Developing Open Source Projects for Safety and Security

Kate Stewart, Senior Director of Strategic Programs
stewart@linux.com

June 25, 2019

Is open source software compatible with safety standards?

Short answer is YES

But…

- Requires major transformation
- Usually forces “forking”
- Is very “expensive”
- Standards are “complex”
- There are many standards…

Safety Standards

- IEC 61508 Generic Standard
- IEC 61511 Industrial Processes
- IEC 61513 Nuclear Industry
- IEC 62061 Machine Safety
- EN 50126/8/9 Railways
- ISO 26262 Automotive

- DO178B/C Aeronautics

- ECSS Space (ESA)

- IEC 62304 Medical devices

“The nice thing about standards is that there are so many of them to choose from.” [Tanenbaum]

Users Demand Accountability

- Feature richness and completeness are not enough
- “Who is liable if something goes wrong?”
- Nobody wants to be first

Photo: https://pixabay.com/illustrations/pointing-accusation-accuse-blame-1991215/

Software Development Culture Clash

Open Source Software Development

Source: https://www.wired.com/story/how-github-helping-overworked-chinese-programmers/

Software Development for Certifications

Source: https://www.occupational-resumes.com/images/Software-Developer.jpg

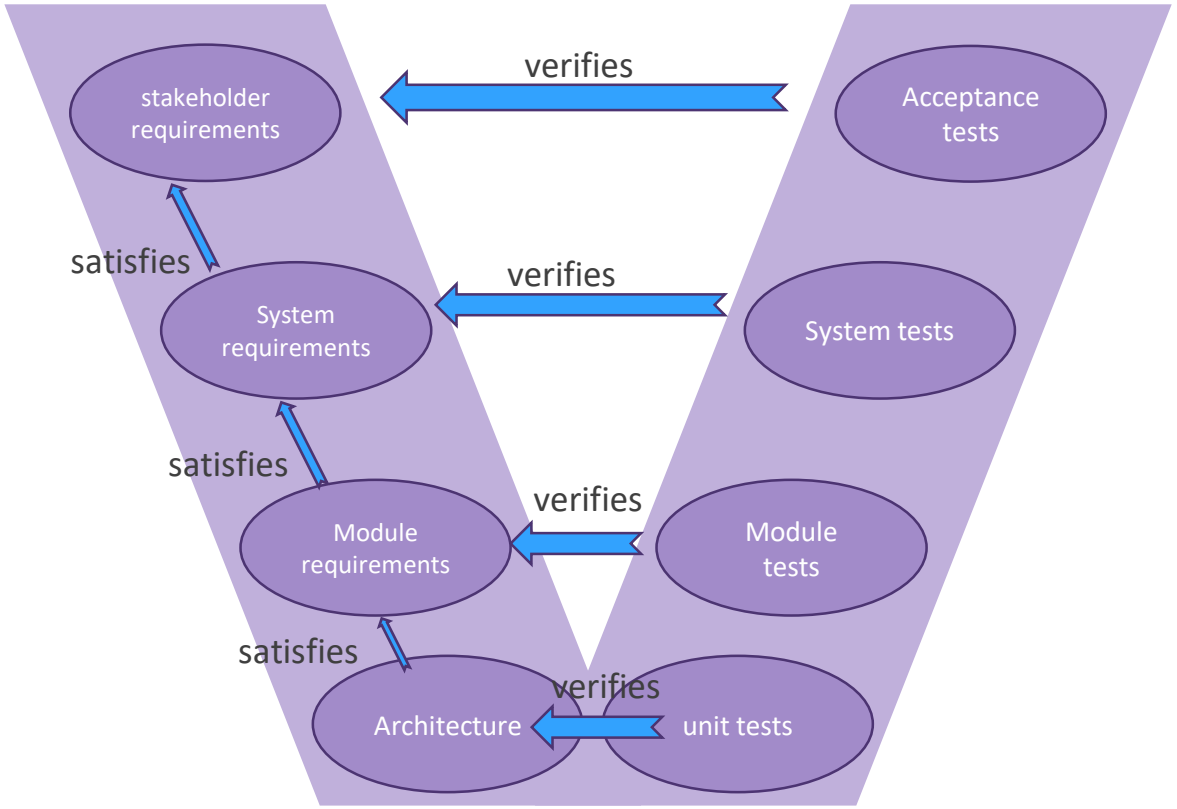
Why Culture Clash?

Open source software is not a problem in itself, however it is difficult to map typical open source development to **the V-model used by Certification Authorities**

- Specification of features
- Comprehensive documentation
- Traceability from requirements to source code
- Number of committers and information known about them
- Certification authority may not be familiar with open source development

V-Model: Requirements Traceability

- Reference links between requirements
- Verification links from related tests
- Satisfaction links from decomposed requirements
- Implementation links from user stories



How to Approach Certifications with Open Source?

- Understand how it will be used in a system
- Define and limit scope early to what is needed for system
- Automate as much of the information tracking as you can
- Auto-generate documents from test and issue tracking systems
- Get proof of concept approval from a certification authority as early as possible

Build on top of Open Source Strengths

- Code is available publicly and can be scrutinized by anyone.
- Code reviews and direct user feedback help improve quality

However…

- Do we have the right set of reviewers?
- Who gets to have the final say?
- How do we guarantee that the reviewer is aware of safety implications?
- For how long should changes be reviewed?

Certifications of Open Source?

- No good examples out there.
- Different perspectives and concerns by experts in each expertise area.

