Linux Plumbers Conference | Richmond, VA | Nov. 13-15, 2023

Putting Linux into Context

Towards a reproducible example system with Linux, Zephyr & Xen

Philipp Ahmann, Robert Bosch GmbH

With work from ELISA contributors:
Alessandro Carminati (Red Hat), Maurizio Papini (Red Hat),
Shefali Sharma, Shuah Khan (LF), Stefano Stabellini (Xilinx/AMD),
Sudip Mukherjee (Codethink), Thomas Mittelstädt (Robert Bosch GmbH),

whoami

BOSCH — Product Manager for Embedded Open Source

ELISA — ENABLING LINUX IN SAFETY APPLICATIONS — Chair of the Technical Steering Committee
Lead of the Systems Working Group

THE LINUX FOUNDATION | Europe — Member of the Inaugural Advisory Board

OSS enthusiast and promoter

ELISA

ENABLING LINUX IN SAFETY APPLICATIONS

ELISA

ENABLING LINUX IN SAFETY APPLICATIONS

SAFETY … don’t mix it up with SECURITY

Premier Members

BOEING
intel
Red Hat

General Members

AISIN
arm
斑马智行 Powered by AliOS
BOSCH
Codethink
Elektrobit
地平线 Horizon Robotics
HUAWEI
LINUTRONIX LINUX FOR INDUSTRY
LOTUS NYO
SAIC 上汽集团 SAIC MOTOR
SUSE
SUZUKI
WNDRVR
ZTE

Associate Members

AUTOMOTIVE GRADE LINUX
OTH OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Industry Support

CIVIL INFRASTRUCTURE PLATFORM
OSADL Open Source Automation Development Lab eG
UL

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

"The mission of the project is to define and maintain a common set of elements, processes and tools that can be incorporated into Linux-based, safety-critical systems amenable to safety certification."

from the technical charter

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Mike Kiev on Unsplash

Working Groups (WGs) - Horizontal

Safety Architecture

Red Hat

Open Source Engineering Process

Codethink

Linux Features

mobileye / intel

Systems

BOSCH

Tool investigation & Code Improvement

BOEING Codethink

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Mike Kiev on Unsplash

Working Groups (WGs) - Verticals

Aerospace

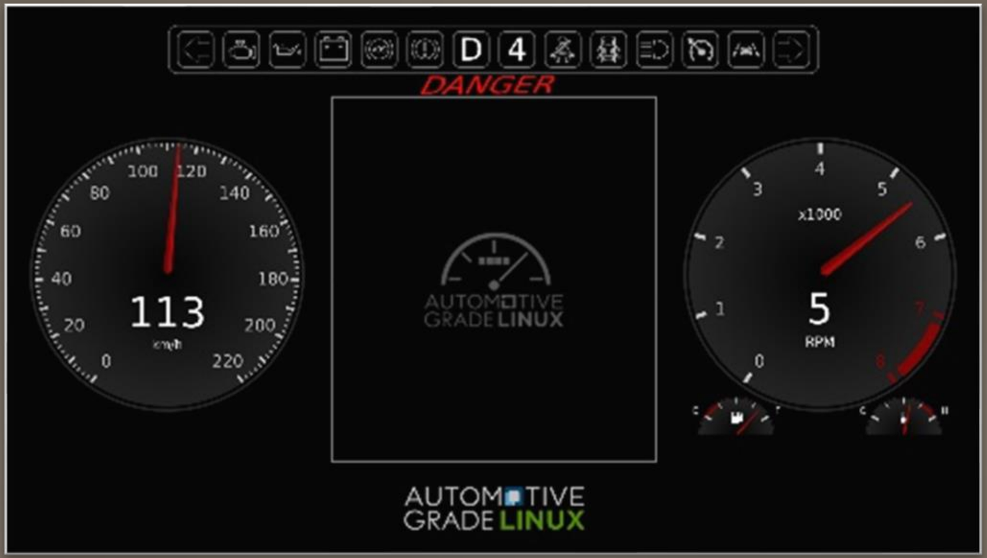
BOEING

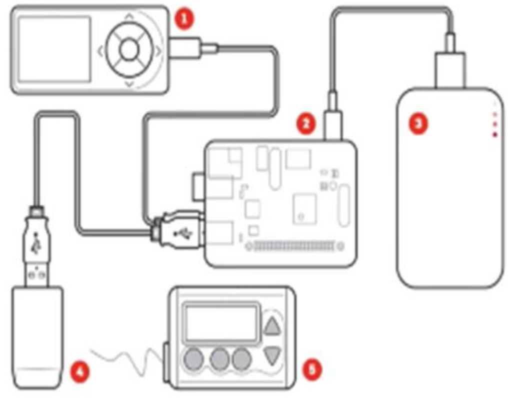
Automotive

BOSCH

Medical Devices

LINUX FOUNDATION





OpenAPS elements

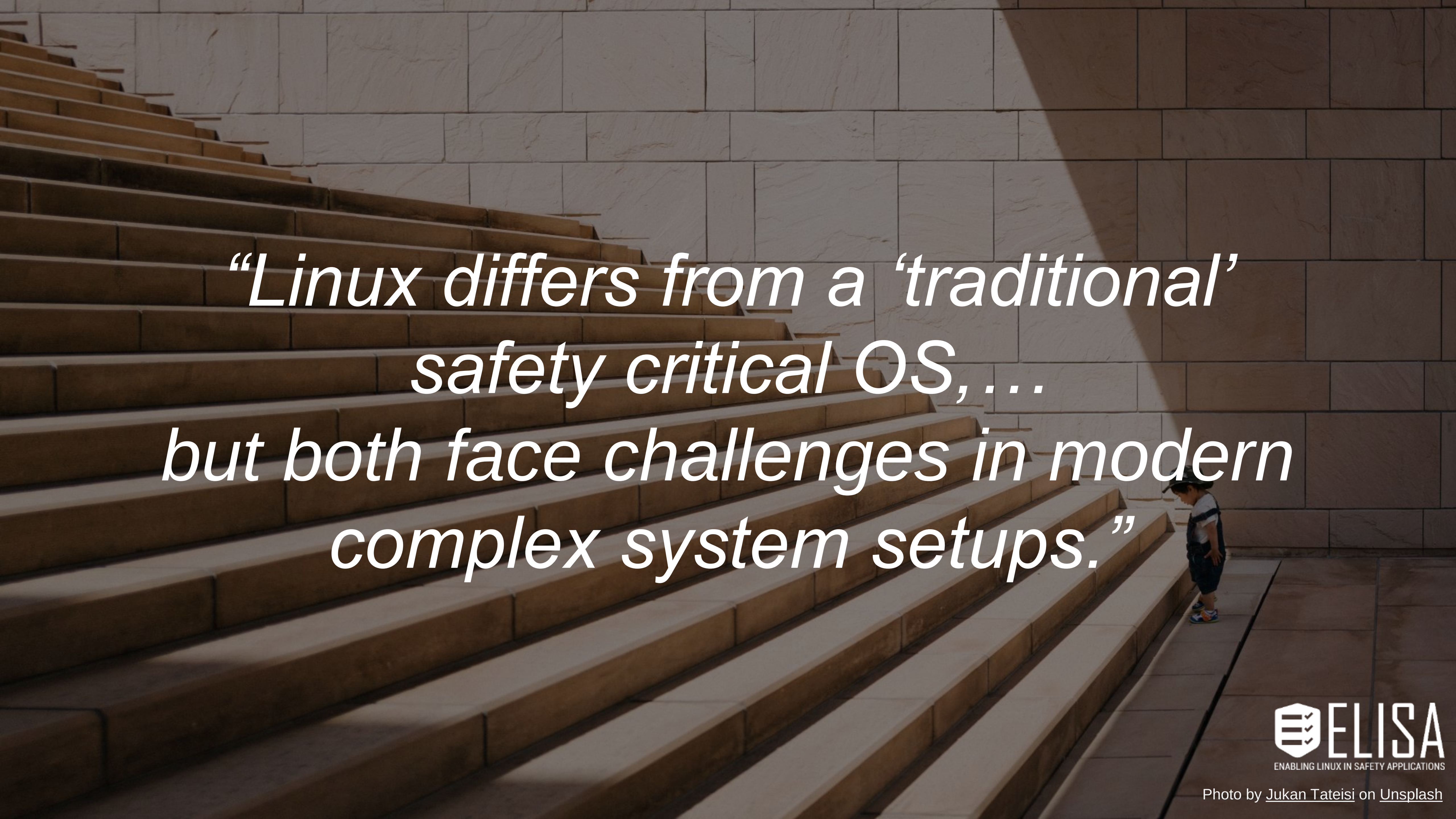
1. Continuous glucose monitor
2. Computer
3. Battery
4. Radio stick
5. Insulin pump

