Verification and Validation of Models and Code

Presenter: Goran Begic
Technical Marketing
Goran.Begic@mathworks.com

MathWorks Symposium

Adopting Model-Based Design within Aerospace and Defense

Agenda

- Introductions
- Workflows for verification and validation

Introductions

- I spend most of my time:

A. Creating specifications and requirements (systems and software)

B. Implementation based on specification and requirements created by somebody else (generating / writing / deploying / debugging code)

C. Other (including both, or none of the above)

Great Demo

- How much time do we need to get 100% MC/DC coverage?

Summary | Details | Help

Coverage Report for ThrustReverserDeployLogic_harness

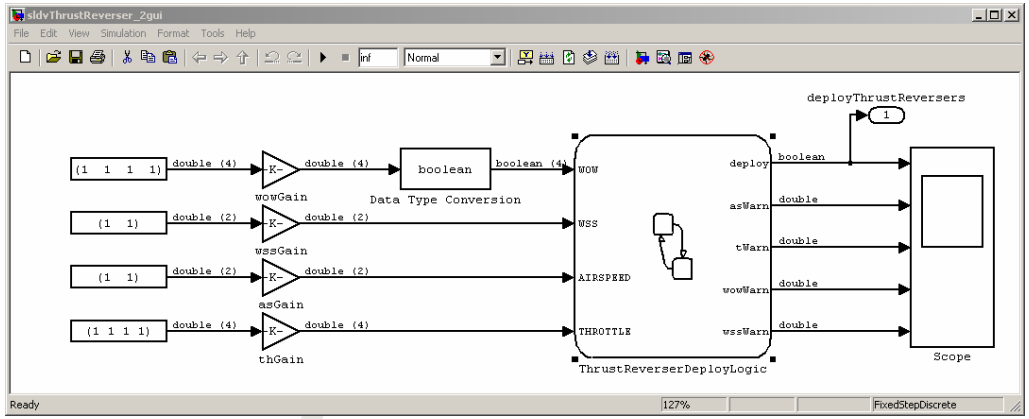
Tests

Test 1

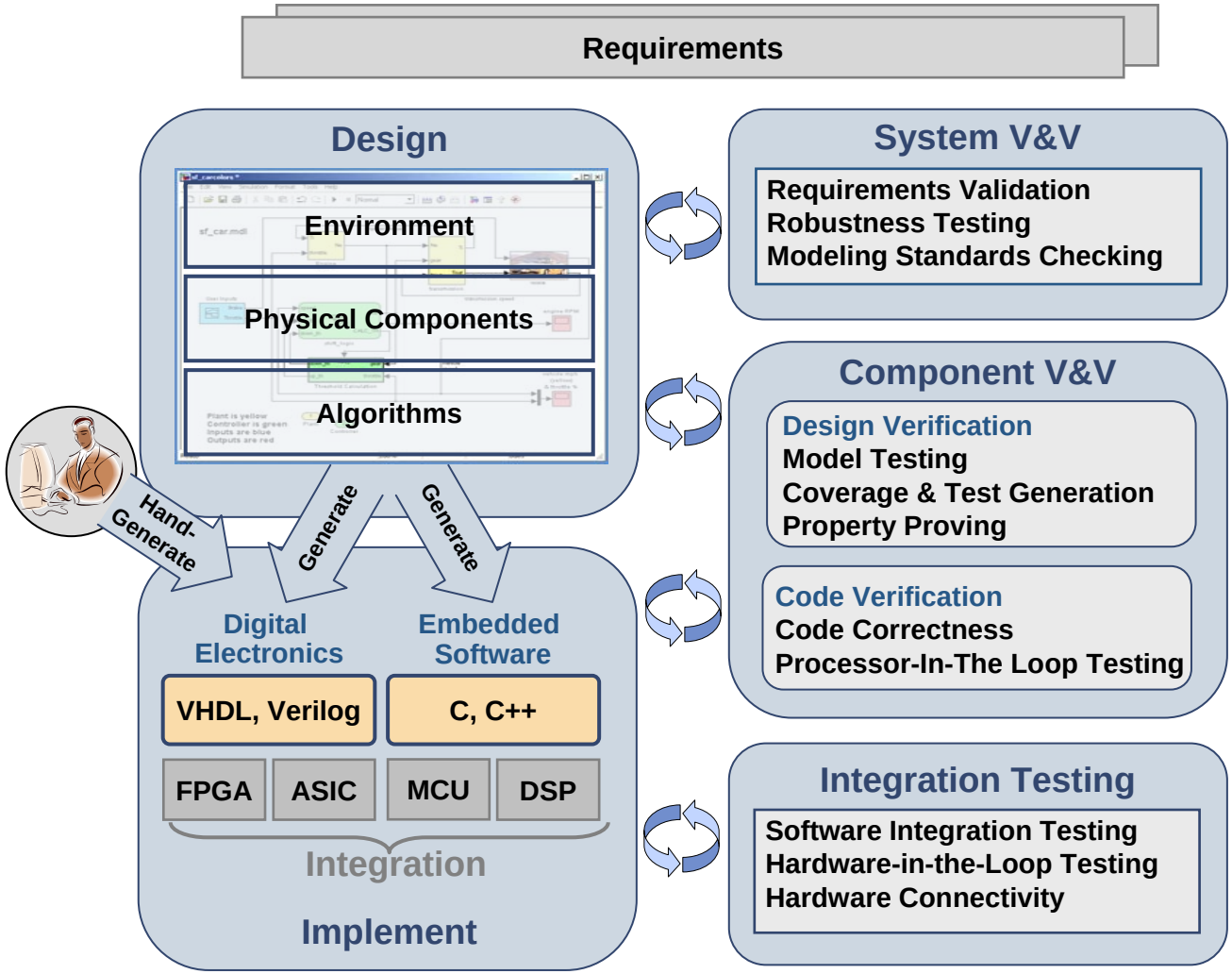
Started Execution: 17-Mar-2008 14:08:14
Ended Execution: 17-Mar-2008 14:08:14

