Verification & Validation with Model-Based Design

Lyle Shipton
Application Engineer
MathWorks
Plano, TX

Background

- University of Illinois at Urbana-Champaign
  - B.S, M.S. Aerospace Engineering
- SpaceX Rocket Development Facility
  - Test Engineer
  - Lead Engineer, Integration & Test
- Eaton Aerospace, Fuel and Motion Controls
  - Lead Aerospace Systems Engineer
- MathWorks, Application Engineering Group
  - Lead Engineer, Aeronautical applications

MathWorks at a Glance

- Privately held
- 4000 employees worldwide
- More than 3 million users in 180+ countries

- Office locations
- Distributors serving 16 countries

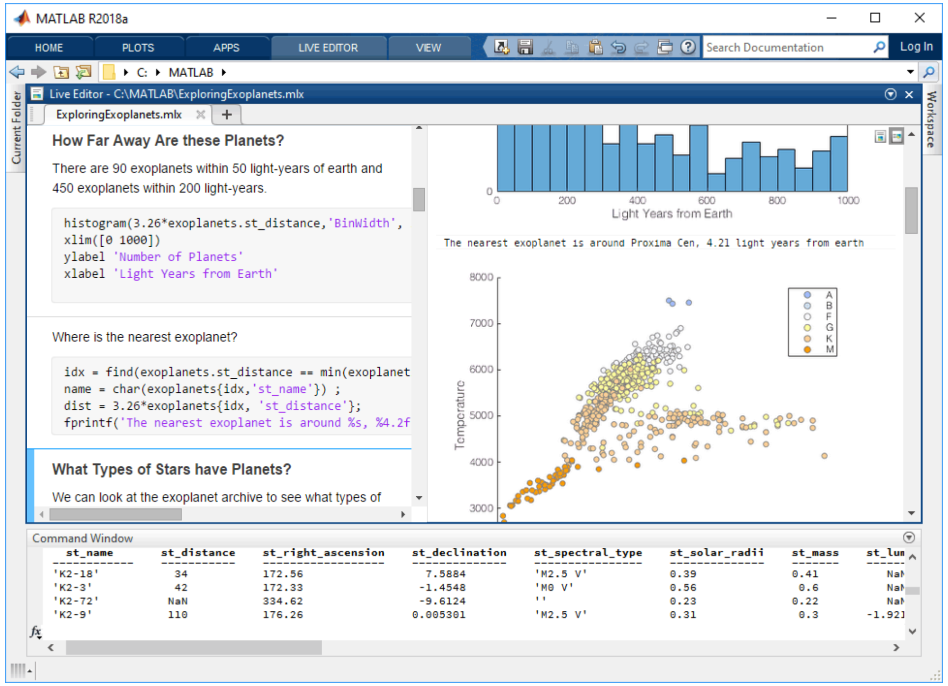
MathWorks

Core MathWorks Products

MATLAB

Math. Graphics. Programming.

- Designed for engineers and scientists
- Professionally developed, tested, and documented
- Toolboxes for:
  - Machine learning, data analytics, deep learning, image processing and computer vision, signal processing and communications, computational finance, robotics and control systems
- Interactive apps that automatically generate programs
- Easily scales to clusters, GPUs, and clouds
- Direct deployment to production enterprise applications
- Automatic conversion to embeddable C and CUDA code
- Integrates with Simulink to support Model-Based Design



MathWorks

Core MathWorks Products

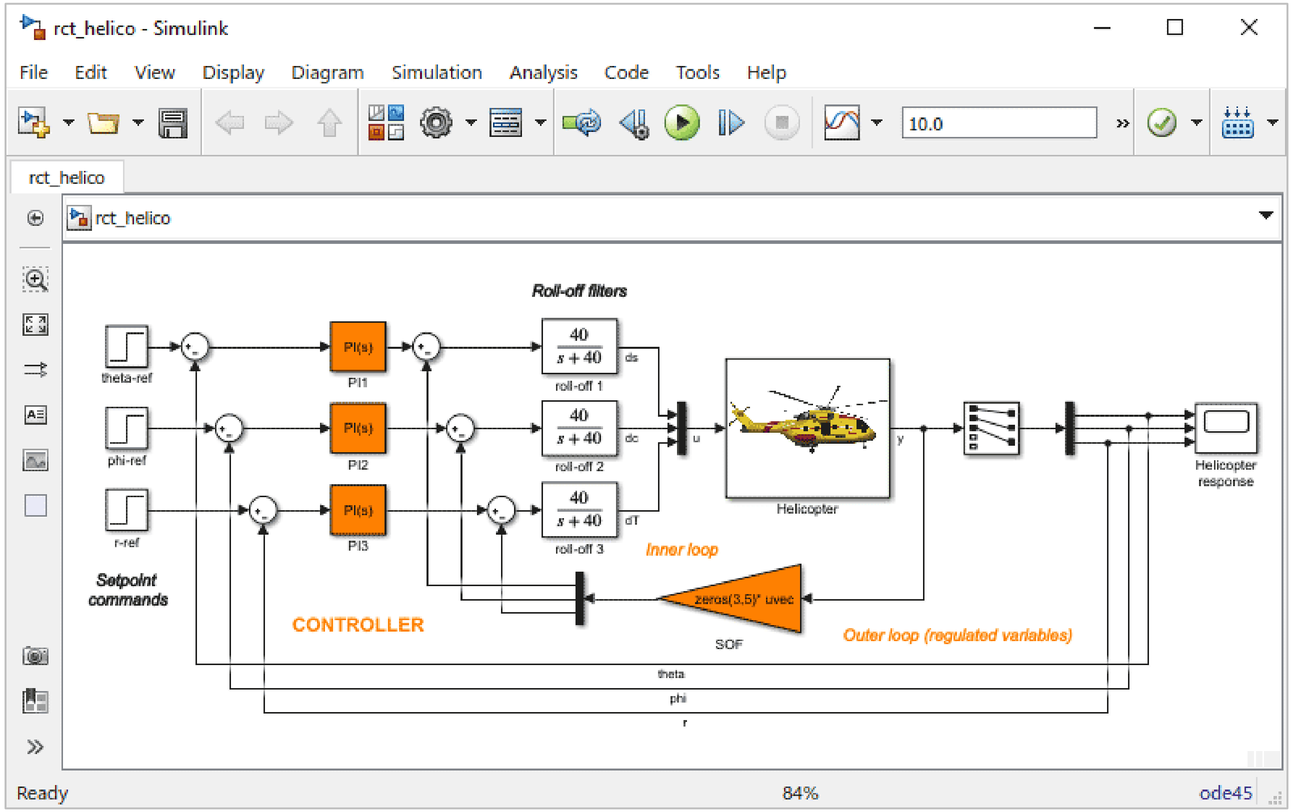
SIMULINK®

Simulation and Model-Based Design

Model and simulate your system
- Use one multi-domain environment
- Model the system under test and the plant
- Simulate how all parts of the system behave

Test early and often
- Test your system under all conditions
- Validate your design with real-time testing
- Trace from requirements to design to code

Automatically generate code
- Generate production-quality C and HDL code
- Deploy directly to embedded processors or FPGA’s/ASIC’s



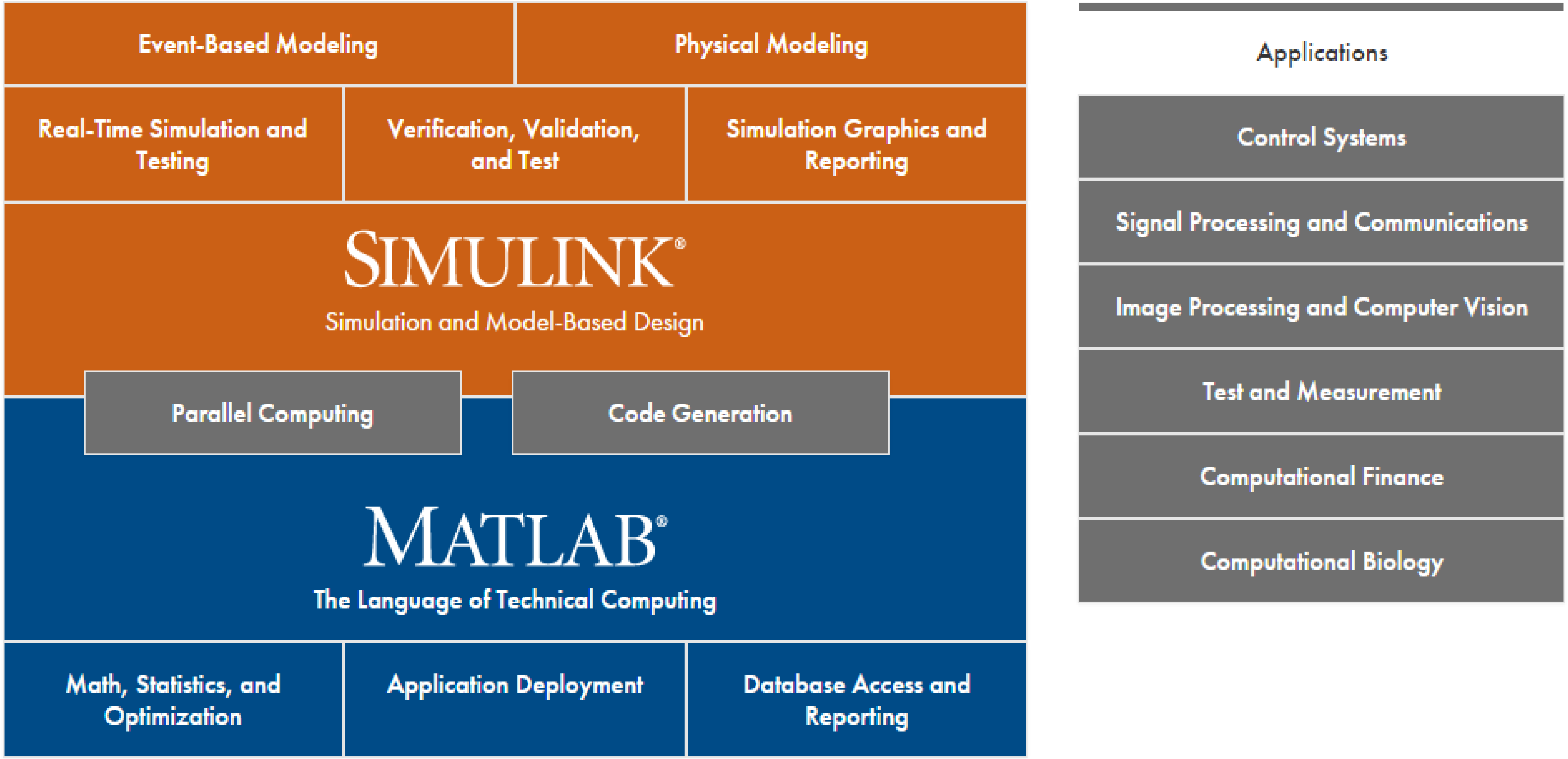
MathWorks

Key capabilities for engineers and scientists

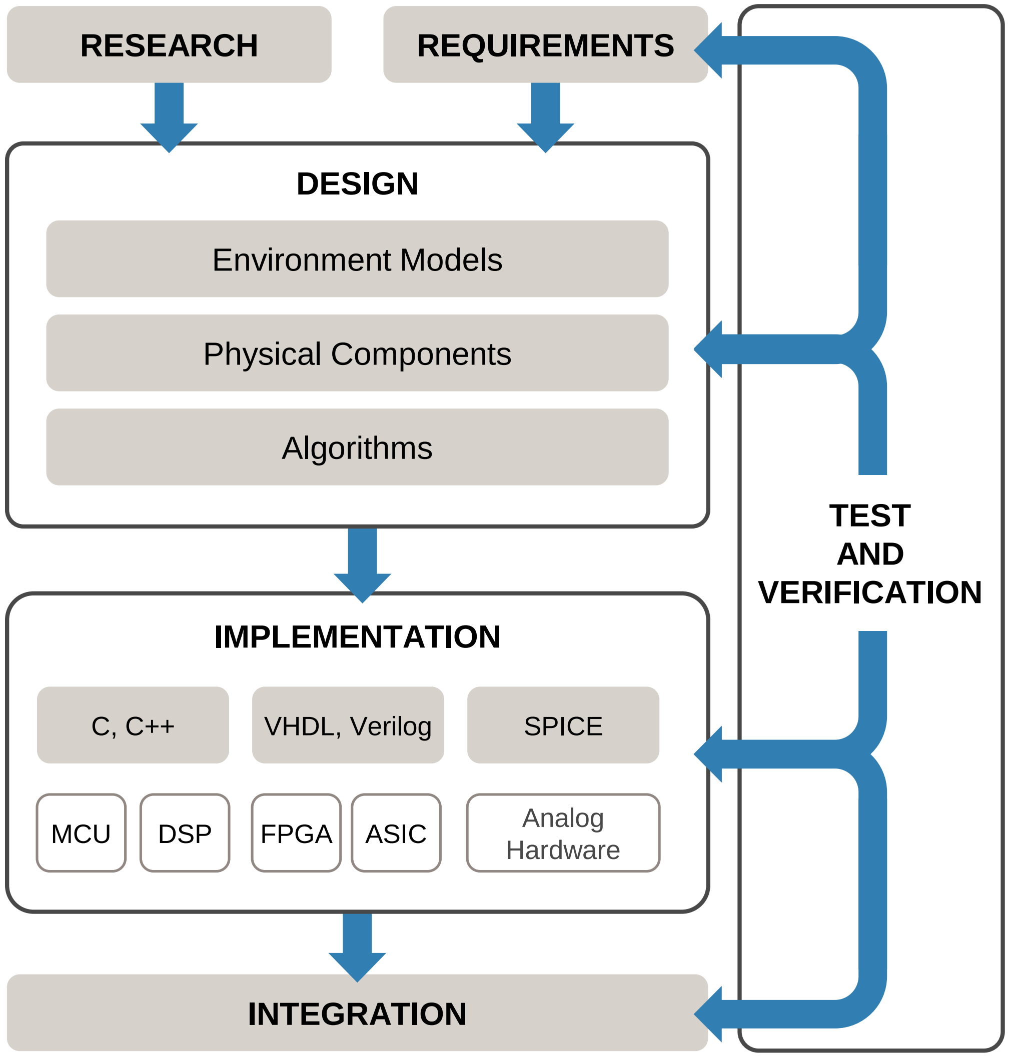
	1985	1990	1995	2000	2005	2010	2015
Verification, Validation, and Test				• Test and measurement	• Model checking	• Code verification • Certification kits • HDL verification	• Test automation • Requirements authoring & mgmt.
Automatic Code Generation			• Rapid prototyping and HIL	• Embedded code • Hardware support packages		• PLC code • HDL code • MATLAB to C/HDL	• HW/SW co-development • GPU code
System Modeling and Simulation		**Simulink**	• DSP designs • Communications systems	• State charts • Physical modeling	• Discrete-event simulation • Video processing	• Computer vision • RF • Phased array	• Robotics and autonomous systems • WLAN/LTE protocols
Data Analysis and Algorithm Development	• Control design • Signal processing	• Optimization • Statistics	• Image processing	• Computational finance	• Computational biology	• Machine learning	• Deep learning • Sensor fusion • Text analytics
Technical Computing	**MATLAB**		• Application deployment	• Student version • Instrument and database connectivity	• Parallel computing	• MATLAB Mobile for phones/tablets • MATLAB Online	• Big data • AWS & Azure support • Enterprise integration • ThingSpeak for IoT

