ecu.test

Product brochure

SOFTWARE TESTING. SIMPLIFIED.

Software development is a complex process in which each new feature requires many test steps. The more complex the software, the more extensive testing is required.

ecu.test is our software for complex functional and system tests of ECU software. The tool is used to create test cases for automotive software in all development stages up to the final vehicle and to execute them automatically in different test environments.

Our customized solutions dynamically scale with test volume – from single-user systems to parallel execution on distributed test systems in internal clusters and public clouds.

ecu.test

key features

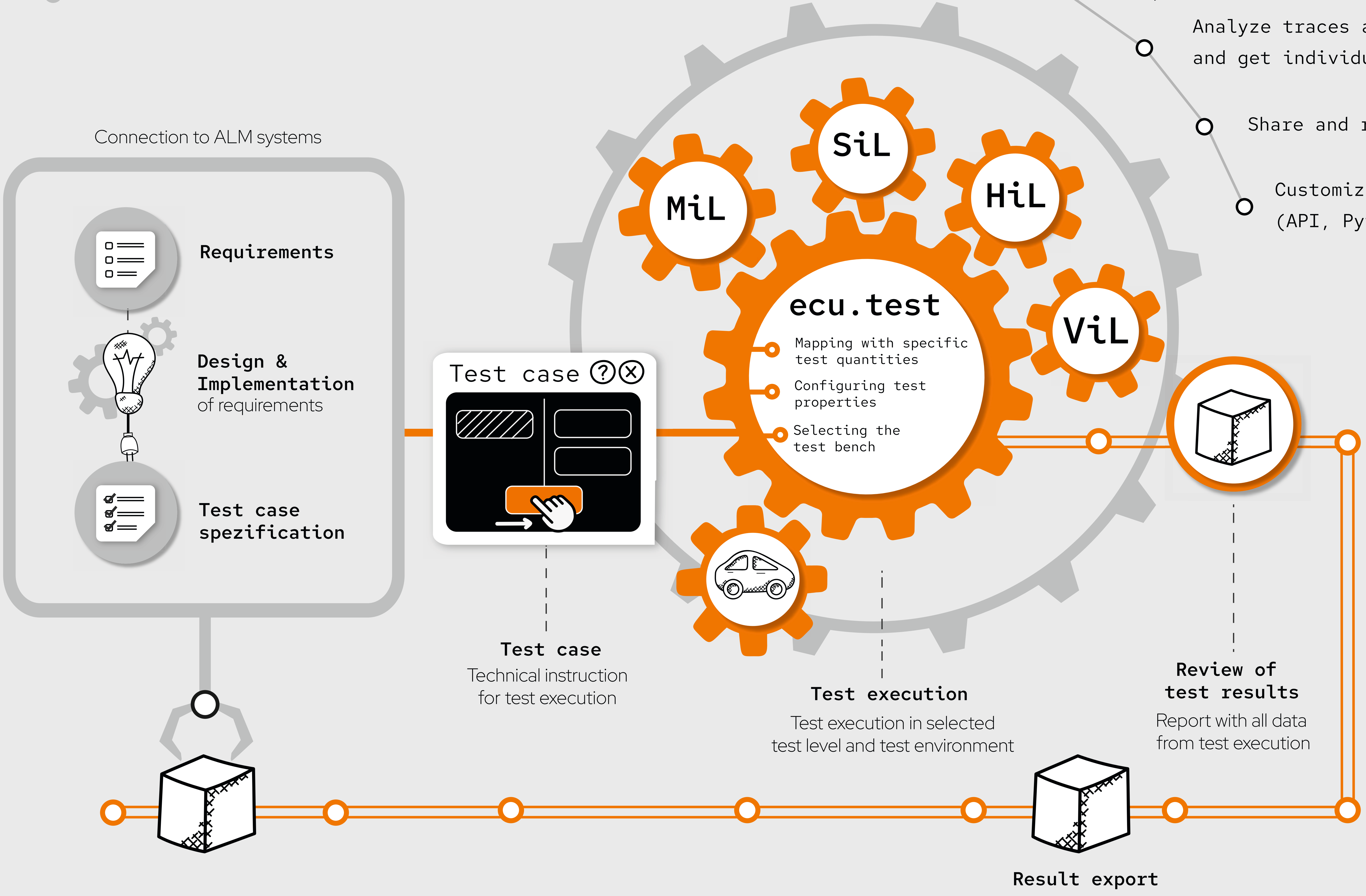
- Test case execution in all stages of vehicle software development
- Integration platform for flexible access to different test tools and test environments (MiL/SiL/HiL/ViL)
- Orchestration of existing tool chains
- Editor for easy creation of test cases across tool boundaries
- Complete test environment automation, even on scaling systems
- Test execution under Windows and Linux, desktop or containers (including Docker)
- Connection to ALM systems
- Cross-team collaboration via Diff and SCM (Git) integration
- Synchronization, analysis and visualization of signal recordings from different sources

HOW TESTING WORKS WITH

ecu.test

It all starts with a requirement for a new, innovative driving function and ends with its release after successful testing.

