SW AS A CAPITAL

DEVELOP REUSABLE SW FROM THE BEGINNING

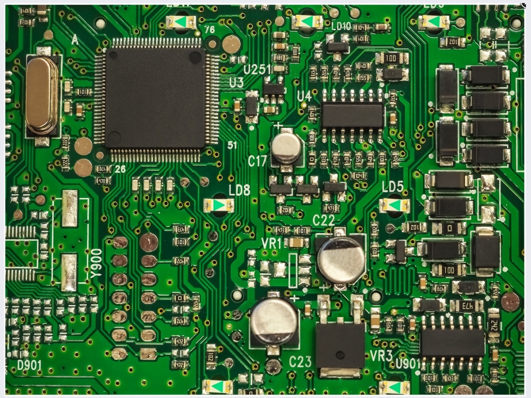
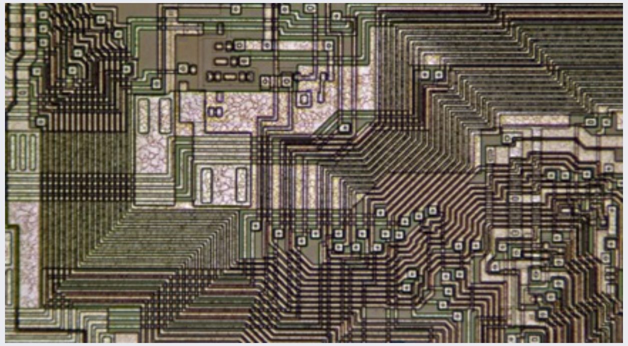
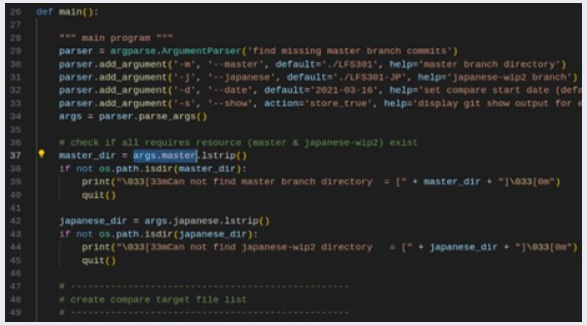
HISAO MUNAKATA
SENIOR DIRECTOR, HPC MARKETING DIGITAL DIVISION

DAN SISCO
SENIOR DIRECTOR, R-CAR GEN5 PROGRAM MANAGER

HIGH PERFORMANCE COMPUTING, ANALOG AND POWER SOLUTIONS GROUP
RENESAS ELECTRONICS CORPORATION

TRANSITION OF INNOVATION SOURCE

HOW SW BECAME THE CENTER OF TECHNOLOGICAL GRAVITY

	In the past	Recently	In the future
Core competitiveness	PCB circuitry	SoC	SW
Technology owner	Tier1	Silicon supplier	SW platformer
Key demand	robustness	integration	portability
Image			
Challenges	ECU interconnect, age-related fault	Performance, power (thermal)	SW update, cyber security

```
def main():

    """ main program """
    parser = argparse.ArgumentParser('find missing master branch commits')
    parser.add_argument('-m', '--master', default='./LFS301', help='master branch directory')
    parser.add_argument('-j', '--japanese', default='./LFS301-JP', help='japanese-wip2 branch')
    parser.add_argument('-d', '--date', default='2021-03-16', help='set compare start date (defa
    parser.add_argument('-s', '--show', action='store_true', help='display git show output for e
    args = parser.parse_args()

    # check if all requires resource (master & japanese-wip2) exist
    master_dir = args.master.lstrip()
    if not os.path.isdir(master_dir):
        print("\033[33mCan not find master branch directory  = [" + master_dir + "]\033[0m")
        quit()

    japanese_dir = args.japanese.lstrip()
    if not os.path.isdir(japanese_dir):
        print("\033[33mCan not find japanese-wip2 directory   = [" + japanese_dir + "]\033[0m")
        quit()

    # ------------------------------------------------
    # create compare target file list
    # ------------------------------------------------
```

Now, many OEMs claim SW has become a primary source of product competitiveness (= SW first concept)