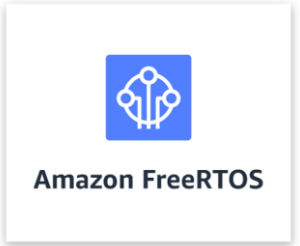
Linux Foundation projects that working towards being able to participate in Safety and Security Certifications

smaller footprint ⟷ larger footprint

Small Footprint: Wishlist for an RTOS

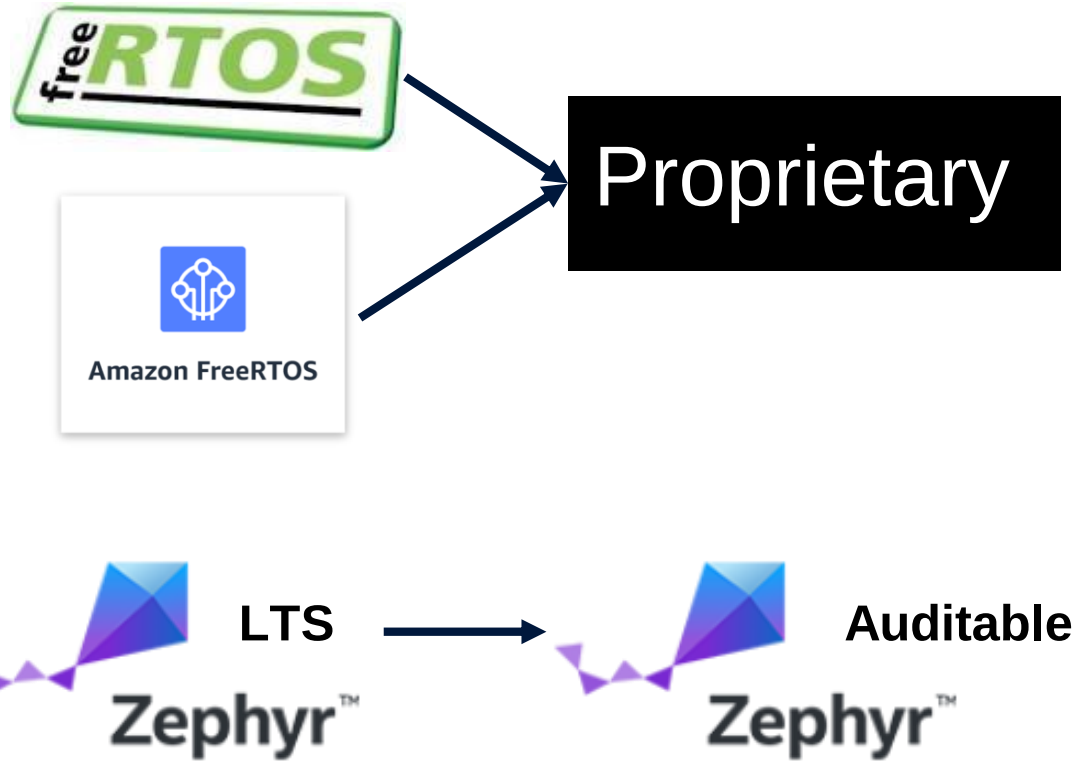
- **Open source** implementation
- **Small and trusted** code base
- **Safety-**oriented architecture
- Built-in **security** model
- **POSIX-compliant** C library
- Supports **deterministic** thread scheduling
- Supports **multi-core** thread scheduling
- **ISO-compliant** development
- **Accountability** for the implementation
- Industry **adoption**
- Certification **friendly interfaces**

Open Source OS/RTOS



Safety Certification Open Source Options?

