AGL past and future:
which innovations should drive next-gen AGL evolution

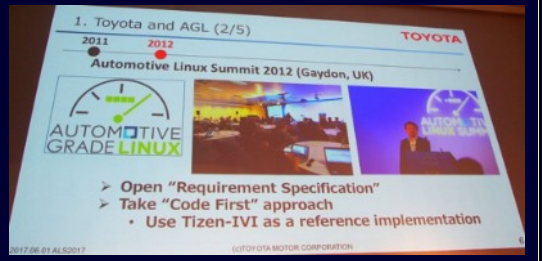
Hisao Munakata
AGL Advisory board (Renesas)

AGL AMM 2021 spring (17th/18th Mar. 2021)

AGL's Future Looks Bright!!

Quoted from Dan's 2020 AMM spring keynote

AGL history



TIZEN

Sep

EG
*not all

App framework
CIAT
UI & Graphics
Reference HW
Instrument & Cluster
virtualization (phase 1)
(phase 2)
IVI-PR

demo: CES 2016, CES 2017, CES 2018, CES 2019, CES 2020

UCB: AA, BB, CC, DD, EE, FF, GG, HH, II, JJ, KK

2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022

AGL history

EG
*not all

IOT BZH — App framework
Baylibre — CIAT
COLLABORA — UI & Graphics
ADIT
Reference HW
Instrument & Cluster
Virtual Open Systems — virtualization (phase 1) — (phase 2)
IVI-PR

Project management

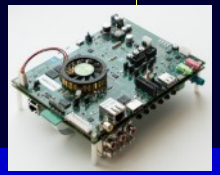
Platinum tier added

IOT BZH
Konsulko Group
igalia

OEM & Tier1

Baylibre
FIBERDYNE
COLLABORA

Contractor

intel
RENESAS

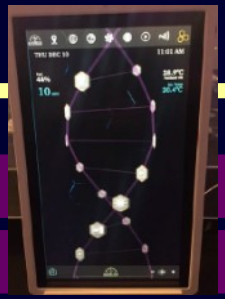
Semiconductor

2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022

Does AGL gravity start shifting ?

vehicle services

vehicle applications (OEM & Tier1)



Linux kernel (device driver & core kernel functions)

Automotive grade silicon & components bring-up

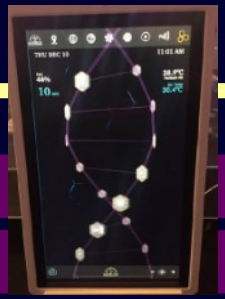
2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022

Does AGL gravity start shifting ?

vehicle services

vehicle applications (OEM & Tier1)

Some AGL members, especially newly joined members seems envisioning more aggressive vehicle-to-cloud interaction now.



Linux kernel (device driver & core kernel functions)

Automotive grade silicon & components bring-up

2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

New challenges to decouple mobile platform bindings

MOBILE NATIVE FOUNDATION

Improving engineering for large-scale mobile applications.

The Mobile Native Foundation provides a place to collaborate on open source projects and discuss wide ranging topics in order to improve processes and technologies for large-scale Android and iOS applications.

https://mobilenativefoundation.org/

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

New challenges to decouple mobile platform bindings

MOBILE NATIVE FOUNDATION

Improving engineering for large-scale mobile applications.

The Mobile Native Foundation provides a place to collaborate on open source projects and discuss wide ranging topics in order to improve processes and technologies for large-scale Android and iOS applications.

https://mobilenativefoundation.org/

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

AGL = Common Automotive SW platform creation

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

AGL = C mmon Auto ive SW pla rm creation

Each OEM willing to operate own vehicle service platform
(No cross automotive-industry common API requested)

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

Service Platform “A”

Service Platform “B”

Service Platform “C”

Service Platform “D”

OEM A

OEM B

OEM C

OEM D

AGL UCB

IT service forks seems demanding “decoupling”

Target agnostic

* data representation
* algorithm

service layer

Some platform dependency

platform layer

Strong HW dependency

library layer

device layer

(A) Common high-level API surface (SOA)

Service Platform “A”

Service Platform “B”

Service Platform “C”

Service Platform “D”

OEM A

OEM B

OEM C

OEM D

(B) Common low-level API surface (HAL)

Dreaming fully portable SW, is that possible?

