Context-Aware Browser

Paolo Coppola[1], Vincenzo Della Mea[1], Luca Di Gaspero[2], Danny Mischis[1], Stefano Mizzaro[1], Elena Nazzi[1], Ivan Scagnetto[1], Luca Vassena[1] *
[1] Department of Mathematics and Computer Science
[2] Department of Electrical, Management and Mechanical Engineering
University of Udine
Italy
coppola@uniud.it, dellamea@dimi.uniud.it, l.digaspero@uniud.it, mischis@dimi.uniud.it, mizzaro@dimi.uniud.it, elena.nazzi@dimi.uniud.it, scagnett@dimi.uniud.it, vassena@dimi.uniud.it

*Authors are listed alphabetically
Name of the corresponding author: Stefano Mizzaro
E-mail address: mizzaro@dimi.uniud.it
Full postal address: Department of Mathematics and Computer Science, University of Udine - Via delle Scienze, 206 - I33100 Udine - Italy
Telephone and fax numbers: +39 0432 558456 +39 0432 558499

Abstract. We propose the Context-Aware Browser, a new approach to context-aware Web content perusal by means of mobile devices. The Context-Aware Browser exploits artificial intelligence techniques and mixes several ingredients: it is a Web browser running on one's own mobile device and capable of automatically and proactively downloading and executing context-aware Web contents, selected by querying a search engine on the basis of the context. The novelty of the proposed approach is twofold. First, we take into account the information provided by the surrounding environment in order to carry out a more refined search of web contents. Second, information is not only automatically pushed towards the mobile device and then filtered: rather, the CAB approach involves a two stage retrieval plus filtering process, that minimizes bandwidth and privacy and security risks. As an overall outcome, the effort required on behalf of the user is minimized as well.

1. Introduction

The typical scenario of a user seeking information on the Web by means of a classical browser requires a somehow significant effort in order to get the desired information (web pages, applications, resources, etc.). The user must provide a query using a web search engine (e.g., Google) and she also has to check the results, in order to see if they really provide the desired information. Otherwise, the query must be refined, re-submitted and so on, giving rise to an iterative process.

Thus, the above mentioned information seeking activity is rather cumbersome for the final user both from a cognitive viewpoint (thinking of the right query to submit, checking the obtained results) and from a practical one (lots of keystrokes, screen scrolls: in general many interactions with the user interface). If this can be acceptable in the everyday use of a desktop system by a computer scientist or by people familiar with such interactions, it becomes a serious issue when the same task has to be carried out on a PDA or a mobile phone. Indeed, the graphical user interfaces and the input/output peripherals of such devices are rather limited when compared to their counterparts on a classical PC; moreover, the user is often a person not very familiar with computer applications and the way search engines usually operate. Nevertheless, in a world where information plays an essential role, it can be useful or even crucial to get the desired information in a quick way even when we are far away from a classical computer. For instance, in an airport it could be useful for a passenger to receive on her own cellular phone the details of her flight (location of the check-in desk, gate number, etc.) without having her to put too much effort in retrieving such information. Moreover, the best solution would be to develop an infrastructure capable of "automatically" providing such data as soon as the passenger arrives at the airport.

In addition, another important, and much discussed, limitation of current search engines is to necessarily ignore the "context" the user is in, when she submits a query. This "ignorance" implies that the answer generated according to the query itself will not take into account many data (potentially suggested by the surrounding environment) that could refine the set of results in a useful way, providing the user with a more suitable and exploitable information. Going back to the previous example, a simple event like the arrival of a passenger (which can be spotted-out, e.g., by means of the Bluetooth device of her cellular phone) could be used to determine the context "the user is in the airport". Then, if the passenger updated her agenda with a note about the flight number, such data could be communicated by a suitable "background" application to the airport server (for example exploiting again the Bluetooth connection), in order to receive back the details of the flight. In this case the event of receiving the flight number would enrich the previously inferred context, determining that "the user is at the airport in order to take that particular flight". This in turn could automatically generate further information "pushing" towards her mobile phone. Notice that, in the whole process, no effort is required on behalf of the user, besides enabling the Bluetooth antenna of her mobile device and activating the above mentioned "smart application". Of course, the above scenario presents obvious privacy and security issues: people could require that their location is kept private, at least in certain situations; data downloaded on one own cellular phone might be malware; and so on. Therefore, suitable policies and mechanisms have to be implemented, in order to protect sensitive information stored on mobile devices and to ensure user's anonymity.

With these considerations in mind, we propose a new approach in the fruition of web contents, i.e., the so-called Context-Aware Browser (CAB for short), which will play the fundamental role of the "smart application" briefly sketched in the previous scenarios. More in detail, the CAB will be able to collect the data from the surrounding environment, to search web contents suitable for the current context, to automatically load and display the selected web contents and to automatically offer web pages and web applications. To have an application capable of such a complex behavior is not simple at all: different artificial intelligence techniques are exploited, as described in the following, and they are crucial to have an effective CAB.

The paper is organized as follows: in Section 2 we briefly survey some related work in the field of context-awareness and mobile computing. Section 3 is devoted to illustrating the CAB overall architecture, while the individual modules are taken into account in Section 4. A typical usage scenario in presented in Section 5, evaluation issues are discussed in Section 6, and final considerations and future work are dealt with in Section 7.

2. Related work: context-awareness and mobile computing

Several research fields are relevant to our approach. We briefly survey the work that has been done on context awareness, on context-aware retrieval, and on frameworks for context-based applications. We discuss more in depth the MoBe framework since, as it will be clear in the following, it is the basis of CAB development.

2.1. Context-awareness

2.1.1. Context

The concept of context is still a matter of discussion and through the years several different definitions have been proposed. They can be divided into extensional and intensional definitions.

Extensional definitions present the context through a list of possible context dimensions and their associated values. In the first work that introduces the expression context-aware [22], the context is represented by the location of the user and the surrounding objects. In a similar way, Brown et al. [6] define context as location, proximity to other people, temperature, daytime , etc. In [17] the concept of context is divided in three categories: computing context (network, display, etc.), user context (profile, people nearby, social situation, etc.) and physical context (light, noise, etc.). Chen et al. [8] add two others categories: time context (day, month, etc.) and history.

Intensional definitions present the concept of context more formally. In [1] the context is defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". For Brazire and Brezillion [3], "the context acts like a set of constraints that influence the behavior of a system (a user or a computer) embedded in a given task". This definition moves from the analysis of a collection of 150 context definitions from several fields of application as sociology, computer science, etc.

Extensional definitions seem to be useful in practical applications, where the abstract concept of context has to be made concrete. However, from a theoretical point of view they are not properly correct, as the context cannot be outlined just by some of its aspects. On the other hand intensional definitions are of little use in the practice, despite they are theoretically satisfying.

2.1.2. Context-aware computing

Context-aware computing can be defined as the use of context in software applications, where the applications adapt to discovered contexts by changing their behavior [8]. From an extensional point of view, a context-aware application presents the following features [18, 11]: context sensing, presentation of information and services to a user, automatic execution of a service, and tagging of context to information for later retrieval.

In most earlier context-aware applications, the notion of context included just a small amount of data. Most researches, for example, focused on time and/or location and other few data [28, 24]. More complex approaches tend to combine several contextual values to generate new contextual information. In [1] primary contexts as location, entity, activity, and time, act as indices into other sources of contextual information. Similarly, in the TEA Project [23] Schmidt et al. use a resolution layer to determine a user's activity starting from basic contextual information.

As in the previously cited works, we intend to combine contexts to determine new, more abstract contexts. Differently from them, however, we do not want to limit the inferences only on an a priori defined set of contextual dimensions, but we aim to develop an inferential infrastructure able to work in a general way.

2.2. Context-aware retrieval

Context-aware retrieval is a rather new field of interest whose importance has increased with the rise of context-aware applications and the increasing amount of information managed by them. A first work in this field is Just-In-Time Information Retrieval agents: a class of software agents that proactively present information based on a person's context in an easily accessible and unobtrusive manner and in a general, task-independent way [20]. Context-aware retrieval can be described as an extension of classical information retrieval that incorporates the contextual information into the retrieval process, with the aim of delivering information relevant to the users within their current context [7, 14].

The context-aware retrieval model includes the following elements [7]:

- a collection of discrete documents;
- a set of user's retrieval needs, captured in a query;
- a retrieval task, to deliver the documents that best match the current query, rated on the basis of a relevance measure;
- the user's context, used both in the query formulation and associated with the documents that are candidates for retrieval.

Two different approaches could be identified [7]: user-driven and author-driven. In the former, on the basis of the user's context, a retrieval request may be issued automatically; in the latter, each document in the collection has a trigger context; if it matches the user's current context, the document is retrieved.

Typically, context-aware retrieval applications present the following characteristics [14]:

- user mobility, i.e., a user whose context is changing;
- interactive or with automatic actions if there is no need to consult the user;
- time dependent, since the context may change;
- appropriate and safe to disturb the user.

