

E-Mobility and Software Model-Based Design for SDV

By Erick Saldana Sanvicente
October 2026

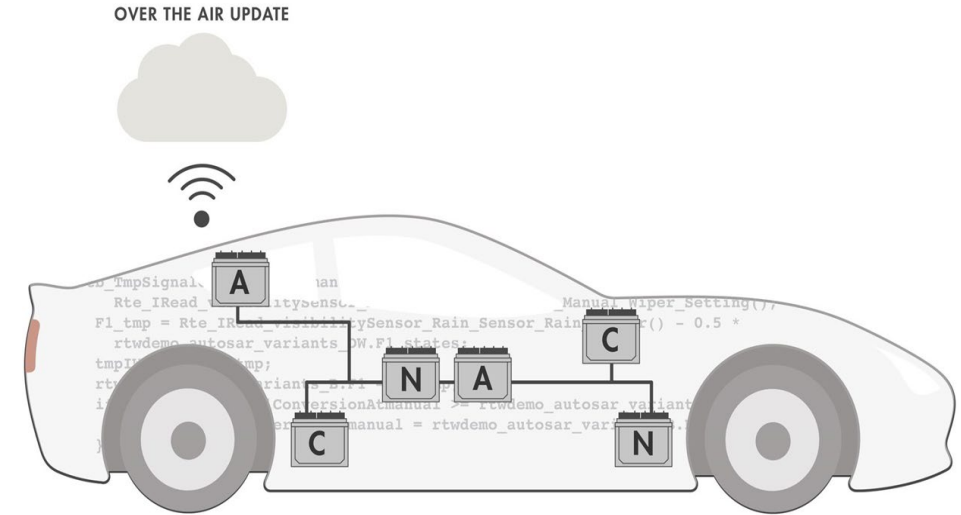
Software-Defined Vehicle

Connected vehicles for a connected world

- The value of a vehicle is increasingly related to the capabilities its software can deliver
- Manufacturer focus is shifting from improving increasing efficiency to improving user experience

For the user

- A vehicle that can have features added and updated, throughout its life
- My vehicle is tailored for my needs
- It can potentially learn from my behavior and usage

