10.20378/irb-58600

# Why Open Source is Driving the Future Connected Vehicle

Robert Höttger, robert.hoettger@fh-dortmund.de

IDiAL Institut, Fachhochschule Dortmund, Otto-Hahn-Str. 23, 44227 Dortmund



*Abstract:*

*With the year 2018, modern cars are driven by sophisticated software interconnected via a heterogeneous Electronic Control Unit (ECU) network that consists of a huge amount of sensors, actuators, processing units, interfaces, gateways etc. Having the evolution towards autonomous driving, electrification, and vast cloud incorporation in mind, questions arise regarding safety, security, intellectual property, open source, standardization, data governance and more.*

*This paper outlines some of those upcoming questions, derives challenges and proposes possible solutions. Such solutions are driven by the ITEA3 project APPSTACLE[1] and its Eclipse project namely Kuksa.*

JEL Classification: L62, L86

Keywords: Automotive, Kuksa, Internet of Things, IoT, Cloud, Cloud Computing, Eclipse Kuksa

[1] Funded by the federal ministry of education and research Germany (BMBF) under funding number 01|S16047D.

# 1 Introduction

In the automotive industry, innovation is predominantly defined by the software that increasingly targets autonomous driving in a dynamic environment. In contrast to the smartphone market however, the automotive domain yet missed to fully exploit its market potentials due to the lack of open standards, connectivity protocols, ecosystems, frameworks, and commonly accessible technologies. While nearly every car manufacturer advanced its infotainment systems as well as connectivity technologies to communicate with the outside world (i.e., the worldwide web, but also local infrastructure, cf. edge computing, or direct communication with the OEM), technologies are proprietary and consequently not used in a cross-vendor manner. On the contrary, the smartphone industry has gained a vast global market with countless applications, vendors, services, and businesses. Just like vehicles, smartphones have a variety of sensors, actuators, and interfaces but only the provisioning of standards, APIs, and services in an open source manner made this industry to what it is today.

With Eclipse Kuksa, initial implementations for an open source ecosystem amongst the connected vehicle domain are given in order to tackle a breakthrough from closed source proprietary development activities towards an open and standardized environment. Such an environment opens the automotive market and lets OEMs, tool vendors, and various tier suppliers cooperate and focus on unique selling points of more advanced applications and infrastructures. Therefore, three major platforms are required to support (a) the vehicle, (b) the cloud, and (c) the IDE as seen in Figure 1: Overview of the Eclipse Kuksa Platforms.

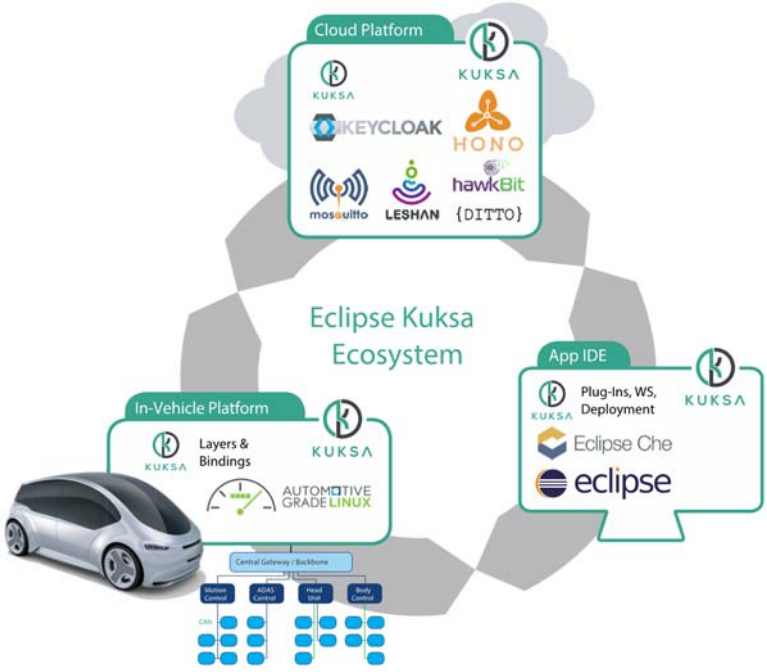


Figure 1: Overview of the Eclipse Kuksa Platforms

To cope with technological challenges, an extensive state-of-the-art analysis has been performed to investigate and identify appropriate technologies, protocols and methodologies that can be utilized, adapted, and extended for vehicle domain purposes. Despite adding necessary features to existing technologies regarding, e.g. security or real-time capabilities, revealed technological gaps are in focus to be filled in order to form a secure and easy-to-use environment for vehicle applications in a cross-vendor manner.

## 2 Related Work

AGL (Automotive Grade Linux) [1] was identified as the most appropriate in-vehicle platform over Apertis, Ubuntu Core, SuSe Embedded, Legato, QNX, and Android Automotive during investigating potential platforms in late 2017.

