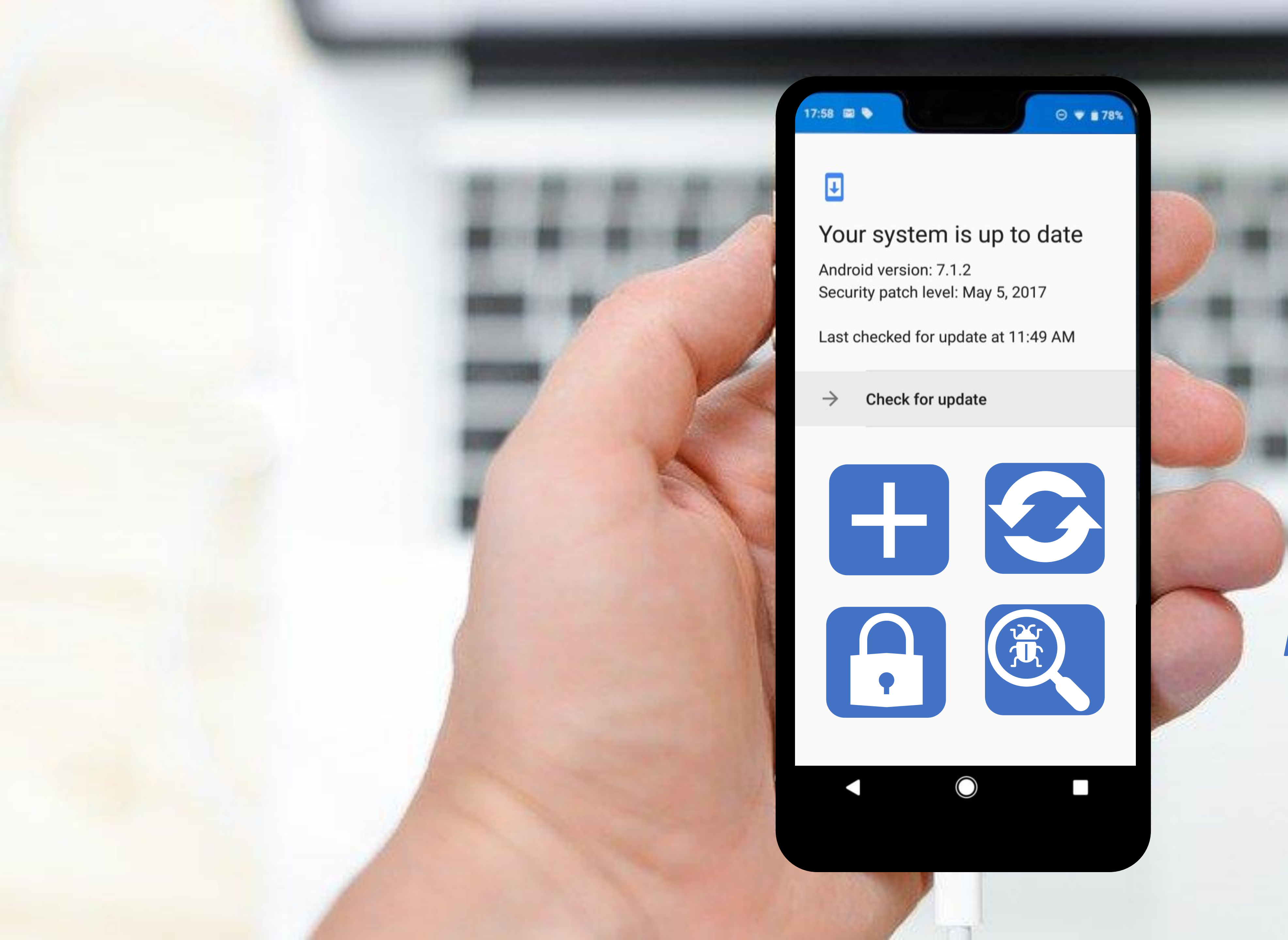
SAFE MODULAR ONLINE UPDATES AND UPGRADES FOR MIXED-CRITICALITY SYSTEMS

Gregor Nitsche, Patrick Uven, Ingo Stierand, Kim Grüttner

31. SafeTRANS Industrial Day, 28.11.2022 Berlin

DLR



Your system is up to date

Android version: 7.1.2
Security patch level: May 5, 2017

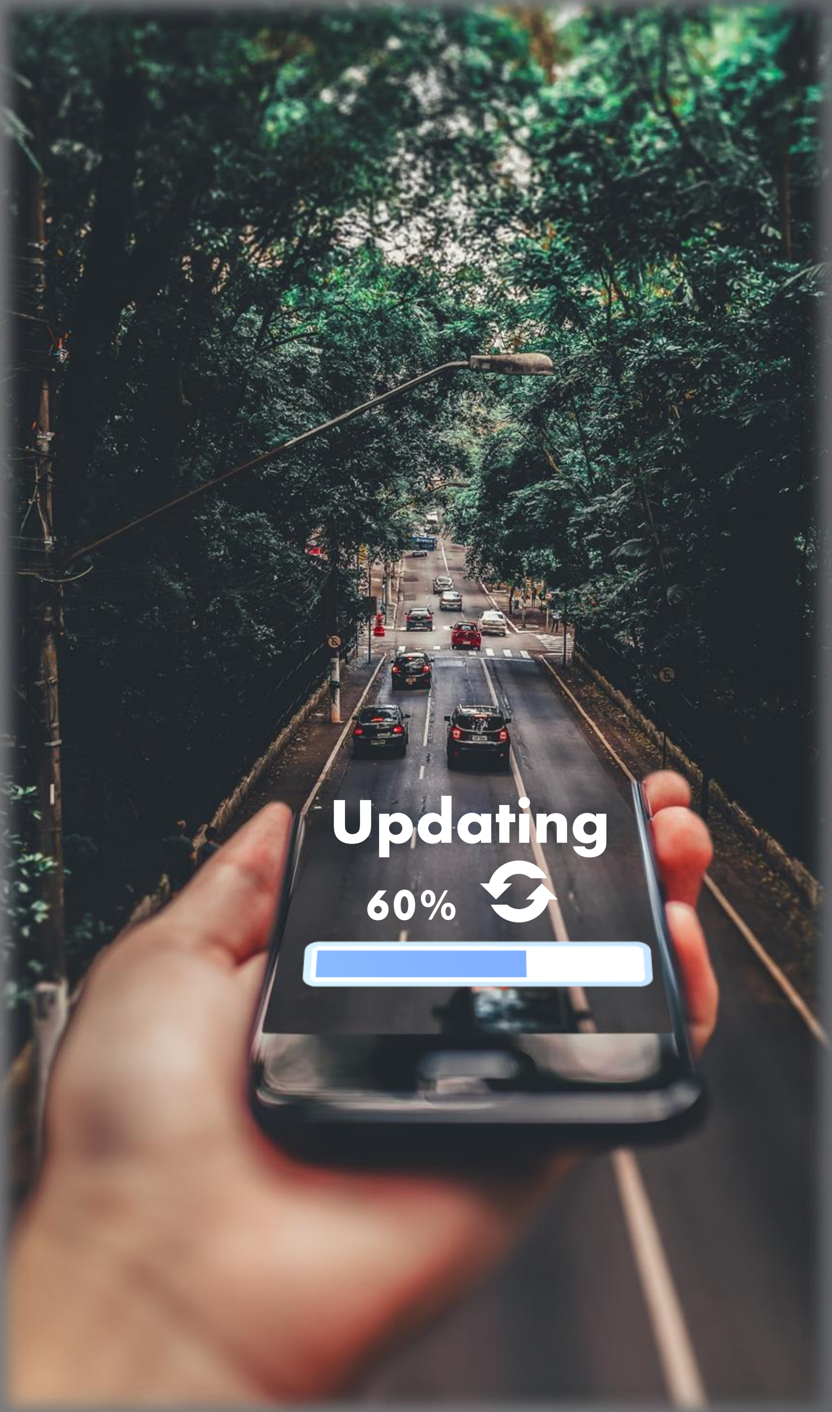
Last checked for update at 11:49 AM

New functionalities

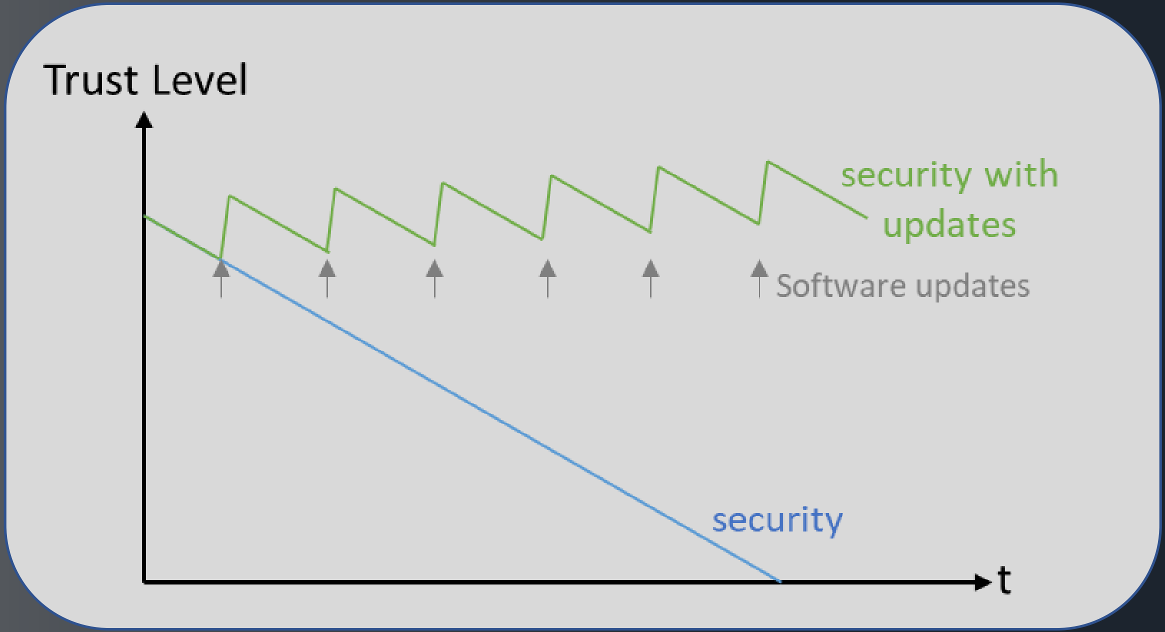
Latest version

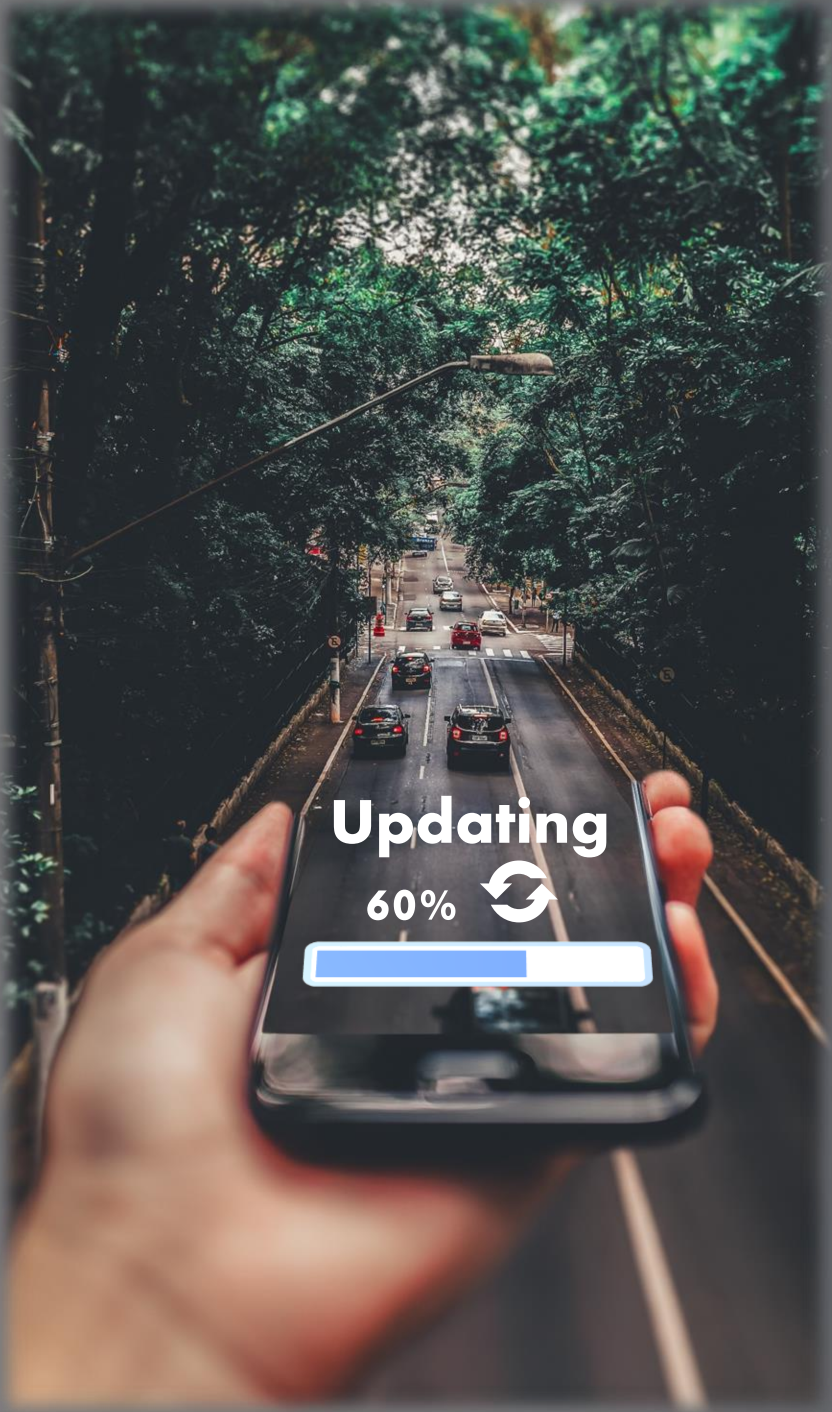
Security patches

Bug fixing

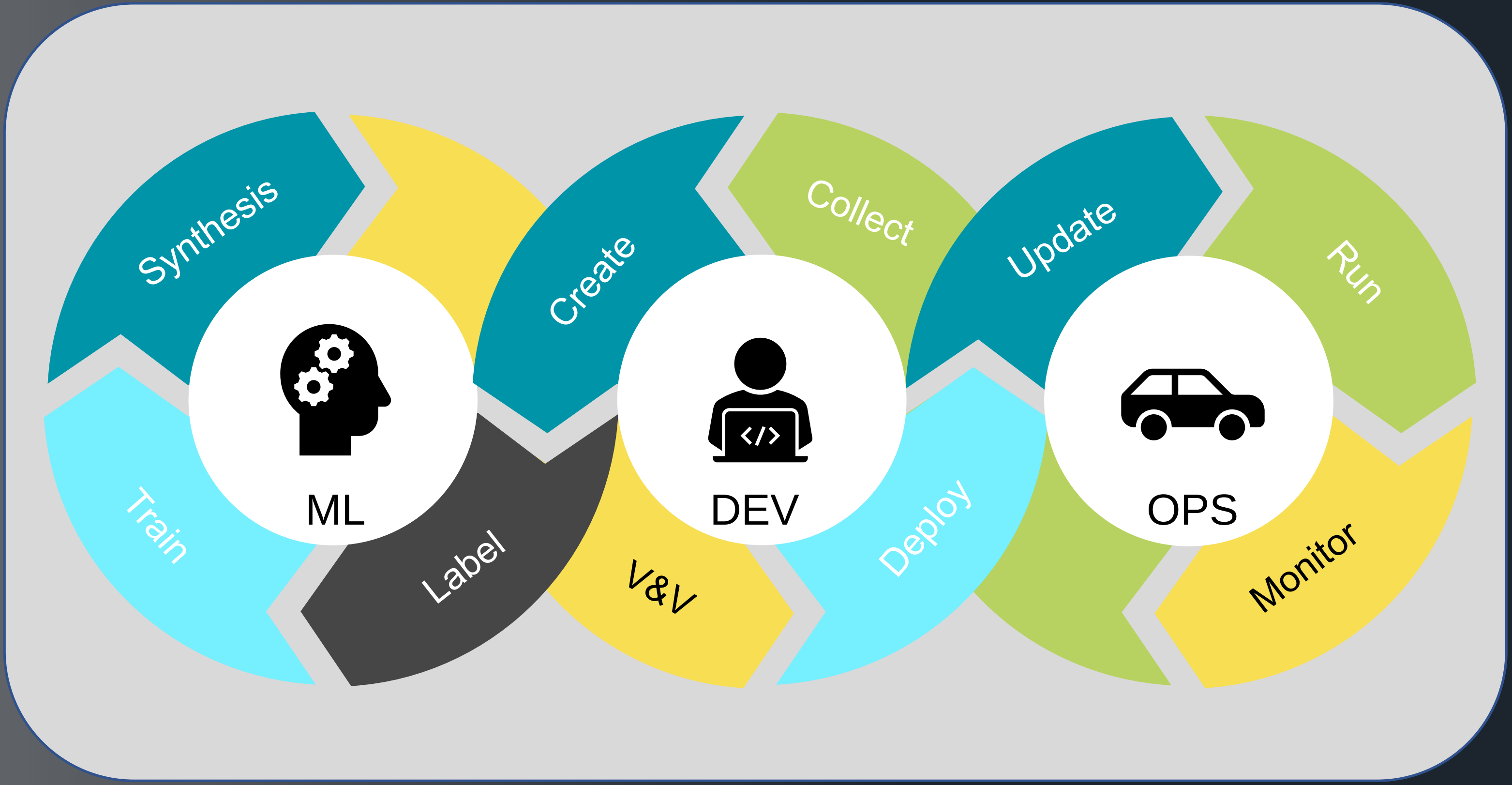


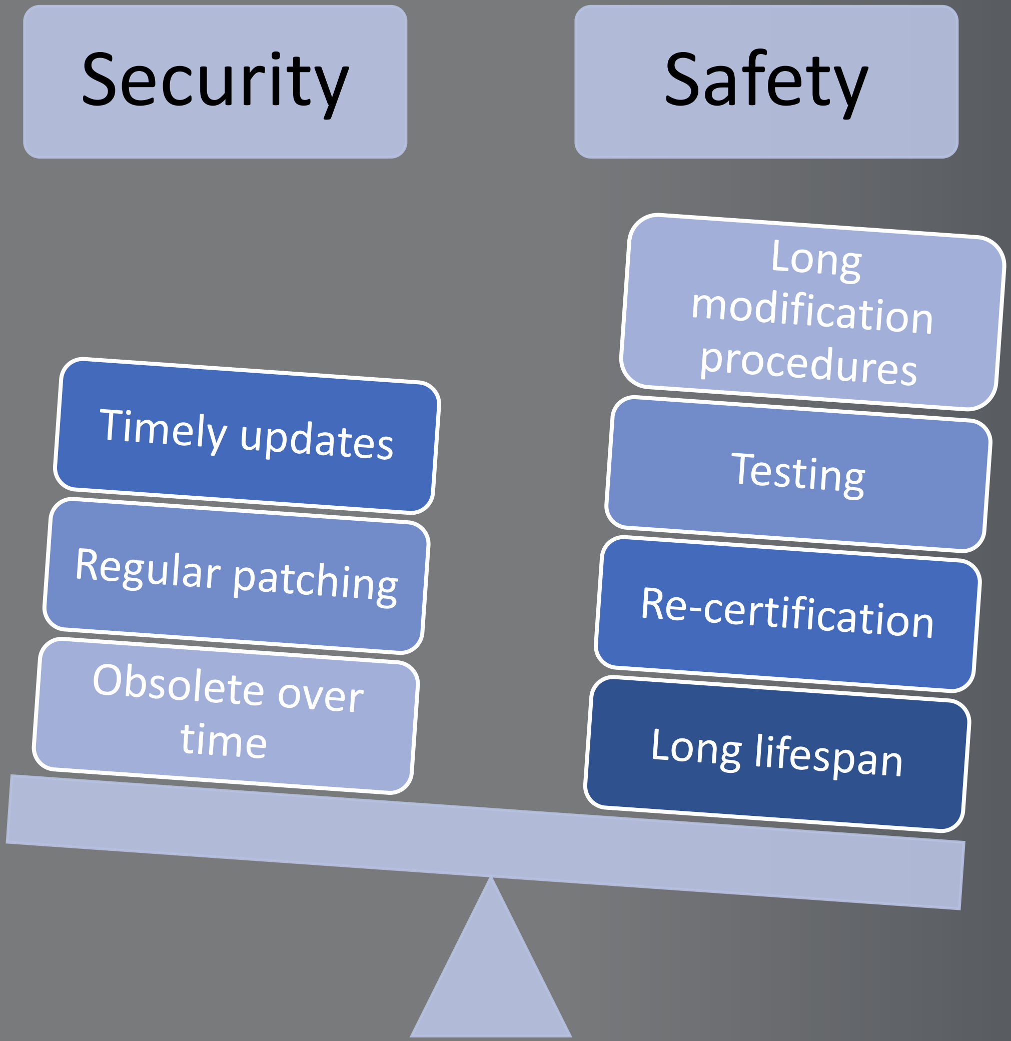
- Connected devices are vulnerable to cyber-attacks
- Security mechanisms become obsolete over time
  - New vulnerabilities disclosed every day
- Updates are crucial to guarantee security (patching)





- Machine Learning is becoming more important and thus the integration with a feedback and update process though MLOps





- **Security demands frequent / critical updates**
  - **Over-the-air (OTA) updates**
- **Functional Safety and OTA updates**
  - **Safety lifecycle (V-model) for critical SW development**
  - **Trust level increases with service time**
  - **Modifications are discouraged**
    - **Standards require an impact analysis, new safety validation, re-certification**

