Context Awareness in Mobile Computing Environments: A Survey

Christos Anagnostopoulos, Athanasios Tsounis, Stathes Hadjiefthymiades
Department of Informatics and Telecommunications, University of Athens
Panepistimioupolis, Ilisia, Athens 15784, Greece
Tel.: +302107299526
bleu@di.uoa.gr, thantsoun@di.uoa.gr, shadj@di.uoa.gr

Abstract

In this survey we report software architectures for context awareness, sensor centric systems and context modeling issues. Defining architecture for supporting context-aware applications explicitly implies a scalable description of how to represent contextual information and which are the abstraction models capable of handling it. Using sensors to retrieve contextual information leads to a sensor network scheme that provides services to upper levels of applications. Operations for capturing, collating, storing and disseminating contextual information at the lowest level and aggregating it into increasingly more abstract models qualifies the context-aware systems.

Keywords: Context model, Context awareness, Contextual information, Sensors.

I. INTRODUCTION

In the recent years we have witnessed rapid advances in the enabling technologies for ubiquitous computing, such as the increasingly emerging pervasive computing standards, embedded sensor technologies, and wide range of wired and wireless protocols. In order to engineer context aware computing systems, it is of high importance to understand and define what are the constituent components of context from an engineering perspective.

We envisage a 3-dimension space into which an architectural structure for a context-aware system should be defined. The space encompasses three logical axes: Context Model, Sensor-centric Support and System Behavior. The 3-dimension space is illustrated in figure 1.

The Context Model is structured around a set of abstract entities, each describing a physical or conceptual object. The definition of a Context Model ranges from raw data oriented models (e.g. spatial information, network measurement, QoS information, environmental information) to abstract based models (decision making, inference engines, task performance, pattern behavior, context reasoning) related to activities of specific contextual abstract entities (e.g. person activity, communication channel).

Obviously, high-level context can not be directly acquired from sensors; it is reasoned from sensor-driven context monitored by sensor-based systems. Sensor based systems exploit the capabilities of sensors in order to capture low level contextual information. Such systems may couple the collected data with a compatible raw data oriented context model (context model-dependent sensor system), or focus on certain sensor management being insensitive to a context model.

Apparently, a context-aware system model should be aware of the notion of context. System Behavior in context-aware computing takes into account the adaptability of a system towards dynamic changes of the contextual information. Systems targeted for the provision of context-aware applications differently handle context adaptation. Such System may continuously react to the highly dynamic nature of context and behave properly at any context alteration, or remain idle and insensitive to context changes.

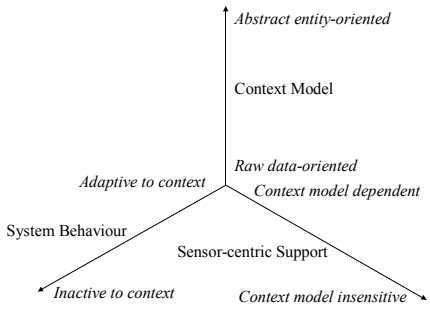

Fig.1. 3D space of Context

A. Definitions

A number of notions and definitions related to context aware computing are present in this survey. We exemplify an array of the appropriate concepts:

Context: A well known definition was given by Dey and Abowd in [41]; according to this "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 integration between a user and an application, including the user and the application themselves". We circumscribe the notion of the context as a set

of situations, which describe humans, applications, and environment related to specific activities.

Contextual entity: Any conceptual or physical object which is derived by the definition of the context including certain contextual information.

Observable: The contextual information retrieved by a deserving contextual entity, so that an application could exploit in execution time.

The contextual information is retrieved by an Observable, which composes the current view of a Contextual entity, embedded in the Context. The above notions could be envisaged in figure 2.

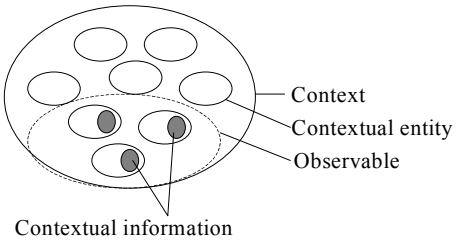

Fig.2. Contextual notions

In the remainder of the survey we first present concepts about context modeling. Such concepts are discussed from two viewpoints: the context representation and context detection. Section 3 refers to available sensor-centric systems for context sensing, producing contextual layered information. These systems are capable of handling the context model in order to meet the needs of context-aware applications. Section 4 reports on infrastructures and architectures that enable and support context services with respect to context monitoring, distributing and reasoning. Section 5 describes a reference model for context-aware system architecture. The concluding section provides an outlook on future developments and open issues.

II. CONTEXT MODELING

The context description initiates a realizable semantic form (e.g. description based on Ontology languages) with which an application should transform it to its own computational logic. The cognitive notion of context emerges from the captured events in which entities get evolved. An operational definition of the notion of context for the design and development of context-sensitive systems is drawing upon the distinction between the notion of an instant snapshot of observable entities (a situation) and the synthesis of these entities over time (a context). Observable entities and their interrelationships can be mapped, at the implementation phase, to colonies of contextors.

A contextor [2] is a software abstraction that models relationships between observables. Contextors share a common I/O structure including control channels and meta-data to ensure and express Quality of Service (QoS). According to their model they can be combined as directed graphs or encapsulated into higher computational units. Coutaz and Rey [2] expand the notion of context on the basis of the notion of situation. A situation at time t is a set of observables named state vector.

Context at time t is a composition of multiple situations over a period of time [t-Δt, t]. A situation refers to users involved in a particular task. Given a set of users U, a task T and two instances of observation at t_0 and t, the Context at time t is the following composition:

$$\text{Context}^{U,T}(t) = \text{COMPOSITION}(\text{situation}^{U,T}(t_0),\ldots,\text{situation}^{U,T}(t)),$$

where $\text{situation}^{U,T}(t)$ is the Situation at t that refers to U for performing T.

The concept of composition is a kind of history function that involves the values of new relations and system state variables as well as the destruction of old ones. Hence, a contextor should be viewed as a computational extension of context related to an Observed System Context.