@DanaMLewis

Dana Lewis‘ OpenAPS project: https://youtu.be/kgu-AYSnyZ8

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Mike Kiev on Unsplash



*“Linux differs from a ‘traditional’ safety critical OS,…
but both face challenges in modern complex system setups.”*

ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Jukan Tateisi on Unsplash

Clash of worlds

(or what is often considered unsafe by safety experts):

- Memory management
- Dynamic memory allocation
- Caches
- Interrupt handling
- Real time scheduling
- …

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

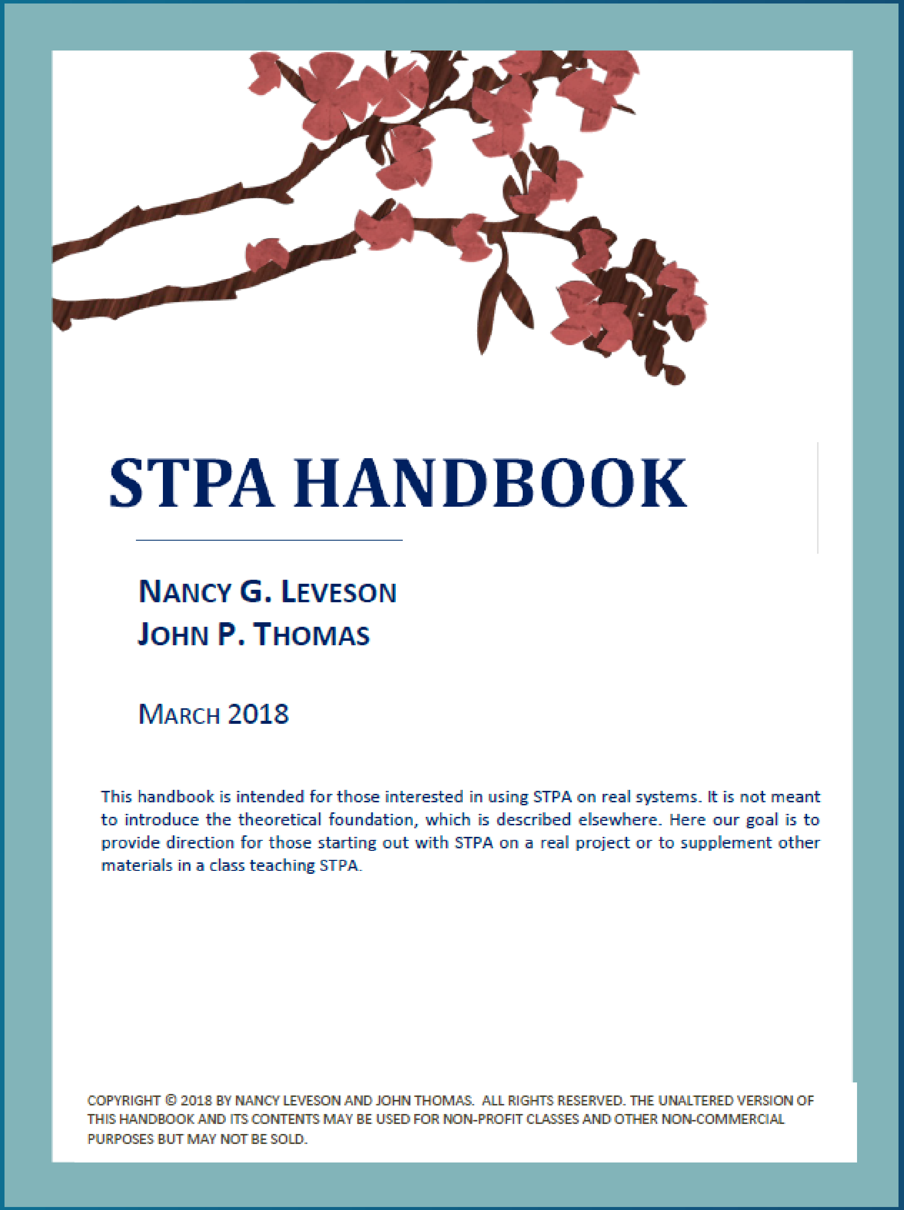
Photo by Jukan Tateisi on Unsplash

Tools + Documentation help to understand complex systems better

- *STPA*
- *strace and csope for workload tracing*
- *ks-nav (graphical representation kernel sources)*
- *real-time analysis*

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Jukan Tateisi on Unsplash



STPA HANDBOOK

NANCY G. LEVESON
JOHN P. THOMAS

MARCH 2018

This handbook is intended for those interested in using STPA on real systems. It is not meant to introduce the theoretical foundation, which is described elsewhere. Here our goal is to provide direction for those starting out with STPA on a real project or to supplement other materials in a class teaching STPA.

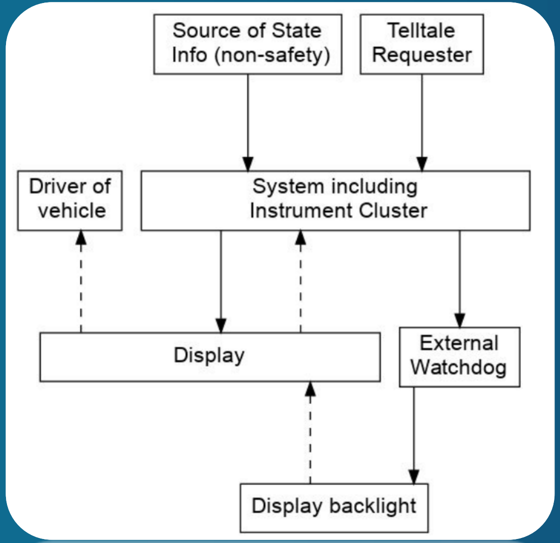
COPYRIGHT © 2018 BY NANCY LEVESON AND JOHN THOMAS. ALL RIGHTS RESERVED. THE UNALTERED VERSION OF THIS HANDBOOK AND ITS CONTENTS MAY BE USED FOR NON-PROFIT CLASSES AND OTHER NON-COMMERCIAL PURPOSES BUT MAY NOT BE SOLD.

https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

System
Theoretic
Process
Analysis

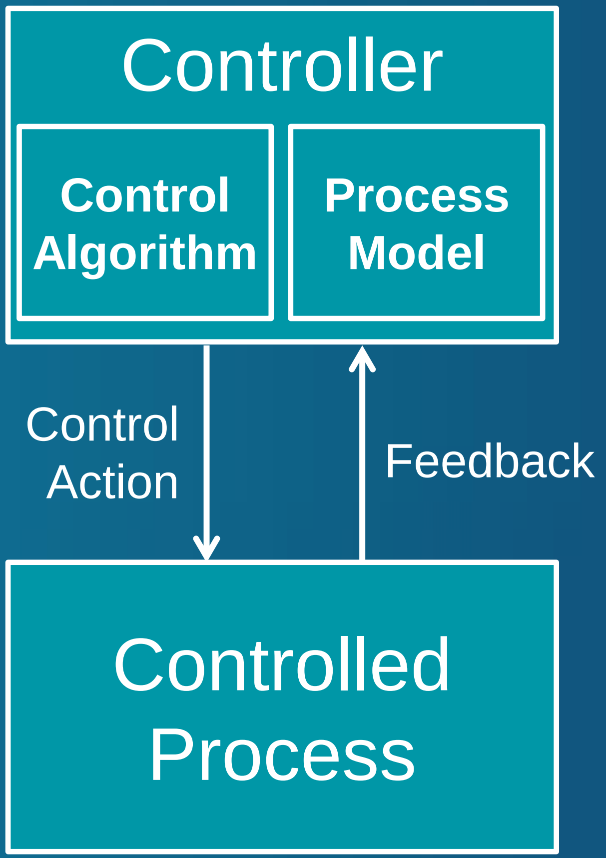
ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

