LDRA
Software Technology

Technical White Paper

ISO 26262 and Automotive SPICE®: A meeting of fire and ice?

www.ldra.com
* Registration required to download the document

Contents

Background

According to Robert Frost's famous poem [1], the disparate forces of both fire and ice each has capacity to bring about the end of the world. To many, the worlds of functional safety and software quality are likewise dissimilar notions. And yet when it comes to software development, "do it right" is a fair summary of most quality or functional safety standards. Perhaps it is no surprise that there is overlap.

Software is almost infinitely malleable, and best practice in software development is constantly evolving. Standards and reference frameworks are necessary to provide points of reference. Without them, the coordination of the multitude of development organisations in the automotive ecosystem would be almost impossible.

ISO/IEC 15504 "*Information technology – Process assessment*" [2] , otherwise known as Software Performance Improvement and Capability dEtermination (or SPICE), provides just such a common point of reference. It consists of a set of technical standards that collectively form a framework to measure the maturity of software development processes.

A domain specific version of SPICE, Automotive Software Performance Improvement and Capability dEtermination (Automotive SPICE, or colloquially ASPICE) [3] was conceived by European car manufacturers to fulfil that same purpose in the automotive sector. ASPICE remains conformant with ISO/IEC 15504's successor, the ISO330XX [4] standard series.

The ISO 26262 standard takes a slightly different perspective. By focusing on functional safety in the automotive sector, it also promotes the development of high quality software but with the specific aim of ensuring that developments are adequately safe, with due consideration of the risk involved should they fail. ISO 26262 is now a de-facto automotive functional safety standard that almost all road vehicles adhere to.

Clearly there is overlap between the scope of ASPICE and that of ISO 26262. What if there is a requirement to comply with both?

ASPICE in practice

The concept underpinning ASPICE is that the continuous and ongoing refinement of software development processes will similarly enhance the quality of the resulting code. The analyses and metrics associated with the framework are equally well suited to development organizations looking to improve their quality, and to purchasers looking to establish the credentials of their suppliers. In the latter case, the HIS scope is usually applied, which is a subset of the ASPICE framework designed for that specific purpose.

The ASPICE Process Assessment Model (PAM) is made up of process model and capability level dimensions. It can be visualized as a two dimensional matrix, where one axis describes a collection of desirable actions, and the other indicates how well and how completely those actions are performed.

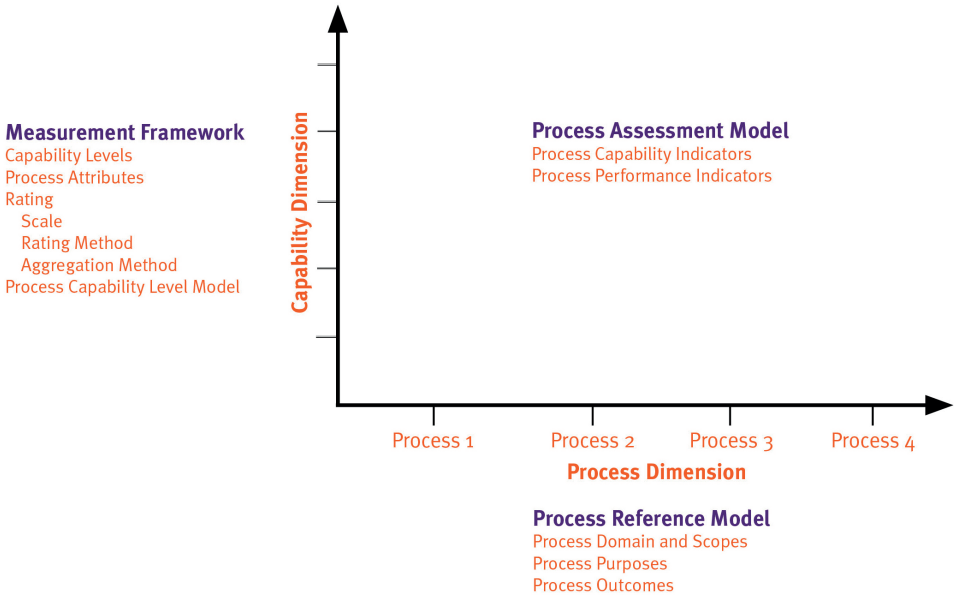

Figure 1: The ASPICE Process Assessment Model [5]

The Process Dimension axis of this PAM matrix is called the Process Reference Model. It is composed of Process Categories which include Process Groups, and these collate individual processes. Each process has a purpose and outcomes, and base practices and work products contribute to achieving one or more outcomes.

