Embedded Linux Conference

Meet an all scenarios os: a distributed OS with feet on the ground

Davide Ricci

Open Source Technology Center, Huawei

THE LINUX FOUNDATION

Disclaimer

Don’t get too attached to “an all scenarios os” nickname because it is temporary and will soon be gone

(thanks, you have served us well “an all scenarios os” but it’s time to move on)

New project name will be announced during EclipseCon 2021 – 25th to 28th of October

as a follow up to recent announcements of collaboration between the Eclipse Foundation and the Open Atom Foundation.

IoT - Problem Statement

- Dumb
- Technological fragmentation
- Reinventing the o.s. wheel
- Lack of interoperability at the edge
- Partial, brand-centric interoperability
- Cloud centric: compute, interoperability
- Top-down, cloud providers driven
- Inefficient, insecure, expensive

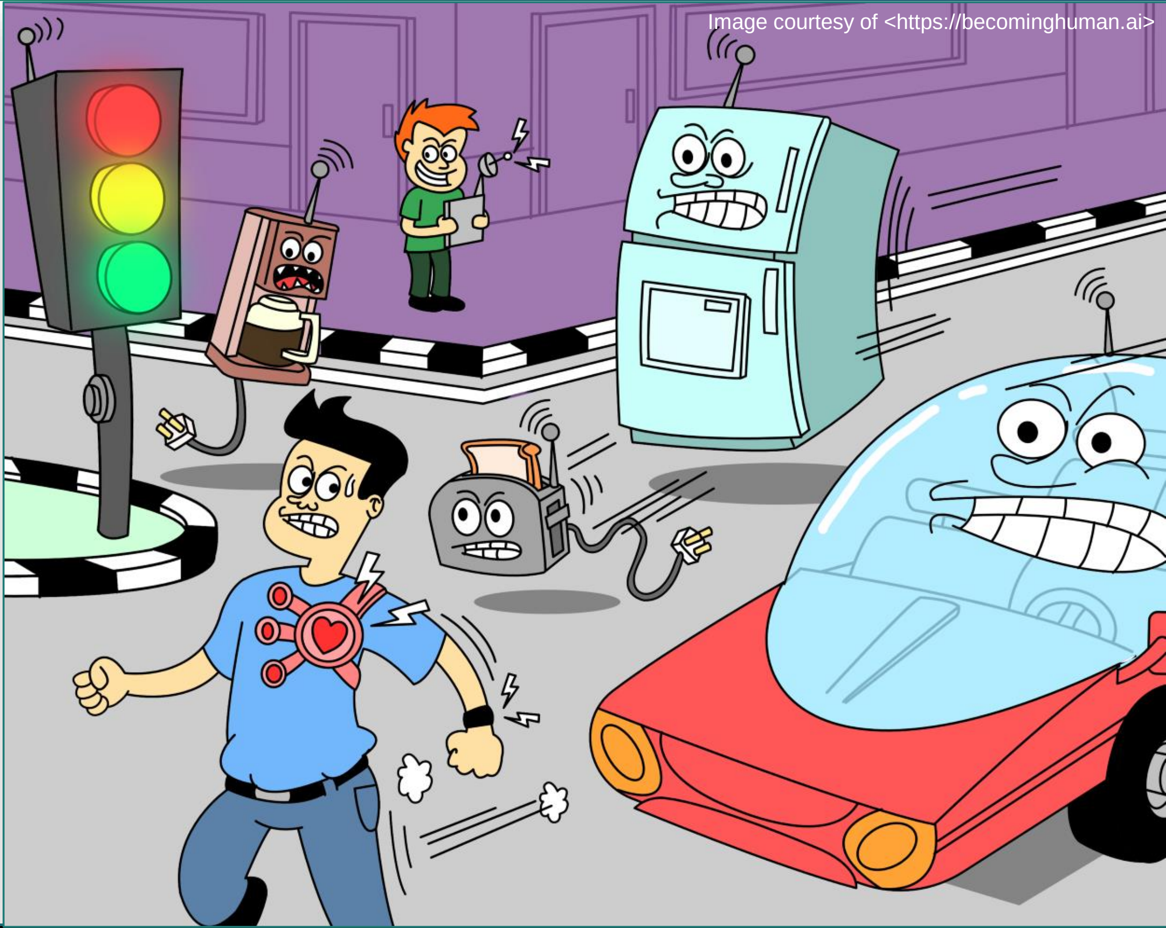


Image courtesy of <https://becominghuman.ai>

IoT - Problem Statement

CONSUMERS

Complexity
Insecurity
Lack of privacy
Turned into products

DEVICE MAKERS/OEMs

Reinventing the wheel
Sub-optimal choices
Becoming Device Dealers
Monetize consumer's data

CONTENT CREATORS

Lack of choice
Lack of standards
Drive o.s. / cloud stickiness
Influence device makers
Monetize consumer's data

Smart Things

- Smart lights, motion sensors
- Smart door locks
- Smart thermostats, radiators, valves
- Smart cameras, doorbells, alarms
- Smart TVs, projectors, speakers
- Smart wearables

- Sensors and actuactors
- MCU, CPU
- FreeRTOS, Zephyr, LiteOS, Linux,...
- From KBs to GBs
- W or w/o display (simple graphics)
- Zigbee, BT,... close range comm
- Java, JS, C, C++ apps

Smart Gateways

- Things to things, things to cloud comms
- Brand specific / isolated comms
- Compute / Storage / OTA / Comms
- CPUs
- Linux, headless
- Gbs
- WiFi, Eth, Zigbee, BT,...close to medium range comm

