Technical report

Context aware querying

Challenges for data management in ambient intelligence

Arthur H van Bunningen
October 2004

Data management group
Department of EEMCS
University of Twente
Enschede
The Netherlands

Contents

1 Introduction **3**
- 1.1 Considerations . . . 3
- 1.2 Outline of this report . . . 4

2 Context **5**
- 2.1 Introduction . . . 5
- 2.2 General overviews . . . 5
  - 2.2.1 A definition of context . . . 5
  - 2.2.2 Some surveys . . . 6
- 2.3 Characteristics of context . . . 7
  - 2.3.1 Introduction . . . 7
  - 2.3.2 Acquiring . . . 7
  - 2.3.3 User related . . . 8
  - 2.3.4 Consequences . . . 9
  - 2.3.5 Demands . . . 12

3 Context and databases **14**
- 3.1 Introduction . . . 14
- 3.2 An architecture . . . 15
  - 3.2.1 Introduction . . . 15
  - 3.2.2 Features of the architecture . . . 15
- 3.3 Modeling context . . . 17
  - 3.3.1 Introduction . . . 17
  - 3.3.2 Description Logics . . . 18
  - 3.3.3 Alternatives . . . 20
  - 3.3.4 Relating the model to sensor data . . . 23
- 3.4 Using the context model to process context aware queries . . . 24
  - 3.4.1 Query augmentation . . . 24
  - 3.4.2 Placing triggers on context . . . 25
  - 3.4.3 Using the context for retrieval purposes . . . 25

4 Scenarios **26**
- 4.1 Introduction . . . 26
- 4.2 The scenarios . . . 26
  - 4.2.1 The phone call . . . 26
  - 4.2.2 The document . . . 28
  - 4.2.3 Looking at pictures . . . 28
  - 4.2.4 The nearest restaurant . . . 30

4.2.5 A web service 31
4.2.6 Searching a recipe 34
4.2.7 Improving life 36
4.2.8 At the university 37
4.2.9 The airport 38
4.2.10 An implementable office scenario 39

5 Future directions **41**

Chapter 1

Introduction

1.1 Considerations

This research starts from the vision of Mark Weiser (1991), to be able to integrate computers in everyday life and by making machines that *fit human environment* instead of forcing humans to enter theirs, which will "make using a computer as refreshing as taking a walk in the woods". Weiser considers two points of most importance: scale and location. The first one so computers can be invisible, the second one to let the computer know its surroundings. Furthermore the computers should be connected. This vision is called *Ubiquitous computing* or *Ambient Intelligence (AmI)*. This last term we will use throughout this report.

Another way to look at fitting human environment is to look at the word fit, which means "to be in proper size and shape for" but also "to be appropriate to", to suit someone. Looking at this definition, for computers to be able to fit human environment they should be of the right size and *appropriate to the user*, which means to adapt to a user, a users context, to be context aware.

Most current context aware systems are small scaled and use only few context information. As a starting point we expect here is an open field for data management techniques; to introduce this scalability.

We want to develop a way to use the context of the user to optimize query processing in a way that queries give *more relevant (context aware)* results and are *executed faster*.

This research is carried out within the MultimediaN project[1] in the subproject AmbientDB. For acquiring context via sensors we try to co-operate with the Smart Surroundings project[2] and furthermore we are looking at corporation with eMAXX[3], our industrial partner in the MultimediaN project, in the field of personalization.

The goal of this report is to give an overview of the many aspects of context and, by taking into account these aspects, provide pointers where data management solutions can address the challenges of context awareness.

[1] http://www.multimedian.nl/
[2] http://wwwes.cs.utwente.nl/smartsurroundings/
[3] http://www.emaxx.nl

1.2 Outline of this report

To be able to see how data management solutions can address the challenges of context awareness we first have to determine what we mean by context. We will therefore give an overview of the different aspects of context in chapter 2. From this overview we derive challenges for data management in chapter 3. This chapter leads to an architecture and a suggested way of representing context. To verify the architecture and the way of modeling and to make the theory more concrete we will, in chapter 4, give some scenarios with their representation in the architecture and for four scenarios, a hint how they could be modeled. We end our report with a short summary of challenges to be addressed.

Chapter 2

Context

2.1 Introduction

In this chapter, we try to define what context exactly is. Because there has already been done much research in this area we will point to previous overviews in section 2.2. However, what we missed during our literature study was an extensive overview of all characteristics of context which we could use to define challenges for data management. This is the reason why we will give such an overview in section 2.3.

2.2 General overviews

2.2.1 A definition of context

One of the most used sources for definitions of context is by Dey and Abowd (1999). They describe in their paper some previous definitions and give the following definitions of context and context awareness:

Context (Dey and Abowd 1999) Context 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 application, including the user and applications themselves.

Context Aware (Dey and Abowd 1999) A system is context aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Gray and Salber (2001) build upon this definition to derive a definition for *sensed context*, where they note that the notion of 'interaction' in the definition of context from Dey and Abowd is ambiguous. They arrive at the following definition:

Sensed context (Gray and Salber 2001) Sensed context are properties that characterize a phenomenon, are sensed and that are potentially relevant to the tasks supported by an application and/or the means by which those tasks are performed.

Lieberman and Selker (2000) take a broad definition from the side of application programming were they notice that traditionally the field of computer science has taken the opposite position: the search for *context-independence*. Their definition is as follows:

Context (Lieberman and Selker 2000) Context can be considered to be everything that effects the computation except explicit input and output.

These definitions help us to get a feeling of what context is. But also make clear that context is a very broad concept. Dourish (2004) says it like this: "'Context' is a slippery notion. Perhaps appropriately, it is a concept that keeps to the periphery, and slips away when one attempts to define it." Dourish in his article mainly objects against seeing context as something which is independent from an activity and he has a point when he for example says that during a conversation the location of the conversation could turn from "context" to "content" when talking about it.

In this report we see context in the broadest sense; *everything which could be related to a certain event.* As this is almost everything, when you read in this report about techniques for sensing, modeling and using context information, you should also be able to apply these techniques for sensing and modeling the world. What *is* important in our definition is the word *related* which suggests that relating concepts is an important property of context.

2.2.2 Some surveys

In this report we will not give a complete survey of context and context aware applications. The reason for this is that there are already many good ones available.

For example Chen and Kotz (2000) give a general survey of research in context awareness where definition, applications, modeling, sensing and an architecture are discussed. They also point to the paper of Dey and Abowd (1999) who next to the defining context as mentioned in section 2.2.1 also survey previous applications and classify them based on used context types (activity, identity, location and time) and context aware features (presentation, automatic execution and tagging). Schilit, Adams, and Want (1994) give an overview of the influential work at Xerox Parc. They describe four categories of context aware applications as listed in table 2.1. Mitchell (2002) surveys applications and

	Manual	Automatic
Getting information	Proximate selection & contextual information	Automatic contextual reconfiguration
Doing a command	Contextual commands	Context-triggered actions

Table 2.1: Context Aware Software Dimensions of Schilit, Adams, and Want (1994).

clusters on the different stages context goes through; context capture, context selection and using context. Korkea-aho (2000) clusters on sorts of applications and lists some frameworks. Brown, Burleson, Lamming, Rahlff, Romano, Scholtz, and Snowdon (2000) try to look at some killer applications for context awareness of which they find seven types. Special attention deserves the survey

of Meyer and Rakotonirainy (2003) which focus on context aware homes. A current overview of applications can be found on the website of Rehman (2001) [1].

2.3 Characteristics of context

2.3.1 Introduction

In this chapter we will give an overview of the different characteristics of context information based on some previous work and own experience. With this overview we hope to make clear where data management systems for use in AmI scenarios would differ normal data management systems. We tried to cluster the characteristics in three categories:

Acquiring Characteristics related to the acquisition of context.

User related Characteristics which focus on the demands of the user.

Consequences Characteristics which follow from the previous two.

We will also include a fourth category which not really encompasses characteristics but some demands for context aware systems which immediately follow from these characteristics. We will call this category **Demands**. The overview is based on several sources which contain smaller overviews of which the most extensive are by: Gray and Salber (2001) with respect to sensed context, Goslar and Schill (2004) who cluster the characteristics in characteristics of pervasive computing, characteristics of contextual data and requirements for a representation format for contextual information and Henricksen, Indulska, and Rakotonirainy (2002) who give the most extensive overview.

For each characteristic will give a short explanation together with some pointers to relevant research.

2.3.2 Acquiring

The most important characteristics of context information follow from how it is acquired (sensed). As Satyanarayanan (2001) and Strang (2004) note, *Pervasive* or *Ubiquitous Computing* evolved from *Distributed Computing* to *Mobile Computing* to its current state, from which follow naturally that this acquiring takes place by distributed sources and should be able to deal with moving objects.

Sensed

The first characteristic as mentioned by Henricksen, Indulska, and Rakotonirainy (2002) and Gray and Salber (2001), is that much context is sensed through sensors or sensor networks, for example location or temperature. Data management solutions in this field focus on seeing the sensor network as a database for which Lazaridis, Han, Yu, Mehrotra, Venkatasubramanian, Kalashnikov, and Yang (2004) list some architectural requirements. The most notable existing systems are Cougar (Bonnet, Gehrke, and Seshadri 2001) and TinyDB (Madden, Hong, Hellerstein, and Franklin 2004).

[1] http://www-lce.eng.cam.ac.uk/∼kr241/html/101_ubicomp.html

Cheap, constrained devices (energy costs)

What adds to the challenge is that this sensing is done by cheap and (therefore) constrained devices. Cherniack, Franklin, and Zdonik (2001) mention the limited computing power of such devices, the difficulty to run applications on this low level and their unreliability. Satyanarayanan (2001) goes in more detail about energy costs and energy management.

Distributed sources

As an important aspect mentioned among others by Henricksen, Indulska, and Rakotonirainy (2002), Dey, Abowd, and Salber (1999) and Goslar and Schill (2004) is that contextual data will come from distributed sources. To get from these distributed sources, Dey (2001) used aggregators to gather context about an entity (e.g. a person) but sensor querying techniques, as mentioned before, also deal with this characteristic. This characteristic is one of the sources of the high interrelatedness discussed in section 2.3.4.

Continuous

As a crucial property of many sorts of context, Jones and Brown (2004) considers continuance, the constantly changing of the user's context which may lead to new actions (*proactiveness*).

Mobile, moving

Very related to this last property is the final characteristic of context acquirement, also mentioned by Jones and Brown (2004); the mobility of the objects for which context is acquired. This leads among others to techniques for indexing moving objects as mentioned by Cherniack, Franklin, and Zdonik (2001), the importance of spatial properties, as we will discuss in section 2.3.4, and dynamic connections (also in section 2.3.4). More information about mobile information access is given by Satyanarayanan (1996).

2.3.3 User related

The following characteristics can be seen as requirements of a user, things which users expect from a context aware system, some of which are listed as "other requirements" by Cherniack, Franklin, and Zdonik (2001). Most of them fall under the so-called "ilities", which are reliability, availability, maintainability, security, responsiveness, manageability, scalability etc. We will not discuss all of them but list five important ones for context aware systems.

Privacy and security sensitive

The most mentioned requirement in discussions about context with other researchers was about the privacy of the user. But also Newman, Eldridge, and Lamming (1991) did notice that during their experiments with tracing users during the day with badges, users did not wear them because of privacy issues. More research about the perceptions of users to these issues has been done by Kindberg, Sellen, and Geelhoed (2004). A very inspiring way of dealing with

this issue is mentioned by Gandon and Sadeh (2004), who use both access rules and obfuscation rules to deliver different context information to different users of context, such as other people or applications. Previous work on this, focused on the location of the user, has been done by Leonhardt and Magee (1998).

Proactiveness

Proactiveness means to process information on behalf of the user so an action can be taken without requiring their attention. This means knowing what a user would want to do with this information and to detect patterns in his or her behavior. It is one of the most important requirements for the ambient to be intelligent. Among other it is discussed by Jones and Brown (2004) and Tennenhouse (2000) even coins the new term *proactive computing* which stands for "the movement from human-centered to human-supervised (or even unsupervised) computing". We will discuss our approach to deal with proactiveness in section 3.2.2.

Tractability

Tractability means that a user can see why something (proactive) happened. Ideally we would like the proactiveness to be understandable and controllable by the user as suggested by DeVaul and Pentland (2000). Also from the ubiquitous computing point of view, it could be argued that a human should be able to know what is happening in the background. Chalmers (2004) makes this clear by saying that, in ubiquitous scenarios, it should also be possible to focus on the tool (the computer) to have it "present-at-hand". An example here could be the dashboard of a car, at which, in case something goes wrong can have the car present-at-hand. Another example is the network signal indicator of a mobile phone. Some research about how users would be able to do this configuration using natural language techniques is described by Weeds, Keller, Weir, Wakeman, Rimmer, and Owen (2004). We will return to this characteristic in section 3.2.2.

Mis-use from users

A last characteristic which is maybe not so relevant but is very specific to context awareness, is that users will probably "mis-use" the intelligence of the ambient. As an example Albrecht Schmidt mentioned the automatic door-opening where we know we can open the door by waving our hand before it, and can mis-use it to keep the door open for someone else, although we will not go through it ourselves.

2.3.4 Consequences

Dynamics of connections

Because we have constrained sensors and mobile objects, we have to deal with dynamic connections; connections can be lost when a sensor is out of reach or temporary unavailable, and have to be re-established when the information is available again. It could be possible that when a sensor is unreachable the

context this sensor provided could be acquired via another sensor or combinations of sensors. Goslar and Schill (2004) suggest because of this that a context database should store how to read values and not the current values itself. DeVaul and Pentland (2000) use a "dynamic decentralized resource discovery framework" which uses semantic descriptions which are registered by the different components at a directory registration service when they are available and are unregistered when they are not available anymore.

Another option is by looking at a goal-oriented approach as done by Saif, Pham, Paluska, Waterman, Terman, and Ward (2004) and Look and Peters (2003) where by using high level goals to acquire certain context and having information of what services are available and what they provide one can combine the different available services and abstract from the actual sensors. Another approach by Michahelles, Samulowitz, and Schiele (2002) uses a self-organized sensor network where autonomous units work together to provide the context of an object.

We have to keep this this characteristic in mind when considering database techniques, because for example the optimization techniques of Deshpande, Guestrin, Madden, Hellerstein, and Hong are based on the assumption that the network topology only changes slowly, so these techniques might be less applicable for AmI scenarios.

Dynamics of context

Not only the acquisition of context is dynamic, but the context of an object can also change very quickly. This is among others mentioned by Henricksen, Indulska, and Rakotonirainy (2002) and is the reason why temporal properties are important which we will discuss later in this section.

Highly interrelated

Not only does high level context depend on low level context but also different sorts of context, for example the temperature of the room and the amount of people in the room, depend on each other. This interrelatedness makes it possible to predict some context parameters based on others. This is among others discussed by Henricksen, Indulska, and Rakotonirainy (2002) and Deshpande, Guestrin, Madden, Hellerstein, and Hong (2004) uses this interrelateness to do some optimizations over TinyDB by using correlation between voltage and temperature. However, as noted by Goslar and Schill (2004), because contextual data structures are so highly interconnected we have to take care that they are not too complex for limited capabilities of human users or local devices. As a solution they suggest to break the data structures down in smaller parts.

Alternative representations

Because we have so many context information which will be acquired from different sensors and could be even from different domains, we will also have a large number of alternative representations. An example how to take care hereof can be by the method of Bressan, Fynn, Goh, Madnick, Pena, and Siegel (1997) who use Prolog rules to convert between different representations.

Different properties

Some context changes quickly (e.g. location) other context slowly (e.g. preferences). Henricksen and Indulska (2004) here distinguish between four typical properties as seen in table 2.2. These types differ both on how dynamic and on

Type	Source
Sensed	Physical and logical sensors
Static	User/administrator
Profiled	User (directly or through applications)
Derived	Other context information

Table 2.2: Properties of context information of Henricksen and Indulska (2004).

how reliable they are.

Imperfectness, uncertainty

Because of the dynamics, the constrained devices, the distributed sources, alternative representations etc. there is almost one hundred percent chance that acquired context information is not perfect. Henricksen and Indulska (2004) even characterize four types of imperfect context information:

- unknown,
- ambiguous,
- imprecise, and
- erroneous.

Imperfectness can lead to fuzzy situations where it is for example unclear in which room a person is. Grimm, Tazari, and Balfanz (2002) therefore introduce *fuzzy situation descriptions* in ontologies. Gu, Pung, and Zhang (2004) and Ranganathan, Al-Muhtadi, and Campbell (2004) both provide a modeling solution for uncertainty by adding a probability predicate and give some references to earlier work. They both refer to Dey, Mankoff, and Abowd (2000) who describe an simple architecture for incorporation imperfectly sensed context and both use Bayesian networks for reasoning about dependencies between context events.

Korpipää, Mäntyjärvi, Kela, Keränen, and Malm (2003) mention uncertainty as well but also combine it with fuzzy situation descriptions, for example *Cold*, *Normal* or *Hot*, and the chances that a situation is like this. According to research by Antifakos, Schwaninger, and Schiele (2004) it does help to display information about the amount of imperfectness to a user when it is used to make decisions.

"Unpredictable"

Next to uncertainty about the current context, we are even less sure about the upcoming context, as described by Cherniack, Franklin, and Zdonik (2001). This means that it is more difficult to do optimization and caching. However because of the high interrelateness it should not be impossible.

Temporal

Because of the dynamics in context temporal data is very important which is among others confirmed by Henricksen, Indulska, and Rakotonirainy (2002). It is therefore not surprising that Chen, Finin, and Joshi (2003b) include a paragraph how to model temporal data and ter Horst, van Doorn, Kravtsova, ten Kate, and Siahaan (2002) introduce the notion of spacetime to reason about context. We will return to these modeling questions in section 3.3.

Spatial

Next to temporal, also spatial information is important. Koile, Tollmar, Demirdjian, Shrobe, and Darell (2003) for example use "activity zones"; regions in which the same activities occur, to trigger certain events. Harter, Hopper, Steggles, Ward, and Webster (1999) describe a context aware application which especially focuses on the users location using *Bats* (an ultrasound position determination system) and Chen, Finin, and Joshi (2003b) introduce next to an ontology for temporal data also one for spatial data.

For acquiring location information a survey with different techniques has been done by Hightower and Borriello (2001) after which the authors specialize on particle filters for location estimation with ultrasound, infrared and WiFi in (Hightower and Borriello 2004).

2.3.5 Demands

In this section we will describe other demands on context aware systems which follow from the previous characteristics but are not really characteristics themselves.

Inference, derivation

Because context can be derived from other context. We must use some kind of inference or derivation to do this. Schmidt (1999) is one of the first who does so by using so called *cues* which takes the value of one sensor and provides a (sub)symbolic output. Korpipää, Koskinen, Peltola, Mäkelä, and Seppänen (2003) considers Bayesian networks to recognize high level context. Combined with the temporal characteristic, research from Höppner (2003) could become relevant who describes some techniques to discover patterns in time-series. Most of our research in this area still has to be done.

Metadata

In section 2.3.4 we saw that context can have different properties. Because of these we will get information about the context information, so called metadata. Gray and Salber (2001) give an overview of the metadata attributes which they include. These are, among others:

- forms of representation,
- information quality,
- sensory source,

- interpretation (data transformations), and
- actuation (for example to shut down faulty sensors)

Storage and logging

Because we want to be proactive and detect patterns in the behavior of the user, context has to be stored. Meyers and Kern (2000) give an overview of different storage questions (where, how, what etc.). An argument for not storing the information at sensor level is because in that case we store too detailed information which causes both too much capacity and makes it harder for detecting patterns.

Chapter 3

Context and databases

3.1 Introduction

In this chapter we will focus on the connection between context and databases, considering the characteristics mentioned in chapter 2.

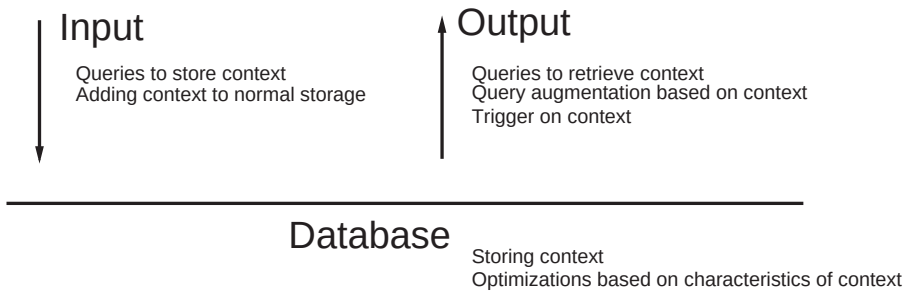

Figure 3.1: Context in a database

In figure 3.1 we show the three points where context can relate to databases;

Input How does context enter the database? Note that as we saw in section 2.3.2, this can be very close to the sensors.

Database How to store context information? Can we do special optimizations based on the characteristics of context? Do we need other query processing methods?

Output How to use context to adapt the queries to the context? This is also the place of *triggers*; a special sort of queries which are used to trigger an action when satisfied.

In this chapter, we will first look how this database view of context can relate to an architecture of a context aware environment in section 3.2. To be able to address the three points mentioned above, we first need a way to model the context, taken into account the characteristics from section 2.3, we will do this in section 3.3. In section 3.4 we will show how we can use this model for intelligent queries.

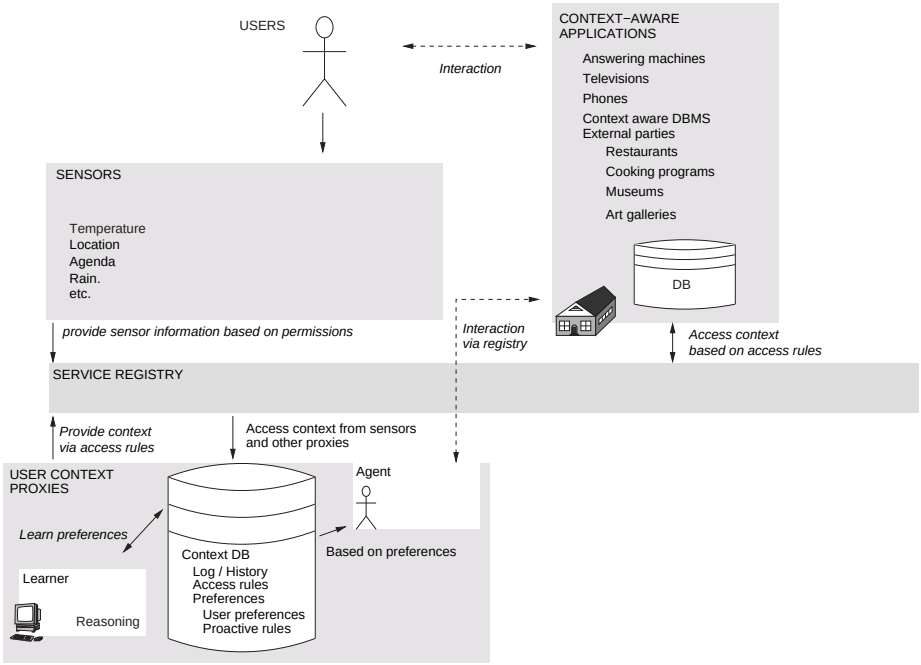

Figure 3.2: An architecture

3.2 An architecture

3.2.1 Introduction

To get more grip of the relation of data management to context awareness we will in this section describe an architecture in which we relate the two together. In chapter 4 we will show for some scenarios how they relate to this architecture. Based on this architecture a prototype will be made, which includes sensor information to do context aware querying. The scenario which focuses most on this prototype is given in section 4.2.10.

Of course many architectures are already defined in literature, especially Judd and Steenkiste (2003) give a good overview. Architectures which are related to our project are from Feng, Apers, and Jonker (2004) and Boncz and Treijtel (2003). Our architecture will be based on the one from Gandon and Sadeh (2004), among others because of their good handling of privacy issues. We will in future research look how to integrate these different structures.

3.2.2 Features of the architecture

Service registry

The basic component of the architecture is the service registry. It is via this registry that external parties and users can acquire context information from different sensors and other users. Example techniques such as are Jini or web services, as described by Gandon and Sadeh (2004), can be used. The main

features of the model of Gandon and Sadeh are that context can be acquired by applications independent of the sensors, respecting the users privacy and in a semantic model. Our use of context services in our model is the same as in their model; service rules explain what piece of knowledge the service can produce, the knowledge needed for calling the service and the function to trigger with its parameters.

Context proxy

Context information of a user is stored in a context-proxy, so it can be accessed also if the user is not connected to the network. It is also this proxy which other devices or users must contact to get context information. The proxy is also the place where information like preferences of a user and a log of his actions with their context is stored. The discovery from low level context information ("there are three people in the room") to high level information ("there is a meeting in this room") will also be done at the proxy because in this way also external information like a users agenda can be taken into account.

Sensors/actuators

The coupling between the sensors and the proxy takes place via the service registry. There are services which are responsible for a certain contextual parameter, such as location or temperature and multimodal services, which by the combination of several context parameters can for example provide more reliable information.

The sensors are also responsible for acquiring the metadata as we mentioned in 2.3.5. Since this is the point were have to collaborate with the Smart Surroundings project we discussed about some metadata which should be available from these modules:

- Accuracy information for the measurement,
- time information for the measurement,
- possibility to add "requested accuracy information" with request, to weigh energy cost and accuracy against each other and
- the possibility of subscribing to a module.

Other lists of possible metadata are given by Michahelles, Samulowitz, and Schiele (2002) who considers feature ID, feature value, sensor type ID, Smart-It ID, sensor location and time-stamp and Judd and Steenkiste (2003) who considers accuracy, confidence, update time and sample interval.

External parties

In our model we explicitely place context aware applications in the same category as other external parties such as restaurants because they will both make use of the context of the user and other external parties. Contrary to Gandon and Sadeh (2004) we do however not have task-specific agents (eg. a restaurant agent which proactively looks for a restaurant based on the users context), they are replaced partly by the *learner and agent* construction and furthermore replaced by specific context aware applications.

An example of an external application could be a context aware multimedia database which stores all videos and scenes you watched. When you would pose to this database the query "Which scene of The Bourne Identity did I watch yesterday before going to the super market". The database could, if allowed, query your proxy for the time you went to the supermarket and based on this present you with the right scene.

Learner and Agent

The claim is that our model "promotes" learning. This is because we store the context and which actions were taken in a log and it is possible to be able to "learn" new "preferences" out of it, which will be stored at the proxy. These preferences will be structured in a way which will be clear to the user. so s/he can edit them. There are already rule-extraction methods such as PRISM etc. the question is how to apply them to the logs.

An agent (see figure 3.2) stands side by side with the user, being able to interact with external parties based on these preferences. This is however not an autonomous agent as seen for example in the scenario of *'Maria'* - *Road Warrior* from Ducatel, Bogdanowicz, Scapolo, Leijten, and Burgelman (2004) because in our case the preferences are clear rules and the agents behavior is completely predictable from these rules. A way of suggesting options can be using default action which can be executed with a single click as in the Satchell system (Flynn, Pendlebury, Eldridge, Lamming, and Jones 2000).

To sum up: *Preferences* are *stored* at the proxy of the user and *learned* (by looking at patterns) from previous experiences or *entered* by the user. The *proactiveness* is realized by deterministically following these preferences by the *agent*.

3.3 Modeling context

3.3.1 Introduction

The purpose of this section is to look at different representation languages for context information and decide which language will be the most promising to build upon for the remaining part of the project. In figure 3.3 we can see in bold which parts of databases related to context are influenced by the representation.

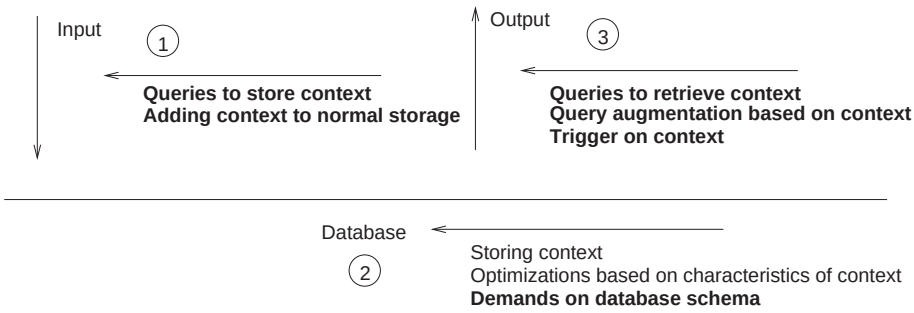

Figure 3.3: Context representation in a database

We see that it is mostly related to the interface of the database. In short:

- Queries can be posed in this representation language.
- Queries can be augmented with the use of context represented in this language.
- Triggers can be formulated in this language.
- It should be possible to represent the database schema in the representation language.

There are several ways of modeling context, which can be divided in several categories. Gu, Wang, Pung, and Zhang (2004) for example differentiate between an *application oriented approach*, a *model oriented approach* and an *ontology oriented approach*. Strang (2004) however does a better job surveying different context modeling strategies. The categories he distinguishes between are:

- profile based (markup scheme models),
- graphical,
- logic based,
- object oriented and (also)
- ontology based.

Both papers conclude that an *ontology based approach* is the best approach for modeling context. We however chose to use Description Logics. This seems strange but actually Description Logics is very related to ontology based approaches and many reasoners for ontologies are even based on Description Logics. Furthermore Description Logics has the advantage of having a clear representation and being decidable (as we will discuss in section 3.3.2). It is also not the case that Description Logics are the same as the *logic based approach* of Strang, who means with this the approach of McCarthy and Buvac (1997).

The rest of this section will be as follows; first we will give a very short introduction to Description Logics. To demonstrate how the modeling in Description Logics would work in practice we will present four scenarios modeled in Description Logics in section 4. In 3.3.3 we will zoom in on five alternatives to Description Logics taken into account the inference we should be able to do for representing context as mentioned in chapter 2.3.5. Finally we will make a start of trying to relate our modeling language to sensors of context and how one would be able to deal with metadata, such as imperfectness, in section 3.3.4. Previous research on selecting knowledge representation systems focusing on Description Logics was done by Speel (1995).

3.3.2 Description Logics

Description Logics are a knowledge representation formalism. Overall they are less expressive than first-order predicate logic (except for the Description Logics with the transitive closure on roles) and if something can be expressed in Description Logics, it can be expressed in First Order Logic. Description Logics

$$
\begin{aligned}
\pi_x(A) &= A(x) & \pi_y(A) &= A(y) \\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D), & \pi_y(C \sqcap D) &= \pi_y(C) \wedge \pi_y(D) \\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D), & \pi_y(C \sqcup D) &= \pi_y(C) \vee \pi_y(D) \\
\pi_x(\exists R.C) &= \exists y.R(x,y) \wedge \pi_y(C), & \pi_y(\exists R.C) &= \exists x.R(y,x) \wedge \pi_x(C) \\
\pi_x(\forall R.C) &= \forall y.R(x,y) \rightarrow \pi_y(C), & \pi_y(\forall R.C) &= \forall x.R(y,x) \rightarrow \pi_x(C)
\end{aligned}
$$

$$
\pi(\mathcal{T}) = \forall \mathsf{x}. \bigwedge_{\mathsf{i}=1}^{\mathsf{n}} (\pi_{\mathsf{x}}(\mathsf{C_i}) \rightarrow \pi_{\mathsf{x}}(\mathsf{D_i}))
$$

Figure 3.4: Translation from Description Logics to First Order Logic

are, more specifically, decidable fragments of First Order Logic. However, next to decidability Description Logics also explicitly distinguishes between the terminological knowledge (the schema), in a TBox, and the assertional knowledge (the concrete situation), in an ABox. In figure 3.3.2 from Sattler, Calvanese, and Molitor (2003) we see how the Description Logics $\mathcal{ALC}$ is represented in First Order Logic for a TBox $\mathcal{T}$. In this figure A is an individual, C and D are concepts and R is a role[1]. The translation of concepts is defined in such a way that the resulting formulas involve only two variables (x,y) and only unary and binary predicates. For our modeling of context we will use $\mathbb{Q}$-$\mathcal{SHIQ}$ from Lutz (2004) this is $\mathcal{ALC}$ with:

- transitive roles,
- role hierarchies,
- inverse roles,
- number restriction and
- representation of numerical knowledge.

Of these properties, expecially the last one is important because we will use it to model time in the recipe scenario (section 4.2.6). $\mathbb{Q}$-$\mathcal{SHIQ}$ is still decidable (ExpTime-complete) (Lutz 2004).

Description logics are closely related to OWL and in fact most the examples we see in this report can be expressed in a subset of OWL. More details on the comparison between OWL and Description Logics are described by Horrocks, Patel-Schneider, and Harmelen (2003).

Automatic reasoning

From the beginning of Description Logics with Kl-One one already had the possibility to do automatic reasoning about subsumption. After Kl-One many systems followed: Classic, Back, Loom, Kris and Crack. And nowadays a new optimized generation of very expressive but sound and complete Description

[1]For examples of modeling in Description Logics, see chapter 4.

Logics systems is available with as the two most important systems FaCT[2] and RACER[3]. These two both make use of the DIG interface[4] which is among others supported by the ontology building software Protégé. FaCT also has a CORBA interface and is programmed in Lisp, for RACER no source code is available.

