according to their transactional requirements, QoS characteristics and user preferences. After each service selection, the current transactional property of the resulting composite service is calculated and a service substitution, in the case of failure, can be performed. The authors in [14] introduced a framework for reliable replacement in the context of transactional services composition driven by QoS parameters. The framework takes into consideration the QoS parameters of the reselection service, transactional risk and the cost of compensation during the replacement process. The utility function is calculated for each candidate service and the path of the optimal service to be replaced is selected again. Reference [15] presented a context-driven approach to adapting transactional web services. The authors have defined transactional properties for web services that allow their composition and execution via policies. This approach supports the management of exceptions through the use of adaptation strategies. In [16], the authors proposed to integrate business transaction recovery workflows and the concept of context-awareness. When a business transaction exception is detected, the assessment of the constraint condition is performed. The exception is ignorable if the constraint is satisfied and the successive service will be invoked. The recovery supported strategies are: re-execution, substitution, user interaction and compensation. The strategy described by [17] takes into account the user's directives in order to propose several recovery plans. Users manually

choose the desired recovery plan among the plans automatically calculated and classified by the system. It supports the following user directives: (i) developers define a set of behavioral correctness properties that must be maintained at runtime, as well as compensation costs; (ii) users provide criteria for choosing among possible recovery plans, depending on the plan duration, the compensation cost, etc. (iii) users manually choose the desired recovery plan among the plans automatically calculated, classified and proposed by the system. Reference [18] presented a method to find the optimal solution for composing transactional web services using a dependency graph and 0-1 linear programming. The approach proposed in [19] allows selecting several candidate services that can be executed sequentially in order to improve the composite service QoS. To do this, the composition of services is modeled as a search problem of the shortest path satisfying certain constraints in a directed acyclic graph (DAG). Other works have proposed several heuristic algorithms [20], [21], and [22] to reduce the complexity of search time in local or global search related to web services composition. In [23], the authors proposed a new approach that combines the use of genetic algorithms and Q-learning to find the optimal composition. In [24], the composition of services taking into account QoS and transactional properties is modeled as a path construction problem in a directed acyclic graph. Transactional properties ensure a reliable execution of services. Graph vertexes represent the candidate services and arcs describe the links between the candidates of two classes of adjacent services. A recent approach in [25] has been proposed to solve the problem of web services composition taking into account conflicts and dependencies between services. Reference [26] proposed DPSA (Distributed Partial Selection Algorithm) which is a service composition algorithm using a partial selection approach. This algorithm first performs a validation of the local QoS constraints where candidate services violating QoS constraints are eliminated at the level of each abstract service in the composition. The authors in [27] introduced EQSA (Energy-centered and QoS-aware services selection) which is a service selection algorithm focused on energy and QoS-aware in the context of large-scale service composition.

Despite the multitude of works that exist in the literature, the adaptability of transactional aspects to context variations remains an open issue. Most adaptation approaches use a particular adaptation mechanism instead of implementing several adaptation strategies (i.e. Dynamic binding, Retry/Redo, Substitution, Dynamic reconfiguration, Behavioral adaptation, Dynamic adaptation of transactional requirements). In addition, the studied approaches do not fully exploit the transactional properties, particularly the Atomicity property (i.e., strict, semi-atomicity, semantic, relaxed) and the behavior of the underlying transactional services (i.e., Vital, Replayable, Replaceable, Compensable) in order to offer proactive and efficient adaptation mechanisms (e.g., adaptation graph, AI planning, branch-and-bound).

3 CATS composition

A composite service can be considered as a structured transaction where services are sub-transactions and the interactions are transactional dependencies [28]. Thus, running

a composite service requires relying on transaction models that are generally distributed, complex, and long-running. In addition, the composite service must support the transactional properties of the participating services. Generally, traditional approaches that guarantee ACID properties are designed for tightly coupled environments and short-lived transactions; which is still not the case in the field of web services. Thus, in this context, the examination of ACID properties is necessary.

3.1 Transactional features of a composite service

Since the services participating in the composition are weakly coupled, each participating service controls the visibility level of the exposed resources:

- Compensable services immediately commit the requested task, their effects will be visible to the system and the associated resources will be released, even before the global commit of the composition. In case of composition abort, compensation operations will be initiated in order to semantically cancel the effects of the executed task. In this case, atomicity and semantic isolation are ensured
- Non-compensable services lock the resources requested by the composite service, until the latter decides either to commit and make visible the effects, or to cancel already made reservations. When the composite service decides to commit, the participating services immediately commit and make visible the effects of the executed tasks. Thus, non-compensable services preserve strict atomicity as well as strict isolation.

Consistency is maintained by respecting the integrity constraints that depend on the composite service. Composition can be successful in several ways, and maintaining consistency is a characteristic that must be defined by the designer of the composite service. Durability in transactional systems refers to the persistence and traceability of the process. Each participating service is responsible for preserving the traceability of its internal process and the composite service must preserve the traceability of the overall process. In the remainder of this article, we are specifically interested in the property of Atomicity. This property plays a very important role in the commit process of transactional services and can significantly impact composition execution, and therefore the performance of context-aware systems.

From what has been presented, a composite service exhibiting transactional properties must:

- Offer to users the opportunity to express their transactional requirements.
- Respect the autonomy of the participating services; it must not place any constraints on the services participating in the composition.
- Adapt to execution context variations by ensuring selection and dynamic replacement of services.
- Adapt to the dynamic change in user requirements.
- Take into consideration that some participating services may be composite as well.