Context-aware retrieval is gaining increasing importance at a fast pace. A recent user survey [26] confirms that users would like context-aware retrieval applications: 20 participants in a study on mobile information needs were asked how they would feel about a tool that could predict their information needs and provide appropriate information at right time; 85% of the participants responded positively, while only the 15% were hesitant and concerned with privacy and control issues.

2.3. Frameworks

2.3.1. Stick-e Notes

Stick-e Notes [5, 6] is a framework developed around an idea which is the electronic equivalent of Post-its: the user can associate notes (texts, HTML pages, sounds, videos, programs, etc.) to different contexts (location, daytime, etc.) in such a way that, when the event will happen again, the relative note will be activated. This framework was conceived as the basis upon which to build in an easier way context-aware applications.

2.3.2. Cooltown

The Cooltown project [15] project aims at establishing a context model based upon the Web and, in particular, upon the concept of "Web Presence"; indeed, the main idea is to associate virtual services to physical entities. This approach is motivated by the fact that many databases and web pages contain a huge amount of information about the physical world; however, such information remains "detached" from the latter. So, physical entities are classified into people, places and things (since, a person during his everyday life can meet people, move to different places and interact with things) and their relative access modality is expanded to a virtual world of web contents, where people, places and things are represented by means of the "Web Presence" context. This essentially amounts to the idea of being accessible in the Web, connecting the advantages of information systems with the physical world and vice versa.

2.3.3. Context toolkit

The Context toolkit [10] is the result of several research activities at GeorgiaTech with the aim of building an architecture supporting the development of context-aware applications. The overall approach is object-oriented, being characterized by three kinds of objects, namely, widgets, servers, and interpreters. Context widgets are essentially software components isolating applications from the activities bound to context sensing. They are reusable components defined by attributes ("pieces" of contextual information available to other components) and callback methods (representing the kind of events that can be notified by the widget). Context servers are collectors used to gather the whole context of a particular entity like, e.g., the user; they must register themselves with each widget they are interested in, acting as proxies for applications. Finally, Context Interpreters manage the interpretation of contextual data, allowing one to present such information in different formats and to combine several contextual data in new pieces of information.

2.3.4. Sparkle

Sparkle [25] is a platform for context-aware application-level state management. It is based on context-aware state capturing and restoring mechanism that can achieve context-aware application migration. In the Sparkle system, applications are composed of facets. Facets are pure functional units which are independent of data or user interface. Facets do not interact with the user and do not maintain any application state. Hence, every application is associated with a container. The container contains the user interface and stores the data and execution state. It also stores the set of functionalities that the application can offer. Facets are housed on facet servers on the network. An application, during execution, will request for a certain functionality. There may be more than one facet which fulfills the same functionality. At run-time, the facet which is most suitable for the run-time environment is brought in and executed.

2.3.5. Hydrogen

Hydrogen [13] is a framework for distributed context-aware applications, based on the idea of not having a centralized component to depend on. Hydrogen is an architecture, which not only contains the context of the location device, but is open to represent the context of remote devices too. In case of encountering another device, context information can be mutually exchanged (via WLAN, Bluetooth etc.) enabling a local representation of the remote device's context. This exchange of context information among client devices is called "context sharing".

2.4. MoBe

MoBe [9] (MoBe stands for Mobile Being, an acronym inspired by the more famous Being Digital by Negroponte) is a general architecture for context-aware distributed applications on mobile devices based on the dynamic and automatic download, configuration, execution, and unload of applications on the basis of the user's current context. Rather than having the applications rigidly installed on a mobile device, the user and device context is used to obtain and start useful applications and to discard the not anymore useful ones. This way, a device is not limited to a set of predetermined functionalities, but allows one to adopt those which are probably more useful for the user at a given time, in a given place.

MoBe architecture is shown in Figure 1 and is composed by the following three layers (from bottom to top):

- MoBeSoul: it is the middleware whose basic responsibility are to sense the surrounding environment, to perform the context inferences and to manage the retrieval of applications;
- Application framework: it consists of the software infrastructure for building the concrete mobile applications;
- MoBeLet: basic context-aware applications built upon this architecture.

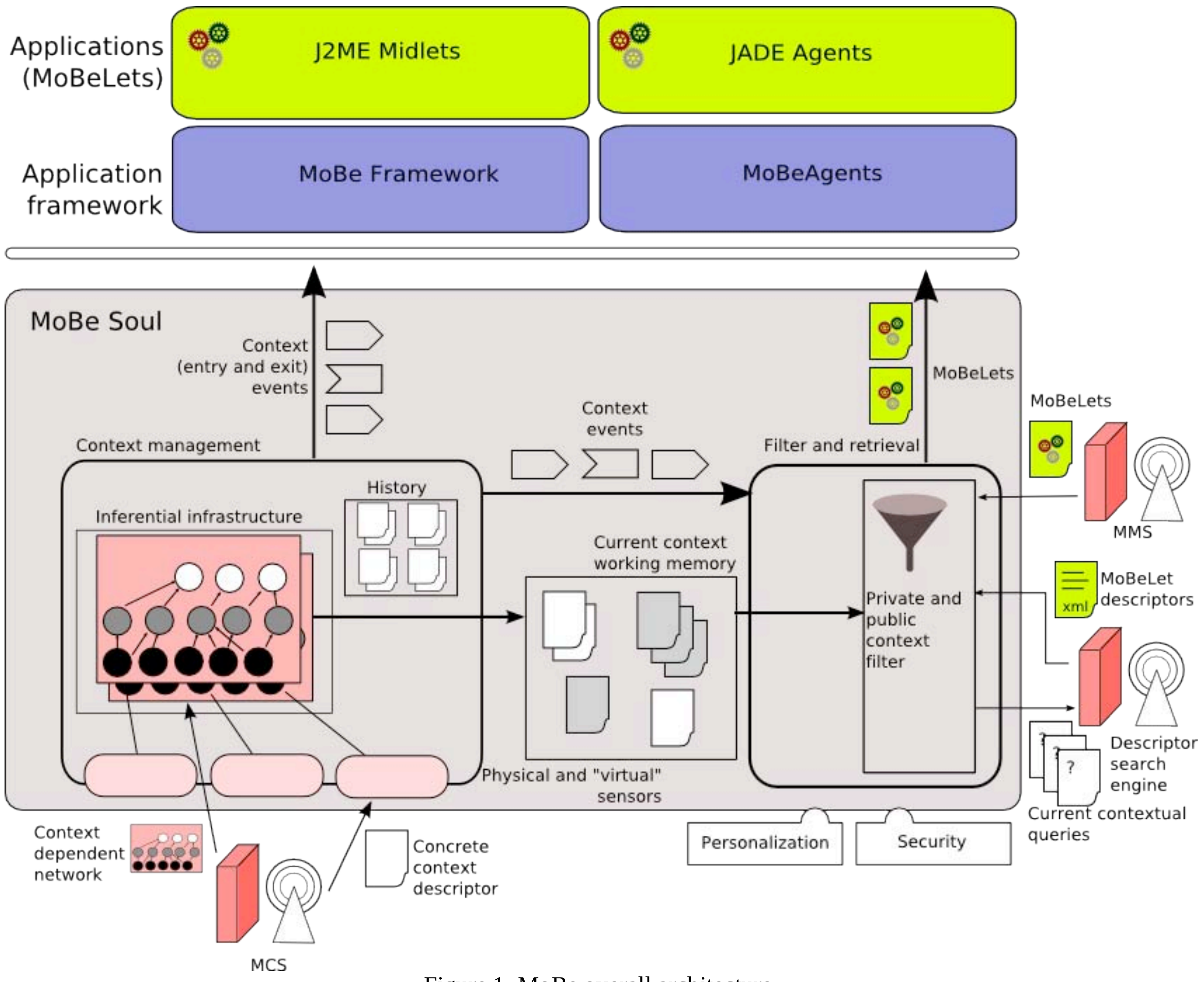


Figure 1. MoBe overall architecture.

The applications, called MoBeLets, are stored on the MoBeLet Server and migrate, transparently to the user, on his mobile device. Each MoBeLet is described by a MoBeLet descriptor , that holds the most important information related to the application, in order to make the retrieval easier (e.g., the type of task carried out) and to decide whether it is suitable or not for the mobile device of the user (e.g., information about the minimal CPU/memory requirements or the kind of needed peripherals/communication media). The MoBeLet descriptor contains all the information relevant to the representation of an application. Descriptors of applications have already been used in several fields as they simplify the retrieval process, making an application more understandable for its use and reuse. As examples we can mention UDDI (Universal Description Discovery and Integration, http://uddi.xml.org/), an XML-based registry used to publish and discover web-services, and SourceForge (www.sourceforge.net), a source code repository where each project is described with a specific technical sheet.

Some ancillary servers exchange information with mobile devices in that environment. In particular the MCS (MoBe Context Server) exchanges information about contexts (and inferential networks), while the MMS (MoBe MoBeLet Server) provides MoBeLets' related information and the actual MoBeLets.