The Capability Dimension axis is called the Process Measurement Framework. It consist of Capability Levels (CL) which are further subdivided into Process Attributes (PA).

In practice, the processes themselves consist of the activities to be performed during the software development lifecycle (Figure 2).

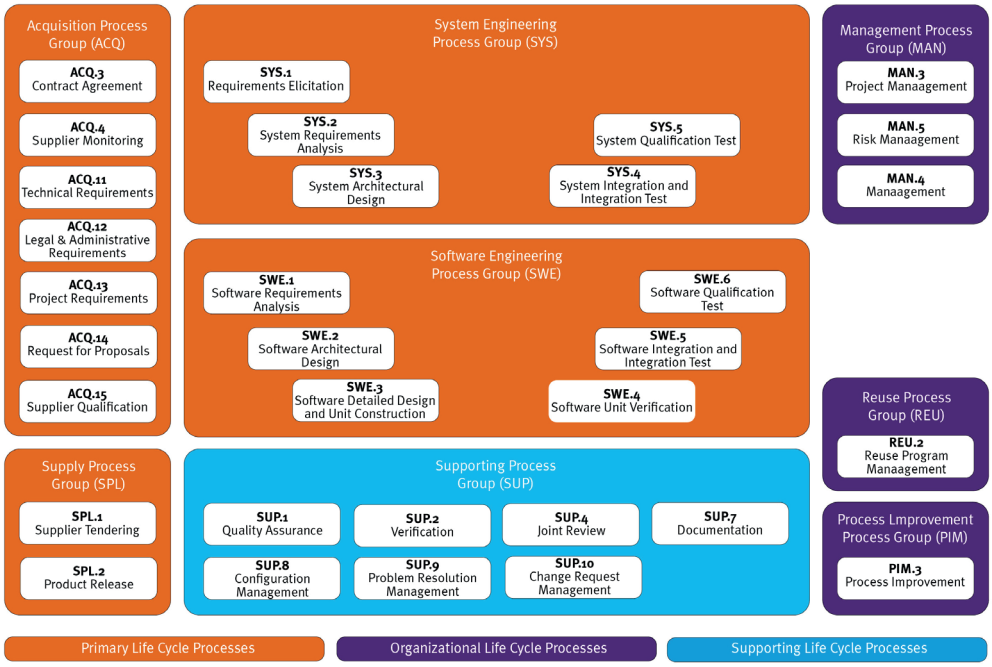

Figure 2: ASPICE process groups in context of the process reference model [6]

PAs are defined within the standard and include considerations such as performance management, work product management, process deployment, and process measurement.

An assessment of PAs involves ranking them as Not Achieved, Partially Achieved, Largely Achieved and Fully Achieved. Process capability levels are derived from these rankings, ranging from Level 0 (Incomplete process - process is not implemented, or fails to achieve its purpose) to Level 5 (Innovative process – process is continually improved to respond to organisational challenge)

The final assessment collating that information is performed using a Process Assessment Model (Figure 3).

		Level 1	Level 2	Level 3	Level 4	Level 5
PA 1.1	Process Performance	Largely	Fully	Fully	Fully	Fully
PA 2.1	Performance Management		Partially	Largely	Fully	Fully
PA 2.2	Work Force Management			Largely	Fully	Fully
PA 3.1	Process Definition			Largely	Fully	Fully
PA 3.2	Process Deployment			Fully	Fully	Fully
PA 4.1	Quantitative Analysis				Partially	Fully
PA 4.2	Quantitative Control				Partially	Largely
PA 5.1	Process Innovation					Largely
PA 5.2	Process Innovation Implementation					Largely

Figure 3: An example Process Assessment Model (PAM) [5]

ISO 26262 in practice

There is an ever-widening range of automotive electrical and/or electronic (E/E/PE) systems such as adaptive driver assistance systems, anti-lock braking systems, steering and airbags. Their increasing levels of integration and connectivity provide almost as many challenges as their proliferation, with non-critical systems such as entertainment systems sharing the same communications infrastructure as steering, braking and control systems. The net result is a necessity for exacting functional safety development processes, from requirements specification, design, implementation, integration, verification, validation, and through to configuration.

ISO 26262 "Road vehicles – Functional safety" was updated in 2018 [7], having first been published in 2011 [8] in response to this explosion in automotive E/E/PE system complexity and the associated risks to public safety. Like the rail, medical device and process industries before it, the automotive sector based their functional standard on the (largely) industry agnostic functional safety standard IEC 61508 [9] which, in turn, drew heavily from the guiding principles of the aerospace standards such as DO-178B [10] and DO-178C [11]. The net result is that proven tools are available to help with the implementation of ISO 26262 which are longer established than the standard itself.

