ISO26262 Compliance Using Approved Software Components for Road Vehicles

A Verocel and RTI Whitepaper

Joe Wlad, Vice President Business Development, Verocel, Inc.

Introduction

Software-driven systems are now the mainstay of innovation in every industry. Nowhere is this trend more profound than in automotive applications. Use of software in cars has spread from basic system diagnostics to information and entertainment systems to complete autonomous driving features. Modern vehicles are now shipped with tens of millions of lines of software that manage engine and transmission controls, braking, steering and a host of diagnostic information on every subsystem. This trend has compelled automotive designers to address safety in a way that includes system, hardware and software design.

There are many standards and guidance documents applicable to development of software for safety-related applications, but most are unique to a given industry. For example, the aviation industry uses RTCA/DO-178C, industrial developers may use IEC61508 and medical device manufacturers may use IEC62304. The automotive industry has adopted ISO26262 as its functional safety standard for electronic systems which include both hardware and software. ISO26262 was adapted from IEC61508 and it has many common requirements, but it also has some unique differences, especially in areas related to determination of safety levels.

A common question from our base of customers is how does one use commercial-off-the-shelf (COTS) software (that may or may not have a proven safety pedigree) in a system destined for ISO2626 approval? Before one can examine how to apply COTS software in an ISO26262 system, one first needs to understand the content and organization of the ISO26262 standard.

ISO 26262, Edition 1 is composed of ten parts and covers the safety lifecycle aspects of electric and electronic automotive systems. Figure 1 below is taken from ISO26262 and summaries the key requirements of each part. Parts 3 through 7 are the core parts that deal with product development from the concept phase through design and production. An outline of the ten sections is given below with brief description of what each part entails.

