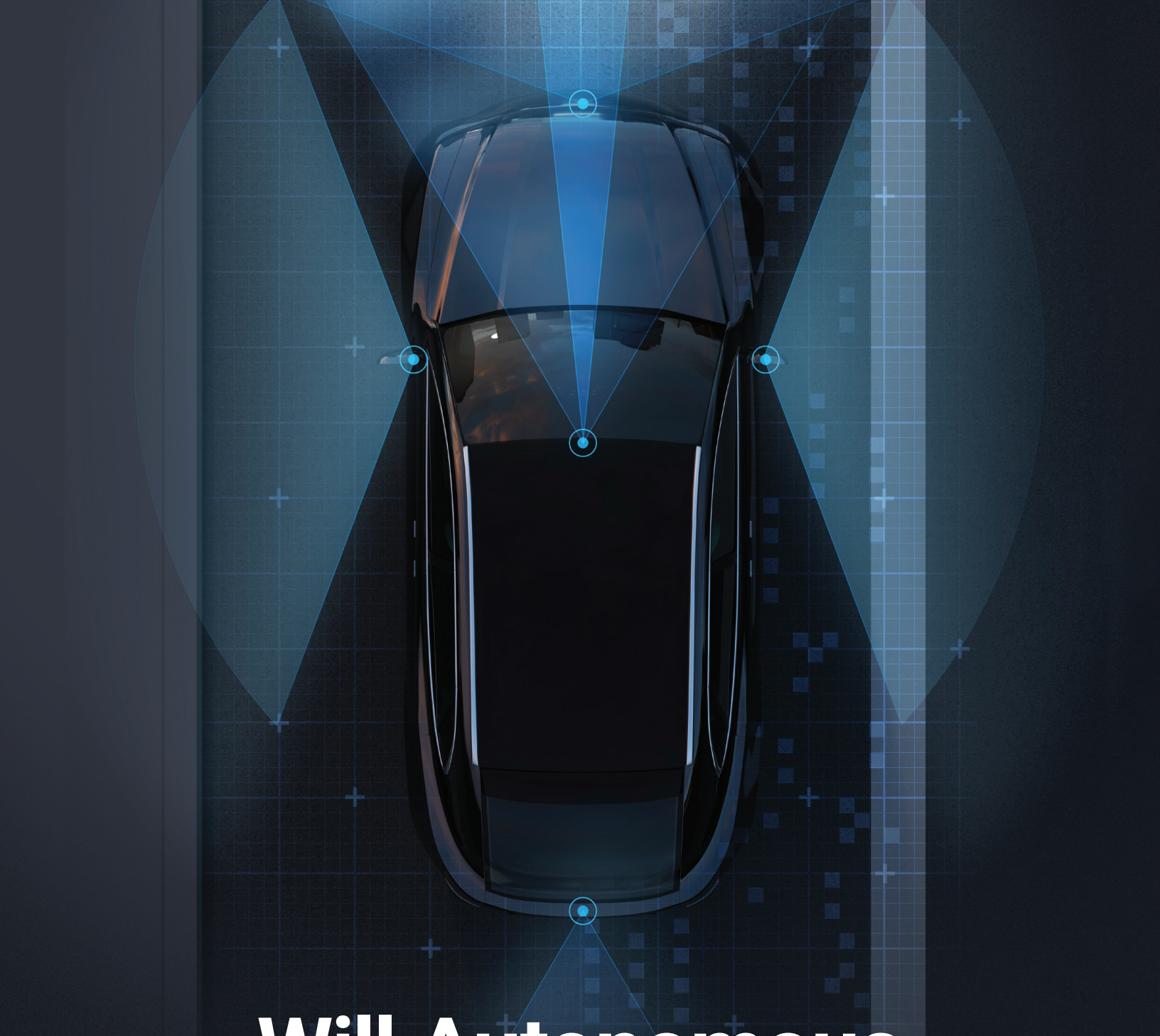


Will Autonomous Vehicles Ever Be Safe?

WRITTEN BY

Christopher Giordano
Vice President UX/UI Technology
The DiSTI Corporation
DiSTI®

Jim Carroll
Chief Technology Officer
Digica
DiGiCA
Intelligent Digital Solutions

Will Autonomous Vehicles Ever Be Safe?

UI Functional Safety in Autonomous Vehicles (AVs)

At first glance, it may appear that autonomous vehicles will not have a significant requirement for User Interfaces (UI) functional safety. There will be fewer driver controls, if any, in an autonomous vehicle, implying a diminished UI. Those controls that remain are likely to be infotainment related and not required to be compliant with functional safety.

However, although traditional driver controls may become less significant, the nature of autonomous vehicles brings new, additional requirements for vehicle control. Standardization of AVs will not occur overnight either, telling us that traditional driver-controlled vehicles and AVs will have to share the road during the transition. During this transition period, AVs will most certainly still require some level of conventional driver controls, which will have safety-related considerations. This paper discusses what those UIs may be and the challenges and potential solutions in developing the software architecture and creation process to accommodate the need for functional safety.