Automotive Safety Integrity Levels (ASILs)

ISO 26262 specifies a number of hazard classifications levels known as ASILs (Automotive Safety Integrity Levels). ASILs range from A to D, where ASIL D represents the most hazardous and hence demanding level.

ASILs are assigned as properties of each individual safety function at the item level, where an item is defined as a "*system or combination of systems, to which ISO 26262 is applied, that implements a function or part of a function at the vehicle level*". The assigned ASIL for a safety function in a safety-related system is dictated by the properties of associated hazardous events.

ISO 26262 supports the decomposition of Functional Safety Requirements (FSRs) in a process often known as "ASIL decomposition" which can help to reduce cost and effort. ASIL decomposition is typically performed manually and must result in redundant safety requirements allocated to design elements of sufficient technical independence.

Figure 4: Decomposition of ASILs throughout the software item can occur over different systems, elements and components

ISO 26262 also represents software development as a V-model, which in many ways is reminiscent of the representation of the ASPICE process groups shown in Figure 2 above.

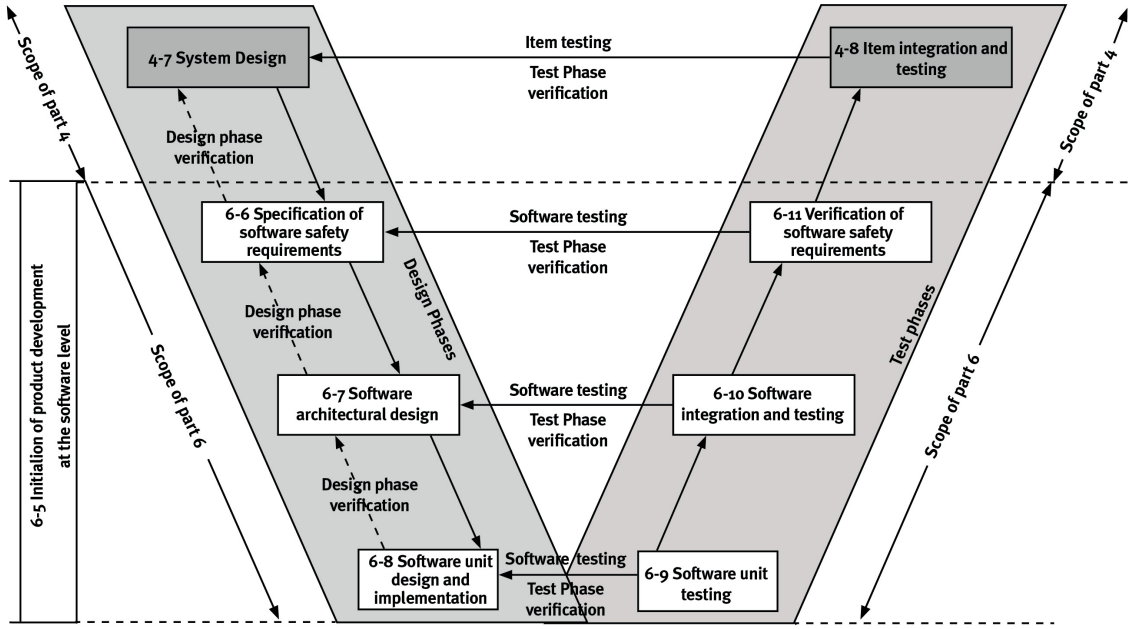

Figure 5: Image from Part 6 of ISO 26262, where the software development lifecycle is represented as a V-shaped model

The amount of effort required at each phase of this development lifecycle is dependent on the ASIL assigned to the software item under development. For example, Figure 6 shows how the ASIL affects the modelling and coding guidelines that are to be enforced.

Topics		ASIL			
		A	B	C	D
1a	Enforcement of low complexity	++	++	++	++
1b	Use of language subsets	++	++	++	++
1c	Enforcement of strong typing	++	++	++	++
1d	Use of defensive implementation techniques	+	+	++	++
1e	Use of well-trusted design principles	+	+	++	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++
1i	Concurrency aspects	+	+	+	+

"++" The method is highly recommended for this ASIL.
"+" The method is recommended for this ASIL.
"o" The method has no recommendation for or against its usage for this ASIL.
Supported by the LDRA tool suite
Verified by the LDRA tool suite

Figure 6: Mapping the capabilities of the LDRA tool suite to "Table 1: Topics to be covered by modelling and coding guidelines" specified by ISO 26262-6:2018

Bringing ISO 26262 and ASPICE together

ISO 26262 specifically references ASPICE's focus on process capability in the context of product safety, saying that "*...an organization's process definitions must address multiple standards at the same time. If a SPICE assessment is performed, then this SPICE assessment and a functional safety audit can be simultaneously performed. There is sufficient commonality in content that can help to avoid duplication of work or process between both standards and to allow synchronization of the planning*".

