THE LATEST TRENDS AND TECHNOLOGIES IN THE IT WORLD

LOGEEK MAGAZINE

5/2017 LTS# LOGEEK NIGHT

A New Dimension in Car & Driver Connectivity

0+

LUXOFT

AUTOMOTIVE ISSUE

TECHNICAL

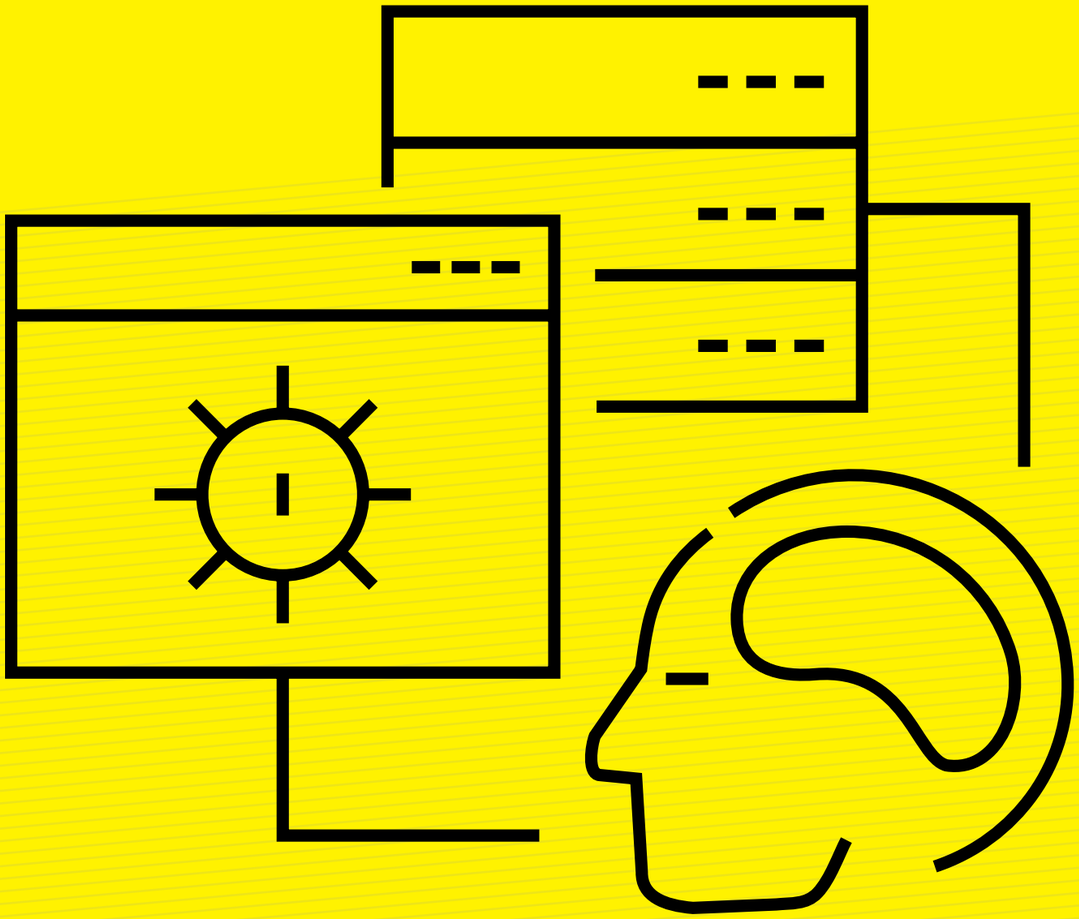
CORPORATE VIDEOS

LoGeek Magazine, 0+
Editor: **Dilya Salakhetdinova**
Comments, suggestions, or want to be a part of the next issue?
Write to LGmagazine@luxoft.com

TENTS

5/2017

LOGEEK



Static Code Analysis:

Optimistic Expectations, Pessimistic Views and the Reality

ALEXEY RESHTA

INTRODUCTION

Static code analysis (SCA) is the process of searching for problems in a program source code before running it. It could be a manual code review process or automatic searching of the code problems using special software tools called code analyzers. This article describes some aspects of using automatic analysis of C++ code. Historically, C++ is one of the most complicated programming languages. It is based on C language and different programming paradigms (procedure programming, objects-oriented programming, generic programming, multithreading programming). The language has plenty of danger operators (as "goto"), objects (as pointers), functions (as strcpy) and types (as union). It is easy to use this language the wrong way.

SAC is very important for developing using C++. The SAC idea is to find code problems immediately after a developer creates the code. The longer code issues remain hidden, the more they will cost to fix. So it's a good idea to run an analyzer and fix any problems before submitting to the repository. Another option is to use it as part of a building session on the building server, where the log of issues is sent to developers after building.

ADVANTAGES

An analyzer can find:

- Hard errors such as using null pointer, using non-initialized variables, buffer overflow, memory leaks, dividing by zero, non-caught exceptions, dangerous access to data in multithreading, etc.
- Some warnings, as present in the code of non-initialized variables, unreachable code, etc.
- Some notes about bad style, the danger of using of program language instructions, etc.

Additionally, some analyzers support:

- Reporting code statistics: Count of files code lines, comments, functions, classes, etc.
- Code issue tracking and code issues statistics
- Reporting issues according to coding standards (for example SEI CERT C++ Coding Standard or MISRA C++)

All analyzer features look useful, but some investigation reveals that there are plenty of developers disappointed by the working of SAC analyzers (for example: "Why Don't Software Developers Use Static Analysis Tools to Find Bugs?)".

DISADVANTAGES

A developer's disappointment can result from a few issues:

- Covering up code problems – there are a lot of false negatives: Some problem is present in the code, but the analyzer doesn't report it. The problem can be found another way, through manual code review, run-time analysis, testing and fixing. Mostly it's a surprise for the developer that used some analyzer and hoped that errors were reported and fixed.
- Spending the developer's time reading and analyzing false positives – the analyzer reports issues that are absent in the code. Such messages must be disabled to be sure that the analyzer report includes only real issues.
- Spending time on setting analyzer parameters – an analyzer has to receive all definitions, including paths that are used by the compiler. Mostly this information is generated by building procedures based on jam, make, bibtbake and other building systems. The integration of an analyzer in such systems is not trivial work.
- Reliability of the building process – if analysis is integrated into the building process, errors can hang up or break down the process..
- Incomparable results from different tools – every analyzer uses its own methods and knowledge base. They often report different problems.

Deficiencies that are common to any software can disappoint developers, too, such as:

- Costs – some commercial tools are expensive.
- Features – needed features aren't covered.
- Support – not all analyzers are supported well.
- Effort – developers need to spend time and effort to study analyzer features and parameters.

- Tool customization – sometimes it's difficult to customize an analyzer to fit the requirements of a project.

As a result, the use of analyzers is questionable for some projects. Another consideration is how they will be used. Some projects will use them under pressure from a customer or a company's processing department that prefers a formal style. They disable issue reports rather than really use it for code quality improvement.

A FEW WORDS ABOUT OUR EXPERIENCES

Our customers require the use of SAC. Lint reports are a part of release delivery. As we develop software for automotive, one of the main demands is the MISRA compatibility report. We use Gimpel PC Lint for analysis.

We explored the idea of using CodeSonar, cppcheck and clang analyzers too. To test how the analyzers work, a small C++ sample was created. The source is here: https://github.com/areshta/SCA-simple-1/blob/master/sca_cpp.cpp. This code includes 38 general C++ issues of different priorities, six of which are catastrophic.

For this example no analyzer reported more than 50% of the errors. Combined together, two analyzers managed to report more than 50%. Some errors were not reported at all.

The results are not published here with the purpose of finding out which analyzer is better. The sample included only a few issues out of thousands of possible code issues. If you create another sample with a different set of errors you would have other results to compare. Results depend on analyzer options, too. The purpose of the investigation was to estimate the situation with SAC in general.

We have a practice of comparing analyzers not only by results, but also by the coverage of different requirements. Each analyzer has its own advantages and disadvantages. For example, Grammatech analyzer CodeSonar is MISRA supporting. Take a MISRA 2008 mapping board. I It looks nice, but if we open the board, we will find the set of MISRA rules 10–15 to be missing. What are these chapters? They are "Derived Classes," "Member Access Control," "Special Member Functions," "Templates," and "Exceptions handling." Surprise! For the most part, OOP C++ is not covered!

