A Flexible Architecture for Managing Vehicle Sharing Systems

A. G. Bianchessi, C. Ongini, S. Rotondi, M. Tanelli, M. Rossi, G. Cugola, S. M. Savaresi

***Abstract*—Vehicle sharing systems are the key to sustainable mobility. They need to possess adaptation features to answer the different user needs, and must be automated to avoid intermediaries between users and system. Finally, they must be based on a wide variety of vehicles and on an open ownership model to become a viable alternative to private vehicles. This letter presents the solution devised in the Green Move project to tackle this challenge. In particular, it focuses on the Green Move vehicle on-board units, for which a dedicated middleware is being developed. The middleware allows developers to create applications that can be dynamically loaded/unloaded while preserving the needed safety levels of the vehicle motion functions.**

***Index Terms*—Transportation systems, electric vehicles, embedded architecture, middleware.**

I. Introduction

New models of sustainable mobility call for the integration of vehicles that differ in type (cars, scooters, bicycles), technology (electric, hybrid or with classical combustion engines) and ownership (they can be publicly or privately owned, fully or partially shared) within the same system, and they must offer end-users common functions, services and interfaces [1].

To realize such systems, suitable, innovative solutions must be found to design a hardware/software architecture that manages the on-board vehicle embedded control units and connects them to a back-end system that provides the mobility services to the final users, preferably through personal mobile devices, such as smartphones, tablets, etc. This calls for solutions that meet the transparency and availability requirements within a distributed environment (see, e.g., [2], [3]). This letter presents the technological solutions adopted in the Green Move (GM) project to address the above issues. Green Move aims at realizing an innovative vehicle sharing system based on zero-emissions electric vehicles. The GM system is innovative for many aspects: the heterogeneity of the vehicles involved, the openness of the ownership model and the interaction protocols between the users and the system, which eliminate all intermediaries. In [4], readers can find an overview of the project and its relation with existing vehicle sharing systems.

The project focuses on electric vehicles (EVs) as a means to achieve a sustainable model of mobility, in particular in the urban context. At the same time, vehicle sharing, possibly including different kinds of vehicles owned by different entities, is crucial to mitigate the obstacles to the distribution of EVs; e.g., their high price compared to that of similar combustion engine vehicles entails that EVs become economically viable only if they cover long distances during their lifetime, a target that is difficult to achieve outside of vehicle sharing solutions. The technology developed within the GM project is not specific for EVs, aside from some low-level details concerning the monitored information (e.g., the battery state of charge), and can be adapted to traditional vehicles, too. Nevertheless, it answers specific needs that arise in vehicle sharing systems with heterogeneous ownership models such as those that can maximize the utility of EVs.

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

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

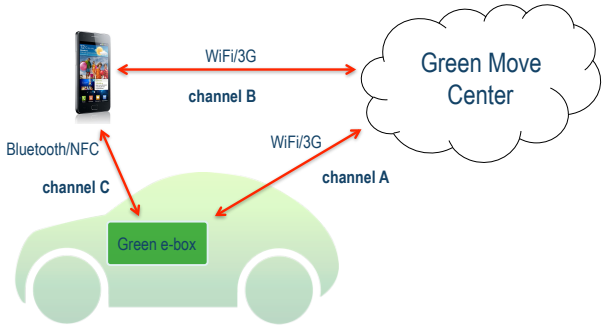

Fig. 1. Schematic view of the overall architecture of the GM system.

A key aspect of the GM system is that it does not involve intermediaries between users and vehicles: reserving, acquiring and releasing a vehicle are all done automatically through software running on the GM users' smartphones. Even keys for accessing vehicles are not required as they are substituted by software keys. This is reflected in Fig. 1, which shows the core of the GM architecture. It includes the *Green Move Center (GMC)*, the *Green e-Boxes (GEBs)*, and the users' smartphones on which the *GM client app* is installed. Communication and coordination among these components are based on a novel middleware, which also allows applications to be dynamically loaded/unloaded on GEBs (more on this later).

The GMC acts both as a web front-end for GM users and as an API back-end with which GEBs interact. The *GM middleware* uses 3G channels (channel A in Fig. 1) to let the GMC communicate with GEBs to manage the fleet. The same channel is used by GEBs to send, at regular intervals, vehicle data such as diagnostic information, usage statistics, and trip data (e.g., current GPS position, speed, state of charge) to the GMC. Finally, the GMC uses this 3G channel to dynamically add services to each GEB, by uploading new software modules that provide additional functions. This is a crucial aspect for a vehicle sharing system with different forms of ownership: as the fleet itself is reconfigurable, so must be the vehicle sharing

functions available on vehicles. For example, a vehicle can be made available for rental through the GM system by its owner when she does not need it, but it reverts to being a private car when used by its owner in pre-defined time intervals.

A GM user interacts with the system through the GM client app installed on her smartphone. It communicates with the GMC through a WiFi or 3G channel (channel B in Fig. 1) to reserve vehicles and to retrieve the electronic key required to unlock a reserved vehicle. The electronic key is exchanged between the GM client app and the GEB through a Bluetooth or NFC channel (channel C in Fig. 1) and it is used to open/close the vehicle doors (if present), and to enable the vehicle drive. Notice that 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/Wi-Fi connection available between the GEB and the GMC (e.g., in an underground parking lot). The same Bluetooth connection can be used to access, from the GM client app, information that is available on the GEB, such as trip statistics.

To manage this complex system and meet the desired requirements of transparency and availability, abstraction mechanisms have been implemented, which allow the seamless use of technologically different vehicles, characterized by different available signals, different on-board networks, a different split between digital and analog signals, and so on. Furthermore, as we mentioned above, the GM middleware allows administrators to manage (add/remove) the features and services of GEBs while preserving the integrity of the low-level safety-related routines of the GM vehicles. The rest of the paper discusses these aspects in detail.

Before presenting the solutions adopted in the GM project, we point out that systems that offer functions similar to those of GEBs are available on the market, such as OnStar [5], MyChevrolet [6] and Viper [7]. Among these, the Viper system is designed to be integrated into any type of vehicle, hence it is the most similar to ours. Nevertheless, the goal of these systems is to allow a vehicle owner to remotely control her vehicle by means of a smartphone, while the GEB has been developed to allow any vehicle to be inserted into a smartphone-based vehicle sharing system.

II. Green Move Vehicles and Embedded Architecture

Being electric is the only requirement that a vehicle has to satisfy to be added to the GM Vehicle Sharing System (GMVSS). For demonstration purposes, a heterogeneous set of 2- and 4-wheeled vehicles with different sizes, ranges and performances are currently integrated into the GMVSS, while others (such as 3-wheeled ones), will be added to the fleet in the next months. In particular, at the time of writing, the GMVSS includes (see Fig. 2) the following vehicles.

Tazzari Zero Evo is a two-seat electric car with a driving range of 140 km, and a maximum speed of 100 km/h. The lithium-ion battery pack requires about 9 hours for a full charge (0-100%), see [8]. This vehicle is suitable for urban mobility and/or short-range interurban trips.

Estrima Birò is a two-seat electric vehicle with a maximum speed of 45 km/h and a range of about 50 km. The Pb-Gel battery pack takes about 9 hours to be fully charged, see [9]. Its extremely compact size makes it suitable for urban (or low-speed suburban) streets.

Fig. 2. Integrated vehicles, from left to right: Tazzari Zero Evo, Estrima Birò, Piaggio Liberty e-Mail.

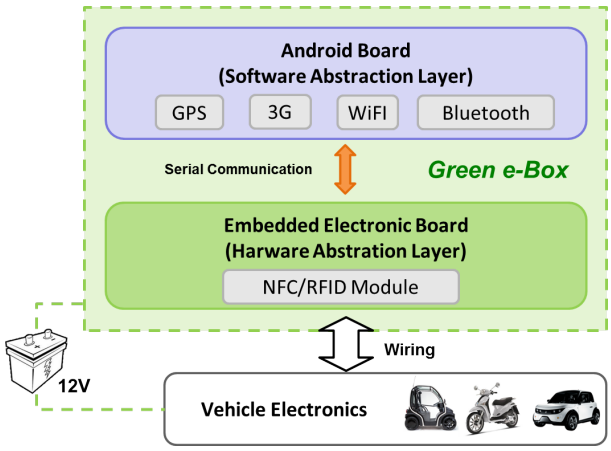

Fig. 3. Green e-Box architecture.

Piaggio Liberty e-Mail is an electric scooter with a top speed of 45 km/h and a range of 70 km. The lithium battery pack requires about 4 hours for a full charge, see [10]. The scooter is suitable for city driving, but, unlike the other two vehicles, it is not for all weather conditions.

To be integrated into the GMVSS, each vehicle has been equipped with a GEB, which allows a vehicle-independent communication among GMVSS elements. It is composed of a low-level embedded board and a high-level Android board (see Fig. 3). In order to have a constant monitoring of each vehicle, even when turned off and not in use, the GEB is directly connected to the permanent 12V line of the vehicle. The GEB is also wired to the vehicle electronics, it communicates with the GMC *via* a 3G channel, and with the users' smartphone *via* Bluetooth or NFC links. In particular, the use of smartphones for dematerializing the interaction with users and for dynamically adding new services as *mobile apps*, is one of the most promising options currently being explored, especially in the field of electric vehicles [11], [12], [13].

Fig. 4 details the GEB architecture, which was designed by following the principles of modularity (separate components are responsible for different functions, e.g. retrieving data from the vehicle) and extensibility (e.g., new functions can be easily added that can exploit the data collected from the vehicle).

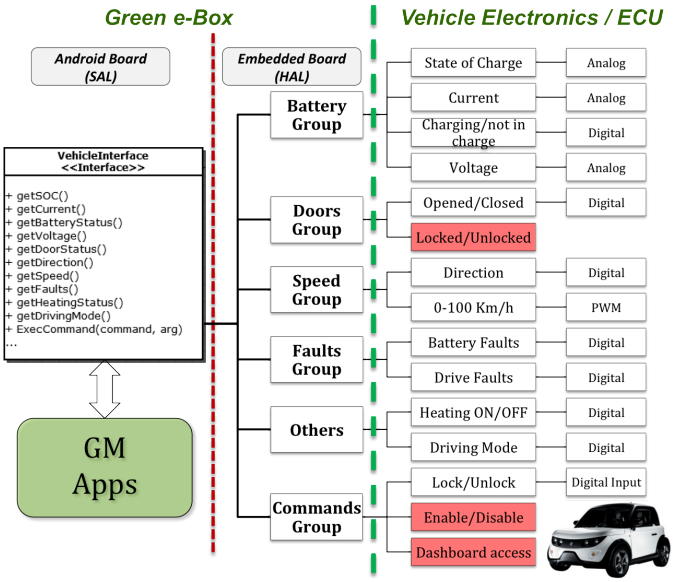

Fig. 4. Green e-Box abstraction layers and measured signals for the Tazzari Zero Evo. The highlighted signals and commands are those not available for the considered vehicle.

The embedded board acts as a Hardware Abstraction Layer (HAL) and it is designed to abstract the vehicle-specific details; thus, it creates a virtual layer between software applications and the actual hardware, providing a general communication protocol to the high-level layers built on top of it. To achieve this, the embedded board has a CAN-bus to retrieve data directly from the vehicle ECU and several analog and digital input/output channels so that the GEB can be installed on a large variety of heterogeneous vehicles, even those without an ECU (*e.g.*, Tazzari and Birò). A microcontroller handles each signal, acquiring the vehicle data at a constant rate and, since the set of available signals is strongly vehicle-dependent, it collects them into well-defined packets so that they can be easily transmitted to the high-level Android board. The vehicle signals are clustered into 6 categories: *battery*, *doors*, *speed*, *faults*, *commands* and *others*. Each signal available on the vehicle must belong to one of the previous categories. For instance, the provided current, the state of charge of the battery, and the battery's state (charging or not) belong to the *battery group*. Fig. 4 shows an example of the signals available on the Tazzari vehicle and how they are grouped by the embedded board. The signals of Birò and Liberty are not reported here for the sake of conciseness.

The Android board provides the Software Abstraction Layer (SAL) which receives (in a vehicle-independent way) the data from the low-level board; the SAL uses a singleton object to store, for each monitored quantity, the last value received; this object also offers, for each quantity, a getter method that allows other GM applications residing on the GEB (see Section III), to easily access the vehicle information. The GEB decouples the high-level fleet management functions from those, implemented in the vehicle ECU, related to the control of the vehicle motion, thus isolating the latter from the former, which guarantees the necessary safety requirements.

