Pervasive Data Management in the Green Move System: a Progress Report*

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
{panigati, rauseo, schreiber, tanca}@elet.polimi.it

Discussion Paper

Abstract. We discuss the use of context-aware and pervasive techniques applied to data gathering, shared services, and information distribution for a vehicle sharing system in the *Green Move* project.

1 Introduction

Pervasive systems, in which a large number of heterogeneous entities are involved, generate interesting issues about energy consumption, network connections, computation resources and, last but not least, data management. Huge amounts of heterogeneous data 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.

The GM system 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 filtering 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.

The paper is organized as follows: we present the data management subsystem of GM in more detail in Section 2; in Section 3 is presented a quick idea about how context is modeled in our approach and specific applications of it. A rapid prototype is described in Section 4. Conclusions and future work are presented in Section 5.

* 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 in the related researches on context awareness.

2 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 ap-

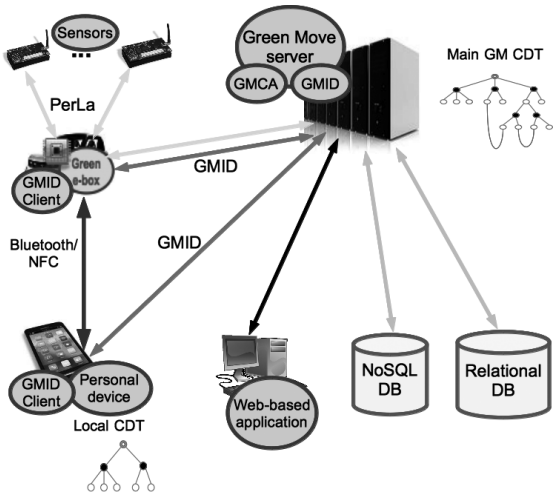

Fig. 1. The GM data management system architecture

plication 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 4) 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. 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 to the users useful information about the trip 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 *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 dy-

namic data coming from sensors are managed in the system using the PerLa framework (see Section 3) 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 Table1 represents the main entities and relations from the GM database; all the data represented are stored in the GM relational database, except for the data related to GPS and SENSOR, which use the NoSQL database.

VEHICLE(id, seats_number, insurance, pub_key, engine_type, *model*, *owner*)
USER(id, name, surname, birthdate, gender, email, pub_key, ident_url, username, passwd, VAT_info, billing_info, is_owner, is_customer)
GREEN_EBOX(id, *vehicle_id*)
GPS(ts, *gb_id*, latitude, longitude, gps_speed, n_satellites)
RESERVATION(id, taking_ts, release_ts, taking_position, release_position, *vehicle_class*, *fare*, confirmed, planned_travel_dist, *service_conf*, *user_id*)
ASSIGNMENT(*reservation_id*, *vehicle_id*, confirmed)
SENSOR(id, ts, *gb_id*, sensor_type, value)

Table 1. Main tables in the Green Move database

Running example: We refer to the following simple scenario: "Mr. Guido Verde" registered himself to the GM car 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 grandaughter 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, ...).

3 Context awareness in Green Move

In GM context is modeled by means of the Context Dimension Tree model (CDT) [2] Specific features of the GM scenario have driven some interesting innovations to the original CDT structure, reported in the following.

The CDT of Figure 2 represents the perspectives envisaged to contextualize GM data (see Section 3). As shown in the figure, a CDT is composed by three basic elements: *1)* a *root node*, *2)* a set of *dimension nodes* (black nodes) that represent the dimensions of interest in a given scenario and *3)* a set of *value nodes* (white nodes) which represent the values for each given dimension. Black and white nodes alternate in each level of the tree.

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`). Parameters are used to represent values where are too many to be represented as nodes of the tree and drawn as squares (value nodes) or double

circles (dimension nodes). From a formal point of view, a context (also called *context instance*) CI is defined as a conjunction of context elements $CI = \bigwedge_{i=1}^{n} ce_i$.

The main objective of the CDT filtering approach is *data tailoring*, that is, to filter the data according to the different context instances [2,3,4].

In the Green Move application, it seems appropriate that the user private profile and specific needs should rest 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* – managed by the GM server – and one or more (personal) *local CDTs*, managed by users on their devices.

In the CDT of Fig. 2, 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):

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

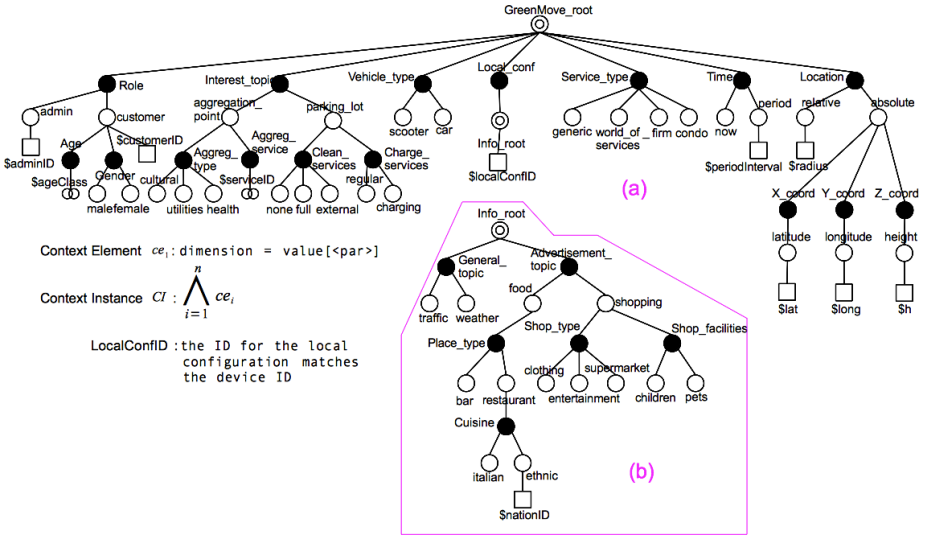

Fig. 2. The primary (a) and local (b) CDTs designed for the GM project

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) sensor 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.

