LINUX FOUNDATION
COLLABORATIVE PROJECTS

CIVIL
INFRASTRUCTURE
PLATFORM

# Time is ready for the Civil Infrastructure Platform

**Yoshitake Kobayashi, Toshiba**
**Urs Gleim, Siemens AG**

Embedded Linux Conference Europe, Berlin, October 13, 2016

# Definition

Civil Infrastructure Systems are technical systems responsible for supervision, control, and management of infrastructure supporting human activities, including, for example,

- Electric power generation
- Energy distribution
- Oil and gas
- Water and wastewater
- Healthcare
- Communications
- Transportation
- Collections of buildings that make up urban & rural communities.

These networks deliver essential services, provide shelter, and support social interactions and economic development. They are society's lifelines.[1)]

CIVIL INFRASTRUCTURE PLATFORM

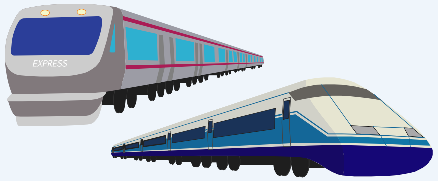
[1)] adapted from https://www.ce.udel.edu/current/graduate_program/civil.html

# Linux is widely used in …

## Transport

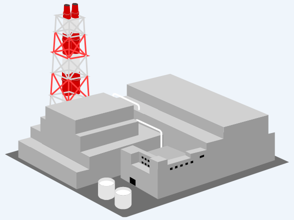
Rail automation

Automatic ticket gates

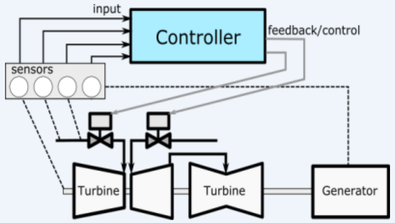


Vehicle control

## Energy



Power Generation



Turbine Control

## Industry

Industry automation

Industrial communication

CNC control

## Others

Healthcare

Building automation

Broadcasting

# Civil infrastructure systems

## Core characteristics

### Industrial grade

- Reliability
- Functional Safety
- Security
- Real-time capabilities

### Sustainability

- Product life-cycles of 10 – 60 years

### Conservative update strategy

- Firmware updates only if industrial grade is jeopardized
- Minimize risk of regression
- Keeping regression test and certification efforts low

## Business needs

### Maintenance costs

- Low maintenance costs for commonly uses software components
- Low commissioning and update costs

### Development costs

- Don‘t re-invent the wheel

### Development time

- Shorter development times for more complex systems

# The evolution of civil infrastructure systems

## Technology changes

### Proprietary nature

- Systems are built from the ground up for each product
- little re-use of existing software building blocks
- Closed systems

### Commoditization

- Increased utilization of commodity (open source) components, e.g.,  operating system, virtualization
- Extensibility, e.g., for analytics

### Stand-alone systems

- Limited vulnerability
- Updates can only applied with physical access to the systems
- High commissioning efforts

### Connected systems

- Interoperability due to advances in machine-to-machine connectivity
- Standardization of communication
- Plug and play based system designs

# Things to be done

- Join forces for commodity components
  - Ensure industrial grade for the operating system platform focusing on reliability, security, real-time capability and functional safety
  - Increase upstream work in order to increase quality and to avoid maintenance of patches
- Share maintenance costs
  - Long-term availability and long-term support are crucial
- Innovate for future technology
  - Support industrial IoT architectures and state-of-the art machine-to-machine connectivity

CIVIL
INFRASTRUCTURE
PLATFORM

**Civil infrastructure systems require a super long-term maintained industrial-grade embedded Linux platform for a smart digital future**

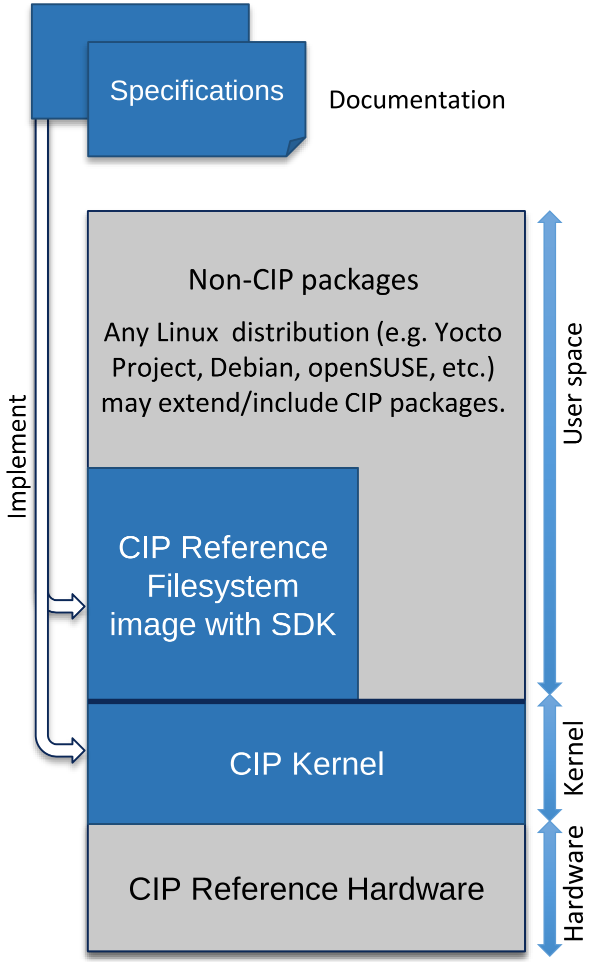
CIVIL INFRASTRUCTURE PLATFORM

# Civil Infrastructure Platform aims to provide industrial grade software

Establish an **open source "base layer" of industrial grade software** to enable the use and implementation in infrastructure projects of software building blocks that meet the **safety, reliability, security and maintainability requirements**.

