From Safety Integrity Level to Assured Reliability and Resilience Level for Compositional Safety Critical Systems

Eric Verhulst[1], Jose Luis de la Vara[2], Bernhard H.C. Sputh[1], and Vincenzo de Florio[3]

[1]Altreonic NV, Linden, Belgium, [2]Simula Research Laboratory, Lysaker, Norway
[3]PATS / Universiteit Antwerpen & iMinds research institute, Antwerpen, Belgium
{eric.verhulst,bernhard.sputh}@altreonic.com, jdelavara@simula.no, vincenzo.deflorio@uantwerpen.be

Abstract:

Safety engineering standards define rigorous and controllable processes for system development. Nevertheless, safety standards differences from distinct domains are non-negligible. We focus in particular on the aviation, automotive and railway standards, all related to the transportation market. We argue that the Safety Integrity Levels are not sufficient to be used as a top level requirement for developing a safety critical system. We argue that Quality of Service is a more generic criterion that takes the trustworthiness as perceived by users into deeper account. In addition safety engineering standards provide very little guidance on how to compose safe systems from components, while this is the established engineering practice. We develop a novel normative concept called Assured Reliability and Resilience Level as a criterion that takes the industrial practice into account and show how it complements the Safety Integrity Level concept. More-over, ARRL can make a significant contribution to foster cross-domain safety engineering.

Keywords: Assured Reliability and Assurance Level, safety engineering, Safety Integrity Level, trustworthiness, Quality of Service, compositional safety.

1 Introduction

One of the emerging needs of embedded systems is better support for safety and, increasingly so, security. The underlying need is trustworthiness. To achieve such properties, systems engineering standards and in particular safety standards were developed. They aim to guarantee safety properties because they concern the risk that people are hurt or killed. Therefore certification is a legal pre-condition to put them in public use.

A first question that arises is why each domain has specific safety standards [9] as they all aim to reduce the same risk of material damages and human fatalities. One can certainly find historical reasons; but also psychological ones. Safety standards, such as IEC 61508 [1], also mainly concern systems with programmable electronic components. The reason for this is that with the advent of programmable components in system design, systems engineering became dominantly a discrete domain problem, whereas the preceding technologies were dominantly in the continuous domain. In the continuous domain components have the inherent property of graceful degradation, while this is not the case for discrete domain systems. A second specific trait is that, in the discrete domain, the state space is usually very large with state changes happening in nanoseconds. Hence it is very important to be sure that no state change can bring the system into an unsafe condition. Notwithstanding identifiable weaknesses, safety engineering standards impose a controlled engineering process resulting in relatively well predictable safety that can be certified by external parties. However, the process is expensive and essentially requires that the whole project and system is re-certified whenever a change is made. Similarly, a component such as a computer that is certified as safe to use in one domain cannot be reused as such in another domain.

This is often in contrast with the engineering practice. Engineers constantly build systems by composing and reusing existing components. This is not only driven by economic benefits, growing complexity but it often increases the trust in a system because the risk for residual errors will be lower, at least if a qualification process is in use. Nevertheless, engineering and safety standards contain very few rules and guidelines on reusing components, which hampers developing safe systems by composition. This paper analyses why the current safety driven approach is unsatisfactory. It introduces a new criterion called the Assured Reliability and Resilience Levels that allows components to be reused in a safety critical context in a normative way while preserving the safety integrity levels at the system level.