STILL, ONE SIZE (SW) DOES NOT FIT ALL

EMBEDDED SYSTEMS HARDLY DECOUPLE HW (SOC) AND SW

HW variation

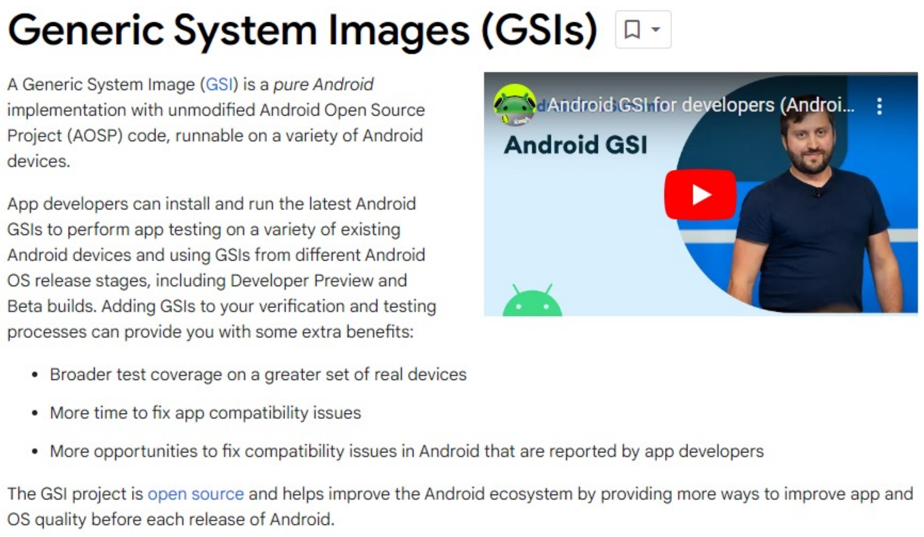
- Require performance (=price) variants
- Expect unique competitive features (display numbers, AI,..)

SW commonality

- Production variants (same generation)
- SW migration (across the generation)

HW/SW boundary

- Application portability (OS interface)
- HW access interface (common API)
- System level boundary



Generic System Images (GSIs)

A Generic System Image (GSI) is a *pure Android* implementation with unmodified Android Open Source Project (AOSP) code, runnable on a variety of Android devices.

App developers can install and run the latest Android GSIs to perform app testing on a variety of existing Android devices and using GSIs from different Android OS release stages, including Developer Preview and Beta builds. Adding GSIs to your verification and testing processes can provide you with some extra benefits:

- Broader test coverage on a greater set of real devices
- More time to fix app compatibility issues
- More opportunities to fix compatibility issues in Android that are reported by app developers

