Mobile Netw Appl (2009) 14:107–119
DOI 10.1007/s11036-008-0104-y

Extracting High-Level Information from Location Data: The W4 Diary Example

Gabriella Castelli · Marco Mamei · Alberto Rosi · Franco Zambonelli

Published online: 25 September 2008

Abstract Services for mobile and pervasive computing should extensively exploit contextual information both to adapt to user needs and to enable autonomic behavior. To fulfill this idea it is important to provide two key tools: a model supporting context-data representation and manipulation, and a set of algorithms relying on the model to perform application tasks. Following these lines, we first describe the W4 context model showing how it can represent a simple yet effective framework to enable flexible and general-purpose management of contextual information. In particular, we show the model suitability in describing user-centric situations, e.g., describing situations in terms of where a user is located and what he is doing. Then, we illustrate a set of algorithms to semantically enrich W4 represented data and to extract relevant information from it. In particular, starting from W4 data, such algorithms are able to identify the places that matter to the user and to describe them semantically. Overall, we show how the context-model and the algorithms allow to create an high-level, semantic and context-aware diary-based service. This service meaningfully collects and classifies the user whereabouts and the places that the user visited.

G. Castelli · M. Mamei (✉) · A. Rosi · F. Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria,
Università di Modena e Reggio Emilia,
Via Amendola 2,
42100 Reggio Emilia, Italy
e-mail: marco.mamei@unimore.it

G. Castelli
e-mail: gabriella.castelli@unimore.it

A. Rosi
e-mail: alberto.rosi@unimore.it

F. Zambonelli
e-mail: franco.zambonelli@unimore.it

Keywords pervasive computing · context-awareness · W4 model · knowledge engineering location-aware services · localization algorithms

1 Introduction

Pervasive computing scenarios comprise a huge number of heterogeneous devices interacting with each other to achieve complex distributed applications. Sensor networks and networks of mobile devices could be employed in a variety of applications including environmental monitoring [1, 22, 26] navigation, and human interaction [24]. One central problem underlying such applications is how to obtain a useful representation of the environment surrounding the system, and how to interpret and understand such a representation to influence the application-level business logic.

In fact, while advances in hardware technologies (e.g., smart phones, wireless sensors and RFID tags) are making possible and economically feasible to collect a vast amount of information about the system context, it is still very difficult to organize and aggregate all the collected information in a coherent and usable representation. In other words there is a huge gap between low-level sensor readings and high-level "situation awareness" [23].

Accordingly, pervasive and autonomic services should evolve from a simple model of "context-awareness", in which they access isolated pieces of heterogeneous contextual data and are directly in charge of digesting and understanding them, towards a model of "situation-awareness", in which services access properly structured and organized information reflecting comprehensive contextual knowledge related to a "situation" of interest and which can be exploited in a standardized way. The

result is that services can reach, with reduced computational and communication efforts, a comprehensive understanding of "situations" around and, consequently, a higher-degree of adaptability and flexibility, with the notable software engineering advantage of enforcing a sharp separation of concerns between context exploitation and context management.

To tackle the above problems and create effective context-aware services, two main tools are necessary:

1. A context model offering a principled and general-purpose way to encode and manipulate context information
2. A set of algorithms to extract practically usable information from low-level and sparse context data

In this paper we present our contribution to these research challenges:

1. 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 in a variety of pervasive and mobile computing scenarios.
2. We propose a number of algorithms to enrich W4 information with semantic data and to extract higher-level patterns from that. In particular, starting from a collection of W4 information representing the user whereabouts, the algorithms we developed identify relevant places, enrich their description with Web-based information, and classify them semantically in an unsupervised way.

The combination of the above two instruments allows to create a diary-like application running on a GPS-equipped handheld device. The application records the list of relevant places visited by the user in terms of W4 information, and use the identified algorithms to extract high-level information about. The result is a W4-based diary describing the user daily life and that could be used either as a stand-alone application allowing a user to browse through his past locations, or as a supporting tool for other applications that could take advantage of the recorded data.

From a general perspective the W4 model and the algorithms to extract high-level context data from locations can be regarded as sorts of middleware-level infrastructures to support location-aware mobile services. While most of the proposed middleware infrastructures focus on "low-level" issues, like communication and interaction among nodes, the W4 model and our algorithms focus on high-level issues such as context-representation and its understanding (e.g., how to represent information about the places, what are the places that matter to the users, what they mean to them). Such kind of infrastructures will become more and more important in the future. As context information will become more detailed and relevant, there will be the need of managing them at the middleware level, providing application with a pre-digested and understandable view of such data [7].

The remainder of this paper is organized as follow. Section 2 presents the W4 model showing how it can be used both to represent and then retrieve context information. Section 3 presents the algorithms that we used to extract higher-level information from W4 location data and describes how the algorithms can be combined in a W4 diary application. Section 4 presents experiments and discusses performance of our implementation. Section 5 presents related work. Section 6 discusses areas for future work and concludes.

2 The W4 context model

The W4 model starts from the consideration that, in a large set of application scenarios, both elementary and higher-level context information represent a "fact" which has occurred in the world. Accordingly, our proposal simply accounts that any of such facts—and therefore any data/knowledge information—can be expressed by means of simple yet expressive four-fields tuples (Who, What, Where, When): "someone or something (Who) does/did some activity (What) in a certain place (Where) at a specific time (When)".

W4 tuples may be created by proper software agents associated to data sources or sensors. Their four-fields structure is flexible and general enough to uniformly deal with information coming from sources as diverse as embedded devices, cameras, users, Web 2.0 sites (e.g., Twitter), and can easily account for incomplete information (i.e., some of the four fields being unspecified). W4 tuples can be stored in suitable shared data spaces, whatever distributed and implemented. Users and services, from everywhere, can retrieve tuples via a simple API, based on Linda-like pattern-matching query mechanisms [2]. Such API supports context-aware queries and incomplete information, to enable services to possibly interact with each other and to enforce autonomic and context-aware functionalities (see Fig. 1).

2.1 Data representation

The four-fields (Who, What, Where, When) of the W4 data model each describes a different aspect of a contextual fact.

