Luxoft
A DXC Technology Company

Always render, never surrender

Why safety-critical renderers are an indispensable part of instrument clusters

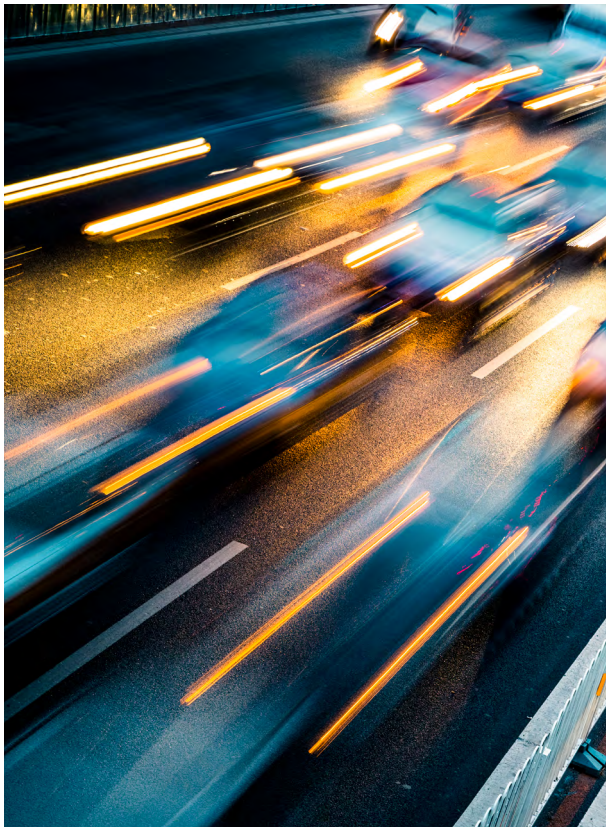
by **Arwed Richert,** Program Manager

The safety of drivers and passengers is the number one priority for automakers. As the amount of software in cars increases, the safety aspect of software in cars soars.

For instrument clusters, displaying telltales is a safety feature that has to work reliably, no matter what: This is the field of safety-critical renderers, or **safe renderers** for short.

But what exactly are safe renderers? Why do we need them and how do they work?

In this white paper, we'll examine the impact of this special kind of software on business and development, then look at its use and technical specialities. We'll also show you a way to integrate safety rendering into your projects without exceeding your development budget.



Safe renderer 101: What and why

Safe renderers became a necessity when functional safety started to be a matter of public interest.

ISO 26262 and its influence

The International Organization for Standardization (ISO) published ISO 26262 "Road vehicles — Functional safety" in 2011. This risk-based safety standard was quickly adopted by carmakers and suppliers — they used it as a guideline for developing products with inherent functional safety.

Since then, ISO 26262 — with an update in 2018, called the second edition — has been the standard in the automotive industry.

Nowadays, manufacturers of hardware and software for road vehicles voluntarily prove that their products fulfill the standard and therefore satisfy current safety regulations. Following the standard shows your company is compliant, and also reduces the risk of potential judicial disputes (because it shows that the developed product complies with state-of-the-art development). The press and general public graciously acknowledge these statements.

Telltales — safety-relevant elements

When it comes to developing software for instrument clusters, one particular type of element gets the focus: **Telltales**.

Telltales — those small icons and indications that tell you if something is wrong with your vehicle — are an integral part of every instrument cluster. Because they are indicators of malfunctions, they are a safety feature and consequently fall within the scope of ISO 26262.

Safety and instrument clusters

Modern instrument clusters are displays behind the steering wheel. As such, all elements of the cluster are rendered, even the telltales.

This is where it gets tricky: Rendering is done by software and software can fail. But, telltales must work, even if software fails.

A way to get around this dilemma is to render telltales in a separate domain of the software that adheres to ISO 26262: A safe renderer.

Safe renderers on the market

Several safe rendering solutions are available on the market. Depending on which solution you choose, you may have limitations concerning the supported operating system and hardware platform. Some renderers offer only limited functionality — e.g., only standard use cases — or create additional costs in the form of royalties.

A free open-source solution that avoids such limitations is the Luxoft Safe Renderer.

In the area of software-defined vehicles with the accompanying digital instrument clusters, safe rendering is a must: Whether you build your own expertise — with lots of effort and time and the arduous search for experts on the market — or bring in a software supplier, you cannot avoid using a safe renderer.