The overlap between the V-models of the two standards is clear to see from the figures above.

In 2017, a team of researchers from academia and industry published their analysis of the "Commonality and differences between ASPICE and ISO 26262 in the context of software development" [12]. Specifically referencing the HIS scope, the team looked to "*have a deeper understanding of the scenario where organisations have a sub-scope of ASPICE adopted (HIS scope) and wish to adopt jointly the ISO 26262 standard as a strategic approach to improving development capability*".

The mappings were performed between ASPICE outcomes and ISO 26262 "shall" statements, to assess whether the existence of the outcome would provide any evidence of the implementation of an ISO 26262 requirement. A graphical representation of the team's findings are shown in Figure 7.

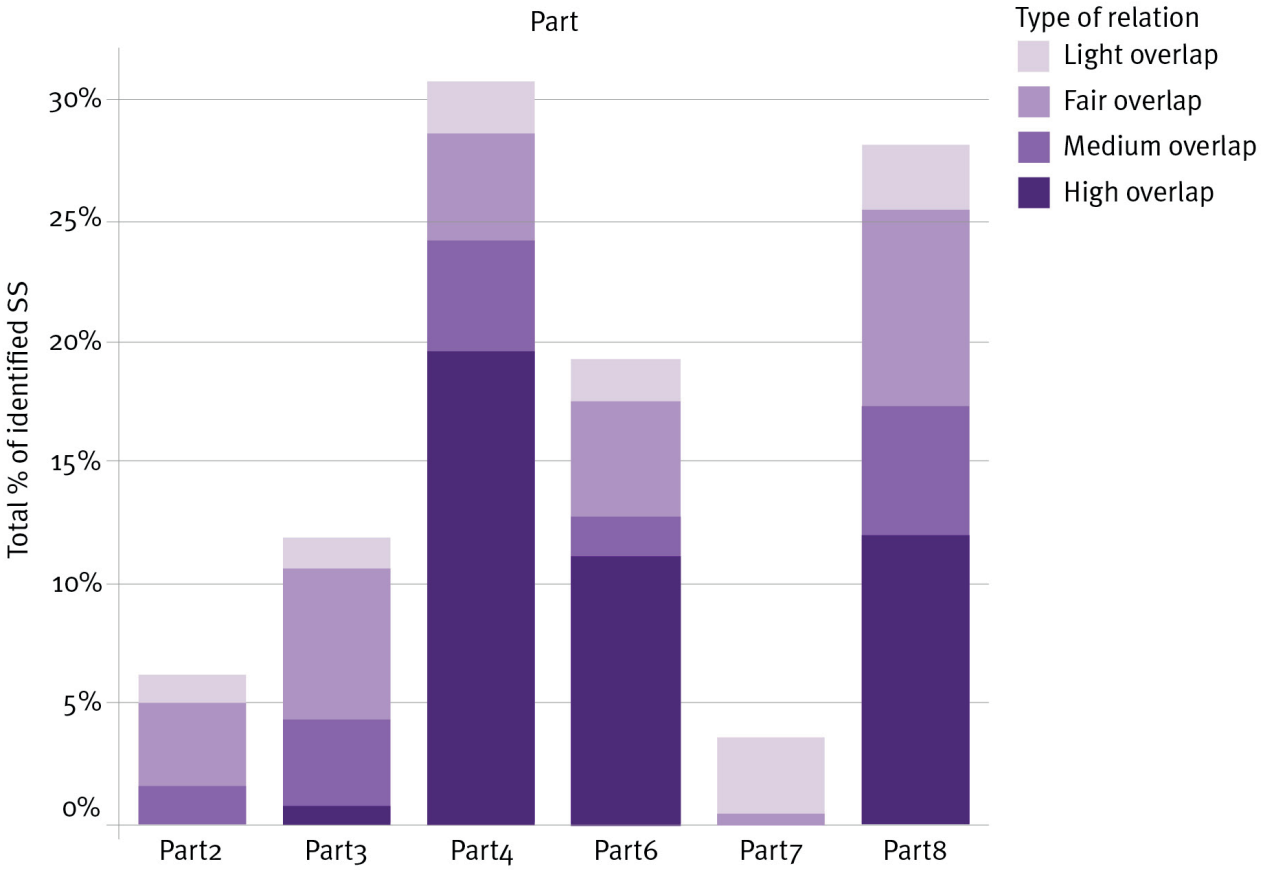

Figure 7: Overall coverage of ISO 26262 by HIS scope from ASPICE

Despite the significant overlap, there are several items referenced by ISO 26262 that are not within the scope of ASPICE HIS.