STPA – Basics



- Relatively new hazard analysis technique
- Very complex systems can be analyzed
- Iterative towards detailed design decisions
- Includes software and human operators
- Provides documentation of system functionality
- Can be easily integrated into (model-based) system engineering process

STPA – Basics



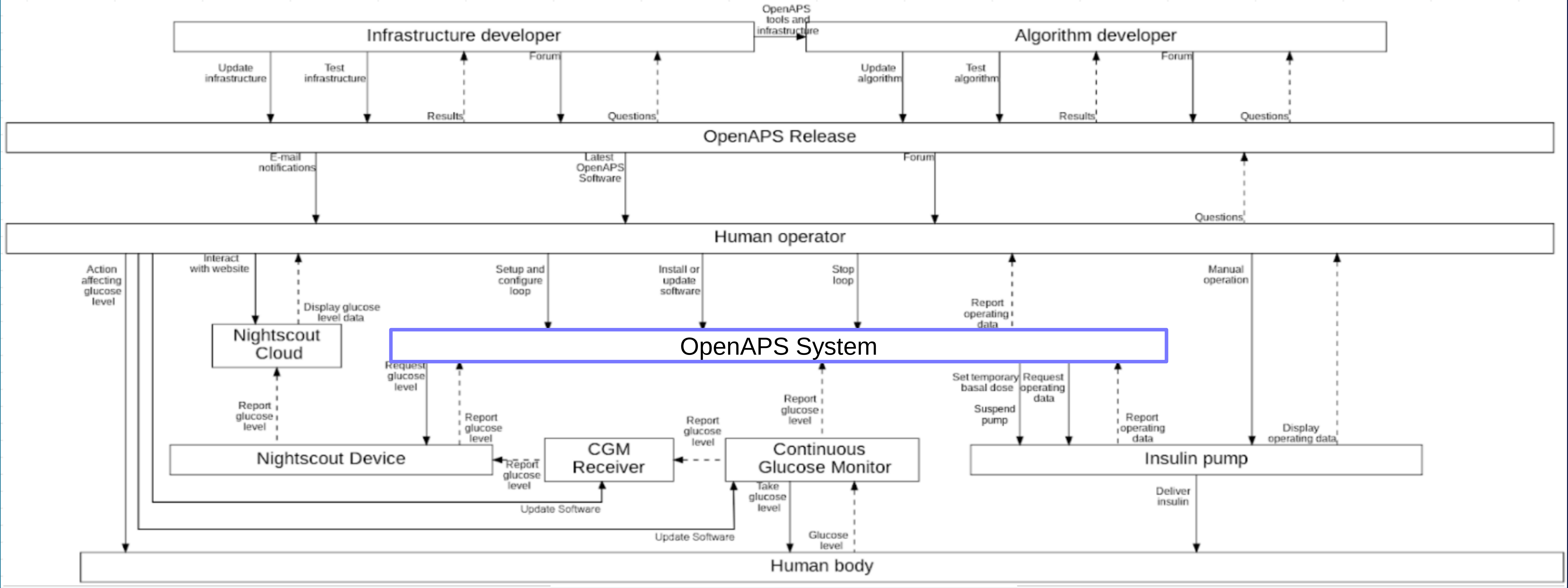
4 key elements

- **Controller** sends
- **Control Action**(s) to a
- **Controlled Process** which provides
- **Feedback** to a controller

A controlled process can be a controller.

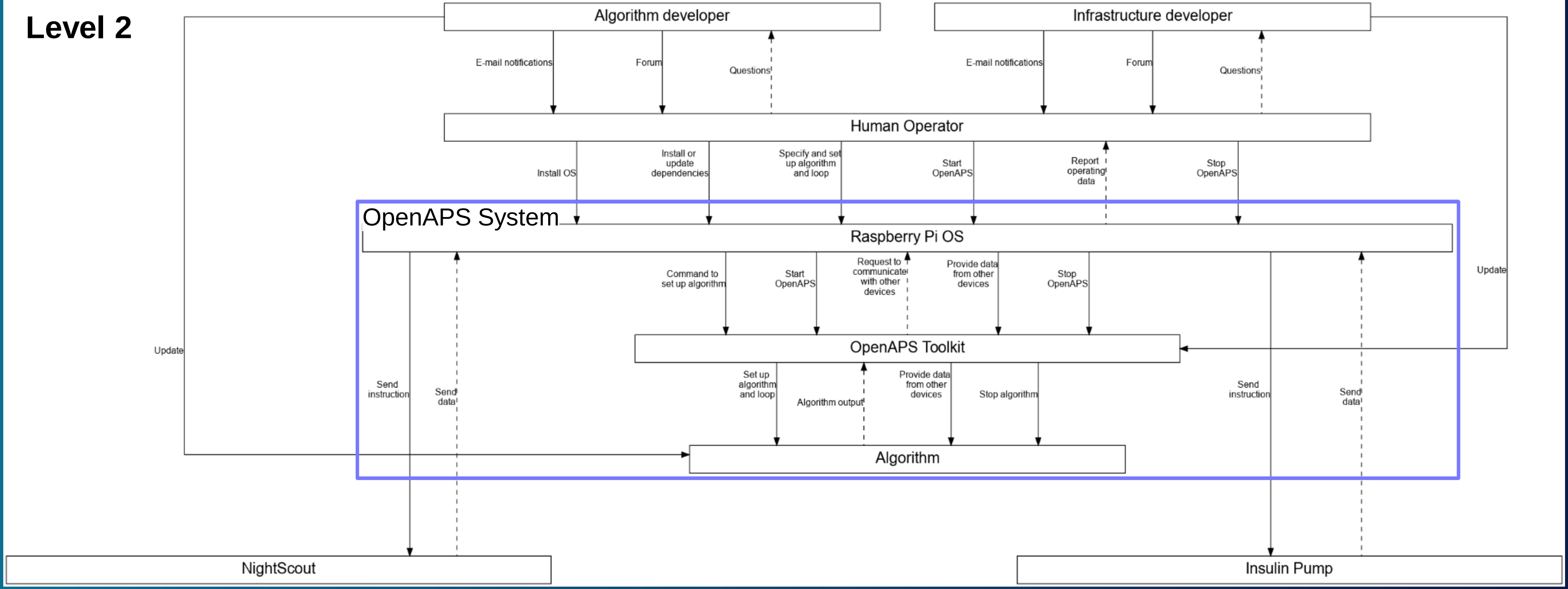
Q: What can be unsafe control actions?

