altia

Low Code/No Code Design Paradigm for Multi-OS Cockpits

Philipp Michel

Senior Solutions Architect

March 2023 – emBO++

altia

Agenda

1. What is a domain-controlled cockpit?
2. Increasing software complexity presents a problem
3. Traditional development process
4. Low code / No code solution

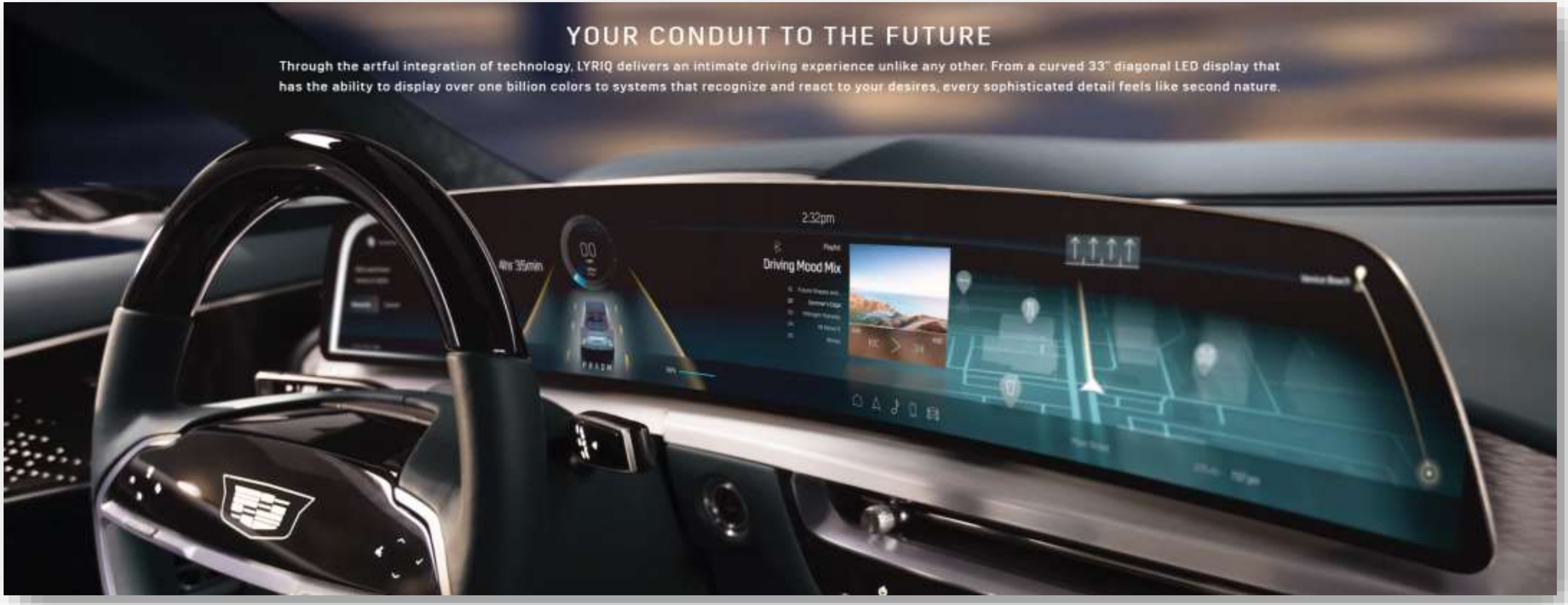
The Automotive UX

OEMs are increasingly bringing cockpit software development in house.

User experience (UX) is a very large value/differentiator for a brand.

What is a Domain Controller?

Cadillac LYRIQ



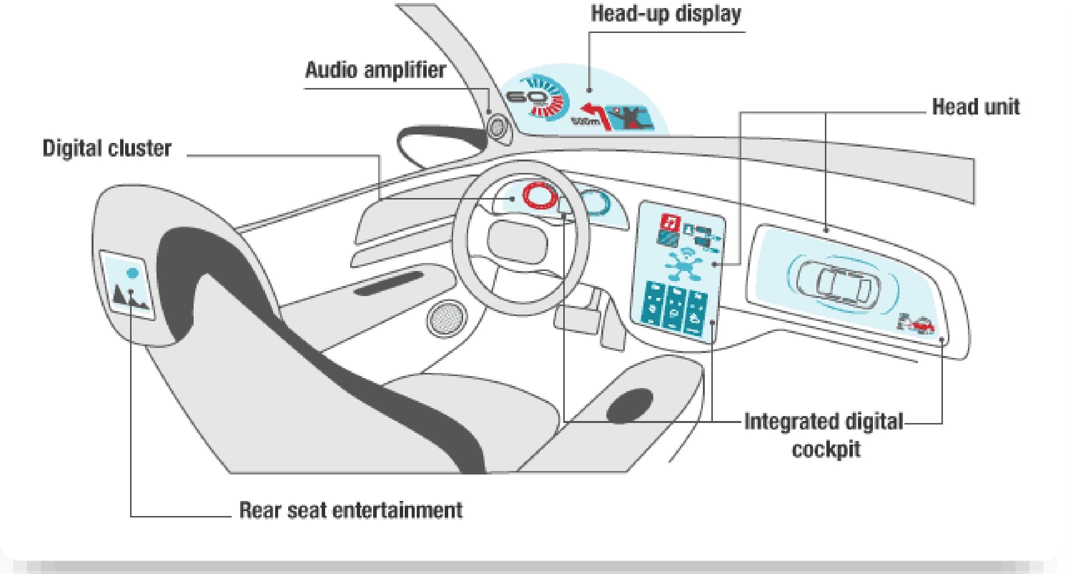
YOUR CONDUIT TO THE FUTURE

Through the artful integration of technology, LYRIQ delivers an intimate driving experience unlike any other. From a curved 33" diagonal LED display that has the ability to display over one billion colors to systems that recognize and react to your desires, every sophisticated detail feels like second nature.