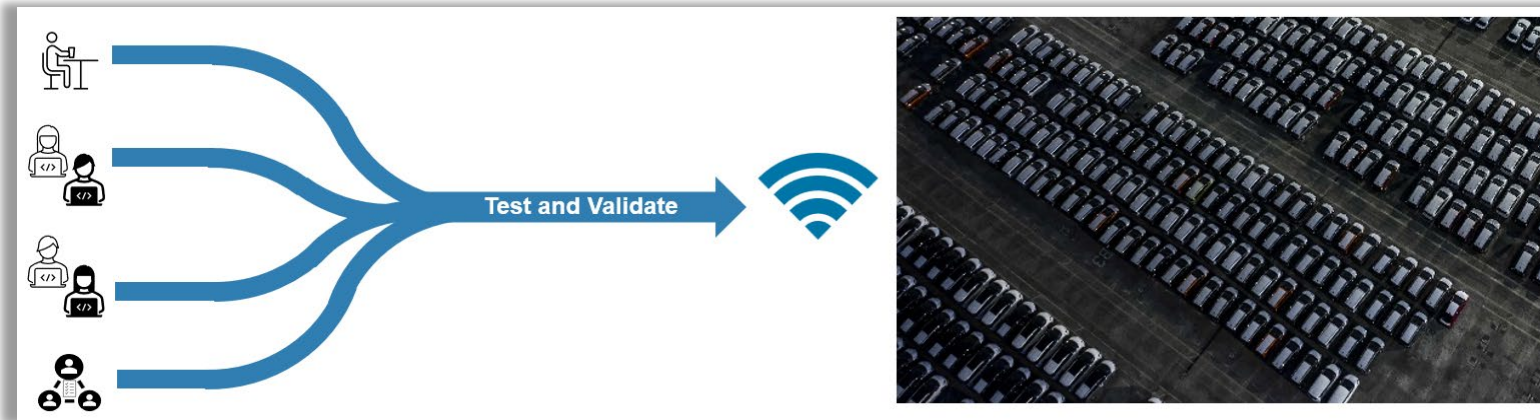


For the maker

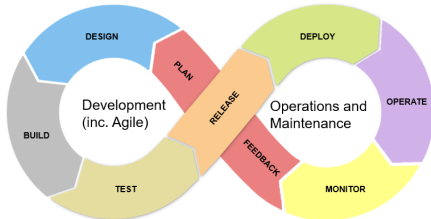
- Build a modular and scalable software platform
- Use the vehicle data creatively to refine the vehicle design
- Over The Air updates of the software – opportunity to provide new features and bug fixes without re-calls

Motivation

Towards SDV – Addressing the development challenges

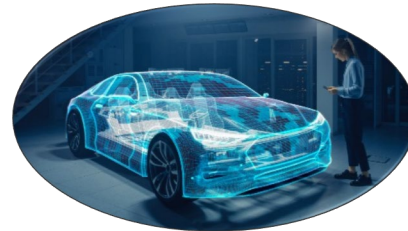


Develop faster without compromising on software quality?



Adopt DevOps workflows and best practices

MathWorks tools enable users to develop while following DevOps best practices



Virtualize systems for analysis and testing

MathWorks together with its partners enables testing in MiL, SiL **and** Virtual PiL

