Design Patterns for Building Context-Aware Transactional Services in PaaS-Enabled Systems

Ettazi Widad [1] , Riane Driss [2] and Nassar Mahmoud [3]

IMS Team, ADMIR Laboratory, ENSIAS, Mohammed V University, Rabat, Morocco

Summary
Pervasive computing is characterized by a key characteristic that affects the operating environment of services and users. It places more emphasis on dynamic environments where available resources continuously vary without prior knowledge of their availability, while in static environments the services provided to users are determined in advance. At the same time, Cloud computing paradigm introduced flexibility of use according to the user's profile and needs. In this paper, we aimed to provide Context-Aware Transactional Service applications with solutions so that it can be integrated and invoked like any service in the digital ecosystem. Being able to compose is not enough, each service and application must be able to offer a well-defined behavior. This behavior must be controlled to meet the dynamicity and adaptability necessary for the new user's requirements. The motivation in this paper is to offer design patterns that will provide a maximum of automatism in order to guarantee short reaction times and minimal human intervention. Our proposal includes a cloud service model by developing a PaaS service that allows CATS adaptation. A new specification for the validation of CATS model has been also introduced using the ACTA formalism.

Key words:
Context-awareness, Pervasive Computing, PaaS, Transactional Service, ACTA, Cloud Service Model.

1. Introduction

With the proliferation of pervasive computing and the availability of a wide range of services, access to information to transact online is anywhere, anytime. In such a flexible, dynamic but less reliable environment, transactional techniques and mechanisms are expected to provide reliability of service and consistency of data. Indeed, in so-called pervasive environments where communicating objects automatically recognize each other without any particular action by the user, the major challenge is to capture and model the intention of use and to resolve its ambiguities. On the one hand, the constraints of the execution environment affect the execution of transactional services resulting in cancellations and unforeseen execution costs. On the other hand, the approaches proposed for service adaptation to the context are essentially based on the creation of personalized services by the specific development the context-awareness code.

Our approach aims to exploit the advantages of different existing approaches. Hence, we focused on the need to design a self-adapting transactional service. We intend to propose solutions to the flexibility problems of transactional systems in order to guarantee adaptation to the varied and variable requirements of applications in terms of transactional properties and to the constraining and variable characteristics of pervasive environments.

To illustrate these challenges, we present the following scenario: "A tourist intends to use an online travel arrangement service to plan a sightseeing visit. An application of such an example can be illustrated by a travel planning scenario. This involves purchasing a flight ticket with a preference on specific conditions (e.g., company, flight schedule, price), booking a hotel room (e.g., proximity to shopping centers, price) and a table in a restaurant (e.g., close to the hotel, culinary specialty). This application can be structured into transactional services, one of which is associated with flight reservation, the second with hotel room reservation, and the third with a restaurant table reservation. Each of the services can use alternatives; for example, the choice between several companies and prices. Flight and hotel room are vital reservations for the user. The commit of the hotel room reservation service can be done before the conclusion of other reservations. Due to compensation, it will be possible to cancel the reservation if at the end the overall composition is not committed. The user can be also granted with the option to change the vitality of services by reserving a hotel room in advance and purchasing a flight ticket regardless of the completion of the restaurant table reservation". To meet these needs, we based our approach on the as-a-service paradigm [1-3]. Cloud computing has offered a new ecosystem where everything is offered as a service, accessible and connectable anywhere and anytime. Software architects are now gradually migrating to service-centric architectures. Applications are now constructed as compositions of micro-services [4-6] integrating more and more features.

In this paper, we proposed a Cloud service model that allows the adaptation of CATS services. This model is built on a PaaS service which introduced (i) design patterns for the construction of a reliable context-aware

Manuscript received July 5, 2023
Manuscript revised July 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.7.11

transactional service self-adaptive system (ii) a formal specification of CATS services based on axiomatic definition in ACTA formalism.

This article is organized as follows. The section II will be devoted to review some basic concepts. Section III gives an overview of related works. In section IV, we present the formal specification of CATS model. Section V details the proposed Cloud service model and exhibits design patterns for CATS execution and adaptation. The section VI presents experimental scenarios demonstrating our proposition. Finally, we conclude in the section VII.

2. Basic Concepts

2.1 Context-Awareness

According to [7], "a system is context-aware if it uses context to provide information and services relevant to the user, where relevance depends on the task requested by the user". Other works like [8] have found that definitions of context-aware systems do not encompass the description of the context management process. Thus, the authors state that "the context-awareness of a software system is its ability to acquire, manage, interpret and respond to context changes in order to provide the appropriate services". For [9], "a system is context-aware if it can automatically change its services or trigger a service as a response to the change in the value of information or a set of information that characterize the service".

