Research Article

Architectural QoS pattern to guarantee the expected quality of services at runtime for context-aware adaptation application

Konan-Marcelin Kouame[1] · Hamid Mcheick[1]

Abstract
Since the year 2000, the use of mobile applications in general and, more specifically, context-aware applications, becomes very popular in the contemporary and even future services. Indeed, in the field of online sales, we have Amazon application; in the banking field we have e-banking applications; in the field of health we have intelligent medical applications, HealthCare application; in the field of public services, we have e-government applications. The fundamental characteristic of these applications is their ability to adapt or to react according to the information of the context. Quality of Services (QoS) and violation of Service Level Agreement (SLA) during adaptation activity are still a challenge that many researches try to address. Before addressing these problems of violation of SLA, it is important to present a concrete research work on implementing adaptation methods to better understand the adaptation activity concept. This concrete work can help companies to address adaptation issues, and better understand integration of OSGi adaptation technics on android system. In addition, the definition of an SLA between application providers and consumers to improve the QoS for application context-aware, is not obvious. This paper proposes a mechanism for addressing these issues, using existing adaptation works for the android platform, an architecture QoS pattern to preserve the quality service levels of the context-aware applications, design pattern observer and aspect oriented programming to realize service adjustment. This QoS pattern can help to guarantee the expected quality of services at runtime for contextual applications. More details of Design QoS Pattern for context-aware application will be given in the future work.

Keywords Context-aware · Adaptation · QoC · QoS patterns · SLA · SLO

1 Introduction

Since the year 2000, the use of mobile applications in general and, more specifically, context-aware applications, is important in ubiquitous environment. Context-aware applications have the capacity to change services or behaviors depending on the information's context, which is collected by sensors. Adaptation activity can impact the services offered by application context-aware. Many researches present methods and platforms to realize adaptation. However, most studies on adaptation methods [5], after classification, do not provide concrete examples of their use. To remedy this problem, a presentation of works using adaptation techniques was carried out. These concrete examples can help the IT manager to use these methods in companies and help to better understand adaptation activity. These concrete works also describe the importance to know the integration of OSGi with android platforms to help android application, to better realize adaptation on runtime.

In addition, the dynamic adaptation of software involves several concepts: context modelling, data acquisition, data aggregation, context-aware fusion, inference, and service change. Errors can occur in context modelling, data acquisition, aggregation of data or inference. Masood argued in [12]: "Context-aware information

✉ Hamid Mcheick, Hamid_mcheick@uqac.ca; Konan-Marcelin Kouame, Konan-marcelin.kouame1@uqac.ca | [1]Department of Computer Sciences and Mathematics, University of Quebec at Chicoutimi, UQAC, Chicoutimi, Canada.

SN Applied Sciences (2019) 1:405 | https://doi.org/10.1007/s42452-019-0415-6

Received: 20 December 2018 / Accepted: 26 March 2019 / Published online: 4 April 2019

fusion is based on data received from a number of sources. (2) Some sensors may give false readings. (3) Some sensors may be offline due to low power levels."

These errors can impact the service offered by contextual applications in general and specifically mobile applications. For example, quality of context (QoC's) errors can have a negative impact of the adaptation engine of context aware applications.

ABID [3] focuses on proposing mechanisms to characterize the quality of the context information to ensure the relevance of the adaptation decisions. ABID concluded that dynamic adaptation impacts the QoS of these applications [2–4]. Therefore, the guarantee of the quality levels offered by the context-aware application is not obvious. Indeed, in [4], the author says: "the increasing pervasiveness of computing services in everyday life, combined with the dynamic nature of their execution contexts, constitutes a major challenge in guaranteeing the expected quality of such services at runtime." Many researches have focused on the quality of context (QoC) and QoS. They conclude that quality of context (QoC) has an impact on context-aware application adaptation decisions [2, 3].

