Quality of Context: What It Is And Why We Need It

Thomas Buchholz, Axel Küpper, Michael Schiffers

MNM-Team, Department of Informatics

Ludwig-Maximilians-University, Munich, Germany

[buchholz,kuepper,schiffers]@informatik.uni-muenchen.de

When people interact with each other, they implicitly make use of context information while intuitively deducing and interpreting their actual situation. Compared to humans, IT infrastructures cannot easily take advantage of context information in interactions. Typically, context information has to be provided explicitly.

Recently, cellular network operators have been showing interest in offering Context-Aware Services (CAS) in the future. For a service to be context-aware it must be able to use context information in order to adapt its behavior or the content it provides. Examples of CASs are restaurant finders, tour guides and dating services. These services will depend on the availability of context information which must be provided at the right time, in the right quality, and at the right place. The quality of this context information is neither identical to Quality of Service (QoS), nor to the quality of the underlying hardware components, i.e., Quality of Device (QoD). Rather, the precision, probability of correctness, trustworthiness, resolution, and up-to-dateness of context information form a new set of quality parameters which we call *Quality of Context (QoC)*.

In this paper, we will discuss what QoC is, what its most important parameters are and how QoC relates to QoS and QoD. These three notions of quality are unequal, but not unrelated. Based on several examples we will show the interdependence between them. We will argue that QoC as a new notion of quality is necessary to allow for the provisioning of CASs in an interorganizational manner.

1 Introduction

12 years after Mark Weiser's first formulation of the ubiquitous computing paradigm [Wei91] the world is still very different from the world Weiser envisioned. He dreamt of an omnipresent information technology (IT) infrastructure pervading everybody's everyday life supporting people in performing virtually any task in such an unobtrusive manner that the IT infrastructure does not attract any attention, but only the functionality it provides. He foresaw a complete fusion of the IT world with the real world. User interfaces would be integrated into artefacts, i.e., objects of everyday's life like chairs, ovens and refrigerators, in a very natural way. Computers would control functions of appliances, e.g., would lower sliding shutters when it gets too bright in a room. To perform its task the IT infrastructure relies on thousands

of sensors providing it with data like the temperature and brightness in rooms, the heartbeat-rate or facial expression of users, the capabilities of the user's equipment and so forth.

To formulate it in a short way: Mark Weiser envisioned a plethora of Context-Aware Services (CASs) making people's lives easier, using implicit input to determine the context in which something happens or is performed and being so well integrated into real-life objects that they can hardly be noticed by the user. A service is considered to be context-aware if it uses context information to adapt its behavior or the content it provides. According to [Dey00] context information *"is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."* Context information has a quality. Two pieces of context information that refer to the same entity, that are of the same type and were taken at the same time can differ in terms of their precision, probability of correctness, trust-worthiness etc. For example, a temperature in a room can be sensed with two different thermometers, one measuring 17 degrees Celsius with a precision of 2 degrees and the other providing a value of 18.2 with a precision of 0.5 degrees. It is obvious that the latter piece of context information is better than the former. We refer to this type of quality as "Quality of Context" (QoC).

Many research prototypes of CASs have been proposed and built [WSA+95, LKAA96, CTB+95, CD00, BMK+00]. But almost all of them had two things in common: They spanned only one administrative domain and had a small scale. Most of them were restricted to a single room or building. An example for a prototype with a rather wide coverage is a tourist guide deployed in the city of Lancaster [CDMF00]. With regard to the commercial state of the art, operators of cellular networks already offer nation-wide Location-Based Services (LBS) like very simple restaurant finders. But these applications take the location of users as a single input. Other types of context information are not considered. Therefore, these very simple LBSs do not belong to the class of CASs we consider here.

Problems that arise when CASs[1] are provided in an interorganizational manner have neither been addressed by the research community nor the mobile network operators, yet. We believe that in the future organizations will specialize on subtasks needed to provide CASs. This leads to the question how the interaction between the partners can be organized.

In Virtual Private Networks, for example, Service Level Agreements (SLAs) are negotiated between interacting partners to specify what functionality costs how much, when it is provisioned with a defined quality. QoS-parameters, fees and penalties for violating service guarantees are determined and agreed upon. We are convinced that in the area of context-aware computing these classical SLAs are not sufficient to allow for efficient co-operation between interacting partners, since there is currently no way to specify the required QoC. Therefore, we argue that current SLAs need to be augmented with additional QoC agreements whenever CASs come into play.