2.2 Cloud Computing

The proliferation of so-called context-aware mobile applications has enabled the emergence of a rapidly evolving field known as mobile cloud computing [10]. The use of the cloud computing paradigm by current mobile applications has made it possible to harness its computing power, memory and storage resources to overcome the resource limitations of mobile devices [11]. The scalability, flexibility, and stability offered by cloud services make it an ideal architecture for use in client applications in a resource-constrained mobile environment. Commercialized cloud applications use Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS) models where software and resources are hosted in the cloud instead of the customer [12].

2.2 Platform-as-a-Service

Along with PaaS is a development platform that supports the lifecycle management of the software. The capacity provided to the consumer is to deploy on the Cloud infrastructure applications created or acquired for a client using programming languages, libraries, services and tools supported by the vendor. The idea behind PaaS is to provide the developer with a platform that contains all the systems and environments necessary for the development, testing, deployment and hosting of complex applications delivered as a service [13]. Hence, the difference between SaaS and PaaS is that SaaS includes completed applications while PaaS offers a platform where hosted applications under development [14]. The advantages of such service are that the developer can take advantage of the underlying infrastructure to develop large applications and accelerate their execution. Reference [13] explained that PaaS can significantly reduce development time and offers also hundreds of easily accessible services.

3. Related Work

Several research works have studied service adaptation in the Cloud. Authors in [15] have developed a framework to support context-aware services in the Cloud. The framework enables context acquisition and adaptation, reconfiguration of candidate services, and execution of reconfigured services. Another concept is proposed in [16] based on virtual services for service composition in the Cloud. Virtualization is used for a successful execution of the composition. Authors in [17] proposed a flexible service model for seamless service integration in order to reduce the costs of providing services in cloud-based environments and provide maximum satisfaction of services suppliers and consumers. Another study presented an abstract model for the federation of services according to their semantics and their QoS parameters [18]. Despite the multitude of researches in the field of context-aware services in the cloud, existing cloud platforms do not provide the necessary services to support the adaptation of transactional services. In this situation, PaaS brings interesting possibilities, especially in situations where several developers collaborate on a project or when other external actors need to interact with the development process. However, this service model is not recommended for the development of applications which require reconfiguration and modification of the underlying infrastructure. In addition, the PaaS is completely dependent of the supplier in terms of availability, platform utilization, interfaces and tools. Reference [19] explained that Paas can be inadequate when it comes to applications requiring high portability, or if the proprietary language offered by the vendor can impact the development process. Indeed, the supplier can offer services, which can be interfaces or languages that he has developed, which can affect the migration of applications from one Cloud to another generating the phenomenon of data lock-in. To overcome this problem, efforts have been made to develop Open Platform as a Service (OPaas). OPaas is a step in

PaaS evolution whose purpose is to provide open access to application programming interfaces (APIs) and standards.

To address the aforementioned issues, several services (Carriots [20], Xively [4], ThingSpeak [21]) has been proposed in the Cloud. These services typically provide an API and various sample applications to use data collected from open platforms (e.g., Arduino). Service platforms play a fundamental role in building and managing context-aware applications. It is crucial to mask the heterogeneity of hardware, software, data formats, technologies and communications that characterize these applications. Some platforms focus on developing architectures that provide vertical interoperability between applications and different technologies. For example, the main objective of Icore [22] and COMPOSE [23] is to develop an open network architecture based on object virtualization which encompasses the heterogeneity of the technologies employed. BlueMix [4] is a PaaS developed by IBM. It enables rapid development of analytical applications, visualization dashboards and mobile IoT applications. AWS IoT [24] is a platform that enables users to connect devices to AWS Services [5] and other devices, to secure data and interactions, process and act on data device, and allow applications to interact with devices even if they are offline. The SPRINT project [25] provides a platform to connect software tools used by industrial companies within the project and allows the integration of different subsystems at the design level. Other platforms like BUTLER [26] or MobilityFirst [27] aim to develop open architectures providing a secure location and context-aware services.

4. Formal Specification of CATSM

4.1 ACTA Adaptation

In this section, we propose to formalize CATS model (CATSM) using the ACTA formalism [28, 29].

ACTA is a framework developed to specify transaction models and define their interactions. This formalism allows defining (i) the effects of transactions on other transactions and (ii) the effects of transactions on objects. The logic of ACTA model is based on the concept of "transaction" by specifying the relationships between transactions and the transactions effects on objects. Hence, the specification of CATSM by ACTA formalism requires transforming this model into a representation based on transactions. By analogy, a context-aware transactional service can be assimilated to a transaction composed of several sub-transactions. Sub-transactions are the activities performed by participating services; their properties are therefore similar to the transactional properties of these services. According to the CATSM, a context-aware transactional service is an adaptable transaction composed of sub-transactions. A sub-transaction can be decomposed into any level of nesting, resulting in a root transaction and a set of component sub-transactions. These transactions are called main transactions. Secondary transactions are either compensating or alternative transactions. In the following, we describe the dependencies between sub-transactions and the root transaction and between main and secondary sub-transactions regarding the context-awareness characteristics.