Smart Mobile

- Phones, Tablets, TVs,...
- Smart Things configuration
- Brand specific smart things apps
- Compute / Storage / HMI / Trainers
- CPUs, Gbs
- Linux, accelerated graphics, rich display
- Medium to long range comms

Putting it all together

Distributed Functionalities	Discovery	Distributed Communications								
	Sensor	Distributed Sensors								
	Actuator	Actuator			Actuator		Actuator	Actuator		
	HMI		HMI			HMI		Dist. HMI		Dist. HMI
	Distributed compute and storage						Distributed Compute and Storage			
	Edge AI	Autonomous Agents					Autonomous Agents and Orchestrators			
Applications	Application Framework	Phone, Maps, Location, ...								
	Applications									
Device Functionalities	Kernel	Zephyr/LiteOS					Linux			
	CPU type	MCU					CPU			
	Number of CPUs	1			2		2	4		8
	Display	Headless	Display				Headless	Display		
	GPU Acceleration	Simple Graphics						Accelerated Graphics		
	Application Runtime Engine	Javascript / C-C++							Javascript /C-C++ /Java	
	Application Framework	GN, Jar, ...								
Device Performance	Communication Range - meters	10			100		1000			
	Energy Consumption	uWatts			mWatts		Watts			
	Memory footprint	kB		MB				GB		
	Processor speed - MIPS	100		500		1000		50000		100000
Device Type		Things					Gateway		Mobile	
	Devices	Speakers, Earbud, Light Bulbs, Doorlocks, Appliances, Watches, Thermostats,...					Transparent GWs		Phones, Tables, In-car	

Mission Statement

- Open Source
- Open Governance
- Industry Driven
- Interoperable (cross-brand)
- User Centric (experience, privacy, security)
- Distributed edge o.s.
  (the cloud is “just” a citizen, not the king)

Edge Devices Collaboration

Based on the W3C semantic web and IEEE distributed agents works

- Distributed agency:
  - Agents have different physical characteristics
  - Agents can play a different role in different scenarios
  - Agents broadcast their characteristics
  - Agents elect one agency coordinator
  - Coordinators trigger coordinated execution, recruit available agents, distribute tasks

Edge Devices Collaboration

- Ontologies are used to describe:
  - Devices characteristics
  - Problems, tasks, routines
- Agency coordinator:
  - Matches agents ontologies with problems
  - Select agents, distribute tasks, executes
  - Agency training is supervised or unsupervised

Edge Devices Collaboration

- African Lion Ontology:
- Classes: African Lion --> Lion --> Animal
- Class: Book About Animals
- Instances: actual book about african lions and its properties

