THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Developing a Community-Driven Standard for Open Source Software Quality

Philipp Ahmann (ETAS),
Gabriele Paoloni (Red Hat)

ELISA
Enabling Linux in Safety Applications

Introduction

ELISA Enabling Linux in Safety Applications

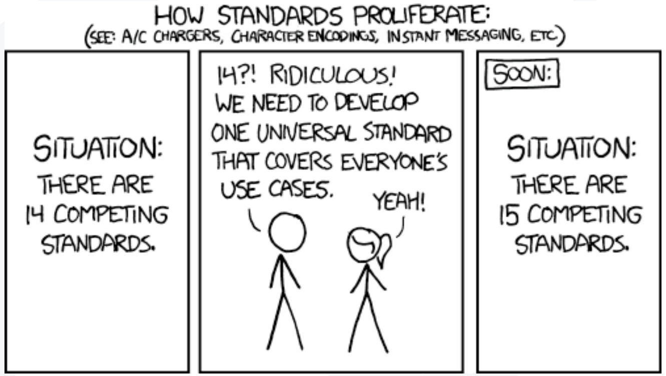
THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

Open Source Good Practices - Yet Another Standard

- Standards are based on v-model
- No OSS project is strictly following v-model. 😱

- ASPICE or CMMI are main argumentation for quality management in Automotive
- Safety standards require a quality baseline to support safety claims.

- No existing quality standard matches established open source development practices! (code-centric, CI driven and agile focus)


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/

Open Source Good Practices - Goal

The goal of this project is to evaluate and document established open source development best practices

&

to provide an assessment guide for the user to rate the quality of open source projects.

Photo by Paul Skorupskas on Unsplash

ELISA
Enabling Linux in Safety Applications

Open Source Good Practices - Overview

Phases

1. Determination of status quo
2. Definition of practices
3. Assessment of pilot projects

Contribution

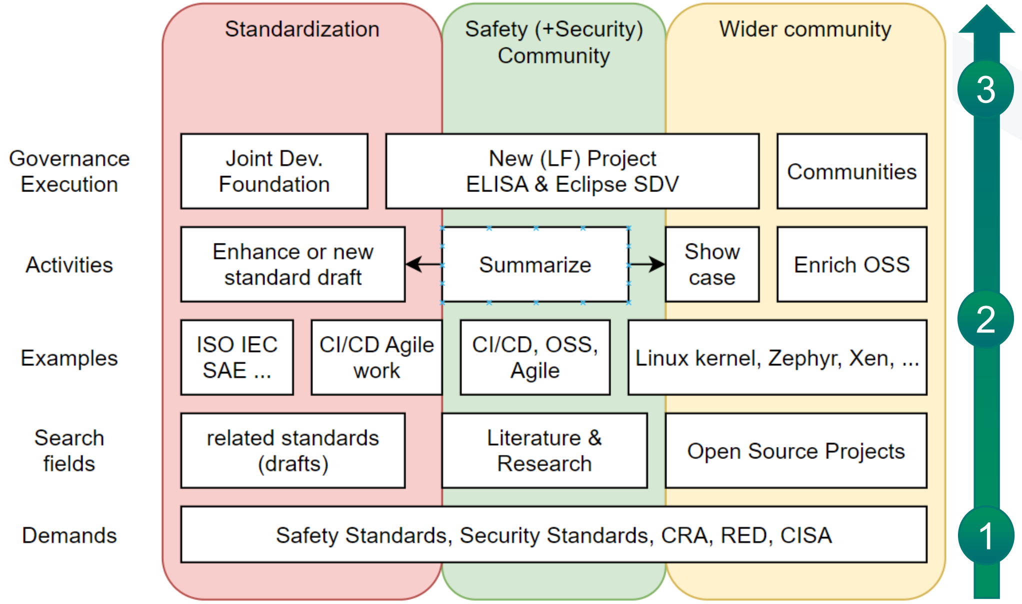
- Academia, Public sector, OSS communities
- Industries: Medical, Robotics, Avionics, Automotive, Railway, Automation, … (SME to industry leaders)

Funding

- Funding to be clarified (PfP or by members?)

Reach out, if you are interested in this effort.

Press Release & Survey under preparation



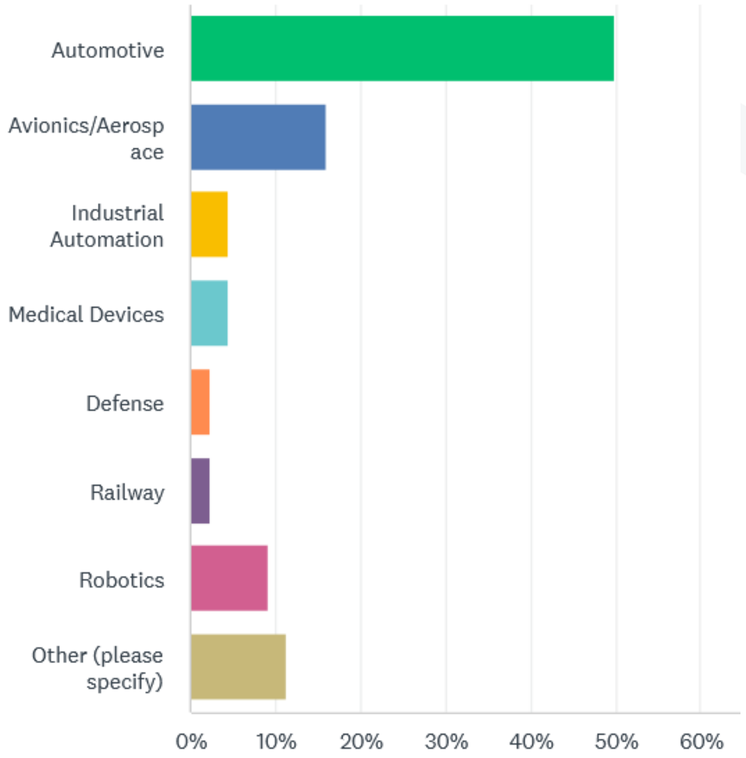
Survey Result Insights

ELISA
Enabling Linux in Safety Applications

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

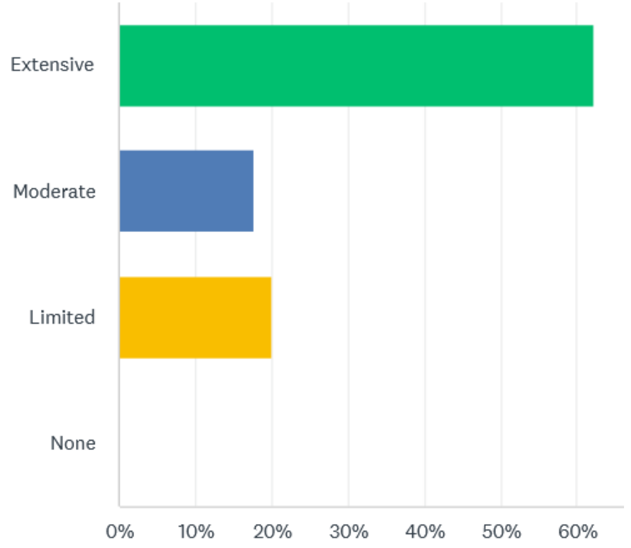
Some Numbers

- ~84% responses from Industry/Company
  - Others: Academia, Pubic Sector, Open Source, Freelancer, …
- 50% from Automotive
- ~9 - 16% from Robotics and Aerospace

