A Simple Model and Infrastructure for Context-aware Browsing of the World

Gabriella Castelli, Alberto Rosi, Marco Mamei, Franco Zambonelli

DISMI – Università di Modena e Reggio Emilia – Via Amendola 2 – Reggio Emilia – ITALY
{name.surname}@unimore.it

Abstract

The imminent mass deployment of pervasive computing technologies such as sensor networks and RFID tags, together with the increasing participation of the Web community in feeding geo-located information within tools such as Google Earth, will soon make available an incredible amount of information about the physical and social worlds and their processes. This opens up the possibility of exploiting all such information for the provisioning of pervasive context-aware services for "browsing the world", i.e., for facilitating users in gathering information about the world, interacting with it, and understanding it. However, for this to occur, proper models and infrastructures must be developed. In this paper we propose a simple model for the representation of contextual information, the design and implementation of a general infrastructure for browsing the world, as well as some exemplar services we have implemented over it.

Keywords: *Context-awareness, Location-dependent Services, Middleware, Sensor Networks, RFID Tags.*

1. Introduction

Two apparently disjoint trends motivate this work. On the one hand, the imminent mass diffusion of pervasive computing technologies such as sensor networks [ChoK03] and RFID tags [Wan06] will soon make available an incredible amount of real-time information about the physical world, its processes, and its objects. On the other hand, the dramatic success of participatory Web tools (aka Web 2.0 technologies) is feeding the Web with information of any kind about any topic. In particular, mapping tools such as Google Earth and Google Maps get continuously enriched by geo-located information coming from very diverse social communities and related to a variety of facts and events situated in the world [But06].

Overall, both the above trends contribute to accumulate information that can be potentially used to build real-time and historical models of a number of facts and processes happening in the world. More pragmatically, the possibility of acquiring detailed digital information about the surrounding context opens up the possibility of exploiting all such information for "*browsing the world*" [Cas06]. The concept of browsing the world considers that, by properly integrating information about the surrounding world coming from both pervasive devices and form the Web, it will be possible for users to gather contextualized relevant information, and for services to effectively support user activities related to interacting with the physical world in a context-aware way.

However, considering that the amount of available information from a variety of sources could become overwhelming, its effective exploitation by users and services calls for proper models to represent such data in an expressive yet simple-to-be-manipulated way, and for proper software infrastructure to organize and provide access to it. Accordingly, the contribution of this paper is twofold.

First, we propose a simple model to represent contextual information about the physical world, for the use of both users' querying activities and context-aware services. The model, which we call "W4", is based on the consideration that most information about the world can be simply represented in terms of four "W"s – *Who, What, Where, When* – and that such a representation enables for very expressive, and flexible data usages.

Second, we describe the design and implementation of a general middleware infrastructure for browsing the world, facilitating the development and supporting the activities of general-purpose context-aware pervasive services. The infrastructure supports PDAs and laptops access to information coming from both pervasive devices and the Web, provides for representation and organization of data in W4 terms, makes available a Java interface for users' queries and for services access to

such data, and it is integrated with both Google Earth / Google Maps for the sake of effective user interfacing.

The remainder of this paper is organized as follows. Section 2 better details the general scenario of browsing the world and the challenges it implies. Section 3 presents the W4 model. Section 4 details the implemented software infrastructure. Section 5 presents some services we have implemented on top of our system. Section 6 discusses related work in the area. Section 7 concludes.

2. Browsing the World

In this section, we better define the scenario in which our research situates, by properly identifying the components involved in the "browsing the world" vision, and by discussing the associated key challenges.

2.1. Scenarios

As stated in the introduction, in the near future, our everyday environments will be densely populated by a variety of embedded devices such as sensor networks [ChoK03], RFID tags [Wan06]. Users in an environment will be able, via wireless interfaces mounted on some wearable computing device (e.g. a PDA or a smart phone), to directly access devices in their proximities to gather information about phenomena occurring in the surroundings or (as in the case of RFID tags attached to objects) about nearby physical objects. In addition, users will be able to access to the Web via some wireless communication technology, to dynamically retrieve any needed information. Other than accessing "traditional" Web information (e.g., html pages and Web services), this also enables users to access geo-located information concerning specific sites geographical areas and general facts and annotations about them, as they can continuously provided via collaborative Web 2.0 technologies by the Web community [Esp01, TerK06]. In addition, it enables users to access information generated by sensors and embedded devices (far in the world or close to him but beside his direct access).