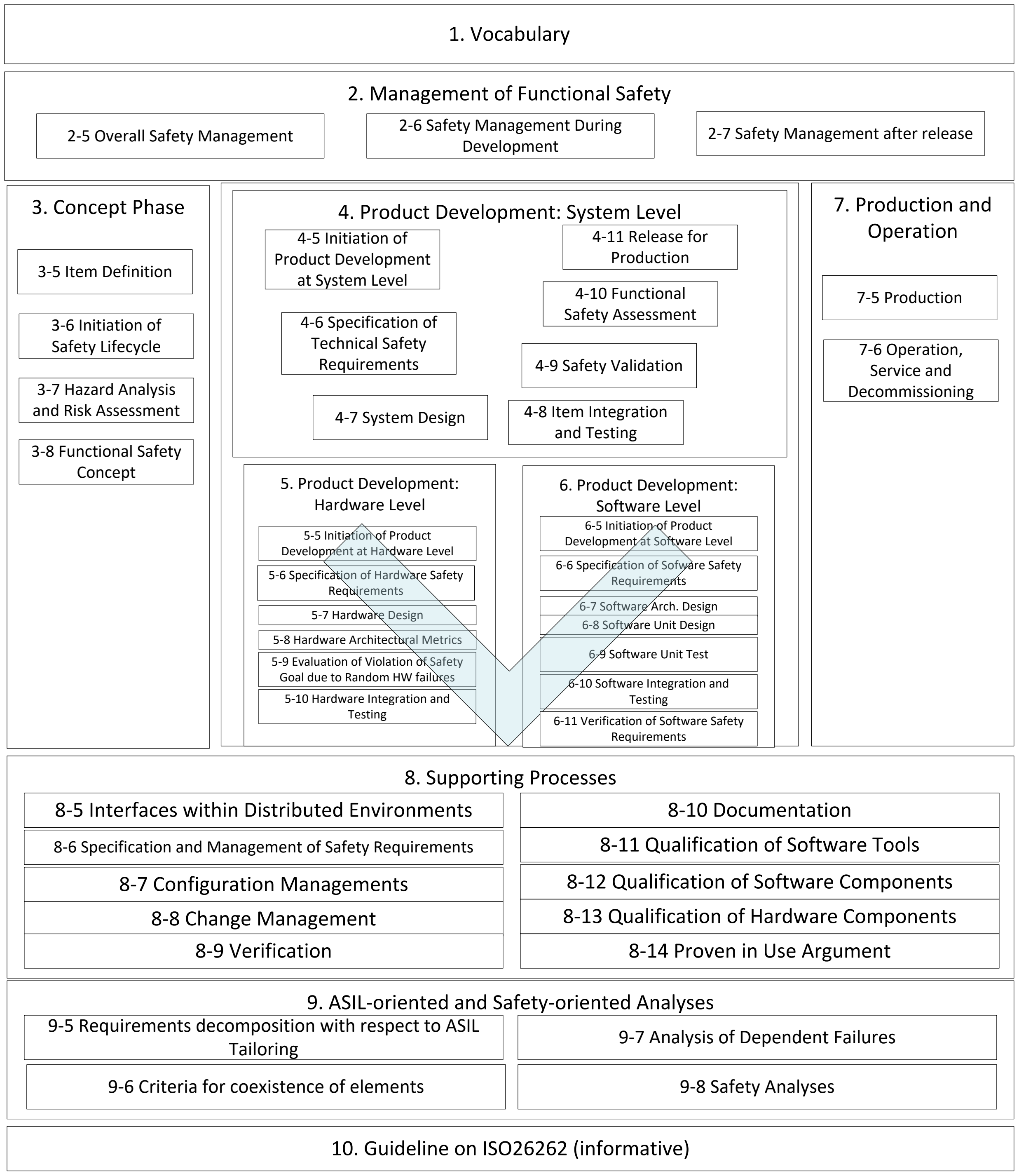

1. Vocabulary
2. Management of Functional Safety
2-5 Overall Safety Management
2-6 Safety Management During Development
2-7 Safety Management after release
3. Concept Phase
3-5 Item Definition
3-6 Initiation of Safety Lifecycle
3-7 Hazard Analysis and Risk Assessment
3-8 Functional Safety Concept
4. Product Development: System Level
4-5 Initiation of Product Development at System Level
4-6 Specification of Technical Safety Requirements
4-7 System Design
4-8 Item Integration and Testing
4-9 Safety Validation
4-10 Functional Safety Assessment
4-11 Release for Production
5. Product Development: Hardware Level
5-5 Initiation of Product Development at Hardware Level
5-6 Specification of Hardware Safety Requirements
5-7 Hardware Design
5-8 Hardware Architectural Metrics
5-9 Evaluation of Violation of Safety Goal due to Random HW failures
5-10 Hardware Integration and Testing
6. Product Development: Software Level
6-5 Initiation of Product Development at Software Level
6-6 Specification of Sofware Safety Requirements
6-7 Software Arch. Design
6-8 Software Unit Design
6-9 Software Unit Test
6-10 Software Integration and Testing
6-11 Verification of Software Safety Requirements
7. Production and Operation
7-5 Production
7-6 Operation, Service and Decommissioning
8. Supporting Processes
8-5 Interfaces within Distributed Environments
8-6 Specification and Management of Safety Requirements
8-7 Configuration Managements
8-8 Change Management
8-9 Verification
8-10 Documentation
8-11 Qualification of Software Tools
8-12 Qualification of Software Components
8-13 Qualification of Hardware Components
8-14 Proven in Use Argument
9. ASIL-oriented and Safety-oriented Analyses
9-5 Requirements decomposition with respect to ASIL Tailoring
9-6 Criteria for coexistence of elements
9-7 Analysis of Dependent Failures
9-8 Safety Analyses
10. Guideline on ISO26262 (informative)

Figure 1: ISO26262 Processes and Requirements (ref. ISO26262)

- Part 1, Vocabulary: provides a definition and description of terms used in the standard
- Part 2, Management of Functional Safety: provides requirements and guidance on how the organization and internal processes will be used during development and post-release
- Part 3, Concept Phase: provides requirements and guidance for item definition, description of the safety lifecycle including the activities and the hazard analysis and risk assessment
- Part 4, Product Development: System Level: provides requirements and guidance on development of system requirements, safety requirements, safety concept, system design, integration testing, safety validation, safety assessment and product release
- Part 5, Product Development: Hardware Level: provides requirements and guidance for initiation of hardware product development, hardware safety requirements, hardware design which includes evaluation of hardware architectural metrics and potential safety goal violations due to hardware failures
- Part 6, Product Development: Software Level: provides requirements and guidance for initiation of software product development, specification of software safety requirements, software design, unit implementation, unit testing, integration testing and verification of software safety requirements
- Part 7, Production and Operation: provides requirements and guidance for development and maintenance of a production process for safety-related elements used in road vehicles as well as achieving functional safety in the production process
- Part 8, Supporting Processes: provides guidance and requirements for overall safety management, interfaces within distributed developments, configuration management, change management, verification, documentation, use of tools, qualification of hardware and software components and proven-in-use arguments
- Part 9, ASIL (Automotive Safety Integrity Level)-oriented and Safety-oriented Analyses: provides guidance and requirements for ASIL decomposition (system, hardware, software and across components) such that safety goals are inherited through safety requirements as well as failure analyses and safety analyses
- Part 10, Guideline (Informative): Provides readers the key concepts of ISO26262, differences with IEC61508 and additional details on ASIL decomposition, safety cases, hazard analyses and risk assessments

