ARRL:
A Criterion For Composable Safety and Systems Engineering

Eric Verhulst, Bernhard Sputh, Altreonic NV

Jose Luis de la Vara, Simula Research Lab

Vincenzo De Florio, University Antwerp

Content

- Safety Integrity Levels
- Some issues with the SIL criterion
- QoS and Trustworthiness
- Technology domains in a system
- Introducing the normative ARRL criterion
- Role of formal models
- An ARRL aware process pattern
- Conclusions
- Note: Work In Progress!

Some background projects

- ASIL project
  - Project with Flanders Drive to develop a common "automotive" safety engineering methodology
  - IEC-61508, IEC-62061, **ISO-26262**, ISO-13849, ISO-25119 and ISO-15998. (+ CMMI, Automotive SPICE)
  - About 350 steps, 100 workproducts, ...
  - ASIL imported in GoedelWorks portal
- EU FP7 IP OPENCOSS
  - Project with 17 EU partners (avionics, raillway, automotive) on reducing the cost and effort of certification
    - ISO-26262, DO-178C/254/..., CENELEC 50126-128-129
    - Cross-domain
    - Product families
- => there is interest and a growing awareness

- LinkedIn discussion grops

Some data for thought

- 1.2 million people killed in cars worldwide
  - 35000 people killed in cars /yr /Europe/US
- 1000 people killed in airplanes /yr /worldwide
  - Why the difference? => many reasons
- The Renault Logan is the most reliable car
  - Why? Less electronics, proven in use design
  - Is it also safer?
- The US is considering to make black boxes a legal requirement in cars
  - What does this mean? What could be the impact?

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”

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
  - It is complex but there are moral liabilities
  - It is not 100% water-tight
- 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.

Safety Integrity Levels (acc. IEC-61508)

Category	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
Negliglible	1	Minor injuries at worst or material damage
No consequence	0	No damages, except user dissatisfaction

- SIL $\cong$ f (**probability of occurrence, severity, controllability**)
  - As determined by HARA
- Criteria and classification are open to interpretation

Safety as a goal across domains

Domain					
General (IEC-61508) Programmable electronics	(SIL0)	SIL1	SIL2	SIL3	SIL4
Automotive (26262)	ASIL-A	ASIL-B	ASIL-C	ASIL-D	-
Avionics (DO-178/254)	DAL-E	DAL-D	DAL-C	DAL-B	DAL-A
Railway (CENELEC 50126/128/129)	(SIL0)	SIL1	SIL2	SIL3	SIL4

Risk reduction factors depend on domain and usage pattern!

Detailed analysis reveals **only partial mapping!**

Problems with SIL definition

- Poor harmonization of definition across the different standards bodies which utilize SIL
- 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
- Risk figures are different for each domain => reuse?
- The **law of Murphy still applies**:
  - The next instant can be catastrophic

The real issue with SIL

- **SIL is a system level concept**
  - But we design using components and reuse
- **SIL cannot be reused**
  - But components can!
  - Engineers always reuse existing components
- **SIL is domain specific**
  - Components are domain independent
- **Composability unclear** issue. Why?
- => we must start at the component level

Safety composibility

- Although this is the practice in systems engineering, it is poorly addressed in the standards
- Example from ISO-DIS-25119 (derived ISO 13849)
  - "The safety-related parts of control systems shall be designed in accordance with the requirements of one or more of the 5 categories specified in ISO-DIS-25119-2:2008-Annex A. When a safety function is realized by an integrated combination of multiple hardware categories, the resulting safety function AgPL is limited by the overall hardware category, including MTTFdc, DC, SRL, CCF, etc."
  - If the embedded software has to implement software components with different SRLs or safety-related and non-safety related software components, then the overall SRL is limited to the component with the lowest SRL, unless adequate independence between the software components can be demonstrated

Redefining SIL to be domain independent ?

- **S** = Safety
  - (includes security)
- **I**ntegrity **L**evel
- Attempt at one SIL definition for all domains
- **If we would follow the standards in spirit**

SIL 0	no impact
SIL 1	material damage
SIL2	harmful impact on people
SIL3	one person dead
SIL4	many persons dead

What's wrong with this definition?

