2012 IEEE 15th International Conference on Computational Science and Engineering

Aspects of Pervasive Information Management: an Account of the Green Move System

Emanuele Panigati, Angelo Rauseo, Fabio A. Schreiber and Letizia Tanca
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Via Ponzio, 34/5 - 20133 Milan, Italy
Email: {panigati,rauseo,schreibe,tanca}@elet.polimi.it

***Abstract*—The Green Move project aims at realizing a zero-emission-vehicle (ZEV) sharing service that also includes pervasive information distribution. In this paper we discuss the use of context-aware techniques applied to data gathering, shared services, and information distribution; we also discuss how a context-aware approach applied to these tasks leads to the reduction of (noisy) information delivered to the users and to the personalized distribution of information. Privacy of data is also a main concern in the realization of the project, and a privacy-safe approach to information distribution and advertising is adopted. The project, grounded in many results on the use of context-awareness already published by the same authors, aims at building a real-life system based on them. Eventually, we briefly describe the rapid prototype produced and the approach employed so far for the realization of the full system.**

I. Introduction

Nowadays technologies enhance most aspects of everyday life. A technology which is able to seamlessly integrate in our way of living as a part of it becomes pervasive [15]: from biomedical monitoring – e.g. continuous health care – to automotive (e.g. self-driving or assisted-driving vehicles), large numbers of very small sensors and embedded systems participate in processes and data flows through the support of many different technologies. Such a large number of involved entities generate interesting issues about energy consumption, network connections, computation resources and, last but not least, data management. Huge amounts of data, coming from possibly large collections of participating entities, have to be collected, re-distributed and analyzed in a reasonable amount of time, in order to obtain useful and up-to-date information.

Such a scenario is instantiated in the *Green Move (GM)* [1] project, whose aim is a zero-emission-vehicle (ZEV) sharing service for the city of Milan. In Green Move the core services are surrounded by a social-like platform to support users in a large urban context. The ZEV-sharing service provides four different *service configurations*, designed to meet different user-category requirements:

condo-sharing for users who live in apartments and decide to share a (set of) vehicle(s) for daily usage (e.g. going to the supermarket, taking children to school, ...). This configuration is usually *two-ways*: the user returns the vehicle to the same place where he/she got it, typically the condo parking;

firm-sharing for firms outsourcing their company vehicles to the GM sharing service. This configuration is usually *two-ways* like condo-sharing;

world-of-services users use a GM vehicle to reach a point of interest – e.g. a museum, or a department store, which has an agreement with GM – offering dedicated services to GM customers (e.g. having the museum ticket charged on the GM monthly bill to skip the queue at the ticket office). This configuration is typically *one-way*: the user shall release the vehicle at a GM reserved parking nearby the aggregation point, any further usage will be independent: the user could reserve a different vehicle (or the same if it is available) for moving away after having enjoyed the service;

generic users whose needs are not represented in any of the previous configurations.

The GM system also aims at providing a complete user experience of core and accessory services, like integrated services offered by GM commercial partners in the city, service and traffic information and advertising based on users' interests and GPS position. To fulfill such objectives we propose a context-aware approach to realize and manage situation-dependent services and support processing of data flows to extract interesting information accordingly. The approach drives the data flows since its gathering phases, even from sensors, selectively retrieving data only in quantity and format useful according to the current context: e.g. driving downtown is different from driving in the suburbs, thus the user reasonably expects different information – like traffic density or the presence of restricted areas – and with different frequencies.

With a growing number of vehicles and users, the amount of collected and exchanged data will make the efficiency of advanced services a critical issue: in this perspective, the use of selective and efficient context-aware data gathering processes, which filter the information on the basis of the context(s) of its acceptor(s), can certainly improve the effectiveness and scalability of the system.

The paper is organized as follows: we present the data management subsystem of GM in more detail in Section II; a perspective about how context is modeled in our approach is presented in Section III and specific applications of the proposed approach in Section IV. A prototype of the core context-aware functionalities is described in Section V. Conclusions and future work are presented in Section VI.

978-0-7695-4914-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICCSE.2012.93

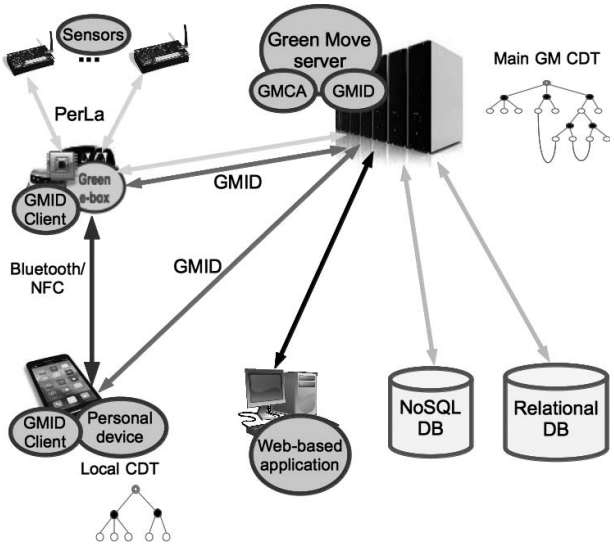

Figure 1. The GM data management system architecture

II. Green Move information management

The architecture of the GM system envisages three main components (see Figure 1): (i) a central platform designed to manage infrastructural aspects and information flows, (ii) on-vehicle components (*Green e-Boxes*) and (iii) the users' personal devices.

