Instrument Cluster Expert Group

IC-EG Japan development team

Topics

- Introduction of Instrument Cluster EG

- What does Instrument Cluster EG aim

- Architecture overview

- Challenge

- Next

Background – Instrument Cluster -

- Instrument Cluster Expert Group launches from March 2019

Instrument Cluster (New EG)

- Create profile for Cluster(HUD)
- Shrink and optimize AGL base as much as possible for low cost system.
- Possible use cases include motorcycles
- Functional Safety for Instrument Cluster

https://wiki.automotivelinux.org/_media/agl_roadmap_tokyo_2019_amm.pdf

AUTOMOTIVE GRADE LINUX

Member of Expert Group

OEM

Suzuki (Leader), Toyota, Honda, Mazda

Denso, Panasonic, Continental, NipponSeiki

DensoTen, AisinAW

Motivation

Create Cluster centric platform

- We want to create a base platform for Cluster, not a platform based on conventional IVI.

- There are different system requirements between IVI and Cluster.

- Instead of a system based on the conventional IVI system, it is necessary to consider a new system suitable for Cluster.

- We want to use AGL as the basis for production. In order to achieve this, we want to define performance targets that can satisfy the quality of the product, and create an AGL ecosystem that satisfies those targets.

AUTOMOTIVE
GRADE LINUX

What does Instrument Cluster EG aim?

- Shrink and optimize AGL base as much as possible for light weight system
  - e.g. Possible use cases include motorcycles

Demonstrate in ALS.
This is realized by one SoC

- Instrument Cluster with simple IVI
  - Target product system image example is as follows.
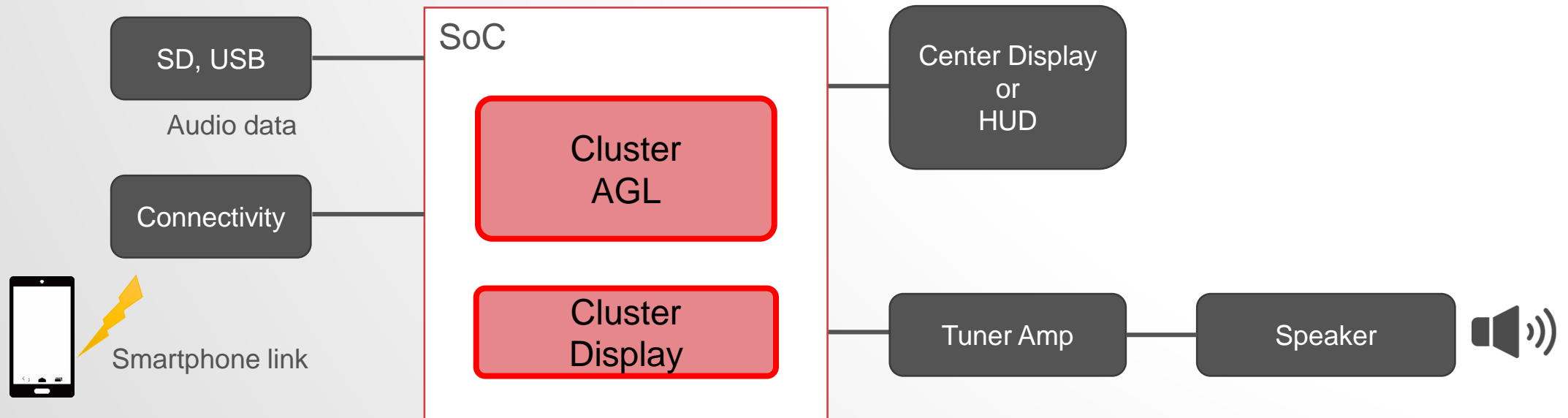
Relationship of IVI

Relationship of IVI and cluster function

- Cluster functions will not change so.

- The larger the HW Spec is, the more functions IVI have.