Why?! If we investigate further, we find that CodeSonar is based on the Common Weakness Enumeration (CWE), which is common for several programming languages. A lot of C++ specific issues are absent there. So, if C++ is supported according to SCA analyzer documentation it is not a warranty that it covers all C++ issues that a developer needs.

Mostly we use Gimpel PC Lint. PC Lint was supposed to support C++11/14, but it does so badly. The new PC Lint Plus does it much better, but is still undergoing beta testing. It might seem strange, but we use the beta version for real projects. It suits our purpose better than the previous Gimpel product.

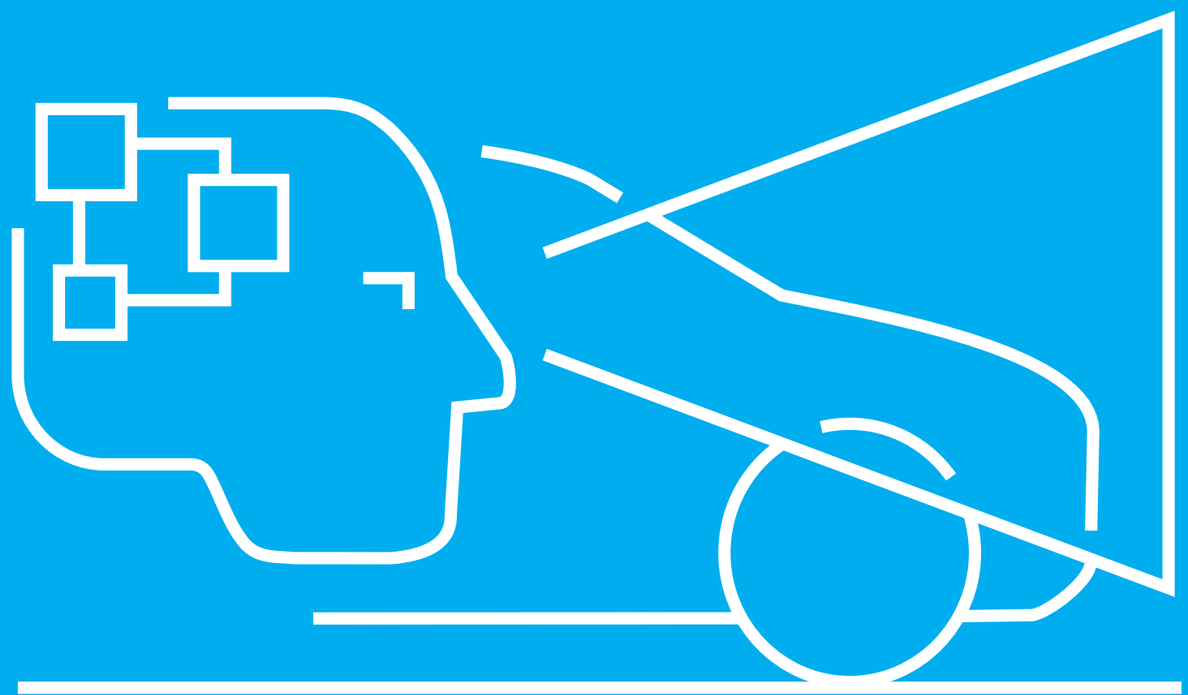
Open source products cppcheck and clang work well. Some our projects utilize them, but it's difficult to find a total list of their messages. You will not find mapping of their rules to programming standards. If your customer requests that you follow MISRA rules you cannot use reports from these tools. You have to use PC Lint or something else that supports MISRA.

SUMMARY

Sure, SAC analyzers are useful, but before using a tool you need investigate it to make sure it matches your project requirements. First of all you need to find out if you will follow some standard as MISRA or CERT. If yes, you will need to investigate popular commercial analyzers. If not, then open source analyzers might be enough. To maximize results, two or more analyzers can be combined to work together.

ALEXEY RESHTA

Aleksey Reshta was born in 1961. In 1986 he graduated from the Odessa Polytechnic Institute. For about twenty years he taught at ONPU (OPI) and worked as a freelancer. Since 2007, he works in Luxoft as a senior programmer. In parallel with the software development, he was engaged in academic and educational programs. He is fond of robotics.



Lane Assist SYSTEMS

TA MINH SON

Nowadays, computing technologies have turned cars into mobile computers. Many of them are designed to minimize the occurrence and consequences of traffic collisions – in other words, they are made for safe driving. Now we have blind spot detection, adaptive cruise control, automatic parking and anti-lock braking systems. Recently, Lane Assist System (LAS) has become one of the most commonly-integrated technologies. It includes lane departure warning (LDW) and lane keeping assist (LKA) systems. LAS is primarily designed to prevent high-speed accidents on highways and freeways. An efficient LAS alleviates accidents related to unintended movement across lane markings. According to the National Traffic Safety Administration (NHTSA), up to 37% of deaths are related to lane deviation. Therefore, systems such as LDW or LKA help save lives.

SYSTEM ARCHITECTURE

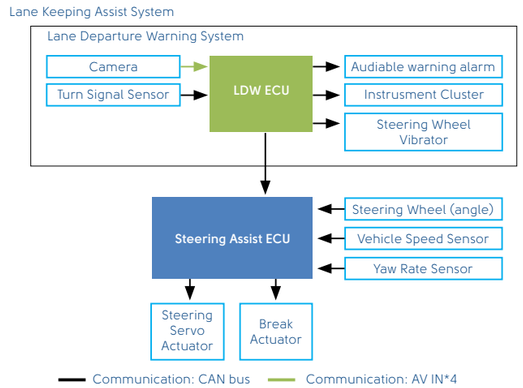


Figure 1. General Lane Assist System Architecture

The system generally employs a camera, which continuously takes pictures/images of the road ahead. Visual images are then sent to an electronic control unit (ECU). The ECU detects the shape and position of lane markings. Systems issue warnings only when the driver does not use the turn signal in the direction of the lane change. To alert the driver, the ECU triggers signals to warning devices such as audible alarms and vibration or visual indicators (HMI).

After the warning, if there is no action on the driver's part, the Lane Keeping Assistance system is activated. Steering assists the electronic control unit, which calculates a target steering torque based on the lane marking data, including steering wheel angle, vehicle speed, and yaw rate, and then sends a steering torque signal to a steering servo actuator and brake actuator which causes the car to return to a safe position.

LANE DEPARTURE WARNING

Lane Departure Warning (LDW) is a warning system that alerts the driver when the car drifts near, onto, or over lane markings. LDW commonly uses beeps, vibration or graphic symbols to notify the driver when the vehicle begins to move out of its lane on freeways and arterial roads unless a turn signal is on in that direction or (in some cars) the brakes are applied.

This technology mainly uses vision sensors with low-cost cameras to identify white or yellow lines in order to analyze and evaluate a situation before issuing a warning. Typically, vision sensors are mounted on the upper edge of the windshield near the rear view mirror. Because of its location, manufacturers focused on reducing the sensor's size. In addition, to better operate in hotter climates and regions, manufacturers also increased the sensor's maximum operating temperature. To address these two challenges, manufacturers developed a low-processing load and a high-performance algorithm, which only require a general-purpose microcomputer as opposed to image-processing integrated circuits (ICs). Eliminating ICs allowed the sensor size to be reduced, which also allowed for an increase in the maximum operating temperature. Some manufacturers also use laser or infrared sensors to monitor lane markings on the road surface. This is part of the circle of safety – the three most common and useful driver assists that protect you from the front (adaptive cruise control and forward collision warning), the side (lane departure warning), and the rear (blind spot detection).

LDW is usually turned on automatically. The driver can disable the lane departure warning, but since it's a safety item that reduces accidents, the majority of automakers default to LDW-on when the driver restarts the car. Some even remember when the driver turns it off. The lane departure warning kicks in between 30 mph and 40 mph (or the rough metric equivalent of 50-65 kph). It isn't intended for low-speed, stop-and-go driving, partly because the camera couldn't capture enough of the lane markings.

Figure **2**. LDW Audible, Vibration, Graphic

LANE KEEPING ASSIST

Lane Keeping Assist (LKA) is built on the basis of the LDW with the ability to adjust the car and return it to the correct lane. It is a far better idea to control the wheel when using electric power steering to keep the car on track. Some systems incorporate adaptive surveillance technology or radar-based surveillance to increase safety. As with the LDW, the LKA is interrupted when drivers switch on the turn signal or when vehicles accelerate to cross another vehicle.