2 Discussion on the Safety Integrity level

2.1 Core concepts defining the Safety Integrity Levels

As safety is a critical property, it is no wonder that safety standards are perhaps the best examples of concrete systems engineering standards, even if safety is not the only property that is relevant for systems engineering projects. Most domains have their own safety standards partly for historical reasons, partly because the heuristic knowledge is very important, or because the practice in the domain has become normative. We take the IEC 61508 standard as an example. It considers mainly programmable electronic systems. It starts from the principle that safety is never absolute; hence it considers the likelihood of a hazard (a situation posing a safety risk) and the severity of the consequences. A third element is controllability. The combination of these three factors is used to determine a required SIL or Safety Integrity Level, categorized in 4 levels, SIL-1 being the lowest and SIL-4 being the highest. These levels correspond to normative allowed Probabilities of Failure per Hour and require corresponding Risk Reduction Factors that depend on the usage pattern (infrequent versus continuous). The risk reduction itself is achieved by a combination of reliability measures (higher quality), functional measures, as well as assurance from following a more rigorous engineering process. The safety risks are in general classified in 4 classes, roughly each corresponding with a required SIL level whereby we added a SIL-0 for completeness. Note that this can easily be extended to economic or financial risks. Note that we use the term SIL as used in 61508 while the table is meant to be domain independent.

Table 1 Categorisation of Safety Risks

Category	Typical SIL	Consequence upon failure
Catastrophic	4	Loss of multiple lives
Critical	3	Loss of a single life
Marginal	2	Major injuries to one or more persons
Negligible	1	Minor injuries at worst or material damage only
No consequence	0	No damages, except user dissatisfaction

The SIL level is used as a directive to guide selecting the required architectural support and development process requirements. For example SIL-4 imposes redundancy and positions the use of formal methods as highly recommended. While IEC 61508 has resulted in derived domain specific standards (e.g. ISO 26262 for automotive [2] and EN 50128 [3] for railway), there is no one to one mapping of the domain specific levels to IEC 61508 SIL levels. Table 2 shows an approximate mapping whereby we added the aviation DO-178C [4] standard. It must be mentioned that the Risk Reduction Factors are vastly different as well. This is mainly justified by the usage pattern of the systems and the different "fail safe" modes. For example while a train can be stopped if a failure is detected, a plane must at all cost be kept in the air in a state that allows it still to land safely.

Table 2 Approximate cross-domain mapping of Safety Integrity Levels

Domain	**Domain specific Safety Levels**				
General (IEC-61508) Programmable electronics	(SIL-0)	SIL-1	SIL-2	SIL-3	SIL-4
Automotive (26262)	ASIL-A	ASIL-B	ASIL-C	ASIL-D	-
Aviation (DO-178/254)	DAL-E	DAL-D	DAL-C	DAL-B	DAL-A
Railway (CENELEC 50126/128/129)	(SIL-0)	SIL-1	SIL-2	SIL-3	SIL-4

The SIL levels (or the domain specific ones) are mostly determined during a HARA (Hazard and Risk Analysis) executed before the development phase and updated during and after the development phase. The HARA tries to find all the hazardous situations and classifies them according to 3 main criteria: probability of occurrence, severity and controllability. This process is difficult and complex; partly because the state space explodes very fast, but also because the classification is often not based on historical data (absent for any new type of system) but on expert opinions. It is therefore questionable if the assigned Safety Levels are accurate enough and if the Risk Reduction Factors are realistic. We elaborate on this further.

Once an initial architecture has been defined, another important activity is executing a FMEA (Failure Mode Effect Analysis). While a HARA is top-down and includes environmental and operator states, the FMEA analyses the effects of a failing component on the correct functioning of the system in terms of the potential hazards. Failures can be categorized according to their origin. Random failures are typically generated by external events, whereas systematic failures are a result of either design or implementation errors. In all cases, when programmable electronics are used, their effect is often the same: the system can go immediately or in a later time interval into an unsafe state. In all cases only an adequate architecture can intercept the failures before they generate further errors and hence pose a safety risk. However, there is no criterion defined that allows us to classify components in terms of their trustworthiness, even if one can estimate parameters like MTBF (Mean Time Between Failures). In the last part of this paper we introduce a criterion that takes the fault behavior into account.

2.2 SIL calculations and non-linearity

SIL determination in the standards is often based on statistical values such as the probability of occurrence and semi-subjective estimations of the severity of the hazard and its controllability. While a human operator can often be a crucial element in avoiding a catastrophe, it is also a subjective and uncontrolled factor; hence it should be used with caution as an argument to reduce functional risk reduction efforts. In general, in any system as envisioned by IEC61508 and the related standards, we can distinguish three levels of technology, as summarized in Table 3.