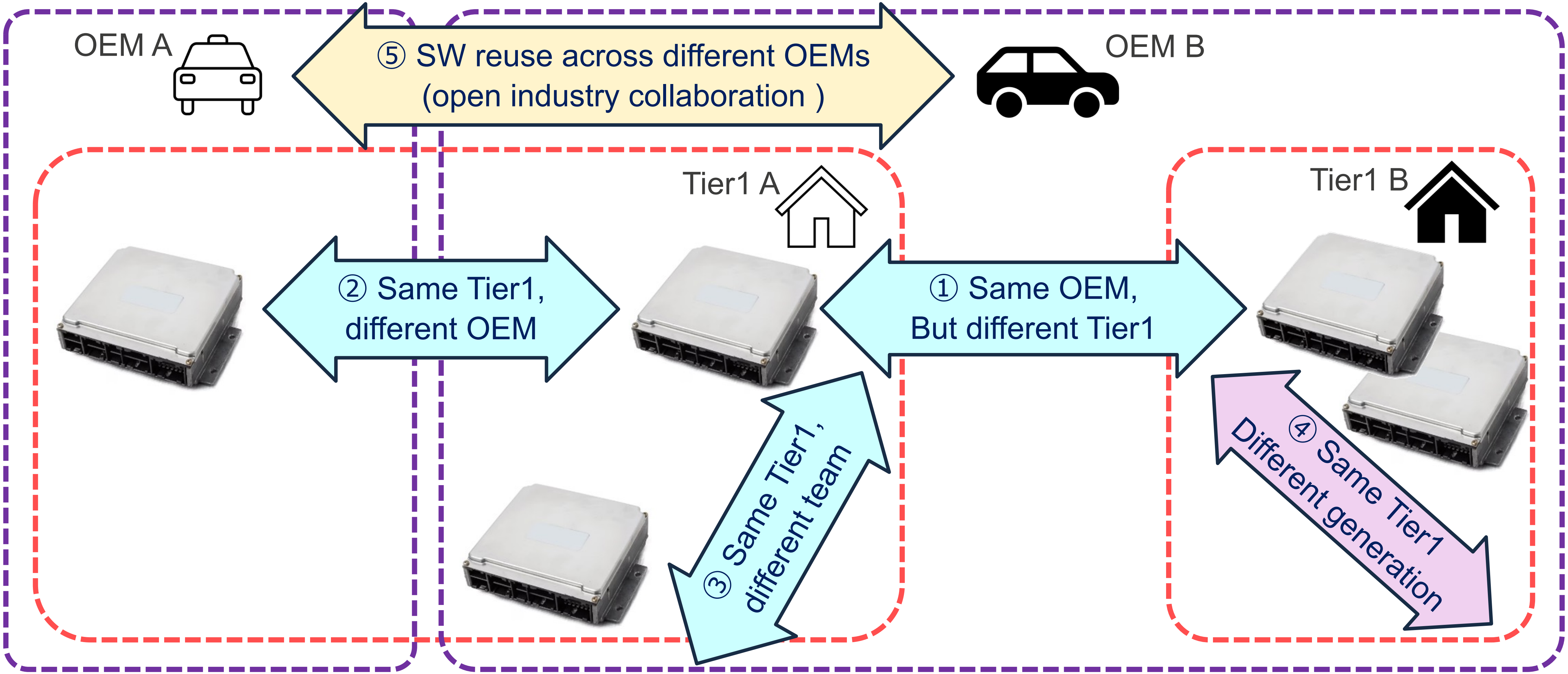
The GSI project is open source and helps improve the Android ecosystem by providing more ways to improve app and OS quality before each release of Android.

https://developer.android.com/topic/generic-system-image

Android mandates handset vendors to support GSI boot, a common binary that can boot multiple devices.

EXPECTED SW REUSE CANDIDATES

… AND VARIOUS BLOCKING FACTORS FOR EMBEDDED SW REUSE



VARIOUS BLOCKING FACTORS FOR SW REUSE

NOT ONLY HW-ORIENTED, BUT ALSO SW-RELATED AND BUSINESS-RELATED

HW origin

- **Performance**
  - Speed
  - Memory size
- **Major IP incompatibility**
  - CPU / GPU / NPU
- **Minor IP incompatibility**
  - Register function assign
  - Channel number
- **Environment**
  - IDE, debug board,..

SW origin

- **Data definition**
  - Common language
- **Coding convention**
  - MISRA-C
- **Code management**
  - Repository access
- **Readability**
  - Document density
  - Language

Business origin

- **Contract**
  - Code disclosure
  - Asset ownership
- **License**
  - Copyright restriction
  - Warranty scope
- **Deliverables**
  - Source code access

SW AS A *CAPITAL*, NOT AN *ASSET*

CAPITAL SHOULD BE RESOURCED TO GENERATE REPEATED FUTURE VALUE

SW as an ***Asset*** (traditional way)

- Acquired to date

Collection of past SW code became bad debt

- Accumulated over time
- Care for the preservation of assets
- The **value diminishes over time**
- Hard to maintain the present value
- Becomes a **non-performing asset**

SW as a ***Capital*** (new direction)

- **Source of future value**
- **Core of the growth strategy**
- **To increase the prospective value, you need to reinvest in the SW**
- **SW management strategy** should include **"What exactly to do for expansion and reinvestment?"**

“CAPITAL TURNOVER RATE” GIVES PERFORMANCE INDEX

IT SHOWS HOW MUCH REVENUE GENERATED FROM THE EXISTING ASSET

Capital Turnover Rate (Accounting terms)

- **Capital Turnover Rate = Annual sales / stockholder equity (net worth)**
  - The ratio indicates how much a company could grow its current capital investment level.
  - Its value varies by type of business. (1.3 ~ 3.5)

- **SW Capital turnover Rate = Reused code / Total developed code**
  - We can calculate the true code reuse rate.
  - That does not include the use of vague experience and know-how.
  - We need to consider how we can improve the true code reuse ratio.