The seven major weighted features were

- Platform features
- Platform runtime
- Application runtime
- Application development and SDK
- App Store
- Licensing
- Developer community

More information about detailed results can be found at the ITEA portal [2].

In general, Eclipse Kuksa [3] results are published at the Eclipse Kuksa GitHub repositories[2], that hold the EPL2.0 license and is industry friendly correspondingly.

With its current status, Eclipse Kuksa provides a customized AGL distribution with layers and bindings that support connectivity protocols such as MQTT, a Cloud API, an In-Vehicle API for OBD data, an Eclipse hawkBit client for over-the-air (OTA) updates, and data format standardization implementations for the W3C standard. In addition, a dedicated cloud instance has been developed based on Eclipse Hono [4] to form the backend for cloud services, e.g., smart traffic services, OTA update managers, and others. Grafana [5] is the counterpart to Eclipse Hono and forms the frontend in order to visualize data in various forms and diagrams. Keycloak [6] is one of the major security technologies used to cover authentication aspects. Eclipse hawkBit [7] is used to cover device management activities as well OTA concepts. A separate APP

[2] https://github.com/?q=kuksa, accessed 23.01.2019.

store[3] has recently been added to Eclipse Kuksa, that communicates with the cloud and allows users to access and install different in-car applications for their vehicles.

## 3 In-Vehicle Platform

The Eclipse Kuksa In-Vehicle Platform is based on AGL and features various additional layers containing different technologies as services in order to meet connected vehicle demands. First of all, an *agl-kuksa* layer contains scripts to combine layers for an automated build system. On top of that, several applications and services exist such as

- *elm327-visdatafeeder* – A hardware specific OBDII data reader that reads data from an OBDII in vehicle interface and sends this data to the *w3c-visserver*.
- *w3c-visserver-api* – This W3C Vehicle Information Specification API is used to have a uniform data exchange across vehicle data.
- *kuksa-hawkBit* – This API supports hawkBit connections in order to communicate with the OTA update mechanisms.
- *direct-access-api* – This API provides CAN message exchange between the cloud and vehicles via Websocket communication.
- *datalogger-http* – This application sends data from the vehicle to an Eclipse Hono instance via http. It is an example application in order to provide an easy to use starting point for developers getting in touch with Vehicle-Cloud data interchange.
- *datalogger-mqtt* – This application sends data from the vehicle to an Eclipse Hono instance via MQTT similar to datalogger-http but using MQTT instead of HTTP.
- *remoteAccess* – This application subscribes to a control topic of Eclipse Hono and receives sent commands.
- *kuksa-appmanager* – This hawkBit-based appmanager deploys in-vehicle applications based on docker containers and other formats.
- *email-notifier* – This example application exchanges data with an email-server in order to send emails to a configurable email address. This can be used in order to notify user about in vehicle error signals for example.

[3] https://github.com/eclipse/kuksa.cloud/tree/master/kuksa-appstore, accessed 23.01.2019.

Given the above applications, APIs, and services, developers get necessary basic functionalities and easy starting points in order to start developing in-vehicle software for future connected vehicles.

## 4 IDE Platform

The Eclipse Kuksa IDE Platform is based on Eclipse Che and adds additional features, i.e., Yocto SDK and target IP definitions. Additionally, build commands are included in order to build custom AGL-Kuksa images based on Yocto, Bitbake, and the OpenEmbedded project. While the final product contains GPL code, build images are not (planned to be) distributed as binary release files. However, application binaries are planned to be accessible for future releases.

The browser-based Kuksa-Che-IDE requires a server deployment and can be accessed from any clients, even mobile phones, in order to program in-vehicle and cloud applications or services. Consequently, development activities are platform independent, do not rely on any local software, improve build stability and performance compared with local IDEs, support multi user development, and many more. Not requiring any form of configuration and having the correct repositories and commands automatically checked out significantly eases software development of Eclipse Kuska software.

Being able to deploy applications to vehicles and at the same time deploying services to the cloud allows implementing new innovative concepts on a less complex environment. Community based parking, smart traffic, navigation, reliable remote maintenance, or platooning are just some of the use cases this environment can support being implemented with.

## 5 Cloud Platform

The Eclipse Kuksa Cloud Platform backend is based on Eclipse Hono and Eclipse hawkBit in combination with an InfluxDB database. A springboot application is added in order to store any receiving data in the InfluxDB database in order to visualize arbitrary data using the Grafana [5] frontend.

The Eclipse Kuksa Appstore is able to communicate with Eclipse Hono and Eclipse hawkBit to transmit and manage application transmissions, installation processes, and user authentications. Different account types (admin, user, group user, OEMs, … ) are supported with different roles in order to manage applications, devices (vehicles), and access rights. Since this technology is currently just a demonstration, any billing system is currently not implemented.