These characteristics show that the transactional properties of composite services are variable, and can change dynamically during their execution, hence the need for

flexibility and adaptability in designing solutions that ensure correct execution of CATS compositions.

3.2 Adaptation mechanisms

Adaptation mechanisms refer to the techniques used to adapt context-aware systems. They represent a large subject of research. In this paper, we focus on the following adaptation mechanisms: dynamic binding, retry/redo, substitution, dynamic reconfiguration, behavioral adaptation, and dynamic adaptation of transactional requirements.

In dynamic binding, the user task is defined as a set of abstract activities, and several services are selected as potential candidates to satisfy each activity [29]. In the execution phase, the selected services are linked to abstract activities according to changes in the context. Service selection is particularly interesting in this case, since it is possible to provide several alternative services for each abstract activity described in the user task. The retry technique is used to invoke the service that failed multiple times, while redo is utilized to rerun the same service using different input parameters. The substitution mechanism consists of replacing the failed service with another service that has equivalent functionality and that meets the execution context requirements. The main difference between dynamic binding and substitution is that the latter allows services to be replaced with alternative services once a service failure or a context change occurs. Dynamic reconfiguration represents a large class of adaptation mechanisms. It includes resources reconfiguration, parameters reconfiguration and service composition reconfiguration [30]. The behavioral adaptation mechanism consists of performing the composition using an alternative behavior. Alternative behaviors are generally obtained by modifying the composition patterns that structure the service compositions and/or by modifying the service granularity (i.e. by merging fine-grained services with coarse-grained services or the opposite). Dynamic adaptation of transactional requirements refers to the ability of a system to change the behavioral profile of its transactions. In fact, during execution, if a transactional service cannot be committed, the user may be allowed to modify the transactional service profile.

4 Behavioral adaptation strategy

In this section, we present the behavioral adaptation strategy which supports the policy-based adaptation approach of CATS services. First, we will introduce the basic concepts associated with this adaptation technique. Then, we will define the mechanism used to ensure behavioral adaptation of CATS services and the underlying adaptation strategies.

4.1 CATS specification

Our proposal allows a dynamic configuration of transactional properties during execution. The proposed model, as designed, is based on the Open Nested Transaction model [31], [32], and [33]. The adaptation aspect is obtained by associating each

transaction with an environment descriptor. An environment descriptor refers to the resources state and the conditions of services and user's runtime environment. An environment descriptor aggregates a set of context descriptors. These descriptors represent a wide range of contextual information related to the user, the transactional service, and the execution environment, that can be quantified and queried. Thus, the context descriptor is a representation of the variable parameters that can influence the execution of a transactional service. In the context of pervasive environment, these parameters depend on the wireless network, communication speed, connection status, memory and cache size, battery capacity, location, user mobility, etc. Each parameter can be qualified by variable states, for instance, the possible states for bandwidth are "high", "medium", "low", and for the memory, "available", "half", "full". Hence, we propose to define a CATS service as a set of activities, each activity can be executed by component transactional services (see Figure 1).

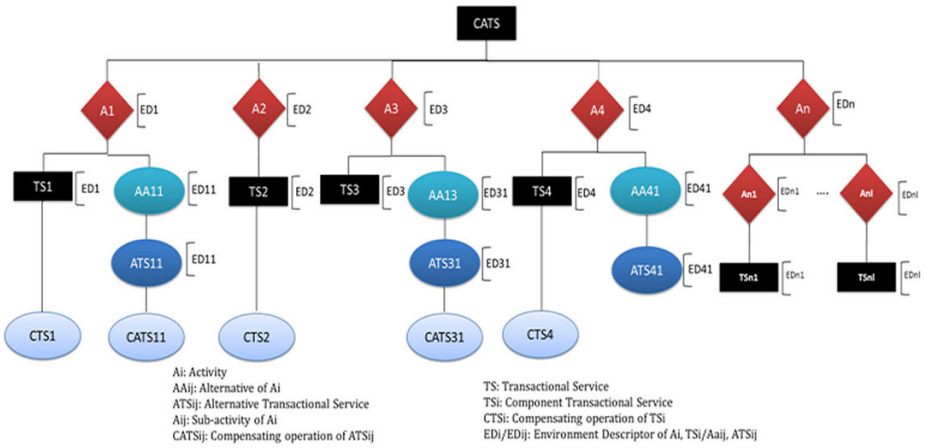

Fig. 1. Succinct illustration of CATS service structure

By analogy to the proposed structure, we model a CATS service using a tree transaction model where internal nodes represent the activities and leaf nodes are the component transactional services. Each activity has a primary transactional service and a set of alternative transactional services, which are associated each with an environment descriptor. In our transaction model, an activity is committed if one of the associated component transactional services (primary/alternative) is successfully executed. However, the activity fails, if none of the associated services is committed. Each activity can have alternative activities; in this case, it is said to be replaceable. An alternative activity is only executed if its primary activity is not committed or if a context change or a particular event occurs. All activities can be critical or non-critical. A CATS service can only be committed if all its critical activities have been committed. An activity is said to be structured if it consists of more than one activity.