A contextor returns the values of a variable that belongs to that context. In figure 3 a graphical representation of a contextor is depicted.

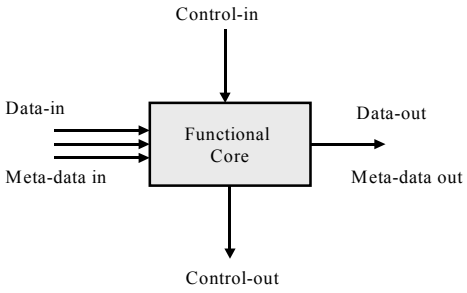

Fig.3. A Contextor

Architecture for context awareness should take into consideration the roles assigned to an entity and relations of this entity with others. Crowley and Reignier [5] envisage the context model as an intermixture of modules, which are defined as transformations of observations. Modules are assembled into reflexive processes, enriched with meta-information, under the direction of a supervisory controller. Federations of processes are dynamically

assembled, according to a model, which is based on ontology for context aware systems. A fundamental aspect of interpreting sensor observations, similarly to Contextors aspects [2], is to form entities by gathering observation. From the perspective of the system, these entities are associations of correlated observable variables.
Context is modeled as a composition of situations relative to a task. Within a situation, context shares the same set of roles and relations. Thus, context determines the collection of observable roles and relations. The model has the following form:

Context (U,T):
$\{Role_1, Role_2, \ldots, Role_n; Relation_1, Relation_2, \ldots, Relation_m\}$

A role may assume a certain action within a task. In [5], a crucial problem is how to provide a mechanism for dynamically composing federations of meta-controllers that observe the entities and relations relative to the user context. This approach is based on a rule based system written in JESS [42] (CLIPS in Java). Meta-supervisors are designed for specific contexts and maintain a model for the current context of user. This model includes information about semantically related contexts that may be attained from the current context, as well as the user and system context events may signal such a change. Figure 4 illustrates the transformation of data and events undertaken by an observational process under the direction of a reflexive controller. The action in which an observational process transforms data and events under the direction of a reflexive controller.

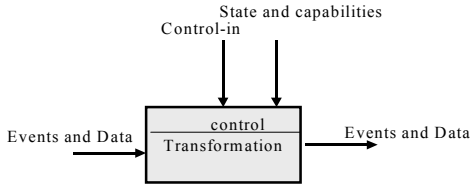

Fig.4. Transformation of Data and Event observed by a controller

Modeling contextual information in pervasive computing supports the basis of building a context management infrastructure. The motivation of context management has derived from the fact that applications will need to be sensitive to context as the latter is dynamically changing. In [11], characteristics of contextual information are specified. Contextual information exhibits certain temporal characteristics. Context can be static, which means that it describes invariant aspects (e.g. a person's date of birth), and dynamic, which refers varying contextual information (e.g. a person's location). Static context can be obtained directly from users. Frequently changing context could be obtained by indirect means, e.g. through sensors. Hence, context could be imperfect, i.e. the relevant information is inconsistent, out of date and incomplete.

Therefore, a context model must support multiple representations of the same context in different forms and at different levels of abstraction. Capturing the relationships that exist between the alternative representations should be mandatory. Context information is highly interrelated. In [11], several relationships are evident between three entities: people, their devices and their communication channels. The context model is based on an object-based approach in which context information is structured around a set of entities, such as physical (devices) or conceptual (persons, communication channels). Properties of objects are represented by attributes and the entities are linked to attributes and other entities by uni-directional relationships known as associations.

The above model classifies association, according to its nature, to two main parts: static association and dynamic association. The context captured by a static association is typically known with a high degree of confidence. The context captured by a dynamic association, through hardware or software sensors (e.g. widget [43]), is not inserted directly to the model but it is transformed in some way to bring it closer to the level of abstraction required by applications, as mentioned in [5]. In addition the model classifies the association according to its structure to two types: simple association and composite association. A simple association could be thought of a link between an object and its property. A composite association could be referred as a communication link from a Person to a set of his/her activities.

Managing such a model it is reasonable to envisage a meta-model of associations known as dependencies. A dependency is a special type of relationship that exists not between entities and attributes but between associations themselves. A dependency can be qualified by a participation constraint, which limits the pairs of associations to which a dependency applies. We could extend the management of this meta-model transforming it to a Bayesian meta-network (a two-level network in this case) [4] intended for context prediction concepts. Finally, the imperfection of the context may direct quality issues that the

model could apply. The information systems community has more extensively researched Context Quality Modeling (CQM). This model borrows ideas from Wang [36] who describes a CQM, in which attributes are tagged with various quality indicators, focused on coverage, resolution, accuracy, repeatability, frequency and staleness of context. The graph in figure 5 depicts the concept of this model including the associations and their dependencies.

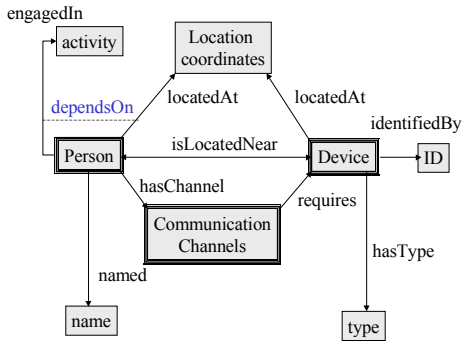

Fig.5. Context model enriched with associations and dependencies

In figure 5 the dependency dependsOn relates two associations (engagedIn and locatedAt). We can infer that an activity A, into which a person P is engaged, is actually depended upon the location of the P, thus decreasing the number of the possible activities that P could undertake in the considered location.

A different view of context, the activity-centric view [34], focuses on the context that surrounds the performance of an activity by an agent. Modeling a context based on this concept focuses on more abstract levels. Since the context being considered covers the realization of the activity by an agent, it is only meaningful when the activity is being performed. Figure 6 illustrates the view of an activity-centric context as stated from Prekop and Burnett [34].

