

Context Awareness in Mobile Computing Environments

CHRISTOS B. ANAGNOSTOPOULOS, ATHANASIOS TSOUNIS and STATHES HADJIEFTHYMIADES

Pervasive Computing Research Group, Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, Athens 15784, Greece
E-mails: *bleu@di.uoa.gr; shadj@di.uoa.gr*

Abstract. In this article, we report software architectures for context awareness in mobile computing environments, sensor centric systems and discuss context modeling issues. Defining an architecture for supporting context-aware applications for mobile devices explicitly implies a scalable description of how to represent contextual information and which are the abstraction models capable of handling such information. Using sensors to retrieve contextual information (e.g., user location) leads to a sensor network scheme that provides services to the applications level. Operations for capturing, collating, storing, and disseminating contextual information at the lowest level and aggregating it into increasingly more abstract models qualify the context-aware systems. In this article, we introduce context aware systems in mobile computing environments, review the basic mechanisms underlying the operation of such systems, and discuss notable work and important architectures in the area.

Keywords: context definition, context modeling, context-aware applications

1. Introduction

In the recent years, we have witnessed rapid advances in the enabling technologies for mobile and ubiquitous computing, such as the increasing pervasive computing paradigm, embedded sensor technologies and wide range of wired and wireless protocols. Specifically, context-aware computing is emerging as the next computing paradigm in which infrastructure and services are seamlessly available anywhere, anytime, and in any format. In order to engineer context-aware computing systems, it is of high importance to understand, apprehend, and define the constituent components of context from an engineering perspective, as well as, from a model-theoretic perspective.

Moreover, mobile and context-aware computing applications respond to changes in the environment in an intelligent manner in order to enhance the computer experience of the user. Specifically, context-aware applications tend to be enhanced mobile applications for the following reasons: (i) user context changes frequently subject to the user's mobility behavior, and (ii) the need for context-aware behavior is greater in a mobile environment (context-aware applications have to take into account user location, network resources, and device capabilities).

One could consider a space into which the structure for a Context-Aware System (CAS) could be defined. Such space, which is depicted in Figure 1, encompasses three logical axes, Context Modeling, Pervasiveness, and, System Behavior. Context Modeling is, mainly, a methodology that is based on a structural logical set of definite or abstract environmental

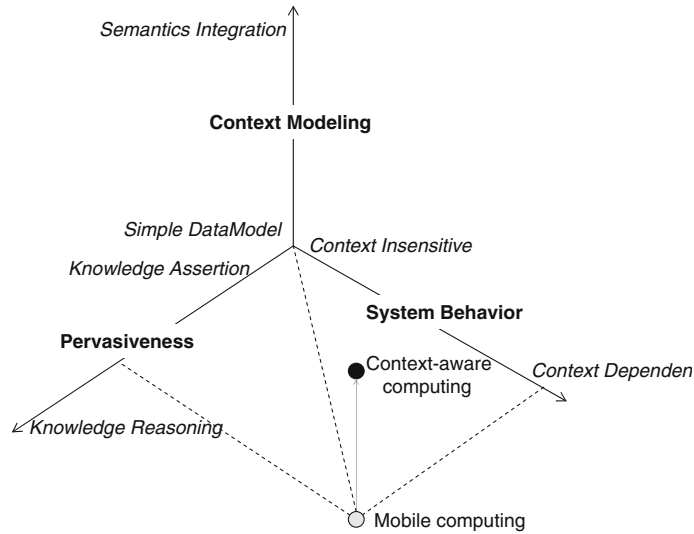


Figure 1. Context-aware logic space.

entities describing a physical or conceptual object, respectively. The definition of a context model may range from unstructured raw-data models (e.g., location, spatial data, network measurement, Quality of Services (QoS) indicators, and environmental state) to logic-based models (e.g., spatiotemporal models, object-oriented models, relational models, and propositional logics). The latter, logic-based models could relate to actual activities of specific entities (e.g., human activities, such as meeting, working, sleeping, and walking).

Additionally, sensor-based systems use sensors, in order to capture low-level context. Such systems couple the collected data with a compatible data model. Obviously, high-level representation of context (i.e., context modeled by logic-based data representations) cannot be directly acquired or disseminated from sensors. Specifically, in mobile computing a well known technique for capturing contextual information is called proximity selection. Proximity selection is primarily based on the user location context. Such context can be (i) resources and devices in the vicinity of the user, (ii) places of interests closest to the user current position, and (iii) computational objects with which the mobile user is currently interacting. It is, also, the reasoning process that produces, or, even, entails (i.e., assertion of *consistent* knowledge) such *new* context derived from the existing and consistent one. One could view that kind of capability (i.e., reasoning) as an aspect of pervasiveness in a context-aware system. The more a system can extend (i.e., assert knowledge) and infer (i.e., entail such knowledge) knowledge for a specific world (e.g., application domain), the more pervasive could it be considered, as it can reason and learn behaviors, or, estimate values for that context.

Obviously, CAS should be aware of the very specific context that falls in its management responsibility including, context storage, dissemination, adaptation, provision, and reasoning. System behavior, with respect to context management, means the ability of the system to adapt and, consequently, react to the expected dynamic changes of the context (e.g., from the fact that a sensor broke down, to a mobile user changing direction, to finding inconsistencies in the knowledge base). Moreover, context-aware systems target to the provision of context-aware

services that are assumed to differently handle context adaptation. Such systems may continuously react to the highly dynamic nature of context and behave accordingly at any context alteration, or, remain idle and insensitive to those changes. The various context models that the majority of context-aware applications adopt are discussed in the following section.

1.1. DEFINITION OF CONTEXT

It is common knowledge that the concept of context is studied in many research areas such as cognitive science [1, 2] and linguistics [3]. However, not only had context been introduced in the area of artificial intelligence (e.g., first-order logic systems [4]), but it had become a widely discussed issue as it had been well formalized in [5, 6]. Specifically, according to such theories about context definition and utilization, one could consider that context is used as a means of solving the problem of generality. In fact, contexts have been used in various applications in mobile devices that deal with modeling situations, beliefs, probabilities, spatial, and temporal relations, web semantics, integration of heterogeneous knowledge resources, and, databases. Moreover, the authors in [7] defined context as a means to formalize theoretical issues concerning reasoning over heterogeneous contextual information of beliefs related to such information.

