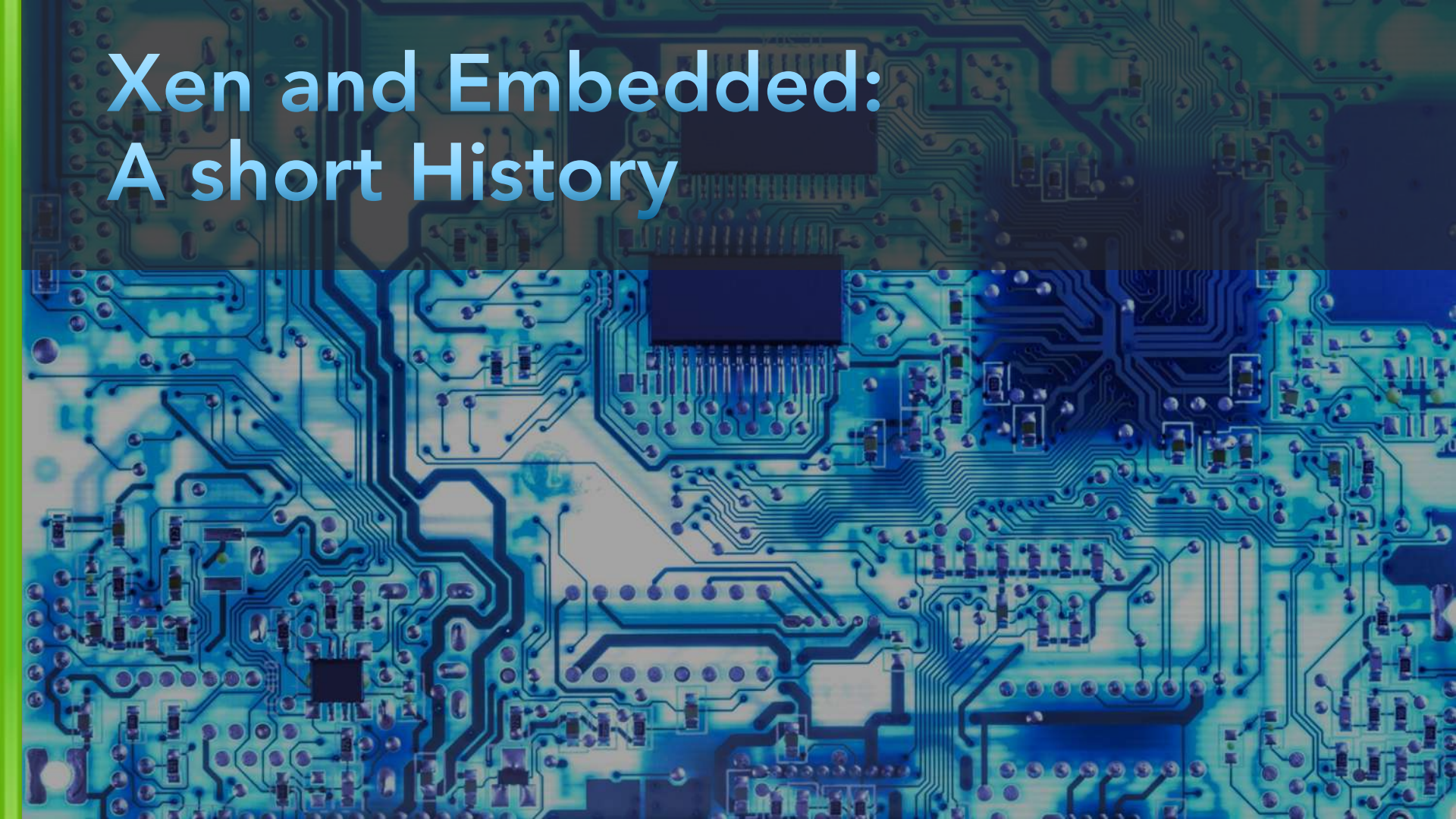
The Road to Safety Certification: Overcoming Community Challenges to Enable Safety Certification

Open Source Summit EU 2019

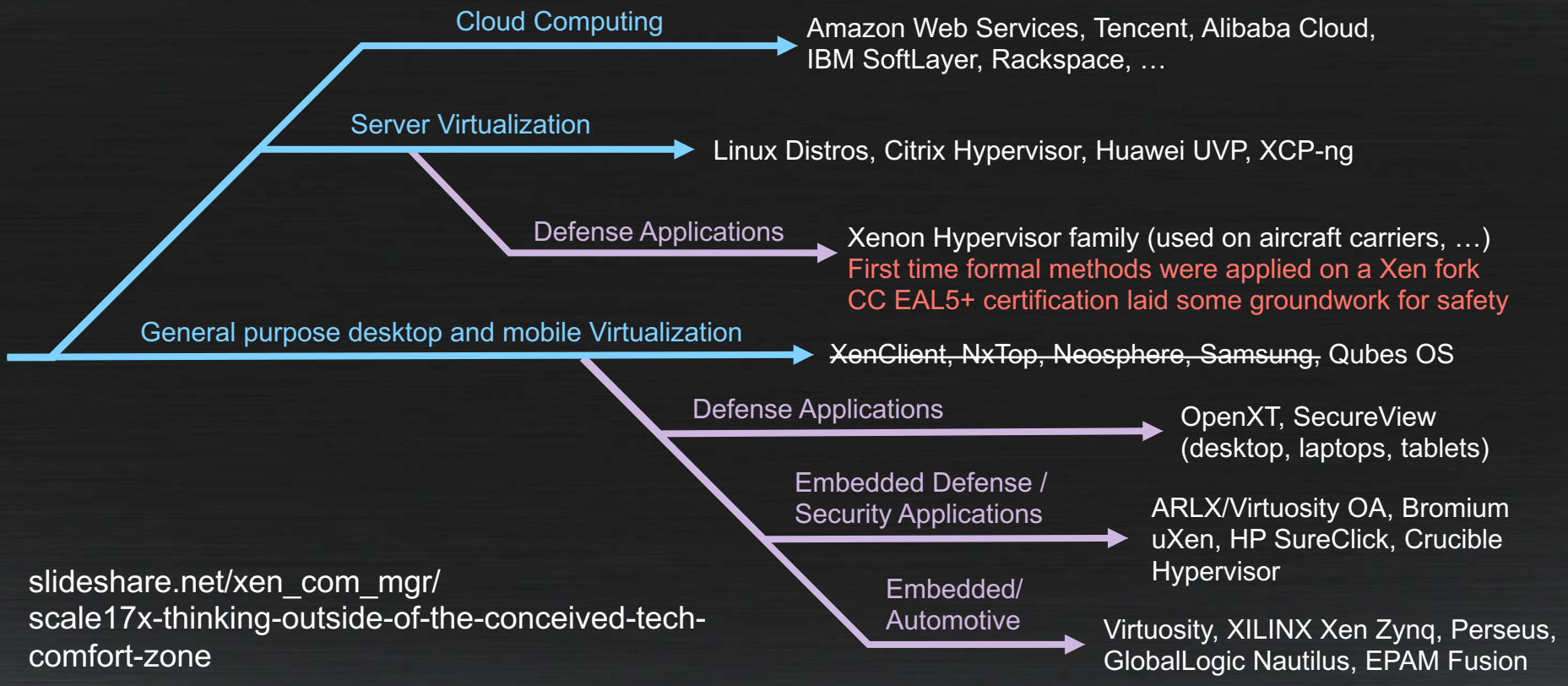
Lars Kurth
Community Manager, Xen Project
Chairman, Xen Project Advisory Board

lars_kurth

Xen and Embedded: A short History



Xen Ideas/Product Genealogy

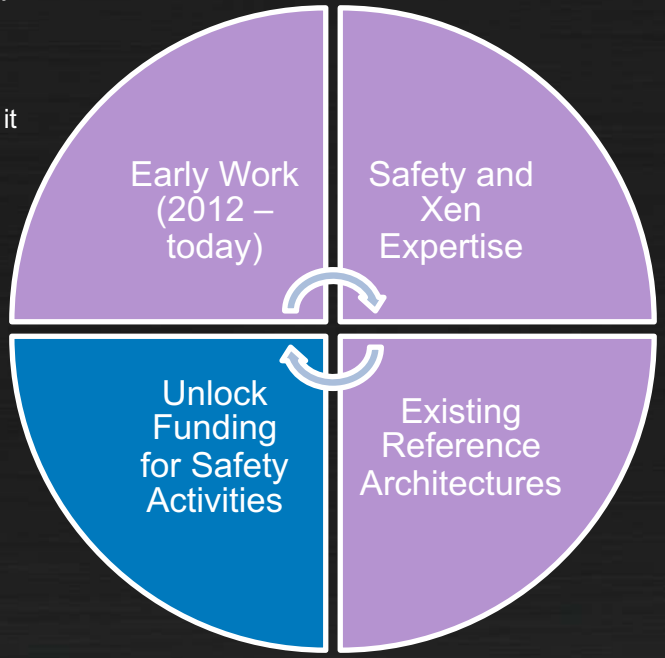


slideshare.net/xen_com_mgr/
scale17x-thinking-outside-of-the-conceived-tech-comfort-zone

Enablers for a Xen Safety Story

- Study by DornerWorks to establish feasibility of whether Xen on Arm could be certified to DO 178b Level A ➔ Cost matrix & Product family (ARLX, Virtuosity OA)
- Study by HORIBA MIRA to assess whether it is possible to safety certify a subset of the Xen Project ➔ EPAM ref platform
- Fill functional gaps (RT, reduce code size, configurability, …) ➔ Reference platforms

- NASA funds Dornerworks to integrate the Xen Project Hypervisor into NASA's new High Performance Space Computing Platform (HPSC)
- Arm announces Xen as key part of their safety reference stack
- Significant funding from a group of vendors to re-write Xen on Arm port for embedded likely (originally designed for servers)
- Other funding routes being considered (e.g. HORIZON 2020, US grants, …)



- Multiple consultancies which know the Xen codebase and various safety standards (DornerWorks, StarLabs.io and EPAM which is nascent)
- All have experience in upstreaming functionality to Xen
- Today: DO 178 centric

- **DornerWorks:** OpenGroup FACE certified Virtuosity OA (military)
- **XILINX:** generic embedded stack
- **EPAM:** automotive stack
- But: all open source, but not all is up streamed
- Some use in production:
  In a non-safety context
  In safety contexts where safety can be isolated outside of Xen

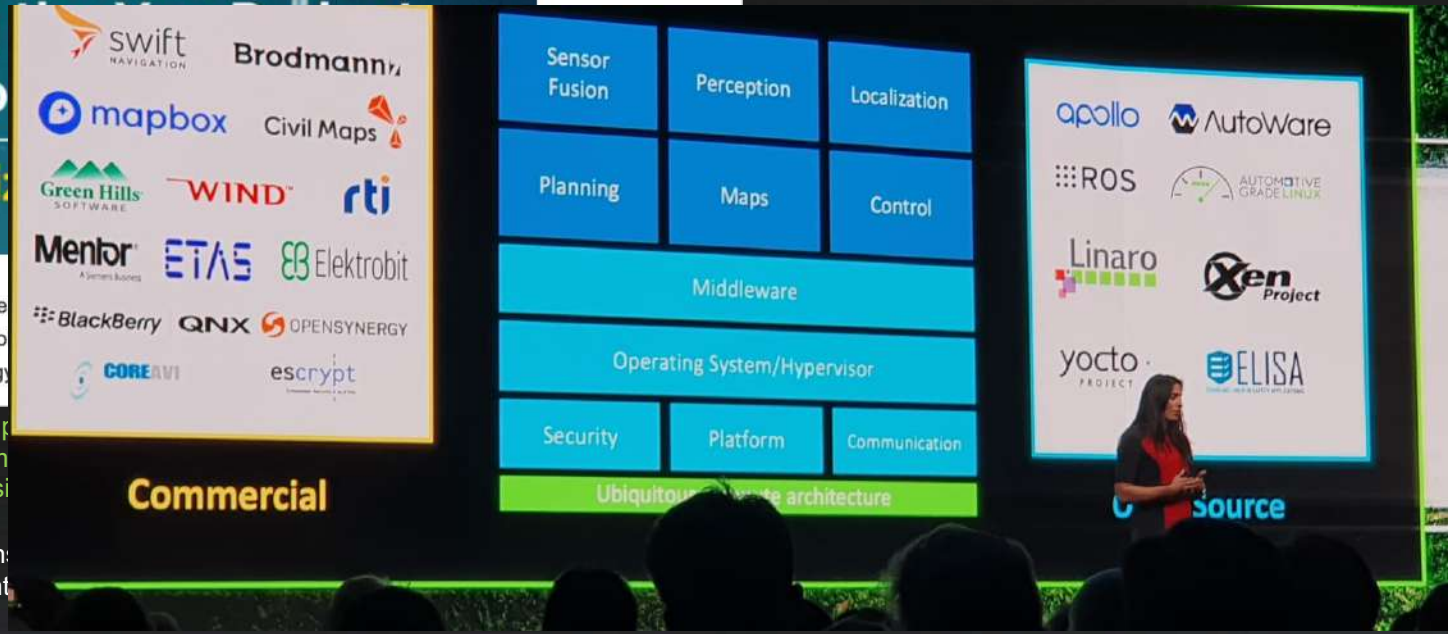
y Story

Home > About > Blog > Xen Project Hypervisor Headed for Space as DornerWorks Begins Phase I SBIR Project with NASA

Xen Project Hypervisor Headed for Space as DornerWorks Begins Phase I SBIR Project with NASA

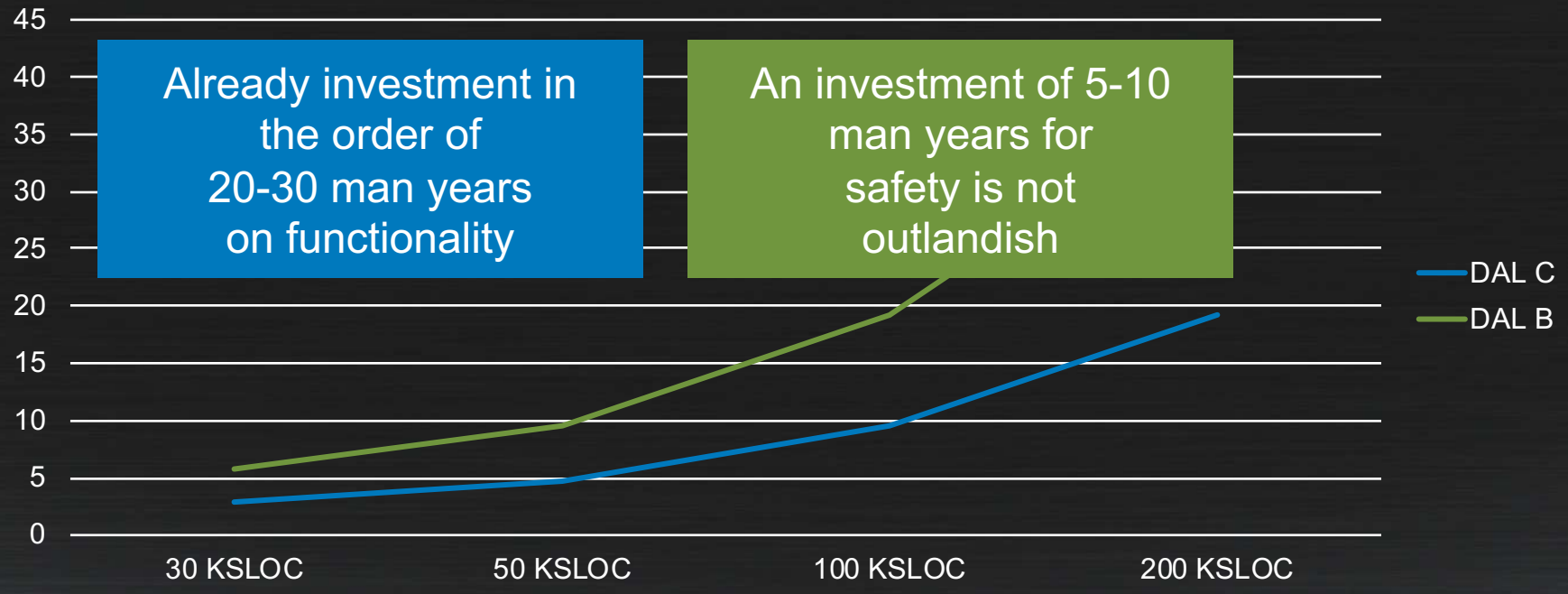
The moment you are barreling out of the e
your rocket's technology can handle the jo
it's hard to deny that space age technology

- Multiple consultancies which know the Xen codebase and various safety standards (DornerWorks, StarLabs.io and EPAM which is nascent)
- Significant funding from a group
vendors to re-write Xen on Arm
embedded likely (originally desi
servers)
- Other funding routes being cons
(e.g. HORIZON 2020, US grant



Certification Costs: Example DO-178

Cost in man years, based on DornerWorks study



Feature Examples specific to Embedded

Schedulers: ARINC, RTDS, Null and other real-time support
Laid the foundation for embedded use-cases and use of Xen as a partitioning HV
Low latency and real-time support

A minimal Xen on Arm Configuration
< 50 KSLOC of code for a specific HW environment

PV drivers (and in future virtio drivers) and GPU mediation for rich IO
Available in various upstreams

OP-TEE virtualization support
Both in Xen and in OP-TEE

Dom0less Xen
For now: allows booting VM's without interaction with Dom0, but Dom0 still exists
2020: an architecture without a Dom0 and/or an RTOS as Dom0

Feature Examples specific to Embedded

Schedulers: ARINC, RTDS, Null and other real-time support

Laid the foundation for embedded use-cases and use of Xen as a partitioning HV

Low

A r

< 5

PV

Ava

OP

Bot

Do

For now: allows booting VM's without interaction with Dom0, but Dom0 still exists

2020: an architecture without a Dom0 and/or an RTOS as Dom0

Key Point:

Xen on Arm, turned out to be a great open source hypervisor for embedded and mixed-criticality use-cases in theory

Despite having been designed for servers!

Safety Certification
The beginning of the journey

Functional Safety

Safety:
Freedom from unacceptable risk of physical injury or of damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment.

Safety Certification:
A safety assessment determines whether your product meets standards and performance requirements created to protect against potential risks, including injuries and even death.

Compliance:
is driven by customer requirements, legislation, regulations, and insurance requirements

FOSS SW and Functional Safety

Requires major changes to the software

Requires tools, infrastructure and expertise

Funding ↔ Confidence

Requires changes in how FOSS projects work

Until recently: assumption was that the two worlds cannot work together

Community Challenges ↔ Trust & Confidence

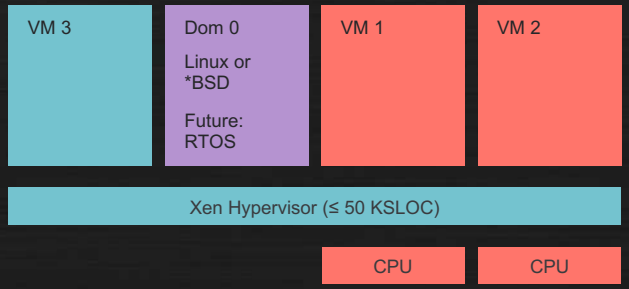
Tooling has a huge impact on Community Challenges

We need tools (ideally FOSS tools) that fit into our Git and CI workflow

Tools Challenges ↔ Funding

Mixed Criticality case

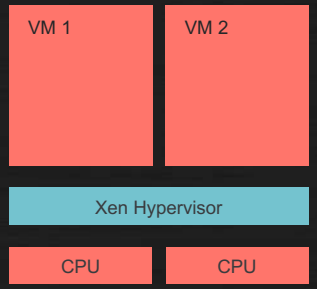
Dom0less VMs (today)



Dom0less VMs loaded by uBoot and booted by Xen (not Dom0), pinned to a CPU via the Null scheduler and I/O handled by device assignment

Dom0 completes boot after VM 1 and VM 2. Static set-up

True Dom0less (2019/20)



Ongoing work to fully implement true Dom0less for small systems

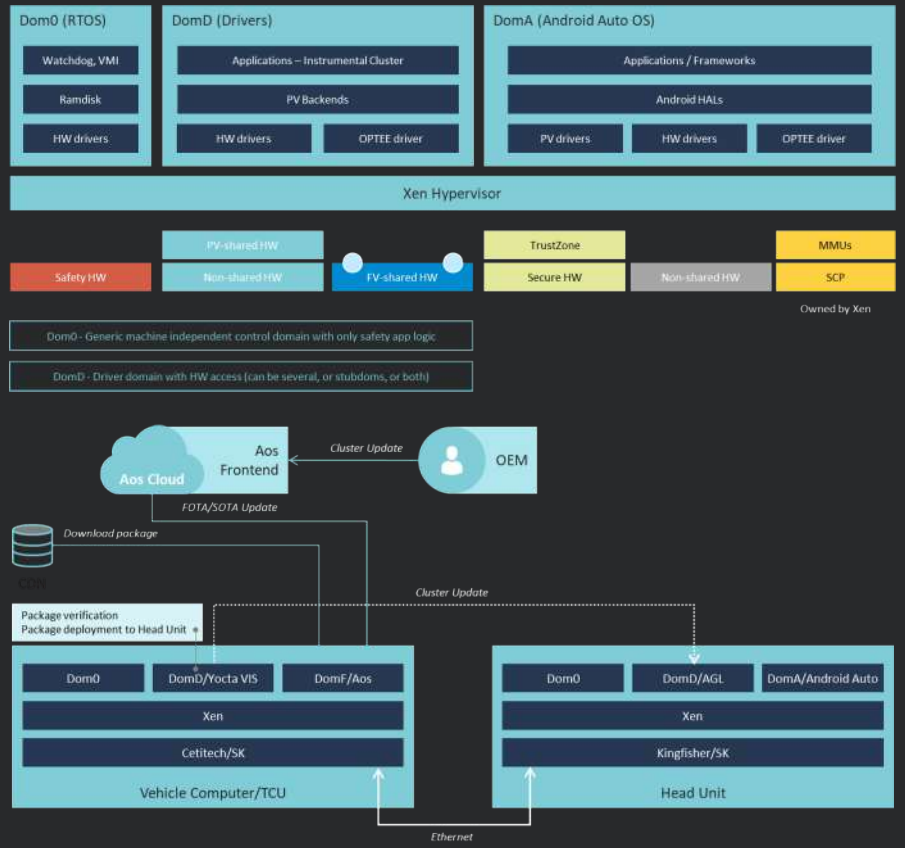
- Shared memory and interrupts for VM-to-VM communications
- PV frontends/backends drivers for Dom0-less VMs

Dom0less initial safety certification scope
Arm64 only

slideshare.net/xen_com_mgr/elc2019-static-partitioning-made-simple

Automotive Case

Mix Safety Digital Cockpit In-Vehicle Computer



AUTOMOTIVE GRADE LINUX

Xen Project

android auto

FuSa SIG with Workstreams

Subgroups meet at least every other week. Partly resourced

Community Reps
Lars Kurth (chair and project mgmt)
George Dunlap (committers)

Stream Owners and Implementers
Lars Kurth

Assessors

Other Members

2-day workshop in March 2019

Create a understanding between the community and industry

Terminology, Concepts, etc.
How safety certification works: look at different standards, routes, requirements
Explain assets and processes

Establish community “red lines”

Principles the community can agree to or would object to
What level of change would be acceptable
Identify potential obstacles

High Level Agreements

Split development model with an open and a closed part

Everything that is valuable to the wider community **ideally** in the open part,
e.g. documentation, **some** tests, traceability, automation and infrastructure,….

Everything that creates code churn if it wasn't open as much as possible:
e.g. coding standards (MISRA)

Changes to the development workflow have to be kept minimal

There must be a benefit the community
Otherwise the community wont carry

There are long-term implications for the community

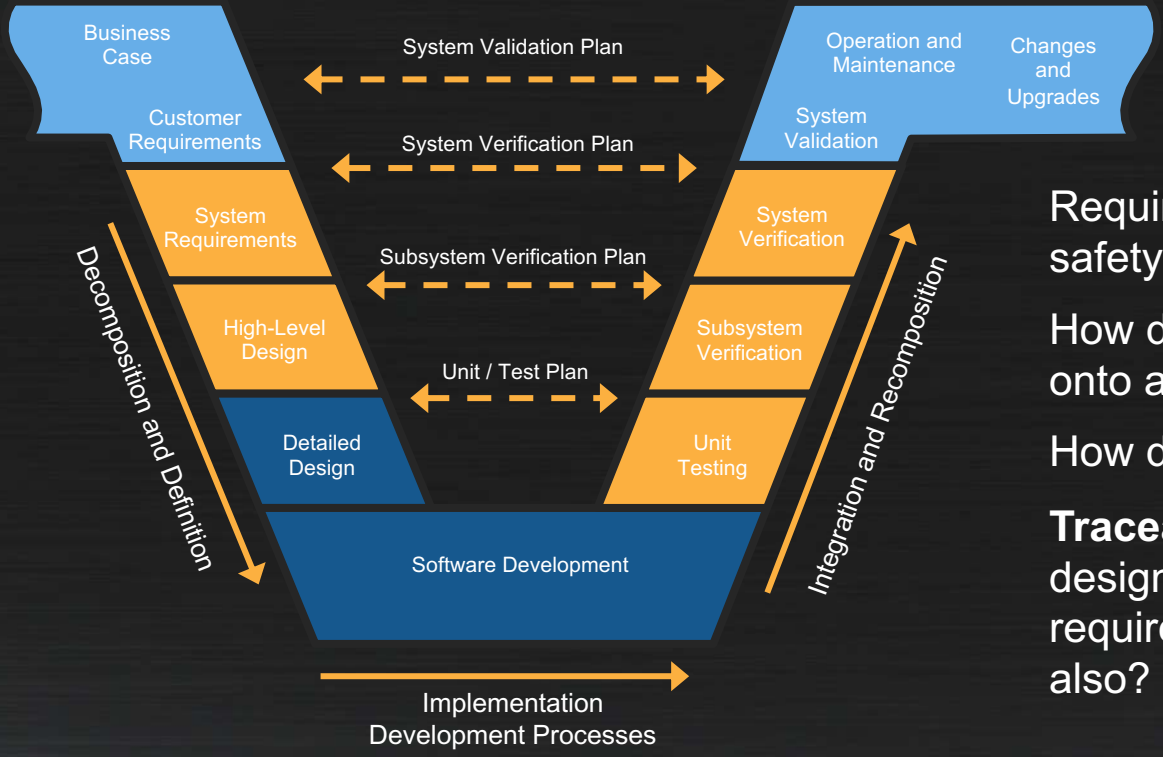
Make-up, scalability, decision making, conflicts – need to be managed
No major new barriers for contributors can be introduced

Goal:
significantly reduce the cost for platform integrators to safety certify Xen derivates

Share as much burden as possible by collaborating upstream

Examples of Challenges that need to be overcome

Development Process and Traceability



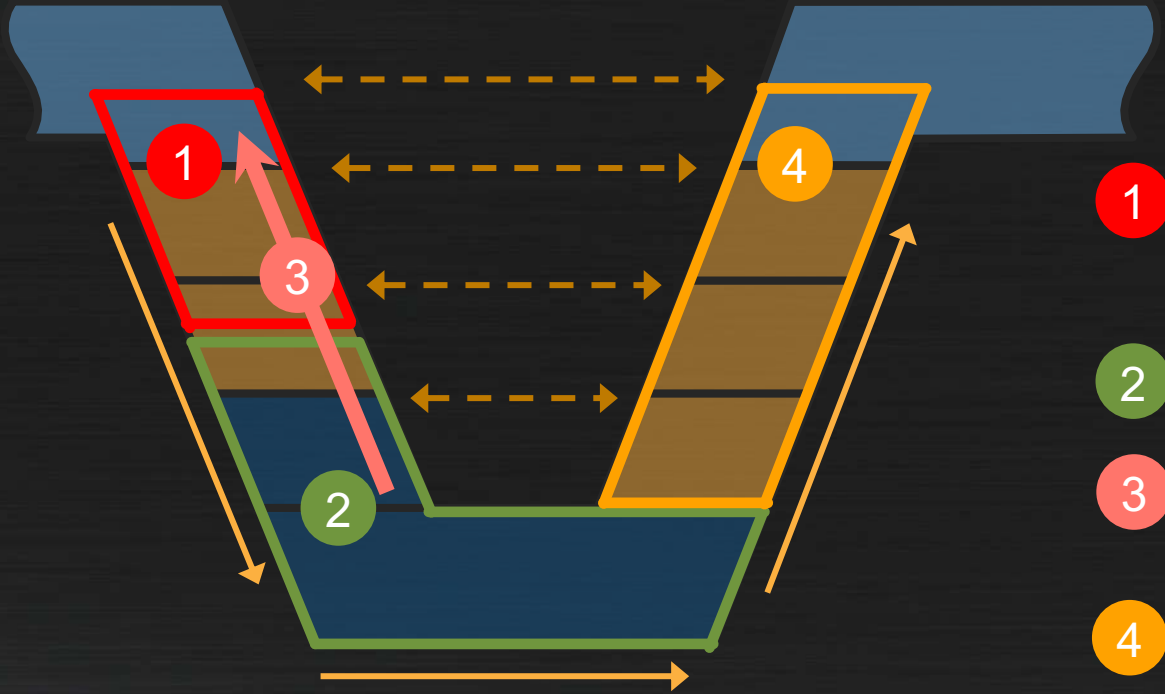
Required by some, but not all safety standards

How do you map this onto a FOSS development process?

How do you get community buy-in?

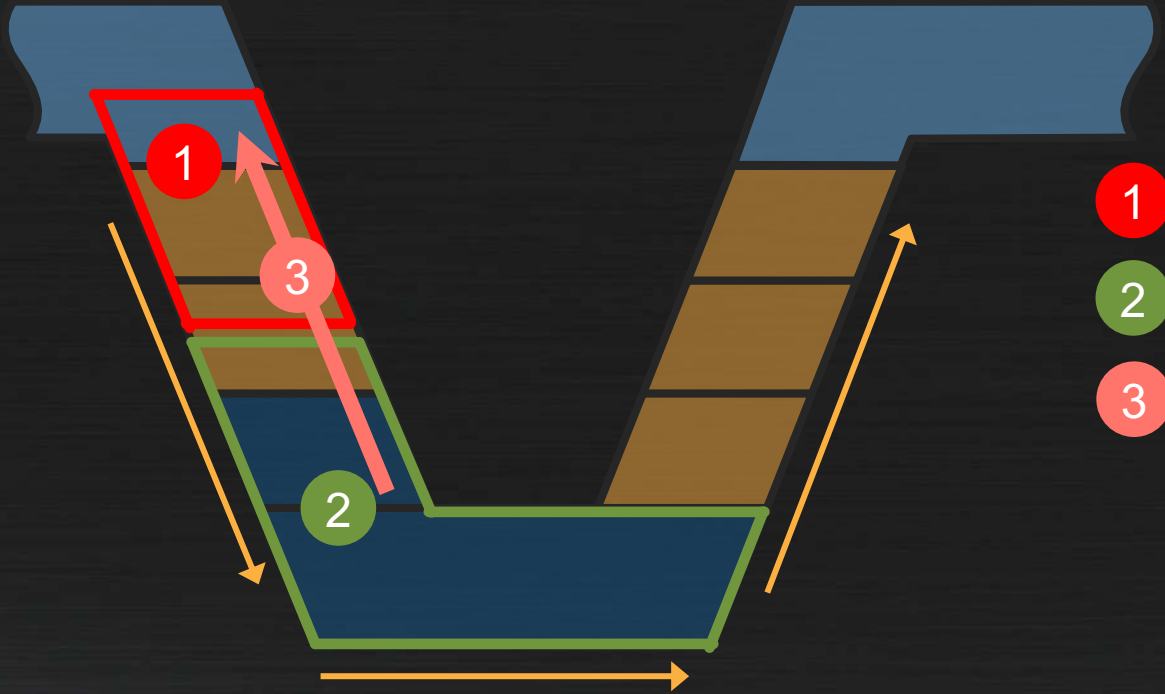
Traceability: how do you prove that design and architecture satisfies requirements and tests verify these also?

What you normally have in FOSS is …



1. Not at at all, or outside
   Not a huge effort to retrofit
   Valuable for developers & users
   Does not change often for a Hypervisor
2. Frequently as good or better than proprietary. Process discipline
3. Not at all. Difficult to maintain manually. Should not change that often for core functionality, once done
4. A subset of this usually exists, but typically tests **code, not requirements/specifications**. That's the most expensive part to address.

What must be upstream: all key inputs …



1 **Documented Requirements**

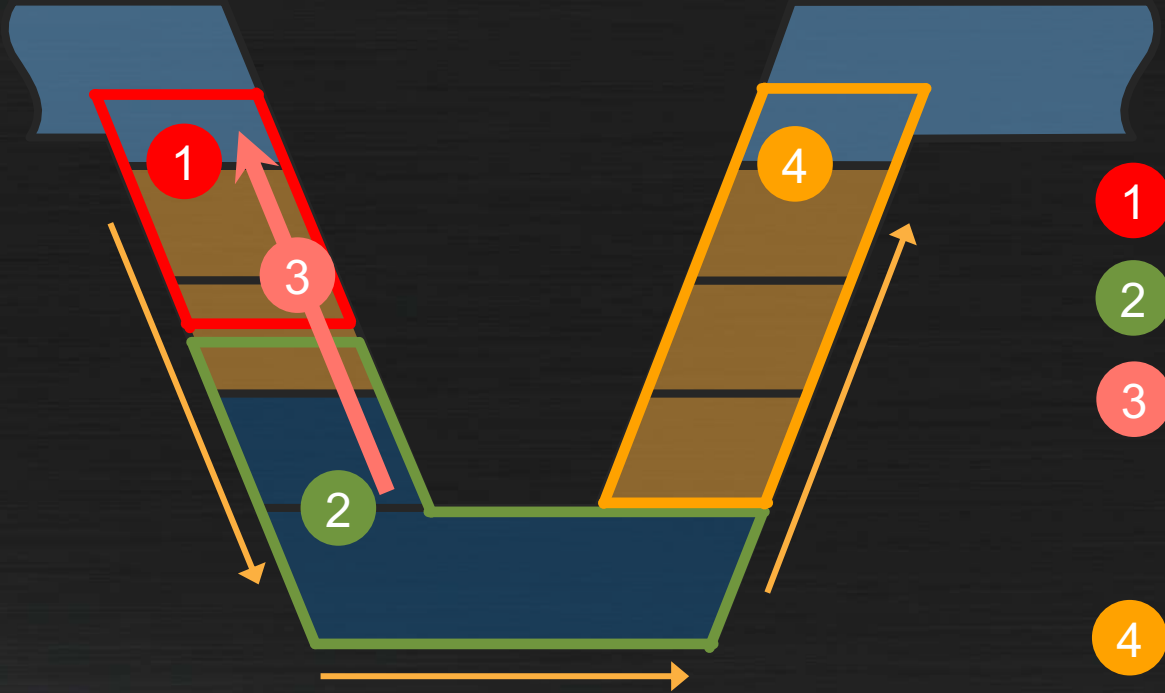
2 **Design, Architectural and API documentation**

3 **Traceability info:**
Between requirements
Between requirements and other docs
Between requirements and code

With appropriate tooling and Information Architecture this can be done in a git-workflow

Candidate tool: DOORSTOP

What must be upstream: all key inputs …



1 **Documented Requirements**

2 **Design, Architectural and API documentation**

3 **Traceability info:**
Between requirements
Between requirements and other docs
Between requirements and code

4 **Validation:**
Can be outside of upstream
Needs a feedback loop to deal with breakage – like OpenStack 3rd party CI

Community Challenges: MISRA C

Picked MISRA C as an example, because …

it is representative of the hardest type of community problems that you should expect if you look at safety certification

Picked hardest and controversial rules to see what would happen!

We did not expect to succeed !

We got stuck early on

MISRA C spec is proprietary

Rule text cannot be copied into a posted patch series ➜
lack of clarity, lack of rationale: leading to unnecessary debate

Interactions w compilers, HW, assembly code problematic

Ended up with 11 iterations and man weeks of review effort

Deviations

Possibility of MISRA C Deviations encourage arguments

Deviations: justification of a class or instance of non-compliance
Deviation Permits: previously approved deviations for a use-case

It's all a bit like like "legal precedent" in common law legal systems:
an expert (assessor) is needed to advise the project on a case-by-case basis

Assessors to advise project are lined up working through difficult example cases

Have core maintainers that have become "advocates"

Round 2: 5000 issue instances

4500 come down to a single hard rule violation

Coding Standard vs Misra

Our coding standard violates MISRA C:2012, 15.6 (95% of Xen MISRA C violations on scope config)

Fixing this causes downstream pain: patch queues, backports, …

```
if ( flag_1 )
    if ( flag_2 )
        action_1 ( );
    else
        action_2 ( );
```

```
if ( flag_1 ) {
    if ( flag_2 ) {
        action_1 ( );
    }
    else {
        action_2 ( );
    }
}
```

Coding Standard vs Misra

Our coding standard violates MISRA C:2012, 15.6 (95% of Xen MISRA C violations on scope config)

Fixing this causes downstream pain: patch queues, backports, …

Choice we have: argue for an exception OR inflict pain on down streams

Exception:

use **-Wmisleading-indentation in GCC 6** to address issue rationale

work with a tool vendor (obvious choice is Arm) to add an equivalent of **-Wmisleading-indentation** to a qualified compilation toolchain

}

Community Scalability

Code review process encourages too much discussion, if there is no up-front plan on how to approach a disruptive set of changes

500 issues to be fixed

- ¼ man year to create fixes (1h per instance)
- On average 2 hours per code review instance = ½ man years by code reviewers from Suse, Citrix, Amazon that could be spent more productively

Need a better approach that is more efficient that focuses on classes of issues, not instances


Safety Certification
Creating a credible plan …

Low tailoring route

Candidates: IEC 61508 or ISO 26262

Build Confidence and Unlock Funding / solve Community problems iteratively

Chicken and egg problems

CI Loop changes

Front-load CI: do as much as possible **before** code review (in progress)
Use bots and automation (in progress)
More tests in “simulated environments” – capacity problem with HW
3rd party CI loop hooks

Folks interested in safety are stepping up to solve long-standing lingering community problems

Coding Standards

Need a process to prioritize rule implementation

Compliance tooling and reporting that fits into CI (issue: © of MISRA)

Goals: Minimize unnecessary discussion, disruption and deliver security benefits to established Xen users

Focus on left side of V model first

While refreshing the Xen on Arm port at the same time

- Effort to identify key APIs and improve documentation (started)
- Code review map (started)

Need docs & traceability tooling story:

- Working through a system SW example (no public examples, but have access to a sanitized relevant example now)
- Ideally a cross-project standard **using tool**s and **Information Architecture**
- **Goal:** Make it easy to keep artefacts up-to-date and integrate validation into CI
  Delivers benefits to established Xen users and developers

Standard Tailoring (being set up)

Gap Analysis with small group of maintainers and assessors
For gaps: clarify community red lines
Based on the outcome: tailoring strategy

Areas which are not yet clear

Testing and Validation
Safety management system that can coexist with generic Xen mainline development
…


Questions
Picture by Lars Kurth


Backup stuff

Certification Costs: Example DO-178b

Level	Requirements	Application	Cost with Experience
DAL E	The software must exist	**Infotainment** Failure is a minor inconvenience	0.11 hour / SLOC
DAL D	High-Level Docs/Tests	**Instruments** Failure can be mitigated by operator	0.13 hour / SLOC
DAL C	Low-Level Docs/Unit Tests, Statement Coverage, and Code/Data Coupling Analysis		0.20 hour / SLOC
DAL B	Branch Coverage	**Engine Control** Failure could kill someone without warning	0.40 hour / SLOC
DAL A	Source to Object Analysis and MC/DC Coverage		0.67 hour / SLOC

Credit/Source: Dornerworks / XPDS14 - Xen and the Art of Certification.pdf

Certification Costs: Example DO-178b

Level	Requirements	Application	Cost with Experience
DAL E	The software must exist	**Infotainment** Failure is a minor inconvenience	0.11 hour / SLOC
DAL D	High-Level Docs/Tests	**Instruments** Failure can be mitigated by operator	0.13 hour / SLOC
DAL C	Low-Level Docs/Unit Tests, Statement Coverage, and Code/Data Coupling Analysis	3-4 times as much without experience	0.20 hour / SLOC
DAL B	Branch Coverage	**Engine Control** Failure could kill someone without warning	0.40 hour / SLOC
DAL A	Source to Object Analysis and MC/DC Coverage		0.67 hour / SLOC

Credit/Source: Dornerworks / XPDS14 - Xen and the Art of Certification.pdf