Since the MoBeSoul component is completely general and can be adapted to different implementations, two implementations have been developed and tested; they are based on: (i) MoBe framework, an ad-hoc J2ME middleware, (ii) MoBeAgents, the extensions of a Multi-Agent framework.

3. The Context-Aware Browser

With the emergence of AJaX technology on desktop web browsers, and on mobile devices browsers as well, it has been natural to think of adapting the previously described MoBe framework to a Web oriented paradigm. Among the aims, reducing the technological gap needed for using such a kind of system was one of the most crucial, as the web browser is now a daily use tool for most people, while mobile software applications like midlets are still somewhat unconventional. Moreover, with a web oriented approach, the currently existing web contents could be re-used (whereas MoBeLets have to be designed from scratch). Finally, we believe that also the developers, in these days, are more familiar with web applications.

Looking at such a web oriented MoBe, one can realize that it is a new kind of system, that we named Context-Aware Browser. In other terms, the main idea behind CAB is to empower a generic mobile device with a browser being able to automatically and dynamically load web pages, services and applications selected according to the current context the user is in. Despite the apparent simplicity of this approach, a more thorough definition has to take into account several features; whence, we can say that the Context-Aware Browser is best described by the sum of the following parts (Figure 2 shows a schematic representation):

- a web browser: CAB is able to interpret and render in a sound way (X)HTML code, to interpret client side (e.g., JavaScript) code, and to fully exploit AJaX implemented applications (moreover, a browser is the standard way of representing the web contents interface to the user);
- a software application for mobile devices: indeed, the latter are the main target of CAB (a device tied to a fixed location cannot fully exploit the context-awareness of our browser, since a fixed location entails a more static context);
- a context-aware application: CAB is able to automatically retrieve and constantly update the contextual information gathered from the surrounding environment (this feature allows CAB to provide contents varying in dependence of the current context of the user);
- a search engine: CAB is able to search both for "traditional" web pages and applications on the Web and for specifically tailored applications as we will see in the following;
- an application able to automatically load contents: the resources retrieved by the search engine are automatically filtered against user's preferences and several other parameters, in order to reduce the cognitive load imposed on the user by automatically selecting the most appropriate contents;
- an application providing web pages and applications: CAB is able to manage both static resources (e.g., plain (X)HTML pages) and dynamic ones requiring user interaction (e.g., web applications).

All the above mentioned parts are needed and equally important in the characterization of the Context-Aware Browser; as a consequence, if we left out even only one of those, we cannot speak of CAB, but of something significantly different and more limited.

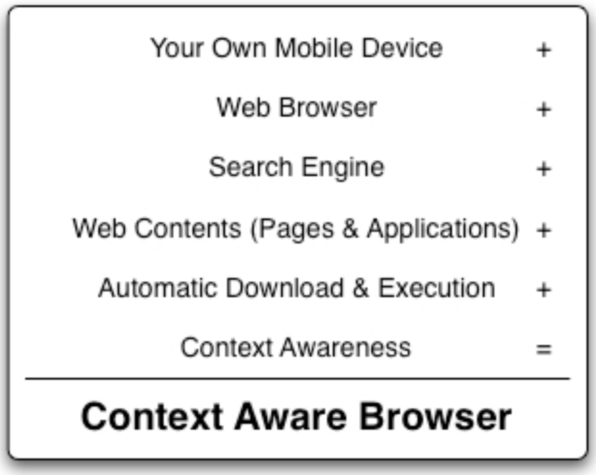


Figure 2. CAB definition.

4. CAB architecture

We provide some more details on CAB implementation. Figure 3 shows CAB overall architecture: it is easy to see some similarities with MoBe (see Figure 1), on both the overall approach and some specific components. Several differences exist, though, and are discussed in the following together with CAB main modules. Essentially, the CAB is based upon a three-tier architecture where the topmost layer (the Browser) manages the interaction with the user (showing web contents and dealing with user's input). The medium layer (the Connector) acts as a bridge between the Browser and the inner layer (the Core) which is responsible for all the internal mechanisms (sensors, inferential network, and filtering engine)

4.1. Context server and Sensors

Context server and sensors are the components responsible for collecting and sending to the CAB Core the contextual information gathered from the surrounding environment. More precisely, when the mobile device equipped with the CAB detects a so-called Context Server, it instantiates a suitable *Location-id* sensor that communicates with the server itself, downloading (in XML format) the current inferential network. This network represents (part of) the knowledge base of the Context manager and it is used to infer in a more detailed way the current user's context. After the local installation of the new inferential network, all the sensors specified by the latter are instantiated as well by the Context manager. Then, such sensors communicate with the current Context server in order to receive the contextual information (location, date/time, temperature, etc.) according to two different paradigms described below. Thus, all data are easily kept up-to-date.

If the user moves to a new environment, the inferential networks might change; whence, the Context manager verifies if the current set of sensors is still meaningful, possibly dropping those that are currently useless and closing the corresponding communication channels towards the Context server. This way, only the needed sensors and the related communication channels are maintained, in order to spare resources and to guarantee an efficient use of the Context Server and of the CAB itself.

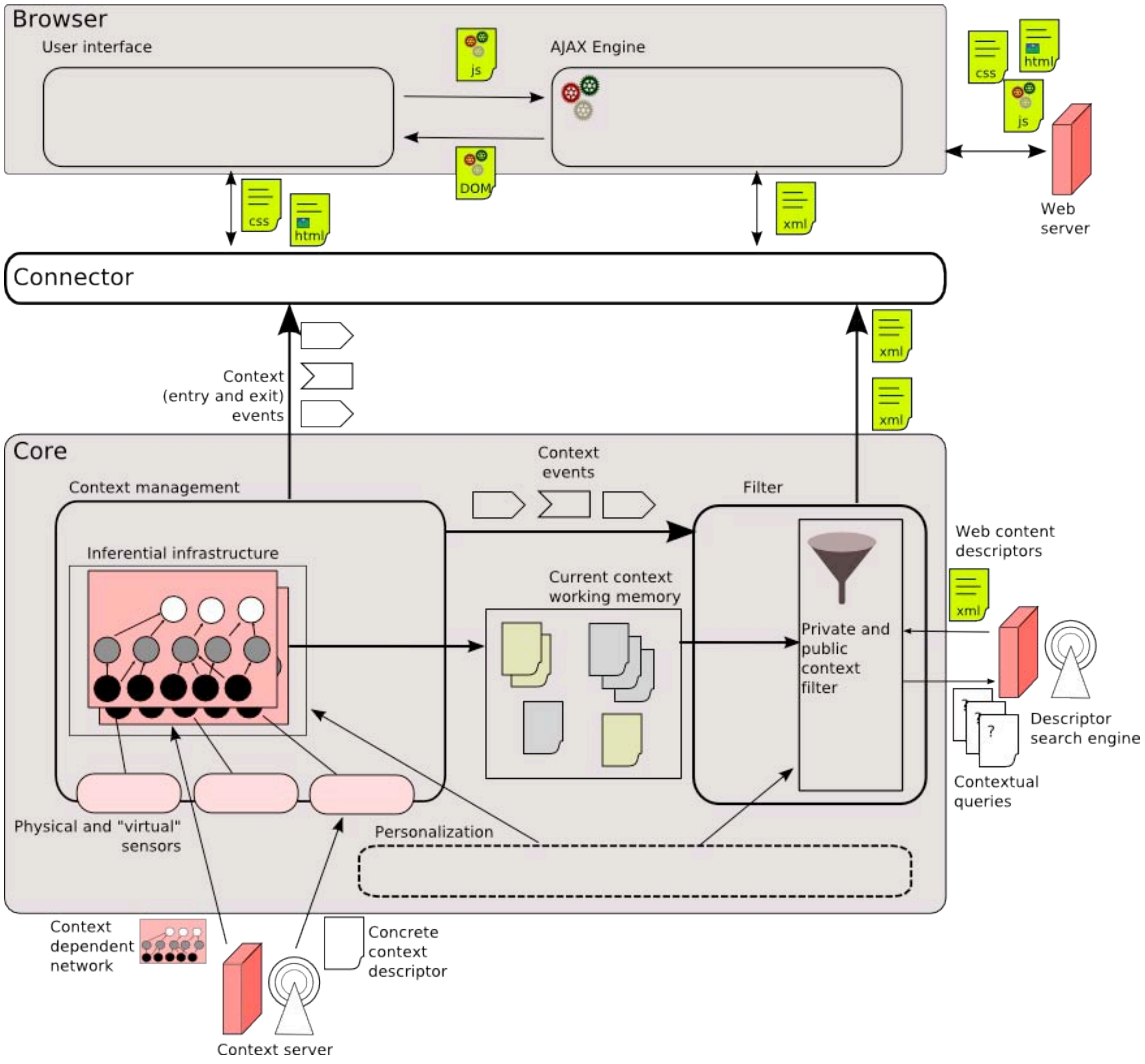


Figure 3. CAB overall architecture.

