

Towards ARRL-7: safer vehicles for resilient Mobility as a Service

Eric Verhulst, CEO/CTO

Altreonic NV

Content

- Safety engineering and Safety Integrity Levels (SIL)
- Some issues with the SIL criterion
- Introducing the normative ARRL criterion
- Illustrated architectures
- ARRL and antifragility
- Autonomous traffic and ARRL-7
- Conclusions
- Note: Work In Progress!

Personal experience as input

- How to develop a processor that lasts 100 years?
 - What is resilience?
 - => a system that lasts 100 years
- OPENCOSS (FP7)
 - Cross-domain safety certification reuse if almost impossible
- GoedelWorks:
 - Unified meta-model for systems engineering
 - Dependency tree is very large: when to stop?
- Autonomous systems going wrong:
 - Uber accident
 - Boeing-737 Max^{Altreonic From Deep Space to Deep Sea}

Systems Engineering vs. Safety Engineering

- System = holistic
- Real goal is "Trustworthy Systems"
 - Cfr. Felix Baumgartner almost did not do it because he didn't trust his safe jumpsuit
- TRUST = by the user or stakeholders
 - Achieving intended Functionality
 - Safety & Security & Usability & Privacy
 - Meeting non-functional objectives
 - Cost, energy, volume, maintainability, scalability, Manufacturability,..
- So why this focus on safety?
- User expects guaranteed "QoS" from a "Trustworthy system"

Safety and certification

- Safety can be defined to be the control of recognized hazards to achieve an acceptable level of risk.
 - Safety is general property of a system, not 100% assured
 - It is complex but there are moral liabilities
- Certification: In depth review => safe to operate
 - "Conformity assessment" (for automotive)
 - Not a technical requirement: confidence, legal
- Evidence makes the difference:
 - Evidence is a coherent collection of information that relying on a number of process artifacts linked together by their dependencies and sufficient structured arguments provides an acceptable proof that a specific system goal has been reached.

Categorisation of Safety Risks

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

(A)SIL [≅] f (probability of occurrence, severity, controllability)

- As determined by HARA
- SIL goals ≅ Risk Reduction Factor
- Criteria and classification are open to interpretation

Problems with SIL definition

- Poor harmonization of definition across the different standards bodies which utilize SIL=> Reuse?
- Process-oriented metrics for derivation of SIL
- SIL level determines architecture (system specific)
- Estimation of SIL based on reliability estimates
 - System complexity, particularly in software systems, makes SIL estimation difficult if not impossible
 - Based on probabilities that are very hard if not impossible to measure and estimate
 - Reliability of software (discrete domain) is not statistical!:
 - The law of Murphy still applies:
 - The next instant can be catastrophic

ARRL: what does it mean?

- Assured:
 - There is verified, trustworthy evidence
 - Process related and architecture related

• Reliability:

In absence of faults, MTBF is >> life-time: <u>QA aspects</u>

• Resilience:

- The fault behaviour is predicted: trustworthy behaviour
- Capability to continue to provide core function
- Level: ARRL is normative
 - Components can be classified: contract

New definition: start from the component up

• ARRL: Assured Reliability and Resilience Level

ARRL 0	It might work (use as is)
ARRL 1	Works as tested, but no guarantee
ARRL 2	Works correctly, IF no fault occurs, guaranteed no errors in implementation) => formal evidence
ARRL 3	ARRL 2 + goes to fail-safe or reduced operational mode upon fault (requires monitoring + redundancy) - fault behavior is predictable as well as next state
ARRL 4	ARRL 3 + tolerates one major failure and is fault tolerant (fault behavior predictable and transparent for the external world). Transient faults are masked out
ARRL 5	The component is using heterogeneous sub-components to handle residual common mode failures

Consequences

- If a system/component has a fault, it drops into a degraded mode => lower ARRL
 - ARRL3 is the operational mode after an ARRL4 failure
 - Functionality is preserved
 - Assurance level is lowered
- SIL not affected and domain independent
 - System + environment + operator defines SIL
- ARRL is a **normative criterion**:
 - Fault behavior is made explicit: verifiable
 - Cfr. IP-norm (comes with a predefined test procedure)







SIL and ARRL are complementary



A system is never alone



What means "anti-fragile"?