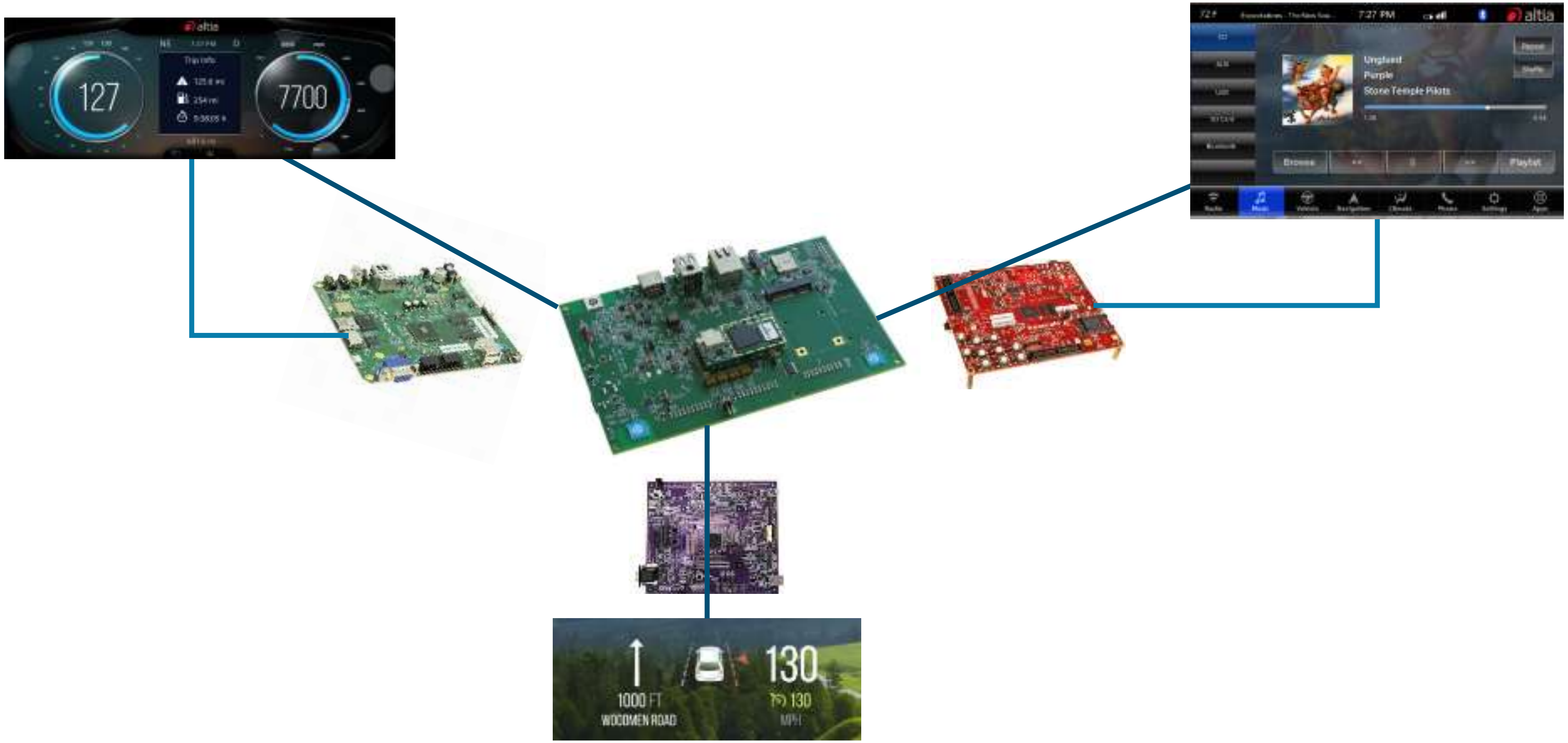
Source - https://www.cadillac.com/

What is a domain-controlled cockpit?

- **Single SoC drives multiple displays**
  - Cluster
  - HUD
  - IVI
  - Passenger
  - RSE
  - Climate
  - Mirror Replacement
  - Etc.
- **Graphical content is synced and shared between displays.**
- **Why?**

