Challenges for Context Management Systems imposed by Context Inference

Korbinian Frank
German Aerospace Center (DLR)
Institute of Communications and Navigation
Oberpfaffenhofen, 82234 Wessling, Germany
korbinian.frank@dlr.de

Nikos Kalatzis, Ioanna Roussaki and Nicolas Liampotis
National Technical University of Athens
Inst. of Communication and Computer Systems
157 73 Zografou, Greece
{nikosk,nanario,nliam}@telecom.ntua.gr

ABSTRACT

This work gives an overview over the challenges for context management systems in Ubiquitous Computing frameworks or Personal Smart Spaces. Focused on the integration of context inference in today's context management systems (CMSs) we address important design decisions for future frameworks. The inference system we have in mind is probabilistic and relies on the concept of Bayeslets, special inference rules extending Bayesian networks. We show that for inference rule creation, storage, inference scheduling and update frequency the best solutions are hybrid, allowing for high flexibility and performance while reducing resource costs. We also see that human expert knowledge cannot be substituted completely in an efficient context-aware system.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies; H.3.2 [**Information Storage and Retrieval**]: Information Storage; H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Management, Performance

Keywords

Context Management, Context Inference, Smart Spaces

1. INTRODUCTION

Ubiquitous Computing Systems offer many challenges to the research community. Abowd and Schilit in [1] identified the following important components for this computing paradigm:

- Scalable Interfaces,
- Ubiquitous software services,
- Ubiquitous information,
- Support for Automated Capture and Access,
- Context-aware computing and
- Technology

Context-aware computing is a particular important one among these as it influences all the other fields, or their usability. The desired seamless integration into the environment is only possible by incorporating the users' context into decision making processes. As already shown in a host of projects and prototypes, context aware service management is fundamental to personalised ubiquitous computing. For an overview, see for instance [13]. These approaches use context in service selection, service ranking, service filtering, proactive service invocation, service handover, service configuration and service deployment – sufficient reasons that context management has to be highly efficient.

The performance of context management however not only depends on efficient access to context information in some knowledge base (be it reading by consumers or writing by context sources), but – more complicated – also on the inference algorithms and applications that produce or enrich context information. This enriched, so called *high-level* context information is necessary, as many decisions cannot be taken on pure sensor input like `temperature` coming from a thermometer, but on complex situations, like `weather`. The concept `weather` is based on a number of lower level concepts (`temperature`, `wind`, `humidity`, `air-pressure`) that can be sensed, for `weather` itself on the other hand there is no particular sensor.

This example demonstrates that a context inference rule itself is also a context consumer. The inference outcome can change every time one of its input information changes, causing new evaluation of the rule. Inference duration consequently becomes a significant parameter influencing response time and Quality of Service. Its computational complexity must not be neglected. Traditional logical (deductive) inference relies on Boolean or propositional satisfiability, the famous *SAT Problem*, which is NP complete [14, 9]. Moreover probabilistic inference in its general case is NP hard [3], even approximated inference [5]. Hence the inference algorithm has to be chosen carefully with regards to its mightiness and its performance, and with it the scheduling and triggering of inference have to be optimised.

In particular with respect to its mightiness (support of uncertainty, missing information and more, see Angermann in [2]) we have chosen Bayesian approaches for context inference. An inference rule is represented by a *Bayesian network (BN)* [15], evaluation is performed by calculating the conditional probabilities. BNs consist of nodes representing random variables and directed edges representing causal influence between the random variables. Every random variable has values and is assigned a conditional probability distribution. For Bayesian context inference random variables represent context attributes of a specified user. Context inference takes into account sensed values for context attributes in the BN as evidence and computes the conditional probability of the target context attribute. The most probable value of this context attribute will be returned together with its probability as confidence level. A concept adapting general BNs for context inference, called *Bayeslets* is described in [7].

The remainder of this paper shows in the next section the requirements to *context management systems (CMSs)* that usually do not take explicitly into account context inference considerations. Section 3 will detail how inference rules have to be created, stored and integrated in order to reduce access time, the integration of the inference itself with a basic CMS, its triggering and result handling is discussed in section 4. Finally section 5 gives recommendations for the design of integrated CMSs and an outlook how we want to pursue this research.