These include:

- Hazard Analysis and Risk Assessment (HARA)
- safety analysis
- the concept of Functional Safety
- the concept of Technical Safety
- safety management
- safety qualification of tools, libraries and components
- system and software safety validation

Where there is no overlap, ISO 26262 practices can extend the scope of ASPICE practices to accommodate the functional safety requirements and the ASPICE principles of continuous improvement applied to them. Where there is overlap the two V-models complement each other very well.

For example, the ASPICE System Requirement specification would also include ISO 26262 safety requirements. The verification and validation processes would also follow the methodologies as mentioned in the ISO 26262 standard, with software unit verification (SWE.4) being a typical example (Figure 8).

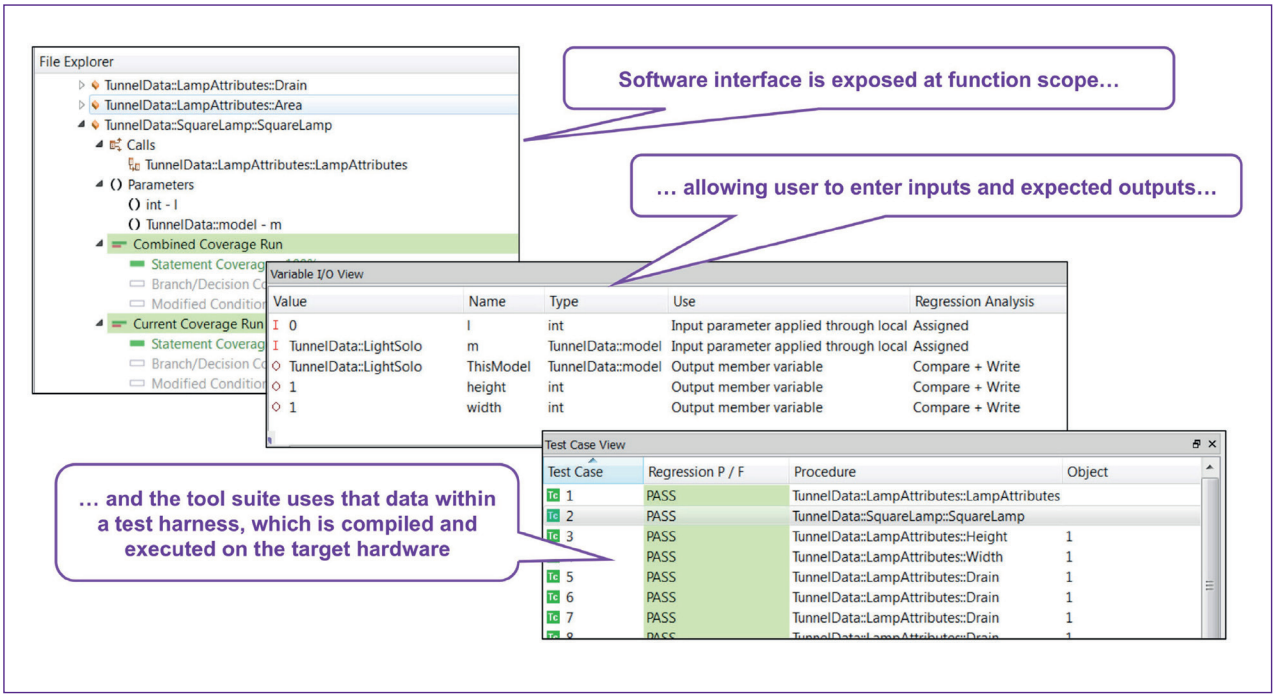

Figure 8: Performing unit test with the LDRA tool suite fulfils both ASPICE and ISO 26262 objectives

Where the overlap between the standards is not 100%, completing a superset of tasks ensures both standards are met.

Automating traceability to standards objectives

However well the objectives of the standards and the application functional requirements are established, a mechanism is required to ensure that they are reflected in the implementation of the project. Where systems developers are obliged to meet the objectives of multiple standards concurrently, it makes sense to incorporate those demands alongside the functional, functional safety, and cybersecurity requirements of the system [13].