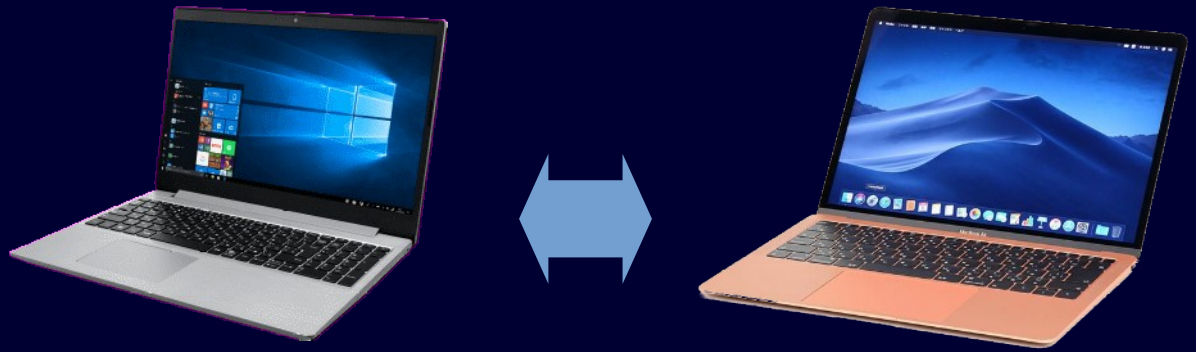
(A) Common high-level API surface (SOA)

(B) Common low-level API surface (HAL)

Dreaming fully portable SW, is that possible?

(A) Common high-level API surface (SOA)

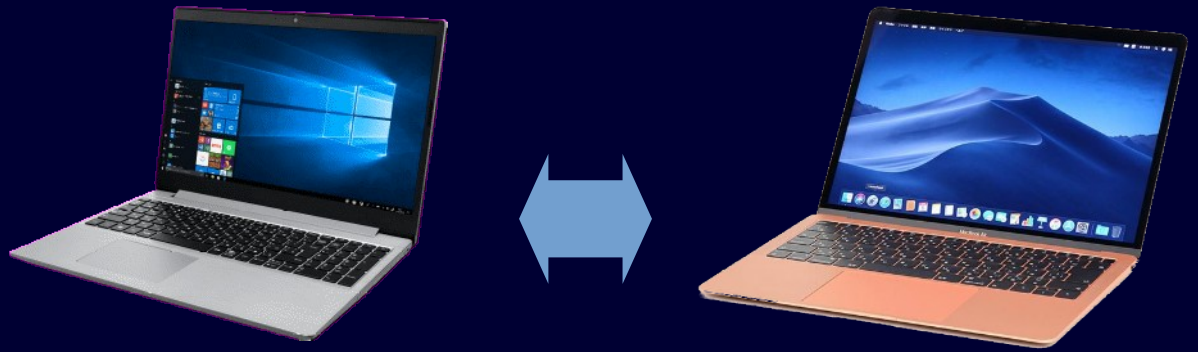
(B) Common low-level API surface (HAL)

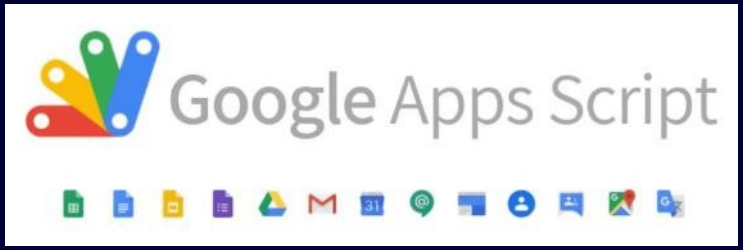


Dreaming fully portable SW, is that possible?

(A) Common high-level API surface (SOA)

(B) Common low-level API surface (HAL)





Dreaming fully portable SW, is that possible?