The remainder of this paper is structured as follows: section 2 introduces two application scenarios to illustrate what CASs could look like that might be offered via cellular networks soon. In section 3 we will propose a role model for CASs and a value chain of CAS provisioning in order to describe which role executes which tasks. In Section 4 a definition of QoC will be given and its most important parameters will be discussed. We will address the differences and the interdependence between QoC, QoS and QoD. An explanation why the new quality notion QoC is needed will follow in section 5. Section 6 presents related work before we conclude in Section 7.

[1] With CASs we will mean in the following services that take more context information into account than just location. Otherwise we will use the term Location-Based Services (LBS).

2 Application Scenarios

To illustrate how a CAS might work and to lay a foundation for our following discussion of QoC we will outline two out of a plethora of conceivable future services offered by service providers based on cellular networks: A ”dating service” and a ”restaurant finder”.

A ”dating service” allows customers to form a virtual community. Customers that subscribe to the service render their personal profile to the service provider, specifying their hobbies, interests, likes and dislikes, and give permission to be informed about events that might interest them. These profiles are kept in a database. A person wishing to get in contact with like-minded people can post events via the dating service. For example, a person on a business trip in Munich can announce that he would like to watch a certain movie in a specified cinema in Munich at 8 p.m. on next Friday. The dating service retrieves from its database persons that, based on their profile, might be interested in joining this business man. Having generated a list of candidates, the service provider asks a context provider which persons out of the list will be in Munich on the specified day. The context provider determines from the candidates' entries in their personal calendars and their current position where they will probably be on next Friday. With this information the context provider can generate a shorter, more accurate list of persons that might have an interest in the posted event. This list is given back to the dating service that informs the candidates via SMS or other means about the newly announced event. Based on this notification, the informed community members can get in direct contact with the person that generated the event, negotiating details like when and where exactly to meet etc. In this scenario, profiles, entries in calendars and location information are used as low-level context information to derive a list of persons that are interested in an event. This list can be considered to be high-level context information.

Another example is a ”restaurant finder”. Participating persons must register with the ”restaurant finder” specifying their personal preferences including price preferences, special likes and dislikes for national kitchens, importance of the vicinity of restaurants etc. When the subscriber is in an unknown area and feels hungry, he simply uses his handheld to inform the restaurant finder that he would like to eat now. This service locates him and determines which means of transportation are at his disposal and what the current traffic situation is like. From this information the service derives where restaurants must lie to be of interest for the requesting person. The restaurant finder retrieves suitable restaurants from a database and filters or sorts the list of restaurants according to the user's preferences, taking also into account the time of day and the weather situation. Thus, in the morning bakeries and cafes are preferred over restaurants and on lovely summer days restaurants providing outdoor seats are assigned to a higher rank. This ordered list is given to the user providing maps and directions to the restaurants if needed. In this service example, user preferences, location information, information about available transportation means, about the current traffic situation, time, and weather situation are used as context information.

3 Role Model for CASs

CASs, like the ones described in the preceding section, will probably be provided in an interorganizational manner, since the generation of context information, e.g., the current weather situation, does not belong to the service providers' core business. Unfortunately, interorganizational aspects of context provisioning as such have not been covered so far by research activities, though there are related areas like Service Management. In order to investigate the interorganizational aspects, especially from the point of view of QoC, we propose a role model for CASs like it is depicted in figure 1. In this model, an actor denotes an individual, organization, department, or enterprise, which offers services to other actors, which consumes services from other actors, or which does both of them. From a technical point of view, each actor autonomously operates and controls its own technical domain, consisting for example, of a number of devices, e.g., sensors, and a network infrastructure interconnecting these devices. An actor may adopt one or several roles. A role represents a certain field of activity of an actor and comprises a well-defined set of tasks an actor adopting this role has to fulfill. Relationships between roles reflect technical co-ordinations, such as protocols, interfaces, agreements concerning the pricing and quality of a particular service, and mechanisms for observing the fulfillment of these agreements.