Users, in turn, can decide to unveil (totally or to some limited extent) their presence in an environment, by making somehow available to the public their identity, location, and/or activities. This can occur by dynamically uploading such information on the Web, or by making it available to other via ad-hoc connections, or even by uploading it into surrounding pervasive devices. In this latter case, pervasive devices such as RFID tags would act as a sort distributed memory infrastructure [MamQZ06]. The location of users will be always available, either because they will carry on a GPS or because of location can be inferred by the patterns of access to pervasive devices (e.g., the access to a RFID tag with a known location implicitly determines the location of the user [Sat05]) (see Fig. 1).

On the basis of the above considerations, the concept of browsing the world, in general terms, consider the possibility of navigating in an information space that – by properly merging and integrating information coming from both pervasive devices and the Web can represent a detailed model of the world, comprising both present and historic fine-grained geo-located data about the world, its entities, its processes, and its social life. In context-aware user-centric terms, which are the ones of more interest here, the concept of browsing the world implies the possibility for users in an environment to access and navigate meaningful information about the surrounding physical world, and for software services to access and manipulate such information to enforce various degree of context-awareness and context-adaptation.

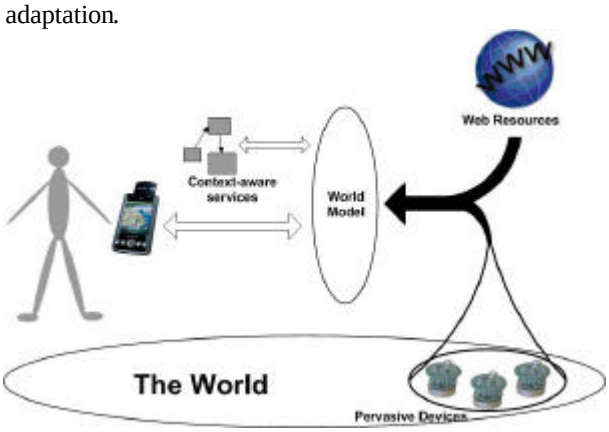

Figure 1. The general scenario of browsing the world.

2.2. Challenges

From the merely technological viewpoint, the "browsing the world" vision could be already turned into reality. Indeed, novel services and Web sites that can be included in the "browsing the world" category appear every day [Cas06]. However, beside specific service implementations, for browsing the world to become common practice based on sound engineered activities, several challenges remains to be addressed. In particular:

1. It would be fundamental to create a general model to represent context information and to build a world model. Spatial information is important but it is not enough. Temporal information must be included, as well as information describing the

activities taking place in the world. The model should enable to deal with incomplete information, and should allow navigation among context information on the basis of what is available at a given time.

2. It would be important to have a general infrastructure supporting the model that should work without requiring or committing to the availability of specific technologies. The infrastructure should be general-purpose, autonomic and adaptable. Relying on this infrastructure, the activities of browsing the world should not be compromised because of say, the temporal unavailability of an Internet connection or the unavailability of a GPS, or of an RFID reader. Consequently, applications built on that infrastructure should not mandate the availability of specific information, but should exploit whatever available information on a best effort basis.

Beyond the horizon, it would be important for such a general model to enable easy processing of data, to facilitate the identification of links between isolated bunch of information. This would enable the creation of complex knowledge networks, and possibly would promote the creation of "new knowledge", as it can be derived by inference from existing information. The attempt to face the above challenges, by defining a simple yet effective model for context data and a general software infrastructure, as preliminary and incomplete as it can be, is the exact goal of our work.

3. The W4 Context Model

We propose a simple model in which context data is expressed by a four field structure: *Who*, *What*, *Where* and *When*. Such a model appears effective in a number of circumstances since it points out some of the main topics that are involved in human thinking: who is acting? What is he/she/it doing? Where and when the action takes place?

3.1. Overview

The goal of our proposal is to develop a general model to manage contextual information. Such information will come from multiple and heterogeneous sources, and would be related to a large number of situations ranging from the description of physical properties in geographic areas, to social facts and processes happening in the world.

In particular, we developed a model in which context data is described by means of a 4-fields tuple: (Who, What, Where, When). We chose this structure because of its evident meaning and flexibility. In fact, a W4-tuple allows to express a situation in a rather natural and human-like way, e.g., "someone or something (*Who*) does some activity (*What*) in a certain place (*Where*) at a specific time (*When*)". We call each of these tuples a *knowledge atom* to describe the fact they represent an atomic unit of context information.

Knowledge atoms are created by a number of software agents running on different (possibly embedded) devices, and will be stored in a suitable shared data space (in section 4, we detail our actual implementation of this space). Knowledge atoms can be retrieved via a pattern matching mechanism, and queries can be expressed as W4 tuples with empty fields (formal parameters). Application agents (ranging from context-aware service providers, to simple interfaces supporting users in browsing information) will access the shared space to retrieve those context information that are suitable for their application task.