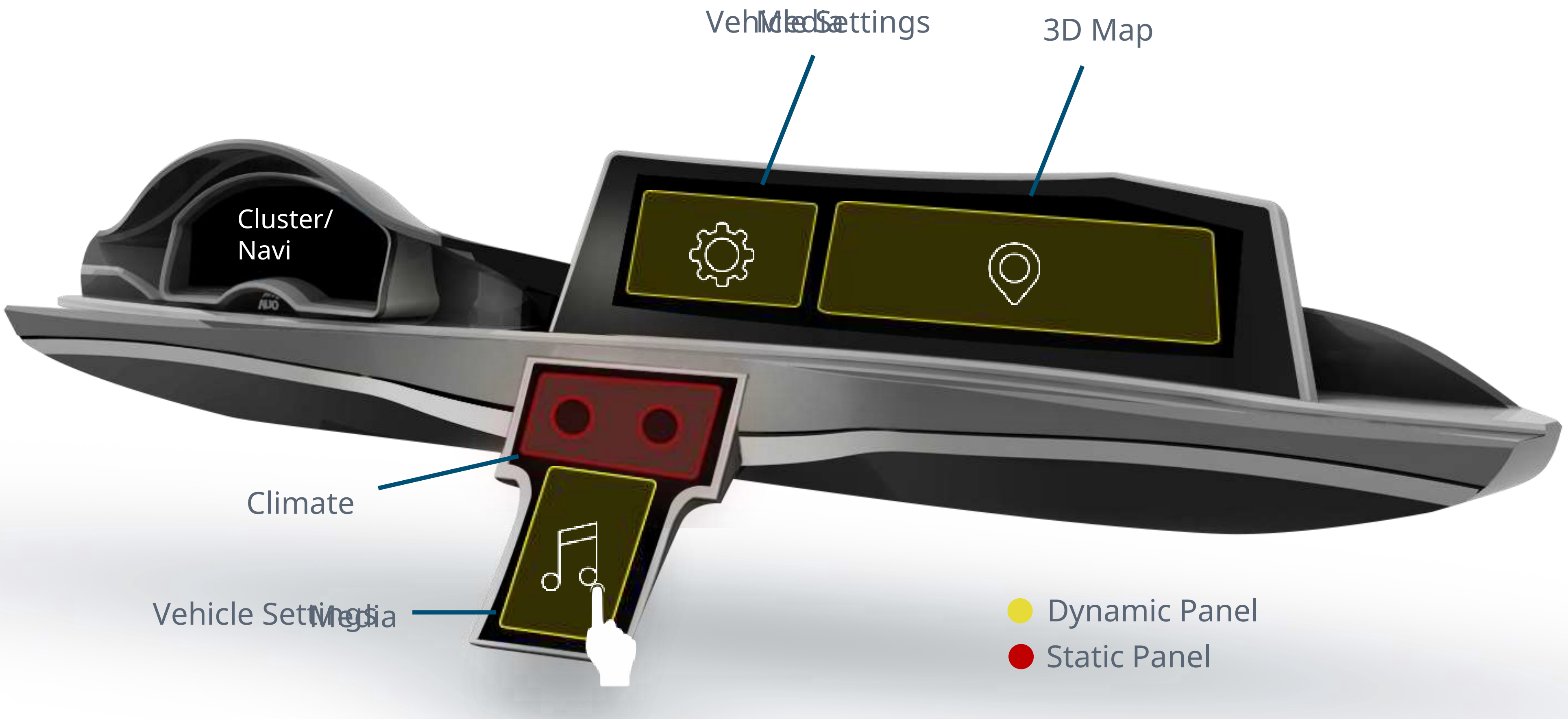


Hardware Consolidation



Interaction Design

altia



Altia: Integrated Cockpit Domain Controller Demo

Cluster
Altia
Simulink
Altia Safety Monitor
RTOS

Passenger Entertainment
Altia
Simulink
Android
RTOS

Requirements – Infotainment Displays

altia

Connectivity

Connected OS

Third Party Apps

Requirements – Infotainment Displays: Content Sharing and Syncing

Jeep Grand Cherokee

Requirements – Cluster Displays

Cluster
Altia
Simulink
Altia Safety Monitor
RTOS

Passenger Entertainment
Altia
Simulink
Android
RTOS

Requirements – Cluster/Functional Displays

altia

Requirements of Infotainment Display, as well as...

Software Robustness

MISRA

ISO ASPICE

Functional Safety

P R N D

ISO 26262

NHTSA and other Regulatory Requirements

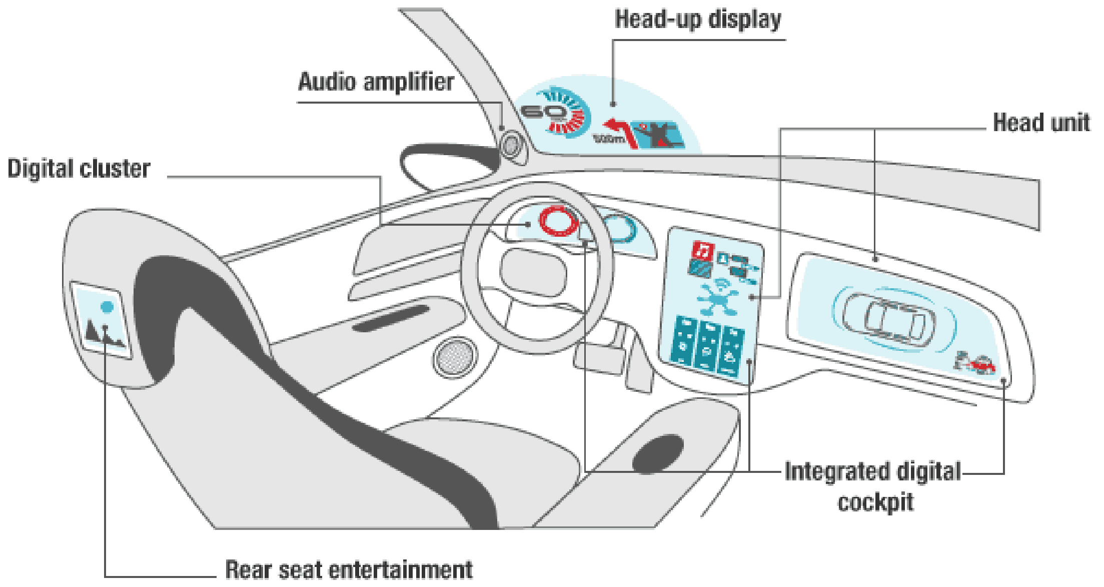
Cyber Security

ISO/SAE 21434

ISO SAE INTERNATIONAL

What is a Domain Controller cockpit?

