The Trustable Software Framework

A new way to measure risk in continuous delivery of critical software

Codethink is an international provider of **expert software engineering**, **solutions**, and **consultancy services**...

mainly based on **FLOSS**.

“Why do you trust software?”

```
curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh
```

(from 2016)

Q: “How could open source concepts, techniques and tools help us to achieve safety in complex systems?”

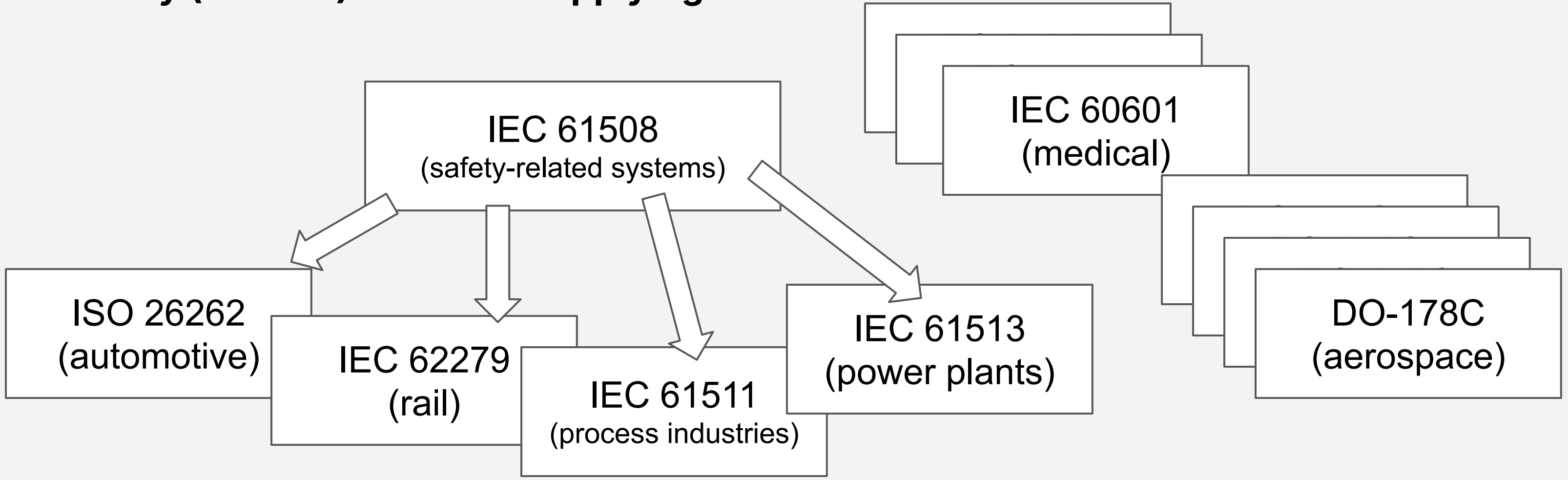
A: “You can’t use open source for safety!”

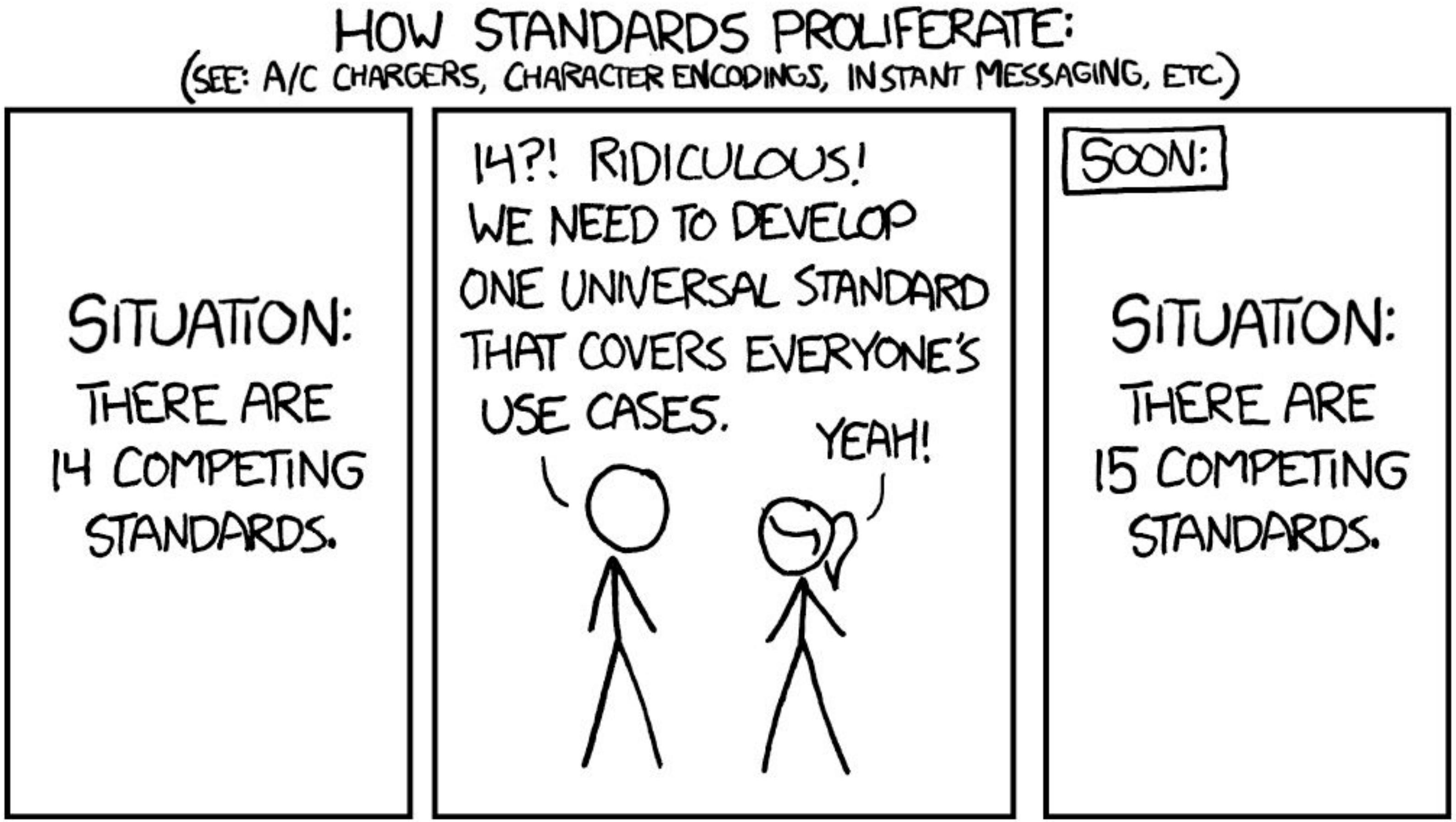
FLOSS

- doesn’t comply with the standards
- doesn’t have “requirements”
- doesn’t have “architecture”
- no traceability

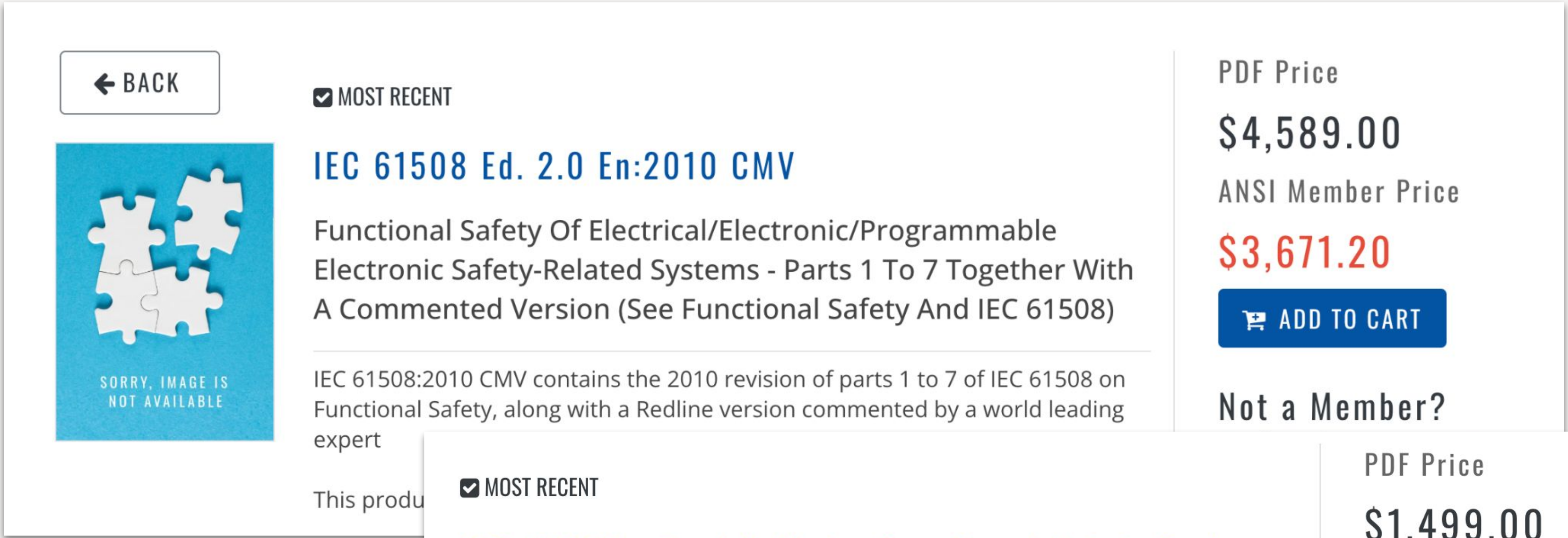
… so the quality is not acceptable

Safety (or FuSa) Standards applying to software…




HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/927/



BACK

MOST RECENT

IEC 61508 Ed. 2.0 En:2010 CMV

Functional Safety Of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Parts 1 To 7 Together With A Commented Version (See Functional Safety And IEC 61508)

IEC 61508:2010 CMV contains the 2010 revision of parts 1 to 7 of IEC 61508 on Functional Safety, along with a Redline version commented by a world leading expert

This produ

PDF Price

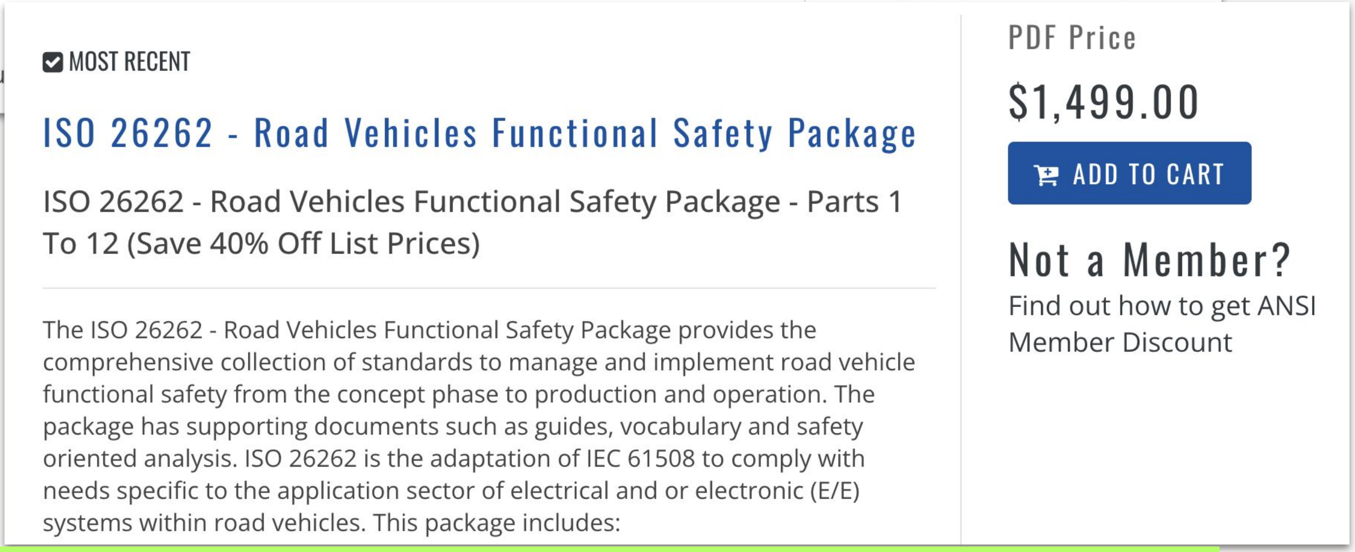
$4,589.00

ANSI Member Price

$3,671.20

ADD TO CART

Not a Member?