(A) Common high-level API surface (SOA)

(B) Common low-level API surface (HAL)

Dreaming fully portable SW, is that possible?

(A) Common high-level API surface (SOA)

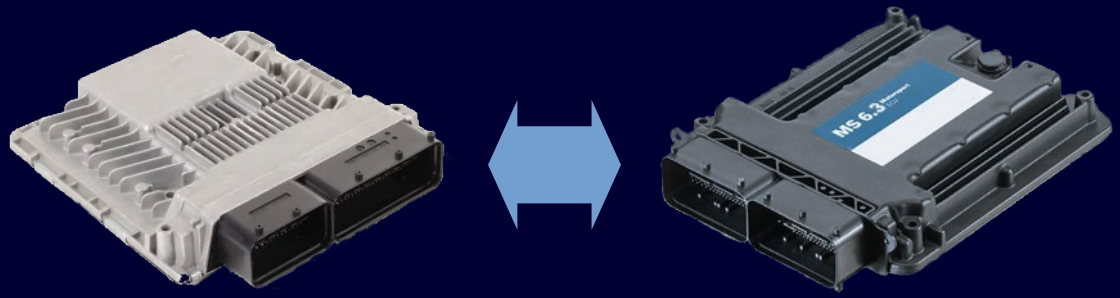
(B) Common low-level API surface (HAL)

https://www.windowscentral.com/windows-10-project-latte-android-apps

Dreaming fully portable SW, is that possible?

(A) Common high-level API surface (SOA)

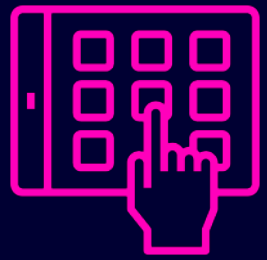
(B) Common low-level API surface (HAL)



Dreaming fully portable SW, is that possible?

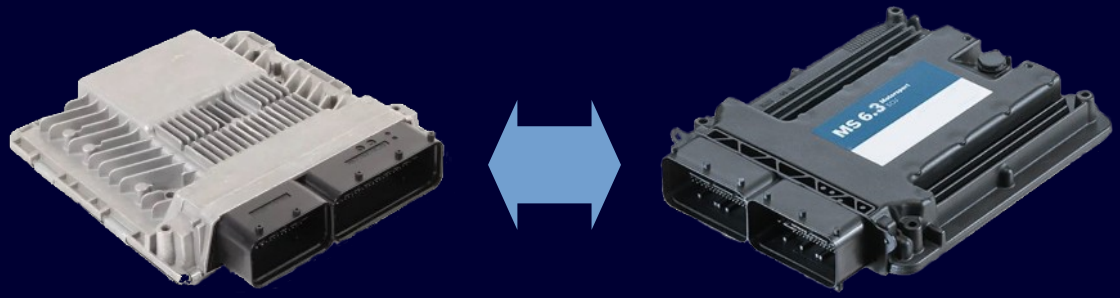
Portability / Reusability

- ECU program
- Data exchange
- Firmware update
- Logging



(A) Common high-level API surface (SOA)

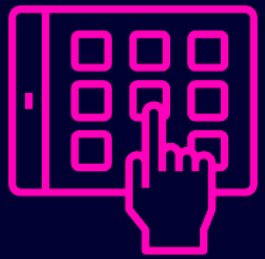
(B) Common low-level API surface (HAL)



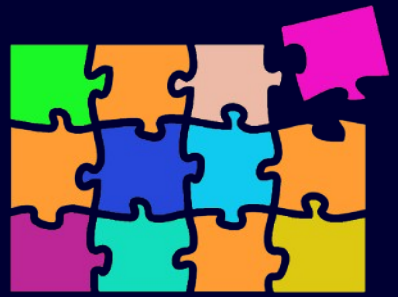
Dreaming fully portable SW, is that possible?

