MathWorks
AUTOMOTIVE
CONFERENCE 2024
Europe

End-2-End framework from Cloud-to-SoC for automotive development for SDV

Thomas Kleinhenz

May 7th, 2024

© Elektrobit 2024 | Public

01 Observed industry challenges

02 How to solve it?

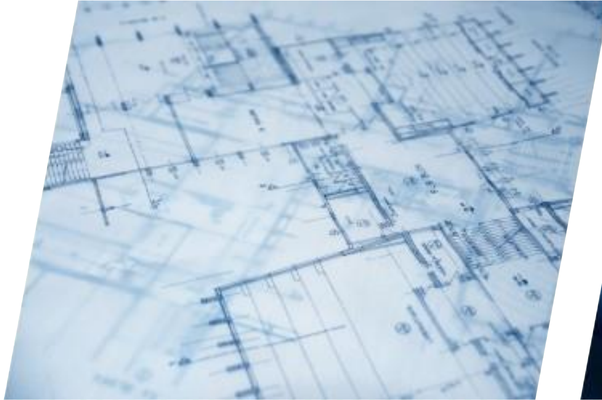
03 What about automation, and virtualization?

04 Do you have some examples?

05 What are the conclusions, and what's next?

Observed industry challenges

Observed industry challenges (... as in focus for this talk)



System definition

Estimation and characterization of system parameter like MIPS, memory footprint (code & data) and power consumption

System evaluation

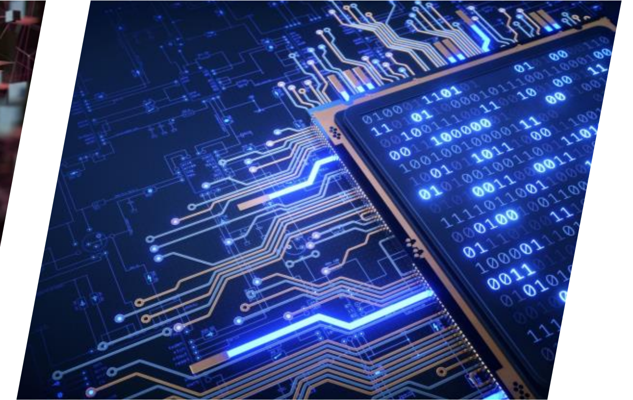
Evaluating the feasibility of new features or functionalities during system definition to ensure that a new feature or functionality can be executed on a preferred HW platform during system definition phase

HW performance forecast for upcoming features and product generations that shall run on the same HW platform

Early integration

Early definition of interfaces (type, signals) including bandwidth requirements

Working in a fully automated cloud environment



Verification

Full functional and non-functional verification of the intended feature or algorithm at low cost and with high degree of re-usability

Natively and seamlessly close the gap between simulation and virtual validation as well as real-world testing

© Elektrobit 2024 | Public

Requirements for any solution

Establish a framework offering a technology and solution that allows

- an analysis to feasibility of running given use cases on an abstracted HW
- a high-level simulation to derive non-functional requirements: MIPS, memory footprint (code & data) and power consumption
- an early integration of interfaces
- a test-harness to verify productive SWC (software-in-the-loop)
- the generation of the software framework