Table 3 Technology levels in a system

Technology level	Dominant property	Dominant faults	Typical safety measures
Software	Discrete state-space, non-linear time	Design faults and logical errors	Redundancy and diversity at macro level, formal correctness
Electronics	Combinatorial state-space, discrete time	Transient faults	Redundancy at micro-level, formal correctness
Material	Mainly continuous or linear properties	Permanent or systemic faults	Adding robustness safety margins

We can now see more clearly why safety standards mostly focus on probabilities and quality. In the days before programmable electronics, system components were "linear", governed by material properties. One only had to apply a serious safety margin (assuming an adequate architecture). Non-linearity (i.e., discontinuities) could happen if there was a material defect or too much stress. Electronics are essentially also material devices and are designed with the same principles of robustness margins, embedded in technology-specific design rules.

With the introduction of digital logic, a combinatorial state machine was introduced and a single external event (e.g. charged particle) could induce faults. The remedy is redundancy: parity bits, CRC codes, etc. Note however that digital logic is not so linear anymore. It goes through the state machine in steps and a single faulty bit can lead to an erroneous illegal state or numerical errors. Software makes this situation even worse as now we have an exponentially growing state machine. And software is clearly non-linear. Every clock pulse the state is changed and even the execution thread can switch to another one. The remedy is formal proof (to avoid reaching undesired states) and redundancy (but with diversity).

Therefore, reasoning in terms of probabilities and quality degrees for digital electronics and software makes little sense. In the discrete domain a component is either correct or not correct. While we can reduce the probability of reaching an erroneous illegal state by, for instance, a better development process or a better architecture, the next event can result in a catastrophic outcome. This must be the starting point for developing safe systems with discrete components if one is really serious about safety. Graceful degradation does not apply to non-linear systems.

2.3 Quality of Service Levels as a more generic approach

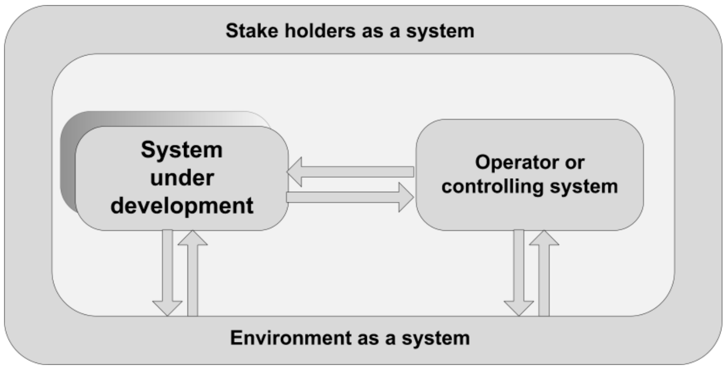

Figure 1 The context in which a system under development is used.

An inherent weakness from the systems engineering and user's point of view is that safety is not a fully integral part of the trustworthiness of a system. A system that is being developed is part of a larger system that includes the user (or operator) as well as the environment in which the system is used. Both do not necessarily interact in a predictable way with the envisioned system and have an impact on the safety properties and assurance. Note that we can also consider security risks as a subtype of safety risk, the difference being the origin of the resulting fault. From the user's point of view, the system must deliver an acceptable and predictable level of service which we call the Quality of Service (QoS). A failure in a system is not seen as an immediate safety risk but rather as a breach of contract on the QoS whereby the system's malfunction can then result in a safety related hazard. As such we can see that a given SIL is a subset of the QoS. The QoS can be seen as the availability of the system as a resource that allows the user's expectations to be met. Aiming to reduce the intrinsic ambiguities of the Safety Levels we now formulate a scale of QoS as follows:

- QoS-1 is the level whereby there is no guarantee that there will be resources to sustain the service. Hence the user should not rely on the system and should consider it as untrustworthy. When using the system, the user is taking a risk that is not predictable.
- QoS-2 is the level whereby the system must assure the availability of the resources in a statistically acceptable way. Hence, the user can trust the system but knows that the QoS will be lower from time to time. The user's risk is mostly one of annoyance and dissatisfaction or of reduced service.
- QoS-3 is the level whereby the system can always be trusted to have enough resources to deliver the highest QoS at all times. The user's risk is considered to be negligible.