We define the term *contextual information* as the retrieved and relevant data that comprise the *context*. Specifically, such *data* are retrieved from heterogeneous and comprise a certain application context. Each piece of such information focuses on a specific epistemic field tightly related to its application context. Furthermore, the authors in [7] defined the concept of context according to two principles: locality and compatibility. The former principle is based on the reasoning only on part of knowledge that is currently available, i.e., what is currently known. The latter principle denotes the compatibility relations being reasoned among diverse contexts.

Moreover, CAS focuses on the contextual information manipulated by a certain application. A well-known context definition related to the application specific context was proposed by the authors in [6]. According to this definition, “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”. From this definition, one could derive that context is a set of situations and actions. Such situations force to change over time, describing humans’ behaviors, application and environmental states, whenever specific actions are applied (e.g., such actions can be considered as the fluent predicates in situation calculus [8]).

In the remainder of the article, we firstly (Section 2) discuss previous efforts on context modeling. Such efforts are discussed from two viewpoints: context *representation* and context *detection*. Section 3, presents the requirements for a mobile computing application to be considered as context-aware. Section 4, refers to existing application frameworks for context *sensing*, producing abstract contextual information (i.e., from raw data selection to abstract data definitions, such as clustering, classification, or, even, approximate reasoning). Section 5, reports on infrastructures and architectures that enable and support context-aware mobile services with respect to context monitoring, adaptation, distribution and reasoning. Finally, in Section 6, a discussion related to applications of large scale context-aware systems is provided, whilst, conclusions along with future considerations and open issues related to context-awareness in mobile computing environments, are discussed in Section 7.

2. Context Modeling

The context modeling is a research area that tries to answer the following questions:

- which are the most appropriate contextual information that can model well enough the specific context in a certain domain (e.g., epistemic field, such as mobility behavior of a user, network congestion control, early warning, and transportation traffic notification),
- which are the relations among such pieces of information,
- how can one take into account the information change, and, how can one react to such change, if necessary.

Hence, modeling context is a technique focuses on how to find and relate contextual information that better captures the observation of certain *worlds* of interest.

In this section, context modeling approaches can be classified into two, not necessarily disjoint, taxonomies: *Context Theoretic* and *Context Conceptual Modeling*. The first class of approaches relies on context modeling with the aid of context theoretic modeling. The second class of approaches addresses the *problem* of context modeling as a *problem* of context conceptual modeling. The former methodology is based on modeling the contextual information as situations and the contextual information changes as actions, which are applicable to certain situations. The latter methodology is based on mapping contextual information into concepts (e.g., predicates or formulae in first-order logic) and on associating each compatibility relation with a conceptual role in conceptual modeling (e.g., binary or n -ary association). Currently, the most useful, in terms of reasoning complexity, mapping of context to concepts is attained by the Description Logics (DL) [9]. Such logics interprets the possible worlds of context parts as a set of interpretation models (e.g., a knowledge base with facts) in order to achieve contextual reasoning.

Further discussion will reveal that context is more than situations and actions, but, several *modals* are attached to its definition to extend its *semantics* [9]. The concept of modal is interpreted as a specific property of the world, into which context belongs. Context model has to bear in mind the appropriate selection of modals over the context. Consider, for instance, the modal *Belief*. This means that, either Alice's location is the R19 office with some degree of belief, or Alice is actually in R20 office, but she may move to R19 office, at a certain time with a certain probability. We report several modals that describe the contextual information change as follows: (i) *Activity* modal, when context modeling refers to activity-oriented context, (ii) *Belief* modal, when refers to belief revision and update, (iii) *Probability* modal, when refers to probabilistic-based context models, (iv) *Time* modal, when dealing with temporal reasoning, (v) *Fuzzy* modal, when reasoning about uncertain and vague information context models, and (vi) *Category* modal, when refers to more *abstract* or more *specific* contexts in terms of their taxonomical interpretations.

It can be noticed that there are several context formulations for such context model methodologies. The most important formulations are these, which have the ability to reason about context and to infer (unknown a-priori) situations or even possible world states of context parts entities. Context models that support such mechanisms are of high interest, and especially those models that can reason in polynomial time of complexity (e.g., currently, this constrain is only satisfied by the *Category* modal based context models [10]).

2.1. CONTEXT THEORETIC MODELING

2.1.1. Context as situation composition

In this situation theoretic context model, the concept of context is constructed by the captured events, or, activities into which contextual entities get evolved. A contexter [11] is a software abstraction that models relationships between observables. Contexters share a common I/O structure including control channels and meta-data to ensure and express Quality of Service (QoS). They can be composed as directed graphs or encapsulated into higher computational units (i.e., compound contexters). Hence, the authors in [11] expand the idea of context on the basis of the notion of *situation*. Specifically, contextual information is modeled as a vector of *snapshots* (i.e., values or contents) of observables, which *match* to predefined situations. The composition or *synthesis* of such *snapshots*, over time, forms context. Observables and their relationships can be mapped to *colonies* of contexters. Context at specific time derives from the composition of multiple situations (e.g., meeting, working) over time intervals. A situation may refer to a *specific* set of users involved in a particular task or set of tasks. Moreover, a contexter should be viewed as a means of retrieval of the values (or contents) of the observables that belonged to the corresponding context. The composition of the contexters implies the composition of the partial contexts, in order for the context-aware application to capture the whole context. Figure 2 provides a graphical representation of a contexter.

Context awareness should take into account the *roles* assigned to a context (i.e., the description of a user's behavior when engaged in a certain activity) and its *relations* with others. The authors in [12] view context as an *intermixture* of *modules*, which are defined as transformations of observations. Modules are assembled into reflective processes, enriched with meta-information, under the direction of a supervisory *controller*. A fundamental aspect of interpreting sensor observations (similarly to Contexters [11]) is to construct context by gathering observation. Hence, context is modeled as a *composition* of situations relevant to a task. Within a situation, context shares the same set of *roles* and *relations*. Then, the context model determines the appropriate collection of roles and relations for a certain task. Let U be the set of users to whom one attempts to assign specific *roles* (e.g., Alice is a "secretary" or "business manager") and their *relations* (e.g., Alice as secretary "presents confidential information" to her boss, but her colleagues cannot obtain such confidential information during the meeting). A role implies a certain action within a task T . The authors in [12] discuss the important problem of how to provide a mechanism for dynamically composing federations of meta-controllers that observe the *roles* and the *relations* over context. This approach is based on a rule-based

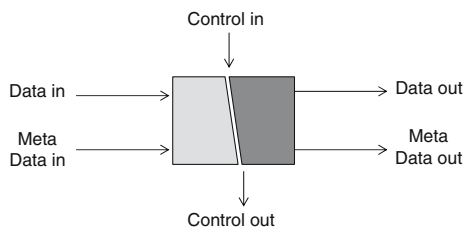


Figure 2. Graphical representation of contexter.

system written in JESS¹. Meta-supervisors are designed for specific contexts (i.e., every plausible situation has its own meta-supervisor) and maintain a *model* for the current context of the *U*. This model includes information about contexts that derive from the current context of *U*.

Figure 3 depicts the transformation of data (i.e., *roles* and *relations*) and *generated* events, undertaken by an observation process (i.e., contextual retrieval process), under the direction of a reflective controller.

2.1.2. Context as activity hierarchy

Considering context as a set of activities, as discussed in [13], focuses on a specific type of context. The latter describes the *performance* (i.e., realization) of an agent's activity. Consider a set of activities *A* as the activities that an agent may perform. A subset of these activities $B \subset A$, is the set of activities that the agent can *really* perform, and the set difference $D = A \setminus B$ is the set of the activities that the agent categorically cannot perform. Hence, context can be considered as the information that describes the performance of the activities of the set *B*. Activities may vary in *scope* (from general to specific). General activities often contain one or more specific activities (e.g., Alice is *moving*, from one room to another, is a more general activity, of these into which Alice is *running*, or *walking*, between the same rooms). The performance of a general activity *generates* a corresponding general context, given that more specific activities have already been performed, and then, have already generated more specific contexts. Such model of context differs from the previous approaches, because it focuses on context-aware applications that support mechanisms for recognition of activities, rather than context-aware applications that focus on *time*, *location* or other contextual information. An indicative representation of this model is provided in Figure 4.

2.2. CONTEXT CONCEPTUAL MODELING

This type of context modeling describes context as concepts and the relations among such concepts as binary or *n*-ary associations. Furthermore, such type of modeling categorizes context according to its prevalent characteristics.

The motivation of adopting several modals to context models has derived from the fact, that, applications need to be *sensitive* and more *reliable* to contextual information, as the latter is dynamically changing. The authors in [14] classify contextual information according to certain characteristics. Basically, the type of the contextual information exhibits certain temporal characteristics. Contextual information can be *static*, which means that it describes invariant

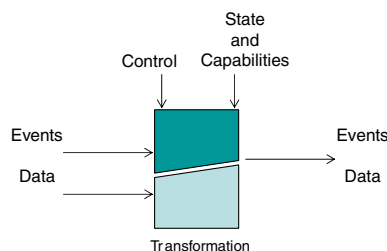


Figure 3. Transformation of data and events observed by a reflective controller.

¹ Rule engine for the Java platform [<http://herzberg.ca.sandia.gov/jess/>].

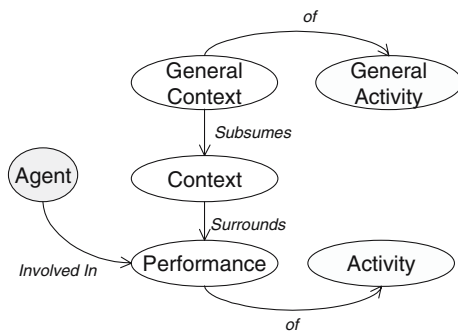


Figure 4. View of an activity-centric context.

aspects (e.g., Alice's date of birth), and *dynamic*, which refers to observations varying over time (e.g. Alice's location). *Static* context can be obtained *directly* from users. Frequently changing context can be obtained *indirectly* (e.g., through sensors). Hence, context could be *imperfect*, meaning that, its descriptive information is either *inconsistent*, or, *out of date*, or, *incomplete*. Moreover, a context model may support multiple *representations* of the same context in different *forms* (e.g., expressions) and at different *levels of abstraction* (e.g., logical representations versus concrete values).

2.2.1. Context as conceptual graph

In [14], a context model is described as a conceptual graph with context modeling, as concepts, and relations, as associations among concepts. In this model, several relations can be considered between three specific contextual entities: *person*, *device* and their *communication channels*. Such context model is assumed to be an object-based approach, in which, context is structured as a set of physical or conceptual concepts (e.g., PDA, persons, and communication channels). Properties of concepts are represented by attributes, and are linked to attributes uni-directional relations. Such model classifies relations, according to the change rate of their values, into two main parts: *static* relations and *dynamic* relations. The context captured by a *static* relation is typically known with a high degree of *confidence* (confidence can be considered as a modal operator). The context captured by *dynamic* relations through hardware or software sensors (e.g., widget [15]) is not *inserted* directly to the model, but is, somehow, transformed to bring it closer to the level of information abstraction required by context-aware applications.

Additionally, the model classifies the relations, according to their structure, into two types: *simple* and *composite* relations. A simple relation could be considered as a link between a concept and its property. A composite relation is a communication link from a Person to a set of her Activities. For managing such context model, it is reasonable to define a meta-model of relations known as dependencies. A dependency is a special type of relation that exists between relations. Specifically, a dependency can be qualified by a participation constraint (i.e., cardinality constraint), which limits the pairs of relations to whom that dependency is applied. Furthermore, the uncertain context may raise quality issues that this model could handle. The information systems community has extensively researched for Context Quality Modeling (CQM) [16]. According to CQM, the observables are tagged with various quality indicators, like coverage, resolution, accuracy, repeatability, frequency, and, freshness of context. Moreover, Figure 5 depicts such conceptual graph and the dependencies between

the corresponding relations. It could be noticed that, the dependency *dependsOn* restricts two relations *engagedIn* and *locatedAt*, which means that, Alice presents some information during the meeting activity given that her location is actually the meeting room.

2.2.2. Context as semantic graph

This approach is based on how to describe context, not as a graph representation, but as a set of propositions in some propositional logic. The statements of propositional languages are visualized as graphs, thus, the aforementioned context model is viewed as a conceptual graph of semantics. In fact, first-order logic systems can, accurately, assert context as predicates that are valid, for certain time intervals, and relations among concepts as n -ary predicates. The context model, in [14], can be described as a set of clauses with unary, or n -arty predicates. Finally, research work on semantics [9] has recently proposed a methodology for modeling such kind of knowledge, as conceptual graphs through Semantic Web languages (e.g., RDF(S) [17]).

