Towards a Context-Aware Adaptation Approach for Transactional Services

Widad Ettazi[1], Hatim Hafiddi[1,2], Mahmoud Nassar[1] and Sophie Ebersold[3]
[1]*IMS Team, SIME Laboratory, ENSIAS, Mohammed V University, Rabat, Morocco*
[2]*ISL Team, STRS Laboratory, INPT, Rabat, Morocco*
[3]*MACAO Team, IRIT Laboratory, University Toulouse 2-Le Mirail, Toulouse, France*

Keywords: Context-Awareness, ACID Properties, Transactional Service, Adaptation, Transaction Model.

Abstract: One goal of ubiquitous computing is to enable users to access and perform transactions from any location, at any time and from any device. In a context-aware environment, transaction management is a critical task that requires a dynamic adaptation to the changing context in order to provide service reliability and data consistency. In this paper, we propose a new approach for managing context-aware transactional services. Then we discuss the various researches that have addressed this issue. Finally, we propose a new model for the adaptability of context-aware transactional services called CATSM (Context-Aware Transactional Service Model) and the adaptation mechanisms to implement this model.

1 INTRODUCTION

Service-oriented architecture is becoming a major software framework for distributed applications such as e-business and enterprise systems. If enterprise applications use service-based technologies, it is expected that they provide transaction support. However, 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, 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. Therefore, it is imperative to take into account the context information in the management of transactions.

This paper investigates into the issue of context awareness and transactional aspects exigencies in service-oriented platforms and surveys how applying context-awareness to transactional services can help to improve data consistency, transactions execution and quality of service.

Despite the multitude of research works on transaction management in service-oriented systems, the notion of context-awareness in the management of these transactions is not yet addressed. Consider, for example, a simple transaction that books a room in a hotel. Current approaches will simply commit the transaction if the required room is available in the hotel. They do not take into account the context information such as *a room should be booked in a hotel which is located nearby*. To meet the variables requirements of transactional services, the need to relax the classical ACID (Atomicity, Consistency, Isolation and Durability) properties has been proposed in many researches since the early 90s. There was a great effort on extended transaction models (Elmagarmid, 1992), (Chrysanthis and Ramamrithan, 1994). This effort has been continued more recently in the context of mobile computing to satisfy the constraints of the execution environment (Segun et al., 2001). Researches have led to different notions of atomicity (strict, relaxed and semantic atomicity), consistency (strict or weak), and isolation (strict or relaxed allowing a flexible interleaving between transactions and a controlled sharing of intermediate results) (Younas et al., 2006).

Several standards specifications have been proposed, including WS-Transaction specification (IBM, Microsoft, BEA, 2005) and Business Transaction Protocol (OASIS, 2002). However, they

Ettazi W., Hafiddi H., Nassar M. and Ebersold S..
Towards a Context-Aware Adaptation Approach for Transactional Services.
DOI: 10.5220/0005382205530562
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 553-562
ISBN: 978-989-758-097-0
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

don't take into account the context information. Many transactional models and techniques have been proposed (Schafer et al., 2008), (Lakhal et al., 2009), (Choi et al., 2008), 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.

This article is organized as follows. We present in the next section the notion of context-awareness and ACID properties. Section 3 will be devoted to a review of related work. We introduce in Section 4 and Section 5 the proposed model for managing CATS and our adaptation approach. We illustrate in Section 6 the application of our proposal with an e-tourism motivating scenario. Section 7 highlights the adaptation mechanism. Finally, we conclude the paper in Section 8 with plans for the future.

2 BASIC CONCEPTS

2.1 Context-Awareness

Context-aware computing appeared since the 90s driven by the work of (Schilit et al., 1994). 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 (Dey et al., 2001), 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".

Unlike traditional transactions, context-aware transactions must adjust to the required context. By context, we mean information:

- Related to the execution context, such as characteristics of the used equipment, software environment, network connectivity, bandwidth, battery level of the terminal, presence in the environment of equipment such as printers and screens;
- About the user, such as his profile, location, preferences and choices;
- Related to climatic conditions, the level of ambient noise, traffic conditions, and temperature;
- Temporal such as date, time, the historical activity information.

