Green Move: a platform for highly configurable, heterogeneous electric vehicle sharing

Andrea G. Bianchessi, Gianpaolo Cugola, Simone Formentin, Angelo Morzenti, Carlo Ongini, Emanuele Panigati, Matteo Rossi*, Sergio M. Savaresi, Fabio A. Schreiber, Letizia Tanca and Edoardo G. Vannutelli Depoli

***Abstract*—Vehicle sharing in urban areas has the potential to be the answer to some of the main issues that hinder the spreading of electric vehicles, in particular for what concerns the high upfront costs of the vehicles, combined with their still limited range, which can induce phenomena such as *range anxiety*. For its potential to be realized, vehicle sharing must be tailored to the multiform needs of its users by offering a wide range of support services that can be selected based on the user preferences. In this paper we present the platform for vehicle sharing developed in the Green Move project, which allows services to be dynamically loaded and unloaded on vehicles, and describe a pair of prototype applications to illustrate its benefits.**

I. Introduction

Electric vehicle sharing has the potential to provide a solution to many of the problems that afflict urban mobility.

Pollution, greenhouse gas emissions are some of the most prominent issues that modern metropolitan areas have to face nowadays: the goals set at the political level in this regard – for example, in 2007, the EU Council stated that, by 2050, developed countries should reduce their overall greenhouse gas emissions by 60-80% compared to 1990 – cannot be achieved without a significant re-think of urban mobility. This typically starts from a shift in the nature of the vehicles on which urban mobility is based, from traditional ones powered by internal combustion engines, to smaller, lighter Electric Vehicles (EVs), which are almost entirely zero-emissions. The economic viability of EVs, however, is still a major question mark, mostly because of technological issues such as battery life or driving range. Some estimates put the break-even point of a battery pack at 80% of its lifetime; assuming a battery pack has a life of 2000 complete discharge/recharge cycles, and the driving range of the EV with a fully charged battery is 80km, the vehicle should cover at least 128.000 km in its lifetime to be economically viable, as depicted in Fig. 1. In an urban scenario, where a single driver covers at most 20km a day, this goal is achievable only if the vehicle is shared among users, thus significantly increasing its usage rate.

A second major problem that plagues urban areas is traffic congestion, and the related issue of scarcity of parking spaces. Vehicle sharing, by reducing the number of vehicles within the city limits, directly helps addressing the latter issue, and often enjoys subsidies from city governments which reduce – or eliminate altogether – parking fees. To curb traffic congestion, instead, municipalities such as London and Milan have introduced so-called "congestion charges" in critical areas; however, vehicles that are electric, shared, or both are usually fully exempt from this charge. Hence, vehicle sharing is incentivized per public policy, which makes it a privileged form of transportation, especially in city centers.

The management of EVs poses peculiar problems, which can be mitigated when the vehicles are part of a coordinated fleet such as a shared one. For example, recharging an EV usually takes hours, during which the corresponding charging station is unavailable. Hence, public charging stations become scarce resources, which require the kind of coordinated management that can be obtained through an ICT-enabled vehicle sharing system. A related issue that negatively affects the attitudes of users towards EVs is the so-called *range anxiety*; that is, the fear that the battery will empty before the destination is reached. In this regard, an innovative, automated vehicle-sharing system can help in two ways: by providing users with constant – but unobtrusive – monitoring of their situation; and by intervening with useful advice on keeping an economic driving style only when this is called for.

For electric vehicle sharing to be part of the solution to the problems of urban mobility, however, it must provide easy access, and address the multiform needs of a wide range of users (e.g. people reaching their workplace, company employees, families with children). This entails that the service should offer as wide a range of vehicles as possible, not only in terms of makers, but also in terms of their kind (2-, 3-, 4-wheeled, 1-2-seats or more, with/without doors, etc.); in addition, it should be highly configurable, to allow for the possibility of closely matching the user preferences with the services offered during the drive.

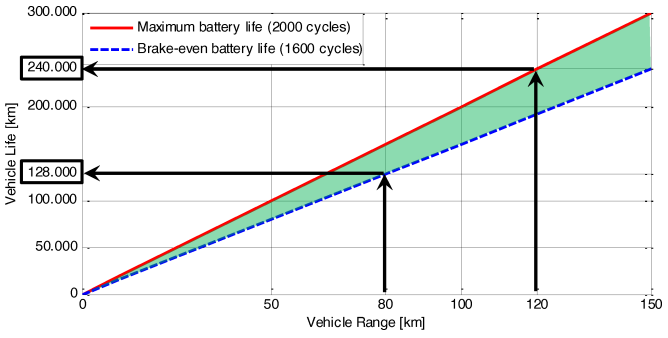


Fig. 1. Trade-off between vehicle range and cost of the battery pack.

The authors are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy. *E-mail: matteo.rossi@polimi.it.

Work supported by Regione Lombardia through project "Green Move".

The Green Move project[1] [1][2] developed a prototype platform for electric vehicle-sharing systems that addresses the issues above in the following way:

- It provides a hardware/software interface, the Green e-Box (GEB), which allows the system to interact with a heterogeneous fleet of vehicles in a uniform way.
- It relies on mobile devices to let users access and interact with the system – take possession/release a reserved vehicle, open/close its doors, enable/disable the drive.
- It uses standard protocols to allow external partner services to access information and operations (e.g., the reservation of vehicles) on the coordination center.
- If offers an infrastructure to customize the software configuration of vehicles by pushing new services on them and by removing existing ones, for example depending on the user preferences.

In this article we focus on the mechanisms developed in the Green Move project to tailor the functions offered by a vehicle to the preferences of its current user. This is achieved through the notion of *dynamic application*, i.e., a piece of software that can be loaded on the GEB of a vehicle at any time, whether it is in use or not, and which extends the functions offered to the user. Dynamic applications can access the data that are present on the vehicle – speed, acceleration, state of charge of the battery, etc. – to create value-added services for the user.

We present a pair of prototype dynamic applications. The first one addresses the issue of *range anxiety* mentioned above, as it gives feedback to the user about her driving style. The second one provides context-dependent advice to the user, such as commercial or cultural suggestions; this application can also be used to coordinate the fleet, for example by notifying drivers of points of interest that are in their proximity, such as available charging stations. The Green Move platform allows these applications to be sent to vehicles depending on their situation. For example, as described in Sect. V-B, the driving style application can be loaded only for users who have indicated, among their preferences, the desire to receive that kind of feedback; or only when the system determines that the user is exhausting the battery while still far from her destination, so a more economic style of driving is necessary to maximize the driving range. Dynamic applications also support a scenario where a function is realized in different ways – for example with different styles for presenting feedback to the driver – and the one that best fits the user preferences is loaded at the beginning of the trip and unloaded at the end. In addition, a mechanism where applications are installed dynamically after the fleet has been deployed allows administrators to remotely perform updates of the functions offered by the GEBs without physically operating on them.