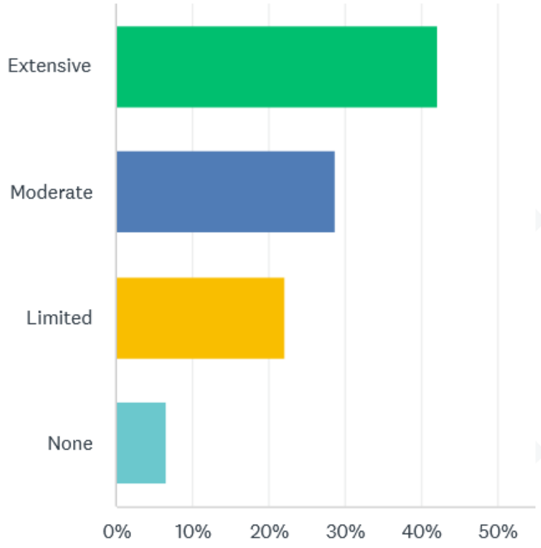


Existing Experience and Expertise

Your experience with…
Open Source Software

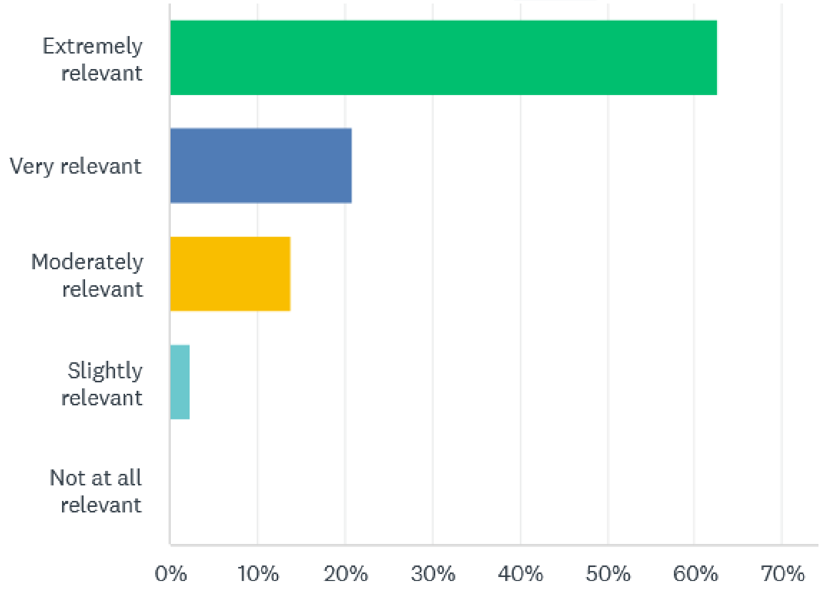


…
Quality management systems



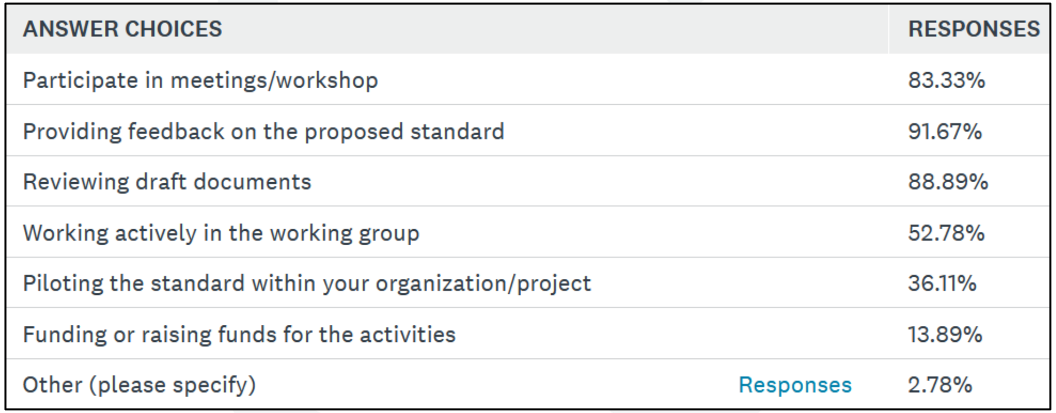
Relevance of Quality Standards

How relevant is a „Quality Standard based on Open Source Best Practices" to your work or interest?



Areas of Participation

- Most respondents prefer to review or participate in meetings/workshop
- 50%+ show interest in actively engaging in the working group (first meeting Friday 20th)
- 35%+ could imagine to pilot the standard



ANSWER CHOICES		RESPONSES
Participate in meetings/workshop		83.33%
Providing feedback on the proposed standard		91.67%
Reviewing draft documents		88.89%
Working actively in the working group		52.78%
Piloting the standard within your organization/project		36.11%
Funding or raising funds for the activities		13.89%
Other (please specify)	Responses	2.78%

ELISA
Enabling Linux in Safety Applications

Biggest Challenges (Extract)

- Automating the process of checking for conformance with standards followed
- No stardardized approach exist how to assess it
- alignment with safety critical standards
- source code size of the open source software
- Non-uniform architecture, coding style, approach to testing
- No standard way to measure quality of OSS
- Come up with robust standards that the don't require to be changed given the rapid evolution of technology -- both applications and tooling.
- The manifold of processes followed by different projects, and the low adherence to these processes by the project

ELISA
Enabling Linux in Safety Applications

One Motivational Challenge Statement:

Challenges are:

- requirements, architecture and design documents, as well as corresponding impact analysis.
- traceability
- coverage and test reports for automotive use case

[This is] in many projects just a matter of reformatting existing data.

Photo by Andrei Shiptenko on Unsplash

ELISA Enabling Linux in Safety Applications

Repeating Theme

- The biggest challenge may be to define what quality actually means?
- Many responses point to apply practices from safety integrity standards.

Important Criteria When Evaluating Quality of OSS

- Solid foundations, active community behind - Community health
- Development team, testing methodology
- Safety certification
- popularity of the project
- Popularity + Traceability (Including SBOM)?
- Normally that it is alive and receives proper fixes for CVEs.
- Modular Design, systematic development
- proven in use criteria... but having a strong review process
- documentation, usage, maintainability. Existing evidence of use in critical infratructure.