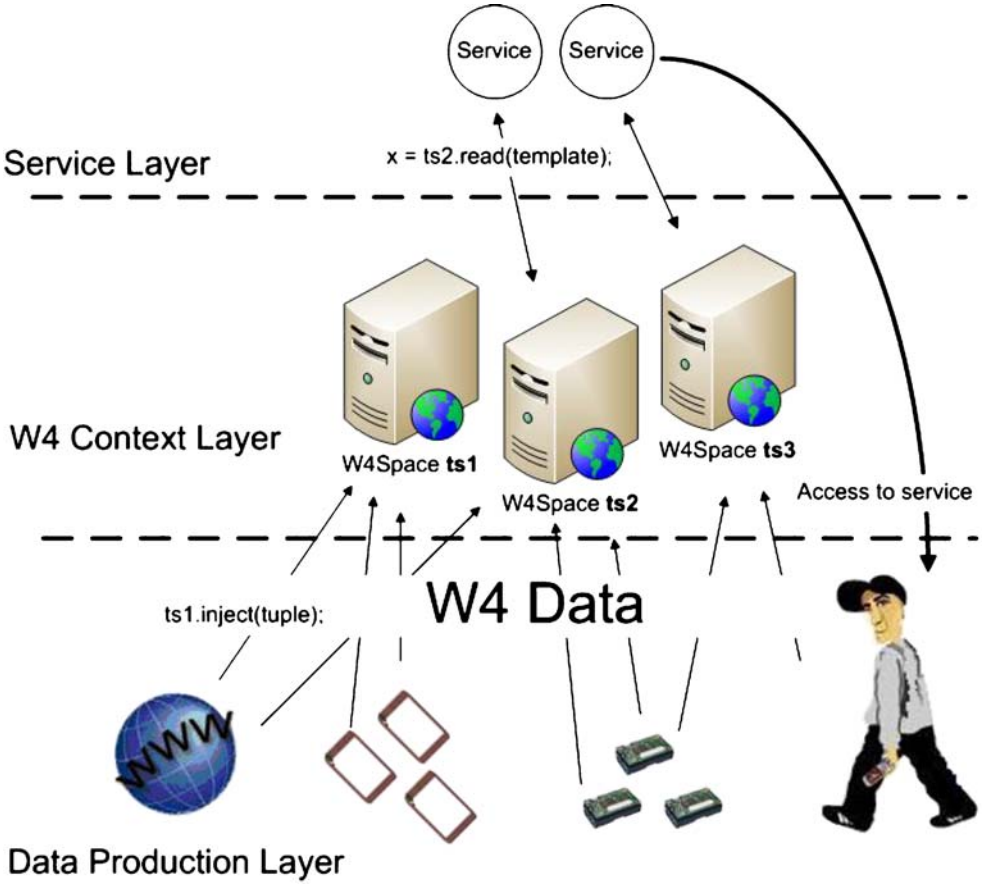

Figure 1 W4 general architecture. Different entities in the data production layer produce W4 tuples. Services access to W4 tuples to create context-aware services. The W4 Context layer decouples the raw context data and services

- The "Who" field associates a subject to a fact, and may represent a human person (e.g., a username) or an unanimated part of the context acting as a data source (e.g., the ID of an RFID tag). The Who field is represented by a type-value pair, in the form of a string, with an associated namespace that defines the "type" of the entity that is represented. For example, valid entries for this field are: "person:Marco", "tag:tag#567".
- The "What" field describes the activity performed by the subject. This information can either come directly from the data source (e.g., a sensor is reading a temperature value), or be inferred from other context parameters (e.g., an accelerometer on a PDA can reveal that the user is running), or it can be explicitly supplied by the user (e.g., via the use of Web 2.0 sites like Twitter). 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".
- The "Where" field associates a location to the fact. In our model the location may be a physical point represented by its coordinates (longitude, latitude), a geographic region (we currently adopt the PostGIS language to describe such regions), or it can also be a logical place (e.g., the engineering department). In addition, context-dependent spatial expressions like "here" or "within:300 m" can be used for context-aware querying, as described in the following of this section.
- The "When" field associates a time or a time range to a fact. This may be an exact time/time range (e.g., "2006/07/19:09.00 A.M.–2006/07/19:10.00 A.M."), or a concise description (e.g., 9:28 A.M.). For example 9:28 A.M.= 2006/07/19:9:28 A.M.±5 min. Also in this case, context-dependent expressions can be defined (e.g., "now", "today", "yesterday", "before") and can be used for context-dependent querying.

The way it structures and organizes information makes the W4 data model general enough to represent data concerning several kind of situations and simple enough to promote ease of management and processing.

2.2 Data access and service engineering

To provide a complete model, it is important to define a simple API to access contextual knowledge and to enable data sources and services to inject new data in the system.

As already introduced, W4 tuples are stored in a collection of shared data spaces. Each data space is intended to contain a set of logically-related tuples. For example, a data space might be devoted to store context information related to a specific geographical area, or describing a situation according to a specific level of "semantic granularity".

To define the API reported in Fig. 2, we took inspiration from tuple-space approaches [2]. A service needs to have a reference to a tuple space to perform one of the API operations on it (see Fig. 1).

The inject operation is equivalent to a tuple space "out" operation: an agent accesses the shared data space to store a W4 tuple there.

The read operation is used to retrieve tuples from the data space via querying. A query is represented in its turn as a W4 tuple with some unspecified or only partly specified values (i.e., a template tuple). Upon invocation, the read operation triggers a pattern matching procedure between the template and the W4 tuples that already populate the data space. A vector of all matching tuples—i.e., those for which all the defined fields match those provided in the template—is returned as the result of the query. In any case, pattern matching operations work rather differently from the traditional tuple space model. In fact, our proposal can rely on the W4 structure to enforce expressive context-aware pattern matching operations, which may exploit differentiated mechanisms for the various W4 fields. Current mechanisms work as follows:

- Who and What. Pattern-matching operations in these fields are based on string-based regular expressions. For example, "user:*" will match any user.
- Where. Pattern matching in this field involves spatial operations inspired by PostGIS operations. Basically, the template defines a bounding box (e.g., "circle, center(lonY,latX), radius:500 m") and everything within the bounding box matches the template. All tuples with a Where field within the circle will match this field of the template. Contextual places such as "within:300 m" can be specified in the template and are translated into actual spatial regions—based on the current location from where the query is performed—before going through the pattern matching.
- When. In this case, the template defines a time interval. Everything that happened within that interval matches the template. Concise time descriptions as well as contextual ones (e.g., "now" or "before") are converted into actual time intervals before pattern matching.

```
void inject(W4Tuple a);
W4Tuple[] read(W4Tuple a);
```

Figure 2 API to inject and read W4 tuples

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

Figure 3 W4 template querying for nearby users working in pervasive computing group

Two simple examples illustrate the querying process. Let us assume Marco is walking in the campus and wants to know if some colleagues are near. He will ask (via a read operation) the query reported in Fig. 3.