We can consider QoS levels are more challenging and ambitious than the SIL levels because they define minimum service levels that must be assured in each QoS level.

2.4 Some data for thought

While risks associated with health and political conflicts are still very dominant as cause of death and injuries, technical risks like working in a factory or using a transportation system are considered more important because they have a higher emotional and visible economic cost, even if the number of fatalities is statistically low. The reason is probably because the perception is that these risks are avoidable and hence a responsible party can be identified, eventually resulting in financial liabilities.

As a result, sectors like railway and aviation are statistically very safe. As an example, about 1000 people are killed every year worldwide in aircraft related accidents, which makes aviation the safest transport mode in the world [5]. In contrast the automotive sector adds up to about 1.2 million fatalities per year worldwide and even developed regions like the USA and Europe experience about 35000 fatalities per year (figures for 2010; [6]). These figures are approximate as the statistics certainly do not include all causalities. Although both sectors have their safety standards, there is a crucial difference. Whereas in most countries aircrafts and railway systems are strictly regulated and require certification, in the automotive sector the legal norms are much weaker partly because the driver is considered as the main cause of accidents. The latter biases significantly the "controllability" factor in the required SIL determination.

Taken a closer look at the SIL classifications of IEC 61508 and the automotive derived ones in ISO 26262, we notice three significant differences:

1. Whereas IEC 61508 and ISO 26262 both define 4 levels, they do not map to each other – in particular SIL-3 and SIL-4 do not map to ASIL-C and -D.
2. The highest ASIL-D level corresponds to a SIL-3 level in terms of casualties, although it is not clear if this means a few casualties (e.g. not more than five like in a car) or several hundreds (like in an airplane.)
3. The aviation industry experiences about 1000 casualties per year world-wide, whereas the automotive industry experiences 1200 times more per year worldwide, even in developed regions 35 times, whereby the automotive highest safety level is lower.

When we try to explain these differences, we identify the following reasons:

1. ISO 2626 was defined for automotive systems that have a single central engine. As a direct consequence of the centralized and non-redundant organization such a vehicle cannot be designed to be fault-tolerant (which would require redundancy) and therefore cannot comply with SIL4.
2. While ASIL-C more or less maps onto SIL-3 (upon a fault the system should transition to a fail-safe state), ISO 26262 introduces ASIL-C requiring a supervising architecture. In combination with a degraded mode of operation (e.g. limp mode), this weaker form of redundancy can be considered as fault tolerant if no common mode failure affects both processing units [7].
3. Automotive systems are not (yet) subjected to the same stringent certification requirements as railway and aviation systems, whereby the manufacturers as well as the operating organization are legally liable, whereas in general the individual driver is often considered the responsible actor in case of an accident. Note that, when vehicles are used in a regulated working environment, the safety requirements are also more stringent, whereby the exploiting organization is potentially liable and not necessarily the operator or driver. Hence, this lesser financial impact of consumer-grade products is certainly a negative factor even if the public cost price is high as well.
4. The railway and aviation sectors are certified in conjunction with a regulated environment and infrastructure that contributes to the overall safety. Automotive vehicles are engineered with very little requirements in term of where and when they are operated and are used on a road infrastructure that is developed by external third parties.
5. One should not conclude from the above that a vehicle is hence by definition unsafe. Many accidents can be attributed to irresponsible driving behavior. It is however a bit of a contradiction that the Safety Integrity Levels for automotive are lower than those for aviation and railway if one also considers the fact that contrary to airplanes and trains, vehicle accidents happen in a very short time interval and confined spaces with almost no controllability by the driver.