- New term quoted by Taleb
- An anti-fragile system gets "better" after being exposed to "stressors"
 - Better: we need a metric => QoS?
 - Stressors: cfr. hazard, faults, ...
 - The issue in safety: rare events (improbable a priori, certain post factum) (Taleb's "black swan"
- What does it mean in the context of safety/systems engineering? Isn't ARRL-5 the top level?
- Anti-fragile = <u>improving resilience by learning</u>

Two example domains

• Automotive:

- 1,2 millon people killed/year: daily event
- Cars get better, but people get killed: safer? QoS?

• Aviation:

- 500 people killed/year: a rare event
- Planes get better, cheaper, safer, energy-efficient
- Railway, telecommunications, medical, ...
 - Similar examples
- What sets them apart?

Assessment in terms of ARRL

• Automotive:

- Vehicle is a **ARRL-3 system**
- Upon fault, presumed to go the fail-safe state
- No (or small) black box, few records, ...
- Automotive is a collection of vehicles

• Aviation:

- Planes are ARRL-5
- Upon fault, redundancy takes over
- Black box, central database,
- Preventive maintenance
- Aviation is a service oriented eco-system

Extended systems (of systems) view



Preconditions for anti-fragility

- Extensive domain knowledge: experience
- Openness: shared critical information
- Feedback loops at several levels between large number of stakeholders
- Independent supervision: guidance
- Core components are ARRL-4 or -5
- The system is the domain
- Service matters more than the component

ARRL-6 and ARRL-7 (inherits ARRL-5)

ARRL 3	ARRL 2 + goes to fail-safe or reduced operational mode upon fault (requires monitoring + redundancy) - fault behavior is predictable as well as next state
ARRL 4	ARRL 3 + tolerates one major failure and is fault tolerant (fault behavior predictable and transparent for the external world). Transient faults are masked out
ARRL 5	The component is using heterogeneous sub-components to handle residual common mode failures
ARRL 6	The component (subsystem) is monitored and a process is in place that maintains the system's functionality
ARRL 7	The component (subsystem) is part of a system of systems and a process is in place that includes continuous monitoring and improvement supervised by an independent regulatory body

Autonomous traffic

- Self-driving cars are the future? (or the goal?)
- Systems engineering challenge much higher than for flying airplanes (100 msec vs 2 min)
- Huge impact: socio-economic "black swan"
- Pre-conditions:
 - Vehicles become ARRL-5
 - System = traffic, includes road infrastructure
 - Standardisation (vehicles communicate)
 - Continuous improvement process
- Hence: needs ARRL-7

Some philosophy: beyond ARRL-7

- Not all systems are engineered by humans
- Biological systems:
 - Survivability (selection) and adaption
 - Build-in mechanism (very long term feedback loops)
 - ARRL-8 ?
 - Inheritance of ARRL-7 ?
- Genetic engineering:
 - Directed selection and adaptation
 - ARRL-9? Or ARRL-7 with bio-components?

Conclusions

- ARRL concept allows compositional safety engineering with reuse of components/subsystems
- More complex systems can be safer if they are designed for resilience
- A unified ARRL aware process pattern can unify systems and safety engineering standards
- ARRL-6 and ARRL-7 introduce systems that include a feedback loop process during development but also during operation
- Maximise feedback = OPENESS

More info:

www.altreonic.com

Further work

- Making ARRL normative and applicable
 - Refinement and Completeness of criteria
 - Normative: components carry contract and evidence
 - Independent of final use or application domain
 - Process evidence + validated properties
 - ARRL-3 and higher: HW/SW co-design?
 - Study link with a system's critical states
 - Apply it on real cases
- Input and feedback welcome