- **Single SoC drives multiple displays**
  - Cluster, HUD, IVI, Passenger, RSE, Climate, Mirror Replacement, etc.
- **Graphical content is synced and shared between displays**
- **Why?**
  - Cost
  - Unified UX
- **Other considerations**
  - Multiple OSs/partitions
  - Multiple graphics pipelines
  - Functional safety content

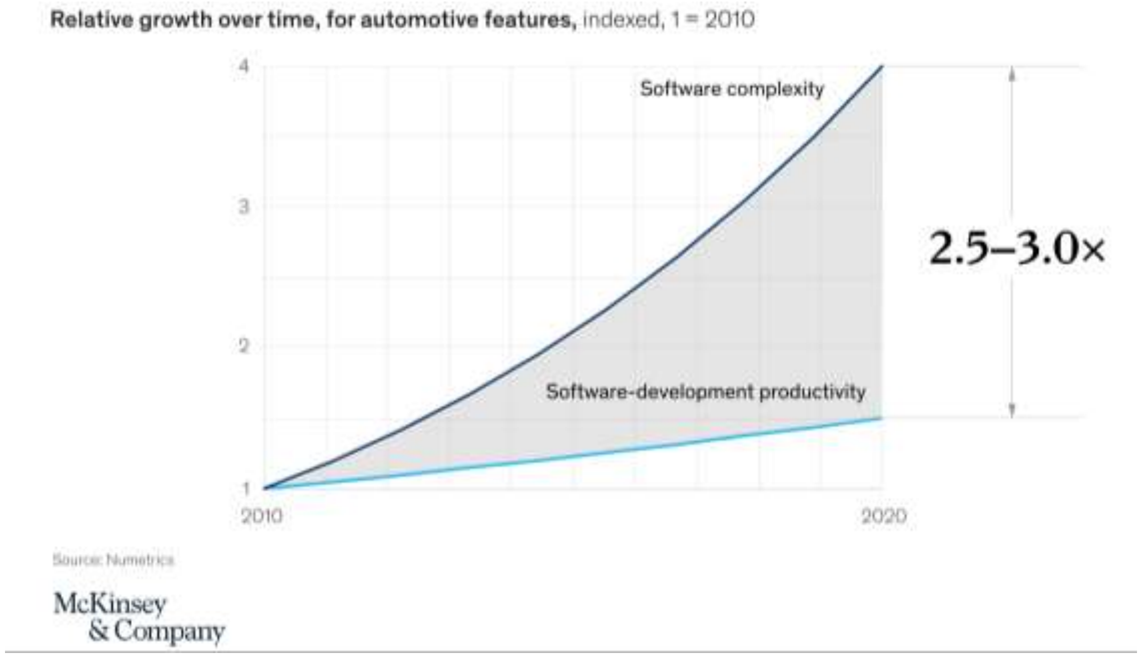


Problem

Embedded and GUI engineers must produce more capability in less time than ever before.

Software Complexity on the Rise

Growth in software complexity more than doubles the growth in software development productivity.



Relative growth over time, for automotive features, indexed, 1 = 2010

Source: Numetrics

McKinsey & Company

Automotive Challenges for Domain Controller Cockpit

altia

Security

Connectivity

Safety

3rd Party Apps

Multi-OS

Non-Traditional Knowledge is Required for Software Development

3D Graphics

- Materials
- Shaders
- File Formats
- Texture Compression

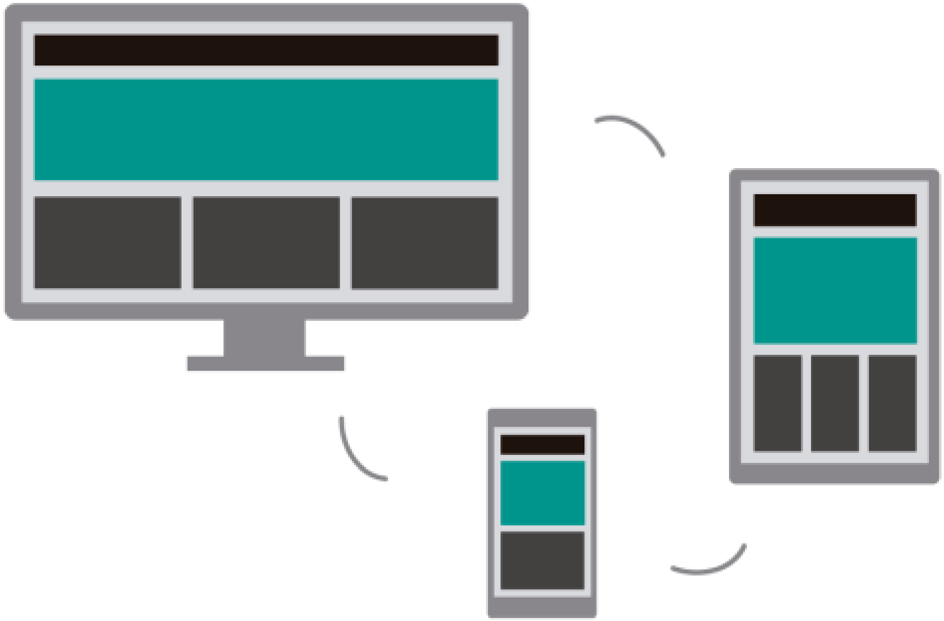
56 MPH
PRND
40
30
20
10
FT
KWH 5.8
297 MI

Non-Traditional Knowledge is Required for Software Development

Advanced Layout

- Style guides (ex. CSS, XML)
- Layout Constraints

  (ex. Flexbox, Android)



Non-Traditional Knowledge is Required for Software Development

- Video Streaming
- Night Vision
- 3D Displays
- HUD Warping
- Etc.