2. REQUIREMENTS FOR FUTURE CONTEXT MANAGEMENT SYSTEMS

One of the first context management systems capable to handle generic context information was developed for the needs of the Cooltown project by HP labs. The Cooltown context management system attempted to resolve problems regarding the context representation, while it combined and exploited context information by introducing a uniform Web presence model for people, places and things [12]. Cooltown envisioned a world where “humans are mobile, devices and services are federated and context-aware and everything has a web presence”. Various prototypes have been developed based on this perception, the most important of which are: museum exhibits that interact with the user, conference rooms that recognize users & automatically adapt to their presence, and radios that play songs based on the preferences of users in the proximity. However the provided context management system did not handle issues regarding context history, inference of context data, quality of context, context privacy and security.

The Owl [6] is a context-aware system, which aimed to gather, maintain and supply context information to clients, while protecting people’s privacy through the use of a role-based access control mechanism. Apart the provision of basic context related functionalities, the Owl project performs an initial research on more complex issues such as history of context, access rights, quality, extensibility and scalability of the contextual information [11].

Another project that provided an architecture for the provision of mobile context-aware services is SOCAM (Service-Oriented Context-Aware Middleware) [10]. This architecture models context information around four main context concepts: person, location, activity and computational entity (e.g. device, network, application, service, etc.). The SOCAM context model is specified in OWL and addresses context sharing, reasoning and knowledge reusing, while providing a service oriented middleware infrastructure for indoor applications, where a central server retrieves context data from distributed context providers and delivers them to its clients after proper processing. Although the SOCAM architecture supports context inference, there is no mechanism for storing and managing context history, while no requirements regarding the protection of the privacy of the user and group context data are addressed.

The IST CONTEXT project [20] is another project that focused on context-awareness. Its main objective is the specification and design of models and solutions for an efficient provisioning of context-based services making use of active networks on top of fixed and mobile infrastructure. CONTEXT has proved that active networks are also powerful in context-aware systems for tackling the issues of context distribution and heterogeneity. It has implemented a flexible and extensible context model, but did not support context inference, privacy & security, quality of context, history of context and group context.

The DYNAMOS project [19] aims at providing mobile users with context-aware services, focusing on proactively notifying them about services they are possibly interested in. The main context information used is the user’s personal profile, which is a combination of its personal information, preferences and schedule, including the user’s activities and a personal calendar. This information is stored and can be shared with other users, rendering the sharing of services possible based on privacy criteria set by the users themselves. Context management functionality is provided by the Contory [18] middleware that is specifically designed for resource-constrained devices, such as smart phones. A disadvantage of this approach is that functionalities that demand many computational and storage resources, such as context inference and context history maintenance & exploitation are not supported. Furthermore, no requirements regarding group context or quality of context are addressed, while the types of context information exploited are very limited.

The CroCo (Ontology-Based, Cross-Application Context Management) [16] context management service aims to support domain independent applications by handling arbitrary context data, provided by context providers and requested by consumers via a service interface. This is achieved by adapting an extensible context ontology allowing the integration of external ontologies describing contextual aspects relevant with various domains. Among others, the basic design principles of the CroCo architecture include the consistency maintenance of the distributed stored data and the reasoning of context information. As stated in [16], the architecture is not sufficiently addressing requirements regarding context history management and user privacy protection, while neither group context concepts are used.

The IST Amigo project (Amigo Integrated project[1]) has also implemented a context management system aiming to establish Ambient Intelligence features in networked home environments. For this purpose, very extended context ontologies have been developed that represent users and their environments, content, services, networks, devices, capabilities, etc., aiming to provide a complete and flexible semantic representation for middleware components and third party applications. Context inference based on ontologies was also supported, as well as history of context maintenance. Nevertheless, in Amigo, no support for distributed context management was provided, nor for group context, sophisticated context query handling or for context security & privacy protection.

Finally, for the needs of the DAIDALOS and DAIDALOS II (Daidalos Integrated project[2]) projects the CDDBMS (Context Distributed Database Management System)[17] has been designed and developed. The CDDBMS is a distributed heterogeneous multi-database system that is built in order to face the requirements of network operators and context marketplaces, while being scalable and lightweight, as it resembles to the web-server schema. It also provides more advanced features, such us context inference, query extension mechanisms and free-text based query handling. It also provides context access control mechanisms capable to enforce privacy and security protection techniques concerning the sensitive context information maintained or traded. A basic mechanism for managing context history is also provided by the CDDBMS. However, neither the quality of context concept is captured in the context model, nor the notion of group context.