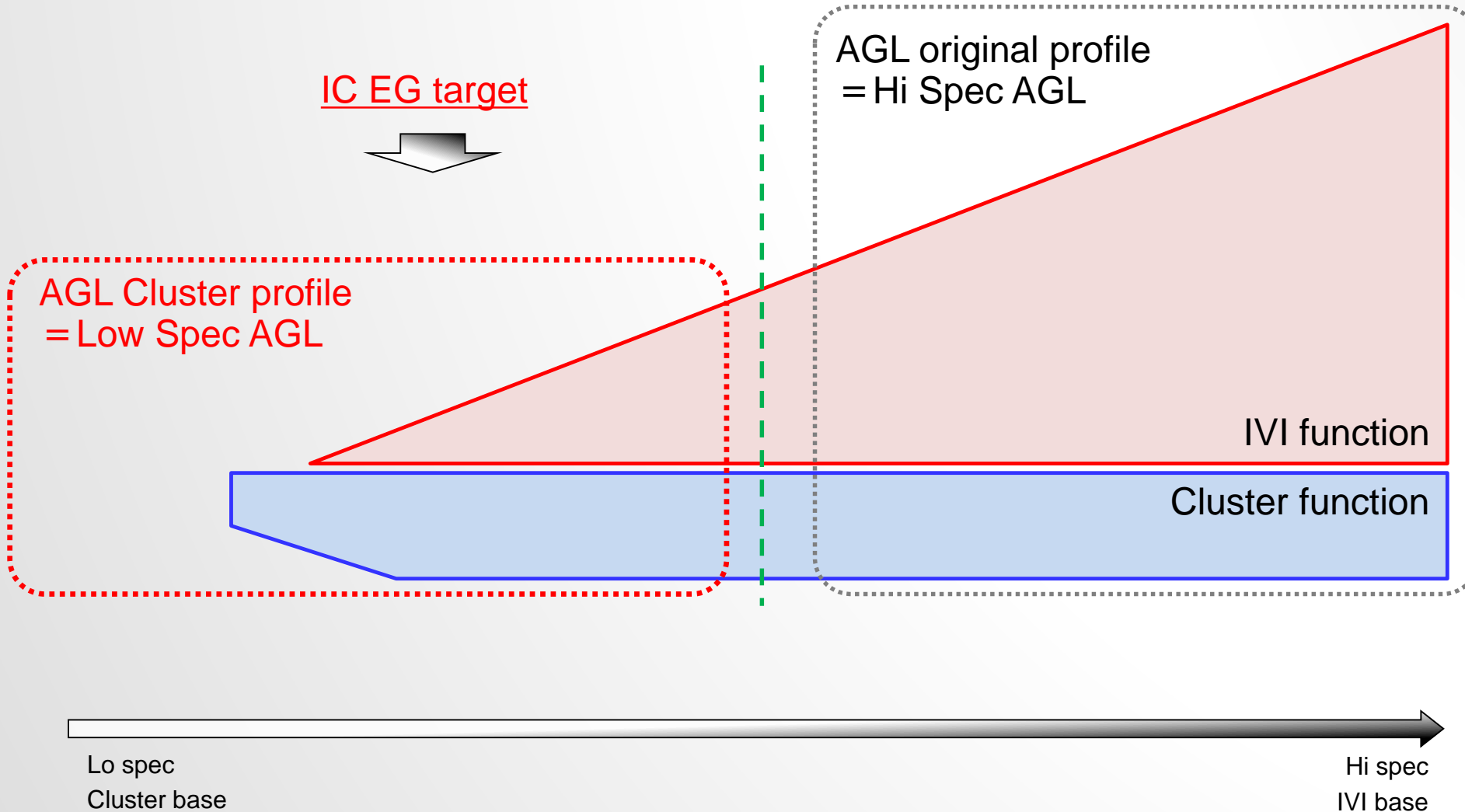## Approach to cluster centric platform

- There are different system requirements between IVI and Cluster

- Quality and robustness is required in cluster. (Functional Safety, Boot time, etc.)

