HyVar NEWSLETTER N°4, April 2018

In this issue:

- ✓ HyVar Final Demonstrator
- ✓ Integration in Industrial Practice

HyVar Final Demonstrator

In the last years we have seen an increased interest in advanced driver assistance systems (ADAS) that enhance vehicle safety and fuel-efficiency by helping with monitoring, warning, braking, and steering tasks.
Although ADAS applications are still in their early days, original-equipment manufacturers (OEMs) and their suppliers realize that they could eventually become the main feature in differentiating automotive brands, as well as one of their most important revenue sources and a first step towards the realization of fully autonomous vehicles.
In the Demonstrator we show how to reconfigure the car dynamically, based on driving style information calculated from CAN (Controller Area Network bus) parameters (such as engine speed, vehicle speed and brake use), and add aftermarket Gear Advice and Brake Advice functionalities only if required. Moreover HyVar allow the user to express preferences about the optional features to be included in the Infotainment digital dashboard. The preferences can be provided through an Android application on the smartphone, allowing the driver's persona and preferences to be moved from car to car.

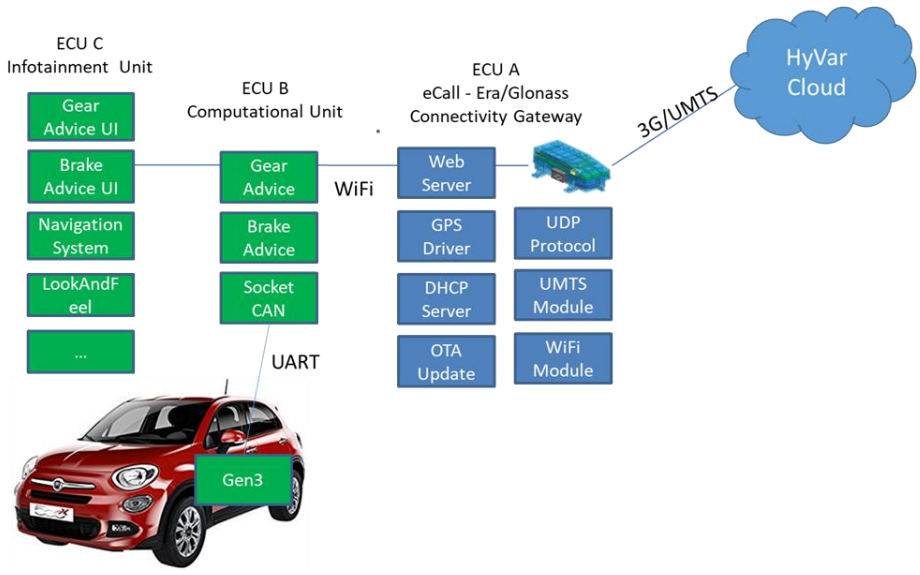

Figure 1 - Final Demonstrator architecture

The End-to-End Final demonstrator architecture includes three ECUs as illustrated in Figure 1.
The first (ECU_A) is already used in the Initial demonstrator to implement the logical functions of:

- Location and positioning module: providing the information required for Emergency call functionality.
- Connectivity Gateway: allowing bidirectional exchange between the vehicle and the HyVar server.

ECU_B represents the computational unit. Its purpose is to collect vehicle status data by listening to the CAN bus, to parse data and forward it adding additional information such as gear and brake advice when the corresponding feature is present.
ECU_C implements the infotainment unit. It handles the graphical user interface and is responsible for displaying real time data to the driver. Moreover it can optionally offer functionality like a navigation system, a background color switcher, a SOS button to manually start an emergency call (see Figure 2).
The final demonstrator has the goal to show the application of the whole basic HyVar process, as well as the new and more advanced functionalities.