HOW TO REALIZE CAPITALIZED SW (= REUSABLE SW)

”REUSE” IS ESSENTIAL FOR SW CAPITALIZATION

Eliminate Assumptions

- **Do not optimize the code**

Optimize may include assumption

- HW configuration (ch. number)
- Data range
- Memory size
- Execution speed (network,..)

Follow Coding Conventions

- Style format
  (tab length, codepage,..)
- Pursuit of readability
  (doxygen support,..)
- Nomenclature
  (adopt common naming)
- Use code control system
  (proper use of git system)

Interoperability with other industries

- Cloud connectivity
- Mobile network connectivity
- Vehicle service connectivity
- common control vocabulary (shared semantics) for data sharing/integration

WHAT COVESA CONTRIBUTES TO REALIZE CAPITALIZED SW

COVESA HOSTS VARIOUS COLLABORATIVE PROJECTS THAT PROMOTE SW CAPITALIZATION

Data Expert Group

- Data Models & Science
  - VSS eco-system
- Common Vehicle Interfaces
  - VehicleAPI (Autosar)
  - Service definitions
  - VISS
  - IFEX IDL framework
- Architecture / Infrastructure
  - Data Layer
  - Data Centric Arch

Automotive AOSP App Framework Standardization Expert Group

- Non-GAS emulator
- Data/Sensor API Definition
  - e.g. camera
- Push Notifications
- Login & Identification
- Payment
- Location Based Services
- Driver distraction
- Display management
- Touchpad support

