

White Paper: Advancing Open Source Safety-Critical Systems

Written by Shuah Khan, Chair of the ELISA Project Technical Steering Committee and Linux Fellow at The Linux Foundation



SPDX-License-Identifier: CC-BY-4.0

This document is released under the Creative Commons Attribution 4.0 International License, available at <https://creativecommons.org/licenses/by/4.0/legalcode>. Pursuant to Section 5 of the license, please note that the following disclaimers apply (capitalized terms have the meanings set forth in the license).

To the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Assessing whether a system is safe requires understanding the system sufficiently. You have to understand what is happening in your system and how various modules and components interact. If the system depends on Linux, it is crucial to understand Linux within that system context and how Linux is used. Also, you have to know how the Linux kernel interacts with various hardware components in the system and the user-space running on top of the kernel. Only with a sound system understanding of the specific context and interactions between the applications, base open-source libraries, such as libc, and the kernel, can you adequately assess if the system is safe or not.

What are the challenges involved in running Linux in Safety-critical systems and building products based on Linux? The primary challenge is selecting Linux components and features that can be evaluated for safety and identifying gaps where more work is needed to evaluate safety sufficiently.

The ELISA project has taken on the challenge to make it easier for companies to build and certify Linux-based safety-critical applications by exploring potential methods to enable engineers to answer that question for their specific system.

Another aspect is understanding the limits. We at ELISA want you to understand that we:

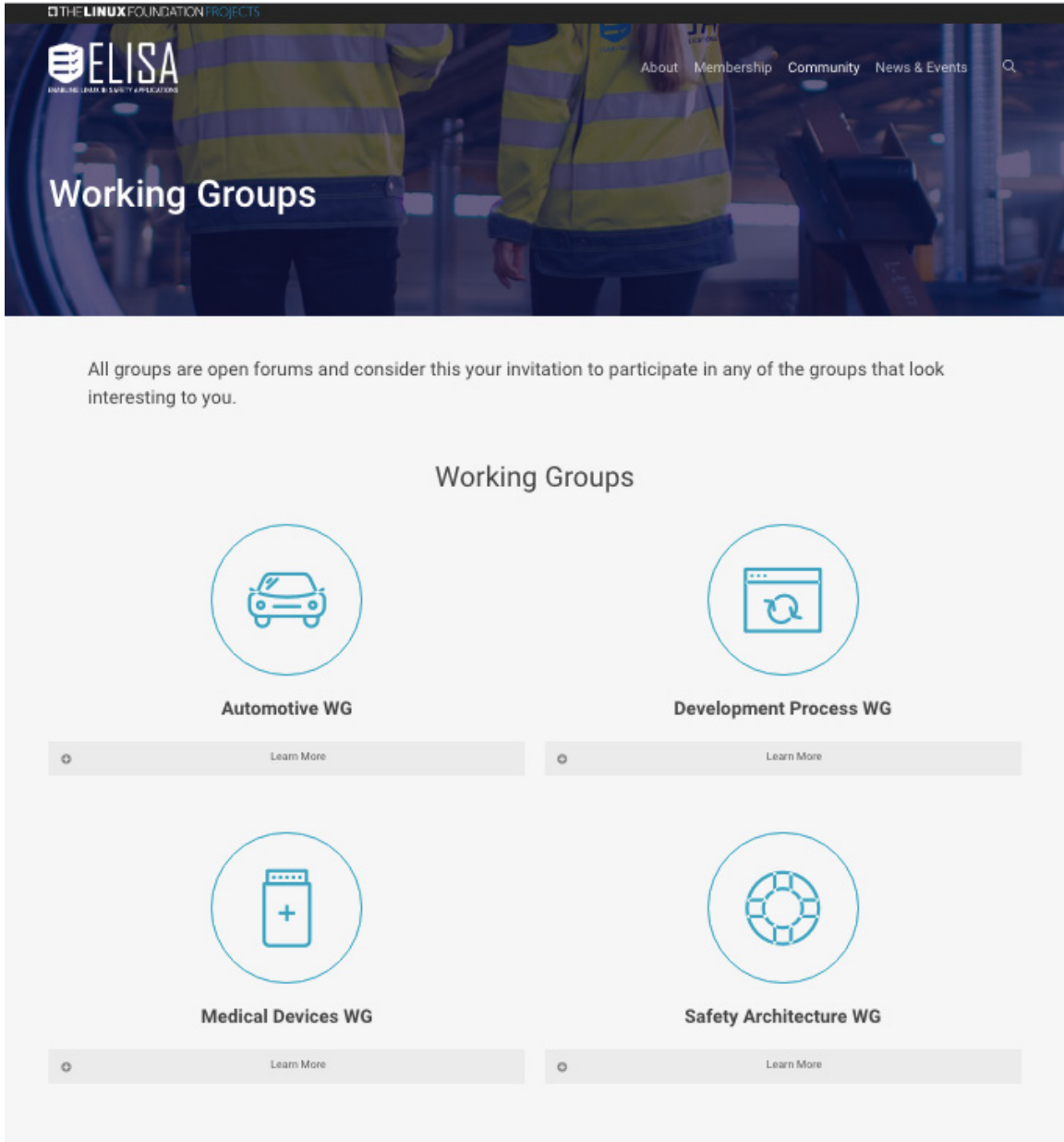
- cannot engineer your system to be safe
- cannot ensure that you know how to apply the described process and methods
- cannot create an out-of-tree Linux kernel for safety-critical applications, being mindful of the continuous process improvement argument
- cannot relieve you from your responsibilities, legal obligations, and liabilities.

As you might already know, Linux kernel releases are time-based, and releases come out once every 8-10 weeks. The Linux kernel community has its own set of development policies and processes that govern reviewing and accepting new content. It is a challenge to keep up with the development process for safety analysis and product support. We don't have all the answers; however, we at ELISA strive to provide a path forward as a home to like-minded peers to collaborate.

Please checkout the [ELISA Working Group](#) activities and join us to collaborate with us.

- Main Technical List: devel@lists.elisa.tech
- Technical Steering Committee: elisa.tech/about/tsc
- Kernel Development Process Working Group: development-process@lists.elisa.tech

- Safety Architecture Working Group: safety-architecture@lists.elisa.tech
- Medical Devices Working Group: medical-devices@lists.elisa.tech
- Automotive Working Group: automotive@lists.elisa.tech



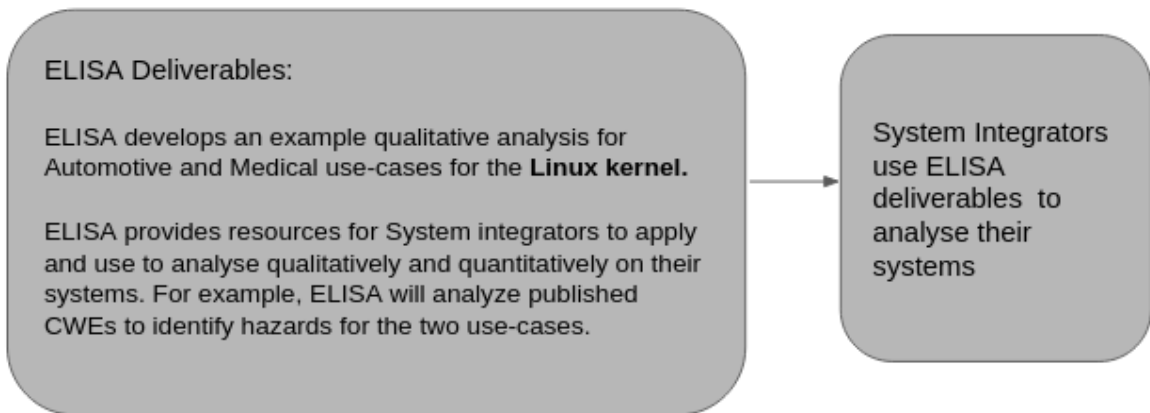
The screenshot shows the ELISA website's 'Working Groups' page. At the top, there is a navigation bar with 'About', 'Membership', 'Community', and 'News & Events'. The main heading is 'Working Groups'. Below this, a paragraph states: 'All groups are open forums and consider this your invitation to participate in any of the groups that look interesting to you.' The page features four group cards, each with an icon, a title, and a 'Learn More' button. The groups are: Automotive WG (car icon), Development Process WG (refresh icon), Medical Devices WG (pill icon), and Safety Architecture WG (globe icon).

Technical Strategy

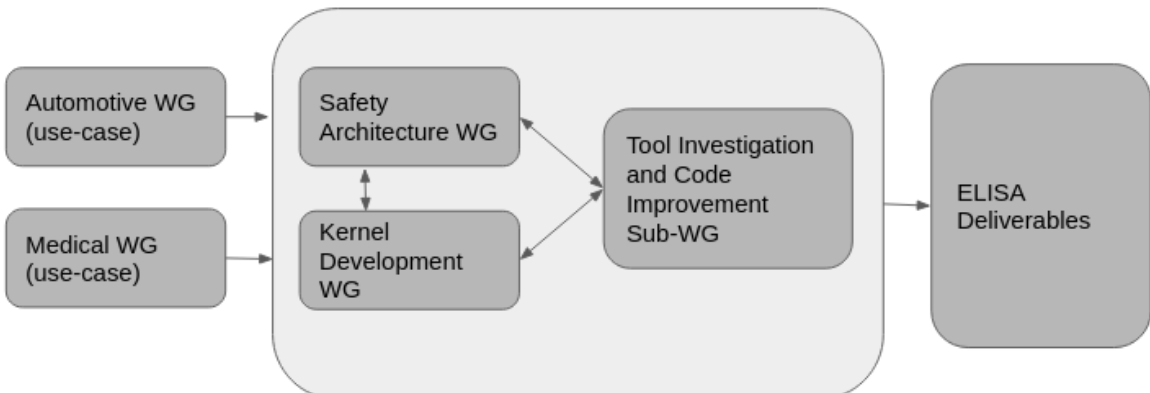
Let's talk about how our [Technical Charter](#) anchors us in defining strategy and deliverables based on Automotive and Medical use-cases.

What are ELISA deliverables?

- ELISA develops an example qualitative analysis for Automotive and Medical use-cases for the Linux kernel.
- ELISA provides resources for System integrators to apply and use to analyze qualitatively and quantitatively on their systems. For example, ELISA will analyze published CWEs to identify hazards for the two use-cases.
- System Integrators use ELISA deliverables to analyze their systems



ELISA Working Groups collaborate in developing a qualitative analysis of the Automotive and Medical use-cases for the Linux kernel.

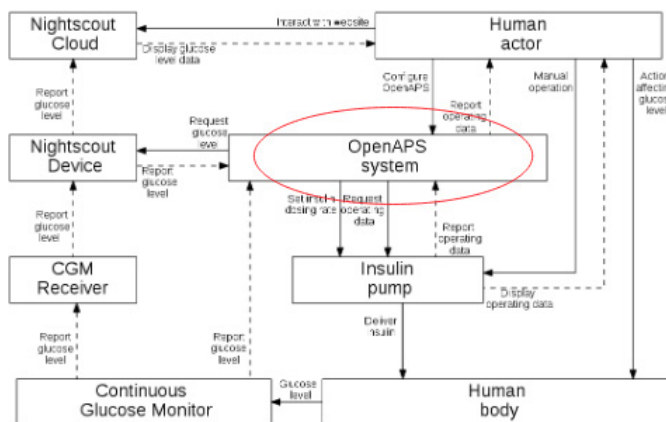


ELISA Technical Steering Committee is responsible for defining the technical strategy. The TSC is currently focused on the following activities with Automotive and Medical use-cases. We welcome other use-cases.

- Refining and finalizing the Technical Strategy
- Exploring how best to represent hazards and state what we tried.
- Defining clearly what ELISA is and is not
- Refining and finalizing executing on the Technical Strategy in WGs
- Deciding on our Publication Strategy (reviews and best practices for ELISA deliverables)

ELISA Medical Devices Working Group is focused on:

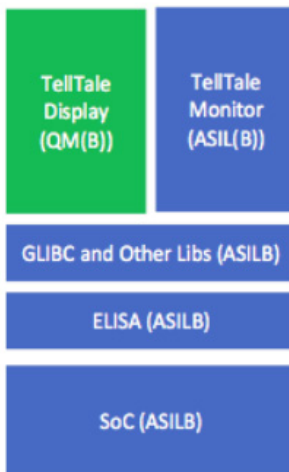
- Finishing Open Artificial Pancreas System ([OpenAPS](#)) analysis
- Reviewing our analysis of OpenAPS (#openAPS) with System-Theoretic Process Analysis ([STPA](#)) experts
- Presenting results of our analysis to the OpenAPS community for feedback.
- Publishing openAPS STPA analysis on Github and put it under version control.



- Use STPA (http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf)
- Take iterative approach, structure according to the handbook
- Next steps: create the next level diagram for openAPS component.
 - Algorithm
 - User space libs/runtime being active/needed
 - Linux syscalls & services
 - Devices/hardware interaction

ELISA Automotive Working Group currently focuses on analyzing automotive use cases with safety implications and collaborating with the [Automotive Grade Linux](#) project to leverage their knowledge in this space. In addition it is:

- Continuing work on and improve the telltale use case
- Consolidating the concept and demo application
- Furthering the concept and architecture refinement



TellTale Display:

- Receive a request to display a certain telltale
- Call the graphic lib routine to display the telltale
- The GPU renders the telltale
- The display engine will compose the framebuffer in memory

Telltale Monitor:

- Receive the same request to display a certain telltale
- Start a watchdog timer programmed with a known maximum acceptable delay to display the telltale
- Start reading back the framebuffer and calculates the CRC on the area where the telltale is expected
- If a match is found with the expected CRC within the watchdog time frame start another watchdog to monitor the telltale to be displayed for all the time it should be
- Keep reading the framebuffer and calculate the CRC till the watchdog elapse

ELISA Safety Architecture Working Group is set out to complete the definition of top-level safety requirements for the Kernel and identify associated entry points within the context of the above example Automotive use-case. The current focus is Automotive use-cases with a plan to include Medical use-cases. In addition, the WG is working on the following within the context of the Automotive use-case.

- Starting the safety analyses for the telltale safety app
- Consolidating and refining the qualification methodology using the telltale use case as the driver
- Starting the Freedom From Interference (FFI) analysis with Linux kernel as the primary focus and taking Non-Safety Related (NSR) user space processes into account. The FFI operates under the premise that a fault in a non or less safety-critical component doesn't lead to a fault in a more safety-critical component. FFI, as defined by ISO 26262, is the "absence of cascading failures between components that could lead to the violation of [some] safety requirement."

ELISA Kernel Development Process Working Group is currently engaged in completing the Linux kernel management process, validation process, and basic hazard classification for safety leveraging [Common Weakness Enumerations](#) (CWEs). Automotive and Medical use-cases are in scope for this work. So how do we do this? We:

- Define system characteristics, failure modes, and attributes of Linux kernel. As an example, Timing and Execution is an important system property to ensure a less safety-critical component doesn't block a more safety-critical component.
- Identify Kernel configurations and features essential to safety-critical systems

For example, **Timing and Execution** is an important system property that will ensure that a less safety-critical component doesn't block a more safety-critical component. [Resource Locking Problems](#) and [Improper Locking](#) related hazards in safety-critical software components could prevent and undermine this System property. It is important to detect and mitigate resource locking and improper locking problems during development and integration. This WG identifies and analyzes available detection and mitigation methods in the Linux kernel and provides resources to System integrators to apply and use on their systems.

The detection and mitigation methods available in the Linux kernel are Static and Dynamic analysis tools and Kernel debug configuration options that can be used to find and fix problems. [CONFIG_LOCK_TORTURE_TEST](#) and [CONFIG_PROVE_LOCKING](#) are a couple of examples that could be used as detection methods for Resource Locking and Improper Locking issues.

The ELISA Tool Investigation and Code Improvement Sub-WG is engaged in aiding the ELISA Kernel Development Process Working Group's goals by using various static analysis methods and tools available in the Linux kernel and making the results available to the ELISA and the Kernel communities.

- Continuous Integration (CI) with static analysis on Linux kernel repositories focusing on configurations relevant to Automotive and Medical use-cases.
- CodeChecker Web service
- Investigate [ECLAIR](#) results, tailor tool to identify actionable results
- Work on fixing Documentation and kernel-doc. This activity benefits ELISA as well as the Kernel community. This Sub-WG is actively engaged in improving Kernel documentation.

- Assist newcomers with onboarding. We recognize the need to help the rest of the ELISA community understand how the Linux kernel development process works. This Sub-WG is engaged in helping ELISA members participate in the Kernel development process. Newcomers are welcome!

As you can see, the ELISA WGs collaborate and work towards providing resources for System integrators to apply and use to analyze qualitatively and quantitatively on their systems. You might ask, “Why can’t ELISA qualify safety-critical systems?” We can’t, as we don’t know the product details to be able to do so. It is the System integrator’s responsibility to use the resources ELISA is developing to analyze their systems.

Please note that the ELISA project will release the resources under the [CC BY 4.0](#). This article is for informational purposes only and is not intended to address any specific implementation.

Now that you have an idea of what ELISA is all about, I invite you to join the discussion to help us define Safety Architecture for Medical use-case and bring us new use-cases. Please join the [ELISA member family](#).