- Fill the gap between capabilities of the existing OSS and industrial requirements.
- Provide reference implementation
- Trigger development of an emerging ecosystem including tools and domain specific extensions

➔**Initial focus on establishing long term maintenance infrastructure for selected Open Source components, funded by participating membership fees**



Specifications
Documentation
Implement
Non-CIP packages
Any Linux distribution (e.g. Yocto Project, Debian, openSUSE, etc.) may extend/include CIP packages.
CIP Reference Filesystem image with SDK
CIP Kernel
CIP Reference Hardware
User space
Kernel
Hardware

CIVIL INFRASTRUCTURE PLATFORM

# Railway Example

3 – 5 years development time

2 – 4 years customer specific extensions

1 year initial safety certifications / authorization

3 – 6 months safety certifications / authorization for follow-up releases (depending on amount of changes)

25 – 50 years lifetime

Image: http://www.deutschebahn.com/contentblob/10862328/20160301+Stw+M%C3%BClheim+Innenansicht+1+(1)/data.jpg

# Power Plant Control Example

3 – 5 years development time

0.5 – 4 years customer specific extensions

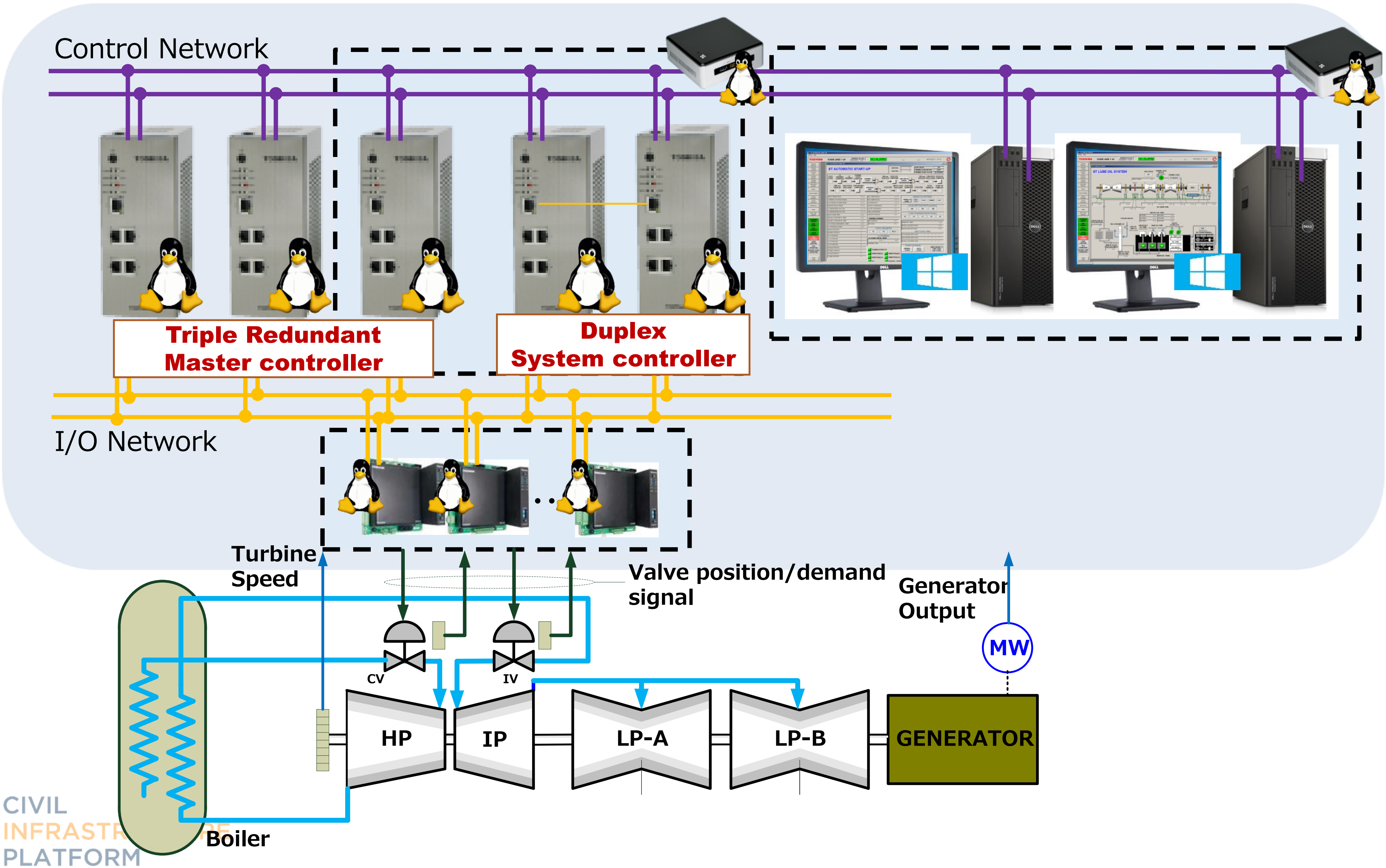
6 – 8 years supply time

15+ years hardware maintenance after latest shipment

20 – 60 years product lifetime

Image: http://zdnet1.cbsistatic.com/hub/i/r/2016/02/29/10863f77-89b2-40c0-9d8c-dbaa5feb65be/resize/770xauto/490141cef9bddc0db66b492698b53a50/powerplant.jpg