A structured activity is modeled by a tree of activities and services. A service can be:

- Compensable or non-compensable: a compensable service offers two operations. The first operation enables the acquisition of the resource exposed by the service; the second one allows the cancellation of the first operation effects. A non-compensable service allows reserving a resource during a determined period, and to commit or cancel the requested reservation.
- Replayable or not replayable: in case of failure, the reexecution of services increases the commit rate, these services are characterized by a number of authorized attempts and a delay between these attempts.
- Replaceable or non-replaceable: each participating service can have alternative services.
- Critical or non-critical: An uncommitted critical service requires the abort of CATS composition and other component services. In contrast, the composition can be committed even if a non-critical service fails.

4.2 The proposed mechanism

Context monitoring is an essential step in adapting transactional services to the context, since it is possible to determine the required context for a correct execution of transactional services, thus ensuring that the violation of contextual requirements is detected and adaptation actions are triggered accordingly. In this sense, the monitoring of context should be carried out a priori and a posteriori of transactional service selection. We refer to this approach as proactive context monitoring. It is used to determine the execution context of all the selected transactional services. On the basis of proactive monitoring, dynamic binding allows to choose the best transactional service (i.e., in terms of qualified context descriptor) to associate for each abstract activity in the user task. Therefore, the other services are detected and rejected simultaneously. Figure 2 outlines the different components involved in the execution of CATS services.

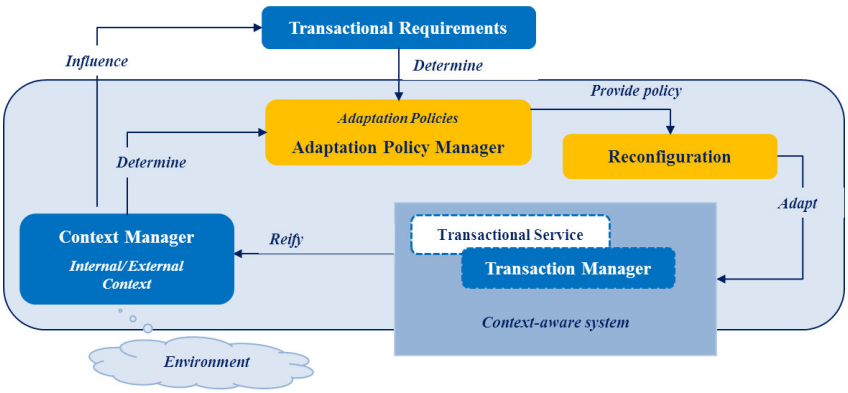

Fig. 2. Adaptation mechanism architecture

As depicted in Figure 3, CATS service composition API takes as input the user request, which includes the description of the user's task and all contextual and transactional requirements.

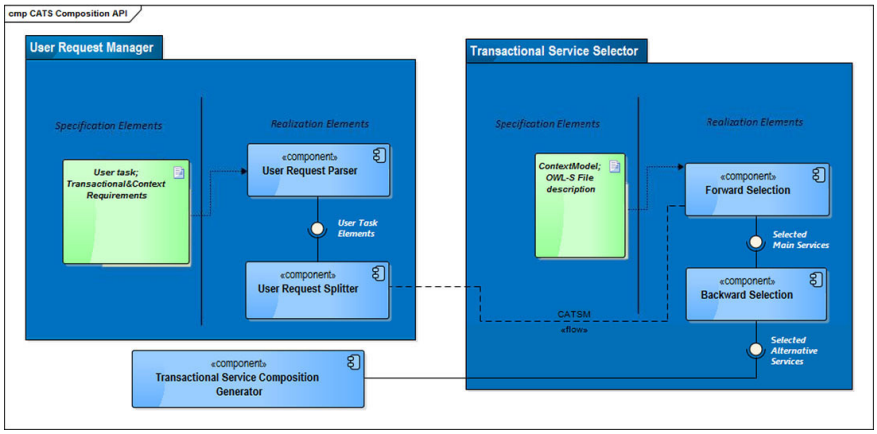

Fig. 3. CATS service composition API

To accomplish the required task, CATS Composition API goes through the following steps:

- The *User Request Parser* component analyzes the specification of the user task and determines its underlying abstract activities and the set of contextual and transactional constraints imposed on the entire task.
- The *User Request Manager* component invokes the Transactional Service Selector to acquire a set of candidate transactional services for each activity in the task.
- In case of failure, the selection algorithm chooses a set of alternative services for each activity, which meet contextual and transactional constraints.
- Based on the highest ranked transactional services, the *Transactional Service Composition Generator* creates an executable composition capable of satisfying the user task.

Service substitution is triggered when one or more running services don't meet the required context. Alternative services can be determined in two ways. First, based on the previously selected services; if these services are no longer available or their environment descriptors don't correspond to the current context, a second solution can be provided. This solution consists of performing a selection of services that checks the current context for the failed activity (or activities). For each activity, the selected services must have an environment descriptor that corresponds to the execution context. Context-driven service substitution is more appropriate when we are dealing with the execution of long-running tasks in very dynamic environments, such that: (i) new services can join the environment while the execution of the user task, and (ii) the discovery duration of the service is negligible compared to the execution time of the user task.

To perform the substitution of services in so-called pervasive environments, we use the Dynamic Binding technique.

Dynamic binding [34] aims to associate a service by activity among those previously selected according to their execution contexts and the associated transactional profile, thus guaranteeing an optimal commit rate for the task composition. To do this, dynamic binding requires a preliminary phase of context monitoring, which determine the services execution context. Hence, dynamic binding checks whether a service provides an environment descriptor similar or different from the one previously associated. If many services meet this condition, the service that provides the most exact similarity function is associated with the considered activity.