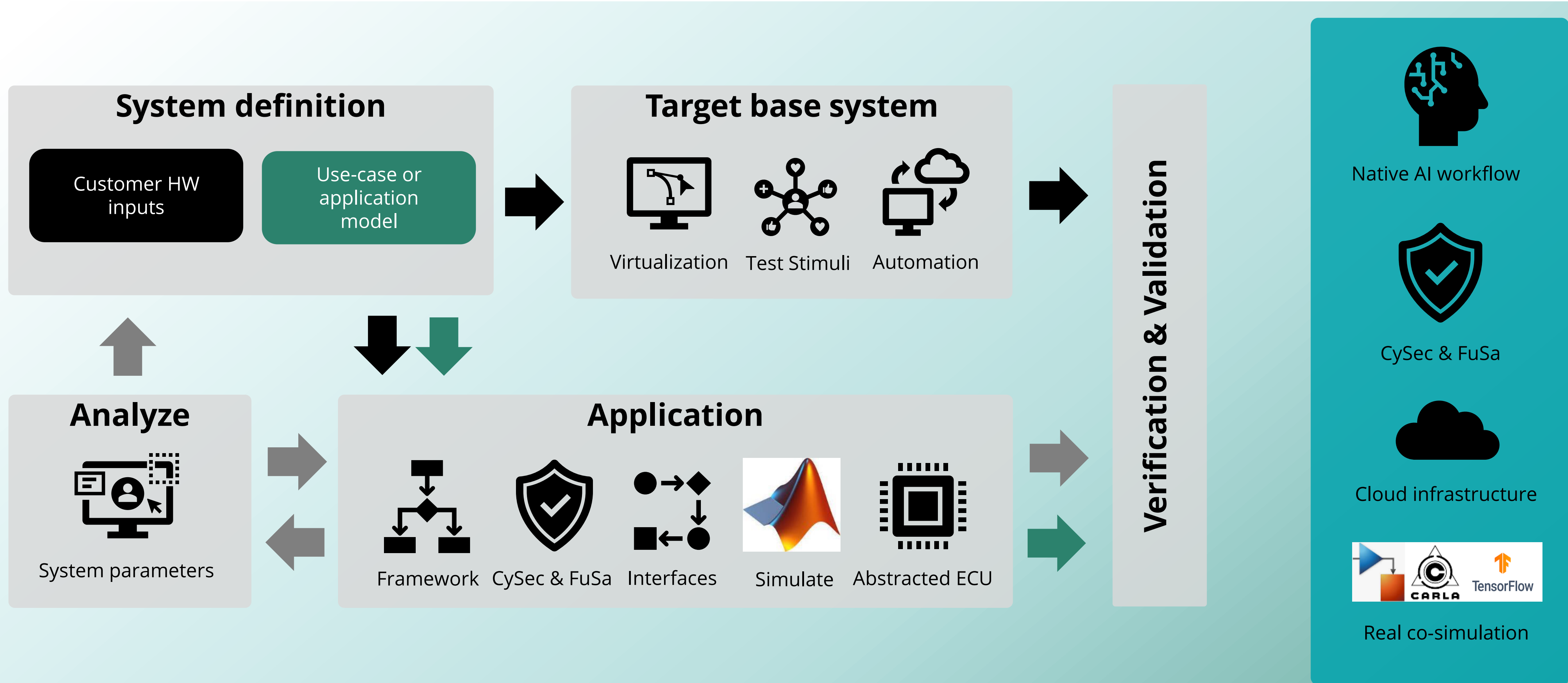
through state-of-art modelling

- MATLAB/Simulink co-design to build abstracted and autogenerated SWC including full requirements management and tracking
- integrate the base SW, SWC, and HW components
- generate Silver virtual ECUs and real ECUs
- validation and test in cloud or real platforms

© Elektrobit 2024 | Public

How to solve it?

Cloud-to-SoC E2E framework for SDV



System Definition: requirements: management and architecture validation/handling
Target Base System: complete middleware, OS and BSW layer plus MCAL
Application: low-level and Autosar application including AI
Analyze: Comprehensive analysis of all system width dependencies, parameter, calibration and results