According to ABID [3], QoS is: "the ability of a service to meet certain requirements for different aspects of the service such as performance, availability, reliability or cost. In order to assess quality of service qualitatively and quantitatively, several measures are considered like the rejection rate, the average time between failures, response time, throughput, financial costs or energy consumption." Mobile application providers like Android, Amazon gives information about the quality of service offered by their platform. Customers consume these services. If we want to design an SLA between application context-aware provider that dynamically adapts their services depending on the contexts and the customers who consume the services offered, what are the relevant metrics on which both parties can engage?

This paper presents our architecture proposal, the specific metric proposal for application context-aware and a classification of concrete adaptation work research.

The rest of the paper is structured as follows. Section 2 describes the Android and OSGi platforms. Section 3 analyses and describes the QoS concepts and the concepts of context aware applications. Section 4 analyzes SLA templates and metrics. Section 5 illustrates the details of the contributions of this paper. Section 6 describes the problem. Section 7 gives an overview of the related works. Section 8 describes the details of our approach for designing a QoS. Section 9 describes the adjustment of the algorithm. Section 10 gives details of QoS formalism Patter. Section 11 gives conclusion and future works.

2 Integration of OSGI and android platforms: concrete adaptation works

A. Integration OSGi with android platforms

Many researches are conducted in the field of adapting mobile applications according to their context in android platform. In [4], Bohez mentions: "although based on the Java programming language, Android does not offer the same benefits as a dynamic software module system such as OSGi. Nevertheless, OSGi would have many advantages on mobile environments." To realize adaptation with android platform software, researchers focus on an integration platform between OSGi and android. We have identified in the literature three important research works on adaptation android platform: *EZdroid, Androsgi and ProSyst's mBS*. We choose Android because it is a popular mobile platform today. Thus, with the information from the literature and surveys, we propose criteria to classify these works. This classification can help IT managers to better address adaptation issues in companies.

B. Concrete adaptation work

Android Applications do not realize dynamic adaptation [4]. Therefore, to provide Android applications with OSGi's performance for dynamic adaptation, several research projects have been conducted. Among these works, we distinguish three main research projects, such as EZdroid, ProSyst's mBS and Androsgi.

(1) EZdroid

EZdroid is an integration platform of OSGi and Android. It uses Apache Felix OSGi and the Android platform. The integration approach is based on a duplication of OSGi components in android applications. However, this integration approach, between OSGi and the android platform, has limitations. Indeed, it is impossible to administer these components with the OSGi platform. In addition, duplicate bundles in android applications require to be recompiled. It is an open source platform [7].

(2) ProSyst's mBS

The ProSyst's mBS Mobile is an integration platform of Android 1.5 and OSGi framework. The strength of this integration approach is that a new type of bundle has been defined, making ProSyst bundles slightly different from traditional OSGi bundles [6]. This approach requires modification of the android application bundles before being used on the ProSyst's mBS platform. According to Adjaz [6], another drawback is needed for the installation of any

Android application, which does not check the availability of services and the dependence of that application [6].

(3) Androsgi

The Androsgi project is an integration platform of OSGi and Android. The quality of this platform is based on Eclipse IDE plugin. According to Bohez [4], this integration facilitates the execution of OSGi on top of Android. This Androsgi platform leverages OSGi's adaptation benefits. The integration approach is based on automating the process of converting OSGi's bundle formats (SDK) into android adk format. In addition, this solution combines the ADT plugin with Bndtools plugin [4].

C. Popular integration OSGi-android work

To understand and analyze the popular OSGi-Android Integration method for adapting applications, we realize a classification base on three criteria: C1-compatibility with eclipse java, C2-level of use and C3-limitations. The information collected made it possible to classify the work according to the three criteria. The results are shown in the Table 1.

The analysis of these results makes it possible to say that the Androsgi integration platform (OSGi and Android) is more successful than the other platforms. This is due to the existence of the Eclipse IDE Tools plugin, ADT plugin with Bndtools plugin [4]. For an IT manager who is interested in Androsgi, it is a concrete project to solve the problem of dynamic adaptation of android applications. After the presentation of integration and adaptation activity; it is important to better understand the concept of SLA Violation.

3 QoS concepts for application context-aware

Several concepts below help to understand the notion of quality of service for context-aware applications in literature.

