Accelerating Powertrain Software Validation: From SiL to Testbed

FPC2026 - AI & SDV Session

Marijn Hollander

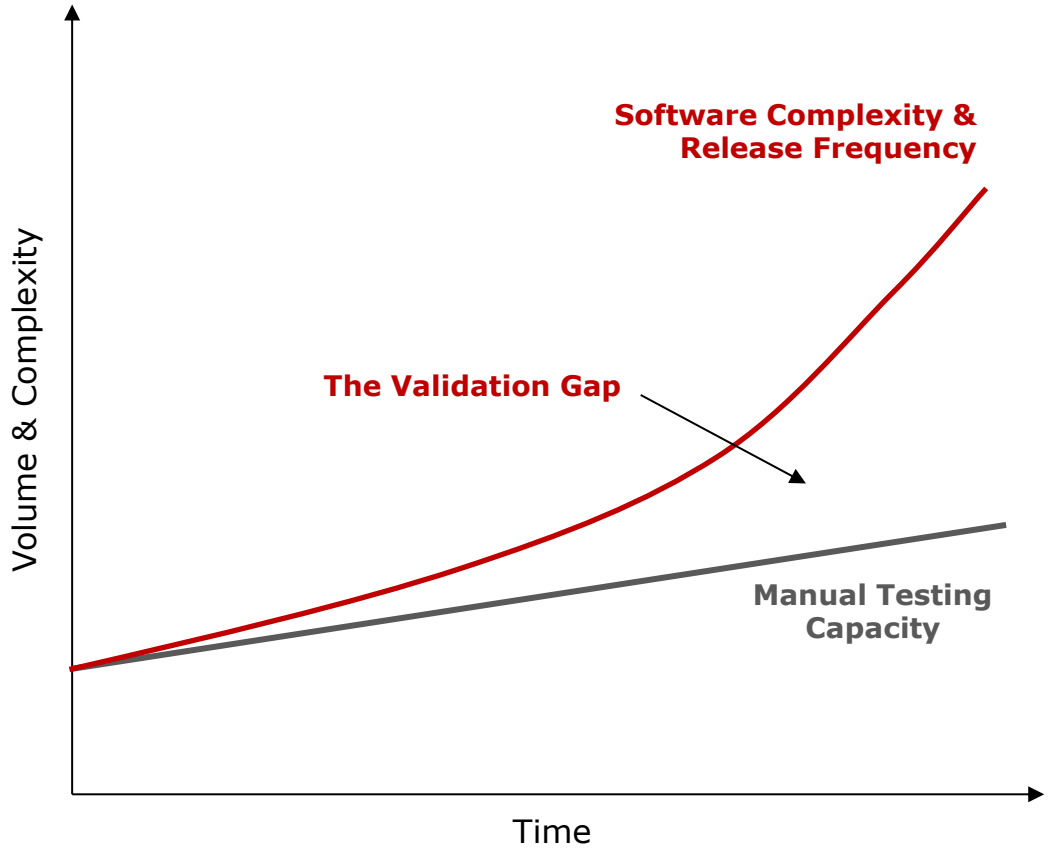
The Validation Gap:
Where Agile Speed Meets Physical Bottlenecks

The Reality:

- Development teams now work in agile sprint with bi-weekly release cycles.
- The demand for software/functional tests is increasing.
- Testing the software together with the system

The Problem: Traditional testing capacity is linear and mostly manual. It cannot keep pace with the growth

The Consequence: Testing with full coverage is becoming impossible, leading to delayed releases or compromise quality



The Human Costs of Disconnected Workflows

The System Engineer

The Goal: Ensure all requirements in the ALM system (e.g., Codebeamer) are met.

The Pain: Traceability is lost in manual handoffs. E.g. I don't know if the test ran against the right dataset

The Calibration Engineer

The Goal: Manage and calibrate dataset for vehicle variants and validate

The Pain: Often excluded from early SiL testing (due to complexity); must wait for physical hardware availability
I must manually flash ECUs

The Virtual Testing Expert

The Goal: Configure the Virtual Configurations and Testing Environments (SiL, HiL, Testbed)

The Pain: I spend my day manually configuring SiL or HiL environments

Quantifying the Manual Burden

Valuable engineering time is being consumed by overhead

Generation/creation of a gateway model, project configuration, plant model interfaces for a dSpace HiL can take **6 days**

- **OEM1: 60% of whole process** only for test case implementation
- **OEM2: ~75** supplier **engineers** for testcase implementation/maintenance
- Scripting of uncontrolled testcases and low standardization