© Elektrobit 2024 | Public

System-level and application-level perspective

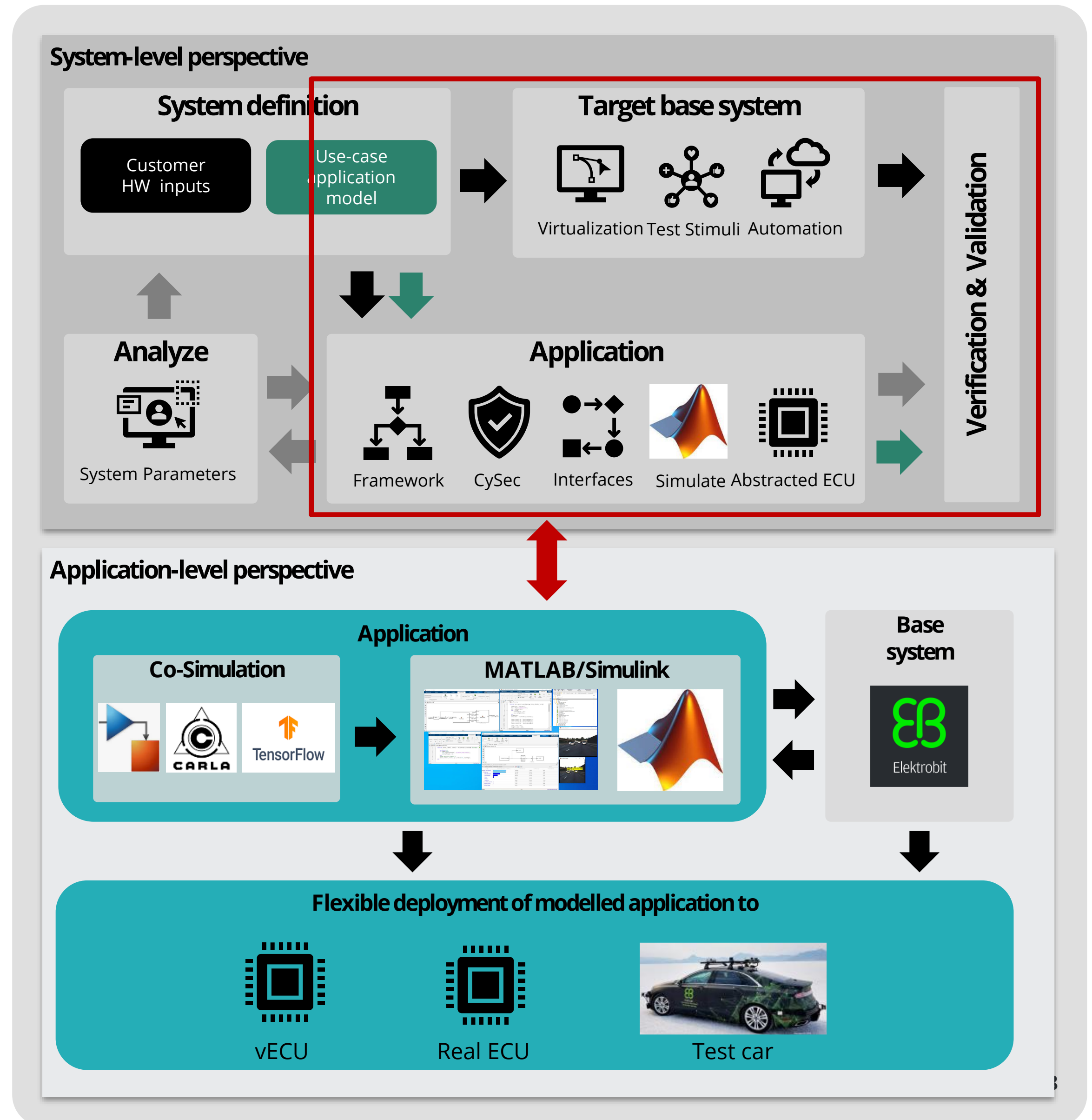
System-level perspective

- Estimation and characterization of system parameter like MIPS, memory footprint (code & data) and power consumption
- Interface definition

Application-level perspective

- Complements system-level-flow to enable an end-2-end closed loop environment in software and hardware for model-based application development
- Instead of an abstracted SWC, a production-grade application model can be used
- Important is interface compliance to allow an exchange of the models for e.g., regression tests

Realization of system-level and application-level in one environment



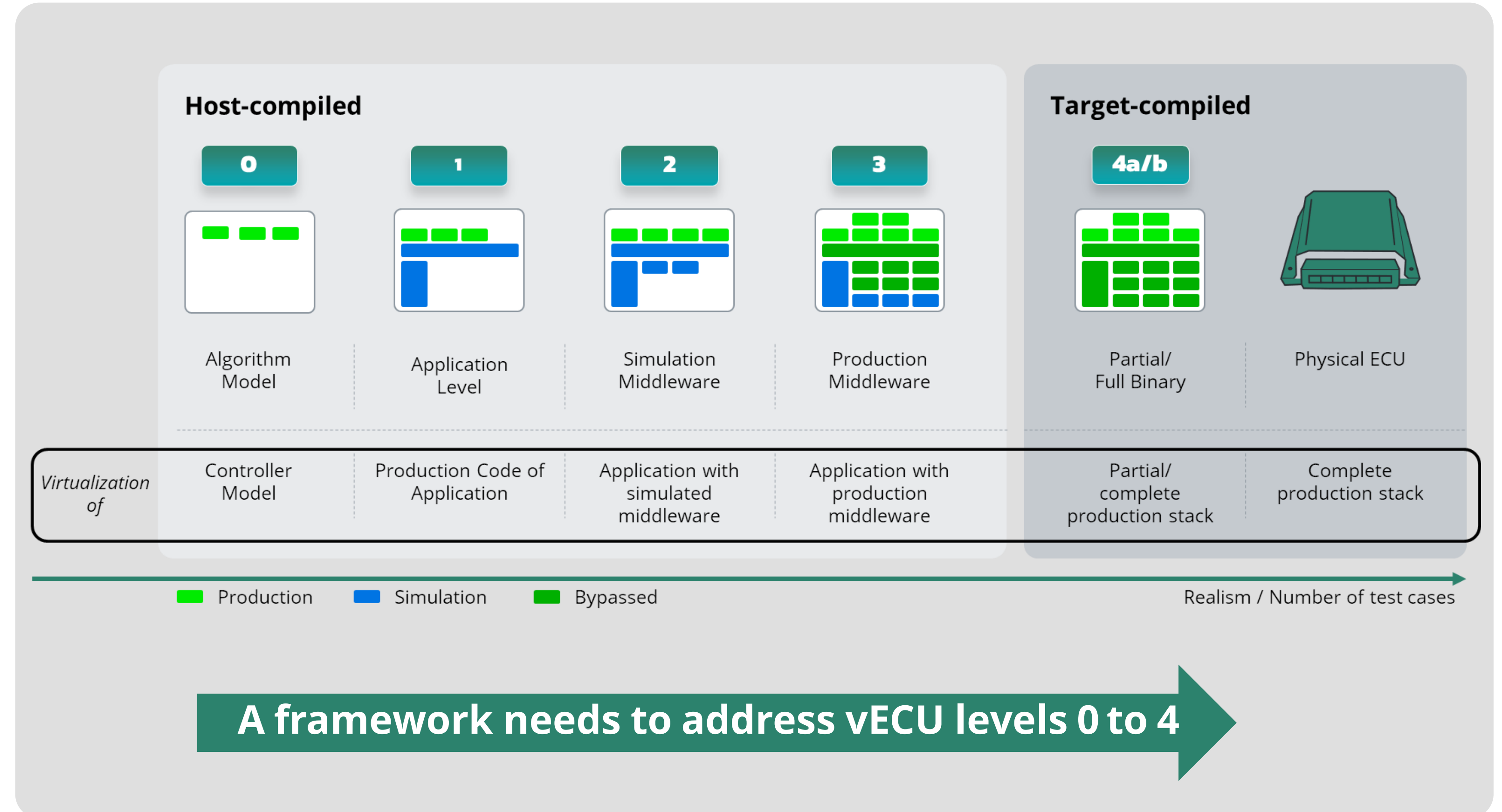
© Elektrobit 2024 | Public

03

What about automation, and virtualization?