Therefore, to ensure adaptation to the context, transactional services must be able to:

- Detect and capture the context information outlined above;
- Interpret the captured information to meet user's requirements;
- Respond to changes in the environment to provide a dynamic adaptation of transactional services.

2.2 ACID Properties

A transaction is the execution of a program containing a sequence of operations (e.g., reads and writes) performed on resources (e.g., databases, objects, components, services, etc.). The execution of a transaction ends with either commit or abort. The commit of a transaction makes the effects of its operations permanent while its abort cancels them (Bernstein, 1997).To control the adverse effects of concurrent modifications in a data system, the transaction defines four properties identified by the acronym ACID (Atomicity, Consistency, Isolation and Durability). The transaction model generally associated with the ACID properties is the flat transaction model (Gray and Reuter, 1993). This one is particularly suited to transactions running in parallel, short and with a limited data handling.

However, the diversity of current application contexts requires to define new transactional models to support transactions of long duration (hours, days, weeks), which handle potentially large and structured data, and for which a degree of cooperation between participants may be required to perform a complex task. Therefore, the ACID properties have limitations. Thus, they must be released because the property of atomicity is a major constraint for long-running transactions. Indeed, the risk of aborting the transaction increases in proportion to its length. In addition, the cost of aborted transactions also increases with the duration and complexity of the implemented process.

One of the main problems with transactional services is that they frequently suffer from failures. For instance, let's consider an e-shopping system that allows people to browse products catalogs in an electronic mall, to select, to book and to buy items. The application execution will not be the same if transactions are launched from a fixed terminal office, from a PDA while traveling by metro or from home using a laptop. Indeed, the characteristics of the mobile environment can have an impact on the commit of transactions:

- The execution time varies due to the limited resources of the mobile unit and the variable communication debit of the wireless networks;
- Communication cuts can occur due to a failure of the battery or outside a network coverage area;
- Low bandwidth affects the energy consumption due to the execution time increase.

Variability and constraints of the runtime environment can affect the performance of transactions leading to aborting transactions and to unexpected execution costs (e.g., communication price, execution time, etc.).

Furthermore, context-related failures occur when required context criteria are not met. For instance, in the case of a restaurant reservation transaction, if the system books a table in a Chinese restaurant while the user has specified his Moroccan culinary preferences, the transaction will be aborted. In this case, even though the service is available, it does not meet the required context.

3 RELATED WORK

This section reviews the research works on context aware transaction management in service-oriented systems.

(Younas et al., 2010) develop 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 protocol for enforcing them. Unlike ACID criteria, SACReD does not impose isolation policy thus allowing transaction to be partially committed. Resiliency property allows for alternative services wherein a given service fails or if it does not meet the required context. However, the proposed model is not generic and does not allow its extension to other degree of atomicity. In addition, the implemented protocol leads to failure when no available alternative service is found.

(El Haddad et al., 2008) present an approach for selecting and composing web services according to their transactional requirements, QoS characteristics and to the end-user preferences. The approach is not completely context-aware, since it doesn't include all context parameters. Moreover, the approach simply leads to a process failure when no suitable service is found.

(Hafiddi et al., 2011) propose a Model-Driven Engineering approach to achieve the context-aware service independently of platforms and application domains. Thus, the basic service focuses solely on the business logic and all adaptations of the ContextViews will be defined separately as Aspects called Adaptation Aspects. However, transactional requirements of context-aware services have not been addressed by this work.

(Serrano-Alvarado et al., 2004) introduce the Adaptable Model Transaction (AMT) according to multiple execution models on fixed and mobile hosts. The AMT model allows programmers to define transactional alternatives for an application task depending on context changes. Even though the approach is interesting, transactional service properties are not taken into consideration.