- Even a perfectly designed and proven system composed of perfect components can fail catastrophically
  - Concorde: 100% safe until the first one crashed due to an improbable external cause
- The **law of Murphy has priority** over probability over a life time: non-linearities!
- SIL is a lifetime (average) indicator for a system, not a criterium for selecting/developing components

Technology levels in a system

Technology level	Dominant property	Dominant fault types	Typical safety measures
Environment	External constraints	Unforeseen interactions	Co-design of infrastructure and system
Operator/user	Human interaction	Human-Machine interface confusion	Analaysis of HMI and testing
Software	Discrete state space, non linear time	Design faults, logical errors	Redundancy and diversity at macro-level, formal correctness
Electronics	Combinatroial state-space, discrete time	Transient faults	Redundancy at micro-level
Material	Mainly continuous or linear properties	Permanent or systemic faults	Robustness safety margins

Safety/robustness margins of linear domains do not apply in the **non-linear domains**: transitions are not statistical

Why is ASIL-D ≠ SIL4 ?

- More or less: ASIL-C = SIL3
  - Switch to fail-safe mode (which is?)
- More or less: ASIL-D = SIL 3.5
  - Supervised operation, best effort upon fault
  - Whereas SIL4 implies fault tolerance and continued operation
- According to generic SIL table =>
  - car has upto 5 passengers => SIL 3.5
  - but 35000 people get killed per year (EU) => SIL4
    - System is not the car but car based transport
    - Car is component in the larger system

New definition: we start from the component

- **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: what does it mean?

- Assured:
  - There is verified, trustworthy **evidence**
  - Process related
- Reliability:
  - In absence of faults, MTBF is >> life-time: **QA aspects**
- Resilience:
  - The fault behaviour is predicted: **trustworthy behaviour**
  - Capability to continue to provide core function
- Level: ARRL is normative
  - Components can be classified: **contract**