4.1 Axiomatic Definition

In order to develop the axiomatic definition of CATSM, we used the following notations:

TS is a context-aware transactional service. It is then considered as an adaptable transaction of the CATSM model:

$$TS = \{ TS_1 (p_1, p_2, p_3, p_4), \dots , TS_n (p_1, p_2, p_3, p_4) \}, n > 0 \quad (1)$$

$$AltTS_i = \{ AltTS_{i1} (p_1, p_2, p_3, p_4), AltTS_{i2} (p_1, p_2, p_3, p_4), \dots , AltTS_{is} (p_1, p_2, p_3, p_4) \}, s \geq 0 \quad (2)$$

$$CompTrs \cap \neg Comp\ Trs = \emptyset \quad (3)$$

$$p_1 \in \{Cp, NCp\} \quad (4)$$

$$p_2 \in \{Rc, NRc\} \quad (5)$$

$$p_3 \in \{Rp, NRp\} \quad (6)$$

$$p_4 \in \{Ct, NCt\} \quad (7)$$

$$P = (p_1, p_2, p_3, p_4) \quad (8)$$

TS_i (p_1, p_2, p_3, p_4) in (1) denotes a main sub-transaction TS_i and its behavioral profile P. $AltTS_i$ in (2) is the set of alternative sub-transactions of TS_i. Equation (3) denotes $CompTrs$ the set of compensable sub-transactions, and $\neg CompTrs$ the set of non-compensable sub-transactions. P in (8) specifies the behavioral profile of each sub-transaction as follows: p_1 in (4) denotes the compensable (Cp) and non-compensable (NCp) sub-transaction, p_2 in (5) denotes the replaceable (Rc) and non-replaceable (NRc) sub-transaction, p_3 in (6) denotes the replayable (Rp) and non-replayable (NRp) sub-transaction, and p_4 in (7) denotes the critical (Ct) and non-critical (NCt) sub-transaction.

$CompTS_i$ is the compensating transaction of TS_i. TS_p denotes a parent transaction.

We present the axiomatic definition of the CATSM model as follows:

$$ES_{TS} = ES_{ts} = \{begin, prepare, commit, abort\} \quad (9)$$

$$EI_{TS} = EI_{ts} = \{begin\} \quad (10)$$

$ET_{TS} = ET_{ts} = \{commit, abort\}$ (11)

$(begin_{ts} \in H)) \Rightarrow AdaptationCondition \wedge (ts\ BD\ TS)$ (12)

$AltTS_{ik}\ (p_1, NRc, p_3, p_4) \wedge (AdaptationCondition)\ EBD\ TS_{im}\ (p_1, NRc, p_3, p_4)$: $begin_{AltTSik} \in H) \Rightarrow (ContextCondition) \wedge (\vdash (begin_{AltTSim}))$ (13)

$TS_p\ AD\ TS_i\ (p_1, p_2, p_3, Ct)$: $(abort_{TSi} \in H) \Rightarrow (abort_{TSp} \in H)$ (14)

$TS_i\ (p_1, p_2, Rp, p_4)\ BAD\ TS_i\ (p_1, p_2, Rp, p_4)$: $(begin_{TSi} \in H) \Rightarrow (abort_{TSi} \rightarrow begin_{TSi})$ (15)

$TS_p\ CD\ TS_i\ (Cp, p_2, p_3, Ct)\ \&\ TS_p\ CPD\ TS_j\ (NCp, p_2, p_3, Ct)\ \&\ TS_p\ SCD\ TS_k\ (NCp, p_2, p_3, Ct)$: $(commit_{TSp} \in H) \Rightarrow ((commit_{TSi} \in H) \Rightarrow (commit_{TSi} \rightarrow commit_{TSp})) \wedge (prepare_{TSj} \rightarrow commit_{TSp}) \wedge ((commit_{TSk} \in H) \Rightarrow (commit_{TSp}))$ (16)

$TS_i\ (NCp, p_2, p_3, Ct)\ CD\ TS_j\ (Cp, p_2, p_3, Ct)\ \&\ TS_i\ CPD\ TS_k\ (NCp, p_2, p_3, Ct)$: $(commit_{TSi} \in H) \Rightarrow ((commit_{TSj} \in H) \Rightarrow (commit_{TSj} \rightarrow commit_{TSi})) \wedge (prepare_{TSk} \rightarrow commit_{TSi})$ (17)

$TS_i\ (NCp, p_2, p_3, Ct)\ SCD\ TS_p$: $(commit_{TSp} \in H) \Rightarrow (commit_{TSj})$ (18)

$TS_i\ (NCp, p_2, p_3, NCt)\ BCD\ TS_p$: $(commit_{TSi} \in H) \Rightarrow ((commit_{TSp} \in H) \Rightarrow (commit_{TSp} \rightarrow commit_{TSi}))$ (19)