What UI is required in AVs

What UI goes into what part of a design is of some debate lately and depends on which sub-components the UI interfaces. For Functional Safety ASIL (Automotive Safety Integrity Level) rated UI, most of the emphasis is historically placed on the Instrument Clusters' critical components. They intrinsically connect to the integrated vehicle parts that control the vehicle's movements, motion, errors, and behaviors. These systems can create an environment for 'catastrophic' failure, resulting in loss of human life.

Varying domains until recently have had very different UI requirements, especially as it pertains to functional safety. With the rapid onset of AVs, ASIL-rated UI takes on an entirely new meaning for those creating the requirements and the use case in the AVs' different components.

Infotainment

For example, the infotainment system is historically part of the non-integrated systems that usually do not pose a safety threat for a vehicle in motion if a failure occurs. With the convergence of subsystems (one SoC managing multiple displays), infotainment is starting to play more of a pivotal role in the driving cockpit.

Functional safety requirements are also making their way from the Cluster and Heads-Up Display (HUD) into a single head unit system that has the power and capabilities to run multiple displays. While efficient from the hardware point of view, it presents unique challenges from the software developer's perspective, which we discuss in detail in this paper. Hardware and display technology have also both come to the point where mass-market automobiles can now employ HUDs. Bringing a new round of UI and software architecture decision making: What data to display on the windshield if that data should have ASIL ratings attached to it and what rating? As both Clusters and HUDs start playing more of a role in the Infotainment system, a functionally safe architecture along with a company's proper safety culture becomes of paramount importance to the success and safety of the brand.

Remote Control

Remote vehicle controls are also a strong candidate for ASIL rated UI, but precisely what the UI should be is not clear or consistent. There are several ways to think of remote capabilities:

- **Driver external to the vehicle in simple low-speed auto parking, or summons information, typically from a remote request key fob or connected mobile smartphone.**
- **Driver in the vehicle driving in traffic and automated take over events such as lane keep assist, emergency auto-braking, or adaptive cruise control.**

As defined in the first case above, remote capabilities are usually "fire and forget" commands to the vehicle. However, the UI still needs to be clear, consistent, and reliable. It does control the movement of the vehicle, which puts some level of ASIL rating to it, but potentially not on the UI itself.

As defined in the second case above, remote capabilities are more about minor adjustments in motion having the vehicle taking over in motion. The driver in this situation may not be able to or have the time to override these activities, hence, a safe and reliable architecture here is also critical.

Emergency Override

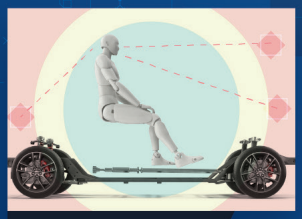
Where the UI is undoubtedly of critical importance in AVs is the concept of emergency overrides, also known as 'take over events.' Imagine your vehicle in motion on the highway at 120 Kph. Suddenly, an external event brings your vehicle to a new state where it does not contain the information required to make a decision. At that point, the vehicle invokes a takeover event, giving the driver seconds to know all the critical systems' states to make potentially life-critical decisions. That information is typically transferred from the vehicle to the driver, either audibly, visually, or both. Multisensory Integration studies show that a combination of audible and visual data produces an improved mean reaction time, hence a more timely cognitive response.[1] Thus, in order to improve the cognitive transfer, clear, concise and reliable UI becomes very important. There is a long-standing debate on moving directly to full autonomy and skipping the partial system, thereby removing the need for take over events. The paper discusses this in further detail after the next section on the different autonomy levels and what they mean.

1 - Rosemann, S., Wefel, I., Elis, V., & Fahle, M. (2017). Audio–visual interaction in visual motion detection: Synchrony versus Asynchrony. Journal of Optometry, 10(4), 242-251. doi:10.1016/j.optom.2016.12.003

A Long Way from Full Autonomous Vehicles

As AV's capabilities continue to progress, as do the system requirements definitions of any of these domains listed above, the UI required is still in flux. One thing is sure; Functional Safety plays an ever-increasing role in the UI and Software architecture as the systems increase in their complexity. Simultaneously, the traditionally driven vehicles versus AVs continuum gap narrows. The transition period between no automation to partial automation and eventually, full automation requires a clear definition.

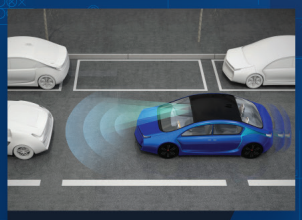
When it comes to autonomous driving, there are five different levels as defined by the Society of Automotive Engineers (SAE)[2]



Constant supervision

LEVEL 0 No Automation

The human does all the driving.



Steering OR Braking

LEVEL 1 Driver Assistance

An advanced driver assistance system (ADAS) on the vehicle can sometimes assist the human driver with either steering or braking/accelerating, but not both simultaneously.

Steering and braking during some circumstances

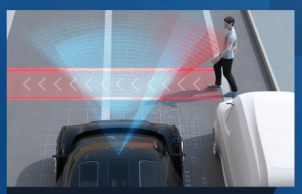
LEVEL 2 Partial Automation

An advanced driver assistance system (ADAS) on the vehicle can itself control both steering and braking/accelerating simultaneously under some circumstances. The human driver must continue to pay full attention ("monitor the driving environment") and perform the rest of the driving task.