MathWorks founded in 1984

MathWorks Product Overview

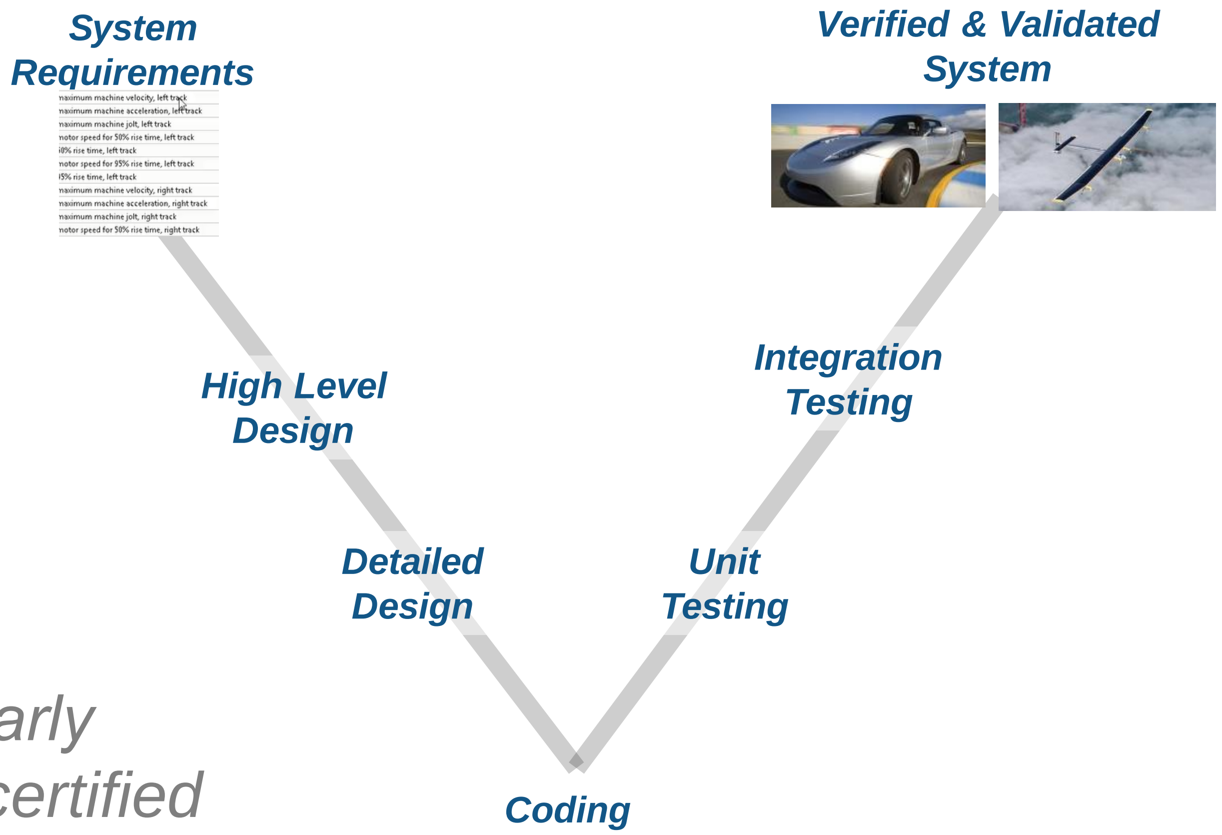


Model-Based Design Workflow



Key Takeaways

1. Find bugs early, develop high quality systems
2. Replace manual verification tasks with workflow automation
3. Learn about reference workflow that conforms to safety standards



"Reduce costs and project risk through early verification, shorten time to market on a certified system, and deliver high-quality production code that was first-time right"   Michael Schwarz, ITK Engineering

Safety of Electronic Systems

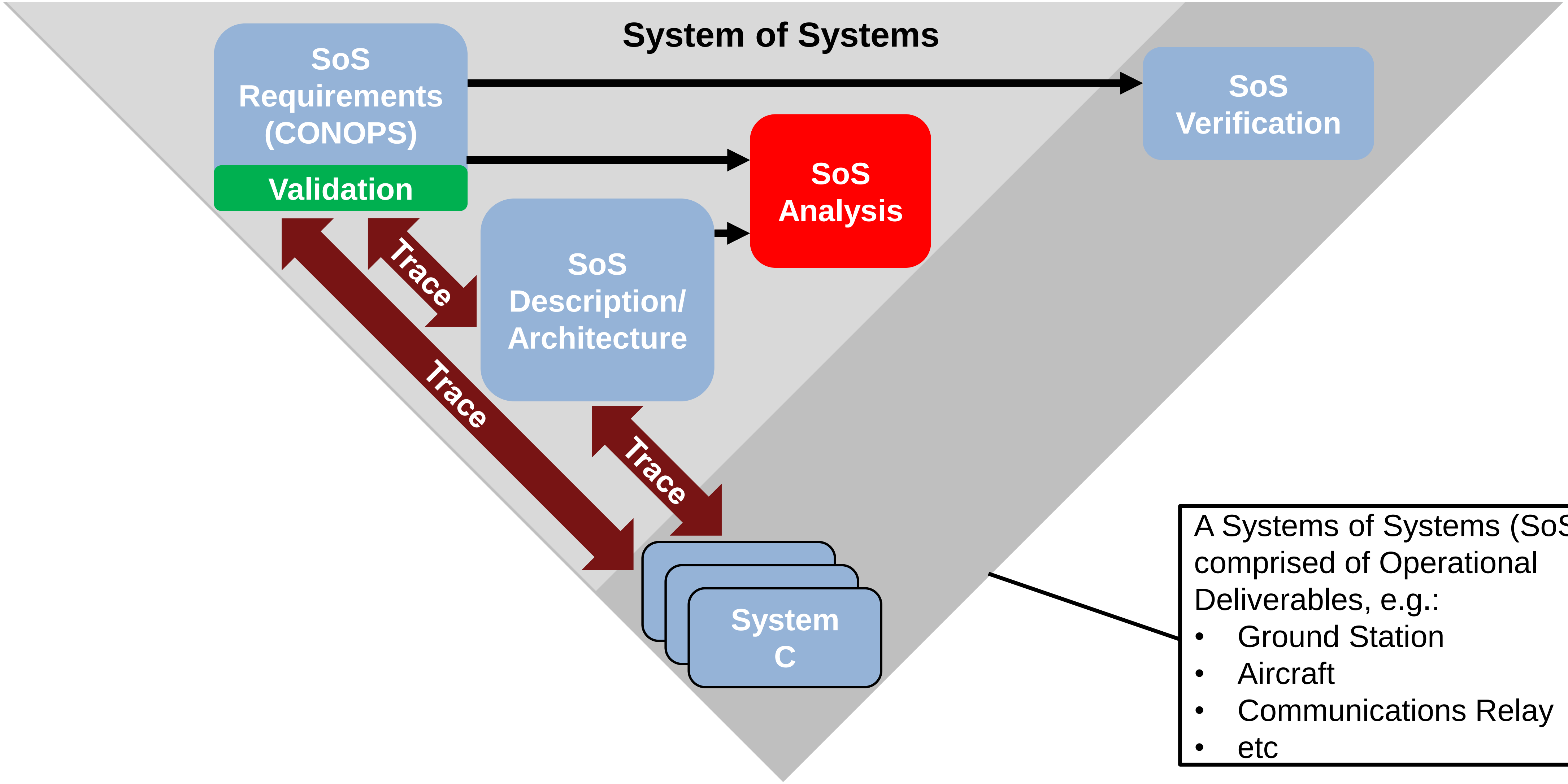
- Critical functionality in industries such as Aerospace, Automotive, and Industrial Automation
- Real-time operation
  – Compute time lag cannot be tolerated
- Predictable behavior
  – No unintended functionality
- Must be robust
  – Program crash or reboot not allowed

Role of Certification Standards

- DO-178 (Avionics)
  - Guidelines for the safety of software in certain airborne systems
  - Level A to E (most critical to least)
  - Verification activities include review of requirements and code, testing of software, code coverage
- ISO 26262 (Automotive)
  - Defines functional safety for automotive electronic systems
  - Automotive Safety Integrity Level ASIL QM, A to D (least to most; derived from severity, controllability, probability)
  - ISO 26262-6 pertains to software development, verification, and validation
- IEC 61508 (Industrial Automation & Machinery)
  - General functional safety standard, originally for process control industry
  - Safety Integrity Level SIL 1 to 4 (least to most; derived from exposure to demand needs and probability of failure)
  - Defines the software requirements and lifecycle for software, that includes validation and verification

Reference Verification and Validation Workflow

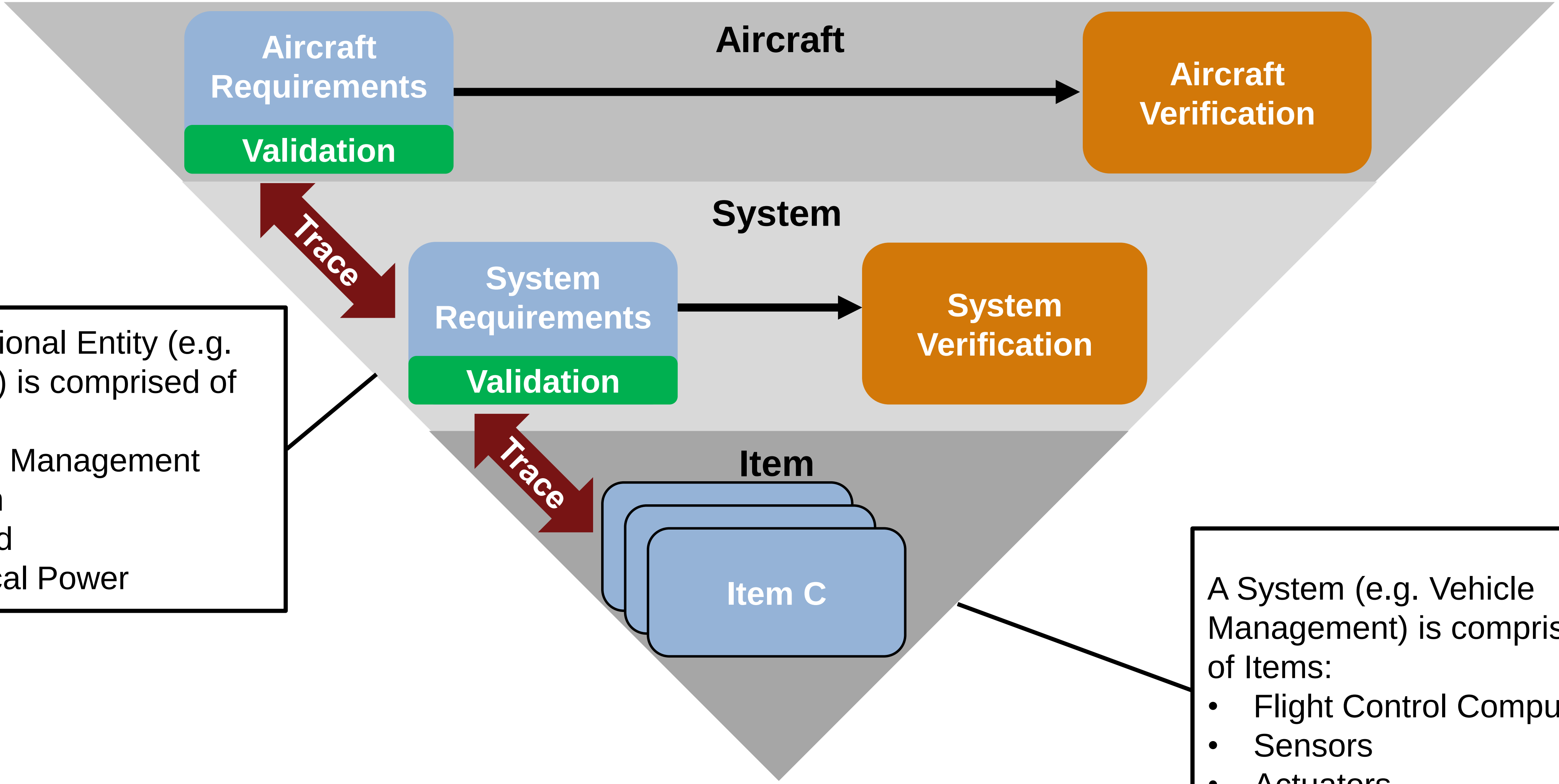
Development Process and Workflow



A Systems of Systems (SoS) is comprised of Operational Deliverables, e.g.:

- Ground Station
- Aircraft
- Communications Relay
- etc

Development Process and Workflow



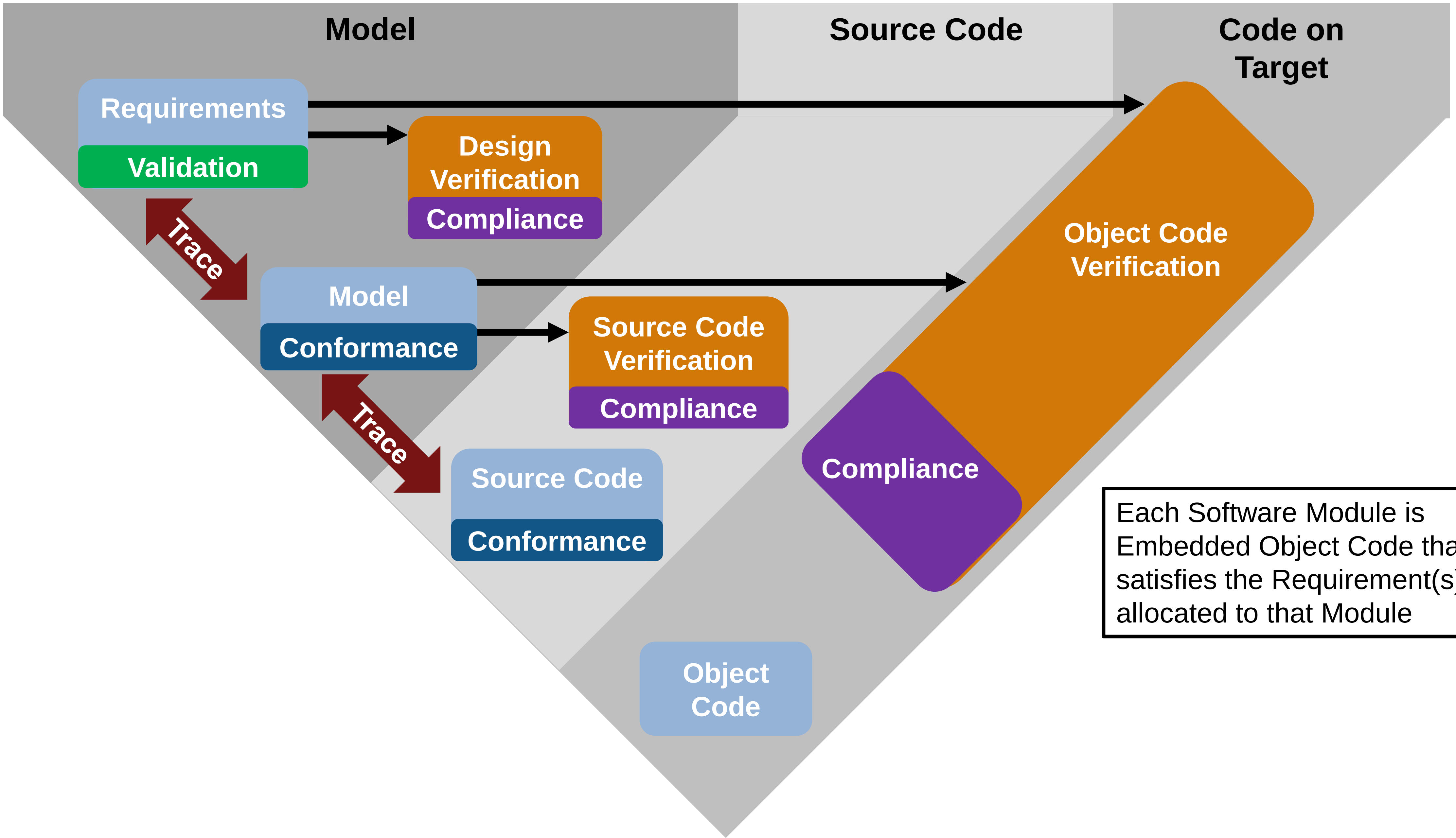
An Operational Entity (e.g. an Aircraft) is comprised of Systems:

- Vehicle Management System
- Payload
- Electrical Power

A System (e.g. Vehicle Management) is comprised of Items:

