ECO-TrackDrive: a tracking system for energy-efficient driving*

Sameh Ben-Aoun[1,*,†], Meriem Belguidoum[2,†] and Ahmed Hadj-Kacem[1]

[1]University of Sfax, ReDCAD Laboratory, BP 1088, 3018 Sfax, Tunisia
[2]University of Constantine 2-Abdelhamid Mehri, LIRE Laboratory, BP : 67A, Constantine, Algeria

Abstract

Driving habits have a crucial role in road safety. They contribute significantly to the emissions of greenhouse gases and other air pollutants. Several studies have been conducted to provide possible solutions, including systems and mobile apps. Researchers have developed several systems for eco-driving and proposed new strategies to improve driving behaviour. However, there is still a lack of such systems to be scrutinized and exploited on the ground, particularly in the Maghreb, to improve real-time eco-driving. This paper presents ECO-TrackDrive; a driver-oriented mobile application. The Proposed approach is based on an IoT design methodology extended with the microservice-based architecture. the requirement specification is based on the modelling language SysML which is very suitable for IoT applications. The paper aims at improving the driver's behaviour, reduce CO2 emissions and save energy. This study intends to achieve the goal without distracting drivers from safe driving.

Keywords

IoT, Microservices, SysML, EcoDrive, Mobile application, Energy efficient driving

1. Introduction

Road transportation is a significant pollutant since it consumes much fuel, and as result, it contributes massively to global CO2 and other pollutant emissions. This issue has started world-wide concern about global warming and the depletion of fossil fuels. For instance, according to the Statista Research Department, transport produced roughly 7.3 billion metric tons of CO2 emissions, with passenger automobiles being the main contributor, accounting for 41% of all transportation-related emissions [1]. That percentage urged many countries to take action on energy preservation and initiatives to reduce carbon dioxide emissions [2], such as adopting an environmentally responsible driving style, as well as managing the rise in fuel consumption and carbon dioxide emissions.

The methods used to reduce fuel consumption and CO2 emissions vary, with promising results using data collected via smartphone or car sensors. The authors of [3] have developed a system for studying driver behavior by installing the necessary sensors in the vehicle and using the Predictive Control Data Analysis Algorithm (MPC). Also, in 2019, under the BeSmart [4]

Tunisian-Algerian Joint Conference on Applied Computing, 13-14 December 2022, Constantine, Algeria
†These authors contributed equally.
sameh.bnaounn@gmail.com (S. Ben-Aoun); meriem.belguidoum@univ-constantine2.dz (M. Belguidoum); ahmed.hadjkacem@fsegs.usf.tn (A. Hadj-Kacem)

project, authors collected information from smartphone sensors to analyze data. The authors of [5] used Carla smartphone sensors to improve driver behavior by classifying driving behavior into categories under different external conditions, results compare the different methods applied and demonstrate a practical ability to detect driving behavior.
There has been great need for a system that contributes to improve real time driving in practice at the lowest possible cost. This can be achieved because of the advantages offered by smartphone sensors to TrackDrive. This application is designed based on a combination of microservices [6] and recommendations aspect. As part of this work we have made SysML system [7] [8] to model hardware elements, and microservices for the architectural side in order to reuse and maintenance services. In order to reduce energy consumption and CO2 emissions and improve, consequently, driver behavior. Unlike other initiatives that rely solely on simulation or do not rely on real time, TrackDrive relies on real time data study and processing and then provides the driver with the necessary recommendations to implement them. Eco-driving appears to be an effective way to save energy, reduce emissions in the transportation industry and reduce risky behaviour.

This study is organized as follows: Section 2 presents the related work, and the proposed platform is discussed in Section 3. The conception and the proposed platform are discussed in Section 4. Section 5 details the implementation. Finally, the article has been concluded in Section 7.

2. Related work

Several efforts have been made to improve fuel economy and reduce emissions from road vehicles, including more stringent vehicle emissions standards [2]. Most of these efforts focused on determining drivers' behavior to rationalize environmental driving.

For example, authors at the University of Southampton, UK [3] proposed a system that uses real-time data from GPS and automotive radar to perform predictive optimization of a vehicle's speed profile and guide the driver toward fuel-saving and CO2-reducing behavior. The fuel-efficient speed profile is determined with nonlinear Model Predictive Control algorithm (MPC) optimization, leveraging GPS and long-range automotive radar measurements to learn road geometry and upcoming traffic. The action layer's visual human-machine interface (HMI) is modeled on «eco-speedometers». For the development of this system, the authors applied two tests: on the simulator and on the road. The repeated measures study conducted in the fixed-base driving simulator indicated an overall reduction of 6.09% in fuel consumption. Despite the success of this study, its exploitation and use are limited to certain types of vehicles as these sensors and techniques used are expensive.