Virtual ECU levels

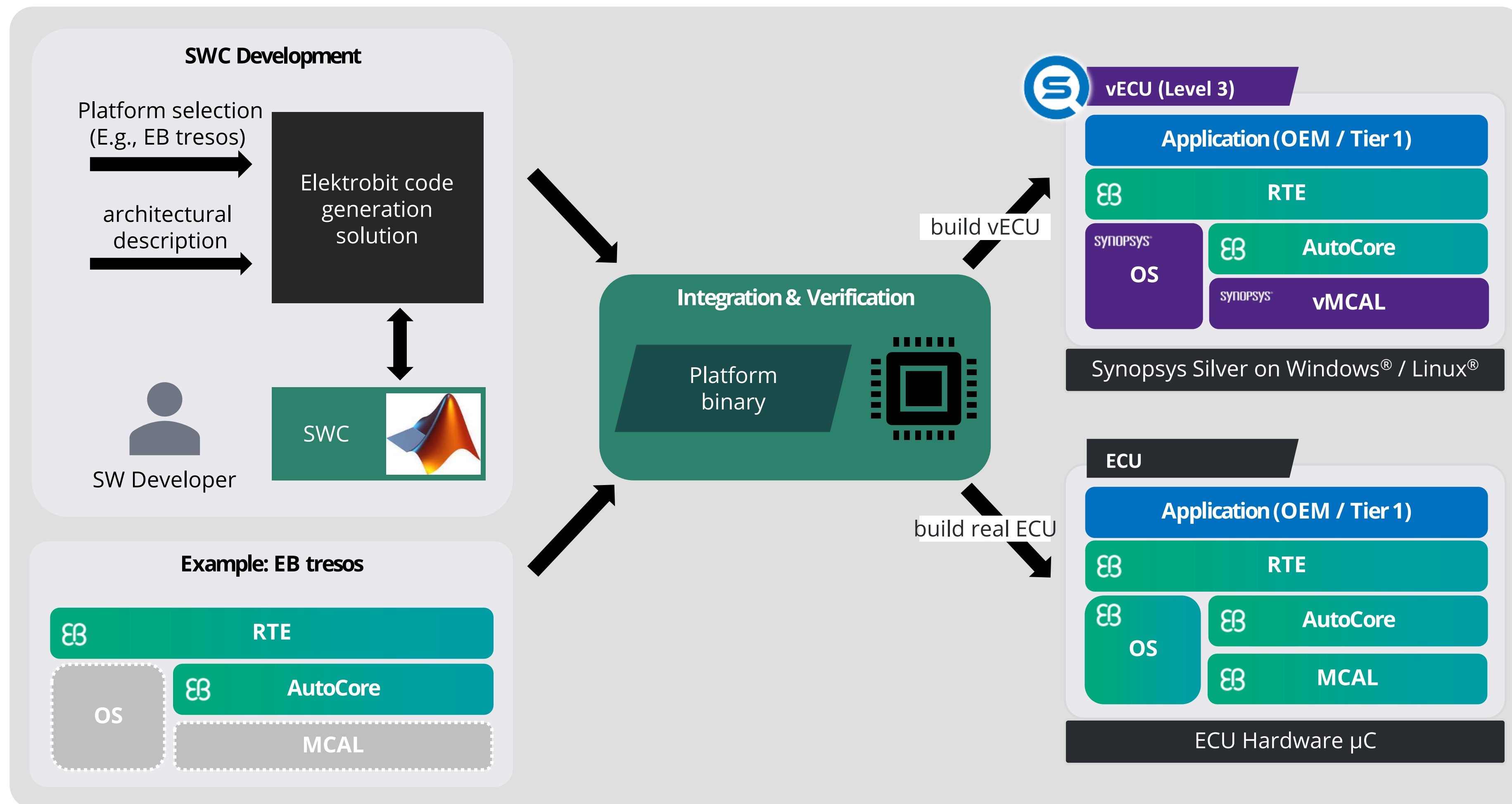
- Level 0: algorithm model, for example in Simulink, of the functionality for fast prototyping up to series release
- Level 1: application-level simulation, hence it includes the production code of the application, but no middleware
- Level 2: simulation middleware
- Level 3: production middleware
- Level 4: full binary, requires HW simulator



Reduced software integration costs by automation

Elektrobit code generation solution allows a template-based approach for integration in any platform including AUTOSAR, ROS2, eCAL