# Power plant runs on the Linux (please visit our booth)



# Why maintaining old kernels?

1. Fear of regressions in newer kernels
   (performance and system stability)

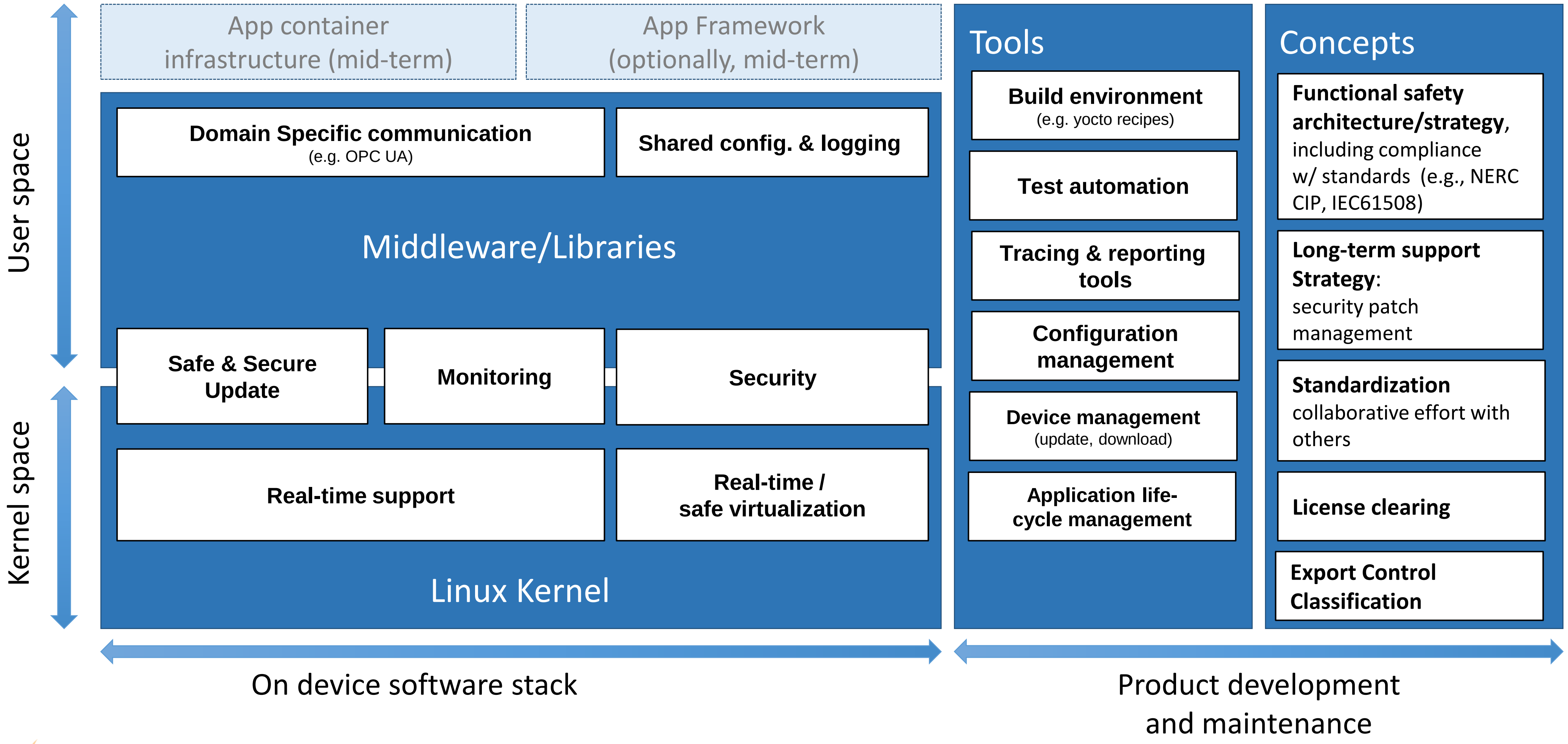
2. Reducing re-certifications costs and time by minimizing changes

3. Reduced number of kernel versions to be provided by SoC vendors
   (like LSK or LTSI)

4. Serving as a common base for vendor-specific kernel forks
   and out-of-tree code
   (yes, we prefer upstreaming…)

see also http://lwn.net/Articles/700530/

# Scope of activities



User space

Kernel space

App container infrastructure (mid-term)

App Framework (optionally, mid-term)

**Domain Specific communication** (e.g. OPC UA)

**Shared config. & logging**

Middleware/Libraries

**Safe & Secure Update**

**Monitoring**

**Security**

**Real-time support**

**Real-time / safe virtualization**

Linux Kernel

Tools

**Build environment** (e.g. yocto recipes)

**Test automation**

**Tracing & reporting tools**

**Configuration management**

**Device management** (update, download)

**Application life-cycle management**

Concepts

**Functional safety architecture/strategy,** including compliance w/ standards (e.g., NERC CIP, IEC61508)

**Long-term support Strategy**: security patch management

**Standardization** collaborative effort with others

**License clearing**

**Export Control Classification**

On device software stack

Product development and maintenance

# Target Systems

Target systems

| | Networked Node 1 | Embedded Control Unit 2 | Embedded Computer 3 | Embedded Server 4 |
|---|---|---|---|---|
| ARM offerings[1] | M0/M0+/M3/M4 | M4/7, A9, R4/5/7 | ARM A9/A35, R7 | ARM A53/A72 |
| Intel offerings[1] | Quark MCU | Quark SoC | Atom | Core, Xeon |
| Architecture, clock | 8/16/32-bit, < 100 MHz | 32-bit, <1 GHz | 32/64-bit, <2 GHz | 64-bit, >2 GHz |
| non-volatile storage | n MiB flash | n GiB flash | n GiB flash | n TiB flash/HDD |
| RAM | < 1 MiB | < 1 GiB | < 4 GiB | > 4 GiB |
| HW ref. platform | Arduino class board | Raspberry Pi class board | SoC-FPGA, e.g.Zync | industrial PC |
| application examples | Sensor, field device | control systems | special purpose & server based controllers | |
| | PLC | | gateways | multi-purpose controllers |

Out of scope:

- Enterprise IT and cloud system platforms.

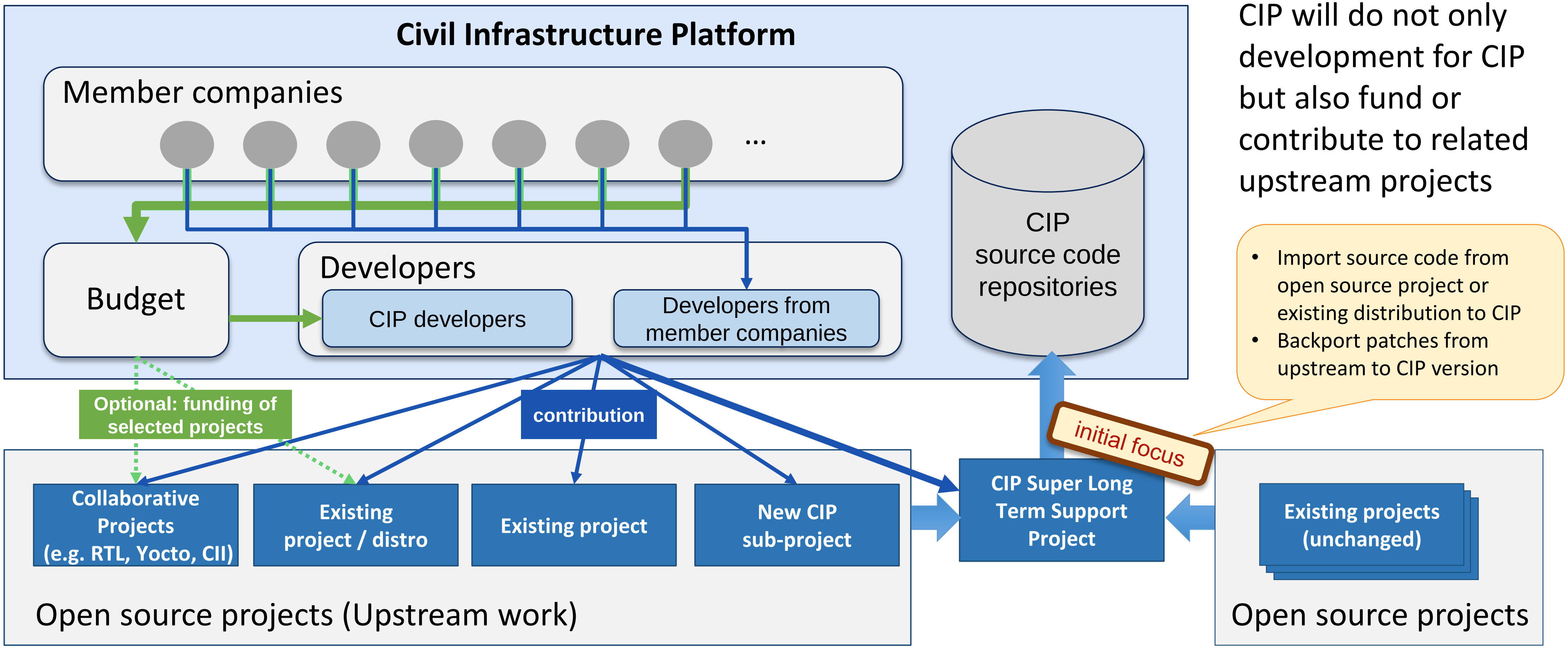
Reference hardware for common software platform:

- Start from working the common HW platform (PC)
- Later extend it to small/low power devices

1) Typical configurations Q1/2016

1 ... 4 Device class no.

# Relationship between CIP and other projects


Civil Infrastructure Platform
Member companies
…
Budget
Developers
CIP developers
Developers from member companies
CIP source code repositories
Optional: funding of selected projects
contribution
initial focus
Collaborative Projects (e.g. RTL, Yocto, CII)
Existing project / distro
Existing project
New CIP sub-project
CIP Super Long Term Support Project
Existing projects (unchanged)
Open source projects (Upstream work)
Open source projects

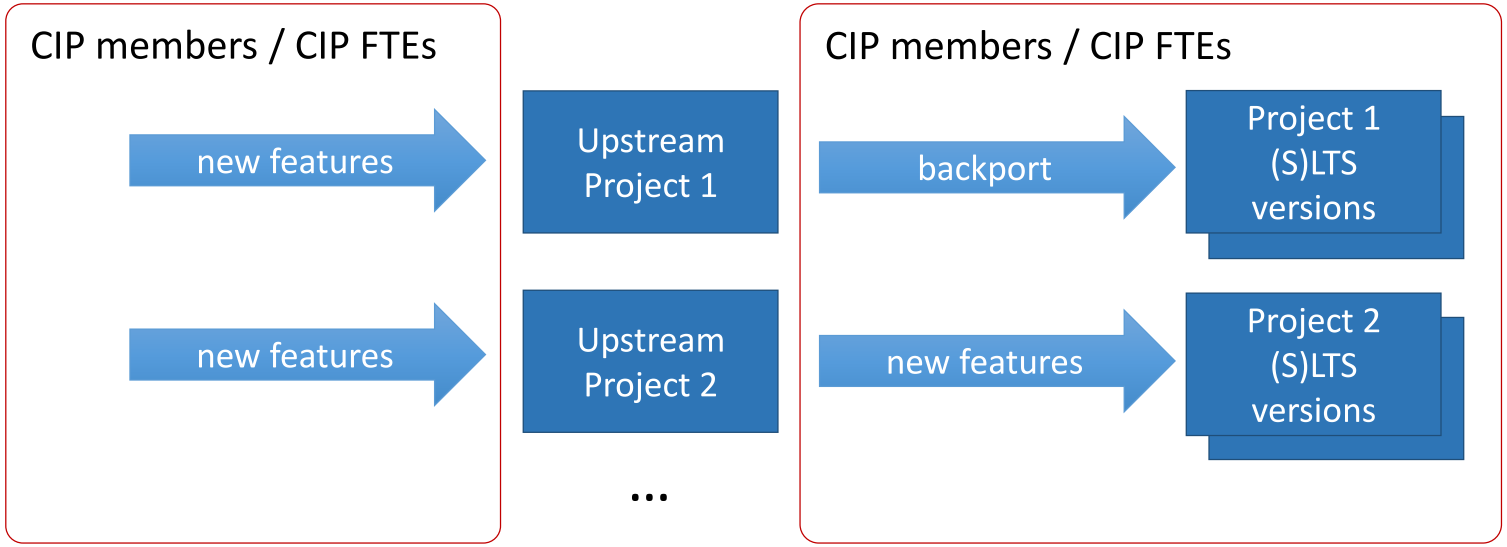
CIP will do not only development for CIP but also fund or contribute to related upstream projects

- Import source code from open source project or existing distribution to CIP
- Backport patches from upstream to CIP version

# Upstream first policy for implementation of new features

## All deltas to mainline to be treated as technical debt

- Avoid parallel source trees, directly discuss features in upstream projects
- Upstream first for fixes and features, just like for stable kernels
- Afterwards back-port to super long-term versions driven by CIP


CIP members / CIP FTEs

new features

new features

Upstream Project 1

Upstream Project 2

...

