Modeling Context Information in Pervasive Computing Systems[⋆]

Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy

School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072 Australia
{karen, jaga, andry}@itee.uq.edu.au

Abstract. As computing becomes more pervasive, the nature of applications must change accordingly. In particular, applications must become more flexible in order to respond to highly dynamic computing environments, and more autonomous, to reflect the growing ratio of applications to users and the corresponding decline in the attention a user can devote to each. That is, applications must become more context-aware. To facilitate the programming of such applications, infrastructure is required to gather, manage, and disseminate context information to applications. This paper is concerned with the development of appropriate context modeling concepts for pervasive computing, which can form the basis for such a context management infrastructure. This model overcomes problems associated with previous context models, including their lack of formality and generality, and also tackles issues such as wide variations in information quality, the existence of complex relationships amongst context information and temporal aspects of context.

1 Motivation

The emergence of new types of mobile and embedded computing devices and developments in wireless networking are driving a spread in the domain of computing from the workplace and home office to other facets of everyday life. This trend will lead to the scenario, often termed pervasive computing, in which cheap, interconnected computing devices are ubiquitous and capable of supporting users in a range of tasks. It is now widely acknowledged that the success of pervasive computing technologies will require a radical design shift, and that it is not sufficient to simply extrapolate from existing desktop computing technologies [1, 2]. In particular, pervasive computing demands applications that are capable of operating in highly dynamic environments and of placing fewer demands on user attention. In order to meet these requirements, pervasive computing applications will need to be sensitive to context. By context, we refer to the circumstances or situation in which a computing task takes place.

⋆ The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science and Tourism of the Commonwealth Government of Australia.

Currently, the programming of context-aware applications is complex and laborious. This situation could be remedied by the creation of an appropriate infrastructure that facilitates a variety of common tasks related to context-awareness, such as modeling and management of context information. This paper addresses the former issue by presenting a model of context for pervasive computing that is able to capture features such as diversity, quality and complex relationships amongst context information. The structure of the paper is as follows. Section 2 examines the nature of context information in pervasive computing environments in order to determine the requirements that a model of context must satisfy. Section 3 characterizes related work in the field of context-awareness, and evaluates its ability to support the requirements outlined in Section 2. Next, Section 4 describes our context modeling approach, and Section 5 presents some concluding remarks and outlines topics for future research.

2 Defining Context

The term context is poorly understood and overloaded with a wide variety of meanings. Various definitions of context have been put forward in the literature, but even these offer few clues of the properties that are of interest when modeling context. In this section we explore some of the characteristics of context information, using a case study as the basis for our discussion.

2.1 Case Study: Context-Aware Communication

One of the most compelling uses of context is in communications applications. Context-aware communication has been widely researched [3–5], and therefore is reasonably well-understood. We discuss a variation of this application here in order to illustrate the nature of context information required by pervasive computing applications, and then return to this case study throughout the paper in order to illustrate our context modeling concepts.

> Bob has finished reviewing a paper for Alice, and wishes to share his comments with her. He instructs his communication agent to initiate a discussion with Alice. Alice is in a meeting with a student, so her agent determines on her behalf that she should not be interrupted. The agent recommends that Bob either contact Alice by email or meet with her in half an hour. Bob's agent consults his schedule, and, realizing that he is not available at the time suggested by Alice's agent, prompts Bob to compose an email on the workstation he is currently using, and then dispatches it according to the instructions of Alice's agent.
> A few minutes later, Alice's supervisor, Charles, wants to know whether the report he has requested is ready. Alice's agent decides that the query needs to be answered immediately, and suggests that Charles telephone her on her office number. Charles' agent establishes the call using the mobile phone that Charles is carrying with him.

The agents in this scenario rely upon information about the participants and their communication devices and channels. For each participant, they require knowledge about the participant's activities (both current and planned), the devices he/she owns, and those he/she is currently able to use. They must also know the relationships that exist between people, such as who supervises whom and who works with whom. Finally, the agents require information about the communication channels that a participant can use and the devices that are required by each channel.

Such information can be collected from a range of sources. Some information must be explicitly supplied by users, such as that concerned with the relationships between people and the ownership of devices and communication channels. Other information may be obtained by hardware or software sensors, such as the proximity of users to their computing devices. Still other information may be derived from multiple sources; for example, user activity may be partly determined by the information stored in the user's diary and partly derived from other related context, such as the user's location. As a result, context information exhibits a diverse array of characteristics, which we now discuss.

2.2 Characteristics of Context Information

In this section, we make a number of observations about the nature of context information in pervasive computing systems. These determine the design requirements for our model of context.

Context Information Exhibits a Range of Temporal Characteristics. Context information can be characterized as static or dynamic. Static context information describes those aspects of a pervasive system that are invariant, such as a person's date of birth. As pervasive systems are typically characterized by frequent change, the majority of information is dynamic. The persistence of dynamic context information can be highly variable; for example, relationships between colleagues typically endure for months or years, while a person's location and activity often change from one minute to the next. The persistence characteristics influence the means by which context information must be gathered. While it is reasonable to obtain largely static context directly from users, frequently changing context must be obtained by indirect means, such as through sensors.

Often, pervasive computing applications are interested in more than the current state of the context; for example, in our case study agents rely not only on information about current activity, but also activities planned for the future. Accordingly, context histories (past and future) will frequently form part of the context description.

Context Information is Imperfect. A second feature of context information in pervasive systems is imperfection. Information may be incorrect if it fails to reflect the true state of the world it models, inconsistent if it contains contradictory information, or incomplete if some aspects of the context are not known. These problems may have their roots in a number of causes. First, pervasive computing environments are highly dynamic, which means that information describing them can quickly become out of date. This problem is compounded by the fact that frequently the sources, repositories and consumers of context are distributed and information supplied by producers requires processing in order to transform it into the form required by the consumer; these factors can lead to large delays between the production and use of context information. Second, context producers, such as sensors, derivation algorithms and users, may provide faulty information. This is particularly a problem when context information must be inferred from crude sensor inputs; for example, when a person's activity must be inferred indirectly from other context information, such as location and sound level. Finally, disconnections or failures can mean that the path between the context producer and the consumer is cut, meaning that part or all of the context is unknown.

Context Has Many Alternative Representations. Much of the context information involved in pervasive systems is derived from sensors. There is usually a significant gap between sensor output and the level of information that is useful to applications, and this gap must be bridged by various kinds of processing of context information. For example, a location sensor may supply raw coordinates, whereas an application might be interested in the identity of the building or room a user is in. Moreover, requirements can vary between applications. Therefore, a context model must support multiple representations of the same context in different forms and at different levels of abstraction, and must also be able to capture the relationships that exist between the alternative representations.

Context Information is Highly Interrelated. In our case study, several relationships are evident between people, their devices and their communication channels (for example, ownership of devices and channels and proximity between users and their devices). Other less obvious types of relationships also exist amongst context information. Context information may be related by derivation rules which describe how information is obtained from one or more other pieces of information. In our case study, a person's current activity may be partially derived from other context information, such as the person's location and history of past activities. We refer to this type of relationship, where the characteristics of the derived information (its persistence, quality and so on) are intimately linked to the properties of the information it is derived from, as a dependency.

2.3 Context Modeling and Management for Pervasive Systems: Requirements

Having identified some of the features of context information in pervasive systems, we now address the issue of how to represent and manage this information.

One approach is to model context using existing data modeling techniques from the field of information systems, and to store and manage the information using a database management system. Alternatively, one of the object-modeling techniques commonly used by software engineers, such as UML, could be employed to construct a model of context information and to support the mapping of this model to an implementation in an object-oriented programming language. However, having explored these approaches, we suggest that they are neither natural nor appropriate for describing context.

We attempted to model the scenario of Section 2.1 using both the Entity-Relationship model and the class diagrams of UML, and experienced particular difficulties in distinguishing between different classes of context information (for example, static versus dynamic information, sensed information versus information supplied by users), representing the temporal and error characteristics of context and expressing relationships such as dependencies. We found the UML constructs to be more expressive than those provided by ER, but also correspondingly more cumbersome. As a result of our experiences, we suggest that the most appropriate approach to modeling context information is using special constructs designed with the characteristics of context in mind. In Section 4, we present such a modeling approach.

3 Related Work

Much of the work in the relatively new field of context-awareness is concerned with providing either frameworks that support the abstraction of context information from sensors or high-level models of context information that can be queried by context-aware applications. In this section, we review both of these areas of research and examine some of the shortcomings of the surveyed approaches.

Both the context toolkit [6] and the sensor architecture of Schmidt et al. [7] support the acquisition of context data from sensors, and the processing of this raw data to obtain high-level context information. The former is a programming toolkit that assists the developers of context-aware applications by providing abstract components (context widgets, interpreters and aggregators) that can be connected together to gather and process context information from sensors. The latter provides a layered model of context processing in which sensor output is transformed into one or more cues, which undergo processing to form an abstract context description comprising a set of values, each associated with a certainty measure that estimates the certainty that the value is correct.

Other work in the field of context-awareness largely ignores the issues of how context is derived from sensors, and focuses more upon modeling context information and delivering this information to applications. The goals of this work are closer to our own. The pioneering work in this area was carried out by Schilit et al. [8], who proposed the use of dynamic environment servers to manage and disseminate context information for an environment (where an environment might represent a person, place or community). The model of context used in this work was extremely simple, with context information being maintained by a set of environment variables. The Cooltown project [9] proposed a Web-based model of context in which each entity (person, place or thing) has a corresponding description the can be retrieved via a URL. This model is relatively informal; entity descriptions take the form of Web pages, which may be unstructured and intended for human (rather than application) consumption. The context modeling approach proposed by the Sentient Computing project [10] is more formal and is based upon an object-modeling paradigm. A conceptual model of context is constructed using a language based on the Entity-Relationship model, and context information is stored at run-time in a relational database. Gray and Salber present a model of context that aims to support design activities associated with context-awareness [11]. Their model is mainly concerned with capturing meta-information about context that describes features such as the format or representation of the context information, its quality attributes, source, transformation processes and actuation (the means by which it can be controlled). The model is informal, being concerned more with supporting the processes associated with the development of context-aware software, including requirements analysis and exploration of design issues, than with capturing context information in a format that can be queried by applications. The Owl context service, currently under development by Ebling et al. [12], aims to gather, maintain and supply context information to clients. It tackles various advanced issues, including access rights, historical context, quality, extensibility and scalability. Currently, only early research results have been published, and the underlying modeling concepts are not yet clear.

These context models exhibit a number of limitations. First, most lack the formal basis that is required in order to capture context in an unambiguous way and support reasoning about its various properties. The most formal models are those that underpin the Sentient Computing approach and the context processing framework of Schmidt et al.; however, these do not address all of the characteristics of context that we identified in Section 2.2. Additionally, many of the models are restricted to narrow classes of context; in particular, several support only sensed context information and its derivatives [6, 12, 11, 7]. Most also ignore temporal aspects of context, including the need to represent histories of context information, and do not address context quality [6, 10, 9, 8]. In the remainder of this paper we present a model of context that addresses these shortcomings.

4 Modeling Context Information

This section presents a collection of modeling concepts, together with an accompanying graphical notation, designed to capture many of the features of context information that are relevant to the design and construction of pervasive systems and applications. These modeling concepts provide a formal basis for representing and reasoning about some of the properties of context information that we

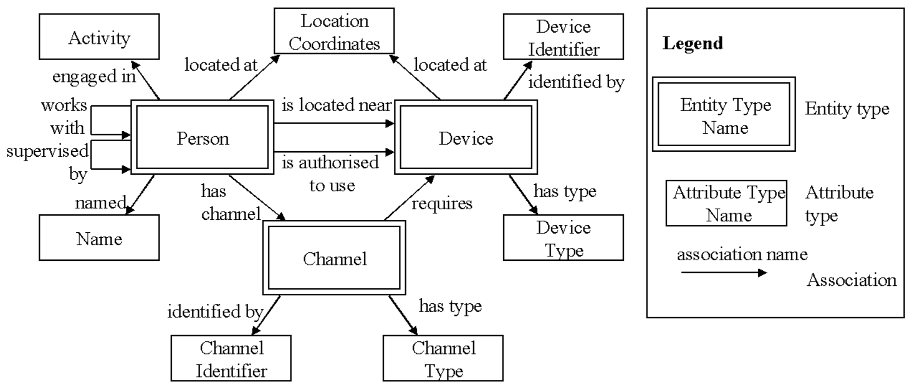

Fig. 1. Modeling the scenario of Section 2.1

identified in Section 2, such as its persistence and other temporal characteristics, its quality and its interdependencies.

The following sections present the modeling concepts incrementally, starting with the fundamental modeling concepts, and then building upon these to express more complex aspects of context. We return to the case study of Section 2.1 throughout our discussion in order to illustrate our modeling concepts by example.

4.1 Core Modeling Concepts

Our modeling concepts are founded on an object-based approach in which context information is structured around a set of entities, each describing a physical or conceptual object such as a person or communication channel. Properties of entities, such as the name of a person or the identifier of a communication channel, are represented by attributes. An entity is linked to its attributes and other entities by uni-directional relationships known as associations. Each association originates at a single entity, which we refer to as the owner of the association, and has one or more other participants. Associations can be viewed as assertions about their owning entity, and a context description can correspondingly be viewed as a set of such assertions. In the remainder of the paper, we use the terms assertion and association interchangeably.

We provide a graphical notation for our modeling concepts in order to allow context models to be specified diagrammatically. This notation takes the form of a directed graph, in which entity and attribute types form the nodes, and associations are modeled as arcs connecting these nodes. We present an example in Figure 1, based on the case study of Section 2.1, to illustrate the notation.

Our example model is constructed around three entity types: people, communication devices and communication channels. Each entity type is associated with a number of attributes: people are associated with names and activities, and channels and devices are associated with identifiers and types.

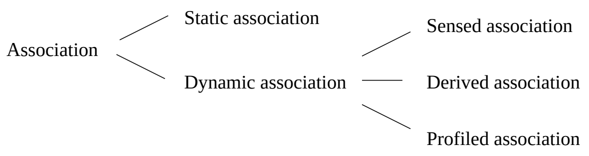

Fig. 2. A classification scheme for context associations

In addition to the associations between the entities and their attributes, several associations exist between the entities. These capture relationships between people (who works with whom, and who is supervised by whom), between people and devices (which devices each person is authorized to use and which devices are currently located with each person), between people and communication channels (which channels belong to each user), and between devices and channels (which devices the user requires in order to use each communication channel).

The model shown in Figure 1 captures the types of context information that are involved in the scenario, but does not describe many of the characteristics of this information that should ideally be known to context-aware applications and their developers. The following sections address this problem. Sections 4.2 and 4.3 present schemes for classifying associations according to type and structure. Section 4.4 describes our approach to capturing dependencies between associations, and Section 4.5 is concerned with characterizing the imperfection of context information.

4.2 Classifying Associations

In Section 2, we recognized the existence of several classes of context information that exhibit markedly different properties in accordance with their persistence and their source. We made the distinction between static and dynamic context, and showed that dynamic context can exhibit a wide range of persistence characteristics, which are linked to the means by which context information is obtained. In this section, we formalize these observations in a scheme for categorizing assertions about context, illustrated in Figure 2.

Static associations are relationships that remain fixed over the lifetime of the entity that owns them. The context captured by this type of association is typically known with a high degree of confidence, and in our example, includes the associations involving device and channel types.

Dynamic associations are all of those associations that are not static. We classify these according to source. Sensed associations are obtained from hardware or software sensors. Frequently, this information is not inserted directly into the model straight from the sensor, but is transformed in some way to bring it closer to the level of abstraction required by applications. Sensed context typically changes frequently, and consequently, can suffer from problems of staleness if there is a long lag between the time readings are taken at the sensor and the time that the corresponding context information is delivered to the client. Moreover, it can be subject to sensing errors arising from limitations inherent

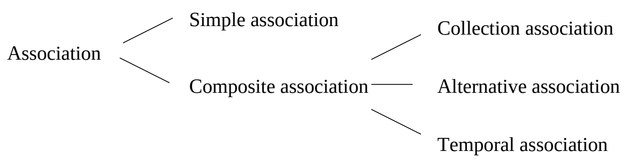

Fig. 3. Structural constraints on context associations

in the sensing technology. Two examples of sensed context in our case study are user and device location coordinates, which we assume are derived from location sensing mechanisms such as GPS receivers outdoors or Bats indoors [10].

Derived associations are obtained from one or more other associations using a derivation function that may range in complexity from a simple mathematical calculation to a complex AI algorithm. This type of context often assumes some of the properties of the class(es) of information it is derived from; for example, derived context information that is obtained from sensed information often has similar or magnified persistence and error characteristics. In addition, derived context as a class typically suffers from its own inherent limitations. In particular, derivation functions are often liable to draw incorrect or imprecise conclusions as a result of their reliance on crude inputs or overly simplistic classification models.

An example of derived context from our case study is the *is located near* relationship that, for each person, describes the set of devices located nearby. Relationships of this type need not be modeled explicitly, but can be derived for a given person by examining the *Location Coordinates* attribute of every device, and comparing it with the *Location Coordinates* attribute of the person.

The third class of dynamic association captures profiled information; that is, information that has been supplied by users. This class of information is typically more reliable than sensed and derived context and longer-lived, but can still suffer from staleness, as users may neglect to update information as it becomes out of date. Examples of profiled context include user names, and the *works with* and *supervised by* associations that exist between people.

The main benefit of classifying context information as we have described is that reasoning about information persistence and quality becomes possible. For example, conflicts can be resolved by favoring the classes of context that are most reliable (static followed by profiled) over those that are more often subject to error (sensed and derived).

4.3 Structural Constraints on Associations

Context information can vary from simple, atomic facts to complex histories. We support these different types of context by further categorizing associations according to structure, as shown in Figure 3.

An association is simple if each entity participating as owner of the association participates no more than once in this role. An example of this type of association is the *named* association of *Person*.

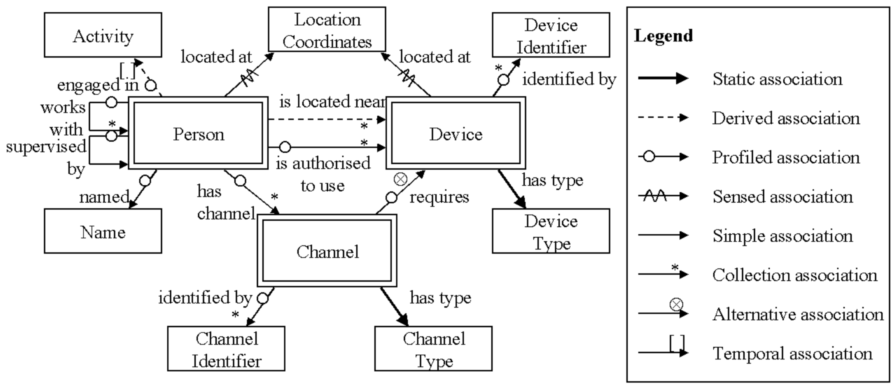

Fig. 4. Modeling the different association types for our case study

An association is composite if it is not simple. We refine composite associations into collection, alternative and temporal associations. Collections are used to represent the fact that the owning entity can simultaneously be associated with several attribute values and/or other entities, for example, people may work with many other people, and may have several communication channels. Alternatives differ from collections in that they describe alternative possibilities that can be considered to be logically linked by the 'or' operator rather than the 'and' operator. One example of an association of this type is the *requires* relationship between channels and devices. By classifying the role as an alternative rather than a collection, it acquires the semantics that a channel requires one of the devices it is associated with, rather than all. This type of association is useful when the context model must capture a number of different representations of the same information as described in Section 2.2, or when two or more sources of context information supply contradicting information and it is desirable to capture each of the different possibilities. Finally, a temporal association is also associated with a set of alternative values, but each of these is attached to a given time interval. This type of association can be viewed as a function mapping each point in time to a unique value. In our example, user activity is captured by a temporal association.

We distinguish the various different types of associations we have discussed in this and the preceding section diagrammatically by annotating the association arcs as shown in Figure 4.

4.4 Modeling Dependencies

A dependency is a special type of relationship, common amongst context information, that exists not between entities and attributes, as in the case of associations, but between associations themselves. A dependency captures the existence of a reliance of one association upon another. We say that an associa-

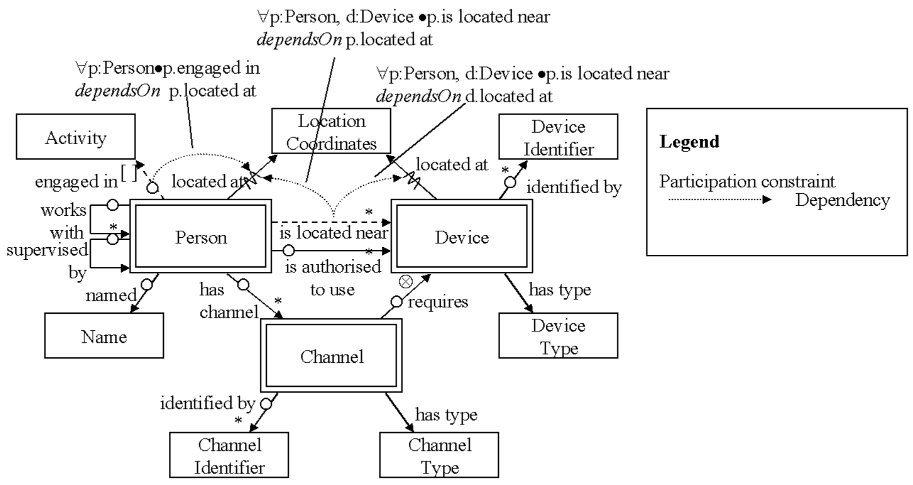

Fig. 5. Context model for our case study, showing the derivation dependencies

tion, a_1, *dependsOn* another association, a_2, iff a change to a_2 has the potential to cause a change in a_1. Each derived association is accompanied by at least one dependency; however, dependencies can also exist independently of derived associations. For example, on a mobile device a change in usage of network bandwidth can influence the battery life; that is, battery life *dependsOn* bandwidth.

The importance of capturing dependencies is pointed out by Efstratiou et al. [13]. Without knowledge of such dependencies, inappropriate decisions can be made by context-aware applications that lead to instability. Moreover, knowledge of dependencies is important from a context management perspective, as it can assist in the detection of context information that has become out-of-date.

We model a dependency, a_1 *dependsOn* a_2, as a directed arc leading from a_1 to a_2 , as shown in Figure 5. A dependency can be qualified by a participation constraint, which limits the pairs of associations to which the dependency applies. We capture three derivation dependencies in the figure. First, we show that the *engaged in* association that links people with activities is dependent upon the *located at* association. We qualify this association to indicate that associations of these two types are linked only if they describe the same person (that is, a person's activity is only dependent on that same person's location, and not on any other person's location). Similarly, we show that the set of devices located near a person is dependent on that person's location as well as the location of all devices.