All aspects and some tasks during some circumstances

LEVEL 3 Conditional Automation

An automated driving system (ADS) on the vehicle can itself perform all aspects of the driving task under some circumstances. In those circumstances, the human driver must be ready to take back control at any time when the ADS requests the human driver to do so. In all other events, the human driver performs the driving task.



All aspects and all tasks during certain circumstances

LEVEL 4 High Automation

An automated driving system (ADS) on the vehicle can itself perform all driving tasks and monitor the driving environment – essentially, do all the driving – in certain circumstances. The human need not pay attention in those circumstances.

All aspects and all tasks during all circumstances

LEVEL 5 Full Automation

An automated driving system (ADS) can do all the driving in all circumstances. The human occupants are just passengers and need never be involved in driving.

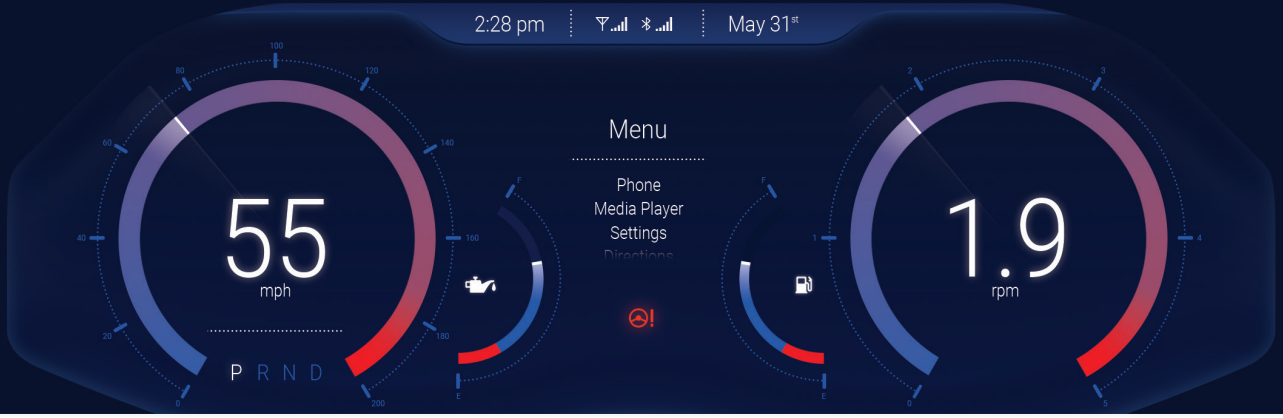
As mentioned earlier, many believe the 'rip the band-aid off' to autonomy is a safer approach and that we should move all vehicles directly to Level 5. In theory, that may be ideal, but there are other considerations such as costs, time to market, systems testing, and the publics' comfort level with such technologies.

The public's willingness to adopt AVs is still a mixed reaction. We are currently in a transition period. We continue to learn more and implement newer Artificial Intelligence systems to make life-critical driving decisions based on the mountain of data acquired. This fact highlights that while not entirely out of reach, we still have a way to go before reaching Level 5 autonomy comfortably and at least regionally. There are too many factors involved currently to make a solid leap to Level 5. The main questions being:

- **Is the technology used fail safe?**
- **Is the safety culture at the OEMs and Tier 1s that design and manufacture the vehicle acceptable?**
- **How does the AV interact with Non-AVs still on the road?**
- **Should there be some level of human control over the vehicles systems in motion and if so, what?**
- **Who is liable for accidents with a pure AV?**
- **Who becomes the centralized authority to define the standards mandated on OEM development while not increasing cost to the level when it stifles creativity?**

While there are proven safety, mobility, and economic benefits to AVs, one requirement is evident as we continue moving closer to complete Level 5 as a standard. There will remain some level of human controls or intervention capabilities involved with AVs, at least in the near term. This issue will likely continue until we prove the technology, and the public accepts that lack of control can lead to a significantly safer environment.

2 - Lynberg, M. (2020, June 15). Automated Vehicles for Safety. Retrieved from https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety



What is Important for UI in AVs

There is always an underlying corporate need for the UI to look great and be consistent with the branding. Some may argue that the UI's look & feel is the primary differentiation from one brand to another. As such, there is always the implicit understanding that one of the primary requirements is that the interface 'look good,' which has subjectively different connotations to different designers.

For safety purposes, the UI should also be intuitive and simple to minimize distractions and interaction times. As to the level of functional safety attached to those UIs, that remains open to interpretation by the OEM and suppliers. There are NHTSA, DOT, SAE, and EC guidelines but no hard fast strict FuSa rules to adhere to as per the specific autonomy level. These guidelines do change and should become more rigorous as the autonomy level increases. Pure autonomy will have its own level of what is important once we get there, however, for the near term during the transition period, from a safety perspective, what is essential for UI in autonomous vehicles is three-fold: simple, performant, and safe.