- Flight Control Computer
- Sensors
- Actuators

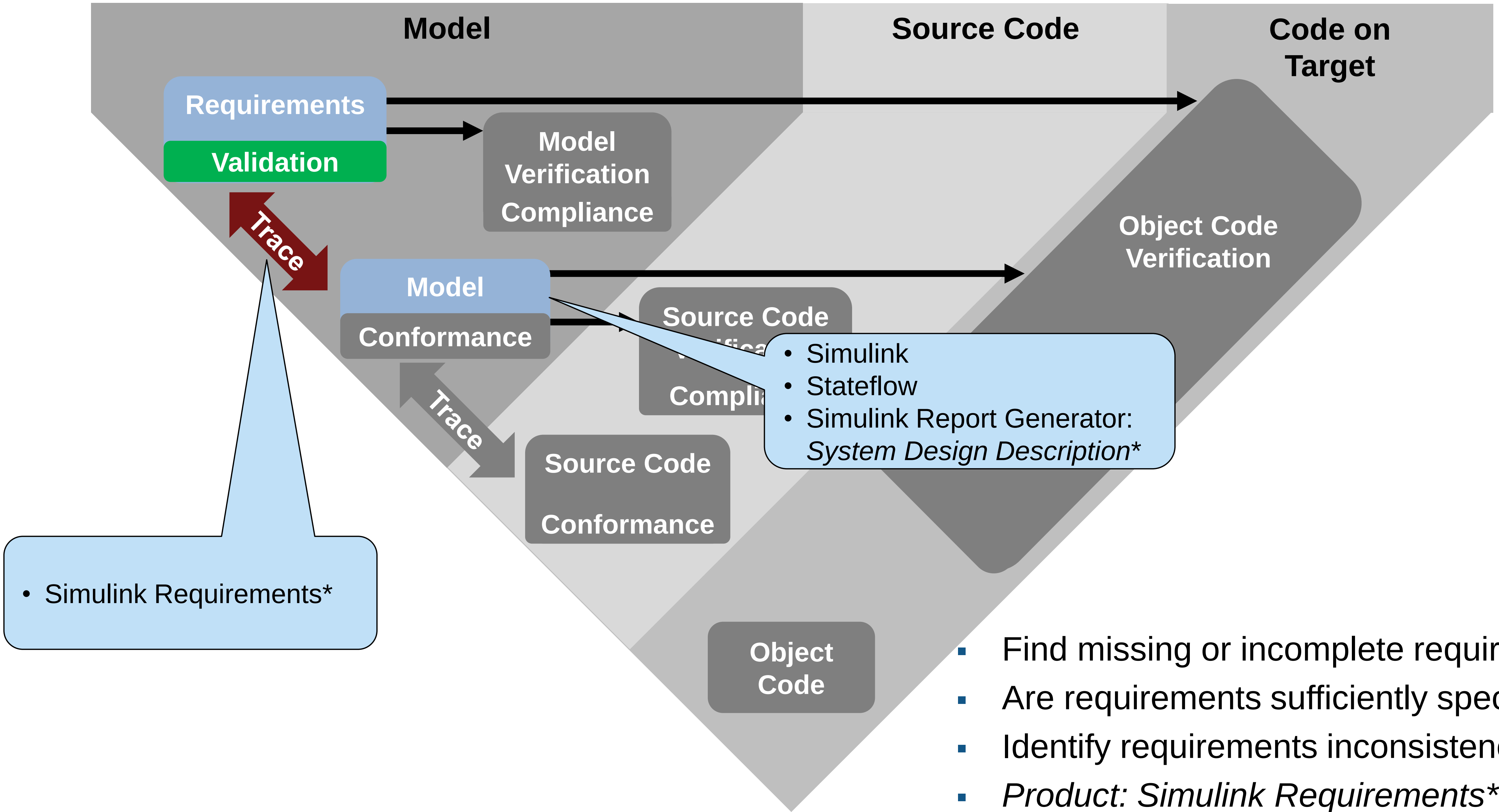
Development Process and Workflow



Each Software Module is Embedded Object Code that satisfies the Requirement(s) allocated to that Module

Verification and Validation Tasks and Activities

Verification and Validation Tasks and Activities



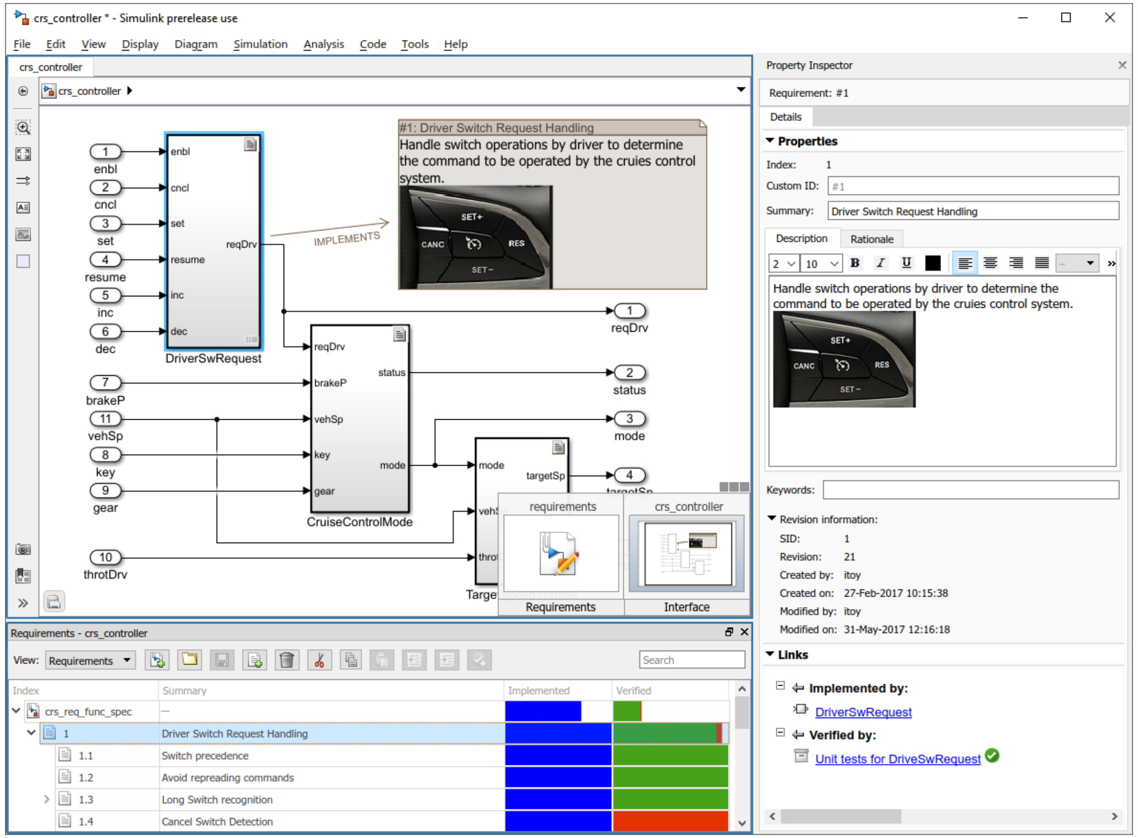
- Find missing or incomplete requirements
- Are requirements sufficiently specified
- Identify requirements inconsistencies
- *Product: Simulink Requirements**

Simulink Requirements

Work with requirements and design together

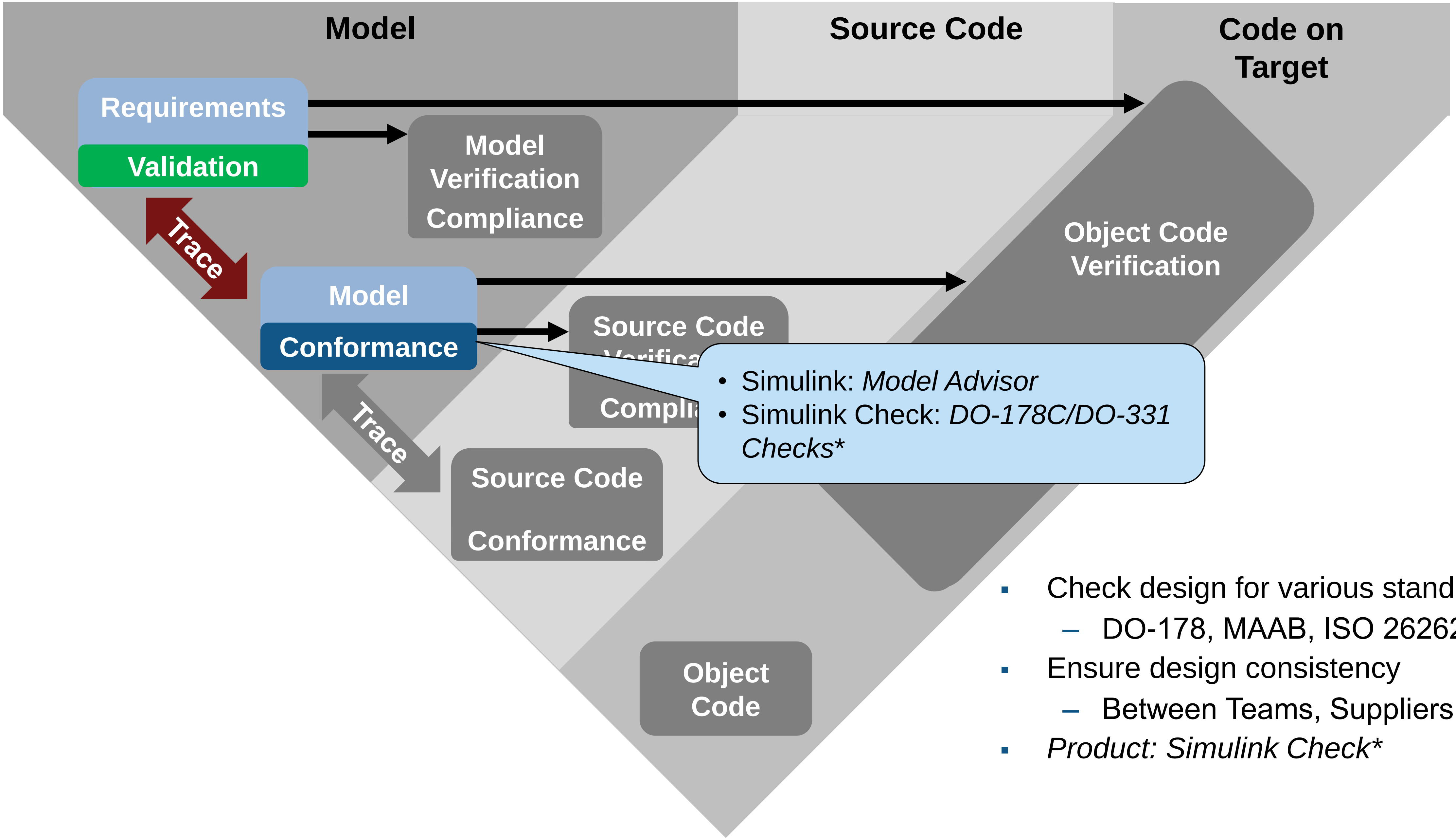
R2017b

- Author, edit and organize requirements
- View and link requirements within the Simulink graphical editor
- Track status and manage requirement changes
- Trace requirements to models, generated code, and test cases



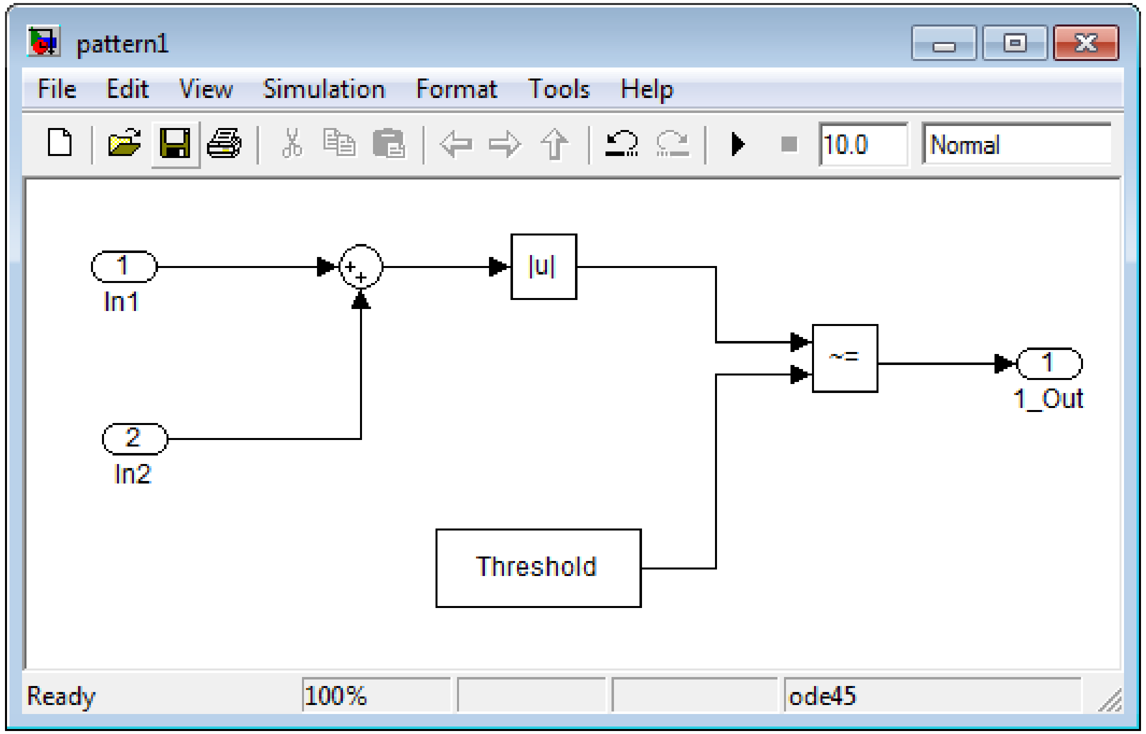
Requirements Perspective

Verification and Validation Tasks and Activities



- Check design for various standards
  - DO-178, MAAB, ISO 26262, …
- Ensure design consistency
  - Between Teams, Suppliers, …
- *Product: Simulink Check**

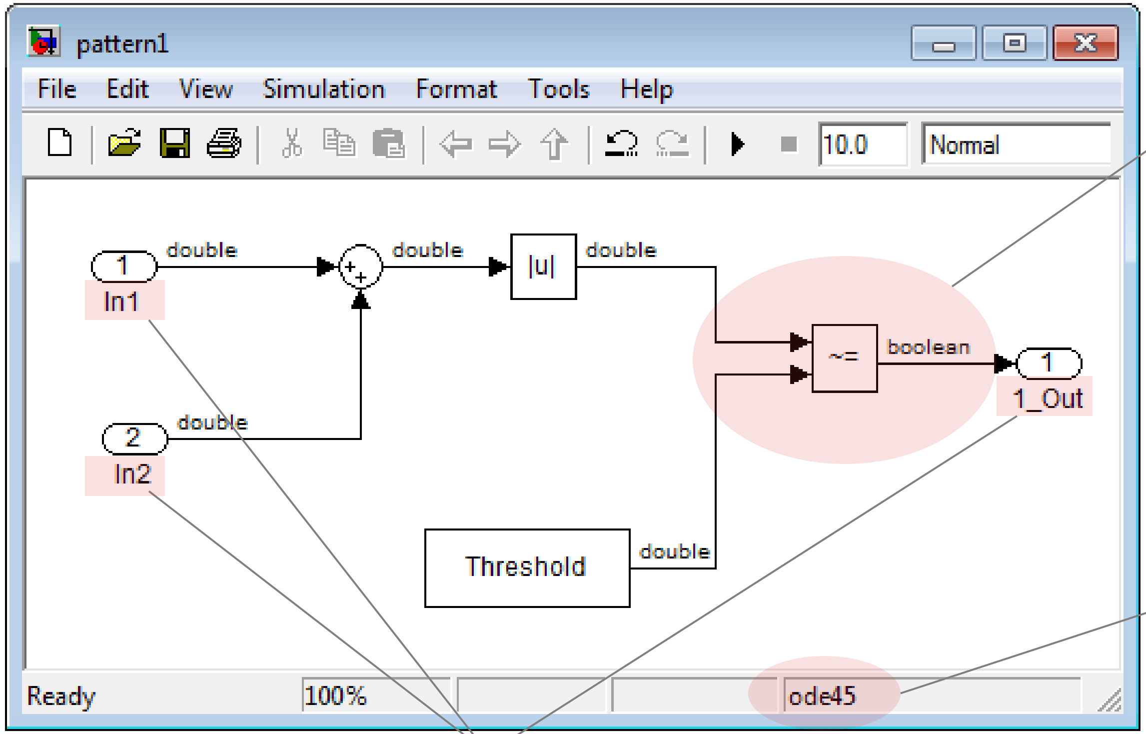
Example



Is there a potential error in this model? It depends…

Example

How about now?



When generating code:

- Floating-point precision issues may lead to incorrect comparison results

Is this a production model?