- Create test cases in GUI via drag&drop
- Combine multiple signal sources
- Execute tests with multiple tools in all test domains (ADAS/AD, Infotainment, Powertrain)
- Do diagnostics (without 3rd party tools)
- Do calibration (without 3rd party tools)
- Analyze traces automatically and get individual reports
- Share and reuse test cases
- Customize your ecu.test (API, Python)



Connection to ALM systems

Requirements

Design & Implementation of requirements

Test case spezification

Test case ⓘ ⓧ

ecu.test

- Mapping with specific test quantities
- Configuring test properties
- Selecting the test bench

MiL

SiL

HiL

ViL

Test case

Technical instruction for test execution

Test execution

Test execution in selected test level and test environment

Review of test results

Report with all data from test execution

Result export

ecu.test

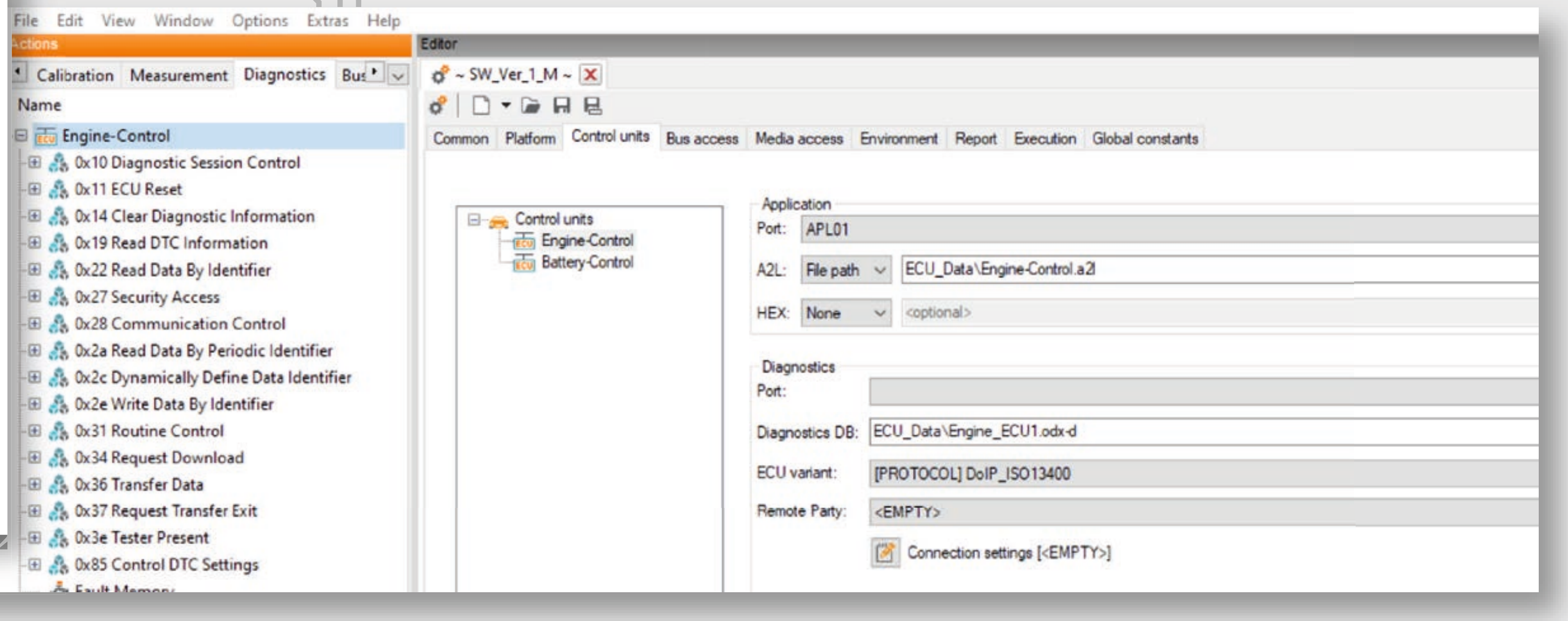
masters test automation

YOUR ADVANTAGES AT A GLANCE

1

Speaks automotive fluently

Whether ARXML, A2L, DBC or ODX – ecu.test processes data in any common automotive format. It can also read image and audio data from the head unit, sensor data from ADAS systems, or touchscreen controls from handhelds.