In the next section we briefly survey existing vehicle-sharing systems to highlight their differences with the Green Move platform, in particular with respect to the issues of configurability and interactions of users and vehicles; we then give an overview of the architecture of the Green Move platform, describe the mechanisms underlying Green Move dynamic applications, and present a pair of prototypical ones.

[1]www.greenmove.polimi.it

II. Related services and platforms

Two key areas in which the Green Move project explored innovative solutions are

- *the interaction between user and vehicle*, which in the Green Move approach is key-less and smartphone-based, without requiring a membership card;
- *the configurability of the services offered*, which can be modified after system deployment through the notion of dynamic applications.

As discussed in the rest of this section, more and more commercial car-sharing services are pursuing a key-less approach similar, though not as powerful, to the one pioneered by the Green Move project [1][3]; conversely, none of them has yet reached the level of configurability that the Green Move platform allows.

Using smartphones to de-materialize the interaction between user and vehicle and to dynamically add and modify available services is, especially in the field of electric vehicles [4][5][6], a growing area of interest. *INVERS*[2], *Convadis*[3] and *Eileo*[4] are the global market leaders in the domain of technologies for car-sharing kits. The vehicle-sharing systems built through these kits – which are the majority – have a number of drawbacks and limitations compared to the Green Move platform:

- their configurability is limited;
- they do not support Bluetooth, nor NFC communication for replacing car keys;
- the ownership of the user database and the rights to the user profiles belong to Convadis/Eileo/INVERS rather than to the vehicle-sharing company;
- they do not offer the possibility to add/remove functions depending on the user, location or period.

Some vehicle-sharing systems have recently included concepts similar to those introduced in the Green Move project. For example, *Enjoy*[5] is one of the first systems which does not require a membership card for entering the vehicle. The opening/closing of the vehicle doors is performed through a smartphone app, but the communication between vehicle and user smartphone is always mediated by a central server, where the commands are processed and then forwarded to the vehicle. This procedure requires time, thus the vehicle doors do not open for many seconds (around 30) after the smartphone sends the command. Interested readers can refer to [7] for a thorough description of available car-sharing technologies.

On the contrary, as explained in Section III, the Green Move platform relies on NFC and Bluetooth communication channels between the user smartphone and the vehicle to directly send open/close commands to the latter. Hence, the interaction is much faster – vehicle doors open immediately after the user sends the command through the smartphone; in addition, it does not require a connection to the server, so it can occur also when this is absent, as in underground parkings.

The topic of service configurability is even less explored than smartphone-based interaction: no vehicle-sharing system

[2]www.invers.com
[3]www.convadis.ch
[4]www.eileo.com
[5]enjoy.eni.com

offers the possibility to configure the services of vehicles depending on the user preferences or the state of the vehicle. Some of the most technologically-advanced services offer, through the vehicles' onboard computers, applications that enrich the driving experience. For instance, Car2go[6] offers two onboard services, the navigator and the Driving Style Assistance System. However, no customization is available; the services behave the same regardless of the user's past habits or vehicle type.

If we broaden the scope to on-board applications in general, most OEMs produce systems for In-Vehicle Infotainment (IVI) and telemetry systems for the monitoring of the state of the vehicle. Most telemetry systems are proprietary, based on real-time operating systems; they are typically not integrated with IVI. IVI systems are also usually proprietary, though lately a few OEMs are converging on common solutions. Mobile solutions for hardware platforms and operating systems are the most commonly used basis for building IVI systems. The recently announced iOS-based *CarPlay*[7] IVI by Apple offers the possibility to install third-party applications – though not dynamically – and to interact with the driver's phone to share the data and audio channels; at the moment it is not clear if it offers any access to vehicle data. General Motors has developed a proprietary solution[8], which allows for a closer integration with the vehicle. In particular, it exposes two sets of Application Programming Interfaces (APIs): In-Vehicle APIs and Remote APIs. In-Vehicle APIs allow developers to access a small set of vehicle data, audio/video capabilities, navigation information, user interfaces and communication channels. Remote APIs offer access to the vehicle data and the ability to send commands to the vehicle. Through these commands one can lock/unlock the vehicle doors, retrieve diagnostic information or retrieve the vehicle's position. This system, which will be available on next-generation General Motors vehicles, offers functions similar to those that have been developed for the Green Move platform; however, unlike the latter, it does not offer any capability of dynamically changing the services offered onboard. The *GENIVI Alliance*[9] has developed an open-source infrastructure for in-vehicle infotainment. The *Tizen*[10] software platform, originally born as a mobile operating system, is now shifting its focus to the IVI market through the GENIVI Alliance open-source platform. In all solutions above the focus is on creating an operating system that offers access to multimedia and network functions, and in some cases to vehicle data. Applications developed for these platforms have to be manually installed and started by the user.

OSGi[11] is a service-oriented component-based framework that allows developers to create and manage dynamically loadable applications. The OSGi infrastructure is built upon three basic abstractions: modules, life cycle management and services. A *module* is a single portion of functional code, wrapped in a deployable unit called *bundle*. The *OSGiContainer* provides the bundles' execution environment and primitives to manage their life cycle. Bundles can be dynamically downloaded, installed and started. *Services* are built upon modules. Every module can offer and consume services. OSGiContainer provides standard modules to manage security issues. The OSGi architecture targets generic Java Virtual Machines rather then Android's Dalvik, although Android implementations are available. The Green Move mechanisms for managing dynamic applications, instead, have been specifically designed for vehicle-sharing systems: They allow for the installation of new components without driver intervention, and for the access to vehicle data. In addition, the Green Move APIs for programming dynamic applications are simple and lightweight, but nevertheless they fit different kinds of applications, as Sect. V-VI show.

To summarize, finding new solutions for an optimal management of electric, flexible and heterogeneous fleets is one of the main challenges facing today's urban mobility. The Green Move project has tackled this issue through a platform that allows both a high level of automation in the interaction between users and system, and powerful customizations of onboard services through the notion of dynamic applications.