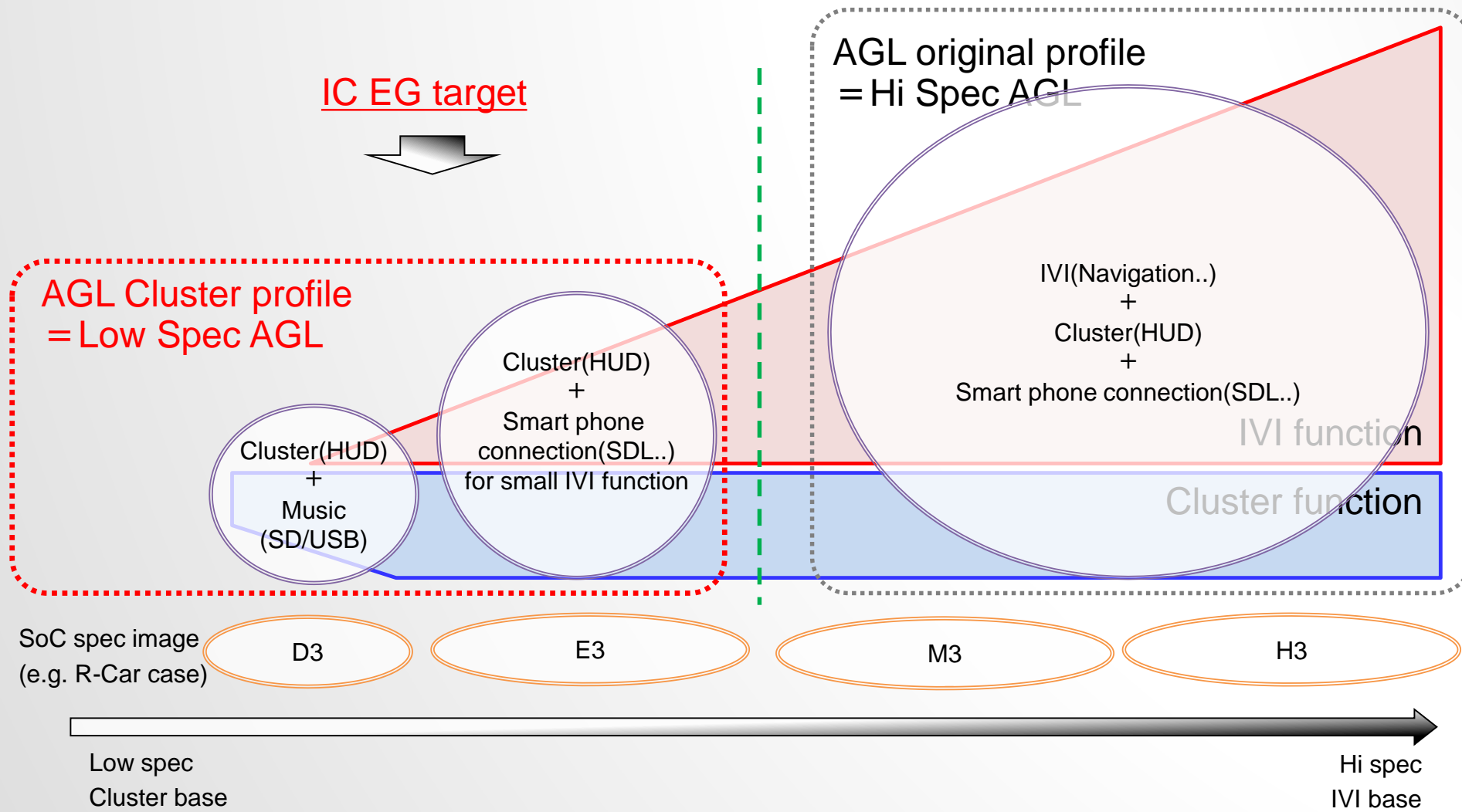➡ We call cluster centric platform as
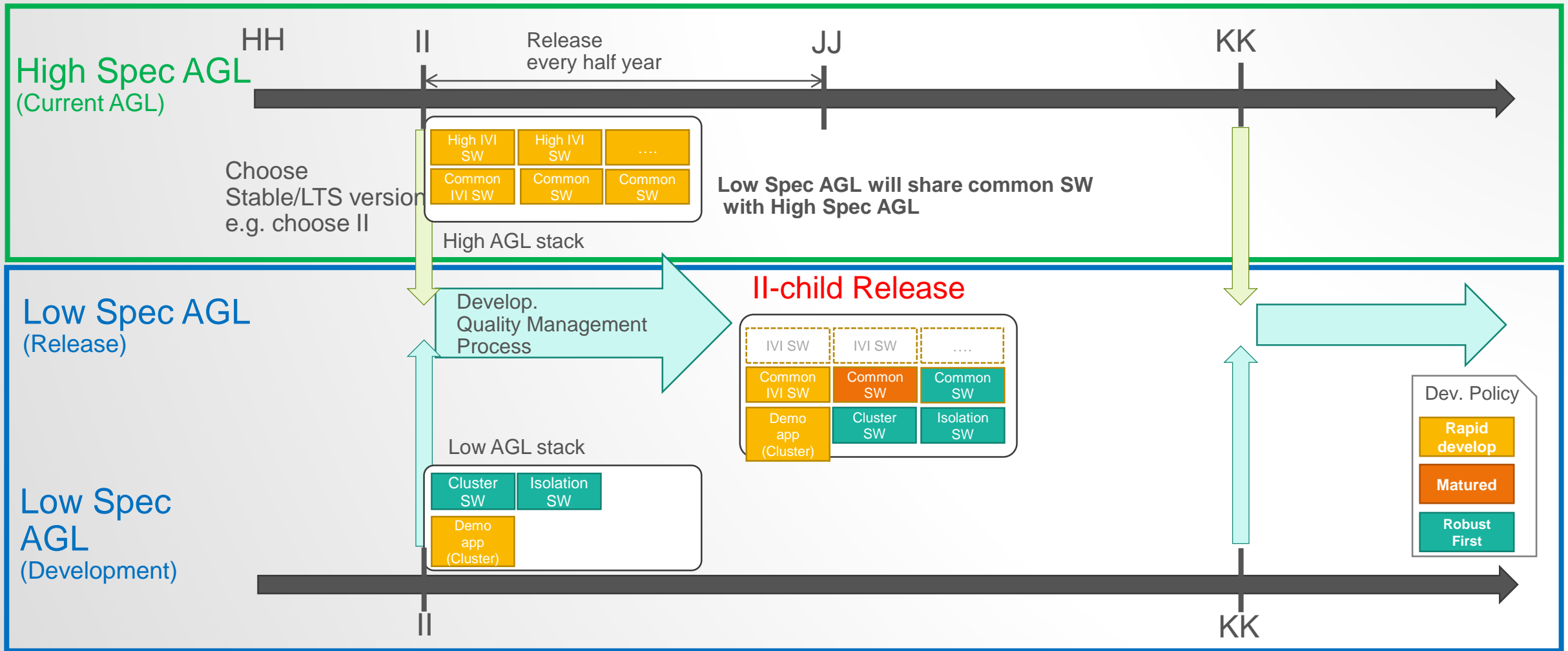
"Low Spec AGL"

Hi Spec and Low Spec AGL relation

IC EG target

AGL Cluster profile
= Low Spec AGL

AGL original profile
= Hi Spec AGL

IVI function

Cluster function

Lo spec
Cluster base

Hi spec
IVI base

Hi Spec and Low Spec relation

IC EG target

AGL original profile
= Hi Spec AGL

AGL Cluster profile
= Low Spec AGL

Cluster(HUD)
+
Music
(SD/USB)

Cluster(HUD)
+
Smart phone
connection(SDL..)
for small IVI function

IVI(Navigation..)
+
Cluster(HUD)
+
Smart phone connection(SDL..)