Each subsection of ISO26262 is formatted in a similar way. The document provides a list of objectives for each general requirement (e.g. initiation of product development at the software level), inputs (e.g., safety plan), requirements (e.g., activities for product development at the software level shall be planned) and work products (e.g., software verification plan). From a holistic point of view all the sections of ISO26262 must be addressed (at the appropriate Automotive Safety Integrity Level) to be considered compliant. The high-level steps to compliance will include creation and approval of a safety plan, safety goals and safety case along with a complete safety lifecycle with bi-directional traceability. The activities and work products will include verification, validation, independent assessment (by an accredited agency) and a complete list of documentation supporting the required activities.

Additionally, the structure of ISO26262 is encapsulated in a way that permits activities, validation and assessment for each major part of development to take place independently. Therefore, one conceivably can take a hardware or software component, assess and approve it in one system or element and then reuse the approval in other systems and assessments. In fact, part 8-12 and 8-13 of ISO 26262 directly address requirements for approval of software and hardware components, respectively.

ISO 26262, Part 6: Software Requirements

As stated earlier, part 6 is a core process requirement of ISO26262 and as such does not include supporting processes of functional safety managements, configuration management and change management and more. The part 6 requirements are specified in a waterfall model of development and verification. While ISO26262 does not assume any particular model of software development (agile, iterative, etc.,) it is convenient to document requirements in way as if the software were designed in a waterfall model. In fact, other software certification guidance and standards such as DO-178C and IECE61508 are written in a similar manner. The first requirement in developing software to be compliant with ISO26262 is to initiate a safety plan and software verification plan. These plans would of course be supported by other process plans (defined in parts 2 and 8 of ISO 26262).

Once the initiation activities are complete, the specification of software safety requirements process takes place. Here there is an inexorable link to the safety concept and overall system design. The system design and/or hardware design will place limitations and burdens on software and these constraints need to be considered as part of the requirements process. A good candidate for a software component approved under ISO26262 would likely have a well-defined and exposed interface into any system or hardware design. This does not mean that any system or hardware would support a given software component but that any limitations or conditions be exposed to an integrator (for example, memory and CPU constraints would be specific requirements if needed). The work products resulting from this phase include a software requirements specification, refined hardware-software interface specification and results of the software verification activities.

The software architectural design process defines the requirements that permit the software to be written without ambiguity. The design may be specified in a number of ways (formally or semi-formally) and ISO26262 has recommendations on what notations should be used depending on the required ASIL. Additionally, ISO26262 specifies properties of software design that are conducive to safety elements (such as restrictions on size and complexity). If the design process employs partitioning of software components, then additional requirements are imposed to demonstrate that freedom from interference is preserved. The methods for verification of the software design varies by ASIL but for the higher criticalities, design walkthroughs, inspections and control and data flow analyses are either recommended or highly recommended. The work products resulting from the design process are a software design specification, failure analysis report (if partitioning is used) and documented verification results.

The software unit design and implementation phase is commonly known as the coding phase, where the software requirements and design information are used to produce software modules. Requirements for the coding phase include having consistent interfaces, a robust implementation, verifiability and testability, among others. ISO26262 requires that the implementation depends on design detail that is sufficient to write each module or function. In other words, if low-level requirements or design information is not uniquely traceable to software functions, a compliance gap may exist. The properties of the implementation required by ISO26262 include no recursions, limited use of pointers, no unconditional jumps and more. The verification activities for software implementation include static code analysis, compliance with coding standards, control flow analysis and data flow analysis and compliance (including traceability) with stated requirements and design data. The outputs of this phase include verification results and completed software implementation.