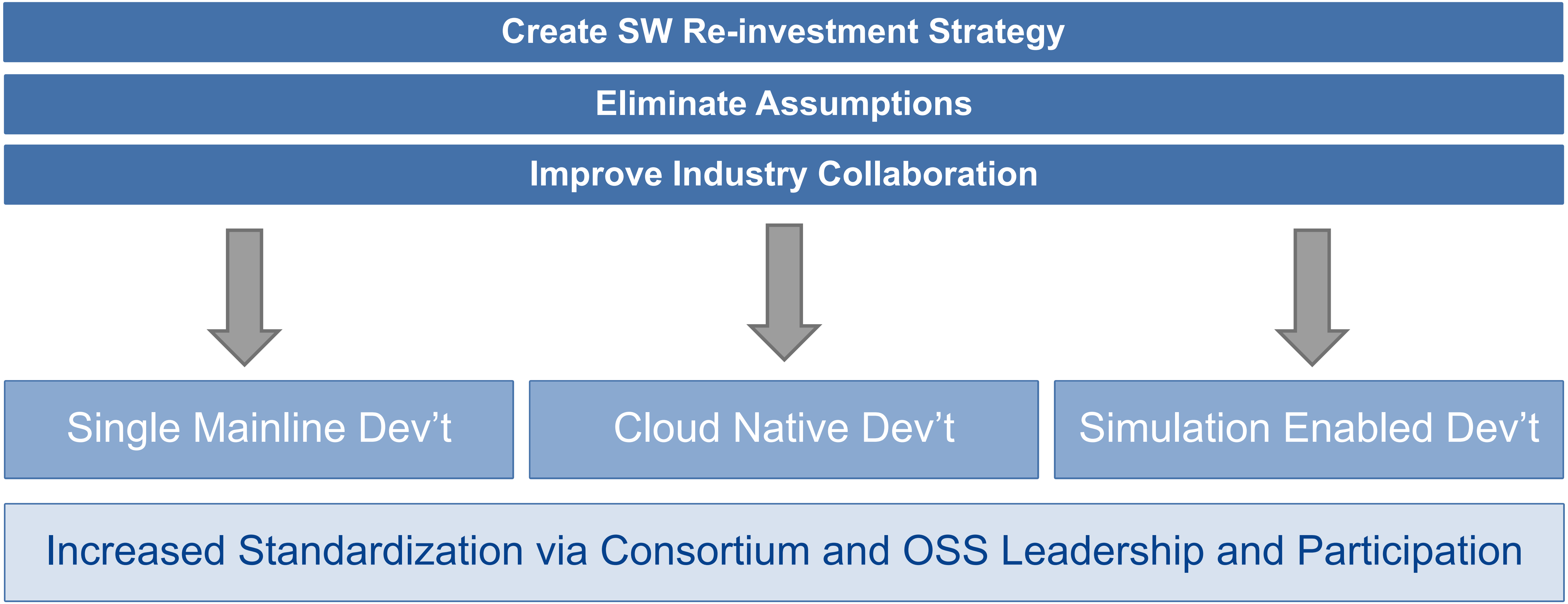
Electric Vehicle Charging Expert Group

- EV Charging Event Data Aggregation
- EV Optimization
- Private Cross OEM Joint Compute for Changing

- Delivering great EV experiences

TRANSLATING SW CAPITALIZATION TO RENESAS

DEVELOPMENT PILLARS TO SUPPORT THE TRANSFORMATION



Create SW Re-investment Strategy

Eliminate Assumptions

Improve Industry Collaboration

Single Mainline Dev’t

Cloud Native Dev’t

Simulation Enabled Dev’t

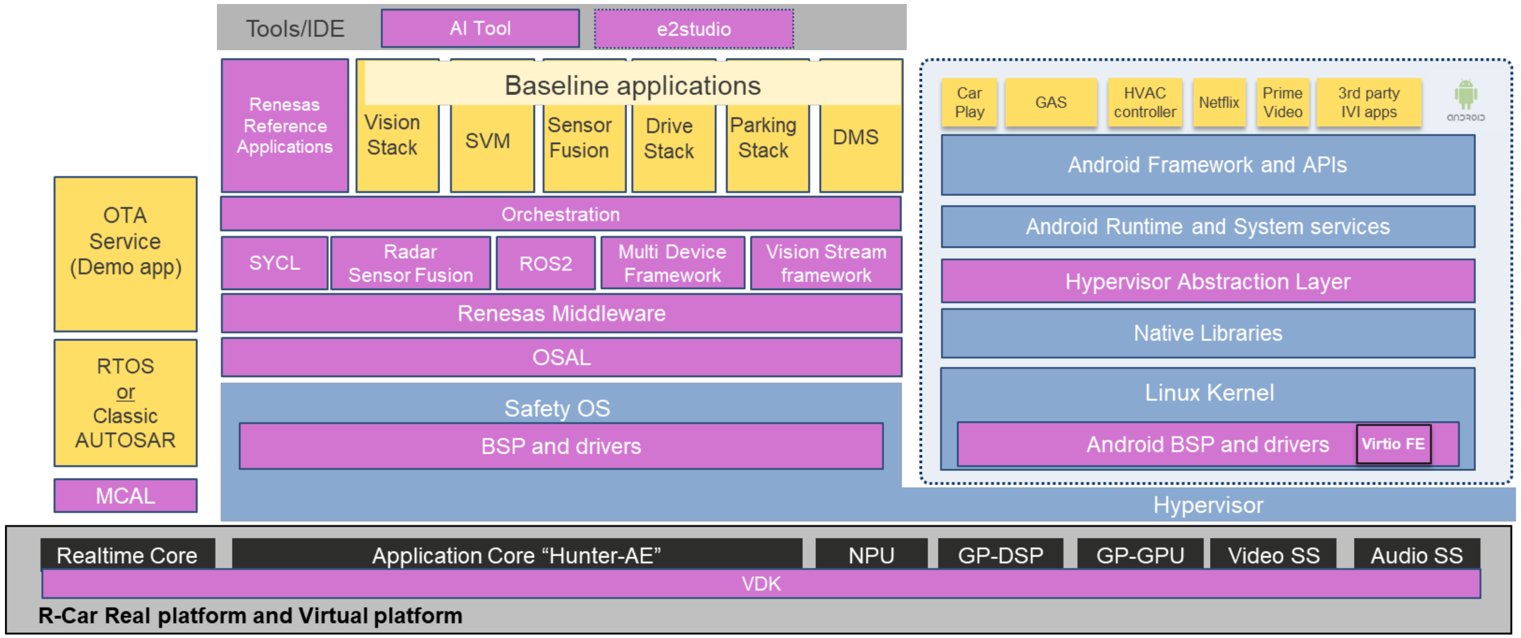
Increased Standardization via Consortium and OSS Leadership and Participation