IVI function

Cluster function

SoC spec image
(e.g. R-Car case)

D3   E3   M3   H3

Low spec
Cluster base

Hi spec
IVI base

Development cycle of Low Spec AGL

High Spec AGL
(Current AGL)

HH   II   Release   JJ   KK
                                  every half year

Choose
Stable/LTS version
e.g. choose II

| High IVI SW | High IVI SW | .... |
| Common IVI SW | Common SW | Common SW |

High AGL stack

**Low Spec AGL will share common SW
with High Spec AGL**

Low Spec AGL
(Release)

Develop.
Quality Management
Process

II-child Release

IVI SW	IVI SW	....
Common IVI SW	Common SW	Common SW
Demo app (Cluster)	Cluster SW	Isolation SW

Dev. Policy

| Rapid develop |
| Matured |
| Robust First |

Low Spec AGL
(Development)

Low AGL stack

| Cluster SW | Isolation SW |
| Demo app (Cluster) | |

II   KK
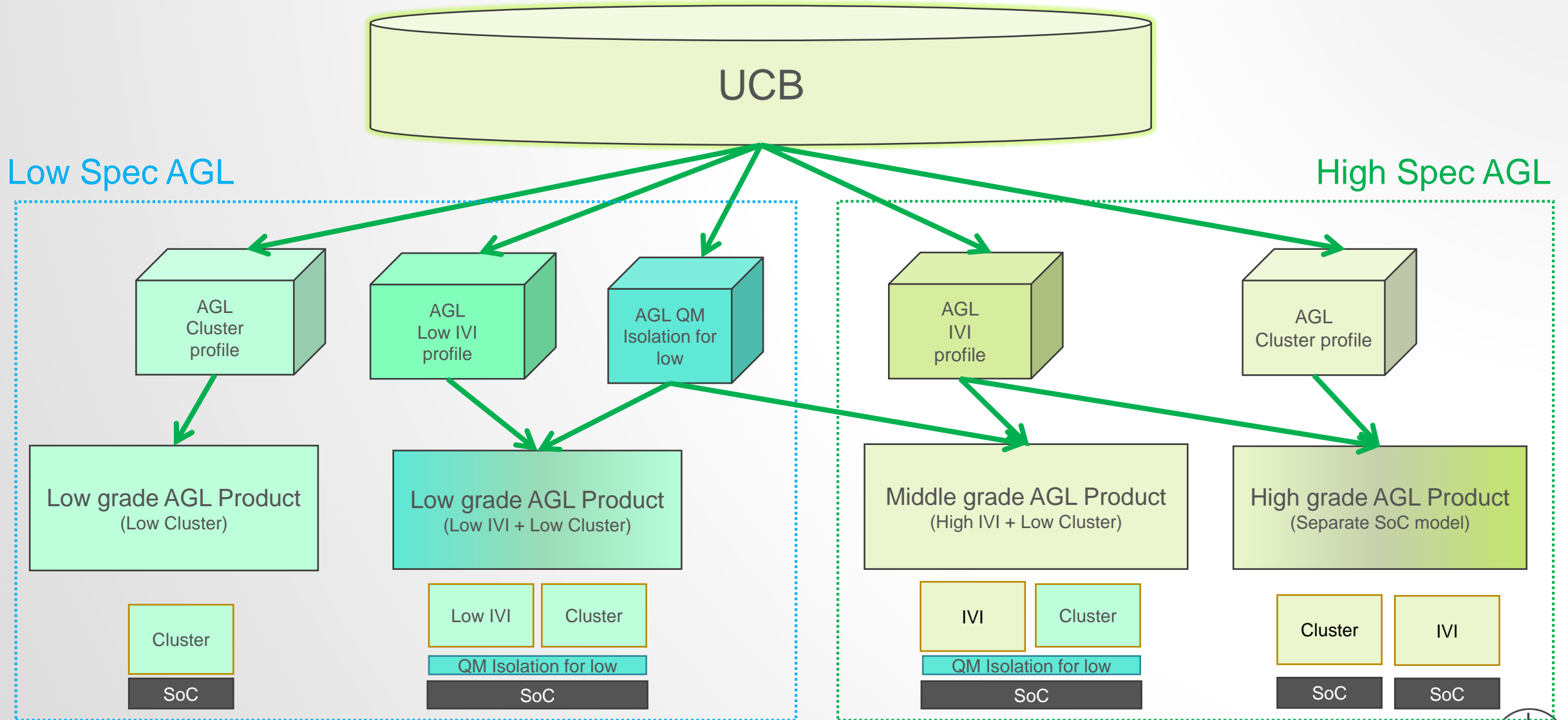
Development speed is different, so Low Spec AGL release will be slow.
Cluster's first priority is robustness, so eventually Low Spec AGL will branch off from High Spec AGL

AUTOMOTIVE GRADE LINUX

Relationship between High Spec AGL and Low Spec AGL

Goal - Realize AGL system suitable for Low Spec AGL

1. Quality and Robustness

- Approach the achievement of functional safety

2. Light weight system

- Work on Low Spec SoC

- Fast boot. Cluster must be ready as fast as current cluster product on RTOS.

3. Scalability and Maintainability

- Integrate cluster with small IVI integrated system because only cluster system is already in market