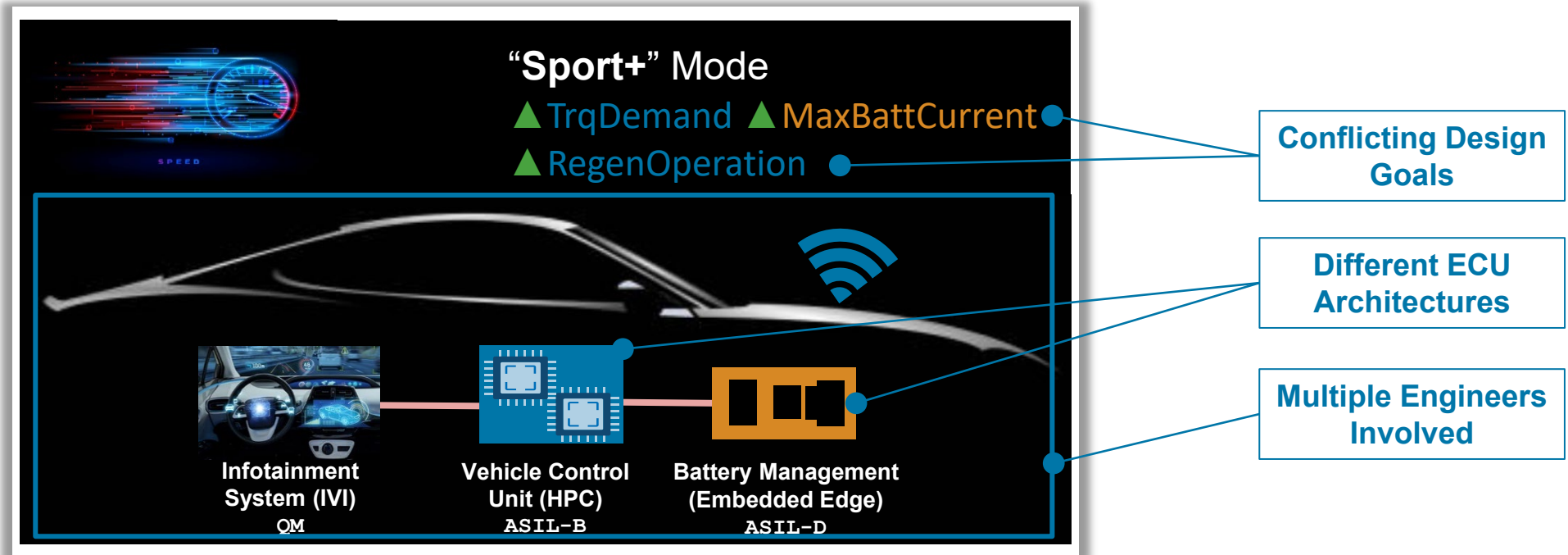


Automate and scale on the cloud

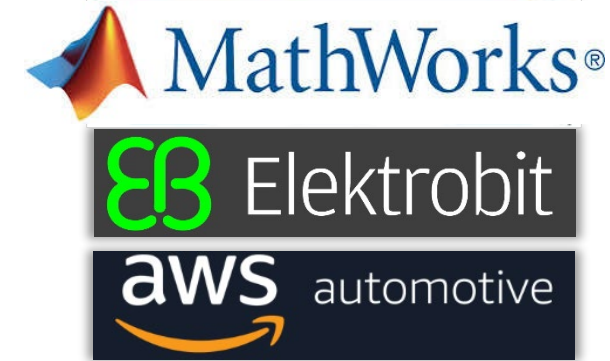
MATLAB and Simulink can be deployed to cloud environments for analysis and testing work

SDV Development in Action

Can we deliver this upgrade without hardware access?



SDV Development in Action



SDV Development in Action

Focus 1: System Level Studies

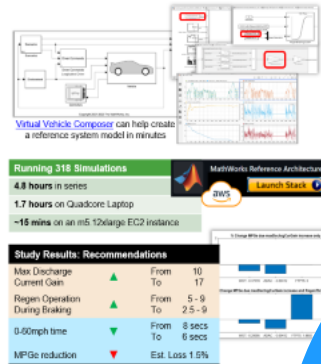
Challenge:

- Identify software parameters we can vary to enhance vehicle performance
- Quantify the effect of changing these parameters of at a component and system level

Study goals	Variables
0-60 mph in <6 seconds	Increased Max Battery Current Lim
Quantity Loss in MPGe	Increased Regen Operation Limits
Quantity gains from regen in different driving conditions	

Solution:

- Virtualize** Build a system level model with detailed component level models where required
- Scale** Scale up simulation studies using parsim on the cloud



SDV Development in Action

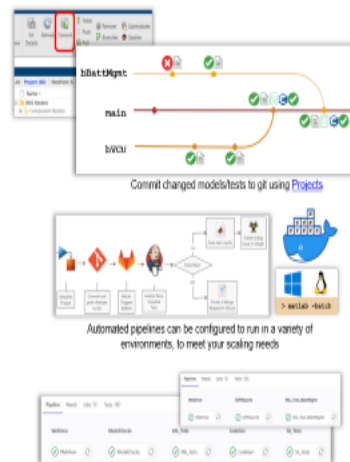
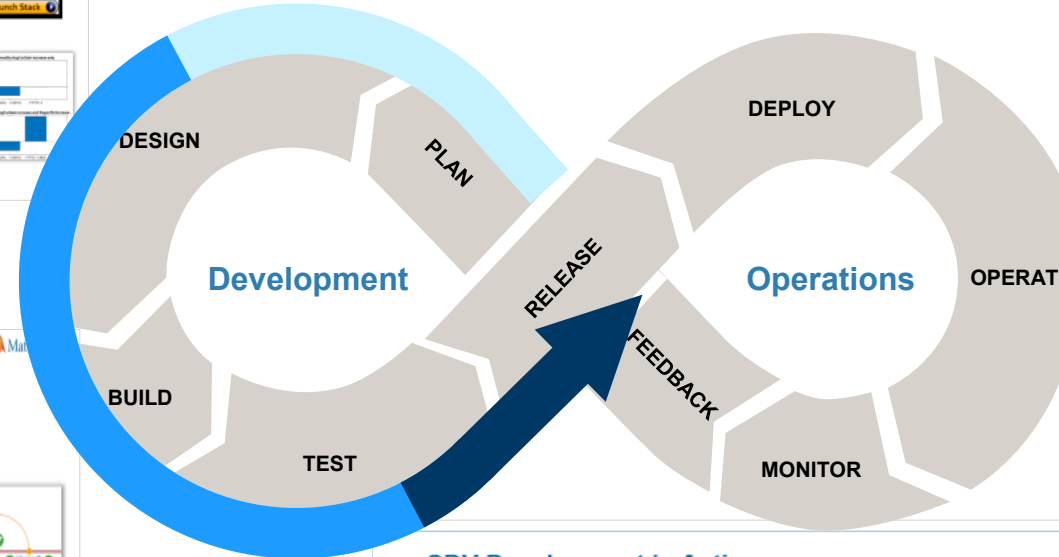
Focus 2: Automated testing and codegen via Continuous Integration