Speel (1995) has fairly large but old overview of description logic systems. In Möller and Haarslev (2004) a more recent overview is given.

Why decidability

The reason why we would like to use the decidable Description Logics is because a user should be able to construct triggers (in the form of concepts, maybe in an intermediate language) and when using Description Logics, a user can be pointed to more general concepts or already know that a trigger will never fire because the concept is not satisfiable. Another case in which I think decidable subsumption could be helpful is when learning behaviors from a user and adding automatic, proactive, triggers. If in a specific domain (for example "recipes") a rule is found based on instances, based on subsumption this rule could, if we know relations between the two domains, maybe be extended to other domains (such as "restaurants"). Because Description Logics is decidable, reasoning about both cases is guaranteed to finish.

Relation to databases

In this report we look at knowledge representation from a Description Logics point of view and discussed Description Logics-reasoners in which (with maybe a few modifications) our scenarios in chapter 4 can be implemented. When looking at real database queries or scalable implementations of our scenarios it could be possible that the Description Logics-part will be concerned with the triggers/augmentation etc. and the database with the ABoxes (the results of the queries) because the ontologies, although very large, will be at least much smaller than the number of instances. At a later stage we can maybe see if other parts of reasoning can be moved to the database-part as well, resulting in a new(?) database management system.

The advantage is that there has already been done much research on the relation from Description Logics to databases. A survey is given by Borgida, Lenzerini, and Rosati (2003). Franconi (2003) also discusses several examples where Description Logics is used for query processing and information access and Pan and Heflin (2004) present a relational database system with capabilities for DAML+OIL inference which makes use of the FaCT reasoner.

3.3.3 Alternatives

Introduction

In this section we will shortly discuss alternative representation techniques to Description Logics which we could use for representing context.

[2] http://www.cs.man.ac.uk/~horrocks/FaCT/
[3] http://www.sts.tu-harburg.de/~r.f.moeller/racer/
[4] http://dig.sourceforge.net/

Variations of First Order Logic

The problem of representation of context in First Order Logic (FOL) is that it gives too much freedom in modeling choices and it is not focused on representing knowledge. Just as with Description Logics there are also decidable fragments of First Order Logic, and there exists some automatic reasoners. The most famous are Vampire[5], Spass[6] and Otter[7] but there exist even more[8].

The relation of Description Logics to modal logic is described by Sattler, Calvanese, and Molitor (2003) where among others they point out that $\mathcal{ALC}$ can be seen as a notational variant of the multi-modal logic K_m.

Variations of RDF/OWL (Semantic web)

The OWL Web Ontology Language is the successor of DAML+OIL and the proposed language for the semantic web and can be used to model knowledge. It is used to model context among others by Wang, Gu, Zhang, and Pung (2004), Gu, Wang, Pung, and Zhang (2004) and Chen, Finin, and Joshi (2003a). The reasoners for Description Logics can also be used for the decidable fragment of OWL (called OWL-DL). There also exists a reasoner for full OWL based on Otter[9] which is however not completely finished. The main difference between OWL-DL and Description Logics itself is that OWL-DL is more focused on the ABox given a certain TBox; so given an ontology they are interested in the instances.

An RDF based W3C proposal for context is Composite Capability/Preference Profiles (CC/CP). However this proposal focuses more on device context and is therefore located in the *Device* Independence group of the W3C. Their mission is to have "Access to a Unified Web from Any Device in Any Context by Anyone". A critical description of this language for device and user information is given by Held, Buchholz, and Schill (2002) in which they propose Comprehensive Structured Context Profiles (CSCP) as an alternative. This language is however also mainly focused on storing static user and device information for adapting applications. Not for representing the dynamics of context.

Topic maps is another modeling language which is used by Goslar and Schill (2004) and Power (2003) to model context. The main representation format of topic maps is *XML topic maps*[10] and reasoning, also called topic map processing, can among others be done by AsTMa[11].

Conceptual graphs

Just as Description Logics, conceptual graphs (CS's) originate from frame systems and semantic networks. It is a knowledge representation method and has the same expressiveness as First Order Logic but is more focused on representation. There exists a decidable fragment: simple conceptual graphs (SG's) (Sowa 1984).

[5] http://www.cs.man.ac.uk/ riazanoa/Vampire/
[6] http://spass.mpi-sb.mpg.de/
[7] http://www-unix.mcs.anl.gov/AR/otter
[8] http://ar.colognet.org/tools.php
[9] http://www.w3.org/2003/08/surnia/
[10] http://www.topicmaps.org/xtm/
[11] http://astma.it.bond.edu.au/

The relation between Description Logics and Conceptual Graphs has been made several times, among others by Baader, Molitor, and Tobies (1999). It consists of four main points

- Conceptual graphs are interpreted as closed formulae, whereas Description Logics concept descriptions are interpreted by formulae with one free variable.
- Most Description Logics only allow unary and binary relations.
- Conceptual graphs are interpreted by existential sentences, whereas most almost all Description Logics allow for universal quantification.
- Because Description Logics use a variable-free syntax, certain identifications of variables expressed by cycles in Conceptual Graphs and by coreference links in Conceptual Graphs cannot be expressed in Description Logics.

For the choice between Conceptual Graphs and Description Logics it is often advised to look at the specific application. For now we chose Description Logics because there is more known about different subsets and reasoning techniques and it has, in our opinion, a clearer way of notation for knowledge.

OpenCyc

OpenCyc is a database of common knowledge together with a reasoning engine and language based on Lisp (CycL). The language and reasoning system on its own are not very special. What is more interesting is that there is already much coherent knowledge available which we might be able to use at a later stage. A nice property of OpenCyc is the notice of microtheories. In OpenCyc one can have microtheories which contain a set of assertions. Each microtheory bundles assertions based on a shared set of assumptions on which the truth of the assertions depend, a shared topic (medicines, diseases, the stock market) or a shared source (e.g. articles from Dow Jones Newswires). All information within a microtheory must be mutually consistent and two microtheories can be related such that one of them inherits the assertions in the other. In OpenCyc we can for example make a microtheory which represents our knowledge about companies, medicines and the stock market as done by Bunningen (2004). These microtheories relate directly to the view on context from McCarthy and Buvac (1997).

Reasoning languages

Next to the automatic reasoners for First Order Logic, OWL and Description Logics there are also exist reasoning languages with their own syntax. The most well known hereof is probably Prolog. This language has a closed-world-assumption (contrary to the open world assumption of Description Logics) so if it cannot prove a goal using the clauses in its database, and the proof attempt fails after a finite number of steps, then the goal is considered to be false. Prolog is among others used by Ranganathan, Campbell, Ravi, and Mahajan (2002), Ranganathan and Campbell (2003) and Chen, Finin, Joshi, Tolia, and Sayers

(2002) for context aware systems. Ranganathan and Campbell even uses a more advanced prolog extension called XSB.

Clips is another reasoning language used by Gandon and Sadeh (2004). The rules of Clips are supported by Jess (a Lisp like rule language binded with Java).

Finally, Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine.

3.3.4 Relating the model to sensor data

Now we have the foundations of a model to represent context data in and can deal with inference and derivation, it is time to look at the other properties of context. In the scenarios we will see some examples how to deal with temporal information. More information about reasoning with time in Description Logics can be found in the survey of Artale and Franconi (2000).

The nice thing of context information is that if we look at our definition, everything which could be related to a certain event, it actually boils down to the word *related*. And in fact we could translate sensor information or other context information directly to roles in Description Logics.

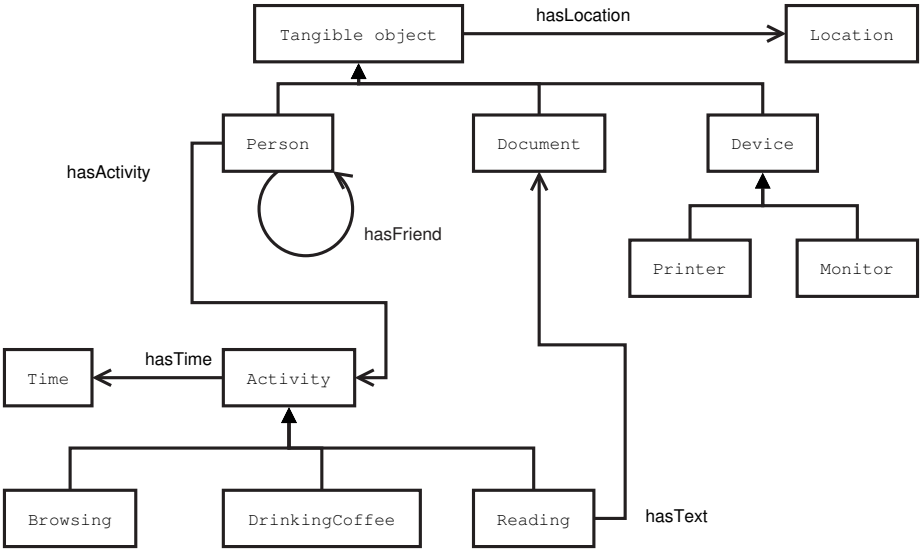

Figure 3.5: A graphical representation of a simple TBox

For example in figure 3.5 a sensor on a printer can determine it's location and could be responsible for the *hasLocation*-role between a printer and a location.

Since a sensor also provides metadata, for example accuracy information, we can add for each role this extra information. An example how this can be done is by reifying the roles to add a probability. This can be seen in figure 3.6. In this figure the *Sensed Location* concept depicts a sensed location of a certain object and includes two roles for relating it to a certain time and probability. To abstract from this reification for reasoning and to make it clearer for the user it could be done automatically.

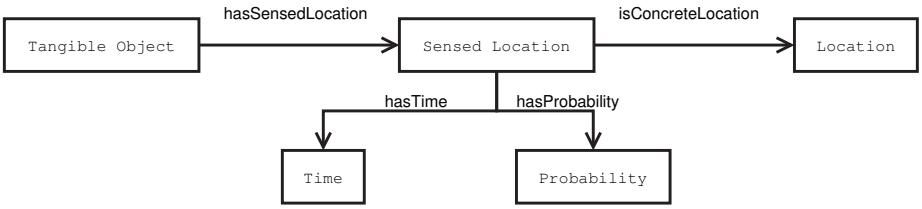

Figure 3.6: The location role reified

3.4 Using the context model to process context aware queries

There are many examples of queries one can do in a context aware environment. Imielinski and Badrinath (1992) and Koile, Tollmar, Demirdjian, Shrobe, and Darell (2003) give examples of queries related to location. Judd and Steenkiste (2003) give some more general, but abstract, context aware queries and some queries for a sensor network are given by Lazaridis, Han, Yu, Mehrotra, Venkatasubramanian, Kalashnikov, and Yang (2004).

We will focus on five "queries":

1. Searching for the first document you read yesterday after the coffee break.
2. Automatically switch on the lights before you have a meeting.
3. Searching for a printer automatically gives the nearest one to which you have access.
4. Proactively get information what is the best thing to do next considering appointments and available time; a sort of deadline management.
5. Searching for someone on data management gives the person who is nearby and not in a meeting.

These "queries" or situations can be categorized in three groups of using context; query augmentation, placing triggers on context and using the context for retrieval purposes. For each group we will give an example how we could realize such a situation.

3.4.1 Query augmentation

In this group are the search for a printer and finding someone on data management.