2.5 The weaknesses in the application Safety Integrity Levels

As we have seen above, the use of the safety Integrity Levels does not result in univocal safety. We can identify several weaknesses:

1. A SIL level is a system property derived from a prescribed process whereas systems engineering is a mixture of planning, prescribed processes and architecting/developing. As such a SIL level is not a normative property as it is unique for each system.
2. SIL levels are the result of probabilities and estimations, while often no analytical historical data is present to justify the numbers. Also here we see a difference between the automotive domain and the aviation and railway domain. The latter require official reporting of any accident, have periodic and continuous maintenance schedules (even during operational use), and the accidents are extensively analyzed and made available to the community. Black boxes are a requirement to allow post-mortem analysis.

3. SIL levels, defined as a system level property, offer little guidance for reusing and selecting components and sub-system modules whereas engineering is inherently a process whereby components are reused. One exception is the ISO 13849 machinery standard and its derivates. Also the aviation sector has developed a specific standardised IMA architecture described in the D0-197 standard that fosters reuse of modular avionics, mainly for the electronic on board processing [14]. ISO 26226 also introduced the notion of a reusable component called SEooC (Safety Element out of Context) allowing a kind of pre-qualification of components when used in a well-specified context. While we see merging nations of reuse in the standards, in general very little guidance is offered on how to achieve a given SIL level by composing different components.
4. An increasing part of safety critical systems contain software. Software as such has no reliability measures, only residual errors while its size and complexity is growing very fast, despite efforts in partitioning and layering approaches that rather hide than address the real complexity. This growth is not matched by an equal increase in controllability or productivity [8]. If one of the erroneous (but unknown) states is reached (due to a real program error or due to an external hardware disturbance) this can result in a safety risk. Such transitions to an erroneous state cannot be accurately estimated up front during a SIL determination
5. The SIL level has to be seen as the top level safety requirement of a system. In each application domain different probabilistic goals (in terms of risk reduction) are applied with an additional distinction between intermittent and continuous operation. Hence cross-domain reuse or certification can be very difficult, because the top level SIL requirements are different, even if part of the certification activities can be reused.
6. A major weakness of the SIL is however that it is based on average statistical values. Not only are correct figures very hard or even impossible to obtain, they also depend on several factors such as usage pattern, the operating environment, and the skills and training of the human operator. Correct statistical values such as the mean value assume a large enough sampling base, which is often not present. Moreover it ignores that disruptive events like a very unlikely accident can totally change these values. As an example we cite the Concorde airplane that was deemed as the safest aircraft in the world until the one fatally crashed. After the catastrophic event it “became” almost instantly one of the most unsafe airplanes in the world.

The last observation is crucial. While statistical values and estimations are very good and essential design parameters, very low residual risks can still have a very high probability of happening. We call this the Law of Murphy: if anything can happen, eventually it will happen. No lives will be saved by referring to their low statistical probability. The estimated probability can be very different from the one observed after the facts.

3 ARRL: Assured Reliability and Resilience Level

3.1 A normative criterion for trustworthiness

Despite the weaknesses of the SIL criterion, safety standards are still amongst the best of the available engineering standards and practices in use. In addition, those standards contain many hints as of how to address safety risks, though not always in an outspoken way.

As an example, every standard outlines safety pre-conditions. The first one is the presence of a safety culture. Another essential principle in safety engineering is to avoid any unnecessary complexity. In formal terms: keeping the project’s and system’s state space under control. A further principle is that quality, read reliability, comes before safety otherwise any safety measure becomes unpredictable. This is reflected in the requirements for traceability and configuration management. Traceability and configuration management are only really possible if the system is developed using principles of orthogonal compensability, hence we need modular architectures whereby components are (re-)used that carry a trustworthiness label. In addition, in practice many components are developed independently of the future application domain (with the exception of for instance normative parameters for the environmental conditions). The conclusion is clear: we need guidance on how to develop components in a way that allows us reusing them with no negative impact on safety at the system level.

In the attempt to deal with the shortcomings of SIL in what follows we introduce the ARRL or Assured Reliability and Resilience Level. The different ARRL classes are defined in Table 3. They are mainly differentiated in terms of how much assurance they provide in working correctly in the presence of faults.

