Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : http://oatao.univ-toulouse.fr/
Eprints ID : 16941

The contribution was presented at CloudTech 2015 :
http://www.macc.ma/cloudtech15/


Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr
A Cloud-Based Architecture for Transactional Services Adaptation

Widad Ettazi\textsuperscript{1,}\textsuperscript{*}, Hatim Hafiddi\textsuperscript{1,2}, Mahmoud Nassar\textsuperscript{1} and Sophie Ebersold\textsuperscript{3}

\textsuperscript{1}IMS Team, SIME Laboratory, ENSIAS, Mohammed V University, Rabat, Morocco
\textsuperscript{2}ISL Team, STRS Laboratory, INPT, Rabat, Morocco
\textsuperscript{3}MACAO Team, IRIT Laboratory, University Toulouse 2-Le Mirail, Toulouse, France
widad.ettazi@um5s.net.ma, hatim.hafiddi@gmail.com, nassar@ensias.ma, ebersold@univ-tlse2.fr

Abstract—Advances in wireless communications and mobility have increased the use of smart mobile applications. As a result of the remarkable increase of mobile devices and the pervasive wireless networks, a large number of mobile users are requiring personalization services customized to their context. The mobile cloud-computing paradigm from a context-aware perspective aims to find effective ways to make cloud services aware of the context of their customers and applications. Another major challenge for context-aware cloud services is to exploit the benefits of cloud computing to manage transaction processing throughout the life cycle of a service. In this paper, we focused on the need of loosely coupled context-supporting components that work with a transaction-aware service infrastructure to adapt services to the context of the user and his mobile device. We propose a cloud-based middleware for transactional service adaptation (CM4TSA) by adding the “Adaptation as a Service” layer into basic cloud architecture, to perform the correct execution of transactional service according to the user context.

Keywords—context-awareness; transactional service; cloud computing; adaptation as a service; transaction model.

I. INTRODUCTION
Transaction processing has always been at the core of data-driven enterprise applications [3]. However, with the introduction and growing popularity of service-oriented architectures, transactional mechanisms have become much more relevant. The loosely coupled nature of distributed service applications defeats some of the essential assumptions of existing transaction-processing systems and shifts the transaction management task from the core infrastructure to the service designer. In parallel, continuing advances in the field of B2B, E-commerce and telecommunications in terms of wireless and mobile devices have brought about the context-aware mobile computing paradigm [6]. Indeed, service-oriented architectures have a number of requirements in a transaction-based infrastructure; transactions must be able to adjust to systems that are not necessarily in a perfect environment. These systems will operate in a flexible, dynamic environment, but less reliable and that presents contextual requirements (e.g., requirements and preferences expressed or implied by the user, connectivity, bandwidth, etc.) that hinder the execution of transactions. Moreover, the proliferation of smart applications is giving rise to a rapidly evolving subject area known as mobile cloud computing [7]. Many of the smart applications are now looking at the cloud-computing paradigm as an enabler to exploit its computing power, memory and storage resources to overcome the resource limitations of mobile devices [10]. The scalability, flexibility and stability offered by cloud services make them an ideal architecture to use in client applications in a resource limited mobile environment. Evolution in these three fields has promoted the birth of a new development paradigm called Context-Aware Transactional Service (CATS) (See Fig.1).

Figure 1. CATS as combination of three computing paradigm.

This paper investigates into the issue of context-awareness and transaction management in service-oriented applications and describes a context-aware transaction-processing system that facilitates coordination and communication of user and environment related context between service consumers and providers hosted in the cloud infrastructure.

The rest of this paper is organized as follows. The next section discusses some backgrounds and summarizes related work in the area. The proposed approach for managing CATS is presented in section 3. We present and describe in section 4, our cloud-based architecture for transactional service adaptation. We illustrate succinctly, in section 5, our approach for the E-tourism scenario. Section 6 concludes our work.

II. BACKGROUND AND RELATED WORK
A. Context-Awareness
Context-awareness is an intrinsic characteristic of intelligent environments [1], [2]. This computing paradigm appeared since the 90s driven by the work of [11]. For the author, the 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 [1], “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”. Context-awareness requires an effective approach for getting the right information at the right time to make the correct decisions for changing or optimizing application’s behavior.