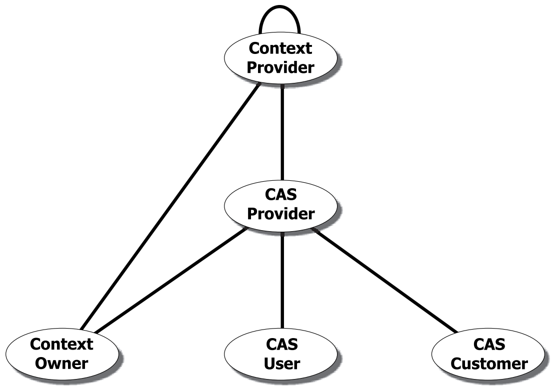

Figure 1: Role Model for Context Aware Services

The central role of our model is the CAS provider, which creates and deploys CASs and offers and sells them to a CAS customer. The CAS customer interacts with the CAS provider in order to negotiate the terms of CAS usage on behalf of one or several CAS users. The CAS provider obtains context information for service adaptation from a context provider, which is usually the operator of context sources. For example, a context provider may be the operator of a cellular network, which tracks a CAS user and delivers that user's current location to the CAS provider. In some cases, the roles of context provider and CAS user may be adopted by the same actor. For example, consider a user's mobile device which is equipped with sensors delivering context information for a CAS requested by that user [KFSB03].

For many CASs it would be desirable that a CAS user has access to context information which is related to another actor. From a security point of view, this is a very sensitive matter, because an actor must always have control about accessing and processing of her context by other actors. It is therefore inevitable to establish the role of a context owner, which represents the entity the context information is about. This context owner must be able to specify access restrictions regarding her context.

CAS providers and context providers will interact to cooperatively provide CASs. To be able to describe which role does what, we need to introduce the value chain of CAS provisioning:

Context sensing: During the context sensing phase all relevant information necessary for determining an object's actual situation is sensed. For example, an object's location may be sensed by a GPS receiver, which is integrated in her mobile device. If some information in the above scenario is stored in a database, maintained, for example, by a professional city map service, then context sensing would comprise the initial registration of the object's coordinates and their updates when the object relocates. The range of possible context sources is usually very heterogeneous and comprises any kind of sensors (physical ones and logical ones like databases).

Context refinement: Context refinement deals with the question of how to generate a concrete context information from attributes that are offered by available context sources. For example, distances could be directly sensed by distance meters. However, today's distance meters are incapable of determining the distance between two objects if there is no line of sight between them. During context refinement, the distance would therefore be calculated from two geographical coordinates and, as the user is presumably not interested in the line-of-sight distance, under consideration of a city map stored somewhere. Context refinement is a multi-step phase. In every step, previously sensed data or already derived context information is taken, and refinement techniques like combination, deduction, filtering, or inter- and extrapolation are applied to derive context information with a certain quality.

Context dissemination: After the refinement phase, context information is delivered to CASs for further usage (push-mode) or to central points at which it is stored for later retrieval (pull-mode), a phase we call context dissemination.

Context usage: Finally, the disseminated context information is used by the CAS to modify its behavior or to adapt the content it provides.

The first three steps are performed by the role context provider. We call this process ”context provisioning”. Nevertheless, CAS providers may also use context information to produce higher-level context information that they need. In the ”dating service” example, the CAS provider derived from the description of the posted event and the database filled with personal profiles a list of candidates that might be interested in joining the business man watching a movie. This list can be considered to be high-level context information. Thus, the role model does not prescribe in what way a CAS provider has to use context information. However, the role of a CAS provider in our concept never operates its own sensors and it never provides context information to other CASs. This means that, if an organization owns sensors or provides context information to other partners, one of the roles it adopts is the role of a context provider.

4 QoC: What it is

Context information possesses a quality that we call Quality of Context (QoC). In this section, QoC will be defined and the most important QoC-parameters will be outlined. Afterwards, we will draw the line between QoC, QoS and QoD and will examine the interdependence between these three notions of quality.

Quality of Context (QoC) is *any information that describes the quality of information that is used as context information. Thus, QoC refers to information and not to the process nor the hardware component that possibly provide the information*.

Our experience shows that the most important QoC-parameters are the following:

1. *Precision:* Precision describes how exactly the provided context information mirrors the reality. Precision is specified with bounds. A GPS-receiver, for example, allows for a precision of about 4 meters, while positioning users via a cellular network like GSM reaches precisions of up to 500 meters in urban areas. Nevertheless, GSM cannot guarantee to provide positioning information with this precision, since the precision is

highly dependent on the density of base stations in the respective area. Precision might be specified on the same scale like the context information or a percentage could be used.

2. *Probability of Correctness:* This parameter denotes the probability that a piece of context information is correct. Consider an organization owning a sensor network of temperature sensors. These sensors might fail and start providing wrong data, e.g., measuring 10 degrees Celsius, while the correct value is 20 degrees Celsius. With the parameter "Probability of Correctness" the original source of the context information estimates how often context information that it provides will be unintentionally wrong because of internal problems.

3. *Trust-worthiness:* Trust-worthiness also describes how likely it is that the provided information is correct. In comparison to the probability of correctness, however, trust-worthiness is used by the context provider to rate the quality of the actor from which the context provider originally received the context information. For example, context provider A sends the information that the account balance of a specified person is 100 USD to context provider B. A states that this information has a probability of correctness of 100%. Nevertheless, in the past B often received incorrect information from A. Thus, context provider B forwards this information to its customers with the remark that the source of the information is rather untrustworthy.

4. *Resolution:* Resolution denotes the granularity of information. Consider a context provider announcing that the temperature in a certain room is 17 degrees Celsius. While this is on average true, there is a hot toaster in one of the room's corners. But the context provider is incapable of offering information at a finer granularity due to the restricted number of thermometers.

5. *Up-to-dateness:* Up-to-dateness describes the age of context information. In general, up-to-dateness will be specified by adding a time-stamp to context information. Thus, a clock synchronization between the context source and the context sink is needed. Very often, it would be more interesting to know how well a formerly provided context information still accurately describes the actual situation. This can either be determined by requesting the current context information or by installing an event-service at the context source, reliably updating the context information if the current context value significantly differs from the formerly provided context information.

There may be much more QoC-parameters than the ones presented here. Some QoC-parameters may even be unique to certain types of context information. Nevertheless, based on our experience and a review of the related literature we believe that precision, probability of correctness, trust-worthiness, resolution, and up-to-dateness are the most important QoC-parameters.

After having discussed some crucial QoC-parameters, we want to point out what the differences are between QoC, QoS and QoD and how these notions of quality are interdependent. While QoC describes the quality of information, QoS refers to the quality of a service. QoC is not equal to QoS, since context information can exist without services. Consider a person knowing that Alice is not married because somebody told him a couple of months ago. Depending on its usage, this piece of information might become context information. The information has an age. Alice might have married in the meantime. And the information might have been incorrect in the first place. Thus, even without any services context information has a quality. Very often CASs cache context information that they received from context providers. When they use the information again, it is not provided by a service. Nevertheless, it possesses a quality.

QoS is any information that describes how well a service performs. Services run on underlying hardware components. These devices also possess a quality, called Quality of Device (QoD). QoD is any information about a device's technical properties and capabilities. QoD restricts the QoS and/or QoC a service using the device can reach. A GSM network, for example, can locate end-devices. This can be done by using the cell ID of the base station to which a mobile phone is currently connected or by triangulation.[2] Choosing one or the other method influences how quickly an end-device can be located (QoS) and how precise the location information will be (QoC). The distance between the base stations (QoD), however, is a factor that limits the reachable precision of the position information (QoC). Thus, QoC, QoS, and QoD are unequal. Nevertheless, they influence each other. There are two possibilities in which one notion of quality can affect another one [Dre02]:

1. *bottom-up-approach:* Impacts via the bottom-up-approach are indicated in figure 2 through arrows that point upwards. In this direction one quality has a direct technical influence on another. Like already mentioned, the distance between GSM base stations (QoD), for example, affects the precision of location information (QoC) that can be generated by the GSM network. In the bottom-up-approach a layer influences all layers above it, i.e., QoD influences QoS and QoC, while QoS influences only QoC.

2. *top-down-approach:* In the top-down-approach, qualities influence each other via the requirements they pose. If it is required, for example, to provide temperature information with a high precision (high QoC-requirements), very exact thermometers must be deployed (high QoD-requirements). Thus, QoC-requirements must be translated into QoD-requirements. These influences are illustrated with arrows that point downwards. In the top-down-approach a layer can have an impact on all lower layers. This means that QoC-requirements can affect QoS- and QoD-requirements, while QoS-requirements merely have an impact on QoD-requirements.

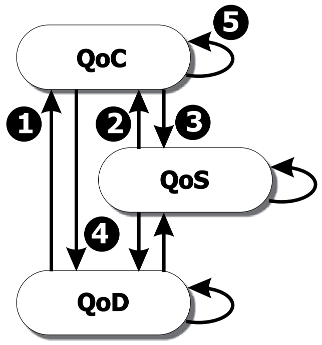

Figure 2: Interdependence between QoC, QoS and QoD

In our discussion, we will focus only on those relations depicted in figure 2 that are connected to QoC. First, we want to illustrate how QoC can directly be affected by QoD and QoS via the bottom-up-approach:

[2]Triangulation is a method where propagation time delays of three base stations, receiving signals from the end-device, are used to calculate the handheld's position.

1. QoD → QoC: A GPS-receiver is capable of positioning entities with a precision of up to 4 meters, while GSM networks are much less accurate. This shows that devices differ in their capabilities to provide high-quality context information.

2. QoS → QoC: Context information grows older during transmission times. Thus, the up-to-dateness of context information suffers from high network latencies. Very often these delays will be negligible, but for some applications even milliseconds might be important. Consider an application that tries to keep a pendulum, that is pointing upwards, in balance. The application uses the position of the pendulum as context information to move a small vehicle, on which the pendulum stands, in the direction that the pendulum inclines to, keeping the pendulum in balance. The age of the pendulum's position information is critical in this example and even transmission delays might affect the proper working of this application.

These are the direct impacts QoD and QoS have on QoC. Like we will show now, QoC can influence QoS and QoD through the requirements it poses (top-down-approach):

3. QoC → QoS: If a high up-to-dateness of the context information is required, the information must be retrieved directly from the sensors. In general, cached results will be too old. If cached results were sufficient, faster results and, thus, a better QoS would be possible.

4. QoC → QoD: QoC-requirements impact which devices will be deployed. If it is necessary to know with a precision of better than 10 meters where a person is located, positioning methods based on GSM infrastructure are inappropriate.

High-level context information must very often be derived from a couple of other pieces of context information. In this process, the QoC of input data affects the QoC of the resulting context information (5.). If the system, for example, wants to find out which task a person is currently performing, the correctness of the result heavily depends on the correctness of the inputs. The system could take a look into the calendar of the person stating that he does some sports at the moment, but the most recent location information indicates that the person is still in the office. Thus, either the calendar information is incorrect or the location information is old. Resolving this kind of ambiguity is a difficult task [MHA00].

5 QoC: Why we need it

After having illustrated in the previous section what QoC is and how QoC is related with QoS and QoD, we will address now why it is necessary to introduce QoC as a new quality notion in addition to QoS and QoD:

QoC agreements: When several actors cooperate to provide CASs, like described by the role model in section 3, there is a need for contracts that not only specify the required QoS, but that also address the QoC. In analogy to SLAs, we call contracts defining QoC-requirements QoC agreements. Note that these contracts may be negotiated together with classical SLAs and may become part of them. QoC agreements are concluded between the actors participating in context and CAS provisioning. For example, a CAS customer, subscribing to CASs on behalf of one or several CAS users, has to enter into negotiations with CAS providers in order to ensure that those use context information with a quality which is better than an agreed upon minimum. QoC agreements must also be concluded between CAS and context providers as well as between different context providers. The latter is necessary if high-level context information is derived from low-level context information. Without QoC agreements, critical parameters within the value chain would remain unnegotiated and would thus be left to the mercy of lucky chance, a situation that is clearly unacceptable for professional context and CAS provisioning. Furthermore, QoC agreements provide a basis for the pricing of context information between the involved actors.

Reconstructing CAS behavior: Context information is used to automatically adapt services or the content they provide. Therefore, the imperfection of context information has a significant impact on the experiences users make with CASs. For example, if the restaurant finder described in section 2 works on the basis of outdated weather information because better information is currently unavailable, the CAS user might be misguided, e.g., he might be directed to beer gardens although the weather is rainy and stormy. In such a situation, it would be desirable to notify the user of the quality of the used context information in order to enable him to adequately assess the quality of provided recommendations. Thus, QoC helps a user to reconstruct, interpret, and estimate the behavior of a CAS, i.e., the delivery of a certain content, the activation or termination of a CAS, or the CAS process workflow. Obviously, QoC should be made available to users only on request or if the QoC is imperfect. Otherwise, CAS users would be annoyed by receiving unnecessary background information.

Selection of appropriate context providers: In many cases, a particular context information can be generated using different value chains. These value chains can be realized by different, competing context providers, each operating an infrastructure of context sources and sub-services in order to derive that particular context information. In such a scenario, it is not unlikely that competing context providers deliver the same context information with different QoC. From the point of view of a CAS provider, QoC is then a valuable indicator to select an appropriate context provider. As an example, consider the context information ”weather conditions” used by the restaurant finder. In the future, some types of mobile devices will be equipped with sensors, which enable to derive the current weather conditions from parameters like temperature, atmospheric pressure, and humidity. In this case, the weather conditions could be directly obtained from the mobile device (the CAS user also adopts the role of a context provider in this case). Alternatively, the city of interest might be covered by numerous weather stations, which are connected by wireless local area or cellular networks. A third context provider might be a nation-wide operating meteorological service which provides weather information for each city downloadable from a database that can be accessed via Internet and that is updated once a day. This scenario is depicted in figure 3.

Obviously, the weather conditions delivered by these context providers significantly differ in their QoC. For example, weather conditions originating from the mobile device might be incorrect, because the user is currently staying indoors, or the weather conditions offered by the meteorological service might be outdated or too coarse-grained. These circumstances would be reflected by the QoC, and the CAS provider could select an adequate context provider based on the offered QoC and the according price for the context information.

Adaptation of context refinement: Like context is used to modify the behavior of a CAS, QoC can be utilized to adapt the context refinement process. Like outlined in section 3, an important task of context refinement is the derivation of new, high-level context information from low-level context information. Often, such a derivation is needed due to the unavailability of appropriate context sources. The quality of the respective low-level context information is an important indicator of whether or not the generation

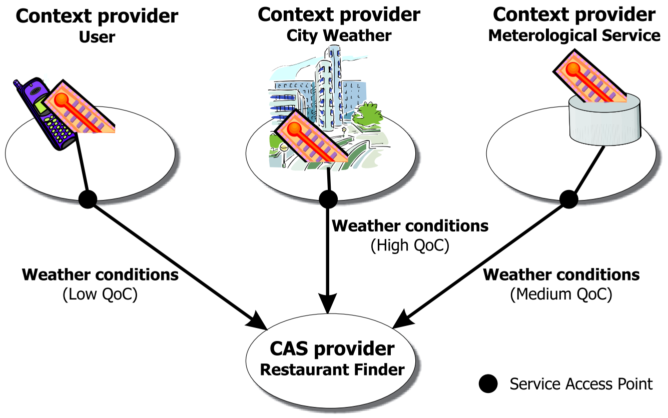

Figure 3: Selection of Context Providers based on QoC

of high-level context information makes sense at all, and, if so, how to determine the quality of the produced context information. As an example consider the context information ”distance” needed by the dating service. The distance between two users is calculated from the low-level context information ”location”. User A has a sophisticated GPS-enabled mobile device, which provides the user’s location with a high precision of approximately 10 meters. However, user B only possesses a simple cellular phone. Her location can only be obtained from her cellular network operator and corresponds to the coordinates of the base station currently serving the user. Typically, radio cells have a radius in the range of several hundred meters to some kilometers. But, the exact precision of a user’s measured location depends on the density of the operator’s base station subsystem. Like demonstrated in figure 4, the distance is derived by an independent third context provider, which has to decide whether the QoC of user B’s location is sufficient to calculate the distance. If so, the QoC of both locations are aggregated in order to determine the QoC of the distance. However, the derivation of high-level context information is only one of several tasks of context refinement. The transformation between different formats of representation or techniques like deduction, filtering, or inter- and extrapolation are affected by QoC in a similar manner.

Adaptation of context dissemination: Context dissemination comprises the distribution and storage of context information. Based on QoC agreements, a context provider can optimize these sub-processes of context dissemination. Like depicted in figures 3 and 4, a context provider makes context information available at service access points. At these access points, caching of context information can be performed. Whenever requests for context information can be answered with cached results, there is no need to execute the entire context value chain on a per-request basis. This improves QoS by lowering response times and saves costs by reducing the utilization of network bandwidth. However, caching unfavorably affects the QoC, especially the up-to-dateness of context information. Therefore, caching and revalidation strategies for context dissemination must be adapted under careful consideration of QoC, QoC agreements, and QoS aspects. As an example, consider again the calculation of the distance between two users like described in figure 4. Locating each of the users and calculating the distance is a time-consuming process. If the QoS-requirements were very tightly set, it might be infeasible to derive the distance during service provisioning on demand. On the other hand, using cached results might violate up-to-dateness requirements. In such