Bidirectional traceability is referenced throughout ISO 26262. It is explicitly mentioned in part 6 paragraph 7.4.2:

"*This implies bi-directional traceability between the software architectural design and the software safety requirements.*"

More generally though, its presence is implied by the need for each development phase to accurately reflect the one before it, such as in part 6 paragraph 9.1, which requires that all requirements are fulfilled with no additional undesirable properties:

"c) to provide evidence that the implemented software unit complies with the unit design and fulfils the allocated software requirements with the required ASIL; and d) to provide sufficient evidence that the software unit contains neither undesired functionalities nor undesired properties regarding functional safety."

ASPICE is rather more explicit on the matter. Requirement SWE.1.BP6 requires compliant projects to "*Establish bidirectional traceability between system requirements and software requirements. Establish bidirectional traceability between the system architecture and software requirements*." In the context of the principle of continuous improvement, it makes very good sense for that traceability to be extended throughout the development lifecycle.

Whether iterative or waterfall models are used, process illustrations show each phase flowing into the next, perhaps with feedback to earlier phases. Traceability is assumed to be part of the relationships between phases but the mechanism by which trace links are recorded is seldom stated. The reality is that, while each individual phase may be conducted efficiently thanks to investment in up-to-date tool technology, these tools seldom contribute automatically to any traceability between the development tiers. As a result, the links between them become increasingly poorly maintained over the duration of projects, particularly when requirements change and time is short.

Figure 9: Deploying the LDRA tool suite to automate traceability to requirements, and to objectives imported from one or more standards

The Requirements Traceability Matrix (RTM) provides the solution to this problem and represents the logical extension of the required traceability between different phases. The links between phases can be ignored, or they can be acknowledged and properly managed. Either way, they are critical.

Figure 10 illustrates this alternative view of the development landscape, reflecting the importance that should be attached to the RTM. Due to this fundamental centrality, it is vital that project managers place the same priority on RTM construction and maintenance as they do on requirements management, version control, change management, modelling and testing

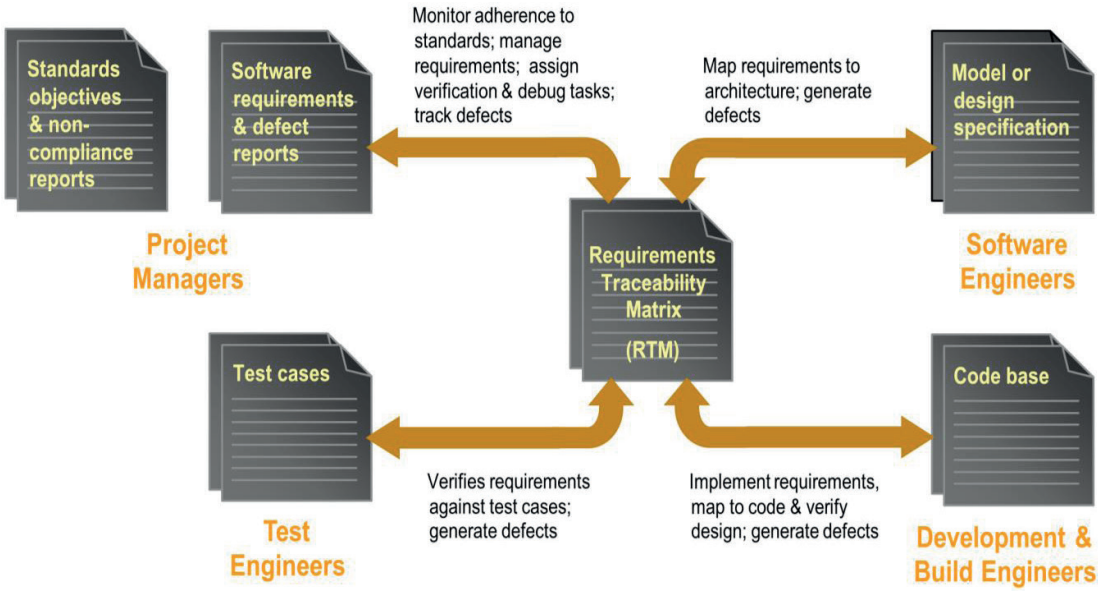

Figure 10 - RTM sits at the heart of the project defining and describing the interaction between the design, code, test and verification stages of development.

The RTM must be represented explicitly in any lifecycle model to emphasise its importance as illustrated in Figure 11. With this elevated focus, the RTM becomes the centre of the development process, impacting on all stages of design from high-level requirements through to target-based deployment.

Normally, such a model would begin at Tier 1, but here the model is extended to collate the objectives from the different standards as represented in Tier 0, and capture them in requirements management and traceability tools.

The functional safety, quality, and other requirements specified by the standards (perhaps security, for example) are collated to form Tier 1, which will also include the application specific software and system requirements.