Then, he will get in return the tuples representing all the colleagues of his group currently around (at least, of all those colleagues having decided to expose themselves via a W4 tuple).

Similarly, Marco can ask if some of his colleagues have gone to work in the morning (see Fig. 4)

We emphasize that the returned answers have not to be "complete" W4 tuples. The pattern matching mechanism also allows for matches between incomplete information. Thus, unlike in traditional tuple space approaches, applications are based on components entering complete and incomplete context information and getting in response refined (but possibly still incomplete) information.

2.3 Data generation

In the W4 model, we rely on the reasonable assumption that software drivers (or, more in general, software agents) are associated with data sources and are in charge of creating W4 tuples and inserting them in some sorts of shared data spaces. In the end, any data source must be somehow associated with some software to gather and store data items, W4 agents have the additional goal of collecting all the necessary information to produce a W4 tuple which is as accurate and complete as possible. This occurs by sensing and inferring information from all the devices and sources available (e.g., RFID tags, GPS devices, Web services), and by combining them in a W4 tuple. Some simple examples may clarify this concept.

Let us assume Marco is walking in the campus park. Agents running on his GPS-equipped PDA, can periodically create the tuple in Fig. 5.

The Who is entered implicitly by the user at the login, What and Where can be derived by the GPS (e.g., the speed of Marco as measured by the GPS can be used to deduce that he is walking), When can be provided both by the PDA

```
Who: user:*
What: works:pervasive computing group
Where: office
When: 2006/07/07:00am- 2006/07/13:00am
```

Figure 4 W4 template querying for users working in pervasive computing group that were in the office in the morning

Who: user:Marco
What: walk:4km/h
Where: lonY, latX
When: 2006/10/17:10.59am

Figure 5 W4 tuple describing user location and activity

Who: user:Marco
What: walk:4km/h
Where: st. Amendola
When: 2006/10/17:10.59am

Figure 7 W4 tuple representing the user state from a higher-level perspective

or by the GPS. Viewing this from a different, more fine-grained perspective, we can imagine that one agent controlling the user profile can create a raw W4 tuple in which only the Who and Where are specified; another agent controlling the GPS agent create a tuple in which only Where and What (i.e., the speed) are specified. The merging of these two raw W4 tuples together can produce the complete one represented in Fig. 5.

Now, let us assume that a Web service makes available geocoding information expressed by means of W4 tuples. For example, the service can produce context information like the one presented in Fig. 6 where Amendola st. (Who) is described by a list of longitude and latitude segments (Where).

The agent running on Marco's PDA can use both the data coming from the GPS and the geocoding service to provide a better localization of Marco. For example, more expressive localization information can be retrieved by associating Marco with the closest street segment. So the resulting tuple describing Marco is the result of the merging between the previous ones (see Fig. 7).

Concerning this point, it is very important to see that one of the advantages of the W4 model is to allow the generation of new W4 tuples on the basis of existing information. The above example, for instance, shows the production of higher-level knowledge (the location of the user expressed by means an address) from lower-level unrelated information (the location of the user expressed by means of GPS coordinates and geocoding data). The obtained new information can be stored in another tuple space containing such kind of higher-level information.

In conclusion, the W4 model well suits to a number of pervasive computing scenarios and it can describe a wide range of context information in a simple yet expressive way. The W4 model is designed also to offer a simple API to inject and query W4 information by agents. In addition the proposed uncoupled interaction mechanism facilitates the integration between information coming from multiple sources. For example as will also be detailed in the rest of the paper, we will present integration between localization data coming from pervasive devices (e.g., GPS) and Web-based mapping tools and resources (e.g., geocoders, and white/yellow page services).

Who: st. Amendola
What: null
Where: (lonY1, latX1), (lonY2, latX2), …., (lonYn, latYn)
When: null

Figure 6 Street description by means of W4 information

Once the context model and the associated API are defined, it is possible to create context-aware applications more efficiently. In the next section we present some algorithms to extract high-level information from W4 data.

3 Context analysis algorithms

In this section we present a number of context analysis algorithms we have experimented on the W4 data model. These algorithms are intended to run on a handheld device provided with GPS localization, and their goal is to incrementally extract high-level information from location data.

At the most basic level, a handheld device connected to a GPS can run an algorithm that continuously collects and stores the user locations in terms of longitude and latitude. Such information ban be easily encoded in terms of W4 tuples and injected in a shared data space for further processing (see Figs. 8 and 9).

Exploiting this data, another algorithm can access the W4 repository to incrementally produce more elaborate context-information. In particular we propose an architecture like the one in Fig. 8 where a number of tuple spaces store context information at different levels of semantic abstraction. Each algorithm we developed takes as input the context representation of the tuple space at one-level below, performs some inference operations, and injects the result in the tuple space one-level above.

Although in this paper we will focus only on algorithms to infer and describe the user location, it is worth reporting that there exist several algorithms and methods to infer what activities the user is performing in a given place. This kind of inference can either involve other sensors (e.g., RFID tags [25], digital wallets [9], and accelerometers [4]) that try to capture the signs of the user activity, or it can be possible to prompt the user asking to annotate what he is currently doing [30]. In any case, the description of the activity can complete the "What" field in the W4 tuple describing the user status.

3.1 Extracting places where the user spends his time

Starting from the raw collection of GPS readings, it is possible to run segmentation and clustering algorithms to infer the places where the user spends most of his time [13].