The software unit testing phase may appear to some to be misleading in the sense that it’s simply structural unit testing of individual software modules. ISO26262 requires that the testing include verification of proper implementation of requirements and design and therefore the test cases should be created directly from the requirements and not from the implementation. Additionally, the unit test phase should demonstrate that no unintended behavior exists in the implementation and that the implementation is robust (e.g., can respond gracefully to abnormal input conditions). Completeness and absence of unintended behavior is demonstrated by showing the required coverage for the associated ASIL (e.g., statement, decision or modified condition/decision coverage). The outputs of this phase includes a verification report demonstrating the test results are correct, complete and produced the required coverage.

The software integration and testing phase is intended to demonstrate that software elements (or components) verified in the software unit testing phase can be fully integrated and completely tested. The integration phase may involve both safety and non-safety related elements. Integration testing will demonstrate correctness of the hardware-software interface as well as verify compliance with the software architectural design. Integration testing includes requirements-based tests, integration tests, fault-injection tests, resource usage tests (stack, heap and execution times among others) and ideally should be performed on the actual target hardware of the software application or applications. The metrics that apply to integration testing are function coverage and call coverage. The functional and call coverage tests may reveal software that is not exercised and in these cases, the software should be removed or otherwise declared as deactivated. An analysis of deactivated code should be done to confirm that presence of the deactivated code does not impair operation of the verified software. It is assumed that the deactivated code is not dead code in the sense that it has no traceability to requirements since the unit testing phase should expose any dead code. The outputs of this phase are the complete integrated software and the verification report confirming successful completion of the integration testing objectives.

The verification of software safety requirements is the last stage of software verification in ISO26262. This stage is used to confirm the software requirements are met in the actual target environment. Up to this point, testing and verification may take place on simulated hardware or reference hardware not intended to be used in the actual production vehicle. The test environment at this stage may be the actual vehicle, integration test bench, or network environments that replicate the actual vehicle architecture related to the software under test. The outputs of this phase is the verification report confirming successful completion of the testing objectives in the actual hardware environment.

COTS Approach to Satisfy Part 6 Requirements

The organization of ISO26262 Part 6 and other ISO parts are highly conducive to approval, assessment and reuse of COTS software components intended to be used in a variety of applications. First, the organization of the software lifecycle requirements and activities in part 6 are staged, meaning that the outputs of one phase are the inputs to the next. There are 3 levels of software testing defined in part 6 and each of these testing is the result of increased software integration. For example, a software component developer may test a component such as a distributed data service or software bus on reference hardware that may later be integrated into an automotive control system. In this case the software component developer would satisfy a subset of the requirements in part 6 (as well as other parts) and leave the remaining compliance objectives to the integrator.

The other parts of ISO26262 that are relevant to software components include part 2 (Management of Functional Safety) and part 8 (Supporting Processes). The software component developer would need to demonstrate that its internal planning processes are aligned with part 2 requirements. At a minimum this would include a functional safety management plan and a quality management plan as well as evidence of a safety culture and personnel who are trained and responsible for enforcing the safety culture in both the development and production phases. Additional plans that would be required (and defined in the safety plan or quality plan) are the verification plan, validation plan and test plans that demonstrate adherence to ISO26262 requirements. Once a supplier can support the requirements of part 2, they have a framework to ensure their software developed under part 2 requirements can support the other parts of ISO26262.

The planning process will also include references to any applicable tool qualification plans. This could include software modelling tools, coverage analysis tools and any other tools used in the development or verification activities.

ISO26262 part 8 is relevant for software component suppliers. Part 8 defines requirements for documentation control, configuration management, change management and requirements for qualification of tools used under ISO26262. Clause 12 of part 8 defines the requirements for qualification of software components, thereby making explicit the notion that approval of software components under ISO26262 is possible.