As part of the BeSmart project, Kontaxi et al. [4] have developed an innovative smartphone application to assess and improve driver behavior and safety by studying the effect of detailed flight characteristics on the frequency of extreme acceleration and braking. It is divided into two phases; The first consists of identifying the drivers' natural driving characteristics, whereby participants receive no feedback from the app about their driving behavior. In the second, participants receive personal feedback, allowing them to identify critical deficiencies and dangerous behaviors. The application is based on recording the driver's behavior using smartphone sensors

Table 1
Comparison study of some related work

	Microservice aspect	Modeling aspect with SysML	Real time	Driving aspect (parameters)
Proposed App	Yes	Yes	Yes	accelerometre+gyroscope+GPS +type (wheel+road)
[3]	No	No	No	accelerometre+gyroscope+GPS
[4]	No	No	Yes	accelerometre+gyroscope+GPS
[5]	No	No	No	accelerometre+gyroscope+GPS

to read and store sensor data. According to the results, the frequency of intense acceleration and braking is closely related to the maximum speed, the percentage of time spent at speed, and the total duration of the flight. One of the most critical limitations of this study is not to interact with the driver in real-time but after specific experiments, reducing their effectiveness ratios.

Another exciting initiative [5] and in the context of providing low-cost solutions to many fleet management and monitoring problems, Hakim et al. proposed a study consisting of collecting data sensors via the Carla simulator available in smartphones (accelerometer, gyroscope, GPS) to classify driver behavior. Then, after integrating pivotal data from multiple sensors, it applies different machine learning algorithms to classify the time series to evaluate which algorithm leads to the highest performance. The results demonstrated the ability of both machine and deep learning models to achieve greater than 88% accuracy in detecting the driving profile of a one-minute trip. Despite the promising results obtained, these results remain limited. Although simulation essentially mimics real-life attitudes, the effectiveness of this study remains limited only by simulation because the focus on real-life data collection remains more important, as results ratios can change between reality and simulation.

The above discussions highlight the extent to which researchers are interested in contributing to the improvement of driver behavior. However, most of these systems do not deal with real-time data processing. For example, all these systems are just mobile app software. It has never been mentioned that it deals with modeling using the SysML system or the architectural side's microservice aspect in favour of reusing services. Also, the studies discussed do not use wheel or road type as settings. Table 1 summarizes the above methods based on these parameters. In this work, we focused on optimizing the vulnerabilities of previous models with all relevant details and implementation. Therefore, we focus on developing an application that works in real-time by taking advantage of smartphone sensors. First, the user may indulge in driving without being distracted. At the same time, he will learn the details of his trip through recommendations. Secondly, this application will help him reduce his carbon consumption, in addition to avoiding road accidents. We also make this app use microservices for the architectural aspect to reuse the services and maintainability, or add microservices without changing the application. To have many microservices, we may have to add or modify them instead of creating a direct application. Anytime it enables us to add components to things or add some kind of sensor.

3. Proposed approach

This research presents an eco-driving mobile application that works in real time, tracks the trajectories of cars to collect data using mobile sensors, and then uses them to reduce carbon consumption and carbon dioxide emissions as well as being alert to avoid road accidents. Under this work we have made SysML system to model hardware elements (many schemes allow to describe the conceptual aspect), and microservices for the architectural aspect in order to reuse the services and their maintenance. Microservices make it easier to add or modify services rather than reintroducing a live application. Basically, this application focuses on the Maghreb countries, where it will be tested and applied according to the microservice architecture designed flexibly through the IoT methodology.

This app collects information through the global positioning system (GPS) to detect the car's location, road, and movement, as well as the accelerometer and gyroscope to acquire the necessary information about acceleration and sudden braking. Regarding the other information (type of vehicle, type of wheels, etc.) the driver has to fill it in manually. Based on the available information, the system will process and analyze this data and will then send real time recommendations to the driver about speed, acceleration, and unnecessary braking, allowing users to adopt an ecological and economical driving style. This approach will help decrease the possibility of accidents because driving style is a critical factor in road safety. The driver's journey ends with a sum of points for calculating user profiles and knowing how safe their behaviour is. All data will be recorded on the cloud.