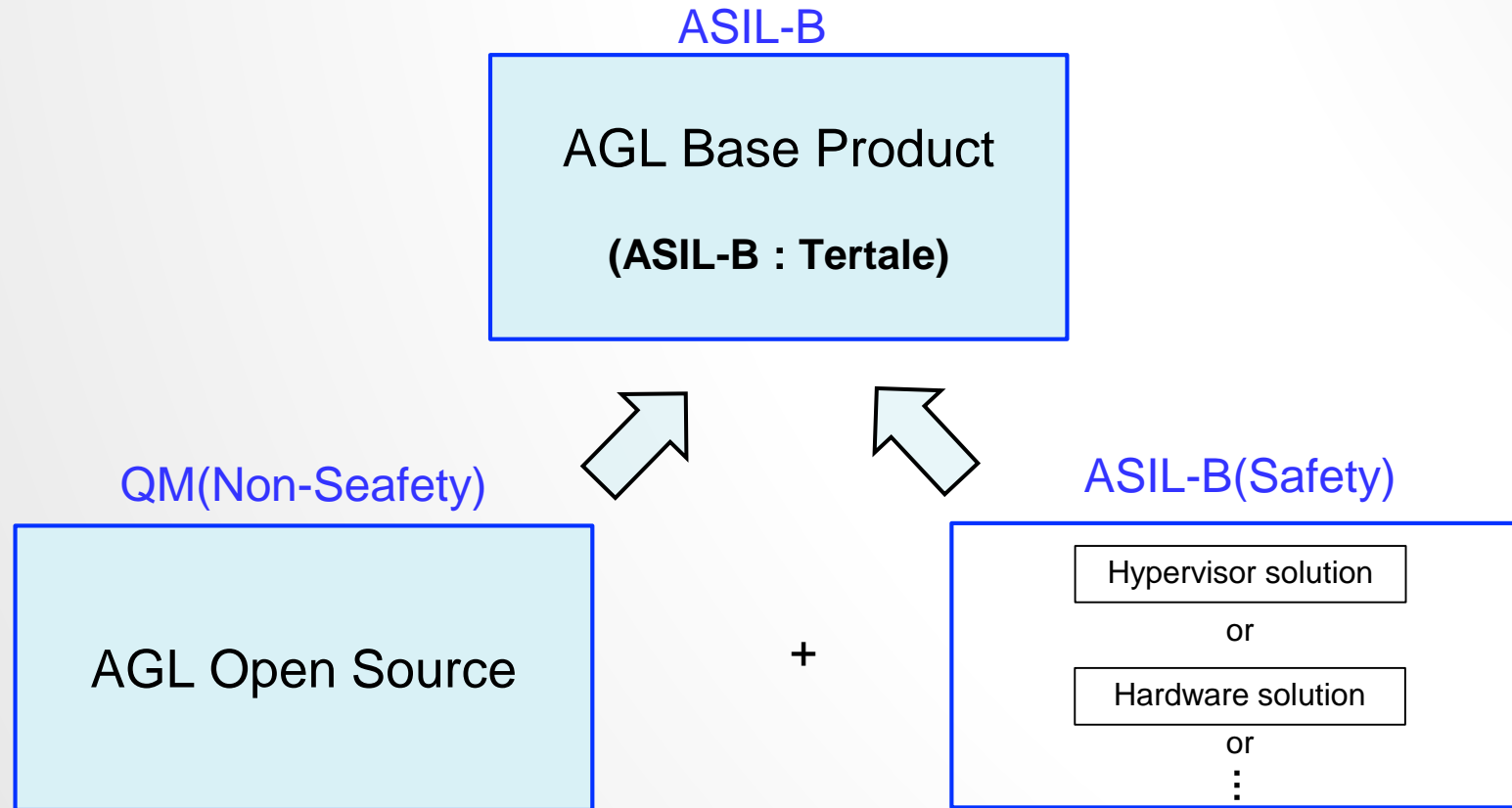- In product, enable to add new features, while keeping 1, 2 goal achieved

AUTOMOTIVE GRADE LINUX

Functional safety

- Functional safety has high dependency on vehicle system architecture.

- It is difficult to get ASIL-B with Linux. In functional safety method, the system should separate main function and safety function.

- Main function is a function of our system.

- Safety function is a function to safe of the vehicle, when the system fails.
  - It should monitor main functions.
    - Which main function must need a monitoring? It depends on your system design.
  - It must be isolated from main function.
  - It is not generic.

- We exclude functional safety from the scope of development.

Functional safety

ASIL-B

AGL Base Product

(ASIL-B : Tertale)

QM(Non-Seafety)

AGL Open Source

+

ASIL-B(Safety)

Hypervisor solution

or

Hardware solution

or
⋮

Achieve ASIL-B for the entire cluster system

Functional safety

Main function is included in the scope

- It requires advanced quality management.
- It requires open innovation.
- It requires cyber security.
- It requires fast boot.
- It requires various functions.
- …

Safety function is out of scope

- What function does it include?
- Which OS do you use?
- Which communication method do you use?

Depends on your system design!

Main function

Safety function

Functional safety will be discussed in the ELISA Project.

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Isolation method

Isolation method is out of scope

- Hardware separation? Using hypervisor?

Depends on your system design!

AUTOMOTIVE
GRADE LINUX

Functional safety

Main function is included in the scope

- It requires advanced quality management.
- It red
- It red
- It red
- It requires various functions.
- …

IC EG Target

   - Achieve Quality Management

Safety function is out of scope

- What function does it include?
- Which OS do you use?
- Which communication method do you use?

Depends on your system design!

Main
function

Safety
function

Functional safety will be discussed in the ELISA Project.

Isolation method

Isolation method is out of scope

- Hardware separation? Using hypervisor?

Depends on your system design!

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

AUTOMOTIVE
GRADE LINUX

What function will relate to Functional Safety ?

- Typically, warning lamp will relate to Functional Safety

- In other case, warning buzzer will relate to Functional Safety

Engine Stop
Transmission gear failure etc

- If those function is realized by main function, it should be monitored by safety side.

Antinomy in automotive quality management

- Automotive system (main function) has many antinomies.