3.2. W4 Data Representation and Generation

In the following we detail the four-fields (*Who*, *What*, *Where*, *When*) that constitute our context representation.

- ***Who*** is the subject described by the context structure. *Who* may be a human person (e.g., Gabriella) or an unanimated part of the context (e.g., a RFID tag). The *Who* field is represented by a string with an associated namespace that defines the "kind" of entity that is represented. For example, valid entries for this field are: "person:Gabriella", "tag:tag#567".
- ***What*** is the activity performed by the subject. This information can be either inferred from other context parameters (e.g., an accelerometer can reveal that the user is jogging), or it can be explicitly supplied by the user. This field is represented as a string containing a predicate-complement statement. For example, valid entries for the *What* field are: "read:book", "work:pervasive computing group", "read:temperature=23".
- ***Where*** is the location to which the context relates. In our model the location may be a physical point represented by its coordinates (longitude, latitude), a geographic region (specifically, our model adopts the Postgis language to describe such a region [postgis.refractions.net], or it can also be represented as a logical place. Logical places like "campus" or "bank" are mapped in their actual location by using a fixed dictionary. Logical places like "here" are mapped via simple algorithms that take into account the user current GPS location. It

is worth noticing that this kind of mapping enforces context-awareness, the same "here" information get multiple meanings depending on the user actual location.

- ***When*** is the time duration to which the context relates. It may be an exact range (e.g., "2006/07/19:09.00am - 2006/07/19:10.00am"), a concise description of a range (e.g., 9:28am), or even a logical value (e.g., "now", "today", "yesterday", "before"). Exact values are represented with a "begin-time-of-day – end-time-of-day" expression. Concise description and logical values are mapped via simple algorithms to the corresponding exact range. For example 9:28am = 2006/07/19:9:28am ± 5min. Again, it is important to emphasize that concise time descriptions and logical times are contextual operators, their meaning depends on the time the query is actually issued.

Software agents are in charge of creating and inserting knowledge atoms in the shared space. Agents sense information from several devices (e.g. RFID tag, GPS devices, Web services) and combine them in order to produce a concise and effective description of what is happening in terms of a W4 tuple. The following examples illustrates the atom generation process.

Gabriella is walking in the campus' park. An agent running on her PDA can periodically create an atom describing her situation.

Who: user:Gabriella
What: works:pervasive computing group
Where: lonY, latX
When: now

The *Who* and *What* information are entered directly by the user at the login of the agent application, *Where* and *When* are dynamically provided by the GPS device.

Gabriella's PDA is connected with a RFID tag reader. A specific RFID agent controls the reader and handles the associated events. When a tag is read, the RFID agent creates a knowledge atom to store the tag information. In particular, either the tag would contain its own description, or the tag ID would be resolved in a dictionary (mapping tag IDs into the associated descriptions) to retrieve associated data. This information, together with the "tag" namespace will fill the *Who* field. *What* is left unspecified. The agent accesses the GPS to retrieve location of the tag and fill the *Where* field. Finally, it completes the When field with the logical value "now".

Who: tag:statue of Ludovico Ariosto
What: -
Where: lonY, latX
When: now

3.3. W4 Interface

Since knowledge atoms will be stored in a shared data space, our model requires two fundamental operations: one for injecting knowledge atoms into the shared space, and one for retrieving atoms from it. In particular, taking inspiration from tuple-space approaches, we defined the following API.

```
void inject(KnowledgeAtom a);
KnowledgeAtom[] read(KnowledgeAtom a);
```

The *inject* operation is trivial: an agent accesses the shared data space and store a knowledge atom there.

The *read* operation, instead, requires some more discussion. The W4 model is suitable not only to represent context information, but for questioning too. A query will be represented by a W4 tuple with missing values (i.e., fields left unspecified). The read operation triggers a patter matching procedure between the query and the knowledge atoms that already populate the data space. Matching atoms are returned as results of the query. In this process, it is important to understand that the pattern matching operations work rather differently from the traditional tuple space model. In fact, our proposal can rely on the W4 structure to enforce more expressive pattern matching operations that have a different meaning for the various Ws.