3. Requirements of Context-Aware Applications

Applications should possess certain capabilities in order to be characterized as context-aware. Specifically, a context-aware application has to take into consideration a set of characteristics related to context modelling, handling and adaptation. Such characteristics could be:

- *Context acquisition*: A mechanism to obtain context data from diverse context sources. Context acquisition could be dealt with hardware sensors delivering information that conforms to a low-level data model.
- *Context aggregation*: A mechanism that provides context storing and integrity. In case of a shared context model, the context aggregation forms a basis for merging correlated contextual information. The context composition is a specific kind of context aggregation, when the involved contexts are compatible with the same, or equivalent, context model.
- *Context consistency*: Context consistency enables the rationality of dynamically changing distributed context models. Such mechanism, regarded as being an extended context aggregation mechanism, maintains the structure of the contextual model into higher levels of abstraction.

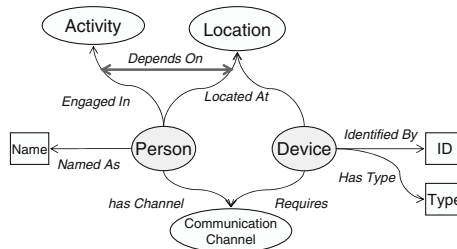


Figure 5. Context conceptual graph enriched with relations and dependencies.

- *Context discovery*: The aim of context discovery is to locate, and access contextual sources, in terms of serving context requests (e.g., discovering the appropriate, or approximate, context pertinent to an entity). Context discovery covers issues, such as, service description, advertisement, and discovery.
- *Context query*: By exploring contextual information, residing in distributed context repositories, a reference model needs a high-level mechanism for posing queries. 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, also, pose design issues as context query language (e.g., RDQL [18]), query optimizations, trigger messages, and, definitions of constraints related to context acquisition.
- *Context adaptation*: The application should be capable of adapting its behavior according to contextual information. Specifically, it, automatically, adapts the system configuration in response to a contextual change. For instance, the application may configure itself to use the display device available to the user. Moreover, context adaptation can be achieved by IF-THEN rules used to specify how context-aware software should respond intelligently to contextual changes.
- *Context reasoning*: Context can be elaborated with reasoning mechanisms. Context reasoning is a process for inferring *new* context, previously unidentified on the basis of a-priori known context. Reasoning tasks check context consistency and deduce high-level context. Such tasks can be realized through logical schemes like first-order predicates and description logics.
- *Context quality indicators*: Context data can come from heterogeneous context sources, such as, sensors, and software services. The lack of a universal context model and application-specific representation of the contextual data undermines the consistency of the sensed information. A mechanism for maintaining predefined sets of quality indicators is very important. Such indicators may be *resolution, accuracy, repeatability, frequency, and staleness* of context.
- *Context integration*: Context model outlines an expressive scheme for context representation and context interpretation. Existing context models vary in the expressiveness they support, in semantics, and in the abstraction level of the conceptual entities. Context model is based on (i) capturing general features of contextual entities, like activities, (ii) specific features, like temperature, and (iii) interrelations between contextual objects, like spatial relations. Contextual information integration (i.e., contextual semantic integration of the *individuals* of an ontology) can be conducted whenever different context models are in accordance, not only, with their semantics, but, also, with their *similar* domains of interest.

4. Context-Sensing Applications

The contextual information captured by a sensor-based system (e.g., [19–21]) is interfaced with an appropriate context model. Contextual information retrieval [22] is partly accomplished through sensor networks. Sensors could be considered as a rudimentary knowledge measurement engine, as they implement raw-data sensing techniques (e.g., location determination, temperature, acceleration, and light intensity measurement). On the other hand, they could be regarded as a more sophisticated mechanism, which can, for instance, determine that Alice’s situation is *at_meeting* (e.g., [23]). Sensor interaction with mobile devices introduces

many challenges to mobile computing. Apart from discovering services, or even resources, and, gathering context in a sensor-aware system, the more important challenge is the realistic detection of the user's context, assuming that it is not a-priori known. Sensor-based context detection for mobile users explicitly empowers human-computer interaction.

In this section, we outline some noteworthy techniques that are used in sensor networks to detect and disseminate contextual information to context-aware applications. Certain sensor-based applications have been selected in order to deal with the high usage of the contextual modals in realistic context-aware applications. Moreover, such modals determine how much context, and, in what type of its representation a context-aware application needs.

4.1. CONTEXT SENSING BASED ON COLLECTING KNOWLEDGE FROM NEIGHBORS

The work described in [24] focuses on a communication scheme for retrieving contextual information through autonomous sensors without centralized control. These sensors called Smart-Its [25], are aware of their sensing capabilities and can report them to their neighbors, if necessary. The idea of introducing an interoperable data format, describing sensor-features among Smart-Its, is based on the Smart Context-Aware Packets (sCAPs). Such protocol may be considered as a document-based approach for collecting sensor-features sharing some similarities with Context-Aware Packets (CAP) [26]. The sCAP is gradually filled with sensed information on its way through the environment. Each Smart-It that receives a sCAP contributes to the required sensor-features and forwards it to another Smart-It in its neighborhood. Combining the features stored in the sCAP allows each Smart-It to make assumptions about the current context. Based on this knowledge, it can forward this sCAP to an *appropriate* sensor for further processing. Figure 6 depicts the interaction of a mobile device using the context detection mechanism of sCAP.

4.2. ORGANIZING CONTEXT-SENSED INFORMATION IN LAYERS OF ABSTRACTION

Intelligent architectures that support multi-granular context descriptions (i.e., different context representations) are required, in order to model complex contextual information. The sensor devices range in complexity, as discussed in [27], from *simple* binary on-off reporting modules to sensors that can decide whenever a person is engaged in a certain activity at a specific location.

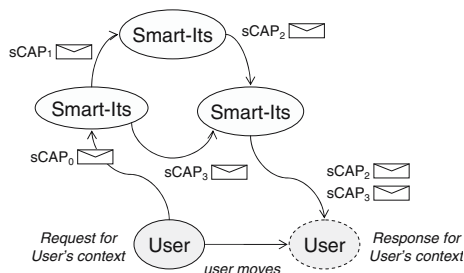


Figure 6. Context detection using sCAPs.