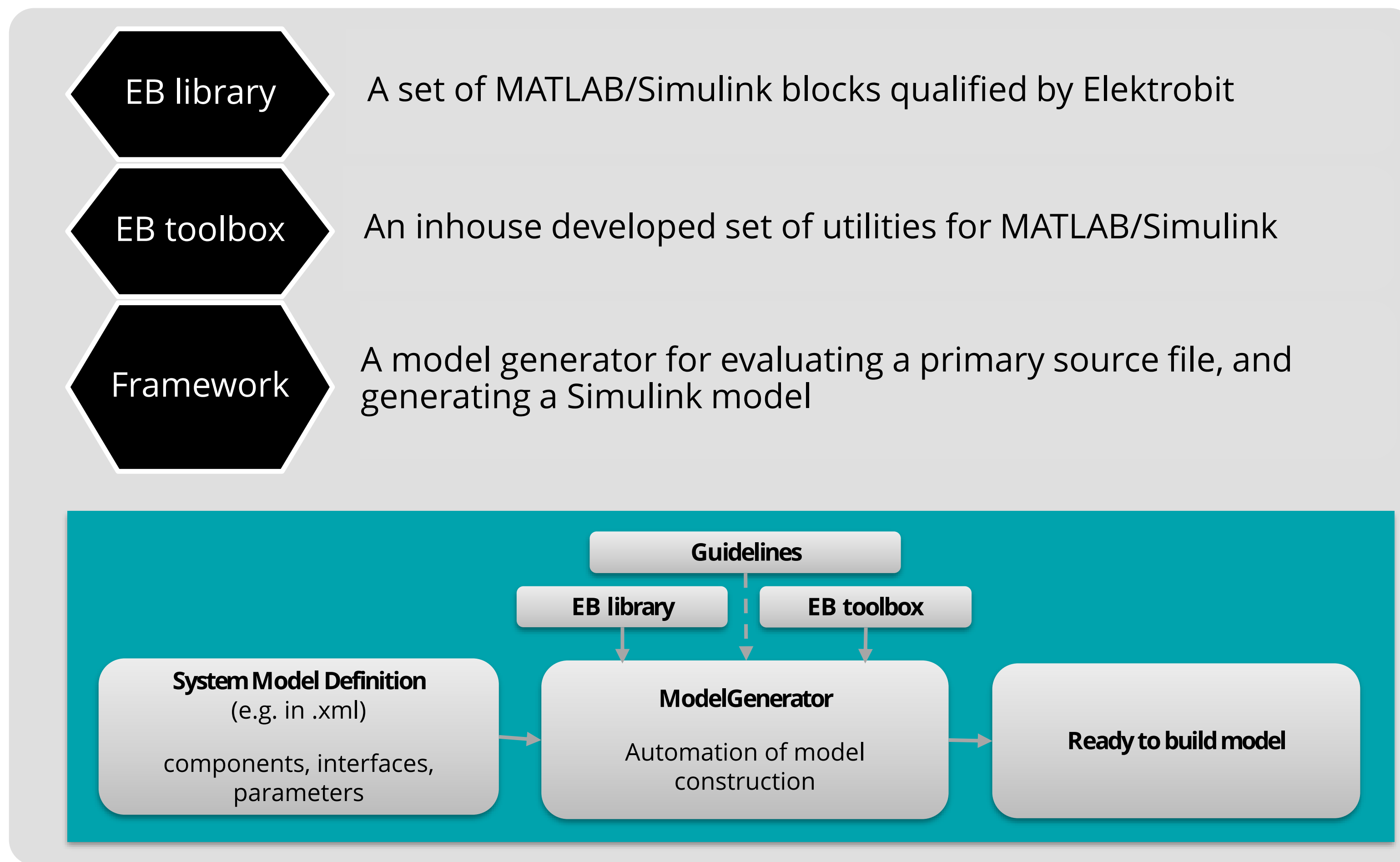
It reduces software integration time by automation and component glue code generation for multiple platforms



MBD Framework, EB Library, and EB Toolbox

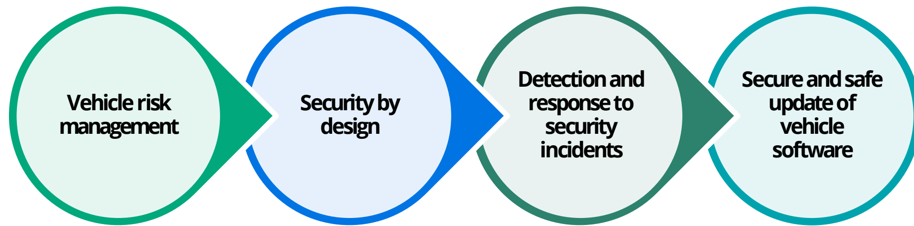
Unify model frame generation and model architecture by using a **one source approach** with high degree auf **automation**

- Creation of interfaces, calibrations, signals, etc.
- Creation and linkage of data dictionaries to defined models
- Forces usage of qualified library block entities in models
- Maintaining consistent structures inside models
- Ready-to-build models
- Model validation at an early stage
- Consistent model configuration, etc.