CIP members / CIP FTEs

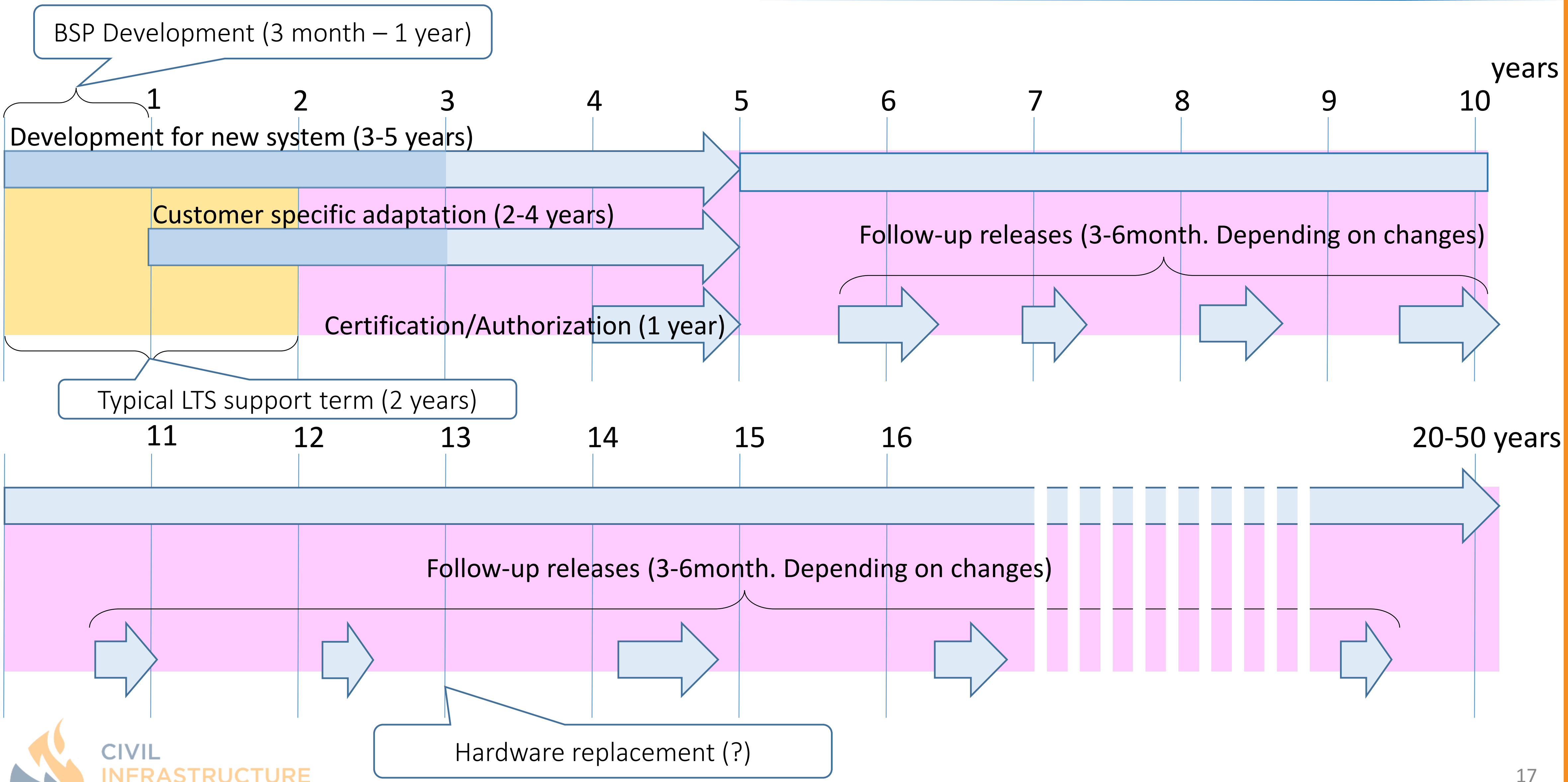
backport

new features

Project 1 (S)LTS versions

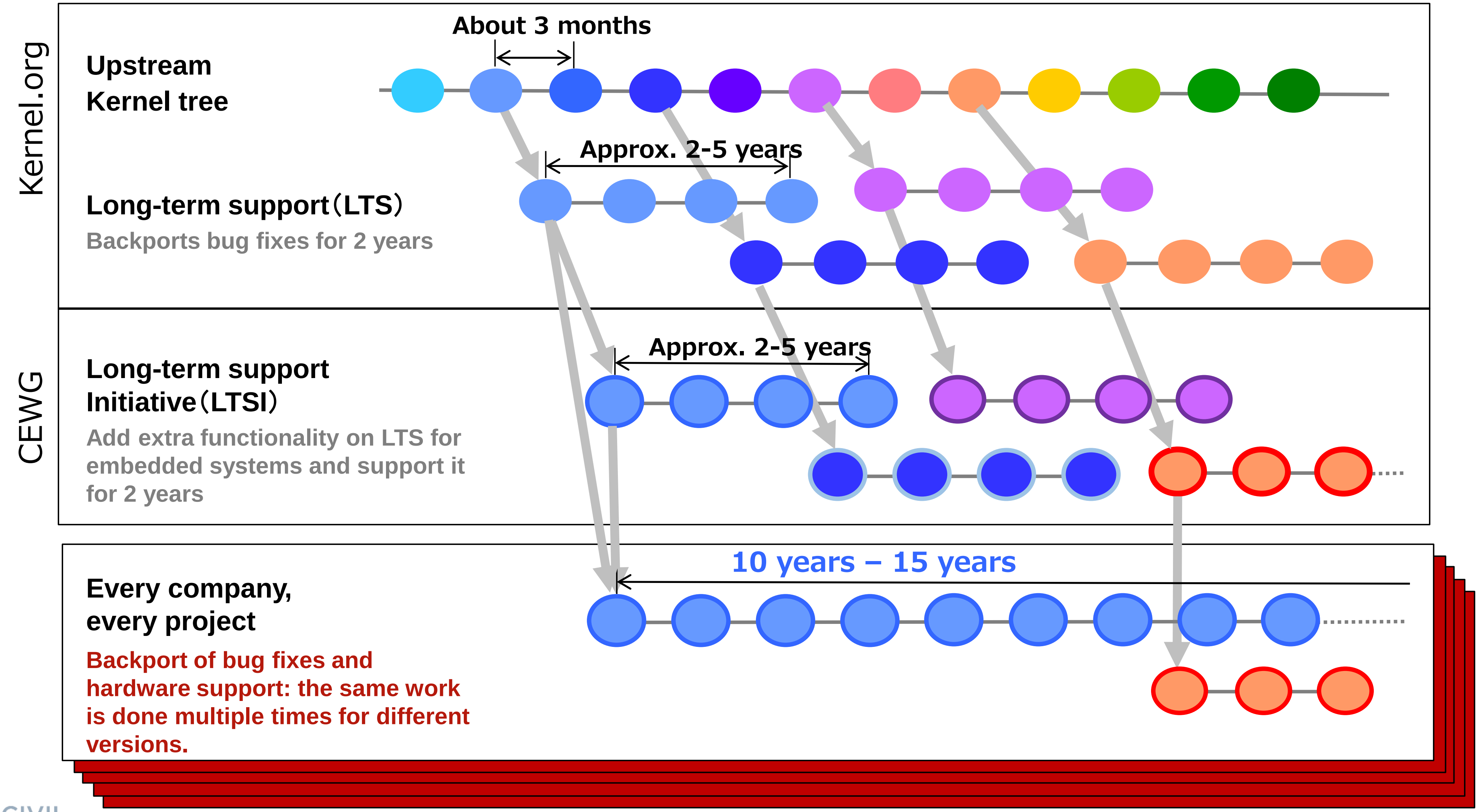
Project 2 (S)LTS versions

# Why super long term support? (Let's look a typical development process)


BSP Development (3 month – 1 year)
years
1
2
3
4
5
6
7
8
9
10
Development for new system (3-5 years)
Customer specific adaptation (2-4 years)
Follow-up releases (3-6month. Depending on changes)
Certification/Authorization (1 year)
Typical LTS support term (2 years)
11
12
13
14
15
16
20-50 years
Follow-up releases (3-6month. Depending on changes)
Hardware replacement (?)

CIVIL INFRASTRUCTURE PLATFORM

# Super Long Term Support - Motivation



Kernel.org

Upstream Kernel tree

About 3 months

Long-term support（LTS）

Backports bug fixes for 2 years

Approx. 2-5 years

CEWG

Long-term support Initiative（LTSI）

Add extra functionality on LTS for embedded systems and support it for 2 years