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns="http://protege.stanford.edu/books2#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:p1="http://isbn.nu/089686328"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://protege.stanford.edu/books2">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="AfricanLion">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Lion"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Lion">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Animal"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="BookAboutAnimals">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#Animal"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://purl.org/dc/elements/1.1/subject"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:about="http://purl.org/dc/elements/1.1/subject">
    <rdfs:domain rdf:resource="#BookAboutAnimals"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="bookTitle">
    <rdfs:domain rdf:resource="#BookAboutAnimals"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <Lion rdf:ID="LionSubject"/>
  <BookAboutAnimals rdf:ID="AfricanLionBook">
    <rdfs:seeAlso rdf:resource="http://isbn.nu/089686328X"/>
    <bookTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The African Lion</bookTitle>
    <dc:subject>
      <AfricanLion rdf:ID="AfricanLionSubject"/>
    </dc:subject>
  </BookAboutAnimals>
  <BookAboutAnimals rdf:ID="LionsLifeInThePrideBook">
    <bookTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Lions: Life in the Pride</bookTitle>
    <dc:subject rdf:resource="#LionSubject"/>
    <rdfs:seeAlso rdf:resource="http://isbn.nu/0736809643"/>
  </BookAboutAnimals>
</rdf:RDF>
```

Edge Devices Collaboration

Autonomous agents

Things: sensors, actuators, compute

Gateways: compute, storage, communication, coordination

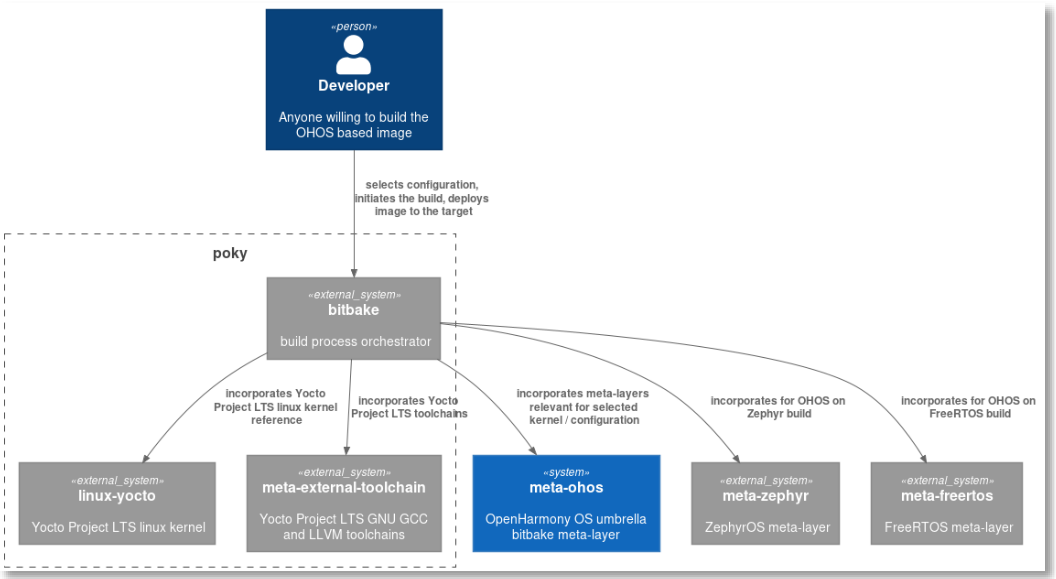
Mobile: compute, storage, HMI, agency supervised training, coordination (not ideal, back-up)