Challenges

End-to-end Security

Challenges

End-to-end Security

Safety

Challenges

End-to-end Security

Safety

HW/SW complexity

SOLUTIONS AND OBJECTIVES

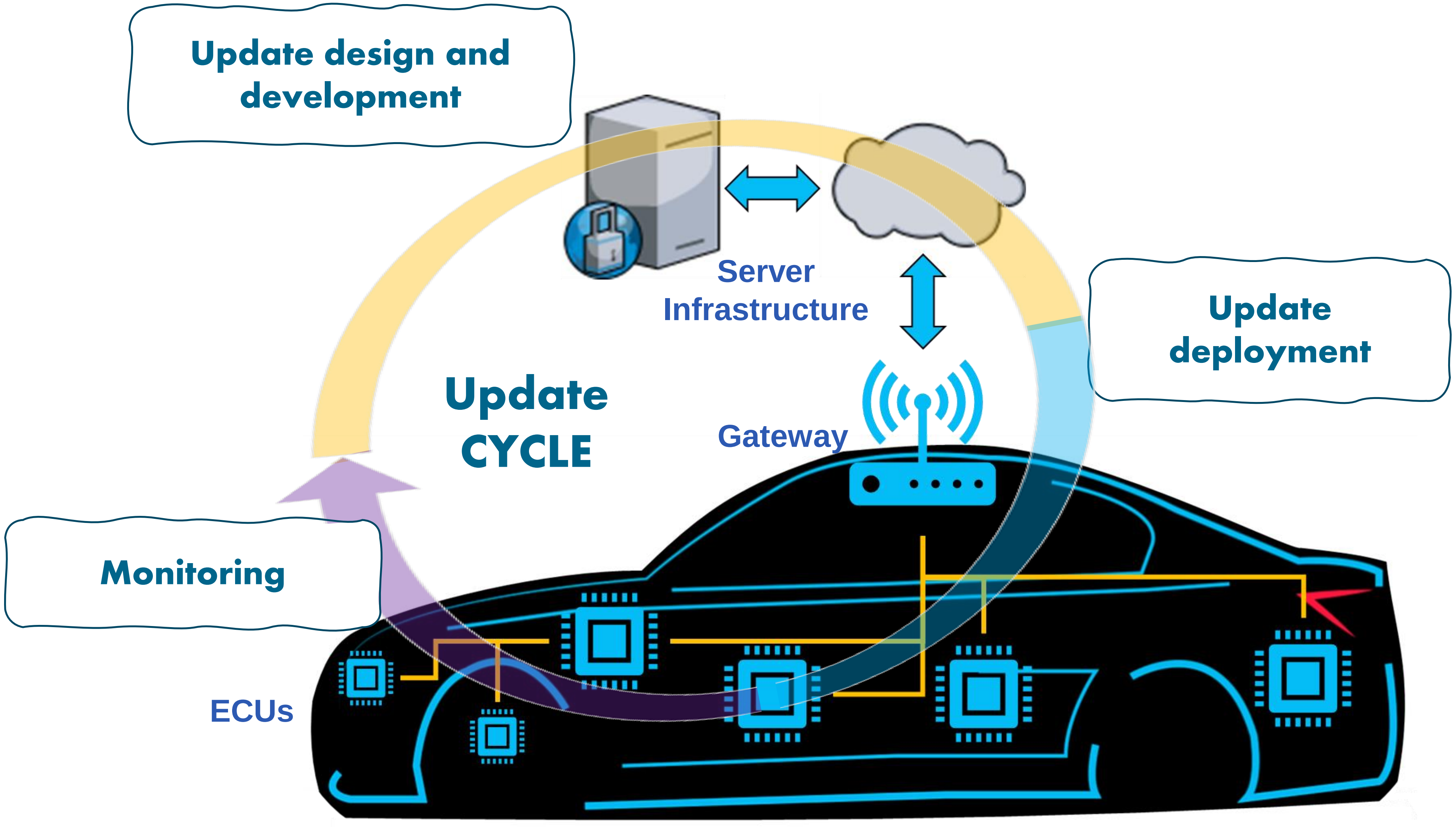
UP2DATE

Safe and secure update framework

Contract-based design

Observability and Controllability

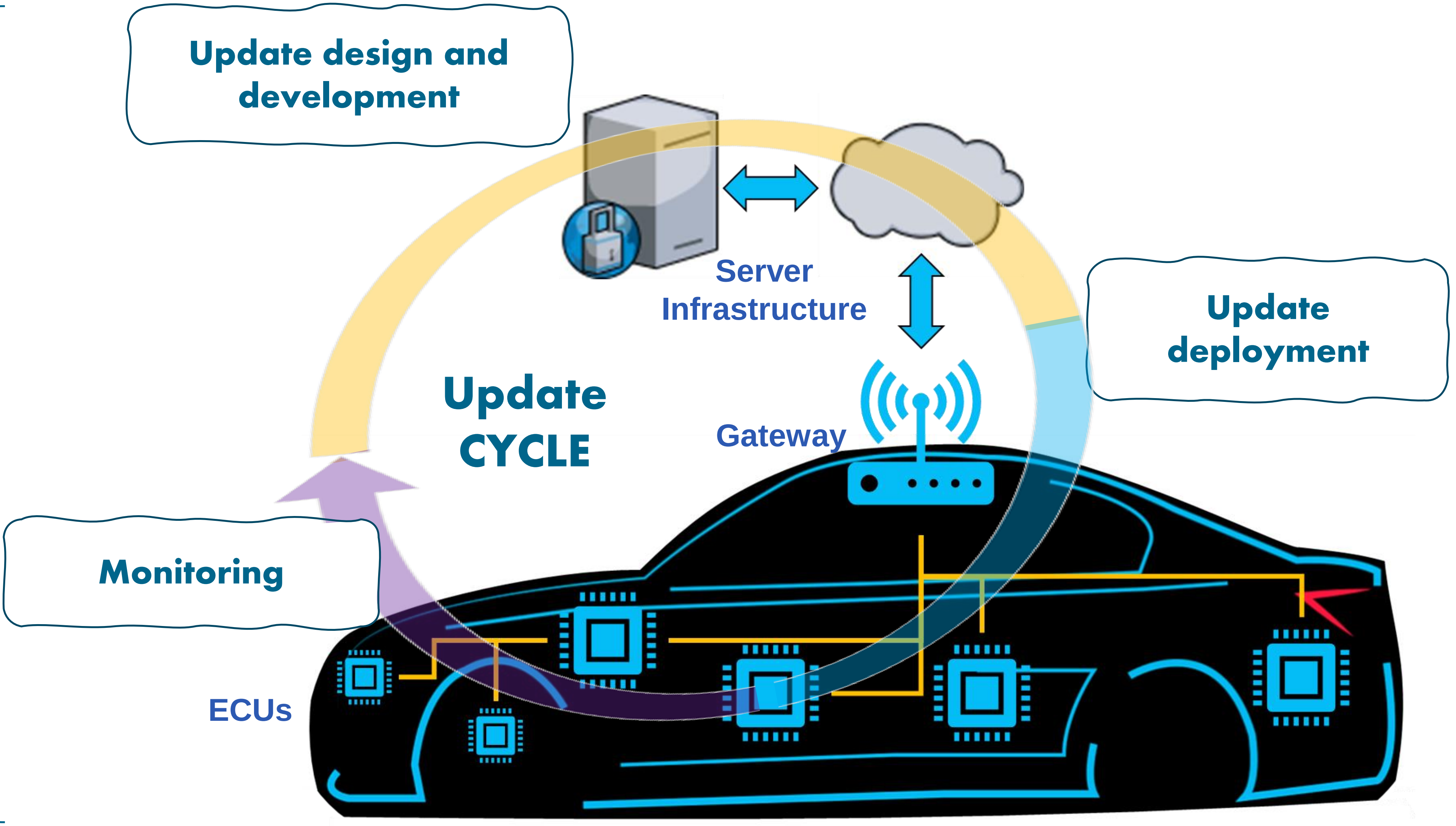
Solution: A holistic OTASU paradigm



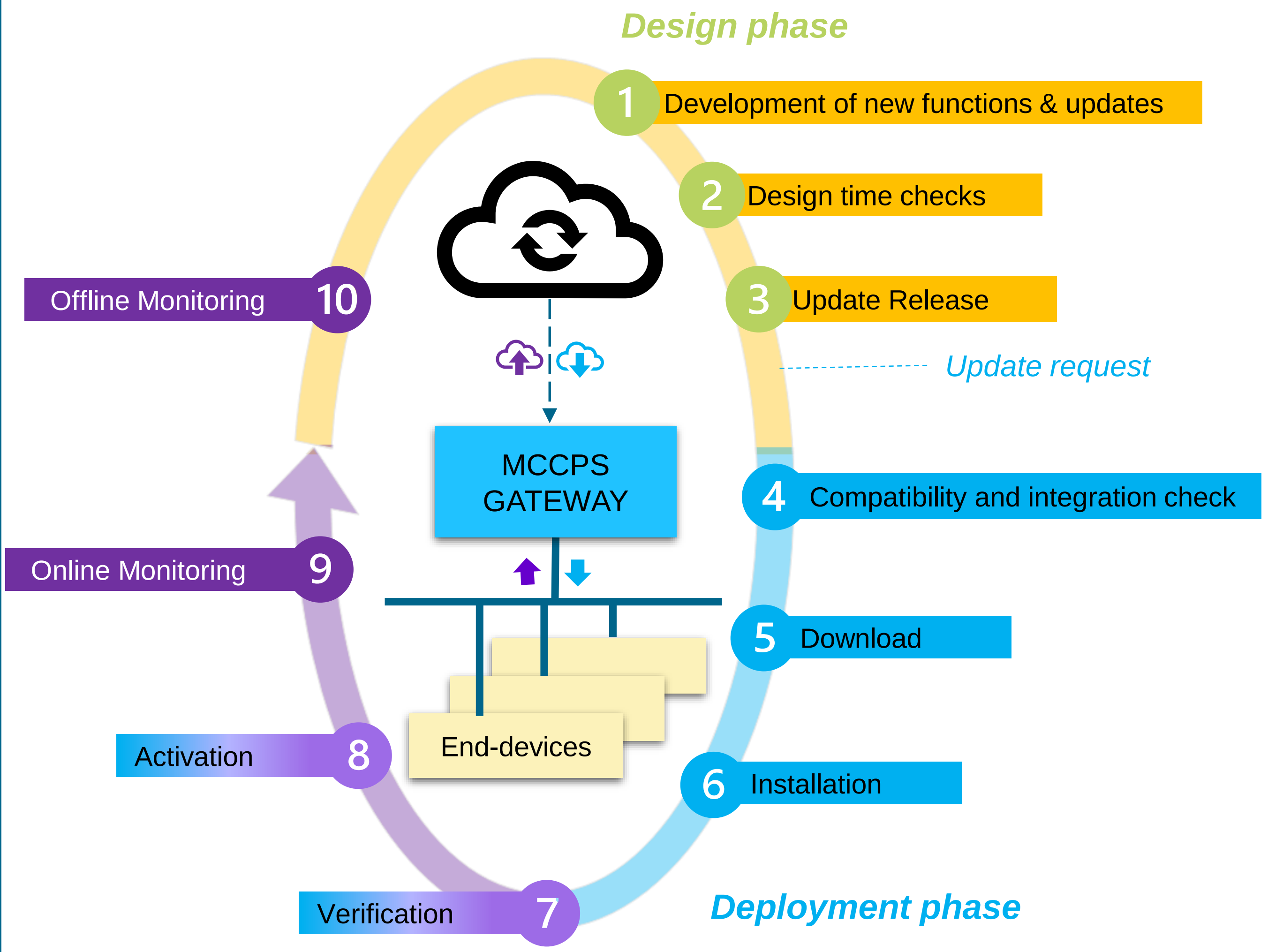
Solution: A holistic OTASU paradigm

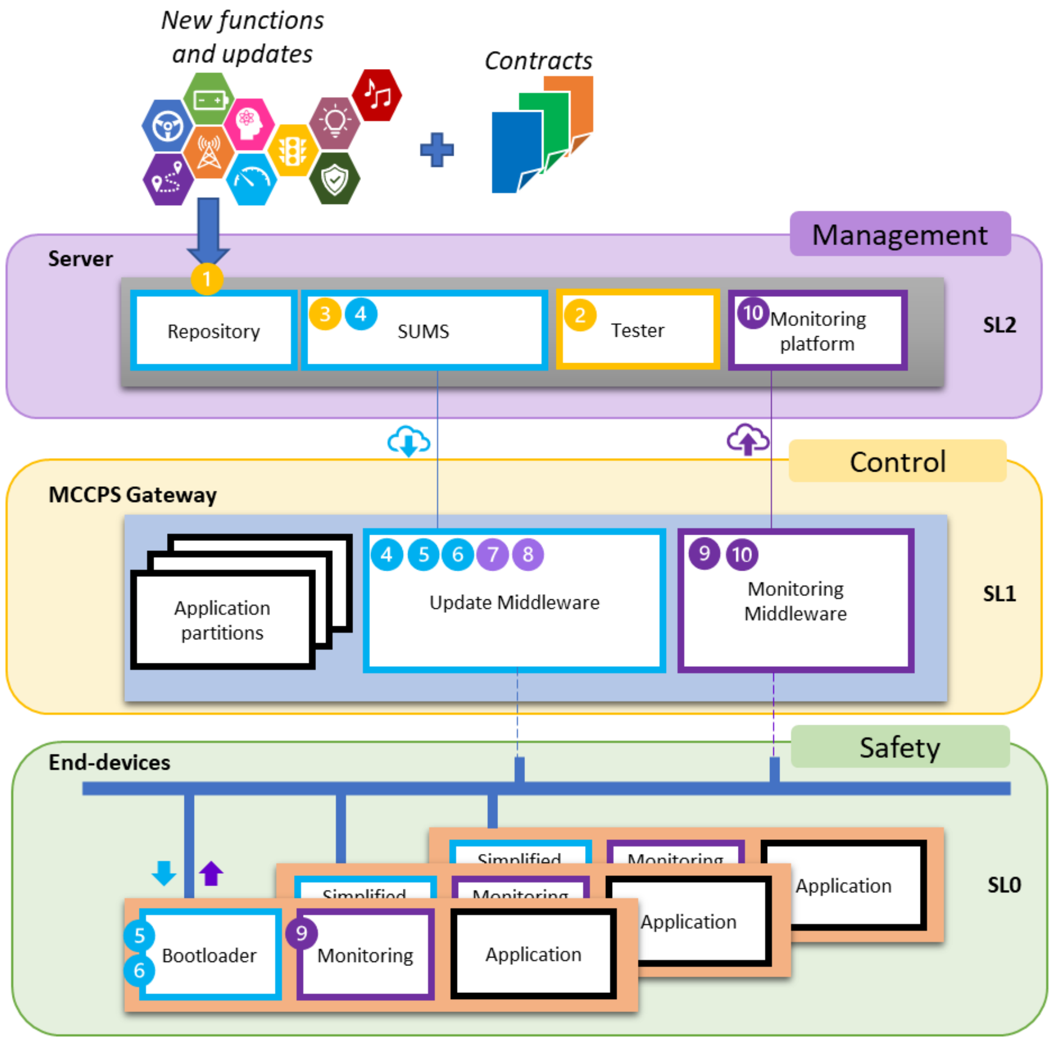
Formalisms, Processes, and Guidelines for:

- SW-Updates
- updateable HW-/SW-Platforms
- Dev-Phase, Ops-Phase, & Deploy-ment
- the Server-Fleet-SoS



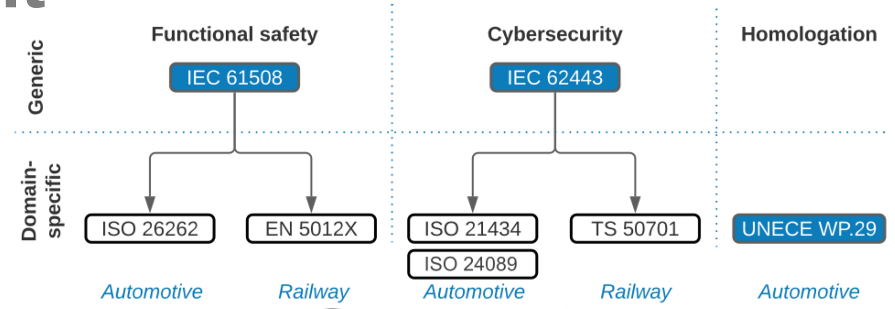
Solution: A holistic OTASU paradigm

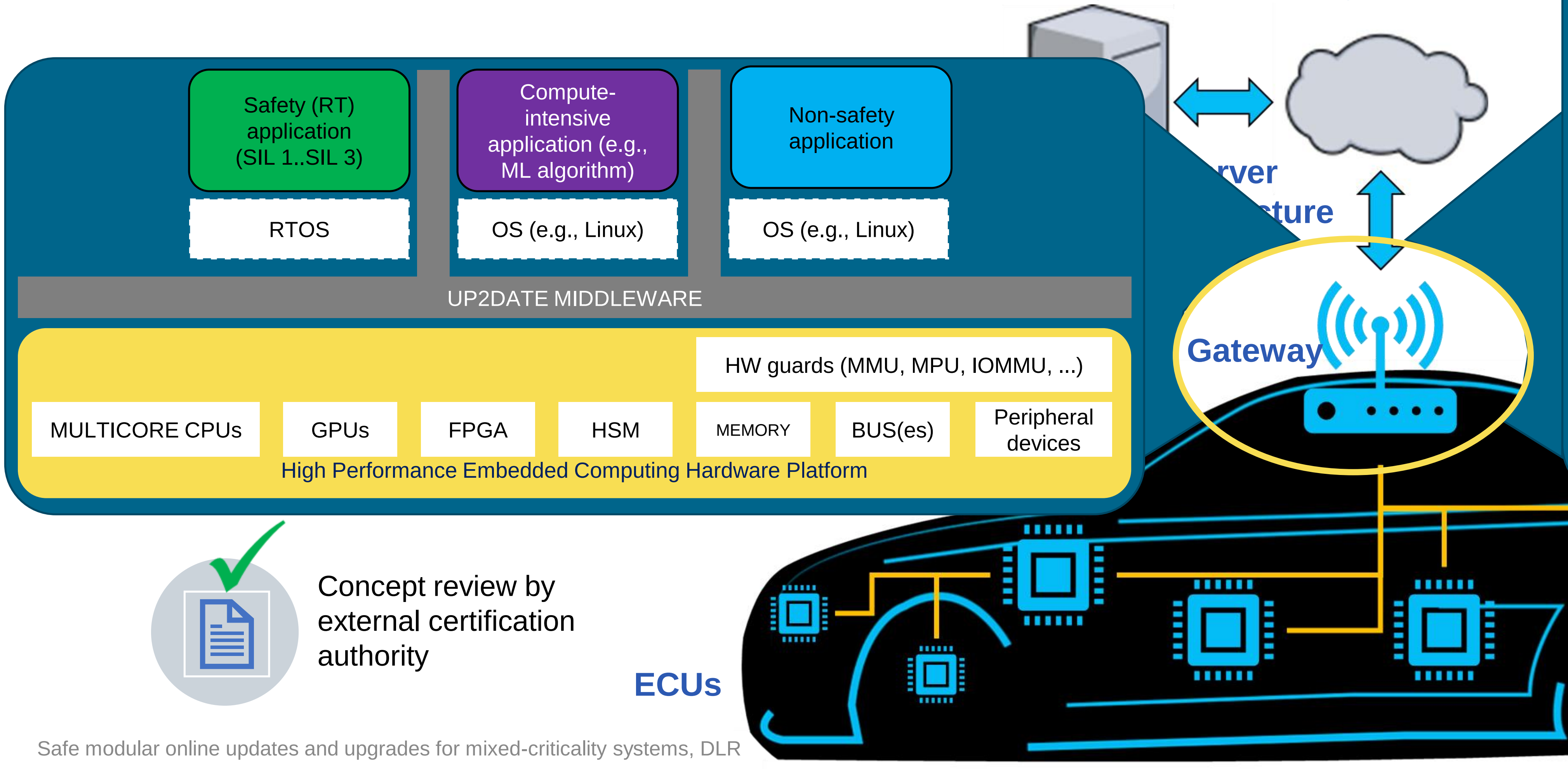




Safety & Security Assessment

- Safe and secure update management procedure
- Safe and secure requirements and mixed-criticality architecture design
- Risk assessment