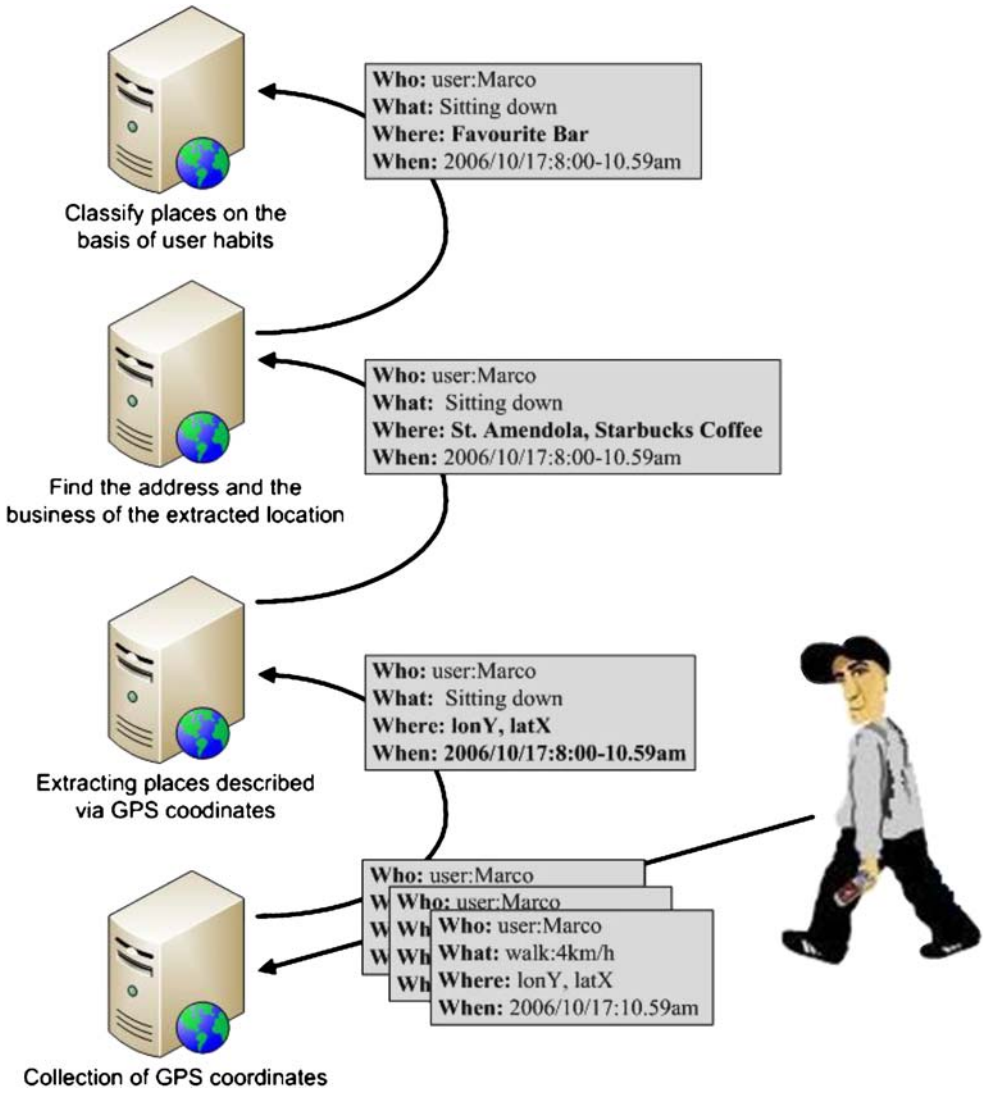

Figure 8 The context-analysis algorithms build on one another to incrementally extract information from the GPS traces. The information is stored in distinct W4 tuple spaces. The base level contains raw information about the user location. Each algorithm we developed takes as input the context representation of the tuple space at one-level below, performs some inference operations, and injects the result in the tuple space at one-level above

Who: user:Marco
What: walking
Where: lon = -73.974, lat = 40.763
When: July, 4, 2006, 4:35:00 pm

Who: user:Marco
What: standing
Where: lon = -73.973, lat = 40.766
When: July, 4, 2006, 4:35:10 pm

Who: user:Marco
What: standing
Where: lon = -73.974, lat = 40.765
When: July, 4, 2006, 4:35:20 pm

Figure 9 W4 tuples representing a log of GPS coordinates

Following an approach similar to the one proposed in [20, 27], the algorithm tags as relevant those places for which either one of the following conditions applies:

1. The GPS signal is lost for at least T seconds and it is re-acquired later on at a distance of less than L meters from where it was lost. This reflects the situation in which a user enters a building and leaves it after some time. Some empirical evaluations let us to set T= 20 min, L=20 m. The constraint on time is important to wash out GPS signal glitches, the constraint on space is useful to avoid those situations in which the GPS has been shut down and the user moves away.
2. The GPS readings over a time window of W seconds are clustered within a radius of R meters from each other. This reflects the situation in which the user stays for a long time in a place like a park or a square. Some empirical evaluations let us to set W=20 min, R= 100 m.

The list of relevant places is built online and incrementally. This algorithm receives and inserts events form the W4 tuple space below, and when a set of coordinates meets one of the above criteria, the algorithm looks in the list of the already discovered places for one closer than $L=10$ meters to the coordinates. If such a place does not exist, a new place is created and the time of visit is recorded. If the place exists, the place coordinates are averaged with the new coordinates, and if enough time has passed since the previous visit (30 min), the time of the new visit is recorded.

The result of this operation is a list of places described in terms of longitude and latitude, and a list of time intervals associated to each of the coordinates indicating when the user has been there (see Fig. 10).

Who: user:Marco
What: standing
Where: lon = -73.974, lat = 40.763
When: July, 4, 2006, 4:35pm-5:41pm

....

Figure 10 W4 description of the places visited by the user

3.2 Extracting addresses and businesses

A simple list of coordinates is only partially informative and the need of translating from positions to places (i.e., adding semantic meaningful tags to the discovered coordinates) has been widely recognized [12]. Information like "the user was at home" rather than "the user was at coordinates (10.873, 44.630)" would be naturally much more informative and easy to use in context-aware applications.

A step in the process of adding semantic information would be to translate from coordinates to addresses. This can be done via standard tracking and geocoding services (as common GPS navigators do). In most of the situations however, because of errors in GPS localization and errors in the process of segmenting and clustering the GPS readings to identify relevant places, it will not be possible to identify the unique address where the user is located, and only a partial estimate can be given. This second algorithm produces the place description reported in Fig. 11.