(Strunk et al., 2009) propose an approach for modeling adaptation in web service compositions to ensure a guaranteed quality of service for the whole composite service. A special adaptation mechanism is the rebinding of single services while the process is executed if the services fail or could not reach the needed QoS level. The approach supports rebinding in BPEL processes and is based restrictively on QoS metrics. Other contextual parameters such as user's requirements are not included in the design process.

(Muralidharan et al., 2008) develop a middleware solution called mConnect: a Context aware real time Mobile Transaction Middleware which handles the multiplicity of devices and provides a context agnostic view to the Transaction (back end) server. Nevertheless, the middleware is restricted to the computational resources available with the handheld devices.

(Rouvoy et al., 2006) propose CATE: a component-based architecture that is based on the Two-Phase-Commit (2PC) protocol and on a context-aware transaction service. The adaptation of CATE is achieved by components reconfiguration to select the most appropriate protocol with respect to the execution context. The adaptation approach is based on the selection of a suitable commitment protocol among 2PC derivatives and hence is limited to the classical commit protocols and does not take into account the context information such as location and time.

(Tang et al., 2008) propose a context-aware transaction model and a context-driven coordination algorithm based on the acceptable final states concept. The transaction model and coordination algorithm can dynamically adapt to the context, significantly improving the success rate of MUC (Mobile Ubiquitous Computing) transactions. However, the transactional model is limited to the compensatable transactions.

The proposed approaches for context-aware service adaptation are based primarily on creating customized services by specific development of the context-awareness code. Other works propose techniques for selecting the service that suits the user's request depending on the context of use, or adaptation by service rebinding and dynamic weaving of aspects that separates the implementation of non-functional requirements from the functional ones. However, this adaptation work has not focused on the transactional aspects of context-aware services.

4 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 (Moss and Hosking, 2006). A transactional service according to CATSM is hierarchical and is based on the transaction model shown in Figure1. In CATSM, the global transaction can be decomposed into a set of sub-transactions TSi, for example, a travel planning service can be represented as a global transaction, while its operations flight booking, hotel, restaurant and art show reservations can be represented by sub-transactions.

To cope with the context-awareness aspect, we associate to the global transaction a Context Descriptor (CD), which refers to the resources state and conditions of service execution environment (See Table1). Context Descriptor is mainly representing the following sub-contexts: transactional service, user, device, environment and wireless network contexts.

- *Device Sub-Context*: operating system, navigator type, supported type of data, screen size, battery level, available memory, computing capacity, etc.;
- *User Sub-Context*: profile, requirements, purpose, etc.;
- *Environment Sub-Context*: location, time, weather, etc.;
- *Transactional Service Sub-Context*: time interval, response time, availability, response rate, etc.;
- *Wireless Network Sub-Context*: Connectivity, bandwidth, cost, stability, etc.

Table1: Example of context descriptor for device sub-context.

Parameter	State
Battery level	High , Moderate, Low
Screen Size	Large, Average, Small
Available memory	Available, Saturated

For more flexibility and resistance to failures, a sub-transaction may be associated to alternative transactions ATSij, for example, in case of failure of the hotel booking, it is possible to book another hotel. 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 case the alternative context descriptor matches the current context, ATSij is initiated instead of the main transaction. A compensation mechanism is also invoked by adding to each sub-transaction TSi and each alternative ATSi a compensating transaction CATSi and CTSi respectively, for example, payment transactions are compensated in case of failure.

Figure 1 shows the general structure of the proposed model:

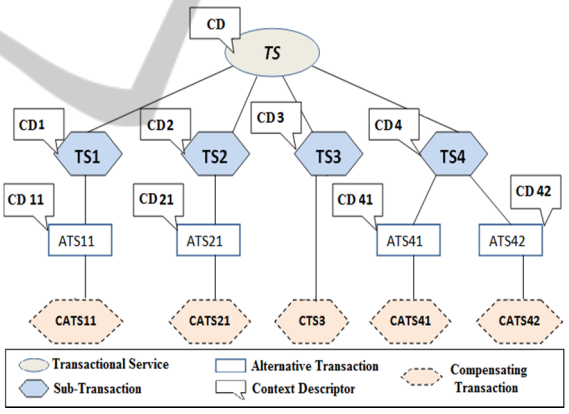

Figure 1: Structure of a context-aware transactional service.

The execution mode of the global transaction, which is a combination of a set of sub-transactions, is defined according to the context changes. In CATSM, we associate to each transaction a property type, namely, *replayable*, *replaceable*, *compensatable* and *critical*. These properties determine the behavioral profile of each transaction.

- *Replaceable transaction*: 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;
- *Replayable transaction*: A transaction is said to be replayable if it can be retried one or

more times after its failure. For example, the operation of sending the reservation document can be rerun;

- *Compensatable transaction*: A transaction is defined as compensatable if it provides mechanisms to undo its effects. The system must allow canceling the payment operation in case of abort of the overall transaction;
- *Critical transaction*: A transaction is said to be critical if it requires the abort of the global transaction after its failure. For example, the flight booking is a critical transaction.

The commit of the global transaction is associated with one of the following three types of atomicity, depending on the application semantic and its requirements in terms of transactional properties. The degree of atomicity can be adjusted dynamically during the transaction execution through an appropriate commit protocol.

- *Strict Atomicity* requires that all sub-transactions vote to commit before the validation of the global transaction. This must consist only of critical and non-compensatable sub-transactions. It is the classic atomicity property that requires that all sub-transactions are committed or none is;
- *Semantic Atomicity* requires that the global transaction consists of critical sub-transactions, some of which can be compensatable. Compensatable sub-transactions can be committed before the completion of the global transaction. In case the latter is aborted, compensating transactions must be executed to semantically undo the effects of transactions that have been unilaterally committed;
- *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.

5 ADAPTATION POLICY FOR CATS

Context-aware systems are generally associated with the specification of rules that define the required behavior of a system in response to a context state. Indeed, in our approach we specify rules that clearly separate the control of adaptation from adaptation mechanisms. Figure 2 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) (Kouadri et al., 2006), (Mostefaoui et al., 2007). CWSDL was developed to improve the WSDL standard of the W3C and provide a platform for retrieving context information. The context information is collected from the runtime environment (users, networks, resources, others services, etc.);
- *Decision phase*: Based on the context information gathered in the first phase and the rules specification, 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 rule is chosen, a reconfiguration mechanism will automatically be responsible for the modifications.

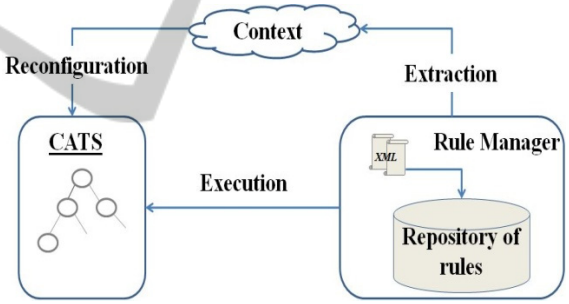

Figure 2: Adaptation approach by rules specification.

The adaptation policies are described in Extensible Markup Language (XML) files. The use of XML provides a hierarchical representation of the data. These structures can be easily checked, browsed and distributed. In addition, the language level of formalism remains affordable by users. Our rules specification describes (See Figure 3):

- Transactional properties and the desired degree of atomicity;
- The structure of CATSM: namely, the sub-transactions and their types, their compensating transactions, their alternatives and the associated context descriptors.

The adaptation rules are deducted from CATS rules specification file. The adaptation policy is based on the ECA strategy (Event, Condition, Action). The events mainly concern the context descriptors. We

```
<?xml version="1.0" encoding="UTF-8"?>
<Rules>
<rule id="Rule-ID">
 <name>property name</name>
 <degree>degree</degree>
 <TransactionalServiceList>
  <TransactionalService id="ID-of-TransactionalService1">
        <ContextDescriptor>
           <ConnectionState>state</ConnectionState>
            <Bandwidth>bandwidth</Bandwidth>
              ....
        </ContextDescriptor>
         <Properties>
             <critical>yes/no</critical>
             <replayable>yes/no</replayable>
             <compensatable>yes/no</compensatable>
             <Replaceable>yes/no</Replaceable>
         </Properties>
            <replay> number of attempts</replay>
     <replaceBy>
      <AlternativeList>
     <Alternative id="ID-of-alternative1">
          <ContextDescriptor>
              <ConnectionState>state</ConnectionState>
               <Bandwidth>bandwidth</Bandwidth>
                ....
          </ContextDescriptor>
       <Compensating id="ID-of-compensating-Alternative">
              </Compensating>
      </Alternative>
          <Alternative id="ID-of-alternative2">
           <ContextDescriptor>
              <ConnectionState>state</ConnectionState>
               <Bandwidth>bandwidth</Bandwidth>
                ....
           </ContextDescriptor>
      <Compensating id="ID-of-compensating-Alternative">
              </Compensating>
          </Alternative>
      </AlternativeList>
     </replaceBy>
      <Compensating id="ID-of-compensating-Transaction">
       </Compensating>
    </TransactionalService>
  <TransactionalService id="ID-of-TransactionalService2">
      <ContextDescriptor> ...</ContextDescriptor>
      ....
  </TransactionalService>
 </TransactionalServiceList>
</rule>
   ....
</Rules>
```

Figure 3: CATS rules specification file.

add constraints by identifying different conditions of the event. The conditions refer to the degree of context descriptor changes and to the behavioral profile of each transaction. Actions are specified according to different conditions in the event.

In case the current context corresponds to the transaction context descriptor:

- If the transaction is replayable then it is replayed according to the number of attempts (in case of failure);
- If the transaction is critical and replayable then it is replayed according to the number of attempts and re-execution parameters are updated (in case of failure).

In case the current context doesn't correspond to the transaction context descriptor:

- If the transaction is replaceable and if current context corresponds to the alternative context descriptor then the associated alternative is executed;
- If the transaction is critical then the main transaction is aborted (in case of failure). In this situation, the committed transactions, if any, will be compensated;
- If the transaction is neither critical nor replaceable then the transaction is just ignored.

6 ILLUSTRATIVE SCENARIO: THE E-TOURISM SYSTEM

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. He may also want to indicate potential hotels and restaurants. Reservations are not of equal importance, and the restaurant table reservation can for example be omitted from the transaction. The intermediate results of some sub-transactions can be validated and made visible once committed. In case of the abort of the overall transaction, the partial validated results require the execution of compensating transactions. The following figure illustrates this example:

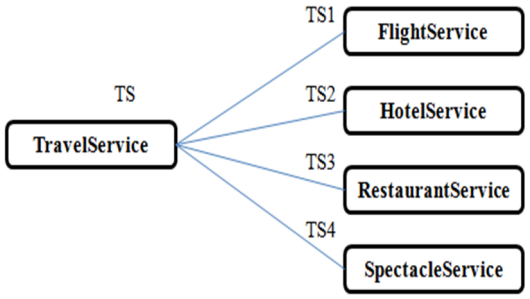

Figure 4: A travel planning service.

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. In this case, context-awareness presupposes that the system must detect in advance any changes and automatically adjust to the environment taking into account contextual information such as location, user preferences and other environmental parameters. Thus, it is desirable in the transaction management, that a transaction can respond to contextual information and adapt its behavior to the context changes.

As shown in Figure 5, the flight reservation is a critical transaction. In other words, this sub-transaction is a crucial task for the travel planning application. Let's imagine that a change in the context occurs during the flight reservation transaction due to a disconnection event. According to CATS specification (cf. Figure 5), flight reservation is a replaceable transaction that may be replaced by an alternative transaction which corresponds to the required execution context. In this case, the adaptation strategy is to execute *FlightServiceAgency2* on local device and synchronize once reconnected (i.e., adaptation action) whenever *replaceable* = yes (i.e., profile = {replaceable}), *ConnectionState* = disconnected, *memory* is available to store data and *bandwidth* is medium (i.e., adaptation condition).

```
<?xml version="1.0" encoding="UTF-8"?>
<Rules>
<rule id="Rule-ID">
 <name>Atomicity</name>
 <degree>Relaxed</degree>
 <TransactionalServiceList>
  <TransactionalService id="FlightService">
       <ContextDescriptor>
           <ConnectionState>connected</ConnectionState>
           <Bandwidth>high</Bandwidth>
           ....
       </ContextDescriptor>
        <Properties>
            <critical>yes</critical>
            <replayable>no</replayable>
            <compensatable>yes</compensatable>
            <Replaceable>yes</Replaceable>
        </Properties>
           <replay>1</replay>
           <replaceBy>
 <AlternativeList>
   <Alternative id="FlightServiceAgency2">
    <ContextDescriptor>
        <ConnectionState>disconnected</ConnectionState>
        <Bandwidth>medium</Bandwidth>
        <available-memory>available</available-memory>
       </ContextDescriptor>
      <Compensating id="canceling-FlightServiceAgency-2">
       </Compensating>
   </Alternative>
   <Alternative id="FlightServiceAgency3">
      <ContextDescriptor>
      ....
      </ContextDescriptor>
      <Compensating id="canceling-FlightServiceAgency-3">
      </Compensating>
   </Alternative>
 </AlternativeList>
           </replaceBy>
       <Compensating id="canceling-FlightService">
       </Compensating>
 </TransactionalService>
    <TransactionalService id="HotelService">
       <ContextDescriptor> ...</ContextDescriptor>
       ....
    </TransactionalService>
 </TransactionalServiceList>
</rule>
  ....
</Rules>
```

Figure 5: Example of CATS rules specification.

7 ADAPTATION MECHANISM

This section addresses the adaptation mechanism. Let's mention that we don't deal in this paper with the architecture of the global context-aware system. Figure 6 illustrates the different modules that are involved in the execution of a transaction in the CATSM.

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

Rule Manager: is responsible for inspecting the adaptation rules specification (XML file) and

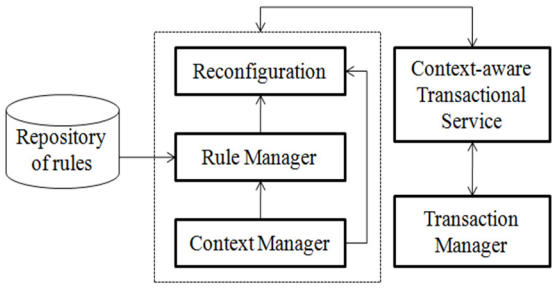

Figure 6: Adaptation mechanism architecture.

converting the adaptation rules into a data format that will be used in the reconfiguration module.

Reconfiguration module: is responsible for evaluating and interpreting the rule based on the context state information provided by the context manager and the inspecting result of the rule manager to trigger the execution of the appropriate adaptation. The rule 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 rule, the module decides which strategy will be triggered (e.g., identify the alternative sub-transaction to run). The reconfiguration module invokes the transaction manager which is responsible for operating the real adaptation (i.e., performing alternative sub-transactions, updating the re-execution parameters).

Transaction manager: Once the CATSM structure is identified (i.e., the alternative sub-transactions and compensating transactions) by the reconfiguration manager, the main coordinator TSC handles the processing and the execution of the global transaction TS, then it submits the sub-transactions to the sub-coordinators TSCi (See Figure 7), which are associated with the different TSi services including flight reservation service, hotel, restaurant and art show booking services. Each TSCi is running its TSi service and exchanges messages with the main coordinator.

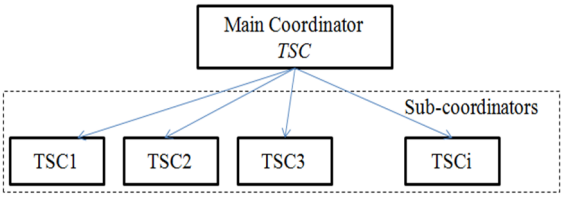

Figure 7: Main coordinator and sub-coordinators.

The following diagram sketches the adaptation mechanism of the proposed protocol.

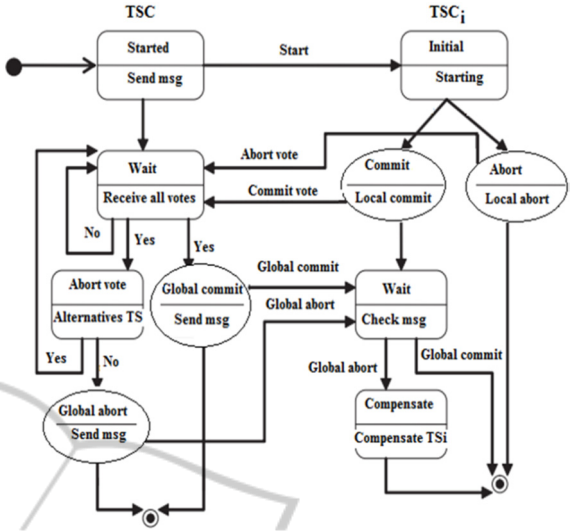

Figure 8: State diagram of the proposed protocol.

The TSC stores information about the overall transaction in log files and sends a "Start" message to TSCi to initiate the sub-transactions TSi. The TSC goes to "wait" state while waiting for TSCi messages.

TSCi records information about TSi transaction in the log file. The TSi execution result (commit or abort) is recorded in the log file (Local commit or Local abort) and sent to TSC to make the decision.

TSC receives the votes of all TSCi. If all TSCi vote to commit, the TSC forces the commit of the global transaction (Global commit) and sends the message to the sub-coordinators.

In case one of the sub-transactions is not committed, if it is replaceable, the corresponding TSCi initiates the alternative transaction and sends the result to the TSC. If it is not replaceable but critical, the overall transaction will be aborted (Global abort).

If TSCi receives an abort message from TSC for the global transaction, compensating transactions will be executed. All execution results are recorded in the log files.

8 CONCLUSIONS

Context-awareness is a common challenge in service-based software engineering. Several researches have been devoted to the design of self-adaptive applications; however, there has been little work on the adaptability of non-functional services.

In this paper, we propose a solution to the problems of transactional applications flexibility in

order to allow adaptation to different types of transactional execution models according to the environment characteristics which are described in context descriptors and the application semantics. This adaptation is based on the specification of rules that provide the ability to replay, the choice of alternative transactions and compensation actions depending on the context. For this, we propose a new model for context-aware transactional services. This model allows the specification of the transactional service and is the basis for all techniques that will be developed. The proposed approach is based on the requirements specification in terms of transactional properties which specify on one hand, the desired degree of atomicity, and allows on the other hand, the choice of an adaptation policy based on the alternative mechanism.

In the short term, we intend to design a self-adaptable transactional service. To model the transactional service, we will use a meta-model with a high level of abstraction to control the definition of all architecture components. In the long term, our objective is to propose a framework for the development of CATS based essentially on models transformation.

REFERENCES

Choi, S., Kim, H., Jang, H., Kim, J., Kim, S. M., Song, J., Lee, Y., 2008. *A framework for ensuring consistency of Web Services Transactions*. Information and Software Technology Journal, Vol.50, pp. 684-696.

Chrysanthis, P. K., Ramamrithan, K., 1994. *Synthesis of extended transaction models using ACTA*. ACM Transactions on Database Systems (TODS). Vol.19, Issue 3, pp. 450-491.

Dey, A., Abowd, G., Salber, D., 2001. *Conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*. Human-Computer Interaction, Vol.16, Issue 2, pp. 97-166.

El Haddad, J., Manouvrier, M., Ramirez, G., Rukoz, M., 2008. *QoS-driven selection of web services for transactional composition*. In ICWS'08, IEEE International Conference on Web Services, pp. 653-660.