The central platform of the system includes GM data and application servers, the main tasks of which are maintaining data in all the needed formats, making the GM web-based application and pervasive information messages and ads available. Thus, the central platform comprises the *GMCA* (GM Context-Aware) and the *GMID* (GM Information Distribution) modules (see Section V) to handle vehicle reservation and assignment, user experience personalization and information distribution in the whole system.

The ICT core of each vehicle connected to the system is known as the *Green e-Box*, an Android-powered board configured with ad-hoc applications integrated within the GM system. By installing the Green e-Box on a vehicle, car manufacturers, companies and private users shall be able to join the system and share their vehicles within the frameworks of the provided service configurations. Each Green e-Box works in association with the GM server: the GM server provides services and data to support security, navigation, traffic and vehicle management, from door unlocking to GPS navigation, while the Green e-Boxes provide local data analysis and an interface to the GM system. In particular, the Green e-Box gathers and possibly pre-processes data from sensors before sending them to the GM server, displays[1] to the users useful information about the trip[2] received from the GMID and acts as local controller for reservation handling.

The user personal device interacts both with the Green e-Box – to handle the vehicle management operations (doors lock/unlock, engine enabling, ...) – using either a a *Bluetooth* or *NFC* connection, and with the GM server by means of the GMID to display useful ads and/or service information.

The system main data store is a relational database which stores all data concerning users, vehicles, Green e-Boxes and accessory information. Highly dynamic data coming from sensors are managed in the system using the PerLa framework (see Section IV-B) and are used for taking immediate actions – like notifying a driver that the battery charge is low or about a traffic jam in the neighbourhood – as well as for subsequent analyses – e.g. for building traffic models. Since sensor data come in streams and transactional *ACID* properties need not be guaranteed, they are stored in a *NoSQL database*, thus allowing fast and effective access.

The conceptual schema in Figure 2 represents the main entities and relations from the GM database. The VEHICLE entity contains data about each vehicle registered with the system and it is related to the GREEN_EBOX entity, which is also responsible for data gathering from sensors, e.g. GPS. User data are collected in the USER entity: from the hierarchy it appears that such entity contains data about both the users that plainly make use of GM services - *customers* - and about users that share their own vehicles through GM - *owners*; however these roles could be not fully disjoint, e.g. a *customer* can also be the *owner* of some GM vehicles. Users are related to their personal devices (`owns` relation) and to vehicles by means of two different relationships *1)* the `owned_by` relationship that connects every *owner* to his/her owned vehicles, *2)* the `makes` relationship, that through the entity `reservation`, connects each user with his own assigned vehicle. Other data about any other kind of service related to a particular reservation are contained in the entity SERVICE and in its related entities.

All the data represented in Figure 2 are stored in the GM relational database, except for the data related to GPS and SENSOR, which use the NoSQL database.

Running example

We refer to the following simple scenario to give some practical examples. "Mr. Guido Verde" registered himself to the GM condo-sharing facility available at his condo, including a parking lot with a recharging station. Once registered, he decided to take full advantage of all services; he specified his data to the system and downloaded the GM application to his smartphone filling the private part of his profile. Besides more occasional usages, Mr. Verde typically uses the electric cars to take his granddaughter to school every morning, and sometimes stops, on the way home, at the supermarket for some shopping. Thanks to the private profile in the GMID client on his smartphone, Mr. Verde can receive interesting traffic and commercial information and ads according to his current context (GM configuration, location, selected interest topic...).

[1]If the Green e-Box is not supplied with a display, this function can be performed by the user personal device.

[2]Except advertisements, that are limited to the user personal device (see Section IV-C)