Axioms (9), (10) and (11) indicate the significant events of CATSM model. As decrypted in axiom (12), the initialization of a sub-transaction is conditioned by the satisfaction of context conditions and the beginning of the root transaction. Conditions on the context are fulfilled when the context current state matches the required values in the environment descriptor. The initialization of an alternative transaction in (13) is conditioned by the satisfaction of context conditions and only an alternative transaction should be initiated. Context conditions are satisfied when the context current state matches the required values in the environment descriptor of the alternative transaction. Axiom (14) describes that if a critical sub-transaction is aborted, then the parent transaction must be cancelled. In axiom (15), a sub-transaction can only be re-executed if the previous attempt has been canceled. Axiom (16) indicates that the commit of the parent transaction depends on the commit of all compensable sub-transactions, and the preparation of all critical and non-compensable sub-transactions. In addition, the commit of the critical non-compensable transaction involves the commit of the parent transaction. The axiom presented in (17) inflicts on the critical non-compensable sub-transaction to commit only after committing all critical and compensable sub-transactions, and preparing all critical non-compensable sub-transactions. Axiom (18) stipulates that the commit of the parent transaction involves the commit of all critical and non-compensable sub-transactions. Axiom (19) implies the initialization of non-critical and non-compensable sub-transactions after the commit of their parent transaction.

To develop our CATSM model formalism, we used the following dependencies:

- *Commit Dependency:* If the transactions t_i and t_j commit, the commit of t_i must precede that of t_j (t_j CD t_i):

  $(commit_{tj} \in H) \Rightarrow ((commit_{ti} \in H) \Rightarrow (commit_{ti} \rightarrow commit_{tj}))$

- *Strong-Commit Dependency:* If t_i commits, then t_j must also commit (t_j SCD t_i):

  $(commit_{tj} \in H) \Rightarrow (commit_{ti} \in H)$

- *Abort Dependency:* If t_i aborts, then t_j must also abort (t_j AD t_i):

  $(abort_{tj} \in H) \Rightarrow (abort_{ti} \in H)$

- *Begin Dependency:* t_j cannot begin if t_i has not begun yet (t_j BD t_i):

  $(begin_{tj} \in H) \Rightarrow (begin_{ti} \rightarrow begin_{tj})$

- *Begin-on-Commit Dependency:* t_j cannot be started until t_i commits: (t_j BCD t_i):

  $(begin_{tj} \in H) \Rightarrow (commit_{ti} \rightarrow begin_{tj})$

- *Begin-on-Abort Dependency:* t_j cannot be started until t_i aborts: (t_j BAD t_i):

  $(begin_{tj} \in H) \Rightarrow (abort_{ti} \rightarrow begin_{tj})$

The dependencies in the ACTA formalism are not sufficient to specify our CATSM model. Thus, in addition to the dependencies defined by the ACTA formalism that is extensible, we have defined the following ones:

- *Commit-on-Prepare Dependency:* t_j cannot commit until t_i prepares (t_j CPD t_i):

  $(commit_{tj} \in H) \Rightarrow (prepare_{ti} \rightarrow commit_{tj})$

- *Exclusive Begin Dependency:* t_i can only be started if no t_j has not begun yet: (t_j EBD t_i):

  $(begin_{ti} \in H) \Rightarrow (\vdash (begin_{tj} \in H))$

5. Cloud Service Model for CATS Adaptation

5.1 The proposed Model

In this section, we present a cloud service model for context-aware transactional services adaptation by introducing a PaaS service. This service encapsulates a set of modules for the execution of user requests and allows

the adaptation of transactional services according to the context. Fig. 1 describes the different layers of the cloud service model.

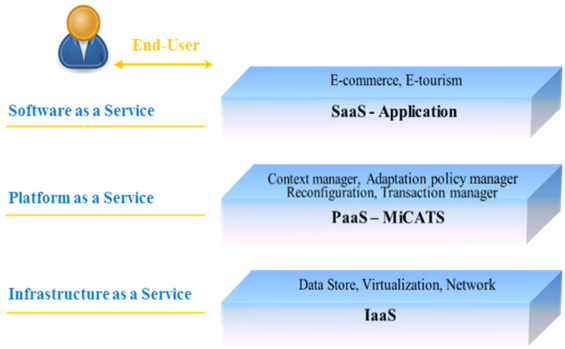

Fig. 1 Cloud Service Model for CATS Adaptation.

We present a global vision of our CATS service adaptation platform as illustrated in Fig. 2. It consists of three main layers: the service layer, the communication layer and MiCATS middleware. As described in Fig. 2, MiCATS is built around three APIs: CATS service composition API, CATS service adaptation API and CATS service execution API. We introduce each layer described in the cloud service architecture as follows:

- *Services layer:*

The evolution of classical information systems into pervasive information systems leads us to no longer consider the latter as a set of logical services. Indeed, with the emergence of context-aware computing, its technologies are increasingly integrated into the physical environment, thus offering innovative services to users who are constantly evolving in this environment. Unlike traditional information systems, context-aware information systems can provide both logical and physical services. In a context-aware information system, users operate in a service space that offers a set of heterogeneous services whose objective is to meet the needs of these users. This space allows the representation of knowledge about users and their environment, in order to select the most appropriate service that can meet the requirements of a given user in a particular context.

- *Communication layer:*

The communication layer aims to connect the services layer with MiCATS middleware for context adaptation of transactional services. User requests, context capture and service invocation are sent through this layer.

- *MiCATS: Middleware for context-aware transactional service:*

MiCATS is the core of our CATS service adaptation platform. Its main objective is to meet user requirements by generating the most appropriate transactional services composition regarding a given context. More emphasis on MiCATS middleware is provided in a previous work [30].

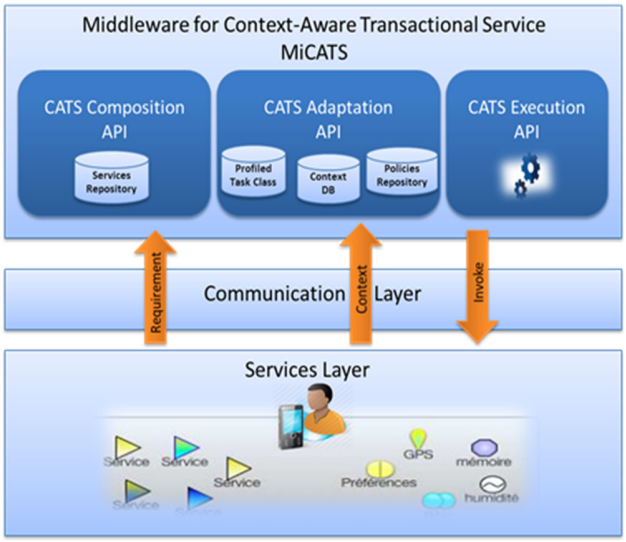

Fig. 2 CATS service adaptation platform.

5.2 Design Patterns for CATS adaptation

CATS service design approach is used to guide designers in order to specify the expected functionalities of their system, as well as the information that will be captured for better adaptation. This is to keep control over the definition of the system and its services, while allowing for a highly dynamic environment. In this section, we will present an overview of the design process and we will detail each of its steps.

The proposed design patterns decouple as much as possible context-awareness design from business aspects design. Therefore, we add to the business design two other dedicated areas: context-awareness and adaptation policy management, inducing two new roles which are the context-awareness designer and the adaptation policy designer. We followed SPEM model (Software process Engineering MetaModel) [OMG, 2008] to define the roles, activities, and produced results in the area of context-awareness. As shown in Fig. 3, the first activity is the specification of environment descriptors taking into account the business logic, the state of runtime infrastructure parameters and the environment in which the user and services evolve. For instance, in the case of flight reservation, the “Weather” parameter is an example of an entity to be observed. The second activity concerns the design of the weights attributed to context dimensions. These weights will be specified according to CATS services application domain and the environmental characteristics of the user. In the case of restaurant

reservation with specific conditions (e.g., restaurant with a terrace), the weight assigned to the context dimension "Temperature", for example, will be of vital importance.

Fig. 4 describes the main activities of the adaptation policy designer. The first activity concerns the specification of the costs associated with the various context parameters.

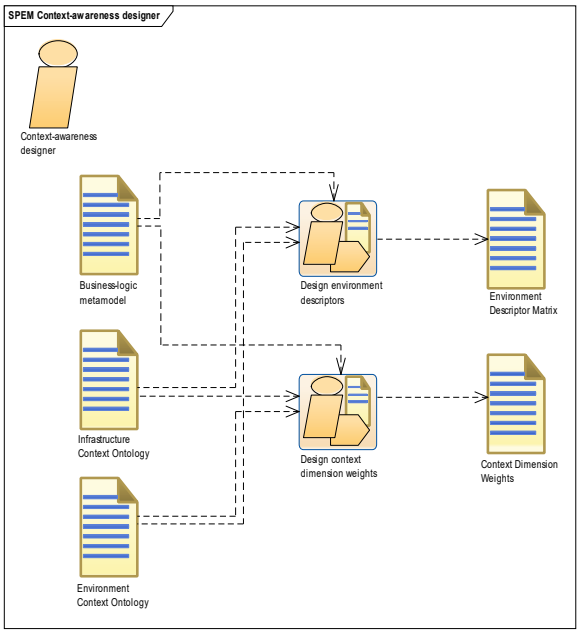

Fig. 3 Activities of Context-awareness designer.