The sensors and the Context server can interact according to two different paradigms: either the formers can interrogate the server at fixed time intervals (pull mode) or the latter can provide the data to sensors as soon as they become available (push mode). The possibility to choose between such operating modes is an extremely useful feature, since pull mode allows to avoid an excessive overload of the Context server when considering contextual data characterized by fast or constant variations without providing significant information (e.g., temperature, humidity, etc.), whereas push mode is useful for data that do not change for long time intervals, but which provide extremely important information (e.g., fire sensors, alarms, etc.).

4.2. Context manager and Inferential system

The Context manager, with its internal inferential system, is the core component responsible for the synthesis of the current context, starting from the contextual information gathered by sensors. In particular, the architecture of this component is composed by the following sub-modules:

- Inferential engine: this part deals with the inference of the user's current context from the low-level sensors' information;
- Registry and Notifier: this component manages the list of all the modules registered with the Context manager, moreover it notifies them context changes (in particular entry and exit events and the update of the probability associated with a context);
- History: this is essentially the database holding the story and the statistics of past contexts.

As soon as the sensors receive contextual information, a notification is made to the Sensor manager. Then, the latter starts the Inferential engine in order to synthesize the current context from the data gathered by sensors. The inferred context is then stored in the Current Context Working Memory, but it is not sent directly to other CAB modules. Indeed, the Context manager uses the Registry and Notifier component to inform the Filter module (see next paragraph) which can autonomously decide whether it is worth requesting the new inferred context. Notice that the procedure just described is run only when new data is available at the sensors level and not on a timely basis. This policy aims at an efficient use of the limited resources of a typical mobile device, since the whole procedure of inferring a new context can be rather costly from a computational point of view.

We describe in more detail the context inference process, that exploits Artificial Intelligence techniques.

4.2.1. Inferring abstract contexts from concrete contexts

The user's current context is composed by an undefined number of contextual values. Each value is described by two elements: an unambiguous ID and a probability value. We split contextual values into two categories:

- *Concrete contexts*: they represent the information obtained by sensors. These contexts can be acquired from the surrounding environment through physical sensors (e.g., temperature sensor), or can be obtained by other software (e.g., calendar) through logical sensors. Some examples are: "temperature 20°C", "current_time 12:31","meeting_at 14:30", and so on.
- *Abstract contexts*: they represent everything that can be inferred from concrete contexts like, for example, "user_at_home", "user_is_shopping", etc.

Concrete and abstract contexts are the inferential system input and output, respectively. However, from a theoretical point of view, this difference is faded since the contexts cannot be unambiguously assigned to one or the other category: the context "temperature 90°C" can be a concrete context as it is obtained from a sensor, or it can be inferred by other contexts (e.g., "user_in_sauna"). The aim of the inferential system it to combine concrete contexts to determine abstract contexts and to combine abstract contexts to obtain new, more abstract contexts.

4.2.2. Two approaches for the inferential system

Two approaches have been taken into consideration to develop the inferential system: rule-based systems and Bayesian networks. As it is well known, a rule-based system [17] is a general mechanism for knowledge representation and management. Although rule-based systems are a relatively simple model, they can be naturally adapted to the context-aware research field. The left and right side of a rule can represent two contextual values and the rule suggests a connection between them: e.g., IF *user is in the bathroom* and *the humidity level is high* and *there is a continuous sound* THEN *user is having a shower*. In addition rule-based systems are not limited to propositional logic but they allow the use of first-order logic variables, a feature that greatly simplifies knowledge management.

Indeed, rule-based systems have already been used in the context-aware research field. Bacon and colleagues [2] describe a multimedia system based on user location, where contextual information is represented as facts in a rule-based system. Zhang [27] proposes a framework for allowing user to program his context-aware application: a user can visually create the rules that are then combined with sensors data to adapt the application to user context.

Even though the use of rule-based systems is reasonable in context-aware computing, it presents a remarkable limit: they manage definite knowledge, whereas almost everything related to contexts is characterized by uncertainty. For this reason other knowledge representation and reasoning mechanisms are needed to deal with uncertainty.

Bayesian networks [19] represent a model for handling probabilistic knowledge, and they can be easily adapted to the context-aware field as well. In such an adaptation, each node can represent a contextual value whereas the edges represent dependence relationships between different contexts, the probability distributions indicate the degree of confidence related to contexts.

Bayesian networks have been used in the context-aware field mainly as system to determine the uncertainty of contexts. In [4] a Bayesian network is used to measure the efficiency of contexts derivation from rough sensors data while in [16] Bayesian networks are used to classify contexts related to a user's everyday activities.

Because of the uncertainty management, Bayesian networks are, in general, more suitable than rule-based systems for contextual inferences. On the other hand, rule-based systems allow the use of variables, which can limit the dimension of the inferential mechanism. For example instead of managing all temperature values we can simplify them to the three abstract values temperature high, low, and normal and use rules generalized with variables to map the real sensed temperature value on the three abstract ones.

In our opinion, both the approaches are important and needed for a complete and functional system as we will describe in the following paragraph.

4.2.3. The inferential infrastructure

We propose a two stage inferential mechanism, where both rules and Bayesian networks are used. We define the combination of rules and Bayesian network as the inferential infrastructure. The input to the inferential infrastructure is represented by concrete contexts. Concrete contexts are processed by a rule-based system in order to simplify the information and map them on the starting node of the Bayesian network, that represents the second and main stage of the inferential infrastructure, where abstract contexts are transformed into concrete ones. The combination of concrete and inferred abstract contexts is the user's current context (Figure 4).

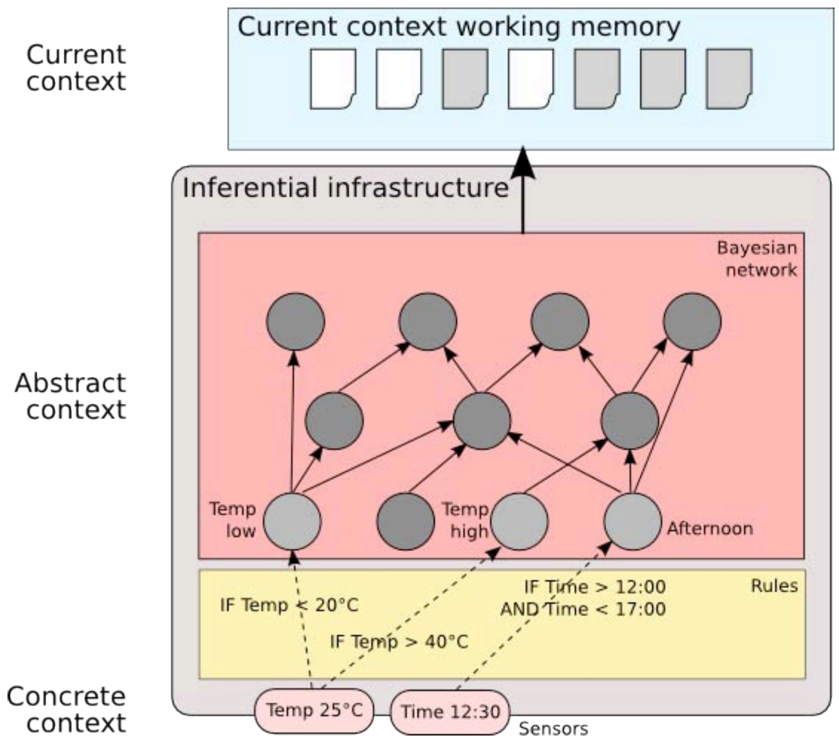


Figure 4. The inferential process.

The following is a simple example. A concrete context is "sound_level 60dB". Through the first rule-based stage this information is mapped into the Bayesian network starting nodes. In this way we simplify the contextual information: instead of managing all the possible sound values we create an abstraction (i.e., "sound low", "sound high"). Without the rules (and the variables), if the management of all the single (discretized) sound values was needed, we should have introduced in the Bayesian network a node for each sound value. Then the Bayesian network starts from this point to infer abstract contexts like "user is listening to music".

4.2.4. Critical issues

We can identify four main limits of the basic inferential infrastructure we have presented so far.

The first limit concerns the size of the inferential network. Since we do not want to limit the inferences only on an a priori defined set of contextual dimensions, the network, in principle, should be omniscient and include information and inferential mechanisms for every possible situation. The possible contextual values that should be managed in such a case are innumerable and, as they have to be built into the inferential network, this would lead to a universal network, which would be unmanageable in practice.

The solution we adopt consists in splitting the inferential network into several subnetworks, on the basis of a single dimension of concrete context. For example, a hypothetical universal network can be split using location as an index: in this way we obtain several networks, each of them relevant to a particular location (e.g., one for home, office, car, etc.). These specialized networks have a smaller size, therefore they are more manageable. Furthermore, if they are still too complex they can be partitioned further.

Single inferential networks are acquired automatically on the basis of the value of the contextual dimension used for the partitioning. The mobile device will receive one or more networks from a remote server and will combine them to infer more precisely the user abstract context. For example, when the user enters her home, if the user location is selected as the partitioning dimension, the device will obtain the inferential network specialized on the contexts related to the home.