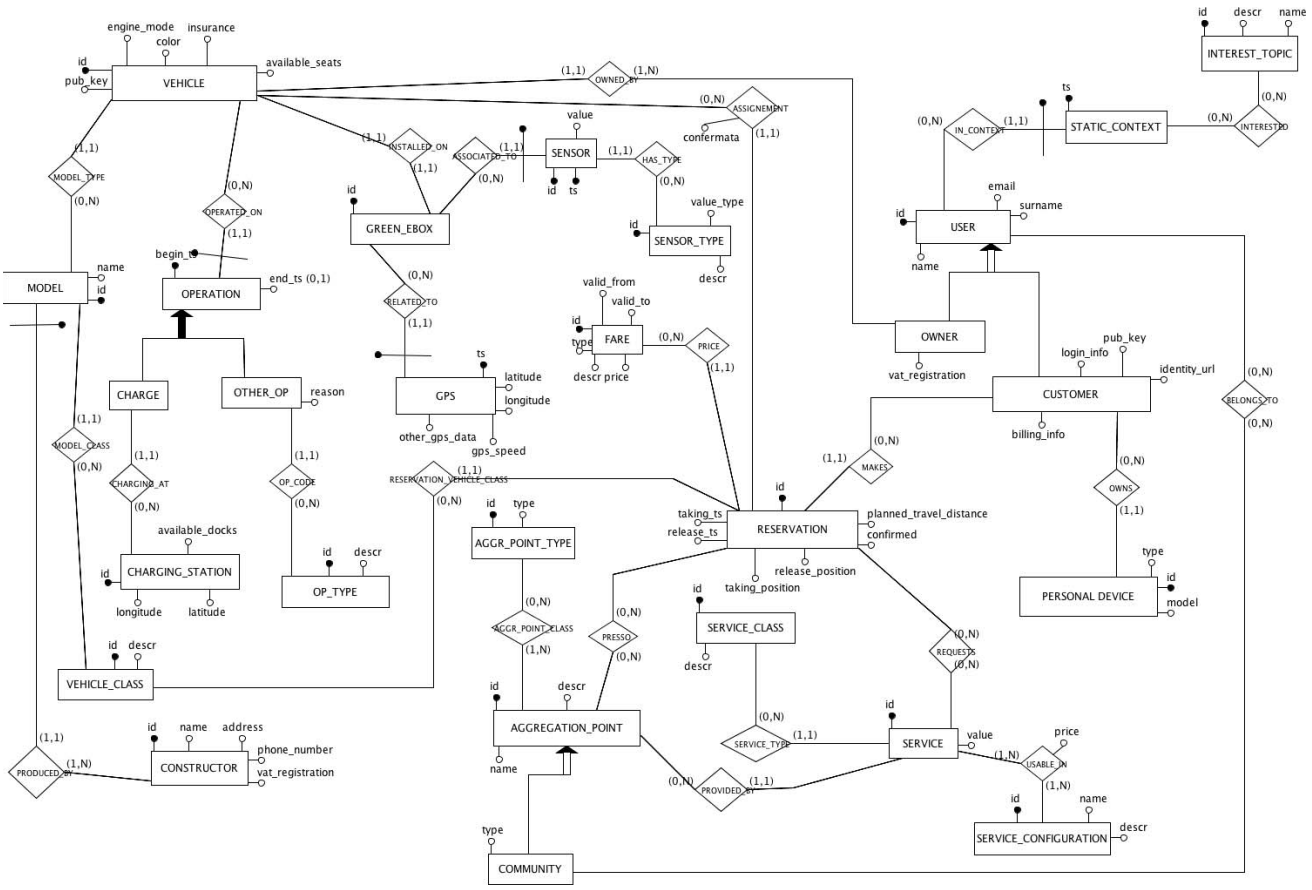

Figure 2. Excerpt from the ER model designed for the GM system

III. Modelling context in Green Move

In GM context is modeled by means of the Context Dimension Tree model (henceforth simply CDT) [3], a powerful and flexible modeling formalism, where contexts [7] are composed starting from an application-dependent set of properties characterizing users, interacting systems and the environment surrounding them. Specific features of the GM scenario have driven some interesting innovations, reported in the following.

A. Context Dimension Tree (CDT)

The CDT of Figure 3 represents the perspectives envisaged to contextualize GM data and to offer personalized car-sharing and information-distribution services (see Section IV-C). A CDT is made of nodes from a set $N = N_D \cup N_V \cup N_{VP} \cup N_{DP}$. Circular black nodes represent envisaged *dimensions* $d_i \in N_D$ for the supported application scenario, i.e. the different perspectives identified by the designer to analyze context. Circular white nodes represent the *values* $v_i \in N_V$ taken by the dimensions; these values can in their turn be analyzed w.r.t. other dimensions, or have *value parameters* $p_i^V \in N_{VP}$ attached to them. A value parameter is represented by a white squared node and allows to refer to specific data: as an example, the value "customer" of the dimension "Role" features the parameter "customerID". Children of a dimension can be values or *dimension parameters* $p_i^D \in N_{DP}$. A dimension parameter is represented by a white double-circle node and acts as a shorthand for representing a high number of possible values of the parent dimension; an instance of a dimension parameter is therefore equivalent to a value.

Context elements ce_i are built starting from CDT nodes; they represent statements of the form *dimension=value*. In particular a value can also contain a parameter (e.g. `customer<IDvalue>`), or can be a dimension parameter (e.g. `ageClass`). From a formal point of view, a context (also called *context instance*) CI is defined as a conjunction of context elements $CI = \bigwedge_{i=1}^{n} ce_i$. An example of context instance is the case of Mr. Verde looking for a cultural aggregation point: here the dimension `Role` takes the value `customer<1101>`, the dimension `Aggregation_type` takes the value `cultural`, the dimension `Vehicle_type` takes the value `car` and the dimension `Service_type` takes the value `condo`.