Table 4 Proposed ARRL Level definitions

ARRL level	ARRL definition
ARRL-0	The component might work ("use as is"), but there is no assurance. Hence all risks are with the user.
ARRL-1	The component works "as tested". No assurance is provided for the absence of any remaining issues.
ARRL-2	The component works correctly, if no fault occurs. This means that it is guaranteed that the component has no implementation errors, which requires formal evidence as testing can only uncover testable cases. The component still provides ARRL-1 level assurance by testing as also formal evidence does not necessarily provide complete coverage but should uncover all so-called systematic faults. The component can still fail due to random errors (example: a bit flip).
ARRL-3	The component inherits all properties of the ARRL-2 level and in addition is guaranteed to reach a fail-safe or reduced operational mode upon a fault. This requires monitoring support and architectural redundancy. The fault behavior is predictable as well as the subsequent state after a fault occurs.
ARRL-4	The component inherits all properties of the ARRL-3 level and can tolerate one major fault. This corresponds to requiring a fault-tolerant design. This entails that the fault behavior is predictable and transparent to the external world. Transient faults are masked out.
ARRL-5	The component inherits all properties of the ARRL-4 level but is using heterogeneous sub-components to handle residual common mode failures.

Before we elaborate on the ARRL criterion, we should mention that there is an implicit assumption about a system's architecture. A system is composed by defining a set of interacting components. This has two important consequences:

1. The component must be designed to prevent the propagation of errors. Therefore the interfaces must be clearly identifiable and designed with "guards". These interfaces must also be the only way a component can interact with other components.
2. The interaction mechanism, for example a network connection, must carry at least the same ARRL credentials as the components it interconnects. Actually, in many cases, the ARLL level must be higher if one needs to maintain a sufficiently high ARRL level at the compositional level.
3. Hence, it is better to consider the interface as a component on itself, rather than for example assuming an implicit communication between the components.

Note that when a component and its connected interfaces meet the required ARRL level, this is a necessary, not a sufficient condition for the system to meet a given ARRL and SIL level. The application itself developed on top of the component and its interfaces must also be developed to meet the corresponding ARRL level.

3.2 Discussion of the ARRL levels

By formalizing the ARRL levels, we make a few essential properties explicit:

- The component must carry evidence that it meets the requirements. Hence the use of the "Assured" qualifier. Without evidence, no verifiable assurance is possible.
- "Reliability" is used to indicate the quality of the component. A high reliability implies that the MTBF will be low and is hence not a major issue in using the component.
- "Resilience" is used to indicate the capability of the component to continue to provide its intended functionality in the presence of faults.
- There is no mentioning of safety or security levels because these are system level properties that also include the application specific functionality.
- The ARRL criterion can be applied in a normative way, independently of the application domain.

- By this formalization we also notice that the majority of the components (software or electronic ones) on the market will only meet ARRL-1 (when tested and a test report is produced). ARRL-2 assumes the use of formal techniques and very few software meet these requirements. From ARRL-3 on, a software component has to include additional functionality that deals with error detection and isolation and requires a software-hardware co-design. With ARRL-4 the system's architecture is enhanced by explicitly adding redundancy and whereby it is assumed that the faults are independent in each redundant channel. In software, this corresponds to the adoption of design redundancy mechanisms so as to reduce the chance of correlated failures.

ARRL-5 further requires 3 quasi-independent developments on different hardware, because ARRL-4 only covers a subset of the common mode failures. Less visible aspects are for instance common misunderstanding of requirements, translation tool errors and time dependent faults. The latter require asynchronous operation of the components and diversity using a heterogeneous architecture.

3.3 Rules of composition for ARLL components

A major advantage of the ARRL criterion is that we can now define a simple rule for composing safety critical systems. We use here an approximate mapping to the different SIL definitions by taking into account the recommended architecture for reaching a certain SIL level.

"A system can only reach a certain SIL level if all its components are at least of the same ARRL level."

