# un code sûr, sécurisé et robuste pour vos applications embarquées Automobiles et Aérospatiales

Atelier Logiciel, Paris La Défense, 27 juin 2023

Organisé par Antycip Technologies et


Detect and Protect.

# Portefeuille Produits

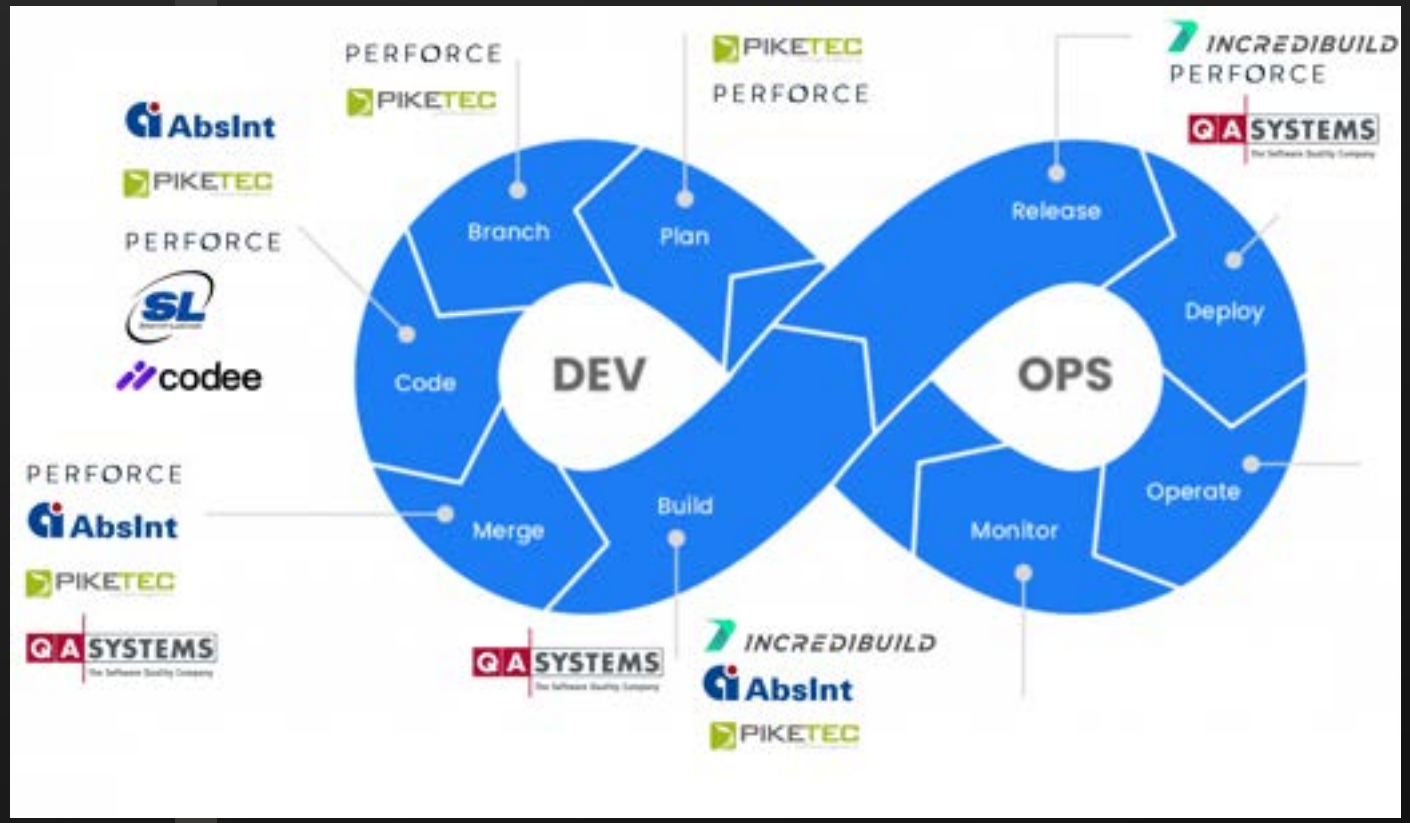
Atelier Logiciel – 27 juin 2023

Nos Métiers:

Commercialisation - Support Technique – Intégration – Ateliers – Formations

Outils pour les R&D’s logicielles:

Sureté Fonctionnelle – Cybersecurité – Qualité – Test – Efficacité Opérationnelle




Detect and Protect.

# Embarqué dans l‘Aerospace & Automotive

Atelier Logiciel – 27 juin 2023

ANTYCIP
A Hensoldt Company.

Similarité + Porosité

Poids croissant des exigences réglementaires

- Cybersécurité
- Sureté de fonctionnement

Complexité croissante, poids croissant du logiciel



HENSOLDT

# Use Cases

Atelier Logiciel – 27 juin 2023

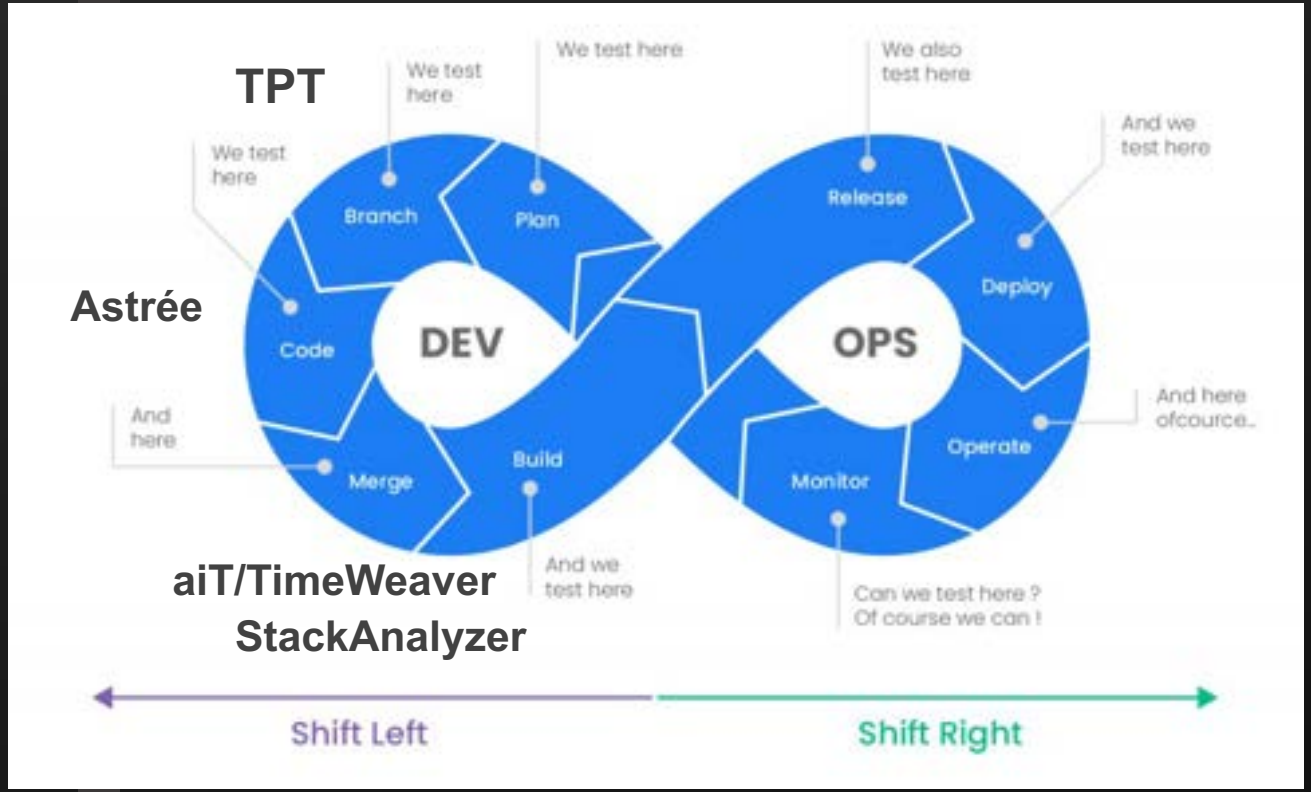
Shift Left, Unit Testing

Dynamic and Static Testing

Functional and Non-Functional Testing

Tools in action:

- TPT on source code
- Astrée on source code
- aiT/TimeWeaver + StackAnalyzer on object code



# AbsInt Angewandte Informatik GmbH

- Provides advanced development tools for embedded systems, and tools for validation, verification, and certification of safety-critical or security-relevant software
- Founded in February 1998 by six researchers of Saarland University, Germany, from the group of programming languages and compiler construction of Prof. Dr. Dr. h.c. mult. R. Wilhelm
- Privately held by the founders
- 40+ employees
- Customers from 40+ countries all over the world
- Industries: Aerospace, Automotive, Railway, Energy (nuclear power, wind energy), Healthcare, …

PIKETEC

*Automotive Software Experts*

- Vendor of TPT - a Test Automation Tooling for embedded systems and software to perform dynamic testing
- Provider of Testing Services (focus on Automotive)
- Founded in 2007 by Dr. Bringmann, Dr. Lüdemann and Mr. Krämer
- 60+ employees
- 150+ customers – 20.000+ User, from 50+ countries all over the world
- Industries: Automotive, Aerospace, Railway, Energy, Healthcare, …

# Agenda

## Atelier Logiciel – 27 juin 2023

- 8H30 - 9H00 Café de bienvenue
- 9H00 - 9H30 Ordre du jour, qui nous sommes, tour de table