III. Overview of the Green Move platform

The main elements of the Green Move platform are shown in Fig. 2. They are the Green Move Center, which coordinates the system, the Green e-Boxes, which constitute the interface between the vehicles and the rest of the system, and the users' smartphones, on which the Green Move client app is installed. The Green Move Center (GMC) coordinates the

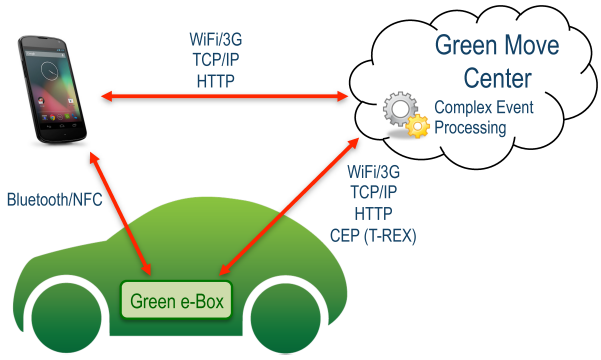


Fig. 2. Schematic view of the overall architecture of the Green Move system.

activities of the Green Move system and offers services such as user and vehicle registration, vehicle reservation/acquisition/release/monitoring, and so on. To retrieve and distribute information among the managed vehicles, the GMC includes the T-Rex Complex Event Processing (CEP) engine [8][9]. In CEP infrastructures [10] we distinguish between event generators (sources) and event consumers (sinks). The former observe primitive events and report about them, while the latter receive event notifications and react to them. Using the nomenclature typical of publish-subscribe systems we say that sources *publish* event notifications (or simply events) and sinks *subscribe* to events. The CEP engine sits in the middle with the task of detecting so-called composite events

[6] www.car2go.com
[7] www.apple.com/ios/carplay
[8] developer.gm.com
[9] www.genivi.org
[10] www.tizen.org
[11] www.osgi.org

from primitive ones through a set of rules – expressed in an ad-hoc language – that are conceived and deployed by rule managers. An example of composite event is "if the vehicle is moving and there is no ongoing rental, then the vehicle is being stolen". The Green Move platform uses CEP technology to both monitor the vehicles and send them configuration commands. For example, Listing 1 of Sect. V shows the T-Rex complex event that is used to realize the driving style scenario described in Sect. I. We have also investigated using the *PerLa* language and middleware [11] next to the T-Rex CEP engine. PerLa offers context management capabilities [12] that can be useful for applications such as the one described in Sect. VI.

Each Green Move vehicle is equipped with a device, the Green e-Box (GEB), through which it interacts with the overall system. From a technical point of view, the only requirement for a vehicle to be introduced in the Green Move system is to have a GEB; this ensures that the interaction between vehicle and system occurs according to standardized protocols. This allows for the possibility of adding to the system vehicles that are heterogeneous not only in their types, but also in their ownership [1]. The GMC communicates with GEBs through 3G channels to manage the fleet. The same channel is used by GEBs to send to the GMC vehicle data – which are distributed at regular intervals as T-Rex events – such as diagnostic information, usage statistics, and trip data (e.g., current GPS position, speed, state of charge). GEBs run the Android operating system, which executes the Green Move vehicle app. This app is the software interface between the vehicle and the rest of the system; in addition to executing core functions such as data retrieval and actuation of commands such as open/close doors, it provides hosting facilities for dynamic applications. Additional details on the architecture and functions performed by the GEB can be found in [2].

The interaction of Green Move users with the system occurs through their smartphones, on which the Green Move client app must be installed. The app communicates with the GMC through a WiFi or 3G channel to reserve vehicles and to retrieve the electronic key that is necessary to access the vehicle. This key is exchanged between the Green Move client app and the GEB through a Bluetooth or NFC channel; it is used to send commands from the user's smartphone to the vehicle, such as open/close the doors (if present), and enable/disable the drive. By using a direct (Bluetooth or NFC) link between the user's smartphone and the GEB, this communication can occur at any time, even when there is no 3G/WiFi connection available between the GEB and the GMC. The same connection can be used by the GEB to send data to the users' smartphones, such as customized advice of various kinds (e.g., commercial).

The architecture outlined above supports the mechanisms described in the next section, which allow administrators to remotely modify the functions installed on GEBs.

IV. Dynamic applications

In the Green Move platform, dynamic applications are bundles of Java code that, packaged into JAR files, can be sent to GEBs, where they are executed within the Green Move Android application running on the GEB. As explained in some detail in [2], a Green Move dynamic application (GMA for short) must implement suitable interfaces, through which it receives access to the resources of the GEB, such as the data collected from the vehicle. Fig. 3 shows the steps through which a GMA is uploaded on a GEB. First (3(a)) a T-Rex event is sent to the GEB, which notifies the latter that a GMA is to be installed; the event contains the address on the GMC where the GMA is to be retrieved. Then, the app running on the GEB accesses the GMC (3(b)) and retrieves the Java code of the GMA (3(c)). Finally, the interfaces implemented by the GMA allow the GEB to instantiate the application and start it (3(d)). Similar mechanisms are used to uninstall a GMA from a GEB, or to reload a running GMA. The GMC gives administrators different options to select the vehicles on which a GMA is to be uploaded: single vehicles (unicast), groups of vehicles (multicast) or the whole fleet (broadcast).

On the GEB side, the component that is in charge of managing GMAs is called GMcontainer, and is shown in Fig. 4. In particular, the GMcontainer listens to the events sent by the GMC; it reacts to them by downloading the GMA code, instantiating it, and removing it when it is no longer needed. In addition, the GMcontainer keeps a persistent registry of the GMAs installed.

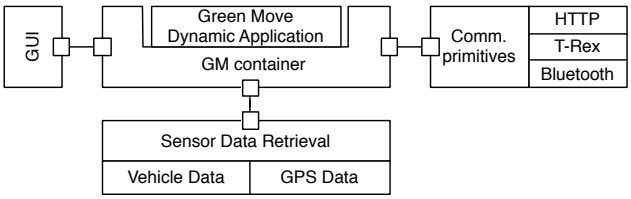


Fig. 4. Schematic view of the Green Move Dynamic App Container.

The interface implemented by a GMA allows it to retrieve from the GMcontainer references to other components of the Green Move environment running on the GEB. In particular, a GMA can use primitives that allow it to: (i) send messages through a variety of communications channels; (ii) acquire the vehicle data (speed, acceleration, state of charge, etc.) from the sensors; (iii) display information on the Graphical User Interface (GUI) of the GEB if one is available.

A GMA can communicate with remote components in several different ways. It can use an HTTP channel that is made available and mediated by the GEB. Such a channel can be used to interact with external services to send/retrieve information (e.g., weather, traffic, advertisements) through standard protocols. For security reasons, the HTTP connection is managed on the GEB by a software proxy which has the possibility to check the traffic and, for example, block access to blacklisted servers. A standard HTTP channel, however, is ill-suited for transmitting (or receiving) streams of data, such as telemetry or composite events. For this reason the GMcontainer offers GMAs access to a T-Rex channel, through which they can publish or subscribe to events. The T-Rex channel allows a GMA to distribute information to other vehicles, if the latter subscribe to the events published by the

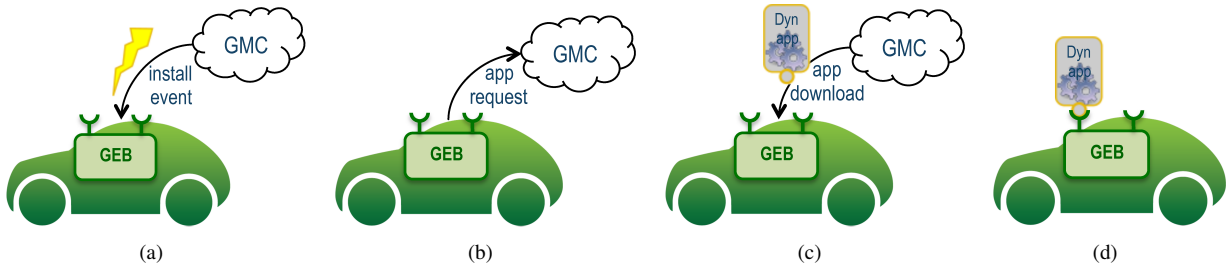


Fig. 3. Installation of a dynamic application.

former. A GMA can also use the Bluetooth channel mediated by the GMcontainer (see Fig. 4) to send messages to the smartphone of the user during the rental. This allows a GMA to communicate with the Green Move client application, or another companion application on the user's smartphone, as explained in Sect. VI.

The GMcontainer offers GMAs the possibility to interact with the vehicle through a high-level interface. The interface is very general, and it can be used to read the status of the vehicle, such as the state of charge of the battery or the speed of the vehicle. However, as mentioned above the Green Move platform targets fleets made of heterogeneous vehicles, so different kinds of vehicles might offer different kinds of data; for example, not all vehicles have doors and, for those that do not, a "door status" is not available. Through this high-level interface a GMA can also send commands to the vehicle. For example, "open/close door" commands can be issued if the GMA resides on the GEB of a vehicle that has doors. As before, the sending/receiving of commands is mediated by the GEB, which can monitor the interaction.

If the GEB has a screen (not all do, such as those of scooters) a GMA can use it to provide information to the user. For example, a GMA might display suitable diagrams to give the user feedback on the driving style. The GMcontainer offers GMAs two different possibilities to show information on the GEB GUI: display a short, purely textual message on the main view of the GEB app; or build a full-screen, graphical view that is under the control of the GMA. The availability of different primitives allows GMA programmers to select the one that better fits the needs of the application, for example from the point of view of the energy consumption.

The next sections present meaningful examples of prototype GMAs that illustrate the features available to dynamic applications, and what benefits can be gained from them.

V. The Driving Style Dynamic Application

The limited driving range of EVs represents a significant obstacle to their widespread diffusion; it is at the root of *range anxiety*, one of the main psychological barriers that hinder the acceptance of green mobility services [13]. Hence, energy-saving strategies which provide drivers with a consistent usable range and which enhance the maximum range are of primary importance for EVs. It is well known that the driver's behavior has a significant effect on energy consumption; then, important energy savings can be achieved through systems that help promote an economical style of driving [14], [15].

Within the Green Move project, a GMA called *Driving Style Dynamic Application* has been developed to help the user save battery. The application computes a quantitative estimation of the driving style of a Green Move user, communicates this information to the GMC, and displays it to the driver, thus inducing her to keep an energy-saving driving profile. As explained later on, the flexibility of the dynamic application framework allows the driving style GMA to be tailored to different user needs.

A. Overview of the application

Fig. 5 shows a general overview of the driving style GMA and its interactions with vehicle and user. The purpose of the application is to gather vehicle data from the GEB, such as current speed and acceleration, and use this information to compute suitable aggregate indicators. In particular, the application uses a custom algorithm to estimate and evaluate the driving style of the Green Move user by computing some real-time indexes. Feedback is provided to the user to nudge her to adopt a more economical driving style, whereas the GMC can use this information to make appropriate enhancements to its service or to provide additional features (adopt personalized fees, alert the user, improve the prediction of the vehicle's range, perform fleet optimization, etc.).

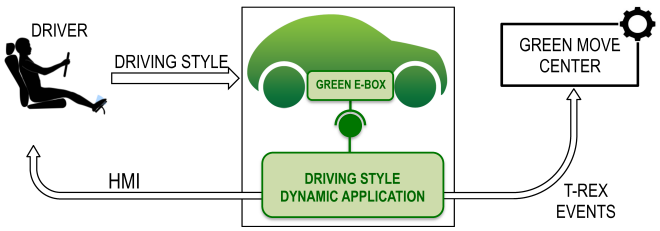


Fig. 5. Overview of the driving style GMA.

1) The algorithm: To give a synthetic evaluation of the user's driving style, some cost functions for the estimation of energy-inefficient behavior are defined. The first step is to obtain a signal that is as accurate as possible of the power consumed during the drive; this is achieved by computing the net power at the wheel defined as

$$P_{\text{wheel}}\left(v(t), \dot{v}(t)\right) = P_{\text{iner}}\left(v(t), \dot{v}(t)\right) + P_{\text{res}}\left(v(t)\right) \quad (1)$$

where P_{iner} is the inertial power and P_{res} represents the power dissipated by the non-conservative forces such as aerodynamic, slope and rolling friction forces acting on the vehicle. Thus,

P_{wheel} is a nonlinear function of the speed $v(t)$ and the acceleration $\dot{v}(t)$; these are the variables upon which one must act to reduce the energy consumption. Estimating accurately the acceleration and speed profiles of a Green Move driver is very simple, since this information is retrieved directly from the GEB through the GMcontainer primitives which give access to the vehicle data.

2) The indexes: After obtaining an accurate estimation of the vehicle energy consumption, the application computes the following two indicators that capture those aspects of power consumption directly influenced by the driving behavior.

Driving Aggressiveness Index γ_1: it stems from the consideration that the best behavior to reduce energy consumption consists in proceeding at a constant speed and, when braking and accelerating, doing so gradually [14]. Thus, we defined γ_1 as the high-pass filter of P_{wheel}, evaluating the smoothness of the power profile chosen by the driver. In fact, the power spent at the wheel (P_{wheel}) can be split into two parts: a low-frequency one, P_{LPF}, which is needed to move the vehicle, and a high-frequency one, P_{HPF} that represents unnecessary inertial power that is wasted in the process [16]. Fig. 6 shows the strict relation existing between driving style (aggressive or smooth) and spectral density of P_{wheel}.