The following side-conditions apply:

- The composition rule defines a necessary, not a sufficient condition. Application specific layers must also meet the ARRL criterion.
- ARRL-4 components can be composed out of ARRL-3 components using redundancy. This requires an additional ARRL-4 voter.
- ARRL-3 component can be composed using ARRL-2 components (using at least 2 whereby the second instance acts as a monitor).
- All interfaces and interactions also need to have the same ARRL level
- Error propagation is to be prevented. Hence a partitioning architecture (using a distributed hardware and concurrent software architecture) is a must.
- ARRL-5 requires an assessment of the certification of independent development and, when applied to software components, a certified absence of correlated errors.

3.4 SIL and ARRL are complementary

The ARRL level criterion is not a replacement for the SIL level criterion. It is complementary in the same sense that the HARA and FMEA are complementary (Figure 2). The HARA is applied top-down whereby the system is considered in its environment including the possible interactions of a user or operator. The goal of the HARA is to find the situations whereby a hazard can result in a safety risk. The outcome is essentially a number of safety measures that must be part of the system design without necessarily prescribing how these are to be implemented.

The FMEA takes a complementary approach after an implementation architecture has been selected. FMEA aims at identifying the faults that are likely to result in errors ultimately resulting in a system failure whereby a safety risk can be encountered. Hence the HARA and FMEA meet in the middle confirming their findings.

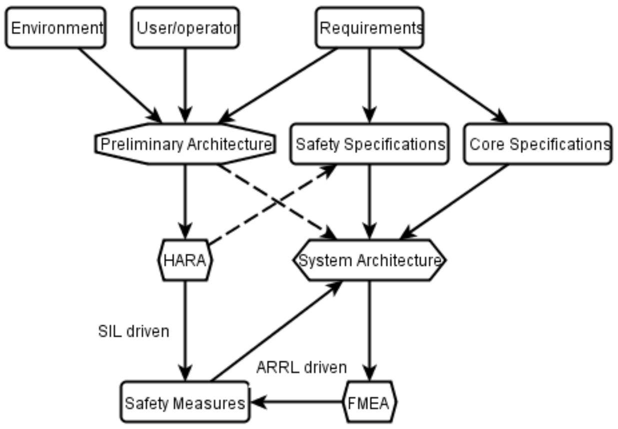

Figure 2 HARA and FMEA correspondence with SIL and ARRL

By introducing the ARRL criterion we take a new step towards making the process more normative and generic – of course still a tentative step because it will require validation in real cases. The SIL is a top level requirement decomposed in normal case requirements (ARRL-1 and -2) and fault case requirements (ARRL-3, -4, -5). From a functional point of view, all ARRL levels provide the same functionality but with different degrees of assurance and hence trustworthiness. Concretely, different ARRL do not modify the functional requirements and specifications of the components. The normative ARRL requirements result in additional functional specifications and corresponding functional support that assures that faults do not result in the core functional specifications to be jeopardized. The non-functional specifications will be impacted as well. For example, the additional functionality will require more resources (e.g. memory, energy and CPU cycles) and is likely to increase the cost price of the system. However, it provides a way to reuse components with lesser efforts across domains or within a product family. For example a computer module (specified for compatible environmental conditions) can be reused between different domains. The same applies to software components. However, this requires that the components are more completely specified than it is now often the case. ARRL level components carry a contract and the evidence that they will meet this contract given a specific set of fault conditions. Note that when using formal methods, each of these ARRL levels also requires different formal models. The higher level ARRL models must model the fault behavior in conjunction with the normal behavior, just like invariants are part of the formal models. By defining a composition rule of ARRL components to achieve a certain level of safety, we also define now safety in a quasi-domain independent way, simplifying the safety engineering process. Note however that any safety critical system still has an application specific part that must be developed to meet the same level of ARRL to reach the required SIL.

3.5 The role of formal methods

In recent years, formal methods have been gaining attention. This is partly driven by the fact (and awareness) that testing and verification can never provide complete coverage of all possible errors, in particular for discrete systems and specifically for software. This is problematic because safety and security issues often concern so-called “corner cases” that do not manifest themselves very often. Formal methods have the potential to cover all cases either by using formal models checkers or by formal proving, both with a sound mathematical base.

