ELISA

ELISA Group

April 27, 2020

Abstract

In this paper we introduce the project and identify central challenges that any safety-critical project using Linux needs to overcome. We present our analysis of the challenges, outline long and short term plans on how to overcome them in the framework of the ELISA group and finally, present a collection of open building blocks that will emerge from those activities for reuse in safety-critical development projects using Linux.

JM: I'm not a great fan of a narrative style (using we). I prefer active verbs from the perspective of the document. For example: "This paper introduces the project and idenitfies..."

Contents

1 The Problem at Hand and Normative Considerations **3**

2 ELISA Strategy and Organization **4**

2.1 Working Groups 4

2.1.1 Development Process Subgroup 4

2.1.2 IVI Subgroup 5

2.1.3 OpenAPS Subgroup 5

2.1.4 Architecture Subgroup 5

2.2 Increasing Rigor Strategy 5

2.3 Development of a Safety argumentation 7

2.3.1 Process - Tailoring and Equivalence argumentation . . 7

2.3.2 Example Use Cases 7

2.4 Development of Open Building Blocks 8

2.5 Marketing and Recruitment 8

3 OSS-Specific Challenges **10**

3.1 Updates and Change 10

3.2 Bug tracking 11

3.3 Regression tracking 11

3.4 Freedom from Interference - Kernel Model 12
3.5 Linux Development Process Analysis 12

4 Overcoming the Challenges 12
4.1 Updates and Change 12
4.1.1 Short term strategy 12
4.1.2 Long term strategy 13
4.2 Bug tracking 13
4.2.1 Short term strategy 14
4.2.2 Long term strategy 14
4.3 Regression tracking 14
4.3.1 Short term strategy 14
4.3.2 Long term strategy 14
4.4 Freedom from Interference - Kernel Model 14
4.4.1 Short term strategy 15
4.4.2 Long term strategy 15
4.5 Linux Development Process Analysis QM 15
4.5.1 Short term strategy 15
4.5.2 Long term strategy 15

5 Open Building Blocks 15
5.1 Reference Architectures 15
5.2 Education and Best Practices Material 16
5.2.1 Safety 101 book 16
5.2.2 Open source 101 for safety people 17
5.2.3 Best practices for open source projects 17
5.2.4 Competition 17
5.3 Kernel Dependency Analysis Tool 18
5.4 Linux Kernel Model 18
5.5 Tailoring techniques - Annex QR 18
5.6 Managing artifacts for certification - PMT 18
5.7 Linux and Quality Management 18

6 Conclusion 18

7 License and Document History 18
7.1 License 18
7.2 Document History 19

1 The Problem at Hand and Normative Considerations

Current safety standards have targeted systems which have low-complexity application software, which do not use hardware concurrency (e.g. multi-core processors), and which use pre-existing and open-source software to a very limited extent. Hence, various domain safety standards (ISO 26262 [9], IEC 62304 [6], EN 50128 [4],...) do not consider pre-existing complex elements, such as Linux and glibc, running on complex multi-core hardware. For example, ISO 26262 has no appropriate classification of Linux, which is a pre-existing software (Part 8-12). But Linux continues to evolve and Part 8-12 applies only to unchanged SWCs (See ISO 26262-2 6-7.4.7). This specific mismatch already indicates that the ISO 26262 committee did not consider Linux-based systems.

An example of the cultural mismatch between the methodology expected by safety standards and the open-source work methodology is how change is handled; very conservative change management vs. encouraging dynamic change. Hence, ISO 26262 fits poorly on that subject as well.

JM: You're starting with an example without first motivating the problem...

Therefore, a dedicated interpretation for open-source software is required, either by interpreting IEC 61508 or by deriving a new domain standard, e.g. extracting objectives from first principles and determining which adjusted measures and techniques provide sufficient evidence of safety.

As a Linux-based system by definition is a mixed-criticality system with parts classified as QM/SIL0 and others with safety classification, there must be a clear definition and criteria for evaluation of QM/SIL0 elements. Lack of a definition of QM/SIL0 in the safety standards, e.g. ISO 26262 and IEC 61508, has already led to some confusion.

Safety standards do not provide rationales, which limits the ability to interpret the objectives of the requirements without violating the intent. Last but not least, the standards are closed. This limits their acquisition, use and acceptance in the open-source community

JM: I don't know what this sentence is referring to. 61508 has objectives for each subclause and 26262 has objectives for each part. I don't really have trouble determining the intent.

On the other side, open source projects frequently lack a formal description of the development process they follow (safety plan) and explicit trace data between the artefacts produced in each development phase (safety case).

To enable Linux to be incorporated in safety-critical systems, the gap between the activities practiced in open source project and those required by safety standards needs to be precisely analysed at both the technical and process levels. All issues must be either mitigated on the user side or addressed by process improvements on the Kernel development side.

JM: What does it mean to mitigate an issue on the user side?

2 ELISA Strategy and Organization

In this section we describe the cornerstones of the ELISA strategy to enable Linux in safety applications and describe the organizational structure of the ELISA effort. Besides the weekly sync call all members take part to coordinate general strategy, planning etc, there are subgroups working on individual aspects described below.

Weekly ELISA Sync Call The weekly ELISA sync call takes place every Friday:

Time Zone	From	To
JST	22:00	23:00
CET	14:00	15:00
CST	08:00	09:00

2.1 Working Groups

As of March 2020, there are three subgroups working on Linux development processes, architecture and various use cases.

2.1.1 Development Process Subgroup

The Development process subgroup aims to analyze the Linux development process in light of the traditional industry development processes as outlined in the safety integrity standards (here [5] and [9]) in order to define a reference process, that is both compliant/equivalent with the safety standards and compatible with OSS development workflows.
Once the reference process is defined, the Linux kernel community can implement additions/changes to the current process to close the gaps between the reference process and the status quo, which gives us a complete equivalence argument between the Linux Kernel development process and safety integrity standards, making the Linux kernel acceptable for use in safety applications from the process side.

Mailing List

Weekly Call The development workgroup weekly sync call takes place every Thursday:

Time Zone	From	To
JST	22:00	23:00
CET	14:00	15:00
CST	08:00	09:00

2.1.2 IVI Subgroup

Add description, maybe weekly meeting details

2.1.3 OpenAPS Subgroup

Add description, maybe weekly meeting details

2.1.4 Architecture Subgroup

A further group targeting architecture in general is currently under formation.

Weekly Call The architecture workgroup weekly sync call takes place every Tuesday:

Time Zone	From	To
JST	22:00	23:00
CET	14:00	15:00
CST	08:00	09:00

2.2 Increasing Rigor Strategy

Due to the immense complexity and huge deviations from traditional V-model style development processes as outlined in safety standards, it makes sense to build from the bottom up and follow an increasingly rigorous approach:
This strategy pertains to all areas, including but not limited to:

Standards

- ISO/IEC 33000 series of standards, CMMI, ASPICA [10, 3, 2]
- UL1998 [12]
- ISO 9001 [11]

- IEC 61508 / ISO 26262 [5, 9]

As QM software development is a basic requirement for developing any good quality software and to eliminate systematic faults to achieve safety, the approach can be to start small i.e, show that LINUX development meets the requirements of a basic software development process (e.g. ISO/IEC 33000 series of standards [10], Capability Maturity Model Integration (CMMI) [3], or Automotive SPICE [2]). Once this is achieved the requirements of UL1998 [12] (less rigorous requirements compared to IEC61508/ISO26262) can be added, followed by requirements from IEC61508 and ISO26262 [5, 9].

Safety Claims ELISA will work on safety claims of increasing complexity, starting with Linux based safety relevant systems of low complexity such as In Vehicle Infotainment systems (IVI), which have no strict timing/performance requirements, opposed to most safety relevant automotive systems. Once for simple example systems a satisfactory safety argumentation is found and documented, it can be expanded to more complex systems.

Architectures Aligned with the increasing rigor approach, ELISA targets architectures of rising complexity to keep the scope as limited as possible at first, expanding to more complex architectures later on.

Need examples

POSIX API Levels / Appliation environment profiles The POSIX standard [7] outlines in part 13 [8] application environment profiles (Effectively minimal subsets of the POSIX APIs required for typical realtime applications) in rising complexities.

- PSE51
- PSE52
- PSE53
- PSE54

```
Propose liasoning process (like DO-178)
Define SIL0/QM (ref: Clause 7-X  formalization as starting point)
Qualify convincing parts of the examples
```

JK: Clarify what was meant by that

2.3 Development of a Safety argumentation

2.3.1 Process - Tailoring and Equivalence argumentation

Since full compliance with any of the safety standards can not be argued for the Kernel development process at the moment, nor is it realistic to expect this to change in the near future, an equivalence argumentation is used to argue suitability for safety application of the Linux Kernel. To that end, it is necessary to map terms and processes of the Kernel development to the according steps in the V- process of the safety standards, and find equivalence arguments as to why the Kernel development process fulfills the intention behind the requirements of the safety standards.
This heavy tailoring/modification (beyond what is outlined within the safety standards as tailoring), requires a systematic approach to achieve confidence in the tailoring arguments. Such a methodology (including practical application examples) has already been developed for tailoring of IEC 61508 within the SIL2LinuxMP project, see [1].
If such an argument can not be made, an actual gap has been found that has to be addressed by modifying and or extending the Linux Kernel development process.
Once this tailoring has been outlined, ideally even earlier, certification authorities are brought to the table to make sure the argumentation is acceptable from their perspective. Annex QR was originally intended to be included into IEC 61508, in the event that this happens in a future edition of the standard, the described argumentation would even be fully compliant. This topic is addressed by the Process subgroup 2.1.1.

2.3.2 Example Use Cases

The core of ELISA strategy is to exercise the construction of safety argumentation at the example of several use cases, which then can be used as blueprints for further projects. Analyzing use cases is crucial

- To stimulate the properties of Linux that need to have a safety capability and to allocate appropriate integrity levels.
- To introduce usecase-specific constraints/requirements for Linux without trivialising the utilisation of Linux
- To use examples to advance certification capability from concrete implementations towards generic usecases, since as we understand it Linux has not been certified in an application agnostic way

- As a basis for applying open source methods and tools to establish a body of knowledge supporting the creation of critical products and systems based on Linux
- As a destination for experimenting with software and designs in a safety-relevant context
- As a means of attracting contributors into the ELISA community

Beyond the two use cases already under consideration (2.3.2 and 2.3.2), a cooperation with AGL (Automotive Grade Linux) is currently being established providing a third use case related to the IVI use case.
In the following we go into more detail on the two use cases currently under investigation by the ELISA group.

OpenAPS The OpenAPS project develops an artificial pancreas System to control insulin pumps.

IVI The in vehicle Infotainment.

JK: Ask openAPS group to write something

JK: Ask IVI group to write something

2.4 Development of Open Building Blocks

The results of the ELISA activities is a collection of reusable building blocks and instructions/examples how to use them, to construct a safety argumentation for Linux based systems.

2.5 Marketing and Recruitment

The problem now is getting acceptance and formal approval that Linux is suitable for use in safety-critical systems and applications. We need to shift the focus that the whole system is safe and sane. End result has to be safe, not just that a form has been filled. Developers, safety experts and regulatory authorities all share the same goal of wanting to make the world a safer place. Safety experts and regulatory authorities have a visible gap of knowledge in dealing with open source software in the safety domain, let alone community based development, highly automated and newer development methodologies. Open source developers often dont understand best practices for designing their software to be suitable for use in safety-critical systems. Linux is pervasively in our ecosystem and our devices already, and will be more use in future, so is a shared point of interest to both communities.
We need to reach out to both of these communities and get them talking together to bridge the gaps. This will require marketing related activities to

raise the awareness, and motivate involvement that aligns with their interests. Once they are engaged, this needs to be a community that they see as beneficial and enjoyable to participate in.
To that end, the next step is the creation of education, best practices and marketing material Gold deck to use to explain the problem and need for participants Pain problems Use Cases for ELISA - medical equipment, industry automation, autonomous systems (vehicle, factories, robots). Why use linux - new technologies available quicker, no licensing fees, total cost over lifecycle. Wider community to draw on for security issues. ELISA is build a wider community focused on safety issues. AI: Kate to take first pass - next Tuesday. Nicole, Nicholas, Olaf to review Wed/Thurs.
How do we want to communicate: Good website (what content do we want to add) what we have to offer, reasons to engage, areas to engage, perspectives to bring to play. :: need people to have vision here, and willing to review. Domain content. Clear instructions on how to engage with workgroups of interest. Standard LF code of conduct, respect for individuals applies. Social media engagement to raise awareness on a regular cadence of communication twitter channel (need specific to ELISA). Use for frequent communication of events relative to community. Public thanks for contribution are motivating for people. Set up FLOCK for ELISA. Free version available. Selective licensing of individuals. ** investigate LinkedIn for more technical related related content. (Whitepapers, content, press releases) Target 2 press release a year. Conferences to target outreach to EW Nuremberg VDA Automotive SYS - annual meetup, by safety and security people. (OpenSource) VDI - IEEE in german, engineering community and some safety. Bitcom - lobbying organization in Germany, digitalization, open source in sept, smart, hub conference in Berlin. OSS - LF events- Safety Safetronic (other) or Safetech (TuV SuD) ? Enable ambassadors for the project able to speak at conferences/events/inhouse/executives Information material to help ambassadors (reference material online, including slides to walk through, whitepapers for offline reading) AI Nicole: Outreach to Bitkom Forum: Nicole to provide overview at next working group meeting. Provide feedback to Kate on Gold Deck. Outreach to media (once content is created), classic press releases. Trigger news sites to pick up. Electronic Net newsletter, headlines.
Strategies to build up organic communities, rallying points. Outreach to target participants for solving this. Open Source Developers Safety Experts Regulatory authorities Open Source Users Professional OEM: EE systems like Car makers, Device makers, Robot makers, etc) Hobbyists with personal need: # OpenAPS codes Academics interested in helping to solve hard problems.
Articulate the compelling rallying points to academics (hard problems), Hob-

byists (personal interests), commercial(safe products) Create an open topic list of unsolved problems. Grab your own problems. Topics that are safety relevant. Get more people to contribute to papers, 3 or 4 people contributing - matching communication between industry review for academic research. Interesting use cases is important.
Initial Thoughts: Visible for customers and users and ones who might want to be engage. We want contributors and sponsors. Want to bring right stakeholders to the discussions. Car makers - not spend licensing fees, but making linux capable. Certifications companies engaged - part of handshake Approval that the whole system is safe and sane. Be sure for self. Want to make the world is a safer place - certification and developers both want same goal. Knowledge of safety and technical properties of system you want to evaluate. Gap of knowledge in dealing with software in safety domain, let alone open source, community based development, highly automated and newer development methodologies. Developers building system, want to make sure they are doing the right thing. Want to avoid harm to people. Did everyone do their job correctly. Informed and educated. We have enough good experts working on this. The person asked needs to know technical properties of system, not just filled out form End result has to be safe, not just that a form has been filled. Formal safety process and standard are only a minimum, and blind faith in standard, is not necessarily creating a safe system. Linux is pervasively in our ecosystem and our devices already, and will be more use in future. Server farms with Linux uptimes are so much better than alternatives. Problem now is getting acceptance and formal approval that Linux is suitable to be used in these safety-critical systems and applications.

JK: Does all this belong in white paper? I think it should be part of a strategy document

3 OSS-Specific Challenges

In this section, we present the major challenges ELISA faces. The following section presents how they are being addressed.

3.1 Updates and Change

Today, the majority of existing certified safety-related products are not updated in the field. This is driven by the fact that reassessment of a safety-critical system is a time-consuming and complex process. Products are developed to avoid hazards with sufficient level of confidence and incremental changes are therefore seldom foreseen. Safety (or security) updates necessary after development ends come with additional costs and risks, while

their safety (and security) benefits are typically rated low and beyond an acceptable level. Hence, in the end, optional or deferred updates are argued to be flawless and complete. Also, existing infrastructure typically does not allow field updates over the air, which further increases the costs for potential updates.

JM: This sentence doesn't make a lot of sense to me....

However, in contrast to traditional devices (e.g., an airbag ECU) nowadays everything else is getting connected. Cloud services are being introduced to all areas of life with an correspondingly increased risks of cyber attacks, especially attacks involving system- and chipset-level exploits such as Spectre/Meltdown/Rowhammer. To prevent security breaches, it is becoming mandatory to release updates within one day of publication, This contrasts strongly with current safety accreditation timeframes. This is not only a problem limited to open source software or Linux, but is a principal challenge for all products providing services in a connected world, including the commercial proprietary ones.

In connected settings every unpatched system must be considered non-secure. The same holds true with respect to functional safety for systems compromised by security vulnerabilities.

To conclude, updates are therefore a major challenge for connected safety-critical systems in general.

3.2 Bug tracking

All OSS projects beyond a rudimentary maturity level have a bug tracking system. However the bug tracking systms are only effective to a certain extent. Tracking bugs is not the most rewarding or prestigious work and there is correspondingly always a shortage of volunteers. Furthermore, depending on the bug tracking and the open source community, low quality bug reports are an issue that further increases the work load without any gain for the project in question. This is not so much a problem for the Kernel bug tracker [reference] but for the distributions downstream, see [reference to short/long term solutions], which absorb the bulk of low quality bug reports.

From a safety perspective, ignoring bug reports is not acceptable, however. A solution must therefore be found to organize bug reporting and tracking in such a way that is manageable.

3.3 Regression tracking

A problem related to bug tracking is regression tracking, i.e. tracking bugs discovered after a version (i.e. an LTS kernel version) has been released which exist in that version and must fixed nonetheless. While this is a general

problem for LTS maintainers, if the version is being used in a safety critical system, the developers must at least be aware that the bug exists. Should the bug impact that safety integrity of the system, the fixes must be backported to the release branch for the safety-critical item, A safety impact analysis must be done and it may reveal that further mitigation measures are necessary..

JK: Refer to Thorsten Lemhuis work, possibly ultimately under the umbrella of the process subgroup?

3.4 Freedom from Interference - Kernel Model

On the technical side, we need to understand better which safety claims can be made for the Linux kernel, and how to insulate against interference. This topic touches all use case subgroups and the yet-to-be-formed architecture subgroup. To create a Kernel model of sufficient granularity, several code analysis based approaches are being investigated. [reference to code analysis] [reference to architecture group]

3.5 Linux Development Process Analysis

A big challenge is to argue the aforementioned equivalence with the conventional development processes envisaged by the safety standards

4 Overcoming the Challenges

We now present our plan to overcome the challenges outlined in section 3.

4.1 Updates and Change

As outlined in 3.1, timely updates are a major issue. ELISA is working towards short and long term solutions as follows.

4.1.1 Short term strategy

As a first step to close this gap, the solution concept has to be judged in a constrained environment (e.g a specific use case or subsystem). In this way the solution's overall feasibility and its reception in safety community can be checked. The potential of state of the art update policies in security-critical systems and DevOps operations to increase software quality should be also be checked in parallel. Emphasis should be placed on understanding which concepts affect reliability and stability. An additional strategy is to

discuss these ideas with proprietary software providers as they must tackle the update challenge as well.

4.1.2 Long term strategy

As a starting point to approach system updates the underlying software has to be arguably safe.
One way to reduce the effort of impact analysis and changes is to partition the system up-front and support the partitioning with a freedom from interference argument for the non-safety-relevant parts. This assumes that even when there are frequent changes to the complete software stack, the impact to safety relevant parts are minimised and become manageable.
Nevertheless, for complex software like the Linux kernel, a structured path during analysis and verification needs to be established and supported by automation. An initial approach in this direction could be analysing existing update policies which have hard requirements on system stability (e.g. for a Linux server) and DevOps approaches to improving product quality. Formalized classification of the type of change with respect to functionality or security fixes or new or updated functionality would help to identify the impact of the changes. Each type may require different actions, but should not impact the overall process and strategy of updating a safety-critical system product. Investing in careful analysis will result in shorter verification cycles.

JM: where do these update policies come from?

For the matter of completeness, not only software is subject to change and update, but also the underlying tools (e.g. compiler or deployment tools). A common method of tool qualification includes testing the tool according to its use cases. It is assumed that a security or bug fix will not have impact to the tool's use cases and the tool qualification suite can be re-executed. In contrast to deployed product software, the tools' feature sets and use cases can be narrowed down to a limited set. This means that the approach towards tool updates most likely differs to the approach to updating product software in the field.
As the whole proposal for software update (e.g. in the field of security update of connected devices) is not yet sufficiently reflected in safety standards close collaboration with standardization authorities and safety community will be required to make fast software updates state of the art.

4.2 Bug tracking

JK: Add results of bug tracking investigation once presented to ELISA group

4.2.1 Short term strategy

4.2.2 Long term strategy

The downstream companies which build safety applications using the Linux kernel must have their own bug tracking systems. These systems would be less prone to being flooded with irrelevant entries and the companies would have a strong incentive to fix the bugs and also bring the fixes upstream. Ignoring the existence of possibly safety-relevant bugs is not acceptable from a safety perspective unless a solid mitigation mechanism and corresponding rationale for doing so can be developed.

JM: Hmmm... ARE there downstream companies building safety applications using Linux?

4.3 Regression tracking

4.3.1 Short term strategy

4.3.2 Long term strategy

4.4 Freedom from Interference - Kernel Model

ELISA is currently focusing its activities in context of two use cases IVI and OpenAPS to understand

- the system's safety requirements that are allocated to the Kernel
- the impact an incorrectly functioning Kernel on the safety claim itself

The following assumptions have been made at the start of the investigations.

- only certain functional layers of the Linux kernel are used for the specific use case.
- it is possible to trace the Kernel functional layers used for the specific use case
- those layers are a small subset of the entire Kernel architecture space.

If the above assumptions can be validated, the criteria for coexistence as defined in ISO 26262 could be used to demonstrate freedom from interference between the safety-related and non-safety-related kernel functionalities and isolate the safety functionality allocated to the Kernel. This reduces the scope of Linux that must be qualified.

4.4.1 Short term strategy

- Identify appropriate methodologies or tools which could be used to trace the control paths in the Kernel
- Identify safety-related and non-safety-related parts of the kernel for the specific use-case

4.4.2 Long term strategy

- Systematically identify the various interfaces between the safety-related and non-safety-related parts of the Kernel that could impact the safety functionality allocated to the kernel.
- Prove that the interaction between the non-safety-related Kernel and the safety-related Kernel functions does not hinder the safety functionality

4.5 Linux Development Process Analysis QM

4.5.1 Short term strategy

Audit process compliance in current development,
Talk to assessors about strategy
ISO 9001 compliance route

4.5.2 Long term strategy

statistical analysis of mailing lists

5 Open Building Blocks

5.1 Reference Architectures

Problem Statement: Deciding ref system architecture and understanding Kernel configuration for the use case
Linux is huge and understanding the configuration for example, defining the scope of Linux Kernel to features based on the selected reference use case is key. While defining the configurations the following should be considered i.e interfaces (APIs, power management), shared resources (system timer, PTP (Precision Time Protocol).
A decision also need to be made on if the idea is to work towards a

5.2 Education and Best Practices Material

There is currently poor awareness of safety in the wider open source community. The community is usually not educated about functional safety and related concepts. Most safety development guidelines are behind a closed curtain (not public domain) and there are no examples for functional safety systems in the open.
To enable the open source community, the following materials are being created

5.2.1 Safety 101 book

This introduction to functional safety for OSS developers gives an overview of the topic.

- What is Functional Safety (1 page ideally, no more than 5)
- Basic worked example that people can readily understand (e.g. train door)
- Specific system (e.g. Raspberry Pi) and context
- Identify possible hazards and losses
- Illustrate some chain of arguments to be taken to make safe
- Introduce distinction between safety and security, and mention things that are outside the safety context (e.g. reliability, robustness)
- Identify some common types of solution or safety strategies
- Perhaps Include a set of questions?
- Summarise concepts such as fail-safe and common strategies
- Introduce standards and processes, avoid too much detail
- Reference to the specific standards and where / if you can see them (or a more detailed summary / discussion)
- Plan for expert and non-expert readability review

JK: Needs rewriting, not sure if we want it in here in that detail

5.2.2 Open source 101 for safety people

This introduction to OSS gives safety engineers, architects and legal people an overview of open source software, the process by which is it developed and how it differs from traditional software development as it is know in industry projects.

5.2.3 Best practices for open source projects

- Safety conscious badge: considering safety as part of a projects goals
- Identify a list of things that are used to build a safety argument that tend to be missing from open source projects
- Build on processes and principles that will be familiar to open source developers
- Template for patches to extract requirements e.g. Coding Guidelines, Coding review templates
- Understand how security considerations can also be applied to safety
- Show how good code fulfills the safety guidelines - common sense approach to safety
- One or more worked example of systems using the solution
- Encourage consumers of a project to document how it is used in a particular
- Recruit some open source projects to try to apply these best practices and identify additional / alternative ideas

5.2.4 Competition

Come up with a safety use case for a Linux / open source project Linux Foundation will sponsor a straw man pre-qualification project Gold badge and title!

JK: Do we want that in the paper, probably not

5.3 Kernel Dependency Analysis Tool

5.4 Linux Kernel Model

To understand and systematically collect all plausible sources of interference that have the potential to influence an application, we need to get a thorough understanding of all the steps an application passes through in its life cycle from startup to termination. In combination with shared resources, a clearer picture should emerge on what can interfere with an application. Strategy towards a solution Mapping the steps an application lives through at the example of a simple application along with creating/identifying a model of what happens with Applications and the Kernel.

5.5 Tailoring techniques - Annex QR

JK: need to write that

5.6 Managing artifacts for certification - PMT

JK: Oskars domain

5.7 Linux and Quality Management

Quality management is the foundation on which all safety integrity is built on, therefore the process by which the Linux Kernel is developed has to be completely understood to make an equivalence argument towards Quality Management as it is codified in QM standards such as ISO 9001. Strategy towards a solution Analogously to the route.pdf for the qualification argument, ISO 9001 should be read, interpreted in the context of the Linux development process, gaps should be identified and rationalized or closed by extending the Process.

6 Conclusion

JK: needs to be written

7 License and Document History

7.1 License

Very unclear w.r.t. publication, what to do.

7.2 Document History

Version	Author	Changes
0.0	ELISA Group	Initial Google Docs draft.
0.1	ELISA Group	• Transferred from Google docs to LaTeX • Restructured Document • Added short term/ long term sections

References

[1] Annex qr. `https://sil2.osadl.org/user/data/SIL2LinuxMP/doc/other/AnnexQR/AnnexQR.pdf`.

[2] ASPICE. `http://www.automotivespice.com/`.

[3] CMMI. `https://cmmiinstitute.com/`.

[4] Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Software fr Eisenbahnsteuerungs- und berwachungssysteme. Standard, DIN Deutsches Institut fr Normung e. V., March 2012.

[5] IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems. Standard, IEC Internationale Elektrotechnische Kommission, April 2010.

[6] Medical device software - Software life cycle processes. Standard, IEC/SC 62A Allgemeine Bestimmungen fr elektrische Einrichtungen in medizinischer Anwendung, June 2015.

[7] IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R)) Base Specifications. Standard, IEEE The Institute of Electrical and Electronics Engineers, Inc, January 2017.

[8] Information technology - Standardized application environment profile (AEP) - POSIX¡(hoch)¿ realtime and embedded application support. Standard, IEEE The Institute of Electrical and Electronics Engineers, Inc, January 2003.

[9] ISO 26262:2018 Road vehicles - Functional safety. Standard, International Organization for Standardization, Geneva, CH, December 2018.

[10] ISO 3300x:2015 Information technology. Process assessment. Standard, ISO/IEC JTC 1/SC 7 Software and Systems Engineering, March 2015.

[11] ISO 9001:2015 Qualittsmanagementsysteme - Anforderungen (ISO 9001:2015); Deutsche und Englische Fassung EN ISO 9001:2015. Standard, DIN-Normenausschuss Qualittsmanagement, Statistik und Zertifizierungsgrundlagen (NQSZ), November 2015.

[12] stub. Standard, UL, March 2015.