- Implementation requires a fixed-step, discrete solver

- Ports do not follow established naming conventions

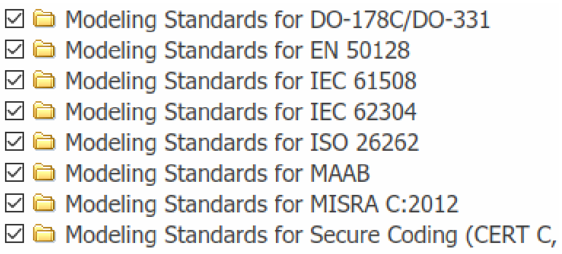
Simulink Check

R2017b

Automate verification and correct models to improve design

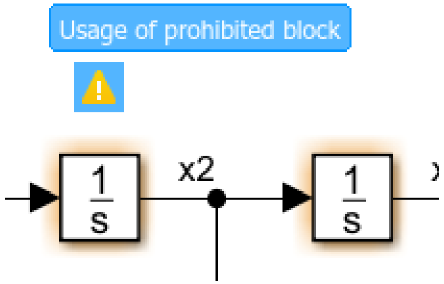
Standards & Guidelines Checks

- Automate compliance to standards
- Create custom checks



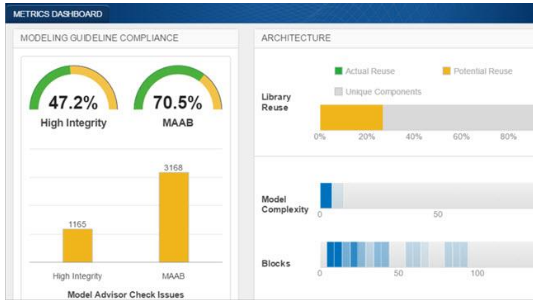
Edit Time Checking

- Find and fix compliance issues while you design



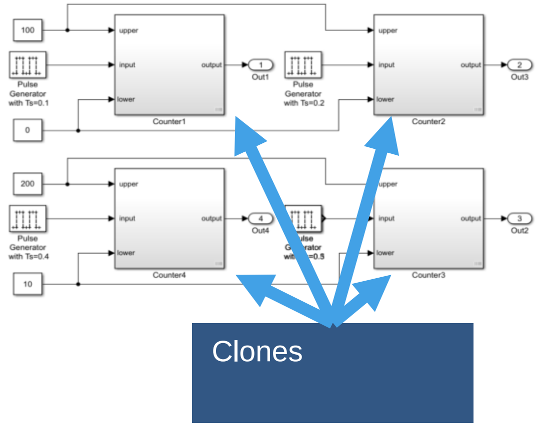
Model Metrics

- Analyze your model for complexity, size, reusability
- Assess design quality

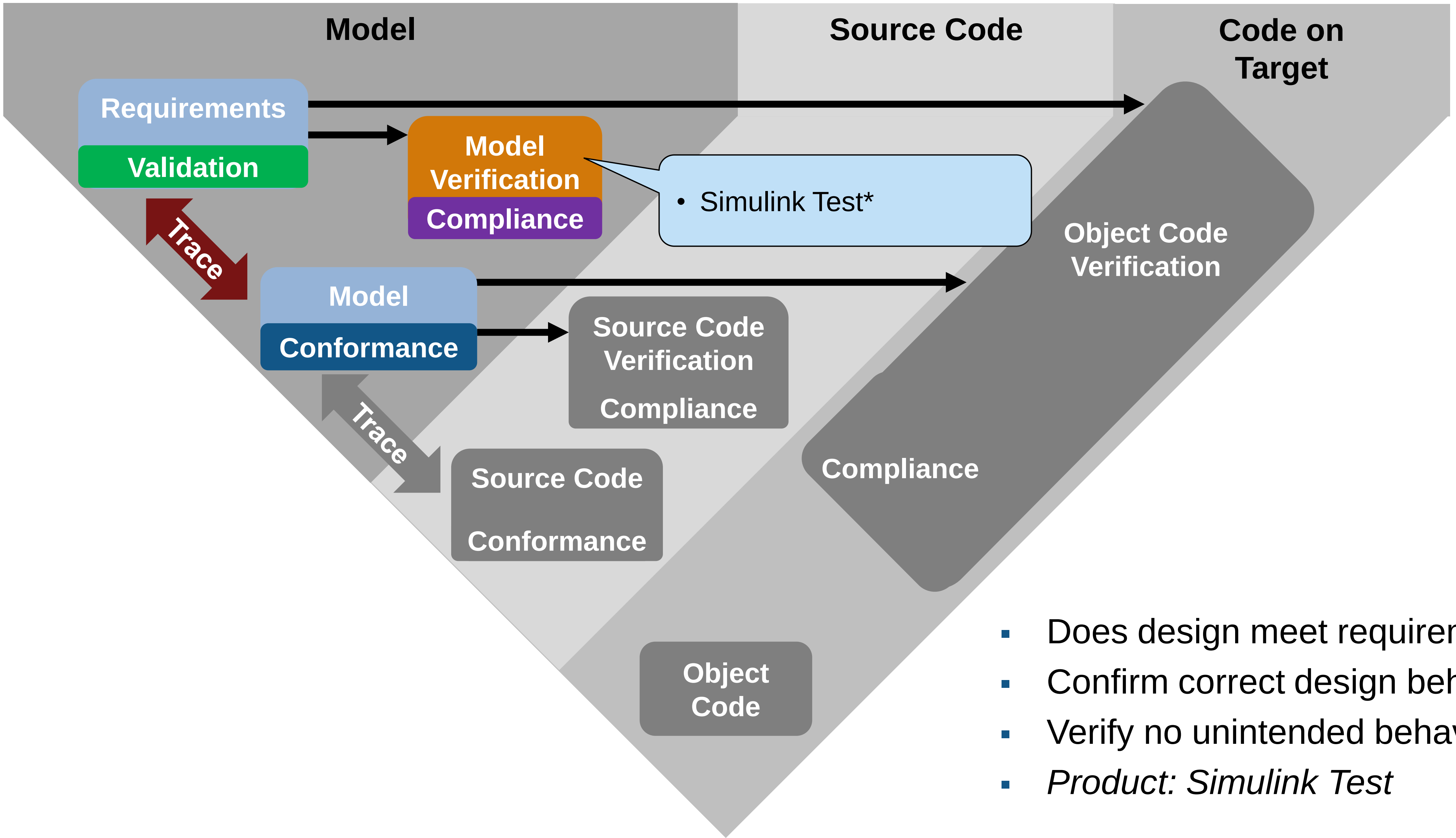


Model Refactoring

- Find clones and modeling patterns.
- Refactor to improve maintainability



Verification and Validation Tasks and Activities

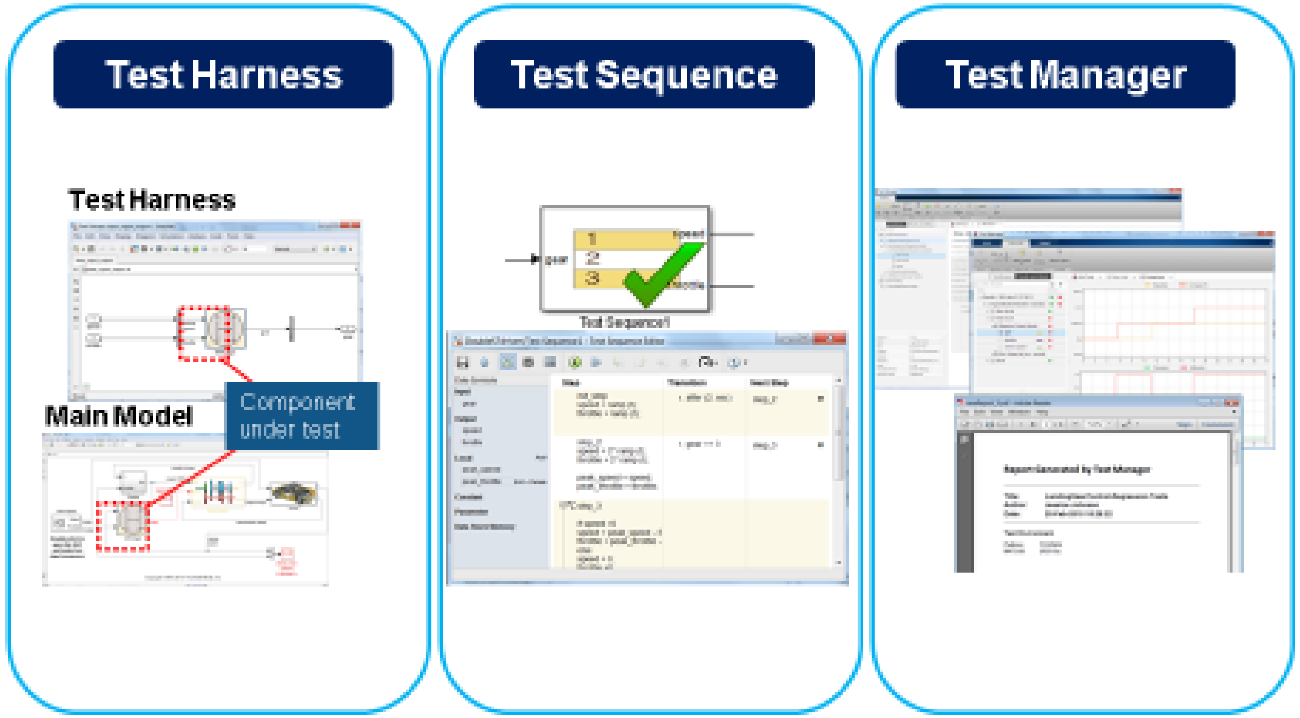


- Does design meet requirements
- Confirm correct design behavior
- Verify no unintended behavior
- *Product: Simulink Test*

Functional Testing Process

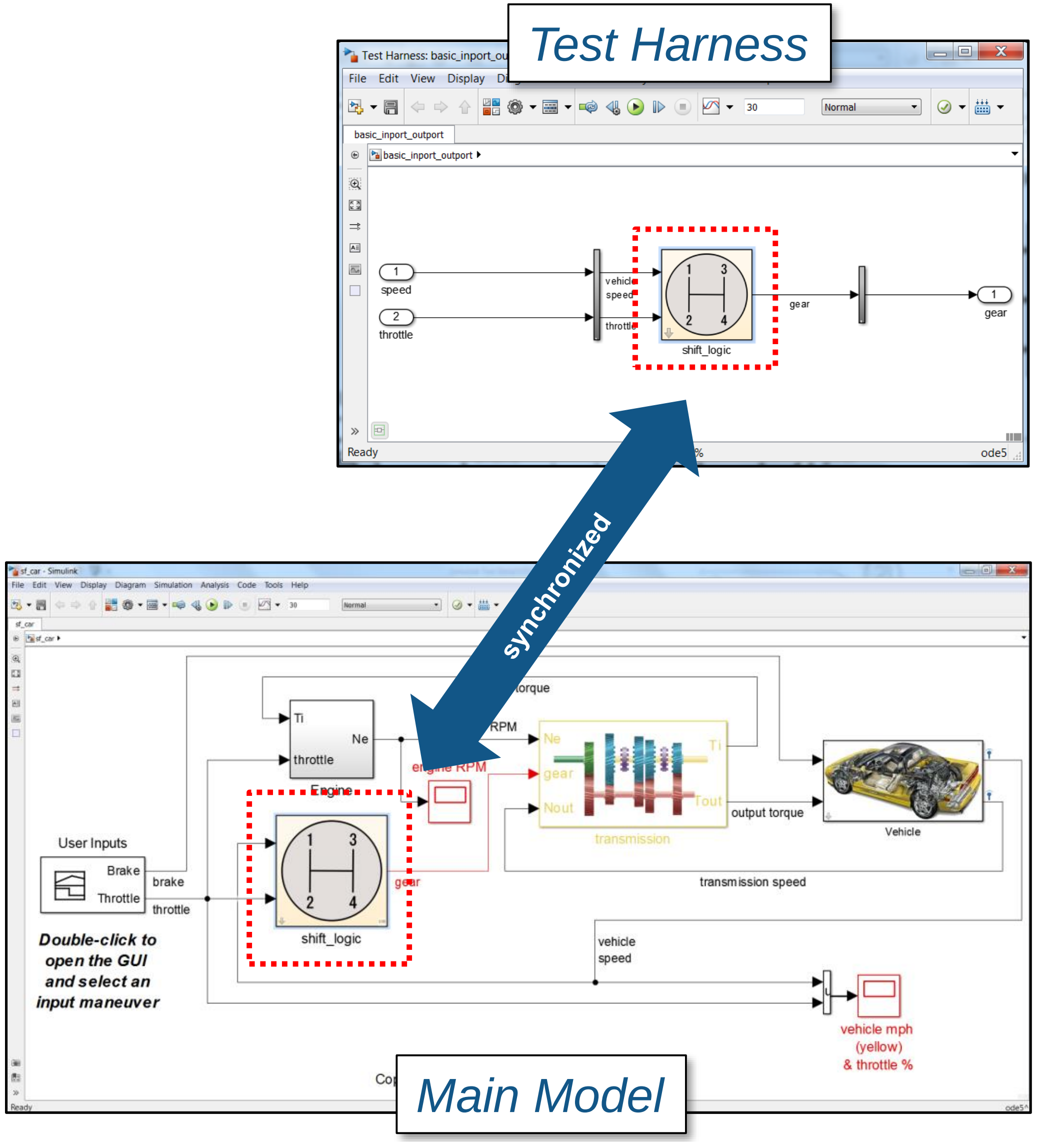
- Author test-cases that are derived from requirements
  - Use test harness to isolate component under test
  - Test Sequence to create complex test scenarios

- Manage tests, execution, results
  - Re-use tests for regression
  - Automate in Continuous Integration systems such as Jenkins



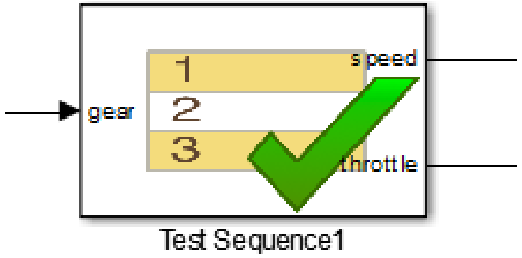
Test Harness

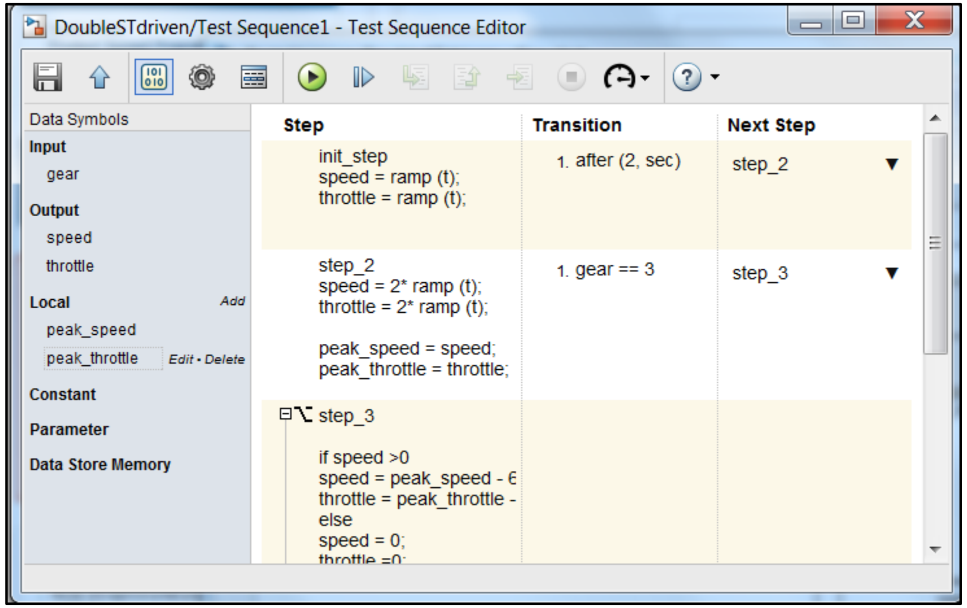
- ✓ Harnesses contained in the model file or external
- ✓ Build harness at unit (subsystem) or system level
- ✓ Synchronized test environment (harness ←→ model)
- ✓ Enables unit testing without requiring new model
- ✓ Configure harness input and output blocks
- ✓ Supports SIL, PIL, HIL