Name

- Engine-Control
  - 0x10 Diagnostic Session Control
  - 0x11 ECU Reset
  - 0x14 Clear Diagnostic Information
  - 0x19 Read DTC Information
  - 0x22 Read Data By Identifier
  - 0x27 Security Access
  - 0x28 Communication Control
  - 0x2a Read Data By Periodic Identifier
  - 0x2c Dynamically Define Data Identifier
  - 0x2e Write Data By Identifier
  - 0x31 Routine Control
  - 0x34 Request Download
  - 0x36 Transfer Data
  - 0x37 Request Transfer Exit
  - 0x3e Tester Present
  - 0x85 Control DTC Settings
  - Fault Memory

- Control units
  - Engine-Control
  - Battery-Control

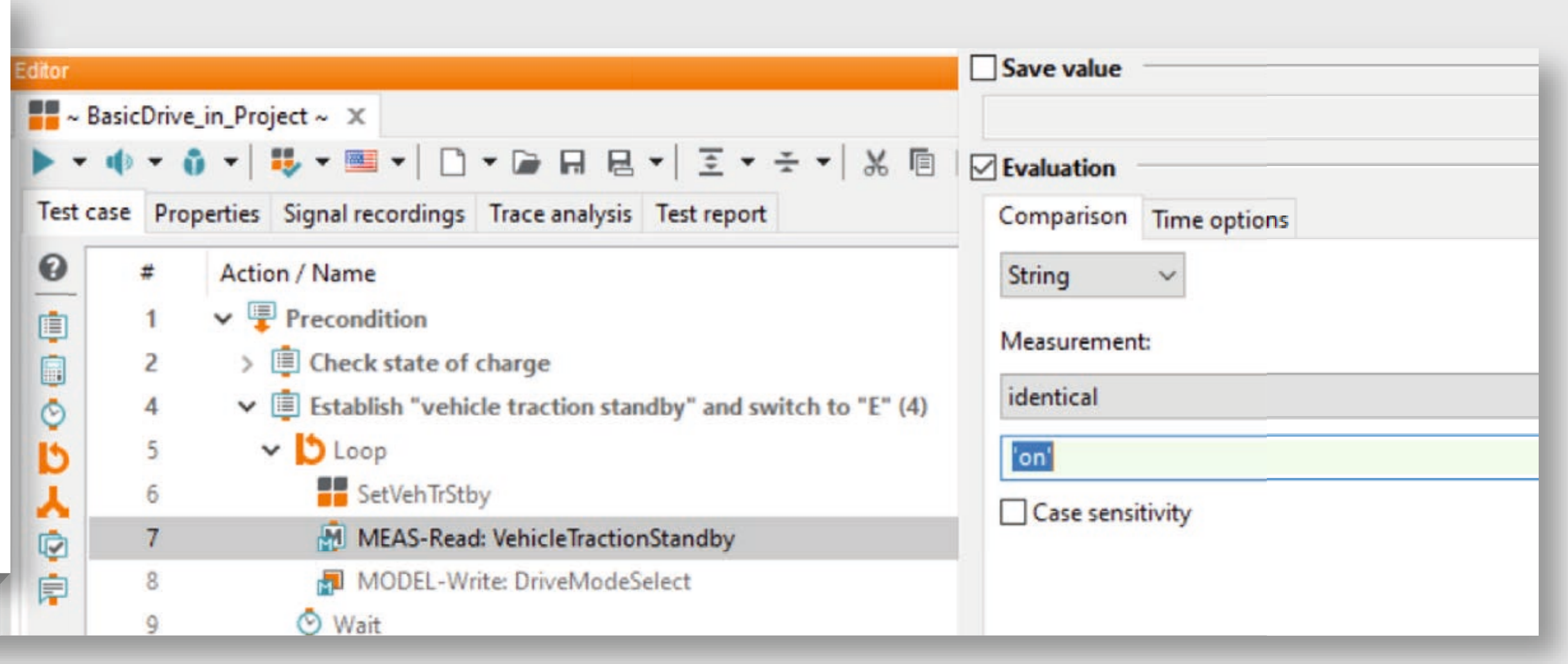
Application
Port: APL01
A2L: File path | ECU_Data\Engine-Control.a2l
HEX: None | <optional>

Diagnostics
Port:
Diagnostics DB: ECU_Data\Engine_ECU1.odx-d
ECU variant: [PROTOCOL] DoIP_ISO13400
Remote Party: <EMPTY>
Connection settings [<EMPTY>]

2

Graphically guided test case creation

Once created, the generic test description makes test cases largely independent of the specific hardware and software of the test environments. This allows tests to be reused in a number of ways at different test levels (MiL, SiL, HiL, ViL) and test areas (Powertrain, Infotainment or ADAS).