Safety (RT) application (SIL 1..SIL 3)
Compute-intensive application (e.g., ML algorithm)
Non-safety application
RTOS
OS (e.g., Linux)
OS (e.g., Linux)
UP2DATE MIDDLEWARE
HW guards (MMU, MPU, IOMMU, ...)
MULTICORE CPUs
GPUs
FPGA
HSM
MEMORY
BUS(es)
Peripheral devices
High Performance Embedded Computing Hardware Platform
rver
ture
Gateway
ECUs

MCS Platform:

- Spatial and temporal system partitioning based on certifiable hypervisor
- Resource management
- Inter-partition communication (IPC)

Security Measures:

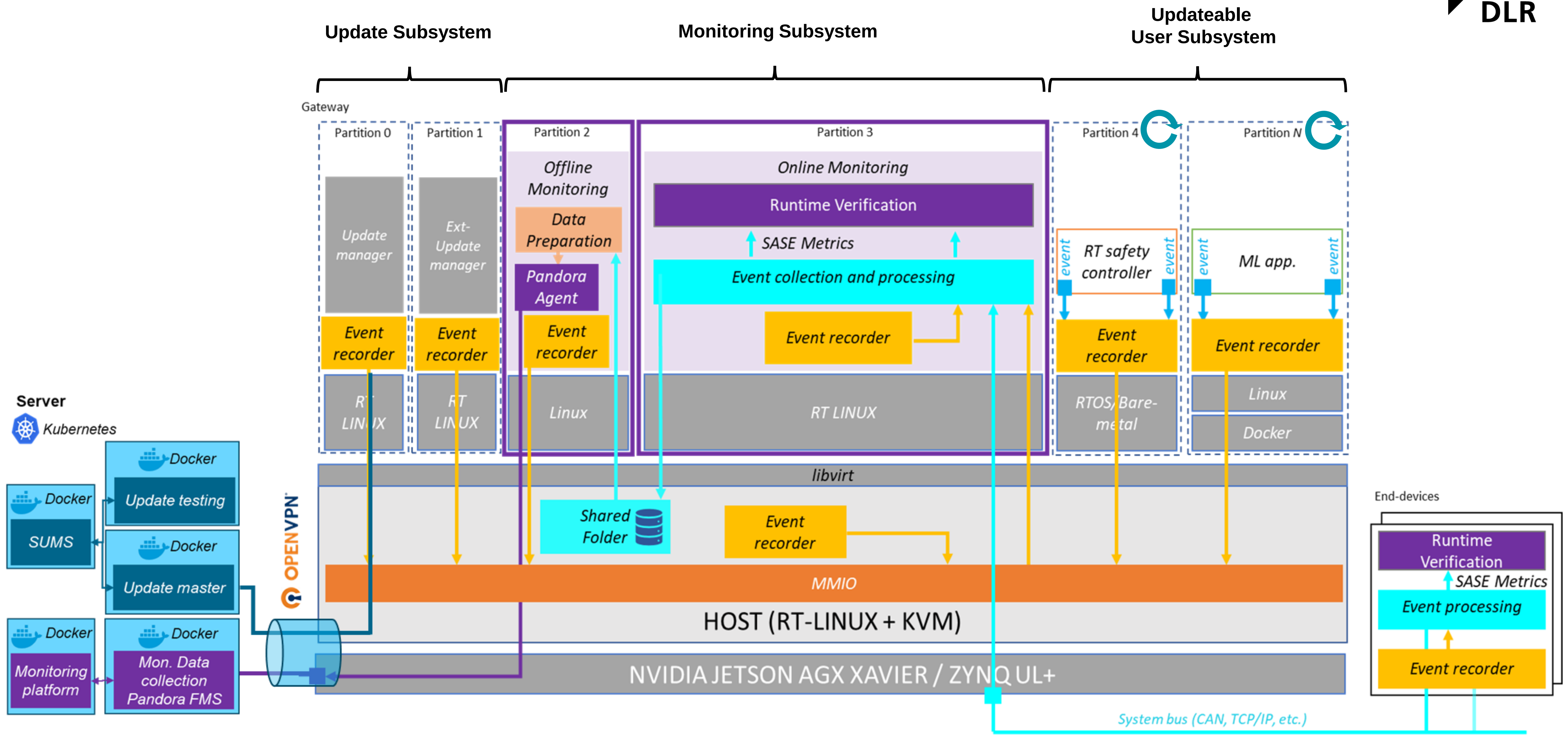
Secure communications:
- Server <-> Gateway
- Server <-> End-Device
- Gateway <-> End-Device

Security Monitoring:
- Health monitoring & Anomaly detection
- Logging
- Security auditor

Secure updates:
- Update authenticity verification

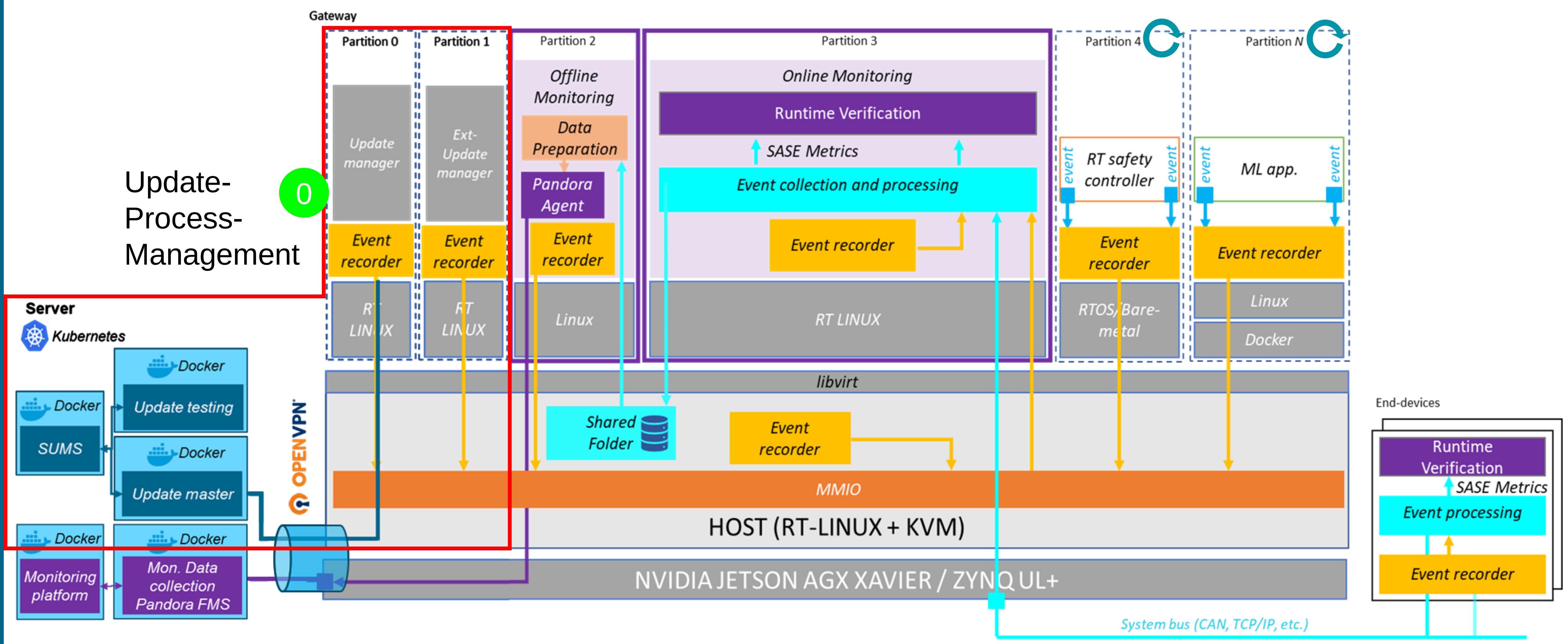
Concept review by external certification authority