Approx. 2-5 years

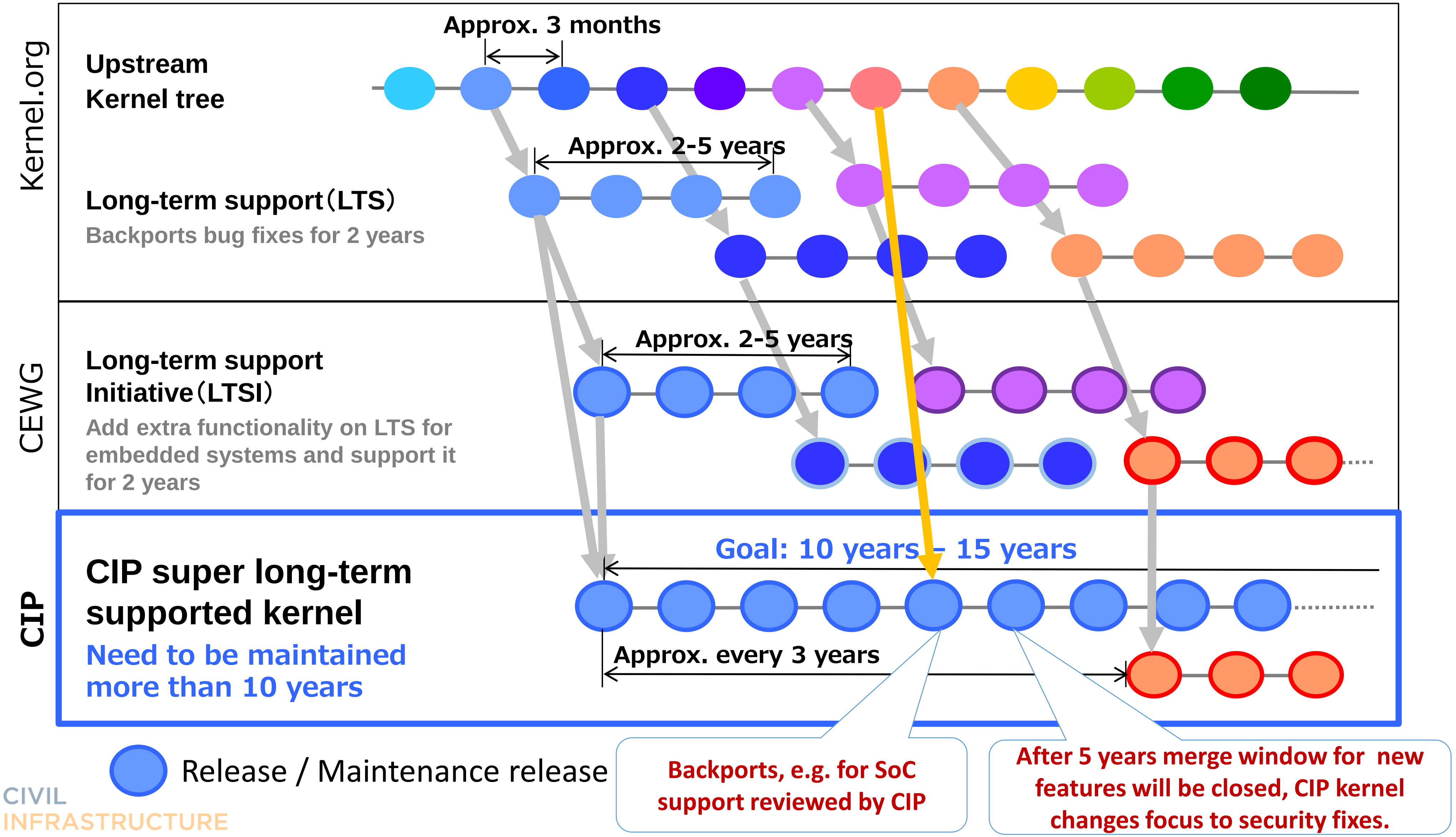
Every company, every project

Backport of bug fixes and hardware support: the same work is done multiple times for different versions.

10 years – 15 years

● Release / Maintenance release

# CIP kernel super long term support (SLTS) overview



Kernel.org

**Upstream Kernel tree**

Approx. 3 months

**Long-term support（LTS）**

Backports bug fixes for 2 years

Approx. 2-5 years

CEWG

**Long-term support Initiative（LTSI）**

Add extra functionality on LTS for embedded systems and support it for 2 years

Approx. 2-5 years

**CIP**

**CIP super long-term supported kernel**

**Need to be maintained more than 10 years**

Goal: 10 years – 15 years

Approx. every 3 years

● Release / Maintenance release

Backports, e.g. for SoC support reviewed by CIP

After 5 years merge window for new features will be closed, CIP kernel changes focus to security fixes.