The Infrastructure Context ontology and the Environment Descriptor Matrix are the inputs of this activity. The second activity consists of establishing a link between the artifacts produced by the context-awareness design process and business logic artifacts through the definition of alternatives. These alternatives are configured taking into account the execution costs defined under the first activity. For example, in the case of an environment that is subject to frequent disconnections, alternatives will be configured based on cost optimization in terms of CPU consumption.

6. Illustrative scenario

In this section, we illustrate a travel arranging system through a CATS service which allows organizing tours in the Moroccan touristic city Dakhla. As shown in Table 1, this service offers the possibility of booking a flight or buying a train ticket, booking a hotel room and a table in a restaurant, contacting a tourist guide, and signing up for surfing sessions.

Fig. 5 shows a succinct prototype for the flight reservation activity "FlightReservation" associated with the transactional service "Reserve Flight". The "Arranging tours" CATS service follows the RACID execution contract (i.e., AtomicityDegree = {Relaxed Atomicity}) which offers more flexibility in the execution of its primitive activities.

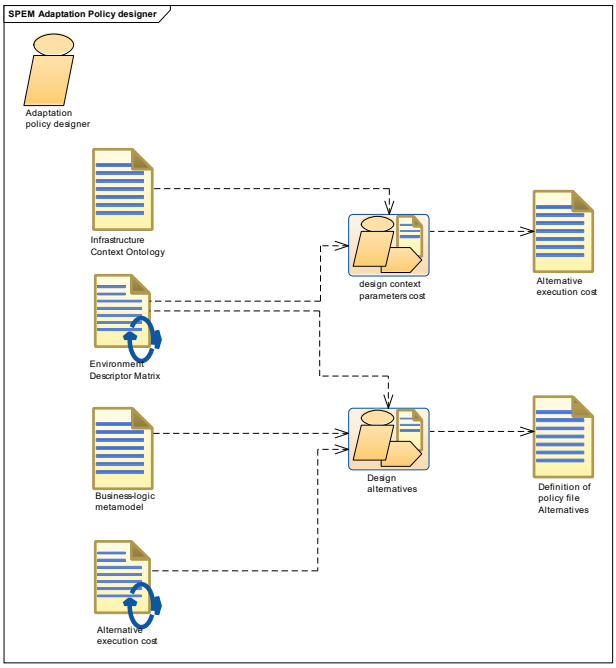

Fig. 4 Activities of Adaptation Policy designer.

Let's consider that a change in the "Weather" context parameter state occurs. According to CATSM specification, the adaptation policy performs a first verification of the condition on the behavioral profile. In our case, the "FlightReservation" activity has an alternative (i.e., p_3 = {Rp}). The second verification consists in checking the condition on the "Weather" parameter (i.e., FlightEnvironmentDesriptor). The adaptation policy triggers service adaptation by executing "Reserve Train" service each time the activity is replaceable and the "Weather" parameter changes from "nice" state to "very_bad" state (i.e., AdaptationCondition). As shown in Fig. 5, "FlightReservation" activity is critical (i.e., $P_F = \{p_1, p_2, Rp, Ct\}$). In other words, it represents a crucial task for the CATS service "Arraging Tours". Imagine that a context change occurs while running the "Reserve Flight" service related to a disconnection event. According to CATSM specification, "FlightReservation" is a replaceable activity. In this case, the adaptation strategy consists of running "Reserve Flight" service on the local device each time the activity is replaceable, the

connection state is disconnected, memory is available to store data, and bandwidth is medium.