Test Sequence/Assessment Block

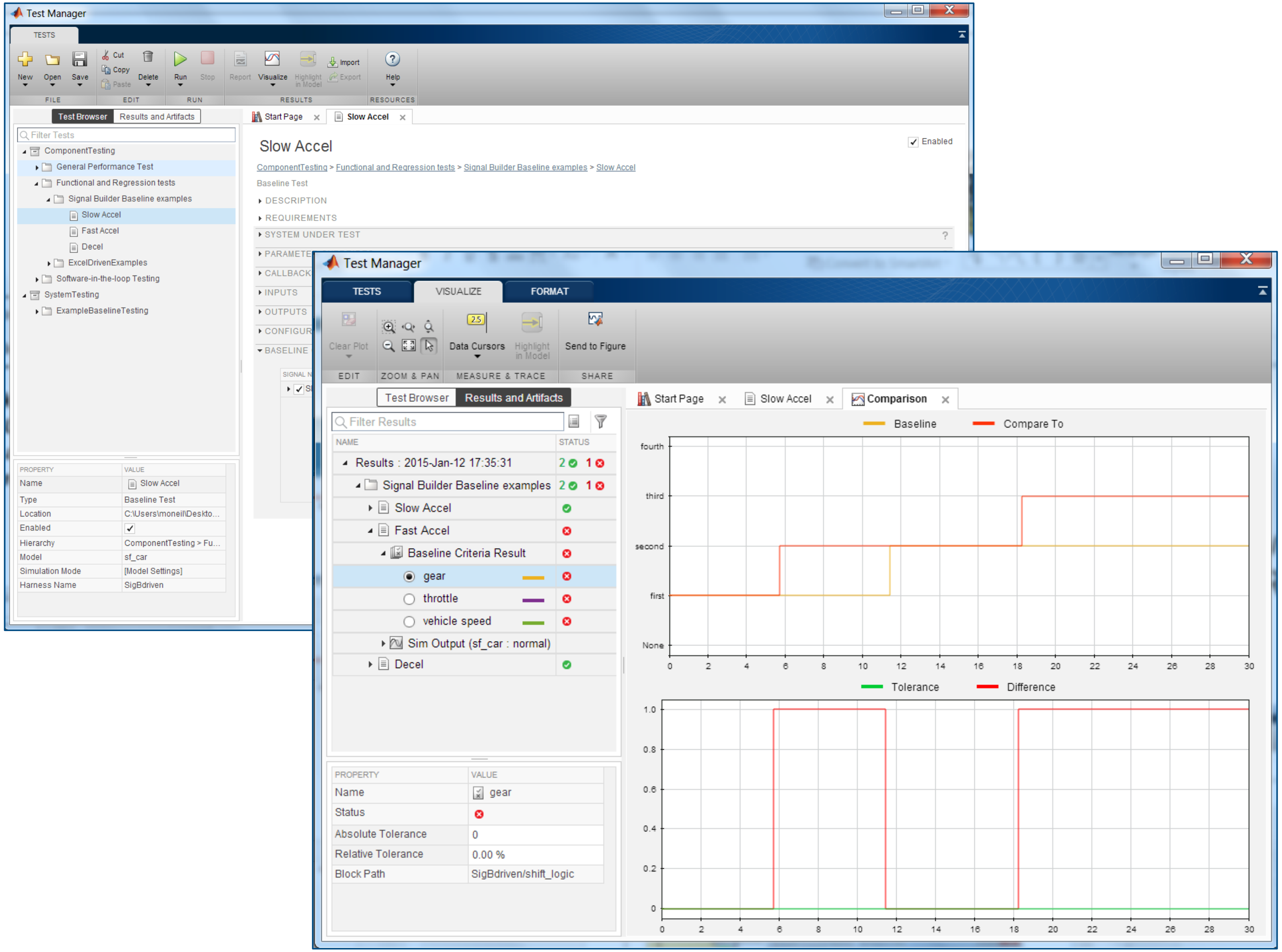
- ✓ Reactive and/or time based test cases
- ✓ Easier translation of test procedures
- ✓ Built on top of Stateflow with extensions for testing (SF license not required)
- ✓ Subset of MATLAB language
- ✓ Steps are temporal or logic-based
- ✓ Create complex test inputs and assessments
- ✓ Supports debugging (breakpoints)



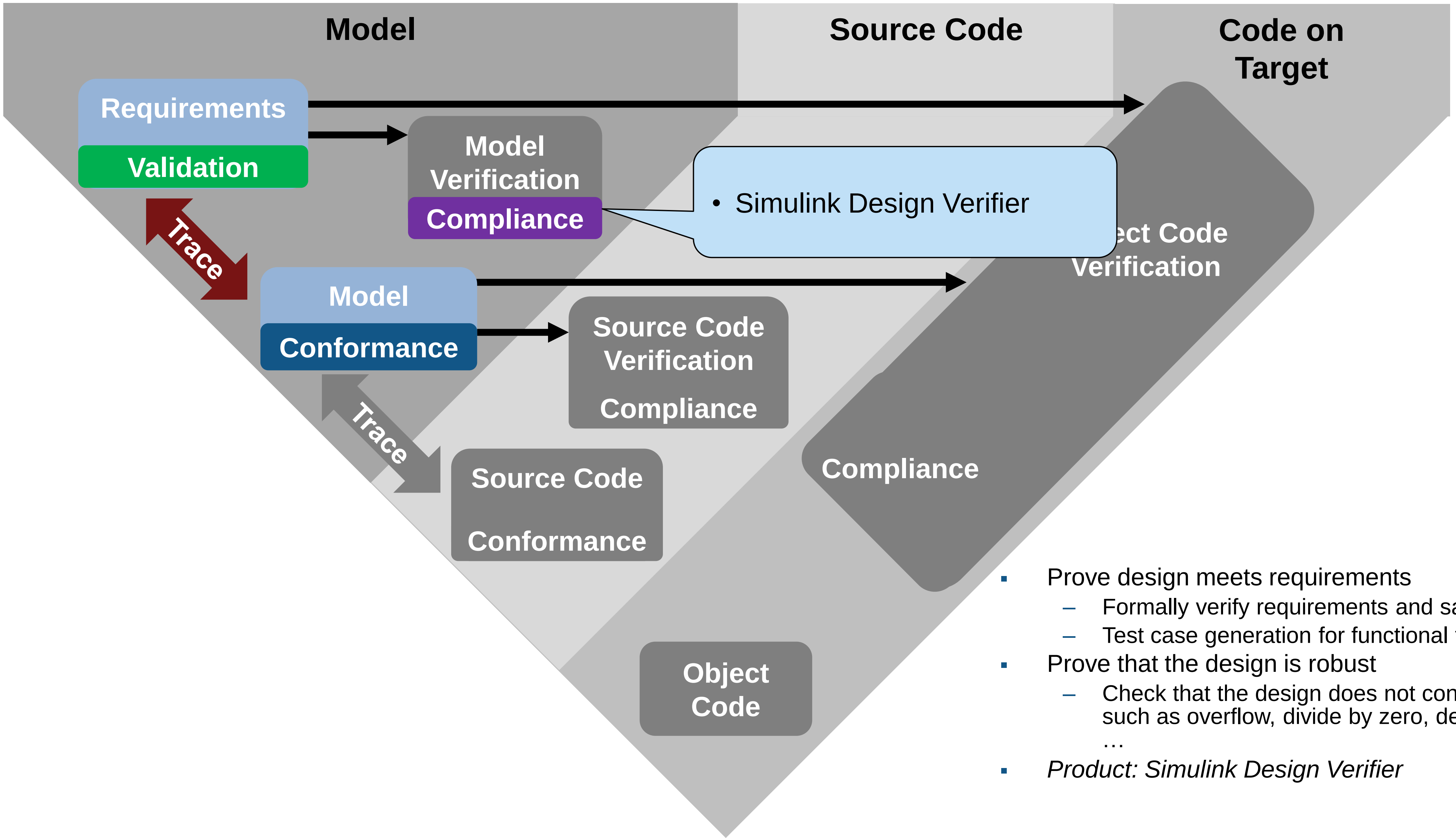


Test Manager

- ✓ Create test cases
- ✓ Group into suites and test files
- ✓ Execute individual or batch
- ✓ View result summary
- ✓ Analyze results
- ✓ Archive, export, report

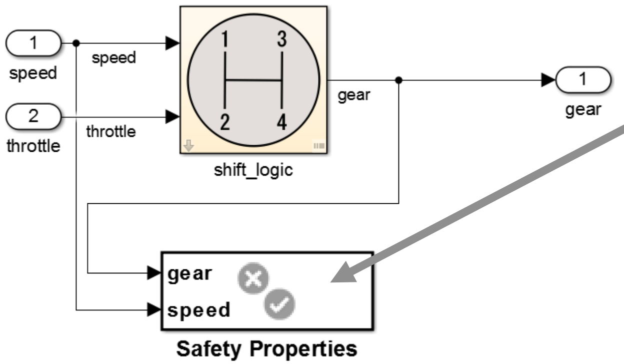


Verification and Validation Tasks and Activities



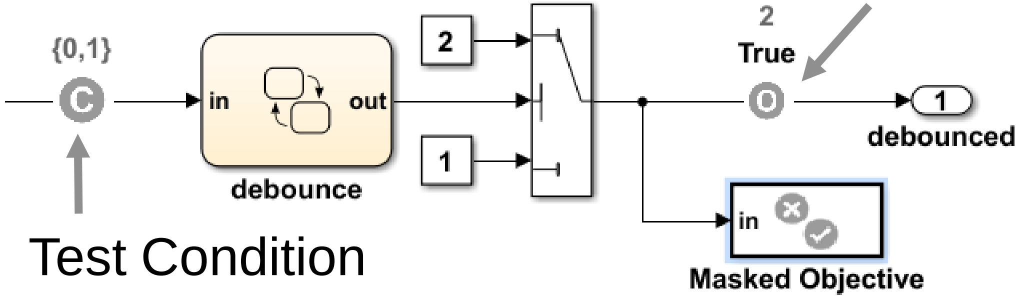
- Prove design meets requirements
  - Formally verify requirements and safety
  - Test case generation for functional testing
- Prove that the design is robust
  - Check that the design does not contain errors such as overflow, divide by zero, dead logic, …
- *Product: Simulink Design Verifier*

Formal Verification with Simulink Design Verifier



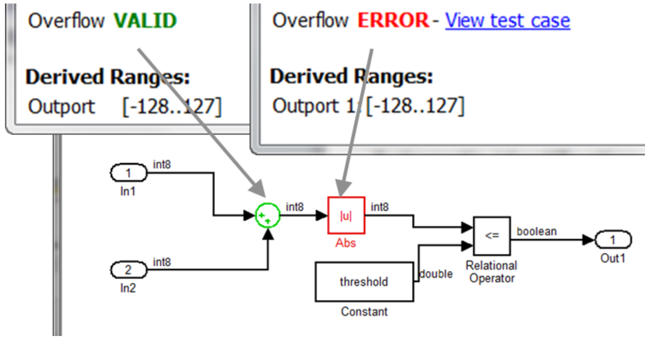
Checks that design meets requirements
- Gear 2 *always* engages when speed ≥ 5 and ≤ 25
- Gear 2 *never* engages when speed < 5 or > 25

Test Objective



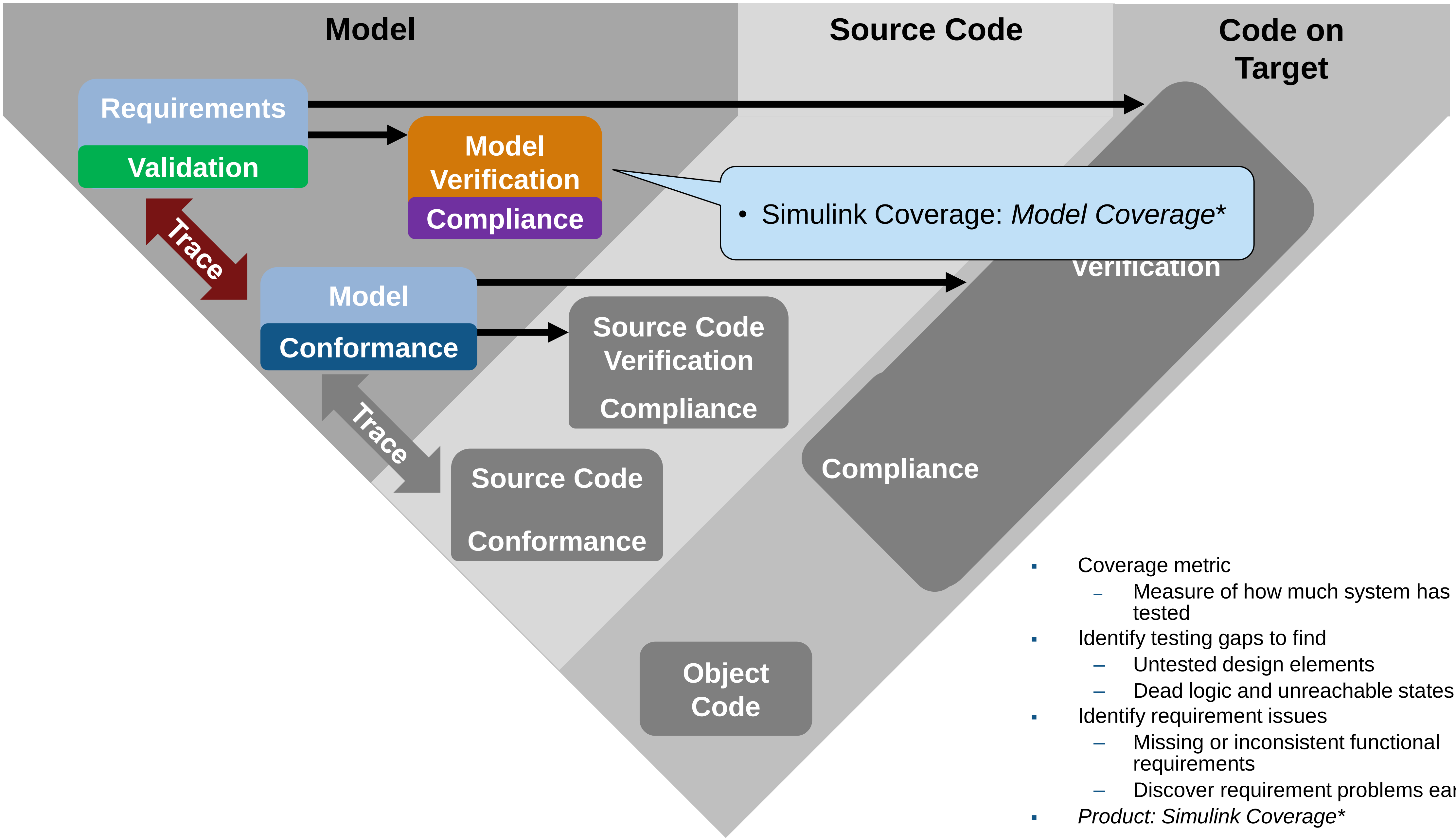
Test Condition

Automatically generate functional test case
- Custom objectives signals must satisfy in tests
- Constraints on signal values for test generator



Detect overflows, divide by zero, and other robustness errors
- Proven that overflow does NOT occur
- Proven that overflow DOES occur