(1) QoC

Table 1 Assessment of adaptation real work

Code	Concrete work	C1	C2	C3
01	EZdroid	+++	++	++
02	ProSyst's mBS	+++	++	++
03	AndrOSGi	+++	+++	+

In [2], Quality of Context is any inherent information that describes context information and can be used to determine the worth of the information for a specific application.

(2) QoS

In [11], Quality of Service (QoS) contracts have been proposed to specify expected quality levels (QoS levels) on different context conditions, with different enforcing mechanisms.

The QoS is a non-functional requirement measure of the overall performance of a service such as reliability, availability, security, affordability, etc. [9–11].

They are various metrics to measure the quality level of applications [11]: 1. performance metric, 2. availability metric and 3. metric reliability:

- Performance metrics: response time, which is the time taken by the application to deliver the result to the user.
- Availability metric: dropout rate, which is the ratio of service requests to the total number of requests, or utilization rate which is the ratio of the time of use of a service to the total time in an adaptation context of the application.
- Reliability metric: The average time between failures that is the expected time between failures inherent in the service, or the average time to recover, which is the average time that a service takes to recover from a failure.
- Cost metrics are the financial cost of using a service of an application that fits the context.

A QoS metric is therefore a way to quantify the level of service.

(2) SLA and SLOs

SLA is the service level agreement between the provider of the service and the service user. The customer may require a level of service to achieve a given objective, the service level objective. (SLO). For example, an SLO [11] can be defined as a quality of service metrics with a value greater than/less than a given threshold, maximize/minimize some quality of service measures, and so on. Therefore, a Service Level Agreement (SLA), is a set of SLOs to be compliant, negotiated between the service provider contextual application provider and the customer. In [9–11], an SLA is composed of several Service Level Objectives (SLOs).

(3) Abbreviation list

All the abbreviations of concepts are described in the Table 2.

4 SLA template for application context-aware

With the increasing popularity of mobile ubiquitous computing where the application context-aware are traded as services, efficient management of quality of service (QoS) has become increasingly significant to both service consumers and service providers. Appropriate Sla Template for context-aware applications must be proposed to prevent from the violation of SLA.

A. SLA template

In [13], Jemal indicates: '(SLA template) An SLA template is a partly completed SLA document and defines initial negotiable SLOs as well as constraints (posed on the negotiable SLOs, such as upper or lower bounds of acceptable values)'. In SLA documents or template, we can identify 3 standard types of metric group: Platform performance Metric, Hardware Performance Metric and Software Performance metric. The specific performance parameters witch can be appropriate for application context-aware are: the ratio of the time of use of a service to the total time in an adaptation context of the application; response time, which is the time taken by the application to deliver the result to the user and The average time between failures that is the expected time between failures inherent in the service.

5 Contributions

We argue that errors from quality of context (QoC) [2] that may occur at each stage of the adaptation process can impact the quality of service provided by context-aware applications. As an example of data acquisition errors, sensors can provide erroneous data. At the inference level, the algorithms, on which the calculations are based, can be erroneous. All these situations can cause SLA violations between context-aware application providers and customers. The SLA is a commitment. According to many research works, the following challenges [9–11] impact the QoS: (1) How can we specify SLA in general for different contexts of dynamic software adaptation? (2) How can we describe the terms of the service level agreement between a contextual application provider and a customer? What kind of service level objectives or penalties can be defined for SLA violations? (3) How can we guarantee quality of service offered by application context-aware? Our contribution is the architecture proposal, the definition of a specific metric proposal for application context-aware and the classification of concrete adaptation work research. This QoS pattern can help to guarantee the expected quality of services at runtime for contextual applications. In addition, we present concrete work based on adaptation methods and platforms in order to help companies addressing software adaptation challenges.

Table 2 Abbreviation list

No.	Description
IT	Information technology
QoC	Quality of context
SLA	Service level agreement
SLOs	Service level object
QoS	Quality of service
GoF	Gang of four
ADK	Android development tools
OSGi	Open service gateway initiative

6 Problem description

Context-aware applications adapt their services according to adaptation decisions. These adaptation decisions depend on the information of the context. In [3], ABI, a researcher states that context-aware applications and services, expect correct and reliable context data to better adapt their functionalities.