Putting it all together

<table>
<tr><td rowspan="6">Distributed Functionalities</td><td>Discovery</td><td colspan="9">Distributed Communications</td></tr>
<tr><td>Sensor</td><td colspan="9">Distributed Sensors</td></tr>
<tr><td>Actuator</td><td colspan="2">Actuator</td><td></td><td>Actuator</td><td></td><td>Actuator</td><td>Actuator</td><td colspan="2"></td></tr>
<tr><td>HMI</td><td></td><td>HMI</td><td colspan="2"></td><td>HMI</td><td></td><td>Dist. HMI</td><td></td><td>Dist. HMI</td></tr>
<tr><td>Distributed compute and storage</td><td colspan="5"></td><td colspan="4">Distributed Compute and Storage</td></tr>
<tr><td>Edge AI</td><td colspan="5">Autonomous Agents</td><td colspan="4">Autonomous Agents and Orchestrators</td></tr>
<tr><td rowspan="2">Applications</td><td>Application Framework</td><td colspan="9">Phone, Maps, Location, ...</td></tr>
<tr><td>Applications</td><td colspan="9"></td></tr>
<tr><td rowspan="7">Device Functionalities</td><td>Kernel</td><td colspan="5">Zephyr/LiteOS</td><td colspan="4">Linux</td></tr>
<tr><td>CPU type</td><td colspan="5">MCU</td><td colspan="4">CPU</td></tr>
<tr><td>Number of CPUs</td><td colspan="3">1</td><td colspan="2">2</td><td>2</td><td colspan="2">4</td><td>8</td></tr>
<tr><td>Display</td><td>Headless</td><td colspan="4">Display</td><td>Headless</td><td colspan="3">Display</td></tr>
<tr><td>GPU Acceleration</td><td colspan="6">Simple Graphics</td><td colspan="3">Accelerated Graphics</td></tr>
<tr><td>Application Runtime Engine</td><td colspan="7">Javascript / C-C++</td><td colspan="2">Javascript /C-C++ /Java</td></tr>
<tr><td>Application Framework</td><td colspan="9">GN, Jar, ...</td></tr>
<tr><td rowspan="4">Device Performance</td><td>Communication Range - meters</td><td colspan="3">10</td><td colspan="2">100</td><td colspan="4">1000</td></tr>
<tr><td>Energy Consumption</td><td colspan="3">uWatts</td><td colspan="2">mWatts</td><td colspan="4">Watts</td></tr>