Location is just one of the dimensions that can be used to partition the inferential network; other dimensions could be time (e.g., a set of applications related to work is downloaded during working hours or according to an agenda) or a combination of location and time.

The second problem concerns the potential mismatch between concepts in the inferential networks and the concrete contexts (e.g., a network could execute inferences starting from a contextual value expressed as "temperature in Celsius degrees" while the temperature sensor on the device manages information in Fahrenheit degrees). These incoherences would of course lead to erroneous contexts inference. To avoid that, we define an ontology of concrete contexts, which provides a shared model for situations, sensors, provided values, and relationships among them. In this ontology, the sensors are categorized on the basis of the contextual information they provide. Similarly, in the development of the inferential network, its concrete nodes (or the starting ones) must also be categorized in the ontology. In this way the Inferential engine is able to obtain the information needed by the inferential network through the appropriate sensors.

The third problem is a consequence of the previous two and it concerns the agreement on contexts representations. Indeed, it is possible to represent the same context in many different ways, leading to a troublesome matching between the context descriptions in the inferential network and those in the specific application descriptors.

Again, a shared ontology allows to link related contexts, by explicitly associating them through the common concepts they share, even when they are expressed in different ways.

Therefore, we extended the ontology previously suggested so that all contextual values (not only the concrete ones) are classified in it. In this way we formalize the contexts definitions and representations, regulating their use. Our ontology is based on WordNet (http://wordnet.princeton.edu), which is both a terminology and a constitutive ontology that supports multilanguages and implements a set of semantic relations between concepts (e.g., synonymy, part-set, etc.). WordNet must be extended since it only provides a basic terminology and a first group of semantic relations. However, the definition of relations specific to the contextual aspects is required to obtain an ontology of contexts. To this aim we devise a domain ontology, which includes concepts and basic relations for sensors and contexts, and can be extended by more specific ontologies. For example, to define the concepts related to "house" we both refer to the basic relations and extend the generic concepts from a previously defined "building" ontology.

A fourth problem concerns subjectivity: since a "one-size-fits-all" approach will most likely be inadequate for users with different needs, we have introduced some other functionalities that allow users to personalize their inferential system. A user can associate to each context of interest in the network two values called privacy and importance. These values refer to two thresholds managed by the user. The first one refers to the sensibility of the contextual information; contexts with privacy value higher than the privacy threshold will not be diffused outside the mobile device. The importance value acts somehow in the opposite direction: it allows a contexts to be made public to remote servers even if its probability is lower than the filtering threshold. Moreover, the user has the possibility to modify and personalize the network by adding or removing nodes (contexts) and changing the probability relations between contexts. This feature is crucial in order to make the network more suitable to model the user's current situation.

Finally, although the inferential system main goal is to infer abstract contexts from the concrete contexts, the prototype could also execute the opposite inference: the user can define manually his abstract contexts (e.g. "watching TV") and from this the other contexts probability will be determined.

4.3. Filter and Descriptor search engine

The Filter module and the Descriptor search engine allow to search for and filter the most suitable contents according to the inferred context. The contextual information, obtained from the Context manager, is represented through a Context Descriptor that holds the most important information related to the user's current context. The Context Descriptor structure is shown in Figure 5.

Name: a context unique identifier.
Description: composed by
    description base: the explicit description of the context, in free text;
    description domain: mainly related to the user location: for example when the user enters the university the description domain of the university is sent to his mobile device.
Location: user's spatial position. It contains both low level data, like GPS coordinates, and semantic enriched information, like "office, 2nd floor".
Time: date and time referred to the current context. It can contain both numerical values like "11:00pm" and temporal labels like "morning".
Activity: composed by
    activity: the activity description;
    activity-type: the activity categorization.
Posture: the user posture, like "lying down", "standing", etc. This information is important in connection with usability concepts: on the basis of the posture a user can manage different cognitive loads and so the application should be different. For example an application to take notes with a lot of function could be useful if the user is sitting, but could be inconvenient if the user is walking.
Probability: likelihood of the context (needed because some contexts are inferred with uncertainty).
Privacy: a numeric value (predefined by default and modifiable by the user) modeling the sensibility of the contextual information. Contexts with privacy value higher than the privacy threshold defined by the user will not be sent outside the mobile device.
Importance: a numeric value (predefined by default and modifiable by the user) that allows a context to be sent to remote servers even if its probability is lower than the filtering threshold. The idea is that important contexts, like for example "car is going to break", should be used in the retrieval process even if they have a low probability value.

Figure 5. The Context Descriptor.

The first operation carried out by the Filter module (see Figure 6) is to determine if the new inferred context is different from the previous one (to avoid useless search operations if the two most recently inferred contexts are indeed equal). If the inferred context is a new one, the Public context filter submodule builds a query on the basis of the public part of the Context descriptor, and sends it to the Descriptor search engine on the server. The search engine carries out the search operation, and sends back to the CAB Filter module the descriptors corresponding to the selected contents. A descriptor is essentially an XML file describing the fundamental features (e.g., URI, size, MIME type, etc.) of the corresponding web content: it is used by the Private context filter component in order to select in a more precise way the contents to be downloaded. For instance, the user could specify to discard all the web resources speaking of a certain topic, or using too many resources, or belonging to a specific domain, etc. The descriptors are created automatically starting from the web content.

Let us remark that the CAB allows to specify which filtering information is public (and may be used to formulate the query to be sent to the Descriptor search engine) and which one is private (and may be used only in "internal" filtering activities); in this way, a reasonable privacy level is reached: only the public part is sent out of the device, whereas the private part is kept secure. The alternative of sending a more detailed query to the Descriptor search engine would be simpler but less privacy-aware and more open to malicious eavesdroppers attacks.

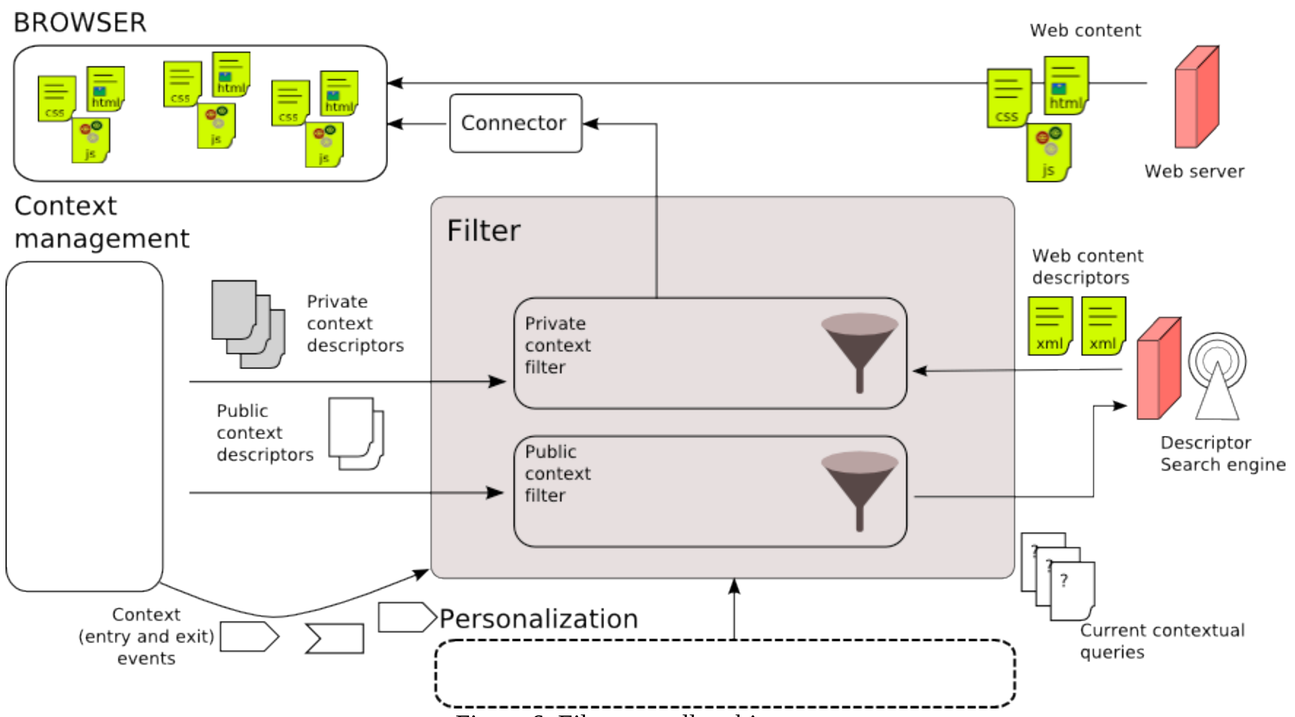


Figure 6. Filter overall architecture.