Vision sensors installed behind the windshield detect lane markings in front of the vehicle and calculate their shape and position relative to the lane. When the system detects the vehicle straying from its lane, LKA applies a slight counter-steering torque, trying to prevent the vehicle from moving out of its lane. The brake actuator controls the brake pressure of each wheel individually to generate intended movement.

Some systems incorporate adaptive by a reaction mechanism, which means that the wheel is adjusted only after the vehicle has left the current lane. Meanwhile, others operate on an active basis, which keeps vehicles in the middle of the lane (these systems are also known as lane centering assist, or LCA). Sometimes the steering change is effected by braking the opposite front wheel and pivoting the car back into the lane. The car can also move back by turning the steering wheel. In either case, the driver can easily overcome the car's intentions by turning the wheel.

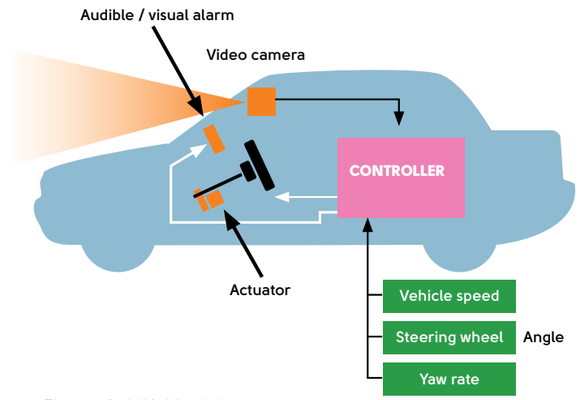


Figure **3**. LKA Modules

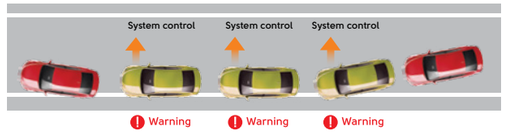
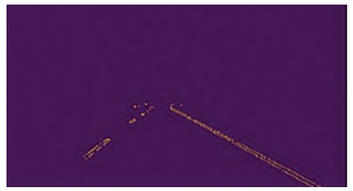


Figure **4**. System controls steering the vehicle back to the correct lane

HOW DOES THE CAR SEE?

One of the solutions providing lane vision is the use of Hough transform and Canny edge detector to help detect lane lines. This comes from real time camera fotage fed from the front-end camera of the automobile. A basic flowchart of how a lane detection algorithm works to activate a lane departure warning is shown below.



Hough transform output

Canny edge detection output

Figure **5**. Lane Detection Algorithms

LIMITATIONS

The limitation of this technology is its dependence on the quality of painted lines on the road: LDW can only be effective when paint lines are easily recognizable. Lane marking dots are sometimes harder to track, especially if their coloring has faded. Low light, fog, rain, snow and extreme weather conditions can make this system ineffective.

CONCLUSIONS

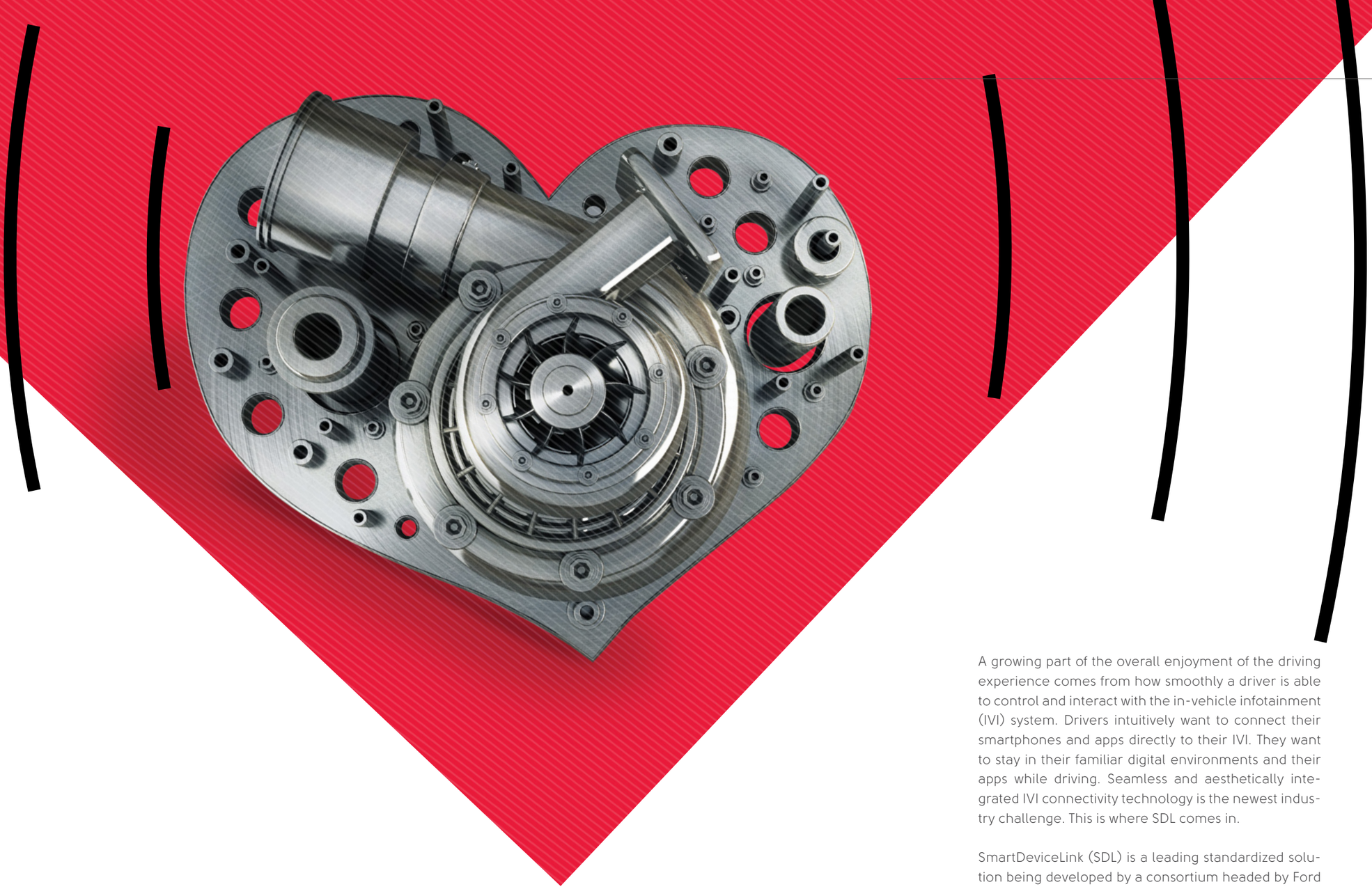
The lane assist system makes sense for driving a lot of highway miles. Lane keep assist makes more sense in keeping the driver from drifting across lanes, and lane centering assist is better still. Although very useful, all lane monitoring and control systems are limited to the quality of lane markings, especially difficult for roads under construction and in bad weather. In the end, these technologies just support safe driving; they are not automatic steering technology. Make sure you are awake at all times while driving, even if the vehicle is equipped with Lane Departure Warning (LDW) or Lane Keeping Assistance (LKA).

REFERENCES

- https://en.wikipedia.org/wiki/Lane_departure_warning_system
- https://www.google.com/patents/US5373911
- https://www.extremetech.com/extreme/165320-what-is-lane-departure-warning-and-how-does-it-work
- http://www.mitsubishi-motors.com/en/corporate/pressrelease/corporate/detail429.html
- http://www.autoguide.com/auto-news/2014/08/lane-departure-safety-systems-passive-vs-active.html
- https://mycardoeswhat.org/safety-features/lane-departure-warning/
- https://mycardoeswhat.org/safety-features/lane-keeping-assist/
- https://www.audi-technology-portal.de/de/elektrik-elektronik/fahrerassistenzsysteme/audi-active-lane-assist
- http://www.nissan-global.com/EN/TECHNOLOGY/OVERVIEW/ldp.html
- http://www.cvel.clemson.edu/auto/systems/lane-departure-warning.html
- https://www.lifewire.com/g00/what-are-lane-departure-warning-systems-534822

TA MINH SON

Son has 8+ years of working experience in C/C++ software development. He has performed various roles in projects: Senior Engineer, Bridge Engineer, Technical Architect and Project Leader. Son has experience in automotive and telecom realms, usually focusing on embedded technologies, but he also has interest in web and mobile applications. In addition, he works on the Continental Singapore Project leading the development team.