#	Action / Name
1	Precondition
2	Check state of charge
4	Establish "vehicle traction standby" and switch to "E" (4)
5	Loop
6	SetVehTrStby
7	MEAS-Read: VehicleTractionStandby
8	MODEL-Write: DriveModeSelect
9	Wait

□ Save value

☑ Evaluation

Comparison | Time options

String

Measurement:

identical

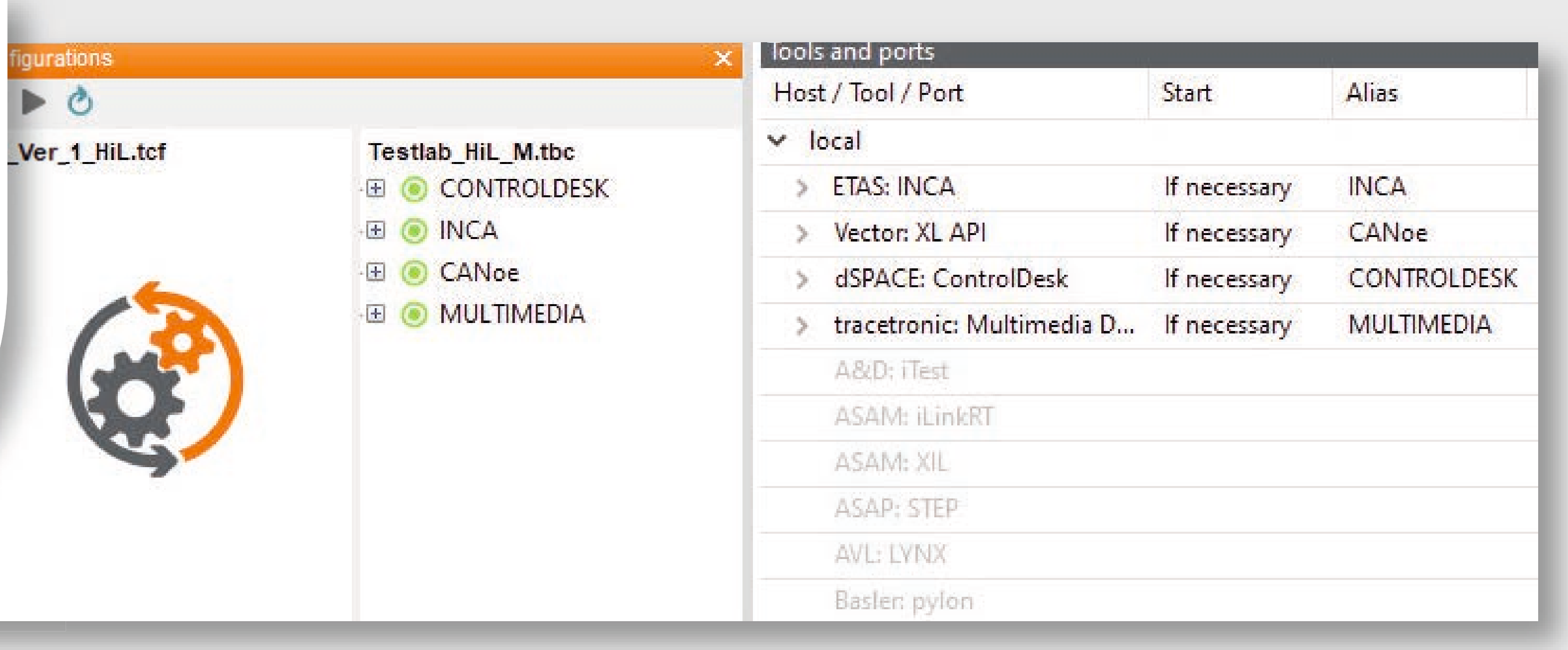
'on'

□ Case sensitivity

3

Combines tools from different manufacturers

More than 80 tools can be connected via standardized access and fully integrated tool interfaces. The test case tool communication is completely abstracted to execute the same test case with different tools. ecu.test can also automate your in-house solutions.