Elmagarmid, A. K., 1992. *Transaction Models for Advanced Database Applications*. Morgan-Kaufmann, ISBN 1-55860-214-3.

Gray, J., Reuter, A., 1993. *Transaction Processing: Concepts and Techniques*. Series in Data Management Systems, Morgan Kaufmann. ISBN 1-55860-190-2.

Hafiddi, H., Baidouri, H., Nassar, M., ElAsri, B., Kriouile, A., 2011. *A Context-aware Service Centric Approach for Service Oriented Architectures.* In ICEIS'11, 13th International Conference on Enterprise Information Systems, Vol. 3, pp. 8-11, Beijing, China.

IBM, Microsoft, BEA., 2005. *Web Services Transactions Specifications*. Retrieved from http://www-128.ibm.com/developerworks/library/specification/ws-tx/

Kouadri, S. K., Maamar, Z., Narendra, N. C., 2006. *Mobile middleware for context-aware service composition*. Chapter In the Mobile Middleware Book, Boca Raton.

Lakhal, N., Kobayashi, T., Toyota, H., 2009. *FENECIA: failure endurable nested-transaction based execution of composite web services with incorporated state analysis*. VLDB Journal.Vol.1, pp. 1-56.

Moss, J. E. B., Hosking, A. L., 2006. *Nested Transactional Memory: Model and Architecture Sketches*. In Science of Computer Programming. Vol. 632, pp. 186-201.

Mostefaoui, S. K., Younas, M., 2007. *Context-oriented and transaction-based service provisioning*. International Journal of Web and Grid Services, Vol. 3, Issue 2, pp.194–218.

Muralidharan, K., Karthik, G. V., Gupta, P., Chowdhury, A.R., 2008. *mConnect: A context aware mobile transaction middleware*. In COMSWARE'08, 3rd International Conference on Communication Systems Software and Middleware and Workshops, Bangalore, India, pp.381-386.

OASIS Committee specifications, 2002. *Business Transaction Protocol,* version 1.0.

Philip, A., Bernstein, E. N.,1997. *Principles of Transaction Processing*, Morgan Kaufmann, ISBN 1-55860-415-4.

Rouvoy, R., Alvarado, P., Merle, P., 2006. *Towards Context aware transaction services.* In DAIS'06*,* International Conference on Distributed Applications and Interoperable Systems, pp. 272-288.

Schafer, M., Dolog, P., Nejdl, W., 2008. *An environment for flexible advanced compensations of web service transactions*. ACM Transactions on the Web (TWEB), Vol. 2, Issue 2.

Schilit, B., Theimer, M., 1994. *Disseminating Active Map Information to Mobile Hosts*. IEEE Network, Vol. 8, Issue 5, pp. 22-32.

Segun, K., Hurson, A. R., Spink, A., 2001. *A transaction processing model for the mobile data access system*. Proceedings of the 6th International Conference PaCT Novosibirsk, Russia, Vol. 2127, pp.112-127.

Serrano-Alvarado, P., Roncancio, C., Adiba, M., Labbé, C., 2004. *Context aware mobile transactions*, Proceedings of the IEEE International Conference on Mobile Data Management, ISBN 0-7695-2070-7.

Strunk, A., Reichert, S., Schill, A., 2009. *An infrastructure for supporting rebinding in BPEL processes*. In EDOCW'09, 13th IEEE Enterprise Distributed Object Computing Conference Workshops, pp. 230-237.

Tang, F., Guo, M., Li, M., 2008. *An adaptative context aware transaction model for mobile and ubiquitous computing*. Computing and Informatics, Vol. 27, Issue 5, pp. 785-798.

Younas, M., Chao, K., Lo, C., Li, Y., 2006. *An Efficient Transaction Commit Protocol for Composite web*

services. In AINA'06, 20th International Conference on Advanced Information Networking and Applications. Vol. 1, pp.591-596.

Younas, M., Mostefaoui, S. K., 2010. *Context-aware mobile services transactions*. In AINA'10, 24th IEEE international conference on advanced information networking and applications, Perth, Australia, pp. 705-712.