TECHNISCHE UNIVERSITÄT
IN DER KULTURHAUPTSTADT EUROPAS
CHEMNITZ

Formal Methods in Design and Test of Integrated Systems

41st Chemnitz Seminar

Test and Reliability Solutions -
New opportunities for electronic components and systems

https://www.tu-chemnitz.de/etit/sse/

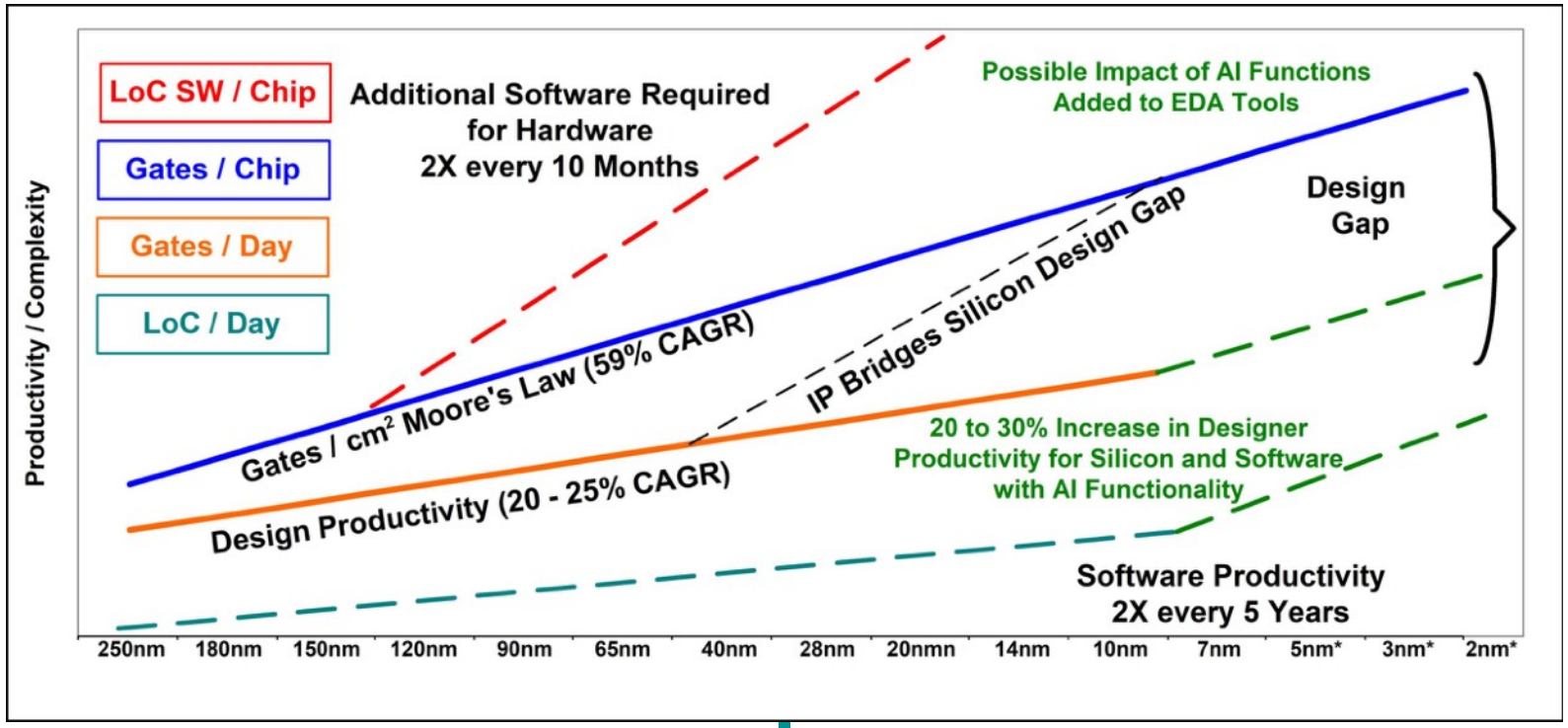
Prof. Dr. Ulrich Heinkel
Professorship Circuit and System Design
University of Technology Chemnitz

Agenda

- Productivity Gap
- V-Model
- What does formal mean
- Example (simpel)
- Formal workflow
- AI for Test

Bridging the Gaps

Source: https://semiwiki.com/semiconductor-services/semico-research/293218-the-impact-of-ai-enabled-eda-tools-on-the-semiconductor-industry/



* compound annual growth rate (CAGR)

- Moore's law: Gates/Chip annual grows rate - doubles every 15-18 Months
- Design Productivity Gates/Day doubles every 3-4 years
- IP reuse: bridges the design gap but verification still needed
- Additional SW needed for HW design/ verification/test for doubling every 10 months

Requirements - Design – Verification – Test Gap:
Complexity outpaces Productivity

Solution Approaches:

Requirements

Design

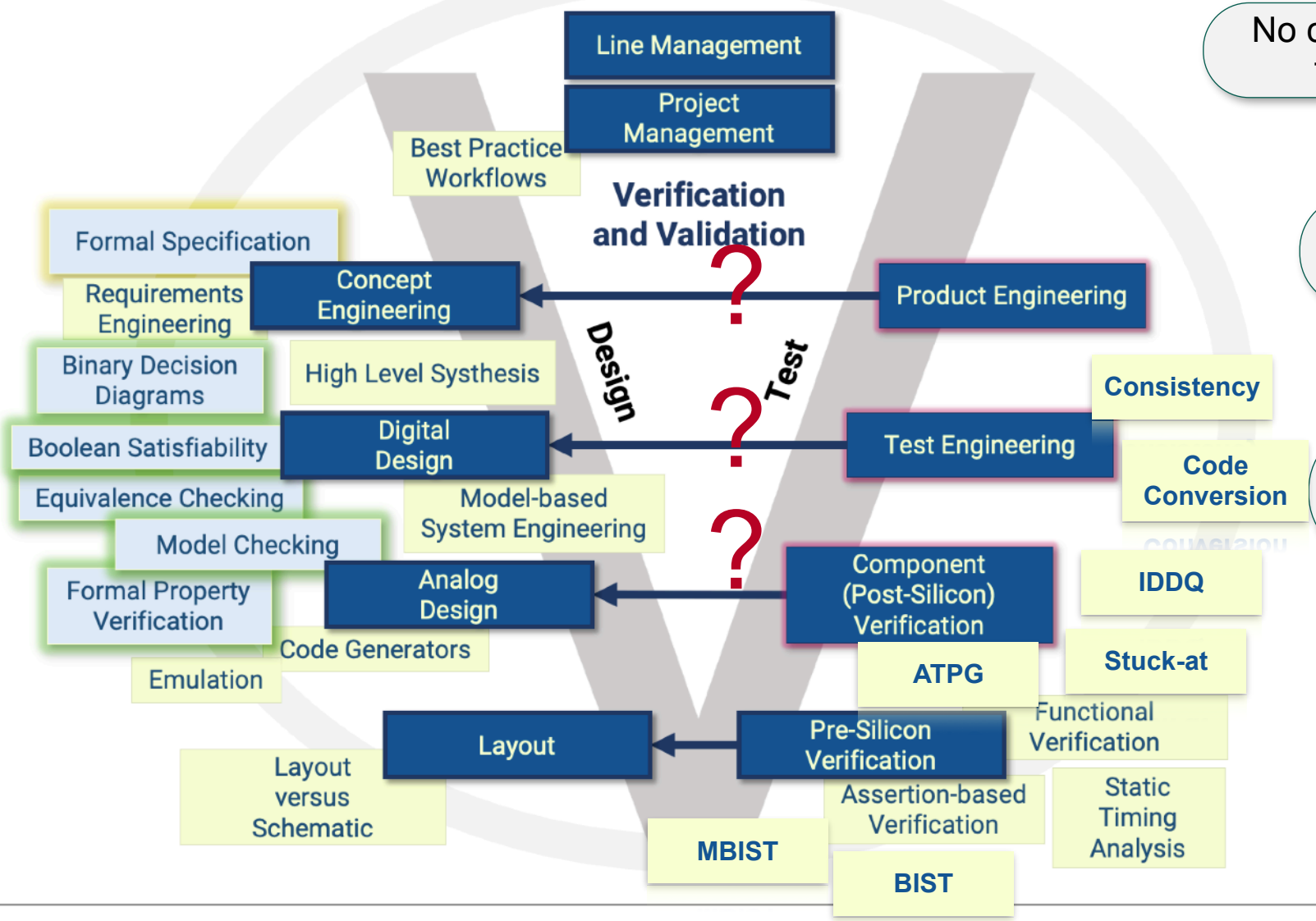
Verification

Test

New methods, new tools
- Specification on high level
- Formal test specification
- Machine-readable specifications
- AI-enabled tools

Match test results on specification
- Backannotation of test results
- Increasing parallelism for quicker test on ATE

Formal V-Model .... CAD (Computer-Aided-Design) ... and Test?



… still room for improvements!

No common data formats to exchange data

Simulation is not sufficient anymore

No propagation of verification and/or test results towards specification

No commonly-used high-level formal test specification format

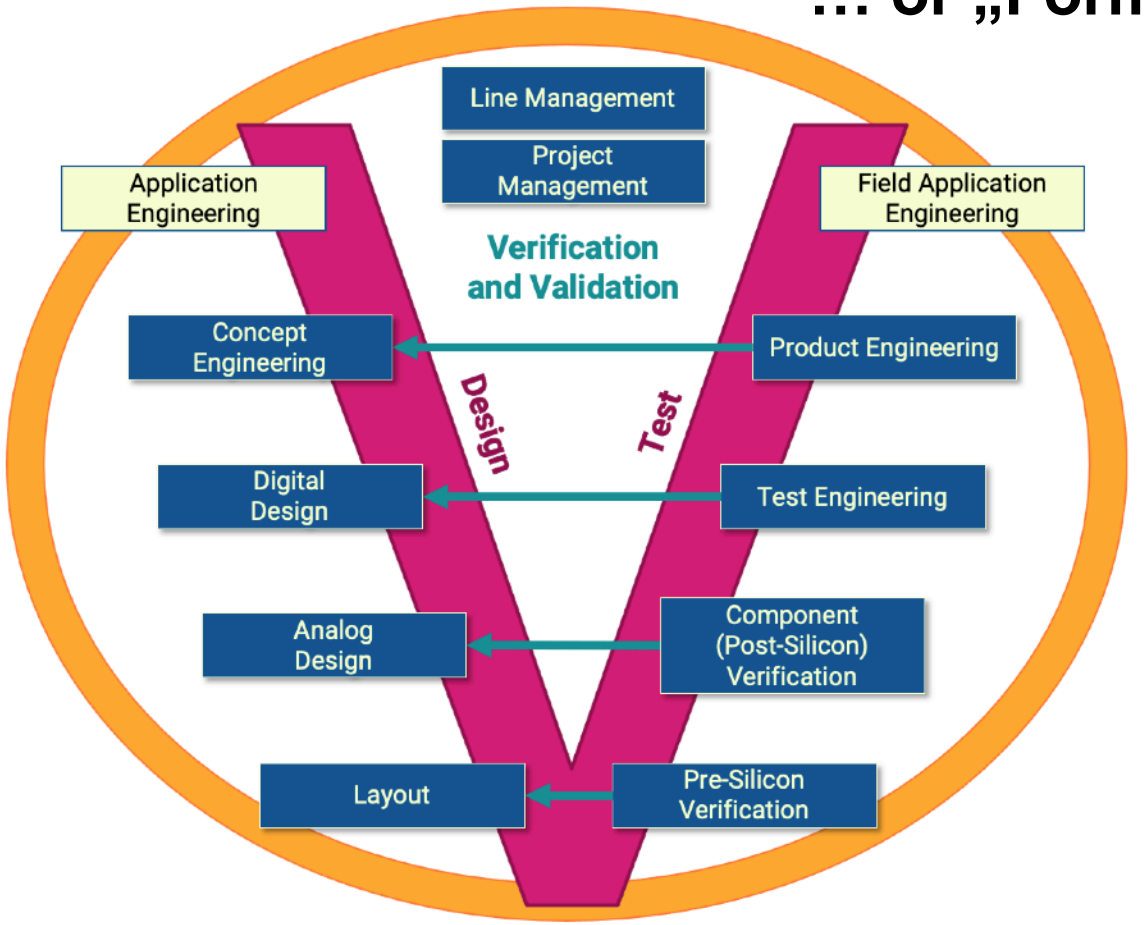
Conversion strategies and tool-independent solutions

→ **A workflow problem**

How can the gaps be bridged?

What does Formal mean …

… or „Formal“ vs. „Formal/Official/Legal/Structured“!



Formal Proof

Equivalence Check
Model Checking
D-Algorithm
Quine-McCluskey
BDD
SAT Solving
…

= Mathematical Proof

Formal "Structured" Process

Methodology framework
Phase model
Waterfall model
V-Model
Spiral model
Agile methodology
..

Keep the controller happy
… but it’s necessary

→ A workflow problem - and we need both "formal" worlds

A German phrase: ***“Wer schreibt, der bleibt”: “Who writes, remains.”***

A formal proof example: Can Robinson escape?

To escape from the island (x_1),

he has to build a boat (x_2)
or he has to be discovered by another ship (x_3). $x_1 \Rightarrow x_2 \vee x_3$

To be able to build a boat (x_2)
he needs wood (x_4). $x_2 \Rightarrow x_4$

To be discovered by another ship (x_3),
he has to make smoke signals (x_5). $x_3 \Rightarrow x_5$

To be able to make smoke signals (x_5),
he needs wood (x_4). $x_5 \Rightarrow x_4$

Robinson can do anything that satisfies this formula.

Robinson's Premise:

$$(x_1 \Rightarrow x_2 \vee x_3) \wedge (x_2 \Rightarrow x_4) \wedge (x_3 \Rightarrow x_5) \wedge (x_5 \Rightarrow x_4)$$

A formal proof example: Can Robinson escape?

SAT-Solving:

The following assignment of variables results in a possible solution:

$x_1=1$	Robinson escapes from the island
$x_4=1$	There is wood
$x_3=1$	He is discovered by another ship
$x_5=1$	He makes smoke signals
x_2 is not determined (don't care)	
	It does not matter if he builds a boat

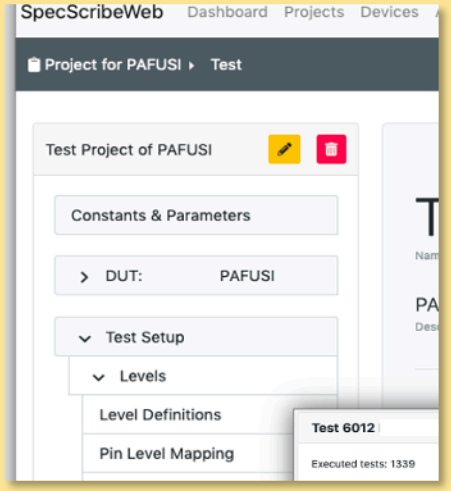
This represents **one** possible escape plan for Robinson.

SpecScribe – A tool for connecting requirements, design, verification and test workflows

Specification first!

Top-down approach

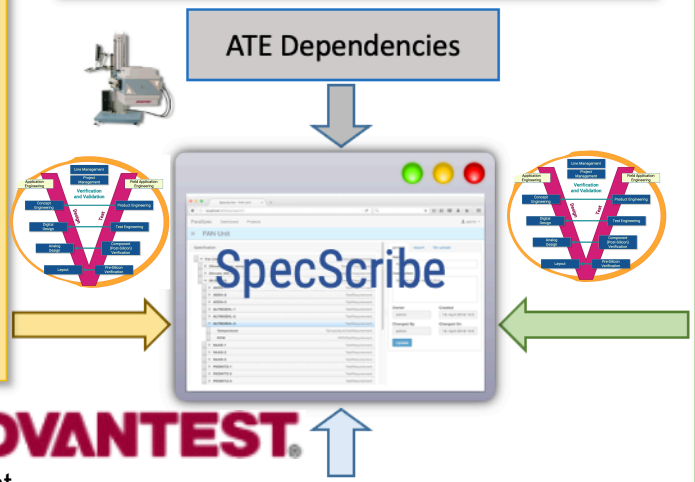
- **Machine-readable specification**: Store formal properties and requirements in machine-readable data formats
- Requirement **tracking**
- Interfaces to **well-known design+verification tools**



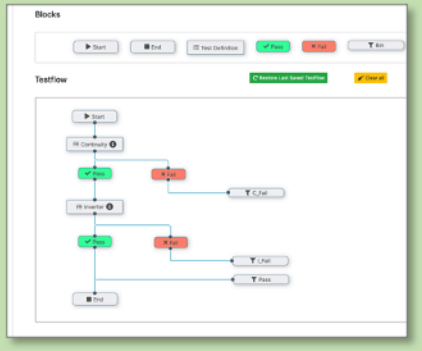
Workflow-driven GUI: GUI-based entry for specification, verification intent and test objectives

Closed-loop approach

- **Forward/top-down:** Generation of verification/test cases
- **Backward/bottom-up:** Backannotation of verification/test results to specification/requirements



Spotlight: Test



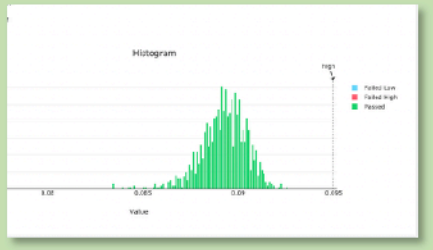
Test flow entry

GUI-based test entry

- Tester-independent descriptions
- Machine-readable test specification
- Expected result definition (Limits etc.)
- Generalization of the target plattform

Analysis

- History
- Yield analysis
- Histograms
- Full-featured test cycle analysis



Pass/fail histogram

Linkage to tools

- Generation of **test setup files**
- Generation of **test programs** (skeletons, C++)
- Test program **verification**

Misc. features

- User management
- Import/export functionality
- Change history
- Technical base: Django web framework + SQL database

✓ UTSL (Universal Test Specification Language)

✓ PSS (Property Test and Stimulus Language)

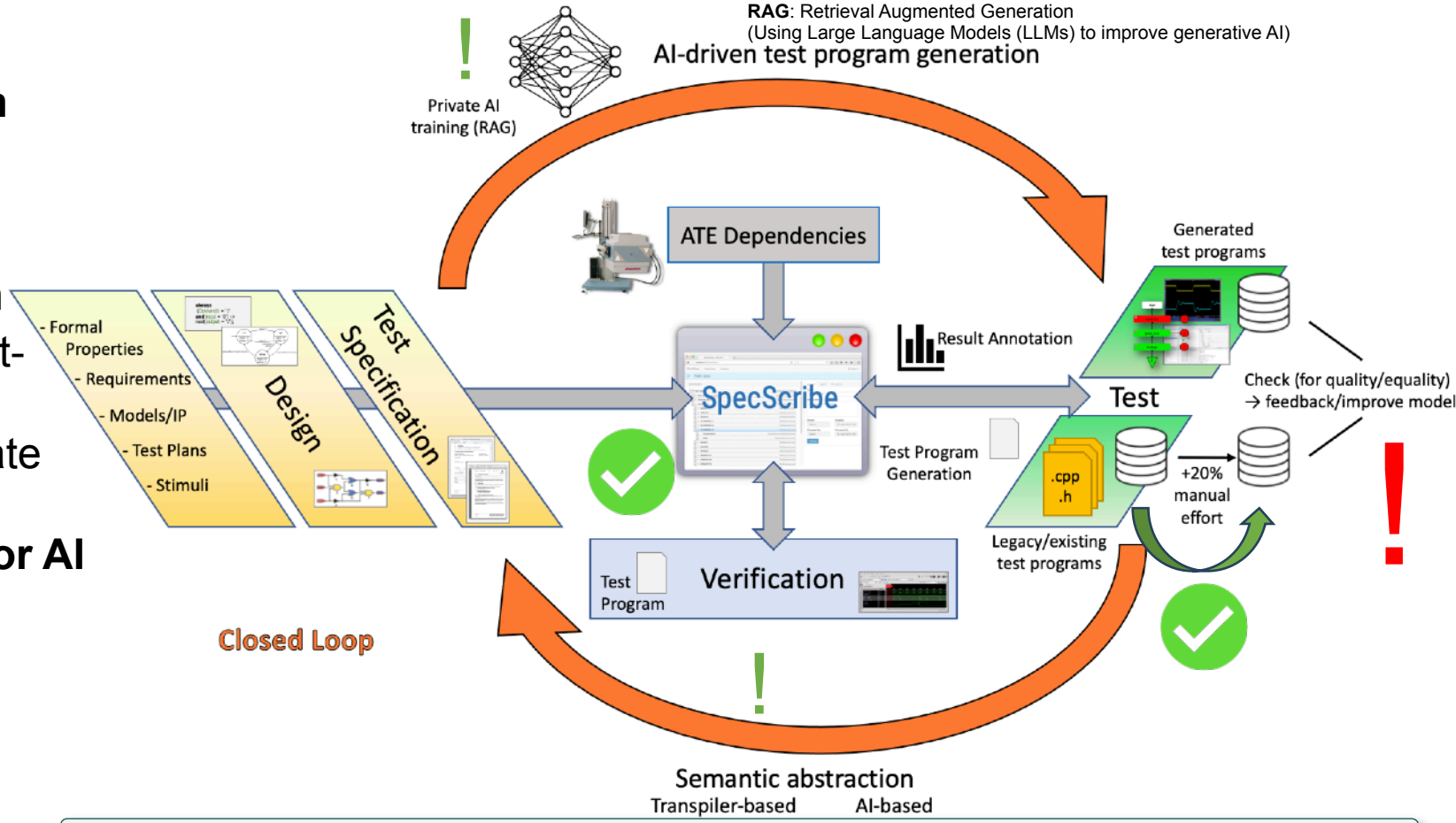
Current research area: Test method conversion

- **Direct conversion:** C++ to Java (Transpiler)
- **AI-based analysis** of test specifications (towards machine-readable specification)

A look into the future … more AI?

Workflow for AI-driven TP-Conversion

- **Local instances – private data**
- **Closed Loop for AI-Training**
  - Get a golden model for comparison
  - Analyzing semantics of existing test-programs
  - RAG: Separate AI model from private data
  - **Workflow Testcases/Questions for AI**
  - **Self-checking Testbench**



Actually not in focus (the "20%"):

- Tester API translation
- Test methods

Transpiler ADVANTEST

- Test Program Conversion between C++ and Java source code (source code translation)
- Reduction of manual effort for test program conversion in case of a test system or software environment switch

Workflow for Design and Test Processes of the future

- Meet in the Middle: Top down, but also inside out and bottom up
- Machine-readable Specifications
- Workflows over all levels of abstraction
- Backannotation of results
- Open/standardized interfaces to well know tools and flows
- Export/Import (VHDL, Verilog, STDF, PSS,...)
- GUI for visualization of results
- Version control
- AI-ready interfaces + AI-integrated features
- ...