_Ver_1_HiL.tcf

Testlab_HiL_M.tbc
- CONTROLDESK
- INCA
- CANoe
- MULTIMEDIA

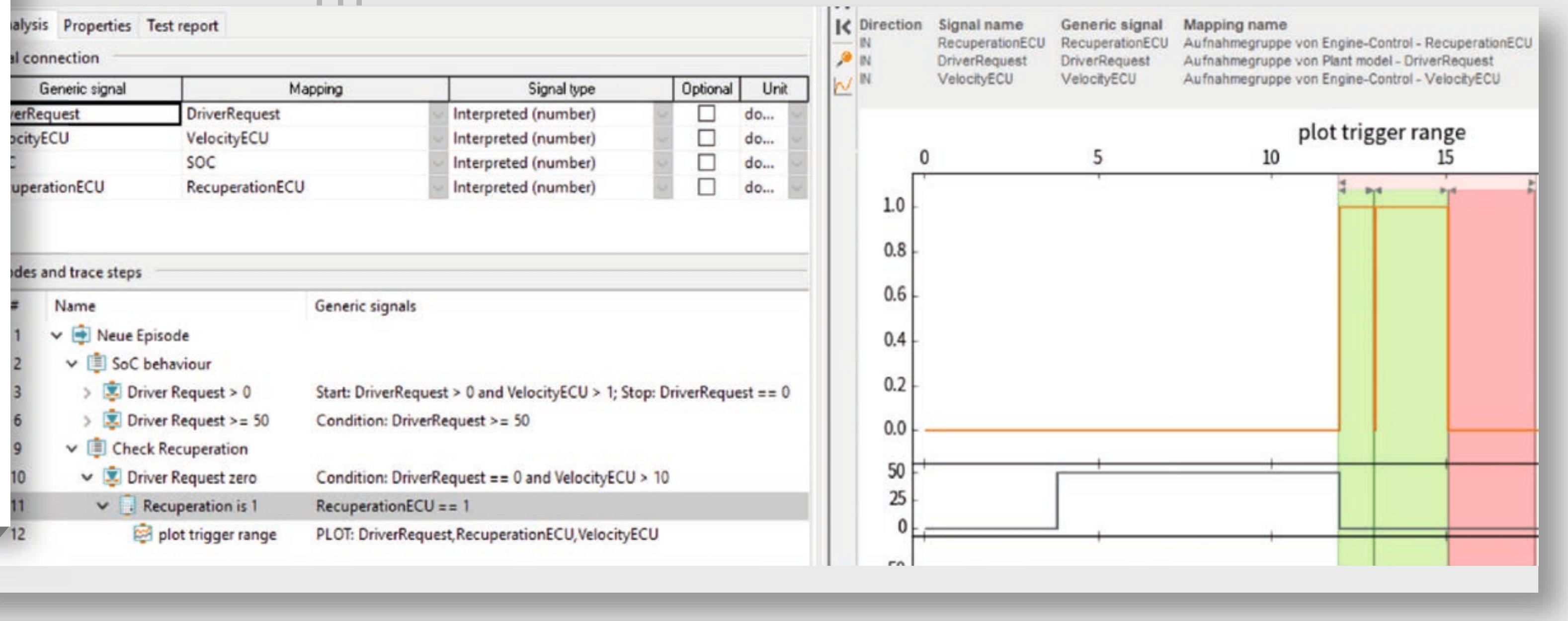
Host / Tool / Port	Start	Alias
local		
ETAS: INCA	If necessary	INCA
Vector: XL API	If necessary	CANoe
dSPACE: ControlDesk	If necessary	CONTROLDESK
tracetronic: Multimedia D...	If necessary	MULTIMEDIA
A&D: iTest		
ASAM: iLinkRT		
ASAM: XIL		
ASAP: STEP		
AVL: LYNX		
Basler: pylon		

YOUR ADVANTAGES AT A GLANCE

4

High-performance signal analysis

For all E/E components and systems in the vehicle, signal curves can be recorded and evaluated over the duration of a test case. Multiple signals from different sources can be automatically synchronized.



Generic signal	Mapping	Signal type	Optional	Unit
erRequest	DriverRequest	Interpreted (number)	□	do...
cityECU	VelocityECU	Interpreted (number)	□	do...
:	SOC	Interpreted (number)	□	do...
uperationECU	RecuperationECU	Interpreted (number)	□	do...