RENESAS PILLARS TO PURSUE SW CAPITALIZATION

SINGLE MAINLINE DEVELOPMENT

Single Mainline Dev’t

- “Single mainline” development co-locating and integrating all SW features at right
- Increased chipset and vehicle system integration planning and test
  - better visibility into SW re-use in real world cases
- Optimizations and customizations become branches from here



Tools/IDE

AI Tool

e2studio

Renesas Reference Applications

Baseline applications

Vision Stack

SVM

Sensor Fusion

Drive Stack

Parking Stack

DMS

Orchestration

SYCL

Radar Sensor Fusion

ROS2

Multi Device Framework

Vision Stream framework

Renesas Middleware

OSAL

Safety OS

BSP and drivers

OTA Service (Demo app)

RTOS or Classic AUTOSAR

MCAL

Car Play

GAS

HVAC controller

Netflix

Prime Video

3rd party IVI apps

android

Android Framework and APIs

Android Runtime and System services

Hypervisor Abstraction Layer

Native Libraries

Linux Kernel

Android BSP and drivers

Virtio FE

Hypervisor

Realtime Core

Application Core “Hunter-AE”

NPU

GP-DSP

GP-GPU

Video SS

Audio SS

VDK

R-Car Real platform and Virtual platform

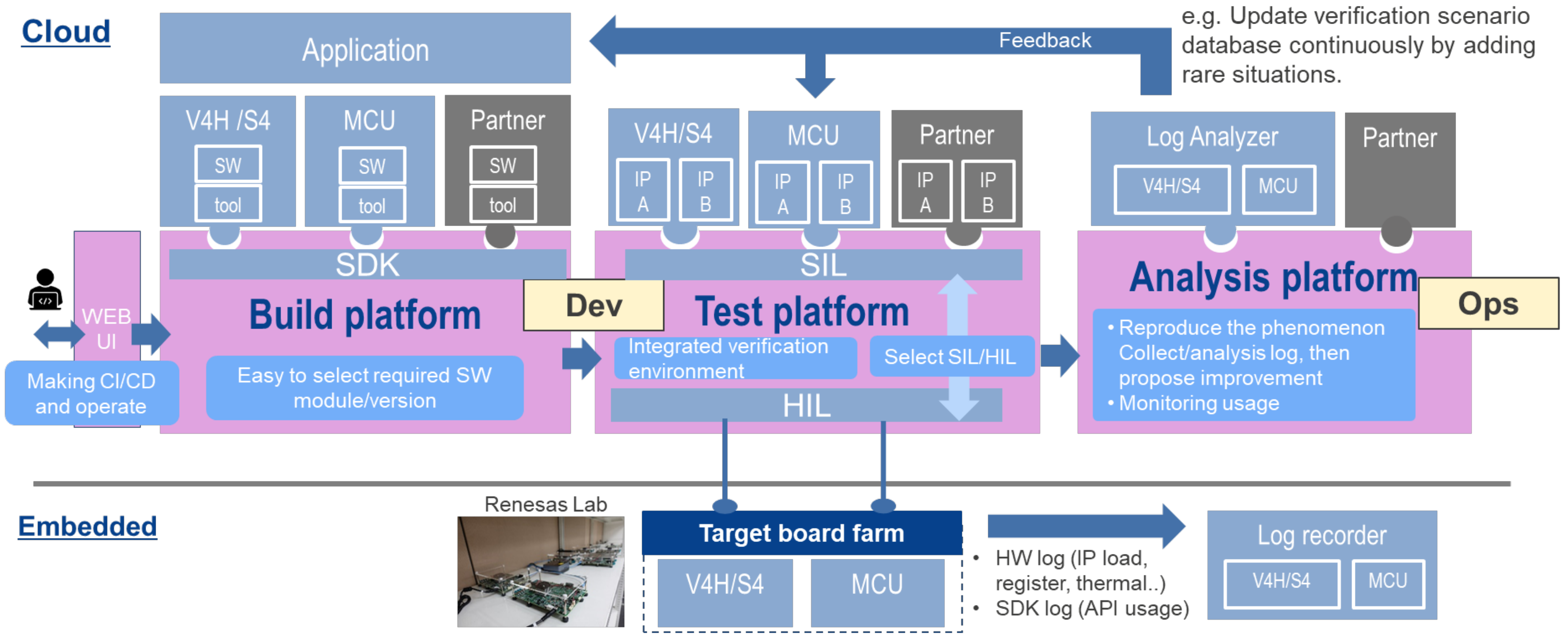
Make smart optimizations to avoid jeopardizing SW re-useability