Portability / Reusability

- ECU program
- Data exchange
- Firmware update
- Logging



(A) Common high-level API surface (SOA)



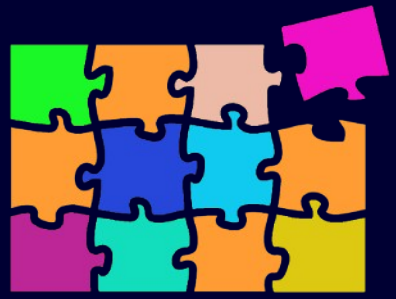
fragmentation
Embedded System
complexity

(B) Common low-level API surface (HAL)

Dreaming fully portable SW, is that possible?

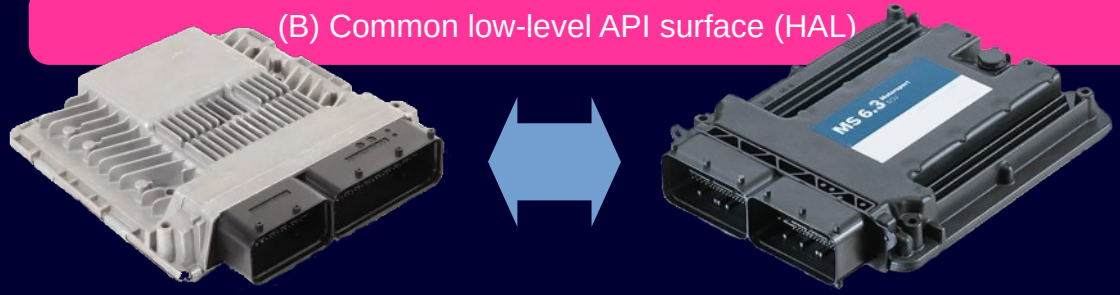
Cloud computing friendly modern IT-oriented SW development scheme might not directly applicable to highly fragmented EMBEDDED SYSTEM (control SW development & maintenance)

(A) Common high-level API surface (SOA)



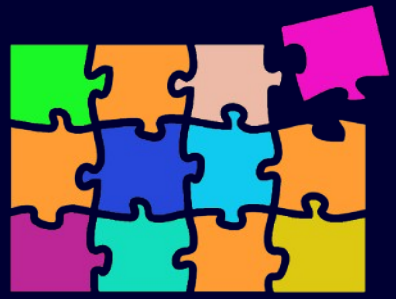
fragmentation
Embedded System
complexity

(B) Common low-level API surface (HAL)



Dreaming fully portable SW, is that possible?

(A) Common high-level API surface (SOA)



fragmentation
Embedded System
complexity

AUTOMOTIVE GRADE LINUX

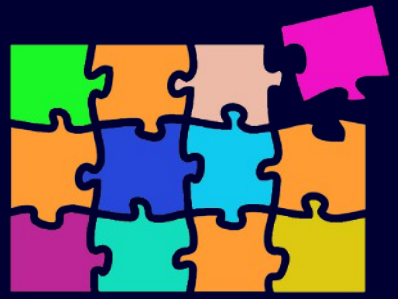
(B) Common low-level API surface (HAL)

Dreaming fully portable SW, is that possible?

IT friendly SW method

- cloud resource utilization
- JS, Python, Kotlin, Dart,..
- dev-ops(CI/CD), test automation

(A) Common high-level API surface (SOA)



fragmentation
Embedded System
complexity

AUTOMOTIVE GRADE LINUX

(B) Common low-level API surface (HAL)

Embedded SW method

- various resource constraint
- C/C++, Clang, Rust,...
- HW abstraction (virtualization)

Now, AGL needs to chase two directions

(B) Common low-level API surface (HAL)

- HW coverage
  - Add new SoC native support
  - Penetrate AGL reference HW platform
- Virtualization support
  - Hypervisor integration (ideally OSS option)
  - VirtIO enhancement and host OS support
- Documentation
  - User friendly step-by-step guidance
  - Pre-build binary AGL environment release

(A) Common high-level API surface (SOA)

- Secure vehicle application portability
  - Provide micro-service delivery scheme
  - Integrate container support
- Security & privacy support
  - FUSA compliance (system isolation)
  - Vulnerability resolution (patch management)
- Common vehicle interface
  - Work with other industry consortium
  - Bridge between automotive & IT industry