Table 1: CATS service "Arranging Tours" description

Arranging Tours Service		
Reserve Flight	Input	UserCredentials, UserProfile, FlightReservationI (cityFrom, cityTo, nbPassengers, departureDate, arrivalDate), CR, TR
	Output	FlightReservationO (FlightDetail)
	Contextual requirements (CR)	Period, AirlineCompany, FlightClass, Price
	Transactional requirements (TR)	Replaceable, Critical, Compensable
Reserve Train	Input	UserCredentials, UserProfile, TrainReservationI (cityFrom, cityTo, nbPassengers, departureDate, arrivalDate), CR, TR
	Output	TrainReservationO (TrainDetail)
	Contextual requirements (CR)	Period, CabinClass
	Transactional requirements (TR)	Replayable, Compensable
Book Hotel	Input	UserCredentials, UserProfile, HotelReservationI (city, nbAdults, roomType, departureDate, arrivalDate), CR, TR
	Output	HotelReservationO (HotelDetail)
	Contextual requirements (CR)	HotelClass, Proximity, BoardBasis
	Transactional requirements (TR)	Replaceable, Critical, non-Compensable
Reserve Restaurant	Input	UserCredentials, UserProfile, RestaurantReservationI (city, nbAdults, roomType, departureDate, arrivalDate), CR, TR
	Output	RestaurantReservationO (RestaurantDetail)
	Contextual requirements (CR)	CulinarSpeciality
	Transactional requirements (TR)	Replaceable, non-Compensable
Book Surf sessions	Input	UserCredentials, UserProfile, ActivityReservationI (city, nbAdults, nbSessions, period), CR, TR
	Output	ActivityReservationO (ActivityDetail)
	Contextual requirements (CR)	AbilityLevel, Age
	Transactional requirements (TR)	Replayable, non-Compensable

```
<policy:Policy rdf: ID="ArrangingTours_Service">
<policy:textDescription xml:lang="en">
  This file describes the adaptation policy for ArrangingTours service which provides a facility
  visits including all services for flight, hotel, restaurant, guide contact and activity plannir
 </policy:textDescription>
<Rules> <rule id="Rule_ID">
  <name> Atomicity </name>
   <degree> Relaxed </degree>
   <TransactionalServiceList>
    <TransactionalService id="FlightReservation">
     <ContextDescriptor>
       <connection_mode> connected </connection_mode>
       <bandwidth> high </bandwidth>
       <weather> nice </weather>
       <communication_cost> expensive </communication_cost>
   </ContextDescriptor>
   <Properties>
       <critical> yes </critical>
       <replayable> no </replayable>
       <compensable> yes </compensable>
       <replaceable> yes </replaceable>
   </Properties>
   <replaceBy>
  <AlternativeList>
    <Alternative id="TrainReservation">
     <ContextDescriptor>
       <weather> very_bad </weather>
    </ContextDescriptor>
    <compensation id="cancelTrainReservation"> </Compensation>
    </Alternative>
    <Alternative id="FlightReservation_companyA">
     <ContextDescriptor>
        <communication_cost> moderate </communication_cost>
        <weather> nice </weather>
    </ContextDescriptor>
    <compensation id="cancelFlightReservation_companyA"> </Compensation>
     </Alternative>
    <Alternative id="FlightReservationLocal">
     <ContextDescriptor>
        <connection_mode> disconnected </connection_mode>
       <bandwidth> low </bandwidth>
       <weather> nice </weather>
       <communication_cost> expensive </communication_cost>
       <memory> available </memory>
    </ContextDescriptor>
    <compensation id="cancelFlightReservationLocal"> </Compensation>
     </Alternative>
    </AlternativeList>
    </replaceBy>
    <compensation id="cancelFlightReservation"> </Compensation>
    </TransactionalService>
    <TransactionalService id="HotelReservation"> .....
```

Fig. 4 “Arranging Tours” service CATSM specification.

7. Conclusion

The positioning of context-aware transactional services in current B2B systems has been felt in recent years. In this paper, we have presented (i) a model for context adaptation of transactional services. This model supports dynamic changes of transactional needs and can adapt to the volatility of services by introducing the concept of alternatives. CATSM model allows transactional services to be classified according to their level of atomicity into four classes: transactional services with strict atomicity, semantic atomicity, semi-atomicity and relaxed atomicity. In addition, the specification of CATSM model by the ACTA formalism has enabled to define the properties of this model and to specify the relationships between its different structures, (ii) a reliable architecture for the support of transactional services in

context-aware systems; which constitutes a framework for the adaptation of this class of services. We have also described the various functionalities supported by the proposed platform. The underlying adaptation architecture is based on a cloud service model.

It is important to point out that our work proposes a set of reflections constituting the first step towards the implementation of a robust OPaaS platform. The main research perspectives that appear at the end of this proposal relate to the research questions that have remained open and that should be explored in order to complete this work: (i) the PoC on real industrial scenarios for final validation through the evaluation and testing of the platform in environments with thousands of instances, (ii) the development of a DSL-based approach for "CATS" services modeling.

References

[1] The OpenCloudware project. The opencloudware project, 2015. Available: http://www.opencloudware.org/

[2] T. Aubonnet, N. Simoni. *Service creation and self-management mechanisms for mobile cloud computing*. In Wired/Wireless Internet Communication - 11th International Conference, WWIC, St. Petersburg, Russia. pp. 43-55, 2013. doi: 10.1007/978-3-642-38401-1_4.

[3] T. Aubonnet, L. Henrio, S. Kessal, O. Kulankhina, F.Lemoine, E. Madelaine, C. Ruz, N. Simoni. *Management of service composition based on self-controlled components*. Journal of Internet Services and Applications, 6(15):17, 2015. doi : 10.1186/s13174-015-0031-7.

[4] IBM Bluemix. IBM Bluemix, 2018. Available: https://www.ibm.com/cloud-computing/ bluemix

[5] Amazon Web Services. Amazon web services, 2018. Available: https://aws.amazon.com

[6] Microsoft Azure. Microsoft azure, 2018. Available: https://azure.microsoft.com/fr-fr/

[7] A. K. Dey, G.D. Abowd, Towards a Better Understanding of Context and Context-Awareness, CHI 2000, Workshop on the What, Who, Where, When, and How of Context-Awareness, The Hague, The Netherlands, 2000.

