Towards a Service-Oriented Approach for Managing Context in Mobile Environment

Waskitho Wibisono, Arkady Zaslavsky, and Sea Ling

Caulfield School of Information Technology, Monash University
900 Dandenong Rd, Melbourne, VIC, Australia
{waskitho.wibisono,arkady.zaslavsky,chris.ling}@infotech.monash.edu.au

Abstract. The current development of context-awareness has introduced various emerging research areas to reduce complexity in developing context aware applications by applying service-oriented approach in managing context and establish context service. The establishment of context service will enable context aware systems to access and utilize context from context providers without paying necessary attention on how context information are composed and managed. Frequent changes of available context providers with different context quality are common phenomena in mobile environment. Hence, dealing with quality of context is a very important issue to provide reliable services for context management in this environment. We have identified some key requirements to establish context service and propose a service-oriented framework to facilitate context management in mobile environment. Furthermore we show our approach to deal with problem in providing appropriate context based on its quality requirements and the preferences of the corresponding context request.

1 Introduction

The proliferation of mobile computing and networking technology has led to the emergence of context-aware applications which are capable of adapting current situations in the environment without having explicit user interventions [1]. This phenomenon has highlighted the need to provide reusable support to facilitate management of context independently from applications by establishing *context service* [2].

Service-Oriented Computing (SOC) is a new computing paradigm to support application development by utilizing services as its essential element [3]. This paradigm has motivated us to use service-oriented approach to develop *context service framework* for mobile environment. The development of the service will enable context-aware applications to access and utilize context from context sources without worrying about details of context management [2][4]. Furthermore, it will enhance the reusability of context sources for multiple applications.

Dynamic changes of available context providers that generate contexts with different qualities are common phenomena in mobile environment. Similar contexts may be available concurrently; however the relevance of these contexts to context-aware applications can vary according to the individual application's requirements and their current situations in environment. Hence, dealing with context quality is also a very important issue to provide a reliable service for context-aware systems in addition to the basic context management mechanisms.

To illustrate these problems, imagine a context-aware application that help a user to find best location for the user to do outdoor activities by providing him/her recommended locations using available environmental information provided by the context service. To perform the task, the application submits necessary context requests to the *context service* and requires a periodic update of contexts from the preferred locations.

In mobile environment, particular environmental information such as temperature, wind or tide of a particular location can be published by multiple context providers. Therefore, similar context information may also be available with different quality information such as *freshness* or *distance* to the desired locations, affected by time and location of data acquisition. Furthermore, other quality information such as *precision*, *probability of correctness* and *spatial resolution* can vary based on individual data aggregation or reasoning technique being used or due to limitation of individual sensors.

In this paper, we have identified several key requirements to establish the context service and propose our service-oriented framework to manage contexts in mobile environment. We adopt ConSpaF [5], a heuristic data fusion technique for situation reasoning based on Context Spaces theory [6], as the basis to infer the *matching confidence* between available contexts in the system and the corresponding context request. The *matching confidence* represents degree that the provided context response can attain quality requirements and preferences defined by applications in their requests. Furthermore, we integrate the notion of *quality of context* into the Context Spaces model and develop our framework prototype as our initial step to establish a context service framework for mobile environment.

2 Context in Mobile Environment

Various definitions have been proposed to define *context* in current literature. However, the interpretation of context often depends on the domain in which context is utilized. Schilit et.al [7] defined important aspects of context information in mobile computing in a user-centered view, "*three important aspects of context are: where you are, who you are with, and what resources are nearby*". On the other hand, contexts can also represent very broad information coming from various and dynamic sources while similar information can be provided by different context sources with different data models [8].

Development of pervasive computing application has to deal with information which is captured from real world by sensing devices. The captured information may have different quality information which can be influenced by sensor limitation, data transformation, aggregation or brokering [9]. On the other hand, the context quality can have significant impacts to influence applications behaviors and their adaptations to the changing environment [10][11]. Therefore, managing the quality of context in addition to the general context management is a very important issue nowadays.

2.1 Quality of Context

Information about quality of context is commonly related to characteristics of sensing devices. Different devices may produce different context with different qualities. For example, an expensive device may be capable to produce information with a high precision compare to low a cost device. Furthermore physical constraint, situation of measurement, transformation process or brokering can also influence quality of the created context information [9].