- Static analysis, conformance with coding style, regular maintenance, traceability from all changes to the public issue tracker, regular releases, code cleaning, documentation for all functions/classes/modules, good architecture, ease of installation, integration of tests into CI infrastructure, community size.
- How and when the system/software/hardware requirements (including interface specification and verification criteria) are documented, verified and released;
  what kind of methods/processes are defined, implemented and used.
  How and when architecture including external and internal interfaces (component <-> componet) including verification criteria created, documented, implemented, verified and released.
  How and when is the software the design created, implemented and documented (i.e. via doxygen), verification criteria;
  How are the verification process for each steps defined, implemented and completed.

Extracted Statement for „Evaluating OSS Quality“

- Who is backing the project?
  How confident can I be that support and maintenance will continue?
  And crucially, if something goes wrong - like losing access or support - what's my mitigation plan? (For instance, bringing it in-house and closing the source, as many companies end up doing.)

- It usually starts by a simple criteria like number of stars, forks, active updates, going to more complicated metrics like the community and code quality.

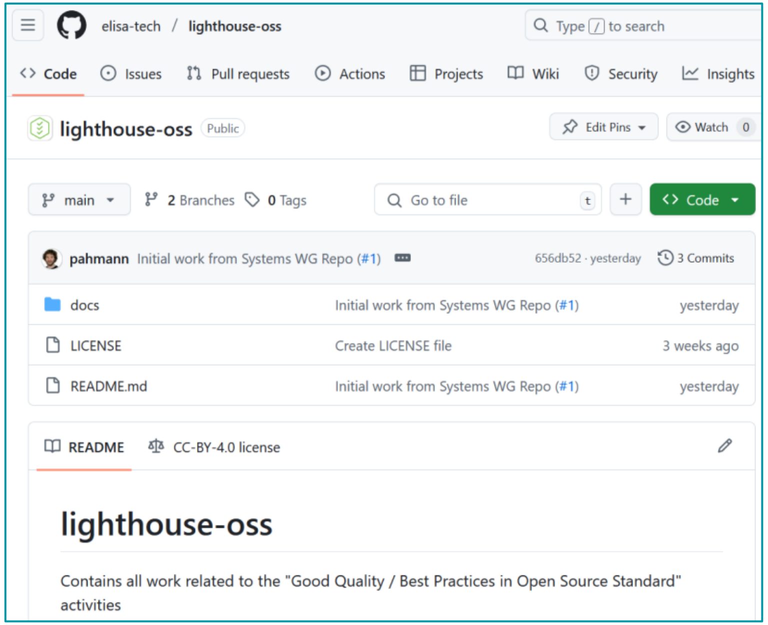
ELISA
Enabling Linux in Safety Applications

Lighthouse-OSS repository (where it comes to life)

ELISA
Enabling Linux in Safety Applications

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Where work will become visible: https://github.com/elisa-tech/lighthouse-oss


elisa-tech / lighthouse-oss

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights

lighthouse-oss Public

main · 2 Branches · 0 Tags

pahmann Initial work from Systems WG Repo (#1) · 656db52 · yesterday · 3 Commits

docs	Initial work from Systems WG Repo (#1)	yesterday
LICENSE	Create LICENSE file	3 weeks ago
README.md	Initial work from Systems WG Repo (#1)	yesterday

README · CC-BY-4.0 license

lighthouse-oss

Contains all work related to the "Good Quality / Best Practices in Open Source Standard" activities

All are welcome to contribute to this repository and prepare the path for documented and measurable best practices in open source development, which can serve as a blueprint for all type of modern software development (incl. commercial proprietary software).

Population ongoing. Contributions welcome!

ELISA Enabling Linux in Safety Applications

The proposed phases and key challenges

1) **Analyzing existing work!** ⬅️ **<u>We are here!</u>**
   a) Which OSS development practices exist?
   b) Which standards exist?

   } What does academia say?

   c) Do they define KPIs that increase quality (i.e. reduce the probability of systematic bug)?
2) **Identifying key performance indicators (KPIs)**
   a) Can we re-use KPIs from step 1)?
   b) Which Open Source project pilots should we consider to define KPIs?
   c) Which additional OSS projects shall we use to validate our KPIs?
3) **Creating a framework to evaluate how well OSS projects align in front of the objectives safety & security standards as well as regulations.**
   a) Are the defined KPIs suitable for supporting an automated framework?
   b) Can we come to a common framework impl. across (some) OSS projects?

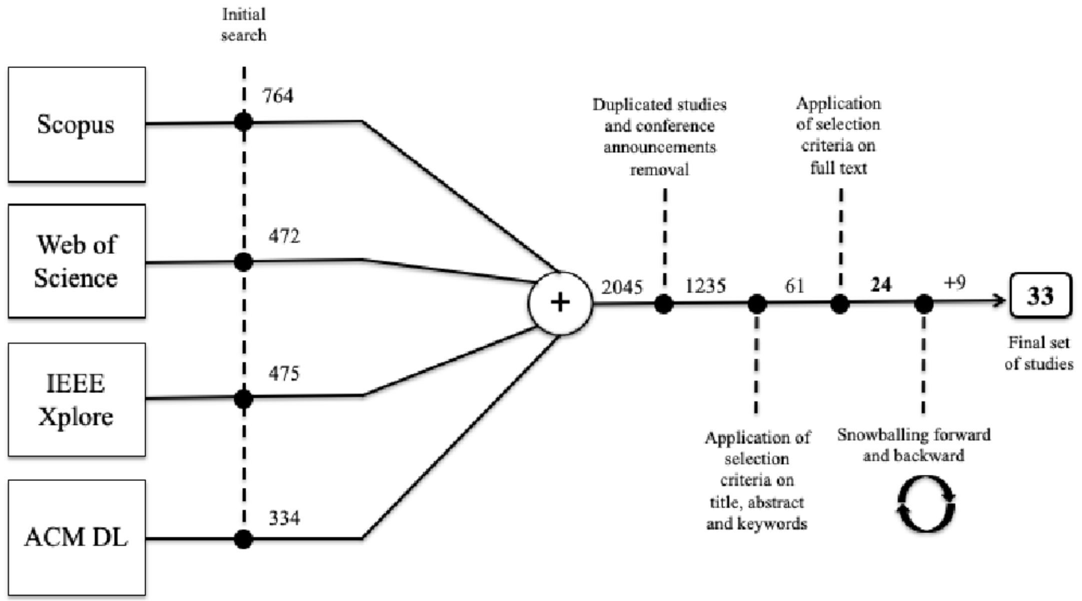
ELISA
Enabling Linux in Safety Applications