4.5 Modeling Context Quality

In Section 2.2, we identified imperfection as one of the characteristics of context information in pervasive systems. Errors in context information may arise as a result of sensing and classification errors, changes in the environment leading to staleness, and so on. As context information is relied upon by applications to make decisions on the user's behalf, it is essential that applications have some means by which to judge the reliability of the information. For this reason, we incorporate measures of information quality into our model of context.

The need to address the varying quality of context information has been widely recognized [14, 15, 12, 11, 7], yet none of the existing work addresses the problem in an adequate or general way. Dey et al. suggest that ambiguous information can be resolved by a mediation process involving the user [15]. However, considering the potentially large quantities of context information involved in pervasive computing environments and the rapid rate at which context can change, this approach places an unreasonable burden on the user. Ebling et al. describe a context service that allows context information to be associated with quality metrics, such as freshness and confidence [12], but their model of context is incomplete and lacks formality. Castro et al. have a well-defined notion of quality based on measures of accuracy and confidence [14], but their work considers only location information. Schmidt et al. associate each of their context values with a certainty measure that captures the likelihood that the value accurately reflects reality [7]. They are concerned only with sensed context information, and moreover take a rather narrow view of context quality. Finally, Gray and Salber include information quality as a type of meta-information in their context model, and describe six quality attributes: coverage, resolution, accuracy, repeatability, frequency and timeliness [11]. Neither their information model nor their quality model are formally defined, as they are intended to support requirements analysis and the exploration of design issues, rather than to support the development of a context model that can be populated with data and queried by applications.

Quality modeling has been more extensively researched by the information systems community. Our modeling concepts borrow ideas from the work of Wang et al., who describe a quality model in which attributes are tagged with various quality indicators [16]. In our model, we support quality by allowing associations to be annotated with a number of quality parameters, which capture the dimensions of quality considered relevant to that association. Each parameter is described by one or more appropriate quality metrics, which represent precise ways of measuring context quality with respect to the parameter.

The types of quality parameters and metrics that are relevant are dependent on the nature of the association. For example, the quality of information about a user's location can be characterized by its accuracy, measured by the standard error of the location system, and freshness, determined by the time the location information was produced and the average lifetime of information related to user location. On the other hand, the quality of an assertion about user activity can be described by the certainty the information source has about the supplied information, measured as a probability estimate, and the overall accuracy of that information source, also described by a probability value. We illustrate the tagging of associations with quality parameters and metrics in Figure 6.

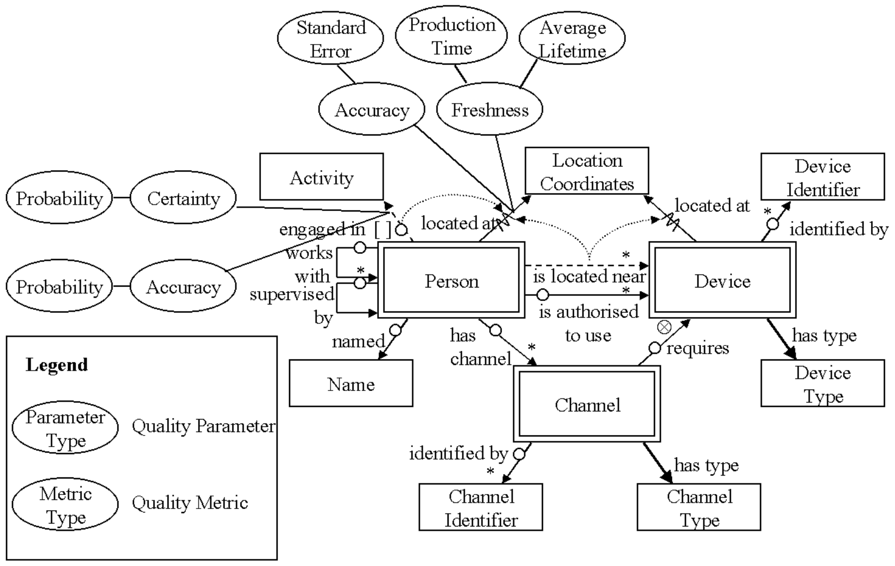

Fig. 6. Context model for our case study, annotated with quality parameters and metrics for two of the associations

5 Concluding Remarks

In this paper, we explored the characteristics of context information in pervasive systems and described a set of context modeling concepts designed to accommodate these. Our concepts were presented using the case study of Section 2.1, but are sufficiently generic to capture arbitrary types of context information, and thus to support a diverse range of context-aware applications.