Bucholz et.al [10] defined *quality of context* (QoC) as "*any information that describes the quality of information that is used as context information*". The quality of context was also described as any inherent information that can be used to determine the worth of information to the applications [9]. The notion of quality of context is different from quality of service since context information has inherent quality metric produced by context sources even when the context is not provided to clients as a service [12].

We refer to [10][11][12] to identify important attributes which determine the quality of context. They are:

- Freshness: defines time elapsed between the determination of context information and its delivery. It represents the age of information.
- Precision: defines how precise the information mirrors the reality.
- Probability of Correctness: defines the correctness probability of the provided information.
- Resolution: defines the granularity of information.
- Spatial validity : defines the spatial location in which the context information is applicable.

In the mobile environment, contexts can be generated by distributed context providers. Their quality values can be closely related to physical location and can be less valuable or even not applicable to the current situation of applications. Moreover, the relevance of context for context-aware applications can also change due to user's mobility. For that reasons, providing context with its quality information is essential.

3 Context Service Framework

There are several challenges that need to be dealt with when developing context service. In this section we describe our framework and describe its constituent components.

3.1 CS-Engine Internal Services

CS Engine is the core of our framework comprising multiple services. It is responsible for facilitating context management and providing relevant context responses for incoming context requests from applications. The services include:

- **Gateway Service**: This service is responsible for handling incoming context requests from applications. A context request can be categorized into either a *direct reply* or a *subscription-based* type of requests. For the direct-reply request, this service can directly invoke the *provisioning service* to obtain the relevant contexts from available contexts in the system. For the subscription-based, the gateway service has to subscribe it to the *messaging service.*
- **Messaging Service**: The *messaging service* has to be established to enable *event-based* interaction for the subscription-based request. This service will send a notification along with the composed context response to the *gateway service* if relevant context of the subscribed request is available.
- **Provisioning Service**: The *provisioning service* is responsible for composing context response for the corresponding context request. This response can be accompanied with *matching confidence* values to represents degrees that the provided context response can attain quality requirements and preferences have been defined in the request. In some case the provision service may need to invoke the *reasoning service* for a particular request that need a new context to be inferred from existing contexts in the systems. The *discovery service* may also need to be triggered to obtain update of the required contexts depends on availability of the registered context providers.
- **Reasoning Service**: The *reasoning service* is a service for deriving high-level context, such as to infer the occurrence of a *situation* given several contexts information by numerous context providers. This component plays a vital role in providing context in mobile environment since necessary context information may not available and context information can be imprecise or inconsistent.
- **Brokering Service**: The *brokering service* facilitates services and flexible mechanisms for context providers to submit their contexts to the system.
- **Storage Service** : This service is responsible for managing and providing services to access the context database. It can also be extended to deal with *security* or *privacy enforcement* issues.
- **Discovery Service** : The service is responsible for managing a list of available context providers in environment. It can also be extended to have capability to notify the available context providers to update their contexts in the system in necessary.

3.2 Service Composition

Figure 1 illustrates the service composition of the CS-Engine including its interaction with distributed context providers and applications. Generally, each context provider has their own data capturing component and may have different