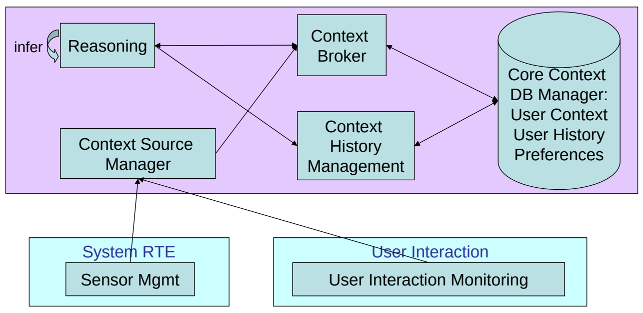
infer
Reasoning
Context Broker
Core Context DB Manager: User Context User History Preferences
Context Source Manager
Context History Management
System RTE
Sensor Mgmt
User Interaction
User Interaction Monitoring

Figure 1: The PERSIST CMS: It consists of five architecture blocks. The Context Broker is the core interface to any consumer, the Context Source Manager is the only access points for all context sources and sensors. Both components store their information in the Context Database via the Core Context DB Manager, the central information unit. Context History Management provides the necessary, pre-processed information e.g. for learning inference rules, the Reasoning finally is the place where inference algorithms are stored and processed whenever necessary.

The ICT PERSIST project[3] is dealing with context management for the needs of Personal Smart Spaces. A Personal Smart Space (PSS) is a set of services that are owned, controlled, or administered by a single user, which are located within a dynamic space of connectible devices and that are capable of self-improvement and of demonstrating pro-active behaviour.

[1]`http://www.hitech-projects.com/euprojects/amigo/`
[2]`http://www.ist-daidalos.org`
[3]`http://www.ict-persist.eu/`

Based on the above described systems the architecture shown in Figure 1 has been designed to fulfil the following set of requirements:

- Efficient Context Modelling and Semantics, in order to represent the entire set of context data (both dynamic such as location, and static such as preferences) that need to be monitored, collected, stored and utilised in PSSs, along with the necessary meta data.
- Distributed Context Management, including distributed context maintenance, access, update and synchronisation, as well as provision of a transparent interface to context management, support of ad-hoc context exchange, real-time and non-real-time context handling.
- Context Query Support. Various context queries need to be supported by PSSs, such as identity-based (or navigational) queries, location-based, semantic-based, and time-based.
- Context Source Management, including sensor data aggregation, context-source discovery, (de-) registration, configuration, etc.
- Preference handling facilities, in addition to the other context management facilities aforementioned, i.e. preference analysis, preference evaluation, and preference condition monitoring.
- History of Context Modelling and Management in order to support history-based context inference and access to past context information. In this respect, the user behaviour & status will also be modelled and recorded.
- Context Event Management, including support for distributed context event creation and propagation.
- Context Inference, i.e. extraction of high level context information from raw context data. In this respect, learning of context inference rules (CIR) needs to be supported, as well as preference learning algorithms, CIR individualisation, context association & pattern extraction / matching, and CIR learning from group knowledge.
- Group context, i.e. context information of group of persons/users, including group preferences. More specifically, the following need to be addressed: efficient group context modelling and representation, group context management & maintenance, group context estimation & inference, context prioritisation & assessment for resource sharing and context/preference conflict resolution.
- Context privacy & security. In this respect, the following need to be supported: access control over individual and group context; context integrity, reliability,

confidentiality and availability; context-based access control; privacy policy learning; etc.

- Quality of context modelling, management and exploitation, including soft context and uncertainty in context values and context inference rules.

- Context sensitivity, i.e. ability to support adaptation of the provided services to the context of their users.

3. CREATION, STORAGE AND ACCESS OF CONTEXT INFERENCE RULES

One challenge for CMSs is the handling of inference rules. When and how they are created, stored and accessed are important parameters to guarantee short response times of a CMS.

3.1 Rule Creation

Next to a "manual" creation of CIRs by a *human expert* or the adaptation of predefined template rules for an individual, there are approaches for automatic creation of Bayesian inference rules. The process used to obtain both network structure and parameters using a combination of expert knowledge and previous observations of the random variables is referred to as *learning*.