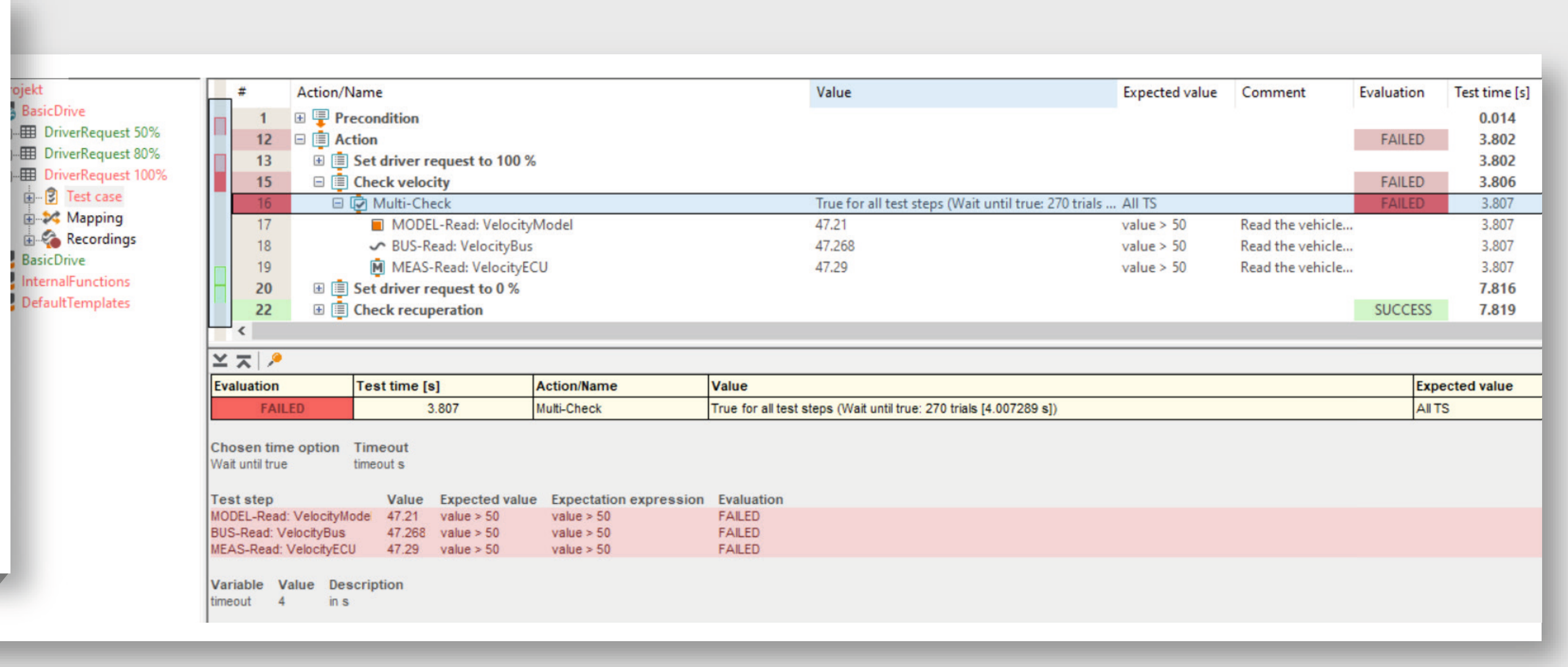
#	Name	Generic signals
1	Neue Episode	
2	SoC behaviour	
3	Driver Request > 0	Start: DriverRequest > 0 and VelocityECU > 1; Stop: DriverRequest == 0
6	Driver Request >= 50	Condition: DriverRequest >= 50
9	Check Recuperation	
10	Driver Request zero	Condition: DriverRequest == 0 and VelocityECU > 10
11	Recuperation is 1	RecuperationECU == 1
12	plot trigger range	PLOT: DriverRequest,RecuperationECU,VelocityECU

Direction	Signal name	Generic signal	Mapping name
IN	RecuperationECU	RecuperationECU	Aufnahmegruppe von Engine-Control - RecuperationECU
IN	DriverRequest	DriverRequest	Aufnahmegruppe von Plant model - DriverRequest
IN	VelocityECU	VelocityECU	Aufnahmegruppe von Engine-Control - VelocityECU

5

Generates structured results

During the test procedure, all relevant data and resulting values are stored. This provides the basis for automatically generated test reports that are complete from the start and contain detailed information on each test step.



ojekt
BasicDrive
DriverRequest 50%
DriverRequest 80%
DriverRequest 100%
Test case
Mapping
Recordings
BasicDrive
InternalFunctions
DefaultTemplates

#	Action/Name	Value	Expected value	Comment	Evaluation	Test time [s]
1	Precondition					0.014
12	Action				FAILED	3.802
13	Set driver request to 100 %					3.802
15	Check velocity				FAILED	3.806
16	Multi-Check	True for all test steps (Wait until true: 270 trials ...	All TS		FAILED	3.807
17	MODEL-Read: VelocityModel	47.21	value > 50	Read the vehicle...		3.807
18	BUS-Read: VelocityBus	47.268	value > 50	Read the vehicle...		3.807
19	MEAS-Read: VelocityECU	47.29	value > 50	Read the vehicle...		3.807
20	Set driver request to 0 %					7.816
22	Check recuperation				SUCCESS	7.819

Evaluation	Test time [s]	Action/Name	Value	Expected value
FAILED	3.807	Multi-Check	True for all test steps (Wait until true: 270 trials [4.007289 s])	All TS