- **Simplicity to minimize the cognitive load and critical data transference to the human during the takeover events.**
- **Runtime performance to ensure the critical content draws as expected when it's expected.**
- **Functional safety to guarantee the critical components are visible precisely when intended by the system safety design strategy.**

Driving into the Future

From a UI perspective, accomplishing all of the above points can be handled employing proven commercialized software tooling, and a well-designed software architecture with safety culture in mind. Hardware and software continue to evolve and become more efficient, handling more data faster. As such, there will never be a one size fits all development process for every system. Rather the flexibility to encompass change and future proof your technology by allowing for fast and efficient adaptation becomes the cornerstone to any OEM and Tier 1 supplier's success.

For example, the recent trend toward a single system governing multiple displays was not easily possible ten years ago. As hardware became smaller, faster, and used less power but could still push massive amounts of data across the bus, the idea of having one more capable SoC governing multiple displays instead of each display having its own SoC became obtainable and financially viable. This change in hardware architecture facilitated the need for a shift in software architecture to blend systems that traditionally did not handle ASIL (IVI) with those that do (Cluster & HUD). This unified system posed a unique set of challenges in the software architecture from the developer's perspective. This challenge is mitigated by using highly flexible, commercially available tools that can adapt while not putting the underlying architecture at risk for fault.

Commercial Tooling

There is a perception that tools designed for functional safety are more expensive than those that are not. This tangential market tool-adoption approach also leverages sunken non-recurring engineering costs keeping automotive OEM efforts costs down for software certification.

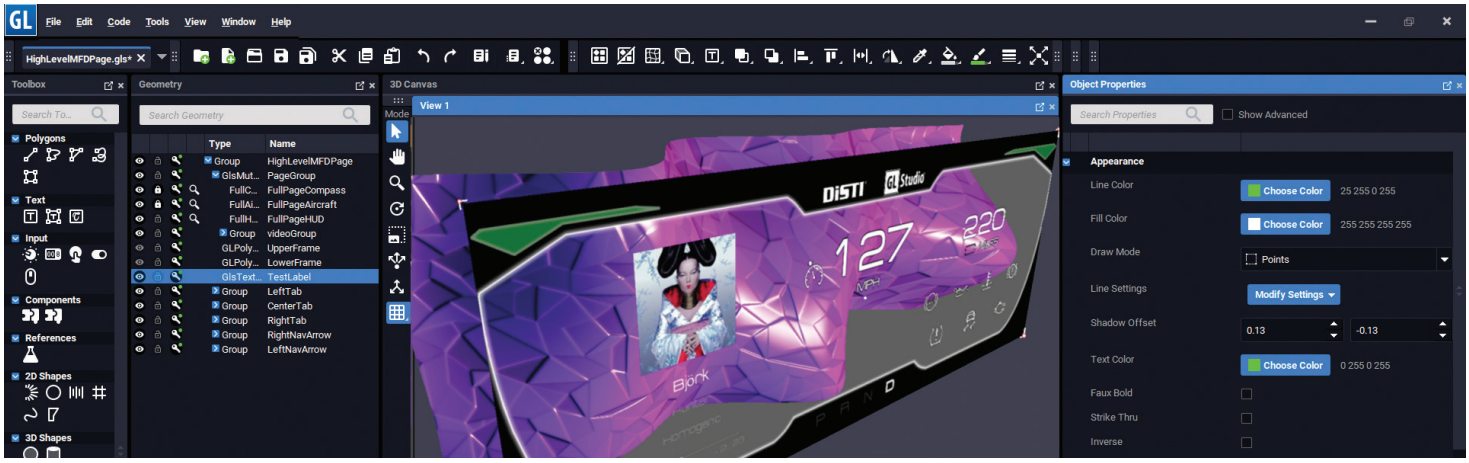
As published in "Aviation Week & Space Technology," November 2017, Rockwell Collins,

> *"It used to be that for a new avionics system, 75 cents of each dollar was spent on engineering and 25 cents on certification by global agencies. Today, that ratio has changed significantly."* [3]

Rockwell Collins' answer is a risk-based system, one of the aspects of automotive industry standards. The message is clear for avionics and automotive – there is much money spent on safety. How can OEMs reduce costs, while at the same time maintaining safety standards and not stifling creativity? **Employ the use of the proper commercial tools that have already borne the brunt of the cost and been proven in tangential related markets for functional safety.**

Automotive software certification has challenged automotive OEMs over the last decade. While certification efforts can improve automotive safety standards, the concern is that it can also stifle innovation and creativity and raise development costs.

In a time when automotive OEMs are looking to compress their development timeline, certification efforts have the perception of encouraging bureaucracy, thereby, conversely extending development timelines.



Semiconductor vendors have recognized the importance of providing FuSa compliant tooling to gain a competitive edge in the automotive sector. Arm offers a compiler feature package to provide safety-critical support; as Arm provides the Silicon IP for the processors included in many SoCs, this means that the semiconductor companies can adopt this support with minimal additional effort. The knock-on effect is that other SoC and Silicon IP vendors must provide support to remain competitive in this sector.