Context information show insufficiencies: sensor errors, incomplete, ambiguous and inconclusive data collected [2]. Thus, ABID. [2–4] states: "In order not to jeopardize the resulting decisions, the quality of the context information must be explicit and be subject to negotiated agreements between the providers and the context users. In the absence of such agreements, critical decisions such as dynamic adaptation of the application with the change of the user interface could be based on unreliable information and would no longer meet the needs of users." These solutions include the implementation of contracts [9] or SLA for QoC. In [2, 3], S. Machara focus on the adaptation problem.

Unfortunately, these solutions have not made it possible to fully resolve these context-quality problems. Machara said [2, 3]: "only Middleware explicitly deals with context quality but limited to location information." The immediate consequence is that adaptation decisions are not completely reliable. The quality levels of the services offered by the applications can be impacted by the quality of the adaptation decisions. It is therefore important to develop service contracts between service providers

and service consumers based on SLAs. In these SLAs, service levels specific to context-aware applications must be defined clearly, not only to improve the quality of services, but to satisfy the customer's requirements. In the absence of such agreements, service level violations due to the poor quality of adaptation decisions can have multiple consequences. For example, we consider the medical sector level, a patient fill the information to a subscription with an application service provider. The application, according to the information collected from the patient's context, must allow the medical system to alert his doctor depending patient's context metric. If the application receives a wrong heart rate, therefore, the change of service based on this rate is not correct. These situations cannot only deteriorate the quality of service, but worse, risk the life of the patient. The implementation of an SLA will help mitigate these situations. The set of solutions to solve these service problems is called quality of service model or QoS Pattern for adapting contextual applications. The concepts that can help to design this QoS pattern are presented in the section below. Overall, our contribution is summarized as follows: 1. Identification of a concrete work to better understand adaptation activity; 2. Identification of architecture proposal for the design QoS Pattern to preserve the quality levels of the context-aware applications.

7 Related work

Many previous software adaptation researchers have performed in specific areas like SOA (Service Oriented Architecture). However, to guarantee the quality of the services during the adaptation activity, a few progress researches have been done.

Tamura [9, 10] argued: "Besides, the dynamic nature of the wireless channel deteriorates or reduce the QoS." Deterioration or reduction of QoS is the same problem of violation SLA of application context-aware. However, the proposal is more specific to healthcare applications (Table 3).

Moreover, other research works focus on guaranteeing quality of service levels and propose a semantic approach. In [10, 11], the strength of this solution is the possibility to systematically exploit design patterns at runtime by dynamically deploying them in the managed software application. Semantic approach has insufficiencies such as [9, 10]:

- The complexity to specify reconfiguration rules in QoS requires adequate training,
- Design patterns at runtime can only help to develop behavior models to predict stability, but not guarantee quality levels during adaptation.

Finally, the SOA approach proposed in [11] is essential during the selection of services according to the decision adaptation. The improvement of this approach can help actors and enterprises to guarantee a good threshold of quality service level (Table 4).

8 Approach and concepts for QoS pattern design

(1) Concepts

In this section, we introduce some concept which can help design a QoS pattern. To better understand the concepts or components defined in the architecture, a table of functionalities is presented (Table 5). These concepts are: Context Information, Context Monitoring, Adaptation Engine, Adaptation Decision, Application Service, user/Profile/Preference SLA, SLA/SLOs, Admission Control Engine, Normal Service Delivery Engine Adjustment Service Expected delivery.