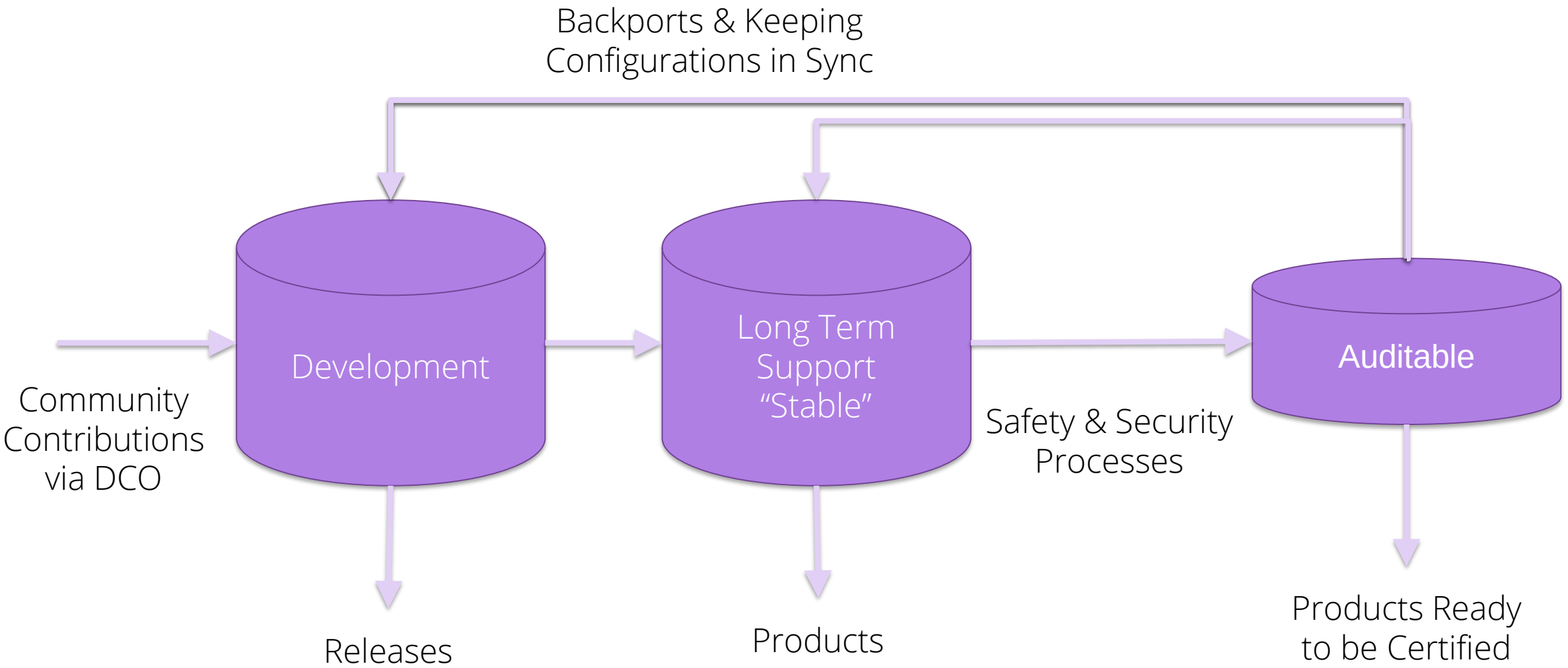
Explicit Path



?

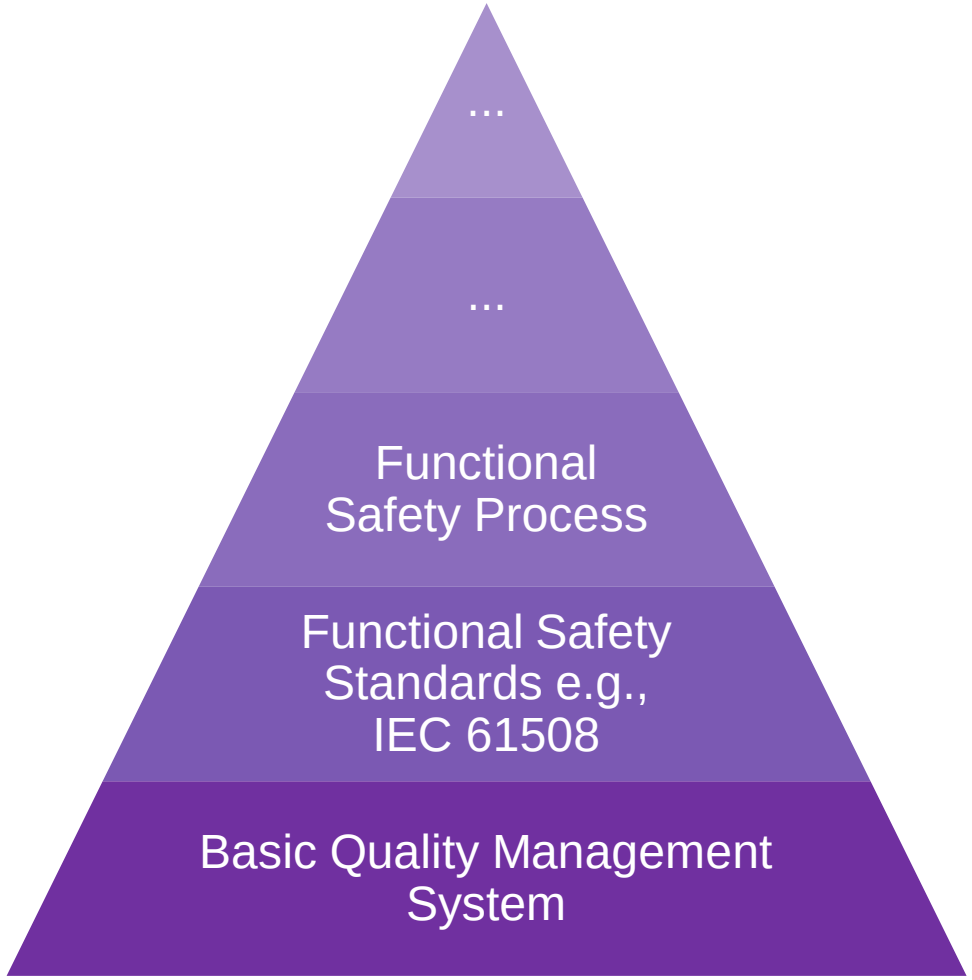
Code Repositories

"**Define and limit scope early**"



Quality Matters

- Quality is a **mandatory expectation** for software across the industry.
- Software Quality **is not** an additional requirement caused by functional safety or security standards.
- Functional safety considers Quality as an **existing pre-condition**.
- Quality Managed (QM) status should be the **goal of any open source project**, regardless of functional safety or certification goals.



Quality as a foundation

Build up from Guidelines familiar to Certification Authorities

Example: **MISRA-C**

- A software development standard
- Focus on safety, security, portability and reliability
- Latest version is MISRA-C:2012
- Contains 167 guidelines in the standard plus 14 new guidelines in Amendment 1
- A guideline can be “mandatory,” “required” or “advisory”
  - **Mandatory** - All code shall comply with every mandatory guideline. Deviation is not permitted.
  - **Required** - All code shall comply with every required guideline. Deviation is allowed.
  - **Advisory** - It is a recommendation. Formal deviation is not necessary

… but be aware there are challenges.

- Can be controversial to use
- Proprietary standard → $
- Proprietary tooling required → $$$

Approach:

- Deviations are key
- Limit to a well-defined scope
- Incorporate in contribution guidelines
- Developer and community buy-in

Rule 15.5 - A function should have a single point of exit at the end

- Most readable structure
- Less likelihood of erroneously omitting function exit code
- Required by many safety standards
  - IEC 61508
  - ISO 26262

How is Zephyr Doing it?