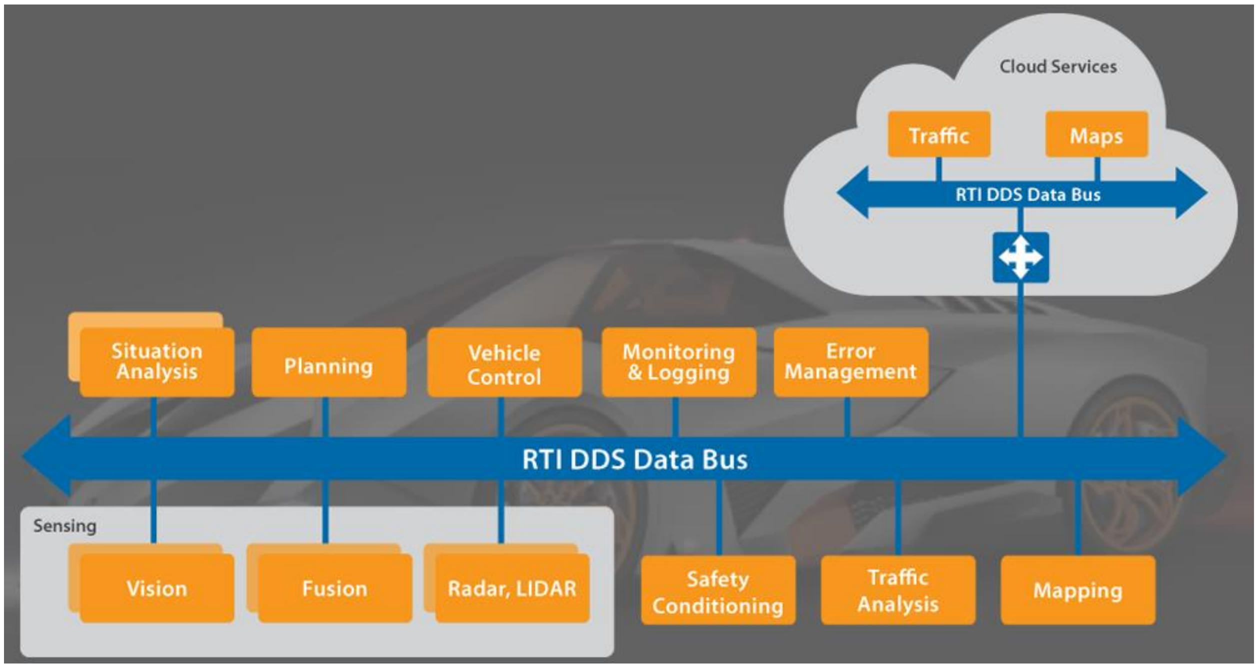
The requirements to qualify a software component under clause 12 of part 8 include:

- Requirements that address functionality, resource usage, response times, behavior under failure conditions and robustness
- Description of the configuration, interfaces and how to integrate the component, description of operation under abnormal conditions and a description of known limitations and workarounds (to include both existing defects and limitations)
- Verification results showing full coverage of all applicable software requirements, including robustness for normal and abnormal conditions and including the required structural coverage analysis applicable to the proposed ASIL
- Documentation of the software identification and configuration, targeted ASIL, hardware compatibility limitations, organization performing the qualification and the results of the verification measures applied to the software component
- The qualification results and the validity of the results must be verified and if required additional activities may need to be performed

The last point above becomes a key factor in determining the value of software component reuse in ISO26262. Establishing the validity of the qualification results in a different environment will always compel some additional activities on the part of the integrator. If the supplied software component does not adequately document the scope of usage, approval and limitations, the integrator may be challenged in trying to apply ISO26262 credit for the component in their respective environment. Additional information on considerations for using qualified software components is presented below.

RTI Connext® DDS Micro as an ISO26262 Qualified Software Component

Real-time Innovations, Inc. (RTI) has a certifiable version of its Connext DDS product that is widely used in mission and safety critical applications. Connext DDS provides software developers and integrators with high-level interfaces for distributing real-time data between devices, applications and subsystems. For example, Connext DDS can be used to stream video, radar and LIDAR data to analytics and autonomous driving applications as well as to integrate those applications with traditional ECUs. Connext DDS is often referred to as "communications middleware" since it is a library that sits between applications and the underlying operating system and network stack, providing developers with high-level publish/subscribe interfaces that abstract low-level networking details.



Connext DDS Cert supports the DDS (Data Distribution Service) family of standards and is a certifiable middleware available with a complete, commercially supported certification package to support ISO26262 certification, including ASIL-D. The package includes all of the evidence required by a certification authority such as TÜV SÜD.