In this case a query for a printer such as the following:

```
SELECT * FROM Printer
```

could be transfered via a rule which say that we prefer nearby objects, like:

```
TangibleObject -->
  ORDER BY (TangibleObject.hasLocation - self.hasLocation)
```

to

```
SELECT * FROM Printer
  ORDER BY (Printer.hasLocation - self.hasLocation)
```

Because of the TBox, as shown in figure 3.5, this rule of having a preference for near objects also applies for printers.

3.4.2 Placing triggers on context

Under this group falls the automatic switching of the lights and the proactive showing of suggested work. The triggers here are the beginning of a meeting or for example the planning of a new appointment. Based on these triggers something should change; the light should switch on or other suggestions for work should be shown. These triggers can be seen as queries, where a trigger fires when the query has results. An example trigger(query), triggering a signal when there is a monitor is in the same room as certain person could be:

$$Trigger = SELF.hasLocation.contains.Monitor$$

Where contains is the inverse of hasLocation.

Triggers are very important because they make a context aware system proactive. Another example could be that we always want to have things which are in our room easily accessible by a device; a sort of cache. This location aware caching is also suggested by Jones and Brown (2002).

3.4.3 Using the context for retrieval purposes

This last way of using context is also sometimes called Past-Recall-Based constraint (Feng, Apers, and Jonker 2004). Under this group falls the searching for a document which you read yesterday after the coffee break. This query could look, when we ignore the yesterday-part, for example like this:

```
SELECT * FROM Text WHERE
 involvedIn.hasTime
  > (SELECT MAX(hasTime) FROM Drinkingcoffee WHERE
      participant = self)
```

Where *involvedIn* is the inverse of *hasText* and *participant* the inverse of *hasActivity*.

Chapter 4

Scenarios

4.1 Introduction

In this chapter we will, to get a clear view of our approach to context awareness, discuss several scenarios. For the first nine scenarios we will shortly describe how they can be realized in the architecture we discussed in chapter 3.2, for four scenarios we will also suggest how they could be modeled in Description Logics as discussed in chapter 3.3. The last two scenarios are different from the rest; the *airport* scenario is just a copy of a scenario by Satyanarayanan (2001) because it contains some special characteristics as we will discuss in section 4.2.9, the last scenario is different because it contains a very specific office scenario which is constructed in such a way that we think we are able to realize it in the course of our research.

Another excellent resource of five well worked out scenarios is given by Ducatel, Bogdanowicz, Scapolo, Leijten, and Burgelman (2004).

4.2 The scenarios

4.2.1 The phone call

Description

John Brown calls the West family, who are out on the beach. As the Brown's are marked as friends of the family, the default action of the answering machine is to put them through a family phone that is active where Ron's phone has highest priority, unless he is in a business meeting.

Special features

Based on the context of a call (the caller) it is treated differently. Furthermore there is knowledge about the context of the phone users (location and agenda) based on which it is decided where to redirect the call.

Possible implementation in the model

The answering machine could be a context aware application. When a call arrives it will, based on the social context of the family (where it has access to), determine that the caller is a friend. Now it will look at its settings to see how to treat friends. When it knows from these settings that it should pass the call to Ron, it will look at the agenda of Ron if he is in a business meeting. If he is not, the call is redirected. All this information is accessed based on privacy profiles of the users as described by Gandon and Sadeh (2004).

Description Logics

To represent this scenario in Description Logics we will first show the schema (TBox) and then its instances (ABox). First the TBox:

$$\begin{aligned} Call &\equiv \forall callerOf.Person \\ Person &\sqsubseteq \forall currentActivity.Activity \\ Family &\sqsubseteq \forall hasFriend.Person \sqcap \forall hasMember.Person \\ friendOf &\equiv hasFriend^{-1} \\ memberOf &\equiv hasMember^{-1} \\ Activity &\equiv Meeting \sqcup Walking \end{aligned}$$

Now the instances can be as follows:

$$\begin{aligned} JOHNBROWN &: Person \\ RON &: Person \\ LOUISE &: Person \\ C_1 &: Call \\ < C_1, JOHNBROWN > &: callerOf \\ WESTFAMILY &: Family \\ < WESTFAMILY, JOHNBROWN > &: hasFriend \\ < WESTFAMILY, RON > &: hasMember \\ < WESTFAMILY, LOUISE > &: hasMember \\ A_1 &: Meeting \\ A_2 &: Walking \\ < RON, A_1 > &: currentActivity \\ < LOUISE, A_2 > &: currentActivity \end{aligned}$$

Using this knowledge the answering machine should now be able to redirect a phone call. Therefore we define a trigger. Costa (2003) defined triggers to be fired when condition which first evaluated to false now evaluated to true. For Description Logics we define the trigger as follows: *If a concrete situation in the ABox is subsumed by the trigger concept, the trigger is fired.*

The next concept can be used as trigger to the answering machine

$$Trigger_1 \equiv Call \sqcap \exists callerOf \circ friendOf.WESTFAMILY$$

When this trigger is activated the corresponding call is found using the same concept, and the result will be C_1. Now the phone can do a query for persons of the family which are able to accept the call.

$$Person \sqcap \neg \exists currentActivity.Meeting \sqcap memberOf.WESTFAMILY$$

The result will be $LOUISE$. Now the phone can redirect call C_1 to $LOUISE$.

Design choices and possible issues

Family context As we saw in this scenario, not only individuals have a context, but families can have one as well. Although this could be derived from the context of the members individually, it is too much too ask from an answering machine to have knowledge of the family structure so there must be general knowledge of a family.

4.2.2 The document

Description

Ron actually does have a business meeting. He has to do a talk about the development of an auto inference database. He quickly needs the document which he wrote last night before dinner in the hotel for this afternoon's talk, so he asks his personal assistant to get this document.

Special features

In this case past context is used for retrieval purposes.

Possible implementation in the model

When Ron writes a document in a context aware word processor, the time at which he works on it is captured by the program. Furthermore, his context is also stored in his proxy. When asking the word processor for the document he wrote last night before dinner in the hotel, the word processor determines this moment by querying the users context (via the service registry) and can look at documents written by the user at the time the user specifies.

4.2.3 Looking at pictures

Description

Meanwhile Louise (Ron's wife) is standing in the art gallery looking at paintings. Because she always asks about the style of the painter when she looks at mostly blue paintings, at the following blue paintings she is, unobtrusively, presented with the style of the painter.

Special features

In this scenario, not only preferences are used (interested in the style of blue paintings) but also learned. Furthermore, there is knowledge about the paintings needed for the inference (the blue color). And finally there is an action triggered by the environment instead of by the user.

Possible implementation in the model

The art gallery publishes information on each painting about colors, style, location etc. At the moment the user takes the action of asking the style of a painting this information is stored. The *learner* examines the log and can look for co-occurring events when the user asked for the style of a painting. In this case, it will find out that if the Louise looks at blue paintings, she is interested in the style so it will add the preference to show the style of the painting when Louise looks at a blue painting.

Description Logics

This scenario is mainly about learning of triggers for which we use timepoints. Here I assume a discrete flow of time and a "point" would be some generally agreed interval for the whole "ontology". So we could say that in the same millisecond/minute (depending on the agreed interval) someone looked at a painting and did something else. The Tbox can be like this:

$$
\begin{aligned}
Person &\sqsubseteq \forall hasActivity.Activity \\
Activity &\equiv Askstyle \sqcup Meeting \sqcup Walking \\
&\quad \sqcup Askcolor \sqcup Looking \\
Activity &\sqsubseteq \exists hasTimepoint.Timepoint \\
Timepoint &\equiv hasEvent.Activity \\
hasTimepoint &\equiv hasEvent^{-1} \\
Looking &\sqsubseteq \exists target.Object \\
Object &\equiv BluePainting \sqcup YellowPainting
\end{aligned}
$$

And the corresponding ABox:

$$
\begin{aligned}
LOUISE &: Person \\
A_1 &: Askstyle \\
A_2 &: Askstyle \\
A_3 &: Askcolor \\
A_4 &: Looking \sqcap target.BluePainting \\
A_5 &: Looking \sqcap target.BluePainting \\
A_6 &: Looking \sqcap target.YellowPainting \\
< LOUISE, A_1 > &: hasActivity \\
< LOUISE, A_2 > &: hasActivity \\
< LOUISE, A_3 > &: hasActivity \\
< LOUISE, A_4 > &: hasActivity \\
< LOUISE, A_5 > &: hasActivity \\
< LOUISE, A_6 > &: hasActivity \\
< A_1, T_1 > &: hasTimepoint \\
< A_2, T_2 > &: hasTimepoint \\
< A_3, T_3 > &: hasTimepoint \\
< A_4, T_1 > &: hasTimepoint \\
< A_5, T_2 > &: hasTimepoint \\
< A_6, T_3 > &: hasTimepoint
\end{aligned}
$$

We do learning based co-occurrence. To deduct interesting co-occurrences, we can look at things which co-occur with an activity which is carried out often, suppose this activity is AskStyle. We can now do the following query:

$$Query_1 = Activity \sqcap hasTimepoint.hasEvent.Askstyle \sqcap \neg Askstyle$$

When we now check for the "least common subsummer" of this query we know which specific activity occurred always while asking for a style. The answer here should be that Louise looked at a blue painting. We have some ideas how to capture things which happened *before* instead of *during* a certain activity which could be done using a Gaussian curve for the time at which an event occurs.

4.2.4 The nearest restaurant

Description

After Ron's presentation is finished, Louise joins Ron to go to a restaurant. Because they are in the north of the city and not familiar with the surroundings, they ask their context aware phone for a nearby restaurant, taken into account their preferences.

Special features

Although the question is just for a restaurant, it should be augmented depending on the user preferences.

Possible implementation in the model

Restaurants advertise themselves globally, in the service registry, including among others their location and kitchen. This could be for example according to a similar ontology as mentioned in agent cities[1]. When the user queries for a restaurant on the device, according to the restaurant ontology on her/his context aware phone, s/he can choose a cuisine. Since preferences for a certain cuisine could be entered beforehand or already be learned, just searching for "a restaurant nearby with company of Louise" is also enough to take these into account; the context aware phone will simply query the proxy of the user for other preferences.

Design choices and possible issues

Advertisement Another possibility is that the user gets information about restaurants without explicitely asking for it. This could be done by the agent, based on the users profile.

What is nearby? The question if a restaurant is nearby depends on the way of transportation, this has to be taken into account as well.

4.2.5 A web service

Description

The presentation was a blast; Ron is invited to become professor at this university. The same night, Louise and Ron decide to move out of their apartment to a nice house. They log in to the "digital desk" of the estate agent where they see the houses based on their preferences. When they have given authorization an appointment is being made and an initial limited amount transfered to the estate agent.

Special features

This scenario focuses on the short term application domain, to be able to see he architecture is compatible with this. The special feature is how the (web)application acquires the house preferences of the user.

Implementation in the model

This scenario could actually partly be implemented with Passport (McKiernan 2002). Passport stores information like birth date, country, name, preferred language etc. and a user can choose which information to share. This could even be credit card information, so payments could be done as well. However

[1] http://agentcities.crm-paris.com/services/RestaurantServiceDescription.html . It doesn't have to be this format, but it is an illustration that it is possible to have a single format for restaurants.

the storage function (which was called myWallet) was canceled. In our model we already store preferences as context and since a web service can, if it is allowed, access this information it can display information based on this context.

