An Object-Oriented Model in Support of Context-Aware Mobile Applications

Felix Dobslaw[1], Aron Larsson[1,2], Theo Kanter[1], and Jamie Walters[1]

[1] Mid Sweden University, Sweden
[2] Stockholm University, Sweden
{felix.dobslaw,aron.larsson,theo.kanter,jamie.walters}@miun.se

Abstract. Intelligent and context-aware mobile services require users and applications to share information and utilize services from remote locations. Thus, context information from the users must be structured and be accessible to applications running in end-devices. In response to this challenge, we present a shared object-oriented meta model for a persistent agent environment. The approach enables agents to be context-aware facilitating the creation of ambient intelligence demonstrated by a sensor-based scenario. The agents are context-aware as agent actions are based upon sensor information, social information, and the behavior of co-agents.

Keywords: Context-Modeling, Context-Awareness, Ontologies, Ubiquitous Computing.

1 Introduction

The growing number of mobile devices and users; the increase in the devices' computational power and the use of Internet as a platform together with new and efficient sensor techniques opens new possibilities for creating intelligent sensor-based mobile services. Recently, vendors of mobile devices such as smart phones and PDAs have started to embed sensors including accelerometers, GPS, light, and short-range radio in the devices. Thereby we are in a position to further exploit access to sensor information in intelligent services. Users of electronic services are also expecting applications and services to provide more value by means of taking into account social and real-time information about the user's real (or perceived) environment.

With the proliferation of mobile services and on-line communities, there is a growing need to be able to provide more customized services based on user's context information in a broader sense than by purely physical data. For instance, services ranging from simple ones such as delivering customized and location-based advertisements, to more complex tasks including social interactions over time and access to shared digital content and media in an intelligent manner. Access to both sensor information and social contexts can substantially increase the perceived value of mobile services to users. Sensor information may originate from user devices equipped with sensors or from sensor networks surrounding

Y. Cai et al. (Eds.): Mobilware 2010, LNICST 48, pp. 205–220, 2010.

the user; providing a physical context. On-line services that manage a social context further provide individuals with an accessible *virtual life*. The exploitation of such information in order to provide intelligent services to users is commonly referred to as ambient intelligence [1]. Ambient intelligence requires a context-aware environment since context information describing the physical context of a user originates from sensor information. In such an environment, the integration of sensors and sensor information is an important feature in order to create physical situational context-awareness.

1.1 Motivation

The work behind this paper is mandated by the fact that mobile services benefit from having access to information regarding both a user's situation and intentions, as well as information regarding the user's social activities in order to support the user in achieving tasks. In order to reach this, a framework enabling intelligent mobile services must be able to support reasoning with sensor information and social information simultaneously. For this purpose, we advocate a common context model incorporating both these aspects of contextual data.

The interest in ambient intelligence as a fundamental facilitator of intelligent mobile services has increased substantially; and the idea of utilizing sensor information in mobile devices is not novel. As for technical solutions enabling the sharing of physical context data; the MobiLife project [2] realized the provisioning sensor information via 3G mobile systems. However, the solution proposed is based upon web-services incapable of maintaining the correspondence between the location of the end-devices and the reachability of sensor information.

Other approaches include SenseWeb [3] and AmbieSense [4]. With respect to the former, applications can initiate and access sensor information streams from shared sensors across the entire Internet. The SenseWeb infrastructure provides support for an optimal sensor selection for each application and the efficient sharing of sensor streams among multiple applications based on web services. However, this is achieved using centralized solutions which negatively impacts on scalablilty. The approach of the AmbieSense project involves the use of so called *context tags* which is a special-purpose hardware device (and a small wireless web-server) mounted in the surroundings and communicating with adjacent mobile devices. In this way, the relevant information can be provided to mobile users based upon the users' current context. While this provides an acceptable means of gathering context information, the requirements with respect to specific hardware, will impact on user adoption and leaves open the possiblity of deriving context in a more seamless manner based on information that is derivable through the deployment of software over existing mobile devices.

With respect to context models and agent environments in [5], an ontology based context model and an architecture for the specification of intelligent context-aware services is presented. The approach to context modeling presented in this paper goes beyond existing approaches, emphasizing the model's extendibility

combined with the ability to translate context information with similar semantics but differing syntaxes. It also addresses the potential issues with regards to query processing times that are inherent with ontology approaches.

The involving of sensing and reacting on the environment has much in common with the field of artificial intelligence; in particular intelligent agent theory and multi-agent systems. In this area, context-awareness of an intelligent agent refers to the agent having the ability to sense and react (perform actions) based upon the state of the environment, cf. [1,6]. Portable devices, such as mobile phones, could then be made seemingly context-aware to the extent that software services implemented in the devices are triggered by context-aware agents and their actions.