In the CDT of Figure 3, the dimensions found immediately below the root, called *top dimensions* (Role, Interest_topic, Vehicle_type, Local_conf, Service_type, Time, Location) determine, through their directly attached values, the main per-

spectives for filtering the data in the GM database; their sub-dimensions provide a more detailed specification where needed.

Application constraints, preventing the construction of unwanted context instances, are specified by logical formulas, as defined in [3], [4]. Examples are *useless-context* constraints, used to prevent conjunctions of any number of context elements from being present in a context instance. Binary useless-context constraints can be graphically represented by means of edges between two white nodes.

The main objective of the CDT filtering approach is *data tailoring*, that is, to filter the data according to the different context instances; this is done by means of *contextual views*, i.e. views selecting only the interesting data. The process starts associating one view, called a *partial view*, with each context element. The view associated with a context is then assembled by appropriately composing the partial views associated with its context elements [3], [4], [12].

B. Local CDTs

The Green Move application needs, especially with respect to privacy, triggered an innovation in the CDT context-modeling approach. Indeed, it seems appropriate that the private profile and specific needs and tastes of a Green Move customer should be unknown to the Green Move server, resting within the user personal device. In this case, we need to distribute the context data to different locations, leading to the introduction of a *combined CDT* comprising a *primary CDT* and one or more *local CDTs*.

A local CDT in the Green Move system is maintained locally to user devices and is used to complete the context-based data filtering. For each Green Move customer, the system will compose a specific combined CDT from the primary one, maintained by the server, and the local one, available on the personal device.

The composition of a local CDT with a primary one must comply with the CDT design constraints described in detail in in [3]; In the CDT of Fig. 3, the dimension `Local_conf` has as possible values the roots of the local CDTs. A *combined context* CC of a combined CDT is then easily defined as the conjunction of a context CP of the primary CDT and a context CL of a local CDT, and thus it is nothing more than a conjunction of their context elements (ce):

$$CC = CP \wedge CL = \bigwedge_{i=1}^{n} ce_i \wedge \bigwedge_{h=1}^{m} ce_h = \bigwedge_{k=1}^{n+m} ce_k$$

IV. Context-awareness in Green Move

There are three main tasks for which a context-aware approach is applied in the GM project: (1) producing a personalized user experience, which involves the management of the whole system and the interaction with users, (2) sensors data retrieval and evaluation and (3) information distribution.

Tasks (1) and (2) are performed by the GMCA, while task (3) is the responsibility of the GMID. In the following we explore such tasks and how the GM system realizes them.

A. Personalized User Experience

Due to the user-centered perspective of the GM project, context-aware techniques are used to tailor the user experience against the users' actual context.

Referring to the running example, we follow Mr. Verde, who has just logged into the web interface to the GM system. He is making a reservation for a car to be used the next morning to take his grandchild to school. Since Mr. Verde performs the same reservation every morning, the system is able to guess that he may need a child seat by analyzing the current context and the history of Mr. Verde's previous context instances.

Contextual preferences are used to rank the data and services, according to the interests demonstrated by the users in the different contexts; for instance Mr. Verde will be offered a vehicle with a children's seat whenever he tries to reserve a car for 8:00 in the morning. This analysis is performed automatically by using the *contextual preference-mining* framework (PreMINE) [11], [2]: indeed, while it would be unfeasible to require a user to answer a large set of questions about his/her interest and preferences in each possible context, it is possible to use mining techniques to extract and learn them directly from historical data.

Once Mr. Verde gets into the car and starts driving around the city, the GMID service is able to identify useful information (traffic jams, street works in progress, ...) with respect to the context data fed to the system (e.g. values for location and time dimensions). The pertinent information is provided to the vehicle Green e-Box, to be displayed on its screen (if present) or on Mr. Verde's personal device, his smartphone, running the client.

If Mr. Verde has already set up his private contextual data on his personal device, the GMID service can send him ads about possibly interesting offers (e.g., since he goes shopping, grocery items on offer or a special sale of vegetables) and other useful information according to his interests. The ranking among interesting and not-interesting information is based on the local context at hand and the matching is performed directly on the the user device in order to preserve privacy.

B. Context-aware Sensors

Available bandwidth in mobile systems is considered a scarcely-relevant issue due to the advent of powerful mobile communication networks. Unfortunately, mobile networks can have quality and/or overloading issues, resulting in decreased net throughput. In a complex car-sharing system with social interactions like GM many information flows have to be managed and the issue needs attention.