<tr><td>Memory footprint</td><td>kB</td><td colspan="5">MB</td><td colspan="3">GB</td></tr>
<tr><td>Processor speed - MIPS</td><td>100</td><td colspan="3">500</td><td colspan="2">1000</td><td colspan="2">50000</td><td>100000</td></tr>
<tr><td rowspan="2">Device Type</td><td></td><td colspan="5">Things</td><td colspan="2">Gateway</td><td colspan="2">Mobile</td></tr>
<tr><td>Devices</td><td colspan="5">Speakers, Earbud, Light Bulbs, Doorlocks, Appliances, Watches, Thermostats,...</td><td colspan="2">Transparent GWs</td><td colspan="2">Phones, Tables, In-car</td></tr>
</table>

Build system– Yocto Project to rule them all

- Some relevant layers
- meta-ohos: root layer
- meta-openembedded
- meta-clang
- meta-zephyr, meta-freertos, meta-liteos
- meta-riscv
- meta-ohos: openharmony components
- meta-seco, meta-st, meta-av96, meta-intel, ...



Build flavours and supported HW

- Build flavours = supported kernels

- Currently supported: Zephyr, Linux

  WIP: LiteOS, FreeRTOS

- Supported images / machines:
  - Linux: allscenarios-image-base (headless), allscenarios-image-extra
  - Linux: qemux86-64, qemux86, qemuarm, qemuarm64, seco-intel-b68 (SECO SBC-B68), stm32mp1-av96 (96Boards Avenger96), seco-imx8mm-c61 (SECO SBC-C61), raspberrypi4-64 (Raspberry Pi 4 Mobel B)
  - Zephyr: zephyr samples
  - Zephyr: qemu-x86, qemu-cortex-m3, 96b-nitrogen (96Boards Nitrogen), 96b-avenger96 (96Boards Avenger96), arduino-nano-33-ble (Arduino Nano 33 BLE and Arduino Nano 33 BLE Sense), nrf52840dk-nrf52840 (Nordic Semiconductor nRF 52840 Development Kit)

Blueprints

- Minimum viable, 80% production-ready reference solutions
- Pre-integrated, tested and maintained (LTS)
- Both makers and production silicon
- Enable and test cooperative use-cases
- Blueprints are NOT: full featured, optimized for cost, size, pretty looking
- Expect to see boards, wires, etc.
- Avalilable blueprints: Doorlock, Transparent Gateway, Touch Panel
- WIP blueprints: Vending machine, mobile phone, smart speaker with vocal assistant, robotic companion,...