Leveraging the experience of tools employed in other similar domains is a very risk-free and low-cost approach to foster innovation. The GL Studio UI Toolkit, for example, has a pedigree of over 20 years and is currently flying in Aircraft, Spacecraft, and Helicopters, and used in Life Critical Medical devices around the world. GL Studio was also the first UI tool to certify to the ISO 26262 ASIL D standard in April of 2015. They have also passed the NQA-1 pre-certification for nuclear facilities, an even more rigorous process than the FAA's DO-178C DAL A certification for avionics.[4] The cost of development in these tools is handled by a wide array of markets and customers in different industries that have had the same considerations and needs for over 30 years, as automotive has had for the last decade. Thus leveraging the experience and existing tools for tangential markets such as Aviation that produce significantly higher quality via more rigorous testing and better performance becomes a low risk and low-cost solution.

All of this together yields a significantly more reliable software architecture in an age when ADAS systems and autopilot are on the horizon. What does UI have to do with this? Take over events or emergency override discussed earlier in this paper.

3 - Statler, K. L. (2017, November 13). The World Needs Seamless Aviation Certification Standards. Aviation Week & Space Technology.
4 - HMI and UI Design Software: Embedded Target Systems. (2019, June 12). Retrieved August 28, 2020, from https://glstudio.disti.com/features/safetycritical/

Embedded Software

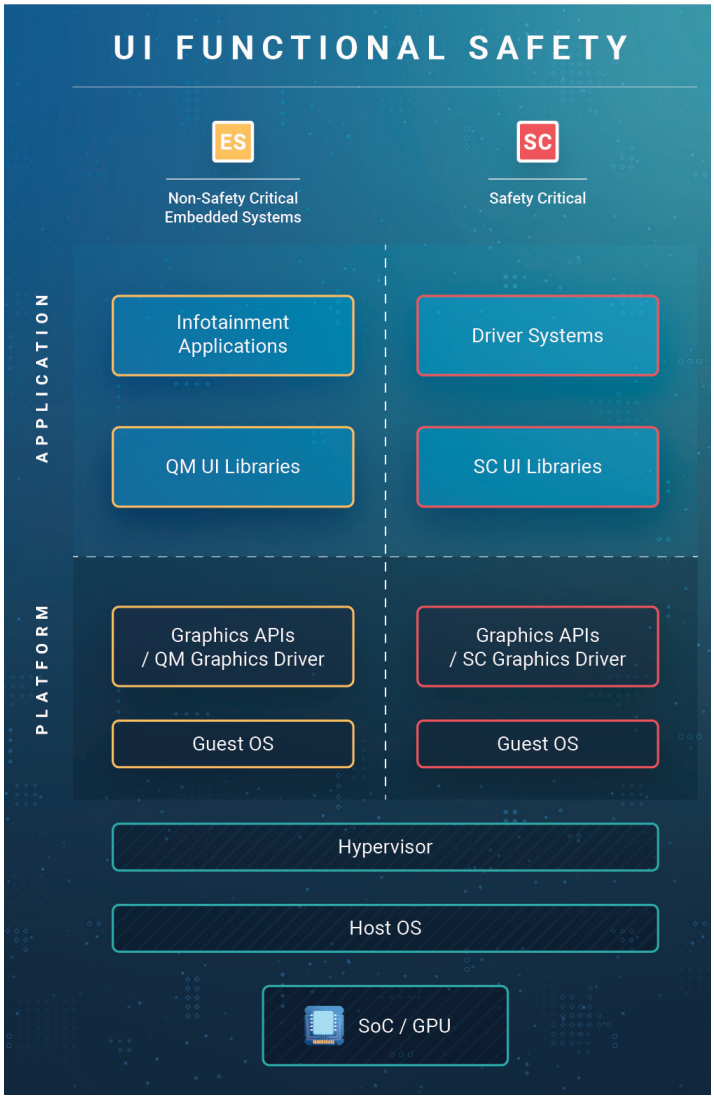
Support for UI Functional Safety in software is in two key domains: the platforms, which enable creating safety-critical applications, and the applications with which users interact.

Platform software comprises many components, including device drivers and other BSP components that directly control the underlying hardware components, such as GPUs, middleware implementing protocols and codecs, and abstraction layers providing consistent APIs to the application layers. Organizations such as Khronos specify industry-standard APIs to be used by application developers, such as OpenGL, OpenGL ES, and Vulkan. Safety-Critical (SC) variants of these APIs are available already with newer variations under development. SC compliant versions of the drivers have also been made available by the GPU vendors to support these APIs.

Where hardware resources are shared between vehicle functions, software virtualization is commonly deployed to achieve separation between functions. In such an environment, the host and guest drivers, and the hypervisor must also be functional safety compliant.

There are also capabilities for hardware virtualization and GPU separation on the same chip. While this is a related issue, which could influence the architecture used, hardware virtualization is outside the software architecture scope of this paper.

As the architecture in the figure to the left shows, software virtualized environments can be complex. This approach complicates the process of defect resolution before production, lengthening the overall development cycle.