But before we get into technical details, let's see which business aspects are influenced by the need for using safe renderers.

The Luxoft Safe Renderer — an open-source alternative

Features:

- Open-source and royalty-free
  - Available for free at **GitHub**
- Extensive safety rendering capabilities
  - Standard elements: Telltales, speed, gear
  - Additional customizable elements
- Platform- and HMI-framework agnostic
- Safety Island support

Luxoft's accompanying offerings:

- Engineering services
  - Platform porting
  - Customer requirements adaptation
- System architecture consulting
  - Partitioning
  - Safe communication path
  - Client-specific engineering and system architecture design

Money talks: Business aspects of safe rendering

Automakers around the world try to become more hardware-independent for all the components they acquire from suppliers. This way, a seamless change from one hardware supplier to another is at least less complicated.

At the same time, original equipment manufacturers (OEMs) ramp up in-house expertise to become even more independent. Instead of relying on suppliers, they try to establish strategic partnerships to develop software based on a common platform. All these strategies are at least partly driven by the urge to reduce costs, for example by switching to a Safety Island approach, which we'll examine in detail in the next chapter.

However, at the beginning of this new orientation, not enough in-house know-how is available. Dependencies to proprietary solutions of suppliers still exist. Moreover, gathering platform- and framework-agnostic expertise within the OEM's context takes time and effort.

As long as OEMs are in the middle of their re-orientation, the dependency risks between the OEM and the hardware suppliers still exist: Putting multiple solutions to work is expensive, as all of them have to be maintained and paid for (hardware, platform, etc.); and no common solution is in sight to unify things (like over-the-air updates, remote diagnostics and other software-related functions).

Ways to step forward

To overcome these obstacles, OEMs have two options: Use technology that is already available (e.g., integrate the Luxoft Safe Renderer into their project), or bring in a software supplier (e.g., work together with Luxoft). Either way, you should consider the following points first:

- Do you need in-house expertise?
  - This would mean ramping up your teams by onboarding new people
- How much and which parts of expertise can be outsourced? How long will the outsourcing take?
  - Contracting a supplier who has skilled experts in place enables a fast project launch and a short transition phase
- How to distribute your development efforts?
  - Check if hardware development can be minimized by focusing on faster software development. For example, developing and using one software platform that supports multiple hardware platforms can save costs

With these business-related decisions made, you can now focus on the definition of the technical aspects of your safe renderer project.

Making the correct choices reduces costs

Why is selecting elements for safe rendering needed? Why not render the whole IC in a safe way?

The answer is simple: Safety is expensive. Developing software that fulfils safety regulations like ISO 26262 is more complicated and costly due to the safety overhead. Certification fees for mandatory external audits also add to the costs.

A simple telltale is a small element, usually just an icon with only a limited number of different states.

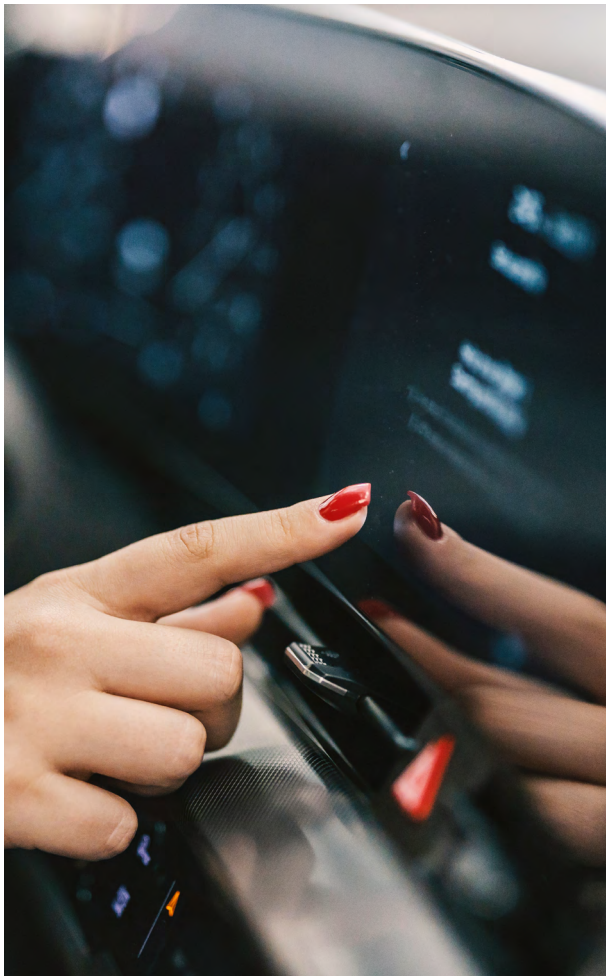
Graphical elements like this can be handled easily. However, implementing bigger elements like gauges with more functionality is difficult enough even without the safety constrictions. Think, for example, about the speedometer: A modern gauge not only needs a smooth needle animation, but also dynamically displayed extra information like the current speed limit, an active speed limiter or cruise control — all on the same scale of the gauge.

So, it is advisable to render only elements according to ISO 26262 that really need to be rendered safely.

A look under the hood: Technical aspects of safe rendering

Starting at the ground line

We can find safety measures for telltales for quite some time: A simple solution from the past is to monitor the current that flows through dedicated LEDs and detect a failure as a missing current.



Taking the display hurdle

With the use of displays (usually LCD-based screens) for fully digital instrument clusters that should also show telltales, new failure detection methods became necessary: A software component that is compliant with Automotive Safety Integrity Level B (ASIL B) has to take over the monitoring and failure detection. We call this component ‘safety-rendering software’.

The main job of safety-rendering software components is:

- Draw safety-related content — like telltales — in a safety-compliant way, for example on the foreground layer of the instrument cluster

This job requires additional tasks:

- The software has to check the output for consistency before actually transmitting it to the display. This can be done using CRC-methods

- Other circumstances have to be validated, like the validity of transmission timings, the display transmission channel, sufficient color contrasts or the availability of display illumination

But the software is not the only part of the system that has to ensure compliancy to safety: The architecture of the whole system has to be considered.

System architecture role

The architectural design has to make sure that a safe environment is available throughout the system. Both software and hardware have to fulfil safety requirements. This includes the complete **software stack** — beginning with the operating system and drivers, the hypervisor (a piece of software that allows the running of multiple virtual machines on a single piece of hardware), the start-up boot chain and so on — as well as the relevant **hardware components** used in the system. Additionally, a **safe communication path** must exist, meaning that safety-related messages have to find their way through the system without being altered or getting lost.

Safe communication paths

Safety-related messages have to travel on safe communication paths. Such a path must consist of safety-compliant hardware and software components only. In addition, the messages have to be protected against potential software and hardware failures. Cyclic-redundancy-checks (CRCs) and sequence counts are some options of choice.

In modern systems, one of the following three approaches is used: A (traditional) Safety Partition approach, a Safety Island approach or a safe display approach.

Safety Partition approach
This approach uses a safety partition inside a safe operating system (OS) environment, like QNX or Integrity. However, as an OS like Linux or Android cannot be seen as safety-compliant, the safety partition approach cannot be used in systems based on these operating systems.

Whereas hardware LEDs might still be used for low-end systems, the challenge is that the hypervisor also needs to be safety compliant. A combination that has been proven to work is a QNX-based hypervisor and OS for the instrument cluster and the safety part, combined with Linux or Android for the in-vehicle infotainment (IVI) part.

Safety Island approach
Modern microcontroller units (MCUs) typically include powerful real-time cores (e.g., CR7, CR52). These real-time cores have their own protection mechanisms and exist outside the hypervised high-end cores, so that they can be used as Safety Islands.

Therefore, the high-end cores of these MCUs can be kept free of safety requirements. As a result, the whole IC and IVI software stack can run inside an open-source environment like Linux or Android. The safety rendering itself runs on the real-time core, which is then called a "Safety Island".

To fulfill safety needs, the Safety Island has to run a safety-compliant OS (like AUTOSAR or Safety RTOS), which has only a small footprint. This solution is more cost-efficient than the traditional approach, but it may add another dependency to the hardware supplier: The Safety Island functionality is typically delivered by the hardware supplier and therefore can't be reused.

This dependency can be overcome with open-source products, like the Luxoft Safe Renderer. These products can be reused on different platforms and projects. Software suppliers like Luxoft can support with porting and implementing.