Jeep Grand Wagoneer

Embedded software and GUI engineers are some of the most difficult to find in the workforce, but they are being given more tasks unrelated to their specialties in the field—effectively diluting the efficacy of each engineer.

Traditional Development Method – Fractured Process

Design

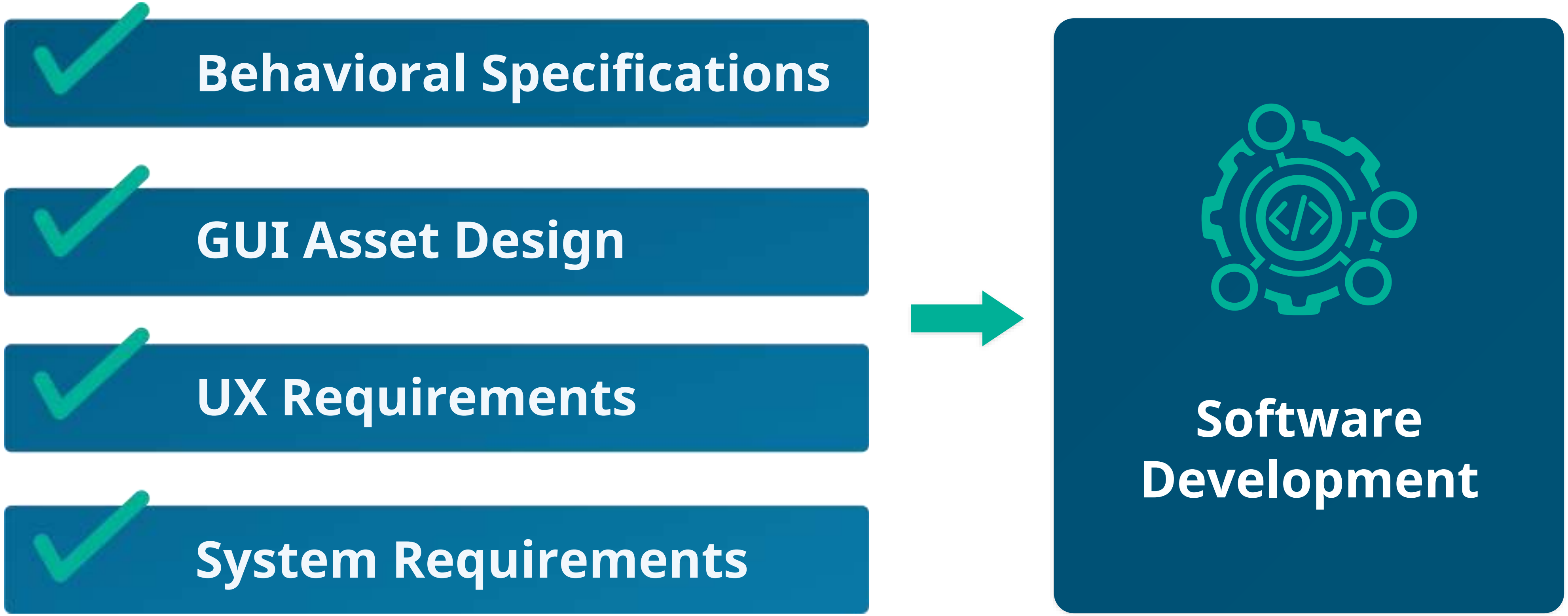
Behavior

System Requirements

Quality Standards

Quality Processes

Traditional Development Process – Serial Process



✓ Behavioral Specifications