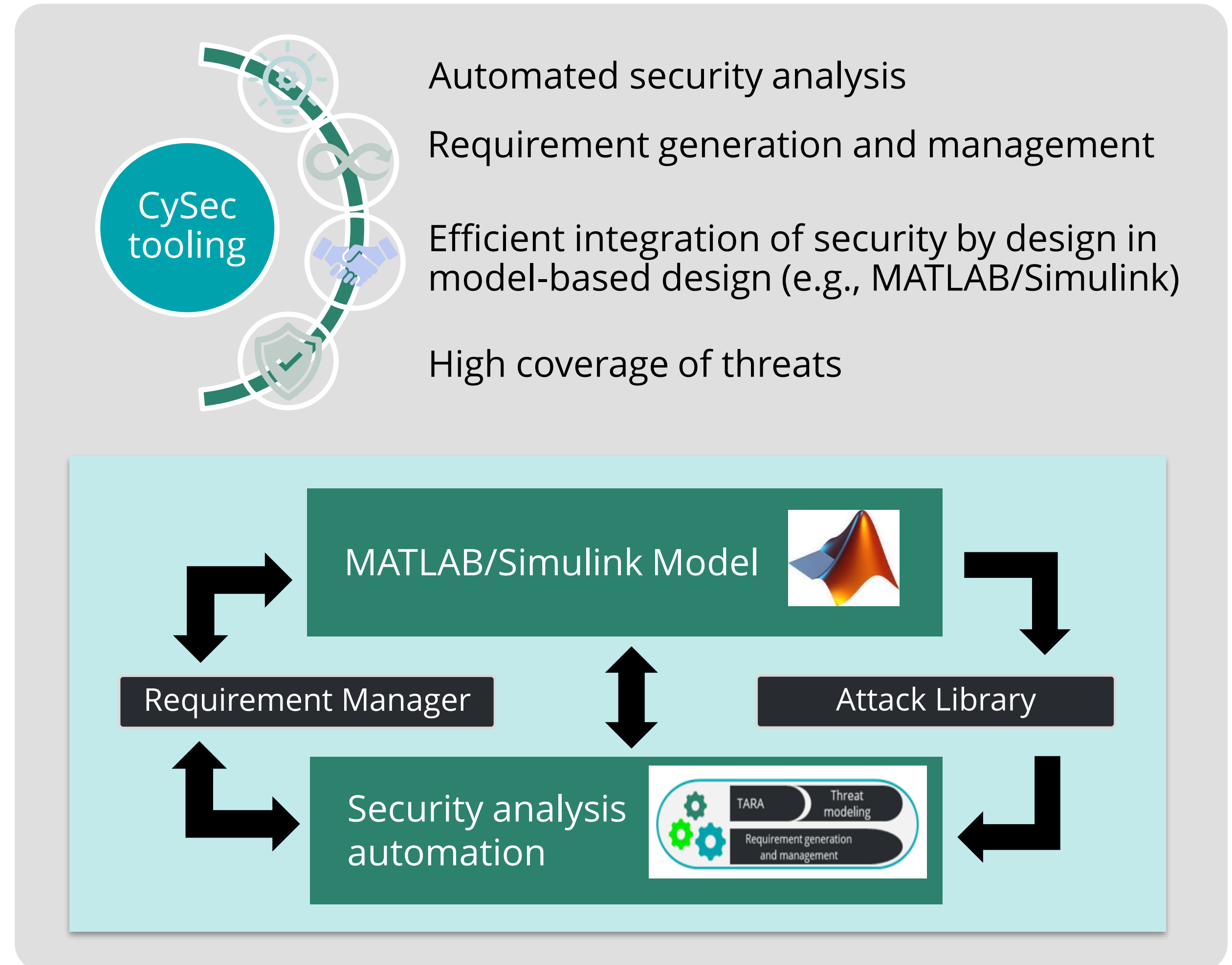
Automated security analysis tooling

The UNECE regulations No. 155 & 156 specify 4 categories:



Most security analysis solutions have many disadvantages:

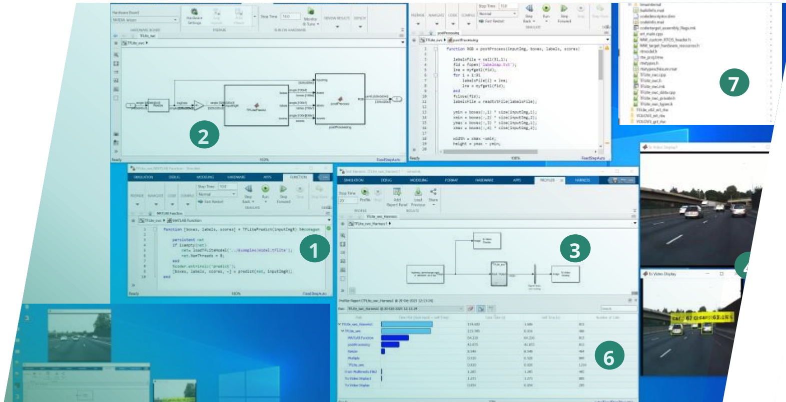
- High cost
- Low coverage
- Inconsistency
- Manual work
- Incompatibility with MBDSE