Safety-Critical compliant tools and software components are maturing rapidly; in the medium term, hypervisors may become less important as such, software products simplify the construction of fully Safety-Critical compliant systems. This method will reduce time to market and development and software licensing costs. In some cases, SC variants of tools incur the same price as their non-SC equivalents - in these cases, using a fully SC compliant may simplify the software architecture with the added benefit of full SC compliance at no additional cost.

In the context of vehicle UIs, application software is primarily the user interface through which vehicle occupants interact with the vehicle. This context includes driver functions such as the cluster, environmental controls, and infotainment. Such applications can be built to be functionally safe by using safety-critical APIs and functional safety UI tooling such as GL Studio.

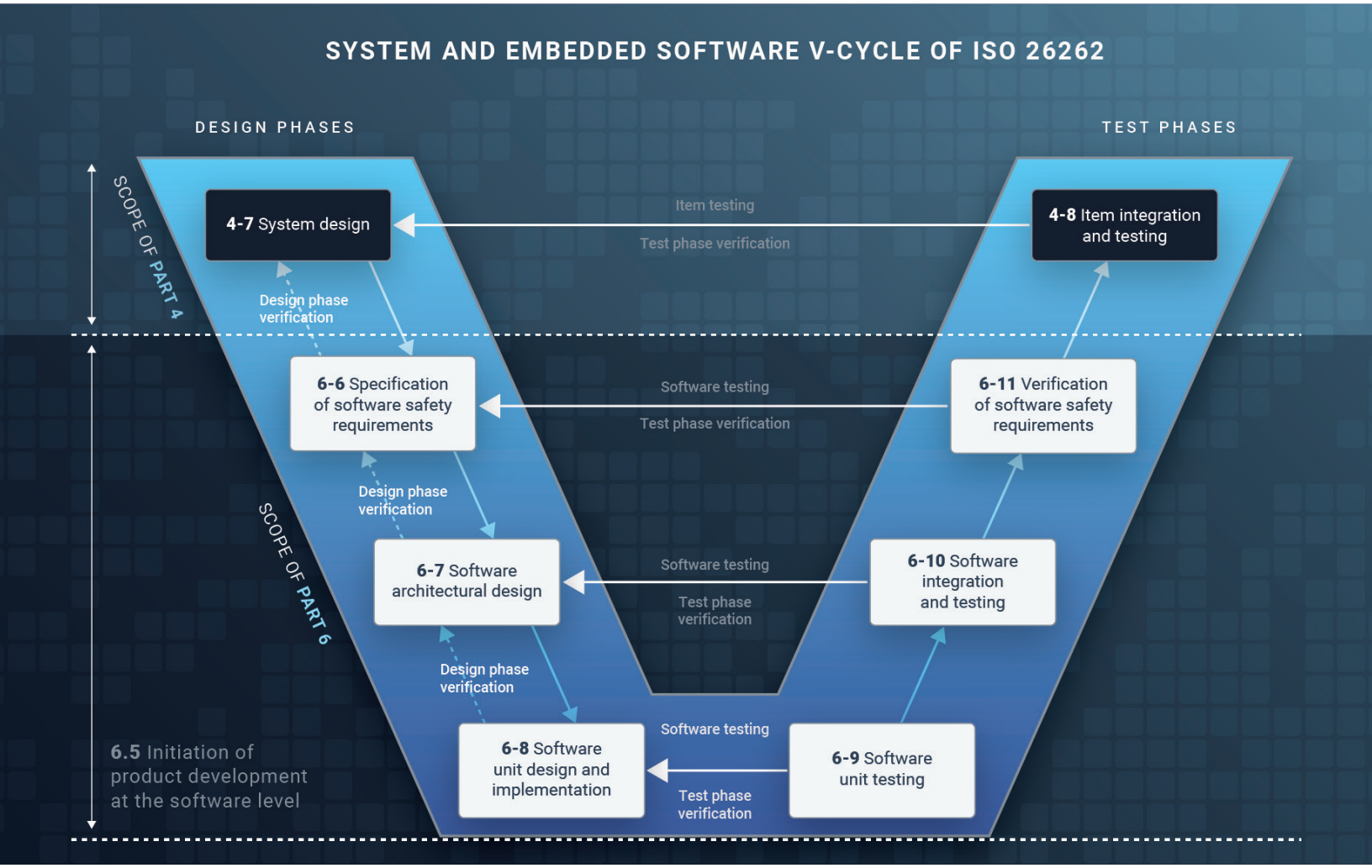
The selection of appropriate functional safety software components is crucial in a project's design phase before any code is programmed. Changing base technology selections during the development cycle will result in significant technical debt, additional integration work, and may compromise the resultant software's functional safety.

Automotive Safety Integrity Level (ASIL)

The key to functional safety is the assessment of risk. Strict processes are imposed during vehicles' development to ensure that the end vehicle users' risk is minimized. A risk classification scheme, known as ASIL (Automotive Safety Integrity Level), is frequently used to assess such risks. There are four risk levels, named ASIL A (lowest) to ASIL D (highest). Hazards assessed as ASIL D represent life-threatening risks, and such developments are subject to the strictest development processes.