IVI   Instrument Cluster

- **Rapid innovation**
  - Adding new future
  - Short-term development

← Antinomy →

- **Advanced quality management**
  - Full path coverage testing
  - Formal verification

- **Rapid bug fix**
  - Rapid security fix
  - Fix! Fix! Fix!

← Antinomy →

- **Careful bug fixes**
  - Does not want to change
  - Hates new bugs

- **Various functions**
  - Many pre-installed applications
  - Install applications from store

← Antinomy →

- **Selected functions**
  - Combinational verification
  - Fast boot

AUTOMOTIVE GRADE LINUX

Monolithic system development

IVI	Instrument Cluster

• Rapid innovation
   • New features
   • Short-term development
	• Advanced quality management
	   • Full path coverage testing
	   • Formal verification

← Antinomy →

If both IVI and Instrument Cluster are running on a single system:

• We have to do full path coverage test and formal verification when we want to add new features to IVI.

• It is not short-term development.

• That's why we will not be able to achieve rapid innovation

AUTOMOTIVE
GRADE LINUX

Monolithic system development

IVI

Instrument Cluster

- Rapid bug fixes
  - Rapid security fix
  - Fix! Fix! Fix!

← Antinomy →

- Careful bug fixes
  - Do not want to change
  - Hate new bugs

If both IVI and Instrument Cluster are running on a single system:

- The IVI system needs rapid security fixes. It's more important than new minor bugs. Because IVI system is connected to the outside.

- On the other hand we do not want to change common software stacks, because even minor bugs can cause serious situations. When we change the system, we have to perform verification.

AUTOMOTIVE
GRADE LINUX

Monolithic system development

IVI

Instrument Cluster

- Various functions
  - Many pre-installed applications
  - Install applications from store

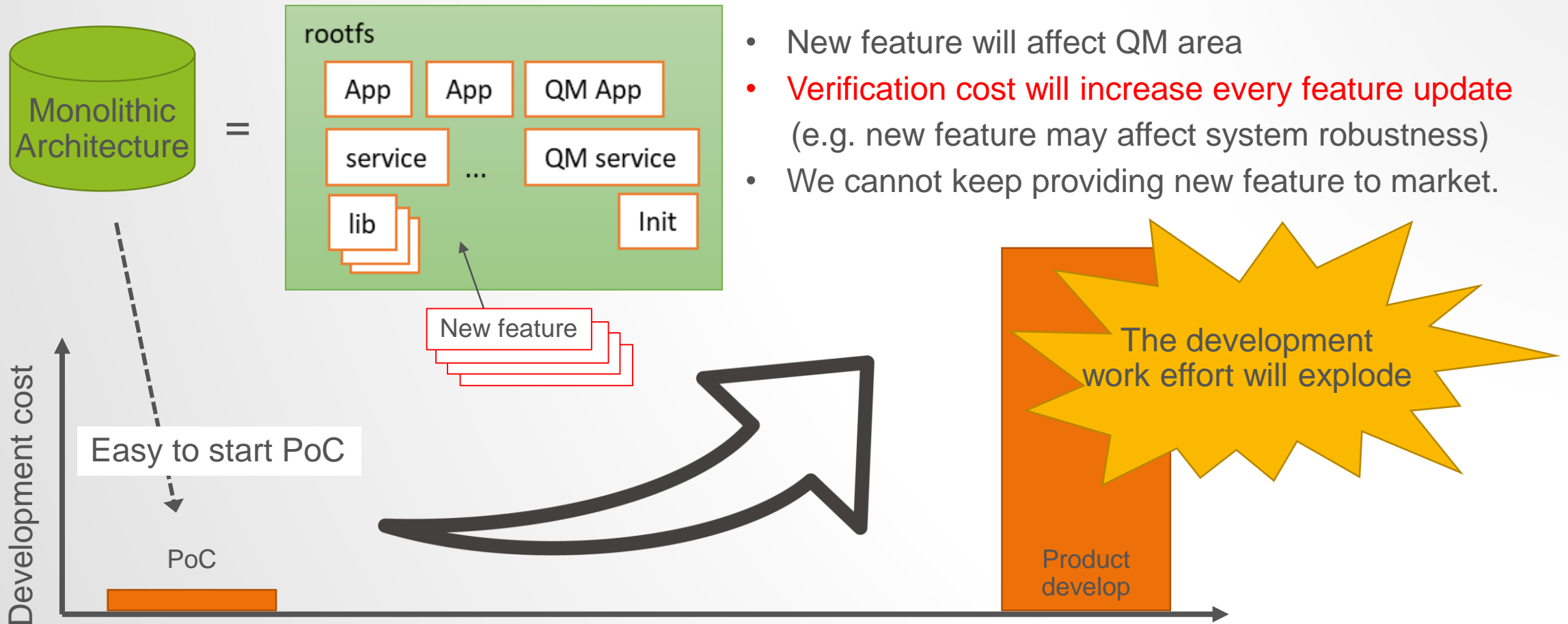← Antinomy →

- Selected functions
  - Combinational verification
  - Fast boot

If both IVI and Instrument Cluster are running on a single system:

- The IVI system has many pre-installed applications. End user can also install applications from the app store. On the other hand, Instrument Cluster has limited functionality because its safety related system requires combinatorial validation.

- It is impossible to perform combinatorial validation on a complex system. It means that it is difficult to integrate Instrument Cluster into an existing IVI system.

- A large application framework for IVI prevents fast boot of Instrument Cluster.