III. Embedded Green E-Box middleware

The functions realized by the GEB can be divided in two categories: core operations of the GMVSS (*e.g.*, user authentication, vehicle monitoring, etc.), and optional functions that, though not essential, provide added value to GM users (*e.g.*, commercial and traffic information). Whereas core operations are known from the design phase of the GMVSS and change infrequently, optional functions could be added or removed after the system deployment (e.g. new commercial agreements between GM and its partners). Consider a scenario such that, after the GMVSS has been deployed, a new commercial deal is struck with an ad provider who developed an application to notify users in real time of discounts available in shops close to their current location. In this case, the application should be installed on all GEBs from the beginning of the commercial agreement, until its end. In addition, the application should be able to read the current location of the vehicle from the GEB, to provide users with accurate information. This suggests that the GEB should have the possibility of dynamically loading and unloading applications which have suitably controlled access to vehicle information retrieved by the GEB itself. Thus, the GEB software should be designed to enable the kind of scenarios outlined above. In particular, it should have a modular structure, where each module oversees a cohesive set of GEB functions (*e.g.*, vehicle data retrieval, communication with the GMC); it should also provide a set of primitives and mechanisms that make it possible to build and manage applications in a dynamic way. In the rest of this section we discuss the choices made and the mechanisms realized to achieve these goals.

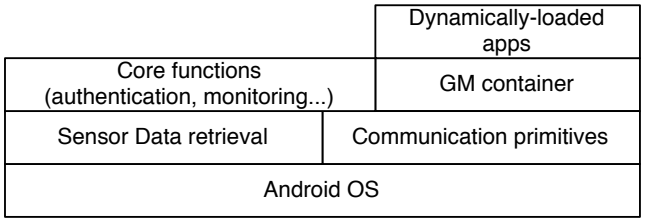

Fig. 5. Layered structure of the functions of the GEB.

The GEB software is built on the Android OS according to the layered structure shown in Fig. 5. At its core are the modules providing the basic mechanisms for the retrieval of data from the vehicle sensors and for the communication with the GMC. They are used to realize the core functions of the GEB, such as, for example, vehicle monitoring, and they are made available to third-party applications through a component called GMContainer.

The GMContainer is part of the GM Middleware (GMM) infrastructure that allows system administrators to load and remove applications from GEBs dynamically. The main components of the GMM are the Code Server, which resides on the GMC, and the GMContainer part of the GEBs.

1) Code Server: The Code Server (CS) is a module of the GMC that allows trusted parties to upload their applications, to verify them and to distribute them to GEBs through the GMM. It also allows administrators to get the current list of devices

running a certain application, to stop any running instance and uninstall it, or to deploy it a second time. Applications are uploaded as signed JAR files, which are verified for authenticity before being made distributable.

2) GMContainer: The main responsibility of the GMContainer is to allow Green Move Applications (GMAs) to be added and removed at run time. GMAs are downloaded when requested and their code is linked to the GEB code at run-time. The application entry point is then executed and a reference to the latter is kept by the GEB for future unloading.

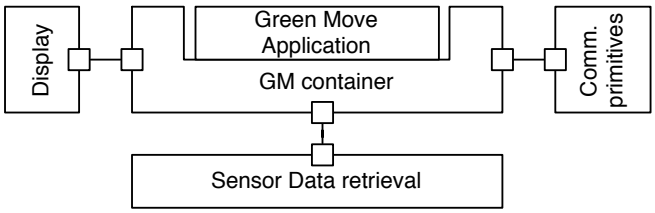

Fig. 6. Interfaces of the GMContainer.

As depicted in Fig. 6, GMAs are provided appropriate APIs for interacting with the GM ecosystem. They allow applications to:

- display text or images to the end user (ads, messages);
- communicate with other GM components (including the GMC) by sending and receiving events according to a publish-subscribe paradigm;
- read sensors data from the GEB embedded board, e.g., the GPS position and the vehicle speed.

GMAs are independent of one another, although no restriction is enforced on their interaction. They are provided to GEBs via the GMC. Their code is downloaded and executed, and then disposed of when the application no longer needs to be active, or system resources need to be freed.