Since frequent data transmission is the most energy-consuming operation and can bring to network congestion, operations on the sensed data (e.g. data aggregation) can be performed locally on the sensing nodes, which can send larger packets at lower frequency, instead of small sets of possibly redundant values [5]. However timeliness constraints might be strong and, in this case, the transmission protocol should ensure a good compromise for a proper real-time behavior

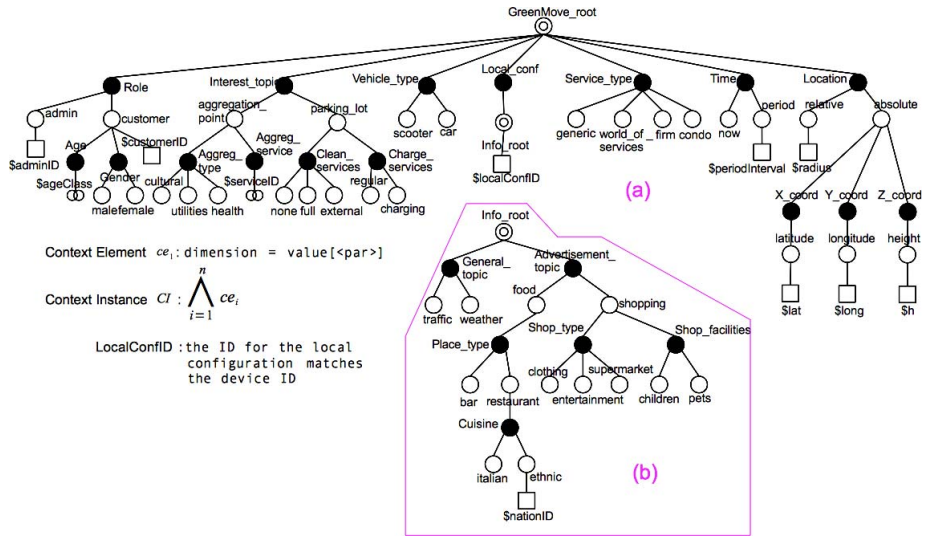

Figure 3. The primary (a) and local (b) CDTs designed for the GM project

of the system (e.g. some key data about road events should always be transmitted as soon as available).

To manage the data produced by sensors, we use the *PerLa* (*Per*vasive *La*nguage) framework [13]: *PerLa* supports locally managed operations on data in a finely controllable and tunable fashion: the framework provides a declarative SQL-like language and a middleware infrastructure suitable for collecting data from different nodes of a pervasive system made of sensors as well as all kinds of generic data-gathering peripherals. The management of the gathered data is performed hiding the complexity of the possibly high heterogeneity of the underlying devices, which can span from RFID(s) to ad-hoc sensor boards or even portable computers and smartphones. Moreover, PerLa supports context-awareness for sensors [14] and can be integrated in a general context-aware system based on the CDT framework.

In our scenario, the data gathering process starts from the moment Mr. Verde unlocks the doors of the assigned vehicle and continues until he gets out of the vehicle releasing it and making it available for the next reservation (the data gathering process restarts for the next user). The whole process is context-mediated by means of PerLa, collecting only data useful for the current user and vehicle context. The data gathered locally from sensors on the vehicle (GPS position, speed, actual power consumption, ...) are pre-processed by the Green e-Box (on which runs a PerLa module) and part of the computation (possibly aggregation) is done by this component. From Green e-Boxes data are pushed to the GM server using the PerLa middleware infrastructure for further processing and storage.

The system information distribution module (*GMID*) works in a synchronous way: it receives a requests from a client, containing its GPS position, and sends back useful data filtered on this position to the client. Different user contexts need different information retrieval frequencies also in order to avoid congestion of the transmission channel; therefore, PerLa also feeds sensors data to GMID. Moreover, the PerLa context language helps us by allowing different settings in different contexts (e.g. different sampling frequencies).

In order to show how to use the PerLa context language, we introduce the following example. After declaring the contexts as described in [14], PerLa allows the user to declare the activities that the system must perform at run-time when these contexts become active.

For instance, Mr. Verde can drive in two different zones of the city: downtown, where battery charging stations are close to each other, or in the suburbs, where they are located farther away. Whether Mr. Verde is driving downtown or not is detected by his GPS position: if the city center (identified by its GPS coordinates) is farther than a predefined distance *max_distance* from the user actual GPS position then he is driving in the suburbs, otherwise he is driving downtown. To give him the needed information, we define in Listing 1 and 2 two different contexts:

Driving in the suburbs (Listing 1) this context will be enabled only if this precondition is true (ACTIVE IF clause); in this case, the system will sample GPS position and battery charge every 60 seconds if the battery charge is $<= 50\%$ (SAMPLING EVERY ... WHERE clause), only if the vehicle is moving and the sensor provides GPS, speed and battery charge data (EXECUTE IF clause); if the battery charge is $<= 35\%$ an alarm is set (SET PARAMETER ... WHERE clause) and thus the system will display the nearest charging station.

Driving downtown (Listing 2) if this context is enabled, the system will sample GPS position and battery charge every 120 seconds if the battery charge is $<= 50\%$, only if the vehicle is moving and the sensor provides GPS, speed and battery charge data; if the battery charge is $<= 35\%$ an alarm is set and thus the system will display the nearest

charging station.

Furthermore, in both contexts the system checks every 5 minutes if a context switch is necessary (REFRESH EVERY clause), modifying the sampling frequency as a consequence.

Listing 1. Suburb context

```
CREATE CONTEXT Suburbs_Driving
 ACTIVE IF lat > center_lat + max_dist
  AND long > center_long + max_dist
ON_ENABLE:
 SELECT lat, long, batt_charge
 SAMPLING EVERY 60 s WHERE batt_charge <= 0.5
 EXECUTE IF EXIST lat, long, speed, batt_charge
  AND speed > 0
 SET PARAMETER 'alarm' = TRUE
  WHERE batt_charge <= 0.35;
ON_DISABLE:
 DROP Suburbs_Driving;
 SET PARAMETER 'alarm' = FALSE;
REFRESH EVERY 5 m;
```

Listing 2. Downtown context

```
CREATE CONTEXT Downtown_Driving
 ACTIVE IF lat <= center_lat + max_dist
  AND long <= center_long + max_dist
ON_ENABLE:
 SELECT lat, long, batt_charge
 SAMPLING EVERY 120 s WHERE batt_charge <= 0.5
 EXECUTE IF EXIST lat, long, speed, batt_charge
  AND speed > 0s
 SET PARAMETER 'alarm' = TRUE
  WHERE batt_charge <= 0.35;
ON_DISABLE:
 DROP Downtown_Driving;
 SET PARAMETER 'alarm' = FALSE;
REFRESH EVERY 5 m;
```

Do note how the computation can be distributed among the system components: all the processing involving battery charge is executed locally to the *Green E-box*, sending data to the GM *server* if and only if all the required conditions are satisfied ($speed > 0$ AND $batt\_charge <= 0.35$) preserving battery charge that might be wasted in frequent, unnecessary transmissions. The results of the PerLa queries are also used by the prototype described in Section V to retrieve data from sensors, whenever needed.

C. Information Distribution

To tailor and distribute information coherent with users' whereabouts and interests we need a powerful and customizable, yet privacy-safe, distribution service: the GMID. To realize such aim we adopt the *PervAds* framework [6], which, in its original terms, defines a pervasive and privacy-respectful approach to advertising. The framework has been customized to obtain a general distribution channel retaining key privacy aspects. The distribution service provides messages, that are *service messages* or *ads*. In PervAds privacy control remains (literally) with the user of the system: (sub)contexts composed from the local-CDT elements remain on the user devices and are used to filter locally the data that arrive from the GM server.

In general, the distribution process comprises three steps:

1) on the central server the GMID system performs a *pre-filtering step* of interesting messages for the client using the part of context obtained from the primary CDT (e.g. age, gender, time and distance among client GPS position and ad/message geolocalized descriptor);
2) the set of pre-filtered messages is sent to the client (e.g. user's personal device), which performs the *filtering step* – the private part of the matching – using configured interest topics (local CDT context);
3) finally, the client displays only the subset of the received messages matching the local CDT criteria: overall, the information has been filtered according to the combined CDT.

The message (e.g. an ad or traffic data) is composed of three parts: *i)* a short caption, *ii)* an (optional) image and *iii)* a data structure (e.g. an XML-like file) describing the topics related to this specific ad or information (chosen among the ones described in the local CDT). The party who wants to broadcast a context-aware message (e.g. a shopkeeper or a municipal traffic-information service) simply uploads it to an appropriately conceived GM web page, and provides metadata about time duration, geospatial information and other possible topics, chosen among the ones mentioned in the CDT (b) in Figure 3 and displayed by the GM advertisers interface. Interest topics help to tailor visualizations of the ads over the customer base, in order to target each advertisement campaign on a set of specific, possibly interested users.

Resuming the running scenario, Mr. Verde has configured his local client selecting the local context, e.g. personal interest topics among the ones described in the local CDT of his user category, like "I am interested in Chinese restaurants" ({`cuisine = ethnic<Chinese>`}) or "I am interested in cloth shops which offer children assistance" ({`Shop_type = clothing, Shop_facilities = children`}) or "I am interested in traffic news" ({`General_topic = traffic`}). The matching between the users' local contexts (note that these data are not stored in the main GM database, but into a structure local to the client) and message metadata is done on the user client, without sending any personally identifiable information to the GM server: the GMID service is only allowed to pre-filter messages on the basis of the primary context, and send them over by means of anonymous data associated with client devices. In our running scenario, examples of the content of the primary context are user GPS position and time information.

V. Rapid prototype

We have implemented a rapid-development prototype based on the GM specifications for data management. The main implementation language is Answer Set Programming [9] (ASP), in particular the DLV system [10]. We apply the techniques described in [12] to implement *(i)* context management and data tailoring for database access, *(ii)* modeling the vehicle reservation system and using contextualized historical data to assign and make forecasts about the availability of the vehicles in a given time interval. As seen in Section IV-B, context management at the level of sensors is directly made by PerLa.

Coherently with the system architecture described in Section II, the prototype is composed of a main GM context-aware component, the *GMCA*, and an information distribution component, the *GMID*.