In our implementation, coordinates associated to places have been translated into addresses using a custom geocoding service. Most of the geocoding services available online (e.g., that provided by the Google Maps API) translate addresses into coordinates. Instead, to create a more descriptive W4 diary, we need the reverse operation: from coordinates to addresses. We developed a "reverse" geocoding service for our region, on the basis of maps

Who: user:Marco
What: standing
Where: 123, 5th Ave, NY, USA
When: July, 4, 2006, 4:35pm-5:41pm

Who: user:Marco
What: standing
Where: 4,5,…,21, 26th St., NY, USA
When: July 5, 2006, 7:00am – 8:00am

Figure 11 Extracting addresses. Because of GPS errors multiple addresses can be associated to a single W4 tuple

available from a commercial navigator software. To take into account GPS and geocoding inaccuracies, and the errors introduced by the place retrieving process, the W4 diary application tries to reverse geocode all the addresses within a radius of 10 m from the place being segmented. Thus, the algorithm actually creates a list of candidate addresses where the user might have been.

A similar algorithm can mine the Web to identify what is in a particular address. The primary source of information in this context would come from yellow- and white-pages services. However, due to the aforementioned localization errors, this process will return in some cases a list of all the businesses performed in the geocoded addresses. Still, in some situations a single exact match could be retrieved like in the case of the user being in a big stadium or entering a big shopping mall. Even more semantic information could derive by searching relevant events that happened in that place at that time. For example, it could be possible to extract from the Web the fact that "the 4th of July parade" took place near the geocoded location at the same time the user was there. This process could create a place representation like the one depicted in Fig. 12.

To perform this operation, in our implementation, we screen-scraped information coming from a widely used online white-pages service[1] in our region allowing to query for who is at a given address. This operation is trivially achieved using the tools provided by the htmlparser[2] software. In particular, each geocoded address belonging to a given place (as provided by the previous step) is looked up in the white-pages and the corresponding business is retrieved. The result of this process is a set of entries labeled with the possible businesses found in that place. This translation process is not completely accurate, since several addresses are not listed in the white-pages (mainly due to privacy constraints). Still, the fact that most public businesses (like shops, etc.) are listed, while several private

[1] www.paginebianche.it

[2] htmlparser.sourceforge.net

Who: user:Marco
What: standing
Where: 4th July Parade
When: July, 4, 2006, 4:35pm-5:41pm

Who: user:Marco
What: sitting
Where: Starbucks Coffee || Uno's Pizza
When: July 5, 2006, 7:00am – 8:00am

Figure 12 Extracting businesses. Because of errors in the previous phases, multiple businesses can be associated to a single place

houses are not, allows to prune out a lot of unlikely addresses being discovered by the previous step. Private spaces like "home"—that are likely not to be listed in the white-pages—can be derived from the algorithm presented in the following subsection.

It is finally worth pointing out that the operations described in this subsection could be automated and improved given the availability of a complete spatial database like the one integrated into commercial navigator systems.

3.3 Extract the meaning of places on the basis of user habits

Given that the user activities are profiled in some way (e.g., the system may know a priori that the user tends to stay at home at night), an algorithm can give labels to places by looking at the temporal patterns in which places are visited. For example, the place most visited at night during weekdays can be meaningfully labeled as "Home". Such kind of analysis can be also used in combination with commonsense information [17, 18] to disambiguate between alternative retrieved places. For example, in Fig. 12 the ambiguity among "Starbucks Coffee" and "Uno's Pizza" can be resolved (at least from a probabilistic point of view) in favor of the former, in consideration of the fact the place has been visited from 7:00 A.M. to 8:00 A.M.

To implement this algorithm, for each place being identified in the first phase, we create a Bayesian network to analyze the temporal pattern in which the place has been visited by the user (see Fig. 13). The Bayesian network is composed of 4 nodes.

1. The *weekend* node represents a Boolean variable used to represent whether a given observation takes place in the weekend or not. This node is always observed on the basis of the information stored in the GPS signal. This information represents the variability in people behavior between weekdays and weekends.
2. The *hour* node is a 24-values discrete node storing the time of day. This node is always observed on the basis of the information stored in the GPS signal.
3. The *kind of place* node is a discrete node modeling what a given place is. In our implementation, we try to classify among five different kinds of places: home, work, restaurant (to indicate any kind of dining place), pub (to indicate any kind of evening entertainment), and disco (to indicate any kind of late-night entertainment). This classification is rather arbitrary, and each user of the application should provide the kinds of place that best match his habits. This node is never observed, and is inferred by probability computations.
4. The *happens* node is a Boolean variable expressing whether the user visits that place at that time. This node is always observed on the basis of the outcomes of the previous localization phase.

The role of the Bayesian network is to encode the routine of the user daily life. This is done by compiling the probability distribution associated to the fact that the user, in a given moment, is in a certain kind of place. For example, the probability of the user being at home during weekdays is depicted in Fig. 14.

Similar tables can be created for other kind of places. In our current implementation, these tables are compiled by hand by each user to whom is asked to self-report the likelihood of being in a given kind of place at a given time. Such kind of data could be derived automatically also by a labeled trace of user's past whereabouts, using standard

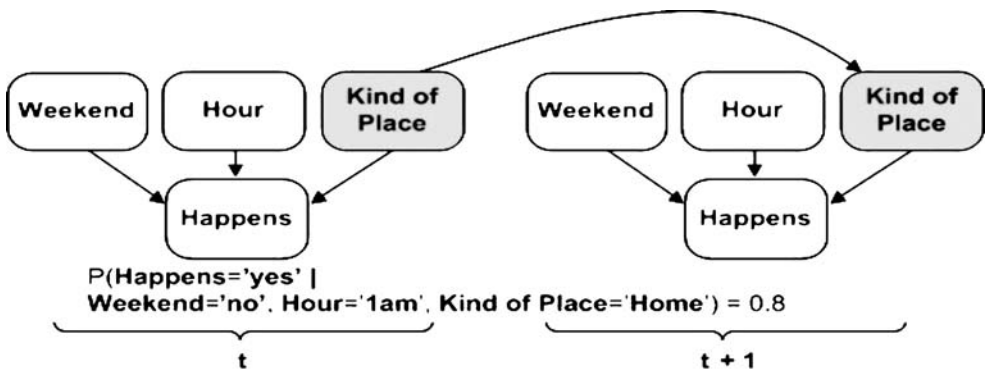

Figure 13 Bayesian network to classify places. White nodes are those that will be provided as evidence

Weekend = false, Kind of Place = home									
Time	**11pm-6am**	**7am**	**8am**	**9am-1pm**	**2pm-5pm**	**6pm-7pm**	**8pm**	**9pm**	**10pm**
P(happens) = true	0.8	0.6	0.4	0.2	0.2	0.4	0.5	0.6	0.7

Figure 14 Conditional probability table describing the probability of the user being at home during weekdays

learning algorithms [23]. Once the tables are filled in, basic inference operations in Bayesian networks will be used to derive the most likely kind of place given the visit pattern.