AUTOMOTIVE GRADE LINUX

Monolithic system development

rootfs

| App | App | QM App |

Monolithic Architecture =

service ... QM service

lib

Init

New feature

- New feature will affect QM area
- **Verification cost will increase every feature update** (e.g. new feature may affect system robustness)
- We cannot keep providing new feature to market.

The development work effort will explode

Development cost

Easy to start PoC

PoC

Product develop

It is difficult to apply monolithic architecture to instrument cluster product

So what solution ?

Answer is QM Isolation

QM Isolation

- Our answer to solve various antinomies is "one more isolation method" with open source.

EG scope

Isolated by Container

Other   Low IVI   Low Cluster

For Advanced quality management
Selected software with full path coverage testing and formal verification.

For rapid innovation and bug fix
The runtime environment is isolated from other software stacks by container to realize rapid innovation.

For Fast Boot
Miniaturized rootfs with minimum functions.

Container host   Container runtime

Safety function

Main functions are isolated based on their QM level, booting time, incident type, etc.

Linux Kernel

Isolation method (low layer)

AUTOMOTIVE GRADE LINUX

What is QM isolation

- Separate and optimize each software stacks using containers.
  - Various isolations such as access control and resource restrictions are performed by the container.
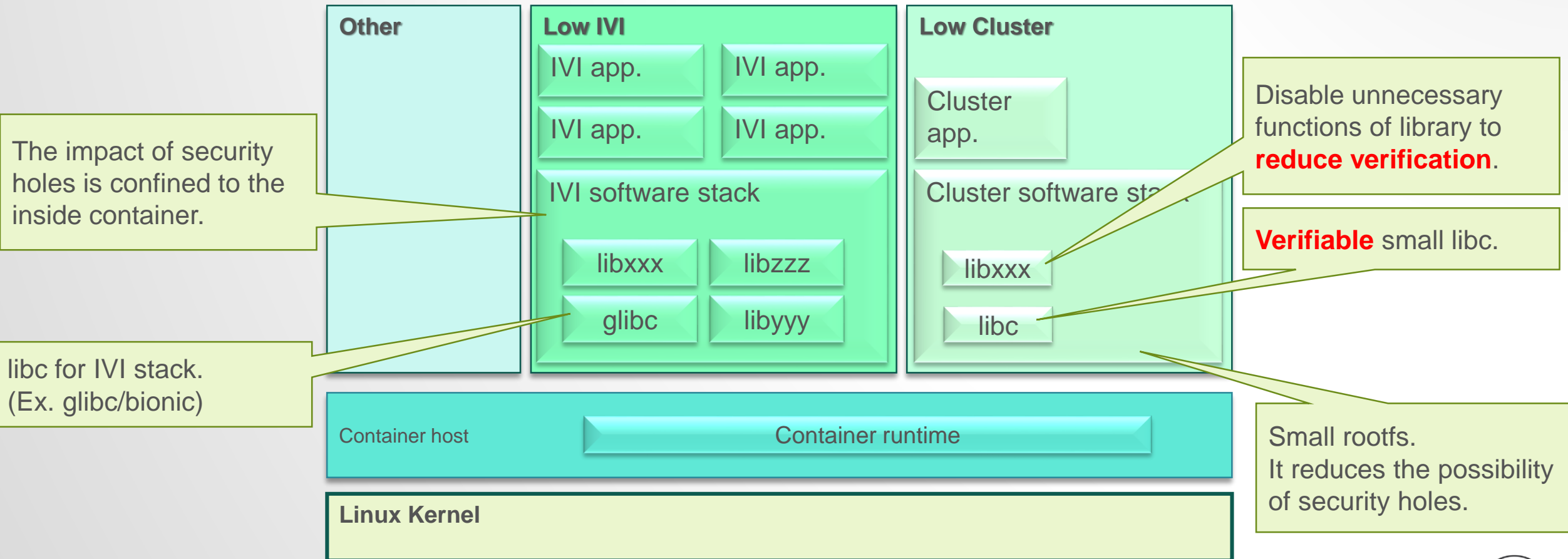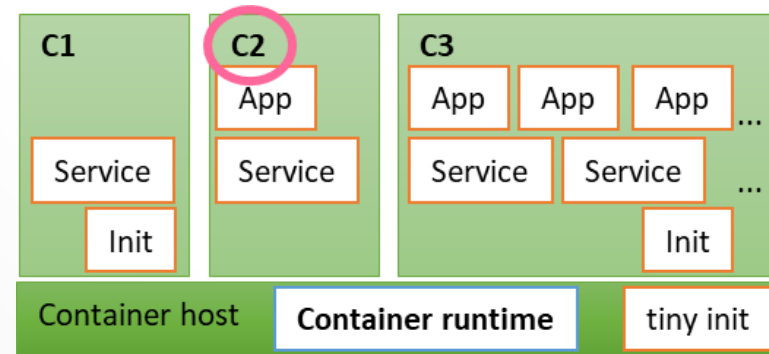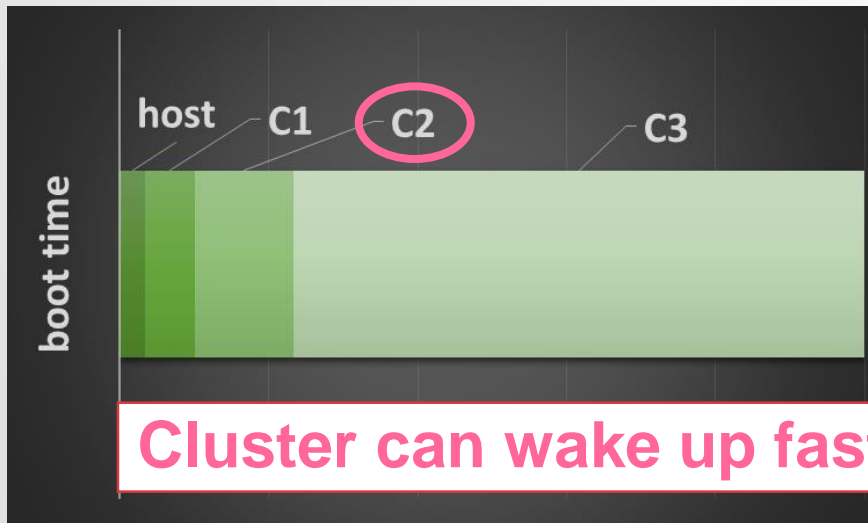
Why does container can boot fast?
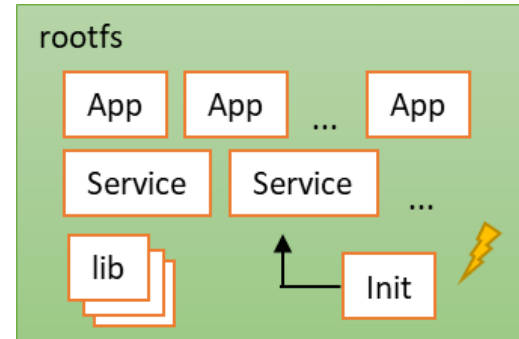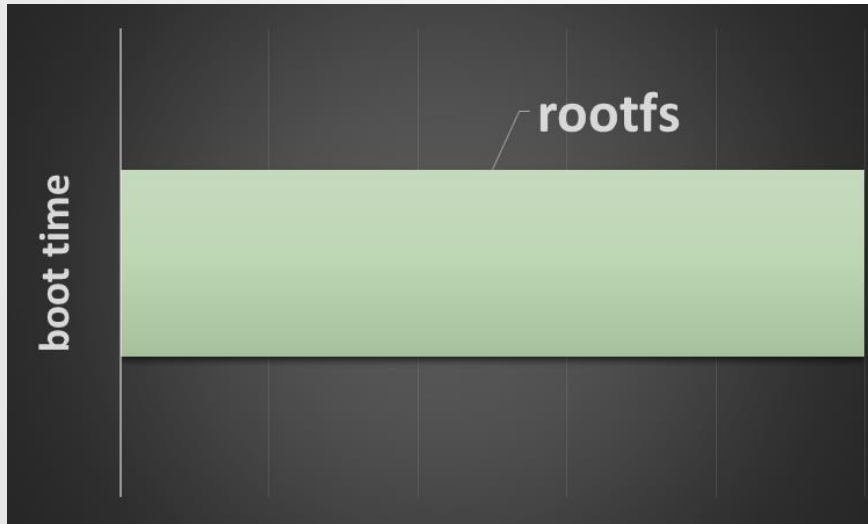
- System has all resources in one rootfs
- System size will be big
- Launch all system components at one time.