Assuming 30 min per flash
- 1 flash per month = 6 hours per year
- 1 flash per day = **8 days per year**

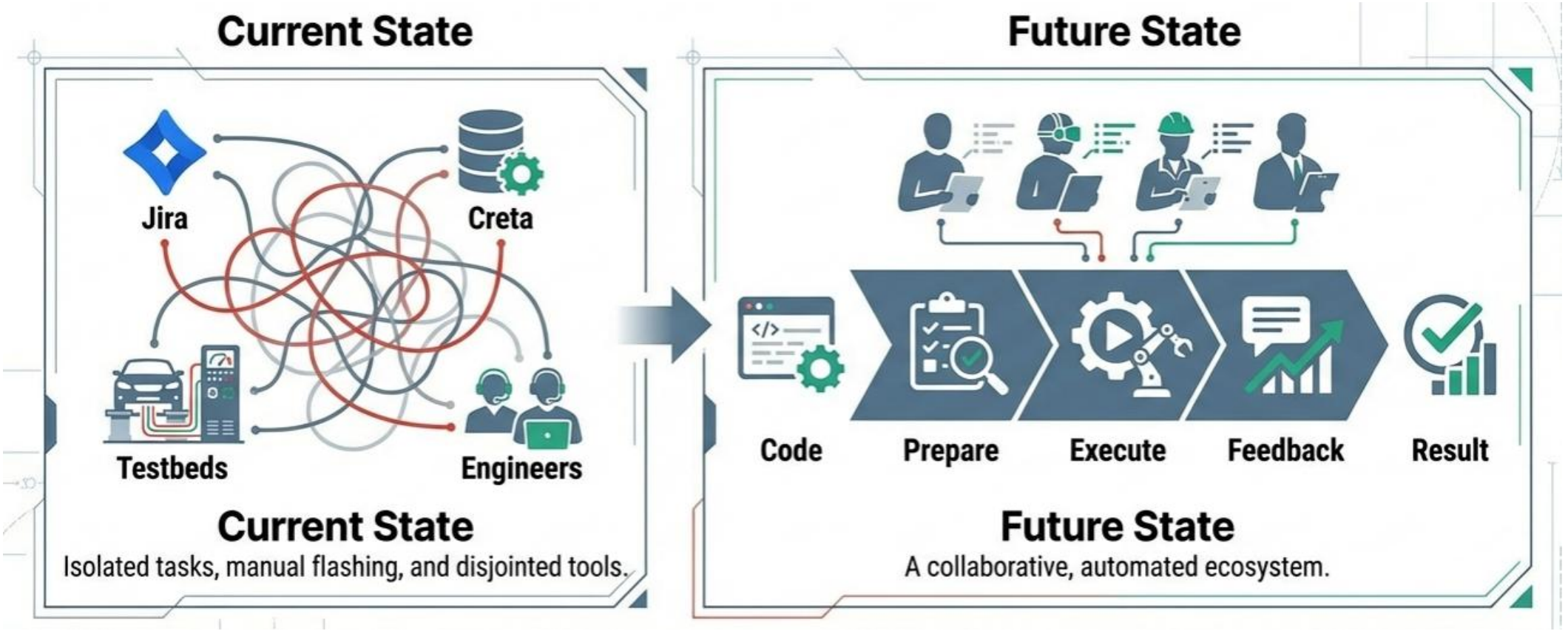
for 1 test environment

State of art the only task that is automated

Test environement is prepared	test cases exist in ALM	(latest) SW files are available	Define Virtualization	get latest SW version & dataset to be tested	translate test cases into create executable tests or test sequences	check if test environment is ready	pull artifacts	(optional) flashing *	execute test	upload results to server / repository	check results	send results to ALM
Prerequisites			Test (Job) Preparation			Test (Job) Execution					Post-processing	

Bottom Line: As software releases scale, this manual overhead causes unacceptable downtime.

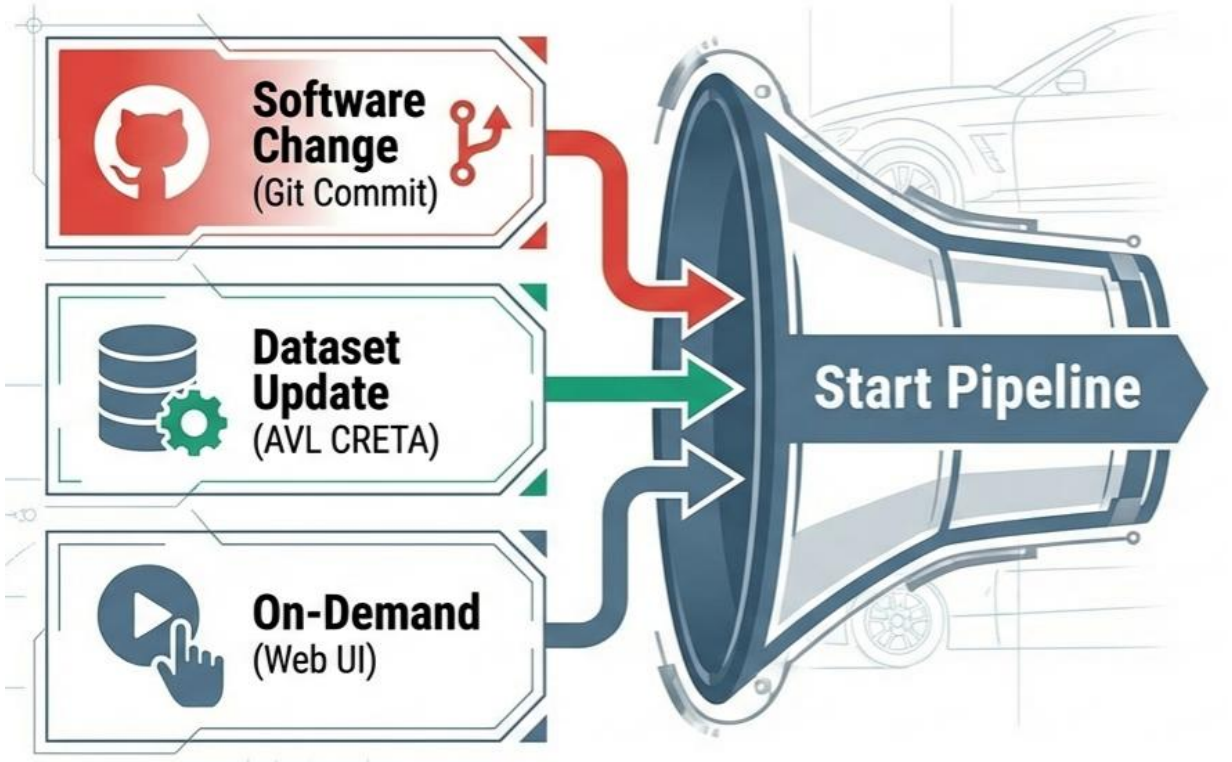
The Goal: A Software Defined Test Factory



The Vision: To align processes and tools to gain speed and obtain full traceability

Step 1: The Trigger

Automation starts the moment a change is made



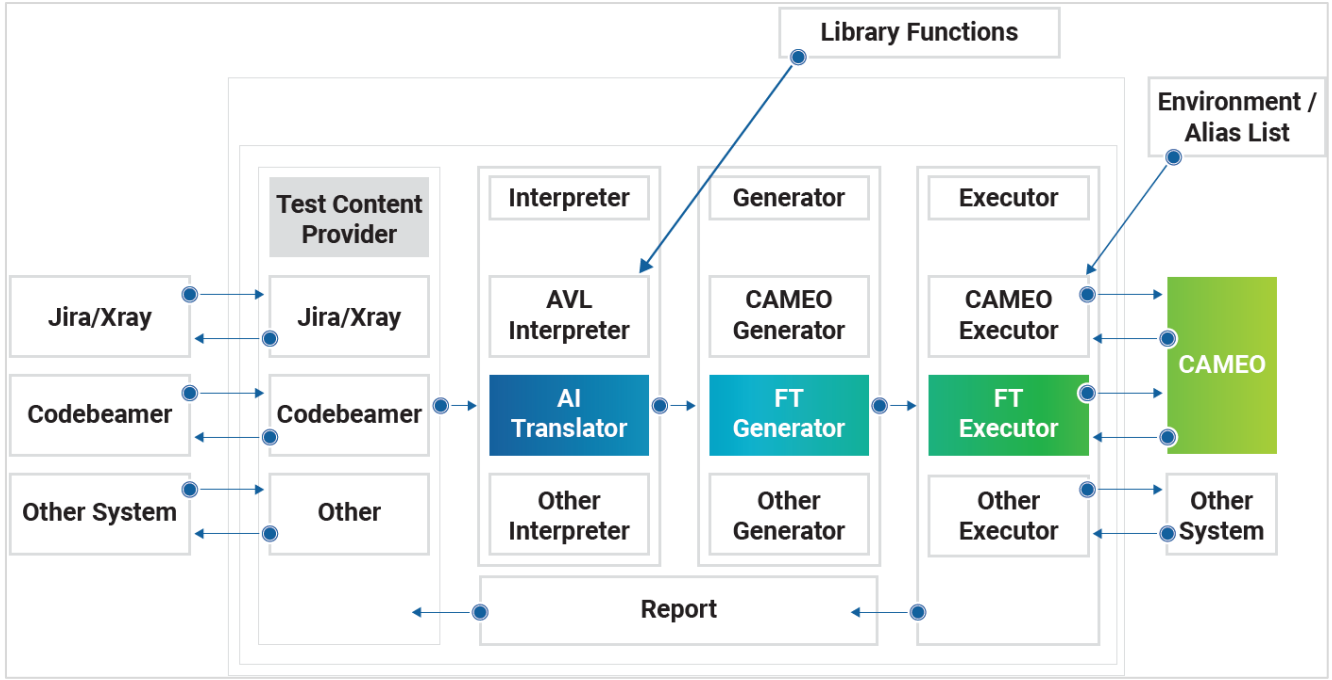
The journey begins automatically via standard development events:

- **A Software Change:** A new software is released to a Git repository
- **A Dataset Update:** A calibration engineer merges a new dataset in AVL Creta
- **On-Demand:** A user manually triggers a specific job via the Web UI

Step 2: Prepare & Translate

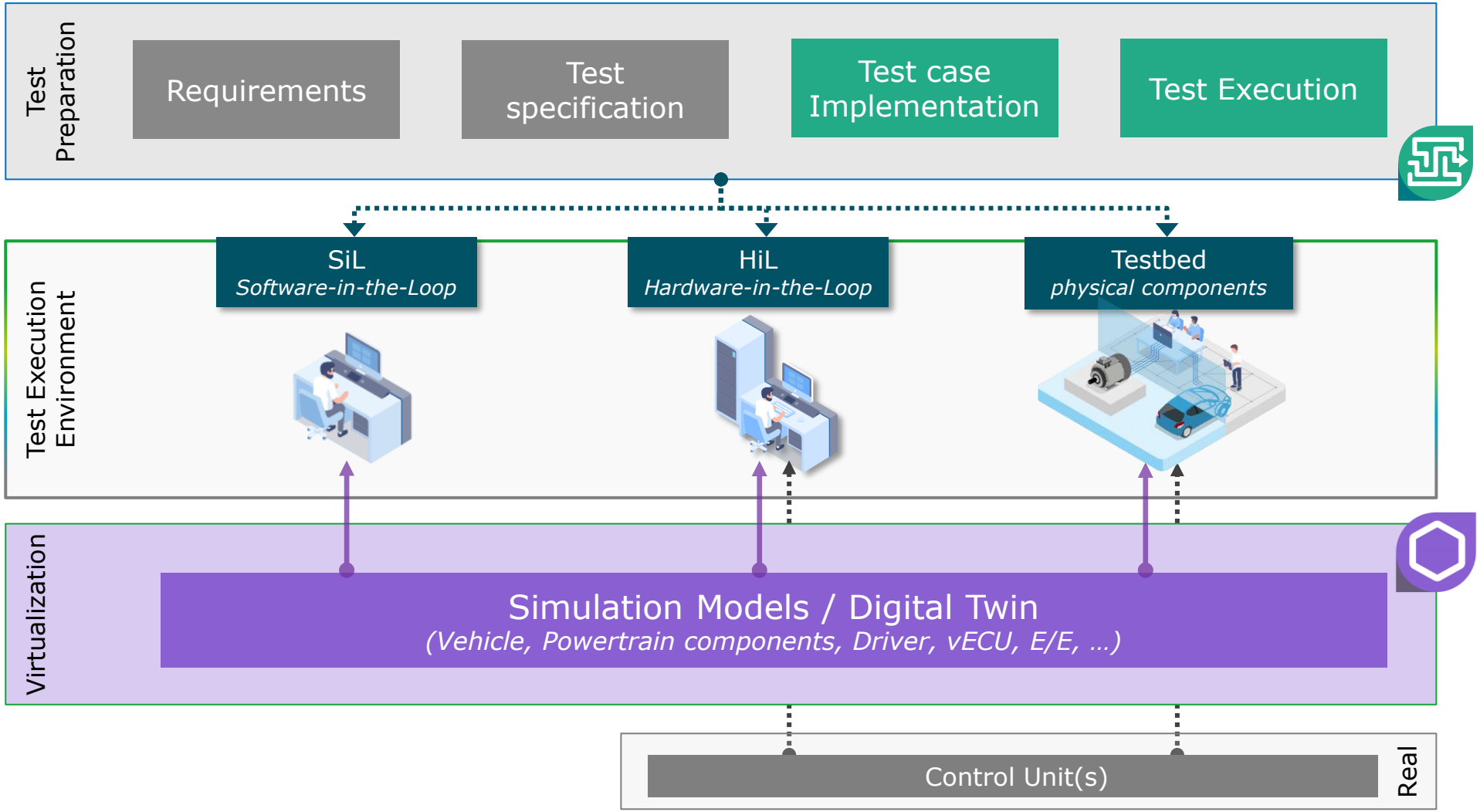
From human requirements to executable scripts-automatically

- **Interfaces to ALM/PLM** trigger on demand AI supported test translation and execution via CI/CD pipelines
- AI driven test generation and test modification is **robust against different language, phrasing and flavors** of requirements
- AVL CAMEO supports test execution **seamless over different test environments**
- New or changed requirements will reflect in **testing within minutes** and provide most relevant feedback to engineers



New or changed requirements reflect in testing within minutes

Seamless Virtualization & Seamless Testing

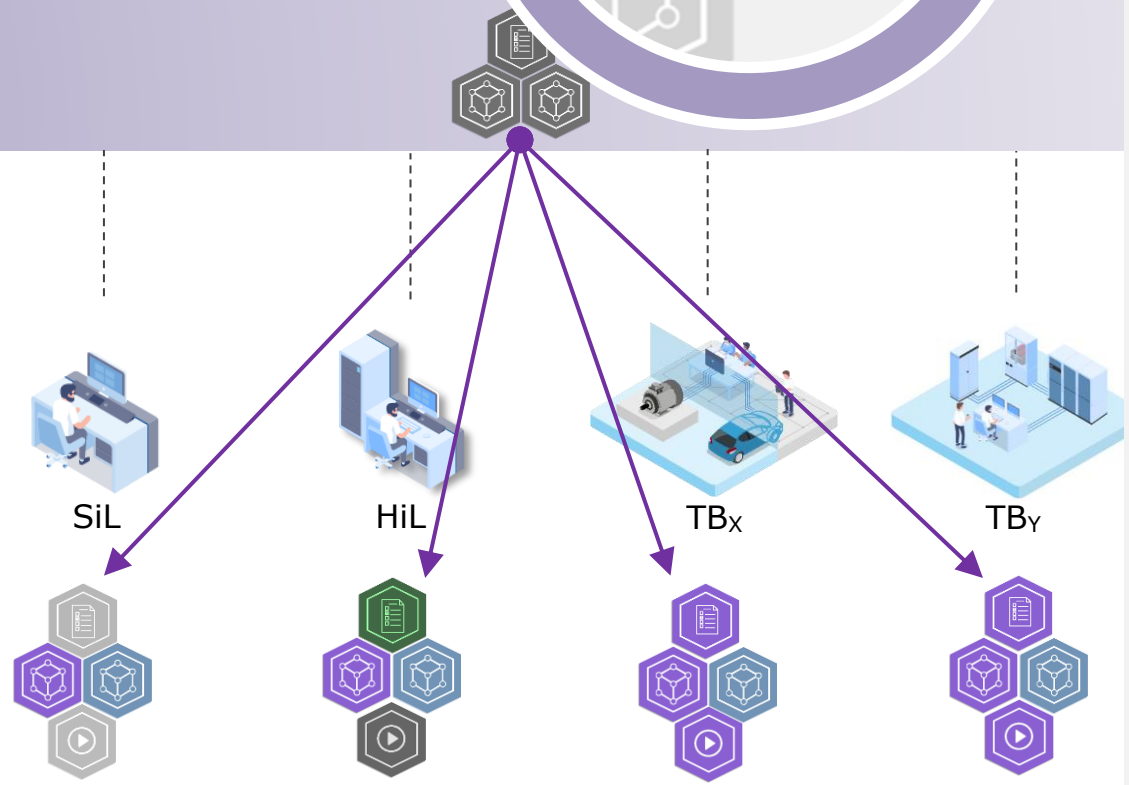


Seamless Virtualization Meta-Model Concept

Neutral configuration layer (meta-model) in **.yaml** file format.

Human readable without specific tools (text editor is sufficient).