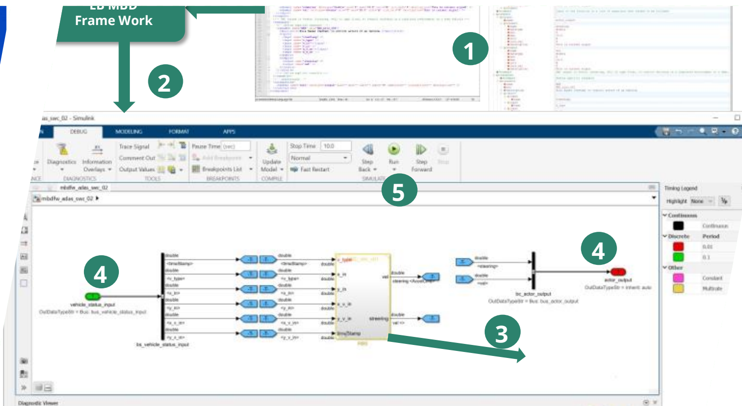


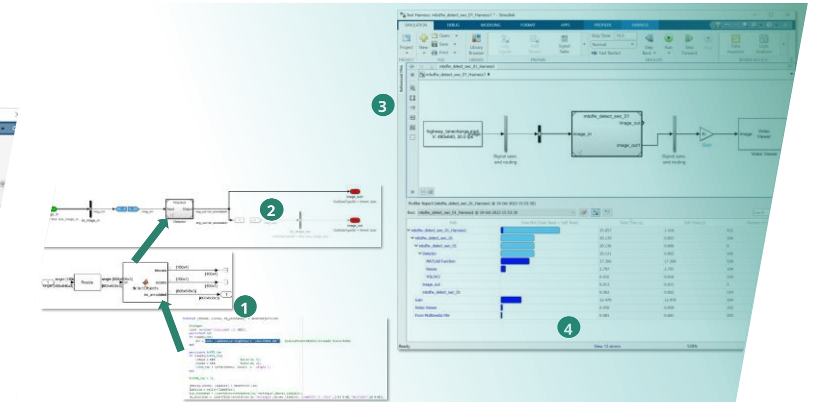
Do you have some examples?

© Elektrobit 2024 | Public

SW performance analysis methodology







ML/DL function based on TinyML

This examples shows 2 aspects:

- How to natively embedded TF-Lite (TinyML) in the overall workflow?
- How to do a performance anaylsis?

System performance analysis

In this example a role-based system component is designed, and its performance evaluated

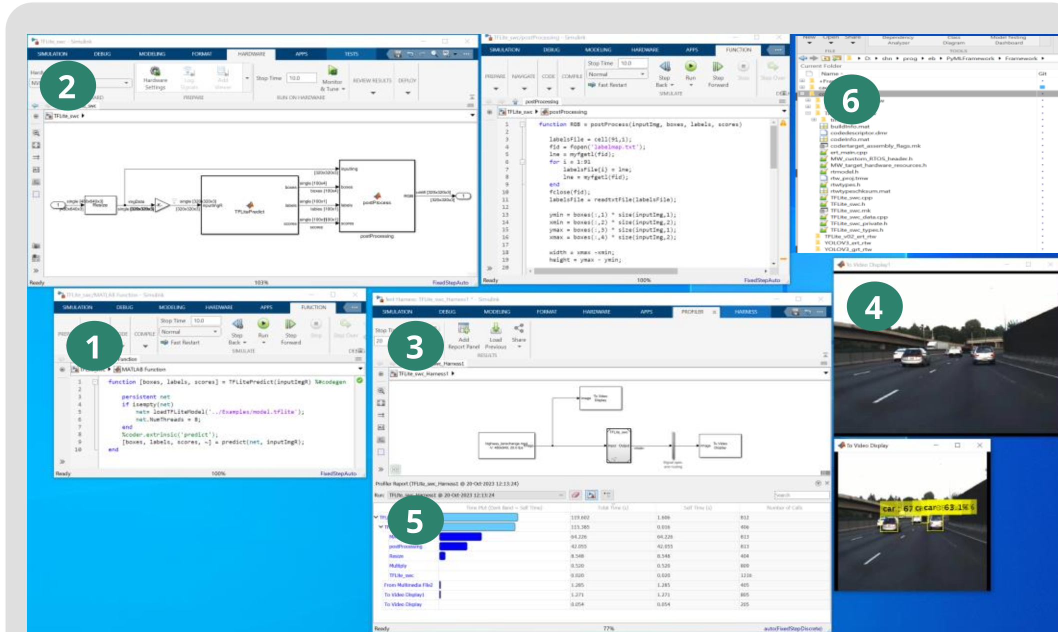
Finally, codes for different platforms is generated and deployed