4. Design methodology

One of the main challenges in the IoT domain is the lack of methods specifically designed for IoT applications as it depends on a large number of intermediate factors that must be considered in advance at the design stage. In this regard, Arshdeep Bahga and Vijay Madisetti in [9] proposed an IoT system development methodology based on the IoT-ARM reference model [10]. In this work we used this methodology where we combined some language schemes to model the system (requirements, usage status, identification block, activity charts) in a few steps of the methodology so that we can represent the different aspects of the system. SysML is the most popular tool for model-based development, it allows the model of physical aspects. To take advantage of the microservice facilities, we have improved this methodology. Therefore, instead of identifying services, we will use the microservice structure (Step E). See Figure 1.

In this paper, we will focus on the first phase, the requirements phase using SysML diagrams which are the use case diagram and the requirements diagram, the definition phase of the microservices architecture, and the final phase of application development.

4.1. Purpose and requirements analysis

Applying the first step to our system, the purpose is to monitor drivers through the capture of data by smartphone sensors, process them, then send recommendations to inform, notify and motivate drivers in real-time to improve their driving styles, and finally store the data

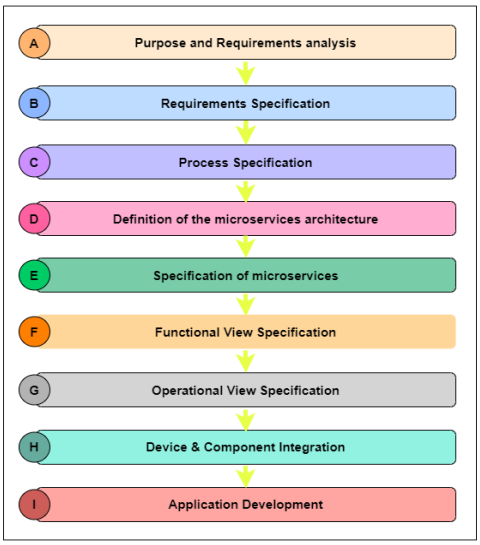

Figure 1: The steps of IoT design.

in the cloud. So the system consists of four microservices that interact with each other : the microservice for capturing data via smartphone sensors, the microservice for processing data to translate it into actionable information, and the microservice for sending recommendations in real time, and the data storage microservice.

4.2. Requirements specification

In this section, we can summarize the first step with the use case diagram and extract the functional requirements. System requirements diagrams describe the main system requirements as well as the sub-requirements needed to achieve them.
Figure 2 shows the use case diagram for the « ECO-TrackDrive » system.

Figure 3 illustrates the requirement diagram for the « ECO-TrackDrive », each requirement has a name, identifier and description. The diagram (see Figure 3) shows that the « ECO-TrackDrive » system consists of the requirements: ManageVehicle, AddNewRoad, CalculationSpeed, ViewRecommendations, SaveRoadAndStart, and ViewHistory: ManageVehicle consists of the requirement LocateVehicle (that is, display the position of the vehicle), of ModifyInformation which allows changing the car information if necessary.

4.3. Definition of the microservices architecture

At this stage, we decompose the system into microservices that interact with each other to meet the system requirements. From a technical point of view, a microservice is realized as an independent process that can be designed, developed, deployed, and operated autonomously. We distinguish between two types of microservice, functional, which realizes the actual business capabilities, or infrastructure, e.g. configuration, etc. Figure 4 shows the microservices

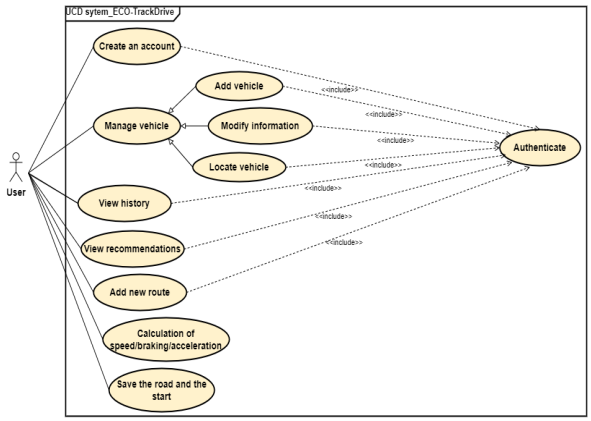

Figure 2: Use case diagram

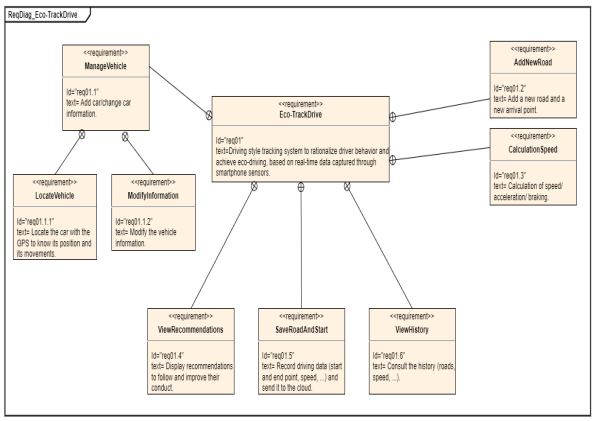

Figure 3: Requirement diagram

architecture of the system. A specific functional microservice realizes each domain.

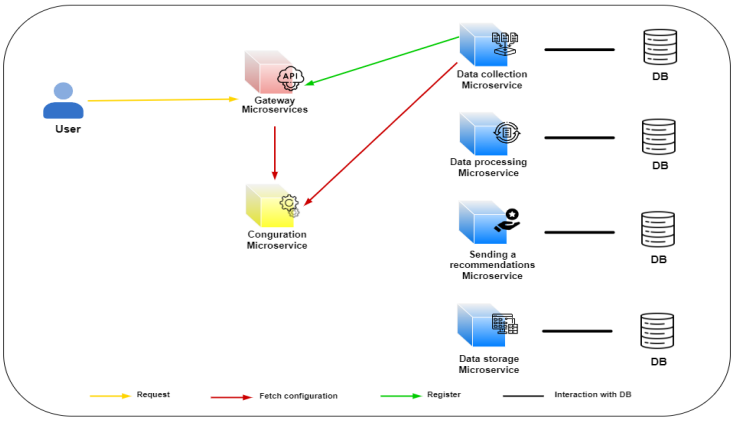

Figure 4: Microservices architecture

The functional microservices are represented with blue cubes (see Figure 4). Multiple instances of a single microservice can be started simultaneously to be autonomous for scalability and fault tolerance, and each microservice has its database to enable autonomy.

Figure 4 the system is also composed of microservices that provide the architecture with infrastructure capabilities: the first one is « *ConfigurationMicroservice* » represented with a yellow cube, allowing to centralize the configuration (a file that contains all the necessary configurations for all the microservices). At startup time, each microservice connects to the « *Gateway Microservices* » represented by a pink cube (see figure 4) to register (with its primary information) and then periodically gives its sign of life (heartbeat every 30 seconds). The « *Gateway Microservices* » allows the location of each microservice. Then, the front end does not directly attack the functional microservices. It goes through the « Gateway Microservices »,

which receives the requests. It sends the request back to the requested microservice to do the necessary processing.

5. Implementation

We implemented a prototype using Flutter, and then we collected and processed data about the driver through mobile sensors.

In terms of front-end development, we used Flutter, a framework for building platform interface; Dart as a text language and interface design. To process and analyze the data, we will use python. The prototype has not yet been fully developed and is currently in the implementation phase. The following forms illustrate how essential steps in the application work.

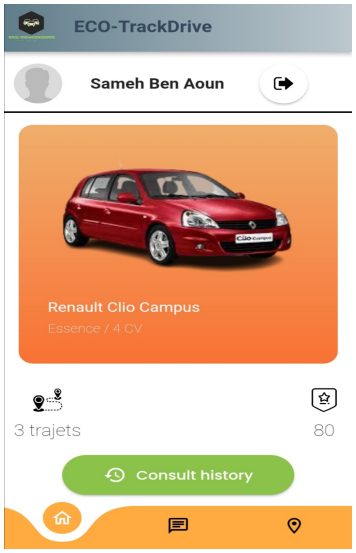

Figure 5: The home page.