Updateable System Architecture



Updateable System Architecture

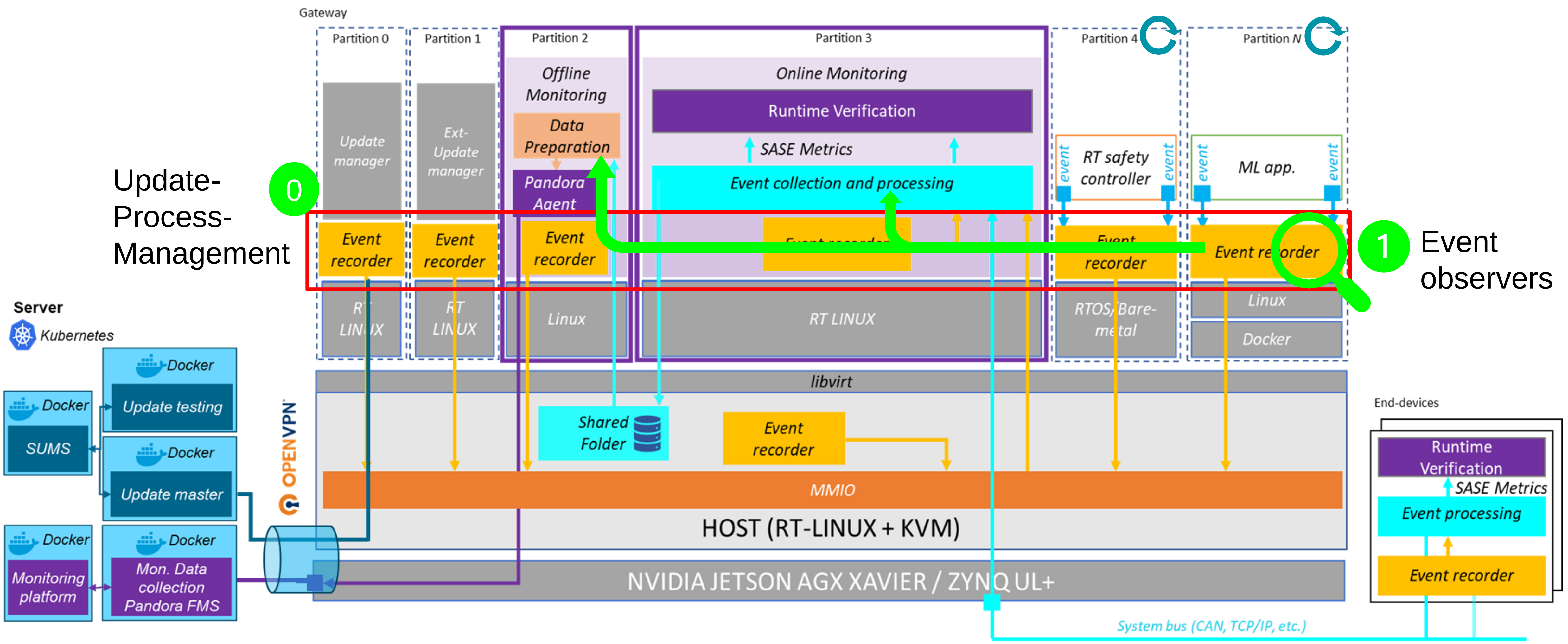
Update-Process-Management



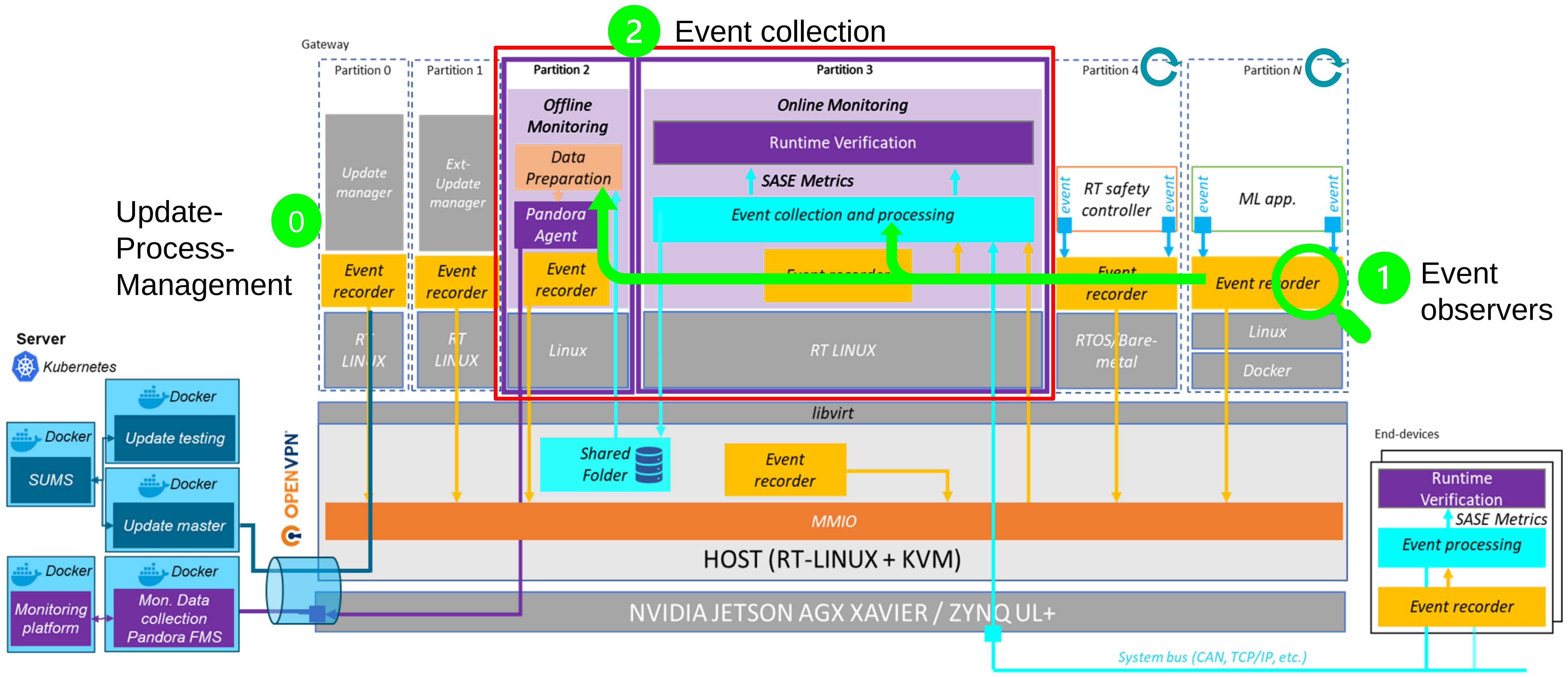
Updateable System Architecture

Update-Process-Management

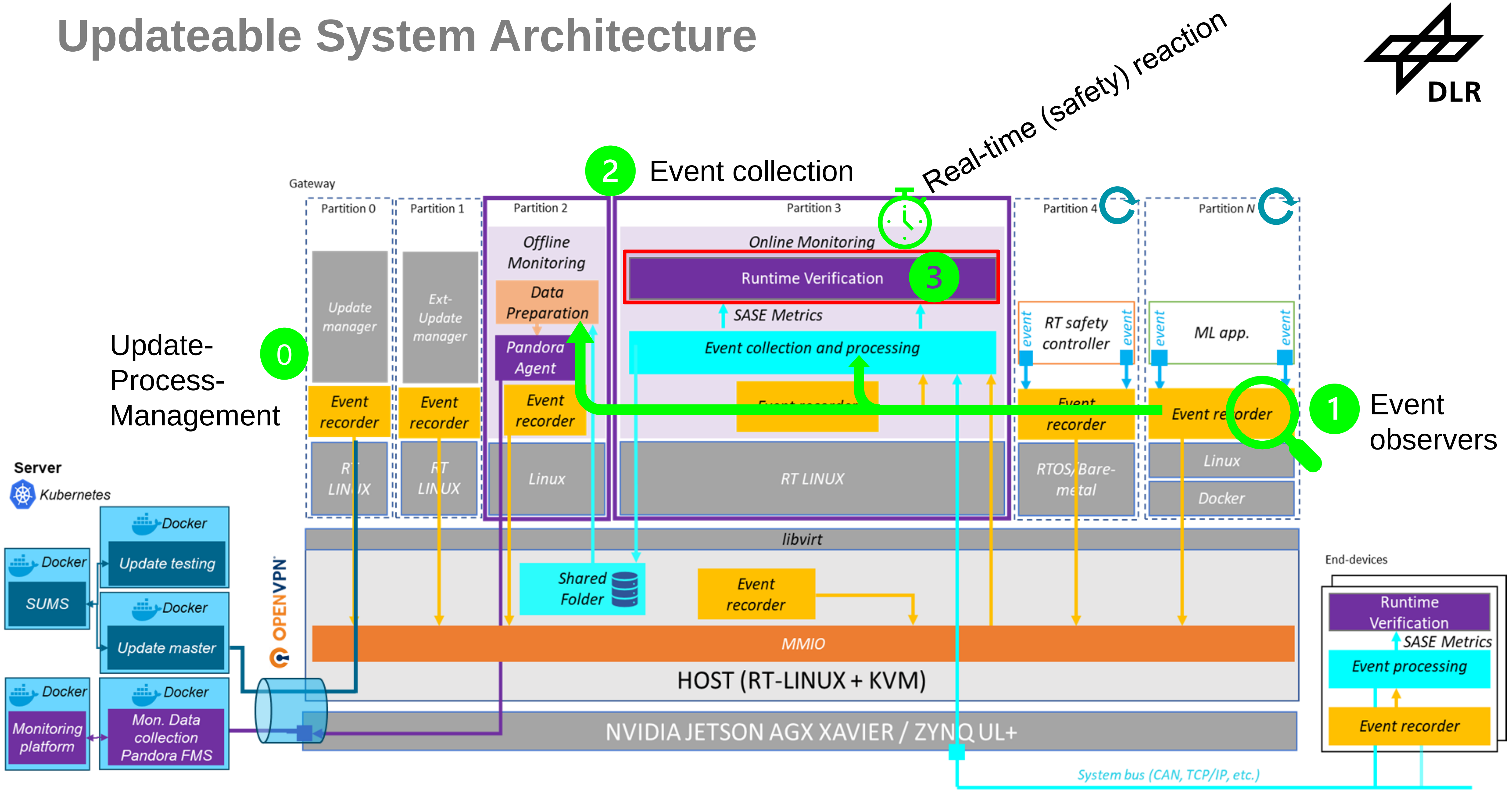
Event observers



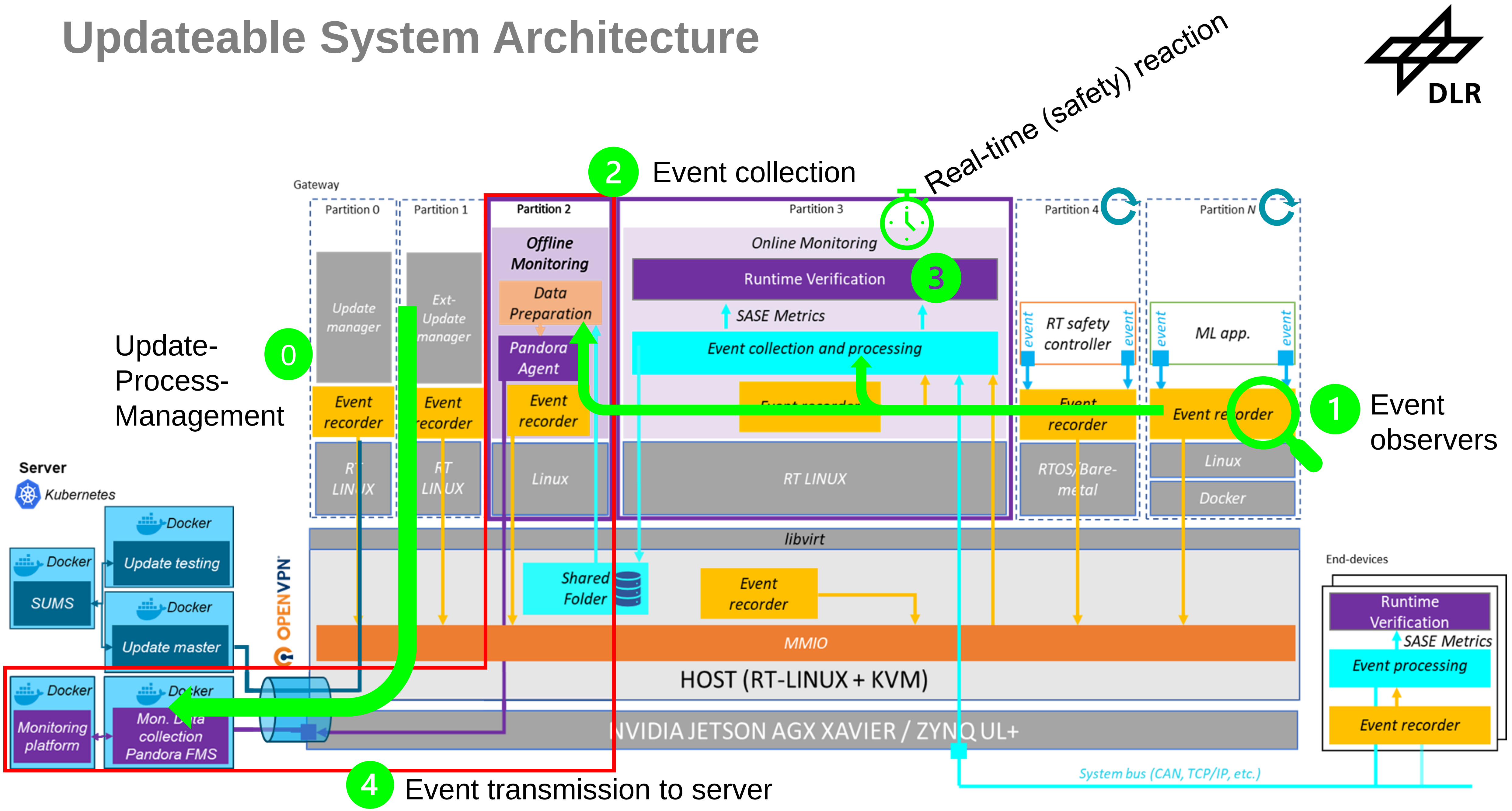
Updateable System Architecture



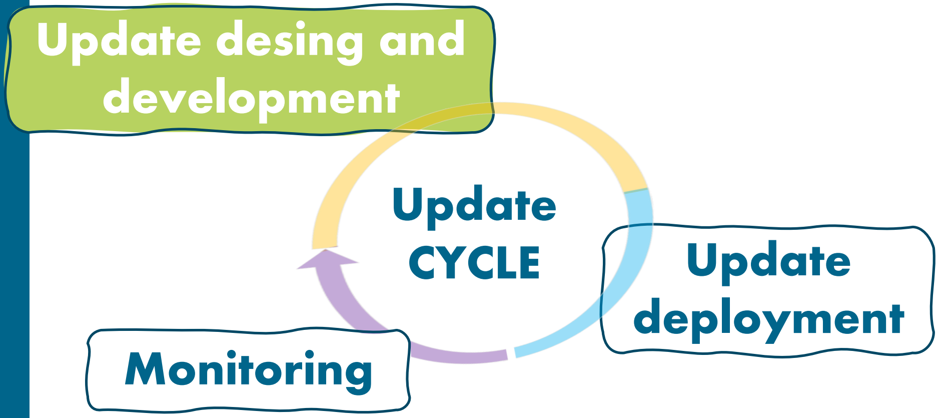
Updateable System Architecture



Updateable System Architecture



Update design and development



7.4.2.4 The design method chosen shall possess features that facilitate software modification. Such features include modularity, information hiding and encapsulation.
IEC-61508-3

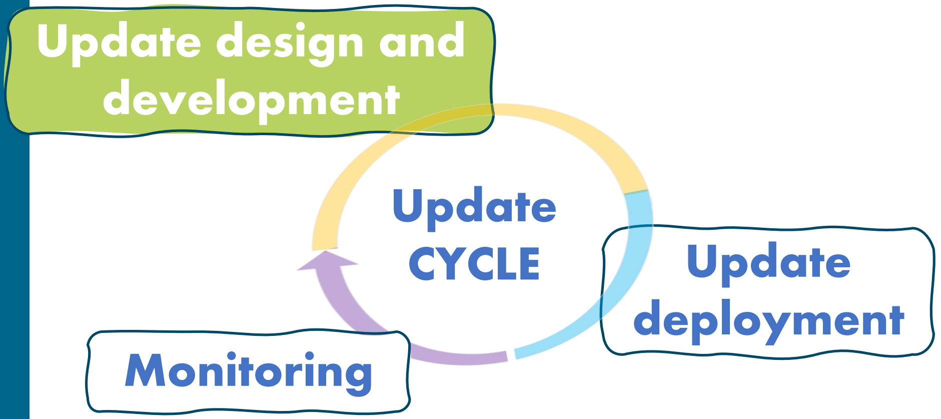
Contract – formalized description
of the conditions of integration
(real-time, resources,
functionality, safety aspects)

UP2DATE update compatibility is defined by:
→ Mutual satisfaction of resource- and metadata-requirements
→ Refinement of (implicit) resource-limits and metadata-criteria
→ Mutual satisfaction of timing-requirements
→ Refinement of timing-specifications

Resource- & Metadata (RMD):
- System-Configurations
- Resource Usage
- Interference upper-bounds
- Power Supply
- Temperature

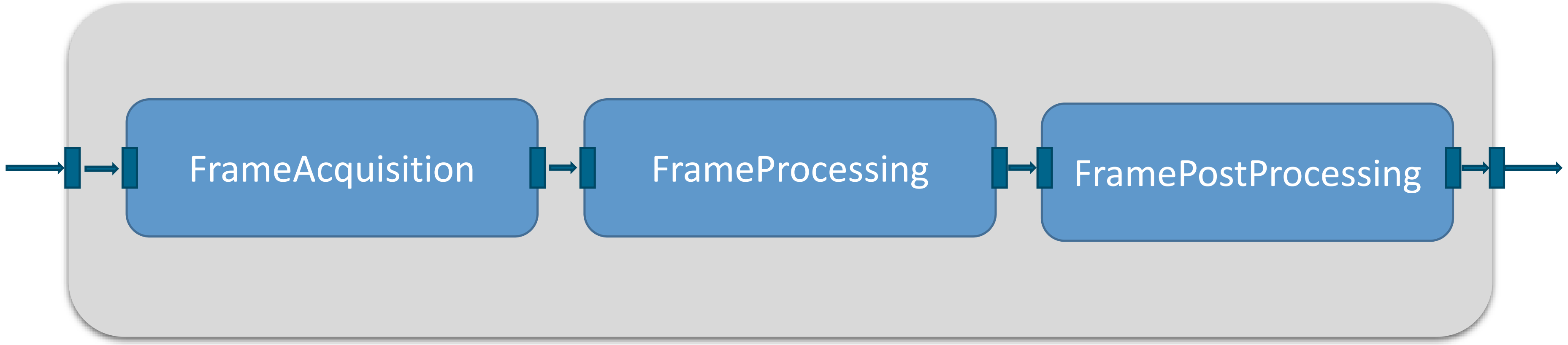
Functional Event Timing (FET):
- Absolute & relative timing of functional events (i.e., control-/data-flow-events)
- Regularity & variance of functional-event timing (i.e., period, delay, jitter,)