Table 3 Structure guide pattern

Purpose/intent	Applicability	Solution
1. Help to define concrete SLA for context-aware application between providers and consumers 2. Improve quality of service offered by applications	1. Existence application context-aware 2. Context to realize software adaptation project	Follow guide process

Table 4 Formalism of guide pattern

Name	Problem	Solution
QoS pattern	How to build a SLA between providers and consumers for application context-aware	Follow guide process

Table 5 Concepts of the architecture

Ref.	Component/concepts	Features
1	Context information	Collect information from the context. Example location
2	Context monitoring	Control and validate context information
3	Adaptation engine	Realize adaptation according to context information
4	Adaptation decision	Inference activity
5	Application	Behavior change activity of the application
6	Service	Features offered by application to users
6a	User/profile/preference SLA	User profile information examples the preference
6b	SLA/SLOs	Service Level for Sla (Contrate between application provider and user)
7,8	Admission control engine	Check if the service level is correct according to sla preference user
9a	Normal service delivery	If Service expected is correct, normal service is delivered
9b	Engine adjustement	Realize adjustment service at run time using AOP if SLA is not respected
9c	Service expected delivery	Service expected is delivered

(2) Architecture for QoS Pattern and approach

The set of concepts above and information of related works will allow to design the architecture of the future QoS pattern.

We consider that a context-aware application has different functionalities or services with different requirements in terms of response time, resulting in a different SLO for each type of response. In ubiquitous environment, they are clients/users with different priorities; it is possible to improve the fulfilment rate of the SLOs for the most important clients by controlling the admission of lower-priority clients. As mentioned by Entrialgoa in [8, 9] to automatically do this, an admission control subsystem has been inserted in the system. This admission [8] must know the SLOs, so that when a new session is initiated by a client, the subsystem can decide whether to accept it in order to improve the fulfilling rate of the SLOs of high-priority clients.

This rate can be calculated by metrics engines using the performance model and propose adjustment so that clients should not suffer the effects of erroneous adaptation decisions (Fig. 1).

Figure 2 below describes the different components of the architecture. This architecture explains the process to preserve the QoS level expected during adaptation activity.

(3) Architecture operating principle

The architecture is composed of 3 essential components: The Observer component, the Admission component, The Adjustment component. The input elements are the services and errors generated by the adaptation engine. The output items are normal services or adjusted services. Once an error is detected by the Observer component, the admission component evaluates the error. If the error can impact the SLA, then the adjustment component adjusts the code by the AOP approach.

9 Adjustment algorithm

(1) General information

To facilitate the simulation of the future QoS pattern, two scenarios with real data have been proposed. These scenarios have been described in Table 5. We present two scenarios:

Scenario 1: Context Information, engine adaptation or inference system is correct as a QoS. QoS Pattern did not adjust Service. Normal service is delivered.

Scenario 2: Incorrect context information or incorrect decision adaptation, therefore, the service level expected is not respected during user's session. There is a case of violation of SLA between the application provider and the application consumer. In this situation, the engine adjustment ensures the correction on runtime based on an AOP approach. This adjustment guarantees the service level expected by current users. Therefore, users do not suffer from the deterioration of QoS level.

(2) Adjustement process

The adjustment is made in the last step. Errors generated by adaptation decisions were saved into a file. The admission control module then evaluates these errors. Only software errors related to SLA metrics are processed. If the admission control module confirmed that the errors have an impact on SLA, the engine adjustment reacts. The injection code (AOP) is needed to prevent the SLA