» Build System Guide » All Scenarios OS Blueprints » DoorLock Blueprint

DoorLock Blueprint

Contents

Security

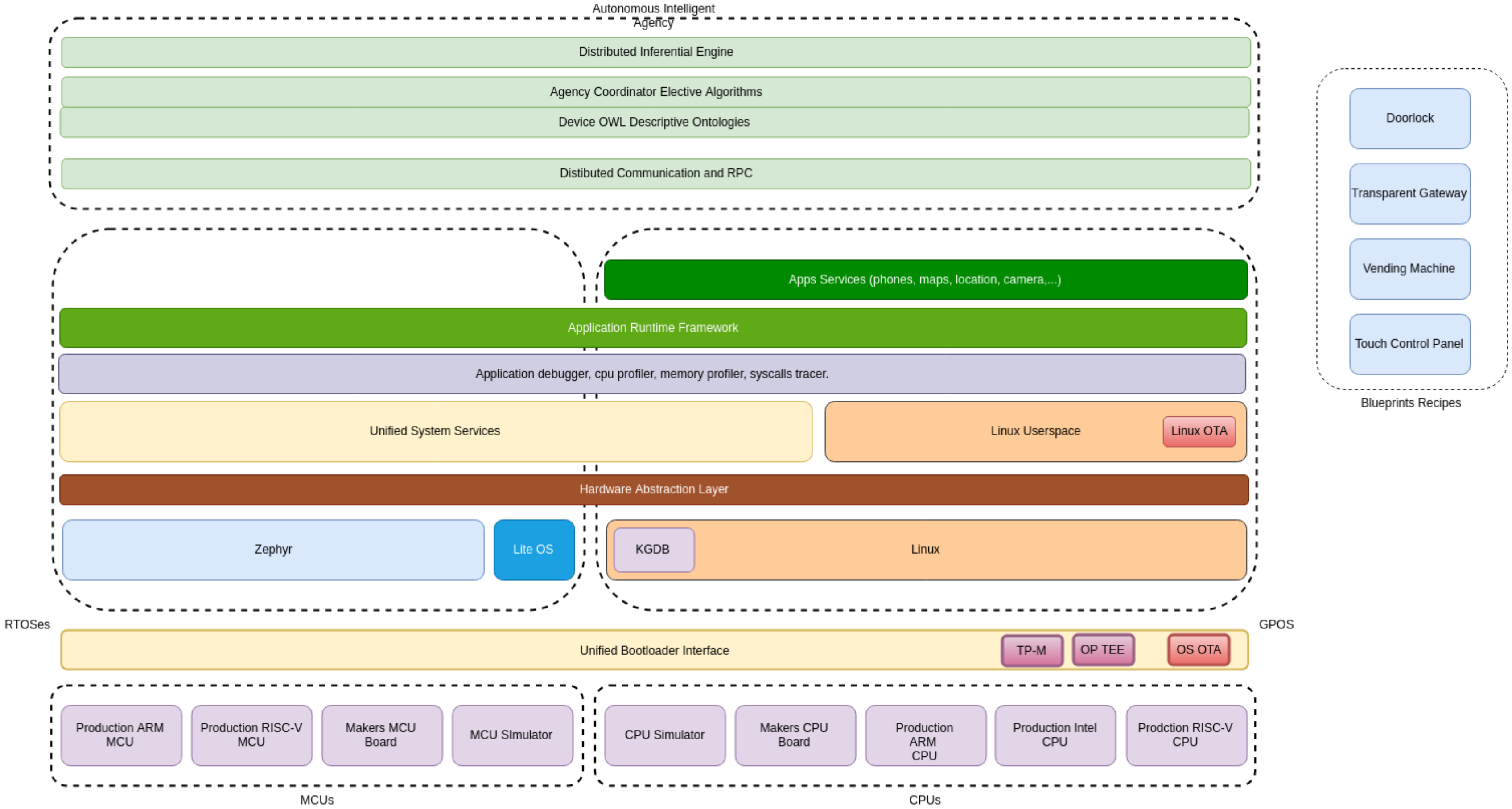
Linux kernel hardening options:

- Memory allocator
- Disabling apparently useful but obsolete features such as COMPAT_BRK, PROC_KCORE, BINFMT_MISC
- SECURITY_DMESG_RESTRICT
- Compiler level hardening via FORTIFY_SOURCE
- Disable physicial memory access and detect unsafe memory permission
- Hardened usercopy from userspace
- Kernel structures data validation
- Under consideration: IOMMU, Panic on Oops

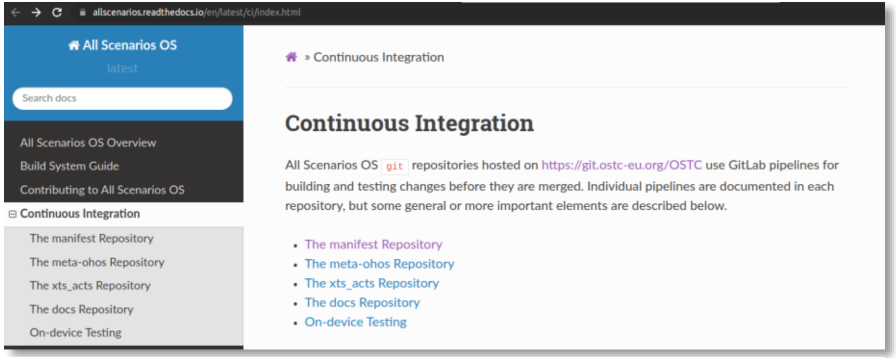
Security related layes / components:

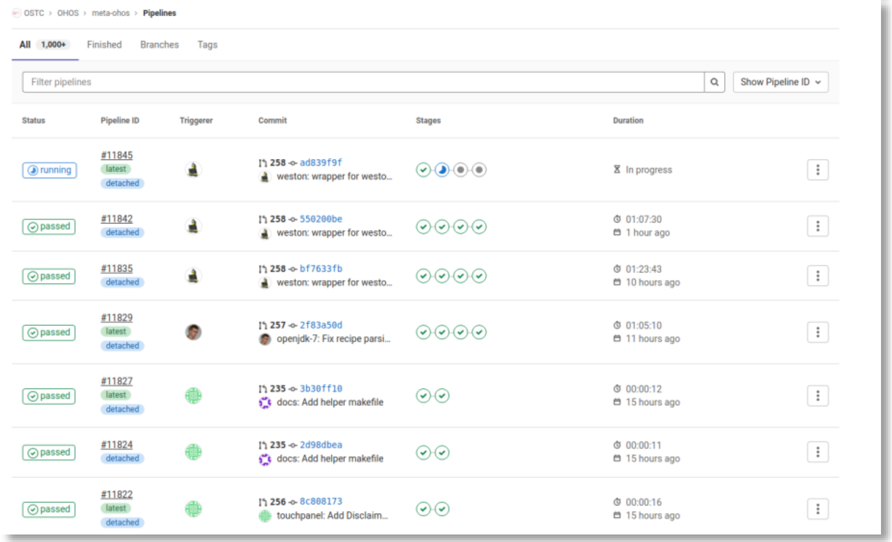
- meta-security
- meta-security-compliance
- meta-security-isafw
- meta-tpm

High level architecture



Continuous Integration



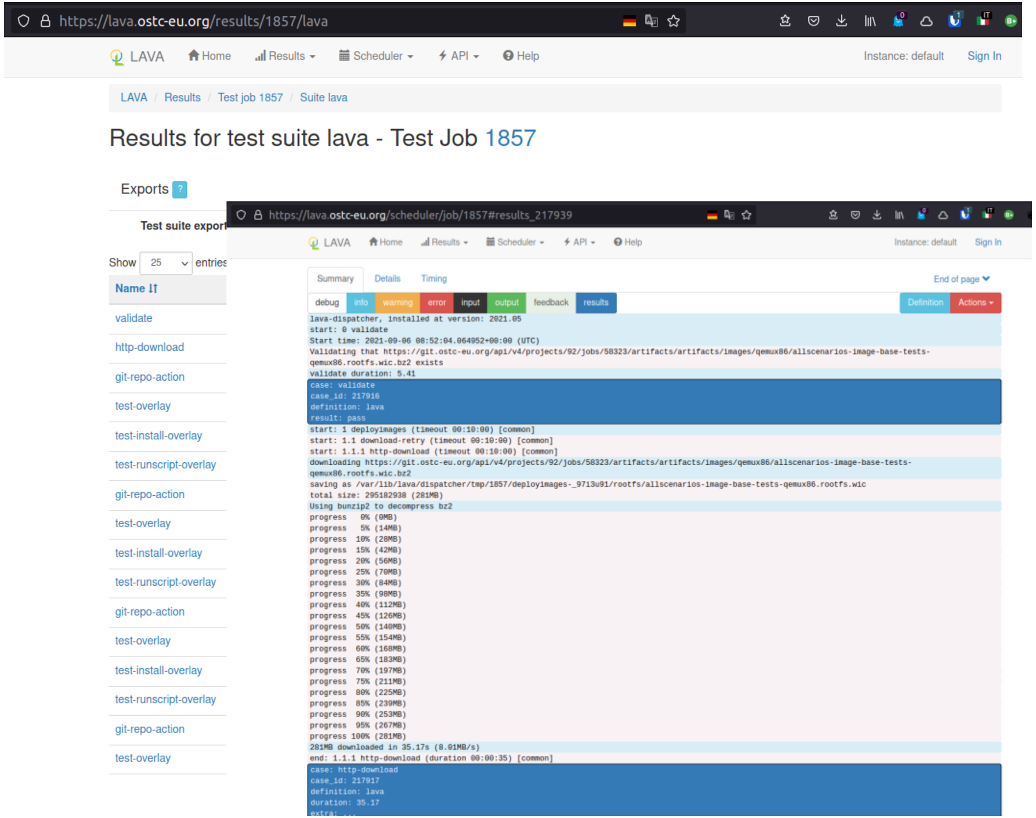


- gitlab runners for builds, with git-repo cache, bitbake sstate and download cache
- strategic placement of jobs across repositories to ease maintenance
- lava for smoke testing on hardware and in virtual environments
- Scancode, Fossology, REUSE, Debian matcher for license compliance and SPDX SBOM
- extra care for fork based workflow in multi-repo world and bitbake recpie (revision pinning) world

Continuous Integration

- Shared jobs for images (machine / flavours) officially supported:
- Currently 14
- Linux-*, zephyr-*, freertos-armv5, blueprints-*
- Hidden jobs as foundation building blocks that shared jobs leverage:
- workspace --> assembles all repos via git repo
- bitbake-workspace --> initialized bitbake build
- build-linux, build-zephyr, build-freertos, build-liteos
- build-recipe, build-image
- build-docs
- lava-test, lava-report
- ip-scan

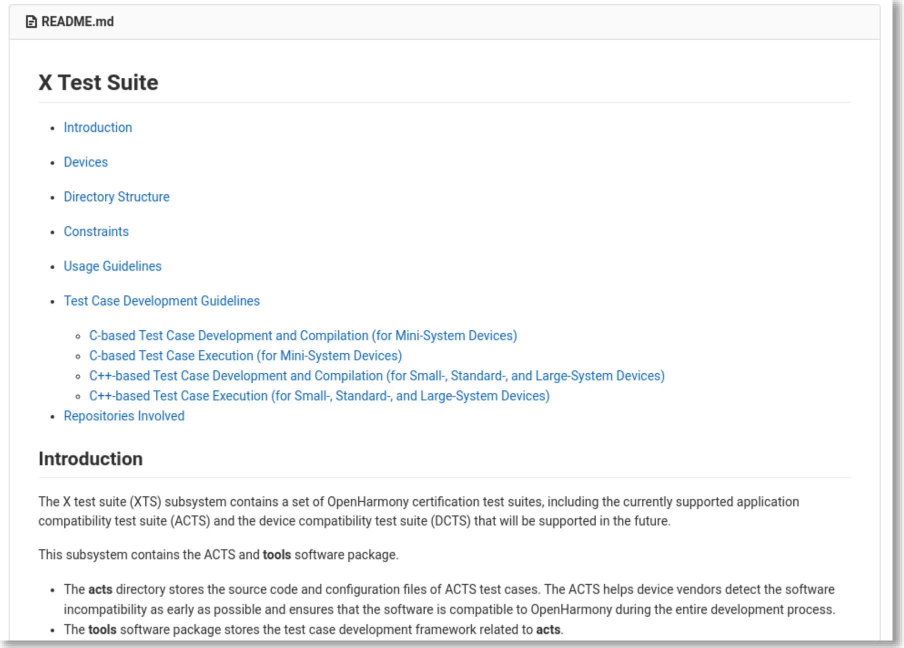
Device Testing



- Decentralized, distributed device testing
- Each member, contributor,... can add physical devices at different locations