4.4. Connector

The Connector is an infrastructure component in charge of sending the web contents descriptors to the browser running on the mobile device. This component is implemented as a J2ME local web server, being able to open HTTP connections and to manage browser's requests. It is divided into two sub-modules:

- HTTP manager: it opens a permanent HTTP connection used to communicate with the mobile device's browser.
- Content manager: it manages the communication of the web contents' descriptors, providing a notification to the HTTP manager, as soon as such descriptors are made available by the Filter module (see the previous paragraph).

4.5. Browser

The Browser CAB module manages the presentation of the web contents to the user and it also automatically starts the most relevant ones. Such features are implemented in two sub-modules: the AJaX engine and the User interface.

4.5.1. AJaX engine

This sub-module allows the CAB to receive and manage the contents' descriptors sent by the Connector, making use of AJaX in order to implement all the features offered to the user in a seamless and efficient way. Again, the architecture of this layer is subdivided in several components (implemented in JavaScript):

- Streaming: it opens and maintains the connection with the Connector module, in order to receive in real-time the contents descriptors;
- XML Parser: it parses the descriptors XML files in order to gather all the relevant information to display in the browser;
- Tab manager: it manages the windows (tabs) displayed in the browser, allowing opening, closing and exchanging messages among them;
- Cookie manager: the application status is stored by means of cookies, managed by this sub-module.

The AJaX engine begins its work immediately following the start of the application, loading the above mentioned JavaScript sub-modules. This operation is triggered by a HTTP request to the Connector, which retrieves from the file system the needed HTML and JavaScript files, sending them to the browser. Then, the Streaming Module instantiates the XMLHttpRequest object, opening an asynchronous HTTP connection with the Connector itself. Thus, the module can provide automatically in a real-time and transparent way the suitable contextual web contents, without having to reload whole pages each time a changing context event happens. Such web contents are provided as XML descriptors; whence, the XML Parser module has to parse such files in order to present the relative information in a suitable and user-friendly way on the browser's screen. In particular, the web contents that have to be immediately presented to the user are sent to the Tab manager. The latter deals with both primitive operations (like opening and closing tabs) and advanced ones (like providing the means to exchange messages between different tabs in complex applications). Finally, the Cookie manager implements the procedures of storing the application status in a safe place and of retrieving it when needed: this is crucial when updating the web page by means of the AJaX technology, without loosing the system status and the data structures already initialized.

4.5.2. User interface

This module provides to the user the main interfaces for using and manipulating the CAB. Essentially they are rendered through the browser, which provides two kind of Tabs:

- Main Tab: it constitutes the main CAB interface, presenting a list of the web contents previously provided by the Connector: the list is dynamic in the sense that context-change events cause its update (the user chooses web contents interacting with such a list).
- Content Tab: this kind of tab renders the pages/web applications downloaded from the relative web servers; hence, their appearance (which is the final result of the use of several standards like, e.g., XHTML, CSS, DOM, JavaScript, AJaX, etc.) may substantially change, when moving from one web content to another one.

Regarding web contents, their nature can be twofold:

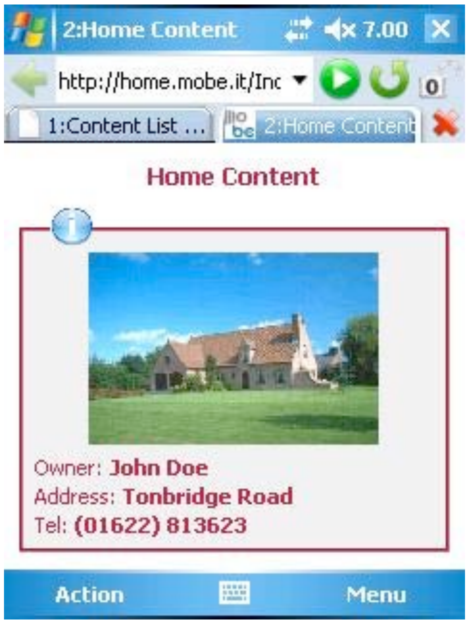
- They can be contextual web pages (either static or dynamic), either ad-hoc developed for CAB or simply preexistent Internet resources; their typical use is to provide some information relative to the current context the user is in.

- They can be more complex contextual web applications (either ad-hoc developed for CAB or preexistent ones); their typical use is to provide non-trivial features to the user, making use of advanced web technologies like, e.g., AJaX or Web Services.

However, it is worth noticing that web contents are downloaded as "classical" web server resources, i.e., there is no need of new, ad-hoc web servers: contextual applications and resources can be provided smoothly by the existing content providers used for the traditional paradigm of web navigation.

5. A usage scenario

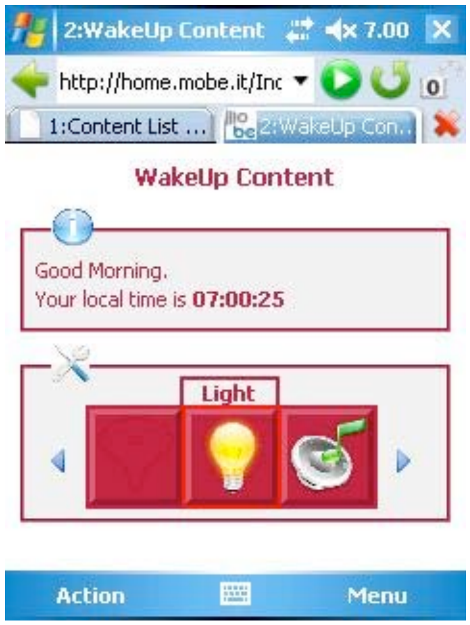
In order to show how the CAB can be adopted in a realistic environment, we propose a scenario describing a typical day of a person using a device equipped with the CAB. We present some snapshots of a working prototype implemented and deployed in cooperation with a company working on domotics household appliances. Figure 7 shows the generic Home Content, downloaded when the user is at home. Let us imagine that it is seven o'clock in the morning and that the user just waked up: this context causes the download and execution of a WakeUp Content (Figure 8) allowing the user to manage the lights (Figure 9), the television and stereo sets, the heater, etc. The home context features several more applications; the full list is shown in (Figure 10).



Home Content

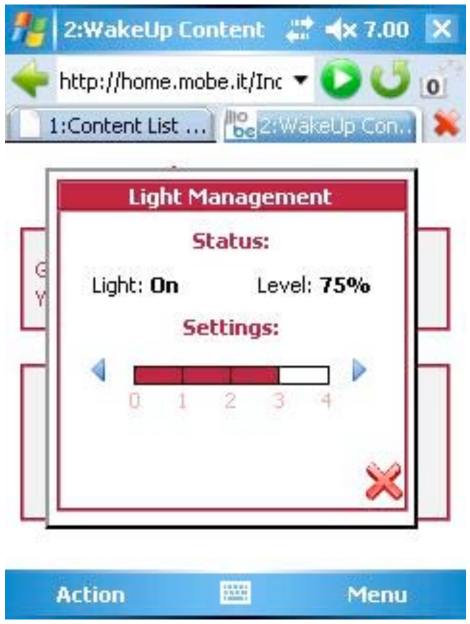
Owner: John Doe
Address: Tonbridge Road
Tel: (01622) 813623

Figure 7. CAB showing the generic content for the "home" context.



Good Morning.
Your local time is **07:00:25**

Figure 8. The WakeUp application.

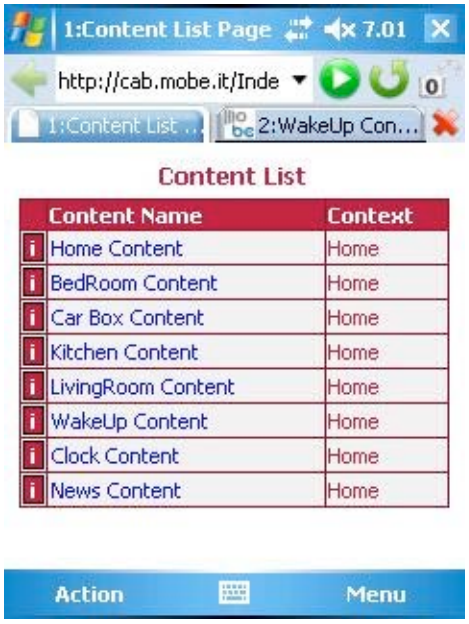


Status:

Light: **On** Level: **75%**

Settings:

Figure 9. Application for light management.