Update design and development

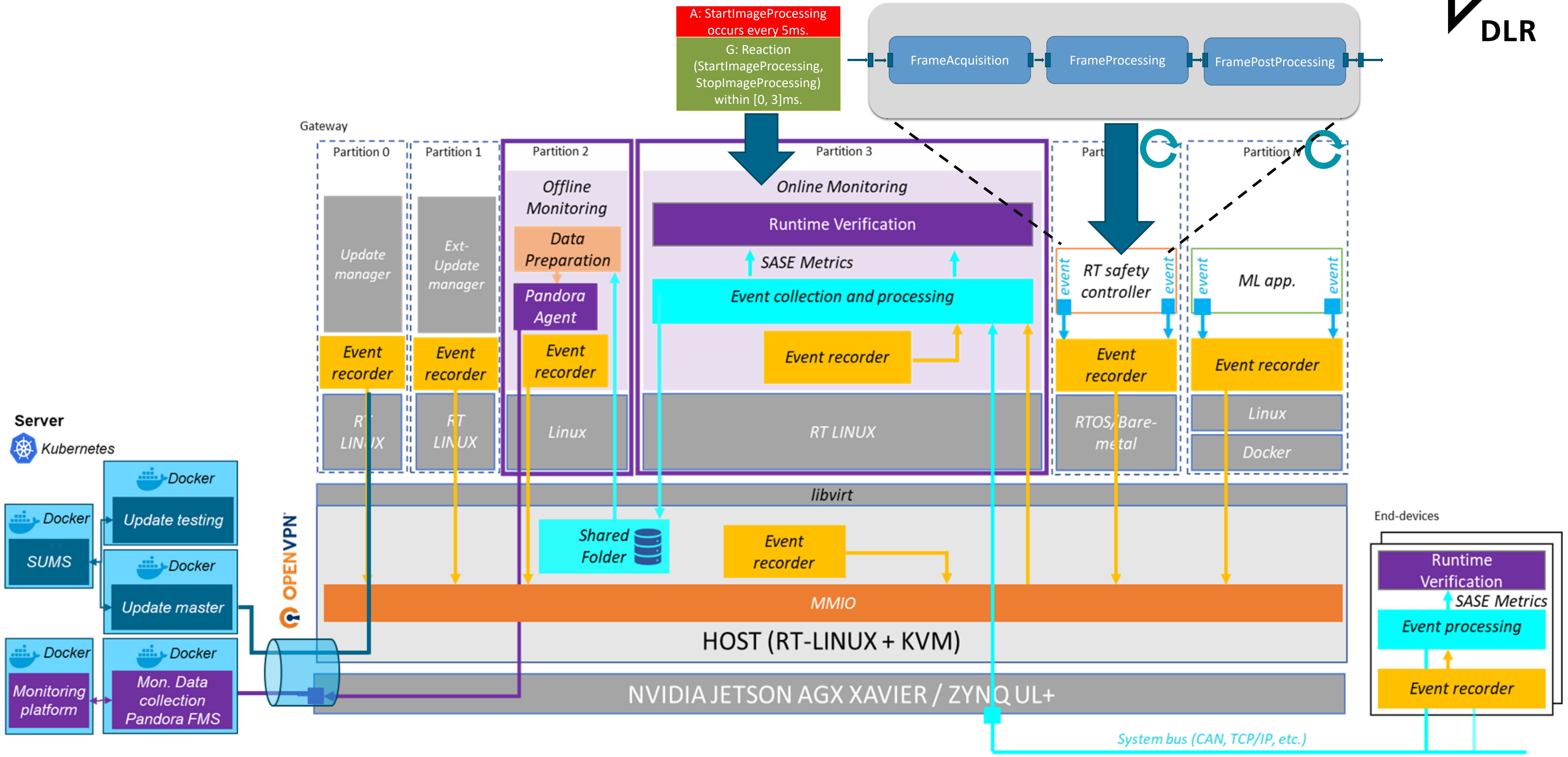


- System and app characterization (contract specification)

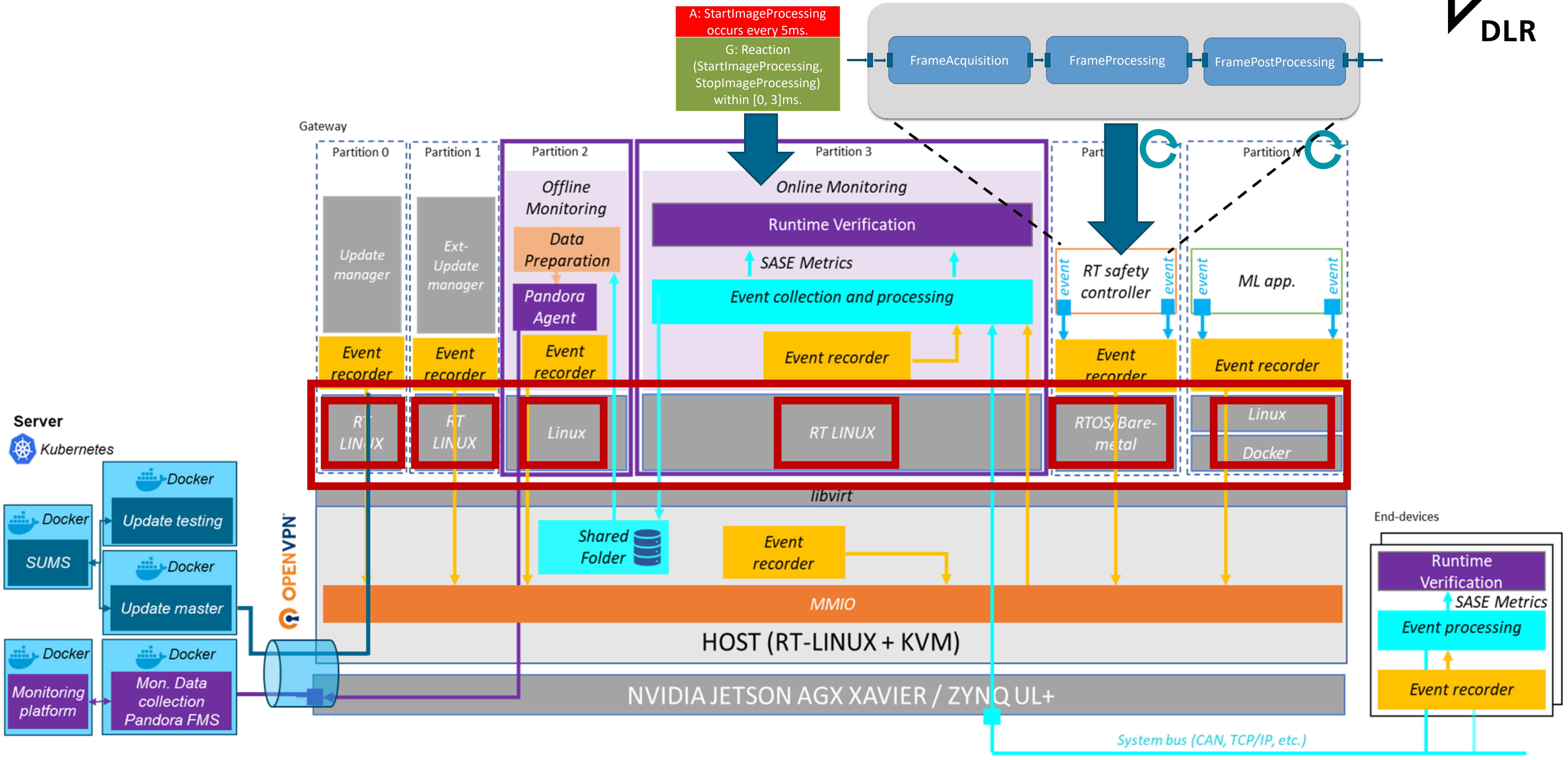
A: StartImageProcessing occurs every 5ms.
G: Reaction (StartImageProcessing, StopImageProcessing) within [0, 3]ms.



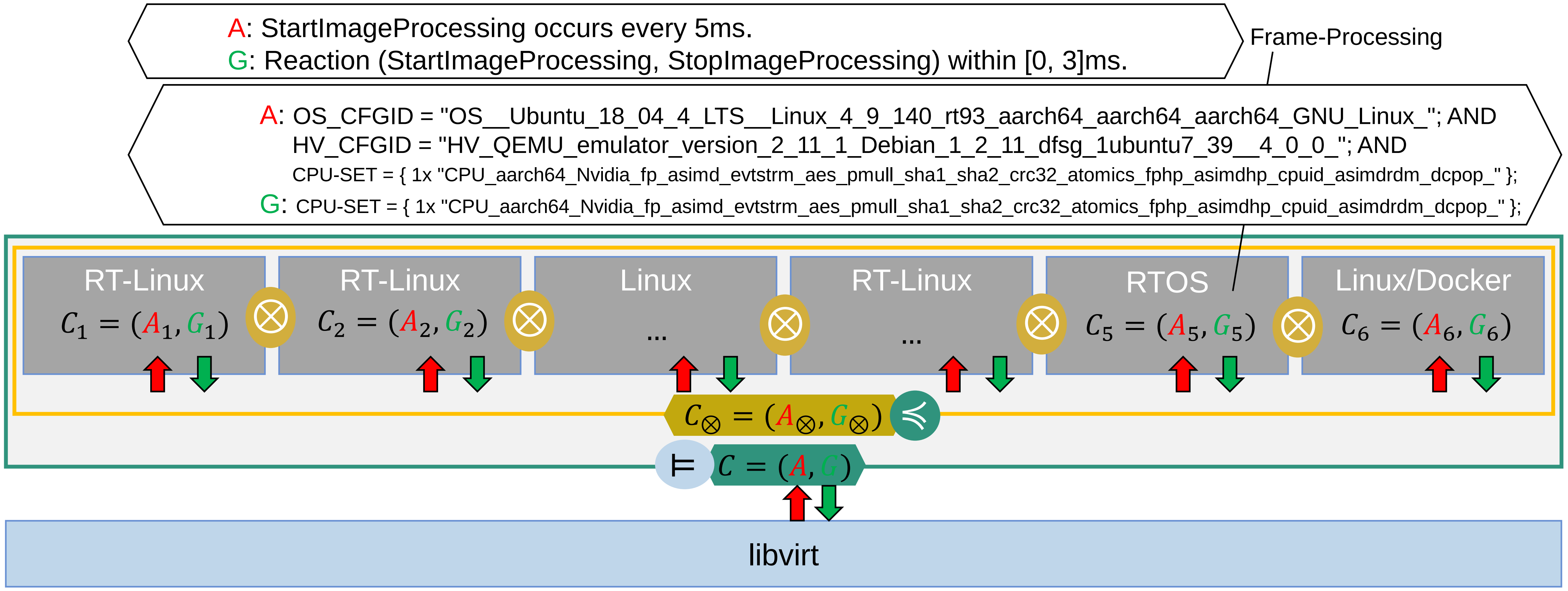
Contract-Based Compatibility (Modularity)



Contract-Based Compatibility (Modularity)



Contract-Based Compatibility (Modularity)



A: StartImageProcessing occurs every 5ms.
G: Reaction (StartImageProcessing, StopImageProcessing) within [0, 3]ms.

Frame-Processing

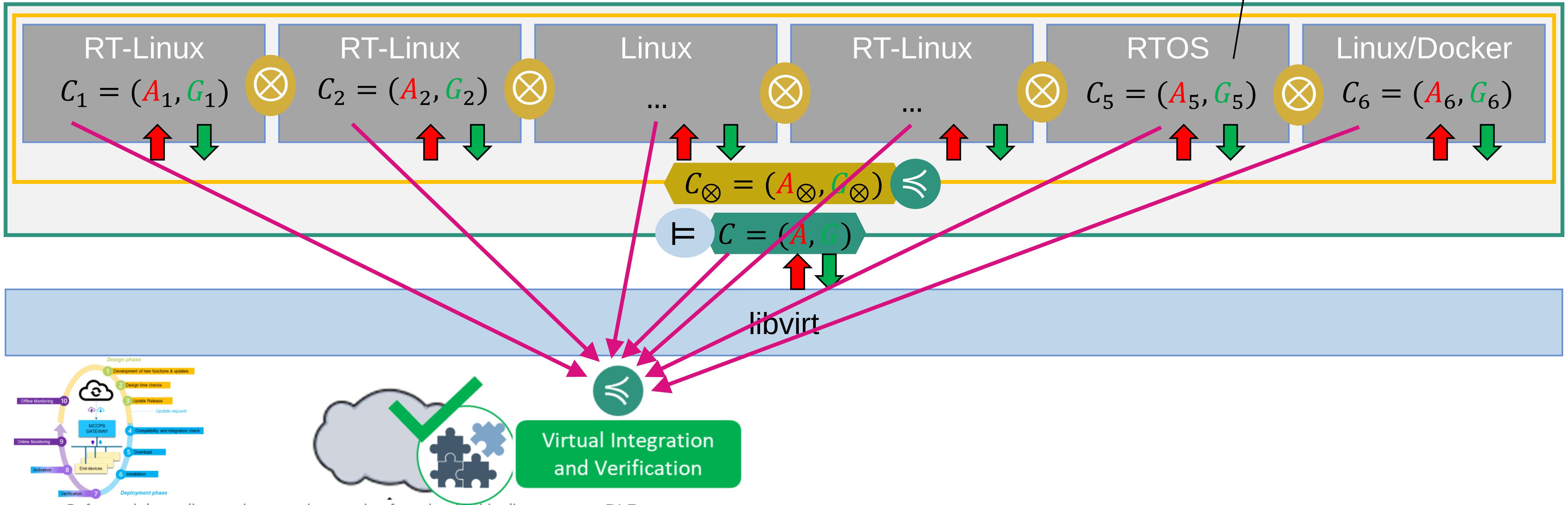
A: OS_CFGID = "OS__Ubuntu_18_04_4_LTS__Linux_4_9_140_rt93_aarch64_aarch64_aarch64_GNU_Linux_"; AND
HV_CFGID = "HV_QEMU_emulator_version_2_11_1_Debian_1_2_11_dfsg_1ubuntu7_39__4_0_0_"; AND
CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };
G: CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };

Contract-Based Compatibility (Modularity)

A: StartImageProcessing occurs every 5ms.
G: Reaction (StartImageProcessing, StopImageProcessing) within [0, 3]ms.

