Developing software for Autonomous Vehicle Applications; a Look Into the Software Development Process

By Andreas Lindenthal and Franz Walkembach, Wind River

The concept of autonomous vehicles or unmanned drones has generated considerable public interest in recent times. While the idea appears technically plausible, in order to make this a reality, development teams face a tough task. This article will discuss the standards developers need to be aware of and the steps they need to take to ensure safety of autonomous and other automotive applications.

Applicable Standards

Thankfully, much of the automotive vehicle safety development methodology has already been formalised within the ISO26262 standard. Titled “Road vehicles – Functional Safety”, the standard is part of the broader ISO61508 Functional Safety Standard for Automotive Electrical and Electronic Systems. First published in 2011, ISO26262 aims to address the potential hazards relating to malfunctions of vehicular electronic and electrical systems. This standard already applies to safety-critical features of all current types of automobiles, not only autonomous vehicles, since even human-driven cars are leveraging advanced driver assist systems (ADAS – Figure 1) such as lane change assist / departure warning systems that can take control of key driver functions such as steering and braking.

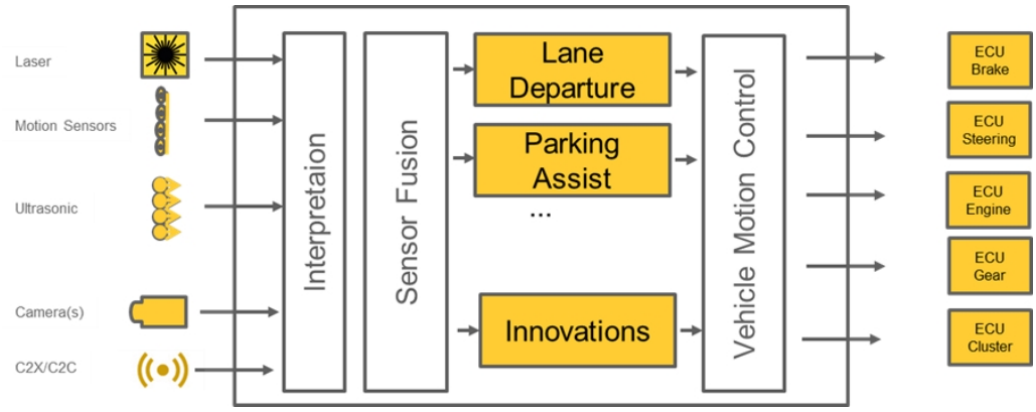

Figure 1: Conceptual design of ADAS systems

Development Process

So, what should a developer do when faced with a new autonomous vehicle software design project? In general, development would adhere to the process depicted in Figure 2 below.

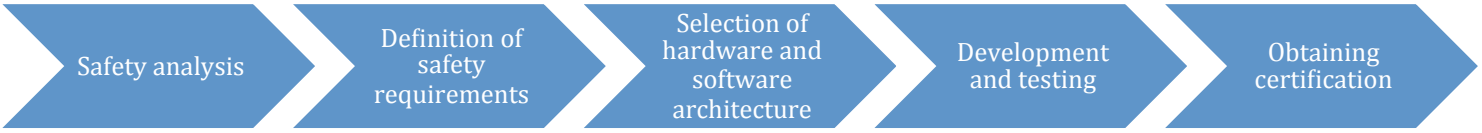

Figure 2: Key steps of the development process

Safety Analysis and Risk Classification

The most important aspect of the development process is to identify all the project requirements and highlight those that have the potential to impact safety. Based on such safety analysis, a mapping exercise is performed that looks at both the software and hardware platform, and assigns a safety risk classification according to the automotive safety integrity levels (ASIL) A, B, C, or D. ASIL D denotes that in the event of a malfunction, the potential for a severe life-threatening or fatal injury requires the highest level of safety assurance. It is highly advisable for developers who have not been directly

involved with functional safety to get specific training in order to become fully aware of how to assess all aspects of software safety.

Software Safety Plan

The formal output document that aggregates all of this information is called the Software Safety Plan. Typically this document is reviewed during a proof-of-concept meeting. System architecture, modules, safety requirements and functions, critical paths and diagnostics are outlined in this document. Diagnostics are a fundamental part of any ISO26262 certification and are used to reduce failure rate. The Software Safety Plan does not go into detail on how the software operates. The Software Requirement Document is the one that identifies the functional software units developers will need to create.

The Software Safety Plan outlines the need for various software components. In most cases, a real-time operating system (RTOS) will be specified as a fundamental software component. A choice of an RTOS requires careful consideration. For example, the developer needs to ensure that the RTOS is certified to IEC61508. By using software that has been certified accordingly allows the developer to leverage certification provided by the RTOS vendor.

The next task is to map the software safety requirements. As discussed earlier, IEC61508 is a basic standard and ISO26262 is an application standard. At the same time, ISO26262 does not automatically inherit IEC61508 certification. A minimum requirement is to provide a compliance matrix from IEC61508 to ISO26262, but this is not ideal, and the goal should be to formally achieve ISO26262 compliance. This involves looking at the whole system, not just the software, and identifying all the safety and non-safety-related functions involved.

Partitioning and Virtualization Help Ensure Safety, Security, and Reduce System Cost and Footprint

A key way to achieve a required level of safety is to establish time- and space-based separation of functions to protect safety-critical applications from being negatively affected by non-safety critical ones. Space, or spatial, partitioning prevents data in one partition from altering data or program code in another partition. It protects code running, for example, on partition 1 from being able to access output devices that are being used by a higher-criticality application running on partition 2. Current 32- and 64-bit CPUs provide a memory management unit (MMU) that can support this feature in software. Time, or temporal, partitioning ensures that a safety-critical application has a guaranteed time frame to execute, e.g., access a processor, shared resource, or physical device, with lower-criticality applications being unable to gain access to it at the same time.

One way to implement time- and space partitioning for safety-critical designs is to use virtualization technology. Virtualization, implemented either on a single-core or a multi-core chip, allows different applications to run in safe and secure partitions, separated from each other and controlled by a hypervisor. Virtualization helps reduce the time, complexity, and costs of system development while accelerating the testing and certification process. Embedded virtualization also allows developers to implement electronic control unit (ECU) consolidation hence reduce the cost and footprint of the system. Commodity embedded processors make it easier for automotive software designers to combine multiple applications on a single processor instead of using several distinct circuits. Eliminating multiple custom boards also greatly simplifies maintenance.

Partitioning helps improve system safety and also aids in boosting software security against external threats, whether from an ill-timed software update or a purposefully orchestrated malicious attack. Each partition can run its own small firewall rather than relying on one main firewall, in which a breach could give an intruder access to the whole system. If one application is compromised, the intrusion is limited to one partition where it can easily be detected and disinfected, saving considerable time and money and reducing safety risks. This also stops intrusions from spreading across system components, particularly from malware. Importantly, it prevents hackers from

accessing the network stack to launch other attacks or take remote control of the vehicle. Selecting a real-time operating system (RTOS) that can support partitioning and virtualization out of the box greatly aids in the certification process. For example, the VxWorks RTOS from Wind River supports virtualization, space, time, and resource partitioning and is certified to IEC61508.

Figure 3: Steps to enhance automotive embedded security

Obtaining Certification

As the project progresses, and like in any other software project, the whole application will be subject to development reviews, testing, verification and validation. Once everything has been internally tested, an independent third-party will need to be employed in order to help obtain ISO26262 and IEC61508 certification. Certification agencies such as TÜV-Nord, Bureau Veritas and Lloyds are examples of such third parties. They will take all the test results, reports and other supporting documentation for ISO26262 and conduct their own analysis of the application to decide whether the software and its development process has followed due process and can be awarded certification. The whole certification process may take several weeks and would involve running many reviews on the autonomous vehicle platform under test.

The arrival of the autonomous vehicle still has a long way to go. However, development teams can begin to carve out their strategies and start to investigate key design considerations and the many certification needs that lay ahead of them. Given the complexities of the autonomous car and critical role of software, it will be increasingly essential for automotive companies to find the right partners with the appropriate expertise who can help them stay on the right track.

+++Ends