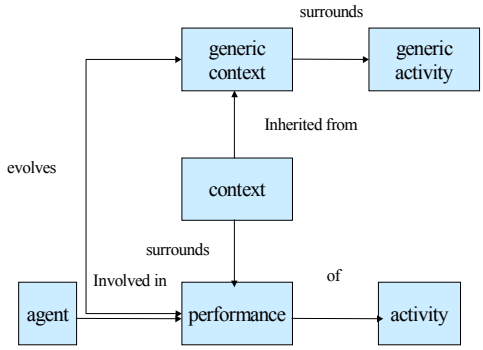

Fig.6. A view of an activity-centric context

Activities vary in scope, from the very broad to the very specific, with broad activities often containing more refined or specific activities. The performance of a broad activity, by the agent, creates a context, assuming that the nested activities have already been performed creating a nested or cascading context. The context surrounding the activity is specialized from a higher-level, more generic context, Context, which surrounds a higher-level more generic activity, Activity. The generic Contexts and Activities can be modeled, a priori, for various domains. A smart room that implements the activity-centric view of context could monitor realization of an activity by the agent and, together with machine learning techniques, evolve Context to better represent the context that surrounds the particular class of activities. The conceptual model of context, based around the realization of an activity by an agent, differs from many of the previous approaches, because it focuses on creating context-aware applications that support cognitive activities rather than context-aware applications that focus on time, location or other elements. The Context Manager, which monitors the agent's activities and makes suggestions to the agent, controls the interactions between the user (agent) and the environment (activities). An example representation of context including an agent performing a set of tasks is shown in figure 7.

Parent Context:
    Organising a Workshop
Agents Involved:
    Self
Resources:
    Calendar,email,names,Travel,Rooms
Process:
    Begin{
    Initial Agenda
    If not Approval Agenda Begin
    Contact Participant
    Book Rooms

```
        Book Travel
}End
```

Fig.7. Context representation based on the tasks performed by the agent

III. SENSOR BASED ARCHITECTURES

The contextual information is captured by a sensor-based system, following an appropriate context modeling. Information retrieval is accomplished through sensor networks. Sensor networks may take into account the nature of information using integrated methods being incorporated in the network. The sensors could be used as a simple retrieval engine implementing raw data sensing techniques, such as location observation of the sensor bearer, but also could be used as a more sophisticated mechanism for determining user situation e.g. attendance of a meeting. Sensor interaction with mobile devices introduces many challenges to mobile computing. Except from discovering services, searching for resources and modeling context in a sensor-aware system a crucial challenge is to detect the user's context (e.g., not a-priori known).

Sensor based context detection for mobile users explicitly empowers human computer interaction. The work described in [18] focuses on a communication scheme for retrieving context through autonomous sensors with no central point of control. These sensors, named Smart-Its [19], know about their own sensing capabilities and can report them to their neighbors, if inquired. The concept of introducing an interchanging format about sensor-features among Smart-Its is based on the Smart Context-Aware Packets (sCAPs), a document-based approach for collecting sensor-features sharing some similarities with context-aware packets (CAP) [29]. The sCAP gets filled with sensed information on its way through the environment. Each Smart-It receiving a sCAP contributes to the required sensor features and forwards it to another Smart-It in its neighborhood. Combining the gained features stored in the sCAP allows each Smart-It to make assumption about the current context. Based on this knowledge it can forward this sCAP to an appropriate sensor for further context investigation.

A sCAP document is organized into three parts: retrieving plan, probable context and retrieving path. The retrieving plan integrates the execution plan determining which sensors should be involved into the context detection. It describes which types of sensors have to be queried for retrieving the current context. The probable context is simple represented by a list of features already retrieved. In this point, an ontology language should be able to provide a more expressive scheme for modeling the probable context, rendering it available to an inference engine. The retrieving path comprises a time ordered list of visited sensors aiming to prevent loops (i.e. revisit sensors twice) and support knowledge about the network topology. The retrieving plan conforms to an execution plan and rewriting rules. The interaction schemes for the application of sCAPs are Pull and Push. Whereas context pull is used when the user explicitly needs the current contextual information, the push scheme is more appropriate for emerging events and interrupts, such as detecting rapid changes in the environment. Figure 8 depicts the interaction of a mobile device using the context detection concept of sCAP.

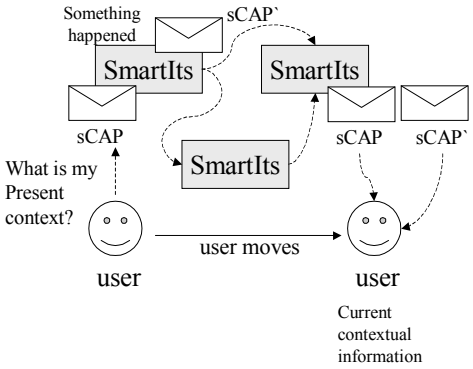

Fig. 8: Detection of contextual information using sCAPs

The user might receive several sCAPs reporting the same context. This should focus on future merging strategies of sCAPs applying an online reconfiguration of sensors or paralleling the context detection process.

After sensing raw data from low level sensors, intelligent environment architecture for multi - granularity context description is required in order to model more complex contextual information.

The sensor devices range in complexity, as asserted in [3], from devices as simple as a binary on-off reporting module, to sensors that can decide which users are engaged in a certain event in the room. Their communication capabilities may range from simple on-off binary signals to supporting Web Services. Sensors are assumed to directly connect to the environment producing raw data at irregular intervals. Merino [3] architecture classifies the sensor's capabilities and focuses on an intelligent environment comprising layers of sensors. Organizing the whole system into layers of complexity is also necessary, given the scale of the number and variety of sensors involved. Sensors in higher layers are capable

of processing raw data in order to support a more abstract context model defined by various context-aware applications.

Merino is a proposed architecture for context layers for the sensed Intelligent Environment (IE). Figure 9 models the Merino architecture.