Chosen time option	Timeout
Wait until true	timeout s

Test step	Value	Expected value	Expectation expression	Evaluation
MODEL-Read: VelocityMode	47.21	value > 50	value > 50	FAILED
BUS-Read: VelocityBus	47.268	value > 50	value > 50	FAILED
MEAS-Read: VelocityECU	47.29	value > 50	value > 50	FAILED

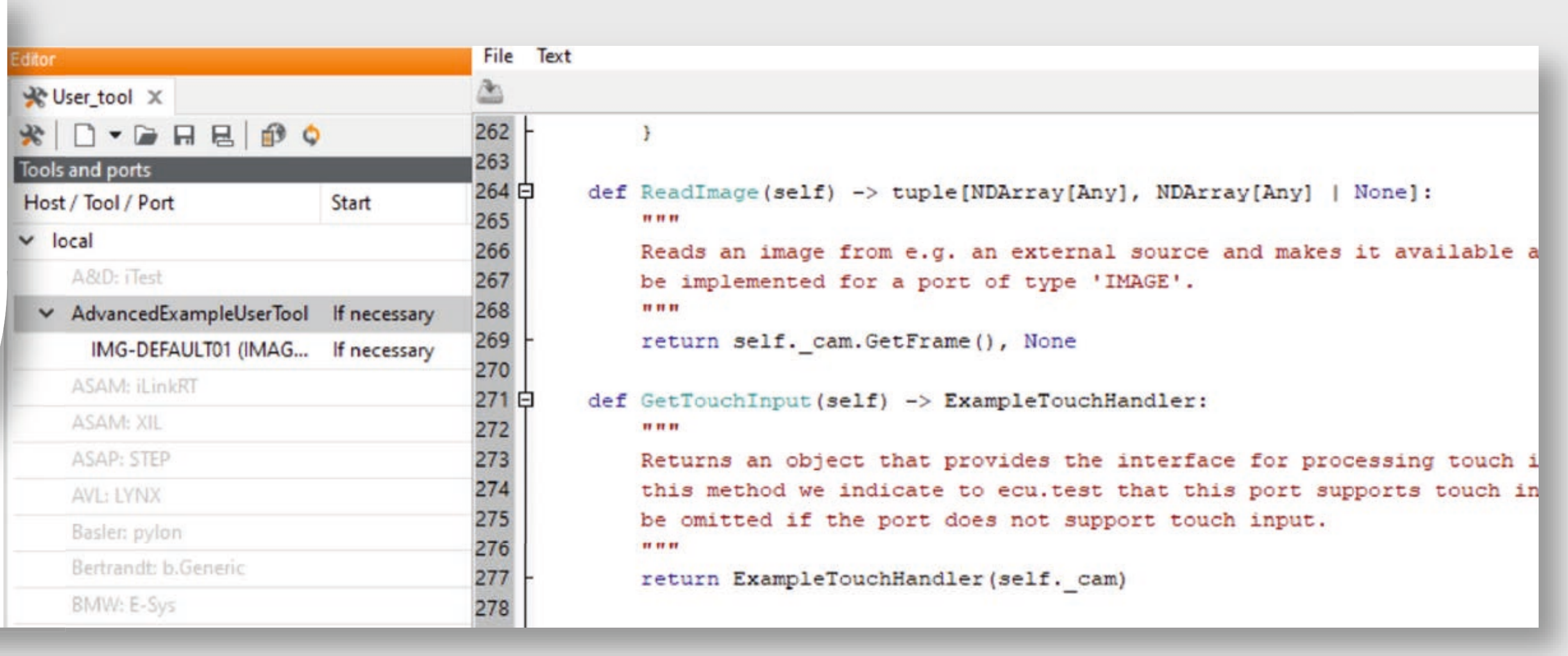
Variable	Value	Description
timeout	4	in s

6

Is highly extensible

1. Everything in ecu.test is accessible via API.
2. The tool can be extended using Python.

This is perfect for your own solutions and extensions. You can quickly and easily customize ecu.test to your needs, both in test case development and trace analysis.



Host / Tool / Port	Start
local	
A&D: iTest	
AdvancedExampleUserTool	If necessary
IMG-DEFAULT01 (IMAG...	If necessary
ASAM: iLinkRT	
ASAM: XIL	
ASAP: STEP	
AVL: LYNX	
Basler: pylon	
Bertrandt: b.Generic	
BMW: E-Sys	

```
        }

    def ReadImage(self) -> tuple[NDArray[Any], NDArray[Any] | None]:
        """
        Reads an image from e.g. an external source and makes it available a
        be implemented for a port of type 'IMAGE'.
        """
        return self._cam.GetFrame(), None

    def GetTouchInput(self) -> ExampleTouchHandler:
        """
        Returns an object that provides the interface for processing touch i
        this method we indicate to ecu.test that this port supports touch in
        be omitted if the port does not support touch input.
        """
        return ExampleTouchHandler(self._cam)

```

10 REASONS TO CHOOSE ecu.test

ecu.test

is always a good idea

Test cases can be automatically executed anytime, anywhere because they are written to be used across all test systems and test domains.