SMARTDEVICELINK

A New Dimension in Car & Driver Connectivity

IEVGEN NEBRAT

A growing part of the overall enjoyment of the driving experience comes from how smoothly a driver is able to control and interact with the in-vehicle infotainment (IVI) system. Drivers intuitively want to connect their smartphones and apps directly to their IVI. They want to stay in their familiar digital environments and their apps while driving. Seamless and aesthetically integrated IVI connectivity technology is the newest industry challenge. This is where SDL comes in.

SmartDeviceLink (SDL) is a leading standardized solution being developed by a consortium headed by Ford and Toyota to manage open source software for smartphone app development and connectivity for vehicles. The SmartDeviceLink Consortium (SDLC) was established in November, 2016 and also includes the Mazda Motor Corporation, PSA Group, Fuji Heavy Industries Ltd. (FHI) and Suzuki Motor Corporation. Luxoft was a technological partner on the project from the very beginning along with suppliers Elektrobit and Xevo. The consortium is focused on significantly increasing the choice for consumers in how they connect and control their smartphone apps on the road.

WHAT EXACTLY IS SDL?

SmartDeviceLink is an open source POSIX-compliant technology platform that allows communication between smartphone applications and in-vehicle software. In-vehicle infotainment systems are connected to third party enabled applications. It can be deployed to Linux, QNX, Windows or other popular embedded OS and can communicate with any mobile device OS such as iOS or Android using Bluetooth, Wi-Fi and USB drives. Ford initially developed SDL and contributed it to the open source community in 2013. Since then Ford has continued to lead the effort for adoption and improvement of the standard. Ford's latest infotainment offering, SYNC 3, comes equipped with AppLink powered by SDL.

WHAT ARE THE ADVANTAGES OF SDL?

The main advantage of SDL is that customers will be able to voice-activate their apps using SDL and stay connected on the move. Pandora, Spotify, Glympse, Audioteka and many other popular apps are already available. Additionally, applications on mobile devices offer almost unlimited possibilities in extending a car's infotainment system. SDL supports various Human Machine Interfaces (HMIs). Traditional car features like radio, climate control and diagnostics are always available in IVI even when a mobile device isn't connected. All additional capabilities become available when a mobile device is connected. Mobile device apps offer almost unlimited possibilities in extending a car's infotainment system. These additional features can be added onto over time. By updating the app on a smartphone the capabilities of the IVI are also updated. SDL provides several other functionalities such as UI Manipulation, Hardware Buttons, Touch Screen, Voice Recognition, Text-To-Speech, Media Data Streaming, Mobile Navigation, Vehicle Diagnostics and Remote Control.

The SDL Ecosystem

The core component of SDL is the software which automotive companies (OEMs) install in their vehicle head units. Integrating this component into their head unit and HMI based on a set of guidelines and templates enables access to various smartphone applications. The iOS and Android libraries are integrated into their applications by app developers to enable command and control via the connected head unit. An optional SDL server can be used by automotive OEMs to update application policies and gather usage information for connected applications.

VIDEO – How Does Agile Messaging Work in a Car?

If you connect your smartphone to your car you can manage messaging easily by voice. You can manage your preferences and set messaging to private mode if you are traveling with friends and guests. See this technology in practice in the video below.

http://bit.ly/2v8j3M8

DIY

AUTOMOTIVE

BOYKO KAZAKOV

One of the major disadvantages of the automotive industry is that every product that is developed for it is proprietary, and the common folk developer cannot actively involve him or herself with the wide range of tools and magnificent software solutions that make modern vehicles tick.

This tendency is slowly changing as most of the bigger automakers like BMW, Ford and others are opening their APIs to outside developers. That way, rapid prototyping is becoming more and more common in our industry. Reducing the time from prototype to actual manufacturing allows for further development and research of modern technologies. It is a fact that the automotive industry is much slower in terms of its evolution compared to the smartphone industry, but all the small steps to get to open source will be justified when more high-level specialists get involved.

Today I will introduce you to an intriguing open source solution that is both free and aimed at giving an equal start to all vendors.

In the following guide you will learn how to build and manage a platform intended for a car infotainment system. IVI, also called In-vehicle infotainment, is a collection of hardware and software in the vehicle that provides audio/video entertainment, including navigation and connectivity. Though this system is not as integral in terms of safety as engine control or the brake system, for example, it still requires significant attention in design and stability. All systems in modern cars are connected via quite a complex network. Usually a system like the IVI can prevent the car from starting if a movie is playing that could be distracting to the driver. In other words, each step of this complex network must be approached with great care.

Before we begin, keep in mind that this is a software solution to demonstrate a principle in the automotive industry. It is not recommended for use in an actual car.

THE HARDWARE

The automotive industry follows a lot of rules. Those rules apply not only to code quality, but also to the embedded solution hardware. Not every processor unit is qualified to be part of a system that human lives depend on. In practice, automotive grade hardware must withstand specific temperatures, physical shock and allow for recovery mechanisms in case of hardware glitches.

Automotive grade vendors provide evaluation boards with specific SoCs (system on chip) but their cost is quite high for the average open source hobbyist.

For the purpose of this exercise, a simple raspberry PI 3 development board will do the trick. The board itself is open hardware and offers a good range of community support, which is perfect for developers who are just starting in the embedded world. For the RPi there are quite a few extensions such as, for example, the CAN hat – this way the device can truly be part of the car area network.

Although Broadcom SoC is not automotive grade, it will be able to run the platform (IVI is usually heavy on graphics and it will strongly rely on the GPU power). Currently there is hardware support for the RPi3/2 in one of the latest "Chinook" branches of the AGL (Automotive Grade Linux), so there is no need to port specific hardware parts in order to run the demo.

Additionally, you will need an HDMI monitor and a mouse. Typically, automotive displays have touch input, but for the sake of the demo the interface can be interacted with via a simple USB mouse. As the tendency for an infotainment display is to use portrait mode, be aware that when the OS boots the screen will rotate 90 degrees and a rotating monitor is a really good option.

LET'S USE YOCTO

The AGL Linux is a Yocto-based operating system. Yocto itself is not an operating system, but a methodology – a set of compilers, tools and source bases that allow you to build your own custom Linux distribution. The process of creation is defined by so-called "bitbake" recipes.

AUTOMOTIVE GRADE LINUX

Like recipes, which require ingredients to create specific dish, the bitbake tool downloads from specified locations, patches, builds and packs a specific part of the operating system.

Those so called recipes are organized in different layers. Each layer has a specific purpose. For example, hardware support for specific platforms is separated into layers. That way, you can have a layer supporting one kind of processor and its kernel, and another that is organized with a completely different kernel architecture. All the recipes specific for the AGL operating system are placed in the meta-agl layer. This layer provides us with build scripts and also predefined configurations that state which parts must be included in the final image.

After invoking them, the build process will be time consuming. As the tooling will need to download almost every source of the system and its application, that will run on top of it later on. Compilation time depends on the performance of the workstation you are actually building. Yocto aims at parallelizing the procedure in different processes but sometimes this can lead to issues. Typically, these are race conditions caused by bad written receipts.

To build an AGL demo platform for Raspberry Pi 3 the build scripts take a couple of arguments.

For the machine argument, we choose raspberrypi3 and append a couple of other features to be set as well **agl-demo, agl-netboot** and ***agl-appfw-smack:***

```
$ source meta-agl/scripts/aglsetup.sh -m raspberrypi3
agl-demo agl-netboot agl-appfw-smack
```

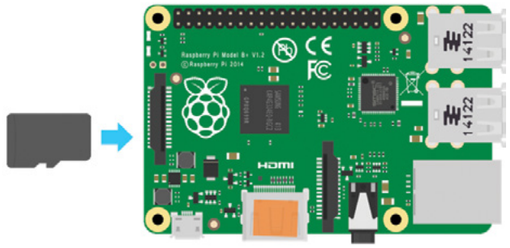
After setting the bitbake environment, the script will show available top-level bitbake variants. The system we need for the demo is baked with **agl-demo-platform:**

```
$ bitbake agl-demo-platform
```

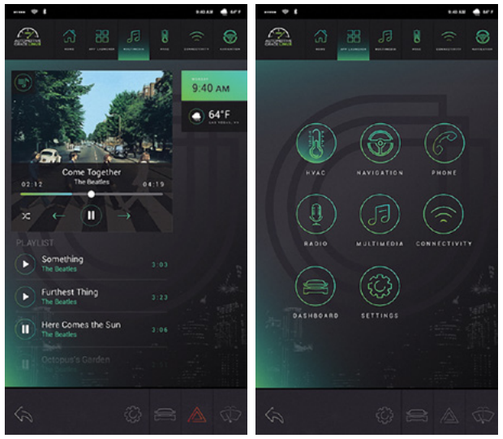
If everything goes okay, the scripts will create an SD-card image that is ready to be burned. There are many ways to flash this kind of image to the SD-card. The best way is to just use ''dd'' for direct copy of the image to the SD card. The output files are usually located in the build directory:

```
$ <AGL_WORKDIR>/build/tmp/deploy/images/
raspberrypi3/agl-demo-platform-raspberrypi3.
rpi-sdimg
```

After inserting the SD-card and powering our raspberry PI, if everything went fine during the build, the Linux kernel will load successfully.



