Whether you're building ECUs for combustion, hybrid, or electric platforms — this approach helps you test more realistically, more flexibly, and more intelligently.

Contact

info@agsotec.de
www.agsotec.de
Weimarer Str. 11
80807 München

Follow our journey!

AGSOTEC GmbH

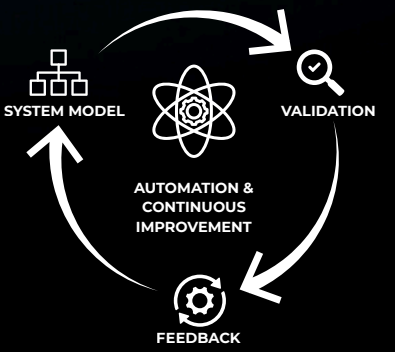
//AGSOTEC

WELCOME TO THE WORLD OF AGILE SOFTWARE TECHNOLOGIES

IMPLICIT TESTING

A new paradigm for scalable validation in embedded systems:

Instead of designing one test case per low-level requirement, we build realistic system-level scenarios and evaluate the system's behavior indirectly — using validation criteria that reflect high-level expectations.



„Think of it as simulation-based testing that checks whether the system behaves correctly under realistic usage conditions, rather than whether it checks off individual requirement boxes."

//AGSOTEC

WHY EXPLICIT TESTING HAS REACHED ITS LIMITS

Traditional testing of embedded control units relies on manually defined test cases derived from detailed requirements. This explicit testing approach demands enormous effort — thousands of test cases that are difficult to maintain and interpret.

As systems grow in complexity, explicit testing reaches its limits:

- **High maintenance and costs**: every software change requires adjusting countless tests.
- **Limited coverage:** only predefined cases are checked — cross-effects remain hidden.
- **Slow Feedback:** Full test runs are resource-intensive and infrequent — delaying bug detection and correction.

Modern development therefore needs a more scalable and system-level validation approach — one that tests how systems behave, not just whether they meet individual requirements.

HOW IT WORKS

The implicit testing methodology is built on three modular pillars:

1. Plant Models

Simulate the physical environment, hardware behavior, and surrounding systems. Built with Simulink and Simscape, they allow realistic testing without real hardware.

2. Test Scenarios

Describe real-world use cases (e.g., urban driving, component failure, temperature changes) using simple script commands. These can be written manually or generated automatically.

3. Validation Criteria

Implemented in Matlab, these check simulation outcomes against high-level behavioral expectations — without knowing how the software works internally.

Each simulation test run combines these three elements: the plant model drives inputs, the scenario defines the situation, and the validation criteria judge the output.