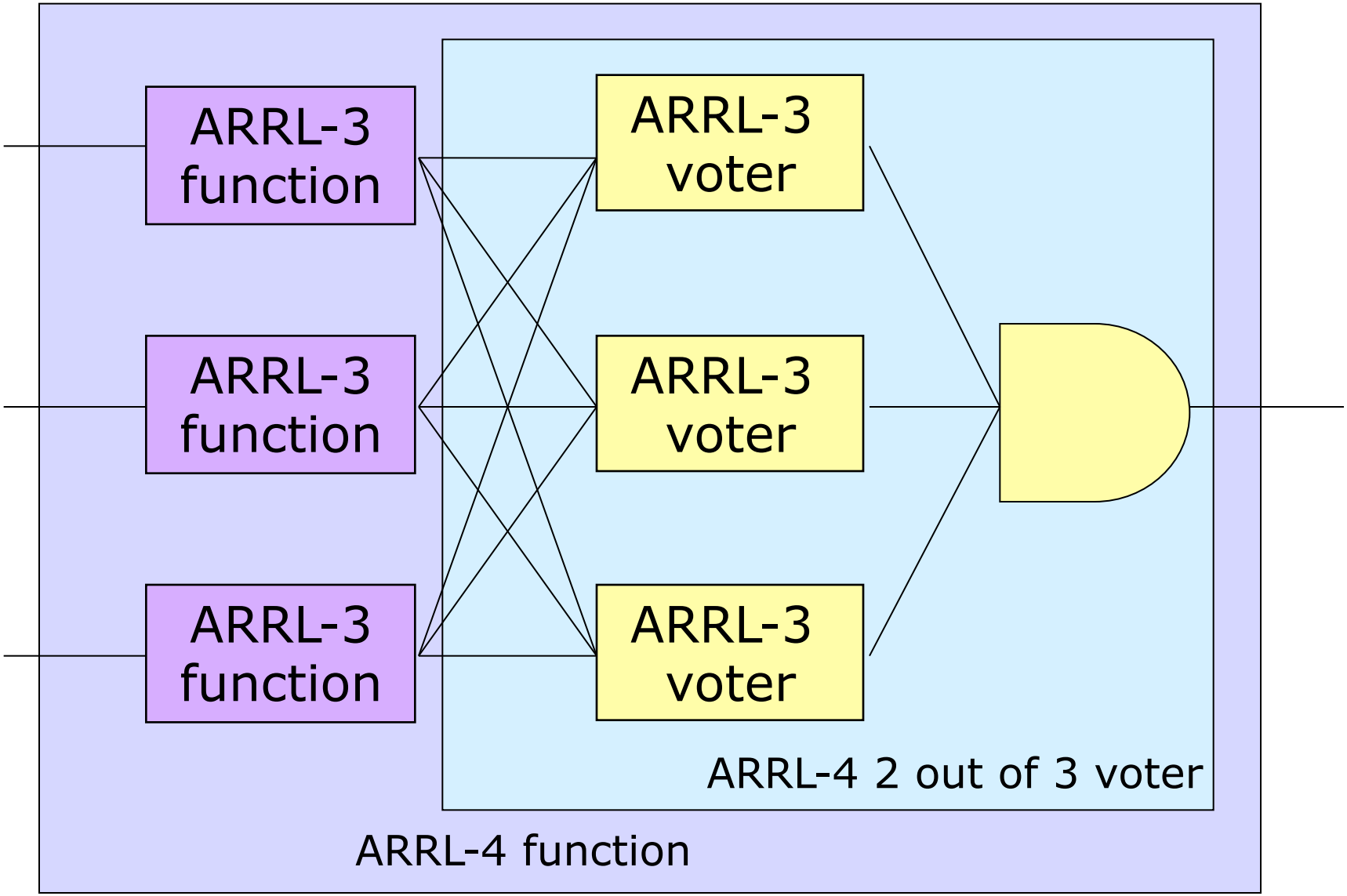
Consequences

- 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)

Composition rule:

- **A system can only reach a certain SIL level if all it components are at least of the same ARRL level.**
  - This is a necessary condition, not a sufficient condition
  - Redundancy can compose ARRL 4 components out of ARRL 3 components (needs an ARRL 4 voter)
  - ARRL3 component can use ARRL 2 components (>2)
- Consequences:
  - Interfaces and interactions also need ARRL level!
  - Error propagation is to be prevented => partitioning architecture (e.g. distributed, concurrent)
  - Using ARRL-3/4 components means that the system becomes resilient: run-away situations leading to critical states are contained.

