Preprint submitted to the International Journal of Modeling, Simulation, and Scientific Computing
You can find the published version at http://www.worldscientific.com/doi/abs/10.1142/S179396231850023X , 2018

A Framework for Ranking Ubiquitous Computing Services by AHP Analysis

Shaghayegh Izadpanah, Hamed Vahdat-Nejad, Hamid Saadatfar

PerLab, Faculty of electrical and computer engineering, University of Birjand, Iran

{sh.izadpanah, vahdatnejad, saadatfar}@birjand.ac.ir

Abstract- With the increasing advances in technology, ubiquitous computing services have been able to satisfy users by providing high quality services. Such services can be found in different areas such as healthcare, social networks, urban transportation, and multimedia. Nowadays there are a wide variety of services with similar functionalities in each of the abovementioned areas. Ranking these services based on Quality of Service(QoS) criteria can help users choose the most appropriate services that meet their preferences. The aim of this research is to create a comprehensive framework based on QoS criteria in ubiquitous environment to rank services. At first by extending the previous research, this paper gathers and classifies the QoS criteria into four classes of architecture, usability, ubiquity, and security. This classification organizes QoS criteria in a hierarchical structure. Afterward, Analytic Hierarchy Process (AHP) is used to propose a customized service ranking framework according to the values of each criterion. To utilize the opinions of experts of the subject, a questionnaire is designed to evaluate the proposed structure of QoS criteria and to determine the weight of each criterion. Finally, comparing the proposed framework with previous approaches indicates that this framework includes a complete set of criteria, which leads to a deep perception of QoS in a ubiquitous computing environment. Moreover, the computed inconsistency ratio indicates that the performed decision-making process has a reasonable consistency.

Key words: Ubiquitous computing, Quality of Service, Service ranking, AHP analysis

1- Introduction

Ubiquitous computing is a new computing model mainly characterized by the invisibility of resources and services to users[1]. The main infrastructure of ubiquitous computing includes user mobile devices, wireless sensor networks, and the Internet. Increasing advances in technology have provided appropriate conditions for the development of ubiquitous computing services. Nowadays, a large number of ubiquitous services are designed and developed in different areas[2, 3] . For instance, assume that a person needs an application to take care of cardiovascular patients. The main features of this application, describing its functional requirements, include monitoring the patient's heartbeat, diagnosing attacks, and reporting them. There are many applications with such functionalities, although each one satisfies different levels of Quality of Service (QoS). It is difficult to compare these applications and choose the most appropriate one, something which needs a comprehensive look at QoS

criteria in a ubiquitous environment. On the other hand, the competitive nature of such environment is undeniable. The diversity of services, implementation complexities, and ubiquitous environment heterogeneities have faced the service providers to the challenge of providing the best service. They instinctively exaggerate in advertising their services. Therefore, advances in ubiquitous computing represent a special need for a comprehensive framework to evaluate the services in that environment.

The aim of this research is to introduce a comprehensive framework for ranking ubiquitous services. For this purpose, at first the QoS criteria in ubiquitous environment are identified. Each of these QoS criteria should be represented quantitatively by a numerical type, percentage, or the size of an interval. Some efforts have been made to create a quantitative view of QoS attributes, and many of them have been quantified with some relationships. However, some of these properties are not still measurable. Finally, services can be ranked by using these quantitative values. Here, it is assumed that services with the same functional features but different quality attributes are compared. Independent of the biased view of developers, this research intends to introduce a framework for ranking ubiquitous services. The output of this ranking shows the fair superiority of a service over another service by considering the relevant criteria.

Reviewing previous studies indicates that ubiquitous environment cannot be evaluated in an integrated way, and it should be investigated from different perspectives such as the evaluation of context information, usability, simulation approaches, and evaluation frameworks[4]. Liu Wei et al. select services in ubiquitous environment by using the genetic algorithm optimization and considering QoS features such as low costs and high efficiency[5]. In a similar study, QoS features such as availability, runtime, cost, and security were used to evaluate the quality of web services in ubiquitous environment[6]. In these two studies, none of the ubiquitous computing criteria such as the quality of context information and the lack of disturbances for users are considered. In another study[7], other criteria such as availability, cost, trust, and access control were used for ranking ubiquitous services. The ranking approach disregards the quality of context and the specific features of a ubiquitous environment such as the invisibility of services.

In this research, after reviewing the QoS criteria investigated in previous studies conducting service ranking and selection, some of such criteria were collected. Given the extensiveness of ubiquitous computing, these criteria were then classified into the proposed architecture, usability, security, and ubiquity classes. Each of these classes includes a number of QoS criteria, which are organized hierarchically. The proposed architecture class consists of scalability, reliability, flexibility, robustness, and integration parameters. Usability class includes user interface aesthetics, offline support, cost of use, and response time parameters. Security class consists of trust, privacy, and integrity parameters. Finally, ubiquity class includes the degree of user distraction, accuracy of context model, context quality, and availability. This primary classification is evaluated and modified by a number of ubiquitous computing experts. Then the AHP technique is employed to rank some services with the same functionality and different QoS features. The calculated inconsistency ratio shows a consistent decision-making. Comparing the QoS criteria considered in ranking indicates that this research has managed to cover a large number of QoS features in ubiquitous services, a fact which is not true about other studies.

The rest of this paper is organized as follows: In the second section, previous studies are reviewed. Section 3 introduces the proposed framework in detail. In section 4, the AHP technique is used to assess the ubiquitous service ranking. Section 5 evaluates the proposed framework. Finally, section 6 concludes the paper and discusses the open research directions.

2- Related works

Reviewing previous studies indicates that it is difficult and complicated to evaluate ubiquitous environments and systems. Complexity is due to the specific features of ubiquitous computing and the vast amount of interactions between systems and heterogeneous infrastructures. Therefore, many researchers have considered and evaluated part of such an environment. These studies include the evaluation of quality of context[8], the evaluation of usability of components in ubiquitous computing services[9-11], user-centric evaluation approaches[12-14], and the evaluation of quality of web services[6, 15]. At first, a brief review of the research conducted on quality of context assessment is provided and then the studies of ranking and selecting ubiquitous services, which are the most relevant to this research subject, are reviewed.

The main difference between ubiquitous computing and traditional services is the use of context information. Each piece of information that can be used to describe the situation of a user or an entity is called context. An entity can be an individual, a place or an object that interacts with the system [16]. Knowing about the quality of context (QoC) and paying attention to it increase the efficiency of context-aware systems and influence the quality of such systems with changes in context values. Any piece of information that describes the quality of data used as context is called the quality of context (QoC)[8]. When there are two services with the same functionalities, the one with a higher quality of context will probably provide a better service[17]. One of the well-known research in this area[8], quantifies the criteria for QoC assessment. It classifies the criteria for QoC into up-to-dateness, trustworthiness, completeness, and significance.

Various web services are provided to users in ubiquitous computing environments. In the last decade, several web service selection approaches have been proposed. Chouiref, et al. [18] propose a framework that enables users to express their preferences by linguistic terms, and enhances the web service selection by leveraging their contexts and profiles. The satisfaction of the candidate web service is expressed by an objective score that takes into consideration not only the user-specified preferences, but also additional preferences extracted from both their context and profile using fuzzy inference rules[18].

Hernandez, et al. present a framework for evaluating the quality of web services. This framework is based on the hypothesis that provisioning the quality of service of a system indicates its ability to respond to users in emergencies and to meet user requirements according to the preferences[6]. This model investigates all aspects influencing the evaluation of web service quality and considers all non-functional criteria including cost, availability, accuracy, runtime, response time, reliability, and penalty[6]. In another research, an ontology of the criteria influencing the quality of web services is proposed. This ontology is used to conduct a hierarchical analysis for ranking web services. The QoS criteria are classified as performance, availability, reliability, security, robustness, economy, integrity, stability, cooperability, and capacity[15].