- Device added under testing can be shared via public cloud infrastructure
- Each site can add one to hundreds of devices
- Sites broadcast their availability to a central repository / directory
- LAVA (Linaro Automation and Validation Architecture) is used:
- lava-test calls LAVA and create a testing job
- lava-report iterates through the active jobs collects results and aggregates them in a report

OpenHarmony application compatibility

- Openharmony is an Openatom foundation project
- Unified ecosystem is achieved by building “Openharmony compatible” o.s. images
- Compatibility is defined in a Compatibility Specification and automated via an Application Compatibility Test Suite (ACTS)
- xts_acts jobs / testing (https://git.ostc-eu.org/OSTC/OHOS/components/staging/xts_acts)
- Openharmony components needed to achieve compatibility packaged in meta-openharmony-x.y.z
- GN class for bitbake in order to build native GN openharmony files


README.md

X Test Suite

- Introduction
- Devices
- Directory Structure
- Constraints
- Usage Guidelines
- Test Case Development Guidelines
  - C-based Test Case Development and Compilation (for Mini-System Devices)
  - C-based Test Case Execution (for Mini-System Devices)
  - C++-based Test Case Development and Compilation (for Small-, Standard-, and Large-System Devices)
  - C++-based Test Case Execution (for Small-, Standard-, and Large-System Devices)
- Repositories Involved

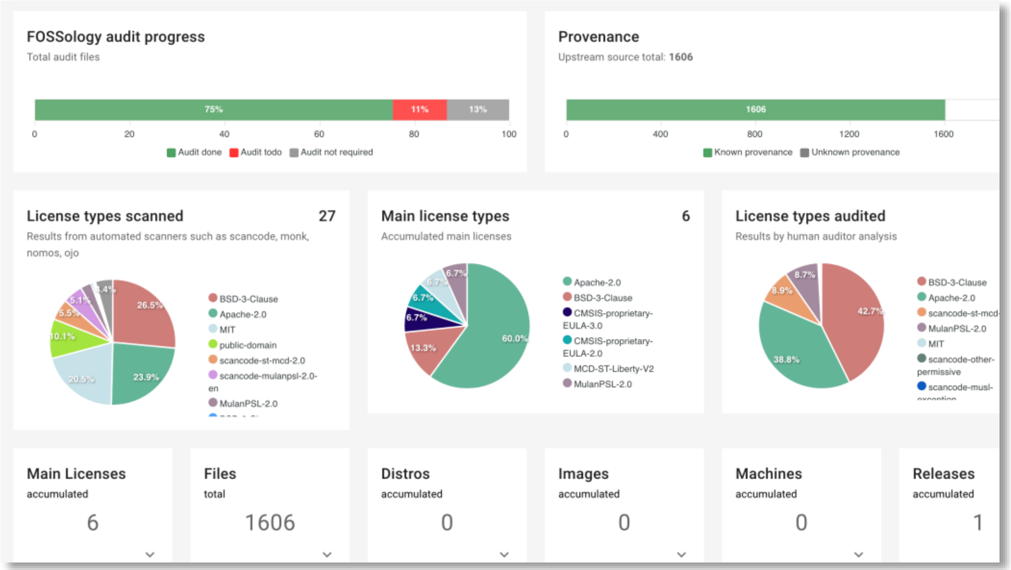
Introduction

The X test suite (XTS) subsystem contains a set of OpenHarmony certification test suites, including the currently supported application compatibility test suite (ACTS) and the device compatibility test suite (DCTS) that will be supported in the future.

This subsystem contains the ACTS and **tools** software package.

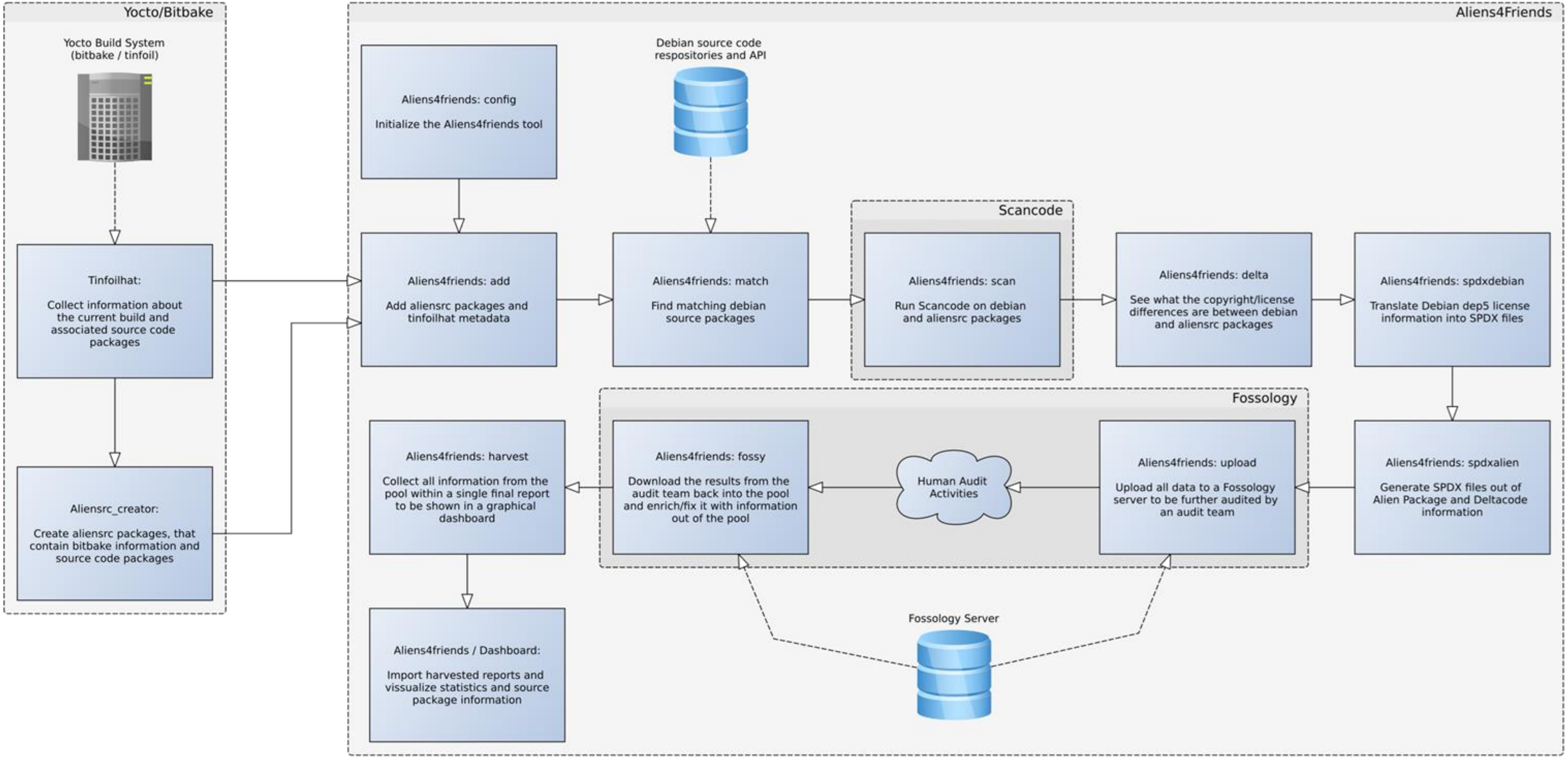
- The **acts** directory stores the source code and configuration files of ACTS test cases. The ACTS helps device vendors detect the software incompatibility as early as possible and ensures that the software is compatible to OpenHarmony during the entire development process.