The Wayland-based graphic system load and the Home Screen Application will be displayed. This example, Human Machine Interface (HMI), is close to a real interface used in the industry. The homescreen demo has a radio interface that is already connected to the service for radio. Additionally, with the HVAC Control you can toggle GPIO-s that can control fans or heating elements.



Using the "afm-util list" command will give you a list of demos installed in the current image. Running "afm--util run <demo id>" will also run the demo from the above-mentioned list.

As the homescreen service is only for demonstration purposes, it can easily be disabled. After that, switching to Wayland's desktop shell is quite easy.

To disable the home screen service:

```
$ systemctl disable HomeScreenAppFrameworkBinderAGL.service
$ systemctl disable HomeScreen.service
$ systemctl disable InputEventManager.service
$ systemctl disable WindowManager.service
```

AUTOMOTIVE APPLICATION

AGL offers quite a unique framework for applications so you can start building right away. It is a good practice to continue using the bitbake environment and create recipes for each application so that the build becomes somehow independent from the host machine and its pre-existing environment. The distribution allows the use of HTML-based as well as QT/QML-defined interfaces. That way, developers can really unleash their imaginations and create really cool user experiences.

For the HTML applications there is the meta-browser layer in Yocto, which provides barebone browser support. This is used to display the web-based content via a specific rendering engine. QT application, however, can be used quite extensively. It can render support as well as provide some web-based content. Where QT shines is the fact that a single application can be responsible for displaying interface and backend services at the same time. Imagine an application that serves a simple interface and also provides middle services for hardware communication with the actual device nodes. Additionally, the AGL offers unique wrappers for RPC and other inter-service communications. The so-called AF-BINDER allows the use of a simple D-BUS or WS connection to services and is quite easily accessible with its versatile JSON API.

In terms of security, AGL uses kernel-level smack isolation, so it's quite possible that we will see cars soon enough that are running on AGL-based distribution.

Wrapping up, AGL Linux is a really good starting point for any developer that wants to be part of automotive development. Involving yourself in the open source community can resolve problems that are holding back major manufacturers, thus ensuring the progress of the entire industry.

BOYKO KOZAKOV is a well-known specialist in the embedded Linux and open source communities. He is working as Senior Developer for various automotive projects for Luxoft Bulgaria.

His specialties range from the electronic design of devices, to software architecture of operating systems, including the graphical design of infotainment systems.

ON THE WAY TO AUTONOMOUS DRIVING

ANDREY BOCHKOVSKIY

"First, there were some amoebas. Deviant amoebas adapted better to the environment, thus becoming monkeys... Then somebody fell on a sharp stick and the spear was invented... Suddenly (in evolutionary terms) some deviant went and built a printing press... Two blinks later and we're switching the batteries in our laptop computers while streaking through the sky in shiny metal objects in which soft drinks and peanuts are served". This is very brief theory of evolution, presented by Scott Adams in his truthful book "The Dilbert Principle". Not surprisingly, a similar evolutionary process is happening on the way from the invention of the wheel, across invention of parking sensor around 1970 and advanced driver assistance systems (ADAS), to the autonomous driving vehicles in the mass market.

Going a little bit into details, namely ADAS can be considered as an essential bridge from conventional to the autonomous driving vehicles. Realities of the year 2017 show many forms of ADAS available covering wide range of functionality from sending beeps or warning messages to the driver to taking partial or full control of the vehicle for the sake of comfort and safety of both vehicle occupants and other traffic participants including pedestrians and occupants of other vehicles. Safety requirements imposed on ADAS shall be considered the key that opens the way for the autonomous vehicles to end users. People, which are going to be users of automated vehicles, must be able to trust that the vehicle will behave adequately in all situations that occur on the roads, including intentional cyber-attacks [1].

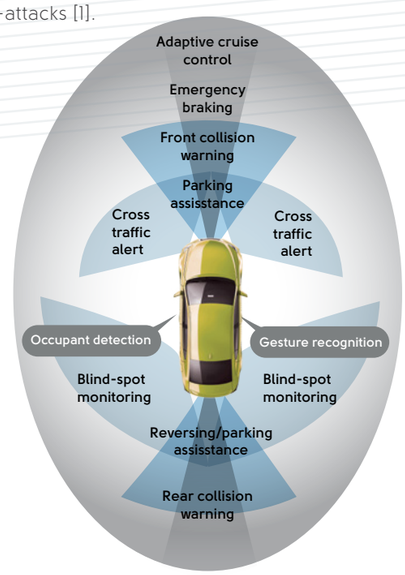


Figure **1**. Automotive sensors

Some ADAS features are built into vehicles by OEM or are available as aftermarket solutions. ADAS is constructed on the basis of inputs from multiple data sources that the modern vehicle may include, such as imaging devices (cameras), light identification detection and ranging (LiDAR), radar, and ultra-sonic sensors, schematically shown in Figure 1 (Image courtesy of Leddar) [2].

Additional inputs are possible from other sources separate from the primary vehicle platform, such as other vehicles, referred to as vehicle-to-vehicle, or vehicle-to-infrastructure systems, such as mobile telephony or WiFi data network.

In fact, these days ADAS is one of the fastest-growing segments in automotive electronics, with steadily increasing rates of adoption of industry-wide quality standards, as well as vehicular safety systems ISO 26262, developing technology-specific standards, such as IEEE 2020 for image sensor quality, and communication protocols such as the vehicle information API.

Yet another component necessary for making the automation of driving possible is terrestrial navigation capability. This segment emerged only after the year 2000, due to the fact that it was then that accurate satellite-based positioning became open to the public. It took another half-decade until Garmin, Mio, Navigon, Magellan, TomTom, and others filled the market with devices of various level of functionality and price.

Luxoft pays significant attention to the automotive line of business and specifically participates in the development and maintenance of the ADAS and navigation systems already in production or preparing to enter production in the near future. Automotive competences of the company are geographically distributed as shown in Figure 1, in order to form a powerful constellation of automotive practices which engage best engineering resources from around the world.

Figure **2**. Luxoft geography

NAVIGATION AND ADAS

These days, drivers increasingly come to rely on technology to keep them on track with the right route and help make their driving more informed and efficient. While manufacturers of vehicles are keen to respond by giving drivers the best possible solutions, Luxoft strives to cover important areas of navigation and ADAS development.

The company places a strong emphasis on interactivity in its car navigation solutions, incorporating 3D modeling and support for Navigation Data Standard (NDS), and offering worldwide usability of related products in a compact and efficient format that are scalable and resistant to unauthorized copying. As a result, Luxoft's partners are afforded innovations that make their systems more convenient, user-friendly and competitive.

Being an official subcontractor of the NDS Association, Luxoft helps to improve automotive grade map technology by defining, developing and promoting the NDS standard. The company also partners with some of the leading infotainment and navigation solution providers such as Harman, Elektrobit and TomTom, all of which use NDS in their navigation systems. NDS has been also adopted by global automakers including BMW, Daimler, Volkswagen, Hyundai and others.

Luxoft develops high-definition map displays based on existing high-end NDS features and its rich experience in visual effects development and hardware-specific (CPU/GPU) optimization. This is augmented by the creation of low-poly 3D models and the remodeling existing models for enhanced looks and performance.

In particular, the company supports a direct route to map visualization that encompasses the data access layer, data management, effective algorithms, texture and model creation, style configuration, rendering and shaders.