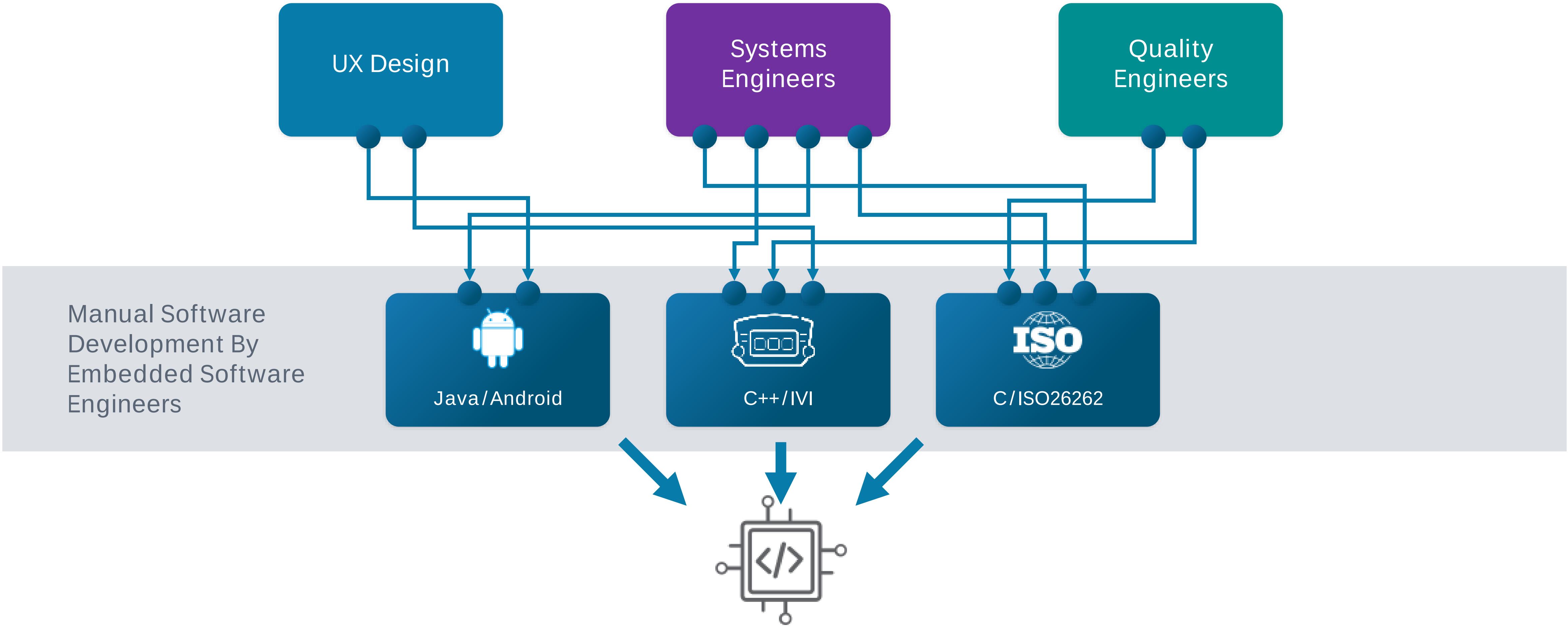
✓ GUI Asset Design

✓ UX Requirements

✓ System Requirements

Software Development

Traditional Domain Controller Development Process

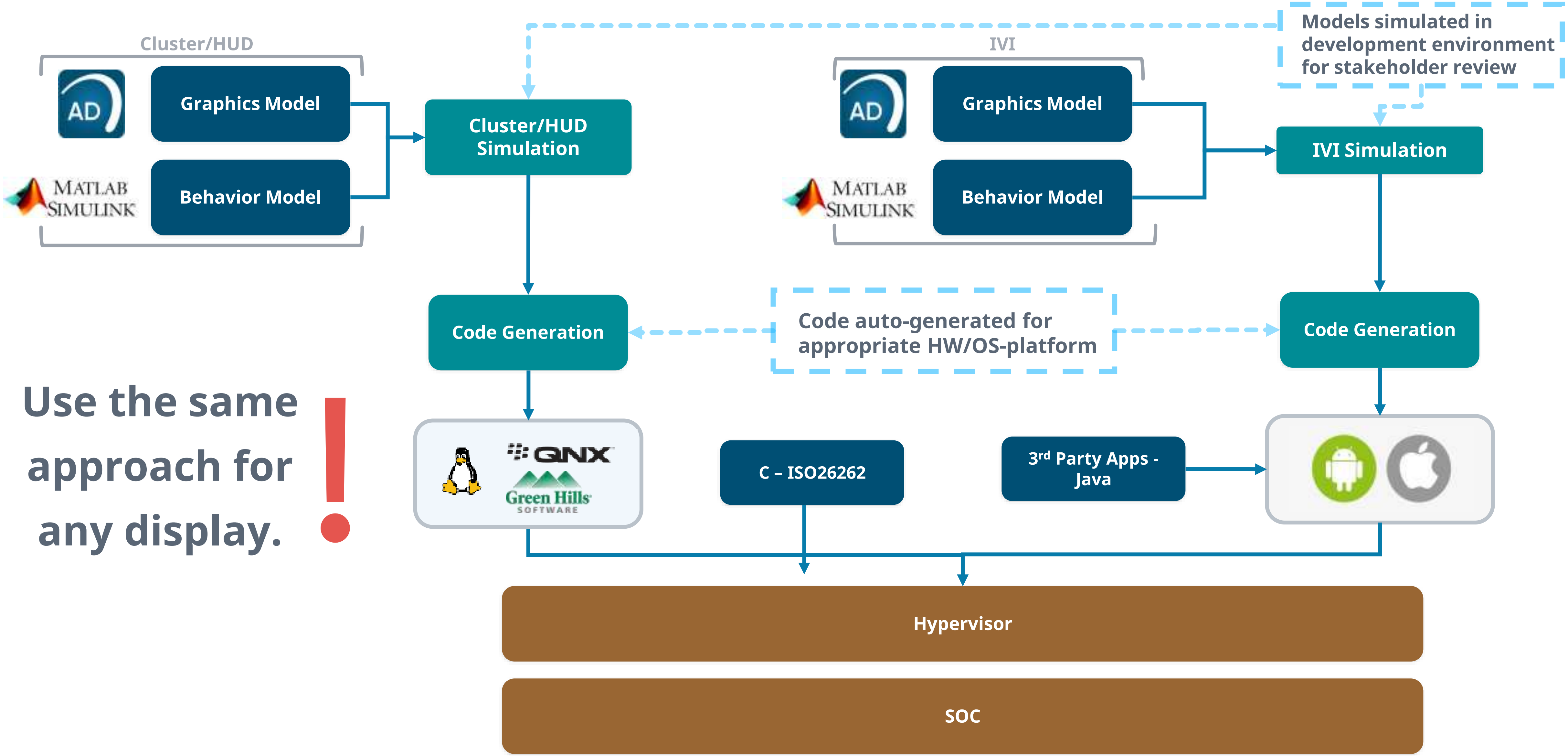


Solution

Share the load.
Decrease the code.

By getting smarter about the tools used by development teams, companies can avoid putting impossible loads on the software and GUI teams. Instead, they can increase productivity and quality simultaneously.