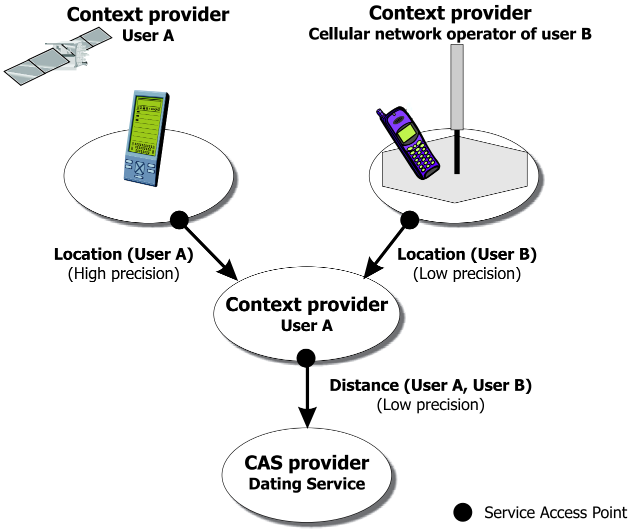

Figure 4: Influence of QoC on context refinement

a situation, the context provider must proactively update the distance stored at the service access points independently of requests for the respective context information, a process which causes high signalling costs. Thus, the determination of QoC- and QoS-requirements influences the cost situation of context providers and, therefore, has an impact on the pricing of context information.

Fine-grained privacy policies: Like described in section 3, a context owner can restrict access to his personal context. Without QoC the context owner could only determine who is allowed to access which part of his context. QoC, however, enables him to specify access policies in a much more fine-grained way. For example, a context owner might grant permission that a certain group might access his current location, but only with a precision of 10 kilometers and with a delay of some hours. Thus, QoC allows for sophisticated privacy policies.

6 Related Work

Some work has been done on the question how to deal with the QoC requirements of CASs. The most important articles that address quality parameters within the context provisioning process are [Dey00], [EHL01], [HIR02], and [GS01]. We will discuss them in turn.

[Dey00] lists some QoS-parameters considered important for the proper functioning of the context toolkit. Reliability, coverage, resolution, frequency and timeliness are explicitly mentioned. While Dey uses the term reliability in its classical QoS sense, coverage, resolution, frequency and timeliness are no QoS-parameters according to our concept. Coverage defines the set of all possible values for a context attribute. A positioning service, for example, that is only capable of locating entities in Germany possesses a coverage that spans Germany. We prefer to call this a capability of a service rather than a QoS-parameter. Resolution describes the actual change of the real-world variable that is required for the context attribute to change. This might refer to the precision of the information which will largely depend on the QoD, i.e., the capabilities of the deployed hardware, or it could describe which granularity has been chosen for the trigger conditions of event services in charge of necessary update procedures. Frequency defines how often a piece of context information needs to be updated, a parameter that we consider to be part of an update-strategy that is conceived to meet the QoC-requirements. Timeliness determines the time that is allowed between an actual context change and the related notification of the application. This parameter belongs to the domain of QoC, but does not address the existence of revalidation-strategies and the deployment of time-to-lives in conjunction with caching. Thus, though Dey addresses quality issues in the context provisioning process, he does not differentiate between QoS and QoC and furthermore mixes his understanding of QoS with parameters we would consider to be part of an update-strategy.

In [EHL01] the concept of a context service is described which is similar to our context provider. Nevertheless, the problem of multiple management domains rests unrecognized. A role model is missing. The main goal of the context service is software reuse, not disintegrating the value chain and specifying co-operation between independent actors to allow for professional large-scale CASs like in our concept. [EHL01] uses the notion of "quality of context information (QoI)" very much in the sense in which we use QoC. But the relation between QoC, QoS, and QoD is left unexplored. The need for augmenting SLAs with QoC agreements is not addressed. Thus, though [EHL01] separates context provisioning and context usage, it still keeps both parts of the value chain within one administrative domain making SLAs and QoC agreements obsolete.

[HIR02] address quality of context through tagging quality parameters to associations between entities and attributes. Each quality parameter may be described by one or more quality metrics. Although this approach is close to ours as far as quality judgements are concerned it does not take into account any underlying context role model which directly impacts any practical QoC handling. It also lacks the ability to differentiate between sensor related QoD parameters, service oriented QoS parameters and context information related QoC parameters. Especially, it is not clear how QoC may be related to context information dependencies.