[8] W. Xiaohang, T. Gu, D. Zhang, J. Dong, H. K. Pung. *Ontology Based Context Modeling and Reasoning using OWL*. 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom'04), March 14, 2004, Orlando, Florida.

[9] M. Miraoui, C. Tadj, C.B. Amar, *Context Modeling and Context-Aware Service Adaptation for Pervasive Computing Systems*, International Journal of Computer and Information Science and Engineering, vol 2, N 3, pp. 148-157, 2008.

[10] N. Fernando, S. W. Loke, W. Rahayu, *Mobile cloud computing: A survey*, in Future Generation Computer Systems Vol. 29 Issue 1, pp. 84-106, January 2013.

[11] R. Kaur, A. Kaur, *A Review Paper on Evolution of Cloud Computing, its Approaches and Comparison with Grid Computing*, in (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 Issue 5, pp. 6060-6063, 2014.

[12] R. Buyya, Y. Chee Shin, S. Venugopal, *Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities*, in 10th IEEE International Conference on High Performance Computing and Communications, pp.5–13, 25–27 September 2008 Dalian, China.

[13] B. P. Rimal, A. Jukan, D. Katsaros, Y. Goeleven. *Architectural Requirements for Cloud Computing Systems : An Enterprise Cloud Approach*. Journal of Grid Computing, vol. 9, no. 1, pp. 3-26, 2010.

[14] T. Dillon, C.Wu, E. Chang. *Cloud Computing : Issues and Challenges*. 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 27–33, 2010.

[15] H. J. La, S. D. Kim, *A conceptual framework for provisioning context-aware mobile cloud services,* in Proceedings of IEEE International Conference on Cloud Computing, pp. 466-473, 2010.

[16] J. Fu, H. W. Tu, M. Biao, J. Baldwin, F. B. Bastani, *Virtual services in cloud computing*, in Proceedings of the 6th World Congress on Services, pp. 467-472, 2010.

[17] Y. Zhu, R. Y. Shtykh, Q. Jin, *A human-centric framework for context-aware flowable services in cloud computing environments*, in Information Sciences, Vol.257, pp. 231-247, 2012.

[18] H. Ma, K. Schewe, Q. Wang, *An abstract model for service provision, search and composition*, in Proceedings of IEEE AsiaPacific Services Computing Conference (APSCC 2009), pp. 95-102, 2009.

[19] B. Kepes. Understanding the Cloud Computing Stack : SaaS, PaaS, IaaS. White paper, 2013.

[20] Azurewatch. Azurewatch, 2018. Available: http://www.cloudmonix.com/aw/

[21] H. A. Soulimani, P. Coude, N. Simoni. *User-centric and qos-based service session.* In IEEE Asia-Pacific Services Computing Conference, APSCC 2011, Jeju, Korea (South), December 12-15, 2011, pp. 267–274, 2011. doi : 10.1109/APSCC.2011.64.

[22] Iot-Icore. Icore-Internet Connected Objects for Reconfigurable Ecosystem, FP7-ICT 287708, 2014. Available: https://cordis.europa.eu/project/rcn/100873_fr.html

[23] Compose. COMPOSE-Collaborative Open Market to Place Objects at your SErvice, FP7-ICT 317862, 2012. Available: http://www.compose-project.eu/

[24] AWS IoT. AWS IoT, 2018. Available: https://aws.amazon.com/iot/

[25] Sprint. SPRINT -Software Platform for Integration of Engineering and Things, FP7-ICT 257909, 2010. Available: http://www.sprint-iot.eu/

[26] BUTLER-uBiquitous, secUre inTernet-of-things with Location and contExt-awaReness - TRIMIS - European Commission, July 2016. Available:https://trimis.ec.europa.eu/project/ubiquitous-secure-internet-things-location-and-contextawareness

[27] D. Raychaudhuri, K. Nagaraja, A. Venkataramani. *MobilityFirst : A Robust and Trustworthy Mobility-centric Architecture for the Future Internet*. SIGMOBILE Mob. Comput. Commun. Rev., 16(3), pp. 2-13, December 2012. ISSN1559-1662. doi : 10.1145/2412096.2412098.

[28] P.K. Chrysanthis, K. Ramamritham, *ACTA: A Framework for Specifying and Reasoning about Transaction Structure and Behavior*, ACM SIGMOD, 1990.

[29] P.K. Chrysanthis, K. Ramamritham, *Synthesis of extended transaction models using ACTA*, ACM Transactions on Database Systems (TODS), vol. 19, Issue 3, September 1994, pp. 450- 491.

[30] W. Ettazi, H. Hafiddi, M. Nassar. *CATS-CAE Reflective Middleware Framework for Adapting Context-Aware Transactional Services: Using a Hybrid Policy-Based Approach*, International Journal of Web Services Research (IJWSR), Volume 17, Issue 2, 2020.