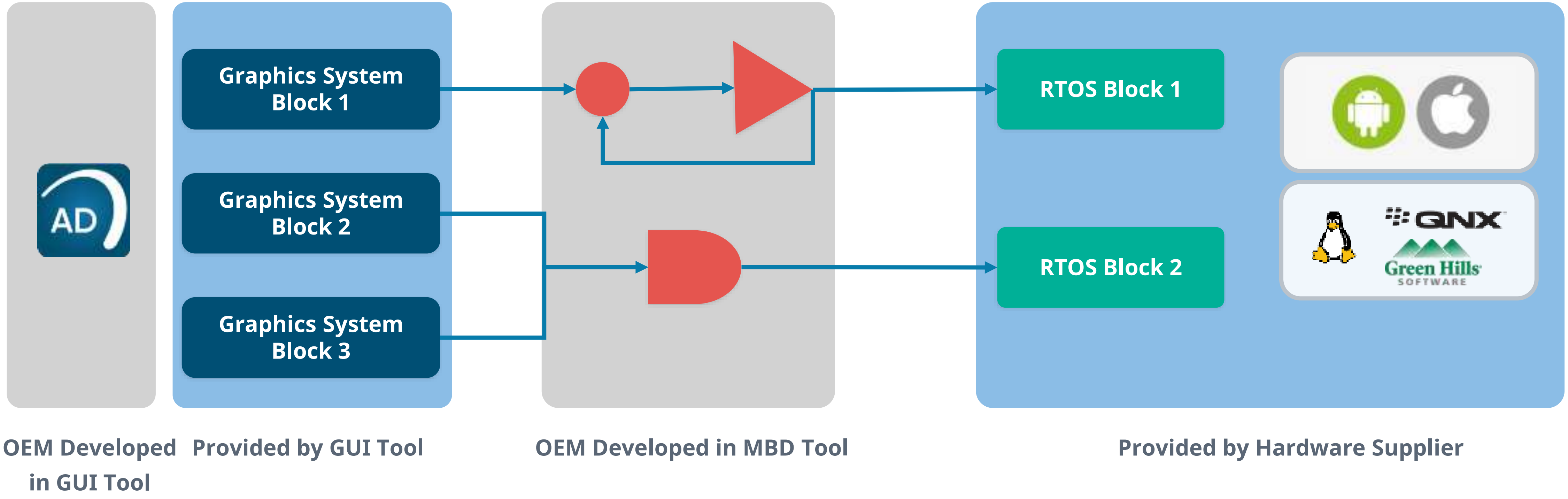
Common Development Methodology



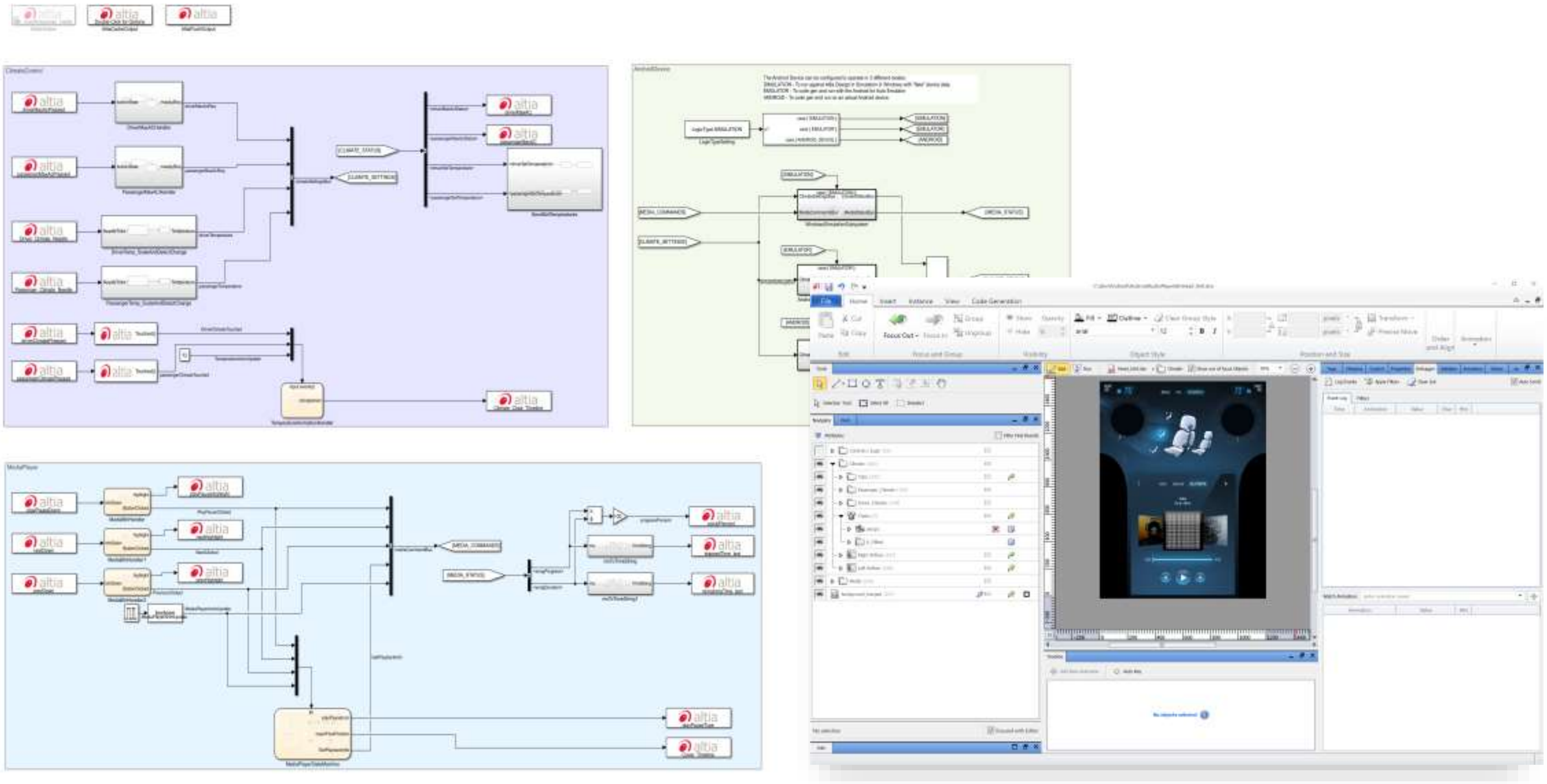
! Use the same approach for any display. !