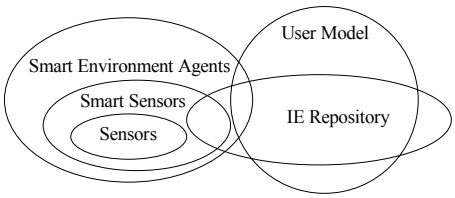

Fig.9. Merino Architecture

The architecture consists of five elements: sensors, smart sensors, smart environment agents, repository, and user model. In the lowest level, sensors are mechanisms in both hardware and software to interrogate both the physical and computational environment. Smart sensors, forming the first layer of the context abstraction, are responsible for aggregating and filtering the raw sensor data into structures that are available in the repository interface. The repository interface is a space where smart environment agents can exploit the processed data by smart sensors providing a rich context environment. Smart environment agents constitute the second layer of context abstraction. These agents may be instantiated by two classes: The class of rich context agent class, and the class of performance enhancers. Rich agents access the contextual information from the repository to form higher-level context information, the so called "rich-context". For example, a rich agent may access location and calendar context information to produce contextual information such as a meeting is on the agenda for now. New contextual information updates the repository, which may be thought as a knowledge base. Performance enhancers include learning and reasoning algorithms to discover patterns, which involve agents to monitor the performance and scalability of the underlying smart sensor layer. The smart agents produce rich context information that is provided on varying levels of granularity. The user model is managed by a Smart Personal Assistant accessing the repository for customizing and configuring the user needs updating regularly the rich context. User models can be stored in objects in the same distributed context database also being accessed by the second level of the architecture.

Sensors of every type are capable of providing services. Context aware services should be published or discovered in a ubiquitous manner. An important feature of context-aware systems is that the users should be able to find out what the system is doing on their behalf supporting those services.

The context-aware service provision is a challenge for adaptive interaction of pervasive services. CAPEUS (Context-Aware Packets Enabling Ubiquitous Services)[29] is a system architecture that independently discovers, selects and executes services with regard to the current context of the user. The adaptation of service interaction depends on the following: context constraints being embedded in service calls and actual context information. Context constraints are utilized to enable service selection and execution processes and also act, as mediators for expressing service needs to the environment. The service needs result from the user's current context or task. The system adopts a uniform document format, called Context-Aware Packets, to express service needs and constraints on a high abstraction level.

The CAP is created by the user and placed in the network in which it gets evaluated. Service needs are expressed by context constraints, which describe the situation and circumstances under which the user intends to use a service. The CAP document is organized into three parts: Context Constraints, Scripting and Data. Context Constraints are used to mediate user's service needs. A constraint is a set of three entities: the Abstract entity, the Relation and the Event. The first relates to the service peer, sensor or person. Relation entity describes dependencies of entities constraining the selection of a desired service. Events, represented by logical conditions, report situations detected by sensors forming a trigger. The scripting part represents simple scripts to be executed during service invocation in order to provide more complex semantics. The data section provides data to be processed by the selected service. A user injects a CAP to a local SAN (Service Access Node). The SAN evaluates the CAP in two phases: selection and execution. The SAN checks whether the CAP is related to a service in its domain. If not, it routes the CAP to the appropriate SAN to meet the service needs. In the second phase, the service is executed according to the contextual constraints, being embedded in the CAP, and the current context of the user. The targeted SAN reads out sensor signals to find out the user's current context in service execution time. An example of service invocation relative to context-constraints is depicted in figure 10.

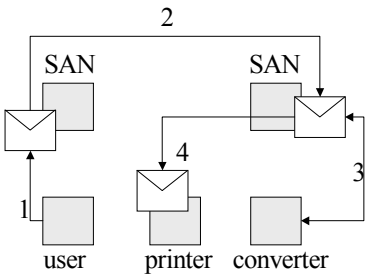

Fig.10. An example using the CAP

A user wants to print a PDF file but the nearest printer can not support this type of file. A CAP is transparently injected in the network (1) and the SAN routes this CAP to the appropriate SAN (2) which can convert the PDF file to the type of file that the printer supports. Finally the converted data in the CAP is printed (4).

IV. CONTEXT-AWARE ARCHITECTURES

The common philosophy of context-aware system architectures [24-28, 30-33, 17] is the hierarchical structure. This abstract structure covers two levels: the operational and the informational. From the operational point of view the system modules that are distributed in a mobile computing environment may serve as:

- Sensors which capture raw data
- Autonomous mediators that process and filter raw data streams exporting them into higher data-representative layers
- Smart agents which can communicate with each other in order to mine knowledge that resides in the system
- Context aware applications that provide innovative services to end users catering for context adaptation and maintenance of an integrity scheme for the user context.

With respect to the informational level, knowledge representation may focus on:

- A simple data model without an expressive scheme
- Modeling large amount of data using classic relational and object-oriented paradigms
- Serving as an ontology that describes distributed system resources such as user profiles, device capabilities and, even, context of applications.
- The operational level coupled with its informational counterpart comprises a universal context aware system model.

Figure 11 depicts the discussed combination of the above models.

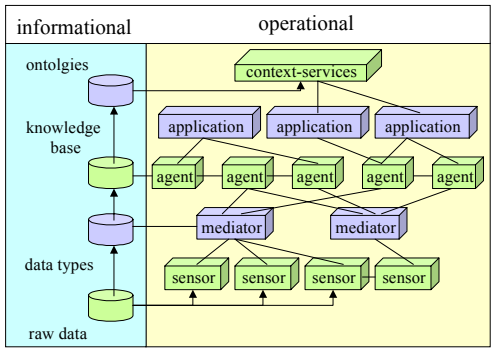

Fig.11. Operational and informational system model

In the following paragraphs we refer to several system architectures that fulfill the requirements of context modeling and implement some of the features of mobile computing environments (e.g. context management, context representation, context security).

A. Operational Level viewpoint

Building context-aware systems involves facing several design challenges to cope with highly dynamic environments and constantly changing user requirements. Such challenges are mainly related to gathering, storage, modeling, distribution and monitoring of context. A proper architectural support is needed to address these challenges. Web Architectures for Services Platforms (WASP) [15] is a project dealing with the definition of a service platform which supports the development and the deployment of context-aware integrated mobile speech and data applications, based on Web Services technology on top of 3G mobile networks.