For a distributed agent environment to operate, a number of basic issues need to be resolved in order to enable and facilitate reasoning in such systems. These basic issues, already highlighted in [7] and [8], include: 1) How to make agents in a distributed system context- aware, 2) How to enable these agents to communicate and interact, and 3) How to facilitate for these agents to reason about sensor and social information. Of particular concern and what constitutes the main challenge in this paper is to show how the first issue is resolved combining sensor and social data.

An additional challenge involves providing the means for the spontaneous sharing of context information from any locally available sensor among agents, also enabling sharing of social information with other agents and enable reasoning with such contextual information. Sharing of remote and local context information requires a context model with the ability to capture the context of users or services and also maintain and update this context during which the service operates. In order to enable reasoning, a context-base and general context model also serving as an agent environment for persistent and context-aware agents is proposed in this paper.

The paper is structured as follows. Section 2, Approach, describes the proposed CII-model and its components in detail. This is followed by an account for how the model could serve as an agent environment and a scenario, highlighting the models features. Section 3, Realization of the Approach, first explains the general architecture for a context-base based on the CII-model, followed by how this context-base provides a necessary foundation for the desired functionality of the MediaSense framework. Section 4 concludes the paper with a summary of its contributions and an outlook on future research and development.

2 Approach

In this paper, a user is the holder of a mobile device and a service is the result of a set of actions initiated by a set of agents which are exploited by a particular application. We conform to the definition of context as given by [9], however slightly extending it to include interactions between applications as this is an important feature of the proposed framework. Thus, semantically speaking, 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 an application or between two applications; including the user and applications themselves.

Our approach for meeting our objectives comprises of a distributed system of agents as first-class objects reasoning over sensor and social information. We conform to the agent paradigm given in [10]; in that an agent is an entity that can perceive its environment through sensors and acts upon that environment through actuators. Thus agents are defined as autonomous entities that make decisions upon perceived stimuli and desires. The environment, or the shared structure of the agents, is the base for enabling agent reasoning and intelligent services. The proposed agent environment (The Context Information Integration Model) operates on an object-oriented context-base organized as an ontology that supports integration of context from customized multi-dimensional domains. The agents are software objects spawned in the context-base; able to move between end devices to be executed locally as a service. They can reason upon the global context by remote access to the object meta model. See Figure 1 for an illustration of the overall approach.

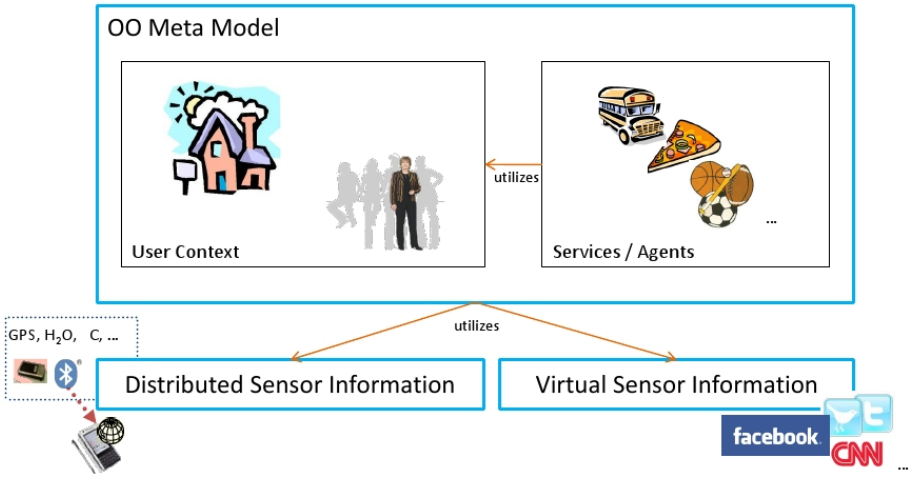

Fig. 1. Illustrative overview of the approach. A user's context is captured from a multitude of sources. Agents are persistent and active objects in the context-base.

2.1 Ontologies

An ontology is a 'formal, explicit specification of a shared conceptualization' [11]. This shared conceptualization or understanding can help with overcoming communication barriers between people, organizations and software systems. Each party in a plot tends to have a particular viewpoint on a matter and diverse, possibly overlapping concepts. This fact constitutes misunderstandings and thereby poor communication, implying 'difficulties in system specification' [11].

The most basic exemplification regarding that, is the bidirectional communication between two individuals not sharing a common language. The language in this scenario contains none or partly overlapping concepts. The application of an ontology would allow for the specification of a meta-language (a shared understanding, an Esperanto) and solve related problems, given that the metalanguage is sufficiently rich in expression. Concepts and relationships for ontologies can vary in representation from highly informal (e.g. spoken language) to rigorously formal (formal semantics, axioms, theorems and proofs of important properties), depending on the purpose of the model. As different problem domains may have syntactically different concepts with the same or similar semantics, enabling overarching context-awareness in several problem domains simultaneously requires some technique for translating between syntaxes and units. When assimilating syntactically different concepts with the same or similar semantics, a translation is required to enable their content to be translated from and to one another. A pattern for the generic resolution regarding this problem is used in the proposed CII-model. This pattern is referred to as Inter-Lingua. Inter-Lingua describes the necessity for one central, globally accepted representation. For each syntactical representation of a concept not conforming with the standard, a translator to the inter-lingua concept and vice versa must exist. In this manner, a valid translation from each model to another is guaranteed. This considerably reduces the system's translation complexity (see [11]).