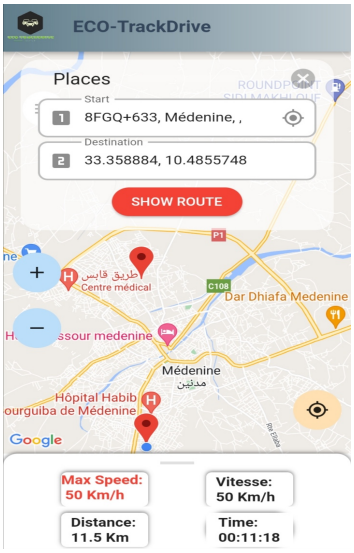

Figure 6: The recommendation.

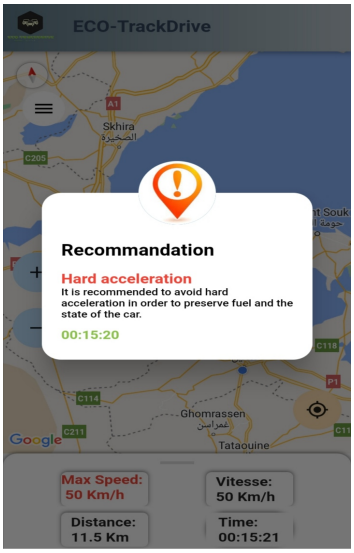

Figure 7: The localisation.

In the first phase, when creating a new account, the user must enter data about his car and register it. Suppose he has an account, in this case, the car data is displayed directly in his profile. The user can locate his car. The user can view their trip report at the end of each trip. The report includes recommendations and tips for the driver, along with the trip duration, the recommended speed, the maximum speed he reached, and the number of kilometers traveled.

In this search, we will focus on key user interfaces. Figure 5 shows the home page interface, where we find a dashboard where the user can consult trip history (rides, period, distance) and see some information about his car. He can also update some information about his account. Figure 6 illustrates the localisation interface enables the user to know his location and the type of road he takes. Figure 7 shows the recommendations, we find recommendations for the user to check his behavior.

6. Acknowledgement

This work was partially supported by the LABEX-TA project MeFoGL: « Méthodes Formelles pour le Génie Logicie ».

7. Conclusion

ECO-TrackDrive is able to detect acceleration, braking, and overspeed instances by integrating smartphone sensors and GPS data. As well as exploiting the type of car wheels and the type of road used. Considering that we are in a master's, we made SysML to model the hardware stuff, the microservices for the architecture aspect for the interest of reuse and maintainability of services. As a future attempt, we are going to add an AI approach. Insurance companies also can make profit of such applications to ensure driver continuity in environmental driving and to overcome some misconceptions when driving. We aim, also, to benefiting from microservices at the level of service reuse and the possibility of maintaining them (adding or modifying).

References

[1] statista, Global transport co2 emissions breakdown 2020, 2020. URL: https://www.statista.com/statistics/1185535/transport-carbon-dioxide-emissions-breakdown.

[2] N. Xu, X. Li, Q. Liu, D. Zhao, An overview of eco-driving theory, capability evaluation, and training applications, Sensors 21 (2021) 6547.

[3] J. Fleming, X. Yan, C. Allison, N. Stanton, R. Lot, Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements, IET Intelligent Transport Systems 15 (2021) 573–583.

[4] A. Kontaxi, A. Ziakopoulos, G. Yannis, Trip characteristics impact on the frequency of harsh events recorded via smartphone sensors, IATSS research 45 (2021) 574–583.

[5] S. B. Brahim, H. Ghazzai, H. Besbes, Y. Massoud, A machine learning smartphone-based sensing for driver behavior classification, arXiv preprint arXiv:2202.01893 (2022).

[6] J. Sorgalla, F. Rademacher, S. Sachweh, A. Zündorf, On collaborative model-driven development of microservices, in: Federation of International Conferences on Software Technologies: Applications and Foundations, Springer, 2018, pp. 596–603.

[7] P. Roques, Modélisation de systèmes complexes avec SysML, Editions Eyrolles, 2013.

[8] S. Friedenthal, A. Moore, R. Steiner, Omg systems modeling language (omg sysml) tutorial, in: INCOSE Intl. Symp, volume 9, 2006, pp. 65–67.

[9] A. Bahga, V. Madisetti, Internet of Things: A hands-on approach, chapter 5, pages 99–115. Arshdeep Bahga and Vijay Krishna Madisetti, 2014.

[10] M. Bauer, N. Bui, J. D. Loof, C. Magerkurth, A. Nettsträter, J. Stefa, J. W. Walewski, Iot reference model, in: Enabling Things to Talk, Springer, Berlin, Heidelberg, 2013, pp. 113–162.