# Announcement: The first CIP SLTS kernel version

4.4

# Announcement: The first CIP SLTS kernel version

- CIP will maintain the Linux kernel 4.4 for more than 10 years

- Selection Criteria for the first SLTS kernel version
  - LTS version, ideally synchronized with LTSI
  - Broadly used for civil infrastructure systems
    - Currently deployed products
    - **Upcoming products**

- Next SLTS kernel version?
  - Will be announced in 2-3years
  - **Synchronize with LTSI kernel version at this timing**

# Super Long-Term Stable Team

- Ben Hutchings is first super long-term kernel maintainer
  - Well-known Debian contributor and package maintainer
  - Currently LTS maintainer for 3.2 and 3.16
- Ben will be supported by one additional developer
- Work started in September 2016
  - Setup of SLTS development and validation process
  - Prepare and perform first SLTS kernel release
  - Support CIP in extending SLTS model to further core packages

# Plans for CIP SLTS kernel development

- Development Process
  - Development process will be similar to LTSI
    - Accept feature backports from upstream kernel
    - CIP will have merge windows and validation periods for feature backporting
  - Important NOTE: If the backport changes the kernel API, it will not be accepted

- Validation
  - Establishing kernel test infrastructure
  - Enhance on-target testing beyond boot-tests
  - Share the results for open spec boards

# CIP Testing Considerations

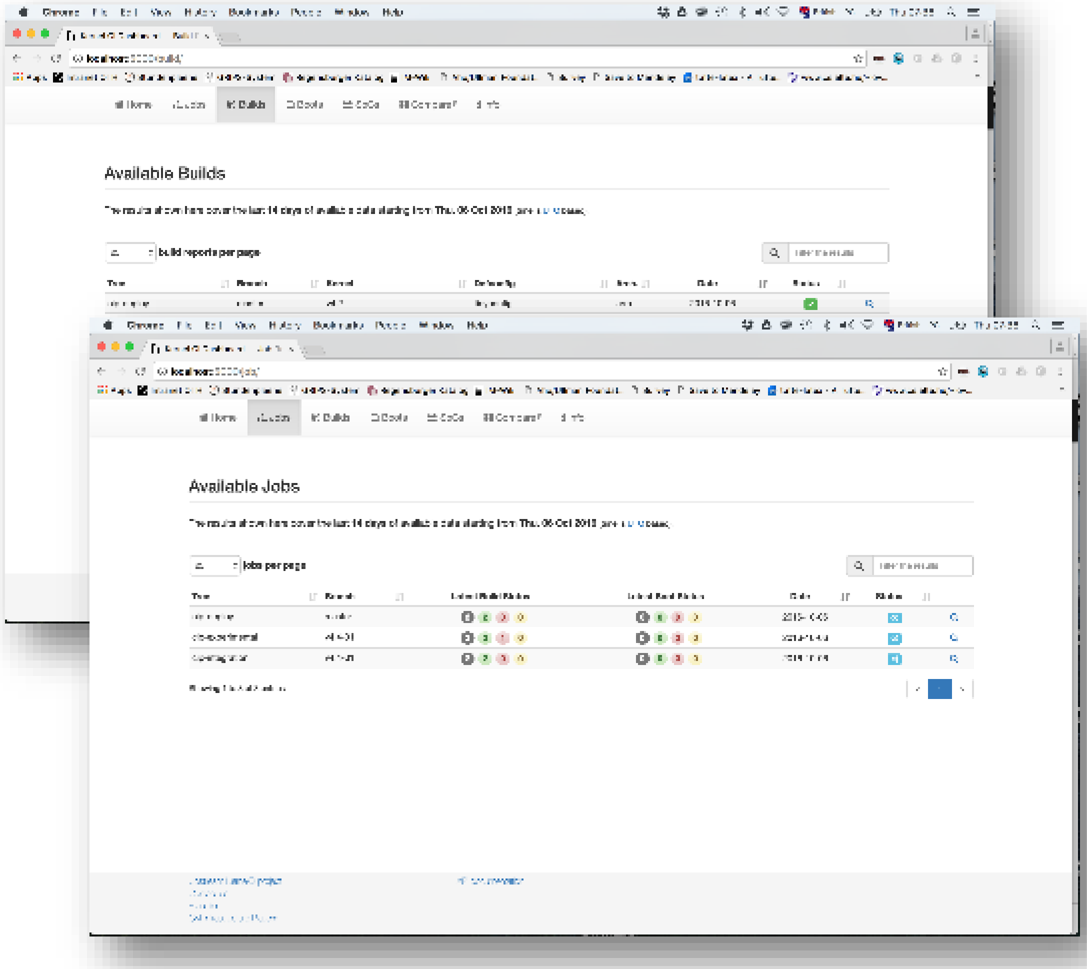
## Testing goals

- Perform testing on real HW (VM: no detail quirks and real-world issues)
- Focus on CIP reference platforms
- Critical Fixes: Build & test within hours on all machines
- No continuous functional testing (for instance, latencies)
- Super-Long-Term result preservation
- Align approach with established community best practices

# CIP Testing Considerations (cont'd)

## Current Status

- Initial CIP-private instance of Kernel CI (vagrant based)
  - Member companies can run local labs
  - HW rack standard (standardized physical and electrical setup) under consideration
- Purely local operation; results via central public web server once fully operational
- Job + Build scheduling: To be defined (likely Fuego and friends)
- Feed results back to Kernel CI?



Available Builds

build reports per page

Available Jobs

jobs per page

**Kernel-CI:** https://kernelci.org/
**Fuego:** http://elinux.org/Fuego

# Selection Criteria for Userspace Packages

- Essential for booting and basic functionality
- Commonly used in civil infrastructure systems
- Security sensitive
- Likely maintainable over 10 years+ period
- **We are open for proposals!**

# Further Candidates for Super Long-term Maintenance

## An Example minimal set of “CIP kernel” and “CIP core” packages for initial scope

### Super Long-term support

Kernel (SLTS)

- Kernel
  - Linux kernel (cooperation with LTSI)
  - PREEMPT_RT patch

Core Packages (SLTS)

- Bootloader
  - U-boot
- Shells / Utilities
  - Busybox
- Base libraries
  - Glibc
- Tool Chain
  - Binutils
  - GCC
- Security
  - Openssl
  - Openssh