- The **tools** software package stores the test case development framework related to **acts**.

IP Compliance

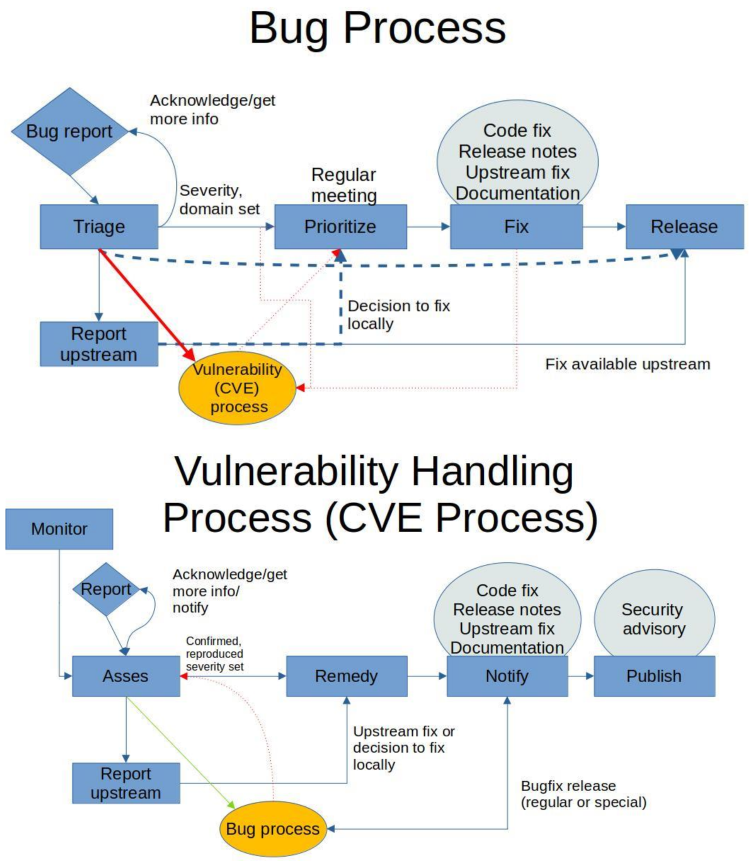


- Openchain Specification 2.0 conformant
- Published at https://allscenarios.readthedocs.io/en/latest/ip-policy/ip-policy_implementation-guidelines/index.html
- Training, R&R, fundings, activity, IP auditing embedded into R&D
- Continuous IP compliance via integrations of IP compliance toolchain in the dev process via gitlab jobs
- Low Resolution SBOM: Merge --> Scan --> SPDX / BOM --> Dashboard
- High Resolution: Dashboard --> IP Auditor --> Fossology
- Releases SBOMs for alpha, beta and official yearly release

IP Compliance

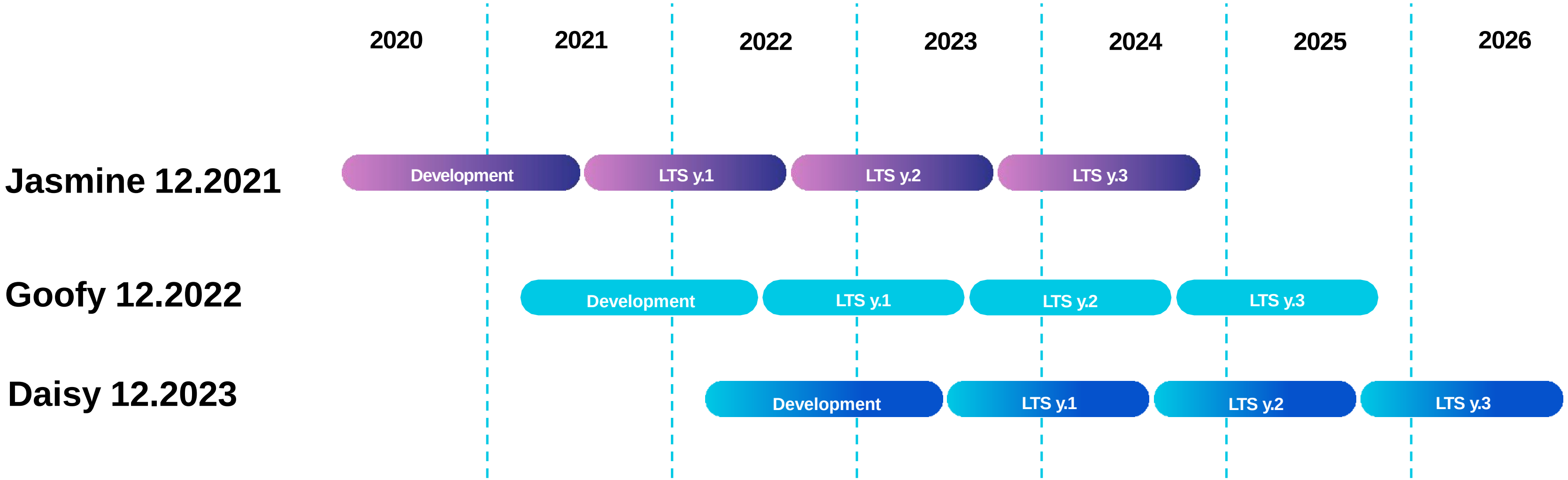


Maintenance and Release life cycle



- One yearly major release (12 months dev cycle)
- 3 years LTS with decrease level of service (based on bug / CVE impact score)
- Dedicated LTS team: maintenance engineers, security response engineers
- Leverage upstream LTS for major components such as linux kernel, toolchain, ...

Maintenance and Release life cycle



	BUGS	CVEs
YEAR 1	All	All
YEAR 2	P1/P2	CVSS 7-10
YEAR 3	P1	CVSS 9-10

Standards and Compliance

OpenHarmony

yocto PROJECT COMPATIBLE

OPENCHAIN

SPDX

REUSE SOFTWARE

arm SystemReady IR

Project Phases

01 **Bootstrap – 12 to 18 months**

- New brand
- Hosting foundation
- Growing active members, design wins and community

02 **De-fragmentation (broad sustainable industry participation) – 12 to 36 months**

03 **Adoption (design wins, shipped devices, apps and content) – 18+ months**

04 **Edge Devices Collaboration – 18+ months**

Thank you

OSTC
Open Source Technology Center

HUAWEI

Linaro

SECO

ARRAY

synesthesia

NOI TECHPARK SÜDTIROL / ALTO ADIGE

ECLIPSE FOUNDATION

TALK TO US

ECLIPSE 2021 CON

Embedded Linux Conference