Safe display approach
If safety rendering on the main MCU isn't possible — e.g., inside a specific platform or project — the display side can handle the telltale rendering. This approach needs an intelligent display with a flash memory and a communication channel:

- The **intelligent display** is either MCU-based or has a safety-compliant display controller with an attached flash area
- The **flash memory** stores telltale images
- The **communication channel** is used to activate and deactivate the telltale rendering

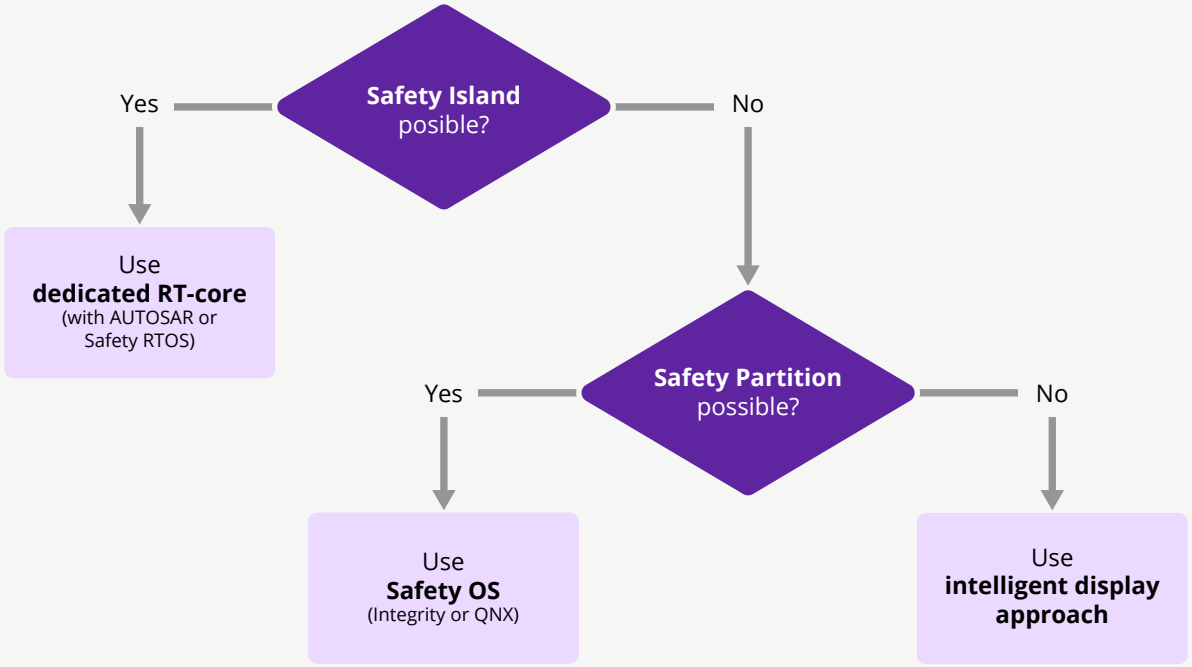
While this is a very reliable solution, it increases costs (additional MCU, flash, ...) and adds to the update complexity (how to update the software on the display side?).

How to find the approach that fits your needs

Following the steps shown in the diagram below, you will find the safety rendering solution that can best minimize your costs.

First, check if you can use the Safety Island approach. If so, use it. If not, you should check if at least a Safety Partition can be used, so you can work with a Safety OS. The intelligent display approach should only be your last resort, as it is the most complicated and expensive solution. No matter which solution you choose, always combine it with a platform strategy.

How to find the best safety approach



Still undecided? **Contact us** for more information about how Luxoft as a software supplier can support you on your journey towards safe rendering. Our experts are also happy to provide you with all the information you need to successfully integrate the Luxoft Safe Renderer into your projects.

About **the author**

Arwed Richert
Program Manager
LinkedIn

Arwed is head of Instrument Cluster inside the Digital Cockpit department at Luxoft. He holds a diploma in Electronics. Starting as a software developer, he later specialized in the area of HMI framework and tooling, before becoming the main development lead responsible for the Luxoft Safe Renderer. Currently, Arwed works as a consultant for safety-compliant system architectures. His expertise of more than two decades in the automotive industry has helped numerous clients reach better system safety.

About Luxoft

Luxoft, a DXC Technology Company delivers digital advantage for software-defined organizations, leveraging domain knowledge and software engineering capabilities. We use our industry-specific expertise and extensive partnership network to engineer innovative products and services that generate value and shape the future of industries.

For more information, please visit **luxoft.com**