Frame-Processing

A: OS_CFGID = "OS__Ubuntu_18_04_4_LTS__Linux_4_9_140_rt93_aarch64_aarch64_aarch64_GNU_Linux_"; AND
HV_CFGID = "HV_QEMU_emulator_version_2_11_1_Debian_1_2_11_dfsg_1ubuntu7_39__4_0_0_"; AND
CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };
G: CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };

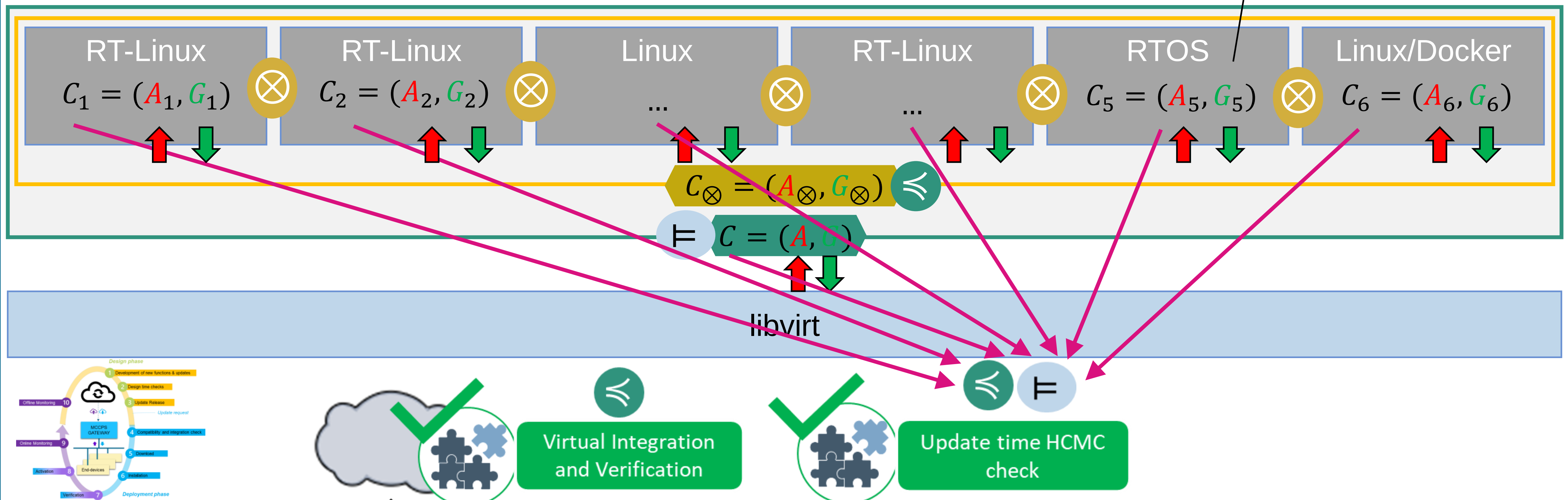


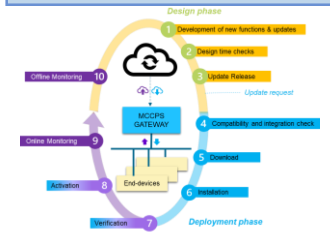
Contract-Based Compatibility (Modularity)

A: StartImageProcessing occurs every 5ms.
G: Reaction (StartImageProcessing, StopImageProcessing) within [0, 3]ms.

Frame-Processing

A: OS_CFGID = "OS__Ubuntu_18_04_4_LTS__Linux_4_9_140_rt93_aarch64_aarch64_aarch64_GNU_Linux_"; AND
HV_CFGID = "HV_QEMU_emulator_version_2_11_1_Debian_1_2_11_dfsg_1ubuntu7_39__4_0_0_"; AND
CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };
G: CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };



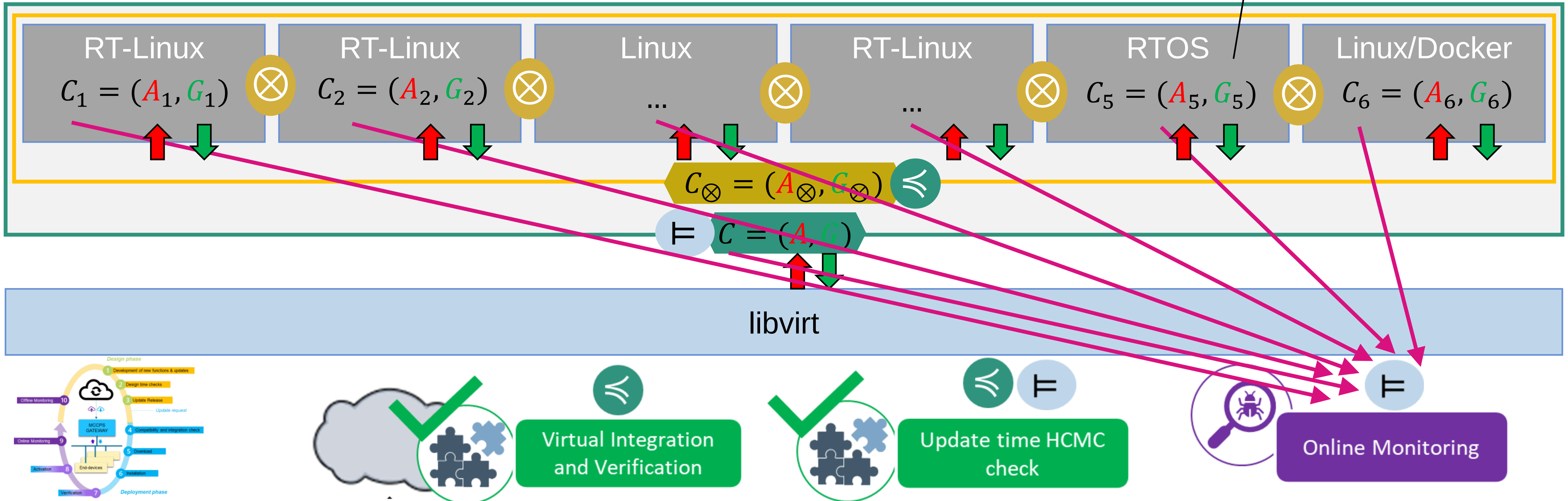


Contract-Based Compatibility (Modularity)

Frame-Processing

A: StartImageProcessing occurs every 5ms.
G: Reaction (StartImageProcessing, StopImageProcessing) within [0, 3]ms.

A: OS_CFGID = "OS__Ubuntu_18_04_4_LTS__Linux_4_9_140_rt93_aarch64_aarch64_aarch64_GNU_Linux_"; AND
HV_CFGID = "HV_QEMU_emulator_version_2_11_1_Debian_1_2_11_dfsg_1ubuntu7_39__4_0_0_"; AND
CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };
G: CPU-SET = { 1x "CPU_aarch64_Nvidia_fp_asimd_evtstrm_aes_pmull_sha1_sha2_crc32_atomics_fphp_asimdhp_cpuid_asimdrdm_dcpop_" };



Contract-Based Compatibility (Modularity)

Compatibility is:

- based on components & composition
- a relation between components

Def: Components are compatible, if they

1. don‘t harm each other and
2. cooperate (interact) as intended

Contract-Based Compatibility (Modularity)

Compatibility is:

- based on components & composition
- a relation between components

Def: Components are compatible, if they

1. don‘t harm each other and
2. cooperate (interact) as intended

Components don‘t harm each other iff:

- a component‘s assumptions are not violated by its environment
- the component‘s guarantees don‘t lead to violated assumptions within its environment

→ Check satisfaction between assumptions and (composed) guarantees

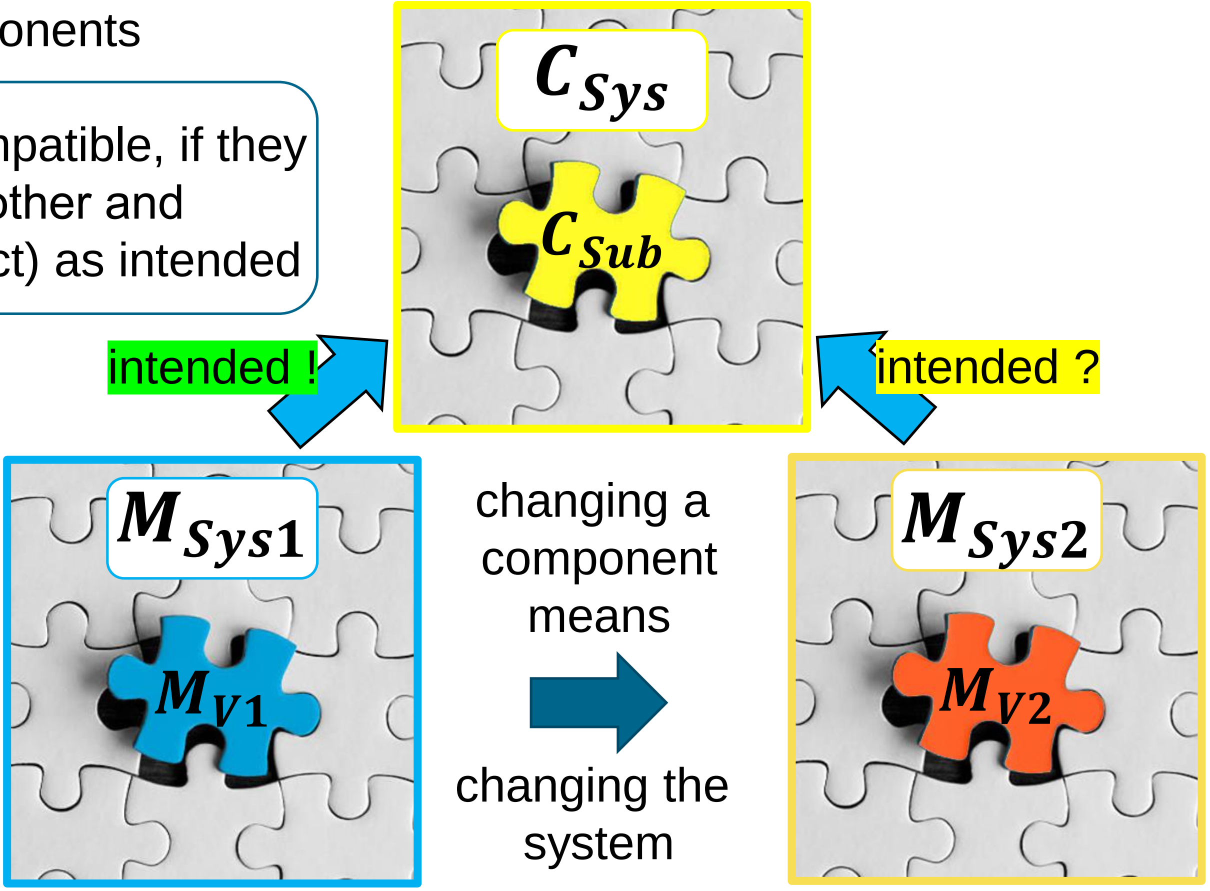
Contract-Based Compatibility (Modularity)

Compatibility is:
- based on components & composition
- a relation between components

Def: Components are compatible, if they
1. don‘t harm each other and
2. cooperate (interact) as intended

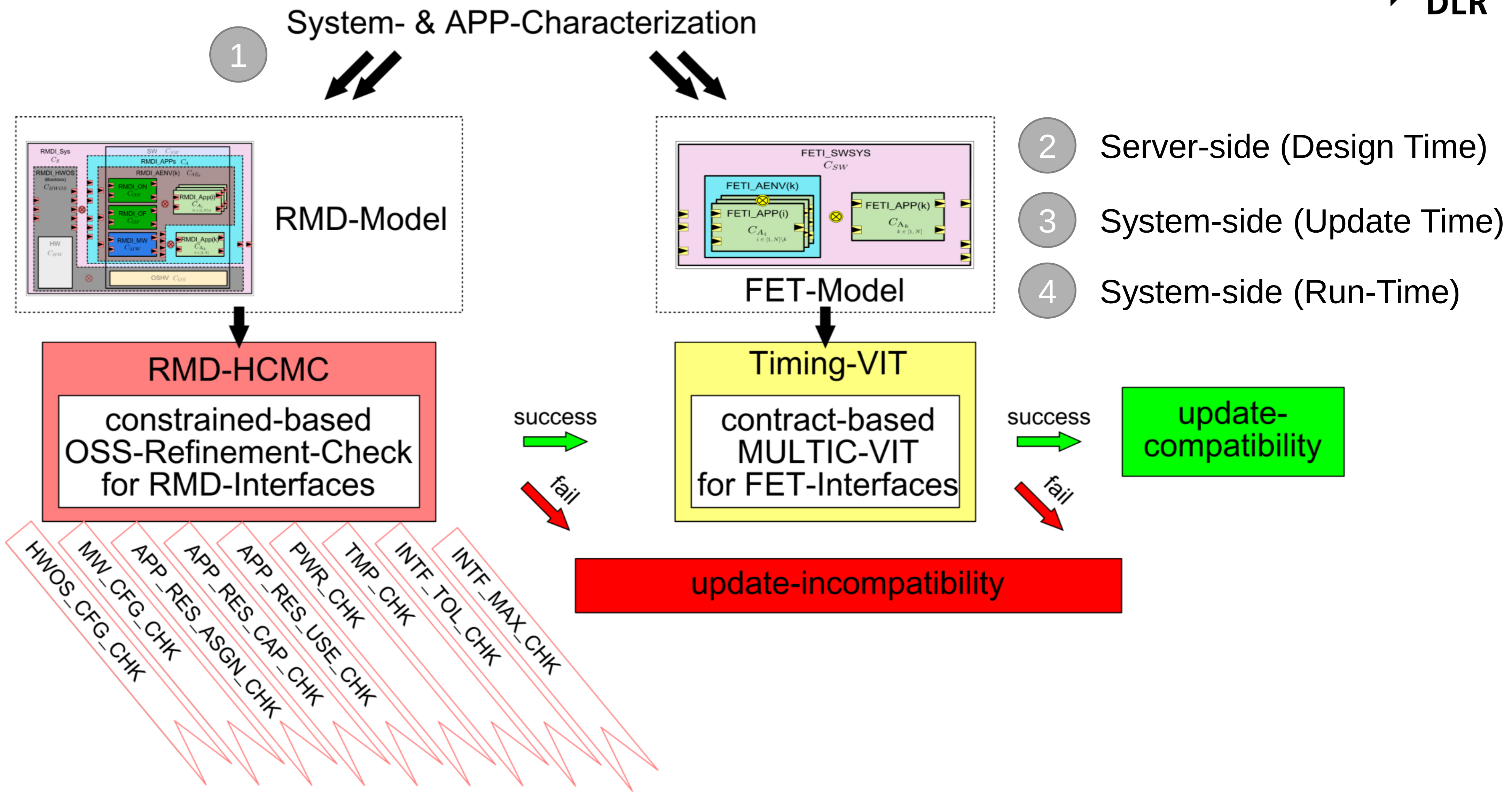
Components don‘t harm each other iff:
- a component‘s assumptions are not violated by its environment
- the component‘s guarantees don‘t lead to violated assumptions within its environment