B. Context-Aware Transaction Processing

To meet the variables requirements of transactional services, the need to relax ACID properties has been proposed in many researches since the early 90s. There was a great effort on extended transaction models [9]. Several standards specifications have been proposed, including WS-Transaction specification [16], [17] and Business Transaction Protocol [15]. However, they don’t take into account the context information. Many transactional models and techniques have been proposed [24], [25], [26], but they have limitations, namely, a non-consideration of the context information and the conception of advanced models with transactional properties that differ from one application to another. Authors in [5] introduce the Adaptable Model Transaction which allows programmers to define transactional alternatives for an application task. Another model is developed for context-aware transactions in the context of mobile systems [4]. This model provides a new set of transactional properties called RACCD (Relaxed Atomicity, Consistency, Context, Durability). The RACCD model considers the context criterion one of the main transactional properties that has not been taken into account in existing transactional models. The new criterion is expected to be more suited to the dynamic nature of context in mobile environments.

C. Cloud Computing

Cloud computing is one of the latest distributed computing paradigms that have emerged [18]. It promises reliable services delivered through infrastructures that are built on virtualization technologies and provides plentiful storage and processing capabilities [10], [14], [8]. Commercial applications of cloud computing are using Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS) delivery models, where software and resources are hosted on the cloud instead of the client [8]. Authors in [19] have proposed a definition of cloud computing, built on five characterististics:

- **Resource pooling**: is defined as the ability to serve a large amount of customers simultaneously and automatically redirect or redistribute client accesses based on load combined with location.
- **Broad network access**: includes facilitation from the provider for using standard protocols for client interaction.
- **On-demand self-service**: includes customization of hardware properties without the need for human interaction with the service provider.
- **Measured service**: consists on measuring service reliability and performance.
- **Rapid elasticity**: deals with the speed of which services are scaled up and down according to need and load.

D. Service Adaptation in the Cloud

There has recently been work on service adaptation in the Cloud. Authors in [20] present a framework for enabling context-aware mobile cloud services. The framework handles context capturing, context adaptation, tailoring candidate services, and running the adapted services. Another concept is proposed in [21] based on virtual services for service composition in the Cloud. The virtualization enables the composition process to succeed. Even though the traditional planning process might fail, the objective is to enable the planning process to go through. An approach focusing on market analysis based adaptation of the service request with cloud services discovery is introduced in [8]. The approach consists on choosing the most efficient service by means of cost-analysis of currently available services in the market. However, it does not take into consideration the context of the client device. Authors in [23] propose a flowable service model for seamless services integration to decrease the delivery costs of services in open cloud environments and to seek maximum satisfaction of both service providers and consumers. Another work presents an abstract model for a federation of services with their semantics and quality characteristics [22]. Despite the emerging research work in the area of context-aware cloud services, existing cloud computing platforms do not provide services to support transactional service adaptation.

III. CONTEXT-AWARE TRANSACTIONAL SERVICE

A. The Proposed Model

In our approach, we propose a new model for context-aware transaction services called Context-Aware Transactional Service Model (CATSM). This approach enables the implementation of context-aware services based on nested transactions models [27]. A transactional service according to CATSM is hierarchical and is based on the transaction model shown in Fig. 2.

![Figure 2. Context-aware transactional service model.](image)

To cope with the context-awareness aspect, we associate to each transaction a Context Descriptor (CD), which refers to the resources state and conditions of service execution environment (See Table 1).
TABLE I. EXAMPLE OF CONTEXT DESCRIPTOR FOR DEVICE SUB-CONTEXT

<table>
<thead>
<tr>
<th>Parameter</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery level</td>
<td>High, Moderate, Low</td>
</tr>
<tr>
<td>Screen Size</td>
<td>Large, Average, Small</td>
</tr>
<tr>
<td>Available memory</td>
<td>Available, Saturated</td>
</tr>
</tbody>
</table>

For more flexibility and resistance to failures, a sub-transaction may be associated to alternative transactions ATSij. We note that according to the context descriptor CDij of each ATSij, only one alternative will be invoked if the transaction to which it is associated has failed. In CATSM, we associate with each transaction a behavior type, namely, **replayable**, **replaceable**, **compensatable** and **critical**. A transaction is said to be replaceable if it may be replaced by an alternative transaction which will be invoked depending on the context descriptor. It is said to be replayable if it can be retried one or more times after its failure. A transaction is defined as compensatable if it provides mechanisms to undo its effects. It is critical if it requires the cancelling of the global transaction after its failure.