STPA – In action (example for OpenAPS)



https://docs.google.com/spreadsheets/d/1vIKNM4hKV3FRt1w9BEt0t7bFgYYgivH3JUsuoe5RmTo

STPA – In action (example for OpenAPS)



ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Deeper level of analysis required workload tracing

Main tools used:

- strace
- cscope

Extract information:

- System Call
- Frequency of call
- Involved Subsystem
- System Call Entry Point

System Call	Frequency	Linux Subsystem	System Call Entry Point (API)
read	3	Filesystem	sys_read()
write	11	Filesystem	sys_write()
close	41	Filesystem	sys_close()
stat	24	Filesystem	sys_stat()
fstat	2	Filesystem	sys_fstat()
pread64	6	Filesystem	sys_pread64()
access	1	Filesystem	sys_access()
pipe	1	Filesystem	sys_pipe()
dup2	24	Filesystem	sys_dup2()
execve	1	Filesystem	sys_execve()
fcntl	26	Filesystem	sys_fcntl()
openat	14	Filesystem	sys_openat()
rt_sigaction	7	Signal	sys_rt_sigaction()
rt_sigreturn	38	Signal	sys_rt_sigreturn()
clone	38	Process Mgmt.	sys_clone()
wait4	44	Time	sys_wait4()
mmap	7	Memory Mgmt.	sys_mmap()
mprotect	3	Memory Mgmt.	sys_mprotect()
munmap	1	Memory Mgmt.	sys_munmap()
brk	3	Memory Mgmt.	sys_brk()
getpid	1	Process Mgmt.	sys_getpid()
getuid	1	Process Mgmt.	sys_getuid()
getgid	1	Process Mgmt.	sys_getgid()
geteuid	2	Process Mgmt.	sys_geteuid()
getegid	1	Process Mgmt.	sys_getegid()
getppid	1	Process Mgmt.	sys_getppid()
arch_prctl	2	Process Mgmt.	sys_arch_prctl()

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

https://github.com/elisa-tech/ELISA-White-Papers/blob/master/Processes/Discovering_Linux_kernel_subsystems_used_by_a_workload.md

Workload tracing documentation is mainlined.

- Understanding system resource necessary to build and run a workload is important
- Linux tracing and strace can be used to discover the system resource in use by a workload
- Additional tools (like perf, stress-ng, paxtest) can help to analyze performance and security of the OS
- **Credits to Shuah Khan & Shefali Sharma for bringing it mainline**
  - /Documentation/admin-guide/workload-tracing.rst


index : kernel/git/torvalds/linux.git

Linux kernel source tree

about summary refs log tree commit diff stats

path: root/Documentation/admin-guide/workload-tracing.rst

blob: b2e254ec8ee846afe78eede74a825b51c6ab119b (plain)

```
.. SPDX-License-Identifier: (GPL-2.0+ OR CC-BY-4.0)

======================================================
Discovering Linux kernel subsystems used by a workload
======================================================

:Authors: - Shuah Khan <skhan@linuxfoundation.org>
          - Shefali Sharma <sshefali021@gmail.com>
:maintained-by: Shuah Khan <skhan@linuxfoundation.org>

Key Points
==========

 * Understanding system resources necessary to build and run a workload
   is important.
 * Linux tracing and strace can be used to discover the system resources
   in use by a workload. The completeness of the system usage information
   depends on the completeness of coverage of a workload.
 * Performance and security of the operating system can be analyzed with
   the help of tools such as:
   `perf <https://man7.org/linux/man-pages/man1/perf.1.html>`_,
   `stress-ng <https://www.mankier.com/1/stress-ng>`_,
   `paxtest <https://github.com/opntr/paxtest-freebsd>`_.
 * Once we discover and understand the workload needs, we can focus on them
   to avoid regressions and use it to evaluate safety considerations.

Methodology
===========
```

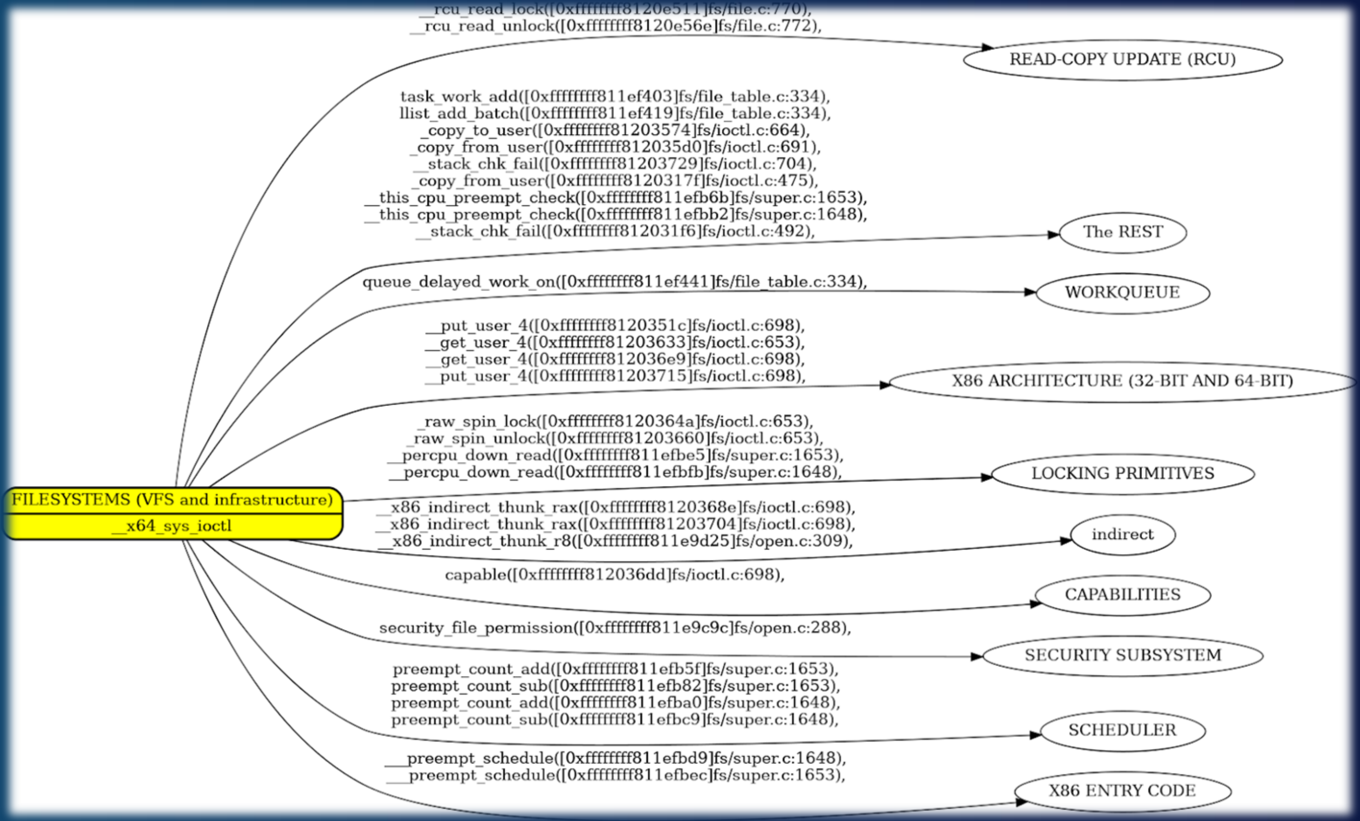
ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Dynamic tracing is supported by Static Analysis Navigator (ks-nav tool)

- Supports the analysis on code/kernel level
- Graphical representation of source code
- Provides insights about the Kernel construction

- **Is there a good place upstream?**

- Credits to Alessandro Carminati & Maurizio Papini (both Red Hat)



https://github.com/elisa-tech/ks-nav

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Use case centric vs. common/generic use of Linux (the core)

- Use cases bring a different point of view and set context, but deal with similar problems
- Requires deep dives
- Deep dive from the past were e.g.:
  - PREEMPT_RT and how to not break it.
  - Real-time Linux analysis tool set.
- All results should end up in upstream documentation
- Helps system integrators to build safe software and improve Kernel quality

Important topics for potential deep-dives:

- Synchronization / timing
- Interrupt and exception management
- Resource access management
- Dynamic memory allocation
- Inter process communication & inter processor communication
- System initialization
- Kernel configuration & trimming

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Possible next documentation: Admin guide for PREEMPT_RT

- PREEMPT_RT mostly upstream, but documentation on use can still be improved.
  - Nothing available so far in the admin-guide kernel documentation.

- Shuah Khan and Elana Copperman presented first results.
  - RT Linux in Safety Critical Systems: the potential and the challenges

- **The Linux Features for Safety-Critical Systems (LFSCS) within ELISA is looking for support by PREEMPT_RT users/experts to bring this forward!**

ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Onur Binay on Unsplash

Photo by Scott Ymker on Unsplash

Photo by Roberto Nickson on Unsplash

Interaction with other communities (outside of ELISA)

- Open source projects focusing on safety-critical analysis

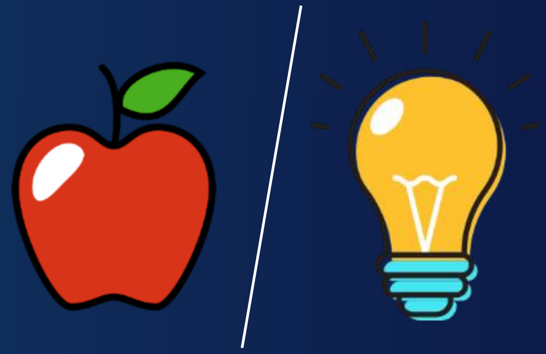
Xen Project · Zephyr™

- Open source projects with safety-critical relevance and comparable system architecture considerations

Automotive Grade Linux · SOAFEE · SDV Eclipse Software Defined Vehicle

- Further community interactions

Yocto Project · SPDX · Linaro

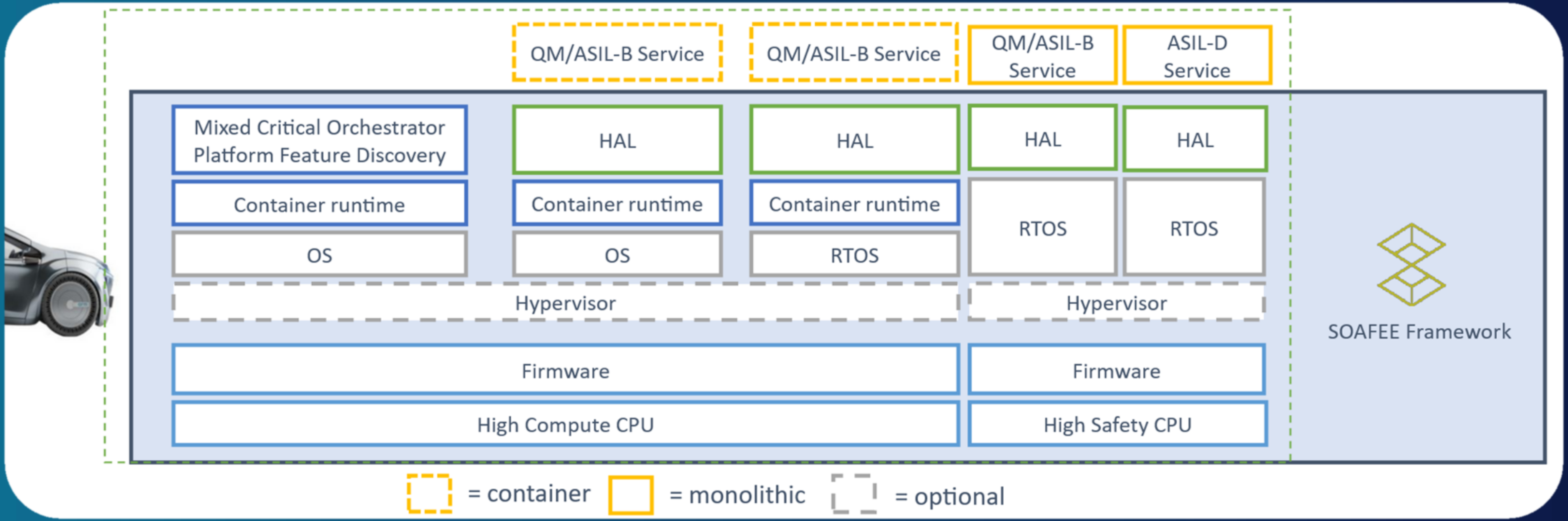


"If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple.
But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas."

— George Bernard Shaw

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

SOAFEE Architecture Vision



https://architecture.docs.soafee.io/en/latest/contents/architecture.html

“When it comes to prototyping systems,
the existing guidelines are limited;
reproducing demos is hard and time consuming.”

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Natalia Y. on Unsplash

Static Partitioning with Xen, LinuxRT, and Zephyr: a concrete end-to-end example

Stefano Stabellini

Embedded Linux Conference 2022

ENABLING LINUX IN SAFETY APPLICATIONS

Content of the Xen end-to-end example

- Build a reference system with default tooling
  - Xen, Linux kernel & rootfs and Zephyr
  - Use ImageBuilder for bootable configuration
  - Xen Device Tree examples
- Give guidance on features (“steps”)
  - Static partitioning
  - Device Assignment
  - Cache Coloring
  - Shared Memory and Event Channels
  - PV drivers

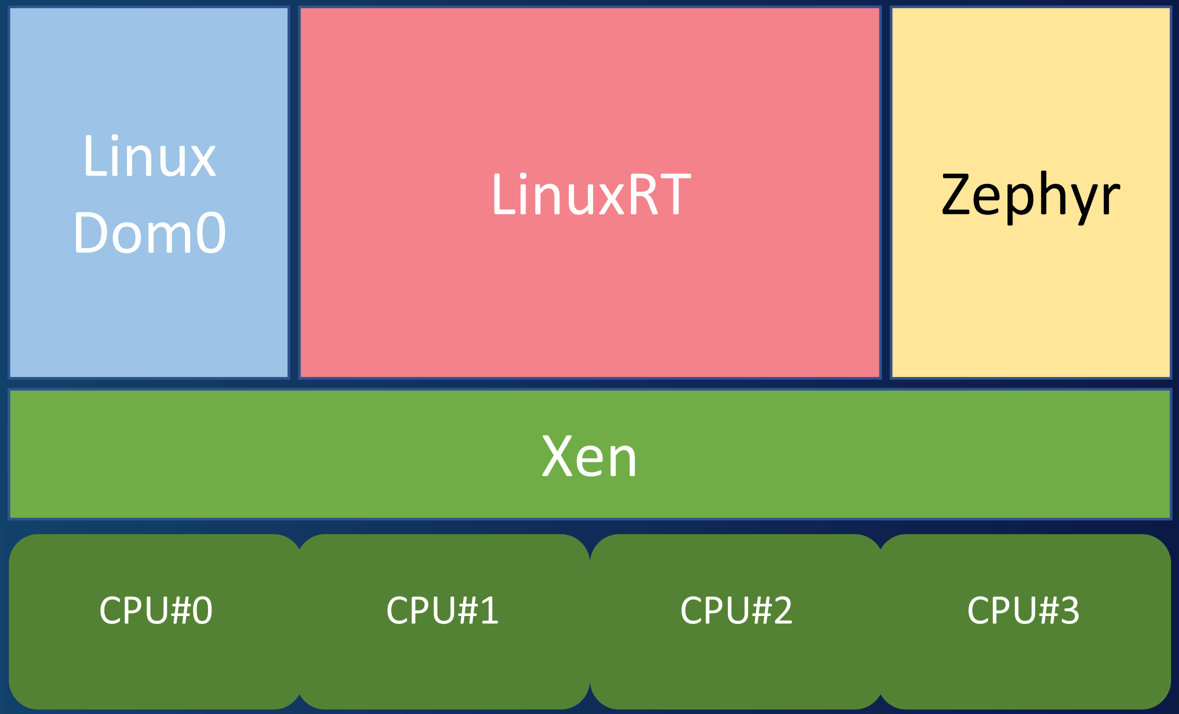


Photo by S. Tsuchiya on Unsplash

Challenges

- New hardware
- Community support
- OS distro
- Tools & CI
- Proprietary drivers
- Images
- SBOM

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Photo by Rob Wicks on Unsplash

Big thanks to…

Thomas Mittelstädt

- Robert Bosch GmbH
- Brings 30 years of experience at multiple operating systems and at build & integration systems. He provides trainings, documentation and technical support to various kind of Bosch users.