The adaptation process, which we present, has been implemented around a number of interfaces. These interfaces are used to ensure a flexible, reusable and exchangeable character of our architecture. The implementations of these are based on the design pattern "strategy pattern" to provide a flexible strategy adjustment, based on a configuration file. The component diagram shown in Figure 4 describes CATS service adaptation mechanism.

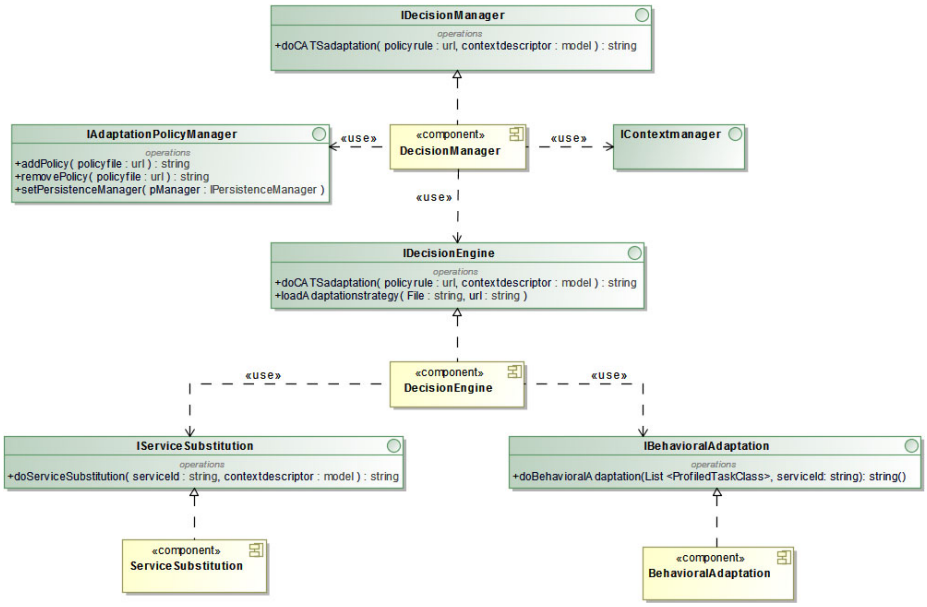

Fig. 4. Components implementing CATS service adaptation mechanism

4.3 Profiled task class concept

In the previous section, we proposed an approach for adapting transactional services to Context. However, the proposed approach is not sufficient to ensure that the mentioned requirements are met during the execution of services composition, since the environment descriptors of the services may change at runtime. To deal with this issue,

we present an adaptation approach for CATS composition based on two adaptation strategies. First, the substitution of transactional services, which consists of replacing the participating services by alternative services. When the service substitution fails, we proceed to the second adaptation strategy, which consists of adapting the behavior of the composition without altering the user's task. This strategy is based on the concept of "Task Class" [35], which we extend by the concept of "Profiled Task Class" to define the set of abstract compositions of transactional services which are functionally equivalent but showing different behaviors. Figure 5 displays the global adaptation approach.

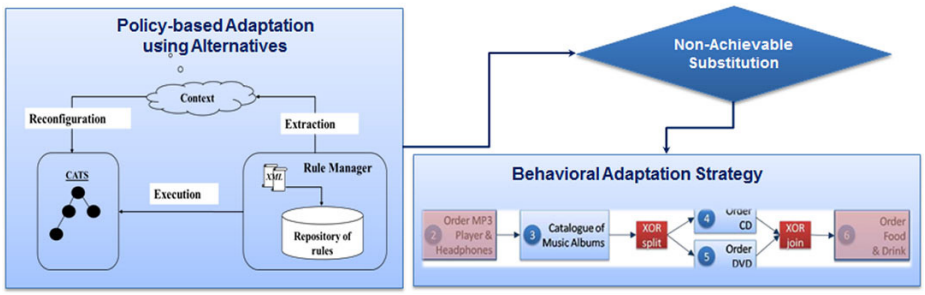

Fig. 5. Illustration of CATS adaptation approach

The behavioral adaptation strategy aims to accomplish the user's task using an alternative abstract composition to accomplish the required task. The concept of "Profiled Task Class" starts from the idea that the user task can be performed in different ways depending on the profile associated with each activity. This is possible either by changing the order in which the abstract activities are executed (i.e., according to their criticality), or by dividing coarse-grained abstract activities into fine-grained activities (i.e., if the activity is replaceable), or by merging fine-grained activities into coarse-grained abstract activities (i.e., according to the criticality of the activity) while respecting the profile assigned to each activity in the user task. To do so, we propose to explore the "Profiled Tasks Class" associated with the user task and to analyze the available abstract compositions. Specifically, the proposed strategy consists of building a sub-composition that is functionally equivalent to the failed one and behaves differently.