The commit of the global transaction is associated with four types of atomicity, depending on the semantic of the application and its requirements in terms of transactional properties. **Strict Atomicity** requires that all sub-transactions vote to commit before the validation of the global transaction. **Semantic Atomicity** requires that the global transaction consists of critical sub-transactions, some of which can be compensatable. **Relaxed Atomicity** is obtained in case the global transaction consists of any combination of critical or non-critical sub-transactions, compensatable or not. If one or more non-critical sub-transactions are aborted, the overall transaction can still be committed. **Classic Atomicity** requires that all sub-transactions are committed or none is. Fig. 3 illustrates our CATS metamodel. This metamodel is based on the following specification:

- **ContextAwareTransactionalService** aggregates a list of **Activity**.
- The **PropertySet** of an **Activity** is the set of transactional properties that this activity supports.
- The **PropertySet** of an **Activity** defines its **ExecutionContract**, which reflects the transaction execution model. This set is a sub-set of {**Strict Atomicity**, **Semantic Atomicity**, **Relaxed Atomicity**, **Classic Atomicity**}.
- Every **Activity** has a **BehaviorSet**, which contains a set of parameters defining the behavior type of each activity.
- The **BehaviorSet** of an **Activity** defines its **Profile**. This set is a sub-set of {**critical**, **replayable**, **replaceable**, **compensatable**}.
- **Activity** may be associated with **Alternative**.
- **Activity** may be associated with **Compensation**.
- Each **Activity** is associated with its **EnvironmentDescriptor** which defines the required environment conditions of execution.
- An **EnvironmentDescriptor** aggregates a list of **ContextDescriptor**.
- A **ContextDescriptor** aggregates a list of **ContextParameter**.
- For a given **AdaptationPolicy** and **ContextDescriptor**, a set of **ContextCondition** is deducted.
- For a given **AdaptationPolicy** and **Profile**, a set of **ProfileCondition** is deducted.
- **ContextCondition** and **ProfileCondition** are specific **AdaptationCondition**.

Figure 3. CATS adaptation metamodel.
An AdaptationCondition can involve the execution of an ordered set of Actions.

An AdaptationPolicy aggregates a set of AdaptationCondition, Action and Rule.

For a given AdaptationPolicy and Action, a Rule is associated.

B. Adaptation Policy for CATS

Context-aware systems are generally associated with the specification of policies that define the required behavior of a system in response to a context state. Indeed, in our approach we specify policies that clearly separate the control of adaptation from adaptation mechanisms. Fig. 4 describes the adaptation process that can be broken down into three phases:

Context gathering phase: this phase describes how to take into account the context information in the service description based on Context Based Web services Description Language (CWSDL) [12], [13]. CWSDL was developed to improve the WSDL standard of the W3C and provides a platform for retrieving context information. The context information is collected from the runtime environment (i.e., users, networks, resources, etc.).

Decision phase: based on the context information gathered in the first phase, the application must decide which reconfiguration operations will be performed to adjust to new circumstances. This step corresponds to the application adaptation strategy.

Reconfiguration phase: once the adaptation policy is chosen, a reconfiguration mechanism will automatically be responsible for the modifications.

IV. CLOUD-BASED ARCHITECTURE FOR CATS

In this section, we describe our cloud-based architecture which is an extension of the basic cloud architecture by adding a special layer “Adaptation as a Service” (AaaS) that enables service adaptation. We apply AaaS into CM4TSA middleware for transactional service adaptation. CM4TSA encapsulates a set of modules for executing the user’s service requests and allows service adaptation according to the context. These modules form the “Adaptation as a Service” that is hosted in the cloud. Fig. 5 describes different layers of the proposed architecture:

End-User
Software as a Service
Adaptation as a Service
Platform as a Service
Infrastructure as a Service

Figure 5. Cloud-based architecture for transactional service adaptation.

We elaborate each module given in the CM4TSA middleware as follows:

Context Manager: it provides context information and the mechanisms to collect and update data in case of context changes.

Adaptation Policy Manager: is responsible for inspecting the adaptation policy and converting the XML file into a data format that will be used in the reconfiguration module. The adaptation policy is determined by transactional requirements.

Reconfiguration Module: is responsible for evaluating and interpreting the adaptation policy based on the context state information provided by the context manager and the inspecting result of the policy manager to trigger the execution of the appropriate adaptation. The adaptation policy processing is performed at the time of loading, time of the transaction initiation or its failure (e.g., to be either rerun or to run an alternative) and when a change in the context state occurs. Based on the context state and the data retrieved from the adaptation policy, the module decides which strategy will be triggered (e.g., identify the alternative to run). The reconfiguration module invokes the transaction manager which is responsible for operating the real adaptation (e.g., performing alternatives, updating the re-execution parameters).

Transaction Manager: Once the CATSM structure is identified by the reconfiguration manager, the Transactional Service Coordinator (TSC) handles the processing and the...
execution of the global transaction. Then it submits the sub-
transactions to the sub-coordinators TSCI, which are associated
with the different services TSi. Each TSCI is running its TSi
service and exchanges messages with the TSC.

V. E-TOURISM SCENARIO

Let’s consider a travel planning system. Mr John plans to
attend an art show scheduled in Marrakech. For this, he intends
to book a flight, a hotel room, a restaurant table, and a place in
the art show. Reservations are not of equal importance, and the
restaurant table reservation can for example be omitted from
the transaction. The following figure sketches our cloud-based
architecture for the proposed system:

![Cloud architecture for travel planning service.](image)

If such a system is designed to be context-aware, the user
once logged in, specifies his preferences in terms of
destinations, will automatically receive the list of flights, and a
list of hotels close to its destination will be displayed. Once the
hotel reservation is made, the user will automatically receive a
list of restaurants according to his culinary preferences and to
the weather. If it is nice, for example, the system will suggest a
restaurant with a terrace. Finally, the system displays a list of
shows scheduled in the country of destination and prompts the
user to register. We should note that each reservation consists
of three operations, namely, the booking operation, the
payment transaction, and the sending of a specific document
for every reservation. This system is a well-known scenario of
a long transaction that requires extra ACID properties. The
compliance with the rules imposed by the ACID model is no
longer recommended in such a situation since a simple change
of context (e.g., low battery or a change in the connection state)
can induce the abort of the overall transaction. Thus, it is
desirable in the transaction management, that a transaction can
respond to contextual information and adapt its behavior to the
context changes.

Let’s illustrate our metamodel with an example. Fig. 7
presents a CATS prototype model in the case of Hotel
Reservation activity. For instance, the HotelReservation is an
Activity of the TravelPlanning service. This activity has a
RACID ExecutionContract (i.e., AtomicityDegree = {Relaxed
Atomicity}) which allows more flexibility in the execution of
its primitive activities (i.e., the reservation, the payment and the
document sending operations). Let’s imagine that a change in
the parameter Bandwidth occurs during the activity execution.
The HotelReservation adaptation policy is to verify first the
activity profile (i.e. ProfileCondition). In our case, the
HotelReservation activity has an alternative (i.e., Profile =
{Replaceable}). The second verification is that of the condition
on bandwidth (i.e., ContextCondition). The adaptation policy
provokes service adaptation which is executing alternative on
HotelReservation (i.e. Action) by running the transaction on
local device (i.e., Alternative On HotelReservation) whenever
the activity is replaceable and the bandwidth level is changed
from a high bandwidth to a low one (i.e. AdaptationCondition)
(See Fig. 7).

![CATS prototype model for Hotel Reservation Service.](image)
VI. CONCLUSION

The dynamic adaptation of service requests for cloud services in transactional systems is an issue that has yet to be sufficiently addressed by current research in this area. In this paper, we proposed a solution to this problem in order to allow adaptation to different types of transactional execution models according to the characteristics of the environment which are described in an environment descriptor and the application semantics. For this, we presented our approach for transactional service adaptation which is based on requirements specification in terms of transactional properties and the choice of an adaptation policy based on the alternative mechanism. We also proposed a cloud-enabled architecture that facilitates the management of such context-aware systems. The proposed platform allows simplified user access, handles demand elasticity and offers a high-level of abstraction.

The next phase of this research will be the design of a context-aware transaction commit protocol and a framework for the development of CATS. We are interested in designing our framework using cloud computing technology, because it enables multiple users to easily access, process, visualize, and search vast amounts of data from different service providers.

REFERENCES