Abstracted meta-model: neutral configuration



KEY VALUES

Create HiL Configuration from MiL project

- No repetitive work for different targets

Advance in the Configuration process with low effort

- Move from MiL to HiL faster than ever before
- Utilize advanced algorithms / AI for model connection

Be platform independent

- Reuse your simulation projects and configurations

Examples of heterogenous tool landscape

Seamless Virtualization
Automated HiL Configuration

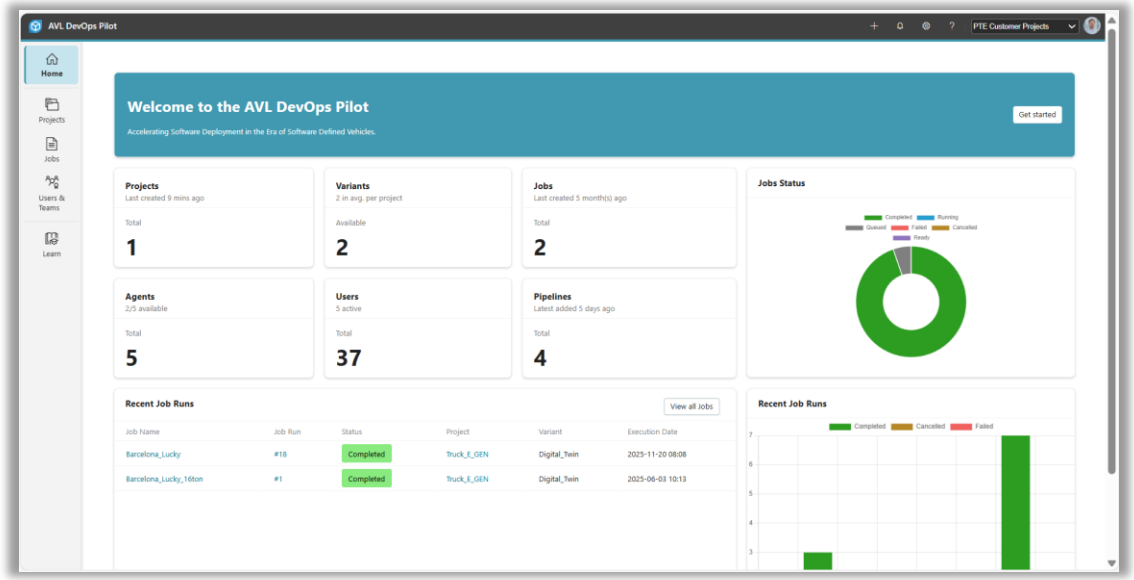
Example HIL Project – @dSPACE SCALEXIO HIL
HEV domain with Engine and Powertrain ECUs

	Manual Workflow		Automated Workflow
Platform independent ***gateway model***	Gateway model manually created for each ECU pin (HW, communication, plant model layer).	***~1-2 days*** → ***1-2 hours***	***Auto-generated*** *I/O model,* interfacing plant models and multiple ECUs; containing function specific subsystems (e.g. Analog, Digital, SENT, CAN, LIN).
HIL platform specific ***project configuration***	Each configuration step handled individually for ECU topology, HW & communication functions, and models.	***~2-3 days*** → ***1-2 days***	***Auto-creation*** of HIL-specific project, based on user configuration & communication files, HW topology, simulation models. ***Auto-configuration*** from ECU pin to HIL module; mapping logical signal chain.
Integration of ***plant model interfaces***	Plant model I/F manually linked to other models, HW I/O ports, and communication modules. Up to thousands of connections.	***~1-2 days*** → ***1-2 hours***	***Auto-linking*** of several plant models with respective interfaces (Model-to-Model communication)

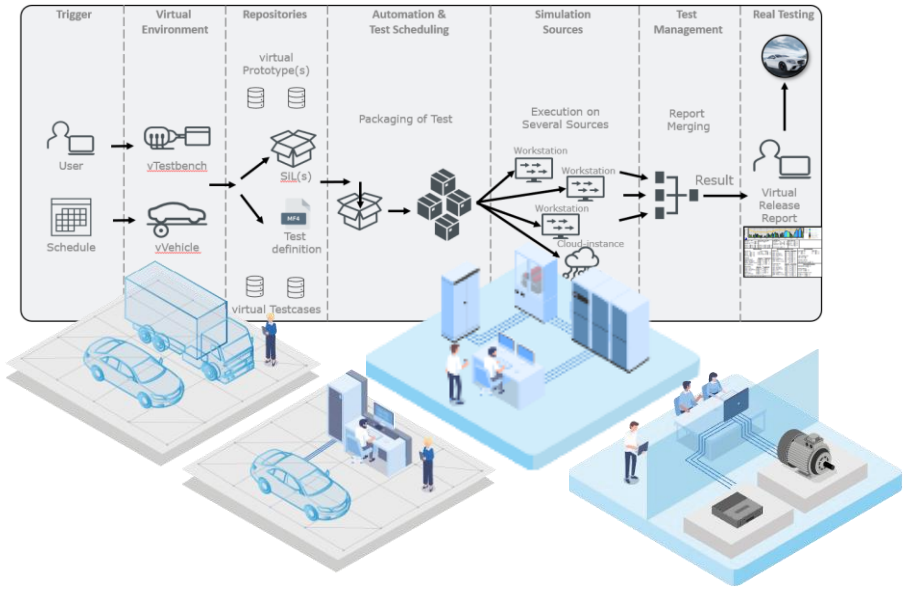
~40%
cost & time saving, overall

Indicative numbers
Based on reference projects

Step 3: Automated Execution



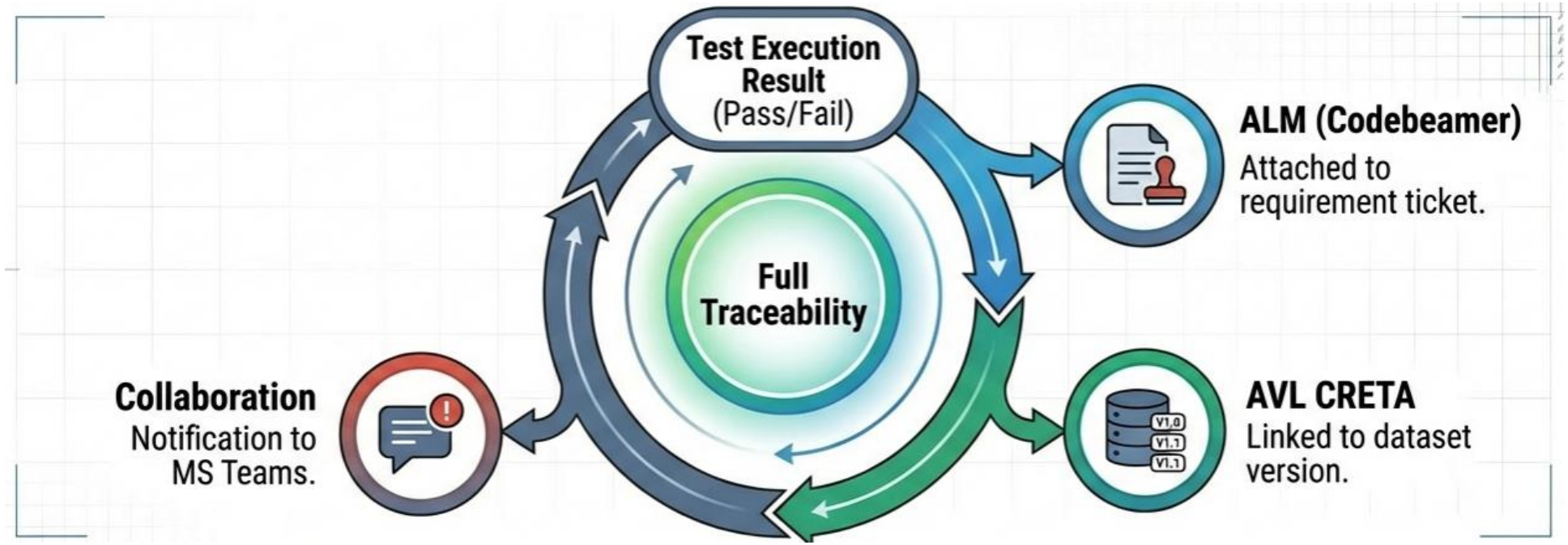
The Office



The Physical and Virtual Factory

The engineer sits in the office while the test environment configures itself and runs the test sequence with out human intervention.

Step 4: Closing the loop

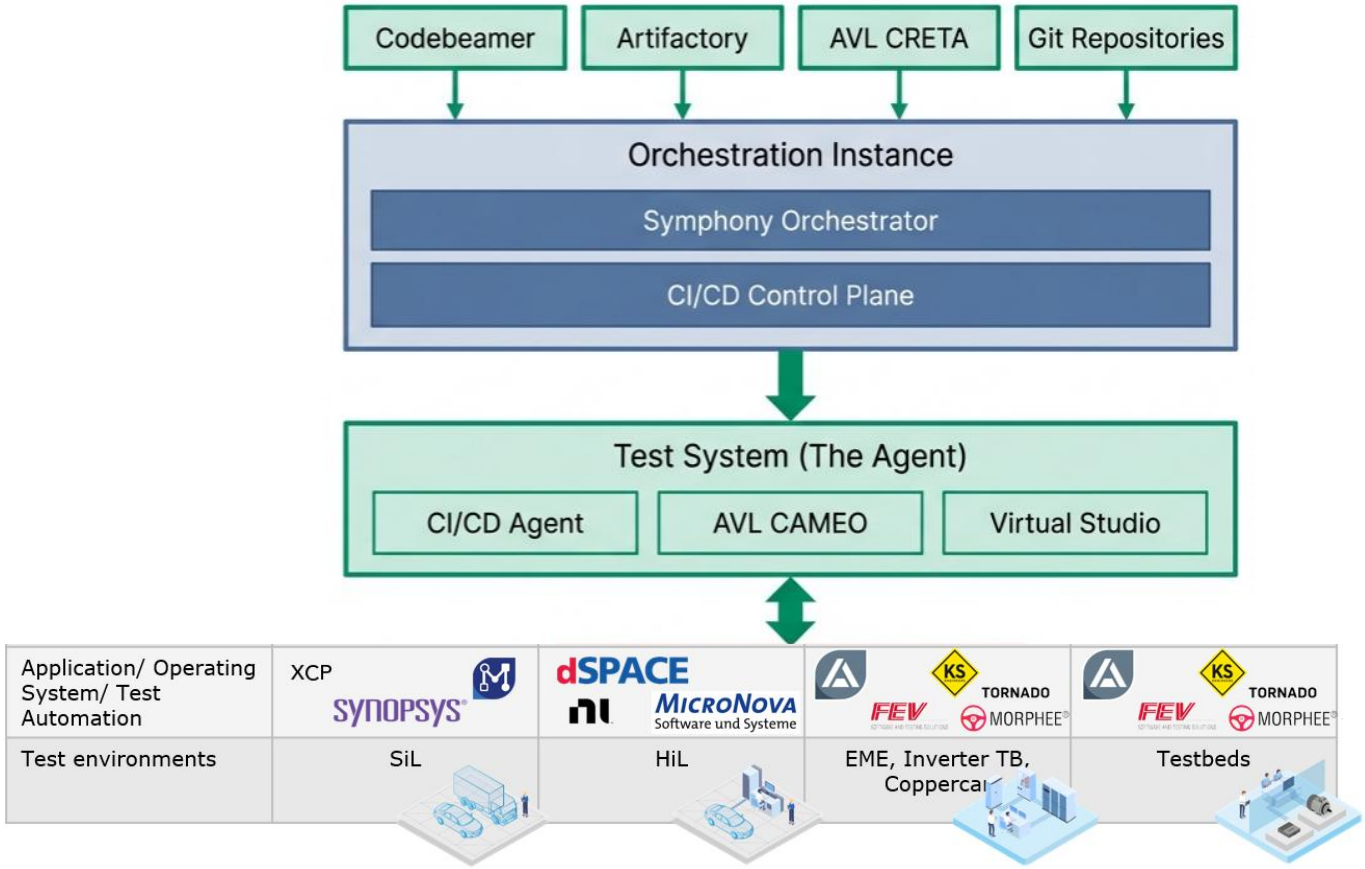


Result are distributed immediately, ensuring that every test in linked back to the specific requirement and dataset that triggered it.

The Orchestrator: AVL DevOpsPilot

End-to-end orchestration of artifacts, simulation models, controller models (vCUs) and testcases.

- Test environments are automatically configured and orchestrated.
- End-to-end traceability of all artifacts and results.
- Integration with AVL Creta
- Neutral configuration layer for SiL and HiL test environments
- Focus on the end-user (test/calibration engineers) via easy-to-use GUI (no experience with coding or CI/CD toolchains needed).
  - Job configurations
  - Environment configuration
  - Auto trigger jobs
  - Re-run jobs (reproduce tests)
  - Result/report upload to different locations

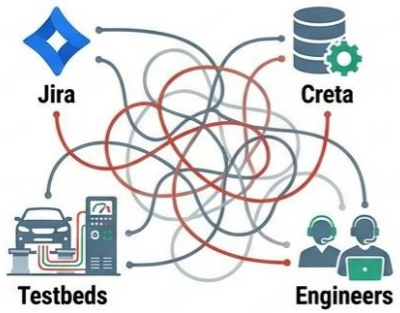


Summary
Focus on engineering instead of configuration

Same toolchain supports heterogeneous test environments

Full traceability of artifacts and results

Reduction of overhead and downtimes by automation of procedures, workflow management and orchestration of test environments



Enablement of calibration engineers and non-devops engineers by easy-to-use UI

Thank you

www.avl.com