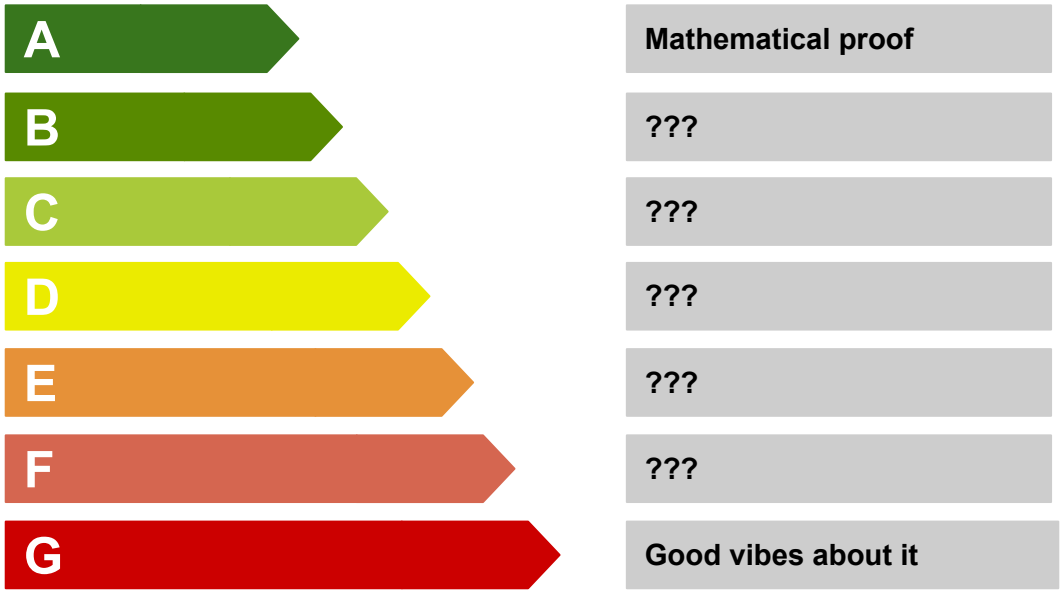
Resilient
Safety Analysis and Qualification

Igor Stoppa

Trusting Safety Practices: when/which/why?

Very wide spectrum of confidence



Arguments Supporting the Safety Qualification

The Safety Qualification involves these dimensions

Safety Analysis of the Architecture

Static Validation of the Implementation

Testing/Validation

Limits of the Safety Qualification process

Qualification relies on a variety of activities:

- **some are deductive**
- **others inductive.**

Qualification is subject to errors, omissions, accidents.

Qualification is a measurement.

Meaningful measurements indicate their accuracy.

Not only in the Eye of the Beholder

What affects the accuracy of a qualification process?

- Qualification steps
  - Correctness
  - Completeness
- Target component(s)
  - Amount of components
  - Complexity of individual components
  - Complexity of their interaction

Correlation Between Qualification accuracy and Targets

Correctness & Completeness of the Qualification depend on the Target Components

What is the likelihood of errors?
What is the likelihood of omissions?

Many complex components are much harder to qualify accurately, than fewer, simpler ones.

Complexity

Growing Complexity vs Decreasing Qualification Accuracy

Example 1: Simple Component

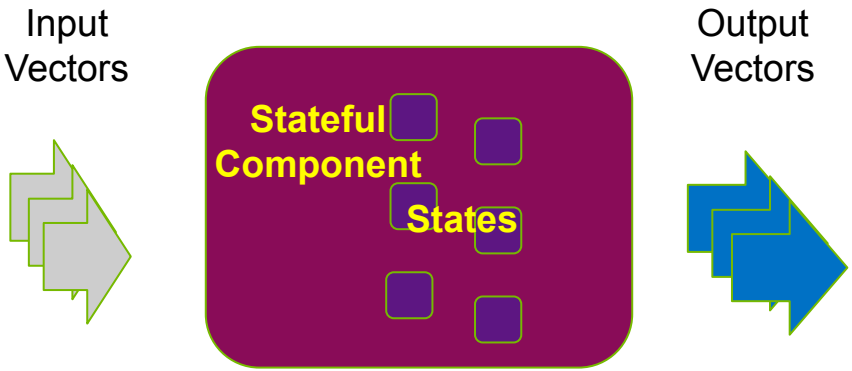
Low Complexity, High Qualification Accuracy

HIGH CONFIDENCE

E.g.: crypto functions, memset, memcpy

Growing Complexity vs Decreasing Qualification Accuracy

Example 2: Medium Complexity Component



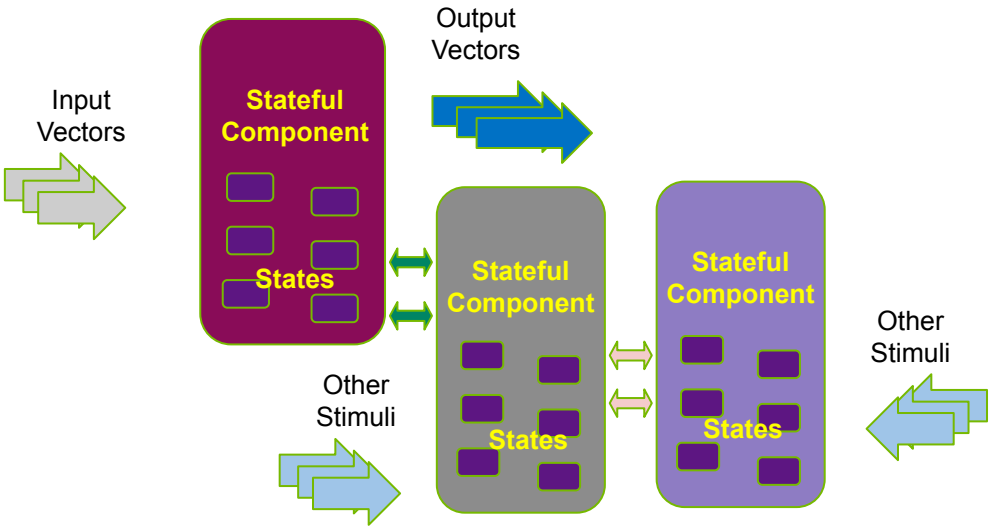
Medium Complexity, Medium Qualification Accuracy

BORDERLINE ACCEPTABLE

E.g.: Simple security modules (Apparmor)

Growing Complexity vs Decreasing Qualification Accuracy

Example 3: High Complexity Component



High Complexity, Low Qualification Accuracy

LOW CONFIDENCE

E.g.:
Memory, Process, Files, Network Management, Complex Security modules (SELinux), cgroups, etc.

Effects on the Safety Analysis

The Linux kernel relies on many loosely coupled components and many advanced programming techniques.

Manual inspection is error-prone.

- Even core developers make mistakes (there is no bug-free code)
- Usually performed by far less people, often lacking domain knowledge

Effects on the Static Code Analysis

Typical Functional Safety metrics are often ill fitted:

- Cyclomatic complexity captures only the local complexity in a unit, but it completely misses out on indirect modules interplay.
- Static Code Analysis can generate overwhelming numbers of false positives. What to do with them?

Effects on runtime measurements

Exponential explosion of path permutations.
Internal states evolving in parallel over many (HW) threads.

- **Code coverage doesn't account for:**
  - **state interdependency**
  - **hidden, asynchronous, code paths**
    e.g. involving exceptions and interrupts

- **FuSa oriented Testing is similarly limited:**
  - **Most of available testing is functionality-oriented**
  - **Available Fault injection is limited in what faults it can inject, and how.**

How complexity affects Trust in the Safety Qualification

Is my safety strategy prepared to be challenged?

Self-assessment

For any given component, am I able to explain:

- what it does? How?
- its inputs?
- the expected outputs?
- how it interacts with other components?
- its implementation?
- how to introduce a new functionality?

The answers will highlight a varying degree of confidence.

Self-assessment

If I cannot explain the code of a component ...

... how good is my safety analysis of it?

Practices

Safety Analysis:

- How exhaustive is it? Can I show completeness?
- Does it account for interference?
- Does it provide objective evidence? Negative testing?

Self-assessment

If I cannot verify empirically a claim …

… how do I know that it's correct?

Practices

Safety Mitigations:

- How comprehensive are they?
- Is their efficacy demonstrable?
- Are the dependencies sufficiently safe?
  e.g. Am I assuming the kernel to not corrupt its monitor(s)?

Self-assessment

My safety story is a chain of dependencies …

… but how strong is its weakest link?

Practices

Testing:

- Am I able to describe the internal states?
- In which ways inputs can affect the internal states?
- In which ways internal states can affect the outputs?
- Am I able to explain the degree of completeness?

Self-assessment

If I rely primarily on black-box testing …

… how can I avoid survivor bias?

Practices

Stress Testing:

- Am I able to justify the relevance of a test pattern?
- Am I able to reproduce potential findings?

Antipatterns

Confidence inversely proportional to complexity
(and internal states)

- Only simple functionality can be tested exhaustively
- As the number of internal states grows, the level of exhaustiveness drops rapidly
- Assuming to be able to test for safety a complex component is a red flag

Antipatterns

Using a simplified model is VERY unlikely to be sufficient

- Creating a model based on partial understanding is a red flag
- Defining a testing campaign based on budget, rather than the analysis of the actual model is a red flag

Antipatterns

(Stress) Testing must be based on solid understanding of the system

- Complex systems are unlikely to be sufficiently understood
- Reproducibility of issues found in complex components can be very unreliable *(Did a bugfix really work? Is the problem still lurking?)*
- Sufficient coverage is equally difficult to prove
- Claiming that understanding “is good enough” is a red flag

How to increase Trust in the Safety Qualification?

Self-Assessment

What is my positioning toward qualification?

- **Am I ready to adjust my safety concept, according to findings?**
- **Am I following the guiding principles of the standards?**

Self-Assessment

What if my assumptions/expectations are unmet?

- **To which extent am I able to recover from a failure in my initial assumptions?**

Self-Assessment

Am I probing my safety concept for failures? Incompleteness?

- Can I detect and report safety relevant failures?
- Do I have evidence that detection and reporting works?
- If not, how can I prove that I'm not victim of survivor bias?

My customer is not a beta-tester

Self-Assessment

If the Qualification strategy doesn't tolerate errors …

… can I prove to be error-free?

Ideas for a Resilient Safety Qualification

- Require only simple components to be qualified (and rely only on those)
- Use alternative approach for more complex components
- Implement Verification of detection and reporting

Ideas for a Resilient Safety Qualification

Require only simple components to be qualified (and rely only on those)

- Less chances of qualification errors
- Simpler scenarios to consider
- Less dilution of the overall effort required
- More maintainable over time

Ideas for a Resilient Safety Qualification

Use alternative approach for more complex components

- Avoid qualifying most complex ones - they have lower confidence
- Carefully evaluate mid-complexity ones, can they be avoided?
- Use independent redundancy mechanisms, to cope with the inherently less reliable qualification
  (YES, everything else equal, it WILL be less reliable, because the component is more complex)

Ideas for a Resilient Safety Qualification

Implement Verification of detection and reporting

- Confirm the correctness of the claims through direct verification (i.e. do BOTH positive and negative testing)
- Assume what is not tested to be broken

Thank You