2.2 The Context Information Integration Model

The Context Information Integration (CII) model, is an ontology with respect to the above definition. Figure 2a shows the concepts in UML class-diagram notion. Top-down, the model is partitioned into three components: *Internet of Things*, *Sensed Context* and *Context Meta Data*. Within the *Internet of Things*, concrete instances of concepts and their relationships to each other can be modeled. The result is a graph where nodes represent concept instances and the directed edges are Resource Description Framework (RDF) triples $\langle subject, predicate, object \rangle$, the standard for resource interrelations on the Semantic Web [12].

An example of a resulting graph can be seen in Figure 2b. Sensed Context synthesizes all physical, virtual and logical sensors as classified in [13]. An example of a physical sensor is a thermometer whereas a virtual sensor could be a Twitter feed. Logical sensors are more sophisticated, integrating different data sources or sensors; for instance, in order to draw conclusions about the state of health of a patient. The Context Meta Data component specifies guidelines for the extension and integration of new concepts into the existing ontology. The following subsections provide a component wise explanation of the concepts.

Sensed Context. `Information Sources` are the perceiving objects in the model. Each `Information Source` holds one value at a time; an instance of `Context Value`. An `Information Source` could be any physical sensor, such as a sensor for GPS, but it could also be any type of source on a local machine or the Internet revealing information such as user profiles and preferences. `Context`

`Values` represent sensed context data in the model, for instance, a GPS coordinate $(123.23, -12.34, 144.40) \in \text{longitude} \times \text{latitude} \times \text{altitude}$. Furthermore, the model allows for `Information Source` inheritance, and thereby its extension by specific sub-concepts. `GPS Sensor` and `Camera` are sub-concept examples for `Information Source`.

Internet of Things. `Entities` represent the things in the model to which context can be attached. `Predicates` give relations between two `Entities` a semantic meaning, forming subject-predicate-object triples that allow for social information. The order in which they are written is relevant (symmetry is not implied). Each `Entity` can be related to multiple other `Entities` in two ways, either as a subject or an object of a triple. `Entities` in the model allow for inheritance in the same way as `Information Sources` do.

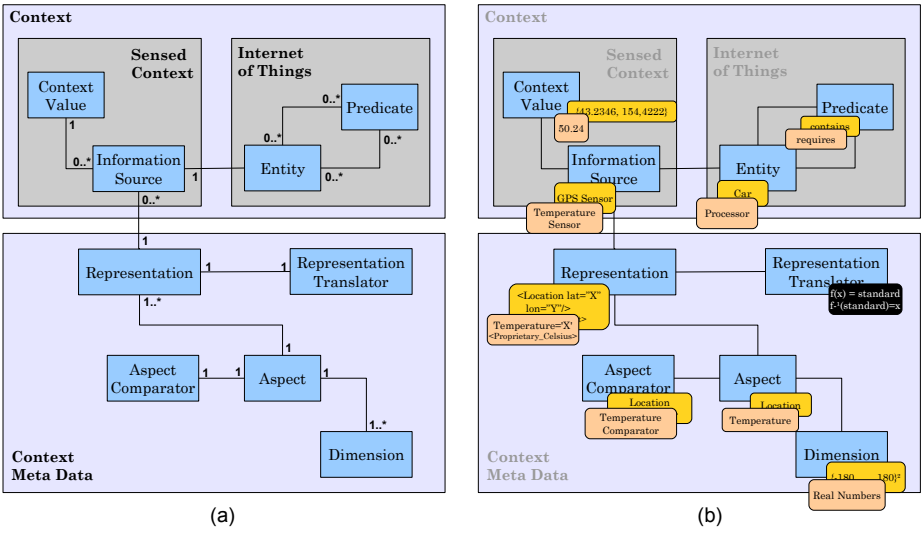

Fig. 2. (a) Shows the CII-model, represented as a UML class diagram. The concepts (classes) are categorized according to their role in the model as *Sensed Context*, *Internet of Things* and *Context Meta Data*. (b) An example of how model components could look like. A simplified scenario could involve a processor with a temperature sensor (red) and a Car with a GPS sensor (orange).

`Context Values` can be attached to and detached from `Entities`. The same accounts for relations between `Entities`, allowing for context dynamically changing over time.

Context Meta Data. The central meta data concept in CII is `Aspect`. Such an `Aspect` could be *location*. Each `Aspect` has any desired amount of `Dimensions` and `Representations`, requiring at least one of each. One particular `Representation` has to be specified as the `Aspects` *standard*. The *standard* serves as the *inter-lingua* for the integration of different `Representations`, and

thereby resolves any syntactical, quantitative, qualitative or unit based distinctions with respect to one `Aspect`.

The `Dimensions` specify the valid domain for all `Context Values` of a particular `Aspect`. The three `Dimensions` for `Aspect` *location* could be called longitude, latitude and altitude; assuming context values within the domain $\{-180, \ldots, 180\}$ for longitude and latitude, as well as "meters over sea level" for altitude ($\mathbb{R}$). The largest valid domain for a `Context Value` is the Cartesian product of all its `Dimensions`.

`Representations` are templates for the presentation of sensed context data. Each `Information Source` holds a `Context Value` of one particular `Representation`. An example `Representation` for the `Aspect` *location* could be called *gmlPoint*, where the `Context Values` template might look like the standard Google representation for location data:

```
<position>
    <gml:Point srsDimension="3">
    <gml:pos>X Y Z</gml:pos>
    </gml:Point>
</position>
```

X, Y, Z stand for the longitude, latitude and altitude values, respectively.

`Representation Translators` are responsible for the translation from one particular `Representation` of an `Aspect` to the standard `Representation` (*mapToStandard*) and vice versa (*mapFromStandard*). In the following example two `Representations`, *gmlPoint* and *proprietary*, for an `Aspect` *location*, are considered. *gmlPoint* serves here as the *standard*. The `Representation Translator` for *proprietary* has to supply the two functions

$$f_{trans}(x_{proprietary}) = y_{gmlPoint} \tag{1}$$

$$f^{-1}_{trans}(y_{gmlPoint}) = x_{proprietary} \tag{2}$$

The `Representation Translator` ensures the transitive translation closure for all `Representations` of one particular `Aspect`, meaning that the model itself is able to translate between all types of `Representations` for that `Aspect`. This is a key feature for the integration of context in different formats and units.

The `Aspect Comparator` enables CII to operate between `Context Values` of the same `Aspect` even if they have different representations (units, syntaxes). It incorporates the functionality of ordering context values with respect to the shared `Dimensions`. Therefore, it is required to specify an `Aspect Comparator` for each `Aspect`. The `Aspect Comparator` enables that CII, without knowing anything about the *Aspects* or context contents, is capable of optimizing its retention. The `Aspect Comparator` provides two functions. One to compare two `Context Values`, returning their order in relation to an implicit but well-defined metric (*compareTo*). The other function returns the relative distance between two `Context Values` with respect to that particular metric (*getDistance*). This data could be of special interest to applications that only know little about a certain, newly explored, `Aspect`. For instance, an agent could use distances and

orders for reasoning and decision making. Naturally, comparison is only possible for measurable data. Figure 2b adds two modeling examples to the general model figure. By extending the CII-model, its semantic expressiveness increases.

2.3 How Context Is Expressed

The real essence of the model is to provide `Entities` with contextual meaning. It is important to link the structures of the CII-model to these meanings, so as to obtain a correlation between the syntactical constructs and their semantic interpretation. This is done in the following definitions.

Definition 1. *The* ***context*** *of an* `Information Source` *at a moment in time is defined by its* `Context Value`*.*

Definition 2. *The* ***simple-context*** *of an* `Entity` *at a moment in time is the set of all attached* `Information Sources` *contexts.*

Definition 3. *The* ***context*** *of an* `Entity` *at a moment in time is its simple-context including the simple-context of all* `Entities` *in its network proximity under consideration of the type of their relationships. How proximity is defined, depends on the eye of the beholder.*

The context of an `Entity` is not expressed in an entirely explicit manner. Context may depend on the subjective interest of the user or other stakeholders, which is why context is partially implicit in the CII-model. That is also the reason why *situations*, unlike other approaches [14], are not taken into account inside the CII-model.

The model viewed at a moment in time always represents a snapshot of the current overall context. It is possible to extend CII, expressing historical context for `Information Sources`. When extending CII with respect to historical context, the 1..1 relation between `Information Source` and `Context Value` becomes a 1..$*$ correlation. Timestamps mark the `Context Values`, so as to allow the determination of their time-frame of validity.

2.4 The CII-Model as an Agent Environment

We assume an environment in which many users (possibly millions) are concurrently following certain interests, where the agent's main objectives are to serve these interests as well as possibly fulfilling their own desires. The importance of the environment in multi-agent systems has been raised in [15,16], arguing in favor of the environment as a first-order abstraction having two different roles: 1) providing the surrounding conditions for agents to exist and 2) to provide an exploitable design abstraction for building multi-agent applications. Herein, the environment is defined by the CII-model extended with protocols for agent communication, conforming with the Agent Communication Language (ACL) agents communication standards from the Foundation for Intelligent Physical Agents (FIPA) [17]. The CII-model serves as a first-class abstraction: it is an

independent building block holding different responsibilities than those of the agents within the system. This means that the behavior of the system is not only a result of the combined actions and desires of the agents; as rules for agent activities with respect to, e.g., accessibility and communication, are defined by the CII-model as the environment.

In this framework, physically an agent is a first-class object represented as an active `Entity` in the context-base. Thus, the existence of the environment is not dependent on the agents in the system, the agents are rather seen upon as a part of the environment. By active, we mean that an agent, as oppose to passive entities (or ordinary objects), is capable of perceiving events, perform actions and make commitments, cf. [18]. Hence, an agent is viewed as a specialization of an object's building on an object-oriented approach, analogous to the agent paradigm of multi-agent Systems Engineering (MaSE) for the analysis and design of multi-agent systems in general, see [19]. This can be realized by hybrid agents that are part of the CII-model, as well as the Java Agent Development Environment (JADE), by inheritance. JADE is an object oriented agent environment, based on the standards imposed by the FIPA [20]. Of further importance, as an agent is represented as an `Entity`, is that the agent's internal states can be persisted which serves as a condition for agent recovery, resilience, accessibility and mobility.

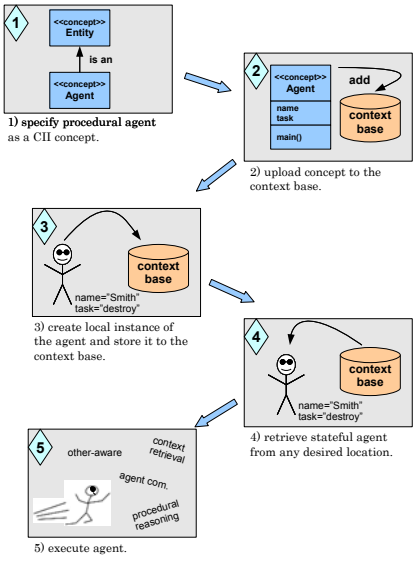

Fig. 3. Illustration of how the agent paradigm concept can be enabled by the CII-model. 1) The agent's internal logic is locally defined, 2) The agent class is persisted, 3) Agent instances can be created and persisted, 4) Agents can be globally retrieved. Other-awareness can be enabled as each agent has an accessible internal state as well as access to the global context information.

2.5 Scenario

A potential scenario benefiting from the CII-model would be a traveler/commuter scenario. Commuters are interested in traveling conveniently, regarding price or (and) travel time. Since public traffic requires the concurrent usage of resources (streets, rails etc.) which is affected by dynamically changing conditions (weather, building sites etc.), planning a multi-hop trip in advance adds a realistic chance of delayed arrivals and subsequently missed connections.

Travelers, equipped with mobile devices, require a service that keeps them up to date about the current opportunities in case of an unplanned schedule change. We suggest the modeling of this service as an agent, such as discussed in 2.4.

When changing from regional busses to local busses, in a city, providers don't always share the same data model or services for the finding of direct connections. However, they could offer agents with a standard interface to allow for context-based recommendations for ad-hoc journeys. Busses are equipped with sensors such as GPS, etc. These sensors are represented in the model as sub-concepts of `Information Source`; `Bus` and `Traveler` as sub-concepts of `Entity`. Each instance is represented by an unique identifier. Travelers are registered to be on a bus by adding RDF triples to the context model $\langle travelerID, isOn, busID \rangle$. In 3.3 we present an approach for how the travelers devices can interact with the sensors, retrieving their context values, updating them to the context model. In response to the state of their computing environment with respect to resources, agents may be executed locally on a device; be opportunistically mobile or in some cases have their state persisted until resources become available.

The advantage of modeling in CII here is that context information can be arbitrarily shared with other domains, since the model is based on the same ontological meta concepts. Car rental, emergency, or event-services could derive their conclusions from, or manipulate the same context pool. Not only could the idea be further extended to air traffic, cab services or trains; with the case of a bus accident, passengers with special needs could be identified since the context database would reveal that they were on the bus. Thus, the ambulance rushing to the crash site could be equipped with the required medication or apparatus. The extendibility of the model allows for all stakeholders to support different formats and units for how the context is expressed, so that existing modules can remain untouched (e.g., kml vs. gml for location information).

3 Realization of the Approach

3.1 The Context-Base

Our proposed architecture for a context-base is data-centric, comprising three layers: the context-aware application, the context-base controller and the physical context storage, see Figure 4a. A data-centric approach is applicable to our problem of global context provisioning, since it addresses the communication via a common repository [21]. However, this repository does not have to reside on a central point. The context-base can be viewed as a middle-ware that hides complexity regarding the distribution of data, processing, application and control from context consumers/providers, providing a high level of distribution transparency. From the perspective of a context-aware application, the context-base is a rich service for the management of context and context models. The application could be: a) manually controlled by end users, or b) perform automated, context related tasks. Developers of context-aware applications either reuse existing domain-model concepts from the context-base or extend the global model. Figure 4b shows the context controller with a higher granularity. The controller has two main tasks with respect to the functionality outlined in 2.2:

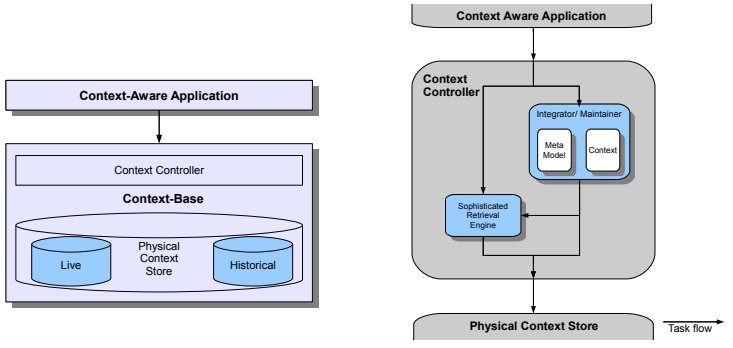

Fig. 4. (a) Views the proposed layered architecture for context attainability, utilized via a context-base, encapsulating the CII-model. (b) Depicts the context controller component. It is responsible for the creation and maintenance of the meta model as well as the integration of domain model components and likewise for the provision of the retrieval engine to the context-aware applications.

1. Handling data operations, mediating between external requests (data from information sources) and the physical data store.
2. Maintaining the context model including alterations, extensions and validity enforcement inside the context-base.

For both tasks, a sophisticated retrieval engine is essential, allowing for access to the global context. The controller is preferably thin and stateless in order to enable the distribution of multiple controllers. The idea behind using multiple controllers is to support concurrent access to the global context-base while offering high scalability. Where the Context Integrator/Maintainer is dealing with Task 1, Task 2 is dealt with by the Meta Model Integrator/Maintainer component. It is crucial to deal with both resilience and robustness as, within a distributed scope, flawed model extensions should only have local impact and should not harm the overall usage of the context-base.

3.2 User Interfaces

Two interfaces are presented, placing the CII functionality at the disposal of the context prospects. From the perspective of the context-base, there are two user groups: context providers and context consumers. Providers create domain models and reuse existing model extensions, and thus modify the schema itself. Consumers query, add, modify, and remove context. Hence, consumers only request or change the context information itself. Context providers supply the consumers with services/applications that are, to some extent, context-based or context-aware. The interface is split in two with respect to this classification (see Figure 5).

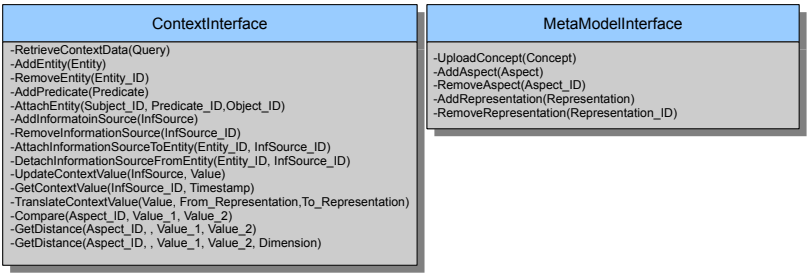

Fig. 5. The two interfaces that allow for context access. ContextInterface (left) allows for the treatment of context, where MetaModelInterface (right) dealing with the treatment of the meta model.

Context Interface. The interface should be made available to all applications having an interest in context storage and retrieval. It allows for context retrieval via a dedicated method *RetrieveContextData* that returns objects. Concrete `Entities` and `Information Sources` can be initialized and stored in the context-base together with their interrelations. `Context Values` can be updated via the ID:s of their respective `Information Sources` (*UpdateContextValue*). Current or historical values can be retrieved by the provision of the ID and the desired timestamp of validity (*GetContextValue*). For the launching of agents, methods for translation, comparison, and distance computation are provided.

Meta Model Interface. The functionality of the Meta Model Interface should be restricted to a system modeler with permission to alter the meta model. `Aspects` and `Representations` can be added, modified, and removed on demand. The most significant method is *UploadConcept*, which allows for a domain level modeler to extend the model with new sub-concepts of `Entity` and `Information Source`, or the addition of `Aspect Comparators` and `Representation Translators`. In an object oriented environment, these concepts would be presented as classes following a standard interface. This integration would be made possible by adaptive software techniques such as a combination of computational reflection, automated compilation and dynamic class loading.