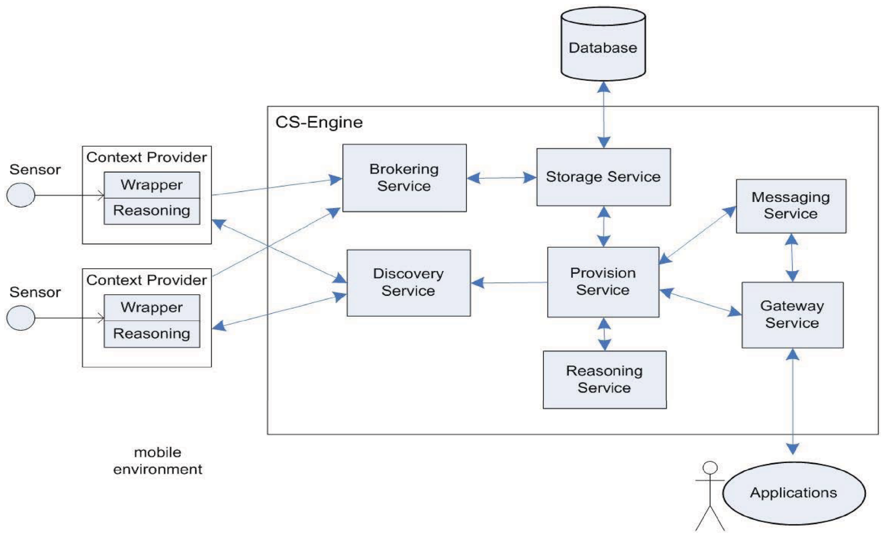


Fig. 1. CS-Engine Services Composition

reasoning techniques to produce contexts from their sensor(s). To provide their contexts to the framework, the context providers can utilize services provided by the brokering service. To obtain context from the framework, applications can submit their context requests to the gateway service and specify type of the request into subscription-based or direct-reply type. Upon receiving the request, the gateway service then decomposes the request to obtain detail of context request including the assigned quality requirements and preferences.

For the *direct-reply* request, the gateway service invokes the provisioning service to obtain context responses accordingly. For the subscription-based, the request will be submitted to the messaging service to facilitate event-based interactions for predefined criteria. Accordingly, the messaging service can organize asynchronous interactions with the provisioning service, to obtain necessary context responses. For both types of context request, the gateway service can then deliver the provided contexts to the corresponding applications.

Generally, processes to generate context response are initiated by extracting details of context request by the provisioning service. The service then gathers relevant contexts from storage service. In special cases, the provisioning service then may ask the reasoning service if it requires derivation of a new context to fulfill context request. In other cases, the discovery service can also be triggered to obtain the latest update contexts from available context providers in the directory list. The Figure 2 depicts the discussed services interaction for processing both types of context requests and the Figure 3 illustrate services interaction for composing context responses.

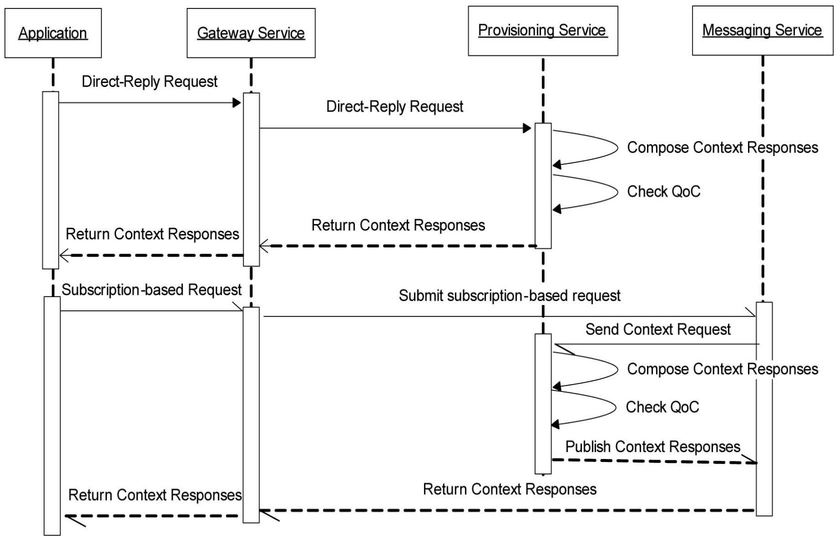


Fig. 2. Service Interaction for processing context request

4 Modeling Context and Context Request

Providing contexts responses for numerous applications requires a formal context representation to deal with heterogeneity of context sources. It will enable a service to provide uniform access of context by both context providers and applications. Furthermore, it can also help various and independent context aware applications to be developed with ease and enable collaborations among them.

Padovitz et.al.[5] proposed a general approach that use *geometrical spaces* intuition to model context and to infer situation for context-aware environment called *Context Spaces*. We use Context Spaces as the basis for context modeling, furthermore we have also developed a model of *context request* as an *object* consist of required context's class and attributes to specify quality requirements and predefined weights to specify relevance of each attributes to other attributes in the context request. In addition, individual contribution for each attribute in the request is also need to be specified. Finally, *matching confidence* values for all of relevant contexts then can be computed. These values represent degrees that the contexts can satisfy quality requirements and preferences of the submitted request.