Starting point: Literature

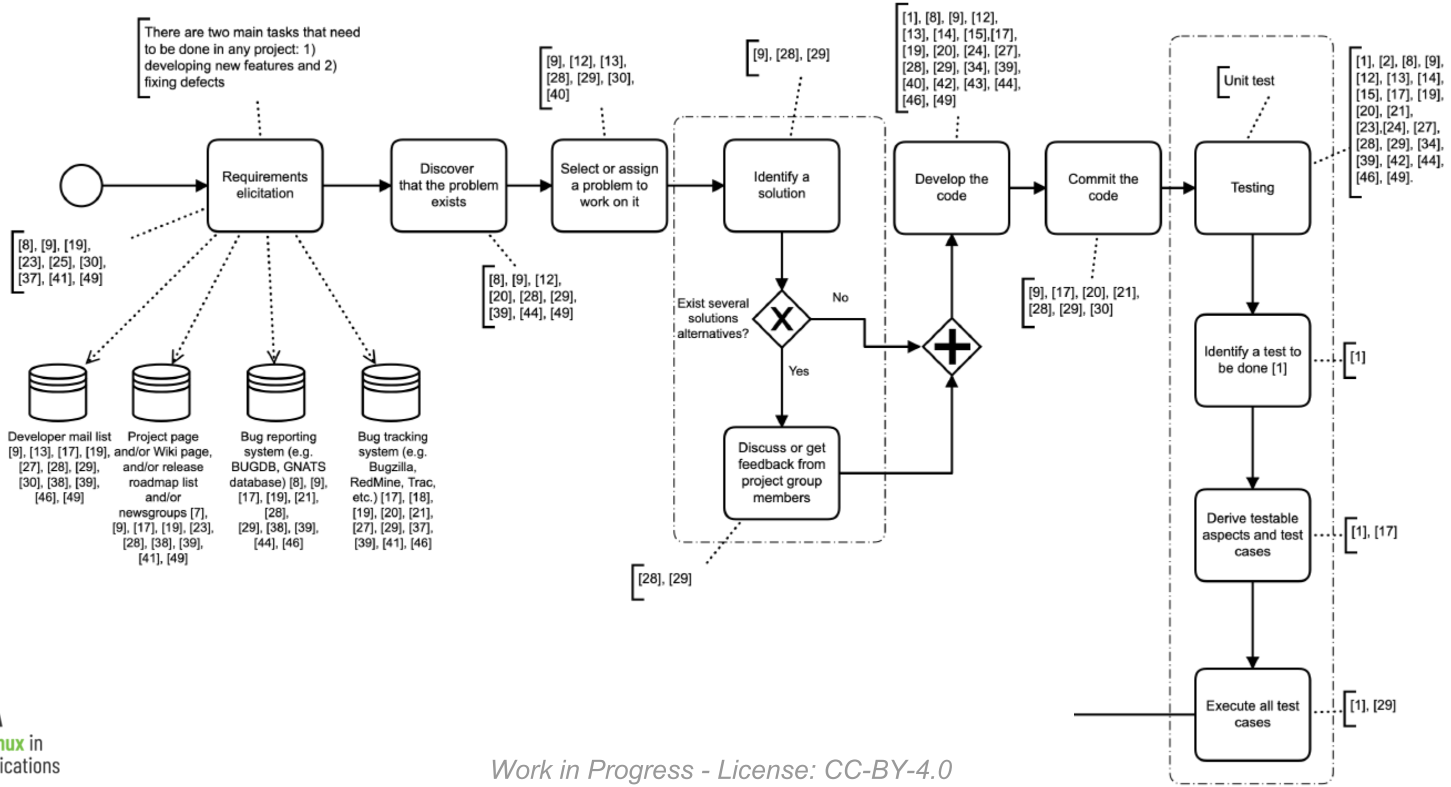
Example: Paper

- Open Source Software Development Process: A Systematic Review
- By: Bianca Minetto Napoleão, Fabio Petrillo, Sylvain Hallé (2020)
- Link: arXiv:2008.05015

Narrow down search field ➡️

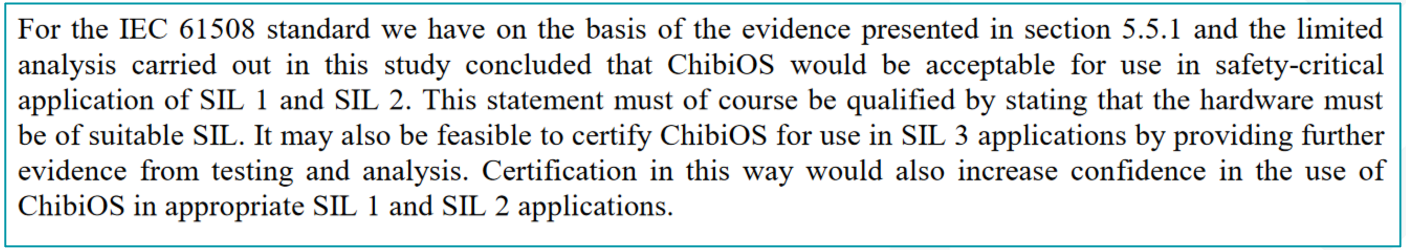

Initial search
Scopus
764
Web of Science
472
IEEE Xplore
475
ACM DL
334
+
2045
Duplicated studies and conference announcements removal
1235
61
Application of selection criteria on title, abstract and keywords
Application of selection criteria on full text
24
+9
Snowballing forward and backward
33
Final set of studies

Example: Paper

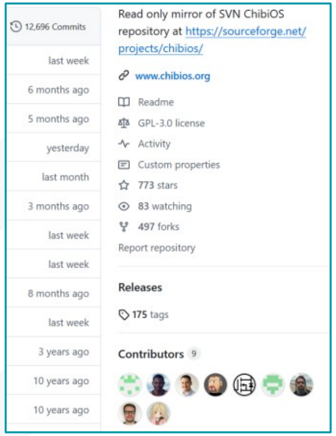

There are two main tasks that need to be done in any project: 1) developing new features and 2) fixing defects
Requirements elicitation
[8], [9], [19], [23], [25], [30], [37], [41], [49]
Developer mail list [9], [13], [17], [19], [27], [28], [29], [30], [38], [39], [46], [49]
Project page and/or Wiki page, and/or release roadmap list and/or newsgroups [7], [9], [17], [19], [23], [28], [38], [39], [41], [49]
Bug reporting system (e.g. BUGDB, GNATS database) [8], [9], [17], [19], [21], [28], [29], [38], [39], [44], [46]
Bug tracking system (e.g. Bugzilla, RedMine, Trac, etc.) [17], [18], [19], [20], [21], [27], [29], [37], [39], [41], [46]
Discover that the problem exists
[8], [9], [12], [20], [28], [29], [39], [44], [49]
[9], [12], [13], [28], [29], [30], [40]
Select or assign a problem to work on it
[9], [28], [29]
Identify a solution
Exist several solutions alternatives?
No
Yes
Discuss or get feedback from project group members
[28], [29]
[1], [8], [9], [12], [13], [14], [15],[17], [19], [20], [24], [27], [28], [29], [34], [39], [40], [42], [43], [44], [46], [49]
Develop the code
Commit the code
[9], [17], [20], [21], [28], [29], [30]
Unit test
Testing
[1], [2], [8], [9], [12], [13], [14], [15], [17], [19], [20], [21], [23],[24], [27], [28], [29], [34], [39], [42], [44], [46], [49].
Identify a test to be done [1]
[1]
Derive testable aspects and test cases
[1], [17]
Execute all test cases
[1], [29]

Example: Thesis

- Evaluation of open source operating systems for safety-critical applications
- By: Petter Sainio Berntsson (2017)
- Link: https://odr.chalmers.se/


For the IEC 61508 standard we have on the basis of the evidence presented in section 5.5.1 and the limited analysis carried out in this study concluded that ChibiOS would be acceptable for use in safety-critical application of SIL 1 and SIL 2. This statement must of course be qualified by stating that the hardware must be of suitable SIL. It may also be feasible to certify ChibiOS for use in SIL 3 applications by providing further evidence from testing and analysis. Certification in this way would also increase confidence in the use of ChibiOS in appropriate SIL 1 and SIL 2 applications.

- https://github.com/ChibiOS/ChibiOS


12,696 Commits
last week
6 months ago
5 months ago
yesterday
last month
3 months ago
last week
last week
8 months ago
last week
3 years ago
10 years ago
10 years ago

Read only mirror of SVN ChibiOS repository at https://sourceforge.net/projects/chibios/
www.chibios.org
Readme
GPL-3.0 license
Activity
Custom properties
773 stars
83 watching
497 forks
Report repository

Releases
175 tags

Contributors 9

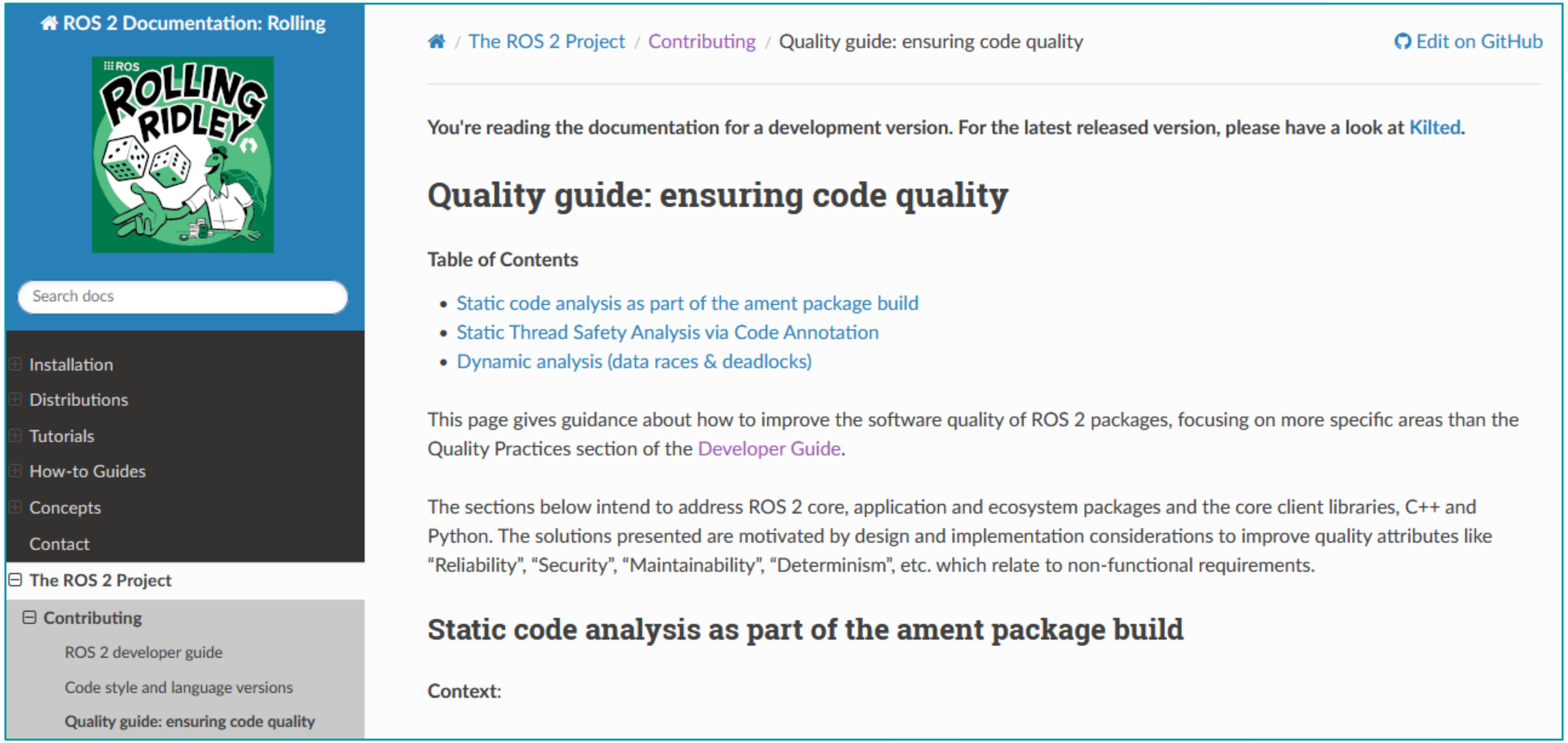
Literature Spot Check

- Relevant data sets and entry points available.
- Others have already done some part of the job.
- Literature may be older, incomplete, incorrect, or becomes outdated.
    - E.g. misconceptions due e.g. evolution of standards, tools (CI/CD)

- Far more papers and topics open to assess… 😉

A starting point:
Project

ROS 2 Quality Guide



ROS 2 Documentation: Rolling

Search docs

- Installation
- Distributions
- Tutorials
- How-to Guides
- Concepts
- Contact
- The ROS 2 Project
  - Contributing
    - ROS 2 developer guide
    - Code style and language versions
    - Quality guide: ensuring code quality

/ The ROS 2 Project / Contributing / Quality guide: ensuring code quality   Edit on GitHub

You're reading the documentation for a development version. For the latest released version, please have a look at Kilted.

Quality guide: ensuring code quality

Table of Contents

- Static code analysis as part of the ament package build
- Static Thread Safety Analysis via Code Annotation
- Dynamic analysis (data races & deadlocks)

This page gives guidance about how to improve the software quality of ROS 2 packages, focusing on more specific areas than the Quality Practices section of the Developer Guide.

The sections below intend to address ROS 2 core, application and ecosystem packages and the core client libraries, C++ and Python. The solutions presented are motivated by design and implementation considerations to improve quality attributes like "Reliability", "Security", "Maintainability", "Determinism", etc. which relate to non-functional requirements.

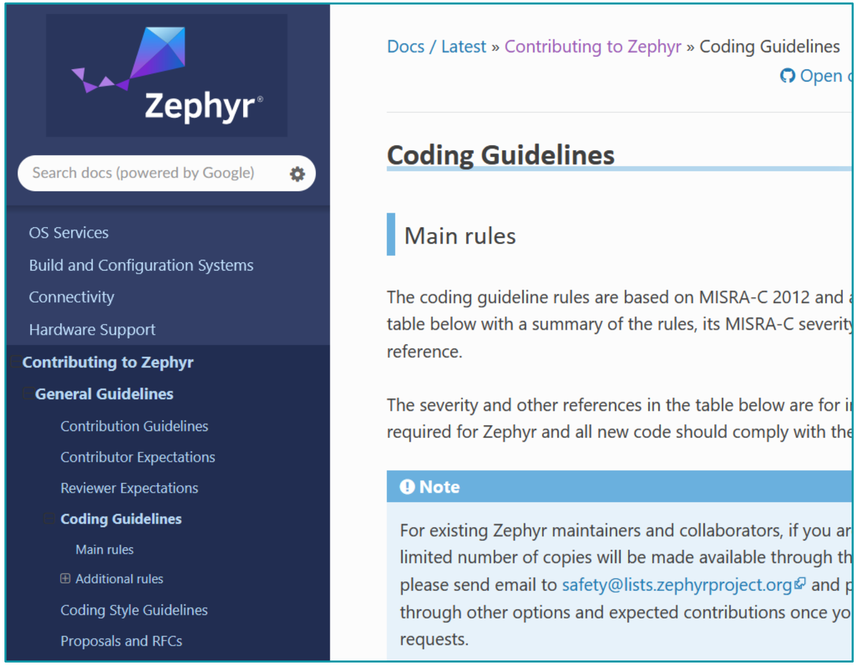
Static code analysis as part of the ament package build

Context:

https://docs.ros.org/en/rolling/The-ROS2-Project/Contributing/Quality-Guide.html

ELISA
Enabling Linux in Safety Applications

Zephyr Guidelines



Zephyr®

Search docs (powered by Google)

OS Services
Build and Configuration Systems
Connectivity
Hardware Support
Contributing to Zephyr
General Guidelines
Contribution Guidelines
Contributor Expectations
Reviewer Expectations
Coding Guidelines
Main rules
⊞ Additional rules
Coding Style Guidelines
Proposals and RFCs

Docs / Latest » Contributing to Zephyr » Coding Guidelines
Open

Coding Guidelines

Main rules

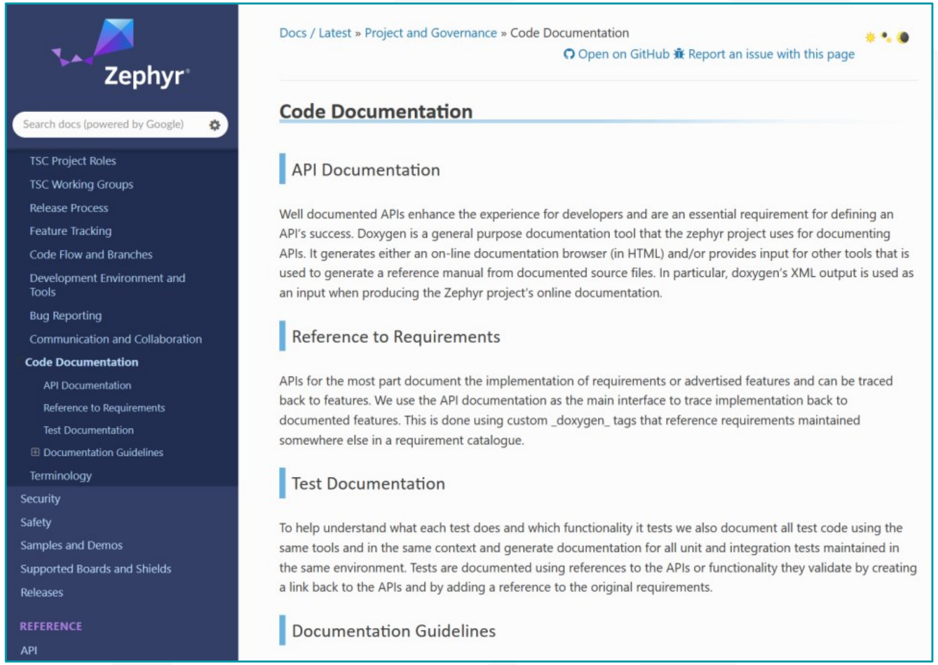
The coding guideline rules are based on MISRA-C 2012 and
table below with a summary of the rules, its MISRA-C severity
reference.

The severity and other references in the table below are for i
required for Zephyr and all new code should comply with the

Note

For existing Zephyr maintainers and collaborators, if you ar
limited number of copies will be made available through th
please send email to safety@lists.zephyrproject.org and p
through other options and expected contributions once yo
requests.

https://docs.zephyrproject.org/latest/contribute/coding_guidelines/index.html



Zephyr®

Search docs (powered by Google)

TSC Project Roles
TSC Working Groups
Release Process
Feature Tracking
Code Flow and Branches
Development Environment and Tools
Bug Reporting
Communication and Collaboration
Code Documentation
API Documentation
Reference to Requirements
Test Documentation
⊞ Documentation Guidelines
Terminology
Security
Safety
Samples and Demos
Supported Boards and Shields
Releases
REFERENCE
API

Docs / Latest » Project and Governance » Code Documentation
Open on GitHub Report an issue with this page

Code Documentation

API Documentation

Well documented APIs enhance the experience for developers and are an essential requirement for defining an API's success. Doxygen is a general purpose documentation tool that the zephyr project uses for documenting APIs. It generates either an on-line documentation browser (in HTML) and/or provides input for other tools that is used to generate a reference manual from documented source files. In particular, doxygen's XML output is used as an input when producing the Zephyr project's online documentation.

Reference to Requirements

APIs for the most part document the implementation of requirements or advertised features and can be traced back to features. We use the API documentation as the main interface to trace implementation back to documented features. This is done using custom _doxygen_ tags that reference requirements maintained somewhere else in a requirement catalogue.

Test Documentation

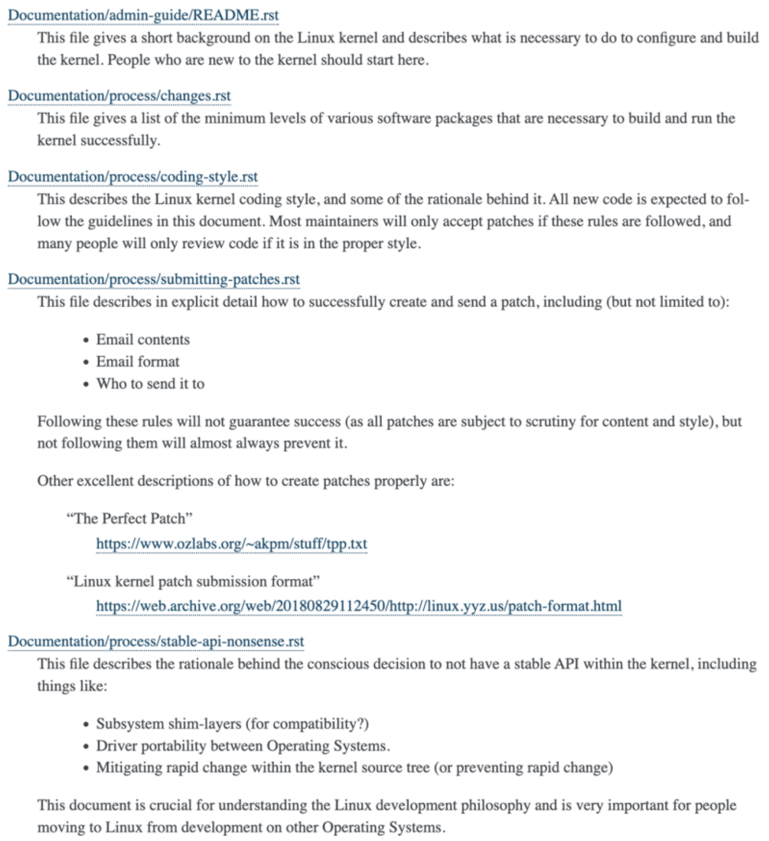
To help understand what each test does and which functionality it tests we also document all test code using the same tools and in the same context and generate documentation for all unit and integration tests maintained in the same environment. Tests are documented using references to the APIs or functionality they validate by creating a link back to the APIs and by adding a reference to the original requirements.

Documentation Guidelines

https://docs.zephyrproject.org/latest/project/documentation.html

ELISA Enabling Linux in Safety Applications

Linux Guidelines



Documentation/admin-guide/README.rst
This file gives a short background on the Linux kernel and describes what is necessary to do to configure and build the kernel. People who are new to the kernel should start here.

Documentation/process/changes.rst
This file gives a list of the minimum levels of various software packages that are necessary to build and run the kernel successfully.

Documentation/process/coding-style.rst
This describes the Linux kernel coding style, and some of the rationale behind it. All new code is expected to follow the guidelines in this document. Most maintainers will only accept patches if these rules are followed, and many people will only review code if it is in the proper style.

Documentation/process/submitting-patches.rst
This file describes in explicit detail how to successfully create and send a patch, including (but not limited to):

- Email contents
- Email format
- Who to send it to

Following these rules will not guarantee success (as all patches are subject to scrutiny for content and style), but not following them will almost always prevent it.

Other excellent descriptions of how to create patches properly are:

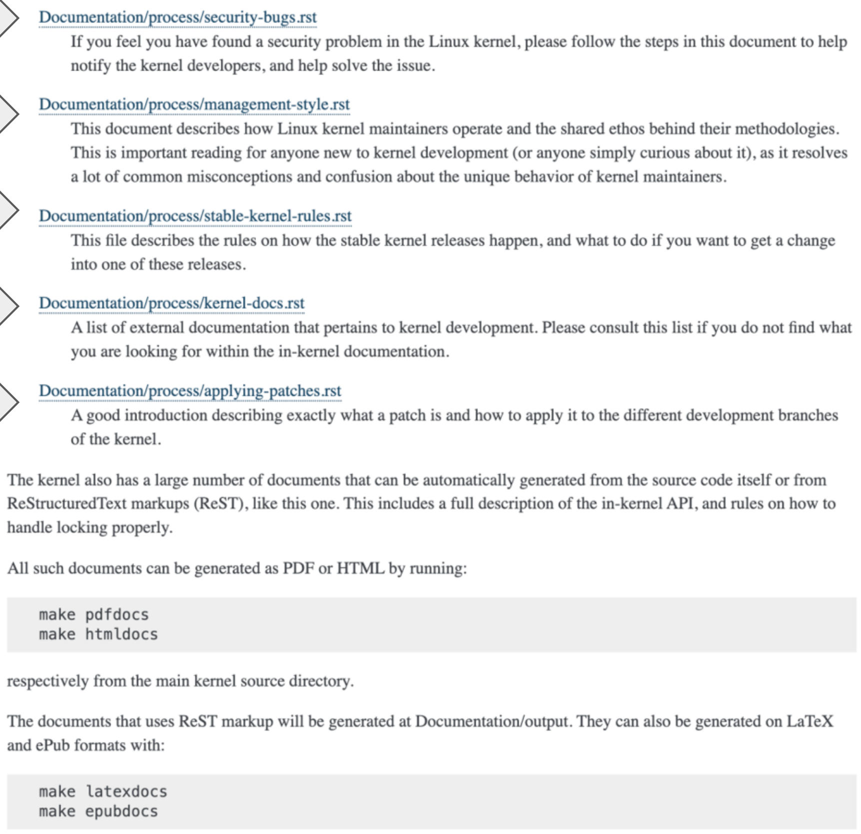
"The Perfect Patch"
https://www.ozlabs.org/~akpm/stuff/tpp.txt

"Linux kernel patch submission format"
https://web.archive.org/web/20180829112450/http://linux.yyz.us/patch-format.html

Documentation/process/stable-api-nonsense.rst
This file describes the rationale behind the conscious decision to not have a stable API within the kernel, including things like:

- Subsystem shim-layers (for compatibility?)
- Driver portability between Operating Systems.
- Mitigating rapid change within the kernel source tree (or preventing rapid change)

This document is crucial for understanding the Linux development philosophy and is very important for people moving to Linux from development on other Operating Systems.



Documentation/process/security-bugs.rst
If you feel you have found a security problem in the Linux kernel, please follow the steps in this document to help notify the kernel developers, and help solve the issue.

Documentation/process/management-style.rst
This document describes how Linux kernel maintainers operate and the shared ethos behind their methodologies. This is important reading for anyone new to kernel development (or anyone simply curious about it), as it resolves a lot of common misconceptions and confusion about the unique behavior of kernel maintainers.

Documentation/process/stable-kernel-rules.rst
This file describes the rules on how the stable kernel releases happen, and what to do if you want to get a change into one of these releases.

Documentation/process/kernel-docs.rst
A list of external documentation that pertains to kernel development. Please consult this list if you do not find what you are looking for within the in-kernel documentation.

Documentation/process/applying-patches.rst
A good introduction describing exactly what a patch is and how to apply it to the different development branches of the kernel.

The kernel also has a large number of documents that can be automatically generated from the source code itself or from ReStructuredText markups (ReST), like this one. This includes a full description of the in-kernel API, and rules on how to handle locking properly.

All such documents can be generated as PDF or HTML by running:

```
make pdfdocs
make htmldocs
```

respectively from the main kernel source directory.

The documents that uses ReST markup will be generated at Documentation/output. They can also be generated on LaTeX and ePub formats with:

```
make latexdocs
make epubdocs
```

https://docs.kernel.org/process/howto.html

ELISA
Enabling Linux in Safety Applications

Project Spot Check

- Examples exist.
- Practices are (even) documented.

- Treasure hunting (for best practices) is possible. 😉

Brainstorming / Feedback from last ELISA Workshop

ELISA Enabling Linux in Safety Applications

THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

Quick search which publications already exist…

- https://www.linkedin.com/advice/0/what-most-common-open-source-software-quality
- Thesis S. Tatschner: Towards a More Sustainable and Secure Software Tooling in Free/Libre Open Source Software Environments
- Tracking software cluster bombs: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4847570
- The process of quality assurance under open source software development https://ieeexplore.ieee.org/document/5958975
- Aspects of Software Quality Assurance in Open Source Software Projects: Two Case Studies from Apache Project https://ieeexplore.ieee.org/document/4301084
- Open Source software best practices and supply chain risk management on www.gov.uk
- MSR (Mining software repositories), ICSE (Int. Conference on Software Engineering), ICSME, ASE (Automated Software Engineering), FSE (Fundaments of Software Engineering)
- Julia Lawall papers
- Draft - RTCA DO-395 is an RTCA document that supplements DO-178C/ED-12C and DO-278A/ED-109A, focusing on the use of Commercial Off-the-Shelf (COTS) and/or Open Source Software (OSS) in airborne systems. It provides guidance on integrating COTS/OSS into software development processes and ensuring their compliance with airworthiness requirements.

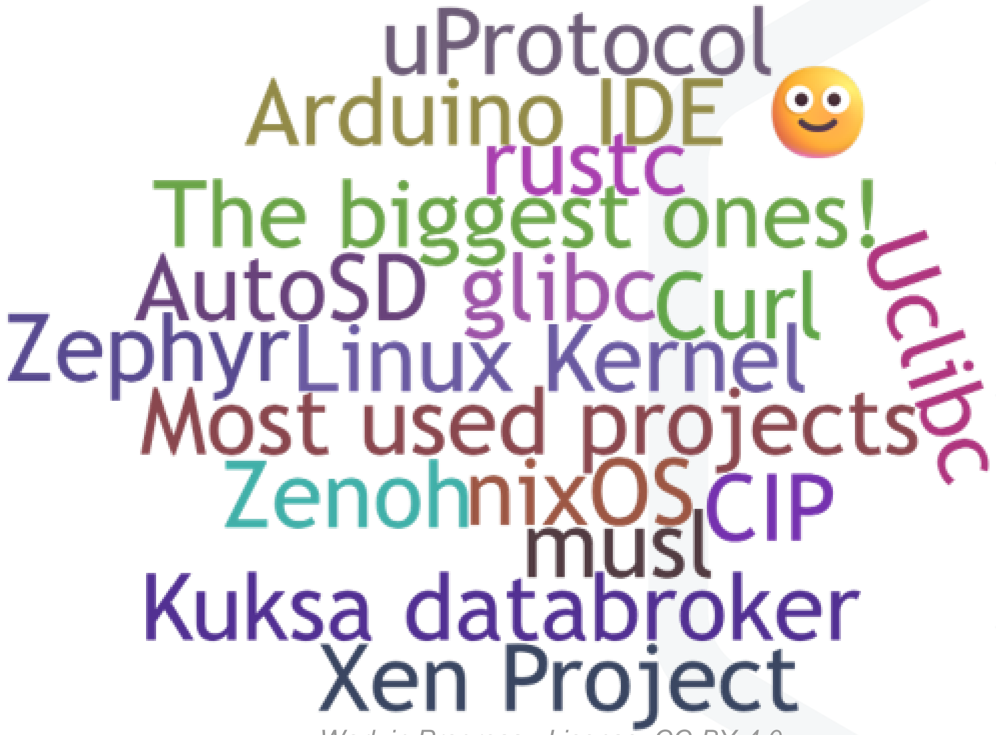
What could be related standards…

DOT/FAA/TC-15/27
ED-12C/ED-109A
SOTIF (ISO/PAS 21448)
EN 50128 ISO 5230
NPR 7150.2 ISO/IEC 12207
ISO 21434 Ece r137
ISO9001 DO 178C AS9100
ISO 17978 ISO PAS 8926 IEC 50716
ISO/SAE 21434
ISO 18974
MIL-STD0882
NASA-STD-8739.8B

Created with: https://wordcloud.timdream.org

ELISA
Enabling Linux in Safety Applications

We asked workshop participants: Which OSS projects should be considered?



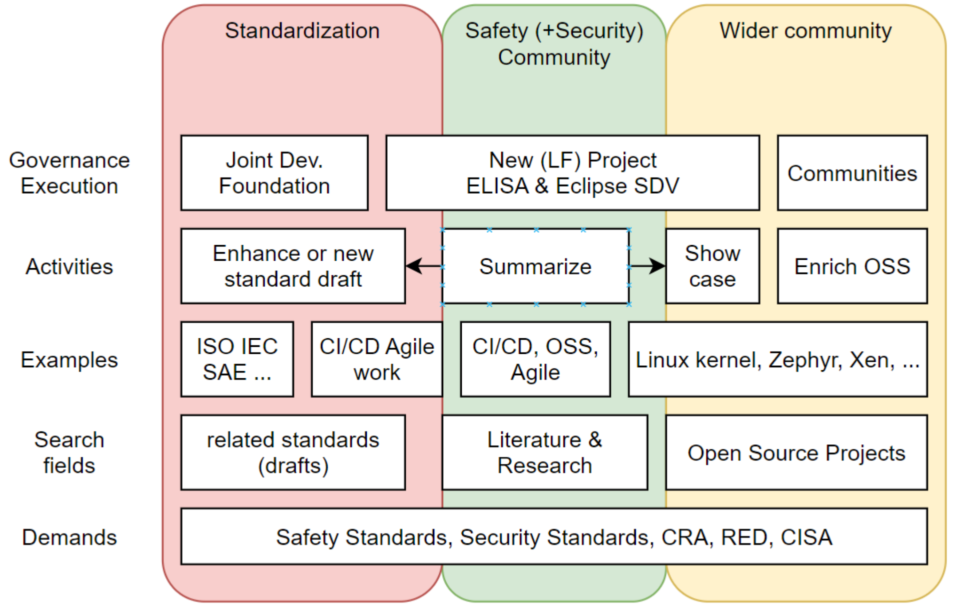
Created with: https://wordcloud.timdream.org

ELISA Enabling Linux in Safety Applications

Open Discussion!

Discussion points

- What (else) to consider?
- Are we missing something?
    - Communities?
    - Technology?
- Is it over-engineering?
- What are the biggest risks?
- Which skill sets do we need to make it a success?



ELISA Enabling Linux in Safety Applications

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA