The Road to Safety Certification: How the Xen Project is Making Progress within the Auto Industry and Beyond

Open Source Summit NA 2019

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

IRC: lars_kurth
Why Virtualize in Embedded Systems?
Consolidation
Reduce cost, size, weight and power consumption
Reduce development costs: platform independence

Security and Safety
Support mixed criticality compositions
(Apps with differing safety, security & real-time requirements)
Safety Certification of the Hypervisor

Embedded Requirements
Minimal IRQ latency
Low or 0 scheduling overhead
Drivers for special I/O devices
Flexible architecture
Xen and Embedded: A short History
Xen Ideas/Product Genealogy

Cloud Computing
Amazon Web Services, Tencent, Alibaba Cloud, IBM SoftLayer, Rackspace, …

Server Virtualization
Linux Distros, Citrix Hypervisor, Huawei UVP, XCP-ng

Defense Applications
Xen Hypervisor family, Magrana Server, …
First time formal methods were applied on a Xen fork
CC EAL5+ certification laid some groundwork for safety

General purpose desktop and mobile Virtualization
XenClient, NxTop, Neosphere, Samsung, Qubes OS

Defense Applications
OpenXT, SecureView (desktop, laptops, tablets)

Embedded Defense / Security Applications
ARLX/Virtuosity OA, Bromium uXen, Crucible Hypervisor
Various Safety Standards

Embedded/Automotive
Virtuosity, XILINX Xen Zynq, Perseus, GlobalLogic Nautilus, EPAM Fusion

slideshare.net/xen_com_mgr/
scale17x-thinking-outside-of-the-conceived-tech-comfort-zone
Future Airborne Capability Environment (FACE™) defines the software computing environment and interfaces designed to support the development of portable components across the general-purpose, safety, and security profiles. FACE uses industry standards for distributed communications, programming languages, graphics, operating systems, and other areas as appropriate.
2012 DornerWorks ARLX
DO-178 with some level A packages, IEC 62304, ISO 26262, MILS EAL, ARINC 653

2016 Virtuosity OA
OpenGroup FACE certified

2016 Star Lab Crucible
Secure embedded virtualization platform for security-critical operational environments, including aerospace & defense, industrial, transportation, and telecommunications
2012 DornerWorks ARLX

DO-178 with some Level A packages, IEC 62304, ISO 26262, MILS EAL, ARINC 653

Virtuosity OA

OpenGroup FACE certified

2016 Star Lab Crucible

2015 Xilinx: Petalinux with Xen

1st Xen distro for embedded with additional functionality
Currently NO safety certification support
2012
DornerWorks ARLX
DO-178 with some Level A packages, IEC 62304, ISO 26262, MILS EAL, ARINC 653

2016
Star Lab Crucible

2015
Xilinx: Petalinux with Xen
1st Xen based stack for automotive
No safety certification

2015
GlobalLogic
No safety certification

2017
EPAM
2nd generation Xen based stack for automotive. No safety certification, but working with community and industry on progressing safety

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

The moment you are barreling out of the earth’s atmosphere at 17,600 miles per hour isn’t the best time to wonder if your rocket’s technology can handle the job. Neither is the question comforting once you’ve made it to outer space, but it’s hard to deny that space age technology is a decade or more behind when it comes to processing power.

dornerworks.com/blog/high-performance-space-computing-platform-nasa-sbir
Summary

2016:
EPAM and Renesas funded a study by HORIBA MIRA to assess whether it is possible to safety certify a subset of the Xen Project

Answer: possible

From 2015 – today:
Close functional gaps, real-time capability, reducing code-size and create reference implementations (EPAM, XILINX, DW)

Answer: suitable platform for some use-cases
Number of gaps to be a general purpose platform still worked on

All is open source, but not all is upstreamed in Xen
The impact on the Xen Project
Features 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
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

Key Point:
Xen on Arm, turned out to be a great open source hypervisor for embedded and mixed-criticality use-cases
Despite having been designed for servers!
Safety Certification
The Final Frontier
Attempts to solve this problem

**FreeRTOS / SafeRTOS**
FreeRTOS-compatible alternatives from Wittenstein
SafeRTOS: proprietary FreeRTOS-rewrite complying with IEC 61508

**SIL2LinuxMP**
Can Linux be Safety certified? Obstacles, tools and processes

**LF Projects with an ambition to become ”easy to certify”**
ACRN
AGL – Virtualization may make achieving key AGL UCs easier
ELISA Project – Develop tools and processes
Xen Project
Zephyr

Each with different history, cultures and problems that have to be overcome
FOSS SW and Functional Safety

Can FOSS SW be used for Functional Safety?
Yes, but there are many barriers

- Requires major changes to the software
- Requires tools, infrastructure and expertise
- Requires changes in how FOSS projects work
  Until recently: assumption was that the two worlds cannot work together

Community Challenges
## Certification Costs: Example DO-178

<table>
<thead>
<tr>
<th>Level</th>
<th>Requirements</th>
<th>Application</th>
<th>Cost with Experience</th>
</tr>
</thead>
</table>
| 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 | Engine Control  
Failure could kill someone without warning | 0.20 hour / SLOC     |
| DAL B | Branch Coverage |                                | 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-178

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

Credit/Source: Dornerworks / XPDS14 - Xen and the Art of Certification.pdf
Already investment in the order of 20-30 man years on functionality

An investment of 5-10 man years for safety is not outlandish
Safety Certification
The beginning of the journey
Xen Project’s starting point

Examples of Xen based embedded products
With some support for safety standards in proprietary spin-offs

Expertise in ecosystem that covers Xen and Safety
Primarily for hire: too small to fund speculatively

Reference implementations with safety in mind
EPAM Stack (automotive), XILINX Stack, Dornerworks NASA stack
Another similar effort in progress elsewhere (generic safety case)

Some limited adoption in niche use-cases today
In a non-safety context
In safety contexts where safety can be isolated in progress
**Mixed Criticality case**

**Dom0less VMs (today)**

- VM 3, Dom 0, VM 1, VM 2
- Xen Hypervisor (≤ 50 KSLOC)
- CPU

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)**

- VM 1, VM 2
- Xen Hypervisor
- CPU

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

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

Mix Safety Digital Cockpit
In-Vehicle Computer
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**

[Logos of collaborating organizations]
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 new barriers for contributors can be introduced
Examples of Challenges that need to be overcome
Development Process and Traceability

How do you map this onto a FOSS development process?

How do you get community buy-in?

How much can be tailored within ISO / IEC?

Agile and ISO / IEC can provide a model which may fit

**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 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

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
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.
Coding Standards: MISRA C

Subset required by most safety standards

10 Mandatory, 111 Required and 38 Advisory rules
Required rules depend on certification level can be deviated from
Justifications of deviations would have to be signed off by an assessor

Partnership with Perforce: access to QA Verify providing
selected community members to results on Xen snapshots

Goal: Experiment and Learn
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
Bike shedding and strong opinions

Some rules will create a flame-war if there is a single argumentative maintainer

E.g. MISRA C:2012, 15.7
"if ... else if" constructs should end with "else" clause

```c
if (x == 0) {
    doSomething();
} else if (x == 1) {
    doSomethingElse();
} else {
    error();
    /* or justification why no action is taken */
}
```
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

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

Fix: A priori agreed strategy and plan on how to approach this
Safety Certification
Creating a credible plan ...
Xen Project Approach

Low customization route
Investigating best customization routes

Focus on left side of V model first
While refreshing the Xen on Arm port at the same time (funding secured)

Collaboration
Collaborate with ELISA and other projects such as Zephyr on common problems and standards

Build Confidence and
Unlock Funding / solve Community problems iteratively
Chicken and egg problems: similar to where Linux was in late 90s
Questions
References

Dom0-less Xen
slideshare.net/xen_com_mgr/elc2019-static-partitioning-made-simple

Xen Project Plans
xenproject.org/developers/teams/embedded-and-automotive

Dornerworks - NASA
dornerworks.com/blog/high-performance-space-computing-platform-nasa-sbir