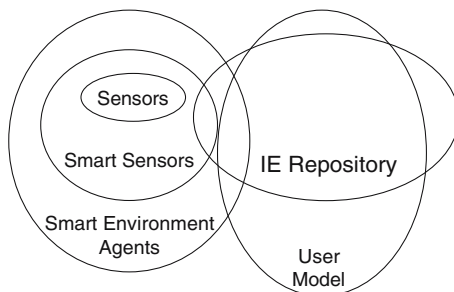


Figure 7. Merino agent environment.

Merino is an architecture for constructing context layers, as depicted in Figure 7. Merino consists of five elements: sensors, smart sensors, smart environment-agents, repository, and the user model. In the lowest level, sensors are mechanisms, both hardware, and software able to interrogate both the physical and computational environment. Smart sensors, forming the first layer of the context abstraction, are responsible for filtering and aggregating the raw sensors data into structures that are available through the repository interface. Such interface is a space where smart environment-agents can make use of the processed data, and, thus, provide a richer context. Smart environment-agents constitute the second layer of that context abstraction. These agents may be classified in two categories: *rich context agents* and *performance enhancers*. The former agent category accesses the contextual information from the repository in order to reformulate it to higher-level contextual information (called *rich-context*). Smart agents produce rich context that may be provided in varying levels of granularity. Moreover, the user model is managed by a smart personal assistant, which has access to the repository, in order to customize and configure the user needs.

4.3. SERVICE ADAPTATION BASED ON USER CONTEXT

The adaptation of the provided services to the user needs depends on the fact that the user context is self-described. This means that, such context describes user needs in terms of *constraints*, *preferences*, and *beliefs*. Such constraints are used for enabling service selection and execution, and, also, act as mediators for expressing service needs to the environment. Such service needs arise from the user context (e.g., a service that reminds Alice for her meeting).

The Context-Aware Packets Enabling Ubiquitous Services (CAPEUS) [28] system *discovers*, *selects*, and *executes* services, with regard to the current user context. CAPEUS adopts the *situation composition* and *conceptual graph* context models. Such models discussed in Section 2. The discussed system adopts a standard document format, known as Context-Aware Packets (CAP), in order to describe service needs and constraints on a logical level. The CAP is initiated by the user and placed in the network, where it is 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 take into account user's service needs. A Context Constraint is further analyzed into three entities: the *Abstract*, the *Relation*, and the *Event* entity. The former relates to the service peer, sensor, or, person. The Relation entity describes dependencies related to the service selection. Events, which are represented by logical conditions, report situations *detected* by sensors, thus forming a *trigger*. The scripting part

represents simple scripts to be executed during service invocation, whilst the data section is the prerequisite data to be processed by the service.

Consider the following scenario: A user injects a CAP to a local Service Access Node (SAN). 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 fulfill these service needs. In the second phase, the service is executed according to the contextual constraints, being described in the CAP, and, if possible, to the user context.

A demonstration of a printing service execution related to contextual constraints is depicted in Figure 8. Alice wants to print a PDF file for her presentation at the meeting, but the nearest printer can not support this type of file. Then, a CAP is transparently injected in the network (step [1]) and the SAN routes this CAP to the appropriate SAN (step [2]), which can convert that PDF file to the type of file that the nearest printer supports. Finally, the converted data in the CAP is printed (step [4]).

5. Context Aware System Architectures

This section provides a brief introduction to well-known architectures for supporting context-aware systems. Such architectures rely on sensor-based context detection. Some context detection is needed in order to: (a) enrich context with semantics (b) provide secure access to the context (c) distribute context to grid applications, and (d) attempt to integrate heterogeneous context semantics with other context-aware applications, thus forming a Semantic Web community. The common underlying philosophy of context-aware system architectures [29–32] is their *hierarchical* structure. Such layered structure covers two levels of context utilization: the *operational* and the *informational* level.

On the operational level, the system modules, which are distributed in a mobile computing environment, may serve as: (a) Sensors that capture raw data (b) Autonomous mediators that process and filter raw data streams transforming them into higher data representations (c) Smart agents that communicate with each other, in order to mine knowledge that resides in the system (d) Context-aware applications that provide innovative services to end users catering for user context adaptation and maintenance of an integrity scheme for that context.

With respect to the informational level, knowledge representation may focus on: (a) A simple data model without semantics or meta-data (b) Modeling large amount of data using relational or object-oriented paradigms (c) Serving as an ontology that describes distributed

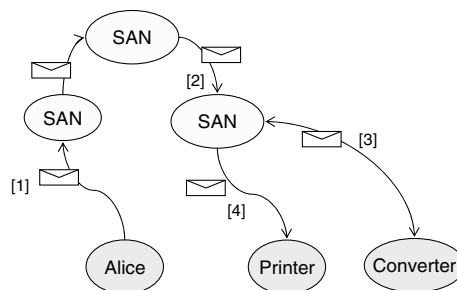


Figure 8. Using CAP in a ubiquitous manner.

knowledge resources, such as user profiles, device capabilities, service features, and, even, contexts of applications, and (d) Context integration, in terms of contextual information retrieval (e.g., similarity methods taking data semantics into consideration). The operational level coupled with its informational counterpart comprises a universal context-aware system model. 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. The following paragraphs refer to several system architectures, which fulfill the requirements of context modeling and implement some of the features of mobile computing environments (e.g., context management, representation, and security).

5.1. ARCHITECTURES FOR CONTEXT-AWARE WEB SERVICES

The Web Architectures for Services Platforms (WASP) [33, Un published] is a project dealing with the definition of a service platform, which supports the development and the deployment of context-aware integrated speech and data applications, based on Web Services technology, on top of 3G mobile networks. The contextual information in this system is modeled by conceptual graphs describing ontologies of context-aware services (Section 2). Such platform provides services to *Context providers*, which communicate through the *Context interpreter* module. The latter gathers context and makes it available to the rest of the platform. The platform consists of a set of *Repositories*, which support the *Monitor* component with knowledge related to the elements involved in WASP. Repositories collect information from the context interpreter (e.g., user preferences and constraints) and make use of the services of the service providers. The Monitor component is responsible for managing application subscription by using a WASP Subscription Language (WSL) and by gathering information from both Repositories and Context Interpreters. Applications use WSL during their subscription, or the configuration of the platform, so as to react to a given set of events. The specific context-model is relevant to Data Entities. Data Entities represent objects of the real world (e.g., person, activity, device, location). Attributes and associations are, also, combined with Data Entities. Different Data Entities must share common contextual representation, allowing the derivation of more complex context. The WASP exploits the Semantic Web Technology, building contextual information using ontologies from ontology-based markup languages (e.g., DAML+OIL, OWL). The platform could be invoked by different conceptual layers, such as from context storage to adaptive interfaces and from service description-discovery to complex service composition.