- Separate filesystem into several containers
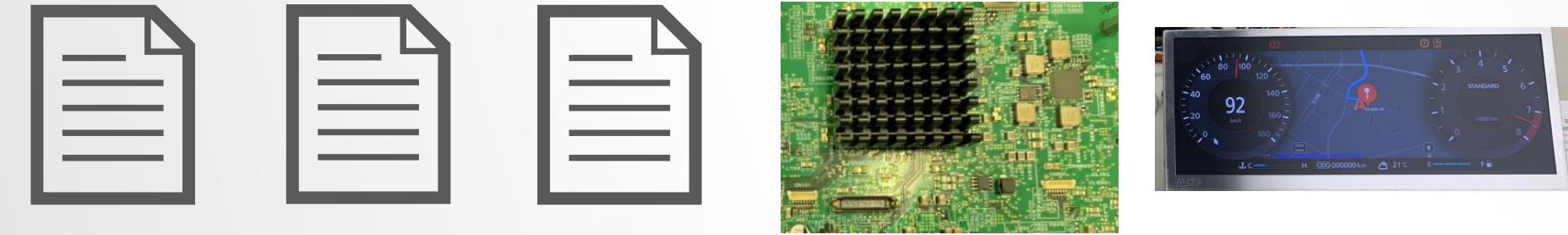- Each container and host size can be small
- Separate booting

Cluster can wake up fast !

Challenge of container architecture

Quality Management for Linux Kernel

- We would like chip vender offer Kernel for QM isolation

- Minimize Kernel land (move drivers to user land)

- Cgroup, namespace and LSM is necessary at the moment
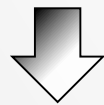
- Collabolate with ELISA Project

Current Work
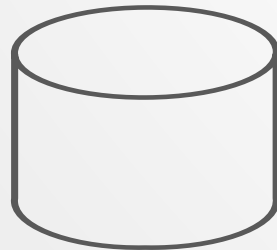

Architecture

Target
Performance

Spec
(Spec EG)

Reference
Hardware
(R-Car E3)

Reference Design

Create source code according to above information

Source code

AUTOMOTIVE
GRADE LINUX

Next…

- Start detail architecture discussion e.g. Container, GUI, Sound, CAN etc.

- Start discussion with ELISA Project for Functional Safety and Quality Management.

- Create Spec

- Create Requirement for Cluster and Low IVI collaborating with Spec EG

Thank you