The behavioral comparison of two abstract compositions is achieved by transforming these abstract compositions into graphs. Indeed, the graphs represent a fairly robust data structure which can have several uses, in particular the specification of compositions behavior. Thus, we can formulate the problem of sub-compositions comparison as subgraphs matching search problem. Such correspondences are generally established by the detection of subgraphs isomorphism [36]. The latter allows to determine the exact correspondence between subgraphs, where only the order of the abstract activities changes. However, when the service substitution fails, changing the order of the activities does not provide any workaround. For this reason, we use the subgraph correspondence approach presented by [35]. The author proposed to use another technique to solve this correspondence problem by adopting flexible correspondence between subgraphs supported by the technique of "vertex subdivision". This technique

is applied using the concept of the "topological minor". Using graph minor theory [37], the relationship between a subgraph and its topological minor can be described as a homeomorphism of subgraphs with disjoint vertices vdSH (vertex disjoint Subgraph Homeomorphism) [38]. The construction of the different combinations in the same class of tasks is dictated by the behavioral profile of each activity forming this task. These alternative compositions are generated using three main methods:

- By changing the order in which abstract activities are executed, critical activities are given higher priority and will be executed first depending on their resource consumption and the current execution context.
- By dividing coarse-grained abstract activities into fine-grained activities, or by merging fine-grained activities into coarse-grained abstract activities.
- By omitting the activity or the sub-composition of activities that are non-critical, in case of failure after using the first two methods.

Our approach is based on the assumption that profiled task classes are built gradually based on learning techniques and user feedback. More precisely, each time a new task is defined, our middleware platform uses planning techniques to automatically generate alternative abstract compositions allowing this task to be satisfied according to the user requirements in terms of functional specifications, the context and transactional semantics of the activities forming the task. Formally, the concept of "Profiled Task Class" can be defined as follows:

– $G_T = (V, E, \beta)$ is the graph representing the user's tasks, V is the set of vertices, $E \subset V \times V$: a set of edges and β: $V \rightarrow P$ is a function which assigns profiles to vertices.

A Profiled Task Class is the set of graphs $G = \{G_1, ..., G_n\}$ where each graph:

– $G_i = (V_i, E_i, \beta_i)$ *with* $i \in [1, n]$ and (G_i is isomorphic to G_T *or* G_i is a subdivision of G_T *or* G_T is a subdivision *of* G_i.)

To optimize the commit rate of CATS composition, the adaptation process is based on two main phases:

Checking the required context: The main coordinator sends a message to each sub-coordinator to verify the required context for execution. If the current context is qualified, each sub-coordinator sends a "Yes" message to the main coordinator indicating that the transactional service can normally be executed. Otherwise, it sends a "No" message. For more detail regarding CATS commit protocol, we refer to a previous work [39].

Commit phase: In the previous phase, if one of the sub-coordinators responds that the current context is not qualified, the decision will be as follows:

– If the transactional service is replaceable and the current context matches the environment descriptor of the alternative service, then the associated alternative service is executed.
– If the transactional service is critical, non-replaceable and no alternative sub-composition in the profiled task classes has been identified, the composition is

aborted (i.e. in case of failure). In this case, the committed transactional services, if any, will be compensated.
– If the transactional service is neither critical nor replaceable, it is simply ignored.

If the current context is qualified, adaptation actions are performed as follows:

– If the transactional service is replayable, it is re-executed according to the number of authorized attempts (i.e. in case of failure).
– If the transactional service is critical and replayable, it is re-executed according to the number of authorized attempts and the re-execution parameters are updated.
– If the service is replaceable, the corresponding alternative service is performed.
– If the service is non-critical, non-replaceable, and non-replayable, it is simply ignored.

Figure 6 shows the different adaptation actions according to the behavior profile of each activity in CATS composition.

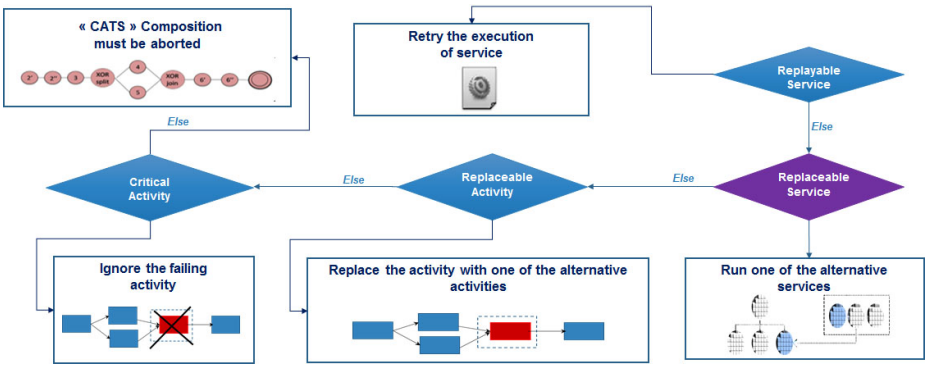

Fig. 6. CATS composition behavior according to activity profile

5 Evaluation

We proceed to a performance evaluation of our CATS composition approach. Our diagnosis was processed on a machine with the following configuration: 2.6 GHz for the processor, 8 GB of RAM, Windows 7 as the operating system and version 1.8 of the Java language development kit (JDK 1.8). We considered two simulation scenarios corresponding to different degrees of complexity. For each of these scenarios, we performed a battery of ten tests to retrieve a significant average of the instrumented parameters. Based on the JProfiler tool (version 8.1.1), we obtained the results presented in Figures 7 and 8.