Round Table

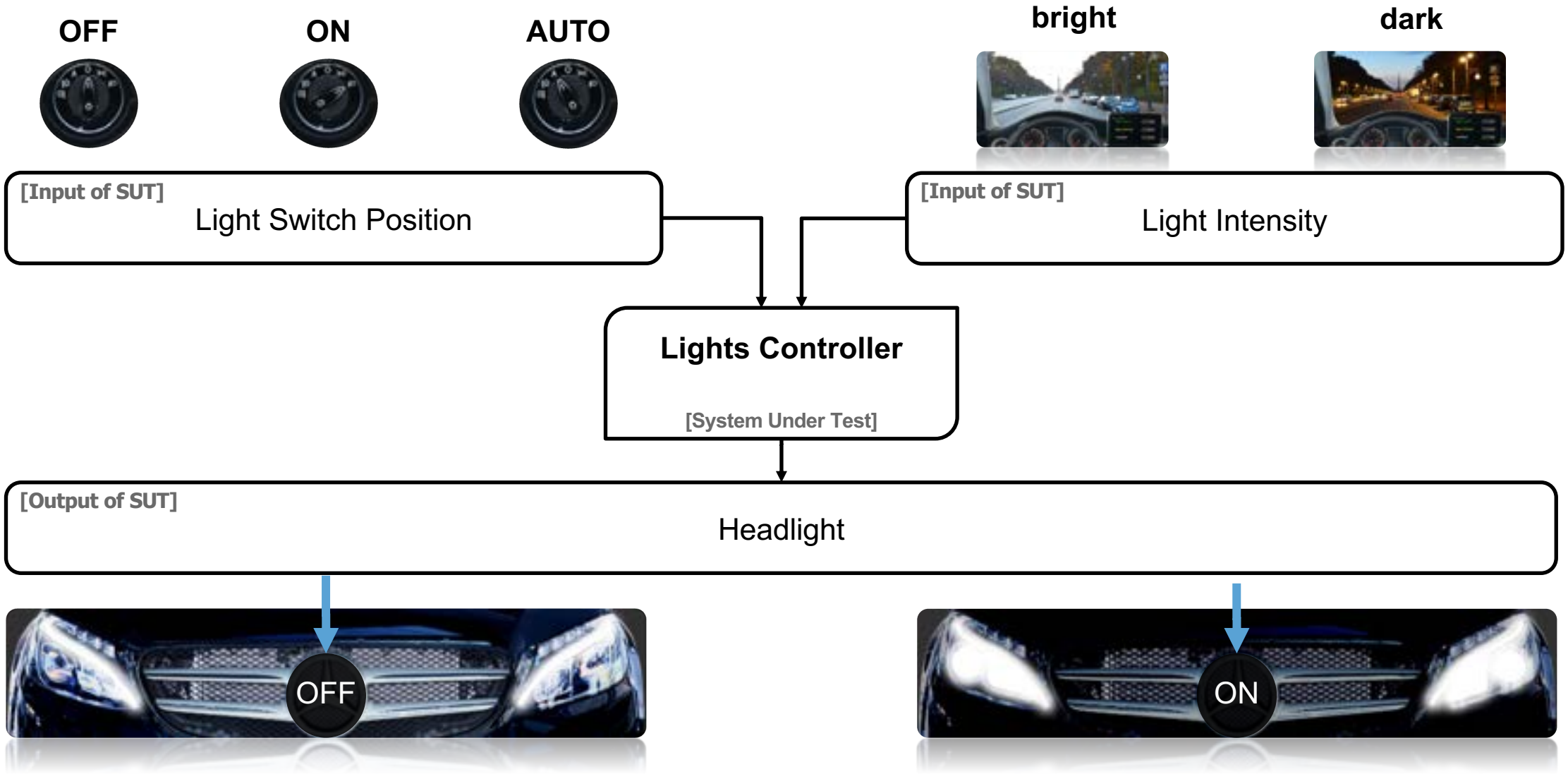
- 9H30 - 10H30 Introduction du projet de démonstration et vérification pratique des règles MISRA C/C++ avec Astrée d'AbsInt. Génération de cas de test, exécution de tests pour le code C/C++, résolution de la matrice de traçabilité avec TPT de PikeTec.
- 10H30 - 10H45 Pause café, réseautage
- 10H45 - 11H30 Détection statique de bugs critiques cachés avec Astrée d'AbsInt, analyse statique de la pile et du timing avec StackAnalyzer et TimeWeaver/aiT d'AbsInt, gestion du changement et intégration dans une CI avec TPT de PikeTec.
- 11H30 - 12H30 Démonstration pratique, hands-on
- 12H30 - 13H00 Q&R, conclusions, évaluation

# Safe, secure, and robust code for embedded applications in automotive and aerospace

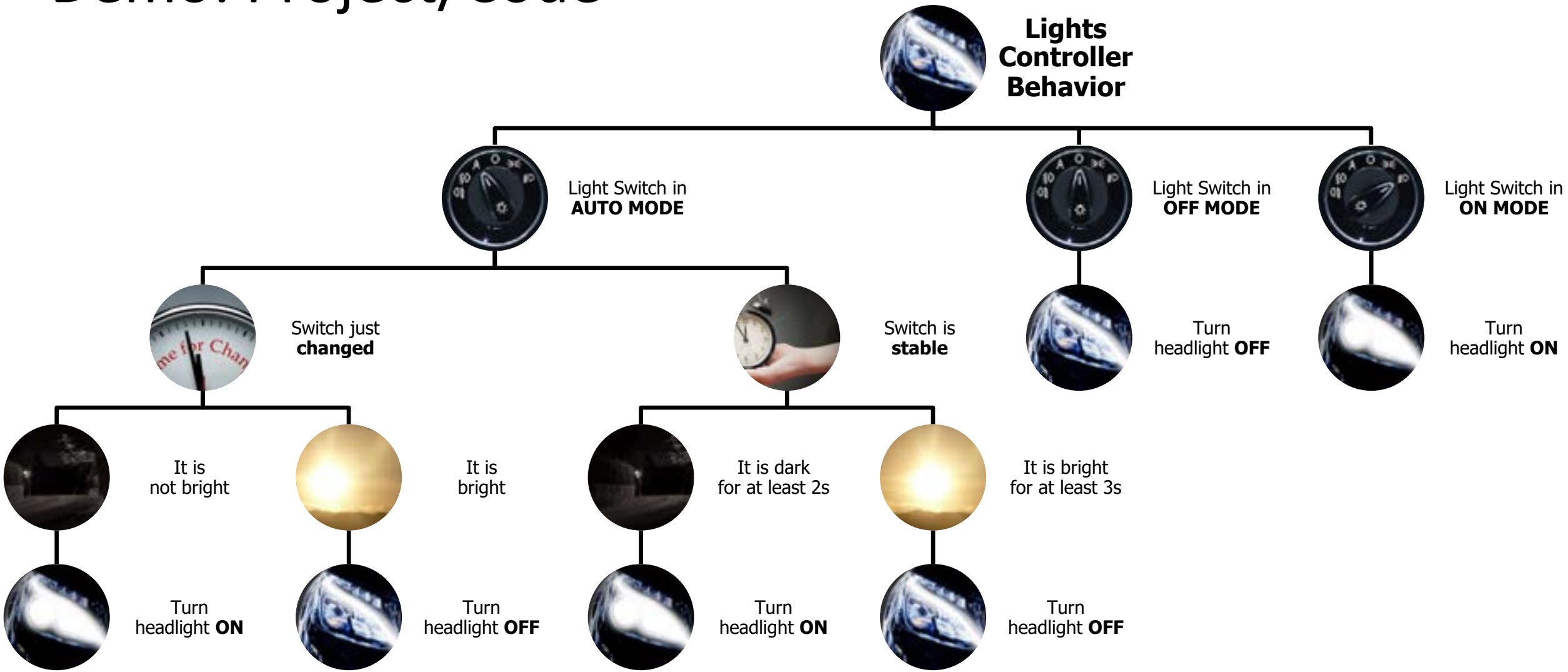
Paris, June 27th 2023

AbsInt GmbH, ANTYCIP Technologies, PikeTec GmbH

# Demo: Project/Code



# Demo: Project/Code



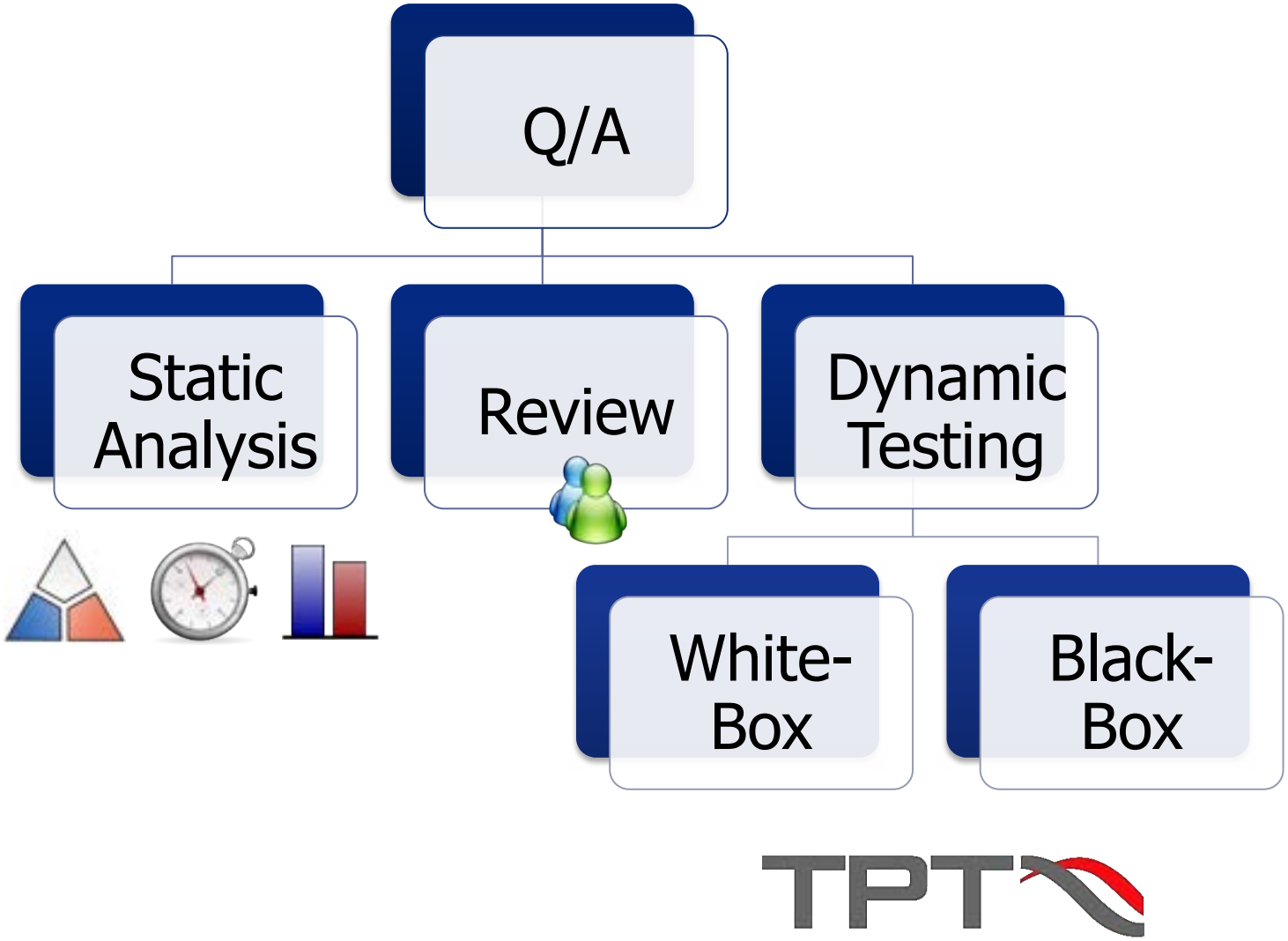
# Functional Safety

- Demonstration of functional correctness
  - Functional requirements are satisfied
  - ➢ Automated and/or model-based **testing**
  - ➢ Formal techniques: model checking, theorem proving
- Satisfaction of safety-relevant quality requirements
  - Compliance with the software architecture
  - No runtime errors (e.g. division by zero, overflow, invalid pointer access, out-of-bounds array access)
  - Resource usage:
    - Timing requirements (e.g. WCET, WCRT)
    - Memory requirements (e.g. no stack overflow)
  - Robustness / freedom of interference (e.g. no corruption of content, incorrect synchronization, illegal read/write accesses)
  - Adequate requirements coverage and structural coverage of testing
  - ➢ **Static analysis**, formal technique (sound): abstract interpretation