Using certified middleware that is conforms to a widely used industry standard has a number of important benefits

- Significant cost savings; by significantly reducing the amount of custom communications and integration logic required, Connext DDS Cert can save tens or even hundreds of thousands of lines of code, avoiding potentially millions of dollars in certification cost.
- Reduced risk; to develop safety software a stringent set of procedures must be followed making the software expensive and risky to develop, or redevelop, if these procedures weren't followed in the original development.
- Leverage experience; RTI has used DDS in many mission and safety critical industries, the software, DDS standard, and certification package are developed based on deep architecture experience from these 1000+ projects.
- Reliability; a new standard or custom development effort does not have the years of project experience and in-field deployments that demonstrates reliability over the long term.
- Interoperability; DDS is supported by a number of vendors and interoperability is ensured with frequent testing against the standard and between the leading vendors.
- Open Architecture; Connext DDS aligns with many open architecture initiatives including the Future Airborne Capability Environment (FACE), UAS Control Segment (UCS) Architecture, and Open Mission Systems (OMS).
- Component Isolation; perhaps the most important benefit of certified middleware, a communications framework certified to the highest safety standard will isolate the various software component of a system and allow them to be certified to different safety levels, even when they communicate or share information between them.

Certified middleware can continue to return costs savings through the entire product life-cycle. After the initial costs savings in development and certification, the combination of component isolation and the loose-coupling of applications can reduce maintenance cost and life-cycle development costs by allowing upgrades and re-certification of individual components without re-certifying unmodified portions of a system. These costs savings can dwarf the original certification cost over the life of the product and can be an important differentiator of any software system.

In the Automotive market, this especially applies to in-car advanced driver assistance system (ADAS) and Autonomous Drive applications where multiple software components must share data. In a well architected system these interdependent systems should be loosely coupled and will need varying levels of certification, depending on their function. For example, a sense-and-avoid braking system would likely need certification to ASIL-D and any software critical to the sensing, decision making, and resulting action would need this certification (i.e. sensors, ECUs, software algorithms, braking, and steering modules). However, functions like navigation and path planning may need to interact with some of the same components but would likely need a much lower certification level or no certification at all. Without certified middleware to isolate and separate these functions, every component would need to be certified to the highest certification, which would be very expensive and would limit the possible features and functions of a system.

Considerations for Using ISO26262 Qualified Software Components

As stated above, ISO26262 part 8, clause 12 addresses the requirements for software reuse. Reuse could apply to a vendor's own software or COTS software procured from a third party. Use of COTS components create some obstacles for integrators when trying to approve these components due to lack of familiarity, use of different standards and plans and likely different verification methods and tools. Even though the COTS supplier can provide the required documentation and data to substantiate ISO26262 compliance, the integrator is still responsible for obtaining ultimate approval for its use. In Verocel's experience, software components for use in ISO26262-approved systems should have the following characteristics:

- Have few, if any hardware dependencies
- Be easily portable to varying hardware platforms
- Have clear boundaries with other software components and hardware
- Be provided in binary or pre-linked form, obviating the need for rebuilding
- Be of limited complexity
- Be adaptable for modification and expansion with minimal change impact

The characteristics defined above offer both economic, technical and certification value to both the supplier and user. It is expected that any approved software component will be used in a variety of applications or it would not be of much use to an integrator. Examples of good reusable software components are operating systems, communication and messaging software (such as RTI Connext DDS Cert), language and graphics libraries and file system interfaces, among others. Verocel has performed certification activities to each of these software components for various customers across aerospace, industrial and medical industries.

Verocel has learned that by providing guidance on how to use and apply the software component and its certification data into a variety of applications, the integrator's certification burden and risk can be minimized. A software component approved under ISO26262 should include the following data and information so the integrator can maximize their familiarity with the component and represent it more easily when submitted for ISO26262 approval.

- ISO26262 Compliance Certificate from an approved entity (such as TüV).
- Software Safety Plan: The safety plan defines the software component and provides an overview of the compliance sought. It also initiates the software planning process and provides an overview of each lifecycle stage, the inputs, activities and outputs of each stage.
- Functional Safety Manual: this document provides a summary of the software component, its characteristics, configuration and pedigree of approval under ISO26262 (including maximum ASIL usage). Additionally, it should include a list of open problems, a list of hazards or vulnerabilities and a summary of workarounds for those problems and vulnerabilities.