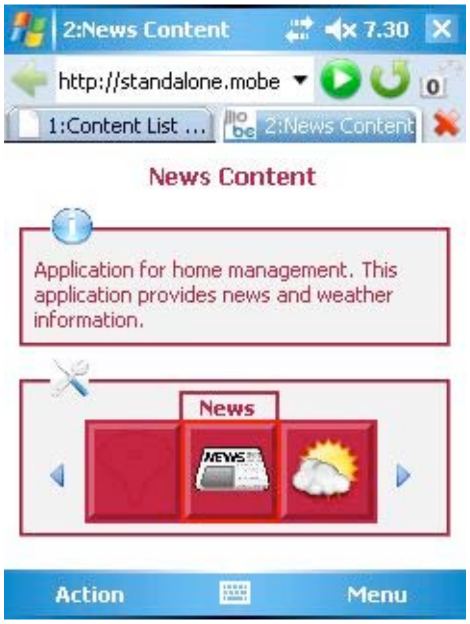


Content List

Content Name	Context
Home Content	Home
BedRoom Content	Home
Car Box Content	Home
Kitchen Content	Home
LivingRoom Content	Home
WakeUp Content	Home
Clock Content	Home
News Content	Home

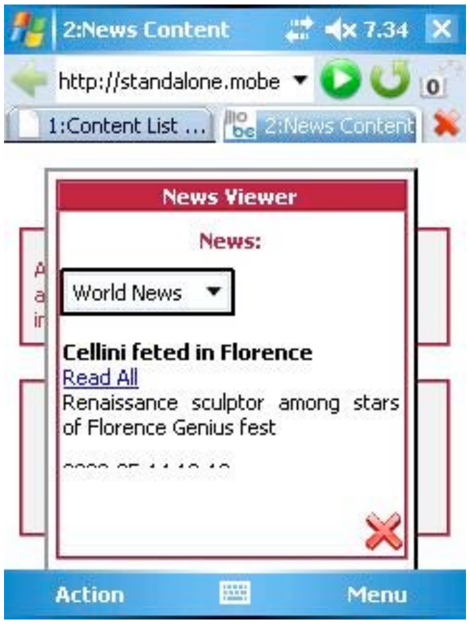
Figure 10. The Contents list.

When the user is having breakfast, e.g., at 7.30 a.m., the CAB provides another functionality to support the user's daily routine: it presents, on the basis of user's preferences, a web content with the news of the day and the weather forecast (News Content in Figures 11 and 12). The news of the day, retrieved by RSS feeds, are a preview (title and brief content) of the complete article, that can be read following a specific "Read All" link.



Application for home management. This application provides news and weather information.

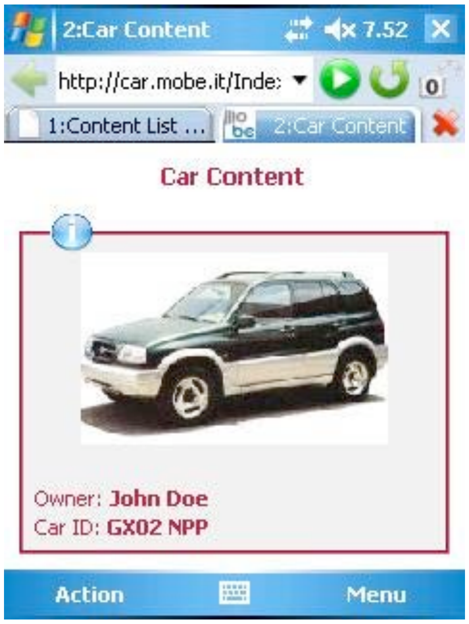
Figure 11. News Content.



Cellini feted in Florence
Read All
Renaissance sculptor among stars of Florence Genius fest

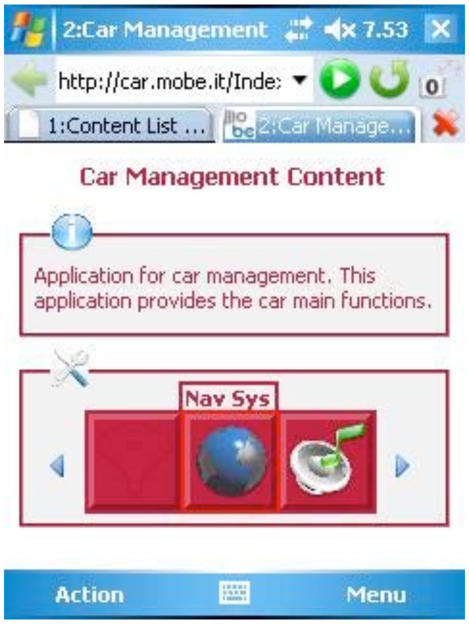
Figure 12. A specific news item.

Our example goes on by the user entering her own car to drive to work. The CAB detects the new contextual information from the external environment (i.e., the presence of the car) and downloads new specific web contents: Car Content (Figure 13) and Car Management Content (Figure 14). These web contents provide information about the car and allow the user to personalize and adapt the car settings (Hi-Fi, navigation system, seat, etc.) to her desire.



Owner: **John Doe**
Car ID: **GX02 NPP**

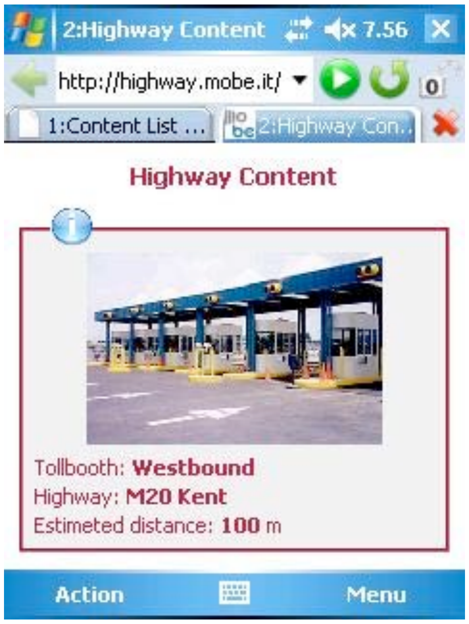
Figure 13. Car Content for the car context.



Application for car management. This application provides the car main functions.

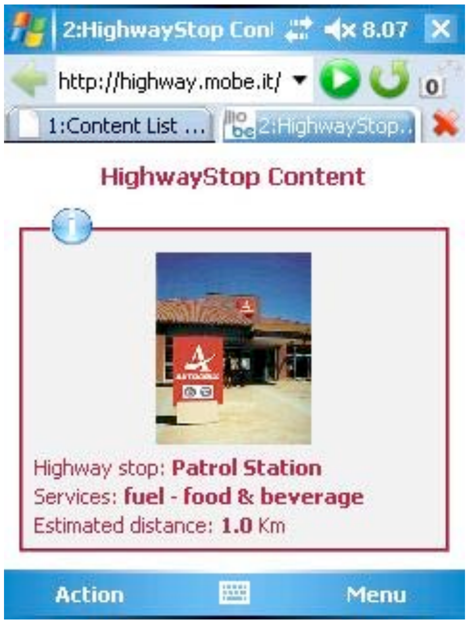
Figure 14. Application for car management; selection of the navigation system.

The user drives into a highway; when she is near the tollbooth, the CAB figures out the new context and retrieves the relevant web content (Highway Content, see Figure 15) providing information about the specific highway route. While driving, the CAB detects the presence of a highway stop, i.e. a petrol station with food and beverage services, and retrieves a specific web content (HighwayStop Content, see Figure 16), which informs the user about the estimated distance and the services available.



Tollbooth: **Westbound**
Highway: **M20 Kent**
Estimeted distance: **100** m

Figure 15. The user drives into the highway.



Highway stop: **Patrol Station**
Services: **fuel - food & beverage**
Estimated distance: **1.0** Km

Figure 16. CAB informs about the next stop.

In this example just a tiny subset of situations have been taken into account. Several further examples of usage scenarios (office, cinema, airport, etc.) can be imagined, thus confirming the adaptability of the CAB and the countless possibilities that it offers to its users.

6. Evaluation

In order to continue CAB development in a justified, informed, and effective way, it is important to evaluate its components, and to understand the effectiveness of the overall content filtering process. Since the CAB can be seen as a context-aware retrieval system, we can take inspiration from evaluation approaches in the information retrieval field.

6.1 MoBe evaluation

In the MoBe project we performed a preliminary evaluation, with particular emphasis on the MoBeLet filtering and retrieval process. We developed a TREC-like benchmark (http://trec.nist.gov/), named MREC (MoBe Retrieval Evaluation Collection), in which the collection is made up by a set of application descriptors and the topics are made up by context descriptors. In particular, MREC is constituted by three components:

- A collection of documents. The documents collection in our benchmark consists of 448 MoBeLet descriptors, either created on the basis of applications descriptions from online software depots, like softonic.com, tucows.com, download.com, getjar.com, etc., or created ad-hoc by ourselves.
- The statements of information needs. In the MoBe setting, the information needs are strongly related to the user's current context. MREC includes 30 context descriptors, which represent different examples of user's contexts in different domain like restaurant, university, theater, car, stadium, trade fair, etc.
- A set of relevance judgments. The relevance judgments have been made by a unique judge using a binary relevance scale: a MoBeLet descriptor is either relevant or nonrelevant for a given context descriptor.

We then used the benchmark to evaluate the effectiveness of different descriptors components, and of structured and unstructured (i.e., free text) data.

The evaluation aimed at comparing the filtering and retrieval process in terms of different approaches for context descriptors (i.e., query formulation) and for MoBeLet descriptors (i.e., indexing of the documents collection). We measured effectiveness by means of three standard IR metrics: Precision, Recall, and MAP (Mean Average Precision). In order to verify the reliability of our benchmark, we also performed an additional experiment. We involved two more judges and measured interjudge agreement by means of F-Measure (a standard IR metric).