Description Logics

From a representational point of view an interesting part of this scenario is that a user, an entity, can like certain concepts like *red houses*. Second, to be able to reason about the color and size of a house, we want to introduce an *exhaustive list* for properties of houses, so a house cannot be small and large at the same time. We therefore use the a so called *value partition* for these properties which means adding a covering axiom (Color, Size) to make the list exhaustive, make the subclasses (Small, Medium, Large, Green, Red and Purple) disjoint and the properties (hasColor, hasSize) functional. The TBox now looks as follows:

$$
\begin{aligned}
House &\sqsubseteq Thing \\
Color &\equiv Green \sqcup Red \sqcup Purple \\
Green &\equiv \neg Red \sqcap \neg Purple \\
Red &\equiv \neg Green \sqcap \neg Purple \\
Purple &\equiv \neg Green \sqcap \neg Red \\
Size &\equiv Small \sqcup Medium \sqcup Large \\
Small &\equiv \neg Medium \sqcap \neg Large \\
Medium &\equiv \neg Small \sqcap \neg Large \\
Large &\equiv \neg Small \sqcap \neg Medium \\
Group &\sqsubseteq \exists hasMember.Person \sqcap \forall hasMember.Person \\
House &\sqsubseteq \exists hasColor.Color \\
&\qquad \sqcap \forall hasColor.Color \sqcap \exists hasSize.Size \sqcap \forall hasSize.Size \\
Person &\sqsubseteq \forall likes.Thing \\
likedBy &\equiv likes^{-1} \\
memberOf &\equiv hasMember^{-1}
\end{aligned}
$$

Where hasSize and hasColor are functional roles (also called attributes or features). The ABox is now as follows:

$$\begin{aligned}
RON &: Person \sqcap \forall likes.((House \\
&\quad \sqcap (hasColor.(Green \sqcup Red) \\
&\quad \sqcup hasSize.Big)) \\
&\quad \sqcup \neg House) \\
LOUISE &: Person \sqcap \forall likes.((House \\
&\quad \sqcap (hasColor.(Red \sqcup Purple) \\
&\quad \sqcup hasSize.Small)) \\
&\quad \sqcup \neg House) \\
H_1 &: House \\
< H_1, C_1 > &: hasColor \\
< H_1, S_1 > &: hasSize \\
C_1 &: Green \\
S_1 &: Small \\
H_2 &: House \\
< H_2, C_1 > &: hasColor \\
< H_2, S_1 > &: hasSize \\
C_2 &: Red \\
S_2 &: Large \\
< RONLOUISE, LOUISE > &: hasMember \\
< RONLOUISE, RON > &: hasMember
\end{aligned}$$

Interesting about this ABox is that we can say that the instance RON likes certain concepts by using a complex concept to describe RON. We can now augment a (Description Logic-)query for a house, which would look like

$$Query_1 \equiv House$$

to one in which it is required that a house which matches the query has something which Ron likes and something which Louise likes:

$$Query_2 \equiv House \sqcap \exists likedBy.LOUISE \sqcap \exists likedBy.RON$$

In the representation of the scenario we also introduced Ron and Louise together as a group. We could try to use this notion to represent the same query:

$$Query_2 \equiv House \sqcap \exists likedBy.(memberOf.RONLOUISE)$$

The problem is here that in this case the liking by one of the two is sufficient (instead of by both of them). An alternative is to define the $likedBy$ of the group specifically (and use other algorithms for combining properties of subclasses from a group. This is probably also a more natural way because one can have different ways of reasoning about what a group likes or who are friends of a group. For such ambiguous properties it is maybe even better not to reason about them but let the users edit this information themselves.

4.2.6 Searching a recipe

Description

It seems like no restaurant fits their needs and because they still have time, they decide to cook together. Back in the kitchen Ron asks for an original recipe. Because the ambient "knows" they want to finish soon, it proposes a fast recipe of which the ingredients are already in their refrigerator.

Special features

Instead of restaurants, recipes cannot advertise on their own, so they have to be provided by other parties (supermarkets?). Another difficulty is how the ambient "knows" that they have little time, because it can not be stored in a preference file.

Possible implementation in the model

There will be different providers of recipes (supermarkets, public domain recipes etc.), who will uniformly publish their information on "the internet" in a common recipe ontology. The matching will be done in the same way as the restaurants. Except for the "fast recipe" part (which is based on context rules). These context rules can be learned from experience by the system, but can also be edited directly, for example a user could have the following rule in its preferences:

```
($timeInCurrentTimeSlot < 1hour) -> Prefer(Recipe.time = fast)
```

Note that the notice of "having little time" is not explicitely stored in the preferences. Instead there is a rule which based on the amount of time in the current time slot says that the user prefers fast recipes. This rule can be entered by the user or can be learned by continously noting that a user select fast recipes when s/he has little time.

Design choices and possible issues

Economic issues Asking oneself if the user will buy recipe books for getting the recipes, they will come with a subscription to a "ambient service" or if they will be provided by supermarkets is of course an interesting question, but is certainly out of scope here.

Description Logics

Specific about this scenario is the notion of time, which is the reason why we chose $\mathbb{Q}$-$\mathcal{SHIQ}$ to model context. Concepts can now have rational numbers as role fillers and one can reason with them by comparing them to rational role fillers of related concepts or testing for a specific number (formally explained by

Lutz (2004)).

$$
\begin{aligned}
Person &\sqsubseteq \forall likes.Thing \sqcap \forall hasActivity.Activity \\
Food &\sqsubseteq Thing \\
Activity &\sqsubseteq \exists_{=1} hasStartTime. >_0 \sqcap \exists_{=1} hasEndTime. >_0 \\
Recipe &\sqsubseteq \forall hasIngredient.Food \sqcap \exists_{=1} takesTime. >_0 \\
participator &\equiv hasActivity^{-1} \\
Foodstorage &\sqsubseteq contains.Food \\
Refrigerator &\sqsubseteq Foodstorage \\
Food &\sqsubseteq Milk \sqcup Eggs \sqcup Cinnamon \\
Wentelteefjes &\sqsubseteq Recipe \sqcap hasIngredient.Milk \sqcap hasIngredient.Eggs \\
&\quad \sqcap hasIngredient.Cinnamon \sqcap hasIngredient.Bread \\
&\quad \sqcap takesTime. < 3
\end{aligned}
$$

The Abox instantiates among others the rational numbers for the start and end times:

$$
\begin{aligned}
RON &: Person \sqcap \forall likes.(Eggs \\
&\quad \sqcup \neg Food) \\
< RON, A_1 > &: hasActivity \\
A_1 &: Activity \\
< A_1, 12 > &: hasStartTime \\
< A_1, 14 > &: hasEndTime \\
R_1 &: Refrigerator \\
F_1 &: Milk \\
F_2 &: Eggs \\
F_3 &: Cinnamon \\
F_4 &: Bread \\
< REFRIGERATOR, F_1 > &: contentOf \\
< REFRIGERATOR, F_2 > &: contentOf \\
< REFRIGERATOR, F_3 > &: contentOf \\
< REFRIGERATOR, F_4 > &: contentOf
\end{aligned}
$$

In this scenario the query rewriting is dependent on the situation. When Ron has little time a fast recipe must be triggered for which the ingredients are in his house and there is an ingredient which he likes in the recipe. The scenario does not tell what happens when there is more time, so we assume in this case he just asks for a recipe.

First when Ron asks for a recipe with the query:

$$query_{i1} : Recipe$$

we look if there is little time, with the following "check":

$$Check_1 \equiv Activity \sqcap \exists participator.RON \sqcap \exists hasStartTime. < 10$$

and the query can be rewritten as follows:

$$\begin{aligned} query_{i2} : Recipe \sqcap \exists hasIngredient.likedBy.RON \\ \sqcap \forall hasIngredient.(REFRIGERATOR.contentOf) \\ \sqcap \exists takesTime. < 2 \end{aligned}$$

which returns the right recipe with instance checking. The queries in this case are different from the queries we saw in section 4.2.5 because we now query for concepts instead of instances. This is done using *instance checking* where reasoning returns all concepts which subsume an instance (here $query_{i1}$ and $query_{i2}$). This means that recipes should not restrict ingredients because in this case extra available ingredients in the refrigerator restrict the result instead of augmenting it.

What is furthermore interesting about this scenario is that the contents of the recipe are described in the TBox which is logical because they are not concrete instances but concepts. The problem however, is how to deal with the combination of having a concrete instance of a recipe (eg. the description how to make the food with a description of the ingredients on concept level). In this scenario we showed that it is possible to find the right recipe-concept based on ingredients and available time. How a concrete recipe is now coupled to this concept is open for further research. Maybe this can be done with web services or a so called "find-concept"; a concept which is stored as instance in the database, and can be searched for using *instance checking*, when found we can look for instances which relate to this concept. As an example of a "find-concept" we could have this rule in the ABox:

$$< CONCRETERECIPE, Wentelteefjes > : hasFindConcept$$

Note that this construction is not part of Description Logics and just an idea how to deal with such situations.

4.2.7 Improving life

Description

The parents of Louise are already in their eighties and are doing very well. Although her fathers back prevents him for walking too long, the ambient helps him with: automatically switching the TV-news on at 20:00, making coffee before guests arrive, putting on music he likes. Furthermore temperature and lightning are adapted to the time of the day and the room which is occupied. Because her parents both need to watch their health, they are happy with the help they get from the ambience; food advice from refrigerator, automatic extraction of chemical samples from the toilet and weight measurement in bathroom to monitor the health condition, quick access to medical information and help functions in their personal device and when unusual things happen the hospital is informed.

Special features

This scenario is not very specific, but describes a promising application; to assist live of sick and elderly people.

Possible implementation in the model

This example is mainly about context aware applications. The TV will, depending on the profile (at home, preferences), play shows. The refrigerator will, depending on health situation, suggest recipes etc.

Design choices and possible issues

Privacy Privacy plays an important role in this scenario for example the question to which amount reporting unusual situations to a hospital is useful for preventing serious issues versus to which amount privacy is given up.

4.2.8 At the university

Description

After dinner Ron has an emergency meeting at the university. Because this meeting is very important, his personal assistant does not intrude him with messages or reminders. Until a phone call from Louise arrives; the assistant notices him very subtile so he can go outside for some minutes to talk with her about their upcoming baby. When he arrives back in the meeting he shows some slides from the presentation which was given during the business meeting this afternoon. The slides are automatically displayed on the projector. However because he wants to show some details, they go to another room with a larger screen where the presentation is automatically "transported to" when he arrives. At the end of the meeting he is noticed that someone working on his research is also nearby.

Special features

This scenario tries to encompass four context aware features of which three are also mentioned in DeVaul and Pentland (2000) namely; central management of different attentions and prioritizing them (messages in the meeting), decentralized contextual resource discovery and allocation (presentation which is automatically displayed on the best available screen) and flexible context sensing and classification based on heterogeneous sensors (how to recognize he is in a meeting). Although this last feature could also be discussed in other scenarios, we will discuss it here explicitely. The last feature, the discovery of someone with same interest is based on the DMe scenario of Ducatel, Bogdanowicz, Scapolo, Leijten, and Burgelman (2004).

Implementation in the model

The central management of different attentions could be done at a device Ron carries with him. This device can depending on Rons context determine what kind of attention to generate. Prerequisite is however that all application requiring attention from Ron will do so via this device.

The contextual resource discovery can take place by the *Agent* based on a rule to display presentation information on the nearest largest device, so it moves with Ron. Another option is that a device on which the presentation was made had limited display capabilities and automatically searches for an unused

screen to display its information on. Of course the *Agent* could also only *suggest* nearby large screens.

The recognition of situations, in this case being in a meeting can be done via learned "preferences" or multimodal sensors. Because the user context contains all information of the different sensors this detection can be learned. An alternative could be using ready-made recognizers which can immediately sense a meeting and store this information.

The discovery of context of other users (with same interest) is possible because this context information is published by their proxies. Based on privacy rules users can discover nearby interesting users.

Design choices and possible issues

Querying over area and privacy Because we do the global query "get everyone who is nearby", we need to take into account both the fact how this can be solved without querying each user in the whole world. Furthermore not every user probably wants to show it's location, so privacy must be taken into account as well.

4.2.9 The airport

As mentioned in the introduction, this scenario is copied from Satyanarayanan (2001).

description

Jane is at Gate 23 in the Pittsburgh airport, waiting for her connecting flight. She has edited many large documents, and would like to use her wireless connection to e-mail them. Unfortunately, bandwidth is miserable because many passengers at Gates 22 and 23 are surfing the web. Aura[2] *observes that at the current bandwidth Jane won't be able to finish sending her documents before her flight departs. Consulting the airport's network weather service and flight schedule service, Aura discovers that wireless bandwidth is excellent at Gate 15, and that there are no departing or arriving flights at nearby gates for half an hour. A dialog box pops up on Jane's screen suggesting that she go to Gate 15, which is only three minutes away. It also asks her to prioritize her e-mail, so that the most critical messages are transmitted first. Jane accepts Aura's advice and walks to Gate 15. She watches CNN on the TV there until Aura informs her that it is close to being done with her messages, and that she can start walking back. The last message is transmitted during her walk, and she is back at Gate 23 in time for her boarding call.*

Special features

This scenario emphasizes combining knowledge from different layers; low-level (network congestion) and high-level (knowledge of boarding time), knowledge of different spaces (to redirect to the right place) and proactiveness; the system knows that user will have little time.

[2] Aura is the ambient environment from CMU.

Implementation in the model

The airport can provide services with among others knowledge where within the airport how many bandwidth is available. The combination of different levels of context is something which we have to take in mind when designing the context storage that we are able to combine these levels in reasoning. And finally, the proactiveness can easily be implemented with the same *timeInCurrentTimeSlot* notion as we used in section 4.2.6.

4.2.10 An implementable office scenario

Introduction

As we saw in section 3.4 there are three important ways in which context can influence doing queries:

1. Using context as a trigger (getting a message when interesting persons are nearby, adapting the function of a button in an application)
2. Using context as query augmentation (only returning nearby printer, only returning headers of e-mail messages when network connection is slow)
3. Using context which is added storage for retrieval purposes (retrieving the documents I printed yesterday)

We decided to work out one scenario with these three different aspects. First, we wanted to focus on an application which can be implemented at our university. Not only to test our techniques and implementation possibilities but also to inspire other people (researchers, companies, students) to think about the possibilities of AmI and how to contribute to this research. Second, the scenario should be a real office/university scenario (not searching of recipes or looking at paintings), so people might actually find it a useful scenario. And finally, the scenario should be a challenge to implement, but not impossible.

The scenario

Ron is researcher at the University of Twente. On Tuesday morning after he finishes his workshop proposal, he has some time left for himself. Since the ambient knows he is in no hurry, he is unobtrusively notified that some of his colleagues of him are having a coffee break. After the coffee break, he meets Alex in the corridor who suggests an article on XQuery. The next day, Chris, a researcher at Philips, is visiting Ron. Fifteen minutes before the meeting Ron looks at the large overview screen in his office, on top there is a summary of common interests between him and Chris, after this, his other appointments for today and because he has little time, no news headlines. Ron prefers to have the lights on in his room when he has a visitor, which he first did by himself but now is known by the ambient which automatically switches on the lights, just before Chris enters the office. They talk to each other about query languages when Ron suddenly remembers the article which was mentioned to him the day before after the coffee break. He does a query for articles at which he looked at after the coffee break when Alex was also with him and that are about query languages. When he get back the right article, he gives it to Chris. When leaving the office of Ron, Chris decides to print the article. This is automatically done on the nearest

printer, to which he is redirected with signs on the wall. Ron's next appointment is already in 30 minutes; he has to leave to Italy for a conference on Description Logics. Based on the topic of the conference, attendees and things Ron still has to do, data is cached on his PDA.

Extensions

When the scenario we described is implemented and works in practice I think the prototype has succeeded. However, of course we can add several interesting extensions, based on the same principles and also in an office environment.

Learning At this moment, it is learned at which moment the light is switched on, other office actions which could be learned are: making coffee (for a several number of persons), starting and stopping the computer or putting the sunscreen up or down.

Adapting environment Next to having the large overview screen at the office, places in the corridor screens can also continously show next appointments and next deadlines, but also for example the documents you already read at that location (with new similar documents) or appointments with nearby people.

Nearest device Instead of printing documents on the nearest printer, also presentations could be shown on the nearest beamer.

Caching We now saw that documents are cached for a certain event on the PDA, other examples of office scenarios where the ambient is pro-active, could be exiting the power-save modus of a printer when the user is probably going to print something.

Other extensions could be done when extending the infrastructure to the whole campus; more persons can be found and one can find more interests of persons.

Scenarios already implemented elsewhere

We believe some of the previous applications to be fairly new, but there are office applications which were already done by others which could complement them:

Sticky notes Displaying a note when at a certain time being at a certain location, as described by Brown, Bovey, and Chen (1997). We can augment this by for example displaying notes when in the neighborhood of a person with the same expertise.

Transporting windows system Having your desktop at the computer at which you are nearest was described by Harter, Hopper, Steggles, Ward, and Webster (1999). This roughly corresponds with the automatic printing on the nearest printer. However, this is not initiated by an action of a user but purely by the location.

Redirecting telephone calls (Semi)automatically redirecting calls to the current room someone is in was described by Want, Hopper, Falcão, and Gibbons (1992).

Chapter 5

Future directions

In this technical report we gave an overview of many aspects of context and how to relate context awareness to databases. The final goal is of course to realize the scenarios and especially the queries posed in section 3.4, but before this is possible, certain tasks have to be carried out.

We must look how our architecture given in section 3.2 relate to the architecture used in AmbientDB and if it is possible to use the AmbientDB infrastructure.

To evaluate the system a prototype should be build as described in section 4.2.10. Based on this we can also come up with some evaluation measures of context awareness on which previous research has among others been done by Brown and Jones (2004) and Ranganathan, Al-Muhtadi, Biehl, Ziebart, Campbell, and Bailey (2004).

Related to our modeling we should better define how sensor information can be related to our context model (via database techniques or context modules) and a well defined way of augmenting queries of which we in this report only gave some examples. To do both it is important to have a well defined ontology for which we have to look at other ontologies related to the characteristics of context and other ontologies which will probably be defined in other MultimediaN projects.

The most important thing however is how to get from sensor information to information which we a computer or user can use to describe situations; to get from fuzzy temperature information to the conclusion of being in a meeting. To do this we need among others inference, probabilistic reasoning and maybe learning. But how should they be applied? And where will it be done? Probably in a sort of context proxy as discussed in section 3.2.2, but should it for example be done with a description logic reasoner or should these capabilities be present in the database? To answer these questions will probably be one of my main research tasks the next three years.

Bibliography

Antifakos, S., A. Schwaninger, and B. Schiele (2004). Evaluating the effects of displaying uncertainty in context-aware applications. In N. Davies, E. Mynatt, and I. Siio (Eds.), *UbiComp 2004: Ubiquitous Computing: 6th International Conference*, pp. 54–69. Springer-Verlag Heidelberg.

Artale, A. and E. Franconi (2000). A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence 30*(1-4), 171–210.

Baader, F., R. Molitor, and S. Tobies (1999). Tractable and decidable fragments of conceptual graphs. In W. Tepfenhart and W. Cyre (Eds.), *7th International Conference on Conceptual Structures: Standards and Practices*, pp. 480–493. Springer-Verlag.

Boncz, P. and C. Treijtel (2003). Ambientdb: relational query processing in a p2p network. In *Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop*, pp. 153–168. Springer.

Bonnet, P., J. Gehrke, and P. Seshadri (2001, January). Towards sensor database systems. In *2nd International Conference on Mobile Data Management*.

Borgida, A., M. Lenzerini, and R. Rosati (2003). Description logics for databases. In *The Description Logic Handbook*, pp. 462–484. Cambridge University Press.

Bressan, S., K. Fynn, C. H. Goh, S. E. Madnick, T. Pena, and M. D. Siegel (1997). Overview of a prolog implementation of the context interchange mediator. In *International COnference and Exhibition on The Practical Applications of Prolog*.

Brown, P., J. D. Bovey, and X. Chen (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications 4*(5), 58–64.

Brown, P., W. Burleson, M. Lamming, O.-W. Rahlff, G. Romano, J. Scholtz, and D. Snowdon (2000, April). Context- awareness: Some compelling applications. In *CH12000 Workshop on The What, Who, Where, When, Why and How of Context-Awareness*.

Brown, P. and G. J. F. Jones (2004). Research issues in context-aware retrieval: evaluation.

Bunningen, A. v. (2004). Augmented trading: From news articles to stock price predictions using syntactic analysis. Master's thesis, University of Twente.

Chalmers, M. (2004). A historical view of context. *Computer Supported Co-operative Work (CSCW), The Journal of Collaborative Computing (to appear)*.

Chen, G. and D. Kotz (2000). A survey of context-aware mobile computing research. Technical report TR2000-381, Dept. of Computer Science, Dartmouth College.

Chen, H., T. Finin, and A. Joshi (2003a). An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review 18*(3), 197–207.

Chen, H., T. Finin, and A. Joshi (2003b). Semantic web in a pervasive context-aware architecture. *Artificial Intelligence in Mobile System*, 33–40.

Chen, H., T. Finin, A. Joshi, S. Tolia, and C. Sayers (2002). Creating context-aware software agents. In *First GSFC/JPL Workshop on Radical Agent Concepts*. Springer Verlag (2003).

Cherniack, M., M. Franklin, and S. Zdonik (2001). Data management for pervasive computing. Tutorial.

Costa, P. D. (2003). Towards a services platform for context-aware applications.

Deshpande, A., C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong (2004). Model-driven data acquisition in sensor networks. In *VLDB 2004*, pp. 588–599.

DeVaul, R. W. and A. Pentland (2000). The ektara architecture: The right framework for context-aware wearable.

Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput. 5*(1), 4–9.

Dey, A. K. and G. D. Abowd (1999). Towards a better understanding of context and context-awareness. Technical report GIT-GVU-99-22, Georgia Institute of Technology.

Dey, A. K., G. D. Abowd, and D. Salber (1999). A context-based infrastructure for smart environments. In *1 st Intl. Workshop on Managing Interactions in Smart Environments (MANSE'99)*.

Dey, A. K., J. Mankoff, and G. D. Abowd (2000, September). Distributed mediation of imperfectly sensed context in aware environments. Technical Report GIT-GVU-00-14, Georgia Institute of Technology.

Dourish, P. (2004). What we talk about when we talk about context. *Personal Ubiquitous Comput. 8*(1), 19–30.

Ducatel, K., M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman (2004). Scenarios for ambient intelligence in 2010. Scientific, IST Advisory Group (ISTAG).

Feng, L., P. Apers, and W. Jonker (2004). Towards context-aware data management for ambient intelligence. In *Conf. on Database and Expert Systems Applications*.

Flynn, M., D. Pendlebury, M. Eldridge, M. Lamming, and C. Jones (2000). The satchel system architecture: mobile access to documents and services. *Mobile Networks and Applications 1*(12), 243–258.

Franconi, E. (2003). Description logics for conceptual design, information access, and ontology integration: Research trends. *Networks - journal of the philosophy of Artificial Intelligence and Cognitive Science, special issue on the Semantic Web 2*, 25–32.

Gandon, F. and N. Sadeh (2004). Semantic web technologies to reconcile privacy and context awareness. *Web Semantics Journal 1*(3), 241–260.

Goslar, K. and A. Schill (2004). Modeling contextual information using active data structures. In *Workshop for Pervasive Information Management*.

Gray, P. D. and D. Salber (2001). Modelling and using sensed context information in the design of interactive applications. In *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, pp. 317–335. Springer-Verlag.

Grimm, M., M.-R. Tazari, and D. Balfanz (2002). Towards a framework for mobile knowledge management. In *4th international conference on Practical Aspects of Knowledge Management*.

Gu, T., H. K. Pung, and D. Q. Zhang (2004). A bayesian approach for dealing with uncertain contexts. In *Second International Conference on Pervasive Computing (Pervasive 2004)*.

Gu, T., X. H. Wang, H. K. Pung, and D. Q. Zhang (2004). An ontology-based context model in intelligent environments. In *Communication Networks and Distributed Systems Modeling and Simulation Conference*.

Harter, A., A. Hopper, P. Steggles, A. Ward, and P. Webster (1999). The anatomy of a context-aware application. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pp. 59–68. ACM Press.

Held, A., S. Buchholz, and A. Schill (2002). Modeling of context information for pervasive computing applications. In *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*.

Henricksen, K. and J. Indulska (2004). Modelling and using imperfect context information. In *Second IEEE International Conference on Pervasive Computing and Communications. Workshop on Context Modelling and Reasoning (CoMoRea'04)*, pp. 33–37. IEEE Computer Society.

Henricksen, K., J. Indulska, and A. Rakotonirainy (2002). Modeling context information in pervasive computing systems. In *First International Conference on Pervasive Computing*, pp. 167–180.

Hightower, J. and G. Borriello (2001). A survey and taxonomy of location systems for ubiquitous computing. Technical Report UW-CSE 01-08-03, University of Washington.

Hightower, J. and G. Borriello (2004). Particle filters for location estimation in ubiquitous computing: A case study. In N. Davies, E. Mynatt, and I. Siio (Eds.), *UbiComp 2004: Ubiquitous Computing: 6th International Conference*, pp. 88–106. Springer-Verlag Heidelberg.

Höppner, F. (2003). *Knowledge Discovery from Sequential Data*. Ph. D. thesis, Technischen Universitat Braunschweig.

Horrocks, I., P. Patel-Schneider, and F. v. Harmelen (2003). From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics 1*(1), 7–26.

Imielinski, T. and B. R. Badrinath (1992). Querying in highly mobile distributed environments. In *the 18th International Conference on Very Large Data Bases*, pp. 41–52. Morgan Kaufmann Publishers Inc.

Jones, G. J. F. and P. Brown (2002). Challenges and opportunities for context-aware retrieval on mobile devices. In *SIGIR workshop on Mobile Personal Information Retrieval*, pp. 47–56.

Jones, G. J. F. and P. Brown (2004, September). Context-aware retrieval for ubiquitous computing environments. In *Mobile HCI Workshop on Mobile and Ubiquitous Information Access*, pp. 227–243. Springer.

Judd, G. and P. Steenkiste (2003). Providing contextual information to pervasive computing applications. In *IEEE International Conference on Pervasive Computing (PERCOM)*, pp. 133–142.

Kindberg, T., A. Sellen, and E. Geelhoed (2004, September). Security and trust in mobile interactions: A study of users perceptions and reasoning. In N. Davies, E. Mynatt, and I. Siio (Eds.), *UbiComp 2004: Ubiquitous Computing: 6th International Conference*, pp. 196–213. Springer-Verlag Heidelberg.

Koile, K., K. Tollmar, D. Demirdjian, H. Shrobe, and T. Darell (2003, October). Activity zones for context-aware computing. In *Fifth International Conference on Ubiquitous Computing (UbiComp'03)*.

Korkea-aho, M. (2000). Context-aware applications survey.

Korpipää, P., M. Koskinen, J. Peltola, S.-M. Mäkelä, and T. Seppänen (2003). Bayesian approach to sensor-based context awareness. *Personal Ubiquitous Comput. 7*(2), 113–124.

Korpipää, P., J. Mäntyjärvi, J. Kela, H. Keränen, and E.-J. Malm (2003). Managing context information in mobile devices. *IEEE Pervasive Computing 2*(3), 42–51.

Lazaridis, I., Q. Han, X. Yu, S. Mehrotra, N. Venkatasubramanian, D. V. Kalashnikov, and W. Yang (2004). Quasar: quality aware sensing architecture. *ACM SIGMOD Record 33*(1), 26–5.

Leonhardt, U. and J. Magee (1998, March). Security considerations for a distributed location service. *Journal of Network and Systems Management 6*(1), 51–70.

Lieberman, H. and T. Selker (2000). Out of context: computer systems that adapt to, and learn from, context. *IBM Systems Journal 39*(3-4), 617–632.

Look, G. and S. Peters (2003). Plan-driven ubiquitous computing. In *Student Oxygen Workshop (SOW'03)*. MIT Project Oxygen.

Lutz, C. (2004). Adding numbers to the shiq description logic–first results. In *Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-2002)*, pp. 191–202. Morgan Kaufman.

Madden, S. R., W. Hong, J. M. Hellerstein, and M. Franklin (2004). Tinydb web page. http://telegraph.cs.berkeley.edu/tinydb/.

McCarthy, J. and S. Buvac (1997). *Computing Natural Language : Formalizing Context (Expanded Notes)*, pp. 13–37. Center for the Study of Language and Information, Stanford University.

McKiernan, P. (2002). Addressing online identity: Understanding the microsoft passport service. *Information Security Technical Report 7*(3), 65–80.

Meyer, S. and A. Rakotonirainy (2003). A survey of research on context-aware homes. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*, pp. 159–10. Australian Computer Society, Inc.

Meyers, B. and A. Kern (2000). <context-aware> schema </context-aware>. In *CHI Workshop on The What, Who, When, Where, Why, and How of Context-Awareness*.

Michahelles, F., M. Samulowitz, and B. Schiele (2002). Detecting context in distributed sensor networks by using smart context-aware packets. In *the International Conference on Architecture of Computing Systems: Trends in Network and Pervasive Computing*, pp. 34–50. Springer-Verlag.

Mitchell, K. (2002). A survey of context-aware computing. Technical report, Lancaster University.

Möller, R. and V. Haarslev (2004). Description logic systems. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider (Eds.), *The Description Logic Handbook*, pp. 282–305. Cambridge university press.

Newman, W. M., M. Eldridge, and M. Lamming (1991). Pepys: Generating autobiographies by automatic tracking. In *Second European Conf. on Computer-Supported Cooperative Work*, pp. 175–188.

Pan, Z. and J. Heflin (2004). Dldb: Extending relational databases to support semantic web queries. Technical report LU-CSE-04-006, Dept. of Computer Science and Engineering, Lehigh University.

Power, R. (2003). Topic maps for context management. In *International Symposium on Information and Communication Technologies (ISICT 03)*, pp. 199–204.

Ranganathan, A., J. Al-Muhtadi, J. Biehl, B. Ziebart, R. H. Campbell, and B. Bailey (2004). Evaluating gaia using a pervasive computing benchmark.

Ranganathan, A., J. Al-Muhtadi, and R. H. Campbell (2004). Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing 3*(2), 62–70.

Ranganathan, A. and R. H. Campbell (2003). An infrastructure for context-awareness based on first order logic. *Personal Ubiquitous Comput. 7*(6), 353–11.

Ranganathan, A., R. H. Campbell, A. Ravi, and A. Mahajan (2002, July). Conchat: A context-aware chat program. *IEEE Pervasive Computing 1*(3), 52–58.

Rehman, K. (2001). 101 ubiquitous computing applications. http://www-lce.eng.cam.ac.uk/∼kr241/html/101_ubicomp.html. Date accessed: 6-10-2004.

Saif, U., H. Pham, J. M. Paluska, J. Waterman, C. Terman, and S. Ward (2004). A case for goal-oriented programming semantics. In *System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing*.

Sattler, U., D. Calvanese, and R. Molitor (2003). Relationship with other formalisms. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider (Eds.), *The Description Logic Handbook*, pp. 137–177. Cambridge University Press.

Satyanarayanan, M. (1996, February). Accessing information on demand at any location: Mobile information access. *IEEE Personal Communications 3*(1), 26–33.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications 8*, 10–7.

Schilit, B. N., N. Adams, and R. Want (1994). Context-aware computing applications. In *the Workshop on Mobile Computing Systems and Applications*, pp. 85–90. IEEE Computer Society.

Schmidt, A. (1999). There is more to context than location. *Computers and Graphics Journal 23*(6), 893–901.

Sowa, J. F. (1984). *Conceptual Structures*. The system programming series. Addison-Wesley.

Speel, P.-H. (1995, August). *Selecting Knowledge Representation Systems*. Ph. D. thesis, University of Twente.

Strang, T. (2004, September). A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*.

Tennenhouse, D. (2000). Proactive computing. *Communications of the ACM 43*(5), 43–50.

ter Horst, H., M. van Doorn, N. Kravtsova, W. ten Kate, and D. Siahaan (2002). Context-aware music selection using knowledge on the semantic web. In *Fourteenth Belgium-Netherlands Conference on Artificial Intelligence*, pp. 131–138.

Wang, X. H., T. Gu, D. Q. Zhang, and H. K. Pung (2004). Ontology based context modeling and reasoning using owl. In *2nd IEEE Conference on Pervasive Computing and Communications, Workshop on Context Modeling and Reasoning*, pp. 18–5. IEEE Computer Society.

Want, R., A. Hopper, V. Falcão, and J. Gibbons (1992). The active badge location system. *ACM Transactions on Information Systems 10*(1), 91–102.

Weeds, J., B. Keller, D. Weir, I. Wakeman, J. Rimmer, and T. Owen (2004, May). Natural language expression of user policies in pervasive computing environments. In *OntoLex 2004 (LREC Workshop on Ontologies and Lexical Resources in Distributed Environments)*.

Weiser, M. (1991). The computer for the 21st century. *Scientific American 265*(3), 94–104.