Specifically, when the previous algorithms identify that the user is visiting a place, the corresponding Bayesian network is retrieved, and the *weekend*, *hour*, *happens* nodes are set to their actual values (the *happens* node is trivially set to *true* to indicate that there is a visit). Then, the application computes the probability distribution of the *kind of place* node. The newly computed distribution will be used as a prior for subsequent visits. This naturally allows evidences to add up, actually enabling the Bayesian network to classify the places on the basis of the visit temporal pattern [19].

The combination of all the presented algorithms can create a W4 diary of the user whereabouts and activities [5]. The result could be used either as a stand-alone application allowing a user to browse through his past, or as a supporting tool for other services that could take advantage of the collected data.

4 Experiments

To test the effectiveness of our algorithms, we collected GPS traces for three weeks from three members of our research team (among the authors) as they went about their normal lives. Each member carried either an i-mate PDA 2K smart phone, or a HP IPAQ RX3700 pda, connected with a Bluetooth GPS reader. GPS signal has been acquired at 0.1 Hz and processed on the fly by the handheld device. Overall, we acquired about 90,000 GPS poses amounting at 360 MB of data. Overall, this resulted in 25 places being identified as relevant. During the data collection weeks, data collectors recorded ground-truth information about the places they have been. Such information has been collected with a simple notepad application running on the PDAs and allowing to write a textual description of where the user has been at a given time. In the following, we present some results obtained by comparing the W4 entries produced by the algorithms, after some weeks of usage, with recorded ground-truth information.

In a first set of experiments, we tried to verify the accuracy of the algorithm to identify relevant places on the basis of the GPS trace log. Following an approach similar to [13], we classify the incorrect results into: (1) *wrong*: the user is in a place, but the algorithm reports he is in a different place, (2) *false negative*: the user is in a place, but the algorithm reports he is moving, (3) *false positive*: the user is moving, but the algorithm reports he is in a place. The results of this experiment are reported in Fig. 15, and they actually show the average of the results obtained by the data collectors. The results we obtained show that the algorithm is correct in 80.6% of the cases. This result is coherent with the results presented in [13] with regard to the A–S algorithm [3] that is the one closer to our implementation. The high-percentage of false negatives (compared to the other cases) is mainly due to the fact sometimes the GPS takes a long time before acquiring the signal. Thus, it can happen that a user leaves a building, and the trace of the GPS is acquired only when he is already far away. In such a situation the place is not detected given the constraint on the maximum distance of spatial disconnection described in Section 2.1.

In a second set of experiments, we tried to verify the results of the (reverse) geocoding service. Basically, the idea is to verify the impact of localization errors in the process of geocoding. It is worth noticing that the maps we used to perform this operation record only the first and the last number of a street segment and span, uniformly, all the other numbers among the segment. This of course introduces further errors in that it does not take in to account the differences in the sizes of the buildings.

Since the place discovery algorithm clusters together points that are closer than 10 m, we counted the number of addresses retrieved within a circle of 10 m radius centered at the relevant place. The results of these operations are displayed in Fig. 16, left, and highlight two aspects. On the one hand, the address of almost half of the places can be retrieved uniquely (this is the case of large buildings—like the departments of our university). On the other hand, some places produce more than ten associated addresses. This is the case of small buildings in the center of the city. It is fair to report that these distributions are rather preliminary since they are based on a dataset of only 25 places (those identified by the diaries of the three data collectors). We are currently conducting a more extensive data collection process that would allow us to identify more stable distributions.

Correct	**Incorrect**		
	Wrong	**False negative**	**False positive**
80.6%	0%	17.3%	2.1%

Figure 15 Errors in the algorithm to identify relevant places on the basis of the GPS trace log

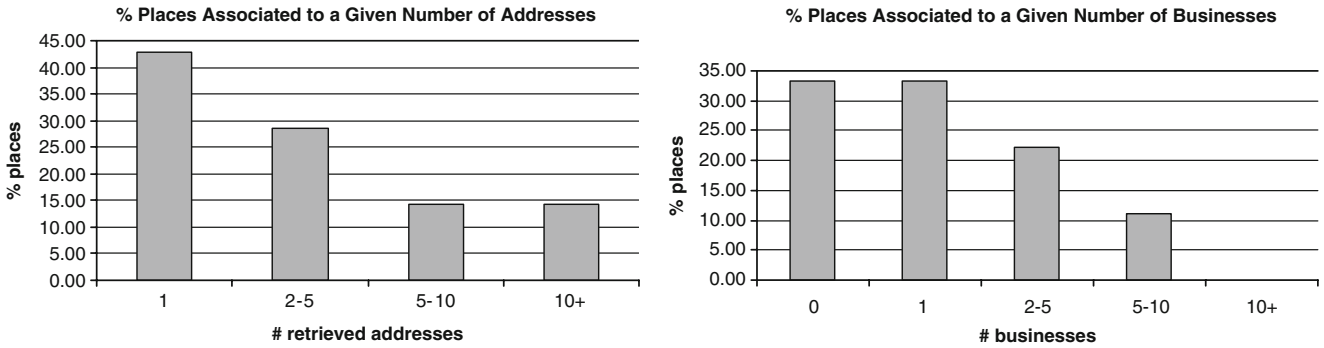

Figure 16 Percent of relevant places corresponding to a given number of addresses (*left*), and businesses (*right*)

In a third group of experiments, we tried to evaluate the performance of retrieving the businesses performed on a given place via the white-pages service. Comparing the results with the ground-truth annotations, we first tried to determine whether the correct place is retrieved. With disappointment, we verified that the actual place could be retrieved in only 40% of the cases. This is either due to localization or white-pages errors. Moreover, due to the multiplicity of addresses being discovered, several businesses can be assigned to a given place. In Fig. 16, right, we report the distribution of the number of businesses found for a given place. It is easy to see that some addresses are not listed in the white-pages, since there are no places with more than ten retrieved businesses. In addition, It is worth reporting that the number of businesses being retrieved is almost independent of whether the correct place has been found or not. Some places, in fact, return a long list of candidate entries not containing the correct one. The main source of errors of this phase is related to the white-pages interface and how it handles street numbers.