Verification Task

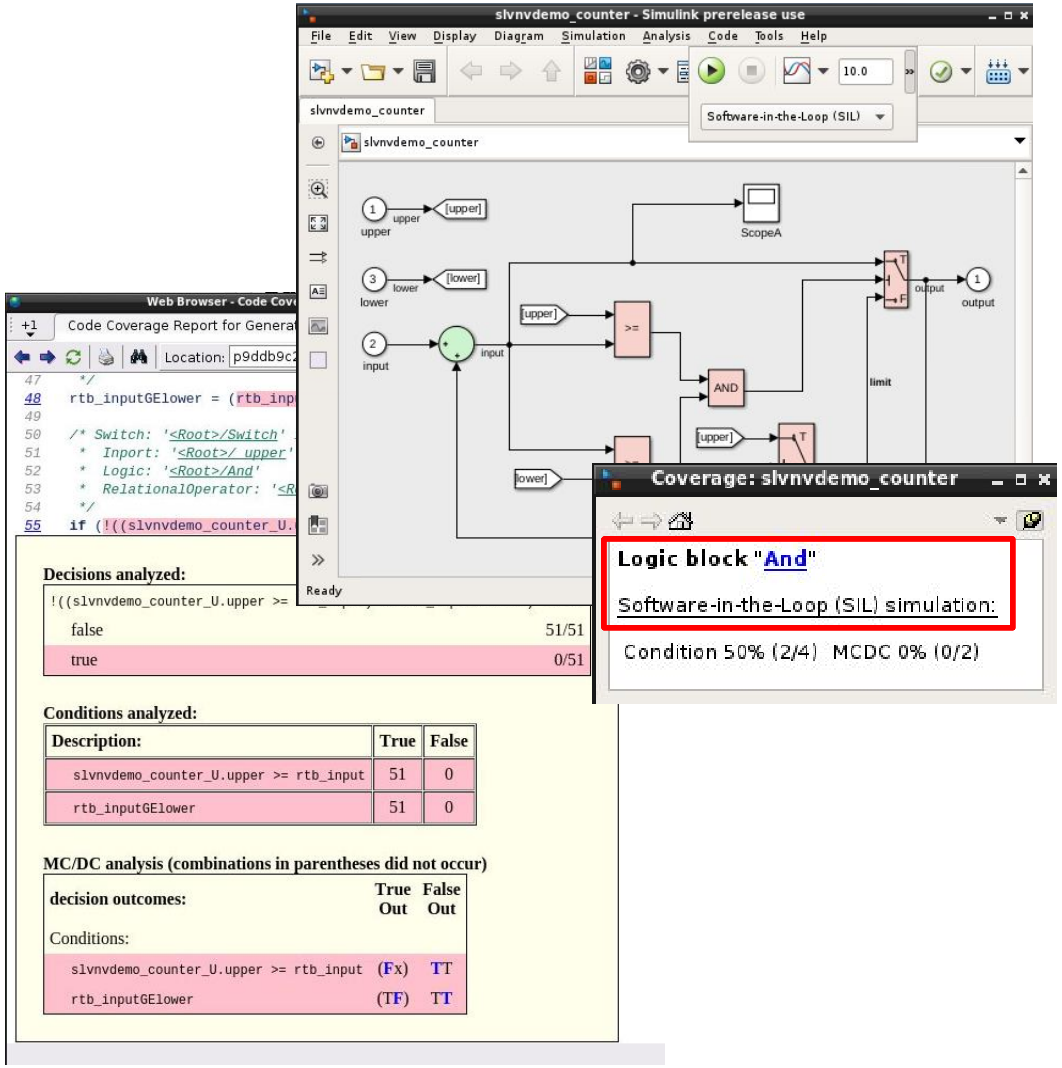


- Coverage metric
  - Measure of how much system has been tested
- Identify testing gaps to find
  - Untested design elements
  - Dead logic and unreachable states
- Identify requirement issues
  - Missing or inconsistent functional requirements
  - Discover requirement problems early
- *Product: Simulink Coverage**

Simulink Coverage

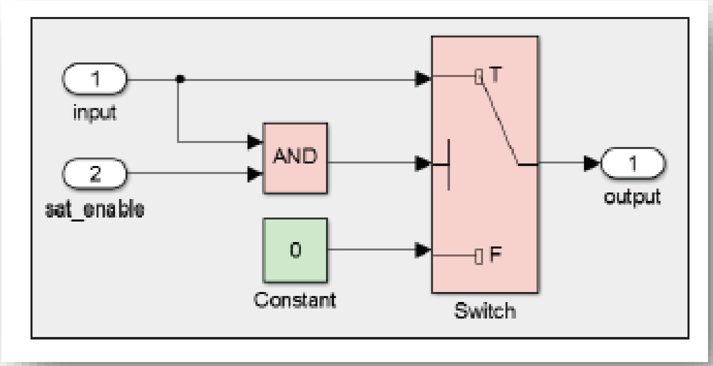
Measure test coverage in models and generated code

- Structural coverage analysis and reports from tests performed on Simulink® models (including C/C++ S-functions)
- Coverage metrics including decision, condition, MC/DC, relational boundary, and signal range
- Coverage analysis of C/C++ code generated by Embedded Coder®
- Coverage result highlighting in blocks, subsystems, and state charts
- Tool qualification support (with DO Qualification and IEC Certification Kits)