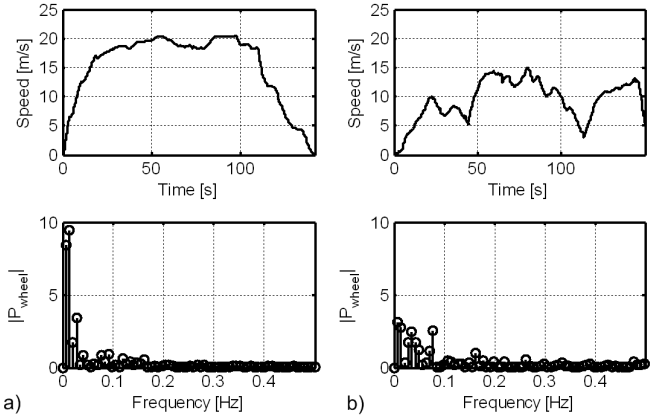


Fig. 6. Speed profile and spectral content of P_{wheel} for (a) a smooth driver and (b) an aggressive one.

Average Driving Style Index γ_2: whereas γ_1 gives an instantaneous information of the user's driving style, this cost function aims to keep track of the driver's behavior over a fixed time interval. The index is defined by equation (2) as the ratio between the power spent by resistance dissipative forces P_{res}, divided by the total power spent P_{wheel}, and multiplied by a forgetting factor parameter δ which attributes more importance to recent data over a time window of M seconds.

$$\gamma_2 = \sum_{t=i-M}^{i} \delta^{(M-t)} \frac{P_{res}(t)}{P_{wheel}(t)} \qquad (2)$$

The detailed formulation of the indicators and a discussion of the implementation of the algorithm can be found in [16].

3) The HMI: It is generally accepted that providing the driver with real-time feedback via dedicated visual representations is a good strategy for saving energy [15], [17]. Therefore, a graphical version of the driving style GMA has been realized thanks to the GMcontainer primitives that allow access to the GEB GUI. In this case, the goal is to provide the user with a real-time feedback of the indexes; this passive control energy-saving mechanism aims at indirectly influencing the driver's behavior by making her aware of the quality of her driving in terms of energy consumption. Fig. 7 shows the graphical user interface that provides a quick and simple representation of the driving style's indexes described above.

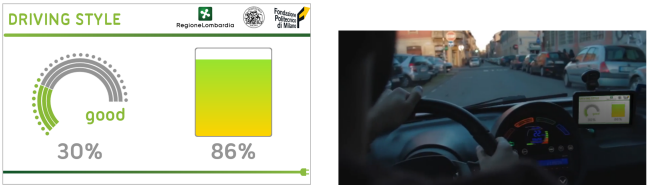


Fig. 7. Driving style GMA running on a Green Move vehicle.

B. Benefits of the dynamic applications framework

The driving style GMA highlights the benefits that can be gained by using the dynamic applications framework. Several reasons led us to developing the application as a GMA.

A heterogeneous fleet needs customized applications, designed specifically for different vehicle models. The access to the GUI of each vehicle changes from model to model (e.g. some vehicles have an embedded tablet, others give access to the proprietary dashboard) as well as the available data (speed, current, voltage, etc.).

The high configurability of the system can lead to custom modifications of the algorithm. Some parameters can be differentiated for each Green Move user in order to tailor the driving style assessment on the user's needs and preferences.

The events generated by the application may need to be dynamically modified depending on the specific needs of the GMC (e.g., change the sending rate, differentiate the number and the type of data to be received, etc.).

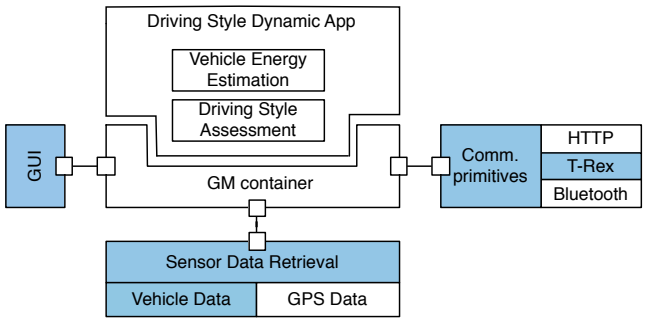


Fig. 8. Features offered by GMcontainer used by the driving style GMA.

As shown in Fig. 8, the application has been developed to exploit some of the features offered by the dynamic application framework, and in particular

- *the GUI's primitives*, to display the driving style indexes on the GEB;

- *the T-Rex event channel*, to send messages to the GMC;
- *the vehicle data retrieval interface*, to gather data about the vehicle (e.g. speed and acceleration).

Another benefit of the dynamic applications framework is that the GMC can monitor the vehicle status and force the GEB to install new applications to gather more data or give driving advice to the user where needed. Consider, for example, the scenario in which a user is driving a Green Move vehicle to the supermarket and then back home. The GMC knows the initial state of charge (*SoC*) of the battery, thanks to T-Rex event *SocWhenTaken* defined in Listing 1. More precisely, a *SocWhenTaken* event is generated when a rental starts (which corresponds to event *Taken*); it contains the last value of the SoC that is retrieved from the vehicle. At regular intervals while the vehicle is moving the GEB sends *VehicleStatus* events to the GMC; when the residual charge of the battery is less than a given threshold, a *SendApp* event, whose T-Rex definition is shown in Listing 1, is generated containing the information needed to load the driving style GMA. The installation is triggered when constraint (3) holds, that is, when the current SoC is less than the SoC consumed since the beginning of the rental times factor k.

$$SoC_{curr} \leq k\,(SoC_{init} - SoC_{curr}) \tag{3}$$

VI. The Custom advice dynamic application

A service that aims to be tailored to the needs of its users must be able to provide them with customized advice without violating their privacy; otherwise, users might be inclined to give inaccurate information about their preferences, which is of little use for personalization purposes. The GMadvisor GMA exploits the mechanisms available to dynamic applications to provide users with customized advice through context-aware procedures, without requiring that user preferences be shared with the rest of the Green Move system.

The GMadvisor GMA is based on the notion of *Context Dimension Tree* (CDT) for describing contexts [18], and on an adaptation of the *PervAds* framework [19], which supports privacy-enforced ads and the distribution of service-related information from sellers to customers. In the rest of this section we briefly describe the features of CDTs and of the PervAds framework, then present the GMadvisor GMA.