In general, the principle to apply when working with ASIL is to work with the highest level that does not incur an additional cost, while complying with industry standards and regulatory frameworks. Such an approach improves product reliability; it will also allow for flexibility in the event of the requirements change. Given the costs of developing software in safety-critical environments, extending its lifetime and reducing long-term costs is critical.

ASIL is a subset of the broader standard, ISO26262, which also provides guidance for product development processes in the automotive industry. Compliance with such measures in the development of vehicle UIs guarantees functional safety.



The process is designed to ensure that risks identified under ASIL are documented, with solutions designed, implemented, and tested appropriately. A “V-model” process is described by the standard, which emphasizes several key features:

- **Engineering process, including traceability by documented requirements, design and implementation.**
- **Structured verification of the outputs of each phase of the development, by peer processes in the test cycle.**
- **Specific focus on the safety requirements of the system, addressing ASIL identified risks.**

The V-model (and variations of it) is well understood and is commonly deployed in embedded software development projects. Its usage applies equally to creating all automotive application stack components, including infotainment and driver applications, UI libraries, and graphics drivers.

All UI software components must be developed according to these standards to be SC compliant.

GPU

Modern UIs are incredibly complex, and pervade every corner of a modern operating system, in every deployed context. The UI experience is at the forefront of the user experience from the moment that the user steps into the vehicle, from the OEM's splash screen through the menu system to the navigation application. Because of the complexity and prominence of the user experience, performance is crucial. All modern UIs are built on hardware that includes a GPU so that graphical functions can be offloaded from the application processor.

The GPU's complexity is such that SoC companies often license GPU designs from third parties to include in their devices. Imagination Technologies and Arm are the leading vendors of "licensable" GPU designs. Other SoC companies design their GPUs that are well integrated with their application processors; this approach is taken by companies such as Nvidia, Qualcomm, and Intel.

The industry has formed bodies such as Khronos to define standard APIs for graphics programming to simplify software integration. OpenGL was first specified for desktop PCs almost three decades ago. It has been through many revisions as the underlying hardware has evolved and has spawned several related APIs such as OpenGL ES, for use in embedded contexts.

Safety-Critical variants of both APIs are available, as are GPU vendor drivers from companies such as CoreAVI. Microsoft specifies a competitor API, Direct3D. Although Windows is commonly used in infotainment systems, most such systems are Linux-based and use Khronos APIs. Apple also defines a competitor API, Metal. Currently, there is no safety-critical variant of Direct3D or Metal; Metal has yet to be deployed in a commercial, automotive system.

As GPUs are also used for more generic computation, Khronos also specifies a compute API, OpenCL. They have developed an alternative to their set of APIs, called Vulkan. This method unifies graphics, embedded, and compute APIs. Vulkan may replace its older "siblings" in due course, but this is by no means a certainty - the programming model is significantly more complicated than its predecessors. To unify graphics and compute in a single API, much of the logic previously encapsulated in the GPU driver must now be implemented by the application developer or the vendor of UI tools and libraries. Although this allows for greater feature differentiation for tool vendors, it introduces significant additional hardware-related complexity into components that have hitherto been abstracted from the hardware dependencies. In 2019, Khronos started work on a safety-critical variant of the Vulkan API.

Other Considerations

Vehicle OEMs are continually striving for a reduced bill of materials cost in the vehicle. This approach may mean removing some SoCs, which implies the sharing of physical processors between vehicle functions. As discussed earlier in the paper, there are GPU virtualization capabilities as well, which would be related to this feature. Where safety-critical and non-safety critical functions coexist on a single processor, it is technically simpler to use a single safety-critical UI software stack across all functions hosted on that processor.

Historically, in-car systems have been based on standard SoC SKUs. SoCs are highly complex devices, which are therefore comparatively expensive. For specific applications, it may be possible to establish a system on lower-cost silicon, such as FPGAs or MCUs, while still achieving the critical requirements of high-quality user interface and SC compliance. Several ISVs provide software renderers; this removes the need for a GPU and the complexities of sharing it between operating systems. A GPU is by far the largest processor included in an SoC, so removing it also results in a smaller system footprint. At the time of introducing APIs such as OpenGL ES, software renderers did not perform well enough for inclusion in an automotive SC environment. Today, their performance is much improved. The possibilities of FPGAs, including GPUs, adds an extra dimension to the architectural decisions.

MCUs are typically used for "smaller" applications, which frequently do not include a user interface. They are already heavily used in an automotive context, as the basis for ECUs for SC components such as braking systems. Like FPGAs, MCUs have also increased in processing power to the point where software renderers perform well enough to be used in a broader range of in-car applications. MCUs have much lower power requirements than SoCs or FPGAs; this may also reduce cost.

Cost savings may be possible through hardware selection, but it is vital to consider the impact on related costs such as software licensing and development.