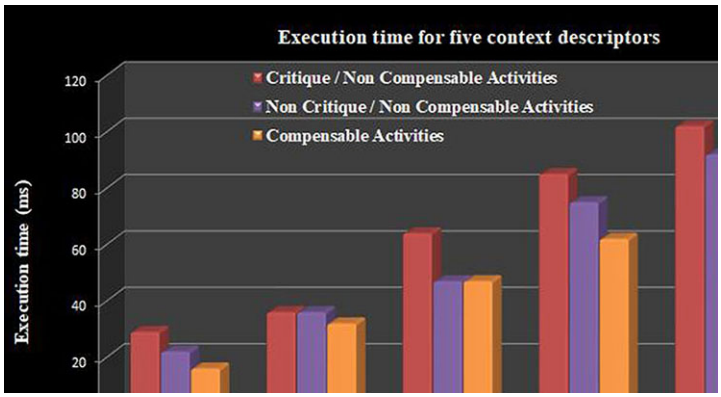

Fig. 7. Execution time for five context descriptors by varying the behavioral profile of activities

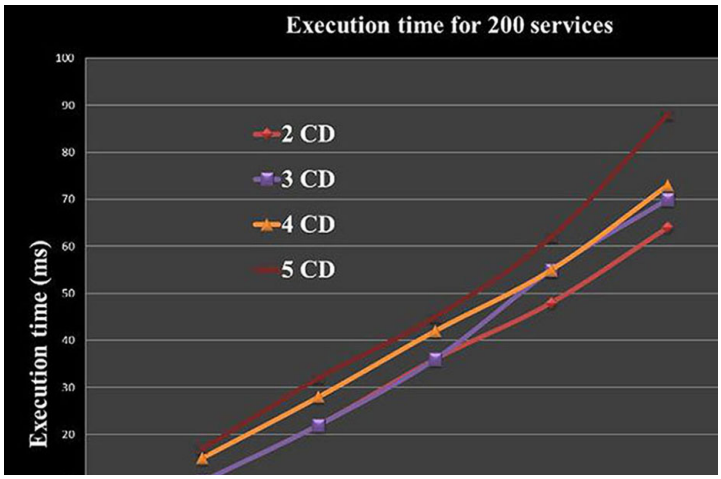

Fig. 8. Execution time by varying the number of context descriptors

The test we have performed consists in varying the number of activities between 10 and 50, and the number of context descriptors between 2 and 5; the number of services per activity has been set at 200. We stipulate that for a given context descriptor, only one context parameter is considered. For precision reasons, we carry out a battery of ten tests and we calculate the average value of the obtained results. Figure 7 illustrates the execution time according to the behavioral profile of activities. We set the number of context descriptors at 5, we vary the number of activities between 10 and 50 and the behavioral profile by group of activities: "Critical/Non-Compensable" Activities, "Non-Critical/Non-Compensable" Activities and "Compensable" Activities.

The obtained results show that the execution time increases (up to 102 ms) with the number of "Critical/Non-Compensable" Activities, which is an expected result. Figure 8 illustrates the execution time according to the number of context descriptors.

We set the number of services per activity at 200 and vary the context descriptors between 2 and 5; the activities are all "Compensable". The obtained results obtained show that the execution time increases (up to 89 ms) depending on the number of context descriptors, which is also an expected result (i.e. a higher number of context descriptors requires more processing time). Both figures illustrate that the execution time increases almost linearly with the number of activities in the composition. In general, CATS compositional approach shows very satisfactory performance (less than 102 ms) vis-à-vis spontaneous interactions with users in pervasive environments.

6 Conclusion and perspectives

One of the main advantages of our work is the exploitation of transactional properties for adapting "CATS" services. Our approach is based on the use of policies as an adaptation model. This model is supported by mechanisms such as compensation, re-execution, substitution, dynamic reconfiguration of policies and dynamic adaptation of transactional requirements of users (i.e. Dynamic binding, Retry/Redo, Substitution, Dynamic reconfiguration, Dynamic adaptation of transactional needs). The policy-based adaptation approach is complemented by a behavioral adaptation strategy that dynamically reconfigures abstract activities of the user task through the concept of "Profiled Task Class". Behavioral adaptation represents an important alternative to the substitution of services widely studied by existing adaptation approaches.

The potential improvements that can be made to our presented work consist in fully validating our composition adaptation approach experimentally. More precisely, we are interested in studying the response time of our selection algorithm in relation to recall, precision and rapidity requirements in pervasive environments. In this sense, we are interested in defining a utility function that allows determining the best behavioral adaptation solution. In order to improve the correspondence between parameters and detect the non-symmetrical substitutability between two services, using machine learning techniques in selection algorithms of alternative compositions will improve the performances based on methods like random-forest and boosting, or techniques like MDP (Markov Decision Process) and Q-learning.

7 References

[1] Schilit, B., Adams, N. Want, R. (1994). "Context-aware computing applications", Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, pp. 85–90, IEEE Computer Society Press, 1994. https://doi.org/10.1109/WMCSA.1994.16

[2] Dey, A. K., Abowd, G. D. (2001). "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction (HCI) Journal, vol. 16, pp. 2–4, 2001. https://doi.org/10.1207/S15327051HCI16234_02

[3] Karlsen, R., Jakobsen, A. B. A. (2003). "Transaction service management an approach towards a reflective transaction service", 2nd International Workshop on Reflective and Adaptive Middleware, Rio de Janeiro, Brazil, June 2003.

[4] Santos, N., Veiga, L., Ferreira, P. (2004). "Transaction policies for mobile networks", fifth IEEE International Workshop on Policies for Dist. Systems and Networks, 2004. https://doi.org/10.1109/POLICY.2004.1309150

[5] Rouvoy, R., Serrano-Alvarado, P., Merle, P. (2006). "Towards Context-Aware Transaction Services", Proceedings of the 6th International Conference on Distributed Applications and Interoperable Systems (DAIS'06), Bologna, Italy, Lecture Notes in Computer Science, Springer-Verlag, vol. 4025, pp. 272–288, June 2006. https://doi.org/10.1007/11773887_21

[6] W3C, Web Services Policy 1.5—Framework, 04 September 2007. [Online], Available: https://www.w3.org/TR/ws-policy/ [Accessed Aug. 15, 2021].