In our future work we will try to improve this result in many directions. First, we will try to integrate our software with commercial spatial databases (e.g., http://www.tomtom.com/pro). This would notably improve the coverage of the addresses and businesses being mapped. Second, we will try to implement more advanced localization algorithms in order to reduce the uncertainty about user location [13]. Finally, we will try to embed commonsense reasoning (e.g., the user does not go to a disco in the morning) to cut off unlikely possibilities. The last resort would be to ask the user. In uncertain situation the W4 diary could pop up and let the user solve possible ambiguities.

Finally, the last set of experiments verified the results of the Bayesian classification. Overall, our approach classifies the places correctly in 64% of the cases. The confusion matrix for the classes being identified is reported in Fig. 17. It is possible to see that home and work places have better classification performances since their associated temporal pattern of visits is defined more precisely. On the contrary, places like pubs, restaurants and discos have a more flexible pattern of visits and thus they are classified less precisely.

5 Related work

In recent years, several models addressing contextual information and context-aware services have been investigated, and several algorithms to extract high-level information from location data have been proposed. In this section, we discuss some relevant proposals in these areas: First we present related work in models to represent context information, second we present related works concerning algorithms to extract information from context data that are suitable to create diary like application.

However, in general, we want to emphasize that a strength of our approach is to develop both these two aspect together in order to let them take advantage of each other.

5.1 Related context models

A first group of research 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. [28] concentrates on the acquisition of context data from sensors and the processing of this raw data through a layered model. Similarly, the

	Home	Work	Restaurant	Pub	Disco
Home	0.75	0	0.25	0	0
Work	0	0.66	0.33	0	0
Restaurant	0	0	0.5	0.5	0
Pub	0.1	0.2	0	0.6	0.1
Disco	0	0	0	0.5	0.5

Figure 17 Confusion matrix for Bayesian network classification: horizontal labels are ground-truth information, vertical labels are classification labels

Context Toolkit [8] focuses upon deriving context from raw data by providing abstract components that can be connected together to capture and process data from sensors. Although powerful, these approaches lack of a common semantic to describe the data. This forces 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 research focuses on developing context models that can be easily queried. The work proposed by Schilist et al. [29] creates a simple context model in which information are maintained by a set of environment variables that can be accessed in a flexible way. Analogously, Henricksen et al. in [11] analyze context by adding several features such as the temporal aspects, information imperfections, 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 while 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 [14] is based on context tuples each describing a single piece of context data in terms of entities (people, places, things), attributes (e.g. the name) and relationships, special kinds of attributes that reference to other entities. Similarly, Egospaces [15] 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 called "view" where every agent holds a personal representation of the world. 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 context information trying to overcome this problem.

Finally, a very interesting proposal is presented in [31]. This work adopts in a seven-field data structure to describe the context. The fields are: subject, predicate, object, time, area, certainty and freshness, with similar meaning to W4. Beyond the fields meaning, the purpose is different: their context model is not for pervasive computing application, but for managing the consistency between data from multiple sources. Similar considerations apply for the system described in [6]. 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.

5.2 Related place extraction algorithms

The recent availability of affordable localization mechanisms and the recognition of location as a primary source of context information has stimulated a wealth of works addressing topics related to place recognition and identification.

One of the earliest work trying to automatically extract user location and compose a diary of users' whereabouts is the PEPYS application [21]. This application uses IR badges and detectors to track user location in an indoor environment. On the basis of such an information, PEPYS infers where the user has been and submits to the user a log of the location being visited as a memory feedback. This kind of indoor localization systems, as well as its more modern incarnation (e.g., [10]) could complement our algorithms to deal with indoor settings.

The work described in [13] compares three algorithms to cluster continuous GPS readings to find relevant places. However all these algorithms are only useful to spot relevant places and identify possible recurrent visit to the same place while the problem of adding semantic information is completely neglected. The works in [3, 16, 20, 27] present similar clustering algorithms.

The problem of adding semantic tags is posed, at least as an open problem, in [12]. Other than clarifying the importance and the need for such a conversion from "positions" to "places", the author illustrates two viable approaches to add semantics. The first approach is based on labeling places on the basis of the activities performed there. The author proposes using RFID tags to infer users' activities on the basis of the objects being touched (e.g., the user touches a fork and a knife, the system infers he is having dinner). Then, the system uses the activity (e.g., having dinner) to label the place (e.g., restaurant). The second approach involves humans assigning labels to places pro actively, and exchanging such labels among users. Neither of the two approaches has been actually realized, and they are mainly left as future work. In any case, once available, they could be well complement and integrate our proposal.

The works described in [23, 24] adopt a Bayesian network to infer high-level user behaviors from low-level GPS readings. While their approach is similar to ours, their goal is different in fact as the algorithms we presented try to classify the places, theirs try to classify user activities and eventually predict where the user will go next on the basis of his past routes. It is worth to report that some user activities can directly identify the place in which they occur. For example, a sharp step in the user speed can reveal the user started/stopped driving the car. This automatically can be used to label that place as a parking place. In general, however, we plan to use these kinds of algorithms to

complete the "what" field in the W4 model and to extract even more information about the user activities.

An important difference between the above presented and our algorithms is that, by relying on the common general W4 model, our algorithms are more interoperable and can work in a plug-and-play manner. It would be very interesting to create W4 interfaces for the other proposals to let them become interoperable with our models and algorithms.

6 Conclusions and future works

Organizing contextual data for their effective and meaningful exploitation by autonomic pervasive services is, and will increasingly become, a challenging issue. As the technology for producing contextual information is becoming widespread, and as a variety of diverse pervasive services are continuously being proposed, there is need of proper solutions to represent, pruning and organizing data. The W4 model that we have developed, as presented in this paper, proposes itself as an effective model supporting context representation in future pervasive and mobile computing services. In this paper we also presented and evaluated several algorithms to extract high-level information from the W4 tuples describing the user whereabouts. There are several directions to improve our work:

With respect to the W4 model, it would be important to develop more applications to test the model effectiveness in a variety of application scenarios. In particular it would be important to understand what kind of context information can be represented when more W4 tuples are combined together (like in the example of Figs. 5 and 7). Another important research topic is to better define and enrich the model itself. On the one hand, it would be important to describe the content of the W4 fields by means of standard ontologies in order to let the W4 model be usable in open and dynamic scenarios. On the other hand, it would be interesting to define principles and tools to assess the reliability and truth-degree of the W4 information.