Challenge:

- Enable multiple engineers to simultaneously develop features in parallel, but can share and sync with the other colleagues
- Test application code while still in development, thereby creating a "fix-as-you-go" workflow
- Automate large scale testing and code generation when development branches are merged to main/release branch

Solution:

- Virtualize** Componentize models and place them under source control
- Test** Test at model level, application software level, and conduct MISRA C checks
- Scale** Setup non-interactive MATLAB on runner computer(s); and perform automated tests, codeGen, and report authoring tasks



SDV Development in Action

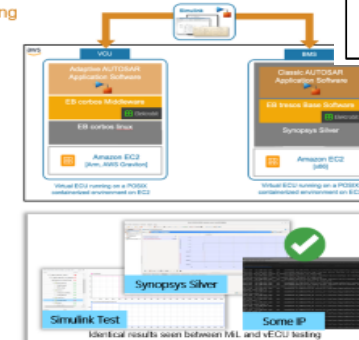
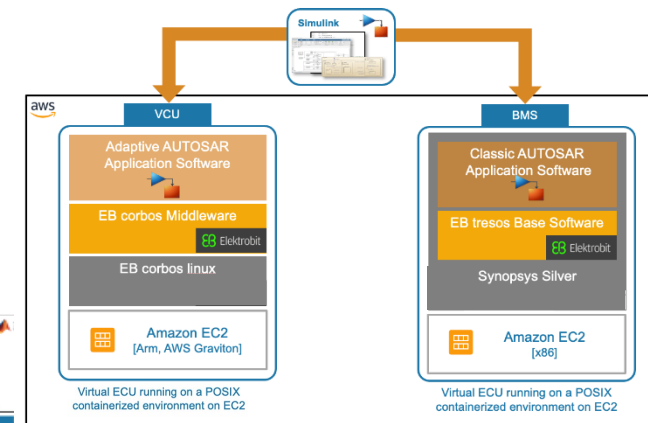
Focus 3: Virtual ECUs deployment and testing

Challenge:

- Ensure application software can integrate into a variety of environments, depending on hardware architecture & middleware
- Verify model based AUTOSAR applications on "road ready" middleware in the cloud without target hardware

Solution:

- Customize application code interface for specific middleware/hardware
 - Classic AUTOSAR
 - Adaptive AUTOSAR / DDS / ROS
- Integrate application code with underlying middleware to deploy within virtual ECUs
- Run virtual ECUs in a cloud native environment and verify the vECU is equivalent to MIL/SIL tests



SDV Development in Action

Focus 1: System Level Studies

Challenge:

- Identify software parameters we can vary to enhance vehicle performance
- Quantify the effect of changing these parameters of at a component and system level

Study goals	Variables
0-60 mph in <6 seconds	Increased Max Battery Current Lim
Quantify Loss in MPGe	Increased Regen Operation Limits
Quantify gains from regen in different driving conditions	

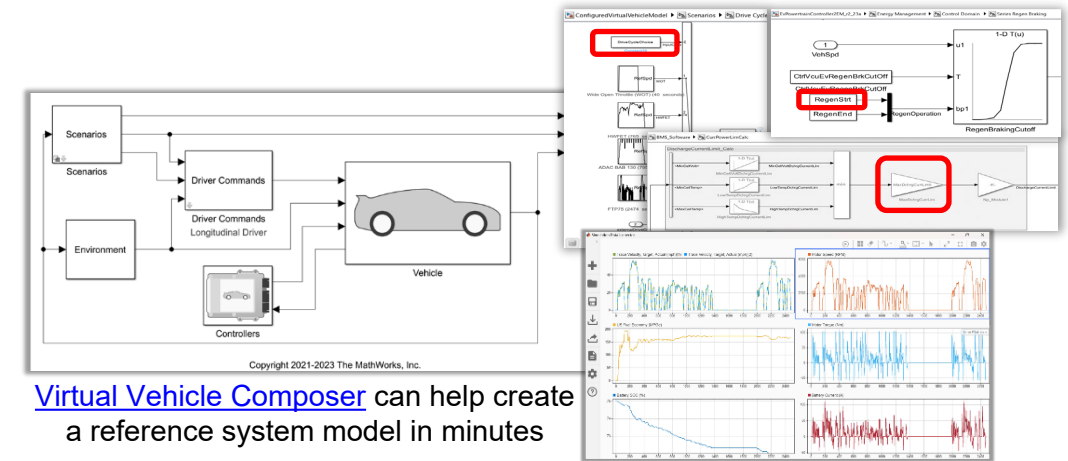
Solution:

Virtualize

Build a system level model with detailed component level models where required

Scale

Scale up simulation studies using `parsim` on the cloud



[Virtual Vehicle Composer](#) can help create a reference system model in minutes

Running 318 Simulations

4.8 hours in series

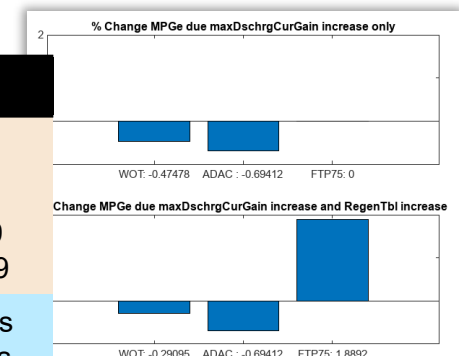
1.7 hours on Quadcore Laptop

~15 mins on an m5.12xlarge EC2 instance



Study Results: Recommendations

Max Discharge Current Gain	▲	From 10 To 17
Regen Operation During Braking	▲	From 5 - 9 To 2.5 - 9
0-60mph time	▼	From 8 secs To 6 secs
MPGe reduction	▼	Est. Loss 1.5%



SDV Development in Action



Focus 2: Automated testing and codegen via Continuous Integration

Challenge:

- Enable multiple engineers to simultaneously develop features in parallel, but can share and sync with the other colleagues
- Test application code while still in development, thereby creating a “fix-as-you-go” workflow
- Automate large scale testing and code generation when development branches are merged to main/release branch

Solution:

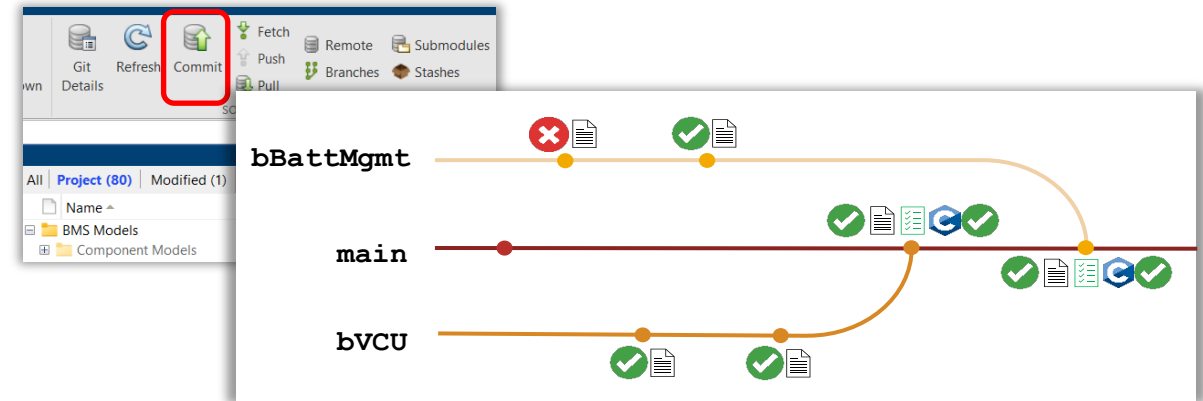
Virtualize

Componentize models and place them under source control

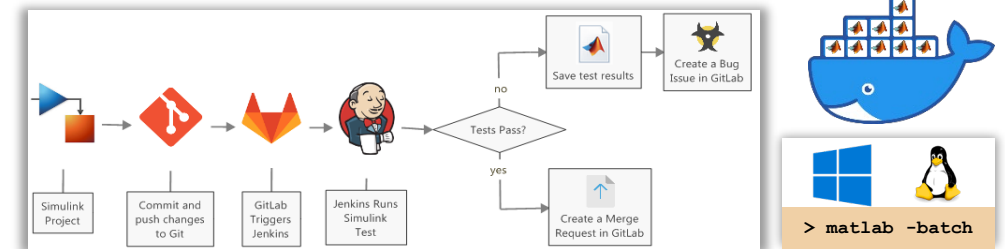
- Test at model level, application software level, and conduct MISRA C checks

Scale

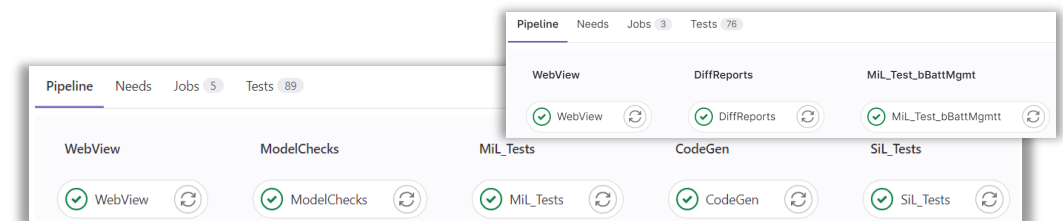
Setup non-interactive MATLAB on runner computer(s); and perform automated tests, codeGen, and report authoring tasks



Commit changed models/tests to git using [Projects](#)



Automated pipelines can be configured to run in a variety of environments, to meet your scaling needs



SDV Development in Action

Focus 3: Virtual ECUs deployment and testing

Challenge:

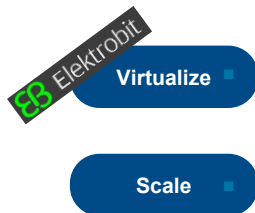
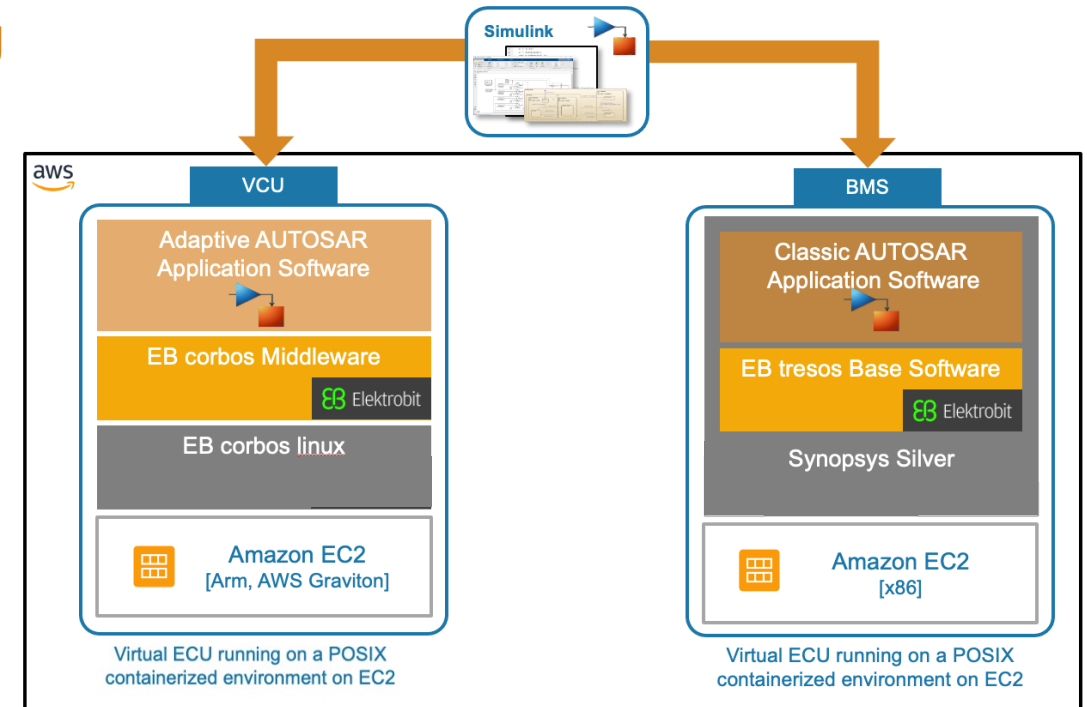
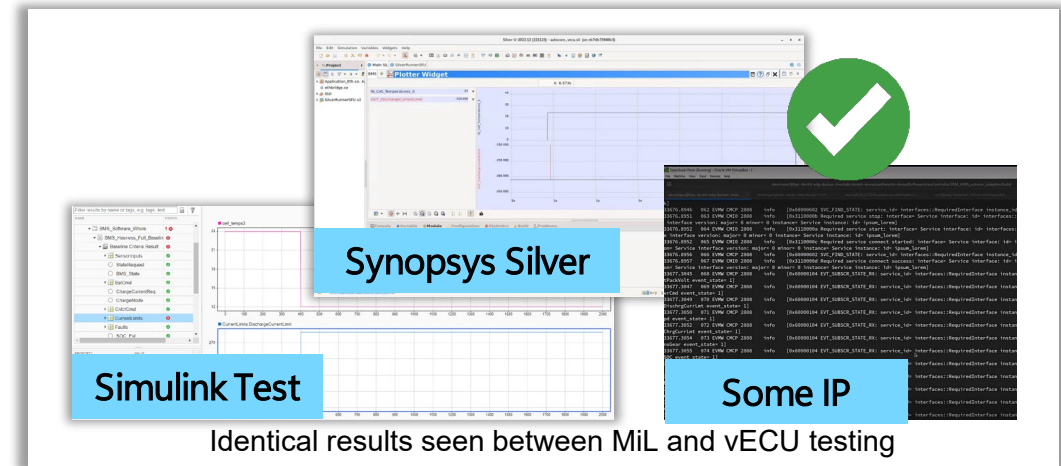
- Ensure application software can integrate into a variety of environments, depending on hardware architecture & middleware
- Verify model based AUTOSAR applications on 'road ready' middleware in the cloud without target hardware

Solution:

- Customize application code interface for specific middleware/hardware
 - Classic AUTOSAR
 - Adaptive AUTOSAR / DDS / ROS

Integrate application code with underlying middleware to deploy within virtual ECUs

Run virtual ECUs in a cloud native environment and verify the vECU is equivalent to MiL/SiL tests

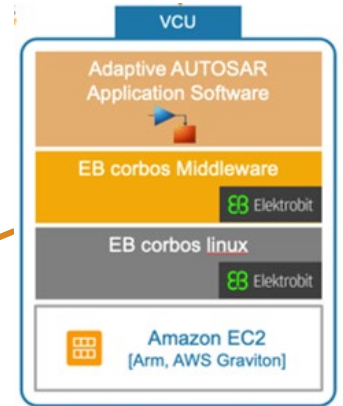


Two virtual ECUs deployment and testing

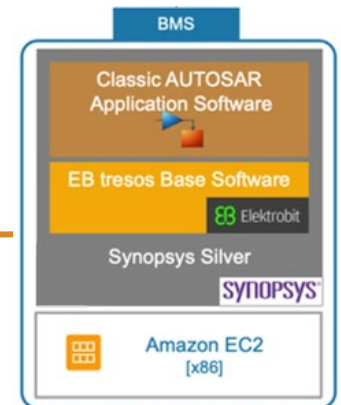


Application code
generated from the
models

Inter ECU
Communication
(via SOME/IP)



Virtual ECU running on a POSIX containerized environment on EC2



Virtual ECU running on a POSIX containerized environment on EC2

Two virtual ECUs deployment and testing



Thank you!