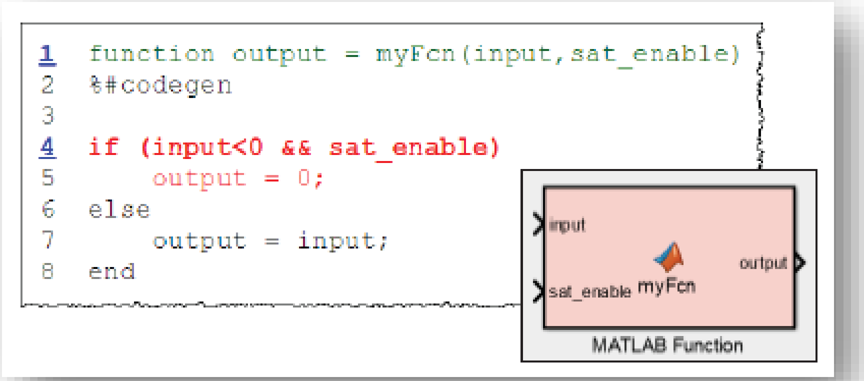


Model Elements That Receive Coverage

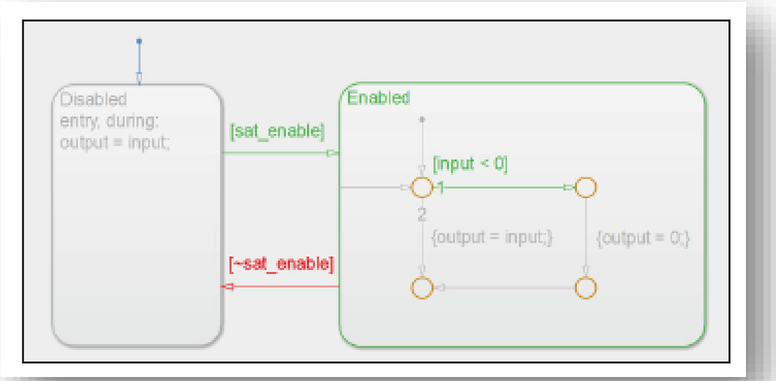
Simulink models



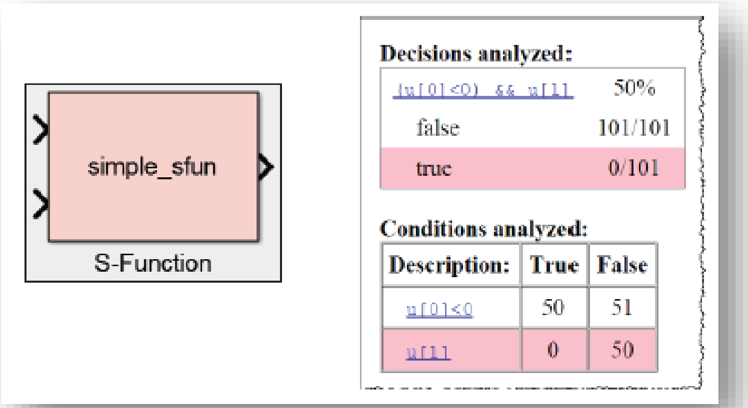
MATLAB function blocks



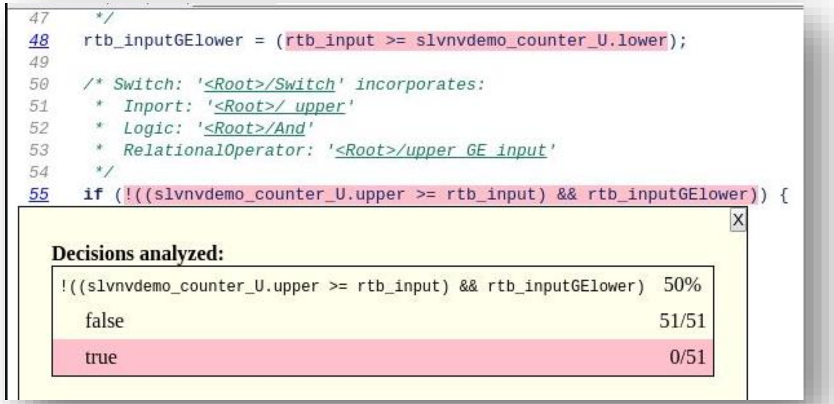
Stateflow charts



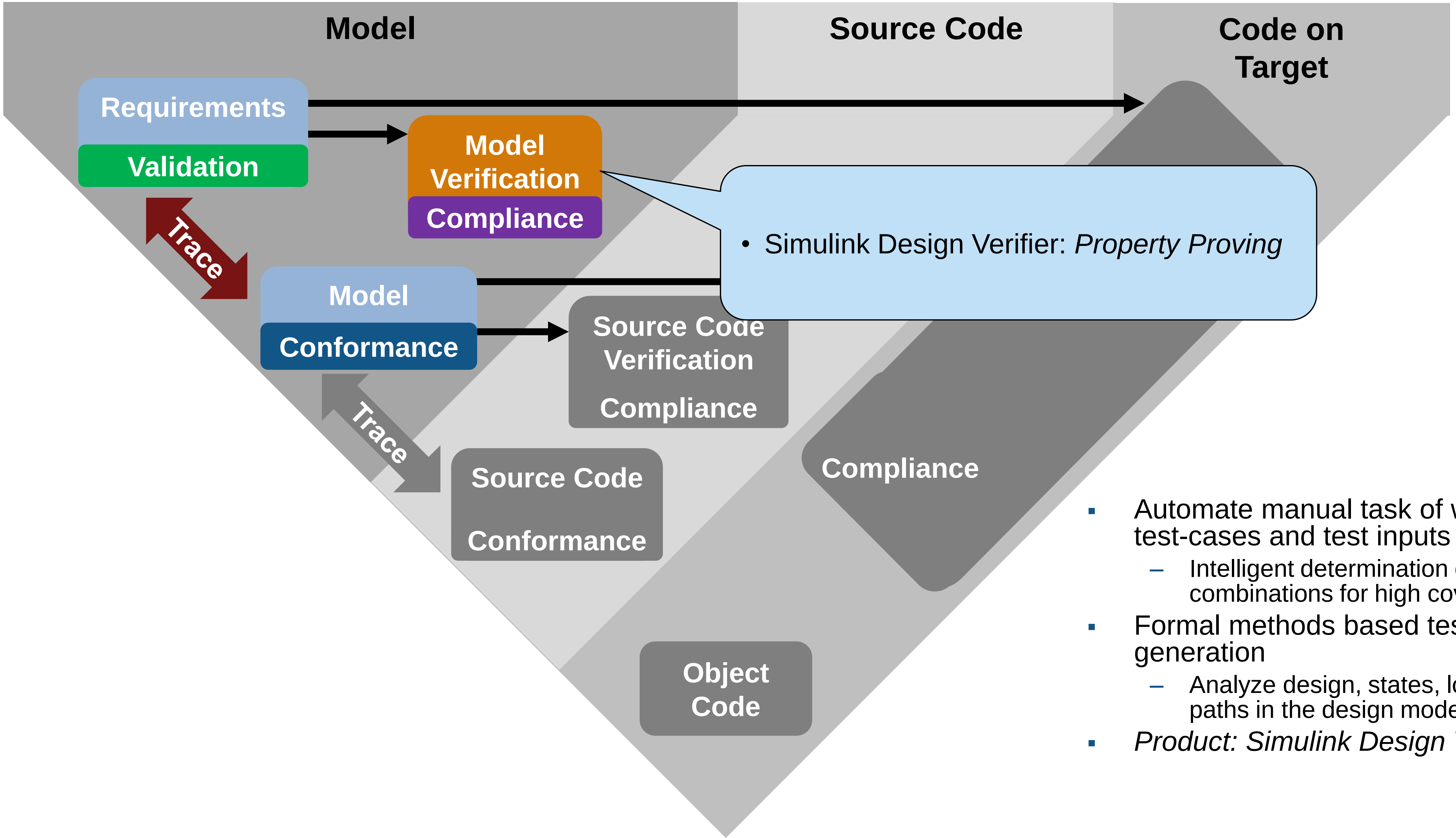
C/C++ code S-Functions



Generated code

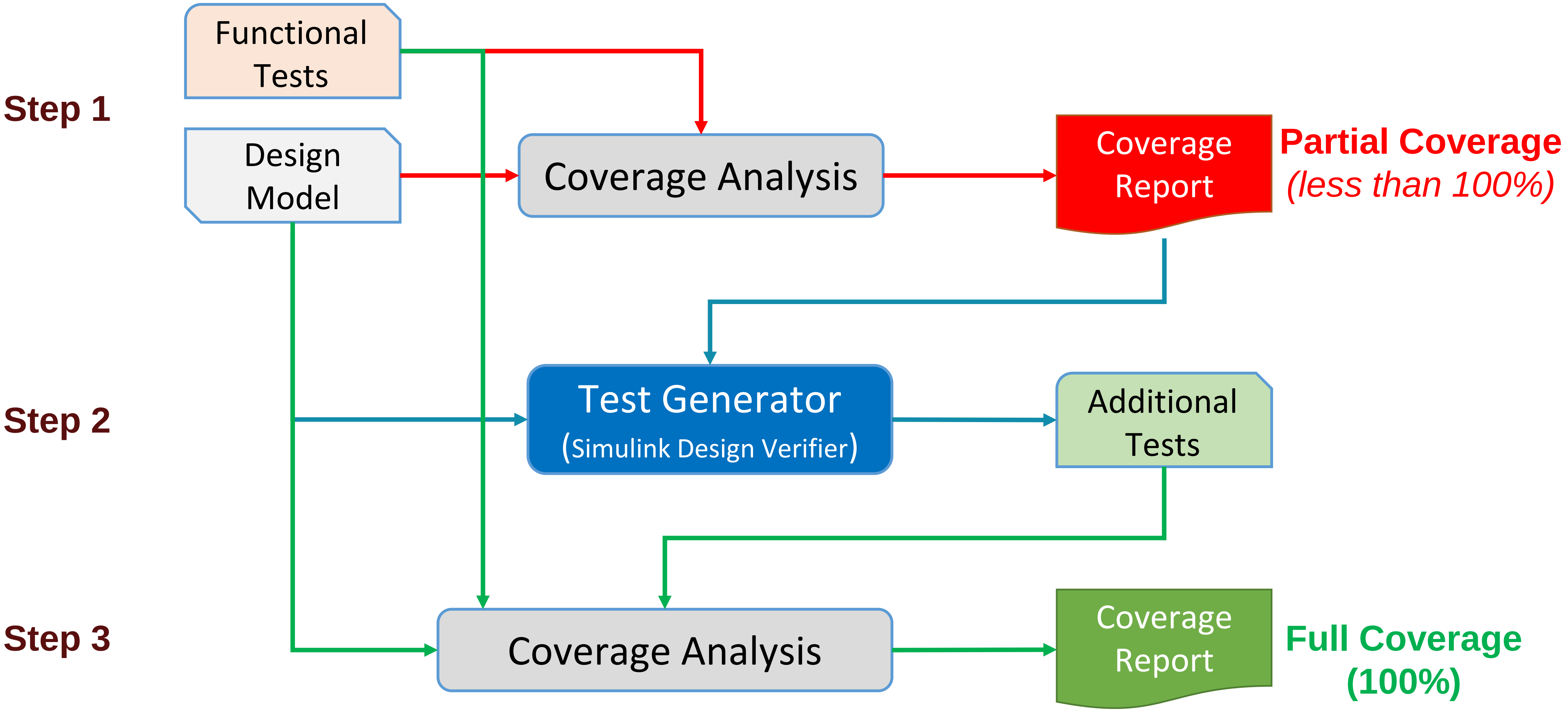


Verification and Validation Tasks and Activites

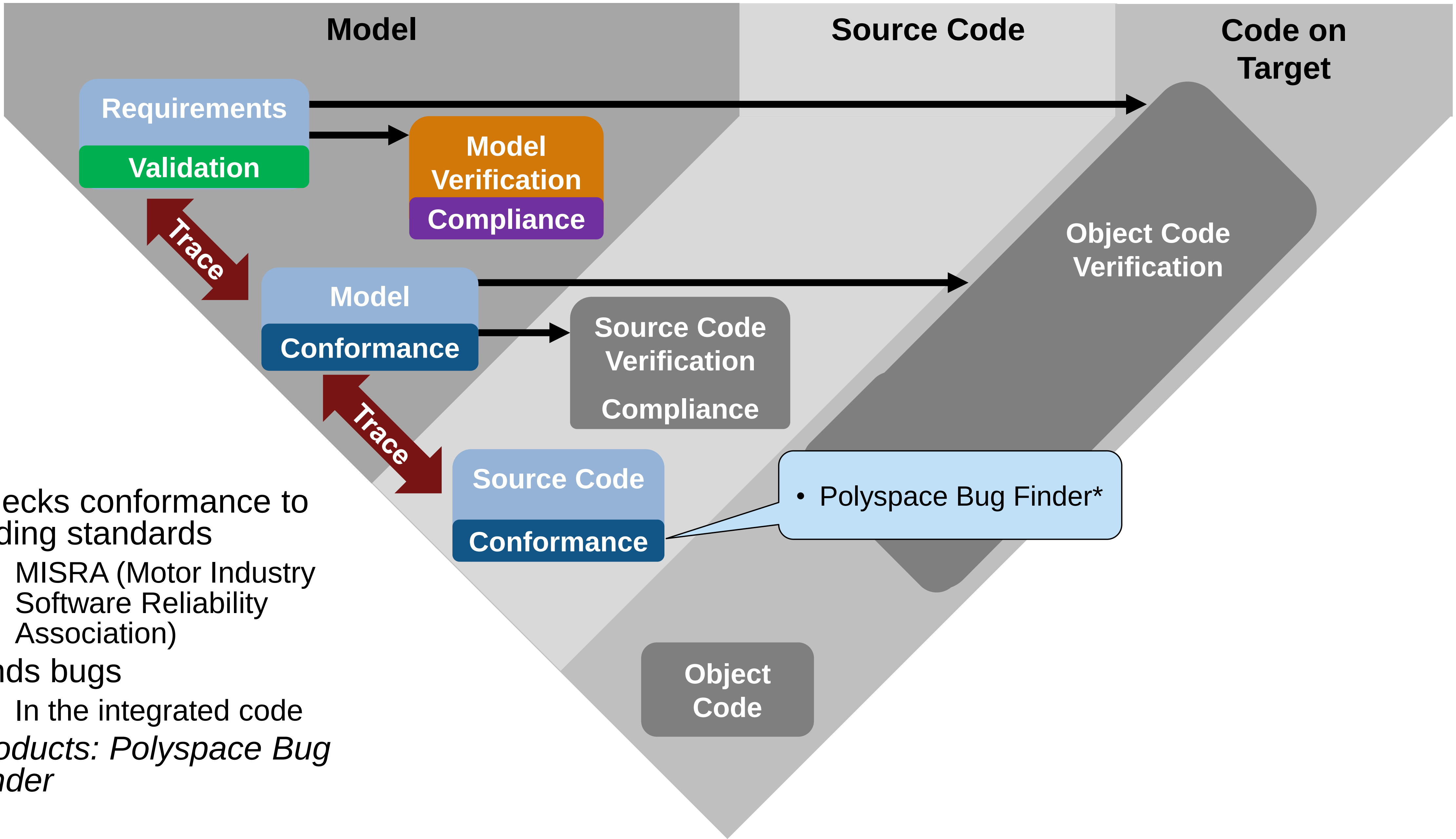


- Automate manual task of writing test-cases and test inputs
  - Intelligent determination of input combinations for high coverage
- Formal methods based test generation
  - Analyze design, states, logic paths in the design model
- *Product: Simulink Design Verifier*

Addressing Missing Coverage

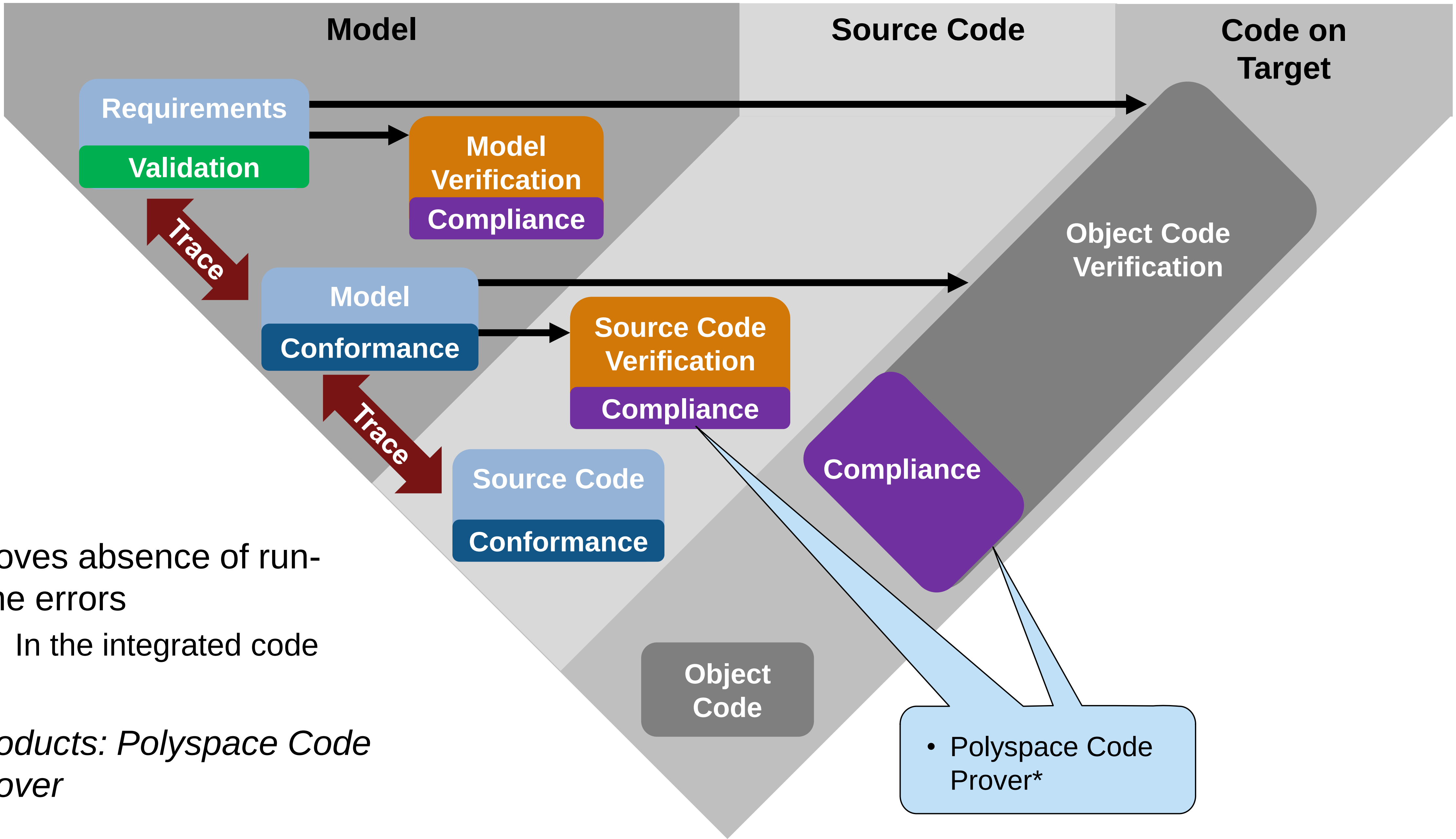


Verification and Validation Tasks and Activities



- Checks conformance to coding standards
  - MISRA (Motor Industry Software Reliability Association)
- Finds bugs
  - In the integrated code
- *Products: Polyspace Bug Finder*

Verification and Validation Tasks and Activities

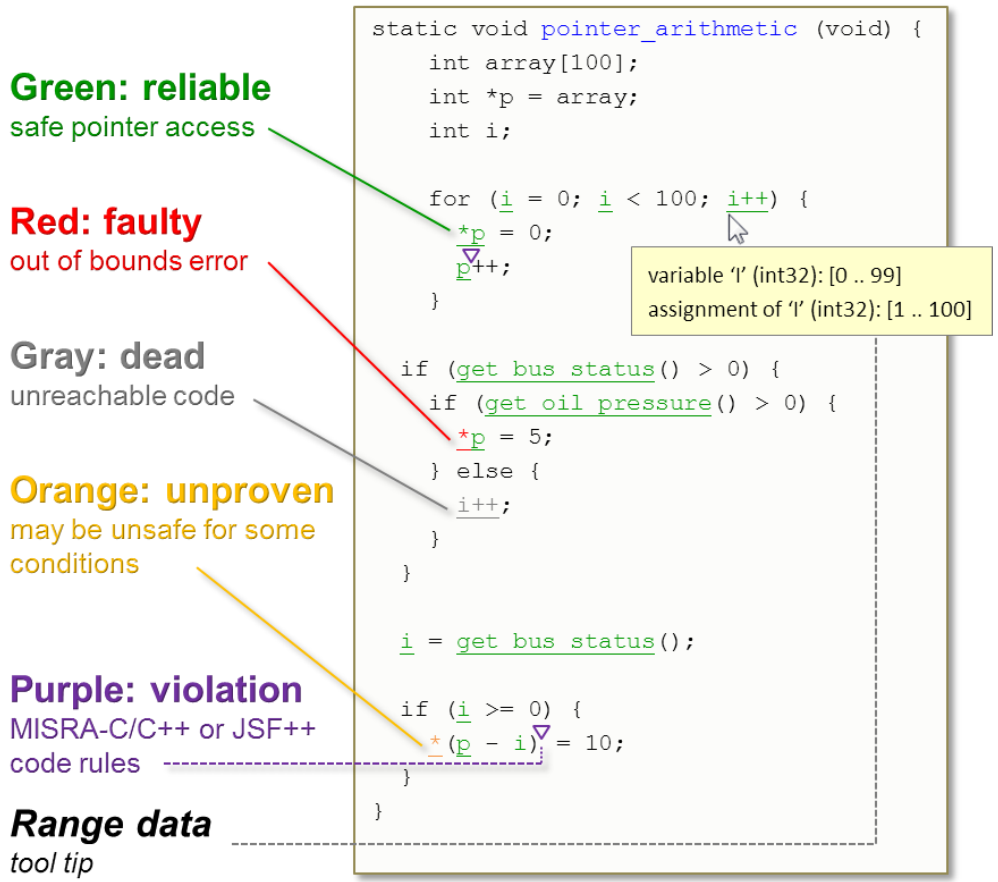


- Proves absence of run-time errors
  - In the integrated code

- *Products: Polyspace Code Prover*

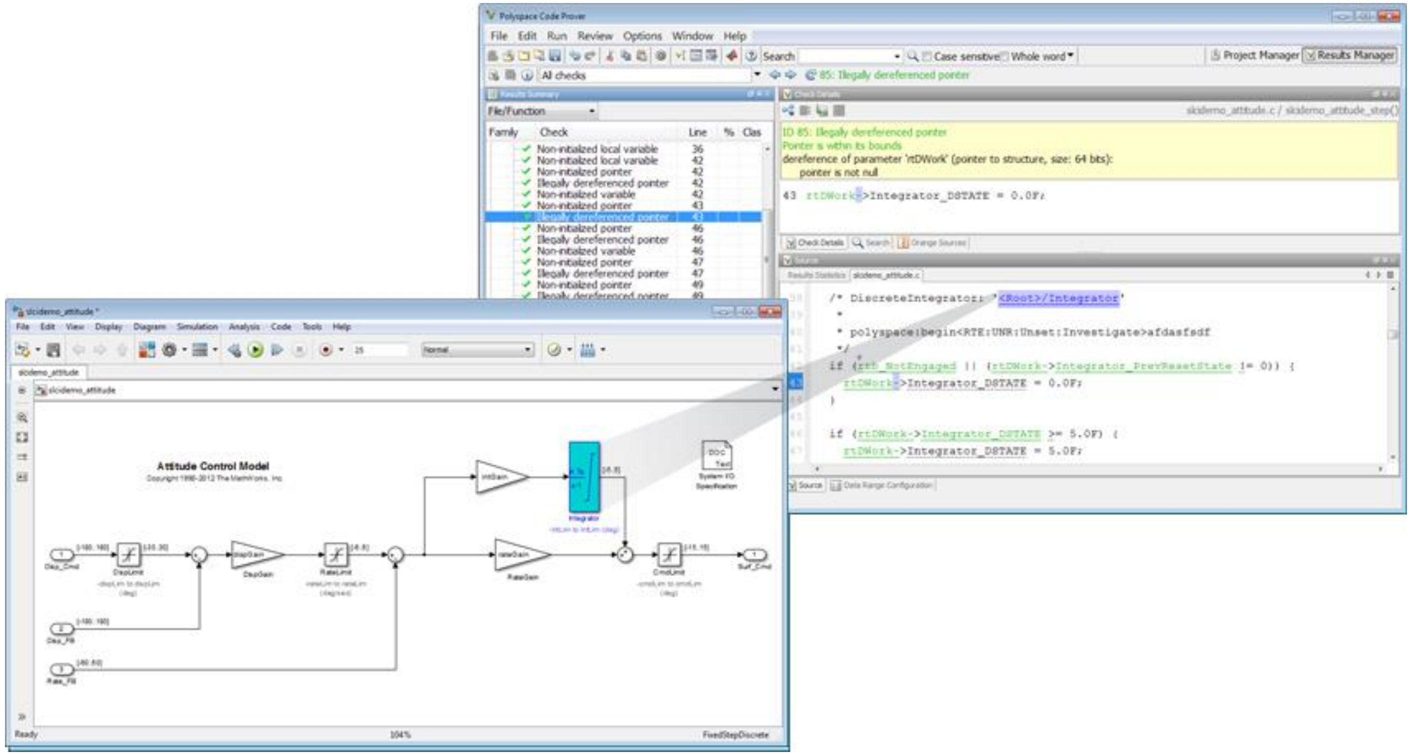
Static Code Analysis Techniques Supported by Polyspace

- Code metrics and standards
  - Comment density, cyclomatic complexity,…
  - MISRA and Cybersecurity standards
- Bug finding
  - Data and control flow of software
  - Check code for security vulnerabilities
- Code proving
  - Formal methods with abstract interpretation
  - No false negatives