By now, several navigation systems have been released with the direct involvement of Luxoft such as BMW NBT, Daimler NTG5, BMW NBT EVO, and BMW EntryNav. As for the latter, it contains MapViewer, which was written from the ground up. MapViewer is a front-end part of the Navigation System for visualization of digital map data that is based on Open Graphics Library (OpenGL) as well as a strong algorithmic and mathematical base.

The Dynamic Points Of Interest (DPOI) system developed by Luxoft is an interactive navigation software platform that combines social network-like online technologies with location-based services. It puts the full scope of local points of interest at drivers' fingertips, alongside the unprecedented interactivity of exchanging location-based data online between cars. DPOI supports clients worldwide from a single server, meaning drivers can add new points or modify existing ones from any internet-connected device.

Moreover, Luxoft has great experience in porting and integrating customer software (SW) to new or alternative hardware (HW) systems, as well as in system analysis, the creation of adapters and scaling of system performance. DPOI is just one example of this, through its ability to be used by any car navigation system.

Engineers of the company do implement navigation controllers that are fully GENIVI compliant [4] and are also able to offer the integration of controllers into other proprietary navigation component interfaces. Compliance to GENIVI is illustrated in figure **3**. This includes the complete development of GENIVI adapter components, which translate GENIVI interface calls to the proprietary interfaces of navigation components, and vice versa.

Over the years, the company has enjoyed countless success stories with integration of third-party libraries, tools and solutions across multiple customer projects and systems. Notable examples include the integration and configuration of the engine from Nuance Communications, Inc. to an NDS database compiler, the successful integration of a variety of tools, and the integration of TomTom navigation engine into a complete system.

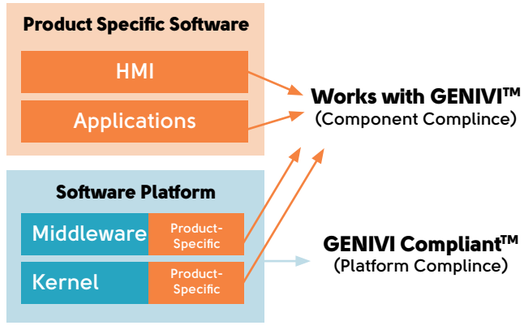


Figure **3**. GENIVI compliance

COMPUTER VISION AND ADAS

Computer Vision and Augmented Reality (CVNAR) is Luxoft's software solution that uses advanced algorithms to create a mixed reality for drivers, combining data from vehicle sensors, map data, telematics and navigation to help drivers feel safe and better informed on the road.

A wide range of CVNAR applications spans almost every facet of modern driving, from driver aids and vehicle-related information to safety innovations.

In particular, CVNAR detects lane markings, road boundaries, other vehicles, pedestrians, road signs, buildings and hazards.

CVNAR technology incorporates artificial intelligence, pattern recognition, image processing, real-time signal processing and more [5]. CVNAR provides augmented navigation, visualization, and additional information in case of advanced parking assistance and adaptive cruise control. It supports low visibility mode and autonomous driving. The approach supplements drivers' visual reality with graphic objects in real time (Błąd: Nie znaleziono źródła odwołania), and works with various output devices such as head unit displays, digital clusters, and head-up displays. CVNAR computer vision and augmented reality software framework was sucessfully used in a demonstration of ADAS and automated driving capabilities presented by QNX Software Systems, a BlackBerry subsidiary, at the International Consumer Electronic Show (CES) 2016. Installed in the QNX reference vehicle, a modified Jeep Wrangler, the CVNAR framework ran algorithms to highlight turn-by-turn navigation arrows, street names, pedestrian crossings, destinations, points-of-interest and forward collision alerts. It is important to mention that the road scene reconstruction engine of CVNAR utilizes the company's proprietary solutions for augmented reality.

YET A LOT REMAINS TO DO

Regarding autonomous driving capability, it needs to be mentioned that with emergence of autonomous cars, various ethical issues arise. The introduction of autonomous vehicles to the mass market seems inevitable due to a potential reduction of crashes and their accessibility to disabled, elderly, and young passengers, yet there still remain ethical issues to be resolved. Those include but are not limited to the moral, financial, and criminal responsibility for crashes, the decisions a car makes right before a (fatal) crash, privacy issues, and potential job losses.

In spite of that, more and more success stories about autonomous vehicles do appear. More vehicles are getting autonomous modes and the mileage accrued using autonomous mode steadily increases. The record of autonomous driving without an accident already counts millions of miles.

Reflecting the step-by-step approach to this global task, the Society of Automotive Engineers (SAE International) defines the corresponding levels of autonomous driving capability from no automation (conventional vehicle), to full automation [6],

where for level 3 and above, the automated driving system is able to monitor the driving environment, replacing the human driver. Considering the nature of the task, two blinks in terms of autonomous driving evolution may constitute decades in real time of automotive practice, as illustrated in figure **4**.

This is why, on the way to autonomous driving, Luxoft is keen to offer both valuable partnership in almost every area of the automotive business that SW is quickly penetrating and also establishing engineering practices where interested engineers can join existing teams and contribute their ideas and efforts to solve one of the most ambitious contemporary automotive challenges. Due to the diversity of technical subtasks, experienced engineers are truly welcome regardless the areas where they have gained their experience, such as embedded systems, telecommunication, digital signal or image processing, computer vision, etc.

Since the year 2000, when Luxoft was incorporated as a development center, we have succeeded due to an unprecedented commitment to customer needs in meeting the demanding requirements of the automotive industry. Let's move into the era of autonomous driving together!