Conclusion

As we continue moving closer to the adoption of full Level 5 Autonomous Vehicles, it is evident this must be accomplished safely. For the many reasons defined in this paper, the AV continuum is slowly narrowing in on the path forward. While it is clear that while we continue to strive towards it, full Level 5 automation is not in our immediate future. There are too many factors involved in one swift and sweeping move direct to Level 5. While there are many benefits to be had from pure autonomy, the automotive industry must overcome infrastructure issues, socio-economic problems, human emotions, and, most importantly, to this paper, technological challenges.

While we need to continue progress, this paper mostly covers the need to do so in a safe and economically viable manner. As mentioned in a well-known study on Model-Based Design, “80% of development costs are spent identifying and correcting defects during the integration phase" of a software project.[5]

WHY MODEL-BASED DESIGN?

Lost opportunity

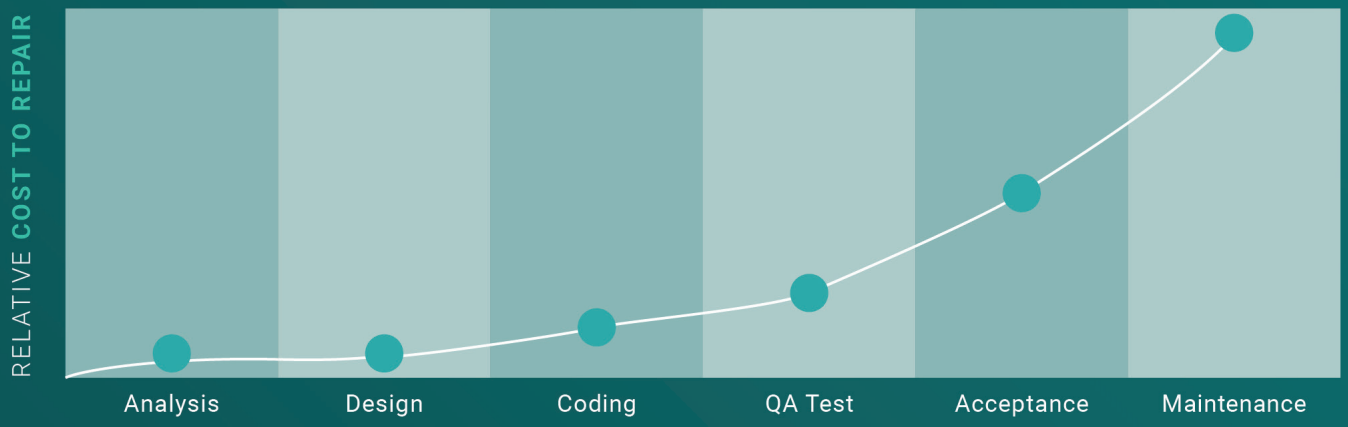
Late to market by six months or more will cost organizations **33% of the five-year ROI** (Source: McKinsey Group)

Cost

- **80% of development costs** are spent identifying and correcting defects during integration
- **More than 40%** of IT development budget will be consumed by poor requirements (Source: IAG Consulting)

Traditional QA Testing

- **25 - 30% delivery time** in testing (Source: IBM study)
- Poor upstream quality yeilds rework
- Compressed schedules make it worse



Software quality optimization: balancing business transformation and risk, Michael Lundblad, program manager, Rational software, IBM Software Group, Moshe Cohen, offering manager, Rational software, IBM Software Group

While not a typically well-accepted principle due to initial costs, the key is to invest in the right hardware and software architecture and workflow upfront. Ultimately, the flexibility required to future proof your software design and accommodating new technologies and architectures comes at a cost upfront. Given the above study, this does save significant effort and thereby costs over the life of your project if properly implemented up front.

The primary goal in developing your embedded graphical software should be safety. Beyond that, the costs and look and feel can be balanced with the appropriate upfront architecture design while leveraging the existing COTS tools that already support safe software that will not stifle creativity or exceed the budget.

5 - Lundblad, M. & Cohen, M. (march 2009). Software Quality Optimization: Balancing business transformation and risk. IBM Software Group

About the Authors

Christopher Giordano

Vice President UX/UI Technology

The DiSTI Corporation

Since 1997, Chris has focused on developing UI and HMI Software starting at the U.S. Navy and University of Central Florida. Chris has worked at DiSTI since 1999 as lead engineer or program manager for over 60 different programs, and eventually the product manager for all DiSTI's UI development tools. Chris managed DiSTI's HMI/UI programs for Boeing, Hyundai, Jaguar Land Rover, Lockheed, NASA, Nissan Motors, Northrop Grumman and The Space Ship Company to name a few and is currently DiSTI's VP of UX/UI Technology. He has successfully managed DiSTI's UX/UI business for over a decade developing a global leadership position as experts in HMI/UI and Functional Safety. Chris holds bachelor degrees in Finance from UNCW and Computer Engineering from UCF graduating with honors.

Jim Carroll

Chief Technology Officer

Digica

Jim Carroll is a highly experienced technical director with a background in embedded systems. Over 25 years, Jim has designed and built large-scale software solutions reaching millions of users working as a software engineer, technical architect and CTO. The majority of Jim's work has been in the semiconductor sector, working with companies including Intel, Arm and Imagination Technologies. He has also been a contributor to the software standards group, Khronos.