Cloud-based development

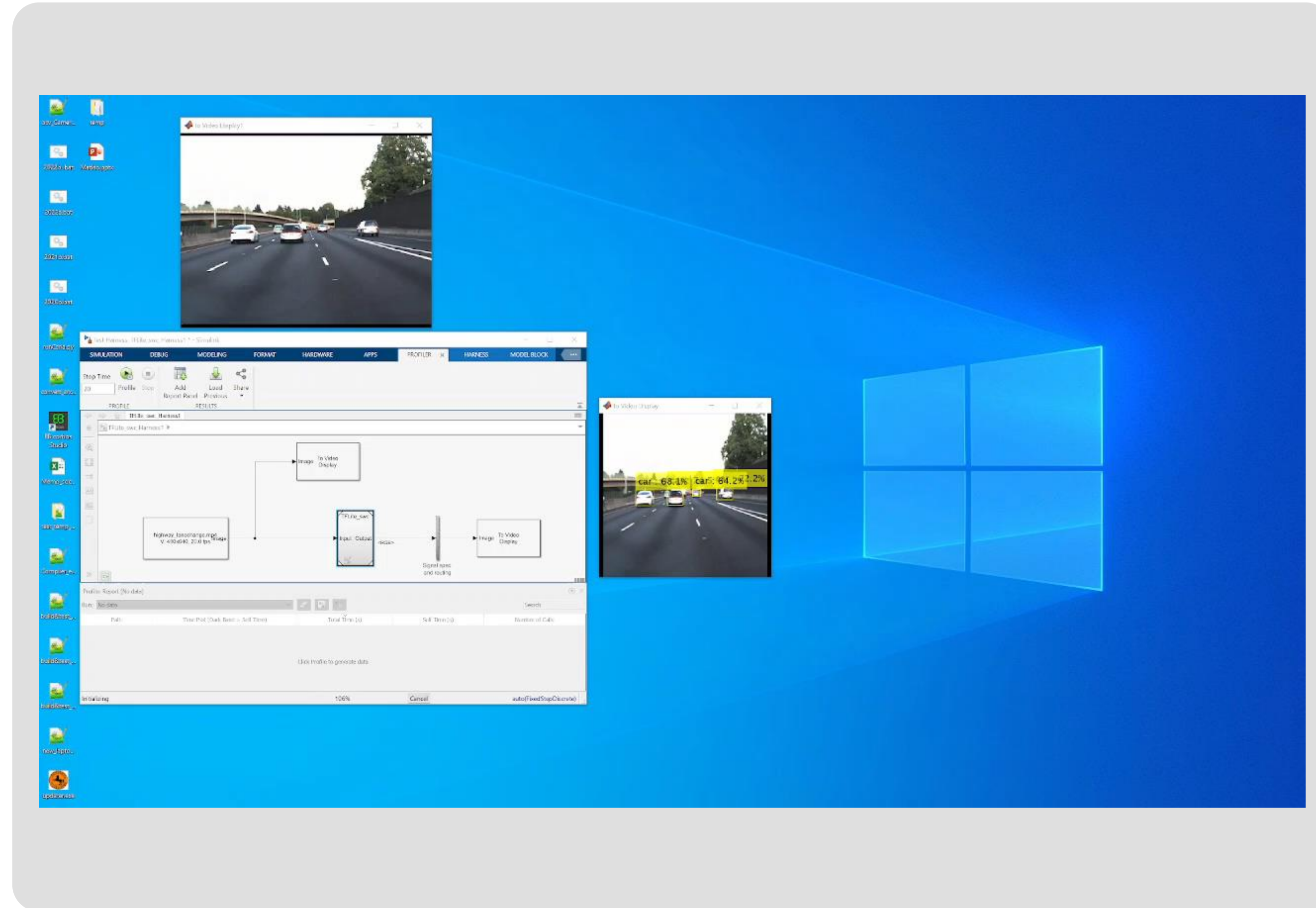
A typical example for a cloud-based development: Autosar adaptive SWC designed with MATLAB/Simulink

© Elektrobit 2024 | Public

Example: ML/DL function based on TinyML



1. TF-Lite (TinyML) object detection and classification
2. System model
3. Test Harness
4. Simulation output
5. Performance analysis
6. Generated code for Jetson Nano



Example: system performance analysis

1. Measure task with performance measurements
2. Evaluate performance with advisor
3. Generate code and compile for different platforms like host, embedded, or HW (FPGA)

Example: cloud-based adaptive AUTOSAR SWC

1. Cloud development framework
2. Generated workspace
3. Docker container setup
4. Use MATLAB/Simulink for code generation
5. Built MATLAB/Simulink Autosar adaptive SWC
6. Virtual deployment and execution on EB linux + Qemu both build for ARM64. Execute the App!

05

What are the conclusions? What's next?

Key take aways

At system definition it is key to **ensure that a new SW feature can be deployed and executed on an existing or new SoC**

Our **End-2-End framework for SDV enables a shift left of the development** through virtualization of the design from systematization, development, up to test of functional and non-functional requirements and integration, and from cloud to SoC

Realizing the essence of SDV by using Elektrobits framework in combination **MATLAB/Simlink** from exploration of innovative ideas, development of software functions, integration, and verification, seamlessly and fast in one framework and consistently form cloud to SoC, both virtual and real.

© Elektrobit 2024 | Public

What's next?

Enhance analysis capabilities

Add further analysis methods like power analysis support

Integration of full vECU models

Add support of full test automation based on Elektrobit's automated test tool for AUTOSAR and EB Assist

Enable full cloud capabilities

Add support of/for further cloud solution including Continental CAEdge (Classic ASR, adaptive ASR, other OS)

Add customizable cloud support enabling flexible usage in various environments, e.g., Continental CAEdge, or customer-owned environments

© Elektrobit 2024 | Public

Contact us

Thomas Kleinhenz

Director, EB-EST-CMS
Elektrobit – Our software moves the world

thomas.kleinhenz@elektrobit.com
elektrobit.com

© Elektrobit 2024 | Public