MOST RECENT

ISO 26262 - Road Vehicles Functional Safety Package

ISO 26262 - Road Vehicles Functional Safety Package - Parts 1 To 12 (Save 40% Off List Prices)

The ISO 26262 - Road Vehicles Functional Safety Package provides the comprehensive collection of standards to manage and implement road vehicle functional safety from the concept phase to production and operation. The package has supporting documents such as guides, vocabulary and safety oriented analysis. ISO 26262 is the adaptation of IEC 61508 to comply with needs specific to the application sector of electrical and or electronic (E/E) systems within road vehicles. This package includes:

PDF Price

$1,499.00

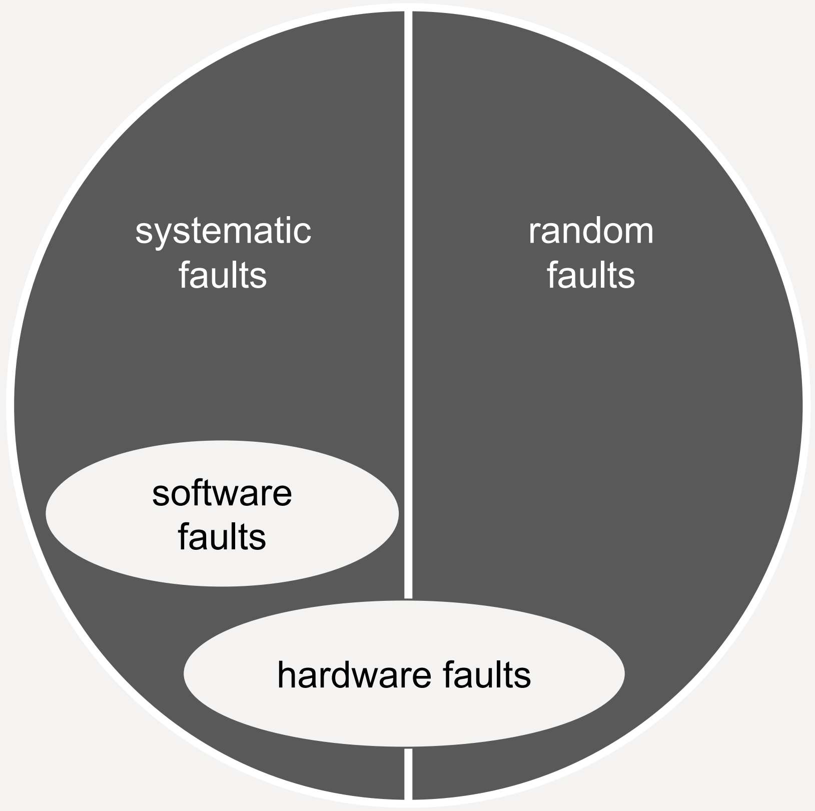
ADD TO CART

Not a Member?

Find out how to get ANSI Member Discount

NOTE: these standards are created by volunteers - subject-matter experts working for free!

ISO 26262 base assumption



This was defensible for microcontroller-based systems, where the expectation was deterministic software behaviour

Implication of “no random faults”

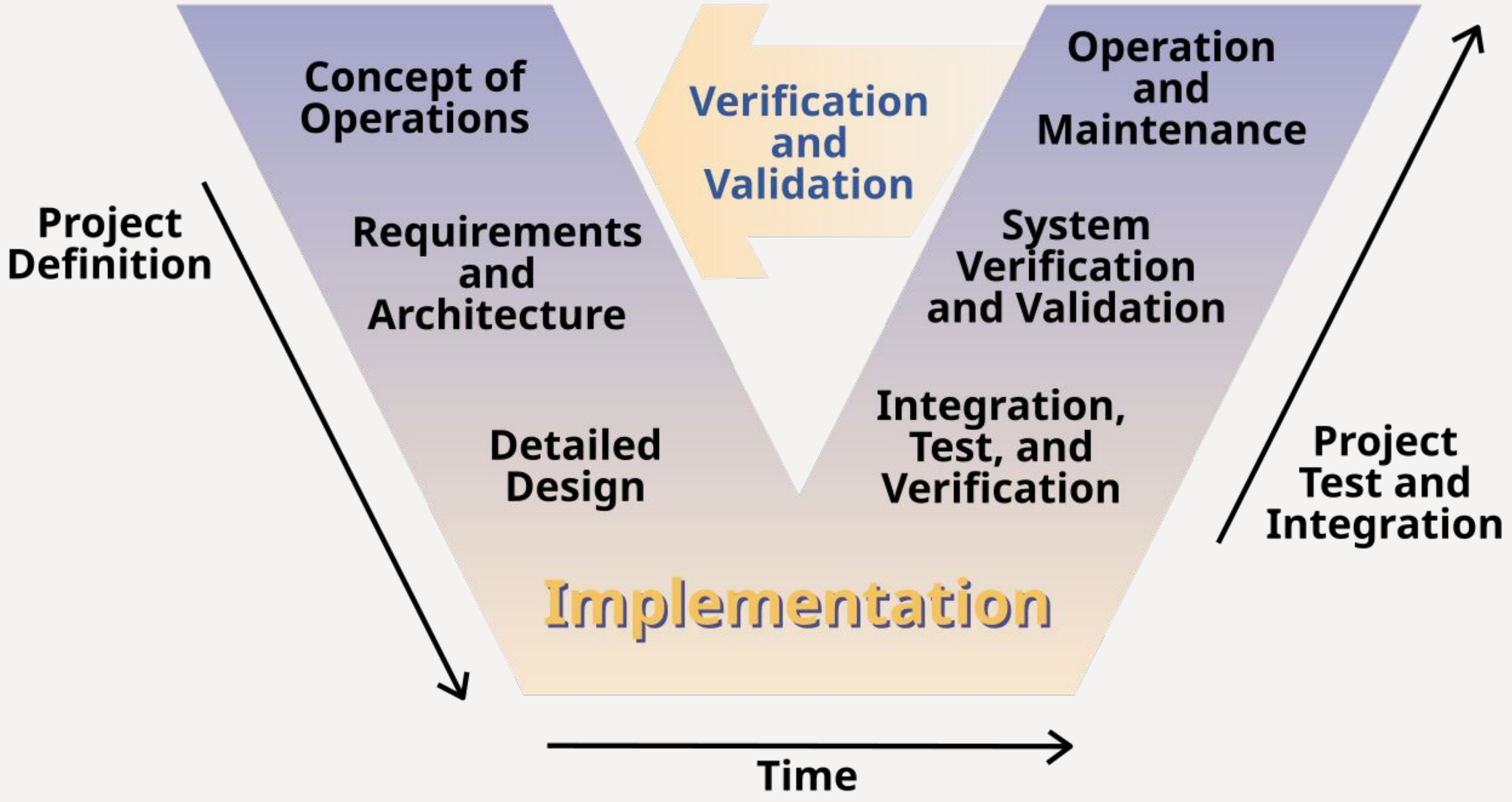
Since all faults are ‘systematic’, focus all your attention on ‘best practice’ processes... specification, architecture, design, testing, traceability.

Unfortunately the standard’s recommended processes are no longer best practice. Software integration is hardly even considered.

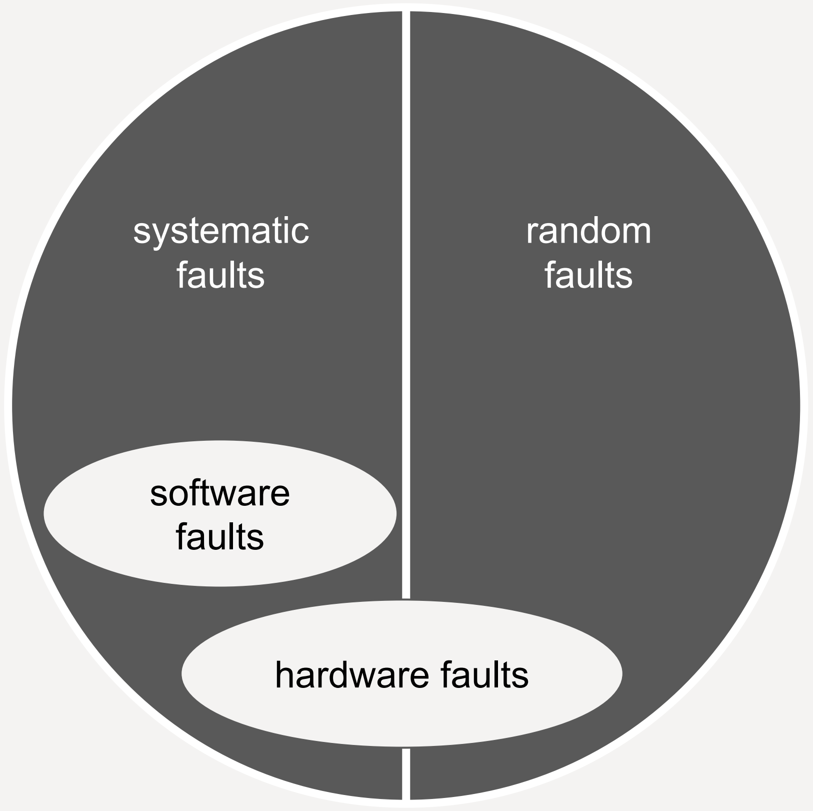
Most modern software is created iteratively, without formal specifications*. The processes and techniques themselves evolve over the life of a project.

* most organisations innovating in software have adopted “Agile”, Open Source or both

The V-Model: Best practice?


Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

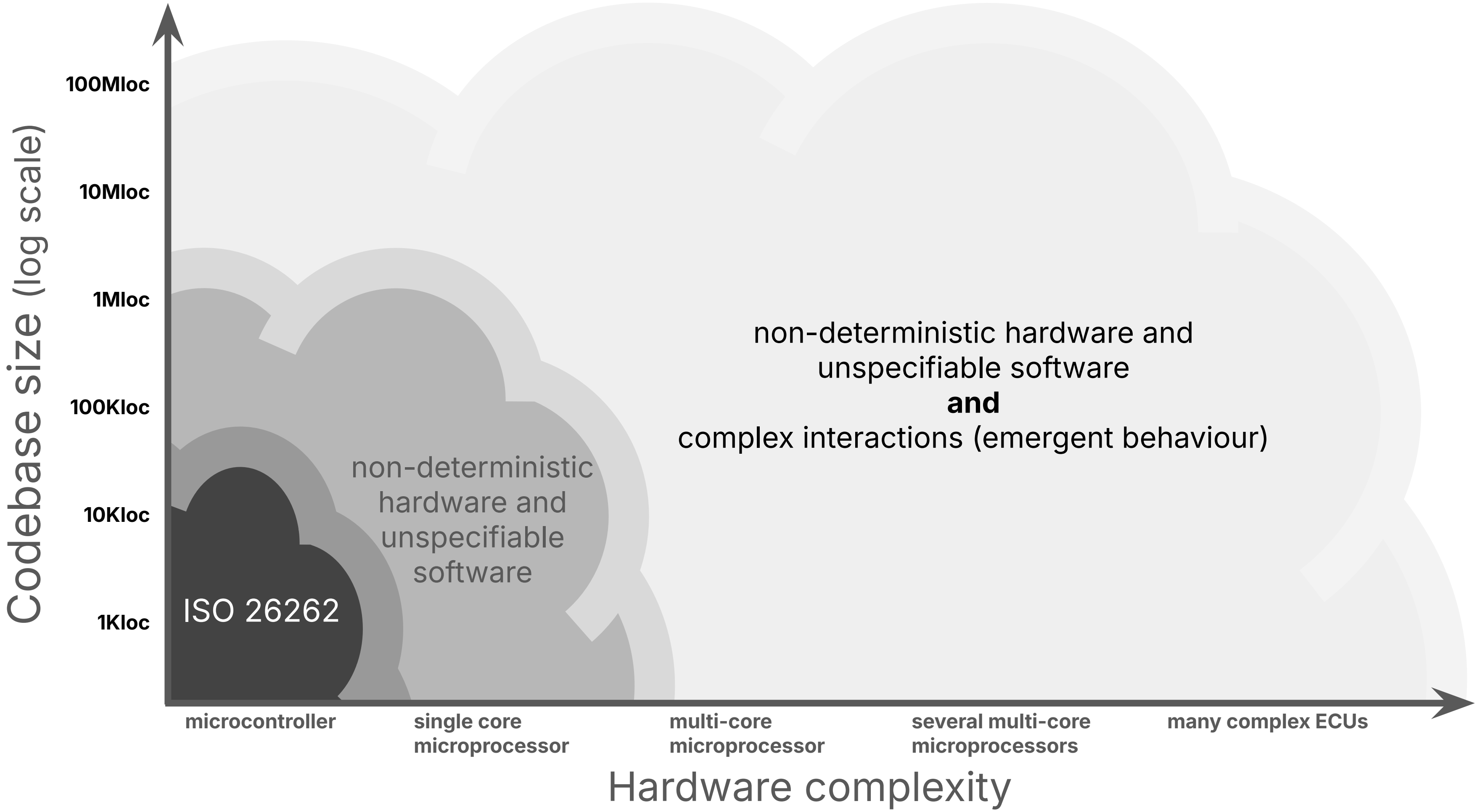
ISO 26262 base assumption



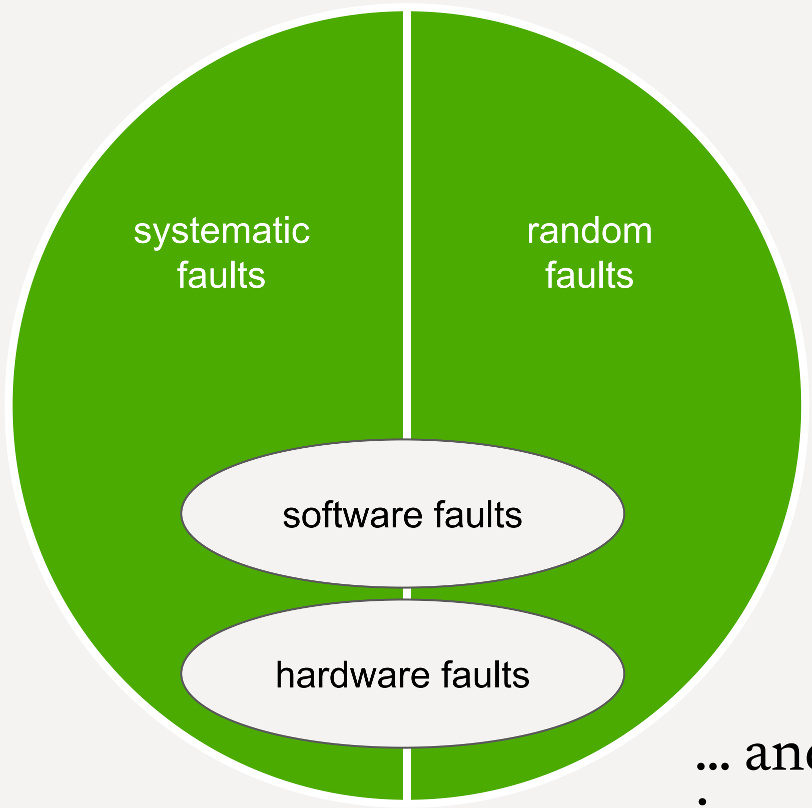
This assumption leads to extreme concentration on process... even to the point of trying to establish new processes for existing software, no matter how mature, successful or widely adopted...

and NO FLOSS

We're not in Kansas anymore


Codebase size (log scale)
100Mloc
10Mloc
1Mloc
100Kloc
10Kloc
1Kloc
ISO 26262
non-deterministic hardware and unspecifiable software
non-deterministic hardware and unspecifiable software
and
complex interactions (emergent behaviour)
microcontroller
single core microprocessor
multi-core microprocessor
several multi-core microprocessors
many complex ECUs
Hardware complexity

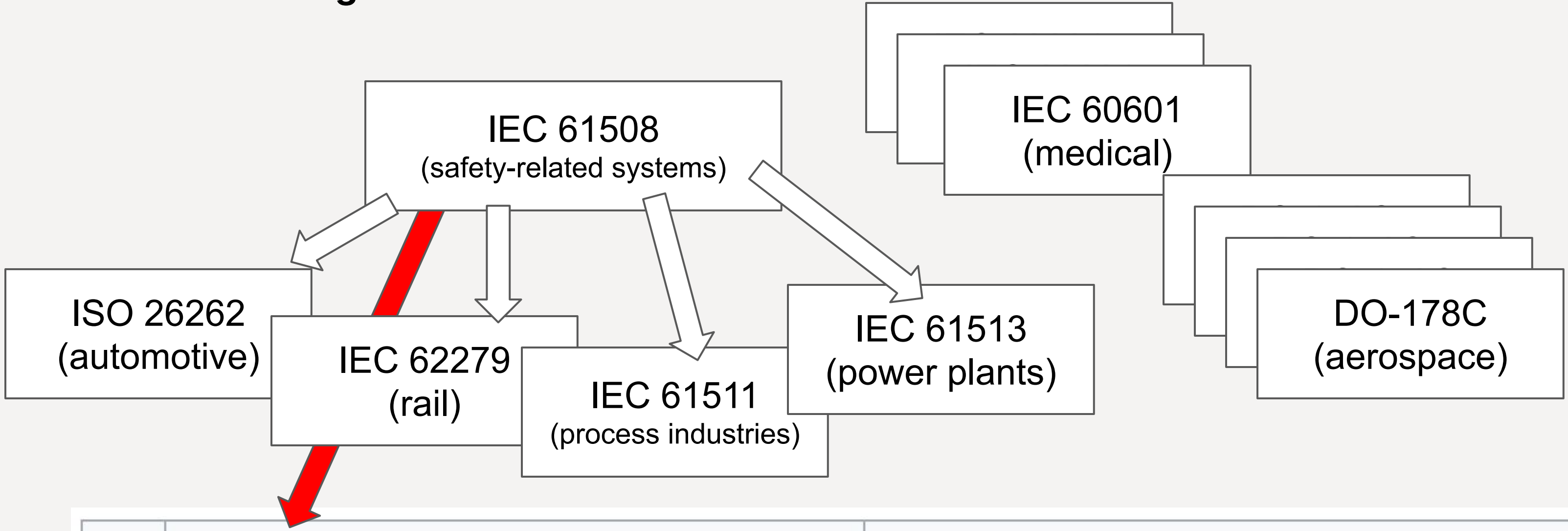
Codethink base assumption

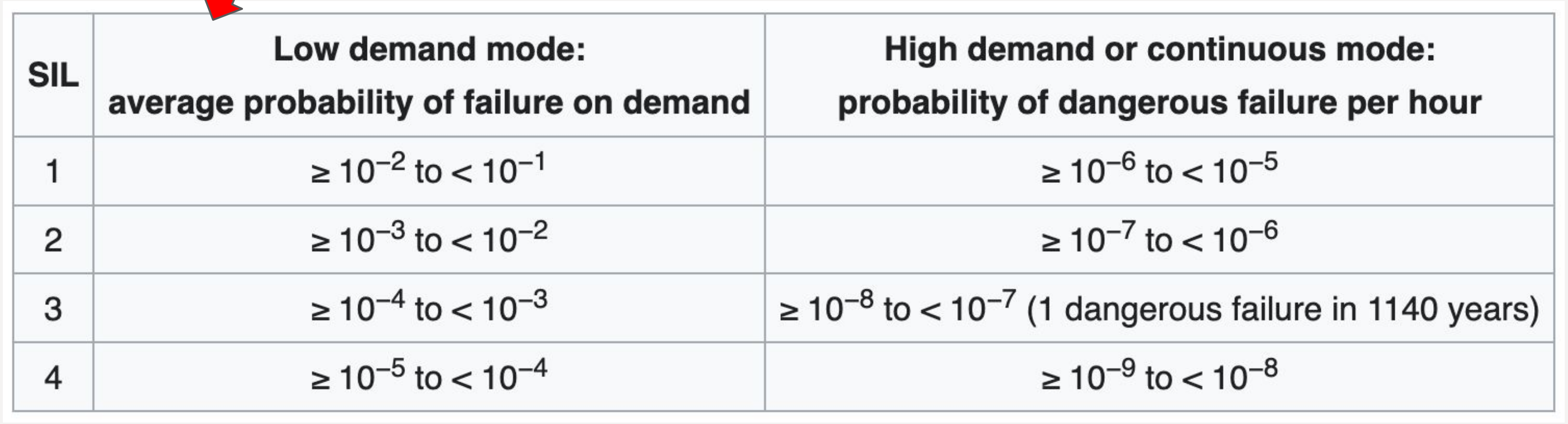


This is where we are now; non-deterministic behaviour in multicore microprocessor systems with L1 + L2 cache, MMU and approx 1MLOC of firmware

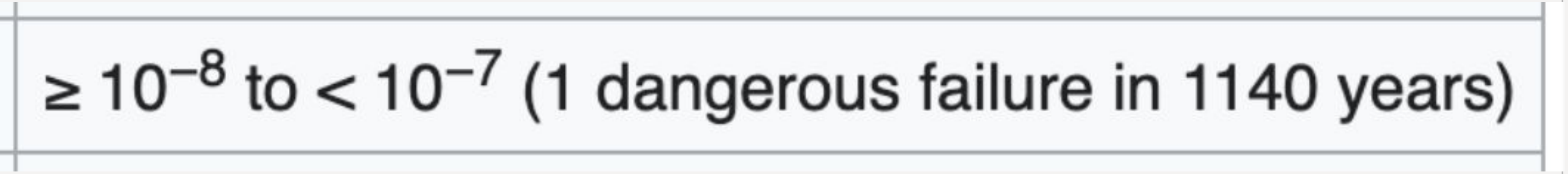
... and millions more LOC in our supply chain

IEC 61508 sets goals for failure rates...



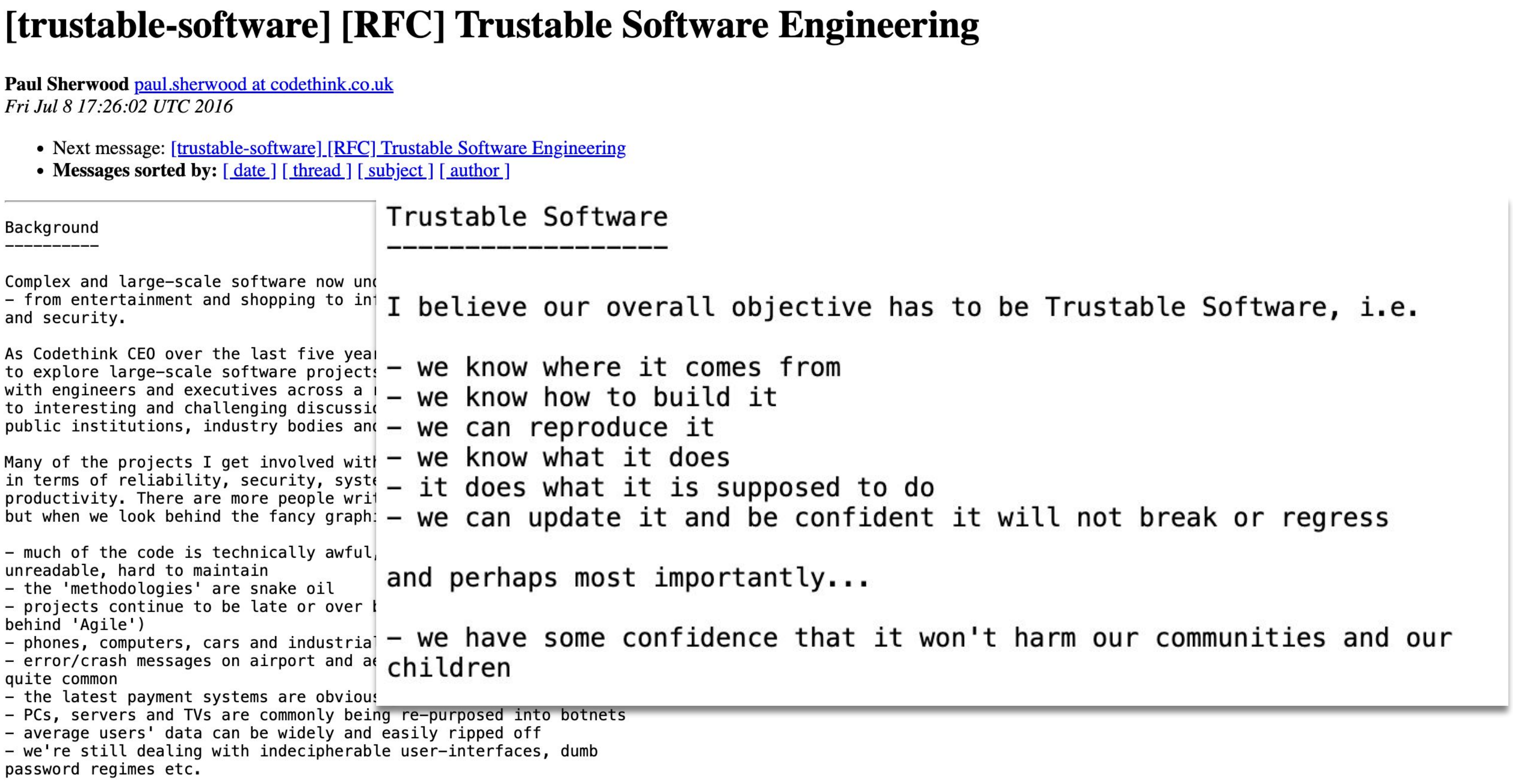


SIL	Low demand mode: average probability of failure on demand	High demand or continuous mode: probability of dangerous failure per hour
1	$\geq 10^{-2}$ to $< 10^{-1}$	$\geq 10^{-6}$ to $< 10^{-5}$
2	$\geq 10^{-3}$ to $< 10^{-2}$	$\geq 10^{-7}$ to $< 10^{-6}$
3	$\geq 10^{-4}$ to $< 10^{-3}$	$\geq 10^{-8}$ to $< 10^{-7}$ (1 dangerous failure in 1140 years)
4	$\geq 10^{-5}$ to $< 10^{-4}$	$\geq 10^{-9}$ to $< 10^{-8}$



$\geq 10^{-8}$ to $< 10^{-7}$ (1 dangerous failure in 1140 years)

This is a very hard target.



[trustable-software] [RFC] Trustable Software Engineering

Paul Sherwood paul.sherwood at codethink.co.uk
Fri Jul 8 17:26:02 UTC 2016

- Next message: [trustable-software] [RFC] Trustable Software Engineering
- **Messages sorted by:** [date] [thread] [subject] [author]

```
Background
----------

Complex and large-scale software now und
- from entertainment and shopping to inf
and security.

As Codethink CEO over the last five year
to explore large-scale software projects
with engineers and executives across a r
to interesting and challenging discussio
public institutions, industry bodies and

Many of the projects I get involved with
in terms of reliability, security, syste
productivity. There are more people writ
but when we look behind the fancy graphi

- much of the code is technically awful,
unreadable, hard to maintain
- the 'methodologies' are snake oil
- projects continue to be late or over b
behind 'Agile')
- phones, computers, cars and industrial
- error/crash messages on airport and ae
quite common
- the latest payment systems are obvious
- PCs, servers and TVs are commonly being re-purposed into botnets
- average users' data can be widely and easily ripped off
- we're still dealing with indecipherable user-interfaces, dumb
password regimes etc.
```

```
Trustable Software
------------------

I believe our overall objective has to be Trustable Software, i.e.

- we know where it comes from
- we know how to build it
- we can reproduce it
- we know what it does
- it does what it is supposed to do
- we can update it and be confident it will not break or regress

and perhaps most importantly...

- we have some confidence that it won't harm our communities and our
children
```

https://lists.trustable.io/pipermail/trustable-software/2016-July/000000.html

Key lesson:

trust should be based on

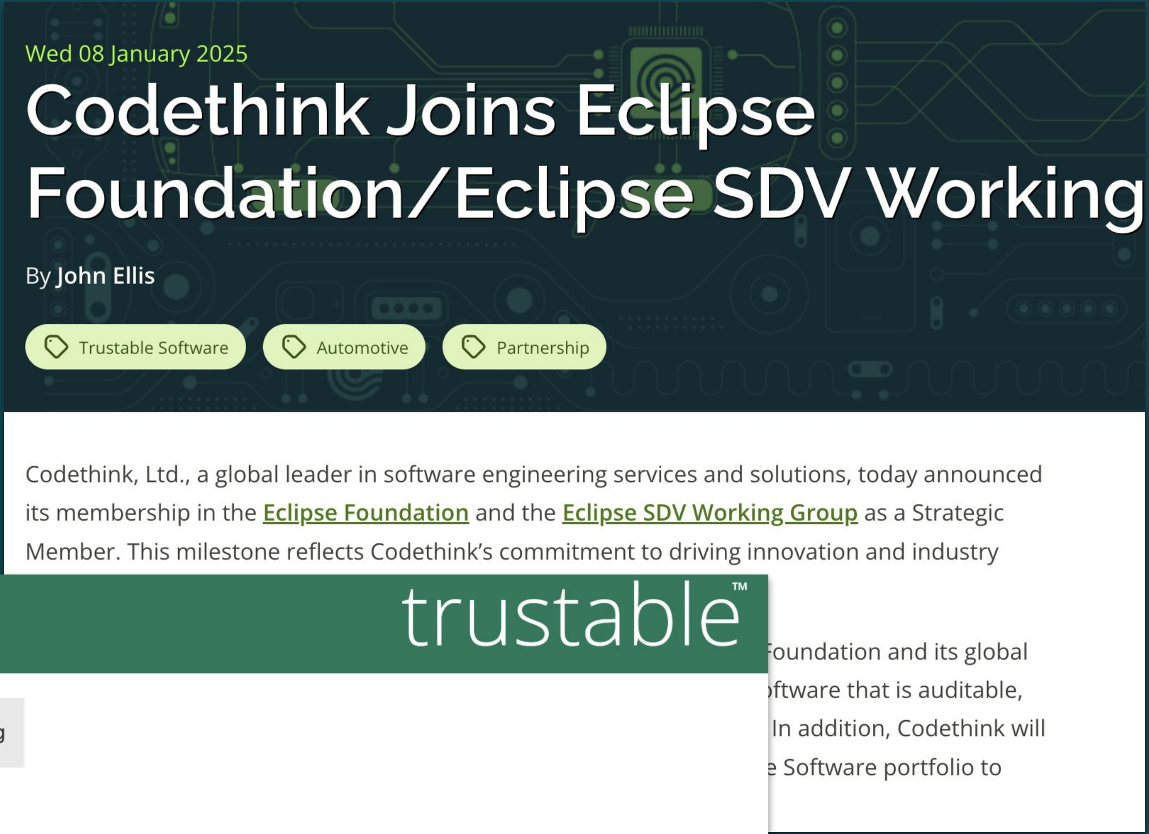
evidence

https://gitlab.com/trustable/documents/-/wikis/hypothesis-for-software-to-be-trustable

Trustable Software Framework

Note: *trustable* as opposed to *trusted, trustworthy*

Moving to Eclipse...



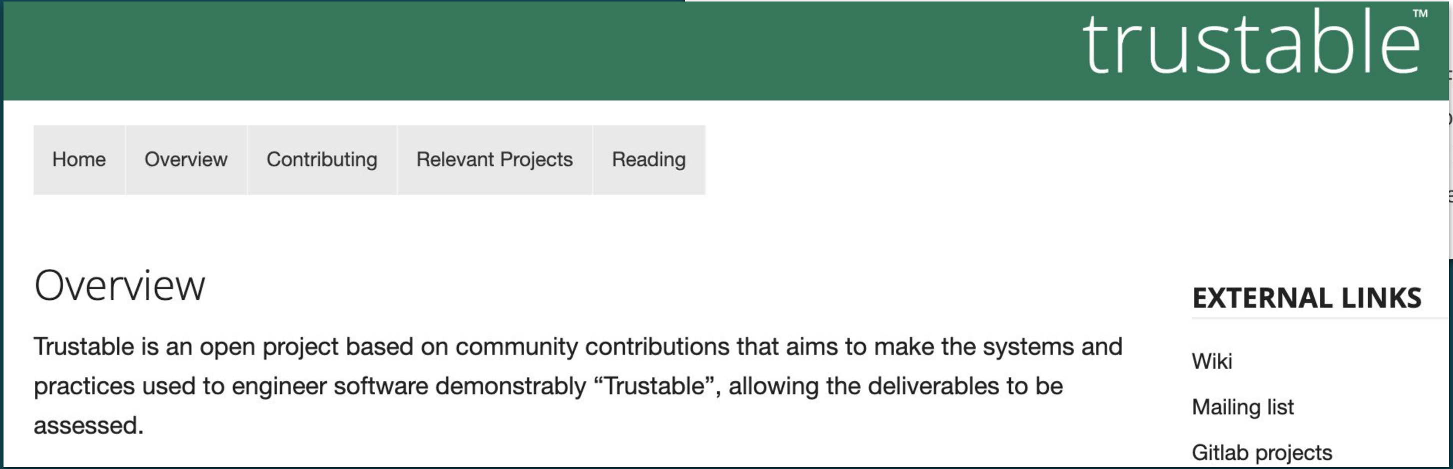
Wed 08 January 2025

Codethink Joins Eclipse Foundation/Eclipse SDV Working

By John Ellis

Codethink, Ltd., a global leader in software engineering services and solutions, today announced its membership in the Eclipse Foundation and the Eclipse SDV Working Group as a Strategic Member. This milestone reflects Codethink's commitment to driving innovation and industry

oundation and its global
oftware that is auditable,
In addition, Codethink will
e Software portfolio to



Overview

Trustable is an open project based on community contributions that aims to make the systems and practices used to engineer software demonstrably "Trustable", allowing the deliverables to be assessed.

EXTERNAL LINKS

- Wiki
- Mailing list
- Gitlab projects

Trustable Tenets

We can offer software as Trustable when we provide evidence to support all of these claims...

1. Provenance
Know where it comes from, who is responsible, and have confidence in them

2. Construction
Know how to build it - **reproducibly** - from source

3. Changes
Upgrade it and be confident it will not break or regress

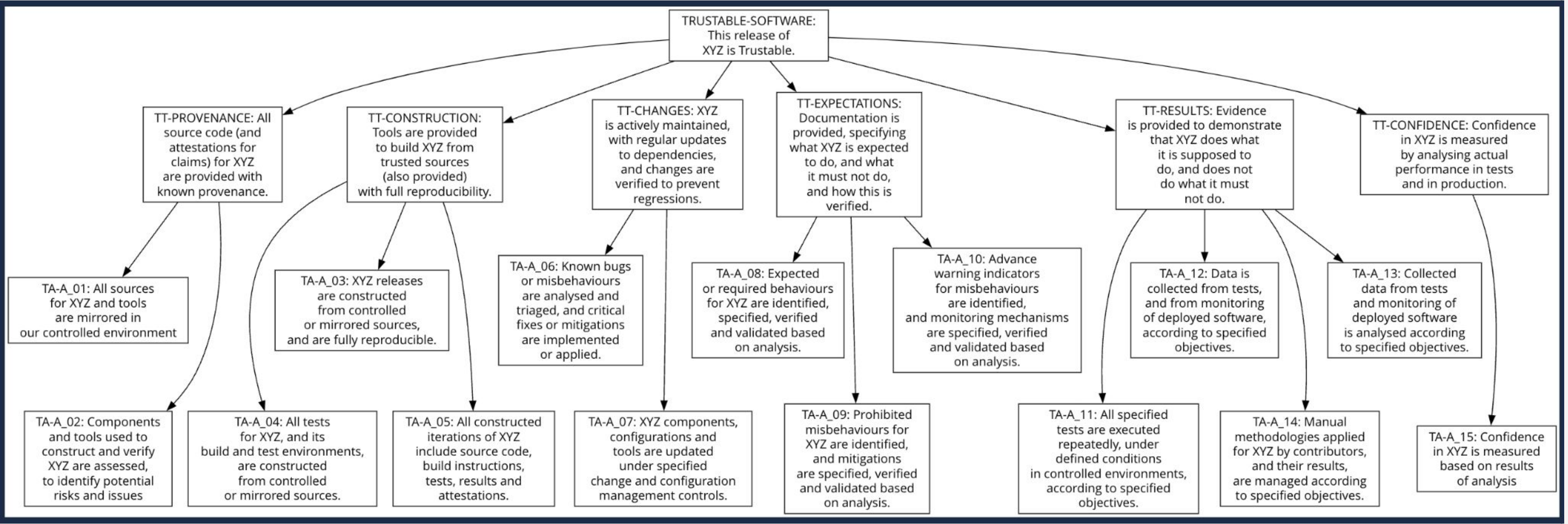
4. Expectations
Know what it must do, and what it must not do

5. Results
It does what it must do, and does not do what it must not do

6. Confidence
Measure and declare our confidence that it will not cause harm

Trustable Software Framework

The Trustable Software Framework (TSF) provides an extensible model for collecting, organising and evaluating evidence for releases of a software project/product ("XYZ"), to allow a consumer to consider to what extent they should trust the software. The TSF broadly identifies six key topics ("Trustable Tenets"), made up of more detailed factors ("Trustable Assertions").



TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.

TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.

TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.

TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.

TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.

TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.

TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.

TA-A_01: All sources for XYZ and tools are mirrored in our controlled environment

TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues

TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.

TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.

TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.

TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.

TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.

TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.

TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.

TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.

TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.

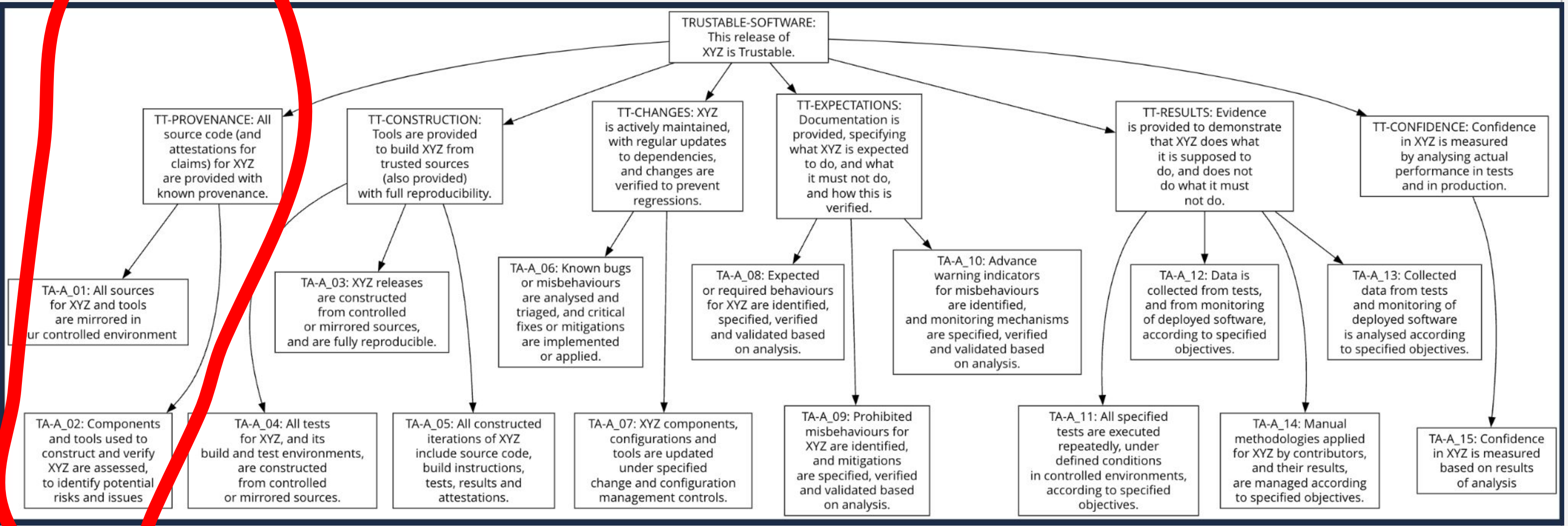
TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.

TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.

TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.

TA-A_15: Confidence in XYZ is measured based on results of analysis

https://gitlab.com/CodethinkLabs/trustable/trustable


TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.

TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.

TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.

TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.

TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.

TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.

TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.

TA-A_01: All sources for XYZ and tools are mirrored in ur controlled environment

TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues

TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.

TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.

TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.

TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.

TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.

TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.

TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.

TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.

TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.

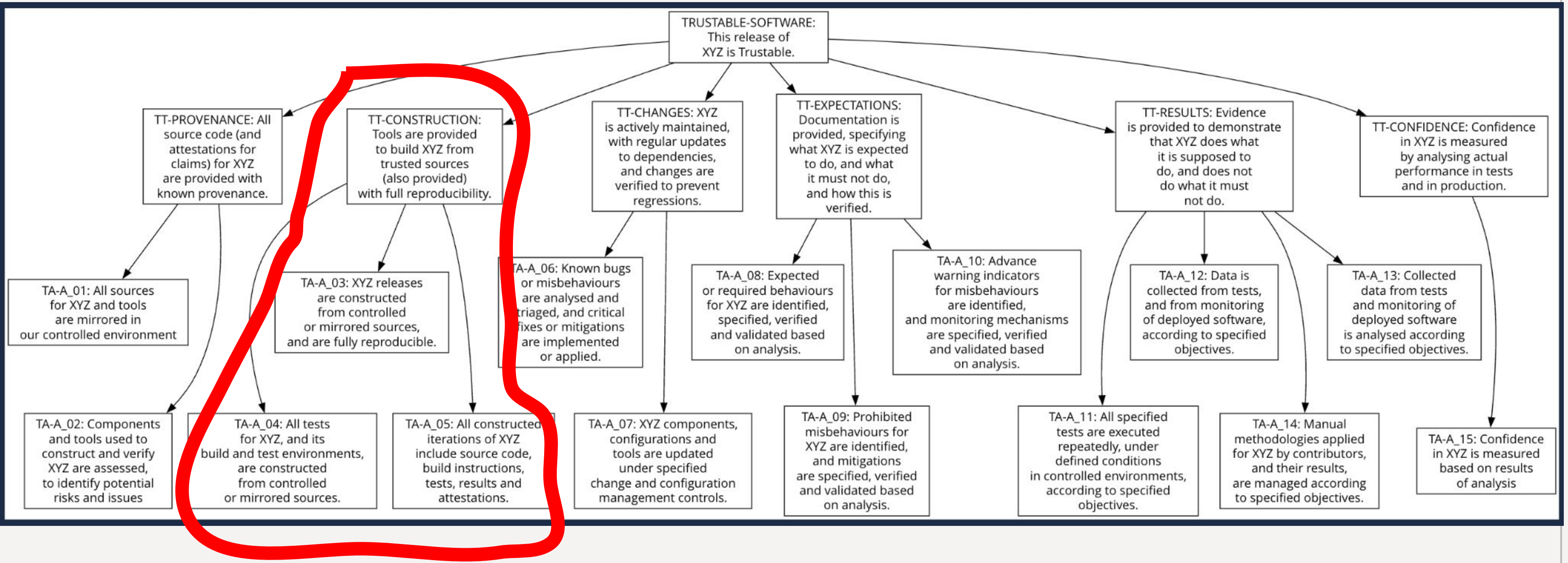
TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.

TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.

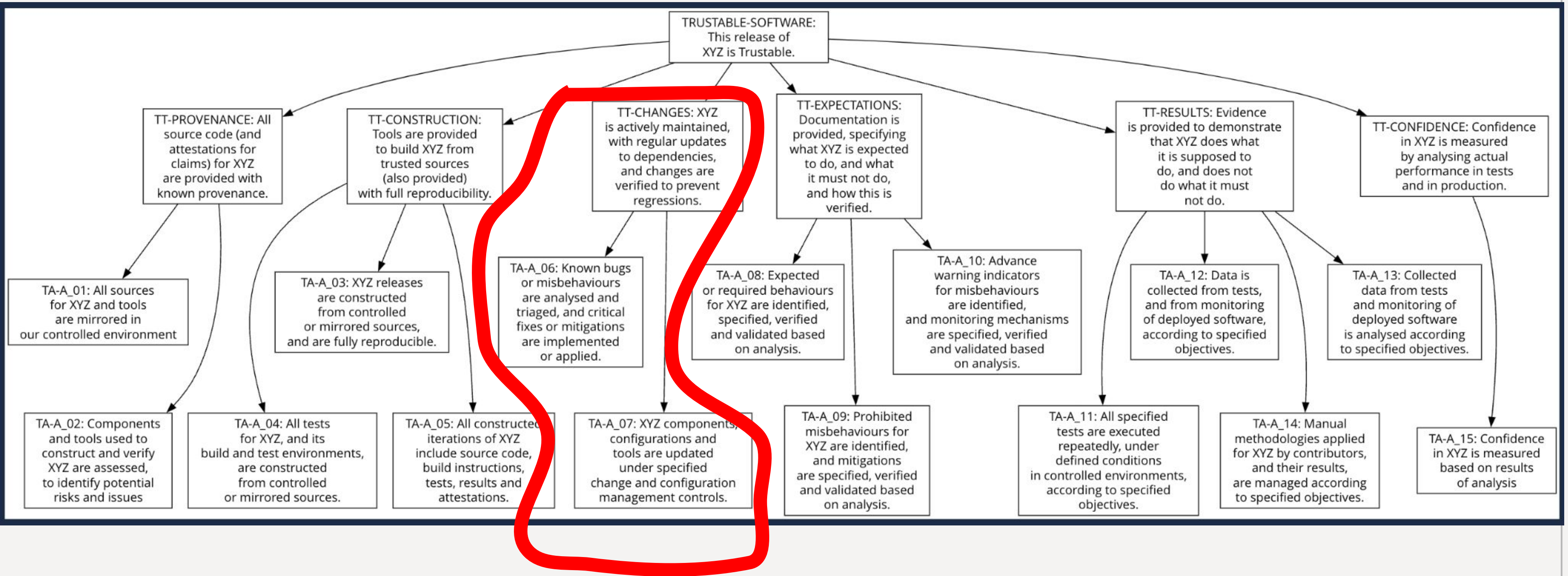
TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.

TA-A_15: Confidence in XYZ is measured based on results of analysis

SUPPLY CHAIN: provenance of all dependencies + toolchain components


TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.
TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.
TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.
TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.
TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.
TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.
TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.
TA-A_01: All sources for XYZ and tools are mirrored in our controlled environment
TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.
TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.
TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.
TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.
TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.
TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.
TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues
TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.
TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.
TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.
TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.
TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.
TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.
TA-A_15: Confidence in XYZ is measured based on results of analysis

CONSTRUCTION: zero trust builds - reproducible, no internet, no root



TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.

TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.

TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.

TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.

TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.

TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.

TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.

TA-A_01: All sources for XYZ and tools are mirrored in our controlled environment

TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues

TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.

TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.

TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.

TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.

TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.

TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.

TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.

TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.

TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.

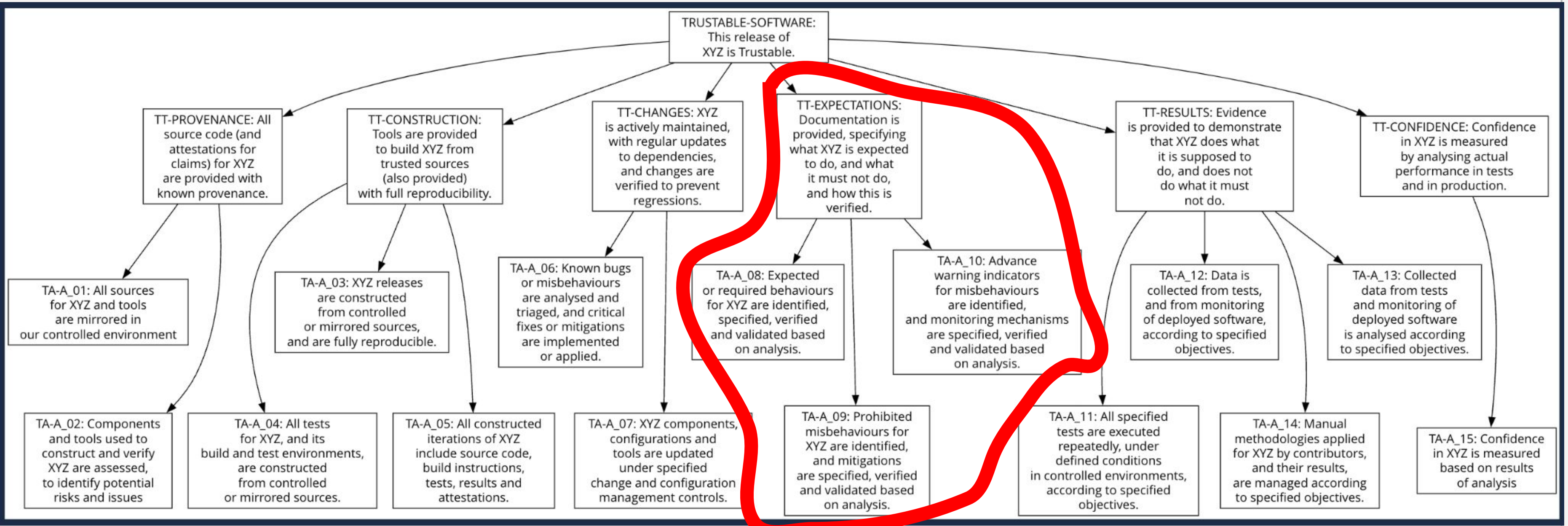
TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.

TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.

TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.

TA-A_15: Confidence in XYZ is measured based on results of analysis

CHANGES: CICD, consuming relevant upstream fixes and releases


TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.

TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.

TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.

TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.

TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.

TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.

TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.

TA-A_01: All sources for XYZ and tools are mirrored in our controlled environment

TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues

TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.

TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.

TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.

TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.

TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.

TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.

TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.

TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.

TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.

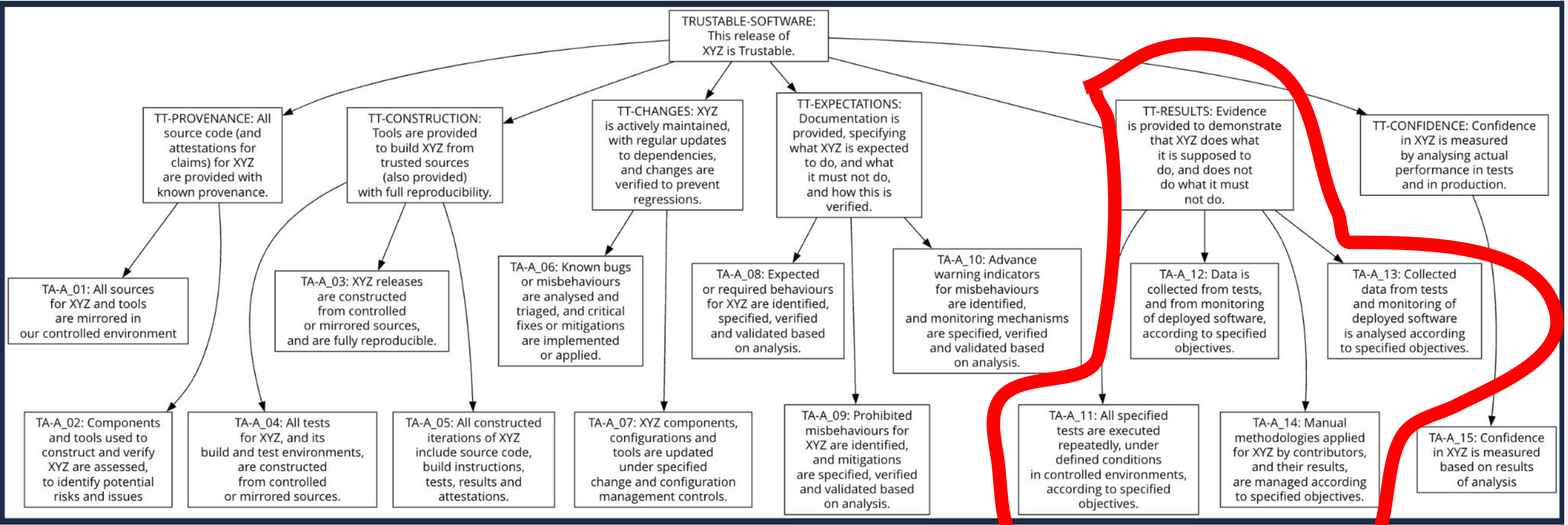
TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.

TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.

TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.

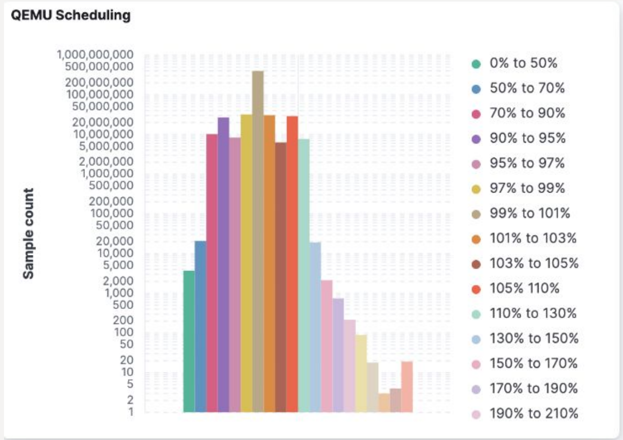
TA-A_15: Confidence in XYZ is measured based on results of analysis

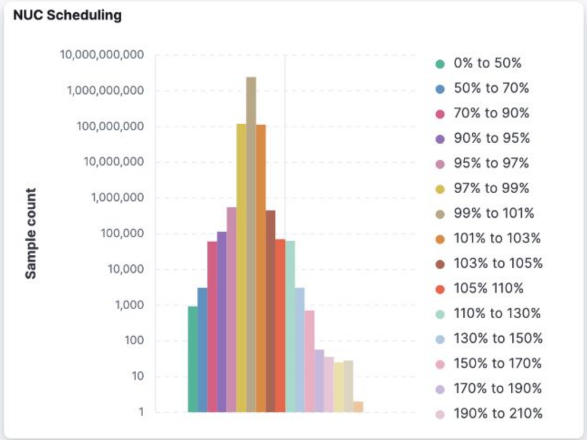
EXPECTATIONS: be clear about what it **must** do, and what can go wrong, with mitigations and warning mechanisms

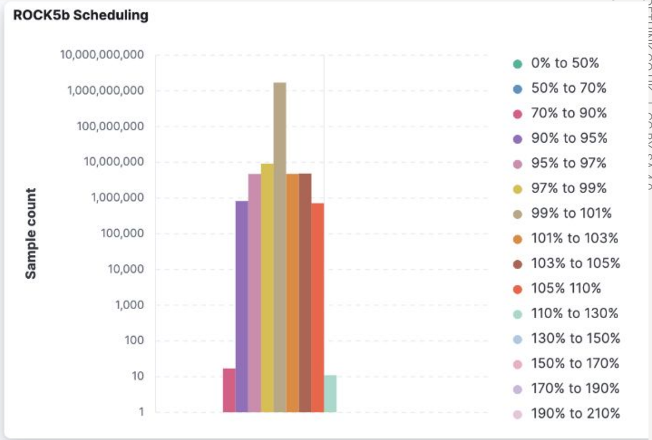

TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.
TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.
TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.
TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.
TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.
TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.
TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.
TA-A_01: All sources for XYZ and tools are mirrored in our controlled environment
TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.
TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.
TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.
TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.
TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.
TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.
TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues
TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.
TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.
TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.
TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.
TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.
TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.
TA-A_15: Confidence in XYZ is measured based on results of analysis

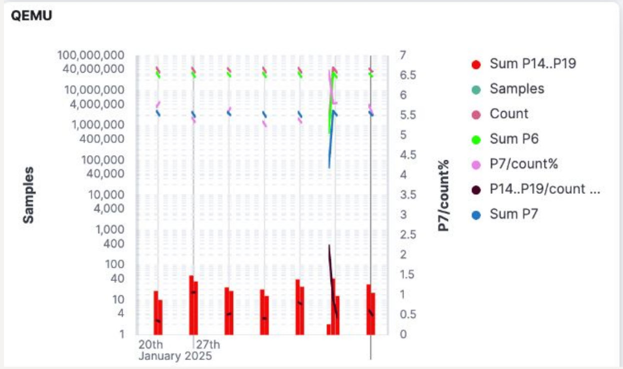
RESULTS: ongoing automated tests, including fault injection tests, with results captured and analysed over time

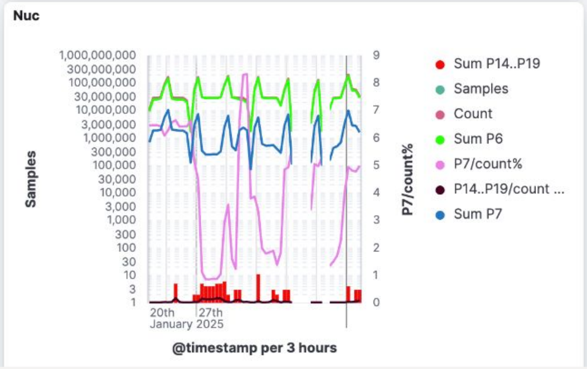
Digression… some results for Linux scheduling

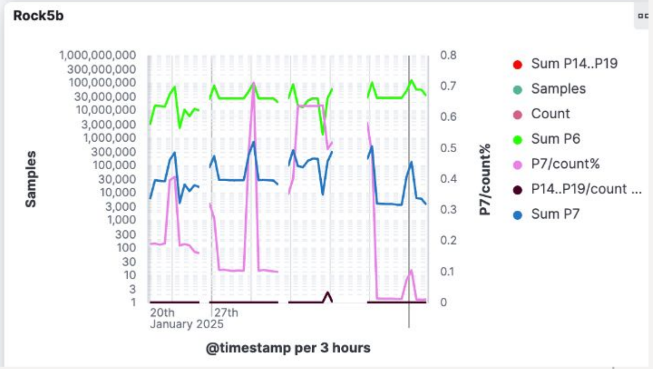


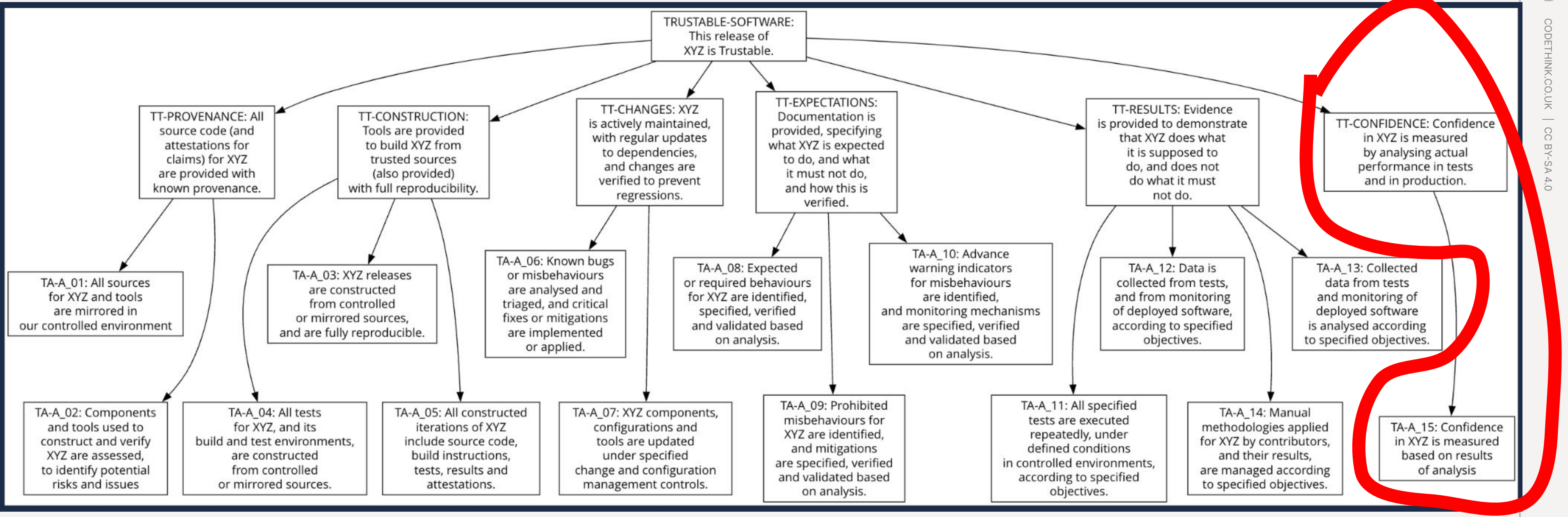











TRUSTABLE-SOFTWARE: This release of XYZ is Trustable.

TT-PROVENANCE: All source code (and attestations for claims) for XYZ are provided with known provenance.

TT-CONSTRUCTION: Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.

TT-CHANGES: XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.

TT-EXPECTATIONS: Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.

TT-RESULTS: Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.

TT-CONFIDENCE: Confidence in XYZ is measured by analysing actual performance in tests and in production.

TA-A_01: All sources for XYZ and tools are mirrored in our controlled environment

TA-A_03: XYZ releases are constructed from controlled or mirrored sources, and are fully reproducible.

TA-A_06: Known bugs or misbehaviours are analysed and triaged, and critical fixes or mitigations are implemented or applied.

TA-A_08: Expected or required behaviours for XYZ are identified, specified, verified and validated based on analysis.

TA-A_10: Advance warning indicators for misbehaviours are identified, and monitoring mechanisms are specified, verified and validated based on analysis.

TA-A_12: Data is collected from tests, and from monitoring of deployed software, according to specified objectives.

TA-A_13: Collected data from tests and monitoring of deployed software is analysed according to specified objectives.

TA-A_02: Components and tools used to construct and verify XYZ are assessed, to identify potential risks and issues

TA-A_04: All tests for XYZ, and its build and test environments, are constructed from controlled or mirrored sources.

TA-A_05: All constructed iterations of XYZ include source code, build instructions, tests, results and attestations.

TA-A_07: XYZ components, configurations and tools are updated under specified change and configuration management controls.

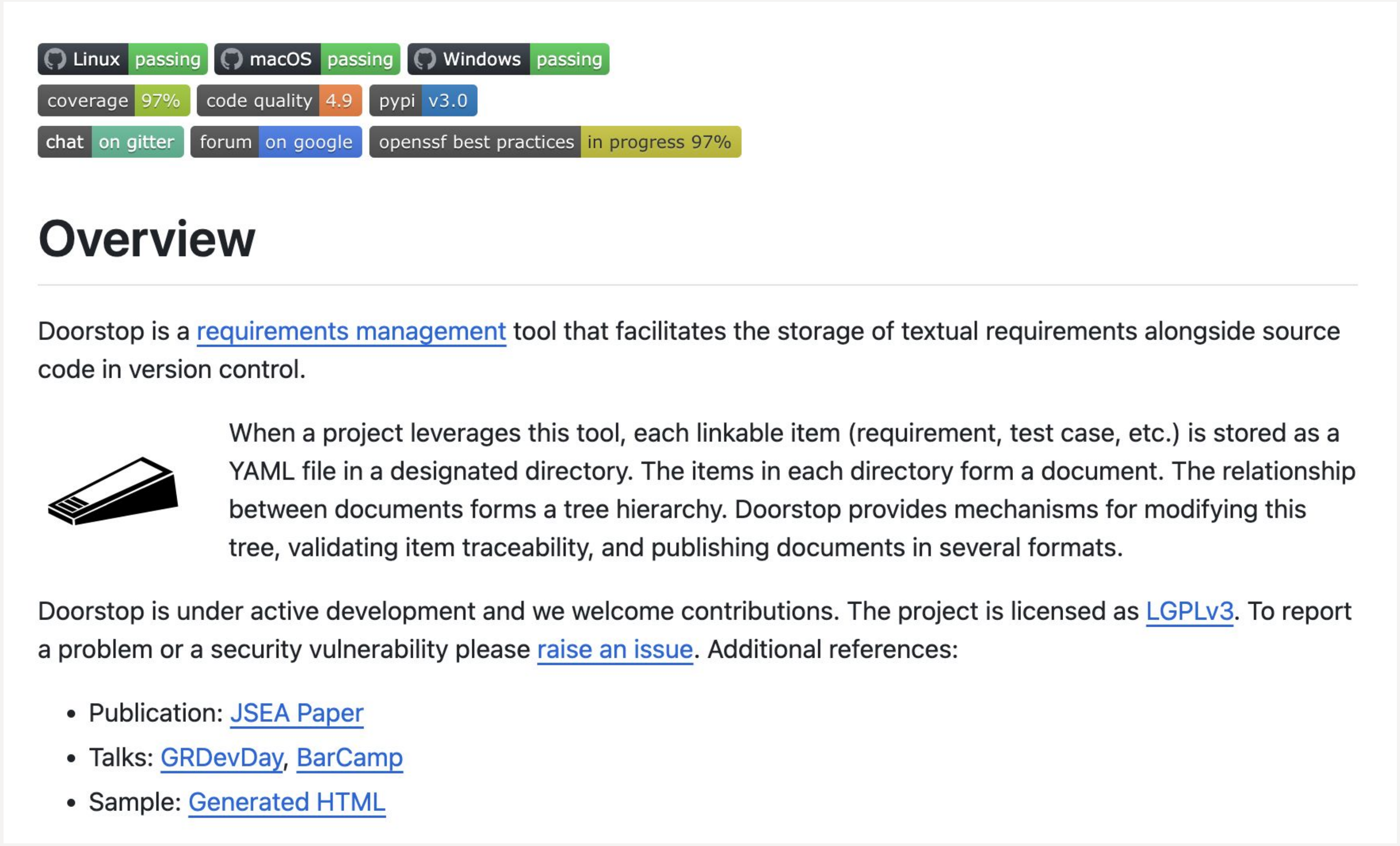
TA-A_09: Prohibited misbehaviours for XYZ are identified, and mitigations are specified, verified and validated based on analysis.

TA-A_11: All specified tests are executed repeatedly, under defined conditions in controlled environments, according to specified objectives.

TA-A_14: Manual methodologies applied for XYZ by contributors, and their results, are managed according to specified objectives.

TA-A_15: Confidence in XYZ is measured based on results of analysis

CONFIDENCE: assess **all of the evidence**, and report confidence scores https://github.com/doorstop-dev/doorstop



Linux passing | macOS passing | Windows passing

coverage 97% | code quality 4.9 | pypi v3.0

chat on gitter | forum on google | openssf best practices in progress 97%

Overview

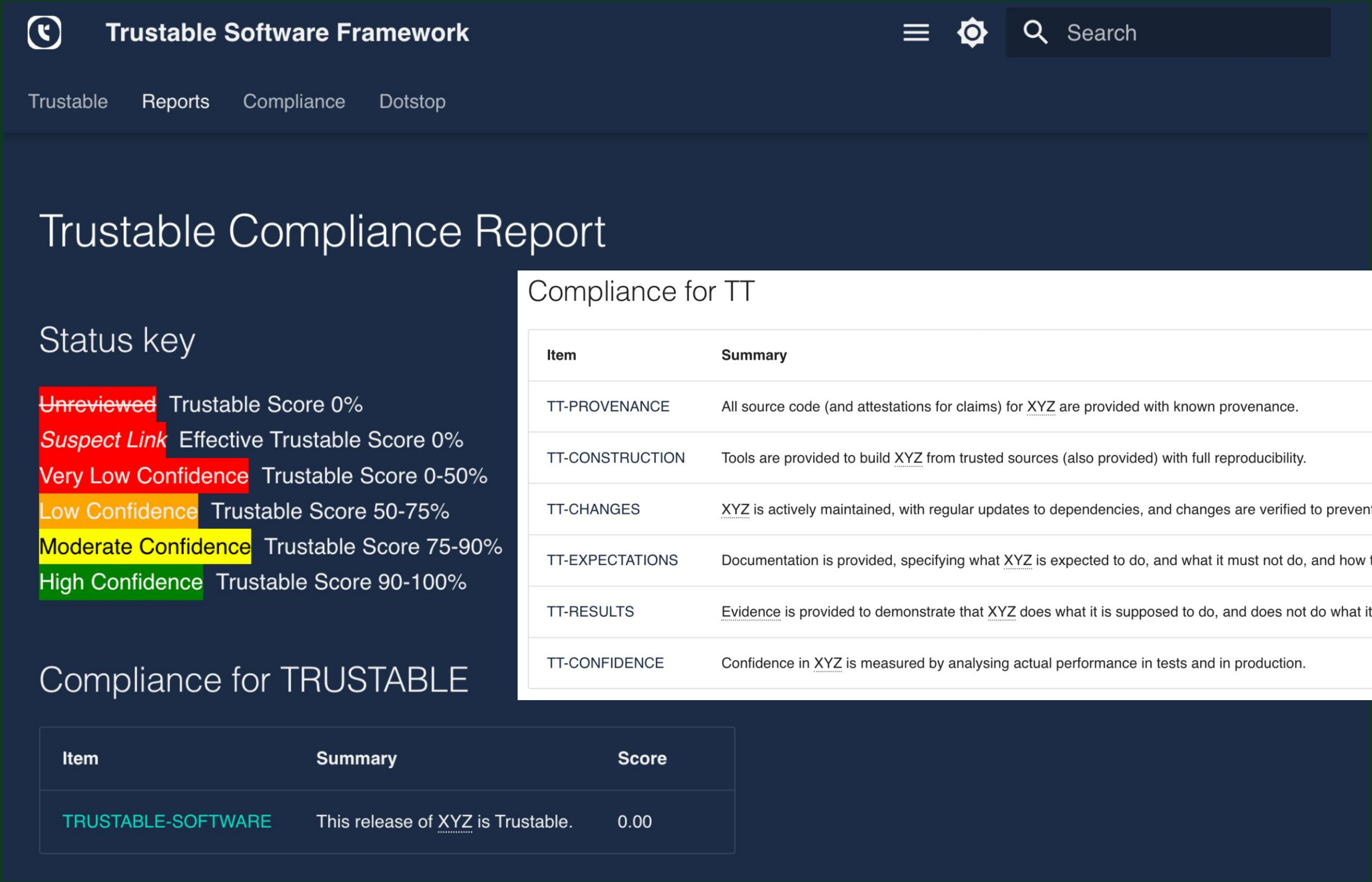
Doorstop is a requirements management tool that facilitates the storage of textual requirements alongside source code in version control.

When a project leverages this tool, each linkable item (requirement, test case, etc.) is stored as a YAML file in a designated directory. The items in each directory form a document. The relationship between documents forms a tree hierarchy. Doorstop provides mechanisms for modifying this tree, validating item traceability, and publishing documents in several formats.

Doorstop is under active development and we welcome contributions. The project is licensed as LGPLv3. To report a problem or a security vulnerability please raise an issue. Additional references:

- Publication: JSEA Paper
- Talks: GRDevDay, BarCamp
- Sample: Generated HTML

https://codethinklabs.gitlab.io/trustable/trustable/doorstop/trustable_report_for_TSF.html



Trustable Software Framework

Trustable Reports Compliance Dotstop

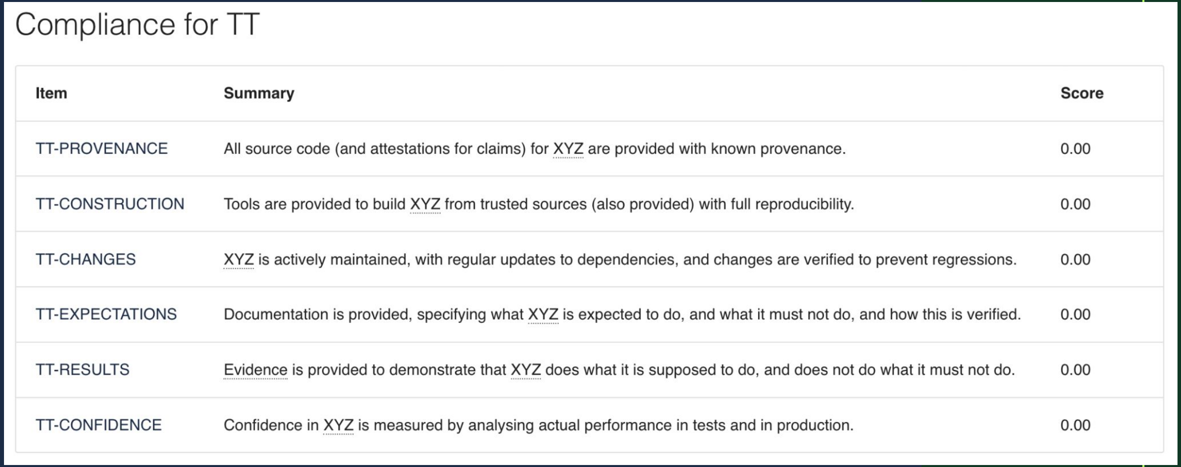
Trustable Compliance Report

Status key

~~Unreviewed~~	Trustable Score 0%
Suspect Link	Effective Trustable Score 0%
Very Low Confidence	Trustable Score 0-50%
Low Confidence	Trustable Score 50-75%
Moderate Confidence	Trustable Score 75-90%
High Confidence	Trustable Score 90-100%

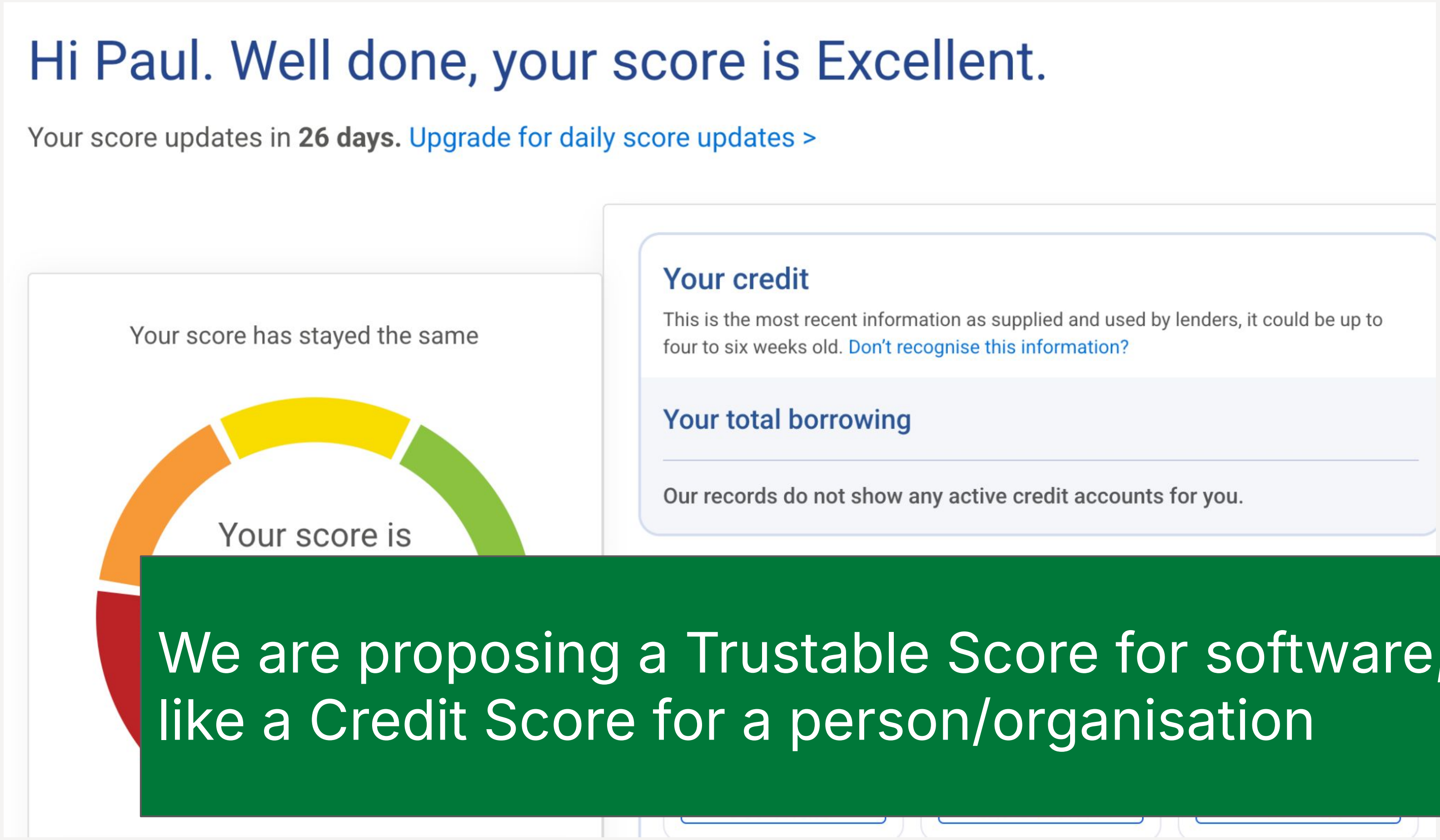
Compliance for TRUSTABLE

Item	Summary	Score
TRUSTABLE-SOFTWARE	This release of XYZ is Trustable.	0.00



Compliance for TT

Item	Summary	Score
TT-PROVENANCE	All source code (and attestations for claims) for XYZ are provided with known provenance.	0.00
TT-CONSTRUCTION	Tools are provided to build XYZ from trusted sources (also provided) with full reproducibility.	0.00
TT-CHANGES	XYZ is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.	0.00
TT-EXPECTATIONS	Documentation is provided, specifying what XYZ is expected to do, and what it must not do, and how this is verified.	0.00
TT-RESULTS	Evidence is provided to demonstrate that XYZ does what it is supposed to do, and does not do what it must not do.	0.00
TT-CONFIDENCE	Confidence in XYZ is measured by analysing actual performance in tests and in production.	0.00



Hi Paul. Well done, your score is Excellent.

Your score updates in **26 days.** Upgrade for daily score updates >

Your score has stayed the same

Your score is

Your credit

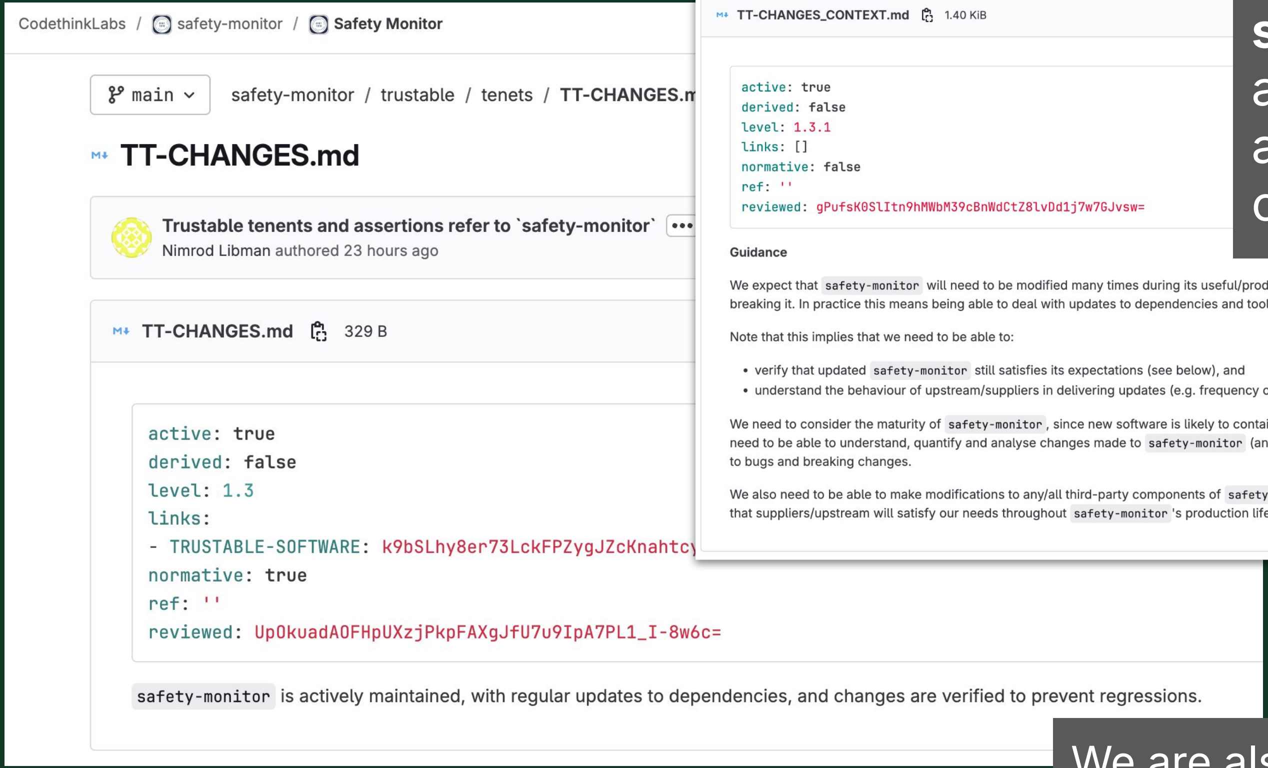
This is the most recent information as supplied and used by lenders, it could be up to four to six weeks old. Don't recognise this information?

Your total borrowing

Our records do not show any active credit accounts for you.

We are proposing a Trustable Score for software, like a Credit Score for a person/organisation

https://gitlab.com/CodethinkLabs/safety-monitor/safety-monitor



CodethinkLabs / safety-monitor / Safety Monitor

main

safety-monitor / trustable / tenets / TT-CHANGES.m

TT-CHANGES.md

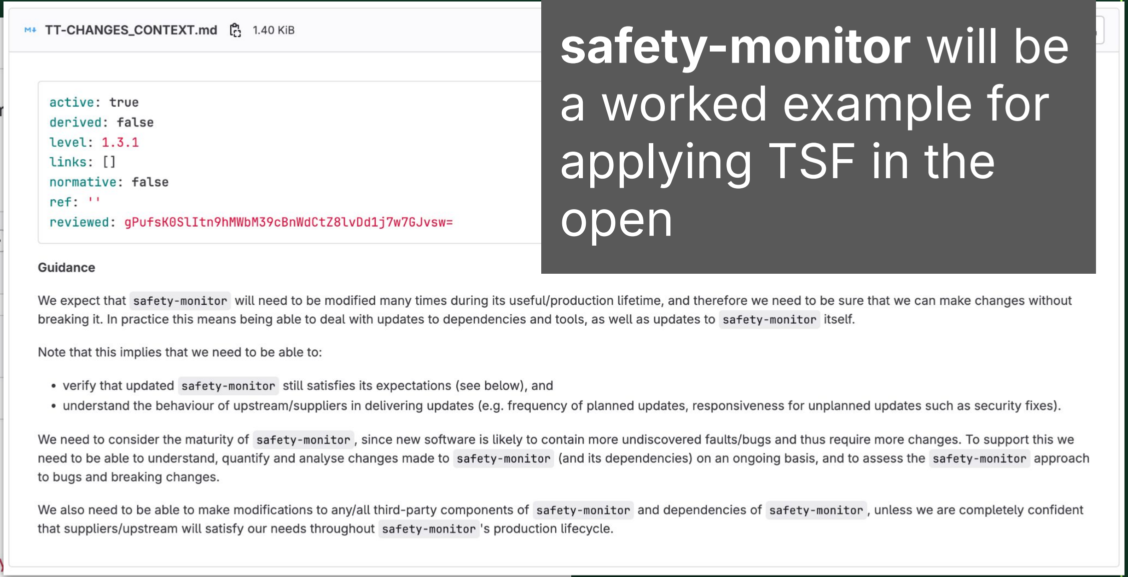
Trustable tenents and assertions refer to `safety-monitor`

Nimrod Libman authored 23 hours ago

TT-CHANGES.md 329 B

```
active: true
derived: false
level: 1.3
links:
- TRUSTABLE-SOFTWARE: k9bSLhy8er73LckFPZygJZcKnahtcy
normative: true
ref: ''
reviewed: UpOkuadAOFHpUXzjPkpFAXgJfU7u9IpA7PL1_I-8w6c=
```

`safety-monitor` is actively maintained, with regular updates to dependencies, and changes are verified to prevent regressions.



TT-CHANGES_CONTEXT.md 1.40 KiB

```
active: true
derived: false
level: 1.3.1
links: []
normative: false
ref: ''
reviewed: gPufsK0SlItn9hMWbM39cBnWdCtZ8lvDd1j7w7GJvsw=
```

Guidance

We expect that `safety-monitor` will need to be modified many times during its useful/production lifetime, and therefore we need to be sure that we can make changes without breaking it. In practice this means being able to deal with updates to dependencies and tools, as well as updates to `safety-monitor` itself.

Note that this implies that we need to be able to:

- verify that updated `safety-monitor` still satisfies its expectations (see below), and
- understand the behaviour of upstream/suppliers in delivering updates (e.g. frequency of planned updates, responsiveness for unplanned updates such as security fixes).

We need to consider the maturity of `safety-monitor`, since new software is likely to contain more undiscovered faults/bugs and thus require more changes. To support this we need to be able to understand, quantify and analyse changes made to `safety-monitor` (and its dependencies) on an ongoing basis, and to assess the `safety-monitor` approach to bugs and breaking changes.

We also need to be able to make modifications to any/all third-party components of `safety-monitor` and dependencies of `safety-monitor`, unless we are completely confident that suppliers/upstream will satisfy our needs throughout `safety-monitor`'s production lifecycle.

safety-monitor will be a worked example for applying TSF in the open

We are also hoping to apply the approach to some of the Eclipse SDV projects

Takeaways...

- fully free and open source - Eclipse project
- automatable (mostly) - intended for use in CICD
- designed to complement established processes
- applicable for existing software, including FLOSS
- extensible e.g. as a basis for mapping to standards
- maybe useful for “manufacturers + stewards” as a basis for driving towards CRA compliance?

Takeaways...

- fully free and open source - Eclipse project
- automatable (mostly) - intended for use in CICD
- designed to complement established processes
- applicable for existing software, including FLOSS
- extensible e.g. as a basis for mapping to standards
- maybe useful for "manufacturers + stewards" as a basis for driving towards CRA compliance?

Lots of work still to do... please help:

- critique the model and the approach, help us increase rigour and confidence in what scores <u>are</u>
- apply trustable scoring to your project, help us improve the method, documentation and correlation between scores and project outcomes

Thank You.

paul.sherwood@codethink.co.uk

https://lists.trustable.io/cgi-bin/mailman/listinfo/trustable-software

Codethink LTD

3rd Floor Dale House,
35 Dale Street,
MANCHESTER,
M1 2HF
United Kingdom