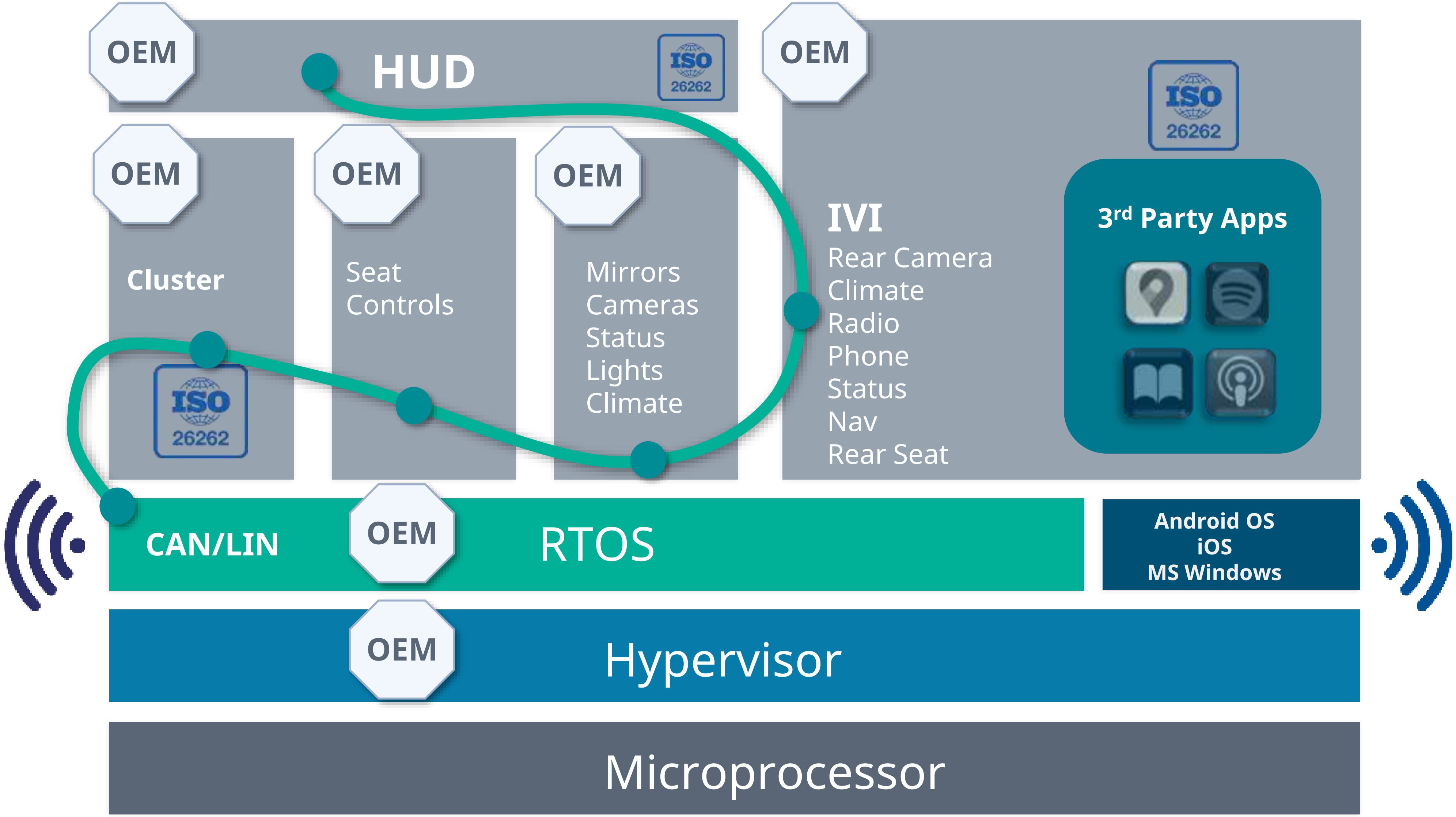
Model-Based Behavior Detail

- MBD System Abstracts Interfaces to GUI and OS
- OEM-required knowledge is abstracted from hardware/operating system and consistent across the entire cockpit.



Simulink Model




OEM
HUD
ISO 26262
OEM
ISO 26262
OEM
OEM
OEM
Cluster
Seat
Controls
Mirrors
Cameras
Status
Lights
Climate
IVI
Rear Camera
Climate
Radio
Phone
Status
Nav
Rear Seat
3rd Party Apps
ISO 26262
CAN/LIN
OEM
RTOS
Android OS
iOS
MS Windows
OEM
Hypervisor
Microprocessor

Model-Based *Development* Benefits

- **Engaging the entire team** in the development process
- **No extra mock-ups and prototypes**
- Extra team support **removes burden from embedded software resources**
- **Embedded engineers can focus** on confirming that the software interacts with electronics hardware
- **Deploy to Android or iOS** and take advantage of connected services
- **Simultaneously deploy to RTOS** environments for mandated and functional safety requirements

Model-Based *Business* Benefits

- **Same design/development paradigm** for ALL OS/HW configurations
- **Less training, fewer tool costs**
- **Lower development cost** due to higher efficiency and elimination of unnecessary steps
- **Faster time to market**—the whole team is contributing to development
- Higher **product quality** and **market penetration**

altia

Thank you!

www.altia.com

info@altia.com
philippm@altia.com