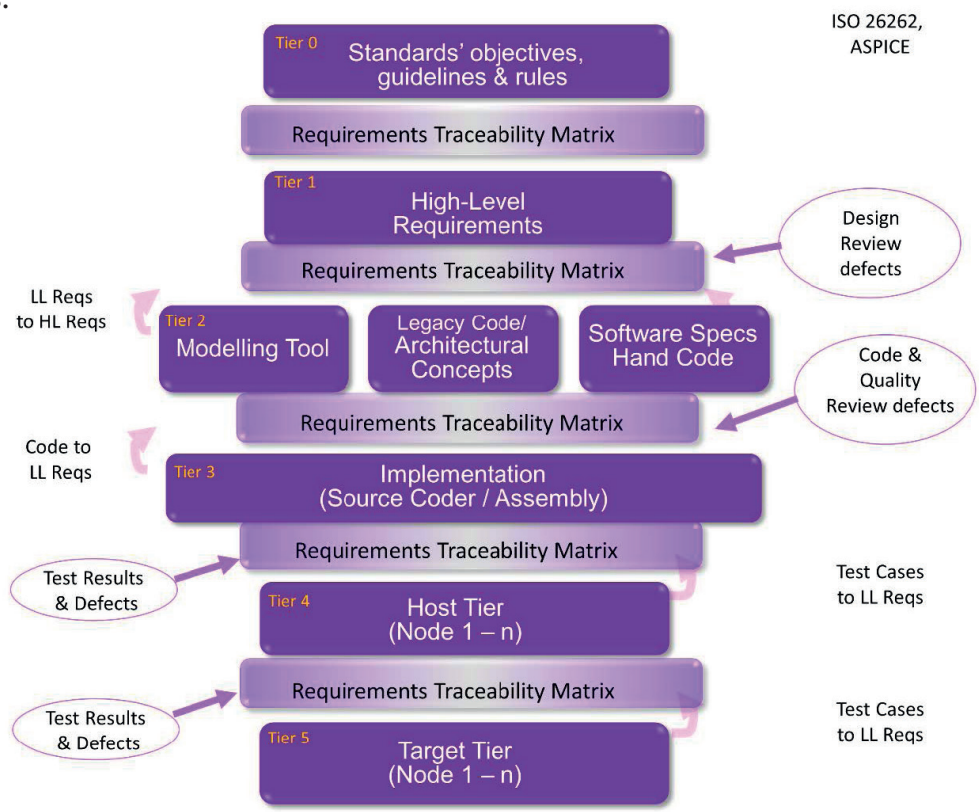

Figure 11 -The extended requirements traceability matrix (RTM) plays a central role in a development lifecycle model. Artefacts at all stages of development are linked directly to requirements matrix and changes within each phase automatically update the RTM.

These Tier 1 high-level requirements might consist of a definitive statement of the system to be developed (perhaps a ABS control module, for instance) and the functional criteria it must meet (e.g., detection of a locked wheel). This tier may be subdivided depending on the scale and complexity of the system.

Tier 2 describes the design of the system level defined by Tier 1. With the ABS example, the low-level requirements might discuss the parameters that define whether a wheel is deemed locked, building on the need to do so established in Tier 1.

Tier 3's implementation refers to the source/assembly code developed in accordance with Tier 2. In the example, it is clear that the detection of a locked wheel is likely to involve several functions. Traceability of those functions back to Tier 2 requirements includes many-to-few relationships. It is very easy to overlook one or more of these relationships in a manually managed matrix.

In Tier 4 host-based verification, formal verification begins. Using a test strategy that may be top-down, bottom-up or a combination of both, software stimulation techniques help create an automated test harnesses and test case generators as necessary. Test cases should be repeatable at on the target system at Tier 5 if required.

In the example, it would be confirmed that function calls to the software associated with the ABS return the values required of them in accordance with the requirements they are fulfilling. That information is then updated in the RTM.

Maintaining the requirements traceability matrix

The construction of an RTM is a laudable aim irrespective of whether a standard insists on it. However, maintaining an RTM in a set of spreadsheets is a logistical nightmare, fraught with the risk of error and permanently lagging the actual project status.

Constructing the RTM in a suitable tool not only maintains it automatically, but also opens up possibilities for filtering, quality checks, progress monitoring and metrics generation. The RTM is no longer a tedious, time-consuming task reluctantly carried out at the end of a project; instead it is a powerful utility which can contribute to its efficient running.

Not only does automation ease the burden of change management, it also provides the facility for impact analysis so that those promoting change can be made are fully aware of its implications. Requirements becoming usable artefacts, able to drive implementation and testing. And many of the trace links can be captured as a by-product of day-to-day development, accelerating RTM construction and improving the quality of its contents.