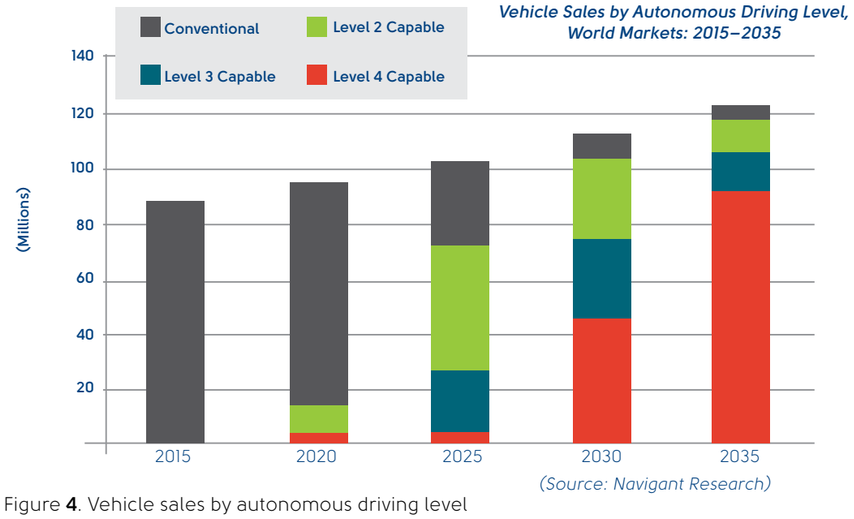


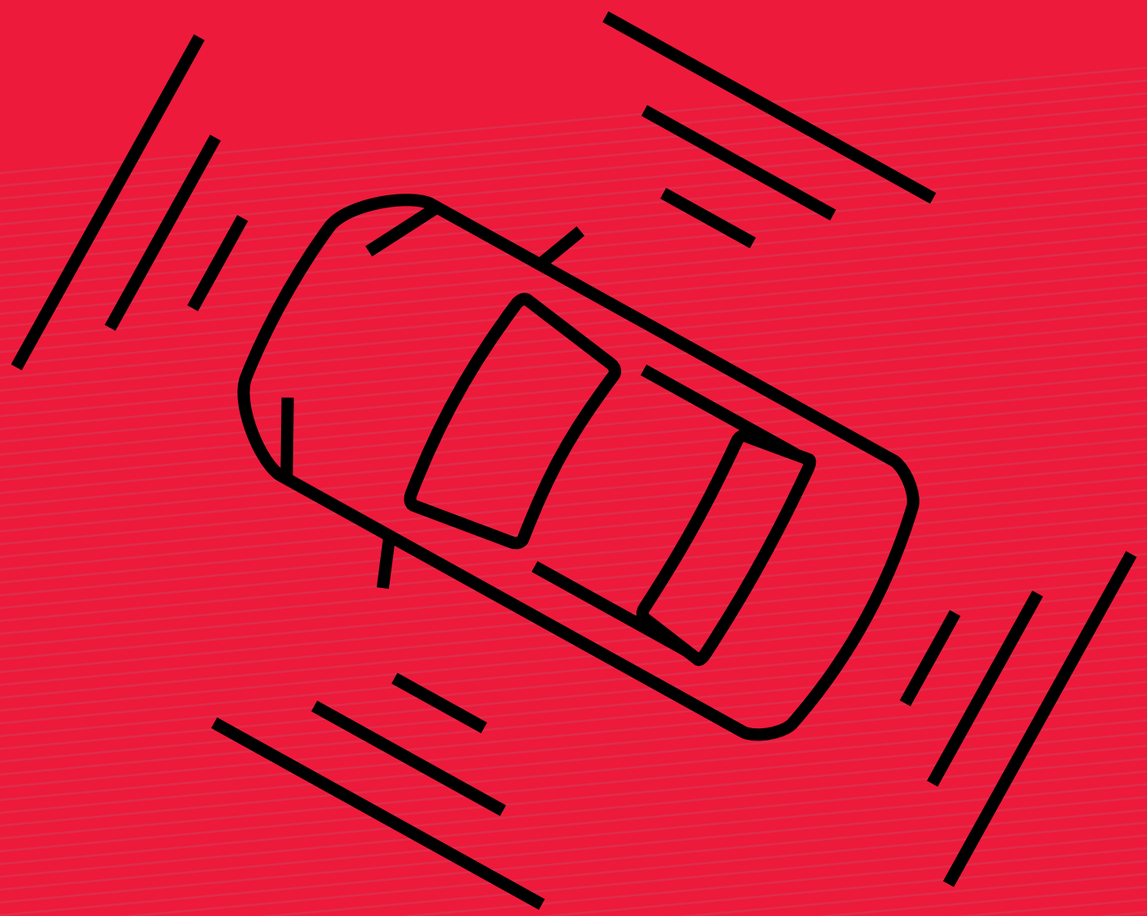
Figure **4**. Vehicle sales by autonomous driving level

REFERENCES

1. Research Report on Autonomous Automotive Cybersecurity, 3Q 2016, https://www.karambasecurity.com/static/pdf/Autonomous-Automotive-Cybersecurity-Report.pdf
2. Steve Taranovich, Autonomous automotive sensors: How processor algorithms get their inputs, July 05, 2016, http://www.edn.com/design/analog/4442319/1/Autonomous-automotive-sensors--How-processor-algorithms--get-their-inputs
3. The History of Car GPS Navigation, Jamie Lendino, April 16, 2012, http://www.pcmag.com/article2/0,2817,2402755,00.asp
4. GENIVI Compliance Programs, https://www.genivi.org/genivi-compliance-program
5. Luxoft Computer Vision and Augmented Reality Software Supports Autonomous Driving and Other ADAS Features, March 21, 2015, John Day, http://johndayautomotivelectronics.com/luxoft-computer-vision-and-augmented-reality-software-supports--advanced-navigation/
6. SAE J3016 Standard Core Reference for Automated Vehicles, September 28, 2016, https://www.sae.org/misc/pdfs/automated_driving.pdf
7. Luxoft Automotive Driving Technology, https://auto.luxoft.com/

ANDREY BOCHKOVSKIY

Andrey has 20+ years of experience developing and leading the development of HW and SW for embedded systems such as satellite navigation receivers of various grades. Among other achievements worthy of mention, this included developing SW in accordance with guidelines for safety-critical software for an airborne navigation receiver. For most of his career, Andreyhas been a technical leader focused on the structured analysis of system requirements and structured HW/SW design.



(ADAS)

KAI RICHTER

Effective timing design, analysis and verification essential to the development of Advanced Driver Assistance Systems (ADAS)

With discussion around the subject of automated driving gathering pace, the development of Advanced Driver Assistance Systems (ADAS) looks set to become the next big thing in automotive electronics. The development of ADAS introduces new challenges and significantly higher levels of design complexity, making the adoption of effective timing design, analysis and verification tools essential in their design.

"Even in terms of where automotive electronics is today, ADAS is a disruptive technology that will fundamentally challenge the way automotive E/E and software systems have been built and integrated in the past," said Dr. Kai Richter, Engineering Director at Luxoft. "ADAS adds enormously to the software cross--dependencies in vehicles, which, in order to execute all the required new functions, will require additional levels of computing performance and network communications far exceeding that of established vehicle E/E systems. To make the step-change needed to turn concepts into reality, the integration of effective timing design, analysis and verification tools into the overall design process will be crucial."

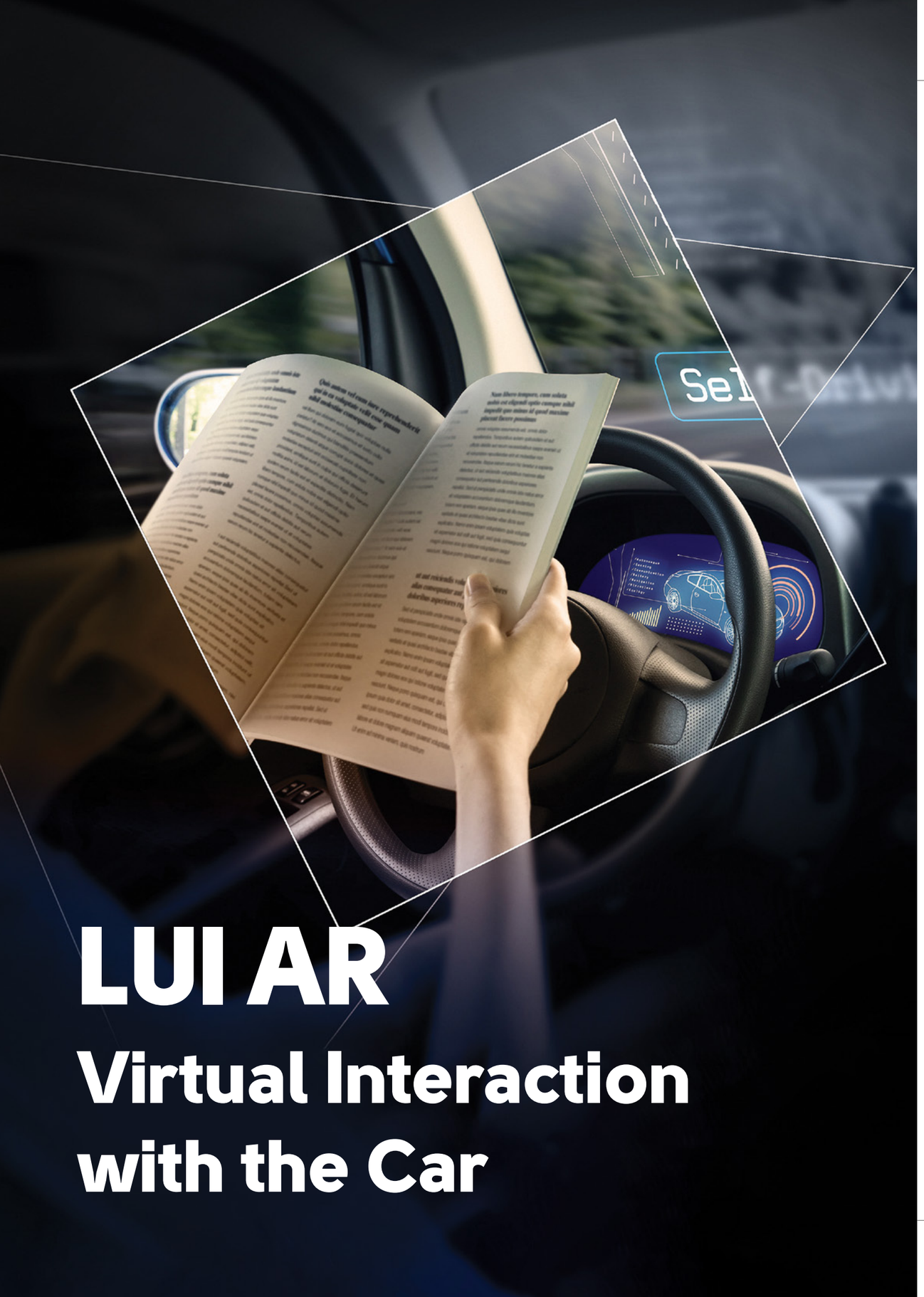
With the development of ADAS, comes the introduction of new types of sensors, new object and trajectory recognition functions and new virtual reality video projection technology, among others. These will also generate data for the automated driving functions to predict and determine how the vehicle should act in terms of the primary control functions, such as accelerating, steering, braking, etc. All these control functions must also cope with new safety and reliability requirements. To facilitate this, in-vehicle computing power will need to dramatically increase, with topologies changing from heavily distributed controllers to fewer, higher--performance computing centers that are connected directly to dozens of sensors and actuators spread over the vehicle.