[GS01] include information quality as (besides others) one type of meta-information. They identify six quality attributes: coverage, resolution, accuracy, repeatability, frequency and timeliness. However, their quality model only considers aspects of the requirements analysis and the exploration of design issues for context-aware applications. Neither do they adequately cope with problems in multi-provider environments, nor do they address the interdependence between QoD, QoS and QoC.

7 Conclusion and Outlook

In this paper, we argued that large-scale CASs will be provided in an interorganizational manner in the future. We proposed a role model and a value chain of CAS provisioning. Based on this, a discussion followed which role performs which tasks. We defined QoC, explained its most important parameters and addressed the differences and interdependence between QoC, QoS and QoD. Several reasons were outlined why the new quality notion QoC is needed. We pointed out that efficient cooperation between CAS providers and context providers and among context providers is impossible without QoC agreements.

In the future, we will evaluate in how far concepts from service management can be used for context provisioning and we will develop the presented concept into a framework. For this, a classification of context information will be needed, since QoC-parameters may differ depending on the class of context information they refer to. We will specify which structure and content QoC agreements should have and will prove the applicability of these agreements by implementing several cooperating context providers and CASs using them. Overall, this article is intended to be a starting point for a lot of research that still needs to be done.

8 Acknowledgement

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering and Prof. Dr. Claudia Linnhoff-Popien is a group of researchers at the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its web-server is located at http://wwwmnmteam.informatik.uni-muenchen.de.

References

[BMK+00] Barry L. Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven A. Shafer. Easyliving: Technologies for intelligent environments. In *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUK2K)*, pages 12–27, Bristol, UK, September 2000. Springer-Verlag.

[CD00] Deborah Caswell and Phillippe Debaty. Creating web representations for places. In *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUK2K)*, pages 114–126, Bristol, UK, September 2000. Springer-Verlag. CoolTown.

[CDMF00] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of Developing and Deploying A Context-Aware Tourist Guide: The GUIDE Project. In *Mobile Computing and Networking*, pages 20–31, 2000.

[CTB+95] Jeremy R. Cooperstock, Koichiro Tanikoshi, Garry Beirne, Tracy Narine, and William Buxton. Evolution of a reactive environment. In *Proceedings of the 1995 ACM Conference on Human Factors in Computing Systems (CHI '95)*, pages 170–177, Denver, CO, May 1995. ACM.

[Dey00] A.K. Dey. *Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[Dre02] Gabrijela Dreo Rodošek. *A Framework for IT Service Management*. Habilitationsschrift, Ludwig-Maximilians-University, Munich, Germany, 2002.

[EHL01] Maria Ebling, Guerney Hunt, and Hui Lei. Issues for Context Services for Pervasive Computing. In *Proceedings of Middleware'01, Advanced Workshop on Middleware for Mobile Computing*, Heidelberg, Germany, November 2001.

[GS01] Phil Gray and Daniel Salber. Modelling and Using Sensed Context Information in the design of Interactive Applications. In *Proceedings of the 8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI 01)*, Toronto, Canada, May 2001.

[HIR02] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling Context Information in Pervasive Computing Systems. In *Proceedings of the First International Conference on Pervasive Computing (Pervasive 2002)*, volume 2414 of *Lecture Notes in Computer Science*, pages 169–180, Zurich, Switzerland, 2002. Springer.

[KFSB03] Axel Küpper, Florian Fuchs, Michael Schiffers, and Thomas Buchholz. Supporting Proactive Location-Aware Services in Cellular Networks. To appear in the Proceedings of the 8th Conference on Personal Wireless Communication (PWC 2003), Venice, Italy, September 2003.

[LKAA96] Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson. Rapid prototyping of mobile context-aware applications: The cyberguide case study. In *Proceedings of the 2nd ACM International Conference on Mobile Computing and Networking (MobiCom '96)*, pages 97–107, White Plains, NY, November 1996. ACM.

[MHA00] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *Proceedings of CHI 2000*, pages 368–375, 2000.

[Wei91] Mark Weiser. The computer of the 21st century. *Scientific American*, September 1991.

[WSA+95] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. The ParcTab Ubiquitous Computing Experiment. Technical Report Technical Report CSL-95-1, XEROX Palo Alto Research Center, Palo Alto, CA, 1995.