Another work [19] describes a model that captures service quality including availability, response time, flexibility, scalability, usability, maintainability, functionality, reliability, connectivity, performance, user interface and security. The framework is especially proposed for the mobile learning environment [19].

In continue, the previous studies related to service selection in ubiquitous computing environments are reviewed. Liu Wei *et al.* use the genetic algorithm for service selection in a ubiquitous environment. The aim of optimization in their genetic algorithm is to reduce the cost and increase the efficiency of the service. For this, the quality of service criteria are classified as cost, time, robustness, security, and

availability[5]. In a similar study, other QoS features such as availability, runtime, cost, and security are used to evaluate the quality of web services in a ubiquitous environment[6].

Scholtz, et al. propose a framework for the evaluation of applications in a ubiquitous environment. They classify the quality of service criteria into attention, adoption, trust, conceptual model, interaction, invisibility, side effects, appeal, and application robustness. Each of these classes include some subcategories with definitions. In some cases, a quantification criterion is presented for some features[20]. In another work, QoS features such as availability, cost, trust, and access control are exploited for ranking ubiquitous services. [7].

In all research conducted on service evaluation and selection, the role of QoC has been disregarded. Similarly, the prominent features of ubiquitous environments such as invisibility of services are ignored or at least not investigated deeply. On the other hand, privacy is a very important problem, which should be taken into consideration.

Finally, utilizing optimization algorithms for ranking where there are few QoS features may seem appropriate. However, there are always some difficulties in implementing and modeling them.

This paper proposes a comprehensive framework of criteria of the quality of ubiquitous services. This framework is meant to collect and classify a considerable number of service quality features. The first proposed classification has been evaluated and modified by some experts in ubiquitous computing through a survey. Then the final proposed hierarchical structure of the QoS criteria, and AHP technique is employed to rank ubiquitous services. In comparison with previous research, the proposed approach includes comprehensive criteria for quality of service assessment. In addition to the criteria presented in previous studies, the proposed approach covers several criteria related to the quality of context. Moreover, several relations have been proposed to compute the QoS criteria. Finally, making use of the AHP technique for ranking services in this area is a new step toward evaluating the quality of ubiquitous services. The low computed inconsistency ratio indicates a consistent decision-making process.

3- The proposed ubiquitous services ranking framework

There are various criteria describing the quality of ubiquitous services. They include a wide range of quality concepts. Selective parameters should be categorized into some classes to provide a comprehensive vision of all criteria related to the quality of ubiquitous services.

After reviewing previous studies and surveying QoS criteria selected by them and resolving the conflicts, a comprehensive set of QoS parameters have been collected. After evaluating these selected criteria, we propose to classify them into architecture, usability, security, and ubiquity classes, each of which has subclasses at several levels. Figure 1 shows the proposed classification. In continue, the details of this hierarchical classification is described.

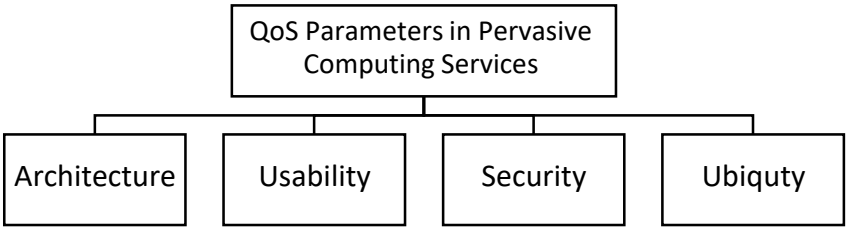

Figure1 The Proposed classification of QoC parameters in ubiquitous computing

3-1- Architecture class: QoS parameters

The first class of QoS criteria includes QoS properties related to architecture. In fact, the design of system architecture has a great impact on the extent to which these properties are achieved in the final system services. There are various definitions of the software system architecture. However, the adopted definition in this study is as follows: "the architecture of a system is defined by a set of structures required by the system, software components of the system as well as their relations, and properties of these components and relations[21]." According to this definition, the proposed architecture class includes scalability, reliability, flexibility, robustness, and integration parameters. Figure 2 shows the class of QoS criteria related to architecture.

The first QoS parameter of the architecture is scalability of a service. Scalability indicates the efficiency of a service in interaction with a large number of entities or users at the same time of an increase in resources[20]. The concept of scalability is very important because it shows the ability of the service to respond to a large number of demands made by users. Given the fact that this criterion has an important role in the QoS provided to users in a large distributed and ubiquitous computing environment, it is necessary to become aware of the service's ability to respond to the increasing demands made by users.

A ubiquitous service is scalable if by simultaneous increase in the number of users and resources, the efficiency could remain fixed. Therefore, a function should be obtained to show this increase, as follows:

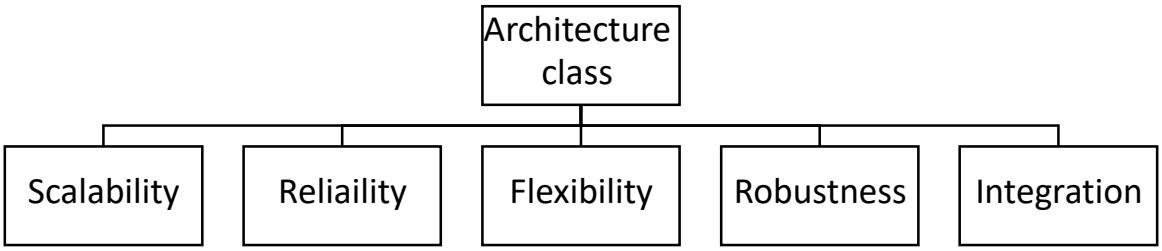

Figure 2. QoS criteria of the architecture class

$$Resources = F(NumOfUsers) \qquad (1)$$

where NumOfUsers shows the number of users, Resources indicates the number and capacity of resources such as memory and processor, and F is the scalability function. Satisfying this equation guarantees the efficiency of the service in large scales. In fact, if the number of users increases, the equation 1 determines the amount of increase in resources to keep the efficiency of the service fixed. The smaller the function, the higher the scalability of the service is. To quantify the scalability level, the largeness of the function is evaluated as follows:

$$Scalability = \begin{cases} 1 & if \quad F \leq linear\ function \\ \frac{1}{n} & if\ F\ is\ a\ polynomial\ function\ of\ degree\ n \\ 0 & otherwise \end{cases}$$

To this end, if the function is linear or smaller, the service has the perfect scalability. If the function is polynomial, the scalability has reverse relation with the degree of the function. If the function is exponential or worse, the scalability degree is zero, because any small increase in the number of users will require a huge amount of resource increase.

The second QoS criterion of the architecture class is reliability, which refers to a degree to which a service performs some specified functions under special circumstances in a specific period of time[21]. Reliability is also defined as the probability at which a service provider responds to requests correctly and during an accepted maximum time[22]. Generally, reliability indicates how a service functions without failures during a specific period and under specified circumstances. Therefore, the mean time to failure promised by provider as well as failures experienced by previous users should be paid attention in order to calculate reliability. The reliability of a service can be computed through Equation 2[23]:

$$reliability = probability\ of\ violation * p_{mttf} = \left(1 - \frac{num\ of\ failures}{n}\right) * p_{mttf} \quad (2)$$

where p_{mttf} indicates the promised mean time to failure. In other words, the service provider promised that it takes at most p_{mttf} to fix every possible failure in their system. *Num of failures* is the number of users who experienced a failure and *n* is number of all users. In a ubiquitous environment, the service reliability is very important. Users perform a part of their daily tasks with intelligent services; therefore, they expect these tasks to be done accurately and timely.

The third QoS parameter of the architecture class is service flexibility. This QoS criterion has been taken into consideration since the very beginning days of software engineering[24]. Flexibility is defined as easiness in changing a system or is components for use in applications or environments apart from what is meant for design and development[25]. Regarding ubiquitous computing, the flexibility of a ubiquitous service also means whether a user can customize it easily. For instance, a user should be able to make some changes in the appearances of shortcuts and application buttons in order to customize and optimize them for themselves. It can generally be stated that a service is flexible if users can change and improve service capabilities and tasks or even add some features according to their preferences[20]. Therefore, Equation 3 is proposed to measure this criterion:

$$flexibility = \frac{number\ of\ tasks\ that\ user\ can\ customize}{total\ number\ of\ tasks} \quad (3)$$

The fourth QoS criterion of the architecture class is service robustness. In computer science, the robustness of a software system refers to the system ability to manage errors during the execution[26]. It means whether the system can take care of errors. Moreover, the robustness can be regarded as the ability of the system to function accurately under unusual circumstances such as facing an inaccurate input[25].

In general, the concept of robustness includes different areas such as programming robustness, machine learning robustness, and communicational network robustness. An appropriate definition of service robustness in the ubiquitous computing area, introduces robustness as a percentage of transient errors

that have not been observed by the user[20]. This value can be calculated by monitoring the service for a specific period of time and enumerating the errors that have been concealed from the user as well as the total number of errors.

The last QoS criterion of the architecture class is integration. System integration is the process of aggregating subsystems into a single system and ensuring that these subsystems can operate as a unanimous system[27].

In the ubiquitous computing environment, several services with limited capabilities work together to achieve a high-level service. For example, nowadays a wide range of services have been designed and implemented for realizing various aspects of smart homes. Each service is responsible for making one part of user requirements smart, Examples of these services are light adjustment, heat adjustment, gate management, and anti-theft service[28]. If all of these services are integrated, the user will literally enjoy living in a smart home. Equation 4 is proposed to measure the service integration:

$$integration = \frac{number\ of\ integratable\ services}{number\ of\ services\ that\ the\ user\ needs\ to\ integrate} \quad (4)$$

3-2- Usability class: QoS parameters

Usability depends on how much a user is comfortable when a desirable task is performed by the service. Usability includes areas of learning service features and efficient use of it to increase satisfaction[21]. Usability is also defined as a degree to which a service can be used by specific users to achieve specific goals with efficiency and satisfaction [29]. In another definition, usability includes two groups of quality properties[30]:

- Learnability: it includes efforts made by different users to learn.
- Operability: it includes software ability to let users utilize the service easily in a specific environment.

This class of QoS parameters in ubiquitous computing includes four criteria of user interface aesthetics, offline support, cost of use, and response time as shown in Figure 3.

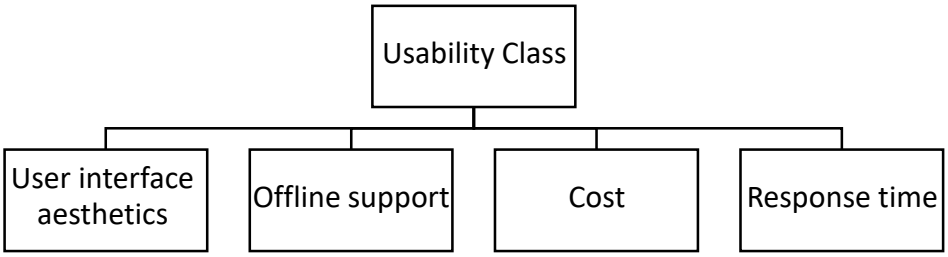

Figure 3. QoS criteria of the usability class

The first QoS criterion of the usability class is user interface aesthetics. It is defined as a degree to which the user interface can make the user satisfied with the application[29]. The user interface aesthetics includes intuitiveness, ease of use, and the aesthetics of graphical components[31]. Different methods have already been proposed for the design and evaluation of the user interface[32]. Furthermore, the user interface aesthetics can be quantitatively measured. Previously, the user interface aesthetics criteria have been investigated and quantified in 14 classes including balance, equilibrium, symmetry, sequence, cohesion, unity, proportion, simplicity, density, regularity, economy, homogeneity, rhythm and order & Complexity [33].

Given different methods of evaluating the user interface aesthetics, one of the best ways is to survey the users. Due to the complexity of predicting users' opinions, the aggregation of them may sometimes be different from the expectations of user interface designers and developers. Therefore, user opinions and tastes always determine the success of efforts made for the beautiful and optimal user interface design.

The second QoS criterion of usability class is offline support. Most of ubiquitous services need to connect to storage and computing servers to provide users with services. However, when the network is disconnected, some limited services may be provided locally on the user's smart device. This feature is important because user may not be able to connect to the Internet in some situations. The following equation is used to create a quantitative criterion for this service quality feature:

$$Offline\ support = \frac{number\ of\ subservices\ that\ are\ provided\ offline}{total\ number\ of\ subsevices} \quad (5)$$

The third criterion of usability class is the cost of using a service. The cost of each service needs to be paid by the user who wants to use it. Although some users are willing to pay any costs to achieve a higher level of quality, the payable cost can finally be influential in comparing two services of the same functionality.

The final QoS criterion of the usability class is response time. If a service has an unacceptable response delay, it will definitely lose users. Especially in a ubiquitous environment, services are interactive and response delay is not acceptable. For instance, in pervasive healthcare systems, if a service diagnoses and reports a heart attack with much delay, the patient’s life is put in danger. Response time is regarded as the time interval between a request is sent by the user and the response is received[20].

3-3- Ubiquity class: QoS parameters

Ubiquity includes all the specific features of ubiquitous computing, which distinguish it from distributed computing. Architecture and usability classes can be used for both distributed and ubiquitous services. However, the QoS criteria categorized as ubiquity are specific to the nature of ubiquitous services. They are the inseparable part of each service in this environment. In fact, if each of these QoS criteria is disregarded, the outlook of ubiquitous computing is missed. This class includes four QoS criteria of distraction, accuracy of context model, quality of context, and availability (figure 4). Among them, QoC has four subclasses including context up-to-dateness, context trustworthiness, context completeness, and sensor error rate.

According to the vision described by Weiser, ubiquitous services will interweave in the user's life. In fact, these services monitor users' needs to meet them in a way that they have to interfere the least[34]. This vision corresponds to the lack of user distraction. In other words, ubiquitous services should make the least distractions. To measure this QoS criterion, the number of times the user is asked to intervene during the service run can be exploited. The fewer times the user is asked to intervene, the less the service has bothered the user, and the higher the QoS is [20].

The next QoS criterion of the ubiquity class is the accuracy of context. User satisfaction with the accuracy of response received from the service depends highly on how accurately the context is

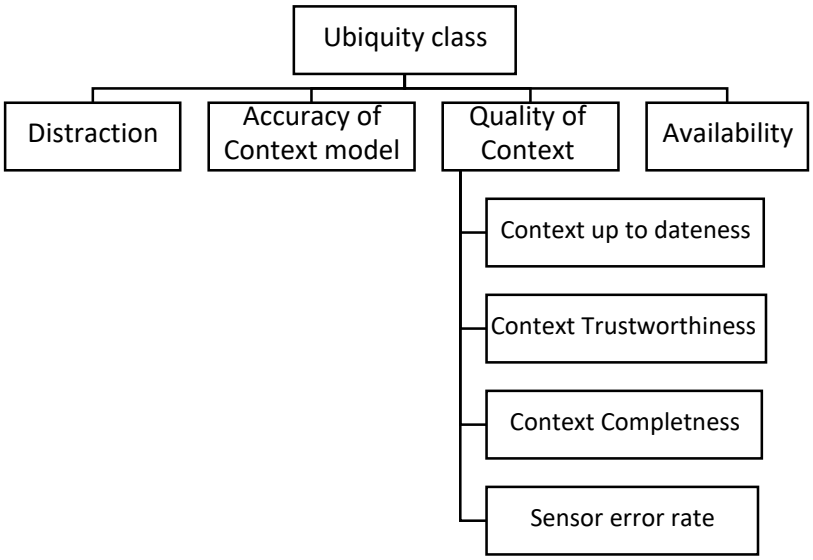

Figure 4. QoS criteria of the ubiquty Ccass

modeled. Each context-aware system utilizes a context model, which helps in perceiving current situation. The accuracy of context model means the adaptation between the exploited context model and the status of the real environment. On the other hand, modeling the environment and context information inaccurately results in the difference between the service response and the response expected by the user[20].

The next QoS criterion of ubiquity class is the Quality of Context(QoC). This criterion is important because context information is the foundation of decisions on services provided by the system. If there are errors in the context values, the response given to the user will be wrong. The QoS is highly important in critical applications such as pervasive healthcare systems. QoC itself has four criteria described as follows:

The first criterion of QoC is up-to-dateness. The up-to-dateness of information indicates its lifetime. In other words, it shows how long it has been acquired. This criterion is especially important for dynamic

context elements. In a healthcare service for instance, if old values of the context information describing a patient's vital signs are used to make decisions, it can have dangerous effects on the user. To measure this criterion, the division of the context age to lifetime is proposed, which is described in Equations 6 and 7 [8].

$$Age(context) = Current\ time - Measurment\ time \qquad (6)$$

$$Uptodateness(context) = \begin{cases} 1 - \frac{Age(context)}{Life\ time(context)}; & if\ Age(context) < Life\ time(context) \\ 0 \quad ; & otherwise \end{cases} \qquad (7)$$

The next criterion of QoC is the trustworthiness of context information. It shows how much the context resource can be trusted. This trust depends on the distance between the sensor and the sensed entity. The farther the distance between an entity and a sensor is, the more doubtful the accuracy of context value is. The following equation is used to measure the trustworthiness of context [8]:

$$Trust(context) = \begin{cases} 1 - \frac{d(s,e)}{d_{max}} * \delta & if\ d(s,e) < d_{max} \\ 0 & else \end{cases} \qquad (8)$$

where *d(s,e)* shows the distance between a sensor and the intended entity. Moreover, d_{max} shows the maximum possible distance between an entity and the sensor, and δ is a statistical parameter, the calculation of which has already been discussed[35].

The third criterion for the quality of context is completeness. Complex context information may include several context elements. The completeness indicates the percentage of pieces of context elements that are available. Therefore, it is computed by considering the rate of available context elements to the total elements that should be collected[35].

$$Completeness = \frac{Available\ Context\ elements}{Total\ number\ of\ Context\ elements} \qquad (9)$$

The last criterion of QoC is the sensor error rate. The higher the sensor error rate is, the lower the perceived QoC will be. The sensor error is usually measured in percentage by the manufacturing company.

Finally, the last QoS criterion for ubiquity is service availability, which means how much the probability of service availability is. The availability of a service can be calculated through the following formula:

$$Availability = \frac{Available\ time}{Total\ time} \qquad (10)$$

3-4- Security class: QoS parameters

The security of a computer system is a measurement of the system ability to protect data and information against unauthorized accesses. Every action meant to damage the system security is called an attack which can have different types[21]. This class of QoS criteria includes trust, privacy and integrity as shown in figure 5.

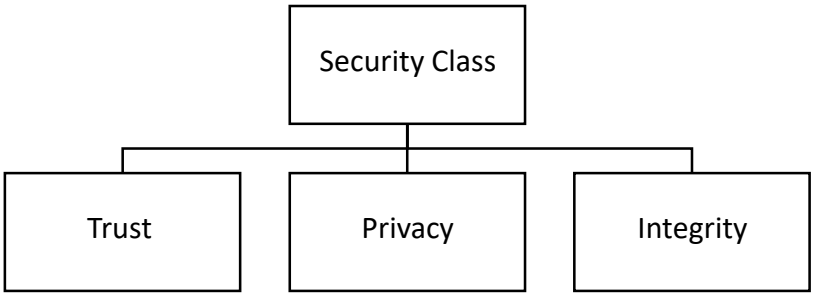

Figure 5. QoS criteria for the security class

Trust is the first QoS criterion of the security class. In ubiquitous computing, different services are provided by unknown service providers to users. Therefore, before a user exploits a service, their trust should be gained.

To measure the total trust of a service, the trust of each user to the service should be computed. In the measurement method, trust is computed between zero and ten. At first, it is initiated into 5 for all users. Then each time a user exploits the service, a feedback of user satisfaction is received between zero and ten. This feedback is used to update the value of trust. In this regard, success ratio is taken into consideration in addition to user feedbacks. Equation 11 shows the trust of the i^{th} user to the service.

$$\text{Trust}_i = \frac{user\ satisfaction_i + success\ ratio_i}{2} \qquad (11)$$

where user satisfaction_i indicates the satisfaction level of the i^{th} user to the service, and success ratio_i shows the success rate of the service for the i^{th} user. To compute the trust of the service, a weight is assigned to each user in a way that loyal users gain importance. The weight is between zero and one; For the first use, the user gets the weight of 0.1. equation 12 shows the trust computation for the service:

$$Trust = \frac{\sum_i (w_i * Trust_i)}{\sum_i w_i} \qquad (12)$$

The satisfaction level of a user in each transaction is dependent on the user satisfaction of the previous transaction as well as her average satisfaction level till now. The α coefficient (is between zero and one) regulates the role of these two parameters. In this research, we assume that the average satisfaction level is more important; thus α is set to 0.3. Equation 13 is used to update the satisfaction level of the user.

$$User\ Satisfaction_i(t) = \alpha * User\ satisfaction_i(t-1) + (1-\alpha) * Avg(User\ satisfaction_i(t)) \qquad (13)$$

The success ratio of the service is updated through equation 14. To this end, the success ratio of the service in the transaction of t-1 as well as the mean success ratio till now are considered. Similarly, the β coefficient is defined between zero and one. In the same way as α, the β coefficient is set to 0.3 in this research.

$$success\ ratio_i(t) = \beta * success\ ratio_i(t-1) + (1-\beta) * Avg\big(success\ ratio_i(t)\big) \qquad (14)$$

The next QoS criterion in this category is privacy. Protecting personal information of users is a well-known issue of the pervasive systems. The developers should always ensure users that their personal information, such as contextual information, will never be revealed to unauthorized parties. Generally,

privacy protection includes actions that the system takes to protect personal information of users. Moreover, privacy can be regarded as the unavailability of user information to unauthorized parties[20]. The privacy of a service for a particular user (user i) is computed according to equation 15:

$$privacy_i = 1 - \frac{number\ of\ revealed\ data\ elements\ of\ user_i's}{Total\ number\ of\ user_i's\ data\ elements} \qquad (15)$$

Therefore, equation 16 is used to compute the privacy of the service by performing a weighted mean to the privacy value of its users:

$$privacy = \frac{\sum_i (w_i * privacy_i)}{\sum_i w_i} \qquad (16)$$

where w_i is the weight of user i, which has also been used in the trust formula.

Integrity is the final QoS criterion of the security class. It means that the service should be protected against un-allowed changes. If an adversary could corrupt the data or functionality of a service, the integrity is violated. In default, the integrity is equal to one. In the case of corruption in data or functionality of the service, the integrity is computed according to the equation 17, as follows:

$$integrity = 1 - the\ percentage\ of\ corruption\ in\ data\ or\ service\ functionality \qquad (17)$$

4- Ubiquitous Services Ranking

After codifying a comprehensive list of QoS criteria in ubiquitous computing and classifying them, it is time to rank services, accordingly. This ranking is fair and unbiased. It is done without human intervention so that it can be automated by the computer. Making decisions on the superiority of a service over another, or ranking, is regarded as a multi-criteria decision making (MCDM) problem.

It is not easy to make decisions on whether a service with all of its features is superior to other services because every important point should be taken into account to make a right decision. There are different methods for dealing with large-scale decision-making problems. The common nature of all of them is that they try to put together all the criteria influencing decision-making in an integrated structure. It is possible that some criteria may contradict each other. Therefore, the MCDM method should be able to put together even these contradicting criteria and provide the best solution. This may be tough for human who may make mistakes.

The second feature of MCDM methods is that they can make the best decision without any partiality and based on only the documents given to them as inputs. This feature makes them reliable so that their results can be trusted. However, the solution provided by an MCDM method can be accepted without any doubts, if only the accuracy of inputs is ensured. More importantly, the criteria for making decisions should be selected and modelled, correctly. Besides, the importance of each criterion should be determined by reliable people or methods.

One of the important components of each MCDM method includes the decision-making alternatives. The aim of each decision-making problem is to determine the superiority of an alternative to another one. Therefore, after selecting the criteria for making decisions and determining the importance of each one, the MCDM method puts all of them in an integrated structure and starts comparing the value of each criterion for each alternative. What principles draw the comparison between alternatives depends on the MCDM method. Finally, the results are aggregated, and the best alternative is chosen.

The AHP technique is one of the most widely used methods to solve MCDM problems. AHP simplifies complicated and unstructured MCDM problems by putting them together in a hierarchical structure. It is based on the pairwise comparisons drawn between criteria, which makes it better than an optimization function and a weighting method. Another advantage of AHP is its flexibility as well as the ability to investigate inconsistencies among decision making criteria. In addition, it divides a complicated decision-making problem into smaller parts and creates a hierarchical structure, which makes the problem easier to understand [23, 38]. In this research, the AHP technique is used for ranking services.

The AHP technique includes three main phases: analyzing the problem, judging the priorities, and aggregating the priorities [39]. The main steps of the proposed ranking approach, which is based on AHP are as follows:

Phase 1 - Creating a hierarchical structure for ubiquitous services:

The first step is to create a hierarchy of parameters for ranking. According to Figure 6, this hierarchical structure has several levels. The aim of ranking is at the top level. Proposed classes of QoS criteria are at the second level. The next levels are the criteria and sub-criteria. Finally, available services are put at the lowest level as ranking alternatives.

Phase 2 – Calculating the weight of each QoS criterion: The importance of each QoS feature should be specified in this phase. To this end, the weighting method is used. These weights are collected through a questionnaire distributed among experts of this area. Different numerical intervals are considered for these weights. In this research, experts determine the importance of each QoS between zero and four. Zero indicates that the QoS criterion has no importance, and four shows the highest level of importance. Then these weights are aggregated by using the mean method and are normalized between zero and one. These weights indicate the superiority of a QoS criterion regarding others.

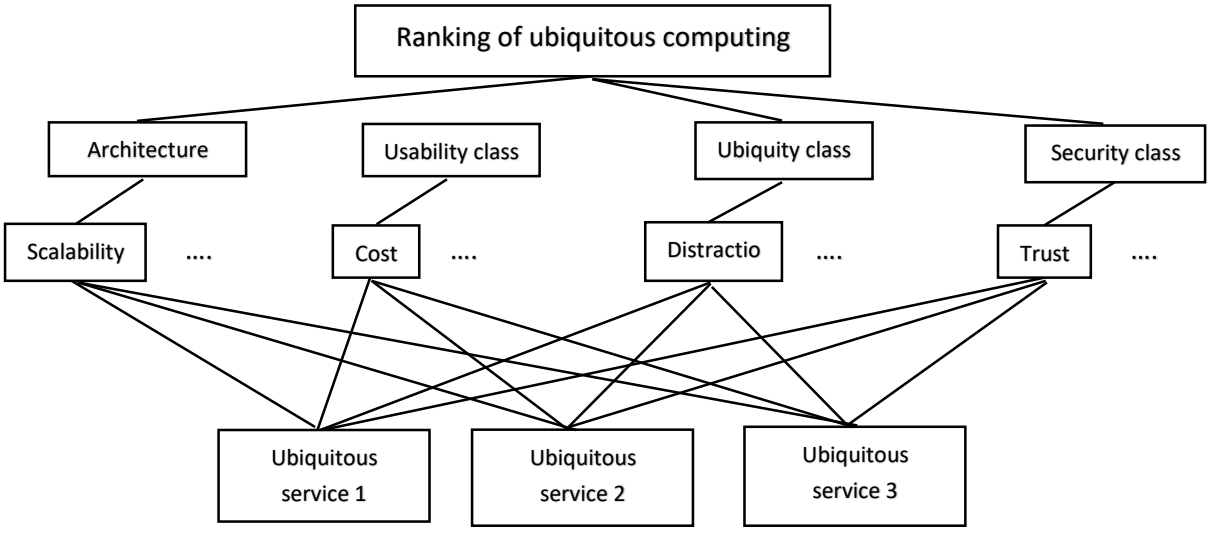

Figure 6. AHP hirarchical structure for ubiquitous computing services

Phase 3 – Calculating the relative value-based weights for ranking services: These weights indicate the superiority of a ubiquitous service to another according to the values of the lowest level QoS criteria

at the AHP structure. The QoS criteria at the lowest level may have different value types. For instance, the value of response time is expressed as a numerical interval, and the value of integration is stated numerically. Moreover, it is possible that some of the QoS criteria may be unknown-valued. Therefore, one of the challenges is to allocate weights to the criteria that are not measurable. For this purpose, the method proposed by Tran *et al*. is used[15].

Assume that w_q is the weight allocated to the criterion q. Moreover, v_i and v_j are the values of q in services s_i and s_j, *respectively*. Then s_i/s_j shows the rank of s_i regarding s_j. When a QoS criteria of two different services are compared with each other, its type should be taken into account. According to the different types of values of QoS criteria, the way of comparison is discussed:

Type 1 – If the values of the QoS criterion are of numbers or percentages, then:

- If the greater value is better: ${s_i}/{s_j} = \frac{v_i}{v_j}$
- If the smaller value is better: ${s_i}/{s_j} = \frac{v_j}{v_i}$

Type 2 – If the values of the QoS criterion are from a sorted interval, then:

$$ {s_i}/{s_j} = \frac{len(v_i \cap v_r)}{len(v_j \cap v_r)} $$

where *len()* indicates the size of a set, which is equal to the number of members. In addition, v_r indicates the required value of the QoS criterion for the user.

These relations can be used to compare two values of a QoS criterion. Therefore, they can be employed to estimate the superiority of each service to another one with respect to one QoS criterion. If the comparison is drawn for *n* different services, the result of each pairwise comparison will be an *n*n* matrix (for only one QoS criterion). This matrix is called the relative service ranking matrix (RSRM); thus, there are such matrices as many as the number of QoS criteria. For instance, Table 1 shows the value of architecture class for three different services. Considering only the integration criterion and given the fact that a greater value is better for this criterion, the RSRM is calculated as follows:

Table 1: the values of architecture class criteria

Top Level QoS criteria	*First level parameters(w)*	*Service1(S_1)*	*Service2(S_2)*	*Service3(S_3)*	*Value type*
Architecture class criteria	Integration(0.19)	1.25	*1*	*0.75*	*Numeric*
	Scalability(0.19)	*1*	$\frac{1}{2}$	$\frac{1}{2}$	*Numeric*
	Reliability(0.23)	*0.36%*	*0.54%*	*0.72%*	*Percent%*
	Flexibility(0.19)	*0.66*	*0.72*	*0.80*	*Numeric*
	Robustness(0.20)	*0.83%*	*0.88%*	*0.86%*	*Percent%*

$$RSRM_{integration} = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{ccc} S_1 & S_2 & S_3 \\ 1 & \frac{1.25}{1} & \frac{1.25}{0.75} \\ \frac{1}{1.25} & 1 & \frac{1}{0.75} \\ \frac{0.75}{1.25} & \frac{0.75}{1} & 1 \end{array} = \begin{array}{ccc} 1 & 1.25 & 1.66 \\ 0.8 & 1 & 1.33 \\ 0.6 & 0.75 & 1 \end{array}$$

After calculating all RSRMs, the eigenvector of each matrix is obtained, which is a vector with *n* columns. In this vector, the more the value of a column, shows the relative superiority of the corresponding service from the perspective of the specific QoS criterion. The vector is also called a relative service ranking vector(RSRV). Similarly, the RSRV of the integration criterion is calculated as follow. This vector shows that service 1 is superior to service 2 in terms of integration, which is in turn superior to service 3.

$RSRV_{integration} = [0.4167, 0.3333, 0.2500]$

Phase 4 – Aggregating RSRVs of QoS criteria:

In this stage, the vectors of each class are aggregated in one matrix named RSRM. For instance, the following matrix results from the aggregation of RSRVs of the architecture criteria. The first to fifth columns of this matrix include the values of RSRVs for integration, scalability, reliability, flexibility, and robustness, respectively.

$$RSRM_{Architecture} = \begin{pmatrix} 0.4167 & 0.2778 & 0.2222 & 0.3028 & 0.3230 \\ 0.3333 & 0.3333 & 0.3333 & 0.3303 & 0.3424 \\ 0.2500 & 0.3889 & 0.4444 & 0.3670 & 0.3346 \end{pmatrix}$$

In the next step, this matrix is multiplied by a column matrix containing the weights allocated to QoS criteria. The result is a column matrix indicating the relative superiority of services with respect to the criteria of architecture class. The multiplication of weights by RSRM for architecture class indicates that the third service bears the highest quality in total.

$$RSRV_{Architecture} = \begin{pmatrix} 0.4167 & 0.2778 & 0.2222 & 0.3028 & 0.3230 \\ 0.3333 & 0.3333 & 0.3333 & 0.3303 & 0.3424 \\ 0.2500 & 0.3889 & 0.4444 & 0.3670 & 0.3346 \end{pmatrix} \begin{pmatrix} 0.19 \\ 0.19 \\ 0.23 \\ 0.19 \\ 0.20 \end{pmatrix} = \begin{pmatrix} 0.3084 \\ 0.3344 \\ 0.3572 \end{pmatrix}$$

The steps to calculating RSRV are repeated for each QoS criteria class including usability, ubiquity, and security. Finally, a matrix is obtained. In this matrix, the first to fourth columns show RSRVs for QoS criteria of architecture, usability, ubiquity, and security, respectively.

$$RSRM = \begin{pmatrix} 0.2899 & 0.2985 & 0.3383 & 0.4011 \\ 0.3562 & 0.3716 & 0.3553 & 0.2954 \\ 0.3644 & 0.3099 & 0.3063 & 0.3035 \end{pmatrix}$$

Then, a weight is allocated to each class and the RSRM matrix is multiplied in the weight vector. In this example, the weights allocated to different classes are regarded as 0.25; therefore, the multiplication shows the final result of ranking based on the proposed method. As it is clear in this example, the first

service has the best total QoS. After that, the second and third services come next, respectively. Therefore, it can be stated that service ranking indicates that the first service presents the best QoS level.

$$RSRV=\begin{pmatrix}0.3084 & 0.3871 & 0.3334 & 0.3170\\ 0.3344 & 0.2993 & 0.3608 & 0.3352\\ 0.3572 & 0.3135 & 0.3058 & 0.3478\end{pmatrix}\begin{pmatrix}0.25\\ 0.25\\ 0.25\\ 0.25\end{pmatrix}=\begin{pmatrix}0.3365\\ 0.3324\\ 0.3311\end{pmatrix}$$

5- Evaluation

To validate the proposed framework and obtain weight of each criterion, a questionnaire[40] has been designed by Google Doc [41]. Afterward, 20 experts of this area all over the world have been invited to complete it. Finally, five of them filled out the questionnaire The questionnaire includes four sections, each of which is allocated to one class of QoS criteria.

The questionnaire has two groups of questions. The first group is mean to validate the accuracy of the proposed framework regarding the criteria and their classes. The responses are either positive or negative. If a response is negative, the reason is also being provided by the respondent. For most criteria, the proposed initial framework was confirmed by experts. In a few cases, some modifications were made according to experts opinions. The final framework has been introduced in the previous section.

The second group of questions have been about determining the weight of each QoS criterion. Each of the experts has allocated a weight to each QoS criterion. These weights have been initiated from zero to four. Then, they have been aggregated by the mean method and normalized to the range of zero to one. Final weights are shown in Table 2.

Table 2: Aggregated and normalized weights to different QoS criteria

QoS Classes & criteria	Architecture class					Usability class				Ubiquity class				Security class		
	Integration	Scalability	Reliability	Flexibility	Robustness	User interface usability	Offline support	Cost	Response time	Distraction	Accuracy of context model	QoC	Availability	Trust	privacy	integrity
Aggregated and normalized weights by experts [0,1]	0.19	0.19	0.23	0.19	0.20	0.31	0.21	0.20	0.28	0.24	0.25	0.26	0.25	0.35	0.33	0.32

In continue, a use case scenario investigates the application of the proposed framework.

"Katherine has recently moved to a new place. Her home has the necessary infrastructure to become smart. It has light sensors, heat sensors, closed-circuit cameras, and a wireless Internet. She is looking for a ubiquitous application to make her home intelligent. After searching into a large number of services which can be chosen by her, she become confused. She cannot confidently choose and buy the most appropriate service out of a large number of applications providing different levels of QoS. On the other hand, she is not familiar with QoS criteria in a ubiquitous environment. She decides to refer to the proposed framework of ranking ubiquitous computing services. She searches for the name of the best three services that make homes smart. The implemented website of the proposed framework

specifies the first ranked services presenting the highest level of quality. Table 3 shows the QoS criteria for them. Therefore, she can choose the best service."

Table 3: The QoS criteria of the three different ubiquitous services

Top level QoS classes	First level criteria	Second level criteria	Service1(S_1)	Service2(S_2)	Service3(S_3)	Value type
Architecture class	Integration		*1.25*	*1*	*0.75*	*Numeric*
	Scalability		*1*	$\frac{1}{2}$	$\frac{1}{2}$	*Numeric*
	Reliability		*36%*	*54%*	*72%*	*Percent*
	Flexibility		*0.66*	*0.72*	*0.80*	*Numeric*
	Robustness		*83%*	*88%*	*86%*	*Percent*
Usability class	User interface usability		*87%*	*83%*	*91%*	*Percent*
	Offline support		*0.12*	*0.09*	*0.06*	*Numeric*
	Cost		*5$*	*7$*	*6$*	*Numeric*
	Response time		*[50-70]s*	*[55-70]s*	*[50-75]s*	*Range*
Ubiquity class	Distraction		*4*	*3*	*5*	*Numeric*
	Accuracy of Context model		*87%*	*90%*	*80%*	*Percent*
	Quality of Context(QoC)	*Up to dateness*	*0.87*	*0.73*	*0.83*	*Numeric*
		Trust worthiness	*50*	*60*	*40*	*Numeric*
		Completeness	*68%*	*59%*	*63%*	*Percent*
		Sensor error rate	*98%*	*92%*	*90%*	*Percent*
	Availability		*99.95%*	*99.99*	*100%*	*Percent*
Security class	Trust		*8.65*	*8.23*	*7.72*	*Numeric*
	Privacy		*0.57*	*0.66*	*0.83*	*Numeric*
	Integrity		*0.84*	*0.92*	*0.86*	*Numeric*
Final weight of each service	-	-	*0.3365*	*0.3324*	*0.3311*	-

The importance of computing the inconsistency ratio is in initial approving the utilized pairwise comparison data. If the inconsistency ratio becomes higher than 0.1, the pairwise comparison matrix loses its validity. Equations 18 and 19 are used to compute the inconsistency index, respecively.

$$I.I. = \frac{\lambda_{max} - n}{n - 1} \quad (18)$$

$$I.R. = \frac{I.I}{I.I.R} \quad (19)$$

, where λ_{max} is the eigenvalue of the matrix and n indicates its dimentions. Since we have $A \times w = \lambda_{max} \times w$ for the matrix A with the relative weight vector of W, we use averaging between array elements of λ_{max} to compute λ_{max}. To computing $I.I.R$ for each matrix, we need the amount of $I.I.R$ of random matrix, which is shown in table 4 according to the matrix dimentions.

Table 4: Inconsistency rate for the random matrix

N	1	2	3	4	5
I.I.R.	0	0	0.58	0.9	1.12

To compute the inconsistency ratio of a hierarchical structure, the inconsistency index of each matrix (I.I) is multiplied by the weight of the related element, and their sum is obtained ($\overline{\text{I. I.}}$). In addition, we multiply the elements weights by I. I. R. and compute their summation ($\overline{I.I.R.}$). The division of $\overline{I.I.}/\overline{I.I.R.}$ results the hierarchical inconsistency ratio. By following these steps for ranking three services of the smart home, the inconsistency ratio is computed as 0.0001393. It shows that the ranking procedure has an excellent consistency.

Table 5 indicates the comparison drawn between QoS criteria introduced in previous studies and the proposed framework. In the architecture class, the integration criterion is disregarded in all previous studies. However, relevant services are provided by different applications nowadays, if these services can be integrated, users will experience a high-level of satisfaction. Besides, it will be possible to implement high-quality services by integrating low-level ones. Among the previous research, scalability is regarded as a criterion only in the study conducted by Jean Scholtz[20] and Tran[15]. This criterion is important because it shows the service ability to respond to a large number of requests made by users. Reliability is regarded as a QoS criterion in studies conducted by Enrique Hernandez[6] as well as Salaja Silas[7] and Tran[15]. Service robustness is also considered in studies conducted by Liu Wei[42] and Enrique Hernandez[6]. Generally, the proposed framework focuses on architecture by putting a significant number of QoS criteria together.

In the usability class, the ease-of-use of the user interface is only presented in the study conducted by Jean Scholtz[20]. In addition, the cost of use is considered in all of the previous research. However, offline support is the only QoS criterion that is considered only in the proposed framework. It enables users to benefit from some offline features when there is no Internet connection.

Ubiquity QoS criteria is one of the contributions of this work. In this category, user distraction as well as accuracy of context model are dealt with only in the study conducted by Jean Scholtz[20]. Availability is considered in all previous studies except the work by Liu wei[42]. However, the quality of context is only investigated in the proposed framework. Since the performance and accuracy of ubiquitous services depend highly on the input contextual information, if the context does not have an acceptable level of quality, it will have a negative impact on the quality of the service.

Finally, in the security class, privacy is only considered in one research. However, nowadays a large number of ubiquitous computing applications use the personal contextual information of users. Therefore, users must be ensured that their private information are not going to exposed. Besides, trust is considered in half of the previous studies, but integrity has been only proposed in one study[15]. Service capability in neutralizing the sabotage actions of adversaries is important and undeniable.

Table 5: Comparison of QoS criteria supported by different approaches

NS=(Not Support)- S=(Support)

6231850023X ,

	papers *QoS* parameters	Liu wei, et al. [5]	Liu wei et al. [42]	Enrique Hernandez [6]	Salaja Silas, et al. [7]	Jean Scholts et al. [20]	Tran, et al. [15]	Proposed Framework
Architecture parameters	Integration	*NS*	*NS*	*NS*	*NS*	*NS*	*NS*	*S*
	Scalability	*NS*	*NS*	*NS*	*NS*	*S*	*S*	*S*
	Reliability	*NS*	*NS*	*S*	*S*	*NS*	*S*	*S*
	Flexibility	*NS*	*NS*	*NS*	*NS*	*S*	*NS*	*S*
	Robustness	*S*	*NS*	*S*	*NS*	*NS*	*NS*	*S*
Usability parameters	User interface usability	*NS*	*NS*	*NS*	*NS*	*S*	*NS*	*S*
	Offline support	*NS*	*NS*	*NS*	*NS*	*NS*	*NS*	*S*
	Cost	*S*	*S*	*S*	*S*	*S*	*S*	*S*
	Response time	*S*	*NS*	*S*	*NS*	*S*	*S*	*S*
Ubiquity parameters	Distraction	*NS*	*NS*	*NS*	*NS*	*S*	*NS*	*S*
	Accuracy of Context model	*NS*	*NS*	*NS*	*NS*	*S*	*NS*	*S*
	Quality of Context(QoC)	*NS*	*NS*	*NS*	*NS*	*NS*	*NS*	*S*
	Availability	*S*	*NS*	*S*	*S*	*S*	*S*	*S*
Security parameters	Trust	*NS*	*NS*	*NS*	*S*	*S*	*NS*	*S*
	Privacy	*NS*	*NS*	*NS*	*NS*	*S*	*NS*	*S*
	Integrity	*NS*	*NS*	*NS*	*NS*	*NS*	*S*	*S*

Apparently, compared with previous studies, the proposed framework provides a comprehensive set of criteria, some of which such as integration, offline support, quality of context, and access control have not been considered in the related works.

6- Conclusion and Open Research Areas

The development of a large number of ubiquitous services with the same functional capabilities and different QoS features have confused users to find the most appropriate service. In this research, the QoS criteria for ubiquitous computing have been proposed and classified. Afterward, the categorized metrics have been evaluated by experts filling out a relevant questionnaire. After necessary modifications, this classification includes architecture, usability, ubiquity and security. Likewise, the importance of each QoS criterion has been determined according to experts' opinions. In the next step, a framework has been proposed to rank services by using the AHP technique. In addition, a case study of three services with different features has indicated how the proposed framework operates. The inconsistency ratio has been computed for the ranking process. It approves the high consistency of the decision-making process. The comparison results have shown that the proposed framework provides a more comprehensive set of QoS criteria than previous studies.

Although many QoS criteria have been quantified in this research, regarding some of the QoS criteria, there is not an accurate measurement formula. In the next step, works can be done on quantifying such features. Quantifying such QoS criteria enables us to replace binary values with numerical values when using multi-criteria decision-making approach. This will increase the ranking accuracy.

Nowadays ubiquitous computing is widely used in various domains, each of which emphasizes different aspects of a service. For example, in healthcare or transportation areas, the response time as well as quality of context are very highlighted. Given the diversity of domains of ubiquitous computing, extracting the QoS criteria specific to each of them is regarded as a future research direction. For

instance, the quality of ubiquitous services can be professionally investigated for the special services in healthcare, transportation or social networks. Definitely, the general model of evaluating QoS cannot be applied to all of these areas, without losing accuracy.

References

[1] C. E. K. geihs, R. Rhichle, M.Wagner, M. U. Khan, "Development Support for QoS-Aware Service-Adaptation in Ubiquitous Computing Application," in *International Confrance on Applied Computing*, 2011, pp. 197-202.

[2] H. Vahdat-Nejad, A. Ramazani, T. Mohammadi, and W. Mansoor, "A survey on context-aware vehicular network applications," *Vehicular Communications*, vol. 3, pp. 43-57, 2016.

[3] S. Poorejbari and H. Vahdat-Nejad, "An Introduction to Cloud-Based Pervasive Healthcare Systems," in *International Conference on Context-Aware Systems and Applications*, 2014, pp. 173-178.

[4] B. Abdulrazak and Y. Malik, "Review of challenges, requirements, and approaches of pervasive computing system evaluation," *IETE Technical Review*, vol. 29, pp. 506-522, 2012.

[5] L. Wei, Z. Lanfei, and D. Jianqiang, "Research on an adaptive algorithm of service selection in pervasive computing," *Journal of Digital Information Management*, vol. 11, pp. 482-488, 2013.

[6] E. L. Hernandez, M. Laforest, and C. Rodriguez, "Evaluation framework for quality of service in web services: implementation in a pervasive environment," Master thesis in INSA Lyon, 2010.

[7] S. Silas, E. B. Rajsingh, and K. Ezra, "An efficient service selection framework for pervasive environments," *International Journal of Wireless and Mobile Computing*, vol. 6, pp. 80-90, 2013.

[8] A. Manzoor, H.-L. Truong, and S. Dustdar, "On the evaluation of quality of context," in *International Conference on Smart Sensing and Context*, 2008, pp. 140-153.

[9] H. J. Kim, J. K. Choi, and Y. Ji, "Usability evaluation framework for ubiquitous computing device," in *International Conference on Convergence and Hybrid Information Technology* 2008, pp. 164-170.

[10] D. Zhang and B. Adipat, "Challenges, methodologies, and issues in the usability testing of mobile applications," *International Journal of Human-Computer Interaction*, vol. 18, pp. 293-308, 2005.

[11] Y. G. Ji, J. H. Park, C. Lee, and M. H. Yun, "A usability checklist for the usability evaluation of mobile phone user interface," *International Journal of Human-Computer Interaction*, vol. 20, pp. 207-231, 2006.

[12] R. Iqbal, J. Sturm, O. Kulyk, J. Wang, and J. Terken, "User-centred design and evaluation of ubiquitous services," in *International conference on Design of communication: documenting & designing for pervasive information*, 2005, pp. 138-145.

[13] K. Höök, "User-centred design and evaluation of affective interfaces," in *From brows to trust*, ed: Springer Netherlands, 2004, pp. 127-160.

[14] L. Arhippainen and M. Tähti, "Empirical evaluation of user experience in two adaptive mobile application prototypes," in *International Conference on mobile and ubiquitous multimedia*, 2003, pp. 27-34.

[15] V. X. Tran, H. Tsuji, and R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for Web services," *Simulation Modelling Practice and Theory*, vol. 17, pp. 1378-1398, 2009.

[16] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and ubiquitous computing*, 1999, pp. 304-307.

[17] H. Vahdat-Nejad, "Context-aware middleware: a review," in *Context in Computing*, ed: Springer, 2014, pp. 83-96.

[18] Z. Chouiref, A. Belkhir, K. Benouaret, and A. Hadjali, "A fuzzy framework for efficient user-centric Web service selection," *Applied Soft Computing,* vol. 41, pp. 51-65, 2016.
[19] M. Sarrab, M. Elbasir, and S. Alnaeli, "Towards a quality model of technical aspects for mobile learning services: An empirical investigation," *Computers in Human Behavior,* vol. 55, pp. 100-112, 2016.
[20] J. Scholtz and S. Consolvo, "Toward a framework for evaluating ubiquitous computing applications," *Pervasive Computing,* vol. 3, pp. 82-88, 2004.
[21] P. C. Len Bass, Rick Kazman, "Software Architecture in Practice," Third edition ed, 2012.
[22] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," in *International conference on World Wide Web*, 2003, pp. 411-421.
[23] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems,* vol. 29, pp. 1012-1023, 2013.
[24] A. H. Eden and T. Mens, "Measuring software flexibility," *IEE Software,* vol. 153, pp. 113-126, 2006.
[25] J. Radatz, A. Geraci, and F. Katki, "IEEE standard glossary of software engineering terminology," *Standards Coordinating Committee of the Computer Society of the IEEE,* pp. 1-84, 1990.
[26] J.-C. Fernandez, L. Mounier, and C. Pachon, "A model-based approach for robustness testing," in *International Conference on Testing of Communicating Systems*, 2005, pp. 333-348.
[27] D. L. Sari, A. S. Prihatmanto, and A. Harsoyo, "Analysis and design architecture Beat me Integrated System," in *International Conference on Rural Information & Communication Technology and Electric-Vehicle Technology* 2013, pp. 1-5.
[28] H. Vahdat-Nejad, K. Zamanifar, and N. Nematbakhsh, "Context-aware middleware architecture for smart home environment," *International journal of smart home,* vol. 7, pp. 77-86, 2013.
[29] D. Zubrow, "Software quality requirements and evaluation, the ISO 25000 series, ISO 25010," *Software Engineering Institute, Carnegie Mellon university,* 2004.
[30] B. McCall, Robert Grady, "ISO 9126 Software Quality Characteristics,9126-1," ed, 1991.
[31] M. Godse and S. Mulik, "An approach for selecting software-as-a-service (SaaS) product," in *International Conference on Cloud Computing*, 2009, pp. 155-158.
[32] D. Stone, C. Jarrett, M. Woodroffe, and S. Minocha, *User interface design and evaluation*, 1 ed.: Elsevier, 2005.
[33] D. C. L. Ngo, L. S. Teo, and J. G. Byrne, "A mathematical theory of interface aesthetics," *Visual mathematics,* vol. 10, 2000.
[34] M. Weiser, "The computer for the 21st century," *Scientific american,* vol. 265, pp. 94-104, 1991.
[35] Y. Kim and K. Lee, "A quality measurement method of context information in ubiquitous environments," in *International Conference on Hybrid Information Technology*, 2006, pp. 576-581.
[36] R. S. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE communications magazine,* vol. 32, pp. 40-48, 1994.
[37] S. Chapman. (2007). *Simetrics for Similarity Measure*. Available: http://www.dcs.shef.ac.uk/_sam/simmetric.html
[38] R. Ramanathan, "A note on the use of the analytic hierarchy process for environmental impact assessment," *Journal of environmental management,* vol. 63, pp. 27-35, 2001.
[39] T. L. Saaty, *Theory and applications of the analytic network process: decision making with benefits, opportunities, costs, and risks*: RWS publications, 2005.
[40] *questionaire*. Available: https://docs.google.com/forms/d/e/1FAIpQLSeit1AaSB-WoCuo-s6Lvn6fhfuPROwOrt2_9i9tmgaCUjBZbA/viewform
[41] *Google Doc*. Available: https://docs.google.com/forms
[42] Liu Wei and H. Binwen, "Design And Simulation Of An Optimum Service Selection Arithmetic In Pervasive Computing," in *International Conferance On Computational and Information Sciences*, 2013, pp. 9-12.