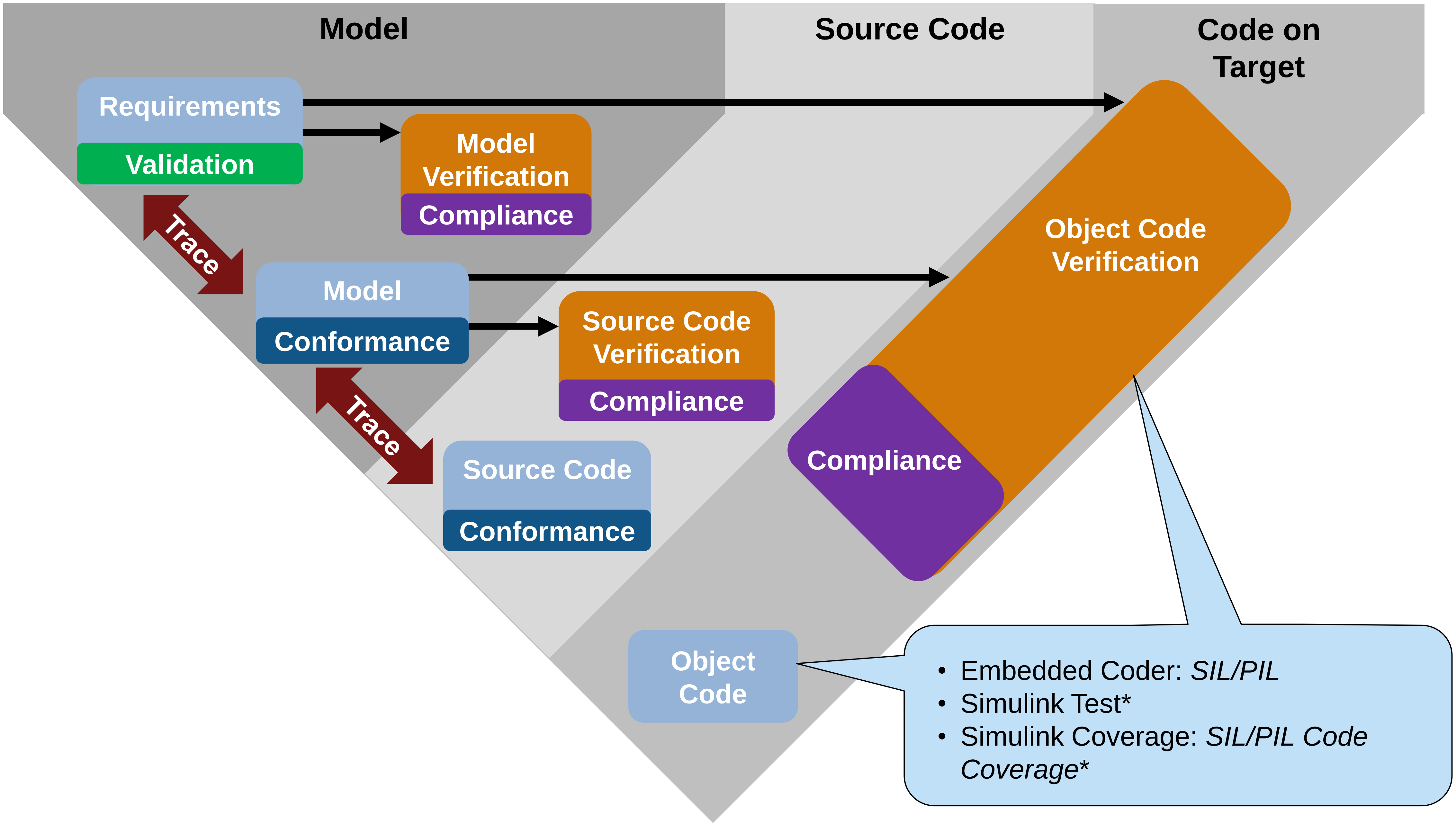
Results from Polyspace Code Prover

Traceability from Code to Model



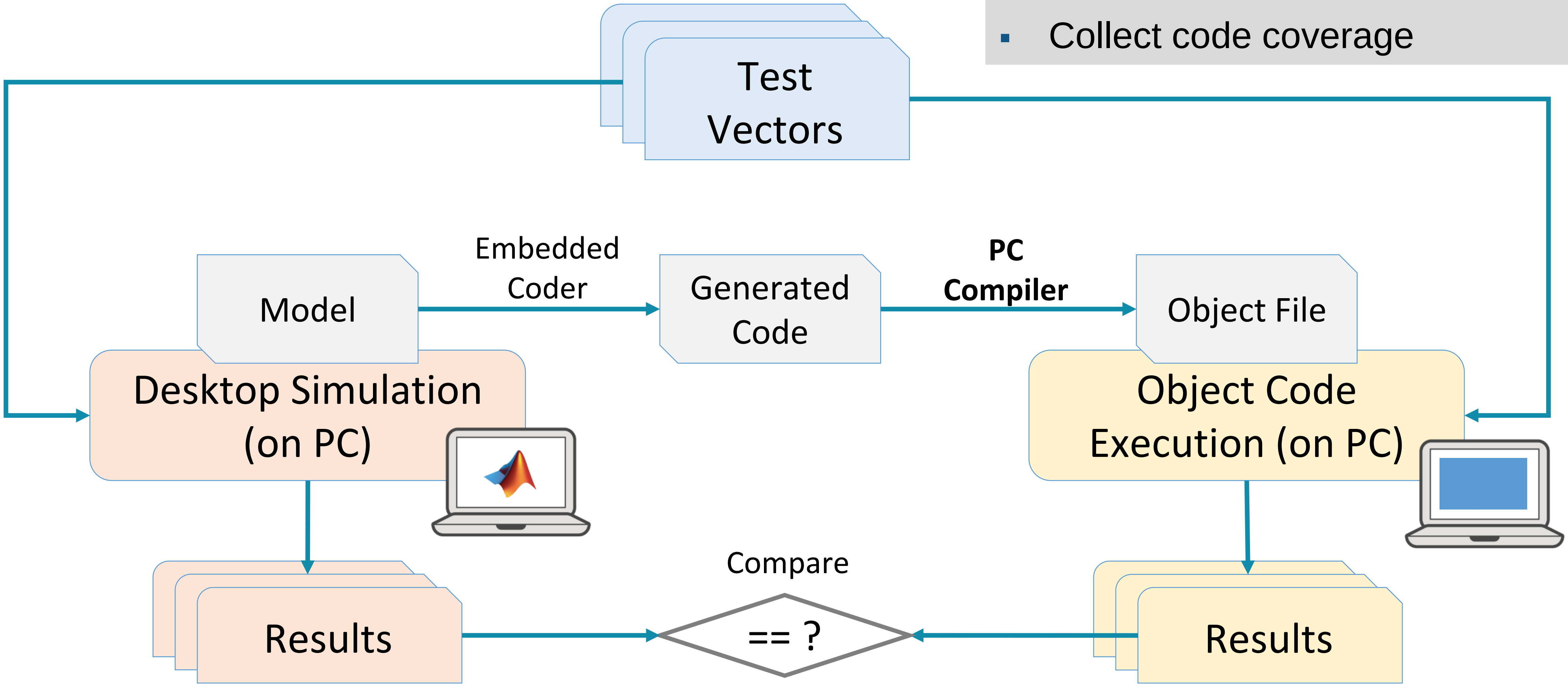
Polyspace Bug Finder and *Polyspace Code Prover* verification results, including MISRA analysis can be traced from code to model

Verification and Validation Tasks and Activities



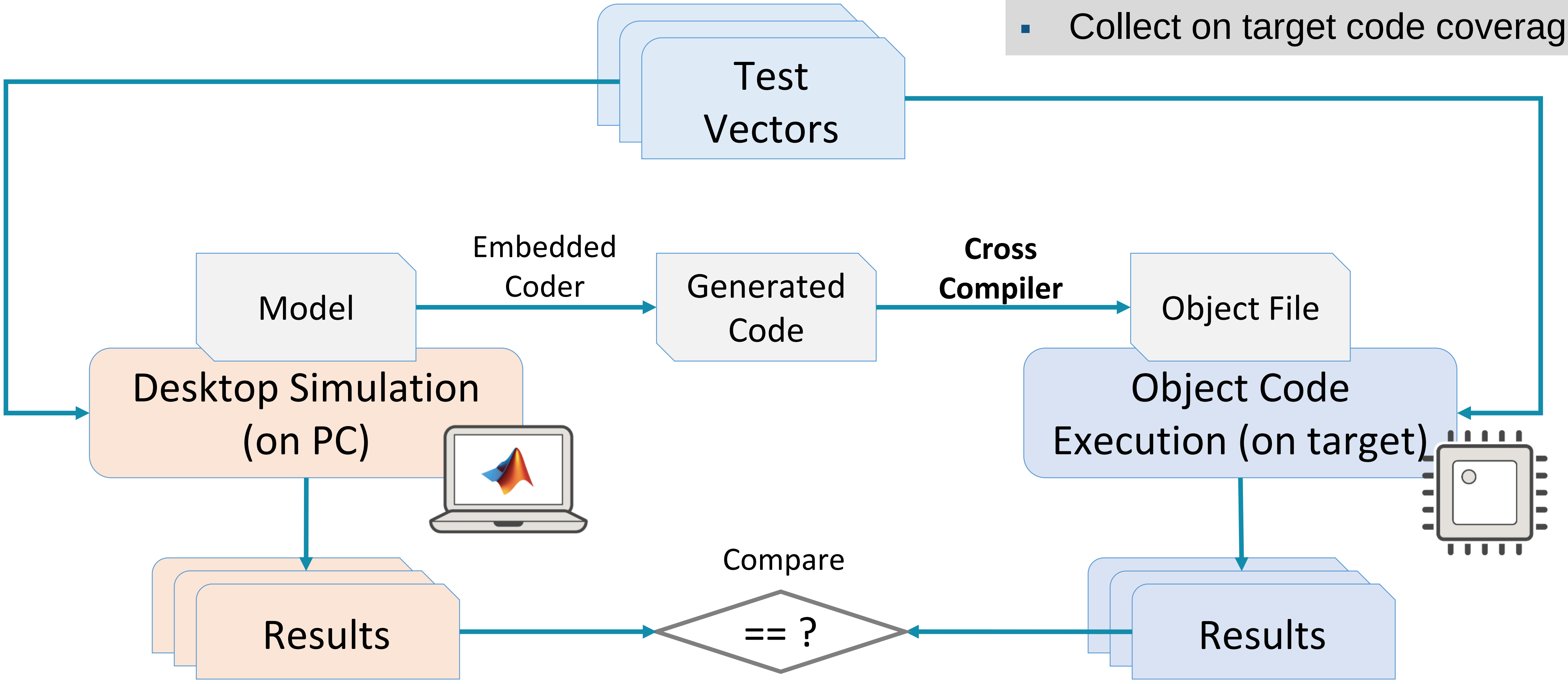
Software In the Loop (SIL) Testing

- Show equivalence, model to code
- Assess code execution time
- Collect code coverage



Processor In the Loop (PIL) Testing

- Verify numerical equivalence
- Assess target execution time
- Collect on target code coverage



MathWorks V&V Solution Summary

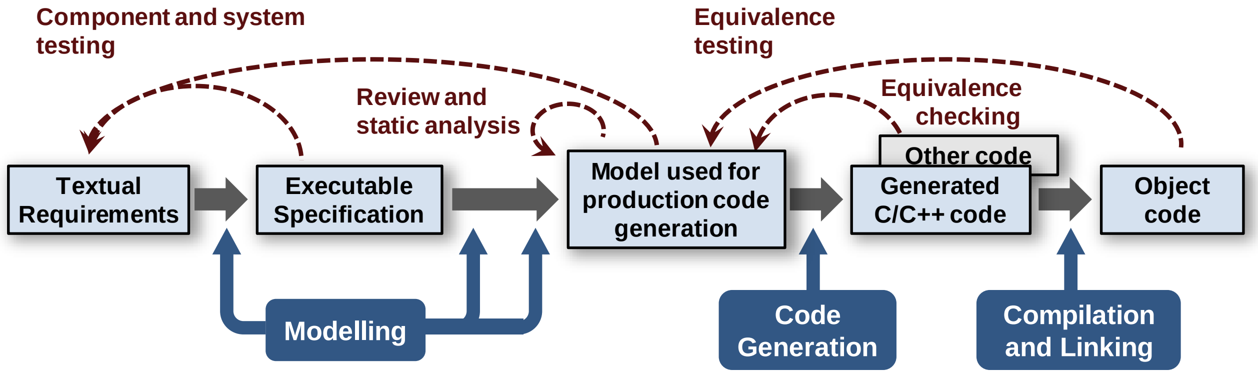
Requirements	Author, manage, and trace requirements
Standards Compliance	Verify compliance with standards and guidelines
Testing	Develop, manage, execute simulation-based tests
Formal Verification	Prove design meets requirements, prove robustness
Coverage Analysis	Measure model and generated code coverage
Static Code Analysis	Check bugs, MISRA compliance, prove code
SIL, PIL	Perform back-to-back testing

MathWorks V&V Product Capabilities

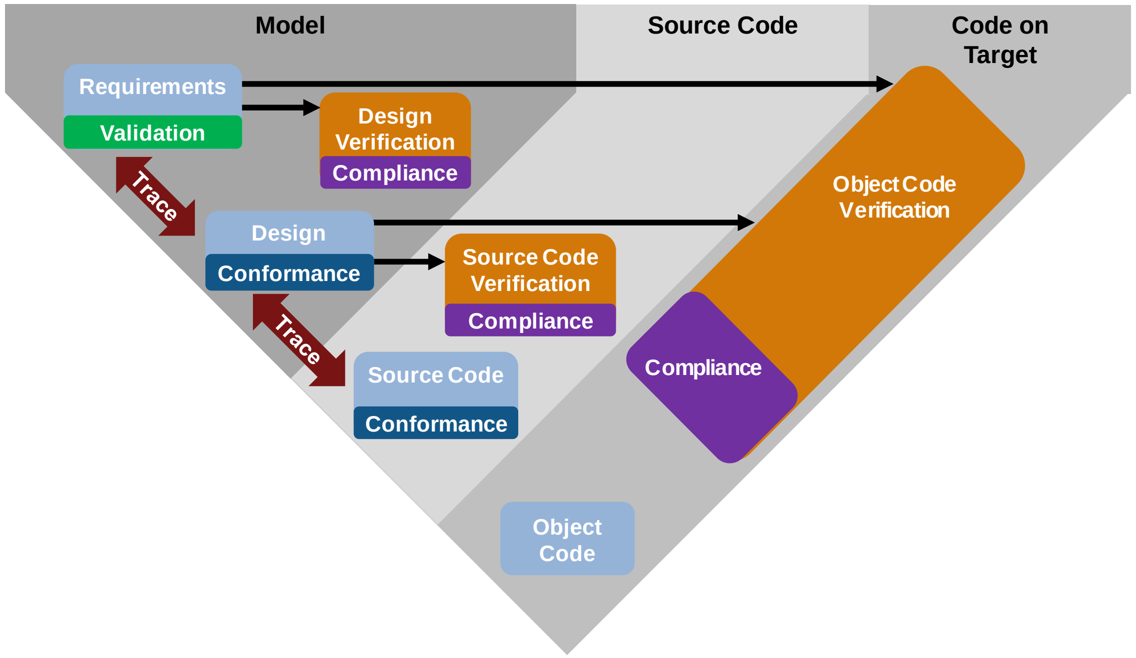
Requirements	Simulink Requirements* *(New in R2017b)*
Standards Compliance	Simulink Check* *(New in R2017b)*
Testing	Simulink Test
Formal Verification	Simulink Design Verifier
Coverage Analysis	Simulink Coverage* *(New in R2017b)*
Static Code Analysis	Polyspace Bug Finder, Polyspace Code Prover
SIL, PIL	Simulink Test

Workflows for Certification Standards

IEC 61508
ISO 26262
IEC 62405
EN 50128



DO-178C
DO-331
DO-333



Bell Helicopter Develops World's First Commercial Fly-by-Wire Helicopter

Challenge

Develop flight software for the first commercial fly-by-wire helicopter and certify it to DO-178B Level A

Solution

Use Model-Based Design to model and simulate the control laws, trace requirements to the model, and generate and verify 16,000 lines of code

Results

- Integration time cut by 90%
- Development iterations reduced from weeks to hours
- Confidence in code quality maintained
- Simulink Code Inspector Qualified by FAA for DO-178B Level A

The Bell 525 Ships 1 and 2 over the Palo Duro Canyon.

"With Model-Based Design we had a successful first flight; there were no issues from a control or integration standpoint. Generating the control law code from our Simulink model with Embedded Coder eliminated the slowdowns caused by manual code generation and freed the team to work on meeting the broader program goals."

- Mike Bothwell, Bell Helicopter

[Link to user story]

BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design

Challenge

Develop flight-critical software for a midsized business jet in compliance with DO-178B Level A standards

Solution

Use Model-Based Design to model the software and systems, run simulations with customer-provided test vectors, trace requirements to model elements, and generate 200,000 lines of certified code

Results

- Development efficiency doubled
- Certification schedule maintained
- Communication between teams facilitated

Primary flight control computers from BAE Systems.

"When we generated code from our Simulink models with Embedded Coder, the team we handed it off to knew it was gold—that it was debugged and fully met the requirements—because we had run it through the Simulink test vectors supplied by our customer. That was a huge advantage on this program."

- Maria Radecki, BAE Systems

Link to user story

ESA and Airbus Create Upper-Stage Attitude Control Development Framework Using Model-Based Design

Challenge

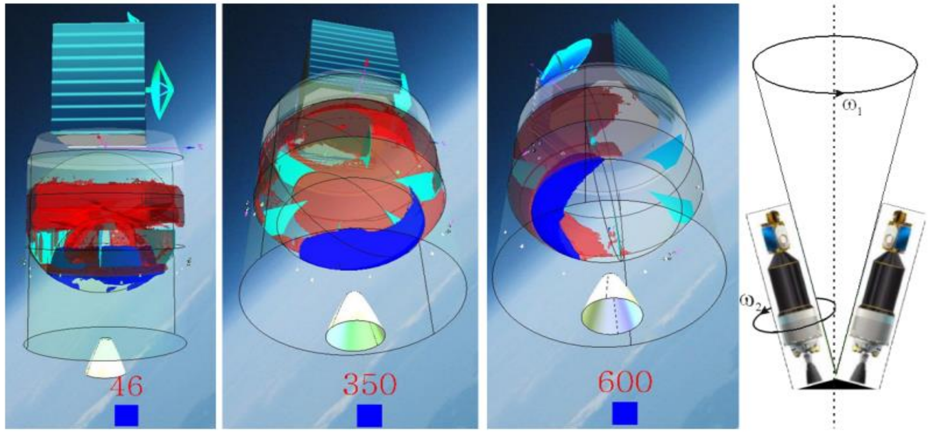
Speed the development of software for controlling complex launcher upper stage missions including the attitude of satellite payloads after they separate from ESA launch vehicles

Solution

Use Model-Based Design to develop controller models and multidomain physical models, run closed-loop simulations, and generate code for PIL testing

Results

- Design iterations reduced from one week to one day
- Failure modes modeled and eliminated
- Comprehensive design framework established



Propellant motion in spinning upper stages at 46, 350, and 600 seconds. Distribution after 350 seconds becomes uneven

"Model-Based Design multiplies the range of capabilities that I have as an engineer. As an individual control engineer I can do what previously took a handful of engineers, because I can create and simulate my own multidomain models. I don't have a wall around me anymore; I am able to better communicate and contribute across disciplines."

- Samir Bennani, ESA

[Link to user story]

Contact Us

508-647-7000

Monday - Friday

Customer Support ...... 08:30-17:30 ET

Technical Support ...... 08:30-20:00 ET