The WASP platform provides services to Context Providers, which communicate through the Context Interpreter module. Context Interpreter tends to gather contextual information, conforming to a specific context-model, making it available to the rest of the platform. Moreover, the platform consists of a set of Repositories, which support the Monitor component with knowledge of the elements involved in WASP. The Repositories collect information from the Context Interpreter (e.g. user profile, entity types) and use services of the Service Providers. The Monitor component is responsible for integrating with the WASP applications, managing their subscription (using WASP Subscription Language - WSL) and gathering information from Repositories and Context Interpreters. Applications use WSL during their subscription, configuring the platform to react to a given correlation of events, potentially involving contextual

information.

The specific context-model is relevant to data entities. Data entity represents objects of the real world (e.g. person, activity, device, location). Attributes and associations are combined with data entities. Furthermore, ontologies are believed to be a key requirement for modeling software system architecture in order to achieve a more expressive scheme of contextual information. Ontologies allow architectural components to share knowledge and reason about information consistency. Different data entities must share common contextual representation, allowing the derivation of complex context. WASP exploits the Semantic Web Technology, building contextual information using ontologies from an ontology-based markup language, DAML+OIL, OWL, [37]. By exploiting semantic knowledge, the platform could be invoked by different conceptual layers, from context storage to adaptive interfaces and from service description- discovery to complex service composition. In figure 12 the WASP platform is illustrated.

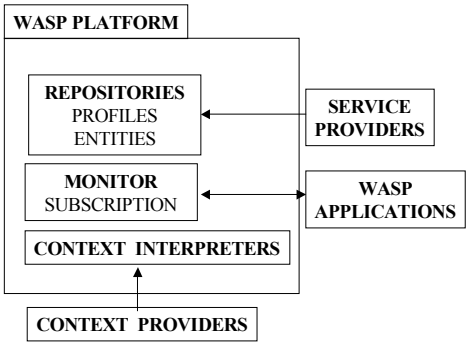

Fig.12. WASP architecture

Among the critical research issues in developing context-aware systems are context modeling, context reasoning, and knowledge sharing and user privacy protection. The work described in [16] addresses these issues by developing an agent-oriented architecture**,** the Context Broker Architecture (CoBrA). CoBrA aims to assist devices, services and agents to become context aware in smart spaces (e.g. an intelligent meeting room). Such an infrastructure requires the following: a collection of ontologies for modeling context, a shared model of the current context, and a declarative policy language, that users and devices may use to define constraints on the sharing of private information and protection of resources. CoBrA uses the OWL [37] to define contextual ontologies, providing a more complex and machine comprehensible representation of contextual information for reasoning and knowledge sharing. CoBrA provides a resource-rich agent, named context broker, to manage and maintain the shared model of context. The architecture defines that a context broker is associated with a certain smart spaces environment. Such context broker is an aggregation of other brokers representing smaller parts of the original smart space environment. This hierarchical approach, with the support of shared ontologies, helps to avoid the bottlenecks associated with a single centralized broker. Context broker can also infer contextual knowledge that cannot be easily acquired from the physical sensors and can detect and resolve inconsistent knowledge that often occurs as a result of imperfect sensing. Additionally, CoBrA provides a policy language that allows users to control their contextual information. A context broker acquires contextual information from heterogeneous sources and fuses such information into a coherent model that is then shared among computing entities in the environment.

Another infrastructure that enables scalable and flexible sensor-based services is that of IrisNet [13].IrisNet employs the aforementioned scheme (hierarchical system model) by forming a two-tier hierarchy of sensing nodes and information brokering (queries) nodes. It substantially reduces network bandwidth requirements through the use of senselets. Senslets are binary code fragments that perform intensive data filtering at the sensing nodes. IrisNet includes advanced sensor devices (called brilliant rocks) which form a wide area sensor network for intelligent context processes. Such processes conform to distributed filtering, hierarchical caching, query routing and context freshness specifications. The two tiers of IrisNet, the Sensing Agents (SA) and the Organizing Agents (OA) can fulfill these specifications. The former collect and filter sensor readings according to a data model and the later perform query-processing tasks on the sensor readings. IrisNet OAs provide a simple way for a service to incorporate support for complex queries. The system enables senselets to be uploaded from OAs to any SA to instruct and perform tasks (e.g. collecting the required information, filtering, caching) and transmit the distilled information to the OA. The OA and SA execution environment uses a service-specific processing and filtering of the sensor feeds which eliminates duplicated and redundant information in order to maintain a useful context-sensitive environment. Figure 13 represents an IrisNet instance.

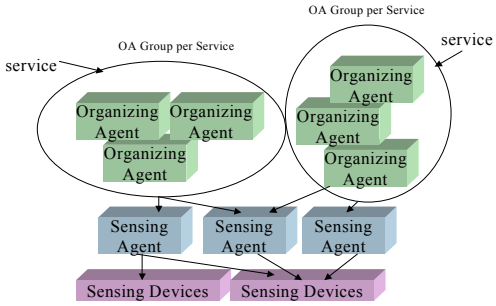

Fig.13. IrisNet instance

The agent-based environment for context-aware mobile services, My Campus [22], revolves around a growing collection of customizable agents capable of automatically discovering and accessing Internet and Intranet services as they assist their users in carrying out different tasks. The power and scalability of this environment directly are attributed to a set of ontologies for describing contextual attributes (e.g. user preferences) facilitating the easy inclusion of new, task-specific, agents. Agents focus on context-sensitive message filtering, message routing and context-sensitive reminding. More sophisticated agents incorporate planning and automated Web Service access functionality.

As already mentioned, a large number of system architectures are based on an agent-oriented scheme achieving a hierarchical structure in context monitoring. Agents form a platform that enables interaction among contextual information, users and resources through an event generator. Therefore in the majority of architectures, the whole system consists of a multitude of active spaces that provide ubiquitous access to system resources, according to the current context of the user. Below, we present an active spaces system.