Modern requirements traceability solutions enable the extension of the requirements mapping down to the verification tasks associated with the source code. The screenshot below shows one such example of this. Using this type of requirements traceability tool, the 100% requirements coverage metric objective can be clearly measured, no matter how many layers of requirements, design and implementation decomposition are used. This makes monitoring system completion progress an extremely straightforward activity.

Conclusions

Developing any system that is required to be compliant with functional safety standards is challenging. But with the ever increasing complexity of automotive systems and the requirement for both connectivity and security, those challenges are of a completely different scale especially where ASPICE is also a factor.

Traditionally, evidence of adherence to standards and the fulfilment of requirements is demonstrated through the use of spreadsheets and databases, updated manually to try to keep abreast of an ever changing situation.

The delivery of a Requirements Traceability Matrix (RTM) is now often contractually imposed on suppliers. Even when not required, many development teams recognize that a RTM is an important 'best practice' for successful projects. However the creation of a useful and error-free RTM can only happen when the requirements are of sufficient quality.

ISO 26262 insists that bidirectional traceability is maintained, as does ASPICE (SWE.1.BP6) from a requirements perspective. In the context of the latter's principle of continuous improvement, it makes very good sense for that traceability to be extended throughout the development lifecycle.

In short, ISO 26262 and ASPICE are not disparate "fire and ice" at all. Perhaps knowledge and power would be a better analogy!

Works Cited

[1] R. Frost, "Fire and Ice," Poetry foundation, 2021. [Online]. Available: https://www.poetryfoundation.org/poems/44263/fire-and-ice. [Accessed 13 July 2021].

[2] International Standards Organisation, "ISO/IEC 15504-1:2004," International Organization for Standardization, November 2004. [Online]. Available: https://www.iso.org/standard/38932.html. [Accessed 13 July 2021].

[3] Automotive Special Interest Group (SIG), "The Automotive SPICE process assessment and reference model," VDA QMC, 2018. [Online]. Available: http://www.automotivespice.com/. [Accessed 13 July 2021].

[4] International Organization of Standards / International Electrotechnical Commission, "ISO/IEC 33001:2015," International Organization of Standards / International Electrotechnical Commission, 2015. [Online]. Available: https://www.iso.org/standard/54175.html. [Accessed 13 July 2021].

[5] S. Neemeh, "What is ASPICE in Automotive?," LHP, 9 June 2020. [Online]. Available: https://www.lhpes.com/blog/what-is-aspice-in-automotive. [Accessed 13 July 2021].

[6] D. J. Schlosser, "Functional safety – processes and organization," Electrobit, 2011-2021. [Online]. Available: https://www.elektrobit.com/trends/functional-safety-processes-and-organization/. [Accessed 14 July 2021].

[7] International Organization for Standardization, ISO 26262:2018 "Road vehicles - Functional safety", International Organization for Standardization, 2018.

[8] Automotive IQ, "Car safety: History and Requirements of ISO 26262," Automotive IQ, 29 June 2015. [Online]. Available: https://www.automotive-iq.com/electrics-electronics/articles/the-history-and-requirements-of-iso-26262 [Accessed 14 July 2021].

[9] International Electrotechnical Commission (IEC), IEC 61508:2010 "Functional safety of electrical/electronic/programmable electronic safety-related systems" (all parts), IEC, 2010.

[10] RTCA, DO-178B, "Software Considerations in Airborne Systems and Equipment Certification", RTCA, 1992.

[11] RTCC, DO-178C "Software Considerations in Airbourne Systems and Equipment Certification", RTCA, 2011.

[12] Oliveira P., Ferreira A.L., Dias D., Pereira T., Monteiro P., Machado R.J., "An Analysis of the Commonality and Differences Between ASPICE and ISO26262 in the Context of Software Development," Communications in Computer and Information Science, vol. 748, pp. 216 - 227, 2017.

[13] M. Pitchford, "AUTOSAR, ISO 26262, SAE J3061: So many rules, so little time!," in Embedded World, Nuremberg, 2020.

Acknowledgments

Automotive SPICE® is a registered trademark of the Verband der Automobilindustrie e.V. (VDA)

www.ldra.com

LDRA UK & Worldwide
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

LDRA Technology Inc.
2540 King Arthur Blvd, 3rd Floor, 12th Main Lewisville Texas 75056
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd.
Unit B-3, Third floor Tower B, Golden Enclave
HAL Airport Road Bengaluru 560017
Tel: +91 80 4080 8707
e-mail: india@ldra.com

ISO 26262 and ASPICE WP v2.0 08/21