3) Implementation: GMAs are Android components which adhere to the single entry point convention enforced by the GMContainer. This means that applications could be coded in any language which can be run on top of the Android Java Virtual Machine (Dalvik), such as Ruby or Python. The current implementation assumes that the application code is sealed in a standard JAR file.

CS allows a publisher to upload JAR files, to verify their integrity according to the publisher's certificate and to select the specific GEBs that have to receive the application, e.g., all of them or only those installed on a specific vehicle type, etc. The CS leverages the T-Rex publish-subscribe middleware, see [14], to communicate with GEBs and send them the "load code" message, which contains the references to the file to load and the entry point of the application.

The GMContainer is implemented as an Android Service. It provides a primitive that allows GMAs to access the API through which sensor data can be retrieved. Similar primitives allow GMAs to access the communication infrastructure used by the GMContainer to communicate with the GMC, and the abstraction layer to access the display to show text and images. Every GMA implements an interface offering a primitive to stop the application, which is responsible for the disposal of its own resources (sockets, running threads and so on).

IV. Concluding remarks and future work

The architecture of the GM system was designed to leverage the classic notions of *component*, *connector*, and *interface*, both at the hardware and at the software level, to provide the flexibility that is necessary to manage heterogeneous fleets of vehicles whose services evolve over time. As such, it is a blueprint for a wide class of systems that include, but are not limited to, next generation vehicle sharing systems. This letter focused in particular on the structure of the on-board unit, the GEB, which acts as the interface between the system and the vehicles, and on the GMM, the middleware infrastructure that has been designed to allow the functions of the GEBs to evolve after the system deployment. The GMM offers developers of GMAs mechanisms to send information not only to the GMC, but also to other vehicles through the event-distribution component of the GMC. As such, it provides capabilities that could be used to develop applications that are based on interactions among vehicles.

Future work on the GEB and the GMM will follow several directions: new modules of the GEB will be deployed to perform various analyses of vehicle sensor data. These will provide the user, dynamically-loaded GMAs and the GMC with rich information such as, quantitative measures of the "driving style", which can be used to optimize performance, sustainability, and/or driving range. The GMM will also be enriched to include context-aware mechanisms through which system administrators can allow/disallow certain operations depending on the current situation, so as to be able to actively enforce safety objectives.

References

[1] A. Kalogeras, P. Foundas, M. Georgoudakis, K. Charatsis, and P. Konstantinopoulos, "Integrated system for smart transport services," in *IEEE Conf. on Emerging Technologies Factory Automation*, 2009, pp. 1–6.
[2] R. Pon, M. Batalin, M. Rahimi, Y. Yu, D. Estrin, G. Pottie, M. Srivastava, G. Sukhatme, and W. Kaiser, "Self-aware distributed embedded systems," in *IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2004, pp. 102–107.
[3] A. Osório, H. Afsarmanesh, and L. Camarinha-Matos, "Towards a reference architecture for a collaborative intelligent transport system infrastructure," *Collaborative Networks for a Sustainable World*, vol. 336/2010, pp. 469–477, 2010.
[4] G. Alli et al., "Green move: towards next generation sustainable smartphone-based vehicle sharing," in *Proceedings of SustainIT*, 2012.
[5] OnStar, LLC, 2013. [Online]. Available: http://www.onstar.com
[6] Chevrolet, 2013. [Online]. Available: http://www.chevrolet.com
[7] Directed Electronics, 2013. [Online]. Available: http://www.viper.com/
[8] Tazzari GL s.p.a., 2012. [Online]. Available: http://www.tazzari-zero.com
[9] ESTRIMA SRL, 2012. [Online]. Available: http://www.estrima.com
[10] Piaggio & c s.p.a., 2012. [Online]. Available: http://www.piaggio.com
[11] 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.
[12] M. Conti, D. Fedeli, and M. Virgulti, "Bluetooth for electric vehicle to smart grid connection," *IEEE Intelligent Solutions in Embedded Systems*, vol. 1, p. 1318, 2011.
[13] 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.
[14] G. Cugola and A. Margara, "Complex event processing with T-REX," *JSS*, vol. 85, no. 8, pp. 1709–1728, 2012.