3.3 The Extendable Model Agent Environment

Providing the necessary structure and technical architecture for enabling intelligent services, focusing on the incorporation and utilization of sensor information is the initial main issue of the MediaSense project [22]. These solutions can at present be summarized into the use of a wireless sensor network gateway (WSNG), utilizing low-energy Bluetooth communication for providing mobile devices with various forms of sensor information, in combination with the Distributed Context eXchange Protocol (DCXP) enabling distributed sharing of context information in real-time. Hence, agents update their state and behavior in response to exchanges of sensor information from other agents or sensor

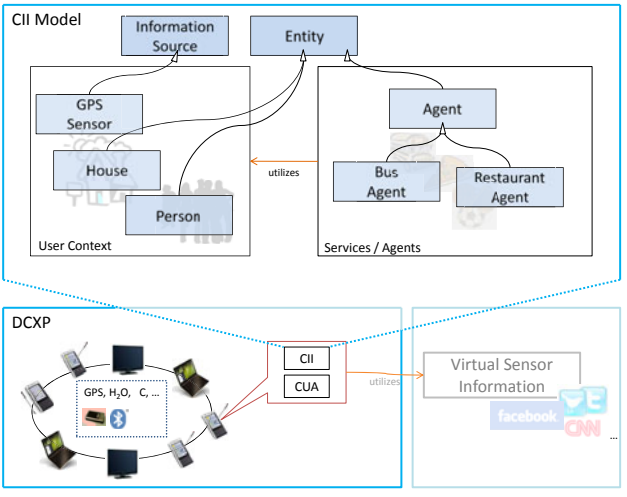

Fig. 6. The DCXP overlay and CII architecture in coordination as a means to globally share context related information that can be persisted and retrieved on demand

sources via DCXP. The framework is event-driven in that it supports publication, discovery, and (conditional) subscription to sensors. The context information may thus be collected from sensors (or sensor networks) that are wirelessly attached to mobile devices through the WSNG.

Although the framework outlined in this paper is done relative to the MediaSense framework, in this paper we define a service as a product of an application in order to serve a user. Hence, this paper treats the architecture of, and conditions for, enabling ambient intelligence, conforming to the paradigm of multi-agent systems and distributed artificial intelligence, with the possibility of utilizing the technical solutions proposed in the MediaSense framework. See Figure 6 for the principal enrollment and utilization of agents in the model. For our purposes of omnipresent availability of global sensor information, the extendable model agent environment is realized on the DCXP framework through the layering of the model on top of the architecture; with the DCXP protocol for the distribution of context information to mobile devices in an overlay network (see [23]). DCXP creates a peer to peer (P2P) overlay network of context-aware nodes, capable of exchanging context information via a conditional publisher/subscriber principle, enforcing a loose coupling. From this perspective however, the interface is architecture independent and simply requires a real-time means of collecting and disseminating the information to support the model. This functionality is realized by the Context User Agent (CUA) which is co-located with the CII platform. Within this distributed model, the CII can exist on multiple nodes, and by extension within multiple domains while having access to the common sensor information. This creates multiple sensor domains with agents located within the domains.

Consequently, the context-base itself can be seen as a service which can be utilized in any service oriented architecture. Combined with the CII-model, the context can be persisted and retrieved from any stationary or mobile node of the system. The resulting architecture can be considered a shared data space with an emphasis on context, due to the integration of the data centric context-base and DCXPs event based approach. This allows for agents and services to communicate decoupled in time since all context, including their own, is persistent. This information could be fused with virtual sensor information such as Twitter feeds.In the MediaSense framework, each node (e.g., a mobile device) in the DCXP overlay can have the functionalities for context propagation and query execution.

4 Concluding Remarks

In this paper we proposed the CII meta model and architecture for the representation of context information through the implementation of an ontology motivated object platform. Such a platform, we envision provides a concrete foundation for building dynamic agent based services and supporting time critical context dependent applications. The model, while being strict with respect to the fundamental concepts, provides for an extendable environment for realizing continually evolving object domains reflecting the inherent fluidity of observable context information. It provides for methods to translate, compare and order context values with respect to the heterogeneity of dependent applications and services.

With respect to the model as an agent environment, it exists as a first-order abstraction with two different roles: a container for agents and other entities (and their relations), and an exploitable abstraction for building multi-agent applications with context-aware agents. The context-base allows for agent persistence, including their internal states, as active first-class objects. This provides the basis for agent recovery, resilience, accessibility and mobility. Furthermore agents derive their context awareness from their ability to act upon sensor information within the model and may subsequently trigger other agents and services within the system. We provide a generic context information layer that has no explicit control of how and what agents infer from or reason upon the context.

The model's applicability was explored in the bus scenario; demonstrating the value it adds to large scale context modeling. We have presented a framework that enables for the implementation of intelligent services in systems of mobile devices. The framework is presented relative to the MediaSense framework for intelligent delivery of any information to any host, anywhere, based on context-aware information regarding personal preferences, presence information, and sensor information. In the MediaSense framework, sensor information from sensors in the vicinity of mobile users is perceived and shared using the distributed context exchange protocol (DCXP).

The model was realized with respect to the MediaSense framework creating an instance of CII in support of context aware applications and services. MediaSense

with the underlying DCXP protocol provided the means for the real-time dissemination of context information in support of CII. CII was developed in Java, utilizing the DataNucleus access platform [24]. This enabled two separate supported services to be implemented: the first, a real-time vehicular safety system exploring the ability to monitor and feedback driving and traffic conditions to commuters in real time; and secondly a presence profile service allowing for dynamic generation and activation of presence profiles in response to changes in context information, supporting seamless media transfer between users' devices. Within this implementation, the context model is centralized with respect to a CUA, however further research will involve the distribution of the CII-model across the entire domain space thus realizing a common, global, content-centric object space.

Acknowledgements

The work was financed in part by the EU Regional Fund and the County Administrative Board of Västernorrland.

References

1. Remagnino, P., Foresti, G.L.: Ambient Intelligence: A New Multidisciplinary Paradigm. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans 35(1), 1–6 (2005)
2. Floréen, P., Przybilski, M., Nurmi, P., Koolwaaij, J., Tarlano, A., Wagner, M., Luther, M., Bataille, F., Boussard, M., Mrohs, B., Lau, S.: Towards a Context Management Framework for MobiLife. IST Mobile & Wireless Communications Summit (2005)
3. Grosky, W., Kansal, A., Nath, S., Liu, J., Zhao, F.: Senseweb: An infrastructure for shared sensing. Multimedia 14, 8–13 (2007)
4. Göker, A., Watt, S., Myrhaug, H.I., Whitehead, N., Yakici, M., Bierig, R., Nuti, S.K., Cumming, H.: An Ambient, Personalised, and Context-Sensitive Information System for Mobile Users. In: Proceedings of Second European Symposium on Ambient Intelligence, pp. 19–24 (2004)
5. Strang, T., Linnhoff-Popien, C., Korbinian, F.: CoOL: A Context Ontology Language to enable Contextual Interoperability. In: Stefani, J.-B., Demeure, I., Hagimont, D. (eds.) DAIS 2003. LNCS, vol. 2893, pp. 236–247. Springer, Heidelberg (2003)
6. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. In: Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 85–90 (1994)
7. Bond, A.H., Gasser, L.: An analysis of problems and research in DAI. Readings in Distributed Artificial Intelligence, 3–35 (1988)
8. Sycara, K.P.: Multiagent Systems. AI Magazine 19(2), 79–92 (1998)
9. Dey, A.K.: Understanding and using context. In: Personal and Ubiquitous Computing, vol. 5, pp. 4–7 (2001)
10. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (1995)

11. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review 11, 93–136 (1996)
12. W3C: Resource description framework (rdf), http://www.w3.org/RDF/
13. Indulska, J., Sutton, P.: Location management in pervasive systems. In: Proceedings of the Australasian Information Security Workshop, CRPITS 2003, pp. 143–151 (2003)
14. Pollich, J., et al.: AmbieSense Reference Information Model (2004)
15. Weyns, D., Schumacher, M., Ricci, A., Viroli, M., Holvoet, T.: Environments in Multiagent Systems. The Knowledge Engineering Review 20(2), 127–141 (2005)
16. Weyns, D., Omicini, A., Odell, J.: Environment as a First-Class Abstraction in Multiagent Systems. Autonomous Agents and Multi-Agent Systems 14(1), 5–30 (2007)
17. Foundation for Intelligent Physical Agents: FIPA ACL Message Structure Specification (2002)
18. Wagner, G.: The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior. Information Systems 28(5), 475–504 (2003)
19. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent Systems Engineering. International Journal of Software Engineering and Knowledge Engineering 11(3), 231–258 (2001)
20. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a FIPA-compliant agent framework. Software - Practice and Experience 31(2), 103–128 (2001)
21. Tanenbaum, A.S., van Steen, M.: Distributed Systems, 2nd edn. Prentice Hall International, Englewood Cliffs (2006)
22. Kanter, T.G., Österberg, P., Walters, J., Kardeby, V., Forsström, S., Pettersson, S.: The MediaSense Framework. In: Proceedings of the Fourth International Conference on Digital Telecommunications, pp. 144–147 (2009)
23. Kanter, T.G., Pettersson, S., Forsström, S., Kardeby, V., Norling, R., Walters, J., Österberg, P.: Distributed Context Support for Ubiquitous Mobile Awareness Services. In: Proceedings of Fourth International ICST Conference on Communications and Networking in China (2009)
24. Dobslaw, F.: An Adaptive, Searchable and Extendable Context Model, enabling cross-domain Context Storage, Retrieval and Reasoning. Master Thesis (2009)