Zephyr™

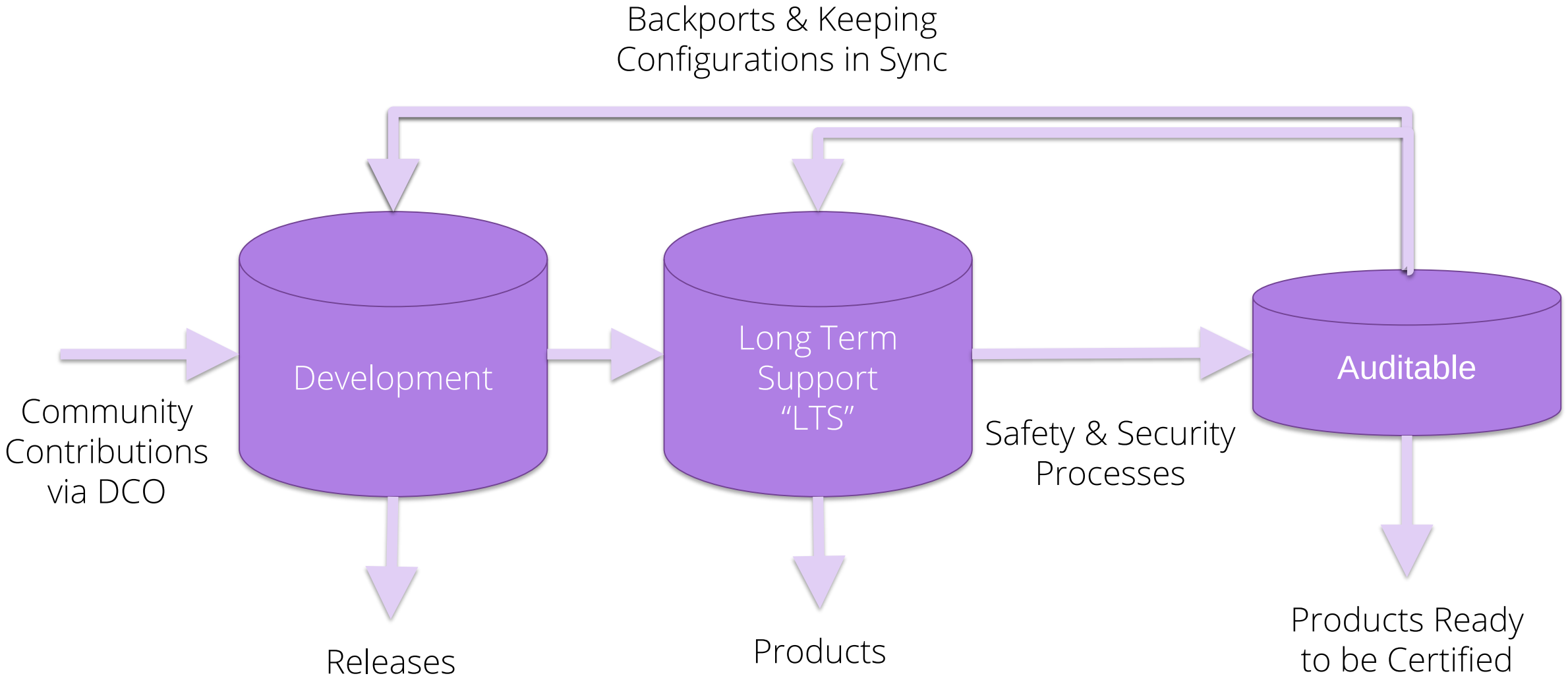
Zephyr Project:

- **Open source** real time operating system
- **Vibrant Community** participation
- Built with **safety and security** in mind
- **Cross-architecture** with growing developer tool support
- **Vendor Neutral** governance
- **Permissively** licensed - Apache 2.0
- **Complete**, fully integrated, highly configurable, **modular** for **flexibility**, better than roll-your-own
- **Product** development ready with LTS
- **Certification** ready with Auditable

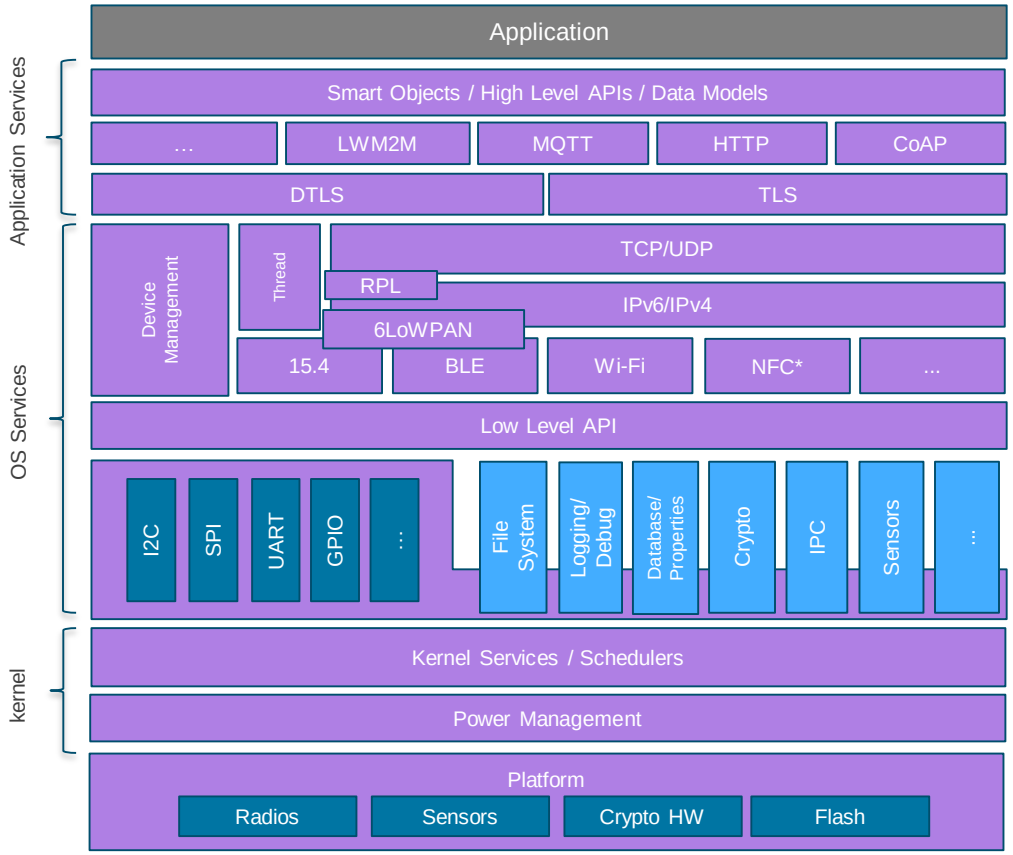
Open Source, RTOS, Connected, Embedded
Fits where Linux is too big

Code Repositories

"**Define and limit scope early**"



Architecture



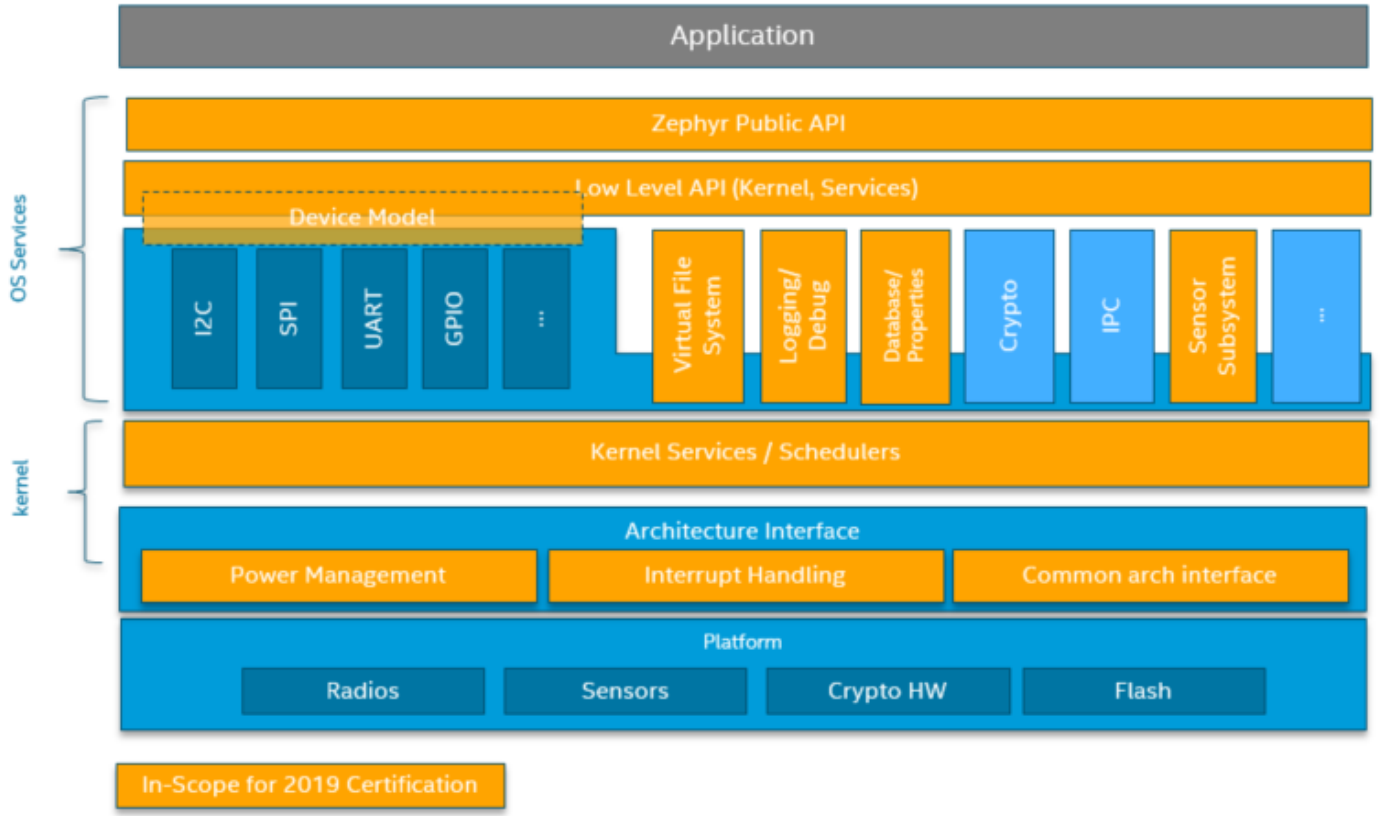
- Highly Configurable, Highly Modular
- Cooperative and Pre-emptive Threading
- Memory and Resources are typically statically allocated
- Integrated device driver interface
- Memory Protection: Stack overflow protection, Kernel object and device driver permission tracking, Thread isolation
- Bluetooth® Low Energy (BLE 4.2, 5.0) with both controller and host, BLE Mesh
- Native, fully featured and optimized networking stack