A. Context-aware data management component (GMCA)

This component supervises all the sharing services provided through the system. For instance, the reservation process provides functionalities like checking available vehicles, foreseeing to vehicles' scheduling, proposing additional services to the users, all taking into account their actual context and historical contextual data.

The GMCA component has been divided into six ASP modules, each dedicated to a specific task:

CDT definition represents the CDT in ASP (see an excerpt in Listing 3)

Application constraint definition through which the designer limits possible context instances excluding context-element combinations

Database access : using Open DataBase Connectivity (ODBC) [8], the DLV program accesses the GM database reported in Figure 2 to build contextual views of the data

Partial-view definition allows, at design time, to associate each context element with the related portion of data

Context-instance recognition is based on context data captured from the user, sensors and mining historical data

Run time contextual-view generation combines the partial views previously associated with the context elements which compose the current context.

At run time, the system detects sensor values (e.g. the position, from GPS coordinates) and collects other contextual information from the user (e.g. current interest topic); this information gives values to CDT dimensions, and thus generates the corresponding context elements. The logic program that encodes the definition, properties and constraints of the CDT is run against the collected context elements, generating one or more models which are in fact the current contest instance(s), in the typical ASP style.

Listing 3. Excerpt from the ASP representation of the GM CDT

```
%dimension(Dimension)
dimension(role).
dimension(interest_topic).
dimension(vehicle_type). [...]

%value(Value).
value(greenmove_root).
value(customer).
value(car). [...]

%dim2val(Dimension, Value).
dim2val(role,customer).
dim2val(vehicle_type,scooter).
dim2val(vehicle_type,car). [...]

%val2dim(Value, Dimension).
val2dim(root,role).
val2dim(root,interest_topic).
val2dim(root,vehicle_type). [...]
```

Once the current context(s) have been generated, the ASP program computes the *contextual views* [12]. Each contextual view provides a version of the database appropriately tailored according to the current context. At design time, together with the CDT, the designer has defined as many *partial views* [12] as the context elements from the CDT; each partial view represents the fragment of the original dataset which has been recognized by the designer as interesting for that context element. The contextual view associated with a generic context is then generated at run time by intersecting the partial views defined for its context elements, and presented to the user for further querying, as required.

As an example, context data together with data about vehicles, users and reservations becomes the input of the DLV program which manages the reservation process (henceforth the *reservation system*). This reservation system evaluates all information at its disposal and generates the vehicle reservation schedule; at the end, reservations are fixed and should not be modified without user intervention. In order to do this, the reservation system evaluates all the vehicles' possible states (a vehicle could be *available*, *reserved*, *out of service*, *in charge* or *under maintenance*) and tries to satisfy all reservation requests (part of this prototype code is shown in Listing 4), although sometimes it might not be possible. The system generates all the possible assignments of vehicles to reservations, ordering such alternatives from the best one to the worst one, according to their ability to satisfy all the constraints (like the last constraint reported in Listing 4). At the end of the process, the best alternative is stored in the database, and vehicles are assigned according to this. Note that the final GM system envisages the optimization of vehicle reservations by means of sophisticated operations research algorithms.

Listing 4. Excerpt from the ASP code for reservations handling

```
% Assign a vehicle to a reservation
assignment(VId,ResrvId) v -assignment(VId,ResrvId)
  :- reservation(ResrvId, _, _, _, _, _, _, _),
     vehicle(VId, _).

% If two reservations overlaps,
%two different vehicles must be assigned to them
:- overlaps_resrv(ResrvId1, ResrvId2),
   assignment(VId1, ResrvId1),
   assignment(VId2, ResrvId2), VId1 == VId2.

% A vehicle can be assigned to a
% reservation iff it is not locked by a constraint
:- assignment(VId1, ResrvId),
   not_avail_assignable(VId2, ResrvId),
   VId1 == VId2.

% Only a vehicle must be assigned to a reservation
:- assignment(VId1,ResrvId),
   assignment(VId2, ResrvId),
   VId1 != VId2.

% Try to satisfy all reservations
:~ reservation(ResrvId, User, StartDay,
     StartHour, EndDay, EndHour, _, _),
   not assignment(VId, ResrvId),
   vehicle(VId, _). [1:1]
```

In addition, the reservation system infers from the user's context data a set of possibly useful services to suggest him/her during reservation confirmation.

B. Information distribution component (GMID)

The GMID component takes care of distributing ads and service messages. The architecture of the GMID includes three submodules: (1) a web interface (mgmtGUI) to manage service messages and ads, (2) a dedicated web-service (the GMID

core service) that coordinates the client-server interaction and (3) a client application (client App) that provides local filtering capabilities and an interface to the web-service.

The mgmtGUI and the GMID core service use Java EE technologies, while the information (ads, information messages) storage solution relies on an ad-hoc geolocalized NoSQL DB.