RENESAS PILLARS TO PURSUE SW CAPITALIZATION

CLOUD NATIVE DEVELOPMENT THROUGHOUT VEHICLE LIFETIME

Cloud Native Dev’t

- “One stop shop”: Single point of entry and operation for complete SW dev’t workload
  - Product Simulation (fast/medium/slow)
  - AI analysis/simulation
  - Documentation/SW access
  - Compute rental mechanisms (HiL, SiL)



e.g. Update verification scenario database continuously by adding rare situations.

- Reproduce the phenomenon Collect/analysis log, then propose improvement
- Monitoring usage

- HW log (IP load, register, thermal..)
- SDK log (API usage)

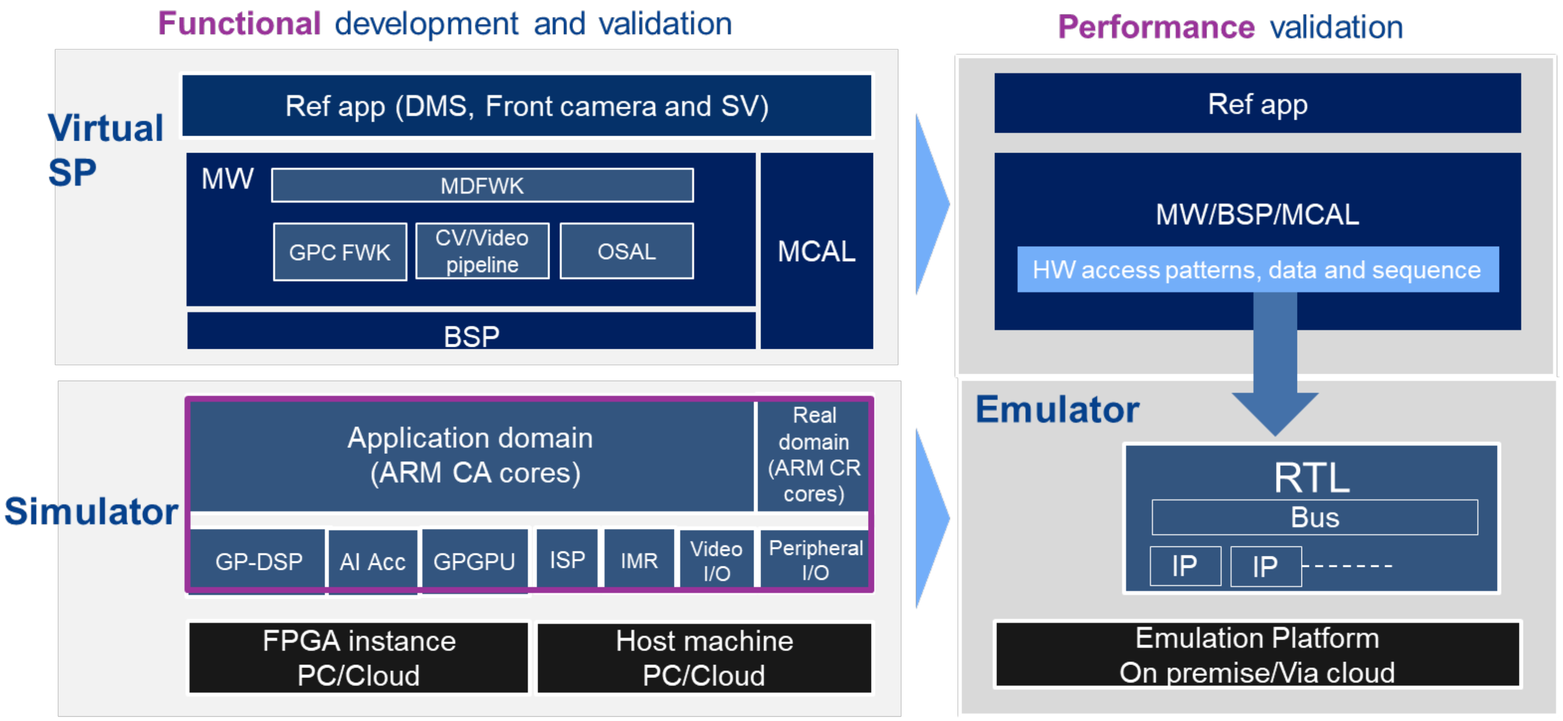
Cloud Native to pull Dev’t Left and Enable Broader and Easier SW Access