UbiqtOS [40] supports dynamic, application-specific and context-aware adaptation by forming a simple, event-based architecture to allow interoperability of low-end devices. UbiqtOS supports heterogeneity and mobility in mobile computing environments. The system consists of the following components:

- An extensible agent engine (SEMAS), which allows context-specific software to be relocated in a device and execute its task.
- An extensible registry (UbiqDir) which allows components to be dynamically discovered in the system using "smart" lookups. It captures and exports changes of context to components residing within devices, thus achieving context aware adaptation.
- A synchronous-event routing system (Romvets) providing a dynamic extension of components installed in UbiqtOS, and,
- finally, a dispatcher module which exports context-specific views of distributed resources in an active space.

One of the most important elements included in the mobile computing context is location. Location may be represented as a combined graph of entities related with spatial information.

Many architectures of ubiquitous computing process the location information through suitable spatial models. The RAUM [8] architecture develops context modeling focusing on the location. The system only supports context generated by an appropriate spatial model. Such model is based on the relative location of entities rather than on their identity. RAUM is a spatial-aware communication model in which, two entities are considered contextually interrelated of their nearby locations rather than of their network identifiers. This model consists of two main parts: the Location Representation Model (LRM) and the Communication Model (CM). The LRM defines how location is represented, stored, and communicated in the RAUM-model and the CM defines how location information is used in the communication among the RAUM entities. Figure 14 depicts the LRM tree.

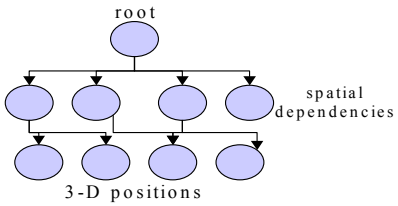

Fig.14. LRM tree representation

In the RAUM - LRM a tree presentation for location selection is adopted. This location-tree consists of three general layers: A tree-root, the semantic sub-layers and a position stated in three-dimensional Cartesian coordinates. Further specialization of the third layer, into sub-sections, enables a finer grained differentiation of locations. The RAUM system makes use of distributed storage of the location information as far as all the involved objects are capable of handling such information. Thus, no central entity for storing and providing the complete location-tree is required. All the objects (e.g. mobile devices) in the system only hold the part of the tree that is pertinent to them. Most objects only have to store the path through the tree representing

their own location. This implies that RAUM is used whenever peer-to-peer communication is required. The systems which were developed using the RAUM model are MediaCup [9], MemoClip [10] and the SmartIts [19]. All these systems contain micro-controller based computing devices.

Context presentation plays a significant role to the user, assuming that the presentation is a combined logic of user, context and terminal features. The rule-based architecture, described in [23], approaches the context adaptation forming three predicates: The fact, the system assumptions and the discourse context. Significant facts can be collected directly from the user (e.g. through a very short questionnaire). Knowing these attributes in advance allows adaptation by setting contents or imposing constraints on presentation level. The system obtains relevant facts also by sensors, i.e. by tracing the user's movement. Besides facts, system assumptions provide relevant context factors, such as Design choices and Dynamic user modeling. The former specify the system's behavior in relation to certain contextual conditions and the latter is based on the correct interpretation of the user's action, interest and knowledge. The discourse context contains both communicative context and textual context. Architecture for context adaptation is based on a component based system. Two modules, the Visitor's Models and the Interaction History store the facts, that the system knows about each user and record both the places visited and the information received. The Physical Organization Knowledge base contains a description of the space in which a user exists or acts. The information stored in these modules makes up the context. Two rule-based modules that realize the adaptive behavior, the Input analyzer and the Compose Engine exploit the context. The Input analyzer interprets the user's implicit and explicit input and decides on whether a new presentation has to be delivered. The latter decides on context presentation. The respective rules are organized in clusters. Each cluster is applied to accomplish a certain task. This rule-based system supports a prototyping approach: changing the rules, not the system code, can alter system behavior. The implementation of this architecture is done by two location-adaptive systems: Hyper-Audio [38] and HIPS [39].

B. Informational Level viewpoint

From the informational development level we meet systems capable of modeling and handling context via integrating context-repositories. The Context Fabric [1] is an infrastructure for context-aware computing. It provides a flexible and distributed context data store, a context specification language for declaratively stating and processing context needs and a customizable privacy mechanism to protect context data for end-users. The context data store consists of two modules; the local context data model and the physical data store. In the logical context data model, the context information is represented using four concepts: entities, attributes, relationships and aggregates. Relationships are special purpose attributes used to express actions among entities. The aggregates group existing entities to form a more sophisticated and complicated representation of context. The second module, the physical data store, is actually responsible for the physical storage of the contextual information. The physical data store distributes and duplicates contextual information pertaining to a user in a more efficient way. It places the context data close to their origin and where it is likely to be used. The responsibility administrating, maintaining and protecting contextual information is therefore an open issue.

The context specification language (CSL) is a declarative way of stating context needs at a high level, providing a clean programming abstraction to the contextual information, in the same way as SQL does for relational databases. CSL statements are locally processed by Context Services which can handle queries such as "Notify me every time a person enters a room". A protection mechanism for maintaining integrity and privacy of context is focused on a restricted-based CSL, which allows context queries to return intentionally ambiguous answers.

V. REFERENCE MODEL

We envisage a reference model for context-aware system by introducing specific functionalities, which facilitate the context-oriented design. Some functionalities, related to context-aware system modeling, could fullfil the specifications of the aforementioned architectures. Such functionalities are listed below:

- Sensing information processes, which are operational modules that observe the status of the sensors and process information received by sensor networks. Such processes include sensing techniques for information retrieval and low-level information representation.
- Context based initiatives, which manage and exploit contextual information. Such initiatives comprise context acquisition, context aggregation, context consistency, context

discovery, context query, context reasoning, context quality issues, and context modeling.

Any initiative, which is considered contextually interrelated with each other, is appropriately placed in an abstract contextual use case diagram forming a reference model for a context-aware system. Such initiatives are defined bellow:

Context acquisition: A mechanism to obtain context data from diverse context sources. Context acquisition could be colligated with hardware sensors, which context data are conformed by a low-level data model

Context aggregation: A mechanism that can provide a persistent storage for distributed context and guarantees integrity of context. In case of a shared context model, the context aggregation sustains a basis to merge correlated contextual information.

Context consistency: Context consistency enables the coherence of dynamically changing distributed context models. Such mechanism, regarded as extended context aggregation functionality, sustains the structure (e.g. relationships among conceptual entities) of the contextual model in higher levels of abstraction.

Context discovery: The aim of context discovery is to locate and access contextual sources in terms of serving context requests (e.g. seeking the appropriate contextual information pertinent to an entity). Context discovery includes issues about service description, advertisement and event subscription.

Context query: By exploring contextual information, residing in distributed context repositories, a reference model needs high-level mechanism for posing queries without explicitly handling underlying data manipulation. Complex context retrieval tasks (e.g. queries as "list all persons in the same conference hall whose presentation is at the same time with mine") must be transparent to end-users. Context query mechanism should pose design issues as context query language, query optimizations, trigger messages and definitions of constraints related to context acquisition.

Context reasoning: Context can be elaborated with reasoning mechanisms. Context reasoning is a process which based on a-priori known context inferences new context, previously unidentified. Reasoning tasks is oriented to check context consistency and deduce high-level context. Such tasks could be implemented using logical schemes as first-order predicates and description logic.

Context quality indicators: Context data can come from heterogeneity context sources such as sensors and software services. The lack of a universal context model and the application-specific representation of the contextual data distort the consistency of the sensed information. A mechanism for pertaining predefined sets of quality indicators related to contextual information is of high importance. Such issues may be resolution, accuracy, repeatability, frequency and staleness of context.

Context modeling: Context model outlines an expressive scheme for context representation and context interpretation. Existing context models vary in the expressiveness they support and in the abstraction level of the presentation of the conceptual objects. Context model is based on capturing general features of context entities as properties (e.g. location, temperature) and interrelations between contextual objects (e.g. spatial relations). Complex context models may exploit the notion of a context meta-model with which any dependencies, aggregations and classifications of conceptual objects could be described (e.g. UML meta-model can define context constraints between user elements related to a task). A uniformed context model could facilitate context sharing and semantic interoperability.

By combining such functionalities, we illustrate the reference model as a use case diagram which incorporates the sensing techniques and the context initiatives. Figure 15 depicts the reference model.

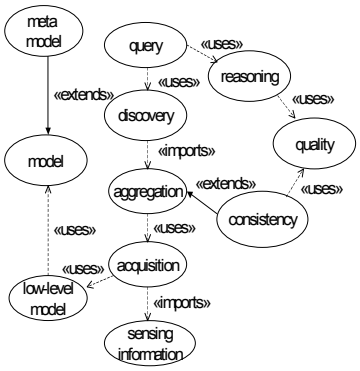

Fig.15. Reference model Use Case Diagram

According to this reference model, we logically place the aforementioned architectures on the table I. The functionalities, related to context awareness regarding to architectures, are illustrated in this table.

TABLE I
FUNCTIONALITIES FOR CONTEXT AWARENESS

		system/model	functionalities
Context Modeling		Contextor	meta model
		Meta-controllers	
		Assosiation Model [11]	model
		Activity-centric	low-level model
Sensor Based Architectures		sCAP	sensing data acquisition aggregation
		CAPEUS	
		Merino	
Context-Aware Architectures	operational	WASP	discovery acquisition quality reasoning
		CoBrA	
		IrisNet	
		My Campus	
		UbiqtOS	
		RAUM	
		HIPS,Hyper-Audio	
	informational	Context Fabric	consistency acquisition query
		CSL	

VI. OPEN ISSUES

A long-term goal of system architectures is making sensors and context platforms flexible and scalable enough to be widely adopted in various context-aware applications. Aiming at Human-Computer Interaction, context-sensing requirements in context-aware computing applications take into account the fact that sensors are highly distributed and their configuration is highly dynamic. Based also on the assumption, that the more complex context model can be decomposed into simpler discrete facts and events, many context models propose a top-down systematic approach providing a clear path letting computers understand context in human-like ways. Issues about privacy and distribution of context information, conforming to an appropriate distribution model of partitioning and replication context, are open while autonomous configuration schemes for sensors, providing service adaptation, are equally crucial. Designing contextual data format and network protocols to allow interoperability by supporting different types of sensors and finding the right balance of developing a universal context model and smart infrastructures are challenges for future context - services. Many systems should also take into consideration the quality of context represented into a model supported by a meta-model scheme of context. Finally, an open issue is definition of a context prediction method supporting the proactivity of context-services in a proactive environment which associates «similar» context models.

REFERENCES

[1] Jason I.Hong, "*The Context Fabric: An Infrastructure for Context - Aware Computing*", CHI 2002, April 20 -25, 2002, Minneapolis, Minnesota, USA. ACM 1-58113-454-1/02/0004

[2] Joëlle Coutaz, Gaëtan Rey, "*Foundations for a theory of contextors*", CLIPS-IMAG,BP 53 France

[3] Bob Kummerfeld, Aaron Quigley, Chris Johnson, Rene Hexel, "*Merino:Towards an intelligent environment architecture for multi-granularity context description*", User Modeling for Ubiquitous Computing, 2003

[4] Vagan Terziyan, Oleksandra Vitko,"*Bayesian Metanetworks for Modeling User Preferences in Mobile Environment*", University of Jyvaskyla, Finland

[5] James L.Crowley, Patric Reigner,"*An architecture for Context Aware Observation of Human Activity*", Project PRIMA, INRIA Rhone Alpes, France.

[6] James L.Crowley, Patric Reigner, J. Coutaz, G. Rey, "*Perceptual Components for Context Aware Computing*", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002

[7] Huandong Wu,Mel Siegel, Sevim Albay,"*Sensor Fusion for Context Understanding*", Robotics Institute, Carnegie Mellon University,IEEE Instrumentation and Measurment Technology Conference Anchorage, AK, USA, 21-23 May 2002

[8] Michael Beigl, Tobias Zimmer, Christian Decker,"*A Location Model for Communication and Processing of Context*", TecO, University of Karlsruhe, Karlsruhe, Germany

[9] Beigl M, Gellersen HW, Schmidt A., "*MediaCups: Experience with design and use of computer-augmented everyday objects*", Computer Networks 2001; 35(4):401-409

[10]Beigl M. "*Memoclip:A location based rememberance appliance*",Proceedings 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2000), Bristol, UK 2000:230-234

[11]Karen Henricksen, Jadwiga Indulska, Andry Rakotonirainy, "*Modeling Context Information in Pervasive Compiting Systems*", The University of Queensland, Australia,2002

[12]Anthony Jameson,"*Modeling Both the Context and the User*", Department of Computer Science, University of Saarbrücken, Germany.

[13] Suman Nath, Yan Ke, Phillip B.Gibbons,Brad Karp, Srinivasan Seshan,"*IrisNet:An Architecture for Enabling Sensor-Enriched Internet Service*",Intel Reaserch Pittsburgh, Carnegie Mellon University, IRP-TR-02-10,December 2002,

[14] Diego Rios,Patricia Dockhorn Costa, Giancardo Guizzardi, Luis Ferreira Pires, Jose Concalves, Pereira Filho, Marten von Sinderen,"*Using ontologies for Modleing context-aware services platforms*", University of Twente, The Netherlands.

[15] WASP Project, [http://www.freeband.nl/ projecten/wasp]

[16] Harry Chen, Tim Finin, Anupam Joshi,"*An Intelligent Broker for Context-Aware Systems*",University of Maryland,Baltimore
[17] Richard W.De Vaul, Alex Sandy Pentland,"*The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications*", The Media Laboratory, Massachusetts.
[18]Florian Michachelles, Michael Samulowitz, "*Smart CAPS for Smart Its – Context Detection for Mobile Users*".
[19]SmartIts, [http://www.smart-its.org]
[20] Guanling Chen, David Kotz, "*A survey of Context Aware Mobile Computing Research*", Dartmouth Computer Science Technical Report TR2000-381.
[21] K. El-Khatib, G.v.Bochmann,"*Agent Support for Context – Aware Services and Personal Mobility*",University of Ottawa, Canada
[22] Norman M.Saden, Enoch Chan, Linh Van,"*MyCampus: An Agent-Based Environment for Context-Aware Mobile Services*", Carnegie Mellon University
[23] Daniela Petrelli, Elena Not, Massimo Zancanaro, Carlo Strapparava, Oliviero Stock,"*Modelling and Adapting to Context*", Cognitive and Communication Technology Division. ITC-irst,Trento, Italy.
[24] Akio Sashima,Koichi Kurumantani, "*Seamless Context-Aware Information Assists Based on Multiagent Cooperation*",Cyber Assist Research Center(CARC),Japan
[25] Christos Efstratiou, Keith Cheverst, Nigel Davies, Adrian Friday, "*An Architecture for the Effective Support of Adaptive Context Aware Applications*", Distributed Multimedia Research Group, Lancaster University.
[26]Shiow-yang Wu, H.S. Cinatit Chao, "*Event Engine for Adaptive Mobile Computing*", National Dong Hwa University, Taiwan, R.O.C.
[27] Bill n. Schilit, Norman Adams,Roy Want, "*Contewxt Aware Computing Applications*".
[28] Olga Ratsimor, Sethuram, Balaji Kodeswaram, Anupam Joshi, Tim Finin, Yelena Yesha, "*Numi: Collaborative Mobile Data Management in Infostation Networks*", Baltimore.
[29] Michael Samulowitz, Florian Michahelles, Claudia Linnhoff-Popien, "*Adaptive Interaction for Enabling Pervasive Services*", University of Munich,2001
[30] Tatsuo Nakajima, "*Pervasive Servers: A framework for creating a society of appliance*", Pre Ubiquit Comput (2003) 7: 182-188,DOI 10.1007/s00779-003-0222-2
[31] Sandeep Adwankar, Venu Vasudevan, "*Mobile Agent based Pervasive System Manager for Enterprise Network*", Mobile Platforms and Services Lab, Motorola Labs.
[32] Ting Liu, Margaret Martonosi, "*Impala: A Middleware System for Managing Automatic, Parallel Sensor Systems*", Princeton University,2003
[33] B. De Carolis, S. Pizzuto, I. Palmisano, A. Cavaluzzi, "*A Personal Agent Supporting Ubiquitous Interaction*", Intelligent Interfaces, Department of Informatics, University of Bari, Italy.
[34] Paul Preko, Mark Burnett, "*Activities, context and ubiquitous computing*", 2002 Elsevier Science PII: S0140-3664(02)00251-7
[35] Philip Gray, Danile Salber, "*Modeling and Using Sensed Context Information in the Design of Interactive Applications*", University of Glasgow, Scotland, Springer-Verlang Berlin Heindelber 2001-LNCS 2254,pp. 317-335,2001.
[36] Wang, R., Reddy, M.P., Kon, H. "*Towards quality data : An Attribute-based approach.*" Decision Support System 13 (1995) 349-372.
[37] Smith, Welty, McGuinness, "*Owl web ontolog language guide*", [http://www.w3c.org/TR/owl-guide/],2003
[38] Petrelli D, Not E, Sarini M,Strapparava C,Zancanaro M, "*HyperAudio = location awareness + adaptivity*", In: Extended Abstract CHI'99. Pittsburgh, May, 1999;21-22.
[39] Benelli G, Bianchi A, Marti P, Not E, Sennati D, "*HIPS: hyper-interaction within the physical space*". In : IEEE Multimedia System '99. Firenze, 1999; 2:1075-1078.
[40] "*Architectures for Ubiquitous Systems*", Technical report, Number 527, UCAM-CL-TR-527, ISSN 1476-2986, United Kingdom,2002
[41] G.G. Abowd and A.D. Key, "*Towards a better understanding of context and context-awareness*" Technical Report
[42] JESS, The rule engine for the Java Platform, [http://herzberg.ca.sandia.gov/jess/f/]
[43] IOS Widgets, [http://mdp.artcenter.edu/ ~vanallen/ios1/2004sp/ios1_wk02d.html]