Personalized User Experience: 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) [5], mining techniques are used to extract and learn preferences 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.

Context-aware Sensors: To manage the data produced by sensors, we use the *PerLa* (*Per*vasive *La*nguage) framework [6] which supports locally managed operations on data in a finely controllable and tunable fashion. Moreover, PerLa supports context-awareness for sensors [7] 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 (e.g aggregated) by the Green e-Box (on which runs a PerLa module) and 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). After declaring the con-

texts as described in [7], PerLa allows the user to declare the activities that the system must perform at run-time when these contexts become active, to give him the needed information. 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. When Mr. Verde is driving downtown the context "*driving in dowtown*" is active (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. Otherwise, if the city center (identified by its GPS coordinates) is farther than a predefined distance *max_distance* from Mr. Verde actual GPS position then the "*driving in the suburbs*" 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. 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.

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 4 to retrieve data from sensors, whenever needed.

Information Distribution: The GMID adopts the *PervAds* framework [8], which, in its original terms, defines a pervasive and privacy-respectful approach to advertising. The distribution service provides messages, that are *service messages* or *ads*. As mentioned before in this Section, privacy control remains (literally) with the user of the system. 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 message simply uploads it on the GM server that will be responsible of its dispatching.

Listing 1.1. 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 1.2. 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;
```

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 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 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 such as user GPS position and time information, and send them over by means of anonymous data associated with client devices.

4 Rapid prototype of the GM context-aware component

We have implemented a rapid-development prototype based on the GM specifications for data management, in Answer Set Programming (ASP), based on DLV system [9]. We apply the techniques described in [4] 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 3, context management at the level of sensors is directly made by PerLa.

As described in Section 2, the prototype is composed of a main GM context-aware component, the *GMCA*, and an information distribution component, the *GMID*.

Context-aware data management component (GMCA): This component supervises all the sharing services provided through the system. For instance, checking available vehicles, foreseeing to vehicles' scheduling, proposing additional services to the users, all taking into account their actual context and historical contextual data.

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 the 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.

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

%value(Value).                     %val2dim(Value, Dimension).
value(greenmove_root).             val2dim(root,role).
value(customer).                   val2dim(root,interest_topic).
value(car). [...]                  val2dim(root,vehicle_type). [...]
```

Listing 1.3. Excerpt from the ASP representation of the GM CDT

Once the current context(s) have been generated, the ASP program computes the *contextual views* [4]. Each contextual view provides a version of the database appropriately tailored according to the current context.

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. This reservation system evaluates context-awares all information at its disposal and generates the vehicle reservation schedule; 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 (*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 1.4). The system generates all the possible assignments of vehicles to reservations requests, 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 1.4). 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.

```
% 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.

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

Listing 1.4. Excerpt from the ASP code for reservations handling

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.

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 simple web interface (mgmtGUI) allowing users (ad designers, shop owners) information messages and ads related information, *(2)* a dedicated web-service (the GMID core service) that coordinates the client-server interaction providing the *pre-filtering step* described in Section 3 by means of the already described ASP logic program and *(3)* an Android client 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. After being configured with the user private profile, the application send its requests to the GMID server, waiting for the set of answers; then the server sends the set of pre-selected information messages and ads to the client App, which performs the final filtering based on the local context and displays the most interesting messages to the user.

(1) and (2) are based on Java EE technologies, while the information storage solution required by (3) relies on an ad-hoc geolocalized NoSQL DB.

5 Conclusion and Future Work

In this paper we gave an account of 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 for the GM project which aims at realizing a zero-emission-vehicle (ZEV) sharing service. 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.

References

1. Panigati, E., Rauseo, A., Schreiber, F.A., Tanca, L.: Aspects of pervasive information management: An account of the green move system. In: CSE, IEEE Computer Society (2012) 648–655
2. Bolchini, C., Curino, C.A., Orsi, G., Quintarelli, E., Rossato, R., Schreiber, F.A., Tanca, L.: And what can context do for data? Commun. ACM **52**(11) (2009) 136–140
3. Bolchini, C., Quintarelli, E., Tanca, L.: CARVE: Context-aware automatic view definition over relational databases. Information Systems **Accepted manuscript (unedited version) available online: 12-MAY-2012** (2012)
4. Rauseo, A., Martinenghi, D., Tanca, L.: Context-aware data tailoring through answer set programming. In Mecca, G., Greco, S., eds.: Proceedings of the 19th Italian Symposium on Advanced Database Systems. (June 2011) 131–138
5. Miele, A., Quintarelli, E., Rabosio, E., Tanca, L.: A data-mining approach to preference-based data ranking founded on contextual information. Inf. Syst. **38**(4) (2013) 524–544
6. Schreiber, F., Camplani, R., Fortunato, M., Marelli, M., Rota, G.: PerLa: A language and middleware architecture for data management and integration in pervasive information systems. IEEE Transactions on Software Engineering **38**(2) (march-april 2012) 478 –496
7. Schreiber, F., Tanca, L., Camplani, R., Viganó, D.: Pushing context-awareness down to the core: More flexibility for the PerLa language. In: PersDB/VLDB 2012, Istanbul (2012) 1–6
8. Carrara, L., Orsi, G.: A new perspective in pervasive advertising. Technical report, Department of Computer Science, University of Oxford (July 2011)
9. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Trans. Comput. Logic **7**(3) (2006) 499–562