A. Representing contexts through Context Dimension Trees

A Context Dimension Tree captures the categories that are relevant to describe an entity, such as for example the user of a system, or a vehicle, or a service. In a CDT the children nodes of the root correspond to the main *dimensions* of interest of the entity being represented. For example, in a CDT that represents the profile of a user, the dimensions along which the user is categorized could be *age*, *gender*, *profession*, *family status*, *hobbies*, *mobility preferences*. Each dimension has children nodes, which represent the *values* that the dimension can take – e.g., the values for *hobbies* could be *theater*, *music*, and *sports*, whereas the values for *mobility preferences* could be *private vehicle* and *public transports*. The structure can be iterated an arbitrary number of times: each value node can in turn have children representing further dimensions in which that value can be decomposed, whose children are also value nodes, and so on. For example, the sub-dimensions of *public transports* could be *local*, *regional*, and *international*, with *taxi*, *bus*, *metro*, *shared car* the values for the *local* dimension.

Given a CDT, a *context* is the conjunction of different dimension-value pairs, any number of which can be selected provided that they obey the following rules:

- No dimension-value pair can be selected if another one deriving from the same dimension is present already (e.g., if the value *shared car* is selected for dimension *local* of *public transports*, this implies that no other value of the same subtree – e.g., *taxi* – can be chosen).
- For each selected dimension node, exactly one value from its children nodes is chosen (e.g., *music* and *shared car*).

In addition, if a value is decomposed in further dimensions, and none of them is selected, by the semantics of CDTs any of them is possible.

For example, the set of values $\{music, public\ transports\}$ could describe a context in which the user is pursuing her musical interests using public transportation (e.g., she is going to a concert using a car sharing service). Additional constraints in the definition of contexts may be applied by means of specific edges between nodes, as described in [20].

Through the context model presented above the GMadvisor GMA can capture both the current user context and the context in which a suggestion must be shown.

B. The PervAds framework

When dealing with the most sensitive user data, one of the main issues that arise is the possibility of improper – and unwanted – disclosure. A user may fear that making her preferences available to a system can lead to a loss of privacy, for example if the system owner or its affiliated partners access her data for marketing purposes. A user may also be wary of possible data theft if she fears the security settings of the central storage are too loose. The PervAds framework [19] mitigates these risks by avoiding that sensitive user data leave the user's smartphone.

The architecture of the PervAds system has been adapted for the GMadvisor GMA. It consists of a server that, through an internet connection, provides a set of ads to the user smartphone, on which a PervAds client runs. When the smartphone client app connects to a PervAds server, the latter sends the former all ads related to shops in its neighborhood; each ad includes a small caption, an optional picture and a set of metadata that describe the target categories of the ad. The PervAds smartphone app receives and filters the ads using the user preferences saved on the device; it assigns each ad a score and displays only those that have reached a certain threshold, or the top-k ones. No user data ever leaves the smartphone, which mitigates against any risk of losing control of the data.

The PervAds framework is based on the CDT context model of Sect. VI-A, which is used to capture the user preferences, the ads metadata describing the targets and the possible contexts of interest.

```
Define SocWhenTaken (String: greenBox_id, int:soc)
From   Taken(greenBox_id=$a) and last VehicleStatus(greenbox_id=$a) within 5 days from Taken
Where  SocWhenTaken.greenBox_id=VehicleStatus.greenBox_id and SocWhenTaken.soc=VehicleStatus.soc

Define SendApp (String: greenBox_id, String: appUrl, String:class)
From   VehicleStatus(greenbox_id=$a) and last SocWhenTaken(greenBox_id=$a) within 1 day from VehicleStatus and
       not Released(greenBoxId=$a) between VehicleStatus and SocWhenTaken and VehicleStatus.soc < SocWhenTaken.soc*k/(1+k)
Where  SendApp.greenBox_id=VehicleStatus.greenBox_id and SendApp.appUrl = "<GMAurl>" and SendApp.class="<GMAclassName>"
```

Listing 1. T-Rex rule forcing the installation of the driving style GMA.

C. GMadvisor

The mechanisms described in Sect. VI-A and VI-B have been adapted to fit the dynamic applications paradigm of Sect. IV and to build a GMA, called GMadvisor, for providing Green Move users with customized advice.

The GMadvisor GMA relies on the following features of the dynamic applications framework, as shown in Fig. 9:

- access to the GEB GUI, to show suggestions on the GEB display (if this is available);
- communication with the user smartphone through the Bluetooth channel, to exchange messages between the user device (which selects the advice to show, according to the framework of Sect. VI-B) and the GEB;
- communication with external servers through the HTTP channel, to interact with the central server containing the advice to be distributed;
- retrieval of vehicle data, to determine, for example, the current position of the user.

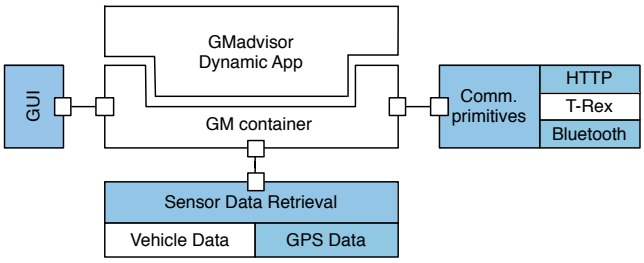


Fig. 9. Features offered by the GMcontainer used by the GMadvisor GMA.

The choice to adapt the PervAds framework into a GMA is driven by two factors. First, not all users may be willing to receive additional information during a trip; this might also depend on the purpose of the trip, as a user might be willing to receive advice – e.g., of commercial nature – during a leisure trip, but not during a business one. Second, by exploiting the HTTP channel provided by the GEB the user avoids consuming her own internet traffic, with its related costs.

The framework providing customized advice to users is based on three components (as depicted in Fig. 10):

- A server storing the set of advice that can be sent to users (e.g., position of charging stations, points of interest).
- The *GMadvisor GMA* installed on the GEB; the application sends to the advice server the user id and position, receives the advices and displays those that match the user preferences as determined by the client app.
- The *GMadvisor client app* installed on the user device; the app is responsible for storing the private user preferences that should not reside on a remote server; depending on them, it filters the advices received by the GMA; it can also display advices to the user if the GEB does not have a GUI.