**REQUIRED BY DO-178B / DO-178C / ISO-26262, EN-50128, IEC-61508**

**REQUIRED BY DO-178B / DO-178C / ISO-26262, EN-50128, IEC-61508**

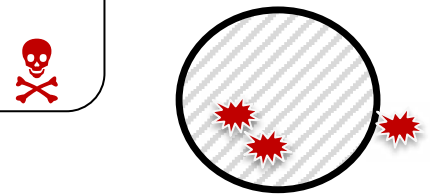
# High Level Q/A



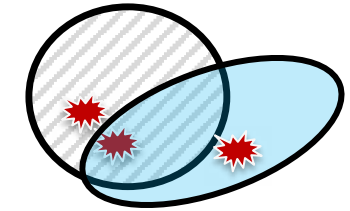
# Static Program Analysis

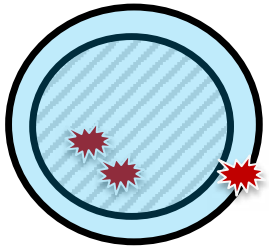
- Categories, depending on analysis depth:
  - Syntax-based: Coding guideline checkers (e.g. MISRA C)
  - Semantics-based

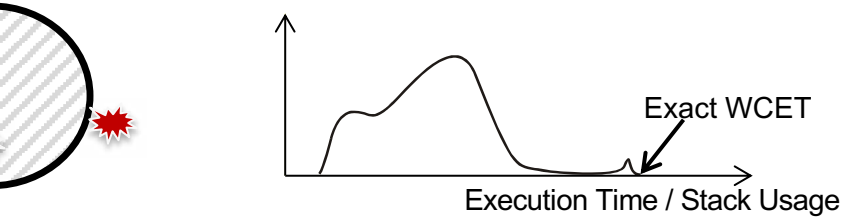
Question: Is there an error in the program?
- False positive: answer wrongly “Yes”
- False negative: answer wrongly “No”

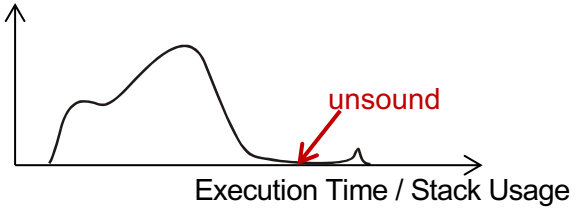


  - Unsound: Bug-finders / bug-hunters.
    - False positives: possible
    - False negatives: possible
  - ☑ Sound / Abstract Interpretation-based
    - False positives: possible
    - No false negatives ⇨ Soundness
      No defect missed



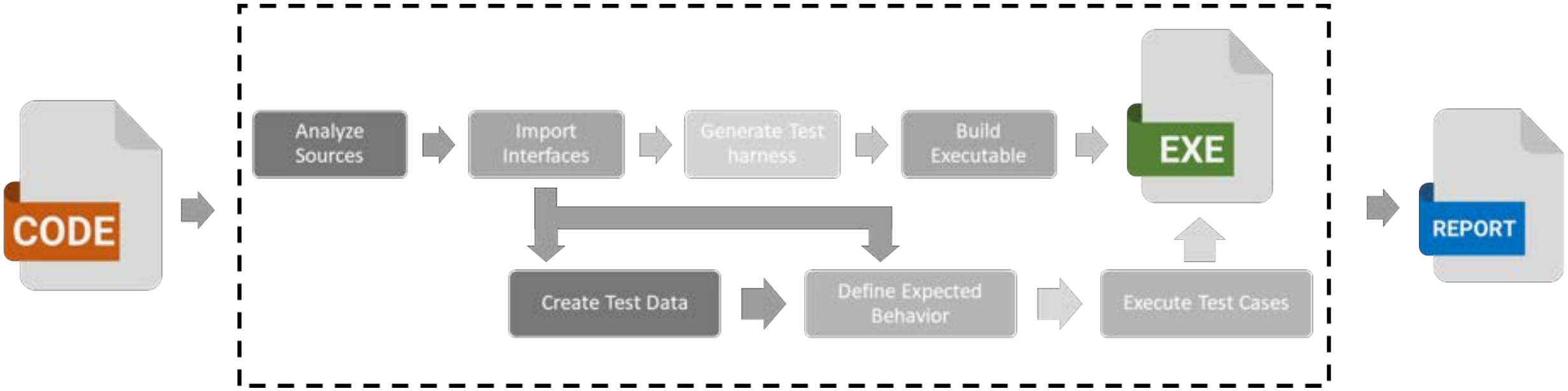





# Dynamic Testing

- Stimulation of a running software to assess its behavior.
- Unit tests, integration tests, system tests and acceptance tests utilize dynamic testing.
- Software must actually be compiled and run.
- Can be done manually or with the use of an automated process.



# AbsInt Angewandte Informatik GmbH

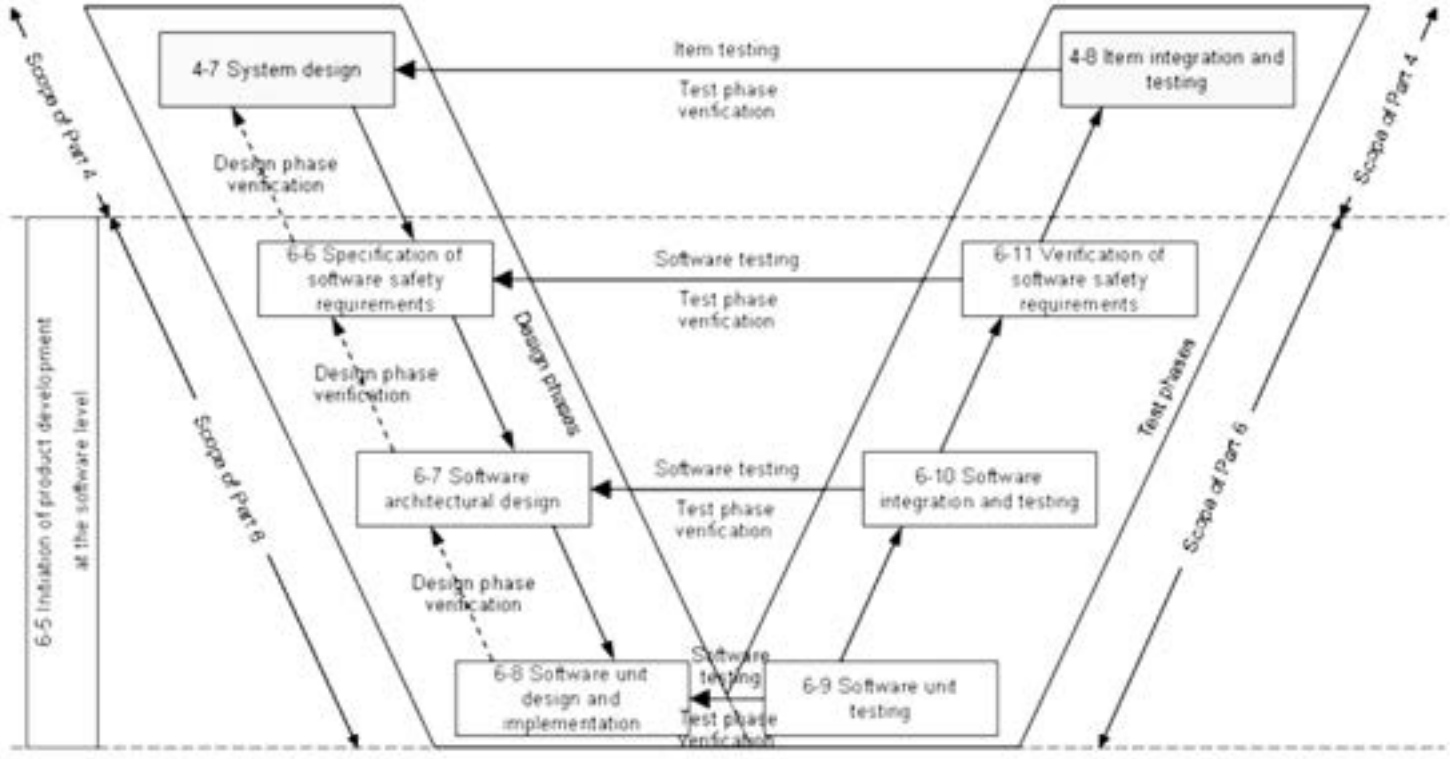
- Provides advanced development tools for embedded systems, and tools for validation, verification, and certification of safety-critical or security-relevant software
- Founded in February 1998 by six researchers of Saarland University, Germany, from the group of programming languages and compiler construction of Prof. Dr. Dr. h.c. mult. R. Wilhelm
- Privately held by the founders
- 40+ employees
- Customers from 40+ countries all over the world
- Industries: Aerospace, Automotive, Railway, Energy (nuclear power, wind energy), Healthcare, …

PIKETEC

*Automotive Software Experts*

- Vendor of TPT - a Test Automation Tooling for embedded systems and software to perform dynamic testing
- Provider of Testing Services (focus on Automotive)
- Founded in 2007 by Dr. Bringmann, Dr. Lüdemann and Mr. Krämer
- 60+ employees
- 150+ customers – 20.000+ User, from 50+ countries all over the world
- Industries: Automotive, Aerospace, Railway, Energy, Healthcare, …