Context information pertaining to a user of Smart Electronic Spaces may in general encompass a very large range of human activities, sensor readings, information such as calendars, as well as such data relating to other people; all represented as random variables (RVs) in a very large BN. Obviously it is impossible to include all sets of such RVs in a representation used in inference, as the resulting BN would be too large. For practical relevance it will be necessary to impose boundaries on which RVs will be used to represent the information "around" her activity. These boundaries can come from constraints in the learning process (only significant causal influence above a certain threshold will be represented as a causal edge, nodes without a continuous, undirected path from the output node will be removed), but also be imposed by access limitations (context information of other persons) and human expertise.

The information used for learning, the context history, is also an important factor for learning. Usual learning methods like the one described in [4] are learning influences from a complete context history, i.e. all high-level context inference values have to be known and present in the history before incorporating them into rules. This implies that they have to come originally from human interactions with the context history, defining their current status on specific context attributes. To enable data collection among different users, a common understanding of existing context attributes have to exist which encompasses the existence of a common context ontology defining the existing context types. Also algorithms for learning from incomplete data sets (algorithms from the class of (structural) expectation maximization (EM) introduced in [8]) will not be able to give semantically meaningful and therefore usable names to new learnt context nodes.

Taking into account these considerations, it is necessary to restrict the scope of learning as far as possible, e.g. by incorporating knowledge from other already existing rules from different users or individualising templates created by human experts. But in addition the limitations on the learning algorithms have to be applied, their efficiency and performance have to be exhausted to achieve best results with shortest delays.

Rule Creation can be triggered either on demand, or by using independent mechanisms. Furthermore the update of existing rules can also be based on regular processes or by incremental incorporation of new knowledge. The decision has to take into account the system's response time, but also the quality and up-to-dateness of the responses.
As a conclusion, a CMS will have to provide all rule creation and update processes. On-demand rule creation adds significant delay on a response, given that learning (even if only roughly) takes some seconds, but is necessary as delay may still be better than giving no response at all. It can remain for the requester to decide if it continues after some seconds or disregards the answers and proceeds immediately without that information. Incremental rule updating keeps all existing rules always up to date at the cost of permanent computation load in the background and cannot create any new rule which is necessary. So even if hardly flexible, resource intensive and not need-oriented, also regular batch processes for learning rules will be necessary to discover new rules. To reduce the computational burden, these processes have to run at typical low-usage times of the systems, for instance during the nights.

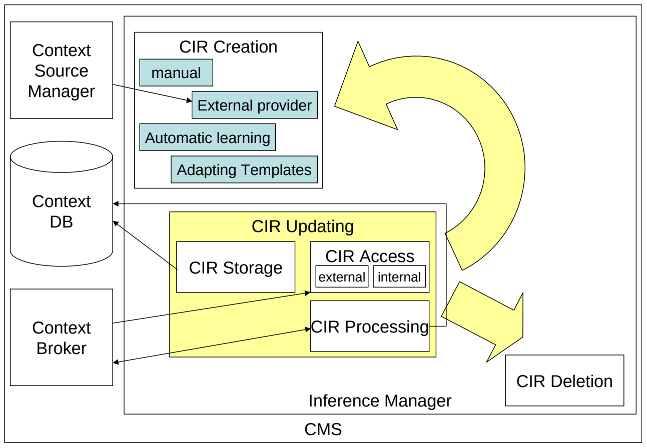

Figure 2: Lifecycle of CIRs in a CMS

3.2 Rule Storage and Access

Once rules are learnt, also their storage and access has to be managed efficiently. There are two options:

(1) Inside the CMS: High-level context inference depends very much on individual persons (e.g. situations where you don't want to be disturbed differ significantly), so personalised versions of each rule (specified by the outcome context information) have to be stored. This would cause intensive load on a centralised back-end server in a large scale ubiquitous computing system given that every user has tens of such rules. It is still manageable though with today's replication and load balancing systems, keeps the information synchronised and allows for access from everywhere. If the CMS is not deployed in the back-end, but on the user's mobile device itself, bigger synchronisation problems arise. Mobile devices are resource limited, may be used by different users and can group and ungroup dynamically with other mobile devices. A consistent state is more difficult to achieve, but users can benefit from reduced communication and higher privacy, if their personal settings are only local.

(2) Outside the system: As an alternative, rules can also be out-sourced. They can be represented as services that firstly register with the CMS as context consumers for the input information and secondly also register as context sources that feed the outcome of the inference back into the system. The deployment of such services can be independent of the CMS. Access control, privacy protection and the integration with the history of context for rule updating is more difficult however. Also the control of the CMS over execution and update is reduced. A positive option with this approach is the possibility to incorporate formerly unknown rules that may be provided with third party services.

Subsuming we can say that it is still preferable to store the rules within the context management system. Some flexibility is sacrificed for higher control and by that up-to-date rules, guaranteed privacy protection and system immanent access control. Third party inference rules still can be offered by approach (2). Within approach (1), a combination of the centralised and user-centric approach is probable. To have the necessary rules on the local devices reduces unnecessary remote requests, but probably cannot have all possibly necessary information. In these cases the best option is to have a back-end based system as fall-back solution. The resulting integration of CIRs in a CMS with its life cycle is shown in Figure 2.

4. INTEGRATION OF INFERENCE WITH CONTEXT MANAGEMENT SYSTEMS

Once a solution about the maintenance of the inference rules has been found, it has to be assured that also the inference process is running as fast as possible and scheduled efficiently, i.e. that inferred context information is sufficiently current, but also to avoid unnecessary resource consumption.

4.1 Bayeslets: a way to reduce inference time

The inference complexity *NP-hard* of Bayesian inference is determined in the end by the number of nodes in the network and the size of the conditional probability tables (CPT) associated to the nodes. The size of the CPTs in turn is influenced by the number of parents, i.e. the number of edges and the number of states, every node can have. Hence we have to reduce the complexity in all parameters:

(1) Number of nodes in the Bayesian Network

(2) Number of values per node

(3) Number of edges per network

As already indicated in section 1, we have developed a concept based on the afore mentioned "Bayeslets" that can reduce the inference time by improving all three factors. Following the principle of *divide & conquer*, a Bayeslet consists of small stand-alone BNs where nodes can be assigned the functionalities *Input* and *Output Node* that serve to dynamically connect them to each other if required (for more information see [7]). Combining these Bayeslets based on an entropy, that only lets connect other Bayeslets that provide a significant win in information and certainty, keeps the number of random variables involved in the inference process at a minimum – reducing parameter (1). Even distributed inference becomes possible if different Bayeslets can be evaluated on different devices. Furthermore it is a working concept for accessing different users' context information in a controlled way.

Parameter (2) will be reduced by a dynamic reduction of value ranges based on an entropy that selects the currently relevant values on a personal basis and adapts quickly to changed settings. Details on this procedure will be published shortly. Appropriately configured learning of Bayeslet inference rules as described in subsection 3.1 moreover can easily limit the number of edges created in a BN, more precisely the number of incoming edges per nodes and thereby parameter (3).

A question to be further investigated on is about the size of a Bayeslet and the separation of complete causal inference networks into Bayeslets. Most probably this separation will have to incorporate human expert knowledge to a large extend like in section 3.1. Although some theoretical limitations can be incorporated, e.g. no *d-separated* nodes [15] should be contained in a single Bayeslet. Sensors can be represented in a single Bayeslet, so that its measurements can be reused for different inference rules or even can be combined with other sensors to increase certainty about the represented concepts. We expect each Bayeslet to contain between $n \in [5, 10]$ random variables with not more than $p \leq 3$ ($< n$) parent nodes, where not more than $s \leq 5$ relevant states per random variable are necessary. This still encompasses huge CPTs with $s^{p+1} = 5^4 = 625$, but without these reductions the CPT size and the inference time would be far worse.

4.2 Inference Scheduling

Such Bayeslets should never be called by the requester itself. It should be transparent to him, if the information is inferred at all or if it was stored directly by a context source. Consequently the CMS is receiving all requests and forwards them to the context inference system if necessary, i.e. if no appropriate current information is available in the CMS. The easiest way to have always the current information stored in the database, is to permanently evaluate the inference rules, but this results in unnecessary evaluations and lots of storage processes that will not have been used before they are overwritten.

The other extreme is to initiate inference only on request, as shown in Figure 3 – which is problematic, as response time increases and more important: it would eliminate the option to register for context updates on high-level context. Hence preferences, proactive service invocation rules or other in-

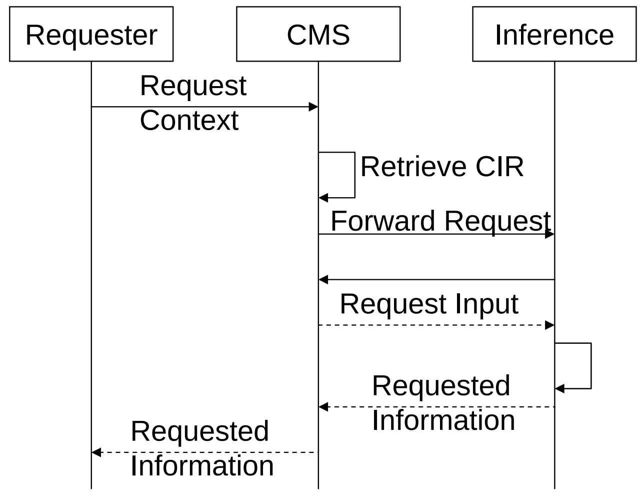

Figure 3: Message Sequence Chart: context inference on demand

ference rules could not be based on them. To enable such context subscription, continuous inference based on the update of the requested rules' input nodes has to be offered.

To avoid any continuous inference, but enable subscription, the only viable solution would be to change the subscription in a subscription for every input-node of the respective inference rule. As soon as any of these events occurs the requester would be notified and can request on-demand inference of the rule. On the other hand this is not efficient with regards to CPU consumption if more requesters register for the same rule. Moreover response time is worse than with continuous inference.

In order to avoid unnecessary evaluation, another time a hybrid solution will perform best. No rule that was not explicitly requested should be evaluated continuously, but as soon as it is requested it should be evaluated. Not on a time based regular basis, but on necessity, i.e. depending on the input information, like Figure 4 shows.

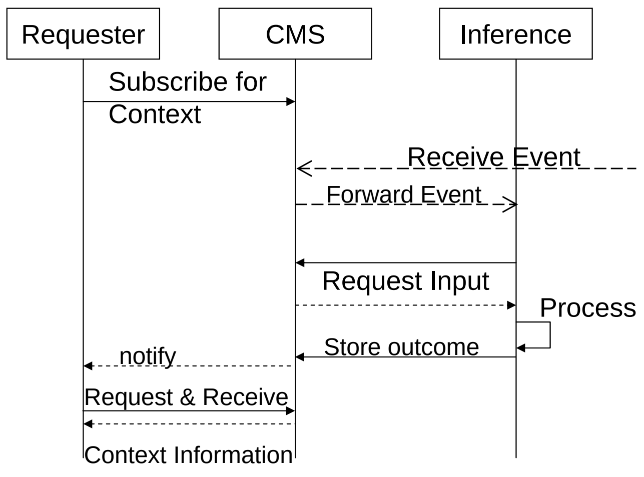

Figure 4: Message Sequence Chart: continuous context inference

Like that we can guarantee for the always up-to-date information. A remaining question however regards the storage of the continuous inference outcome. It may be either returned directly to all subscribers or be saved in the CMS. The former is neglecting the needs of on-demand context consumers that could also benefit from this already performed evaluation, the latter could cause problems with the up-to-dateness of information, if the CMS detects available context information and does not cause re-evaluation for a new on-demand request. Concluding, storage in the CMS is preferable, but this information has to be marked as inferred and assigned with a measure of up-to-dateness.

4.3 Update Frequency

A remaining problem with the compromise solution proposed above is an estimation of the update frequency of an inference rule $f_{inference}$, that depends on the update frequency of its input information $input_i$, f_{input_i}. To always have the most up-to-date information, the following equation holds:

$$\max_i(f_{input_i}) \leq f_{inference} \leq \sum_i f_{input_i}$$

If there is one input information coming from a sensor with frequent measurements (e.g., an *Inertial Navigation System (INS)* provides measurements in the order of 100 Hz) this would cause so frequent rule evaluation that storage and inference only may take a few milliseconds. If the relevancy of this inference is not too high, this would be a waste of resources. On the other side there are already cases of inference, e.g. the estimation of hazardous situations in cars, where the prediction frequency has to be higher than the update rate of input sensors.

Connected to such considerations, we also have to ask: would sensors store data with their rate of 300 Hz or more in the Context DB? On the one side no, to reduce costs and resource consumption, on the other side yes to allow processing of the raw data by other interested services. Probably, a good solution is to abstract from the sensor itself by means of Bayeslets. The concept represented by the sensor then can already be discretized, its values partitioned to meaningful, semantical rich states. These states would change less frequently though still provide all the input necessary for other applications. We can assume that safety critical applications bring along their own sensors where they can access the raw measurements.

All in all we learn that the necessary update frequency per context information depends on the context information and its use case. It will not be possible to completely abstain from making use of expert knowledge or human made ontologies providing this information.

5. CONCLUSIONS AND OUTLOOK

In this paper we have discussed five questions regarding the integration of CIRs in CMSs:

1. When and how should CIRs be created?
2. Where should they be stored and how accessed?
3. How can inference time be reduced?

4. How has inference to be scheduled or triggered?
5. When has inference to be updated based on its input?

Question 1 requires a combination of batch learning processes and incremental learning, applying scope restrictions with algorithmic and automatic means, as well as with the help of human expertise. Storage of the resulting CIRs, question 2, should be inside the CMS, on mobile devices as far as possible, but backed up in the back end. A promising approach for question 3 is the *Bayeslet* concept described in section 4.1, question 4 requires a hybrid solution of on-demand and continuous inference with results stored back into the CMS under appropriate meta information. For question 5 finally, we propose not to store high-frequent raw sensor measurements in the CMS, but already semantically enriched, less volatile information that is valuable for many purposes.

As the next step, these assumptions will have to be verified in a realistic evaluation of a context management systems, as it is developed in the project ICT PERSIST. Interesting input to be expected is also about the kind of user interaction with a context aware system on mobile devices, which will be necessary to incorporate expert knowledge.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 215098 of the *Persist* (PERsonal Self-Improving SmarT spaces) Collaborative Project.

6. REFERENCES

[1] G. D. Abowd and B. N. Schilit. Ubiquitous computing: the impact on future interaction paradigms and HCI research. In *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*, pages 221–222, New York, NY, USA, 1997. ACM.

[2] M. Angermann, P. Robertson, and T. Strang. Issues and requirements for Bayesian approaches in context aware systems. In *LoCA*, pages 235–243, 2005.

[3] G. F. Cooper. Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, Stanford, CA, May 1990.

[4] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 09(4):309–347, October 1992.

[5] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artif. Intell.*, 60(1):141–153, 1993.

[6] M. R. Ebling, G. Hunt, and H. Lei. Issues for context services for pervasive computing. In *Proceedings of the Advanced Workshop on Middleware for Mobile Computing*, Heidelberg, Germany, 2001. Springer.

[7] K. Frank, M. Röckl, and P. Robertson. The Bayeslet concept for modular context inference. In *Proceedings of UBICOMM08*, Valencia, Spain, 2008. IEEE Computer Society.

[8] N. Friedman. The Bayesian structural EM algorithm. In *In UAI*, pages 129–138. Morgan Kaufmann, 1998.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.

[10] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 270–275, 2004.

[11] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In F. Mattern and M. Naghshineh, editors, *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, pages 79–117, London, UK, 2002. Springer-Verlag Berlin Heidelberg.

[12] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. People, places, things: web presence for the real world. *Mob. Netw. Appl.*, 7(5):365–376, October 2002.

[13] F. Klan. Context-aware service discovery, selection and usage. In *Proceedings of Workshop Grundlagen von Datenbanken 06*, pages 95–99, 2006.

[14] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. In *19th International Conference on Automated Deduction (CADE-19)*, 2005.

[15] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[16] S. Pietschmann, A. Mitschick, R. Winkler, and K. Meißner. Croco: Ontology-based, cross-application context management. *Semantic Media Adaptation and Personalization, International Workshop on*, 0:88–93, 2008.

[17] C. Pils, I. Roussaki, T. Pfeifer, N. Liampotis, and N. Kalatzis. Federation and sharing in the context marketplace. In *LoCA*, pages 121–138, 2007.

[18] O. Riva. Contory: a middleware for the provisioning of context information on smart phones. In *Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, pages 219–239, New York, NY, USA, 2006. Springer-Verlag New York, Inc.

[19] O. Riva and S. Toivonen. The DYNAMOS approach to support context-aware service provisioning in mobile environments. *J. Syst. Softw.*, 80(12):1956–1972, 2007.

[20] S. Xynogalas, M. Chantzara, I. Sygkouna, S. Vrontis, I. Roussaki, and M. Anagnostou. Context management for the provision of adaptive services to roaming users. *Wireless Communications, IEEE*, 11(2):40–47, Apr 2004.