Architecturally, Ethernet will come to dominate on--board and in-vehicle communications, providing the conduit for new functionality as well as existing control signals that currently use CAN and FlexRay.

In fact, the newly developed generation of ADAS hardware platforms look like large E/E systems already. Several automotive and general purpose cores, graphics and other special co-processors run parallel software that communicates through Ethernet within a single box. The current reference implementations include a mix of time-triggered and event-triggered components and a host of possible safety hazards. To simplify portability, provide interoperability between vehicle domains and application protection, new concepts in software integration, such as virtualization and partitioned scheduling, will be required.

"It is no simple task to efficiently utilize these platforms," continued Dr. Kai Richter. "Also, they have to be utilized optimally as the software usually grows faster than the platform developers can imagine. Timing analysis is the key enabler for this optimization and, to assist in the process, we provide customers with consultancy services for requirements collection and the development of optimized system architectures for the highly integrated systems found in the ADAS domain today. They also value our ongoing commitment to ensuring that our tools are kept fully up-to-date for the requirements and new technologies being introduced."

Symtavision Tools add worst-case timing analysis for Ethernet communication in several flavors including AVB and time-triggered Ethernet, and also cover the ADAS software technology for both AUTOSAR and non-AUTOSAR operating systems, with timing analysis for partitioned and hierarchical schedulers (with or without hypervisor). In addition, the end-to-end timing analysis supports heterogeneous mixes of both event and time-triggered applications and architectures, as well as statistical simulation and worst-case analysis, and compatibility with the existing OSEK, CAN, LIN, and FlexRay modules.



LUI AR
Virtual Interaction with the Car

LUI AR – A NECESSARY HUMANLIKE DRIVING COMPANION THAT'S RELIABLE, HONEST AND TRANSPARENT

The autonomous vision is more prevalent in the automotive industry now more than ever, and is continuing to move like a wildfire. Many automotive companies are focusing on the technical implementation of autonomous driving to make it a reality. This is a huge calculation process that makes sure both the steering wheel angle and acceleration values are precise. With these accurate measurements, it's feasible to say we'll be seeing more self-driving cars on the road in the near future – but this new reality doesn't come without its own challenges.

HANDING OVER YOUR RESPONSIBILITY TO A VEHICLE

It's important to note what not many consider when they jump into the driver's seat, but may often be considered subconsciously. In a self-driving car, *people give the responsibility of their lives to the vehicle*. Allowing this release of control requires a much higher level of trust in the autonomous car than in a human-driven one. Convincing human beings to build trust in their car (especially if they are using an unfamiliar car from a shared car provider) can be difficult.

But let's take a step back. What if we thought of the car as a humanlike companion rather than a cold-blooded machine? Building trust in a human-human relationship can be complex and sometimes requires a lot of time, but there are key humanlike qualities that can be attributed to the human-machine relationship, such as:

- Being reliable
- Being honest
- Being transparent [BS2]

If a machine adheres to these rules, *it's easier to hand over the responsibility*.

In addition, the machine also has to communicate in a way that is natural and easy to understand. For a passenger in a self-driving car, it should not be any different than talking to another living, breathing human being. So as technology marches forward, new communication opportunities arise that are appealing to users.

HOW WILL AR/VR COMMUNICATE WITH THE FUTURE?

The truth is, augmented reality (AR), virtual reality (VR), hand tracking and holograms are shaping the future of human-machine interactions. By allowing for a more enhanced user experience, the human-machine interface (HMI) becomes more user friendly, natural and intuitive for the user.

To adapt to the change of the future, Luxoft created an AR/VR solution that introduces a humanlike machine companion to the passenger. This system, the Luxoft User Interface AR (LUI AR), will pave the way for autonomous vehicles to get you from point A to point B.

THE LUI AR IS AN AUTONOMOUS SYSTEM YOU CAN COUNT ON

Using LUI AR means letting the car take the wheel without any need for concern. Through high-tech sensors, the car can alert the passenger of obstacles or changes before they happen. For instance, it follows other drivers' maneuvers, such as lane changes or identify sharp bends in the road ahead.

The human-machine interface (HMI) also detects the weather. The car alerts passengers when bad weather could be an issue, offering alternative routes and plans. For example, the car could offer an evasive route, or something reasonable to pass the time while the storm blows over, like shopping or dining options. Of course, the car also informs the passenger about nearby places of interest without bad weather issues. But unlike a typical GPS, the user can interact with a virtual model of the destination, such as a castle or a museum.

In addition, this solution allows the car to keep you informed of other things, such as when it has finished charging its battery, found an available parking spot, alerted the user for maintenance and so on. It also asks the user questions, such as if and when it should pick you up, or even if it should drive through an automated car wash beforehand.

SIRI FOR YOUR CAR

Have you ever wondered what it would be like to talk to your car? LUI AR allows you to add your car to the contact list of your favorite messaging app. And when you send messages, the car replies in a humanlike nature, feeling extremely natural to communicate with.

CONCLUSION

By providing a new way of the future, LUI AR improves the customer journey every time the user steps into the car. This solution provides a stunning and engaging user experience to get you from point A to point B and anywhere in-between.

The future is coming, so let's get prepared.
Contact us to find out more!

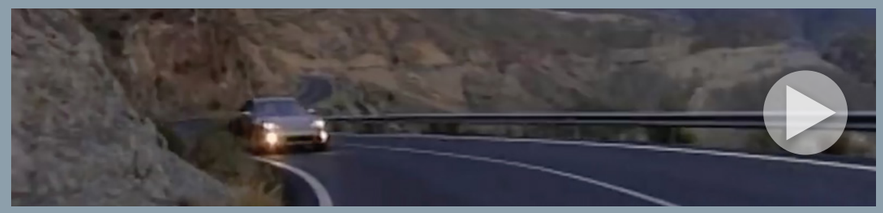
CAR HMi AWARD
UX REDEFINED 2017

FINALIST
Most Innovative
HMi feature

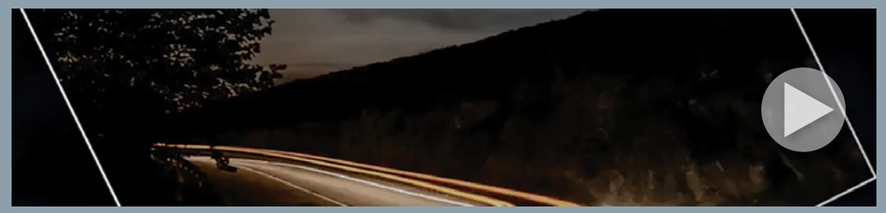
LUXOFT

MOVIES

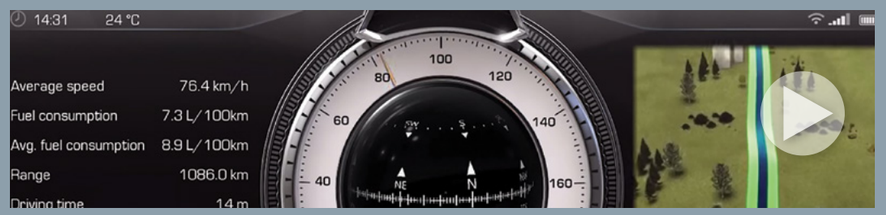
Watch all video related to our automotive Expertise. http://bit.ly/2wSIUe8



Rinspeed. http://bit.ly/2vL2kk8



Luxoft Perspectives present an exclusive interview with Michael Bykov
(Managing Director, Luxoft Automotive Solutions) and Prashant Kelker (Digital Advisor, ISG). http://bit.ly/2uhHR51



Populus. http://bit.ly/2vL5XGD

Messaging co-pilot. http://bit.ly/2uSwe4v

Future Vehicle Concepts – LUI AR Luxoft User Interface Augmented Reality. http://bit.ly/2v4EMED

Check out our YouTube channel at www.luxoft.com/youtube

LUXOFT

Photo credits:shuttertock.com

LOGEEK
MAGAZINE

Editor: Dilya Salakhetdinova

Comments, suggestions, or want to be a part of the next issue?
Write to LGmagazine@luxoft.com

career.luxoft.com