RENESAS PILLARS TO PURSUE SW CAPITALIZATION

SIMULATION ENABLEMENT FROM THE START

Simulation Enabled Dev't

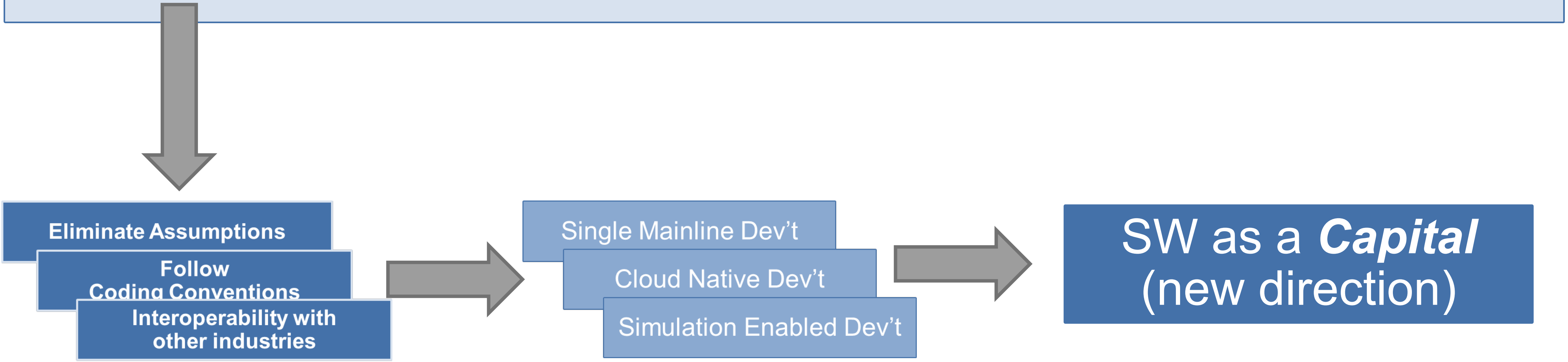
- Renesas Automotive SW IDE includes simulation components since 2021
  - Multi-device simulation
  - Enhanced debug/trace
- Now we expand on this
  - Flexibility of simulation (speed/accuracy)
  - Deeper AI toolchain simulation



Decouple SW Value and Strategy from HW Access

OUR JOURNEY…

Renesas is making great efforts to **master our own journey towards SW capitalization**, and to **enable our customers and partners** to master their own journeys…



Eliminate Assumptions

Follow Coding Conventions

Interoperability with other industries

Single Mainline Dev’t

Cloud Native Dev’t

Simulation Enabled Dev’t

SW as a ***Capital*** (new direction)

GENERAL METHODS

RENESAS APPROACH

Renesas.com