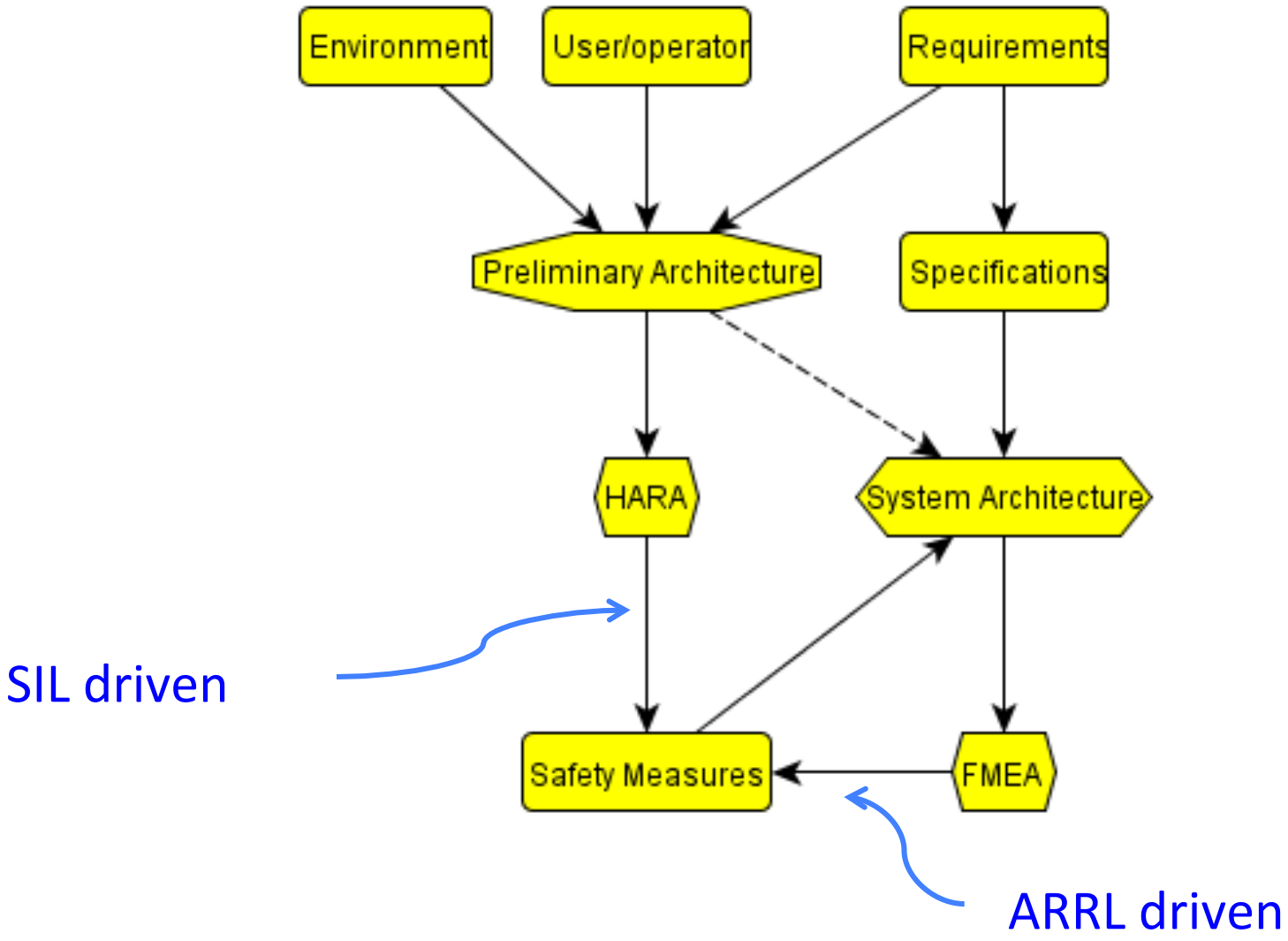
Generic example



Common mode failures => **ARRL-5**

- ARRL-4 assumes independence of faults in each redundant channel
- Covers only a subset of the common mode failures
- Less visible are e.g. common misunderstanding of requirements, translation tool errors, time dependent faults => require asynchronous operation and diversity/heterogenous solutions
- Hence we can define an ARRL-5 as well

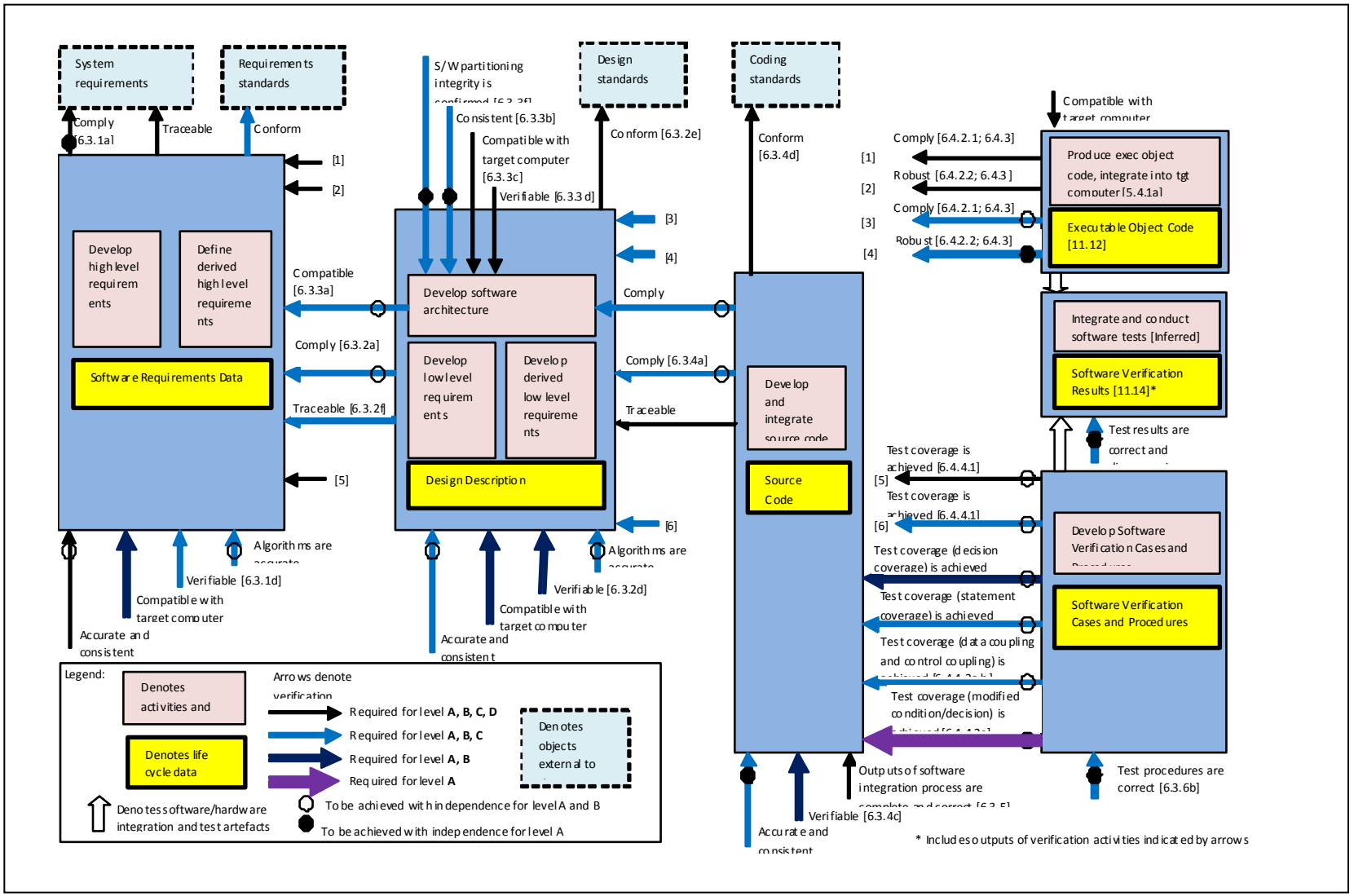
SIL and ARRL are complementary



What about a unified SE process?

- If ARRL can be generic, what about the process to be followed?
- We need a common process pattern
- ARRL allows this by separating core functionality from “safety” functionality and architecture: **fault behavior is explicit**
- Clean boundaries between ARRL levels vs. unclear decomposition of SIL levels in a system

Current practice (example from DO-178B - SW)



RTCA-DO/178B Software Development and Verification Processes

Phase	ARRL-1, ARRL-2	ARRL-3 additional	ARRL-4 additional	ARRL-5 additional
Requirements capturing	Normal cases Test cases	Fault cases (safety and security cases)	Requirements on fault tolerance.	Requirements on diversity and independence.
Specifications derivation by refinement	Functional and non-functional specifications derived from requirements	Safety and security specifications derived from safety requirements by analysis (HARA). Explicit fail-safe mode.	Specifications on selected fault tolerant architecture.	Specifications on selected diversity support.
Model building by refinement and specifications mapping	Architectural model. Simulation model. Formal models from ARRL-2 on.	Formal models. All models include safety and security support.	See ARRL-3. Models include fault-tolerant functionality.	See ARRL-4 Heterogeneous models.
Model analysis and verification/testing	On normal case architecture and models.	Evidence of a fail-safe architecture	Evidence of a fault tolerant architecture	Evidence of an heterogeneous / design diverse fault tolerant architecture
Implementation	Manual or code generation	See ARRL-2, code generation recommended	See ARRL-3	See ARRL-4
Integration and validation	Does the ARRL-1, -2 implementation meet SIL-1 or -2 level?	Does the ARRL-3 implementation meet the SIL-3 level?	Does the ARRL-4 implementation meet the SIL-4 level?	Does the ARRL-5 implementation meet the SIL-5 level?

ARRL driven process flow:

each level adds to a lower level

Dependency analysis

- SIL is a top level Requirement, decomposable in
  - Normal Case requirements (ARRL 1 &2)
  - Fault Case Requirements (ARRL 3 & 4 or even 5)
    - Will require extra resources
- More complex systems possible:
  - If all components/sub-systems are ARRL-4
- Consequence: critical state is reduced (resilience increased)

Further work

- Make ARRL normative and applicable?
  - Refinement of criteria
  - Completeness of criteria
  - Normative: components carry contract and evidence
    - Independent of final use or application domain
  - Impact on System Engineering process
  - Impact on System Architecture
    - Link with a system's critical states
  - Apply it on real cases
- Input and feedback welcome

Conclusions

- Unified system and safety engineering is feasible
- Unified safety certification is not yet feasible (standards and SIL differ too much)
- ARRL concept allows compositional safety engineering with reuse of components/ subsystems
- More complex systems can be safer
- A unified ARRL aware process pattern can unify systems and safety engineering standards

More info:

www.altreonic.com

White paper as work in progress available