→ Check satisfaction between assumptions and (composed) guarantees

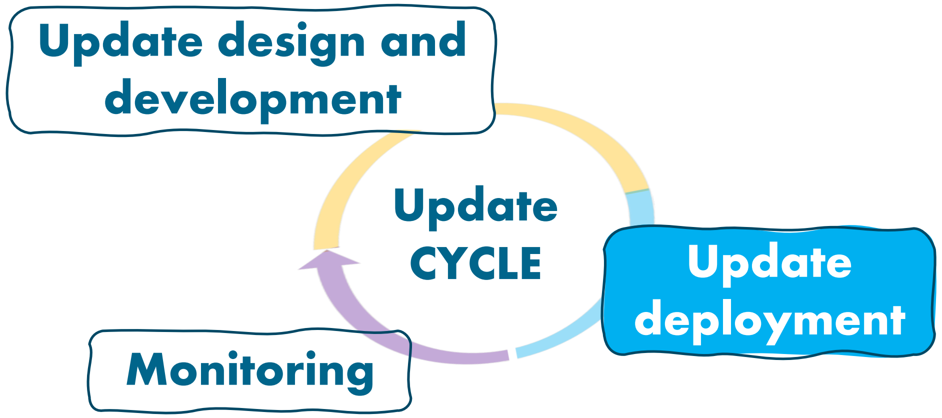


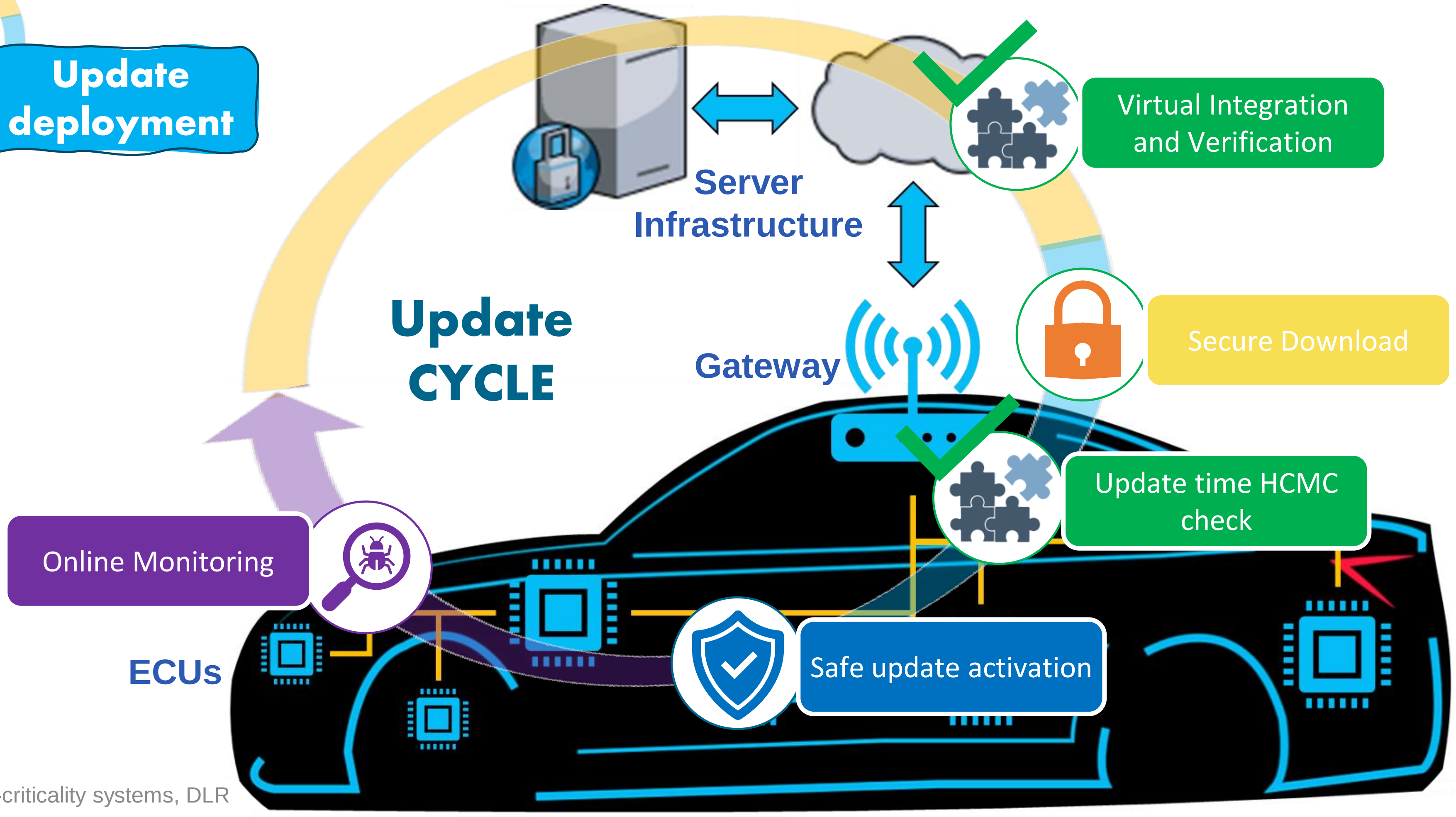
→ Check refinement between a system-specification and different versions of system-compositions

Contract-Based Compatibility (Modularity)

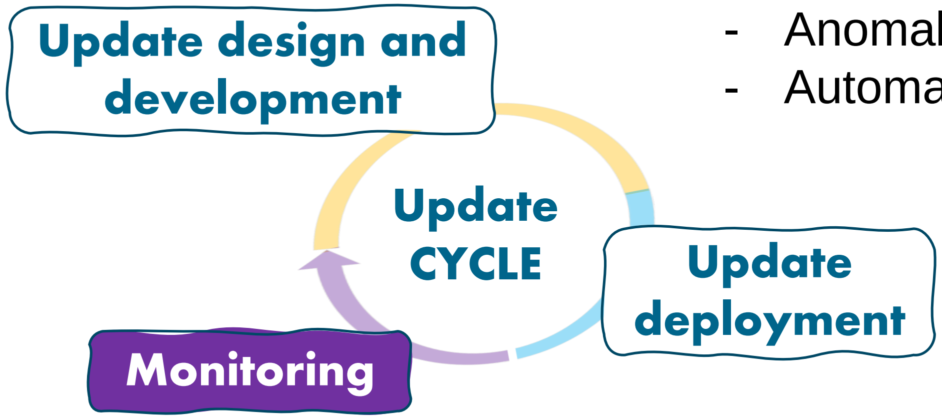


Deployment


Update design and development
Update CYCLE
Update deployment
Monitoring

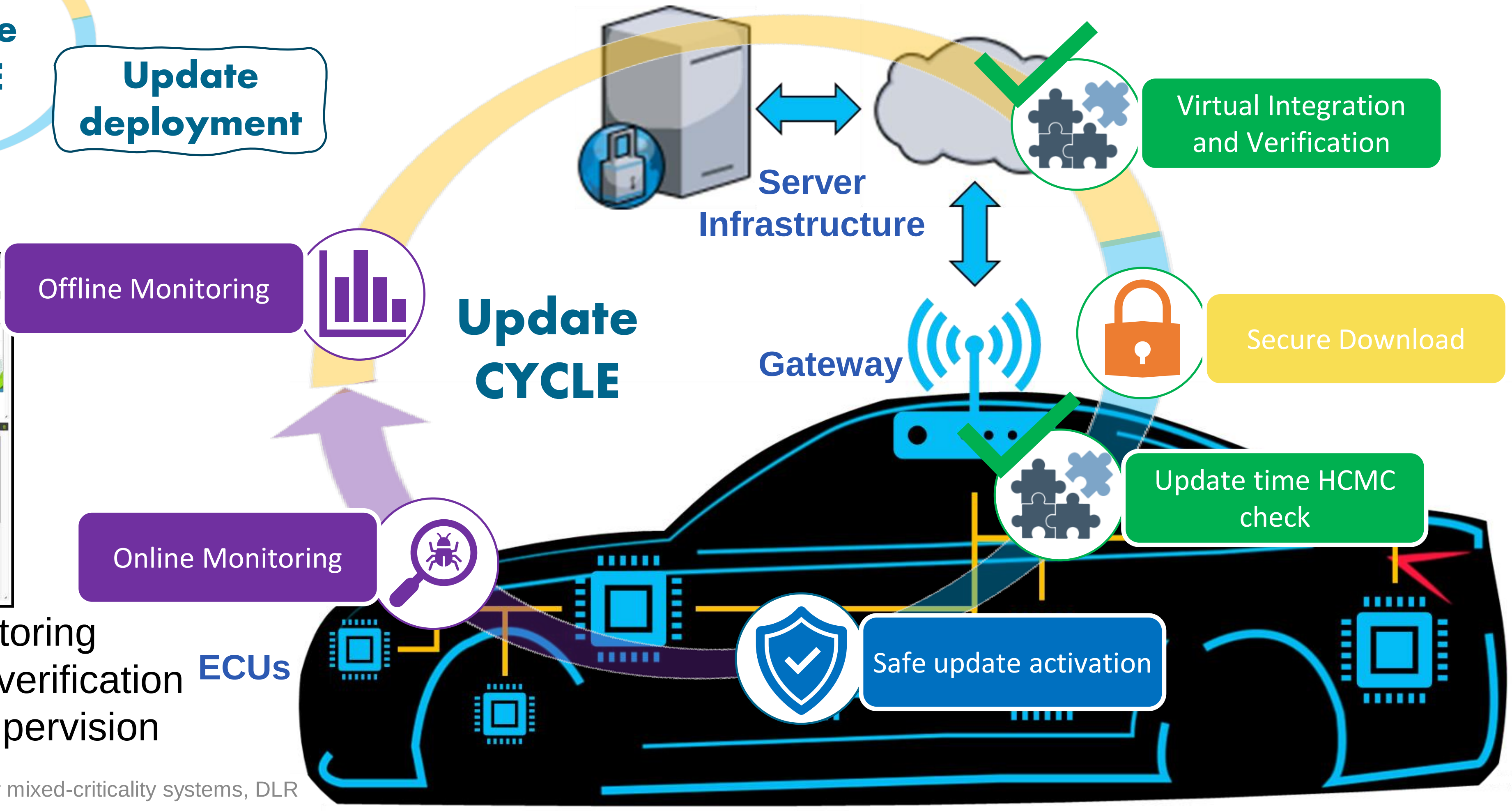

Server Infrastructure
Virtual Integration and Verification
Update CYCLE
Gateway
Secure Download
Update time HCMC check
Online Monitoring
ECUs
Safe update activation

Monitoring



Offline monitoring

- Anomaly detection
- Automated security auditor



Online monitoring

- Runtime verification
- Safety supervision

Conclusion and open challenges

- Proof-of-concept modular update process and middleware (server to gateway to end-device) ✓
- Contract based compatibility checking and online monitoring promising approach ✓
- No functional (SOTIF) properties considered so far ⚠
- Only static resource properties supported so far ⚠
- Interference challenge on shared resources still not sufficiently solved ⚠
  - Today‘s COTS HW still not designed appropriately (see „CAST-32A” and “AC 20-193” for Avionics Multi-Core Processing)
  - Multi-Core Processing Platform with robust partitioning required
    - Robust Resource and Time Partitioning not only between software applications hosted on the same core, but also between applications hosted on different cores of an MCP or between applications that have threads hosted on several cores
  - Joint HW/SW approach required for partitioning hypervisor with guaranteed robust partitioning
  - Our contract based approach would highly benefit from such robust partitioning, since it allows
    - to exploit the power on incremental update compatibility checking
    - and a shift towards virtual verification

Contact info & credits

Dr. Kim Grüttner

+49 441 770507-300
kim.gruettner@dlr.de
DLR.de

German Aerospace Center (DLR)
Deutsches Zentrum für Luft- und Raumfahrt e. V.
Institute of Systems Engineering for Future Mobility (SE)
Head of Department of System Evolution and Operation (EVO)
Escherweg 2 | 26121 Oldenburg

Bundesministerium für Bildung und Forschung

Horizon 2020 Programme

Bundesministerium für Bildung und Forschung

V&V4NGC
Methoden, Prozesse und Werkzeugketten für die Validierung & Verifikation von Next Generation Car

DLR