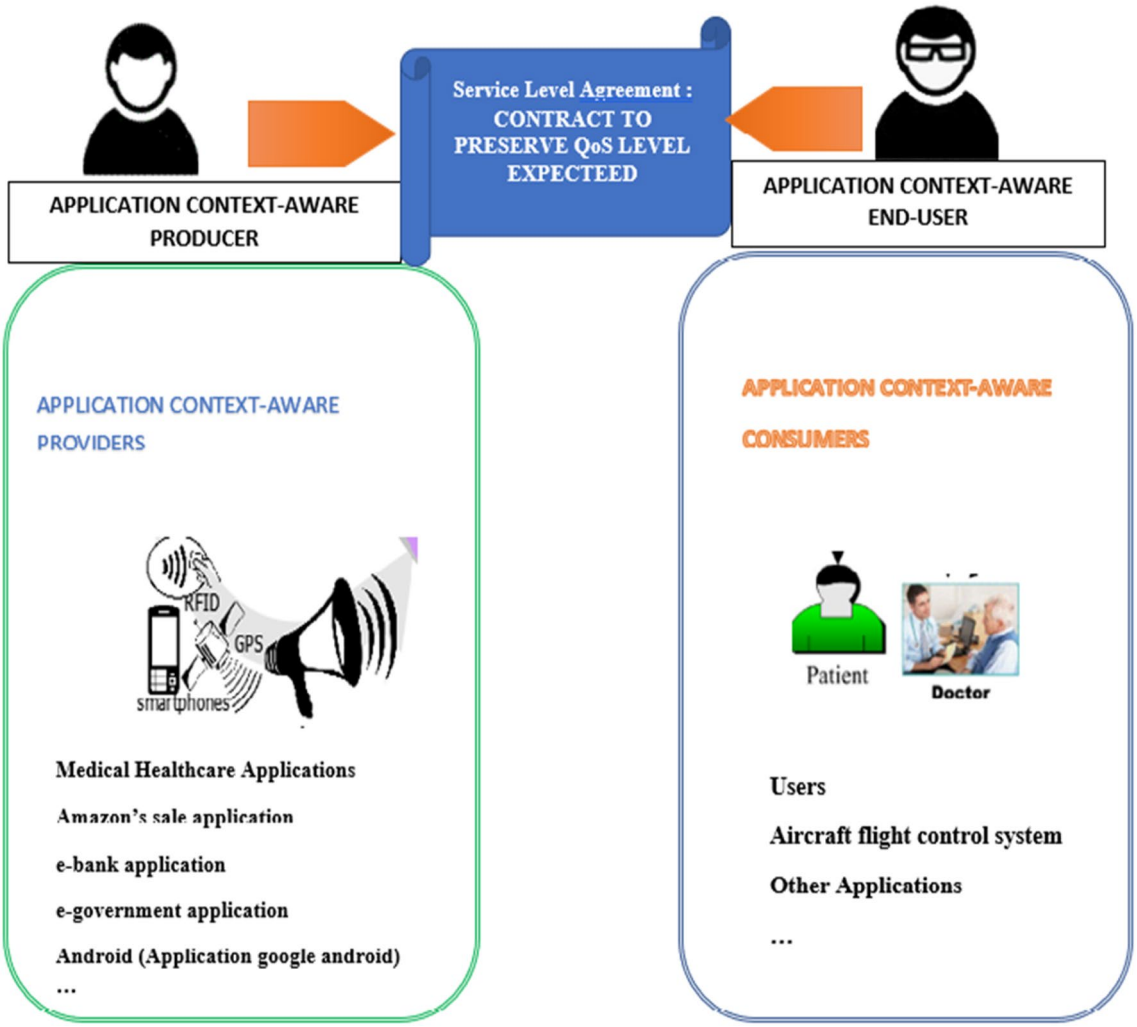

Fig. 1 Contract view between application context-aware providers and application context-aware consumers

violation. If the admission module evaluates that either errors do not have an impact of the SLA, then there is no adjustment. The normal service is delivered, and SLA is preserved (Table 6).

10 Formalism of QoS pattern

This section proposes a formalism to design QoS pattern to guarantee the expected quality of services at runtime during adaptation of Context-aware applications and to solve the problem of violation of service level agreement (SLA) context-aware for applications.

GoF design models present flexible solutions to common software development issues. In particular, these patterns manage changes in interfaces and behaviors of an application. Each pattern model is structured as follows: purpose/intent, applicability, solution structure and implementation examples [1]. The QoS pattern has the same structure. In general, a pattern does not give details of software implement. However, our QoS pattern provides instructions to solve a problem of SLA for context-aware applications. We describe the process and concepts that can help to design this QoS pattern.

*Advantages of QoS pattern: help to define concrete SLA for context-aware applications between providers and consumers; improve quality service offered by applications; prevent violation SLA.

In addition, the description of a guide pattern follows a fixed formalism [1]: Name; Description of the problem to be solved;

Description of the solution is the elements of the solution, with their relations. The solution is called guide pattern.