Fully featured OS allows developers to focus on the application

2019 Auditable Scope

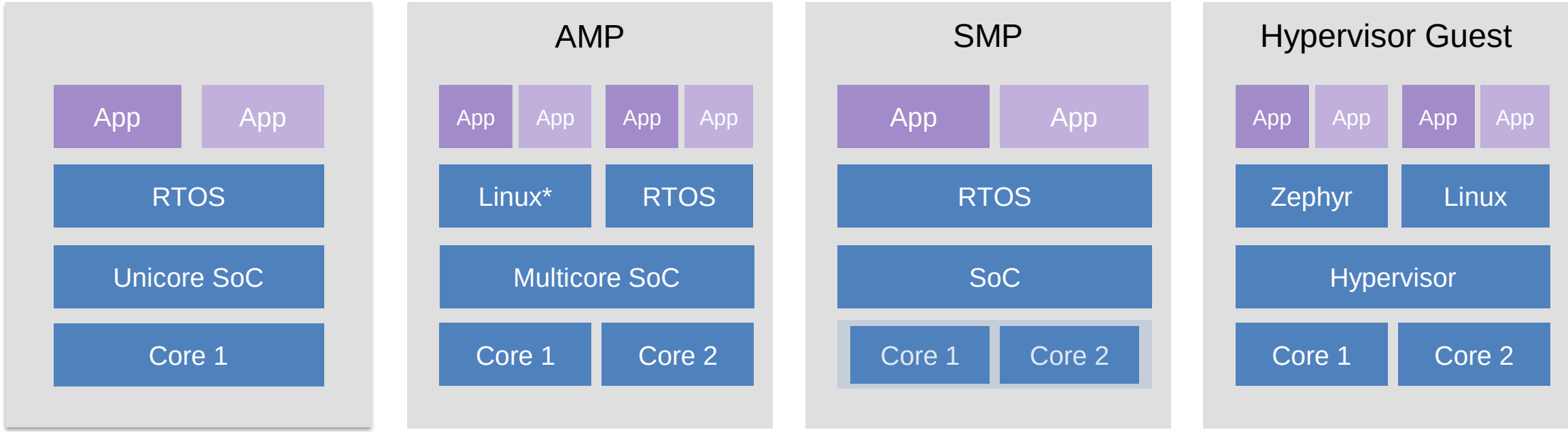
Not in scope:

- Platform drivers or BSPs
- No platform specific power management implementation, only device and kernel part of PM.
- No Filesystem or sensor driver implementation, only interface and infrastructure to support those on top of existing APIs



See: https://www.zephyrproject.org/zephyr-project-rtos-first-functional-safety-certification-submission-for-an-open-source-real-time-operating-system/ for more details

Zephyr Use Cases



Single Core MCU

Supported with OpenAMP

Supported on Xtensa* and x86_64

Supported with OpenAMP

Safety and security requirements grow with complexity of use cases

Zephyr OS: Long Term Support (LTS - 1.14)

It is:

- **Product Focused**
- **Current with latest Security Updates**
- **Compatible with New Hardware**: We will make point releases throughout the development cycle to provide functional support for new hardware.
- **More Tested**: Shorten the development window and extend the Beta cycle to allow for more testing and bug fixing

It is *not*:

- **A Feature-Based Release**: focus on hardening functionality of existing features, versus introducing new ones.
- **Cutting Edge**

Building in Security for LTS → Auditable

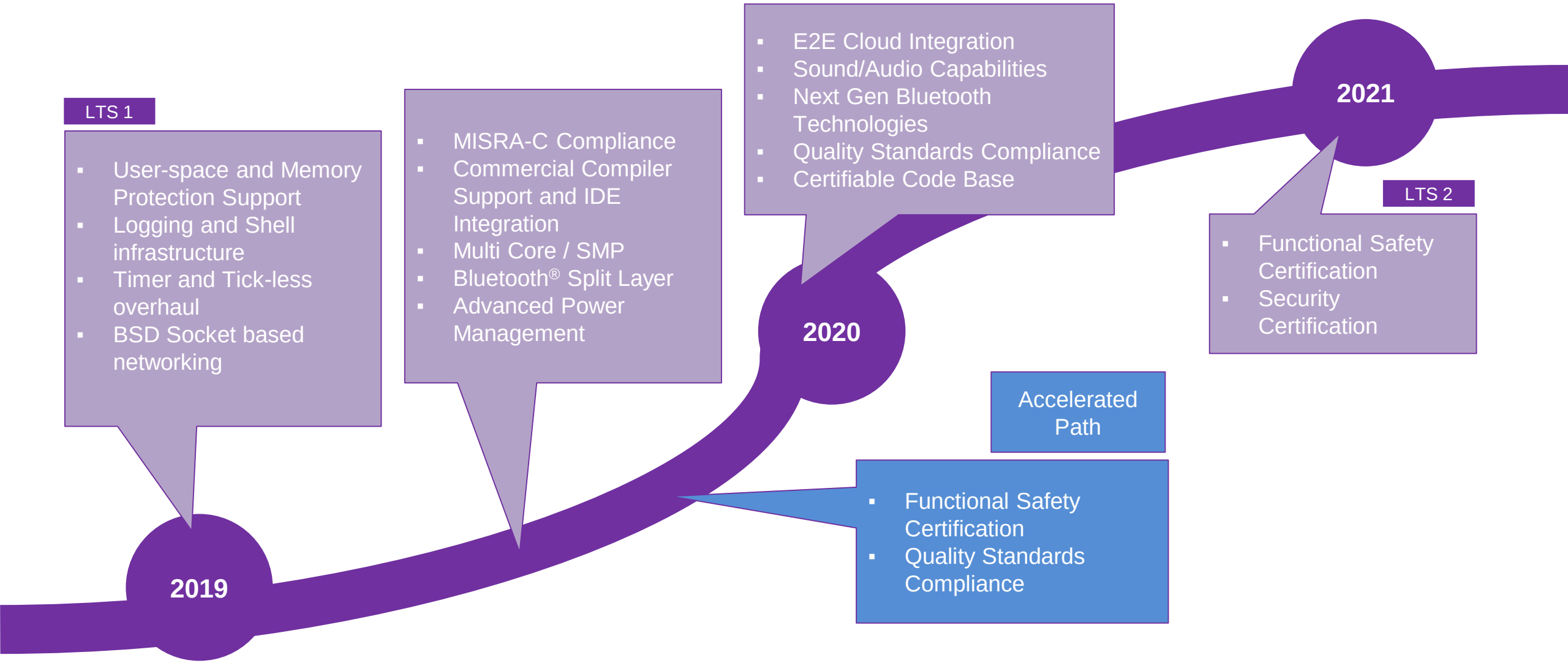
- Established Security Working Group, meets bi-weekly.
- Secure Coding Practices have been documented for project.
- Zephyr Project registered as a CVE Numbering Authority with Mitre.
- Security Working Group has vulnerability response criteria publicly documented
  - addressed weaknesses and vulnerabilities already
- Passing Best Practices for projects as defined by CII
  - https://bestpractices.coreinfrastructure.org/projects/74
- Leveraging Automation to prevent regressions:
  - Weekly Coverity Scans to detect bad practices in imported code
  - MISRA scans being incorporated, to evolve to conformance and address issues.

Zephyr OS: Auditable Code Base

- Initial and subsequent certification targets to be decided by Governing Board.
- An auditable code base is being established from a subset of Zephyr OS.
  - Code bases will be kept in sync.
  - More rigorous processes (necessary for certification) will be applied into the auditable code base, and then moved to development

Processes to achieve selected certification to be determined by Security Working Group and Safety Working Group in coordination with Technical Steering Committee.

Zephyr Project Roadmap



Larger footprint: How is ELISA doing it?

ELISA: Enabling Linux in Safety-Critical Applications

To assess whether your system is safe, you need to understand your system sufficiently.

➔ If your system‘s safety depends on Linux, you need to understand Linux sufficiently for **your system context and use**

Safety by Process Argument

Compliance to Objectives of Safety Standards by Development Process Assessment:

- ➔ Linux has been **continuously developed for 27+ years**
- ➔ **Continuous process Improvement is in place**.
  - ◆ When technical or procedural issues in the kernel development are identified and pressing, the community addresses them.
- ➔ Evidence for the **requisite process quality** and **process improvement quality exists** already.
- ➔ This **evidence can indicate** that all objectives of a safety integrity level 2 for **selected parts and properties are met**.

Safety-Critical Linux Process Approach

The difference between safety-critical Linux and main-line Linux **is the way you use it**.

- *Understand* your system and *understand* Linux
- Make sure your system uses Linux based on the ***selected*** properties of Linux where ***you can assure*** quality exists already.

How to Make Linux-Based Systems Safe

An Organization‘s shared knowledge of the system and Linux make the system safe

- ***Processes and Methods*** to understand:
  - the qualities of the complex software system
  - the qualities of the Linux kernel

- ***Education*** on these topics is the key to your safety product development.

Path forward for “Closing the Gap” & Culture Clash

Industry needs an operating system for complex algorithms and software suitable for safety-critical systems

Basis available:

- Functional safety is about managing risk in product development
- Risk in Linux-based systems can only be understood with knowledge of the system and kernel
- Starting point for understanding Linux in safety-critical systems is available

Collaboration proposal:

- Further development of the basis requires industry collaboration

ELISA Collaboration Goals

Shared development and effort on:

- Understanding safety engineering of complex systems
- Creating risk assessments of the kernel subsystems and features
- Gathering evidences of kernel development process compliance
- Developing supporting tools
- Creating material to train and educate engineers

Key Elements of an Effective Collaboration

- Establish well-defined governance and project steering
- Focus to establish and maintain good community health
- Keep educating on functional safety and process assessment
- Share a common system to focus on common activities
- Participation of safety experts, hardware vendors and Linux community

What Will Success Look Like?

Assets for safety certification of Linux-based systems

- consisting of a complete process, selected kernel features and tools, and previous process assessments
- shown feasible with a reference system(s)
- usable by properly educated system integrators
- maintained over industrial-grade product lifetimes
- well-known and accepted by safety community, certification authorities and standardization bodies in multiple industries
- positively recognised and impacting the Linux kernel community
- hardware collateral from multiple supporting vendors

Mission Statement

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

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.

Project Goals

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

- A set of elements, processes and tools to support safety certification of Linux-based systems.
- A Linux-based reference system to focus the activities of the project.
- Companies are able to incorporate the output of the project into products.
- The open source community, safety community, regulation authorities, standards bodies and system developers all accept the project work.

Project Deliverables

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

- Work with certification authorities and standardization bodies in multiple industries to establish how Linux can be used as a component in safety critical systems
- Continuous feedback and contributions to open source community
- Inclusion of the safety community to gain acceptance of project results
- Broad hardware vendor participation
- Tooling to support use of Linux in safety critical systems
- Incident and hazard monitoring
- Education and evangelism
- Reference documentation and use cases

Understanding the Limits

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

The collaboration:

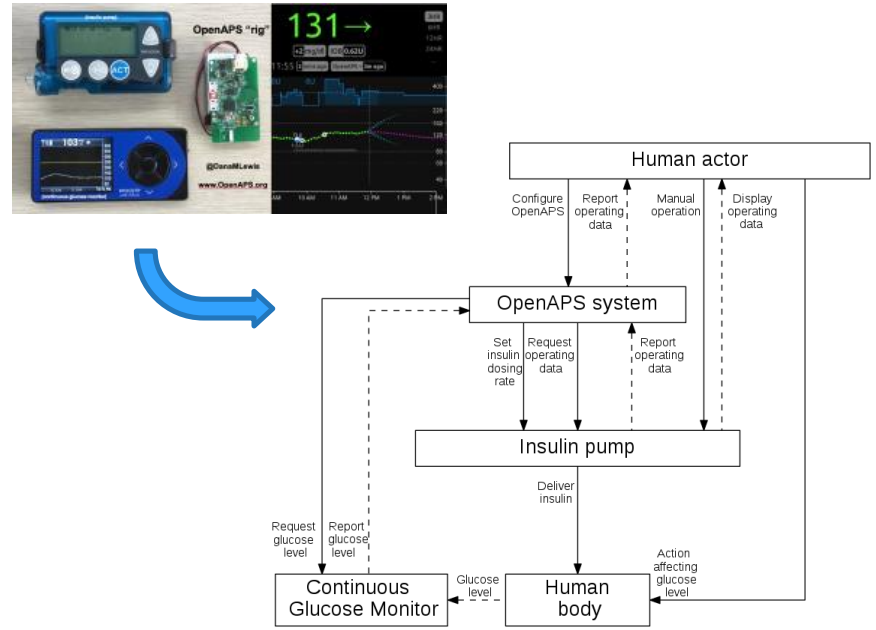
- ***cannot engineer*** your system to be safe
- ***cannot ensure*** that you know how to apply the described process and methods
- ***cannot create*** an out-of-tree Linux kernel for safety-critical applications (Remember the continuous process improvement argument!)
- ***cannot relieve*** you from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

But it will provide a **path forward** and peers to **collaborate** with!

Initial Outcomes from Workshop #1

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

- Use Case: **openAPS** (Artificial Pancreas System)
  - https://openaps.org/
  - Next steps:
    - STPA analysis of role of Linux in closed feedback loop located on Raspberry PI.
    - Identification of key interfaces and dependencies
    - Accurate identification of all components in system (we may need new tooling here: tracing technology and system SBOM dependency)



- Use Case: **Autonomous Driving**
  - Building on the SIL2LinuxMP work
  - Next steps: breaking problem down

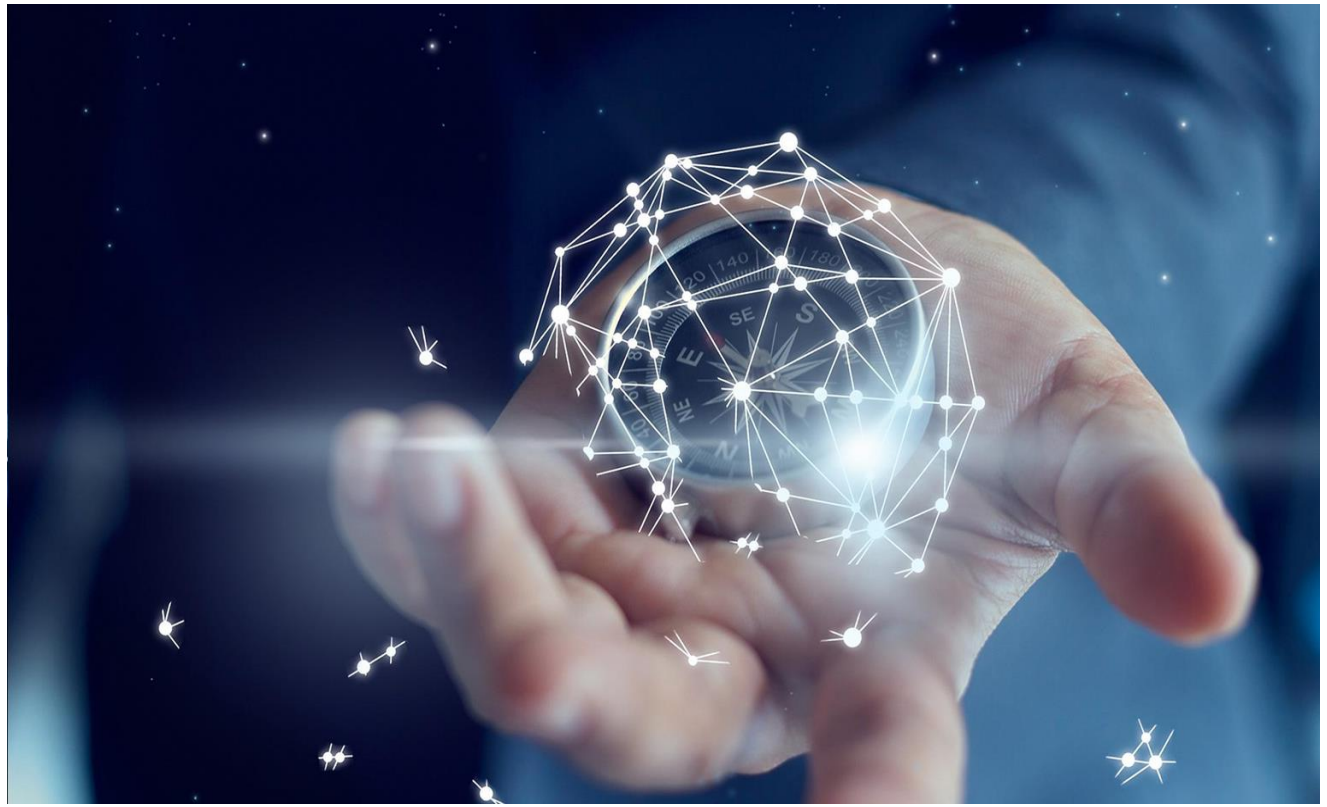
THE LINUX FOUNDATION

Image Source: https://diyps.org/2018/06/23/presentations-and-poster-content-from-danamlewis-at-2018ada/

Summary for Zephyr & ELISA

- Functional Safety and Security can **coexist** with open source projects
- Quality needs to be driven at the open source **project level**
  - Need to showcase quality processes and plans for process improvements publicly
- Manage **developer** and **certification authority** expectations
  - Work within a well defined certification scope and focus on interfaces to system.
  - Understand the system where you want to use certified open source and get early buy in on design from certification authorities.

Questions?



For more information on these Linux Foundation projects see:

- https://www.zephyrproject.org/
- https://elisa.tech/

Or contact:
stewart@linux.com

Thank you!