As result, the subjectivity of the main judge does not seem an issue for our MREC benchmark, that seems reliable at least to a reasonable extent. Concerning MoBe, we have some evidence on the effectiveness of single and aggregate fields of both context and MoBeLet descriptors. For instance, we have had an indication on how to improve the matching process between location and time fields in the two descriptors, so that it seems reasonable to relax the current exact matching in these two fields. Similarly, a refinement of the activity field seems necessary in order to better use the information it conveys. Also, we understood that structured and unstructured fields have complementary effects, and that a mixed approach seems more effective. Therefore, the introduction of ontologies for both the context and the application descriptors seems a useful method to enhance data interoperability and retrieval effectiveness.

6.2 Evaluation issues

Although the proposed benchmark and the obtained results are rather limited, they have been useful to the development and improvement of the proposed system. Since this experience has been positive, we intend to repeat the evaluation, adapting it to the CAB features and refining it on the basis of the results of MoBe evaluation.

Evaluation has been and still is an issue of paramount importance in information retrieval. Depending on resources and aims, different evaluation metrics are chosen and different evaluation approaches are adopted: user studies and benchmarks; lightweight and heavyweight [12]; etc. The evaluation of context-aware retrieval systems in general, and of CAB in particular, is even more complex, for several reasons: the more complex scenario, the highly interactive nature of context-aware retrieval, the lack of previous experiences, benchmarks, and methodologies, etc.

Moreover, the evaluation of an extremely novel and highly interactive context-aware retrieval application like CAB puts us in a paradoxical situation: on the one hand, users and tasks seem needed, thus a user study based on some task analysis seem mandatory. On the other hand, user studies would be far too expensive at an early development stage, and there are no realistic users and tasks until the system is launched on a large scale, and if a "wrong" system is launched to start the wave and have some users, then the system will fail, and there will be no users at all. Finally, to fully evaluate the approach, some contextual data should be available on the Web, which is not yet the case.

Pushing this line of reasoning to its extreme consequences, one might wonder if evaluation somehow hinders development. Indeed, when a paradigm shift is going to happen, it is quite unlikely that current evaluation methodologies can cope with the new scenario: they will evaluate the revolutionary system on the basis of the current evaluation techniques, that could be not appropriate for such a system and could lead to negative results. To give some examples: were SMS (the short text messages) evaluated? Was the Web evaluated? The obvious answer to these questions is no: these services have been used in creative and unforeseen ways by the users-far beyond the limits that evaluation methodologies well established at that time. This is an extremist's position, to be taken with caution, since evaluation is fundamental for new systems; however, we should perhaps remember that when we have a hammer in our hand, everything looks like a nail, and evaluation might be the hammer.

Considering these issues and difficulties, and the still initial stage of development, it is not sensible to invest too many resources to get detailed results; even a rough and quick evaluation is useful.

7. Conclusions and future development

In this paper we have presented the Context-Aware Browser, a new approach to context-aware Web content perusal by means of mobile devices. The novelty of the proposed approach is twofold. First of all we take into account the information provided by the surrounding environment in order to carry out a more refined search of web contents. Secondly, information is not only automatically pushed towards the mobile device and then filtered: rather, the CAB approach involves a two stage retrieval plus filtering process, that minimizes bandwidth and privacy and security risks. As an overall outcome, the effort required on behalf of the user is minimized as well.

Often, one of the major criticisms against the effectiveness of novel information seeking approaches is that they are perceived as too much complicated by "casual" users, i.e., users not acquainted with the use of web search engines. We believe that CAB does not suffer from this point of view, since, as already mentioned, it minimizes the interaction with the user automatizing the retrieving and management of web contents.

As future work, since many data are exchanged between the CAB and the remote servers, we plan to provide authentication methods in order to certify web contents through digital signatures. Moreover, we believe that the CAB approach could improve the information seeking activity on all mobile devices, we are currently porting the CAB to the iPhone, Android, and Symbian architectures, in order to enlarge as much as possible the number of hardware platforms supporting our approach. The porting task turns out to be rather simple, since most of the code can be straightforwardly reused thanks to the widespread availability of J2ME implementations and of AJaX-enabled web browsers on different platforms. Of course, it would be also interesting to implement a full evaluation in a controlled environment, in order to measure users satisfaction. In this respect, we are currently deploying mobile guides, based on the CAB and therefore accessible on one own's device, for a museum and an archaeological site.

Finally, we aim at integrating CAB into the social networks and Web 2.0 worlds, since it would allow mobile users to produce, in an easy way, personalized and contextualized web contents, comments, evaluations, etc.

References

[1] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith and P. Steggles. Towards a Better Understanding of Context and Context-Awareness. HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pp. 304–307, Springer-Verlag, 1999.

[2] J. Bacon and J. Bates and D. Halls. Location-oriented multimedia. IEEE Personal Communications, 4(5):48–57, 1997.

[3] M. Bazire and P. Brezillon. Understanding context before using it. Proceedings of CONTEXT 2005, pp. 29–40, Springer-Verlag, 2005.

[4] G. Biegel and V. Cahill. A framework for developing mobile, context-aware applications. Proceedings of Second IEEE International Conference on Pervasive Computing and Communication, PerCom 2004, 2004.

[5] P. J. Brown. The stick-e document: a framework for creating context-aware applications. In Proceedings of EP'96, Palo Alto, pages 259-272, January 1996.

[6] P.J. Brown and J.D. Bovey and C. Xian. Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications, 4(5), pp. 58–64, 1997.

[7] P. J. Brown and G. J. F. Jones. Context-aware retrieval: Exploring a new environment for information retrieval and information filtering. Personal Ubiquitous Computing, 5(4):253–263, 2001.

[8] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Techreport TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[9] P. Coppola, V. Della Mea, L. Di Gaspero, S. Mizzaro, I. Scagnetto, A. Selva, L. Vassena and P. Zandegiacomo Riziò. MoBe: A Framework for Context-Aware Mobile Applications. Proceedings of Workshop on Context Awareness for Proactive Systems CAPS 2005, pp. 55–65, June 2005.

[10] A. K. Dey and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction Journal, 16:97-166, 2001.

[11] A. K. Dey and G. D. Abowd. Towards a Better Understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing, June 1999.

[12] D. J. Harper and D. G. Hendry. Evaluation light. In Proceedings of the second MIRA workshop, pages 53–56, 1997. Technical Report TR-1997-2, Department of Computing Science, University of Glasgow, Glasgow. http://www.dcs.gla.ac.uk/mira/workshops/padua_procs/.

[13] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, and J. Altmann. Context-Awareness on Mobile Devices – the Hydrogen Approach. 2002.

[14] G. J. F. Jones and P. J. Brown. Context-aware retrieval for ubiquitous computing environments. In Mobile HCI Workshop on Mobile and Ubiquitous Information Access, pages 227–243, 2003.

[15] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, and B. Serra. People, places, things: Web presence for the real world, February 2000.

[16] P. Korpipaa, M. Koskinen, J. Peltola, S. Makela and T. Seppanen. Bayesian approach to sensor-based context awareness. Personal Ubiquitous Computing, 7(2):113–124, 2003.

[17] A. Newell and H.A. Simon. Computer Augmentation of Human Reasoning. Spartan Books, Washington DC, USA, 1965.

[18] J. Pascoe. Adding generic contextual capabilities to wearable computers. In Proceedings of the Second International Symposium on Wearable Computers, Pittsburgh, Pennsylvania, October 1998. IEEE Computer Society Press.

[19] J. Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers, Inc., 1988.

[20] B. J. Rhodes. Just-In-Time Information Retrieval. PhD thesis, MIT Media Laboratory, Cambridge, MA, May 2000.

[21] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 85-90, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[22] B. Schilit and M. Theimer. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5), pp. 22–32, 1994.

[23] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, W. V. de Velde. Advanced interaction in context. In Proceedings of first International Symposium on Handeld and Ubiquitous Computing (HUC'99), pp. 89-101, Sept. 1999, Springer-Verlag

[24] A. Schmidt, M. Beigl, H. W. Gellersen. There is more to context than location. Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC98), Rostock, Germany, November 1998. http://citeseer.ist.psu.edu/schmidt98there.html

[25] P. P.L. Siu, N. Belaramani, C.L. Wang and F.C.M. Lau. Context-Aware State Management for Ubiquitous Application. Embedded and Ubiquitous Computing. In EUC, pages 776–785, 2004.

[26] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan. A diary study of mobile information needs. In CHI 2008 Proceedings, pages 433–442, Florence, Italy, Apr. 2008.

[27] T. Zhang. An architecture for building customizable context-aware applications by end-users. In Proceedings on Second International Conference, PERVASIVE 2004, 2004.

[28] D. West, T. Apted and A. Quigley. A context inference and multimodal approach to mobile information access AIMS 2004, Artificial Intelligence in Mobile Systems 2004 (in conjunction with UbiComp 2004), Sept 7, Nottingham, England.