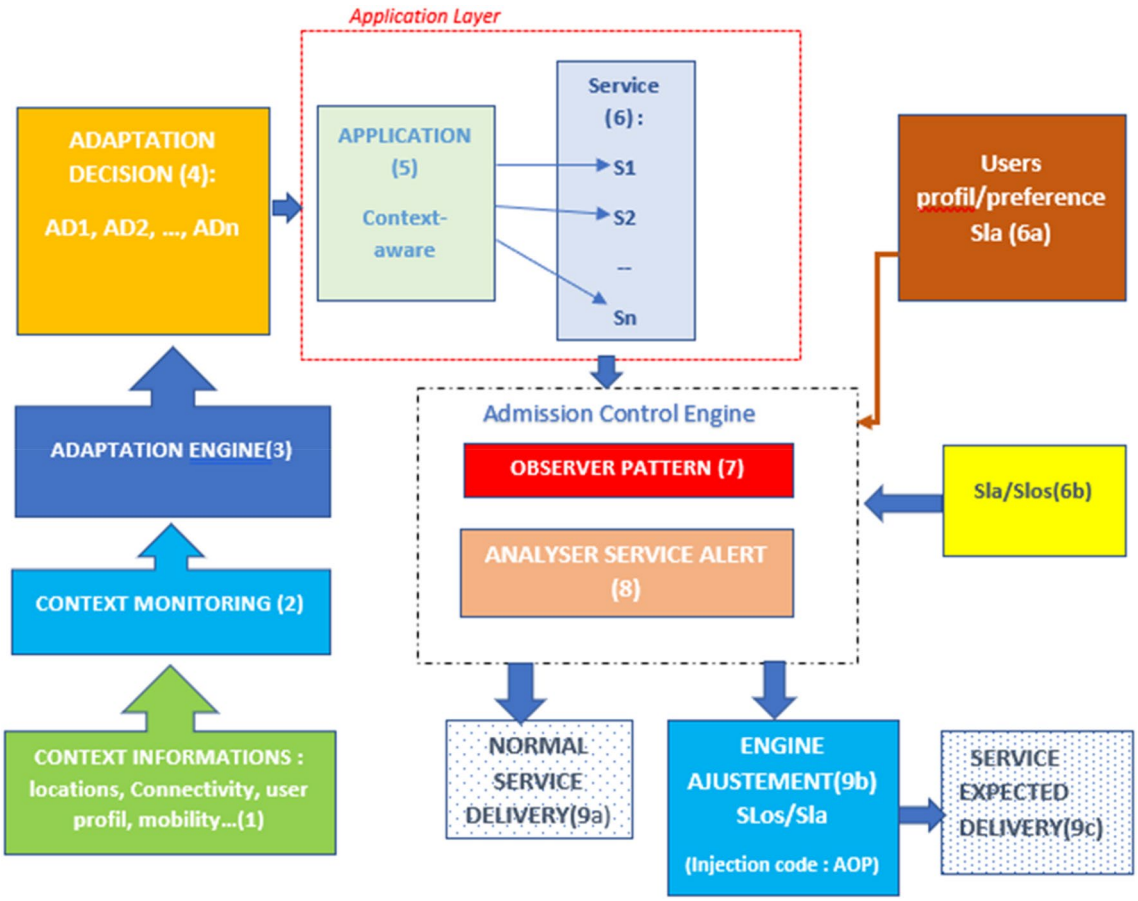

Fig. 2 An architecture for design QoS pattern for SLA preservation

11 Conclusion and future work

A first part of this article presents a classification and analyze s dynamic adaptation works of contextual applications. In particular, we analyse the dynamic adaptation of mobile applications using Android and OSGi platforms. The second aspect of the article highlights the challenges, and approaches to solving QoS problems related to the dynamic adaptation of applications.

Our contribution made possible to propose an architecture to solve the problem of violation SLA between application providers and consumers, during adaptation activities. In addition, this architecture can help to design a QoS Pattern for applications to solve a recurrent problem of SLA. We also describe and analyze two approaches in software industry to design the architecture for QoS pattern: 1. Aspect Oriented Programming (AOP), and 2. Design pattern Observer. AOP helps to inject code in order to adjust expected service. In particular, design pattern observers allow detecting eventual errors in the service proposed by inference system. This QoS pattern can also help to guarantee the expected quality of services at runtime for contextual applications. The design and validation QoS pattern will be our future research works.

Table 6 Adjustement algorithm

Context information/ Variable context/user profil	Condition	Adaptation	Adjustement
Variables context Heart rate: H1; Body Température: T1; Location: L1=Longitude, l1=latitude Provider Sla(p)/Slop(p) application provider ***Consumer/user/customer*** Sla(c)/Slos(c) user ***Features/services offered*** S1: Alert medical system S2: Call nearest Ambulance SN: ----------	**Sceneraio 1: Context Information Correct** (H1⩽70 BPM) AND (T1⩾ 40 °C) AND (0⩽L1 102.2) (AND 0⩽11⩽50.0.1 Sol(p)=Slo(c)	**Scenario 1:** IF ADAPTATION DECISIN IS AD1, CALL S1 ELSIF AD2 CALL S2 ELSE …….	**Scenario 1:** Normal Service Delivered with Slo(p)=Slo(c) 9a
	Scenario 2: Error Context information Or error inference or engine adaptation (HI⩽50 BPM) AND (TI⩾30 °C) AND (0⩽L1⩽102.0.2) AND 0⩽l1⩽50.01 Slo(p)!=Slo(c)	**Scenario 2:** Adaptation decision is AD3 and the decision expected is AD2, not AD3 S2 is expected Not S3	**Scenario 2:** ENGINE ADJUSTEMENT DETECTS(9b) Realize adjustment service at runtime using AOP(9b) Service expected is delivered

Acknowledgements This work was sponsored and funded by Natural Sciences and Engineering Research Council of Canada (NSERC), and computer science and mathematics department of the University of Quebec at Chicoutimi (Quebec), Canada.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Gamma E, Johnson R, Vlissides J, Helm R (1995) Design patterns: elements of reusable object-oriented. Addison-Wesley, Boston
2. Marquez SM (2015) Models and algorithms for managing quality of context and respect for privacy in the Internet of Things. Software Engineering [cs.SE]. Université Paris-Saclay
3. Abid Z (2012) Management of the quality of context for ambient intelligence. Doctorat en co-accreditation Telecom SudParis et Universite d'Evry-Val-d'Essonne, Paris
4. Bohez S, De Cominck E, Verlen T, Dhoedt B (2015) Androsgi: bringing the power of OSGi to android. ACM, Pittsburgh, pp 1–10
5. Kouamé KM, Mcheick H (2018) Overview of techniques of software adaptations. Université Québec à Chicoutimi(UQAC), Chicoutimi, pp 141–148
6. Aghiles A, Bouzefrane S, Huang D, Paradinas P (2011) An OSGi-based service oriented architecture for android software development platforms. Arizona University
7. EZdroit project (2011). http://www.ezdroid.com/
8. Entrialgo J, García DF, García J, García M, Valledor P, Obaidat MS (2011) Dynamic adaptation of response-time models for QoS management in autonomic systems. J Syst Softw 84(5):810–820
9. Tamura G, Casallas R, Cleve A, Duchien L (2014) QoS contract preservation through dynamic reconfiguration: a formal semantics approach. Sci Comput Progr 94(3):307–332
10. Sodhro AH, Luo Z, Sangaiah AK, Baike SW (2019) Mobile edge computing based QoS optimization in medical healthcare applications. Int J Inf Manag 45:308–318
11. Tripathy AK, Tripathy PK (2018) Fuzzy QoS requirement-aware dynamic service discovery and adaptation. Appl Soft Comput 68(2018):136–146
12. Khattak AM, Akbar N, Aazam M, Ali T, Khan AM, Jeon S, Hwang M, Lee S (2014) Context representation and fusion: advancements and opportunities. Sensors 14(6):9628–9668
13. Abawajy J, Fudzee MF, Hassan MM, Alrubaian M (2015) Service level agreement management framework for utility-oriented computing. J Supercomput 71(11):4287–4303