The set of possible advice stored on the server can be large. Hence, the system should balance the computational load between the server and the instances of the GMadvisor GMA installed on GEBs, and also avoid sending too many suggestions to GEBs, thus exhausting the available bandwidth. To achieve this, the information to be sent to GEBs is pre-filtered at the server level based on the user position, the time and the user's data that is associated with her registration with the Green Move system (non-sensitive information such as age, gender, etc.). The pre-filtered advice received from the server is further filtered on the client app installed on the user's smartphone using the sensitive profile information stored there.

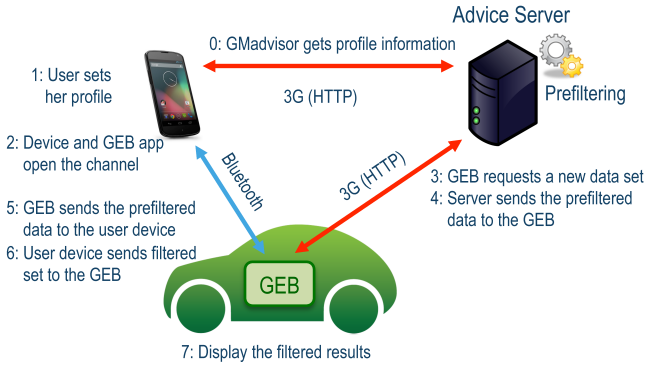


Fig. 10. Schematic description of the GMadvisor GMA and service.

Fig. 10 shows the steps performed and the interactions among the different components in a usage scenario. First, the user installs the client app on her device and selects her preferences according to a CDT retrieved from the central server. During the trip, the client app exploits the Bluetooth channel to interact with the GMadvisor GMA installed on the GEB. In response to the GMadvisor GMA sending the id and position of the user, the advice server sends back a set of pre-filtered advice, whose metadata is relayed by the GMA to the client app for further filtering. Finally, the selected advice is displayed on the GEB if a GUI is available there (otherwise the advice is shown on the user's smartphone).

VII. Dynamic Applications in Development

The applications described in the previous sections have been fully integrated in the Green Move system as GMAs. The framework described in Sect. IV, however, is general,

flexible, and open, and it can be exploited to enrich the range of services offered during trips through applications possibly developed by third parties. To illustrate the extensibility of the services offered, in this section we briefly present two other GMAs that are currently being developed: Feedback Dynamic Pricing and Active State-of-Charge Control.

1) Feedback Dynamic Pricing: In vehicle-sharing systems that allow one-way trips (i.e., where the destination differs from the origin) the distribution of vehicles across stations tends to be unbalanced: some stations end up with too many vehicles (i.e., they are attraction points) whereas others have too few (possibly zero). This poses a number of problems regarding fleet management so, even though one-way trips are very attractive for users, they are sometimes forbidden in vehicle-sharing systems. In the Green Move context this problem has been addressed from a novel perspective, by reformulating it as a control problem according to the *Feedback Dynamic Pricing* rationale (see [21] for details). The key consideration is that, assuming that people are sensitive to changes in the price of the service, a vehicle-sharing system can be accurately modeled by a dynamical system, and its balance can be actively and real-time controlled by acting on the trip fee. With reference to Fig. 4, the primitives offered by GMcontainer that are necessary to realize the Feedback Dynamic Pricing GMA are the GPS localization provided by the vehicle data retrieval interface, access to the GEB GUI and to the HTTP channel. The mechanism of Feedback Dynamic Pricing could also be used to address the issues of coordinated access to charging stations mentioned in Sect. I.

2) Active State-of-Charge Control: One way to overcome the driving range limitations of EVs is to adopt strategies based on the active control of the vehicle dynamics. More precisely, a pro-active Energy Management System can control the state of charge of EVs by enforcing – rather than simply suggesting – limitations on the driving style [5]. In fact, on some Green Move vehicles the GEB can set bounds on the values of certain parameters, such as speed and acceleration. This opens the possibility of developing a GMA that actively controls the state of charge of the vehicle battery. The discharge rate is tracked by means of speed and acceleration controllers, which intervene to constrain the vehicle dynamics according to high-level energy management policies [22]. Through the GEB GUI made available by GMcontainer (see Fig. 4), the Active State-of-Charge Control GMA allows the user to set the information about the planned route and other preferences. It also uses the vehicle data retrieval facilities offered by GMcontainer to obtain the information on the current state of charge of the vehicle battery. Finally, the application runs a control algorithm which actively modifies the dynamic behavior of the vehicle by ensuring that speed and acceleration are kept within prescribed bounds.

VIII. Conclusions

The Green Move platform for electric vehicle-sharing systems is highly configurable, thanks to the mechanisms that allow applications to be dynamically loaded onto the vehicles' onboard computers before and during trips, and then be removed when they are no longer needed. This flexibility allows us to tailor the services offered during the vehicle rental to the user's needs and preferences, as witnessed by the prototype applications described in this article. We are currently working on improving the framework of the dynamic applications, in particular in relation to its security aspects. We are also building mechanisms that exploit dynamic applications to automatically re-configure the services offered by vehicles, depending on the state of charge of the battery, to manage the energy consumed and the level of service offered in a joint optimal manner.

References

[1] G. Alli, L. Baresi, A. Bianchessi, G. Cugola, A. Margara, A. Morzenti, C. Ongini, E. Panigati, M. Rossi, S. Rotondi *et al.*, "Green move: towards next generation sustainable smartphone-based vehicle sharing," in *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*. IEEE, 2012, pp. 1–5.

[2] A. Bianchessi, C. Ongini, S. Rotondi, M. Tanelli, M. Rossi, G. Cugola, and S. Savaresi, "A flexible architecture for managing vehicle sharing systems," *IEEE Embedded Systems Letters*, vol. 5, no. 3, pp. 30–33, 2013.

[3] A. Luè, A. Colorni, R. Nocerino, and V. Paruscio, "Green move: An innovative electric vehicle-sharing system," *Procedia-Social and Behavioral Sciences*, vol. 48, pp. 2978–2987, 2012.

[4] S. Di Martino, C. Giorio, and R. Galiero, "A rich cloud application to improve sustainable mobility," *Web and Wireless Geographical Information Systems*, vol. 6574/2011, pp. 109–123, 2011.

[5] A. Dardanelli, M. Tanelli, B. Picasso, S. Savaresi, O. di Tanna, and M. Santucci, "A smartphone-in-the-loop active state-of-charge manager for electric vehicles," *IEEE ASME Transactions on Mechatronics*, vol. 17, no. 3, pp. 454–463, 2012.

[6] M. Conti, D. Fedeli, and M. Virgulti, "Bluetooth for electric vehicle to smart grid connection," *IEEE Intelligent Solutions in Embedded Systems*, vol. 1, pp. 13–18, 2011.