The mgmtGUI (1) provides users (ad designers, shop owners) with a simple interface to manage information messages and ads; it acquires the target interest topics from the CDT to be matched and attaches such metadata to messages and ads for subsequent filtering.

The GMID core service (2) provides the *pre-filtering step* described in Section IV-C, which reduces the data to be transmitted to the user client to a few sets of useful ads and information messages using the primary user context. It knows the combined CDT and, in particular, is aware of the local CDT structure that will be used to perform the last filtering on the client. The GMID core service will use contextual views generated using the already described ASP logic program and the information about its messages and ads to perform the pre-filtering step (see Section IV-C).

The client App (3) is an Android application that coordinates both the user personal device (e.g. a smartphone, where it has to be installed) and the Green e-Box (where it is provided by default) overseeing the interaction with the GMID core service. During the first access, the application must be initialized by asking the user to choose among the possible interest categories on the local CDT and if traffic information has to be displayed. After the initialization, the client App will begin interaction with the GMID core service, periodically updating its reference data with a unique ID different from the userID; once the reference data have been received, the GMID core starts serving the client. The set of selected information messages and ads is sent to the client App, which performs the final filtering based on the local context and displays only the most interesting ones to the user, as described in Section IV-C.

VI. Conclusion and Future Work

Urban mobility and energy saving are among the driving factors in future smart-city planning and development. The Green Move project aims at realizing a zero-emission-vehicle (ZEV) sharing service, that also includes ubiquitous information distribution in order to assist the driver both in driving and in personal activities. In this paper we gave an account of its context-and-preference-aware information collection and dissemination service based on the Context Dimension Tree for context-aware data management, on the PerLa sensor language, and on the personal advertising platform PervAds, which allow to provide the right information to the right person at the right moment and place. The generality of the used platforms has allowed rapid prototyping of the main system components, and will foster a rapid transformation from the prototype state to the full system. The prototype will be installed and tried on the GM fleet of electric vehicles in the city of Milan from next September.

VII. Acknowledgments

This research is funded by the Regione Lombardia project Green Move. The project is also partially supported by the European Commission, Programme IDEAS-ERC, Project 227977-SMScom and by the Industria 2015 project SENSORI.

We would like to thank all the researchers of Politecnico di Milano involved in the Green Move project, and especially Angelo Morzenti and Matteo Rossi. Moreover, we thank Davide Martinenghi, Giorgio Orsi and Lorenzo Carrara for their support during the design of the *Green Move* context-aware subsystem and of the PervAds framework.

References

[1] G. Alli et al. Green Move: towards next generation sustainable smartphone-based vehicle sharing. In *Proc. of SustainIT 2012*, 2012. To appear.

[2] D. Beretta, E. Quintarelli, and E. Rabosio. Mining context-aware preferences on relational and sensor data. In F. Morvan, A. M. Tjoa, and R. Wagner, editors, *DEXA Workshops*, pages 116–120. IEEE Computer Society, 2011.

[3] C. Bolchini, C. A. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca. And what can context do for data? *Commun. ACM*, 52(11):136–140, 2009.

[4] C. Bolchini, E. Quintarelli, and L. Tanca. CARVE: Context-aware automatic view definition over relational databases. *Information Systems*, Accepted manuscript (unedited version) available online: 12-MAY-2012, 2012.

[5] C. Cappiello and F. A. Schreiber. Quality- and energy-aware data compression by aggregation in wsn data streams. In *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '09, pages 1–6, Washington, DC, USA, 2009. IEEE Computer Society.

[6] L. Carrara and G. Orsi. A new perspective in pervasive advertising. Technical report, Department of Computer Science, University of Oxford, July 2011.

[7] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[8] K. Geiger. *Inside ODBC*. Microsoft Press, Redmond, WA, USA, 1995.

[9] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[10] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, 2006.

[11] A. Miele, E. Quintarelli, and L. Tanca. A methodology for preference-based personalization of contextual data. In M. L. Kersten, B. Novikov, J. Teubner, V. Polutin, and S. Manegold, editors, *EDBT*, volume 360 of *ACM International Conference Proceeding Series*, pages 287–298. ACM, 2009.

[12] A. Rauseo, D. Martinenghi, and L. Tanca. Context-aware data tailoring through answer set programming. In G. Mecca and S. Greco, editors, *Proceedings of the 19th Italian Symposium on Advanced Database Systems*, pages 131–138, June 2011.

[13] F. Schreiber, R. Camplani, M. Fortunato, M. Marelli, and G. Rota. PerLa: A language and middleware architecture for data management and integration in pervasive information systems. *IEEE Transactions on Software Engineering*, 38(2):478 –496, march-april 2012.

[14] F. Schreiber, L. Tanca, R. Camplani, and D. Viganó. Towards autonomic pervasive systems: the PerLa context language. In *Proc. of the 6th International Workshop on Networking Meets Databases Co-located with SIGMOD 2011*, June 2011.

[15] M. Wieser. The computer for the 21st century. *Scientific American*, 265:94–104, September 1991.