Practice has shown that using formal methods can greatly increase the trustworthiness of a system or component. Often it will lead to the discovery of logical errors and incomplete assumptions about the system. Another benefit of using formal methods during the design phase is that it helps in finding cleaner, more orthogonal architectures that have the benefit of less complexity and hence provide a higher level of trustworthiness as well as efficiency [13]. One can therefore be tempted to say that formal methods not only provide correctness (in the sense of the ARRL-2 criterion) but also assist in finding more efficient solutions.

Formal methods are however not sufficient and are certainly not a replacement for testing and verification. Formal methods imply the development of a (more abstract) model and also this model cannot cover all aspects of the system. It might even be incomplete or wrong if based on wrong assumptions (e.g. on how to interpret the system’s requirements). Formal methods also suffer from complexity barriers, typically manifested as a state space explosion that makes their use impractical. The latter however is a strong argument for developing a composable architecture that is using small but well proven trustworthy components as advocated by the ARRL criterion. At the same time, the ARRL criterion shows that formal models must also model the additional functionality that each ARRL level requires. This is in line with what John Rushby puts forward [11] whereby he outlines a formally driven methodology for a safe reuse of components by taking the environment into account.

The other element is that practice has shown that developing a trustworthy system also requires a well managed engineering process whereby the human factor plays a crucial role [10]. Moreover, processes driven by short iteration cycles whereby each cycle ends with a validation or (partial) integration have proven to be more cost-efficient as well as more trustworthy with less residual issues. Formal methods are therefore not something like a miracle cure. Their use is part of a larger process that aims at reaching trustworthiness. In the context of the ARRL criterion they increase the assurance level considerably because of the completeness of the verification – a goal that is only marginally reachable by only testing.

4 Conclusion

This paper analyzed the concept of Safety Integrity Level (SIL) and put it in a wider perspective of Quality of Service and Trustworthiness. These concepts are more generic and express the top level requirements of a system in the perspective of a prospective user. We have discussed some weaknesses in the SIL concept, mainly its probabilistic system specific nature whereas engineering is often based on composition using components of sub-systems. A new concept called ARRL (Assured Reliability and Resilience Level) was introduced defining a normative criterion for components and their interactions.

As future work, the concept will further be validated and applied in the context of safety critical applications. Several aspects like refinement, completeness, specifications as a contract and evidence and the impact on the system engineering process needs further study. The ARRL criterion, being normative, is a very promising approach to achieve composable safety across different domains and systems in a product family.

5 References

[1] http://www.iec.ch/functionalsafety/
[2] http://www.iso.org
[3] http://www.cenelec.eu
[4] http://www.rtca.org
[5] http://www.baaa-acro.com/. Aircraft Crashes Record Office (ACRO).
[6] "Global status report on road safety 2013", available from http://www.who.int/iris/bitstream/10665/78256/1/9789241564564_eng.pdf
[7] L.R. Goel, Vinay Kumar Tyagi, A two unit series system with correlated failures and repairs, Microelectronics Reliability, Volume 33, Issue 14, November 1993, Pages 2165-2169.
[8] http://www.pats.ua.ac.be/content/publications/2008/adidrds.pdf. On the Requirements of New Software Development. Vincenzo De Florio & Chris Blondia. International Journal of Business Intelligence and Data Mining, Vol. 3, No. 3, Inderscience, 2008.
[9] A cross-domain comparison of software development assurance standards. E. Ledinot, e.a. ERTS2012-, Toulouse.
[10] Nancy G. Levenson. Engineering a safer world. MIT Press. 2011.
[11] Rushby 2012] John Rushby, Formal Aspects of Component Software, Lecture Notes in Computer Science, Composing Safe Systems, Springer
[13] E. Verhulst, R.T. Boute, J.M.S. Faria, B.H.C. Sputh, and V. Mezhuyev. For- mal Development of a Network-Centric RTOS. Software Engineering for Reliable Embedded Systems. Springer, Amsterdam Netherlands, 2011.
[14] RTCA DO-297/EUROCAE ED-124 Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations.