ELISA

ELISA Group

March 27, 2020

Abstract

In this paper we introduce the project and identify central challanges 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 activites for reuse in safety critical devlopment projects using Linux.

Contents

1 The Problem at Hand and Normative Considerations

Domain safety standards have been developed based on the practices of companies that have not been using hardware concurrency (e.g. multicore processors), software with low complexity and very limited use of pre-existing and open-source software. Hence, domain safety standards (ISO 26262 [9], IEC 62304 [6], EN 50128 [4],...) have been written without considering dominant use of 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 as it is a "pre-existing software (Part 8-12)" and it is evolving, but 8-12 applies only to unchanged SWCs (See ISO 26262-2 6-7.4.7). Already this specific mismatch indicates that the standard committee did not consider Linux-based systems.

A cultural mismatch between the expectation of safety standards and the open-source work methodology is how change is handled, encouraging dynamics of change vs. very conservative change management. Hence, the safety standard has a poor fit on that subject.

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

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

Last but not least, safety standards are closed, limiting their reception, use and acceptance in the open-source community and lack the provision of rationales, which limits the ability to interpret the objectives without violating the intent.

On the other side, open source projects frequently lack a formal description of their executed development process (safety plan) and explicit trace data (safety case).

To enable Linux to play a part in safety critical systems, this gap needs to be precisely analysed on both the technical and the process level, and all issues either mitigated on the user side or addressed by process improvements on the Kernel development 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 Sycn Call The weekly ELISA sync call takes place every friday:

Time Zone	From	Till
CET	14:00	15:00
CT	8:00	9:00

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	Till
CET	14:00	15:00
CT	8:00	9:00

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	Till
CET	14:00	15:00
CT	8:00	9:00

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 Information 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
```

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 es 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.

Ask openAPS group to write something

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 don't 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 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: Visibile 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.

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 major challenges ELISA faces and how they are being adressed in the following section.

3.1 Updates and Change

As of today the majority of existing certified safety related products are not updated in the field after certification and launch. This is driven by the fact that reassessment of the item is a time consuming and complex process. Incremental changes are seldom foreseen as the product was developed to avoid hazards with sufficient level of confidence. Later changes come with additional costs and risks, while the benefit for safety (and security) introduced by software updates is typically rated low and beyond an acceptable level.

Hence in the end updates optional or deferred is 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.
However, in contrast to traditional devices (e.g., an airbag ECU) nowadays everything gets connected. Cloud services are introduced to all fields of life with an increased risk of cyber attacks, including attacks involving system and chipset level exploits such as Spectre/Meltdown/Rowhammer. To prevent security breaches, releases of updates within less than one day may become mandatory, which strongly contradicts with current certification timeframes. This is not only a problem limited to open source software or Linux, but becomes a principal challenge for all who want to provide services in a connected world, including commercial proprietary software providers.
In a connected setting every unpatched system must be considered non-secure, the same holds true with respect to functional safety compromised by security vulnerability.
In a connected setting every unpatched system must be considered non-secure, and by extension also non-safe
To conclude, updates are seen as a major challenge for connected safety-critical systems in general.

3.2 Bugtracking

All OSS projects beyond a rudimentary maturity level run a bug tracking system, however the respective communities only maintain the bug trackers to some degree. Maintaining a bug tracker is not the most rewarding or prestigious work, and there is always a shortage of volunteers. Furthermore, depending on the bug tracker and the audience, low quality bug reports are an issue further increasing the work load without any gain for the project in question. This is not so much a problem of the Kernel bug tracker [reference] but of the distributions downstream, see [reference to short/long term solutions], which absorb the bulk of low quality bug reports.
From a safety perspective however, ignoring bug reports is not acceptable, we therefore need to find a solution to organize bug reporting and tracking in such a way that is manageable.

3.3 Regression tracking

A problem related to bugtracking is the tracking of regressions, i.e. bugs introduced into the code before a branch (i.e. an LTS kernel version) is forked from the mainline of development and fixed afterwards. Such fixes need to be back-ported to the branch in question, that is used in a safety

critical item, or at the very least, the developers need to be made aware and handle the situation themselves.

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 investigated. [reference to code analysis] [reference to architecture group]

3.5 Linux Development Process Analysis

A big challenge is to argue the afforementioned equivalence with conventional 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 major issue. ELISA works towards short and long term solutions in the following way.

4.1.1 Short term strategy

As a first step to close this gap, the concept has to be judged in a constrained environment (e.g specific use case or subsystem). In this way overall feasibility and reception in safety community can be checked. In parallel understanding state of the art update policies of security critical systems and DevOps approaches towards increased software quality should be checked to understand which concepts on reliability and stability are used there Additionally, and to be open minded to get in touch with proprietary software providers to discuss these ideas as they will have to 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 arguable as safe. One way to reduce the cost of impact analysis and changes is by upfront system partitioning, with a freedom of interference argument for non safety relevant parts. To avoid a complete time consuming change impact analysis, an upfront system partitioning with a freedom of interference argumentation to non safety critical parts needs to be established. . By achieving this, it is assumed that even in case of frequent changes to the complete software stack the impact to safety relevant parts are significantly reduced and become manageable. Nevertheless, an automation-supported structured path as support during analysis and verification needs to be established for complex software like Linux Kernel. A hint on how to go in this direction involves the analysis of existing update policies which have hard requirements on system stability (e.g. Linux server) and DevOps approaches towards improved product quality. Formalized classification of the type of change will help to identify the impact of the change with respect to bug or security fix as well as new or updated functionality. Each type may require different actions, but should not impact the overall process and strategy of updating a safety product. Careful analysis will benefit in shorter verification cycles. For the matter of completeness, not only software is subject to change and update, but also the underlying tools (e.g. compiler or deployment). 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 use cases of the tool and the tool qualification suite can be re-executed. In contrast to deployed product software, feature sets and use cases of tools can be narrowed down to a limited set. This means that the approach towards tool updates most likely differs to the way of 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 Bugtracking

Add results of bug-tracking investigation once presented to ELISA group

4.2.1 Short term strategy

4.2.2 Long term strategy

The companies downstream building safety applications using the Linux kernel operate their own bug tracker which are less prone to being flooded with irrelevant entries and also have a strong incentive to fix the bugs and also bring the fixesm upstream. Ignoring the possibly safety relevant bugs is not acceptable from a safety perspective, unless a solid mitigation mechanism and according rationale for doing so can be developed.

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 focussing its activites in context of two use cases IVI And Open APS to understand

- the safety requirements of the item that is allocated to the Kernel
- the impact of incorrect functioning of the Kernel on the safety claim itself

The following assumptions are made at the start of the investigation.

- only certain function layers of Linux kernel might be used for the specific use case.
- it is possible to trace the functions layers of the Kernel used for the specific use case and it is a small subset of the entire Kernel architecture space.

If the above assumption is validated “Criteria for coexistence” discussed by ISO26262 could be used to support Freedom from interference of the safety related functionalities and non-safety related functionalities of the Kernel to affect the safety functionality allocated to the Kernel. This will help to reduce the scope of Linux that will need to be qualified.

4.4.1 Short term strategy

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

4.4.2 Long term strategy

- In a systematic way identify various interfaces between the safety related and non-safety related parts of the Kernel that could impact the safety functionality allocated to the kernel.
- Proving that the interaction of the non-safety related functions of the Kernel with the safety related part does not hinder the safety functionality

4.5 Linux Development Process Analysis QM

4.5.1 Short term strategy

Audit of 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

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 project's 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!

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

need to write that

5.6 Managing artifacts for certification - PMT

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, ISO9001 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

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 für Eisenbahnsteuerungs- und Überwachungssysteme. Standard, DIN Deutsches Institut für 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 für 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 Qualitätsmanagementsysteme - Anforderungen (ISO 9001:2015); Deutsche und Englische Fassung EN ISO 9001:2015. Standard, DIN-Normenausschuss Qualitätsmanagement, Statistik und Zertifizierungsgrundlagen (NQSZ), November 2015.

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