5.2. AGENT-BASED CAS

Among the critical research issues in developing context-aware systems are context modeling, context reasoning, knowledge sharing, and protection of user context. The system presented in [33], addresses such issues by developing an agent-oriented architecture, the Context Broker Architecture (CoBrA). The CoBrA is considered as a large scale implementing paradigm, which models its contextual information through semantic graphs (e.g., ontologies). Such graphs describe user profiles, user preferences, and device capabilities. 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 functionality: (a) a collection of

ontologies for modeling contextual information (i.e., Context Conceptual Modeling methodology) (b) a shared model for the current context, and (c) a declarative policy language that users and devices may use for defining constraints on their sharable resource (e.g., personal agenda). The CoBrA uses languages from the Semantic Web for defining contextual ontologies providing not only a semantically richer context representation, but, also, making use of the ability of reasoning and sharing knowledge. CoBrA provides a resource-rich agent, called *context broker*, to manage and maintain the shared context into consistency. A context broker is associated with a certain smart spaces environment. It may be considered as an aggregation of other brokers representing *smaller* parts of the original smart space environment. Such hierarchical approach, with the support of shared ontologies, is capable of avoiding the bottlenecks associated with a single centralized broker. The Context broker can, also, infer contextual information that cannot be easily acquired from sensors. Furthermore, it can detect and resolve inconsistent knowledge that often emerges as a result from imperfect context sensing. Moreover, CoBrA provides a policy language that allows users to *control* the provision of 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 inside the environment.

Another infrastructure that enables scalable and flexible sensor-based services and makes use of the agent technology is that of Irisnet [34]. IrisNet adopts a hierarchical system model, which forms a two-tier hierarchy of sensing nodes and information brokering nodes. It, substantially, reduces network bandwidth requirements through the use of *senselets*. Senselets are binary code fragments that perform intensive data filtering at the sensing nodes. Irisnet includes advanced sensor devices, which form a wide area sensor network for context processes. Such processes refer to distributed filtering, hierarchical caching, query routing, and context freshness evaluation (e.g., when to update the contextual repository). The two tiers of agents are: (a) the Sensing Agents (SA), and (b) the Organizing Agents (OA). The former category of agents collects and filters sensor readings according to a data model, whilst the later category performs query-processing tasks on those 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 in order to instruct and perform tasks (e.g., collecting the required information, filtering, and caching) and to transmit the *distilled* information to the OA. The OA and SA execution environment uses a service-specific processing and filtering over the sensor feeds, which eliminates duplicated and redundant information. Figure 9 represents an Irisnet instance.

The My Campus system [35] is an agent-based environment for context-aware mobile services. It revolves around a growing collection of customizable agents capable of automatically discovering and accessing Internet and Intranet services. The scalability of such architecture is attributed to the use of ontologies describing contextual information (e.g., user preferences and constraints). Moreover, agents focus on context-sensitive message filtering, message routing, and context sensitive reminding. More *sophisticated* agents incorporate planning and automated Web Service access functionality.

5.3. CAS BASED ON SPATIAL INFORMATION

One of the most important concepts in the mobile computing context is that of *location*. Architectures, like ParcTab[36] process the location information through suitable spatial models

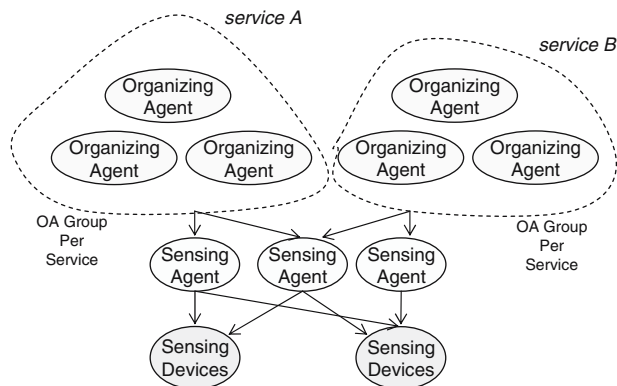


Figure 9. Iris net instance.

(e.g., the Geometric Model WGS84²). RAUM [37] system develops context-aware applications based on the contextual information retrieved by the user location. The system supports context generated by an appropriate spatial model (i.e., Context conceptual modeling methodology). Such context model is based on conceptual graphs representing hierarchies of symbolic locations (see Section 2). Spatial contextual information (location symbolic model) is based on the relative location of entities (e.g., users) rather than on their identity. The RAUM is a spatial-aware communication model, in which, two entities are considered contextually relevant (i.e., the relative location information is the compatibility relation between those entities) through their locations rather than their network identifiers. Such 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 architecture, whilst the CM defines how location information is used in the communication among the RAUM entities. Figure 10 depicts the RAUM – LRM adopted a tree presentation for location selection. Such logical representation consists of three general layers: (a) a tree-root (b) the semantic sub-layers, and (c) a location expressed in three-dimensional Cartesian coordinates. Further specialization of the third layer into sub-sections enables a more fine grained differentiation of locations.

The RAUM system makes use of distributed storage of the location information, as long as all the involved objects are capable of handling such information. Hence, no central entity for storing and providing the complete location-tree is required. All the objects (e.g., mobile

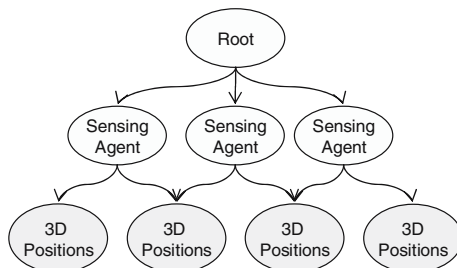


Figure 10. RAUM Tree spatial model.