- Compliance Matrix (may be part of the safety manual): This matrix shows the Part 2, 6 and part 8 objectives that the software component fulfils. The matrix summaries each requirement, the associated evidence of compliance and to what extent credit is taken for each objective. For any objectives where full credit is not taken, a summary of the required activities by the integrator should be included.
- Configuration Index or Version Description Document: Provides an unambiguous report on the exact configuration of the software component (source and binary) including configuration of any associated build tools, environment and all documentation associated with the approved component. Methods of how to ensure binary identicality of the software component in the user's environment should be provided.
- User's Guides and Manuals: If not provided as part of the safety manual, guides and manuals should be provided describing how to install, operate and used the software. Included would be description of the interfaces along with any limitations on use of those interfaces.
- Verification Results: The verification results would include information on reviews of requirements, design and code, test cases and results. Also included are the test results and coverage analysis for the software component.
- Test Vectors: The supplier should be furnished with a copy of the test vectors so they can repeat the test cases in their environment as a means to establish equivalence to the results supplied by the component developer.
- Tool Qualification Data and Results: Documents showing the tool qualification plan, tool qualification level and qualification results should be included, especially if the integrator needs to make use of these tools in their environment.
- Vulnerability Analysis or Hazard Analysis: This document would provide details on the hazards and vulnerabilities summarized in the safety manual. This information will give the integrator additional insight into the rationale behind each hazard and mitigation technique. This information will assist the developer in composing their safety analysis at both the system and hardware levels.
- Traceability Data: This information shows direct linkages between requirements, design, code, test cases and results as well as traceability to reviews of each activity including change management of each lifecycle data item.
- Partitioning Analysis (optional): A partitioning analysis may be required if the software component supports some level of ASIL separation (such as an operating environment). The partitioning analysis will show the integrator how isolation is preserved should components at a lower level of ASIL fail. The partitioning analysis will include some assumptions and perhaps limitations if there are dependencies on the hardware or a hardware-software interface
- Integration Guide: If not already included as part of the safety manual, an integration guide provides a summary of the information provided in safety manual, compliance matrix, version description document, vulnerability analysis and test vectors. Ideally, the integration guide would be the integrator's roadmap to ISO26262 using the documentation set provided by the component supplier.

Summary and Conclusions

The trends in automotive design are for ever-increasing use of software, integration of software components at various levels of safety, and regulatory compliance with ISO26262. Additionally, there is a trend for more autonomous features in road vehicles that bring about complex safety goals and requirements. Project teams seeking to integrate COTS software into their automotive will need to vet those software components closely. Beyond just the hardware compatibility, the integrator will need to show regulatory compliance with Part 6 (among others) of ISO26262. For a software component, this can be tricky because not all “certifiable” software components are equal in value and capability.

It is important to recognize that beyond an ISO26262 accreditation for a COTS component, lies an integration activity that imposes burdens on the integrator. Any software component used in an ISO26262 system should have some evidence, guidance and documentation that ensures the integrator can reuse that certification evidence without undue burden or risk. Before integrators attempt to use a COTS component, they should determine how the component supports the considerations raised here. Lack of a vulnerability analysis, safety manual, partitioning guidance as well as user’s guides and manuals will make ISO26262 approval using a COTS component more burdensome. Companies like Verocel and RTI have used their experience in other industries to align their offerings properly for the automotive designers to enable successful ISO26262 compliance.

About Verocel

Verocel (www.verocel.com) was founded in 1999 and has been dedicated to ensuring the safe behavior of software and complex hardware in multiple industries. Verocel has been a leader in defining a gold standard for software certification practices. Based on our experience, Verocel has developed a number of techniques and processes that make our work efficient and compliant with the rigors of standards such as DO-254, DO-178B/DO-178C, IEC61508, IEC62304 and ISO26262 and more. The Verocel team has experience on scores of projects from nuclear reactors, industrial control systems, flight management computers and aircraft displays, among others. Verocel’s unique skills lie in constraining COTS software and hardware for safety-critical usage and have applied these skills to components such as operating systems, communication protocols, system-on-chip microprocessors and logic devices.

About RTI

RTI provides the connectivity framework for the Industrial Internet of Things.

The RTI Connext® databus software forms the core nervous system for smart, distributed applications. RTI Connext allows devices to intelligently share information and work together as one integrated system. RTI was named "The Most influential Industrial Internet of Things Company" in 2014 by Appinions and published in Forbes.

Our customers span the breadth of the Internet of Things, including medical, energy, mining, air traffic control, trading, automotive, unmanned systems, industrial SCADA, naval systems, air and missile defense, ground stations, and science.

RTI is the world's largest embedded middleware provider, privately held and headquartered in Sunnyvale, California.