- **Who and What.** Pattern matching operations in these two fields is based on string-based regular expressions. For example, a patter like "user:*" will match any user.
- **Where.** Pattern matching in this field involves spatial operations (again inspired by Postgis operations). Basically, the template defines a bounding box. Everything within the bounding box, matches the template. For example, a pattern like "circle,center(lonY,latX),radius:500m" defines a circle centered at (lonY, latX) with a 500m radius. Tuples with a *Where* field within the circle will match the template. Logical places have to be translated into actual spatial regions before going through the pattern matching.
- **When.** In this kind of pattern matching, the template defines a time interval. Everything that happened within that interval matches the template. Concise time descriptions and logical times will be converted into actual time interval before pattern matching.

The following two examples illustrate the querying process.

Gabriella is walking in the campus, and wants to know

if some colleague is near. She will ask (read operation):

Who: user:*
What: works:pervasive computing group
Where: circle,center(lonY,latX),radius:500m
When: now

Analogously, Gabriella can ask if some of her colleagues has gone to work in the morning:

Who: user:*
What: works:pervasive computing group
Where: office
When: 2006/07/19:09.00am - 2006/07/19:10.00am

It is important to emphasize that returned answers have not to be "complete" W4 atoms. The pattern matching mechanism also allows matches between incomplete information. Thus, following this approach, applications are based on components entering complete and incomplete context information and getting in response refined (but possibly still incomplete) information.

3.4. Future Extensions

There are two important extensions that could be valuably added to the model.

On the one hand, the current model is somewhat limited by the lack of a reference ontology that could add semantic relationships to the concepts in the W-fields. With such an ontology in place, knowledge atoms could be related also if their fields do not match exactly. Moreover, application agents would be able to manipulate the retrieved context information in a more meaningful way.

On the other hand, although pattern matching operations proved rather flexible to retrieve context information, in our future work, we would like to exploit the W4 structure to better navigate the context repository. More specifically, we would like to link together the various knowledge atoms to form a knowledge network where it would be possible to navigate from one W4 tuple to the other. From this perspective, the W fields could be used as links to other knowledge atoms, so that it would be possible, for example, to follow the *Where* link to get further information on where a given entity is located. Our idea, is that the possibility of querying this network, instead of a flat tuple space, would allow much more semantically rich questions and inferences. In particular, new knowledge could be produced by navigating the knowledge network and combining and aggregating existing information into new knowledge atoms.

It is worth reporting that we already started experimenting with this kind of extension. Specifically, links among knowledge atoms can support localization when the GPS signal is unavailable. Let us focus on a simple example.

Gabriella enters a building. She can read RFID tags around to localize (we actually placed RFID tags in our department containing location information of where the tag is). The knowledge atom associated to one of such tags would be:

Who: tag:room_A
What: -
Where: lonY,latX
When: -

Reading such a tag, would cause Gabriella to localize and produce a knowledge atom like:

Who: user:Gabriella
What: works:pervasive computing group
Where: lonY, latX
When: now

If someone wants to retrieve who is actually in a given room, he can issue a query like:

Who: user:*
What: *
Where: room_A
When: now

This query can be answered correctly by using the link (i.e., cross-reference) between Gabriella – (lonY,latX) – room_A. The system first retrieves the actual location of the room, the queries for users located at that coordinates.

4. The "Browsing the World" Infrastructure

To enable the concept of "browsing the world", we designed and implemented an infrastructure based on the W4 model. In this section we first present the general architecture underlying our infrastructure, then we will detail the parts that fulfill the W4 model. In particular, we present how we integrated our system with Google Earth – Google Maps to display location-based context information in an effective way.

4.1. The W4 Architecture

A general infrastructure to enable human-centric browsing of the world must include services for data acquisition, data integration, and data visualization. The architecture we have implemented is organized as follows:

1. Putting humans at the center, our architecture considers users with portable computing devices (i.e., laptops or PDAs), integrating localization devices (i.e., GPS), devices to acquire information

from the physical world (i.e., RFID readers and sensors), and means to connect to the Internet (i.e., WiFi and/or UMTS connections).

2. Data representing contextual information about the world, there included user data, data coming from pervasive devices, and more generally data available on the Web, is represented by means of W4 tuples and stored in the local tuple space to be later accessed by application agents.
3. A number of Web-accessible tuple spaces can be used to store/retrieve W4 information. Each space could host information related to either a limited geographic area (e.g., the campus tuple space), or to a specific topic (as happens today for Google Earth – Google Maps mash ups [Rou05]). Application agents will decide whether an information should be uploaded to one of the Web-accessible tuple spaces, or if it should be kept local.
4. A RFID reader (in the form of a wearable glove) connected to the laptop or to the PDA via a serial interface can be used to collect information from RFID tags dispersed in the environment. This information, enriched with the physical location where it has been collected (as provided by the GPS, device) is stored in the local tuple space.
5. Data coming from sensor network nodes (Crossbow MICAz) can be directly accessed by a suitable agent that collects sensed data and store them in the data space. Data can be enriched with the physical location of the actual sensors and converted in the W4 format. Alternatively, sensor data could be collected by a base-station and sent directly to a Web-accessible tuple space.
6. Specific services can be realized by means of application agents (i.e., autonomous software components) running locally on the user portable device and accessing, via the W4 model, both the local and Web-accessible tuple spaces. Also, application agents can interface with a local GIS client (Google Earth or Google Maps) to turn data into a user-centric perspective.
7. On need, agents can dynamically connect to the Web to retrieve additional information to integrate with that coming from the W4 tuple spaces.