² World geodetic system [<http://www.wgs84.com/>]

devices) in the system *hold* only the part of the tree that is *pertinent* to them. In this respect, most objects have only to store the path through the tree by representing their own location. This implies that, RAUM is used whenever peer-to-peer communication is required.

6. Applications of Large Scale Context-Aware Systems

In this section, we refer to certain abovementioned indicative context-aware systems. The discussed systems depict the enabling technology for building and testing mobile and ubiquitous computing scenarios, and the emerging functionality and collective context-awareness of information artifacts.

Smart-Its [25] are small-scale, embedded devices that can be attached to everyday objects to augment them with sensing, perception, computation, and communication. Several device prototypes are based on Smart-Its using diverse micro-controller platforms. Smart-Its are used for multi-sensing contextual information for mobile users, similarly to the Smart-CAPs [26]. Moreover, *Smart-Its Friends* is a context-aware application based on this technology. As soon as a device becomes connected, the application notifies the mobile user with a brief tone. This notification also occurs after a *friend* has been temporarily out of range and, thus, disconnected. Two Smart-Its-augmented objects can be connected and, then, notify the mobile users when they venture within a certain range, acting in support of *proximity awareness*.

The RAUM system is currently used for inter-device communication in ubiquitous computing. The RAUM location-aware application runs on desktop computers, Palm Pilots, and micro-controllers. The *Smart-Door-Plate* context-aware application is used to yield context information for applications in human computer interaction. Specifically, it displays the name of the room, the names of the people working in that room, or events taking place there (e.g., a meeting). The information for recognizing that a meeting is in progress is inferred from the context established by MediaCups [38]. The later is a computer-augmented coffee cup with infrared communication, which distributes status information, like temperature, sound and usage in specific time intervals.

Finally, a smart meeting room system called *Easy-Meeting* explores the use of multi-agent systems, Semantic Web ontologies, reasoning, and declarative policies for security and privacy. Building on the CoBrA system [33], *Easy-Meeting* provides relevant services and information to meeting participants based on their situational needs. The CoBrA intelligent broker agent maintains a shared context model for all computing entities in the space and enforces user-defined privacy policies. In addition, CoBrA system models its context using a shared conceptual graph (Section 2) for context-aware computing application, called Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA). Such ontology is expressed using the Semantic Web languages and includes modular component vocabularies to represent intelligent agents with associated beliefs, desires, and intentions, time, space, actions and events, and user profiles. SOUPA can be extended to support the applications of CoBrA, and MoGATU [39], a peer-to-peer pervasive data management system.

7. Conclusions and Open Issues

In this article, emphasis was placed on context-aware mobile computing. Essentially, the form of such computing paradigm is broader than mobile computing because it concerns not only mobility of users and devices, but, more importantly, information surrounding the user and

computational entities. In context-aware mobile computing, the application adapts not only to changes in the availability of computing and communication resources but also to the presence of contextual information. Issues related to context modeling and how context aware systems deal with contextual information are discussed. We reviewed the basic mechanisms underlying the operation of such systems. We discussed notable works and important architectures reported in the relevant literature. A long-term goal of system architectures is rendering sensors and context platforms flexible and scalable enough to be widely adopted in various context-aware mobile 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 allowing computers to 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 the future. Systems should also take into account the quality of context represented into a model supported by a meta-model scheme of context. Finally, an open issue could be the definition of a context prediction method supporting the pro-activity of context-services in advanced mobile computing environments.

References

1. E. Giunchiglia and F. “Giunchiglia, and Ideal and Real Belief about Belief”, *Proc. FAPR'96*, Bonn, Germany, 1996, pp. 261–275.
2. F. Giunchiglia F, “Contextual Reasoning, Epistemologia”, special issue on I Linguaggi e le Machine, Vol. XVI pp. 345–364, 1993.
3. G. Fauconnier, *Mental Spaces: Aspects of Meaning Construction in Natural Language*, Cambridge, MA, MIT Press, 1985.
4. R. Weyhrauch, “Prolegomena to Theory of Mechanized Formal Reasoning”, *Artificial Intelligence*, Vol. 13 No. 1, pp. 133—176, 1980.
5. J. McCarthy, “Notes on Formalizing Context”, *Proc. 13th IJCAI'93*, Chambéry, France, pp. 555–560, 1993.
6. A. Dey and G. Abowd, “Towards a Better Understanding of Context and Context-Awareness”, *Proc. CHI'00*, The Hague, The Netherlands, 2000.
7. C. Ghidini and F. Giunchiglia, “Local Models Semantics, or Contextual Reasoning = Locality + Compatibility”, *Artificial Intelligence*, Vol. 127, No. 2, pp. 221–259, 2001.
8. H. Levesque, F. Pirri, and R. Reiter, “Foundations for the situation calculus”, *Electronic Transactions on Artificial Intelligence*, Vol. 2, No. 3–4, pp. 159–178, 1998.
9. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P Patel-Schneider, *The Description Logic Handbook – Theory, Implementation, and Applications*, Cambridge, Cambridge University Press, 2003.
10. D. Grossi, F. Dignum, and J. Meyer, “Context in Categorization”, *Proc. CRR'05*, Paris, France, 2005.
11. J. Coutaz and G. Rey, “Foundations for a theory of context”, *CLIPS-IMAG, BP* Vol. 53, pp. 283–302, 2000.
12. J. Crowley and P. Reigner, “An architecture for Context Aware Observation of Human Activity”, *Project PRIMA, INRIA*, France.
13. P. Prekop and M. Burnett, “Activities, Context and Ubiquitous Computing”, *Computer Communications*, Vol. 26, pp. 1168–1176, 2003.
14. K. Henricksen, J. Indulska and A. Rakotonirainy, Modeling Context Information in Pervasive Computing Systems, *Proc. Pervasive 2002*, Zurich, Switzerland, pp. 169–180, 2002.