With respect to the algorithms, we think that much more information about the places could be retrieved from the Web. This could be very useful especially in the cases in which localization is precise enough to return a single or a couple of addresses. Such kind of retrieved information could be a precious source of information to estimate the user profile. Commonsense data could be exploited to effectively discriminate among several candidate places [17, 18]. For example, if a person went to a restaurant at noon, it is very unlikely that he will go to another restaurant at 2 pm. Other kind of sensing devices and algorithms could be employed to extract more information about the place. Moreover, some GPS clustering techniques that have been used in related works [13] could improve the performance of our implementation.

In conclusion, we think that the combined development of the context model and the information extraction algorithms, guided by the W4 diary service, is a fruitful way to develop them both and obtain results that can be also generalized to other scenarios.

Acknowledgements Work supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

References

1. Abdelzaher T, Anokwa Y, Boda P, Burke J, Estrin D, Guibas L, Kansal A, Madden S (2007) Mobiscopes for human spaces. IEEE Pervasive Comput 6(2):20–29
2. Ahuja S, Carriero N, Gelernter D (1986) Linda and friends. IEEE Comput 19(8):26–34
3. Ashbrook D, Starner T (2003) Using GPS to learn significant locations and predict movement across multiple users. Personal Ubiquitous Comput 7(1):275–286
4. Bannach D, Lukowicz P, Amft O (2008) Rapid prototyping of activity recognition applications. IEEE Pervasive Comput 7 (2):22–31
5. Bicocchi N, Castelli G, Mamei M, Rosi A, Zambonelli F (2008) Supporting location-aware services for mobile users with the whereabouts diary. Int Conf Mobile Wireless Middleware, Operating Syst Appl, Innsbruck, Austria
6. Bravo J, Hervas R, Chavira G, Nava S (2006) Modeling contexts by RFID-sensor fusion. Conf Pervasive Comput Commun Workshops, Pisa, Italy
7. Castelli G, Rosi A, Mamei M, Zambonelli F (2007) A simple model and infrastructure for context-aware browsing of the world. Int Conf Pervasive Comput Commun, White Plains, NY, USA
8. Dey AK, Abowd GD, Salber D (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum-Comput Interact 16(2–4):97–166
9. Fitzgerald M (2004) Your digital wallet. MIT Technology Review, 24 August
10. Hahnel D, Burgard W, Fox D, Fishkin K, Philipose M (2004) Mapping and localization with RFID technology. IEEE Int Conf Robotics and Autom, New Orleans (LA), USA
11. Henricksen K, Indulska J, Rakotonirainy A (2006) Developing context-aware pervasive computing applications: Models and approach. J Perv Mobile Comput 2(1):37–64
12. Hightower J (2003) From position to place. Workshop on Location-Aware Computing, Seattle, WA, USA
13. Hightower J, Consolvo S, LaMarca A, Smith I, Hughes J (2005) Learning and recognizing the places we go. Int Conf on Ubiquitous Comput, Tokyo, Japan
14. Hong J (2002) The context fabric: An infrastructure for context-aware computing. Conf Comput Human Interaction, Minneapolis, MN, USA
15. Julien C, Roman G (2006) EgoSpaces: Facilitating rapid development of context-aware mobile applications. IEEE Trans Softw Eng 32 (5):281–298
16. Kang J, Welbourne W, Stewart B, Borriello G (2004) Extracting places from traces of locations. Int Workshop on Wireless Mobile Appl Services on WLAN Hotspots, Philadelphia, PA, USA
17. Liu H, Singh P (2004) ConceptNet: a practical commonsense reasoning toolkit. BT Technol J 22(4):211–226

18. Lenat D, Guha RV (1990) Building large knowledge-based systems: Representation and Inference in the Cyc Project. Addison-Wesley, New York
19. Mamei M, Nagpal R (2007) Macro programming through bayesian networks: Distrib Inference Anomaly Detection, Int Conf Pervasive Comput Commun, White Plains, NY, USA
20. Marmasse N, Schmandt C (2000) Location-aware information delivery with commotion. Int Symp Handheld and Ubiquitous Comput, Bristol, UK
21. Newman W, Eldridge M, Lamming M (1991) PEPYS: Generating autobiographies by automatic tracking. European Conf Computer Supported Cooperative Work, Amsterdam, The Netherlands
22. Paskin M, Guestrin C, McFadden J (2005) A robust architecture for inference in sensor networks. Int Symp Information Processing in Sensor Networks, Los Angeles, CA, USA
23. Patterson D, Liao L, Fox D, Kautz H (2003) Inferring high-level behavior from low-level sensors. Int Conf Ubiquitous Comput, Seattle, WA, USA
24. Patterson D, Liao L, Gajos K, Collier M, Livic N, Olson K, Wang S, Fox D, Kautz H (2004) Opportunity knocks: A system to provide cognitive assistance with transportation services. International Conference on Ubiquitous Computing, Nottingham, UK
25. Philipose M, Fishkin K, Perkowitz M, Patterson D, Fox D, Kautz H, Hahnel D (2004) Inferring activities from interactions with objects. IEEE Perv Comput 3(4):50–57
26. Riva O, Borcea C (2007) The urbanet revolution: Sensor power to the people. IEEE Perv Comput 6(2):44–44
27. Schmid F, Richter K (2006) Extracting places from location data streams. International Workshop on Ubiquitous Geographical Information Services, Munster, Germany
28. Schmidt A, Aidoo KA, Takaluoma A, Tuomela U, Van Laerhoven K, Van de Velde W (1999) Advanced interaction in context. Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany
29. Schilit B, Adams N, Want R, Context-aware computing applications. Workshop
30. Twitter: What are you doing? http://Twitter.com
31. Xu C, Cheung SC (2005) Inconsistency detection and resolution for context-aware middleware support. Symposium on the Foundations of Software Engineering, Lisbon, Portugal