The whole system has been realized using the Java language. Web accessible tuple spaces has been implemented through a Postgres database with spatial extensions. The local tuple space is simply implemented by a Java Vector. The RFID reader and the sensors are accessed via JNI and sockets respectively. User interface is provided by Google Earth (for laptops) and Google Maps accessed via the Minimo browser (for PDAs).

Figure 2. User centric infrastructure for browsing the world

4.2. W4 Tuple Space

All the information coming from the embedded devices (GPS, RFID and wireless sensors) is represented by means of W4 tuples, and stored in a local tuple space. Application agents access this space to retrieve W4 context information supporting their activities. Thus, application agents are completely decoupled from low-level embedded devices, and so they access and deal with contextual information only in terms of the W4 model. In addition, the availability of a local tuple space allows the system to work also in absence of a network connection and allows to minimize the generated data traffic (and its associated costs). Since this tuple space has to run on portable devices, it has been implemented by a simple Java Vector accessible with the W4 interface as described in 3.3.

Other than the local tuple space, our infrastructure comprises also a number of Web-accessible servers allowing multiple users to exchange information and to conduct global queries. In general, an application agent will access the local tuple space and some of these remote spaces to retrieve a detailed description of its context. Our current implementation of a remote tuple space consists of a Tomcat Web server giving access to a Postgres database that store the W4 tuples. We realized JSP and Servlets implementing the W4 interface. Our Postgres database is based on a single table consisting of the four Ws fields. Thus, it actually

resemble a bag of tuples.

A general problem is for application agents to decide which information has to be sent to the World tuple space and which has to remain only locally confined. This decision may depend on many factors, such as privacy issues (e.g., a user may not be comfortable of constantly sending his GPS location on the Web) and scalability reasons. In our current implementation, where the user base is extremely limited and scalability issues are not compelling, we simply upload to the global server all the knowledge atoms whenever the wireless network is available.

4.3. W4 Query Engine

The W4 query engine is the component that is in charge of managing the W4 queries, translating on need logical values (e.g. when = now) into actual ones (when = 2006/07/19:9:28am ± 5min), and performing pattern matching operations. We developed two implementations of the query engine.

1. The query engine running on the local tuple space has been developed in Java. It basically, scans the local Vector of tuples and uses String parsing methods and simple geometric algorithms (to handle *Where* clauses) for pattern matching.
2. The query engine running on the Web accessible tuple space dynamically translates W4 queries in SQL to execute them on the Postgres database. In this implementation, query pattern matching is supported either natively by SQL or by the Postgis spatial extensions.

Both these two engines allow to deal with spatial queries both in terms of actual geographic areas (expressed in term of longitudes and latitudes), and in terms of logical places (e.g., rooms).

It is worth emphasizing that the current implementation is only a first prototype and the current tuple space and query method is rather naïve. However, in future implementations, we will enrich the current infrastructure so as to manage, organize and integrate data in a more complex and clever way. In particular, as discussed in 3.4, we would like to abandon the current flat tuple-based implementation and structure context information in networks of knowledge. Such network-based representation would be more naturally distributable and could make our infrastructure more adaptive and autonomic.

4.4. The Graphical Interface

We developed a flexible graphical subsystem that can be easily employed on both laptops and PDAs. In particular, our GUI interfaces with the GIS tools made available by Google: Google Earth and Google Maps to display retrieved context information as placemarks in a specific geographical area (see Fig. 3, 4, 5). Our graphical subsystem is based on the Keyhole Markup Language (KML), fully supported by Google Earth (at the moment only available for desktop and laptop computers), and at least partially supported by Google Maps and Google Maps for Mobile (that can be accessed also by PDAs and smart phones). This language allows to enrich geographical images coming from the Google GIS software with custom placemarks, images, 3D objects, etc. Thus, our graphical interface just translates proper W4 tuples in a corresponding KML file and dynamically provides it to the Google software. The fact of leveraging on existing and assessed technologies (KML) is an important asset of our implementation, since it allows users to access the system by using consolidated platforms (Google Earth / Google Maps).

It is worth noticing that the KML language allows also to specify the user viewpoint on the map. This naturally supports context awareness, in that an agent could decide to center the map where relevant information are located. In particular, by centering the map on the user, the agent can provide a user-centric representation of the world, where the user can literally *see* nearby resources.

5. Application Examples

To test our model and infrastructure, we developed some simple applications highlighting the flexibility of the W4 representation. In all these examples, we implemented a software agent that:

1. receives either static or dynamic queries from the user.
2. accesses the World tuple space to retrieve suitable context information.
3. creates a KML-formatted answer, and displays it either in Google Earth (for laptops) or in Google Map (for PDAs).

We are aware that the applications we developed are not completely novel, and other researches present similar services [Rou05]. However, our aim is to show that the W4 model can support a variety of application in a uniform and intuitive approach.

5.1. The Journey Map

A first application we have experimented allows to provide context-aware information to a user equipped with a GPS device and a RFID reader. In particular, we focused on the scenario in which a tourist wants to

automatically build and maintain a diary of his journey. To this end, the proposed service allows to keep track of all the user movements and have them displayed on the map of the visited places. Moreover, the support for RFID allows to access likely-to-be-soon-available tourist information stored in RFID tags attached to monuments and art-pieces. From the diary perspective, this allows to store the visited art-pieces' location together with their description on the journey map.

The W4 model can accommodate a number of interesting queries in this scenario. A first query allows to retrieve information about RFID tags being read.

Who: rfid:*
What: *
Where: lonY, latX
When: now

This query retrieves the knowledge atom describing the tag being read (recall that there is a RFID agent that is in charge of reading nearby tags and represent them in W4 format). We implemented this as a static query that the agent asks cyclically to the local tuple space. If a tag is found, its content (properly parsed and enriched with Web-retrieved information) is used to create a KML placemark that will be displayed in the user interface (see Fig. 3).

Another functionality we realized for the journey map application allows an agent to recover user past locations from the World tuple space. This service could be useful to review a past tour and to check the places where the user has been. The associated W4 query can be expressed in the form:

Who: user:Gabriella
What: *
Where: *
When: yesterday

Similarly as before, we implemented this service as a static query. The agent queries the World tuple space, retrieves a list of past GPS traces and displays them as a KML-ploryline in the user interface (see Fig. 5).

It is worth noticing that data coming from sensor network could be accessed via a similar W4 queries. The application can be configured so as to retrieve and display environmental data coming from sensor nearby.

Figure 3. (top) The RFID-reader embedded in a glove allows to identify tagged objects. (bottom) The RFID tag becomes a placemark with Web-retrieved information in the GIS software.

Figure 4. GUI showing user's past GPS traces

5.2. The People Map

A user equipped with a GPS device can decide to share his location with other users and, analogously, he may wish to be aware of the location of others users. For example, a group of friends can share their actual GPS locations (represented as knowledge atoms) with each other. This can happen either by uploading knowledge atoms to the World repository, or by exchanging them in ad-hoc way and storing them in the local tuple space only. Either way, collected knowledge atoms can be used to display users' locations on real-time a map (which can also highlight other interesting Web-retrieved information for the group, such as museums or hotels, depending on the specific interests of the group). It is worth noticing that our current implementation deals with privacy by leaving up to the individual user to decide whether to: share its position or not (and with which accuracy), make it available only to a restricted group of users, or to make it publicly available but only in an anonymous way.

In this application, we developed two interfaces to access the W4 tuple space. A first interface allows to directly compose a query in terms of the W4 fields (see Fig. 5-top). A second interface offers a precompiled menu that is automatically translated into a W4 query (see Fig. 5-bottom).

In both cases, the results are then visualized at the correct location (i.e., in the form of Google Earth / Google Maps placemarks). Since the answer to a query depends often on the location of the mobile users, the results of these queries dynamically change as the users change their location in context-aware fashion.

Figure 5. Map showing users real time locations with neighbor university facilities. (top) PDA user interface with the browser Minimo. (bottom) laptop user interface with Google Earth.

6. Related Works

In recent years, several models addressing contextual information and context-aware services have been investigated, and several infrastructures approaching -- to some extent – our concept of "browsing the world" have been proposed. In this section, we discuss some relevant proposals in these areas.

6.1. Related Context Models

A first group of researches focuses on models for context-aware information trying to create high-level and general-purpose context representation from low-level sensor data.

The work by Schmidt et al. [SchATT99] concentrates on the acquisition of context data from sensors and the processing of this raw data through a layered model. Similarly, the Context Toolkit [DeyAS99] focuses upon deriving context from raw data by providing abstract components that can be connected together to capture and process the data from sensors. Although powerful, these approaches lack of a common semantic to describe the data. This force developers to build new query languages and new components that strongly depend on the kind of information at hand. On the contrary the W4 model provides a common semantic to deal with multiple context information in a coherent way.

Another group of researches focuses on developing context models that can be easily queried. The work proposed by Schilist et al. [SchAW94] creates a simple context model in which information are maintained by a

set of environments variables that can be accessed in a flexible way. Analogously, Henricksen et al. in [HenIR02] analyze context by adding several features such as the temporal aspect, information imperfection, etc. These approaches lead to a long list of all the characteristics of context, lacking in simplicity. In fact it becomes very difficult to browse the list effectively. Instead, the W4 model avoids this problem by organizing the characteristics of the context in the 4 fields discussed.

A third group of researchers describe context via a set of tuples with name-value pairs. The Context Fabric model [Hong02] is based on context tuples each describing a single piece of context data in terms of entities (people, place, thing), attributes (e.g. the name) and relationship, special kinds of attributes that reference to other entities. Similarly, Egospaces [JulR02] provides a structured notion of context as name-value pairs. Egospaces addresses context-aware programming in ad-hoc environments populated of agents by proposing an egocentric notion of context, i.e. every agent holds a personal representation of the world - that representation is called view. The main shortcoming of both these approaches is that it is difficult to browse the context description because of the lack of a predefined structure in the data. The W4 representation, instead, strongly structure our context information.

Finally, a very interesting proposal is presented in [XuC05]. This work adopts in a seven-field data structure to describe the context. The fields are: subject, predicate, object, time, area, certainty, freshness, with similar meaning to W4. Beyond the fields meaning, the purpose is different: their context model is not for browsing the world application, but for managing the consistency between data from multiple sources. Similar considerations apply for the system described in [BraHCN06]. This work describes RFID tags with a structure similar to ours. However, this is not a general model, since it is applied to RFID tags only. The strength of our approach, instead, is to be general purpose and able to represent a large number of context information.

6.2. Related Infrastructures

In the past few years, several infrastructures to integrate context information with GIS-like tools have been presented.

Some streams of works is simply based on representing Web information overlaid to geographical maps [But06, Rou05]. Examples include representations of avian–flu-outbreak reports [declanbutler.info/Flumaps1/avianflu.html], celebrity sightings [www.gawker.com/stalker] and real estate information [www.forsalebyownercenter.com/google-earth-real-estate.aspx]. More dynamic applications, combine collaborative technologies (e.g., blogs and wiki) to GIS tools. MapWiki [TerK06], for example, is a Wiki collaborative environment where all the contents are located on a map. The contents can be edited and moved across the map and accessed on a location basis. All the above projects represents promising starting points of a future in which a wide range of information will be properly conveyed by novel GIS services. However, most of the above researches are special purpose and lack of a general architecture to manage and integrate pervasive, Web and GIS data. Furthermore, in opposition at our user-centric vision, their aim is to produce a centralized view of the world.

Other works concerned systems characterized by the presence of exploratory users and a surrounding environment. Users move across the environment and access information exploiting different type of embedded sensors. Example of this systems are TinyLime [CuGG05] and the system proposed in [MamQZ06].

TinyLime is a middleware for wireless sensor networks that departs from the traditional setting where sensor data is collected by a central monitoring station, and enables instead multiple mobile monitoring stations to access the sensors in their proximity and share the collected data through wireless links. Similarly, the work in [MamQZ06] describes the implementation of a tuple-based distributed memory realized with the use of RFID technology. By accessing in a wireless way the re-writable memory of such RFID tags according to a tuple-based access model, it is possible to enforce mobile and pervasive coordination and improve our interactions with the physical world. From a certain point of view we can consider these systems as “World Browsing” systems. Nevertheless we can say that our new system is certainly more complete and structured. These systems indeed base their functionality on specific technologies (either sensors or RFID tags), they aren’t based on a well-structured context model and further they don’t exploit information coming from the Web (as our does).

A further interesting project is FLAME2008 [WeissVoG04]. Users through their PDA access to services (related to the 2008 Olympics games in Beijing) expressly fitted on their needs: FLAME2008 elaborates them on the base of activities and situations carried out by the user. The infrastructure is very interesting, in particular for its use of ontologies, and appears to be very complete. Nevertheless we noticed

that it's too bound to a specific application domain and it doesn't perform any mechanism for generate and store new. Our model is certainly more user centric and it has been developed to adapt itself to a generic context, and to be fully functional even without the support of a centralized infrastructure.

7. Conclusions and Future Works

In this paper we presented a simple model and infrastructure to *(i)* access context information coming from embedded devices and Web resources in a comprehensive framework, and *(ii)* to effectively visualize it –possibly with mobile devices -- with novel GIS tools. Our future research in this area will mainly focus on two aspects. On the one hand, we will try to integrate ontologies in our model to improve its expressiveness and flexibility. In particular, ontologies will allow more semantic forms of pattern matching among W4 tuples. On the other hand, we will try to go further than the current flat (i.e., tuple space) data organization and link knowledge atoms in suitable knowledge networks allowing a better and more semantic navigation of context information.

References

[BraHCN06] J. Bravo, R. Hervas, G. Chavira ,S. Nava, "Modeling Contexts by RFID-Sensor Fusion", IEEE International Conference on Pervasive Computing and Communications Workshops, Pisa, Italy, 2006.

[But06] D. Butler, "Virtual Globe: the Web-Wide World", Nature, 439:776-778, Feb 2006.

[Cas06] G. Castelli, A. Rosi, M. Mamei, F. Zambonelli, "Browsing the World: Bridging Pervasive Computing and the Web", International Workshop on Ubiquitous Information Systems, Munster, Germany, 2006.

[ChoK03] C.-Y. Chong, S. P. Kumar, "Sensor Networks: Evolution, opportunities, and challenges", Proceedings of the IEEE, 91(8):1247-1256, 2003.

[CuGG05] C.Curino, M. Giani, M. Giorgetta, A. Giusti, A.L. Murphy, G.Picco: "Mobile Data Collection in Sensor Networks: The TinyLIME Middleware"; International Conference on Pervasive Computing and Communications, Kauai Island (HW), USA, 2005.

[DeyAS99] Anind K. Dey, Gregory D. Abowd, Daniel Salber. "A Context-Based Infrastructure for Smart Environments", International Workshop on Managing Interactions in Smart Environments, Dublin, Ireland, 1999.

[Esp01] F. Espinoza, P. Persson, A. Sandin, H. Nystrom, E. Cacciatore, M. Bylund, "GeoNotes: Social and Navigational Aspects of Location-Based Information Systems", International Conference on Ubiquitous Computing, Atlanta (GE), USA, 2001.

[HenIR02] K. Henricksen , J. Indulska , A. Rakotonirainy, Modeling Context Information in Pervasive Computing Systems, International Conference on Pervasive Computing, Zurich, Switzerland, 2002.

[Hong02] J. Hong., "The Context Fabric: An Infrastructure for Context-Aware Computing.", Conference on Computer Human Interaction, Minneapolis (MN), USA, 2002

[JulR02] C. Julien , G. Roman, "Egocentric context-aware programming in ad hoc mobile environments", Symposium on Foundations of Software Engineering, Charleston (SC), USA, 2002.

[MamQZ06] M. Mamei, R. Quaglieri, F. Zambonelli, "Making Tuple Spaces Physical with RFID Tags", Symposium on Applied Computing, Dijon, France, 2006.

[Rou05]W. Roush, "Killer Maps", Technology Review, 11 September 2005

[Sat05] I. Satoh, "A Location Model for Pervasive Computing Environments", International Conference on Pervasive Computing and Communications, Kauai Island (HW), USA, 2005.

[SchATT99] A. Schmidt , K. A. Aidoo , A. Takaluoma , U. Tuomela , K. Van Laerhoven , W. Van de Velde, "Advanced Interaction in Context", International Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, 1999.

[SchAW94] B. Schilit, N. Adams, and R. Want. "Context-aware computing applications". Workshop on Mobile Computing Systems and Applications, English Lake District, UK, 1994.

[TerK06] Y. Teranishi, J. Kamahara, S. Shimojo, "MapWiki: A Ubiquitous Collaboration Environment on Shared Maps", International Symposium on Applications and the Internet Workshops, Phoenix (AZ), USA, 2006.

[Wan06] R. Want, "An Introduction to RFID Technology", IEEE Pervasive Computing, 5(1):25-33, 2006.

[WeissVoG04] N. Weißenberg, A. Voisard, Rüdiger Gartmann, "Using Ontologies in Personalized

Mobile Applications", International Symposium on GIS, Washington (DC), USA, 2004.

[XuC05] Chang Xu , S. C. Cheung, "Inconsistency detection and resolution for context-aware middleware support", International Symposium on Foundations of Software Engineering, Lisbon, Portugal, 2005.