[7] Micalsen, T., Tai, S., Ravellou, I. (2002). "Transactional attitudes: Reliable compositions of autonomous Web services", Workshop on Dependable Middleware Based Systems, March 2002.

[8] Pires, P. F., Benevides, M. R., Mattoso, M. (2003). "Building Reliable Web Services Compositions", Web, Web-Services, and Database Systems, LNCS 2593, pp. 59–72, Springer, 2003. https://doi.org/10.1007/3-540-36560-5_5

[9] Gaaloul, W., Bhiri, S., Godart, C. (2004). "Discovering Workflow transactional behaviour Event-based Log", 12th International Conference on Cooperative Information Systems CoopIS'04, 25–29 October 2004, Larnaca, Cyprus. https://doi.org/10.1007/978-3-540-30468-5_3

[10] Lakhal, N. B., Kobayashi, Y., Yokota, H. (2009). "FENECIA: failure endurable nested-transaction based execution of compo site Web services with incorporated state analysis", VLDB Journal, 18(1), pp. 1–56, 2009. https://doi.org/10.1007/s00778-007-0076-8

[11] Liu, A., Li, Q., Huang, L., Xiao, M. (2010). "FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services", Services Computing, IEEE Transactions on, 3(1), pp. 46–59, Jan 2010. https://doi.org/10.1109/TSC.2009.28

[12] Younas, M., and Mostefaoui, S. K. (2010). "Context-aware mobile services transactions", 24th IEEE international conference on advanced information networking and applications AINA, Perth, Australia, pp. 705–712, 2010. https://doi.org/10.1109/AINA.2010.157

[13] El Haddad, J., Manouvrier, M., Rukoz, M. (2010). "TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition", IEEE Trans. Serv. Comput., 3(1), pp. 73–85, January 2010. https://doi.org/10.1109/TSC.2010.5

[14] Ying, Y., Zhang, B., Zhang, X., Zhao, Y. (2009). "A Self-healing composite Web service model", Services Computing Conference, IEEE Asia-Pacific, pp. 307–312, December 2009. https://doi.org/10.1109/APSCC.2009.5394108

[15] Maamar, Z., Benslimane, D., Anderson, A. (2006). "Using Policies to Manage Composite Web Services", IEEE IT Professional, 8(5), September/October 2006. https://doi.org/10.1109/MITP.2006.124

[16] Jiuxin, C., Junzhou, L., Zhang, S., Zheng, X., Bo, L., Zhu, G., Zhang, B. (2012). "A Context-Aware Recovery Mechanism for Web Services Business Transaction", pp. 352–359, IEEE SCC 2012.

[17] Simmonds, J., Ben-David, S., Chechik, M. (2010). "Guided Recovery for Web Service Applications", Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE '10, pp. 247–256, New York, NY, USA, 2010. https://doi.org/10.1145/1882291.1882328

[18] Gabrel, V., Manouvrier, M., Murat, C. (2014). "Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming", In International Conference on Service Oriented Computing, pp. 108–122. Springer, 2014. https://doi.org/10.1007/978-3-662-45391-9_8