Teams distributed across the globe can easily work together, as workspaces, packages and libraries can be shared and reused.

Test throughput can be significantly increased by carrying out time-consuming analyses away from expensive test benches.

Test results can be reused in other systems, as they can be converted into all formats.

Data is always available and easy to share with teams because it is stored in XML format and managed in Git.

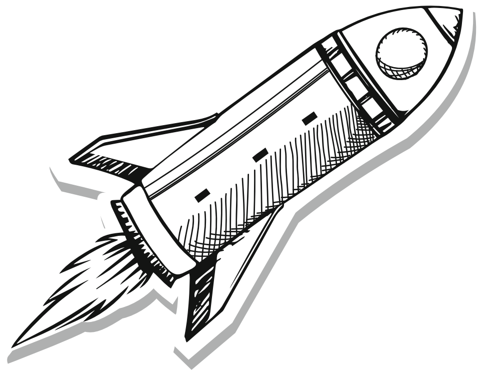
Software defects are quickly identified because tests are repeatable and provide reliable results.

Test cases can be developed iteratively and executed at any time because there is always a direct connection to the test system.

New users can create test cases effortlessly because the GUI is user-friendly and the underlying tools are easily accessible.

Test steps can be reconstructed in detail because the structure of the automatically generated reports matches the test case structure.

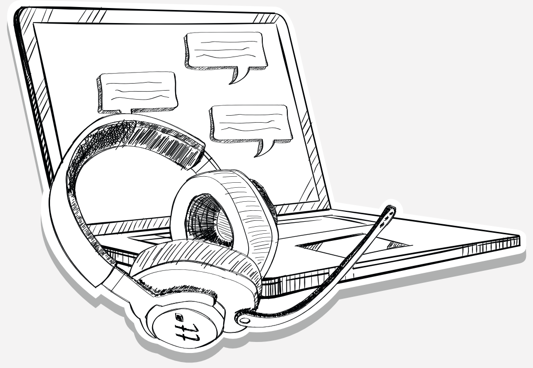
Testing in heterogeneous test environments is possible without restrictions, since other (in-house) tools can be seamlessly integrated into existing systems via standardized API interfaces and Python scripts.



RELEASES

We deliver four software releases per year, keeping all users informed of the latest features, enhancements, previews, and discontinuations.

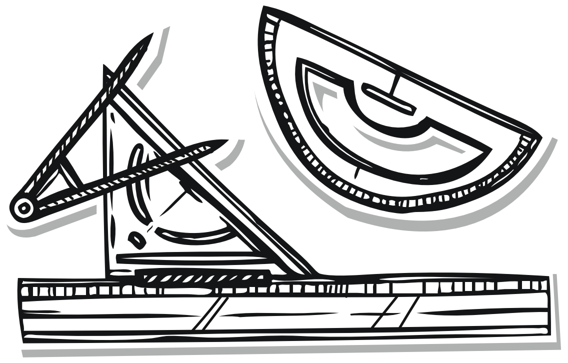
As part of our incremental, agile development process, we provide numerous pre-release versions.



SUPPORT

We provide comprehensive technical and strategic support from all our locations:

- Initial setup
- General use of the software
- Problem analysis and solution
- (Best practices) advice
- Test system development
- Adapting test methods and strategies to your workflow



ENGINEERING

As an end-to-end solution provider, we develop continuous workflows to execute software tests in great detail and comprehensively, with a very high degree of automation.

We can also customize your existing systems – from simple test cases to complex, dynamically scaled automation solutions in the cloud.

,tracetronic

tracetronic supports automotive manufacturers and suppliers in the development of highly complex automotive software with software products and customized services. The focus is on solutions for an automated test process in all phases of software development – from unit tests to integration tests in the vehicle.

tracetronic was founded in 2004 as a university start-up at the Technical University of Dresden and is now a global company. It currently employs over 400 professionals, students and trainees.

The tracetronic group is headquartered in Dresden, Germany, with additional offices in Munich, Ingolstadt, Stuttgart, Hamburg as well as in the USA, South Korea, Japan and China.
Our products and solutions are used by more than 400 customers in over 30 countries worldwide, including Audi, BMW Group, Bosch, BYD, Mercedes, Stellantis, Valeo, Continental, Daimler, Ford, John Deere, Magna, Porsche, Rivian, Siemens and Volkswagen.