# Development Process

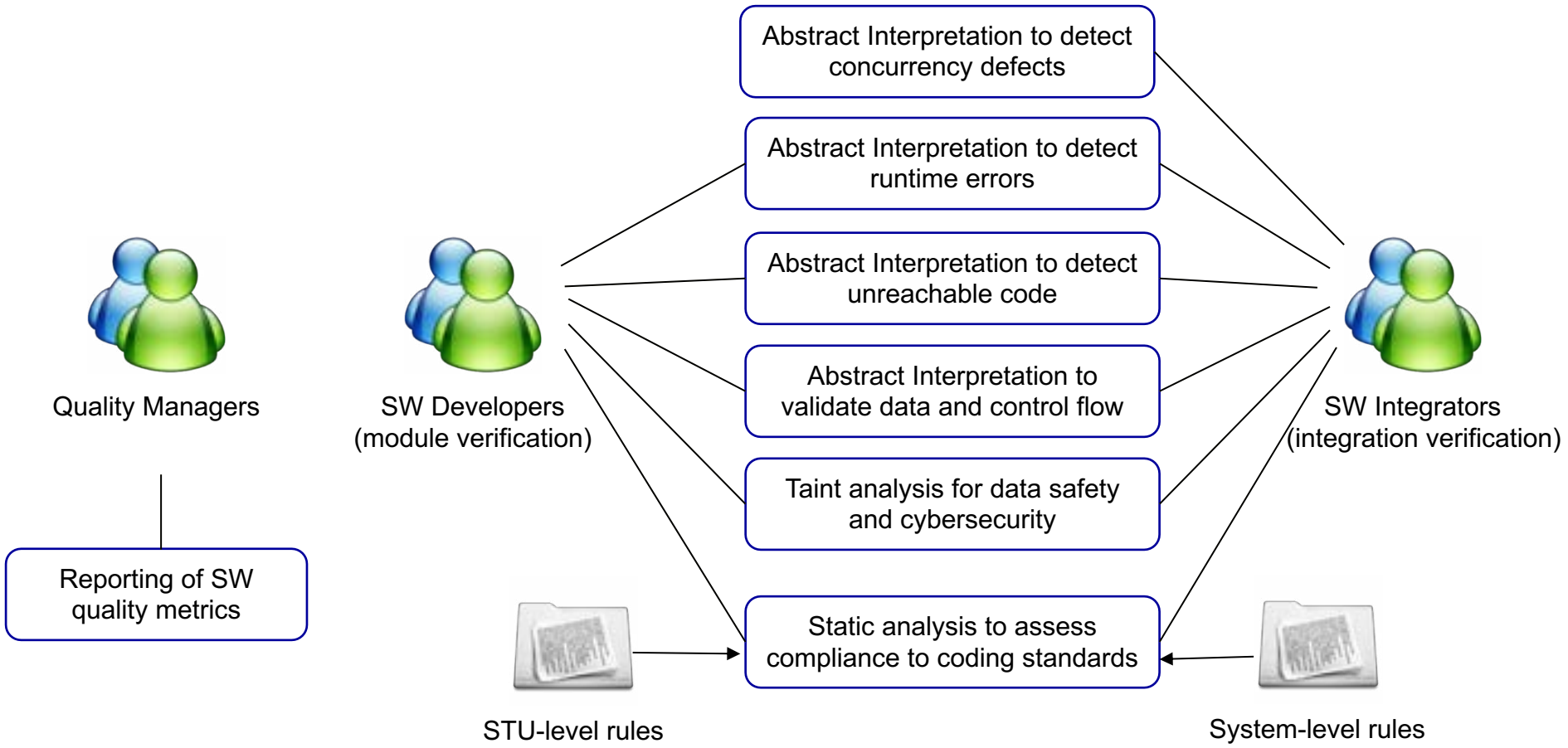



Excerpt from:
*ISO 26262-6 Road vehicles - Functional safety – Part 6: Product development: Software Level, 2011.*

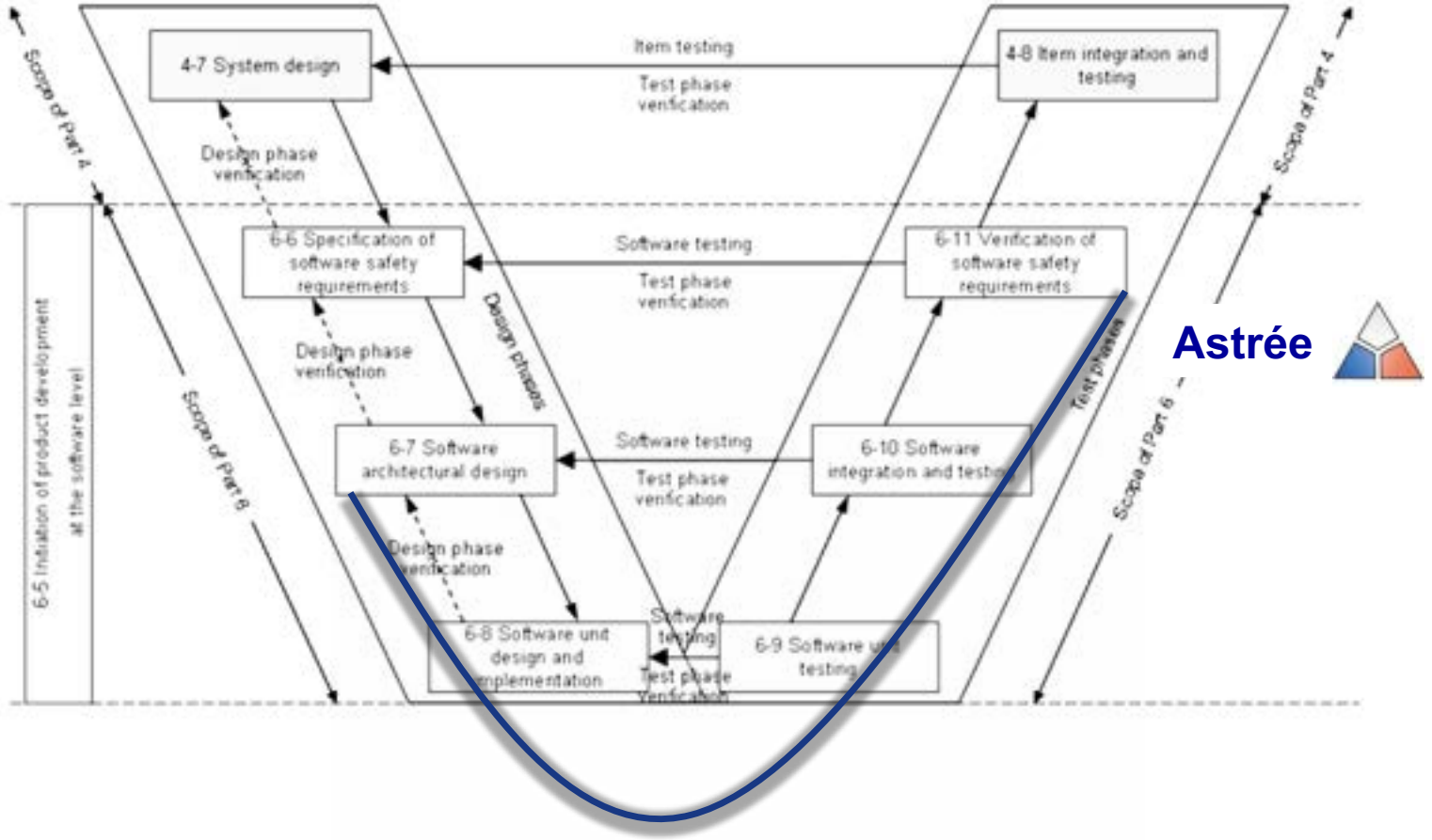
# Astrée

- Sound static Analyzer based on Abstract Interpretation designed to prove the absence of runtime errors and data races in C programs
  - No alarm (potential runtime error / data race) ⇨ no such errors in the code
  - Essential for functional safety and cybersecurity
- Reference customer: Airbus flight control software (DO-178B level A)
  No false alarm on >755.000 LOC, analysis time 6h.
- Beyond runtime errors: Taint analysis, data and control flow analysis, alias analysis, static assertions, …
- Qualification Support Kits (QSK) enable automatic tool qualification according to ISO-26262, DO-178B/ DO-178C, IEC-61508, IEC-60880, etc. up to the highest criticality levels
- Support for model-based code generation: Tool Box for TargetLink, model link to MATLAB/ SIMULINK
- Open formats, full continuous verification support
- Coding guideline checker included (RuleChecker): MISRA C/C++, SEI CERT C/C++, Ad. Autosar C++, …

# Astrée Use Cases



# Development Process




Excerpt from:
*ISO 26262-6 Road vehicles - Functional safety – Part 6: Product development: Software Level, 2011.*

TPT

ASCET
SIMULINK
TARGETLINK
C | C++

AUTOSAR
SILVER
CANoe

VTD
CarMaker
OpalRT

trace32
PLS UDE

XIL-API
dSpace
Speedgoat
INCA

**+ many more**

# 32

**Plug & Play supported technologies**

# 1

**Co-Simulation Platform**

Model- /
Software- /
Processor- /
Hardware- /
in-the-loop

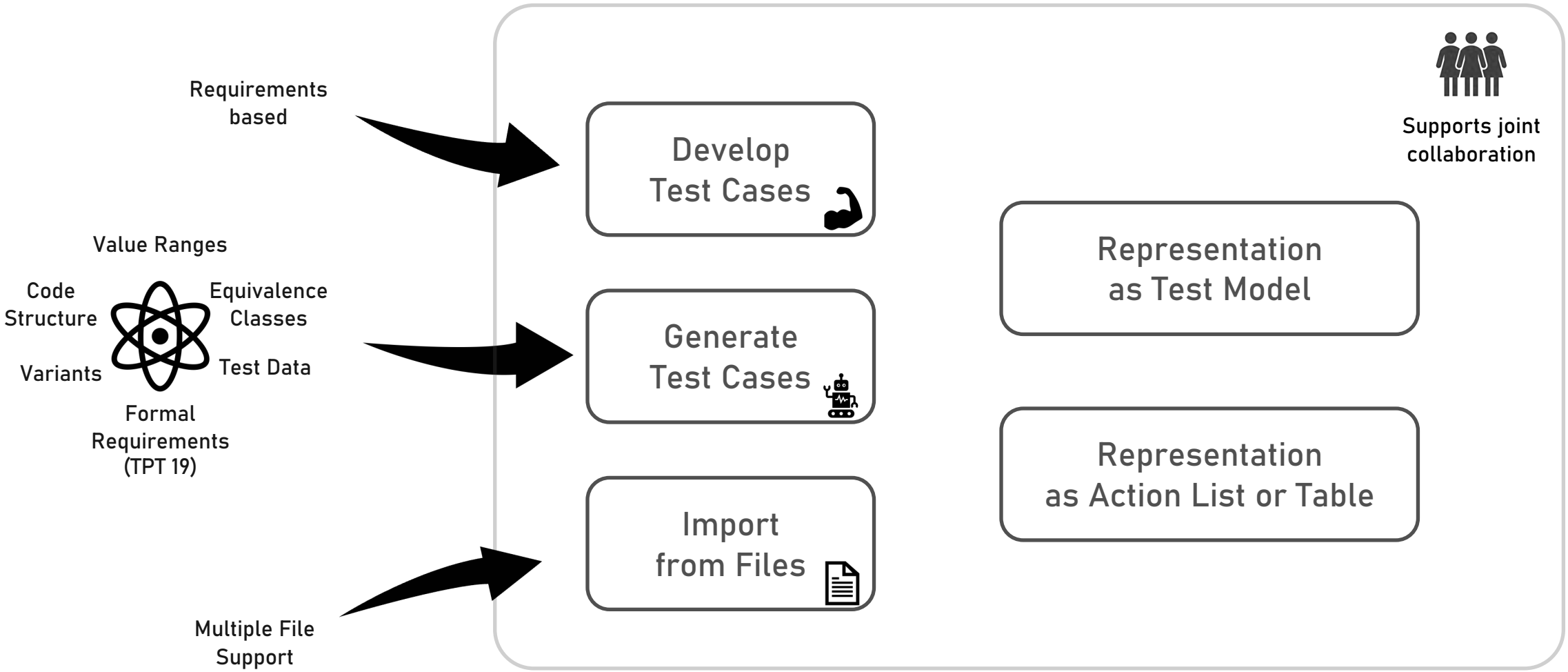
Automation of
Vehicle-Tests

Unit tests
Software tests
Integration tests
Hardware tests
System tests

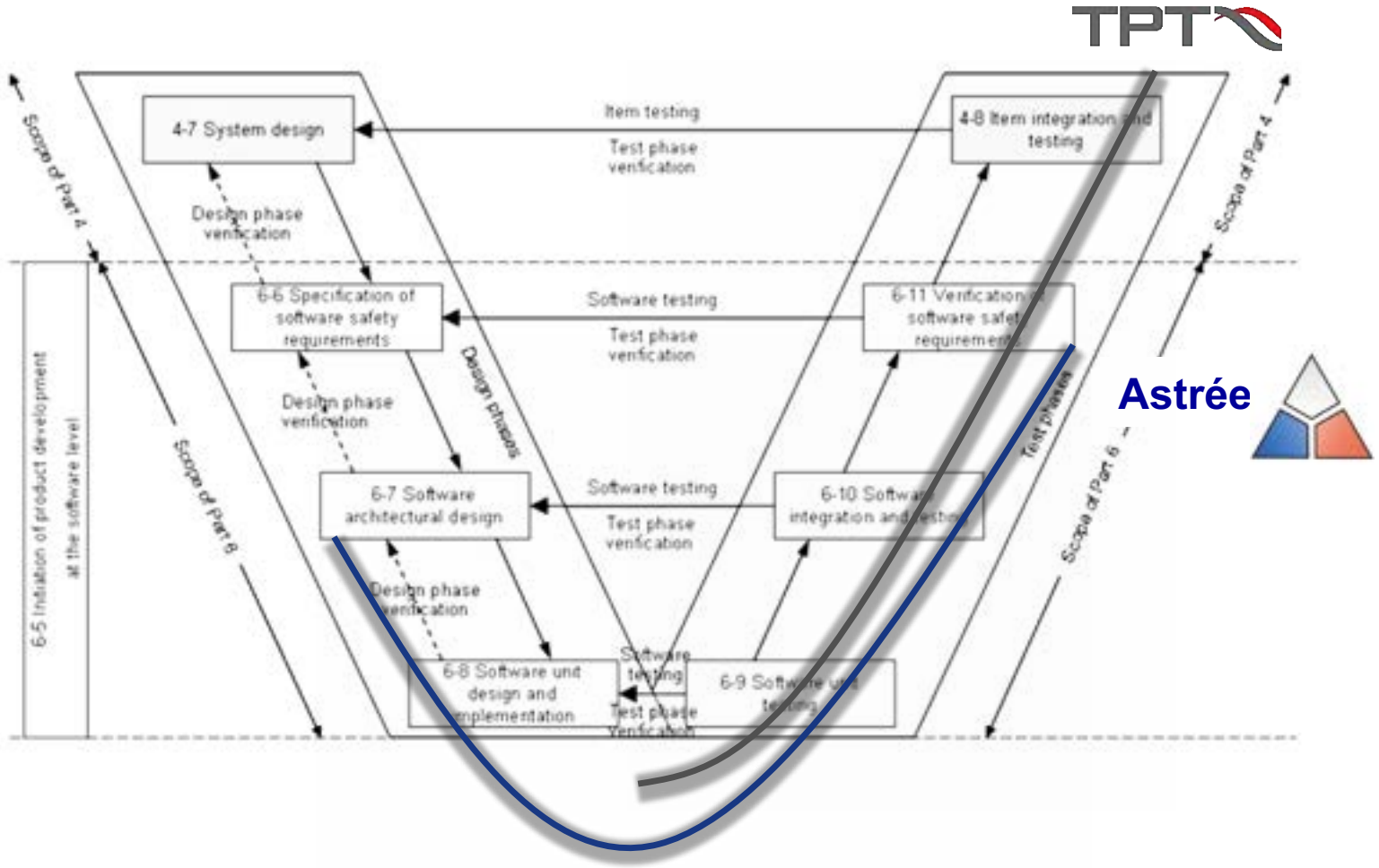
**Back-to-Back & Regression Test Execution**

**Safety Critical Development according to**
ISO 26262
DO 178C

TPT


Requirements based
Value Ranges
Code Structure
Equivalence Classes
Variants
Test Data
Formal Requirements (TPT 19)
Multiple File Support
Develop Test Cases
Generate Test Cases
Import from Files
Supports joint collaboration
Representation as Test Model
Representation as Action List or Table

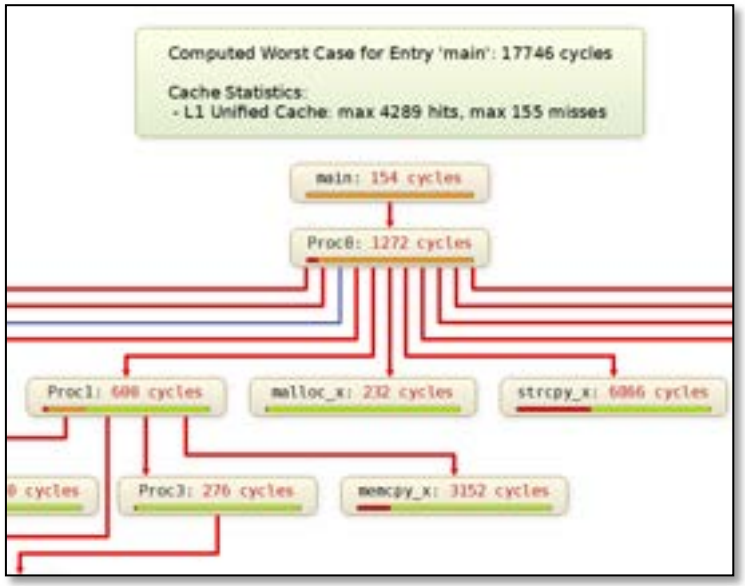
# Development Process




Excerpt from:
*ISO 26262-6 Road vehicles - Functional safety – Part 6: Product development: Software Level, 2011.*

# aiT / TimeWeaver

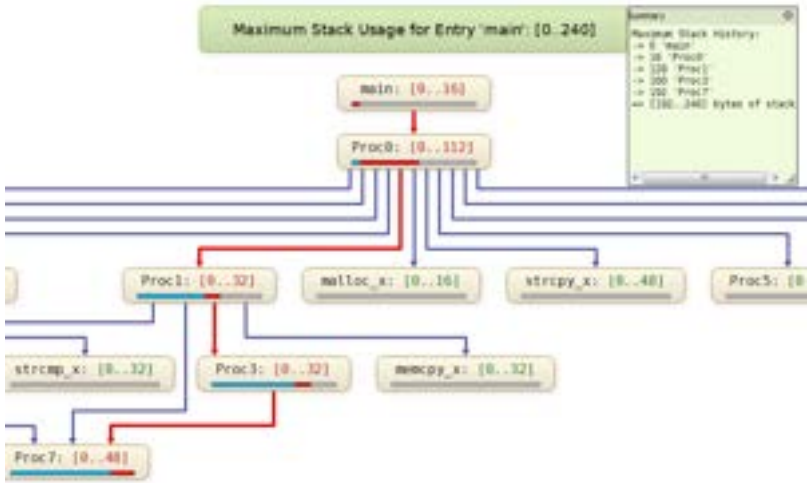
- WCET results determined automatically
  - For timing predictable architectures:
    static analysis (aiT)
  - For high-end multi-core architectures:
    Combines static analysis and non-intrusive tracing (TimeWeaver)



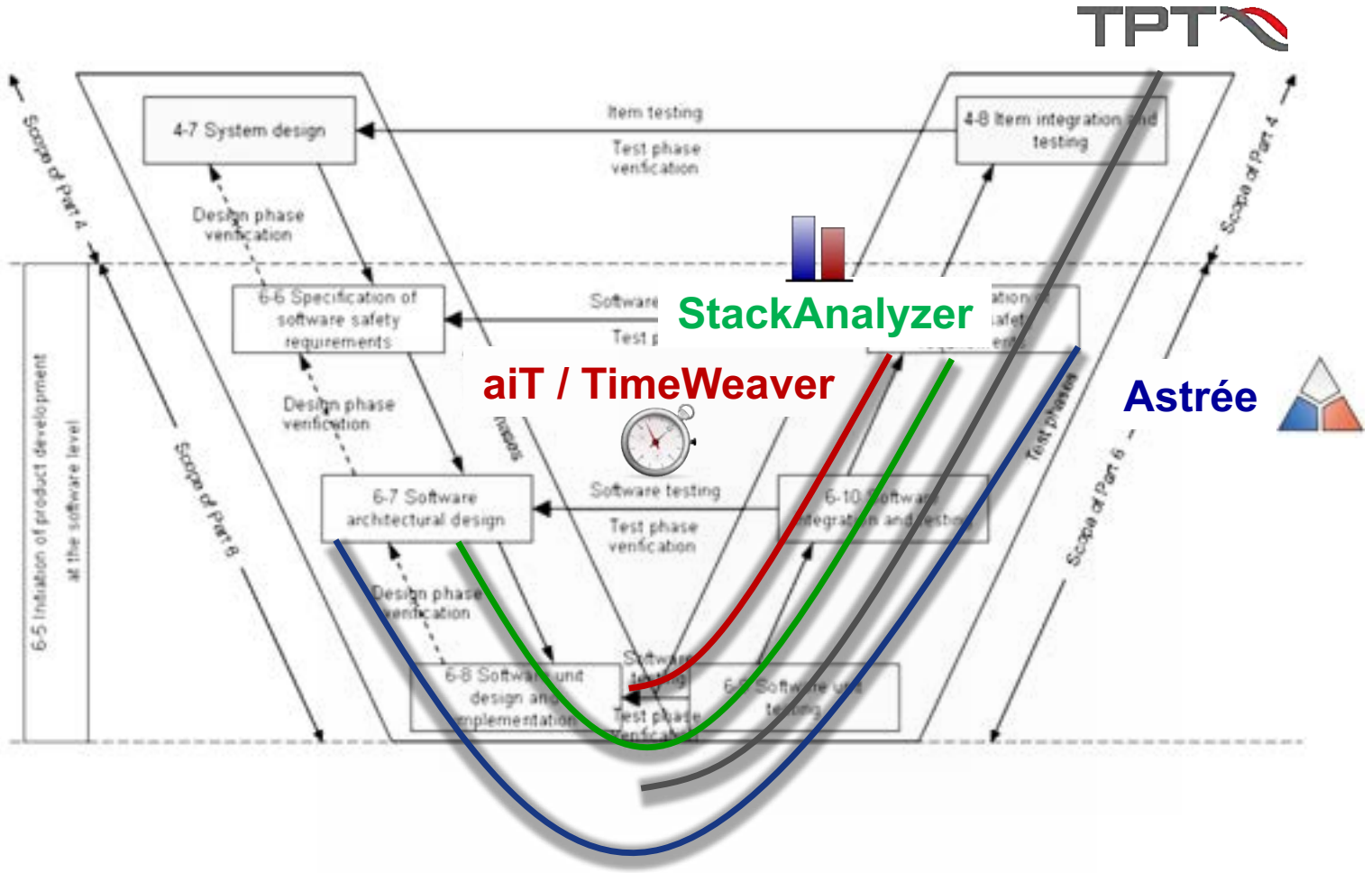

- Valid for all inputs and all execution scenarios
- No modification of your code or tool chain required
- Seamless integration into development tool chain
- Automatic tool qualification, e.g. according to ISO-26262 up to ASIL-D/TCL3.
- Application areas
  - Timing verification
  - Feedback for optimization
  - Software integration
  - Architecture exploration

# StackAnalyzer

- StackAnalyzer reduces risk of failures in the field: no more stack overflows.
- Computed bounds are valid for all inputs and all execution scenarios.
- Analyses the executable binary
- No code instrumentation needed
- Taking into account
  - loops and recursions,
  - inline assembly,
  - object code libraries, and
  - link-time optimizations
- Available for numerous target architectures
- Tool Couplings available, e.g., to dSPACE TargetLink, Esterel SCADE.

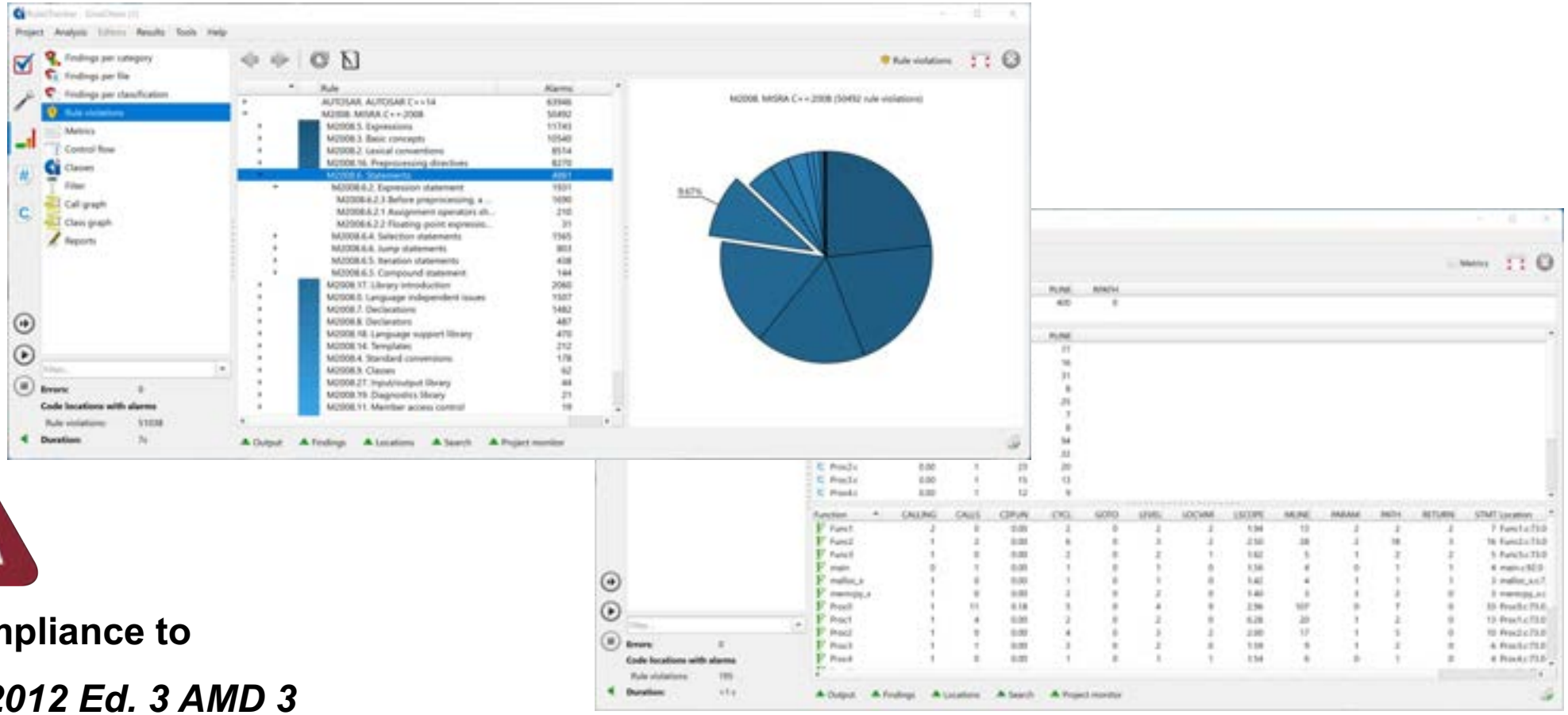


# Development Process




Excerpt from:
*ISO 26262-6 Road vehicles - Functional safety – Part 6: Product development: Software Level, 2011.*

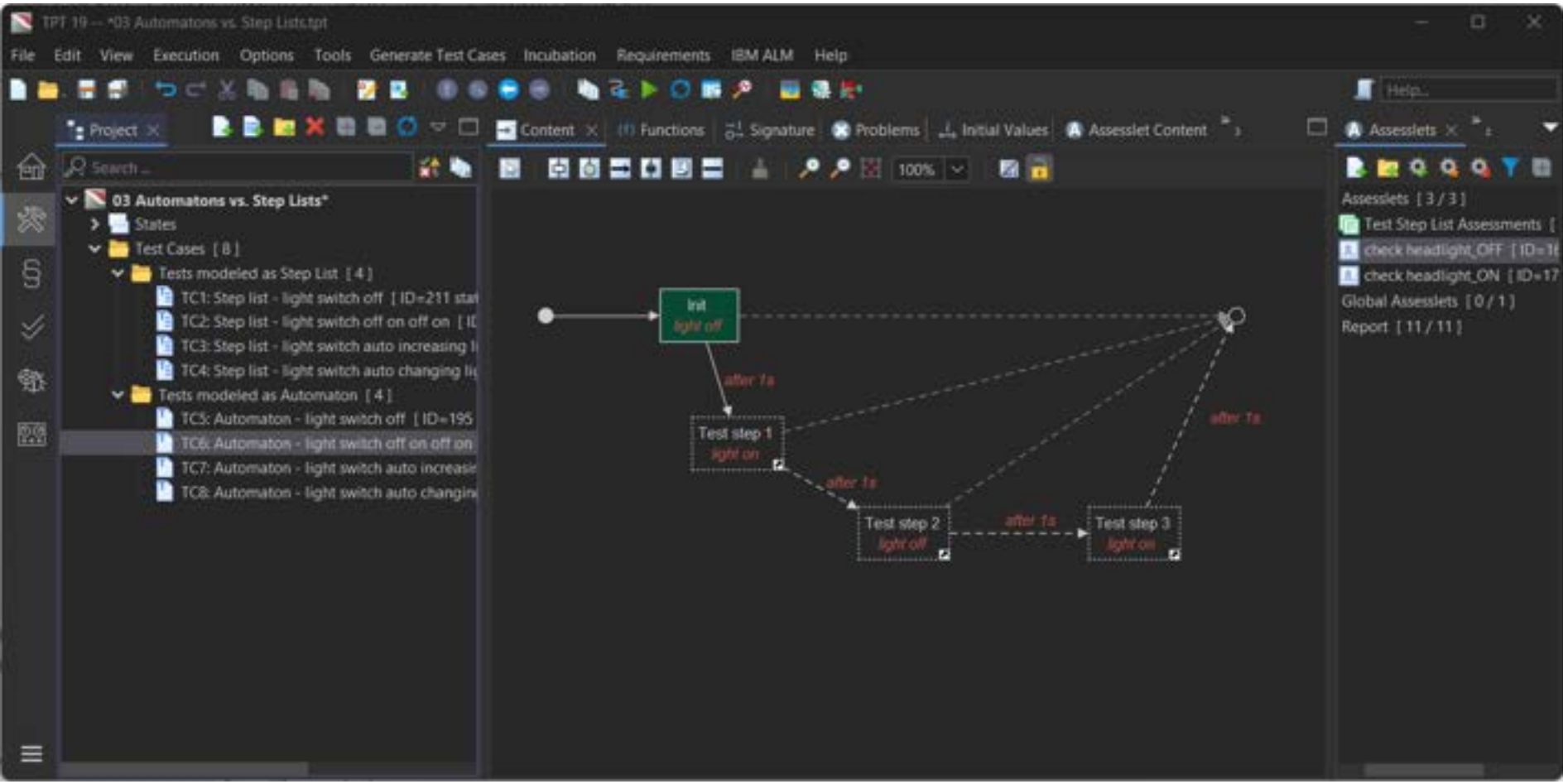
# Demo: Coding Rules



**Check compliance to**

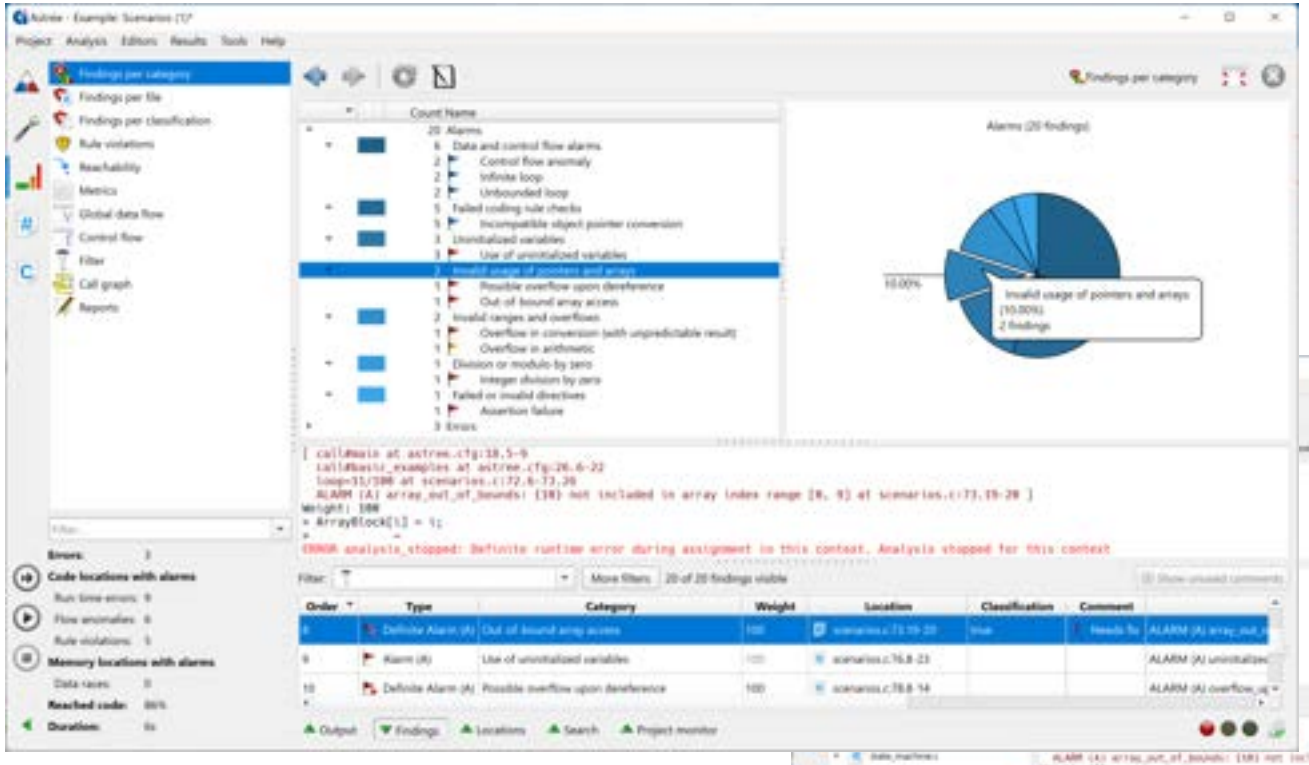
***MISRA C:2012 Ed. 3 AMD 3***

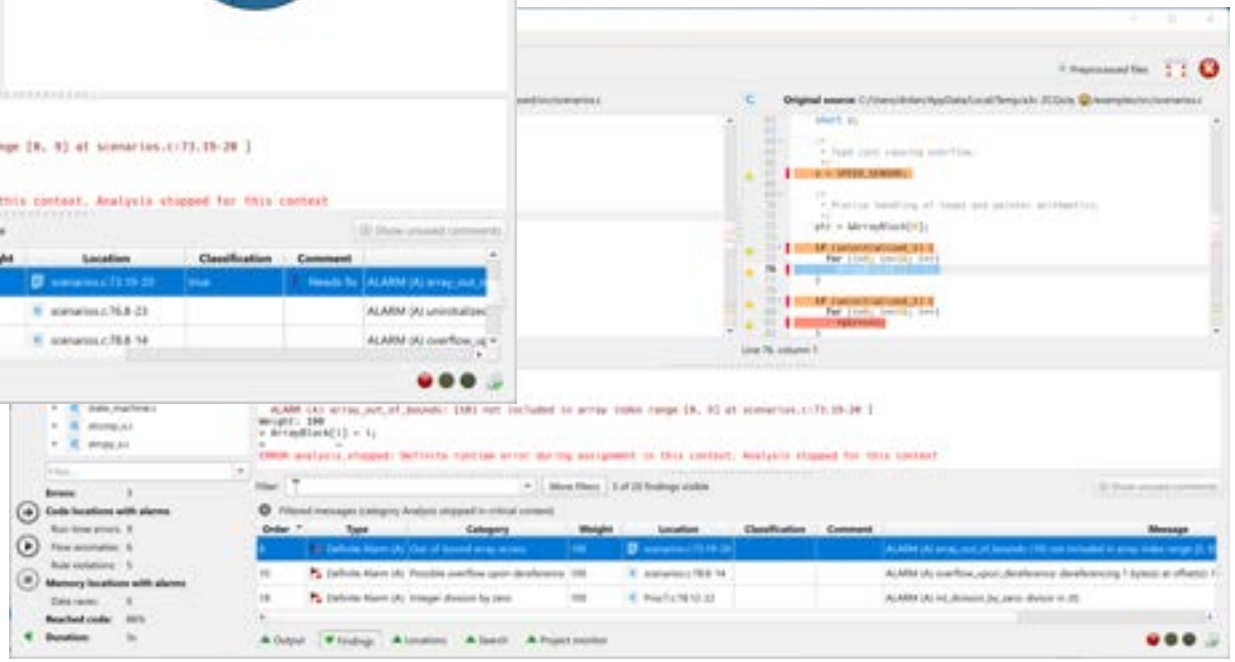
# Demo: Dynamic Testing




- Import Req.
- Testcase Creation
- Testcase **Generation**
- Assessments Creation
- Reporting
- Analyze

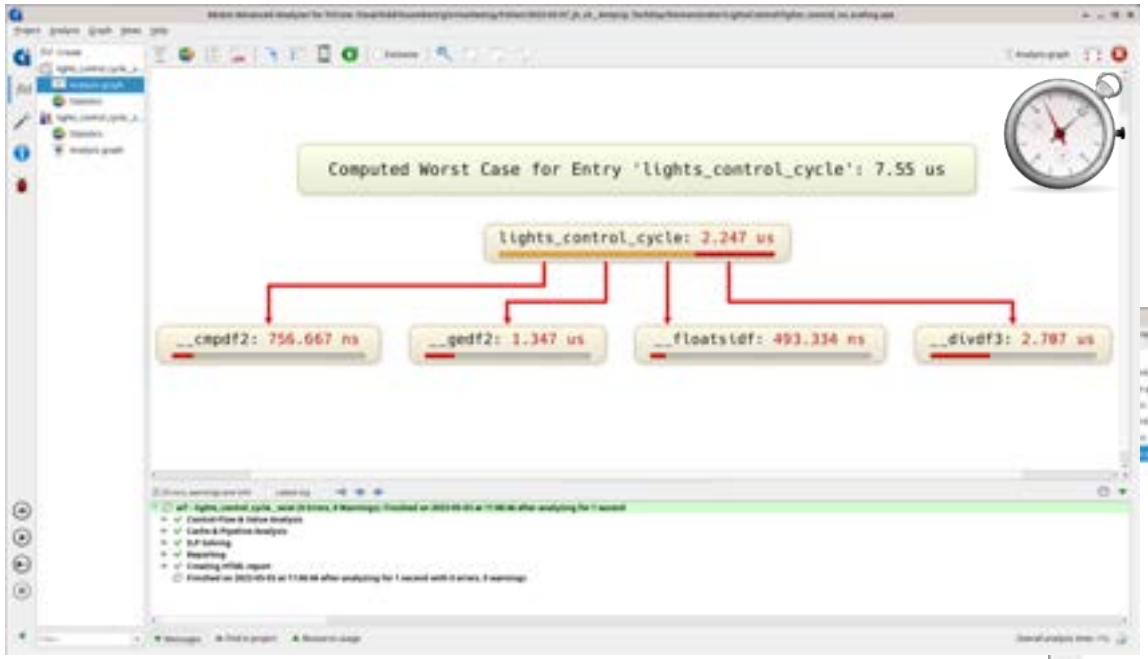
# Demo: RTE/Semantic Analysis

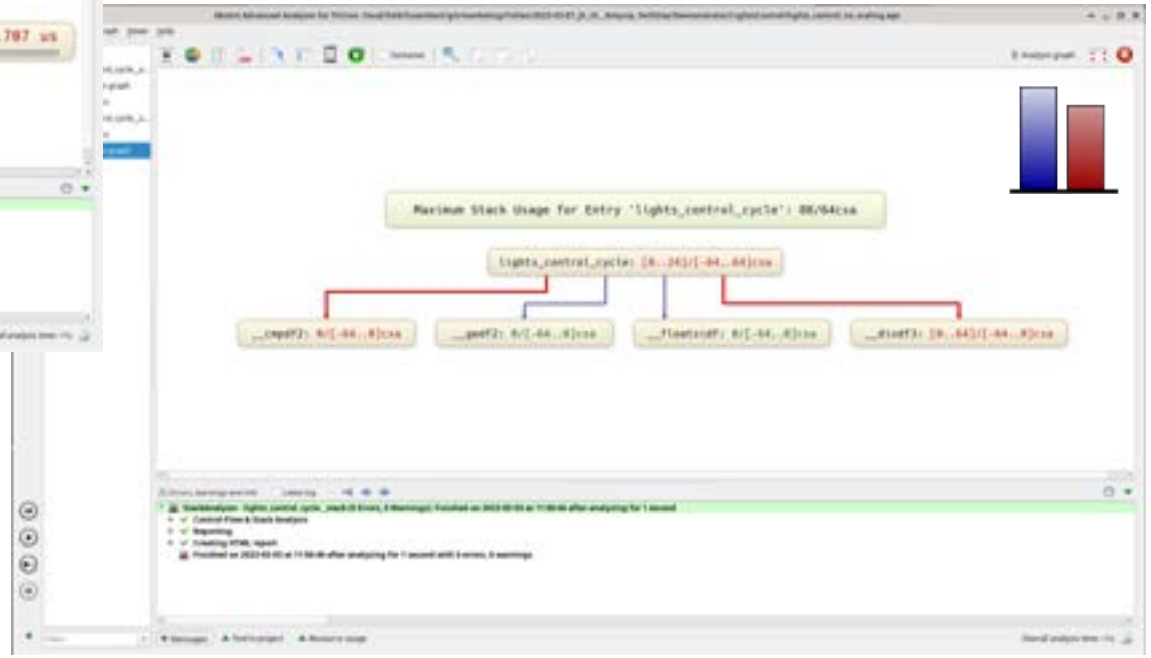




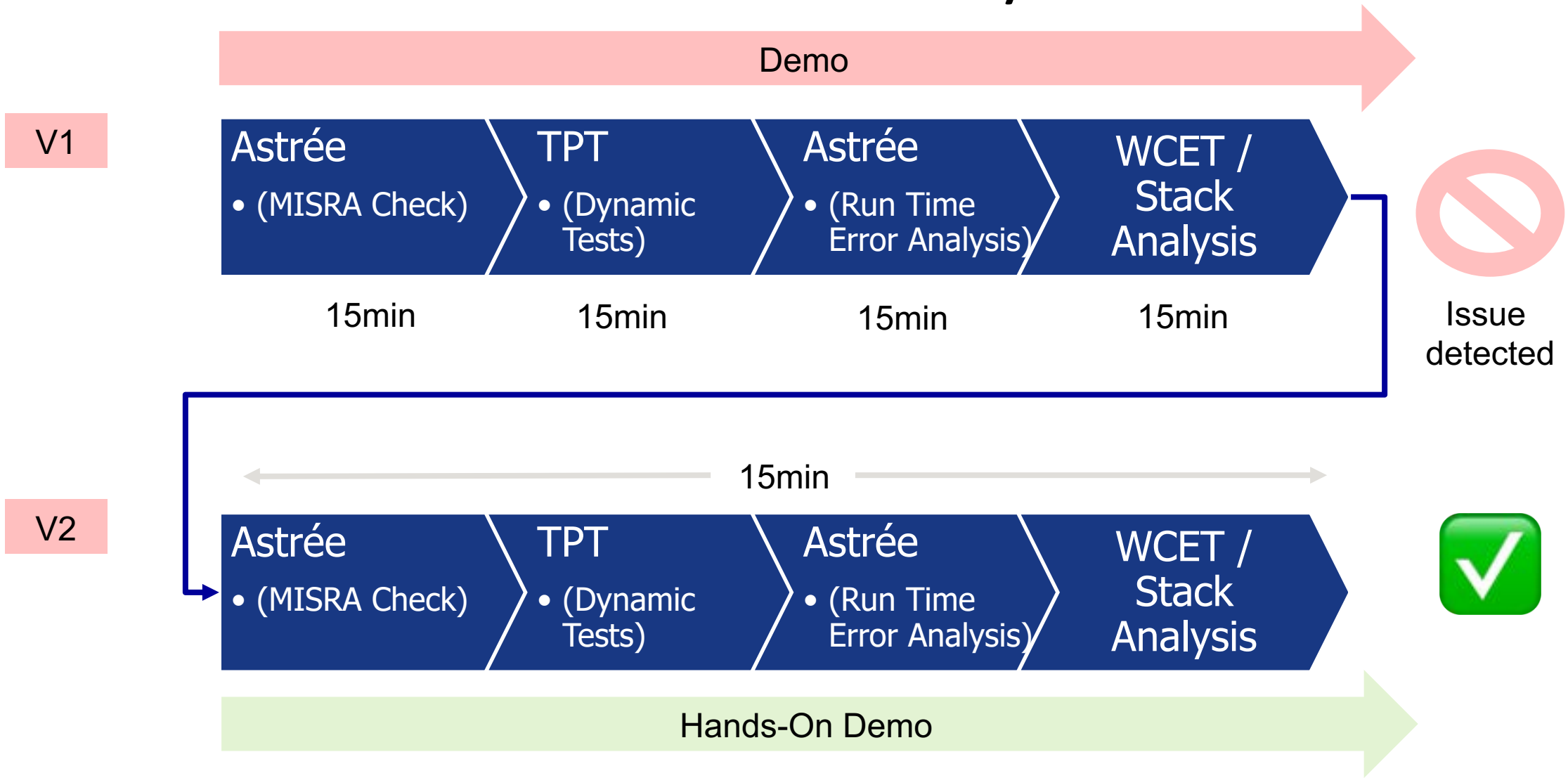
**Guarantee (!) absence of runtime errors and other critical defects**

# Demo: WCET and Stack Analysis

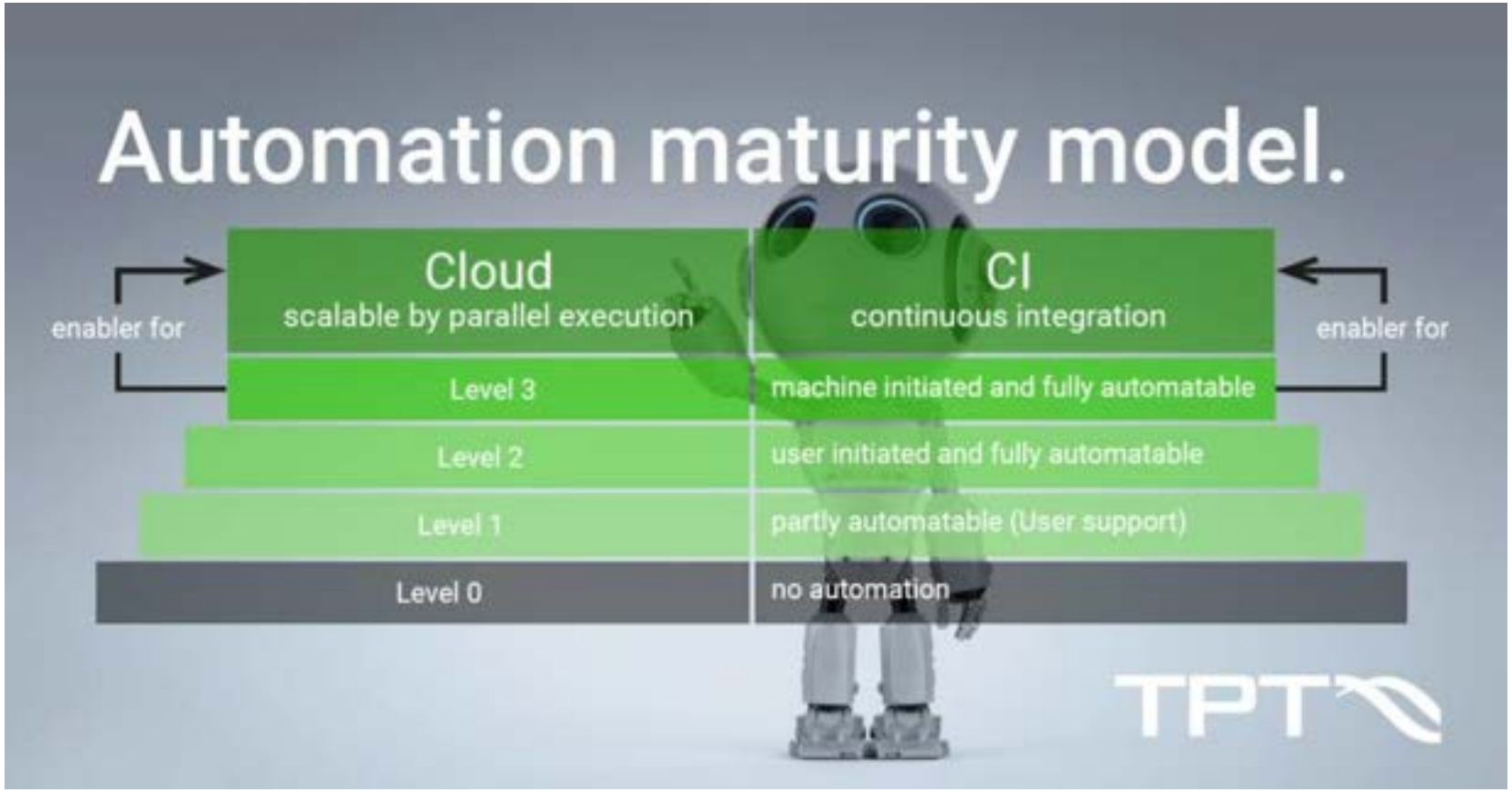


# Demo Summary



# Automation




All Products are ready for use in Continous Integration Environments.

# Questions and Discussions



email: info@absint.com
http://www.absint.com

# The Big Picture – Code-Level Verification

Testing
Model Checking
Theorem Proving
functional correctness

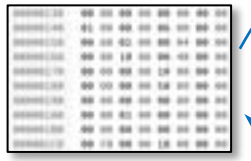
Source Code

Compilation

CompCert/
AbsInt

## Abstract Interpretation

Worst-case execution time analysis (aiT/AbsInt)
No stack overflows (StackAnalyzer/AbsInt)



Executable
Machine Code

Hybrid WCET Analysis
(TimeWeaver/AbsInt)

## Abstract Interpretation

No run-time errors
No data races
Static functional assertions
(Astrée/AbsInt)

## Testing

(TPT/PikeTec)

## Static Analysis for Coding Guidelines

MISRA C/C++, CERT, CWE, C Secure, …
(Astrée/AbsInt)

# Demo: Taint Analysis