[19] Llinás, G. A. G., Nagi, R. (2015). "Network and qos-based selection of complementary services", IEEE Transactions on Services Computing, 8(1), pp. 79–91, 2015. https://doi.org/10.1109/TSC.2014.2299547
[20] Luo, Y.-s., Qi, Y., Hou, D., Shen, L.-f., Chen, Y., Zhong, X. (2011). "A novel heuristic algorithm for qos-aware end-to-end service composition". Computer Communications, 34(9), pp. 1137–1144, 2011. https://doi.org/10.1016/j.comcom.2010.02.028
[21] Comes, D., Baraki, H., Reichle, R., Zapf, M., Geihs, K. (2010). "Heuristic approaches for qos-based service selection", International Conference on Service-Oriented Computing, p. 441–455. Springer, 2010. https://doi.org/10.1007/978-3-642-17358-5_30
[22] Do Prado, P. F., Nakamura, L. H., Estrella, J., Santana, M. J., Santana, R. H. (2013). "A performance evaluation study for qos-aware web services composition using heuristic algorithms", ICDS, The Seventh International Conference on Digital Society, pp. 53–58, 2013.
[23] Elsayed, D. H., Nasr, E. S., Alaa El Din, M., Gheith, M. H. (2017). "A new hybrid approach using genetic algorithm and q-learning for qos-aware web service composition", International Conference on Advanced Intelligent Systems and Informatics, pp. 537–546. Springer, 2017. https://doi.org/10.1007/978-3-319-64861-3_50
[24] Wu, Q., Zhu, Q. (2013). "Transactional and qos-aware dynamic service composition based on ant colony optimization", Future Generation Computer Systems, 29(5), pp. 1112–1119, 2013. https://doi.org/10.1016/j.future.2012.12.010
[25] Zhao, Y., Tan, W., Jin, T. (2017). "Qos-aware web service composition considering the constraints between services", Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing, pp. 229–232. ACM, 2017. https://doi.org/10.1145/3127404.3127451
[26] Chen, M. (2015). "QoS-aware Service Composition and Redundant Service Removal". PhD thesis, Concordia University, 2015.
[27] Khanouche, M. E., Amirat, Y., Chibani, A., Kerkar, M., Yachir, A. (2016). "Energy-centered and qos-aware services selection for internet of things", IEEE Transactions on Automation Science and Engineering, 13(3), pp. 1256–1269, 2016. https://doi.org/10.1109/TASE.2016.2539240
[28] Bhiri, S., Perrin, O., Godart, C. (2005). "Ensuring required failure atomicity of composite web services", Proceedings of the 14th international conference on World Wide Web, WWW'05, New York, NY, USA, pp. 138–147, 2005. https://doi.org/10.1145/1060745.1060769
[29] Di Penta, M., Esposito, R., Villani, M. L., Codato, R., Colombo, M., Di Nitto, E. (2006). "WS-Binder: a Framework to Enable Dynamic Binding of Composite Web Services", Proceedings of the international workshop on Service-oriented software engineering. ACM, New York, NY, USA, pp. 74–80, 2006. https://doi.org/10.1145/1138486.1138502
[30] G Amundsen, S. L., Eliassen, F. (2006). "Combined Resource and Context Model for QoS-Aware Mobile Middleware", ARCS. pp. 84–98, 2006. https://doi.org/10.1007/11682127_7
[31] Moss, J. E. B. (1985). "Nested Transactions: An Approach to Reliable Distributed Computing". Cambridge, MA: M.I.T. Press, 1985.
[32] Gray, J., Reuter, A. (1993). "Transaction Processing Concepts and Techniques". Series in Data Management Systems, Morgan Kaufmann, 1993.
[33] Bernstein, P. A., Newcomer, E. (1997). "Principles of Transaction Processing for the System Professional", Morgan Kaufmann, ISBN 1-55860-415-4, 1997.
[34] Châtel, P., Malenfant, J., Truck, I. (2010). "QoS-based Late-Binding of Service Invocations in Adaptive Business Processes", IICWS, pp. 227–234, 2010. https://doi.org/10.1109/ICWS.2010.74

[35] Ben Mabrouk, N. (2012). "QoS-aware Service-Oriented Middleware for Pervasive Environments, Mobile Computing". Universite Pierre et Marie Curie—Paris VI, 2012.
[36] Labriji, A., Charkaoui, S., Abdelbaki, I., Namir, A., Labriji, E. H. (2017). "Similarity Measure of Graphs". International Journal of Recent Contributions from Engineering, Science & IT (i-JES), vol. 5, Issue 2, pp. 42–56, 2017. https://doi.org/10.3991/ijes.v5i2.7251
[37] G Robertson, N., Seymour, P. D. (1995). "Graph Minors .XIII. The Disjoint Paths Problem". Journal of Combinatorial Theory, Series B 63, 1, pp. 65–110, 1995. https://doi.org/10.1006/jctb.1995.1034
[38] Xiao, Y., Wu, W., Wang, W., He, Z. (2007). "Efficient Algorithms for Node Disjoint Subgraph Homeomorphism Determination", CoRR abs/0709.1227, 2007.
[39] Ettazi, W., Hafiddi, H., Nassar, M. (2018). "A Context-Driven Commit Protocol for Enhancing transactional Services Performance in Pervasive Environments", International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC), Volume 10, Issue 4, 2018. https://doi.org/10.4018/IJAPUC.2018100102
[40] Chouiref Latreche, Z., Belkhir, A., Hadjali, A. (2013). "Advanced Profile Similarity to Enhance Semantic Web Services Matching". International Journal of Recent Contributions from Engineering, Science & IT (i-JES), vol. 1, Issue 1, pp. 13–20, 2013. https://doi.org/10.3991/ijes.v1i1.2963
[41] Misbah, A., Ettalbi, A. (2018). "Automatic Conversion of a Conceptual Model to a Standard Multi-view Web Services Definition". International Journal of Recent Contributions from Engineering, Science & IT (i-JES), vol. 6, Issue 1, pp. 43–56, 2018. https://doi.org/10.3991/ijes.v6i1.8285

8 Authors

Widad Ettazi is Professor of Software Engineering at National Higher School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University in Rabat, Morocco. Her research interests are Context-Aware Service oriented Computing, Cloud Computing.

Hatim Hafiddi is a full Professor of Software Engineering at National Institute of Communication (INPT), Rabat, Morocco. He is head of the EVEREST (InnovatiVE REsearch on Software, Systems and daTa) research Team / STRS Laboratory. He holds an Engineer degree and a PhD degree in Software Engineering from National College of IT (ENSIAS), Rabat, Morocco. His research interests are Context-Aware Computing, Integration and interoperability, Smart service oriented systems, and Model driven Engineering.

Mahmoud Nassar is a full Professor at National Higher School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University in Rabat, Morocco. He is Head of IT architecture and Model driven Systems development (IMS) team of Rabat IT Center. He received his PhD in Computer Science from the INPT Institute of Toulouse, France. His research interests are Context-Aware Service-Oriented Computing, Component based Engineering, Model-Driven Engineering, Cloud computing, and Cloud Migration. He leads numerous R&D projects related to the application of these domains in Embedded Systems, smart cities, e-Health, e-Tourism.

Article submitted 2021-08-02. Resubmitted 2021-10-17. Final acceptance 2021-10-24. Final version published as submitted by the authors.