## 6 Why Open-Source

Governance, IP management, collaborative infrastructure, and recommended development processes are just some of the benefits developers can make use of when conducting a project under the Eclipse Foundation umbrella. With the latest GitHub report[4], over 31 million users share experience and knowhow in order to learn, share, and work together to build software across more than 100 million repositories as of November 2018. Consequently, having open source code on GitHub has proven to help in reaching project goals, get consumer attention, and build a community to further enhance, sustain, and maintain the code (for free!).


Figure 2: A business friendly ecosystem with a common platform and advanced products

Additionally, an open source software community can ensure sustainability, mutualization, standardization, and necessary project metrics. Sustainability, for example, can ensure the continuous code maintenance even if committers leave the project. Mutualization can concern efforts and resources in order to invest manpower into the unique selling points and advanced added value products (cf. Figure 2) that may even significantly advance in terms of functionality since interfaces can be used across broader application fields and technologies. Standardization also plays an important role. An example is the MQTT protocol and technology that was published open source by IBM after 15 years of internal development. After just five years of being used open source, it became an ISO/IEC standard. Finally, metrics ensure an easy way of deriv-

[4] https://github.com/about, accessed 23.01.2019.

ing community activity. Regular releases, number of commits, pull requests, closed and open issues etc. are great indicators of code liveliness.

Furthermore, having vendor-neutral governance structures can ensure a free collaborative environment based on intellectual property management, licensing, project processes, ecosystem development environments, and a safe infrastructure as shown in Figure 3. Consequently, open source governance can achieve independent control of how decisions are made, policies are established and disputes are resolved as a matter of successful collaboration. Also, collaboration amongst companies requires due diligences on the commonly built intellectual property that is provided by Eclipse such that software is available to being used by anyone even within commercial products. The infrastructure that has been evolved over a long period of time is suitable for any kind of projects and processes behind this infrastructure, makes development easy, comprehensible, and efficient.

In addition, open source software (OSS) has proven to be a successful model e.g. Linux, Apache HTTP, Mozilla Firefox, Eclipse Java IDE and others. Since OSS is free, people can easily try out the software without any efforts and developers can gain experience with various technologies early without requiring significant training. Technologies can be customized and time to market periods can be much shorter. By joining an OSS community, companies can mutualize maintaining costs with the community behind the OSS. Source code can be more effectively aligned to what the user really need whereas code quality and stability can be higher since the community can report and fix bugs.


Figure 3: Basics of open collaboration

## 7 Conclusion

With the provisioning of the App store and all related frameworks, APIs, and platforms, Eclipse Kuksa will form a standardized open source basis to leverage open source technologies for the connected vehicle domain.

Development parties will be given the opportunity to concentrate on unique selling points such as autonomous driving components without having to invest efforts towards infrastructure, basic services (e.g. security, OTA update services, ...), or mandatory basic platforms for the vehicle, the cloud, or the IDE.

With the Eclipse Public License (Eclipse Kuksa is licensed EPL2.0), involved parties can still keep extensions, additions, or modifications proprietary and hence possess their development IPs in order to comply to company or European commission regulations.

As another consequence, OEMs, suppliers, and tool vendors will significantly benefit from modular software components, free community technologies, and new businesses arising from an ecosystem that unifies data, protocols, APIs, and connectivity solutions across the automotive domain.

In order to retain the technological innovation and compete with FAANG (Facebook, Apple, Amazon, Netflix, Google) that will eventually try to conquer this tremendously growing market, i.e., the vehicle sector, OEMs, suppliers, and tool vendors need to stand up against their spurious bias to inherently construct silos of intellectual properties and relinquish the benefits of open source communities across the globe and its technologies.

## 8 Bibliography

The Linux Foundation, “The Linux Foundation Projects - Automotive Grade Linux, ” 2019. [Online]. Available: https://www.automotivelinux.org. [Accessed 23 January 2019].

APPSTACLE Consotium, “D1.1. APPSTACLE . Specification of In-Car Software Architecture for Car2X Applications,” 2019. [Online]. Available: https://bit.ly/2Mrxcyq. [Accessed 23 January 2019].

The Eclipse Kuksa Project, “Eclipse Kuksa: Open Platforms for Connected Vehicles,” 2019. [Online]. Available: https://www.eclipse.org/kuksa/. [Accessed 23 January 2019].

The Eclipse Hono Project, “Eclipse Hono - Connect. Command. Control.,” 2019. [Online]. Available: https://www.eclipse.org/hono/. [Accessed 23 January 2019].

Grafana Labs, “Grafana: Tope open platform for beautiful analytics and monitoring, ” 2019. [Online]. Available: https://grafana.com. [Accessed 23 January 2019].

Red Hat, Inc., “Eclipse Keycloak: Open Source Identity and Access Management For Modern Applications and Services,” 2019. [Online]. Available: https://www.keycloak.org. [Accessed 23 January 2019].

The Eclipse hawkBit Project, “Eclipse hawkBit: IoT. Update. Device.,” 2019. [Online]. Available: https://www.eclipse.org/hawkBit/. [Accessed 23 January 2019].