### Maintain for Reproducible build

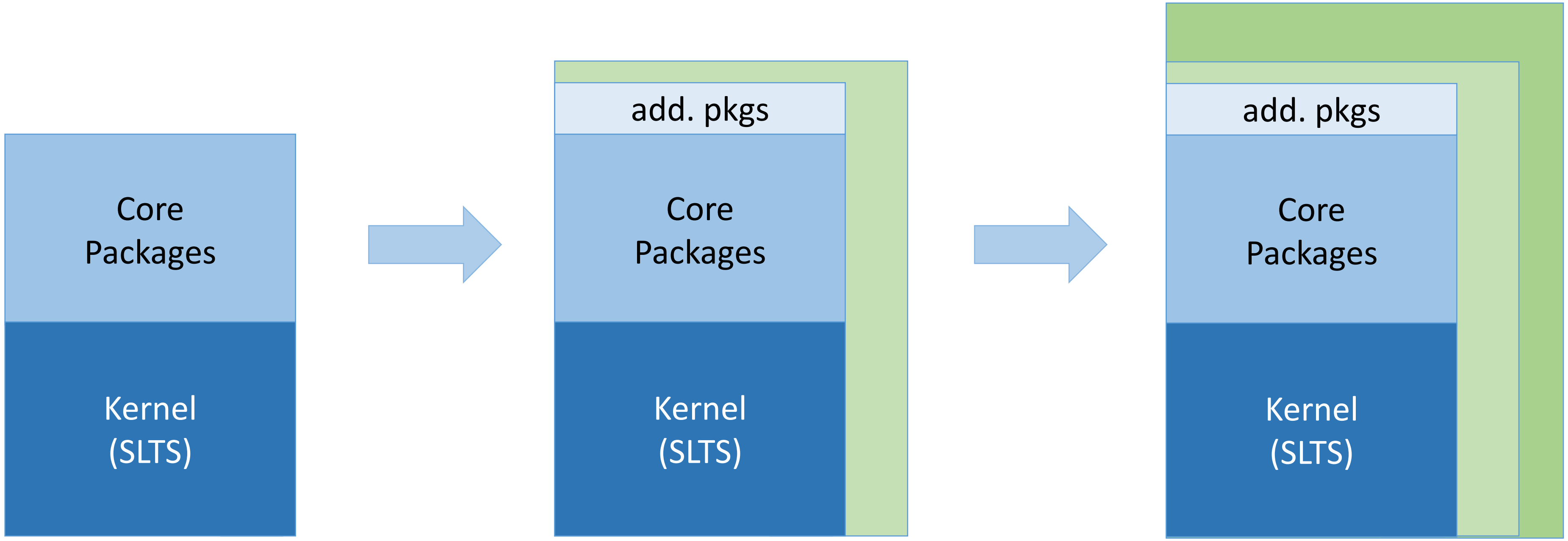
Dev packages

- Flex
- Bison
- autoconf
- automake
- bc
- bison
- Bzip2
- Curl
- Db
- Dbus
- Expat
- Flex
- gawk
- Gdb
- Git
- Glib
- Gmp
- Gzip
- gettext
- Kbd
- Libibverbs
- Libtool
- Libxml2
- Mpclib
- Mpfr4
- Ncurses
- Make
- M4
- pax-utils
- Pciutils
- Perl
- pkg-config
- Popt
- Procps
- Quilt
- Readline
- sysfsutils
- Tar
- Unifdef
- Zlib

*NOTE: The maintenance effort varies considerably for different packages.*

# Development plan

CIP will increase the development effort to create industrial grade common base-layer



Phase 1:

- Define supported kernel subsystems, arch.
- Initial SLTS component selection
- Select SLTS versions
- Set-up maintenance infrastructure (build, test)

Phase 2:

- Patch collection, stabilization, back port of patches for CIP kernel packages
- Support more subsystems
- Additional core packages

Phase 3:

- Domain specific enhancements, e.g. communication protocols, industrial IoT middleware
- Optionally: more subystems
- Optionally:  more core packages

# Currently under discussion

- CIP should collaborate with other similar efforts
  - LSK (Linaro Stable Kernel)
  - Other distributer or SoC vendors
- Selection of features for backporting
  - PREEMPT_RT
    - PREEMPT_RT might be merged into separate branch
  - KSPP (Kernel Self Protection Project)
- Testing infrastructure (KernelCI + Fuego)
- Kernel maintenance policy
- Userland package selection

# Milestones

- 2016:
  - Project launched announcement at Embedded Linux Conference 2016
  - Requirements defined, base use cases defined, technical & non-technical processes established (license clearing, long-term support), maintenance plan
  - Common software stack defined, related core projects agreed (e.g. PREEMT_RT, Xenomai), maintenance infrastructure set up
  - Domain specific extensions defined, tool chain defined, test strategy defined
  - Maintenance to be operational and running

- 2017:
  - Realization phase of selected components

- 2018:
  - Advancement, improvements, new features

# Please join!

**Provide a super long-term maintained industrial-grade embedded Linux platform.**

CIVIL
INFRASTRUCTURE
PLATFORM

## Current members

| | | | |
|---|---|---|---|
| **Platinum Members** | HITACHI<br>Inspire the Next | SIEMENS | TOSHIBA |
| **Silver Members** | Codethink | Plat'Home<br>There, we are. Internet of Things | |

# Why join CIP?

- Participate in **project decisions** through the governing board and/or committees; leverage an ecosystem of like-minded participants to help drive project priorities as a community.
- Provide **technical direction** through a TSC representative enabling fast engagement and input into the technical direction of the project
- Demonstrate support for CIP.
- Priority access to any events, sponsorship and marketing opportunities. Potential events include:
  - Embedded Linux Conference
  - LinuxCon
  - Collaboration summits
  - Other community events
- Visibility on the CIP website and in membership collateral

# Contact Information and Resources

To get the latest information, please contact:

| | |
|---|---|
| Noriaki Fukuyasu | fukuyasu@linuxfoundation.org |
| Urs Gleim | urs.gleim@siemens.com |
| Yoshitake Kobayashi | yoshitake.kobayashi@toshiba.co.jp |
| Hiroshi Mine | hiroshi.mine.vd@hitachi.com |

Other resources

| | |
|---|---|
| CIP Web site | https://www.cip-project.org |
| CIP Mailing list | cip-dev@lists.cip-project.org |
| CIP Wiki | https://wiki.linuxfoundation.org/civilinfrastructureplatform/ |

CIVIL
INFRASTRUCTURE
PLATFORM

# Questions?

# Thank you!