15. IOS Widgets, [http://mdp.artcenter.edu/~vanallen/ios1/2004sp/ios1_wk02d.html].
16. R. Wang, M. Reddy, and H. Kon, Towards Quality Data: An Attribute-Based Approach, *Decision Support System*, Vol. 13, pp. 349–372, 1995.
17. D. Brickley, and Guha, “RDF Vocabulary Description Language 1.0: RDF Schema”, W3C Working Draft. <http://www.w3.org/TR/PR-rdf-schema> (visited April 2004).
18. A. Seaborne, “RDQL – A Query Language for RDF”. W3C Member Submission. <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/> (visited February 6th 2004).
19. R. Want, A. Hopper, V. Falcao, and J. Gibbons, “The Active Badge Location System”, *ACM Transactions on Information Systems*, Vol. 10, No. 1, pp. 91–201, 1992.
20. M. Beigl, H.W. Gellersen, and A. Schmidt, “MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects”, *Computer Networks*, Vol. 35, No. 4, pp. 401–409, 2001.
21. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.W. Gellersen, “Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts”, *Proc. UBICOMP*, California, USA, 2001.
22. T. Bauer and B. Leake, “Exploiting Information Access Patterns for Context-Based Retrieval”, *Proc. IUI’02*, Florida, USA, pp. 176–177, 2002.
23. J. Cooperstock, “Making the User Interface Disappear: the Reactive Room”, *Proc. Centre for Advanced Studies on Collaborative research*, Toronto, Canada, 1995.
24. F. Michahelles and M. Samulowitz, “Smart CAPS for Smart Its – Context Detection for Mobile Users”, *Proc. IHM-HCI*, Lille, France, 2001.
25. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.W. Gellersen, “Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts”, *Proc. UBICOMP*, California, USA, 2001.
26. M. Samulowitz, F. Michahelles, and C. Linnhoff-Popien, “Adaptive Interaction for Enabling Pervasive Service”, In: *MobiDE01*, USA, 2001.
27. B. Kummerfeld, A. Quigley, C. Johnson, and R. Hexel, Merino: “Towards an intelligent environment architecture for multi-granularity context description”, *Proc. User Modeling for Ubiquitous Computing*, Johnstown, USA, 2003.
28. M. Samulowitz, F. Michahelles, and C. Linnhoff-Popien, CAPEUS: “An Architecture for Context-Aware Selection and Execution of Services Distributed Applications and Interoperable Systems”, *Proc.DAIS*, 23–40, Krakau, Poland, 2001.
29. T. Nakajima, “Pervasive Servers: A Framework for Creating a Society of Appliances”, *Personal and Ubiquitous Computing*, Vol. 7, No. 3–4, pp. 182–188, 2003.
30. C. Efstratiou, K. Cheverst, N. Davies, and A. Friday, *An Architecture for the Effective Support of Adaptive Context Aware Applications*, Distributed Multimedia Research Group, Lancaster University.
31. B. De Carolis, S. Pizzutilo, I. Palmisano, and A. Cavalluzzi, “A Personal Agent Supporting Ubiquitous Interaction accepted to Workshop on User Modelling for Ubiquitous Computing”, *Proc.UM03*, Johnstown, USA, 2003.
32. WASP Project, [<http://www.freeband.nl/kennisimpuls/projecten/wasp/>].
33. H. Chen, T. Finin, and A. Joshi, “An Ontology for Context-Aware Pervasive Computing Environments, Special Issue on Ontologies for Distributed Systems”, *Knowledge Engineering Review*, Vol. 11, Issue: 3, pp. 197–207, 2003.
34. S. Nath, Y. Ke, P. Gibbons, B. Karp, and S. Seshan, *IrisNet: An Architecture for Enabling Sensor-Enriched Internet Service*, Intel Research Pittsburgh, Carnegie Mellon University, IRPTR-02–10, Bologna, Italy, 2002.
35. N. Sadeh, E. Chan, and L. Van, “MyCampus: An Agent-Based Environment for Context-Aware Mobile Services”. *Proc. UBIAGENTS*, pp. 34–39, 2002.
36. B. N. Schilit, N. Adams, R. Gold, M. Tso, and R. Want, The PARCTAB Mobile Computing System, In: *Workshop on Workstation Operating Systems*, pp. 34–39, 1993.
37. M. Beigl, T. Zimmer, and C. Decker, “A Location Model for Communicating and Processing of Context”, *Personal and Ubiquitous Computing*, Vol. 6, 5–6, pp. 341–357, 2002.
38. M., Beigl, H.-W. Gellersen, A. Schmidt, 2002. “MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects”. *Computer Networks*, Vol. 35, No. 4, Special Issue on Pervasive Computing, Elsevier, Amsterdam, pp. 401–409, March 2001.
39. F. Perich, “On Data Management in Pervasive Computing Environments”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 5, pp. 621–634, 2004.



Christos B. Anagnostopoulos has received his B.Sc. in Computer Science from the Department of Informatics and Telecommunications at the University of Athens, Greece in 2001 and his M.Sc. in Computer Science - Advanced Information Systems from the same department in 2003. He is now a Ph.D. student in the University of Athens - Department of Informatics and Telecommunications. His research interest is focused on Contextual Reasoning, Web Semantics, Ontological Engineering and Uncertainty Management. Since 2004 he is a member of the Pervasive Computing Research Group of Communication Networks Laboratory of University of Athens.



Athanasios Tsounis has received his B.Sc. in Computer Science from the Department of Informatics and Telecommunications at the University of Athens, Greece in 2002 and his M.Sc. in Computer Science - Advanced Information Systems from the same department in 2003. He is now a Ph.D. student in the University of Athens - Department of Informatics and Telecommunications. His research interest is focused on Smart Agents and Ontological Engineering.



Stathes Hadjiefthymiades received his B.Sc., M.Sc. and Ph.D. degrees in Informatics from the Dept. of Informatics and Telecommunications, University of Athens (UoA). He also received a Joint Engineering-Economics M.Sc. from the National Technical University of Athens. In 1992 he joined the Greek consulting firm Advanced Services Group. In 1995 he joined the Communication Networks Laboratory (CNL) of UoA. During the period 2001–2002, he served as a visiting assistant professor at the University of Aegean, Dept. of Information and Communication Systems Engineering. On the summer of 2002 he joined the faculty of the Hellenic Open University, Patras, Greece, as an assistant professor. Since December 2003, he is in the faculty of the Dept. of Informatics and Telecommunications, University of Athens, where he is presently an assistant professor. He has participated in numerous EU & National projects. His research interests are in the areas of web engineering, mobile/pervasive computing and networked multimedia. He has contributed to over 100 publications in these areas. Since 2004 he co-ordinates the Pervasive Computing Research Group of CNL.