Sudip Mukherjee

- Codethink
- He has been a mainline kernel contributor since 2014. Sudip is also a Debian Developer and has worked in multiple automotive projects for Codethink's clients.

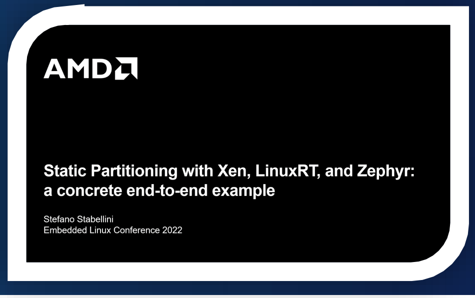
ENABLING LINUX IN SAFETY APPLICATIONS

Major challenges during setup of XEN systems

- Select target board with
  - Hardware support for XEN, especially SMMU controller
  - XEN community support
  - Documentation for build and setup
  - Licenses compliant to OSS project
- Setup of Yocto build environment
  - Amount of computer resources
  - Network and Host dependencies
- Finding valid descriptions
- Build image parts based on descriptions
- Finding community support at occurring build problems
- Understanding XEN setup and structure

Evaluated targets

• **Renesas RCAR 3.0 family** *(link to Wiki of eLinux)*	• **Xilinx Zynqmp and Ultrascale family** *(link to product page)*
+ XEN hardware support + Functional XEN systems (also graphic)	+ XEN hardware support + Functional XEN systems + Good documentation and open source support of Xilinx
- Proprietary licenses for essential parts like graphic - Not available at standard market	- Graphic at Zcu102 atm not able to be handled by XEN - Zcu102 well supported, but additional complexity due to FPGA programming



ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Evaluated targets – cont.

- **Qemu systems for Xilinx**
  (link with some hints for setup with XEN at Xilinx boards)

  + XEN support

  + Functional XEN systems

  + no hardware needed

  - Only for development, not for hardware related demo cases

- **Raspberry Pi systems**

  - Hardware support not sufficient for security requirements of XEN

- **NXP i.mx8 systems**

  + Good hardware support for hypervisor like XEN

  - Less community support

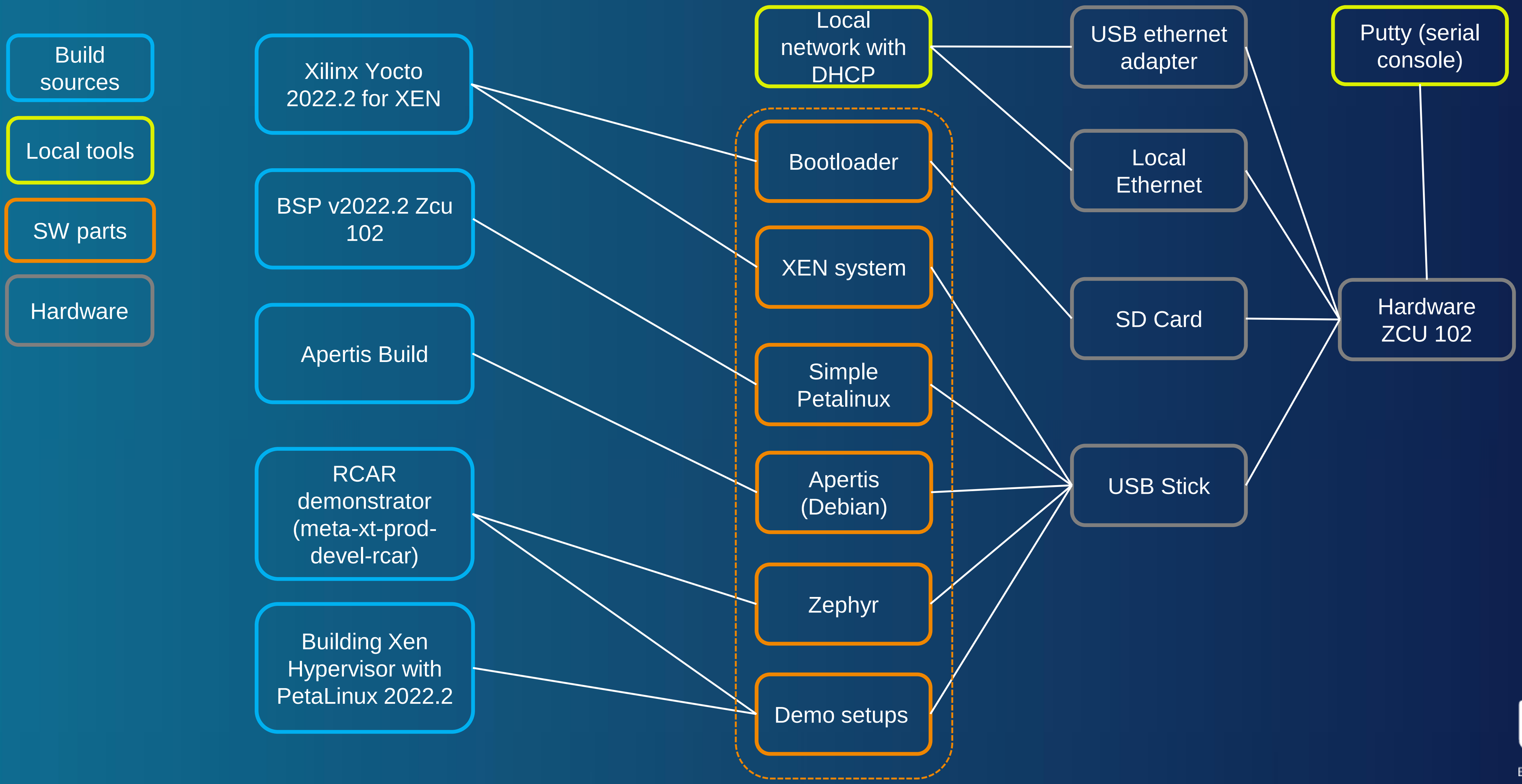
ENABLING LINUX IN SAFETY APPLICATIONS

Used hardware

- Board ZCU102 (link to description)
    - Reference manual (link)
    - SD card 16GB for boot loader
    - USB Stick 16GB for demonstrator setup
    - USB-Ethernet-Adapter (DLINK)
- Environment for setup
    - Local DHCP server (VM with system networkd)
    - Putty for serial console
    - USB Keyboard (for TTY console)
    - HDMI screen

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Overview of the XEN example system



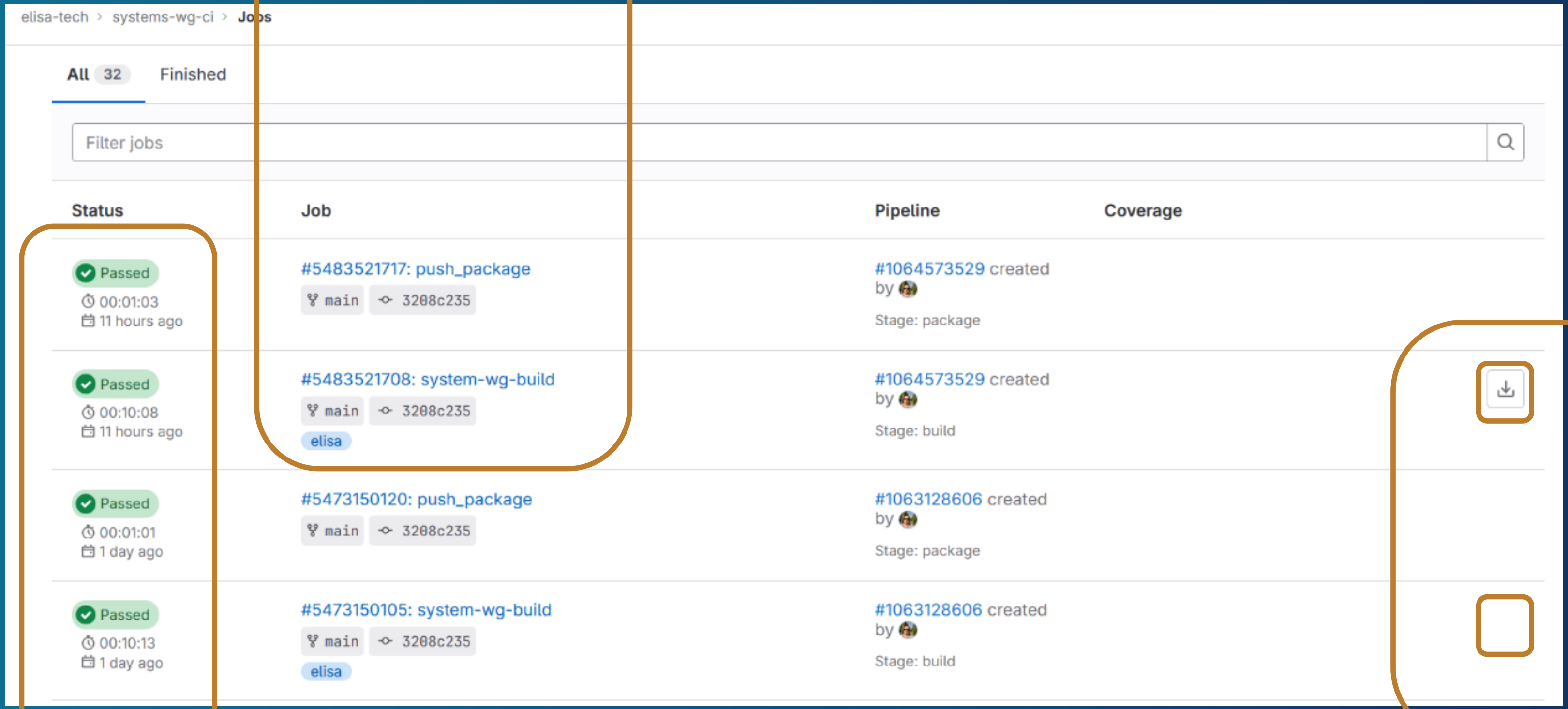
External parts of system images

- Xen Hypervisor (link for build description)
  - Image, ramdisk, device tree
  - Boot.bin
- Petalinux (link for binaries from "BSP")
  - Image, ramdisk, (device tree: not used for XEN)
- Zephyr (atm got from demo for Renesas RCAR, link for build description)
  - Image
  - Configuration file for XEN
- XEN configuration files (created on description at link)
- Apertis (Debian based, specific image, but general build instructions at link)
  - Image, ramdisk, (device tree: not used for XEN)
- XEN image builder (link for download and usage)

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

CI enablement: https://gitlab.com/elisa-tech/systems-wg-ci

Build & packages



Artifacts download of recent images

Runs daily

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

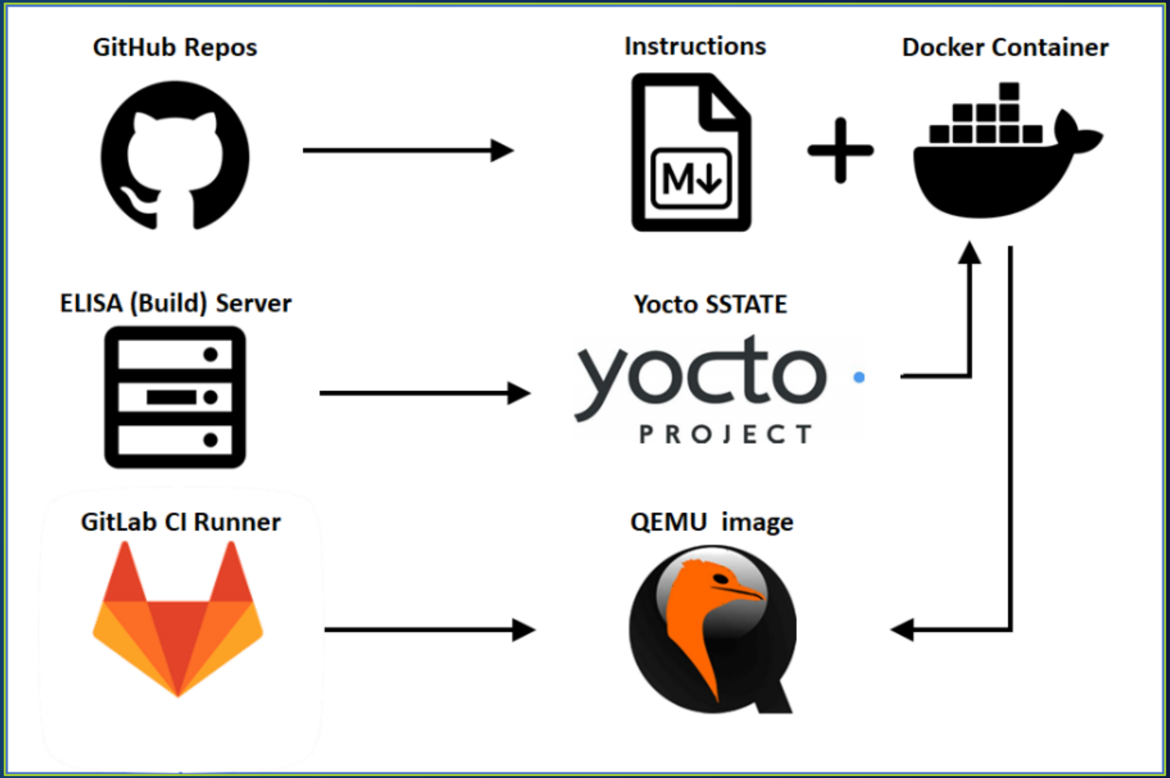
meta-elisa: Various starting points provided

- Plain and native from source
  https://github.com/elisa-tech/meta-elisa

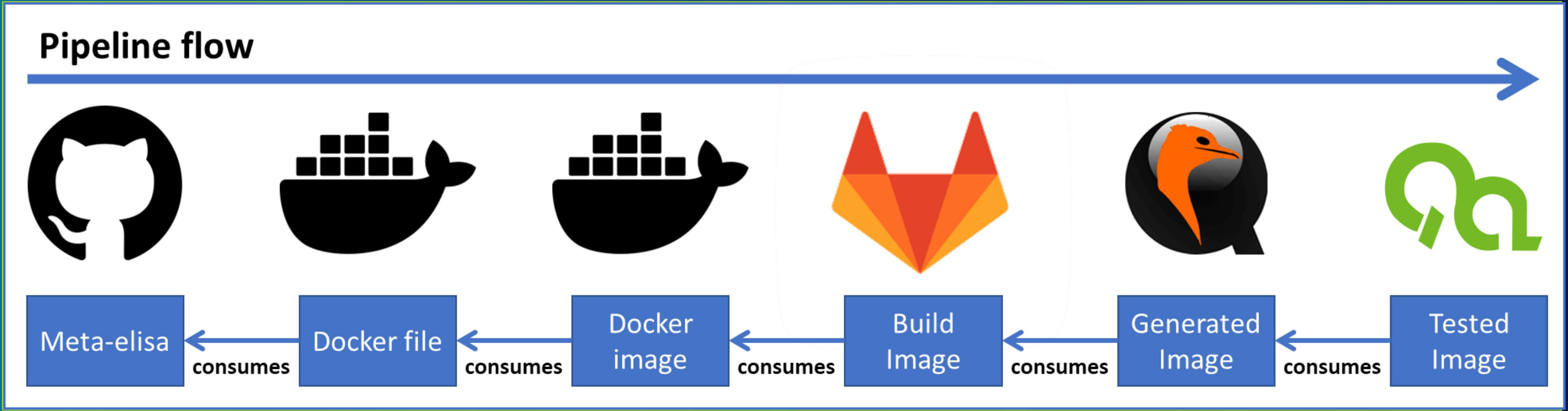
- Using docker container
  https://github.com/elisa-tech/wg-automotive/tree/master/Docker_container

- With cached build using SSTATE
  modify "`conf/local.conf`" after the "`source`" command before the "`bitbake`" command

- Download binaries directly from build server
  https://gitlab.com/elisa-tech/meta-elisa-ci



Pipeline dependencies

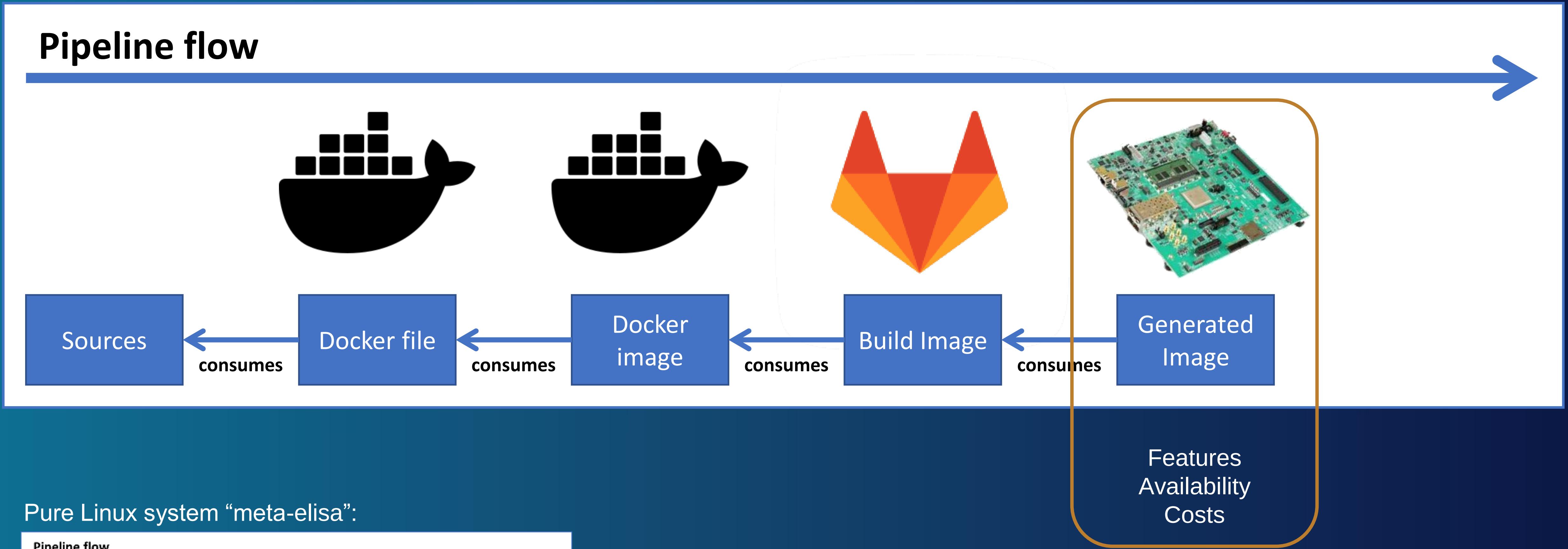


Full description in the blog

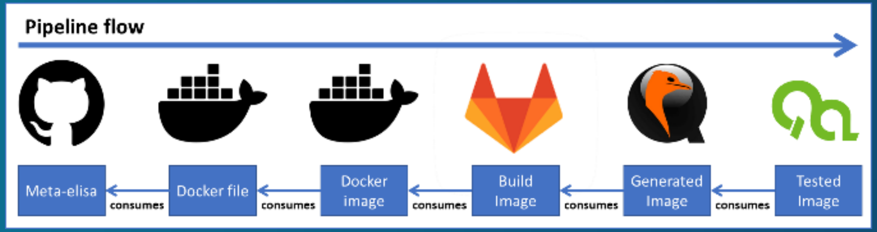
https://elisa.tech/blog/2023/04/05/elisa-ci-enablement-automation-tools-for-easier-collaboration/

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Limitations of the current implementation.



Pure Linux system “meta-elisa”:

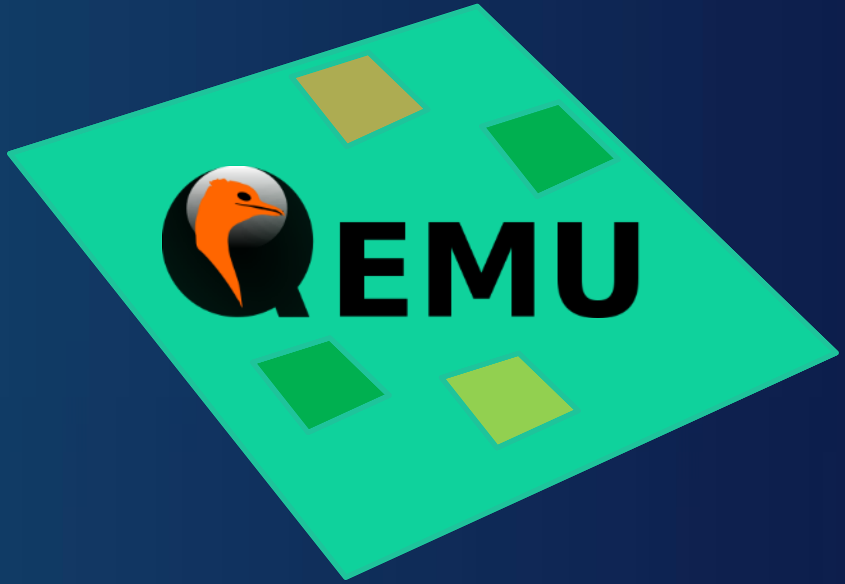


ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

From hardware to qemu (again?!)

- QEMU increases availability, lowers costs, but misses some features (like HW interfaces)

&



- Uncovered topics:
  System diversity, hardware prototyping, virtual GPU performance, “real µC” involvement

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Open questions…

- What is a good hardware to extend the PoC scope?
- Are there further existing examples where open source, security, safety and compliance come best together?
- Which alternative real-time operating system and virtualization should be incorporated?

Photo by Ana Municio on Unsplash

Benefits provided by the ELISA project

- **Provide system engineering and safety competencies**
  - Workload tracing upstream in kernel mainline
  - Tools for kernel analysis in ELISA github repo
- **Provide a start into safety critical system creation**
  - Hosting seminars to educate system creators/integrators
  - Create a knowledge base around Linux in safety critical environments
  - Provide a working example system for easy start into system creation
- **Want to get much more of this system experience upstream**

Where do we want to do, where do we need help, … (path to grow)

	Completed example “workload tracing”	**Ongoing activity “PREEMPT_RT guide”**	**Planned work “reproducible example system”**	**Future activities “critical kernel core components docu”**
Available inputs (to work on task)	-	RT and RT tooling deep dives	Interaction with Xen and Zephyr community	none yet!
Target activity	Tools to get a better understanding Creating a SBOM and porting to yocto	Provide a guide on how to work with real time inside the kernel	Help people understand to create complex Linux based systems	Identification of mission critical kernel parts supporting safety of products
Where we could need help	Help in porting the Raspbian demo to Yocto incl. kernel build	Contribution from PREEMPT_RT users to create the document	Proposals for well supported community hardware with HV support	Deep dives into special topics: - memory, interrupts, …
Place for the results	Documentation in Kernel admin-guide yocto with SBOM for openAPS commuity	Documentation in Kernel admin-guide	ELISA project github	- Kernel patches - Documentation in admin-guide - manpages

Thank you!

Linux
Plumbers
Conference

Richmond, Virginia | November 13-15, 2023