4.1 Context Spaces

The Context Spaces[6] defines *situation* in pervasive environment as a collection of *accepted regions* in a multidimensional spaces. It also defines *context attribute* as any type of data that is used in the process of situation reasoning that can be associated with sensor data and denoted as a_i^t. As an example, a value of

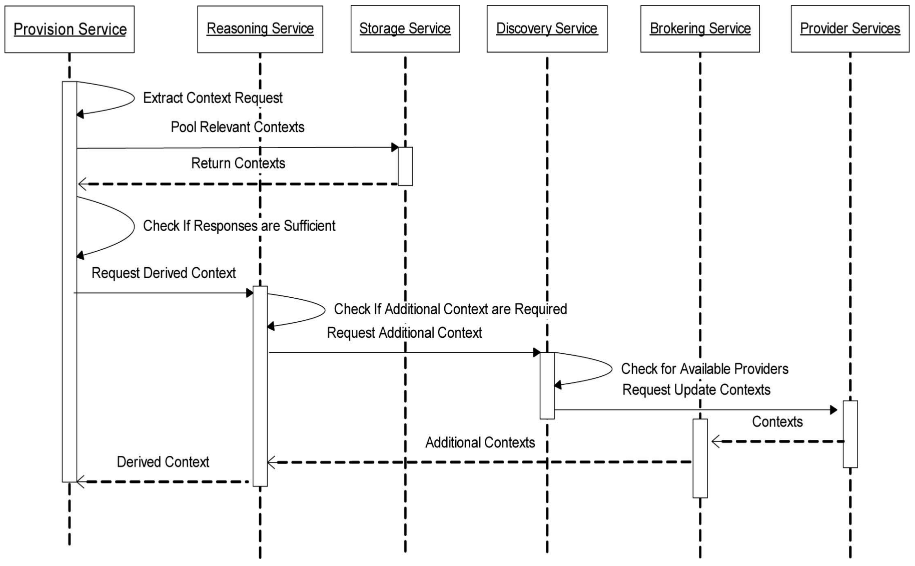


Fig. 3. Services interaction for composing context responses

sensor reading i at time t can be represented as context attribute a_i^t. A vector consists of a collection of context attributes at time t forms a *context state*. The context state is denoted as $C^t = \{a_1^t, a_2^t, \ldots a_n^t\}$ that represent a current state of application while n is defined as the number of context attributes. Context Spaces defines a real situation in context-aware environment as a *situation space* and denoted as $R_i = \{a_1^R, a_2^R, \ldots a_n^R\}$ that is a collection of n acceptable regions corresponding to a predefined situation. A region of acceptable values can be defined as a set that satisfy some predicates [5]. The Figure 4 illustrates this concept and show an example of a context state being inside $((C_i)$ or being outside (C_j) of a defined situation space R_i.

A heuristic based *data fusion* technique for situation reasoning called ConSpaF [5] was proposed to compute *degree of support* to verify situation occurrences for the Context Spaces model. The fusion approach assigns *weights* for the corresponding regions of the situation space R_i to represent the relative importance of a region to other regions to infer a situation; and *individual contributions* that specify individual support of a region that a corresponding situation is occurring. Ideas of *symmetrically/asymmetrically contributing* of the context attribute were defined. The *symmetrically contributing* attribute will increases the confidence of situation occurrence if the value of a particular context attribute is inside the corresponding region; otherwise it will decreases the confidence if it located outside the corresponding region. On the other hand, the *asymmetrically contributing* attribute will not decrease the confidence if the value of the value of the context attribute is outside the acceptable ranges of the corresponding region.

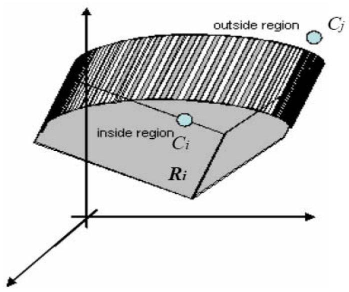


Fig. 4. Visualitation of context state and situation space[6]

4.2 Modeling Context Request

In our approach, we combine context *quality attributes* with the *context attribute* for our *context data*. It is defined as a tuple (a_i^t, Q) where a_i^t is the *context attribute* a_i^t and Q is the set of context quality attribute-values and denoted as a vector $Q = \{q_1, q_2, \ldots q_k\}$ where $k \in N$ represents the number of quality attributes for the context attribute a_i^t. To model a context request, we create an object called *request space* that represents a context request. The object is composed of the corresponding *context class* and some additional properties consist of regions that represent quality requirements including their assigned weights and individual contributions. We adopt the concept of *situation space* in the Context Spaces to compose the request space. The algorithm 1 shows steps to compose a context request in our framework as illustrated Fig.5.

4.3 Inferring Degree of Matching Confidence of Available Contexts

In mobile environment, distributed context providers may produce similar contexts with different qualities. In this sub-section, we discuss how we deal with this problem by adopting ConSpaF [5] data fusion technique to compute *matching confidence* of relevant context data with the corresponding context request concerning its quality requirements and preferences.

To measure this value, Algorithm 2 starts the process by collecting relevant context data in the system followed by extracting their quality attributes into a vector object called *ContextQualityState*. This vector represents the value of quality attributes for the corresponding context data. To compute the degree of matching confidence for each corresponding context data, the Algorithm 2 then invoke Algorithm 3. This algorithm is an adoption of ConSpaF [5] data fusion approach for situation reasoning that is utilized to infer the required matching confidence of a particular context data with the corresponding context request. Figure 7 depicts details steps of the discussed Algorithm 3.

```
-----------------------------------------------------------
Algorithm 1 : Composing Context Request
-----------------------------------------------------------
ContextRequest=new ContextRequest(ContextClass,threshold)
TotalRegionNumber=QocRequirements.totalRequirements
 for  i=1 to TotalRegionNumber
  Region[i] = new(QoCAttribute)
  for j=1 to QocRequirements[i].predicateNumber
    QocPredicate[j]= new Predicate(QocRequirement[i].predicate[j])
  end for
  Region[i].AddPredicate{QoCPredicate[1],..,QoCPredicate[j]}
  Region[i].Set(RelevanceWeight,SymmetricStatus)
  ContextRequest.AddRegion(Region[i],IndividualContribution)
  ContextRequest.SetTimeValidity(timeValidity)
 end for
ContextResponse = GatewayService.Put(ContextRequest)
```

Fig. 5. Algorithm to compose context request

5 Implementation

In this section, we discuss our initial implementation of the framework. We have simulated a number of distributed context providers and described a scenario to highlight problems in providing relevant contexts in mobile environment in a situation where multiple context providers offer similar context information with different values and quality information.

5.1 Application Scenario

To illustrate the problems, imagine a context-aware application that helps a user to find best location for the user to do outdoor activities by providing him/her the choice of best locations using updates of environmental contexts. To obtain these contexts, the application needs to compose necessary context request(s) and subscribes the request along with the quality requirements and preferences to the context service framework. In addition, the application may also need to get a periodic update of necessary context information from all of preferred locations. However, specific environmental information such as temperature, wind or tide at such a particular location in mobile environment can be generated by multiple context providers simultaneously. Therefore, similar context information may have different value of their quality attributes such as, freshness or distance to desired location that relate to time and location of data acquisition; or different values for other quality attributes as identified in section 2.1 which can be influenced by individual aggregation and reasoning technique or the limitation of sensors. In the simulation, we assume that each context provider publish their contexts within the framework along with their quality information.

```
----------------------------------------------------------------------
Algorithm 2 : Extracting QoC attributes and composing context response
----------------------------------------------------------------------
for i=1 to totalContextData do
 if ContextData[i].class == ContextRequest.ContextClass then
   new ContextQualityState
   for each QoCAttribute in ContextData[i] do
      Attribute = QoCAttribute.ExtractAttribute
      ContextQualityState.add(Attribute)
   end for
   requestSpace = ContextRequest.getRequestSpace
   MatchingConfidence =
   Algorithm3.Infer(requestSpace,ContextQualityState)
   if MatchingConfidence > requestSpace.treshold then
    Response[i]= new Response(ContextData,MatchingConfidence)
 end if
end for
Return (Response[1],...,Response[n])
```

Fig. 6. Algorithm to extract QoC attributes and composing context responses

```
-------------------------------------------------------------
Algorithm 3 : Computing Matching Confidence
-------------------------------------------------------------
for each Region in RequestSpace do
 QualityAttribute = ContextQualityState.getAtribute(Region)
 regionContribution =  getContribution( Region,QualityAttribute)
 accumulatedSupport += regionContribution * Region.getWeight
 if Region.Optional == false then
      productSupport * = regionContribution
 end if
end for
q2,q1 = ComputeCoefision(Region[1].Weight,., Region[n].Weight)
MatchingConfidence = q1*accumulatedSupport+q2*productSupport
Return(MatchingConfidence)
```

Fig. 7. Algorithm to compute matching confidence

For an example, local weather stations may publish their observations every 4-6 hours while national bureau of meteorology may update their information on daily basis. Furthermore, there may be numerous individual context providers such as private boats or sophisticated vehicles that are equipped with environmental sensors and communication system which can report occasionally weather information using cellular network or other available connection to the system. Table 1 shows examples of similar contexts produced by distributed context providers including their distances of observation to the required location and Table 2 shows the corresponding quality information we use to simulate our scenario.

As we can see from the Table 2, similar contexts can have different quality information that are influenced by factors that as we have discussed before. For instance, information from national bureau of meteorology may have high probability of correctness (PoC) but may have a low freshness due to its daily

Table 1. Available contexts and their providers

ID	Source	Class	Data(m)	Distance (m)
Provider 1	Bureau Meteorology	Tide	2.4	50000
Provider 2	Local Stations A	Tide	3	70000
Provider 3	Local Stations B	Tide	2.2	30000
Provider 4	Local Stations C	Tide	2.1	45000
Provider 5	Private Observation	Tide	1	5000
Provider 6	Private Observation	Tide	0.8	13000
Provider 7	Private Observation	Tide	1.8	8000
Provider 8	Private Observation	Tide	2.5	70000
Provider 9	Private Observation	Tide	1.2	15000
Provider 10	Private Observation	Tide	0.9	3000

Table 2. Quality attributes values of the available contexts

ID	Freshness(h)	Resolution (km)	Precision(m)	PoC
Provider 1	20	50	0.2	0.85
Provider 2	4	12	0.15	0.85
Provider 3	6	15	0.1	0.9
Provider 4	5	10	0.15	0.95
Provider 5	1	1	0.3	0.75
Provider 6	0.8	1	0.25	0.8
Provider 7	2	1	0.25	0.8
Provider 8	4	5	0.25	0.7
Provider 9	0.5	0.8	0.3	0.75
Provider 10	0.8	0.8	0.3	0.7

update mechanism. Moreover, the information has a lower resolution since it covers area of 50 square kilometers or even more. On the other hand, a certain environmental information can also varies or even change frequently in each location within smaller areas required by applications.

Distributed local weather stations may provide their contexts that have a spatial resolution between 10-15 square kilometers. However, the context information they provide can have different *precision* or *PoC* in relation to individual sensing equipments or reasoning technique they may use. In addition, *distances* and *freshness* of their observation to particular locations required by the application can vary. In some cases, numerous individual context providers such as private boats that report their observations to the system might have high freshness and spatial resolution in their information, however their information can have low PoC or precision which varies according to their sensing equipments. In some cases, they may have closer distances and higher freshness for particular locations of the application. An application need to determine criteria for each quality attributes in the context request. One example of specifying these criteria is shown in Table 3.

Table 3. Requirements for quality attributes

Freshness	PoC	Precision	Resolution	Distance
< 6 h	> 0.7	< 0.3	< 11 km	< 10000 m

Table 4. Quality attribute preferences

Parameter	Asymmetric / Optional	Importance (1-5)	Contribution
Freshness	No	5	0.95
PoC	No	3.5	0.85
Precision	Yes	3	0.75
Resolution	Yes	3	0.7
Distance	No	4.5	0.9

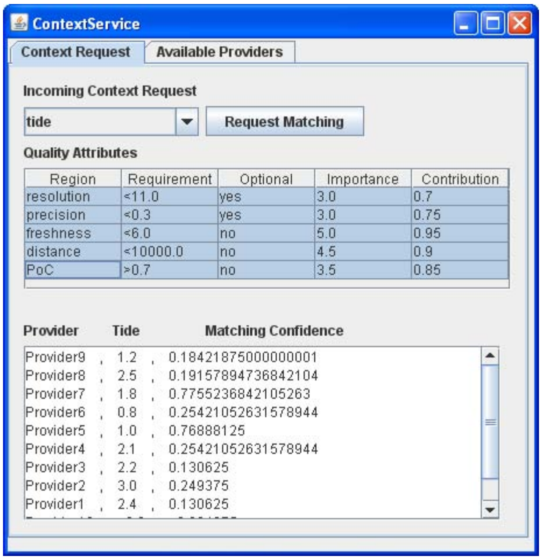
An example of quality preferences of a submitted context request is illustrated in Table 4. For each quality attribute in the context request, an application then determines weights (from 1 to 5) to represent relative importance of an attribute compared to other attributes of the context request. The individual contribution for each quality attribute is also need to be specified. It defines individual support of a quality attribute in the data fusion process to compute the matching confidence. This table shows that the application has decided that freshness and distance are more important compared to other quality attributes of the submitted context request.

5.2 Initial Implementation

We have implemented the prototype of the context service framework using *Java*. We have also simulated mobile context providers who provide their context information over network into our context service framework. A client prototype to simulate the submission of various context requests has been developed to show capability of the framework. This is shown in Fig. 8.

The sorted computation result of matching confidence between available contexts and the defined context request has been defined earlier are shown in Table 5. The table shows that context data produced by provider 7 and provider 5 have higher matching confidence for the defined context request and defined criteria compared data from other providers. Referring to the previous tables, the QoC information of context provided by provider 7 can fulfill all the requirements of the context request while provider 5 can meet all requirements except the precision requirement (< 0.3). On the other hand, information data provided by provider 1, has the lowest matching confidence since it can only attain PoC and precision requirements and fail to accomplish other quality requirements e.g. distance, freshness and resolution.

The differences in matching confidence values are also influenced by the assigned weights, individual contributions and symmetrically contributing status of the request as we have discussed before. Finally, by considering the specified



Incoming Context Request

tide

Quality Attributes

Region	Requirement	Optional	Importance	Contribution
resolution	<11.0	yes	3.0	0.7
precision	<0.3	yes	3.0	0.75
freshness	<6.0	no	5.0	0.95
distance	<10000.0	no	4.5	0.9
PoC	>0.7	no	3.5	0.85

```
Provider     Tide        Matching Confidence
Provider9  ,  1.2  ,  0.18421875000000001
Provider8  ,  2.5  ,  0.19157894736842104
Provider7  ,  1.8  ,  0.7755236842105263
Provider6  ,  0.8  ,  0.25421052631578944
Provider5  ,  1.0  ,  0.76888125
Provider4  ,  2.1  ,  0.25421052631578944
Provider3  ,  2.2  ,  0.130625
Provider2  ,  3.0  ,  0.249375
Provider1  ,  2.4  ,  0.130625
```

Fig. 8. Basic experimental run

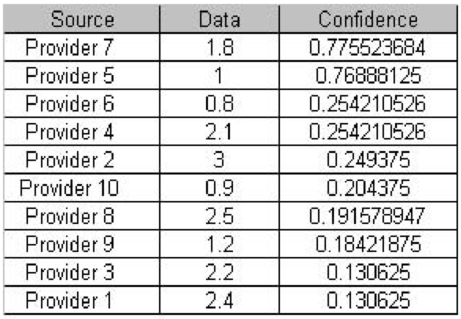
threshold of confidence level of the context request, the acceptable contexts can then be delivered to the corresponding application.

6 Related Work

There are many existing approaches of managing context in pervasive environment that relevant to our work. Context Toolkit [13] introduced widgets as an abstraction layer to hide the detail of low-level context sensing mechanisms and uses aggregators to mediate interaction between application and the widgets as well as to provide context aggregation service. Context Broker Architecture (CoBrA)[14] is an agent-based approach for supporting context aware systems in a smart space. CoBrA implements ontology as its data model and enable sharing information and for reasoning the context among the agents. Context Managing Framework (CMF)[15] uses a centralized blackboard-based approach to store and share context from any available context source terminals. The CMF incorporates ontology as the basis for its context data model and provides common vocabulary and knowledge among users.

Some existing approaches have already brought QoC issues to manage their context. CIS [4] and CMF [15] use some QoC parameters for their query interface to enable application to specify the minimum QoC of the request but they do not specify how to compose context responses based on quality attributes requirements and preference of the application if there are multiple context providers that offer similar contexts .

Table 5. Sorted computation result



Source	Data	Confidence
Provider 7	1.8	0.775523684
Provider 5	1	0.76888125
Provider 6	0.8	0.254210526
Provider 4	2.1	0.254210526
Provider 2	3	0.249375
Provider 10	0.9	0.204375
Provider 8	2.5	0.191578947
Provider 9	1.2	0.18421875
Provider 3	2.2	0.130625
Provider 1	2.4	0.130625

In this paper we integrate the notion of quality of context into the Context Spaces modeling technique and propose algorithms to provide context responses based on quality preferences and requirements of the submitted context request from applications using a heuristic data fusion technique.

7 Conclusion

Frequent changes to incurred context providers, providing varying context quality is common in mobile environment. Hence, dealing with quality of context in managing context is a very important issue to provide services for context-aware systems in the mobile environment

We have proposed an initial service oriented approach to facilitate context management in mobile environment by establishing the context service framework. Our preliminary implementation shows the capability of the framework to provide relevant context in situation where multiple context providers offer similar contexts with different quality information.

Further investigation and development still need to be done to address challenges to facilitate context management in mobile environment such as trustworthiness of context providers, context ambiguity, mobility and frequent disconnection of context providers and their impacts to quality of context. In the next phase of our research, we plan to develop service for data fusion to be incorporated with the reasoning service and enable adaptation of the discovery service to deal with these issues.

References

1. Baldauf, M., Dustdar, S.: A Survey on Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
2. Lei, H., Sow, D.M., Davis II, J.S., Banavar, G., Ebling, M.R.: The Design and Application of a Context Service. Mobile Computing and Communication Review 6(4), 45–55 (2002)

3. Papazoglou, M.P., Georgakopoulos, D.: Service-Oriented Computing. Communications of ACM 46(10), 25–28 (2003)
4. Judd, G., Steenkiste, P.: Providing Contextual Information to Pervasive Computing Applications. In: Proceeding of The First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), Fort Worth, Texas, pp. 133–142 (2003)
5. Padovitz, A., Loke, S.W., Zaslavsky, A., Burg, B., Bartolini, C.: An Approach to Data Fusion for Context-Awareness. In: Fifth International Conference on Modelling and Using Context, Paris, France, pp. 353–367 (2005)
6. Padovitz, A., Loke, S.W., Zaslavsky, A.: Toward a Theory of Context Spaces. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshop, Orlando, Florida, pp. 38–42 (2004)
7. Schilit, B.N., Adams, N., Want, R.: Context-Aware Computing Applications. In: IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1994), Santa Cruz, CA, US, pp. 89–101 (1994)
8. Broens, T., Quartel, D., Sinderen, M.V.: Toward a Context Binding Transparency. In: Proceedings of the 13th EUNICE Open European Summer School, Enschede, The Netherland, pp. 9–16 (2007)
9. Krausse, M., Hochstatter, I.: Challenges in Modelling and Using Quality of Context. In: Magedanz, T., Karmouch, A., Pierre, S., Venieris, I.S. (eds.) MATA 2005. LNCS, vol. 3744, pp. 324–333. Springer, Heidelberg (2005)
10. Buchholz, T., Kupper, A., Schiffers, M.: Quality of Context: What It Is and Why We Need It. In: Proceedings of the 10th International Workshop of the HP OpenView University Association (HPOVUA), Geneva, Switzerland (2003)
11. Sheikh, K., Wegdam, M., Sinderen, M.V.: Quality of Context and Its Use for Protecting Privacy in Context Aware Systems. Journal of Software 3(3), 83–93 (2008)
12. Huebscher, M.C., McCann, J.A.: Adaptive Middleware for Context-aware Applications in Smart-homes. In: Proceedings of the 2nd workshop on Middleware for Pervasive and Ad-Hoc Computing, Toronto, Ontario, Canada, pp. 111–116 (2004)
13. Deu, A.K., Abowd, G.D., Salber, D.: A Context-based Infrastructure for Smart Environments. In: Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE 1999), Dublin, Ireland, pp. 114–128 (1999)
14. Chen, H., Finin, T., Joshi, A.: An Intelligent Broker for Context Aware Systems. In: Adjunct Proceedings of Ubicomp 2003, Seattle, USA, pp. 183–184 (2003)
15. Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.-J.: Managing Context Information in Mobile Devices. IEEE Pervasive Computing Magazine 2(3), 42–51 (2003)