To manage high personalized over-the-air update the Cloud need to collect and capitalize on information from car (sensors, user profiles) before determining (assembling, distributing) appropriate update.
For example, suppose a car has initially the base version of user interface.
The cloud collects the information from car and identifies that the driver is a user with attitude to use wrong gears, and with attitude towards harsh braking.

HyVar solution triggers the installation of Gear Advice and Brake Advice and sends update request to the car. The update package is downloaded during the drive and applied at switch-off. At key-on the new adapted version of user interface is displayed.

Figure 2 - Graphical user interface

Integration in industrial practice

The integration of the Domain Specific Variability Language (DSVL) with industrial practice is essential for adoption and usability. To this end, we adapted the DSVL to support common development practices. UML statecharts provide a flexible possibility to model the behavior of systems and are an industrial standard. Statecharts are a modeling technique to give a behavioral description of a system. They consist of states and transitions between these states. Transitions can be guarded and, thus, are only triggered if a certain condition is satisfied. Moreover, statecharts can model concurrent behavior and can be structured hierarchically.
In the DSVL, we employ Yakindu statecharts (https://www.itemis.com/en/yakindu/state-machine/).
To provide executable programs, code generators can use statecharts to generate executable code. Generally, statecharts are independent of a programming language, therefore, arbitrary code can be generated out of statecharts. Within this project, the target language for the statecharts in the DSVL are C and Java.

However, the statecharts created with the DSVL can be utilized with other code generators, e.g., to generate Abstract Behavioral Specification Language (ABS) code.
As we provide support for statecharts, existing systems can be integrated with the Scalable Hybrid Variability for Distributed Evolving Software Systems (HyVar) methodology.
Extractive development of an SPL based on an existing system modeled by statecharts is possible. To this end, the existing software is analyzed and features are determined. Subsequently, implementation artifacts are extracted according to the determined features.
With the integration of statecharts in the DSVL, current industrial practice is supported and industrial users can integrate their existing systems modeled by statecharts. Moreover, as arbitrary code can be generated out of statecharts, the methodology of HyVar can integrate with existing target platforms.

HyVar Events

- ✓ General Meeting in Tromso, January 2018

Next Events

- ✓ HyVar Final Review in Brussels, Apr 24, 2018

HyVar project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644298

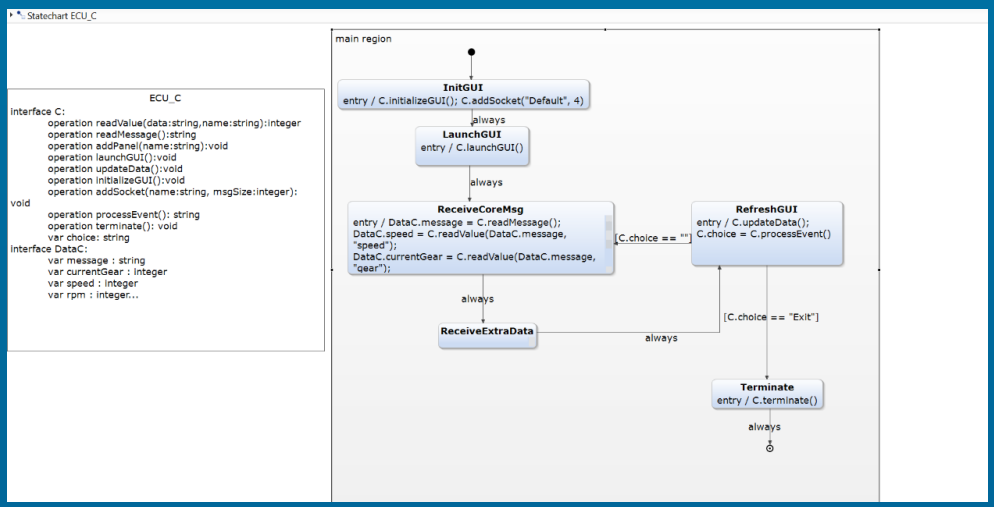

Figure 3 Statechart of the Base User Interface