We are currently in the process of developing a context management system founded upon the modeling constructs that we have presented. This system will allow abstract models described in our notation to be mapped with little effort to corresponding implementation models that can be populated with context information and queried by applications. It will be responsible for a range of management tasks, such as integration of context information from a variety of sources, management of sensors and derived context, detection of conflicting information, and so on. Concurrently, we are implementing the context-aware communication application that we have described. We have already used this case study to validate the context modeling concepts we presented in this paper, and, next, we hope to use the implementation of the case study to validate our context management infrastructure.

Aside from our implementation efforts, we envisage several areas for future work. These involve the extension of our context modeling concepts in order to address key issues for pervasive computing systems, such as privacy and distribution of context information. A privacy model is required in order to prevent abuses of context information, particularly personal information, by limiting its dissemination. Similarly, a distribution model is needed to support the appropriate partitioning and replication of context information across pervasive systems. This model must balance the requirement for a globally consistent view of context with the need for timely retrieval and continued access to information during periods of network disconnection.

References

1. Norman, D.: The Invisible Computer. MIT Press, Cambridge, Massachusetts (1998)
2. Henricksen, K., Indulska, J., Rakotonirainy, A.: Infrastructure for pervasive computing: Challenges. In: Informatik 2001: Workshop on Pervasive Computing, Vienna (2001)
3. Hong, J., Landay, J.: A context/communication information agent. Personal and Ubiquitous Computing: Special Issue on Situated Interaction and Context-Aware Computing **5** (2001)
4. Schmandt, C.: Everywhere messaging. In: 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99). (1999)
5. A, S., Takaluoma, A., Mäntyjärvi, J.: Context-aware telephony over wap. Personal Technologies **4** (2000)
6. Dey, A., Salber, D., Abowd, G.: A context-based infrastructure for smart environments. In: 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99). (1999)
7. Schmidt, A., et al.: Advanced interaction in context. In: 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe (1999)
8. Schilit, B., Theimer, M., Welch, B.: Customising mobile applications. In: USENIX Symposium on Mobile and Location-Independent Computing. (1993)
9. Kindberg, T., et al.: People, places, things: Web presence for the real world. Technical Report HPL-2000-16, Hewlett-Packard Labs (2000)
10. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: Mobile Computing and Networking. (1999) 59–68
11. Gray, P., Salber, D.: Modelling and using sensed context in the design of interactive applications. In: 8th IFIP Conference on Engineering for Human-Computer Interaction, Toronto (2001)
12. Ebling, M., Hunt, G.D.H., Lei, H.: Issues for context services for pervasive computing. In: Middleware 2001 Workshop on Middleware for Mobile Computing, Heidelberg (2001)
13. Efstratiou, C., Cheverst, K., Davies, N., Friday, A.: An architecture for the effective support of adaptive context-aware applications. In: Mobile Data Management (MDM), Hong Kong, China, Springer (2001) 15–26
14. Castro, P., Chiu, P., Kremenek, T., Muntz, R.: A probabilistic room location service for wireless networked environments. In: UbiComp 2001 Conference, Atlanta (2001)
15. Dey, A., Mankoff, J., Abowd, G.: Distributed mediation of imperfectly sensed context in aware environments. Technical Report GIT-GVU-00-14, Georgia Institute of Technology (2000)
16. Wang, R., Reddy, M.P., Kon, H.: Towards quality data: An attribute-based approach. Decision Support Systems **13** (1995) 349–372