[7] A. Bianchessi, C. Ongini, G. Alli, E. Panigati, and S. Savaresi, "Vehicle-sharing: Technological infrastructure, vehicles, and user-side devices - technological review," in *International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2013, pp. 1599–1604.

[8] G. Cugola and A. Margara, "Complex event processing with T-REX," *Journal of Systems and Software*, vol. 85, no. 8, pp. 1709 – 1728, 2012.

[9] ——, "Low latency complex event processing on parallel hardware," *Journal of Parallel and Distributed Computing*, vol. 72, no. 2, pp. 205 – 218, 2012.

[10] ——, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys*, vol. 44, no. 3, pp. 15:1–15:62, 2012.

[11] F. A. 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*, vol. 38, no. 2, pp. 478–496, 2012.

[12] F. Schreiber, L. Tanca, R. Camplani, and D. Viganò, "Pushing context-awareness down to the core: more flexibility for the perla language," in *International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases*, 2012, pp. 1–6.

[13] N. S. Pearre, W. Kempton, R. L. Guensler, and V. V. Elango, "Electric vehicles: How much range is required for a day's driving?" *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1171 – 1184, 2011.

[14] V. Manzoni, A. Corti, P. De Luca, and S. Savaresi, "Driving style estimation via inertial measurements," in *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2010, pp. 777–782.

[15] C. Vagg, C. Brace, D. Hari, S. Akehurst, J. Poxon, and L. Ash, "Development and field trial of a driver assistance system to encourage eco-driving in light commercial vehicle fleets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 796–805, 2013.

[16] A. Corti, C. Ongini, M. Tanelli, and S. Savaresi, "Quantitative driving style estimation for energy-oriented applications in road vehicles," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2013, pp. 3710–3715.

[17] M. Van der Voort, "Fest. a new driver support tool that reduces fuel consumption and emissions," in *International Conference on Advanced Driver Assistance Systems (ADAS)*, 2001, pp. 90–93.

[18] C. Bolchini, C. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca, "And what can context do for data?" *Communications of the ACM*, vol. 52, no. 11, pp. 136–140, 2009.

[19] L. Carrara, G. Orsi, and L. Tanca, "Semantic pervasive advertising," in *Web Reasoning and Rule Systems*, ser. Lecture Notes in Computer Science, vol. 7994, 2013, pp. 216–222.

[20] C. Bolchini, E. Quintarelli, and L. Tanca, "Carve: Context-aware automatic view definition over relational databases," *Information Systems*, vol. 38, no. 1, pp. 45–67, 2013.

[21] A. Bianchessi, S. Formentin, and S. Savaresi, "Active fleet balancing in vehicle sharing systems via feedback dynamic pricing," in *16th Conference on Intelligent Transport Systems (ITSC)*. IEEE, 2013.

[22] A. Dardanelli, M. Tanelli, B. Picasso, S. Savaresi, O. di Tanna, and M. Santucci, "Speed and acceleration controllers for a light electric two-wheeled vehicle," in *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011, pp. 2523–2528.

Andrea G. Bianchessi received the M.Sc. in Computer Engineering from the Politecnico di Milano in 2011. He is currently a Ph.D. Student in the Automation Engineering section at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano in the MOtor VEhicle control team (MOVE). His main research interests are about the design and control of advanced vehicle sharing systems.

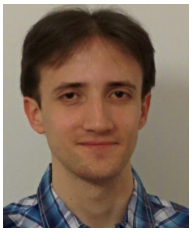
Simone Formentin received his Ph.D. degree in Information Technology within a joint program between Politecnico di Milano and Johannes Kepler University Linz, Austria. He currently is an assistant professor at Politecnico di Milano. His research interests include data-driven controller tuning methods and automotive applications.

Gianpaolo Cugola is Associate Professor at Politecnico di Milano. His research interests are in the area of Software Engineering, Distributed Systems, and middleware technology for highly reconfigurable applications, with a special attention to the issue of Complex Event Processing. On these and other topics he authored a hundred papers published in international journals and conferences.

Angelo C. Morzenti is Full Professor at Politecnico di Milano, where he chairs the Computer Science and Engineering Study Programme. His research interests are centered on languages, methods, and tools for the specification, the analysis and the design of computer-based systems with a particular emphasis on real-time, embedded, and distributed systems.

Carlo Ongini received the M.Sc. in Computer Engineering from the Politecnico di Milano in 2011. He currently is a Ph.D. candidate in Information Technology at the Department of Electronic, Information and Bioengineering of the Politecnico di Milano. His main research interests are about automotive applications, with focus on the design of energy-oriented driver assistance systems.

Matteo Rossi is an assistant professor at Politecnico di Milano. His research interests are in formal methods for safety-critical and real-time systems, architectures for real-time distributed systems, and transportation systems both from the point of view of their design, and of their application in urban mobility scenarios. He has previously participated in several national and international projects, including MADES and Green Move.



Emanuele Panigati received the MSc degree in Computer engineering in 2010 and is currently a Ph.D. candidate in Informatics engineering at the Department of Electronic, Information and Bioengineering of the Politecnico di Milano in the PErvasive Data GRoup of EngineErs. His research interests include context-aware systems, complex event processing systems, data stream management systems and RDF/OWL reasoning systems.

Sergio M. Savaresi is Full Professor in Automatic Control at Politecnico di Milano, where he chairs the Systems&Control area of Dipartimento di Elettronica, Informazione e Bioingegneria. His main interests are in the areas of vehicles control, automotive systems, data analysis and system identification, non-linear control theory, and control applications. On these topics he has published more than 300 scientific publications.

Fabio A. Schreiber is Full Professor of Databases and of Pervasive Data Management at the Department of Electronic, Information, and Bioengineering of the Politecnico di Milano. His current research interests include: data management in pervasive systems, Database Systems for context-aware applications, Very Small and Mobile Database design methodologies and applications. He authored a hundred papers published in international journals and conferences.

Letizia Tanca is Full Professor at Politecnico di Milano, where she chairs the Computer Science Dept. Area. She teaches courses on Databases and Information System Technologies. She is the author of more than 120 papers on databases and database theory, deductive and active databases, graph-based languages, semantic-web information management, and recently on context-aware knowledge management and Big Data analytics.

Edoardo G. Vannutelli Depoli received the M.Sc. in Computer Engineering from the Politecnico di Milano in 2007. He is currently research assistant at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano. His main research interests are about the design and development of mobile applications, with particular focus on transportation systems and on the support of people with disabilities.