Summary

Model Hierarchy/Complexity:		Test 1					
		D1		C1		MCDC	
1. ThrustReverserDeployLogic_harness	46	100%		100%		100%	
2. .... Test Unit (copied from ThrustReverserDeployLogic)	45	100%		100%		100%	
3. ..... ThrustReverserDeployLogic	45	100%		100%		100%	
4. ..... SF: ThrustReverserDeployLogic	44	100%		100%		100%	
5. ..... SF: airspeedOK	2	100%		NA		NA	
6. ..... SF: isRolling	4	100%		100%		100%	
7. ..... SF: onGround	12	100%		100%		100%	
8. ..... SF: throttleOK	12	100%		100%		100%	
9. ..... SF: thrustReversers	4	100%		100%		100%	
10. ..... SF: warnings	8	100%		NA		NA	
11. ..... SF: airspeedSensors	2	100%		NA		NA	
12. ..... SF: throttleSensors	2	100%		NA		NA	
13. ..... SF: wowSensors	2	100%		NA		NA	
14. ..... SF: wsSensors	2	100%		NA		NA	



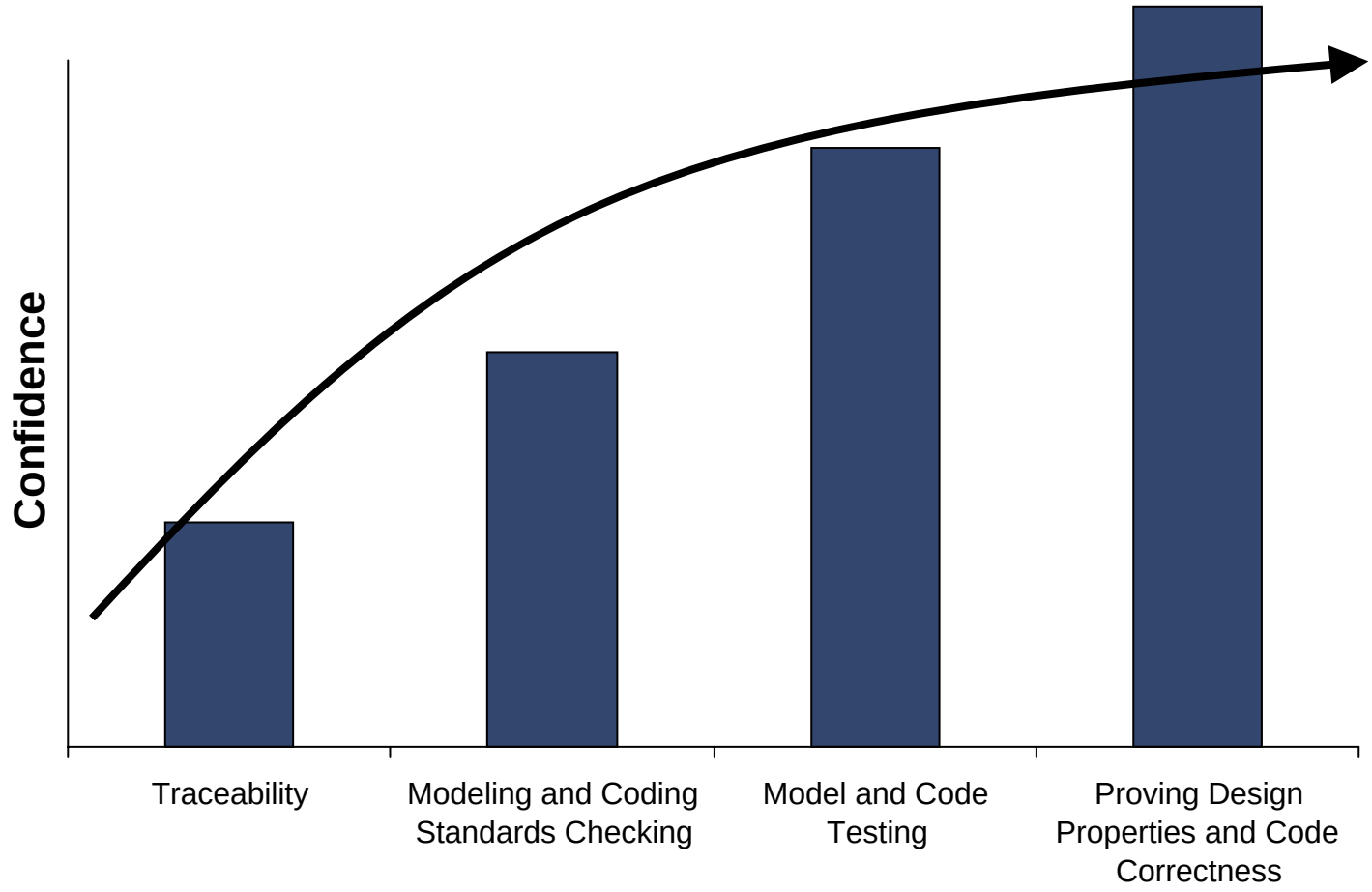
Address the Entire Development Process



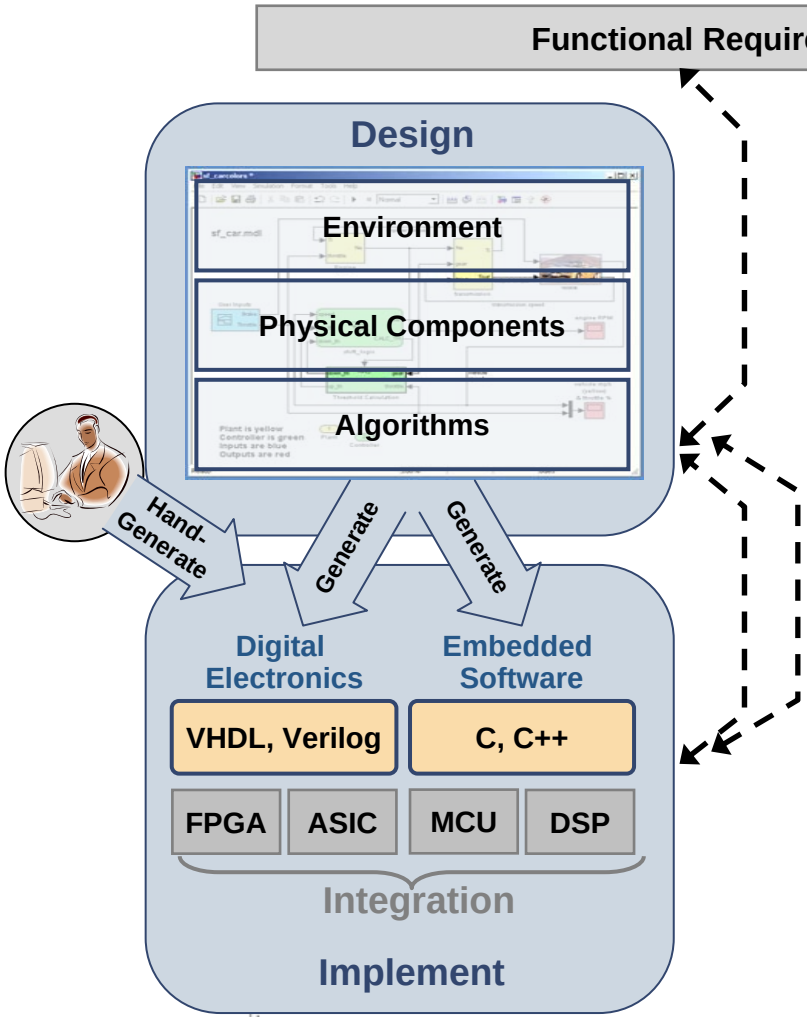
Methods for Early Verification and Validation

- Traceability
  - Requirements to model and code
  - Model to code
- Modeling and Coding Standards
  - Modeling standards checking
  - Coding standards checking
- Testing
  - Model testing in simulation
  - Processor In the loop
- Proving
  - Proving design properties
  - Proving code correctness

Increasing Confidence In Your Designs

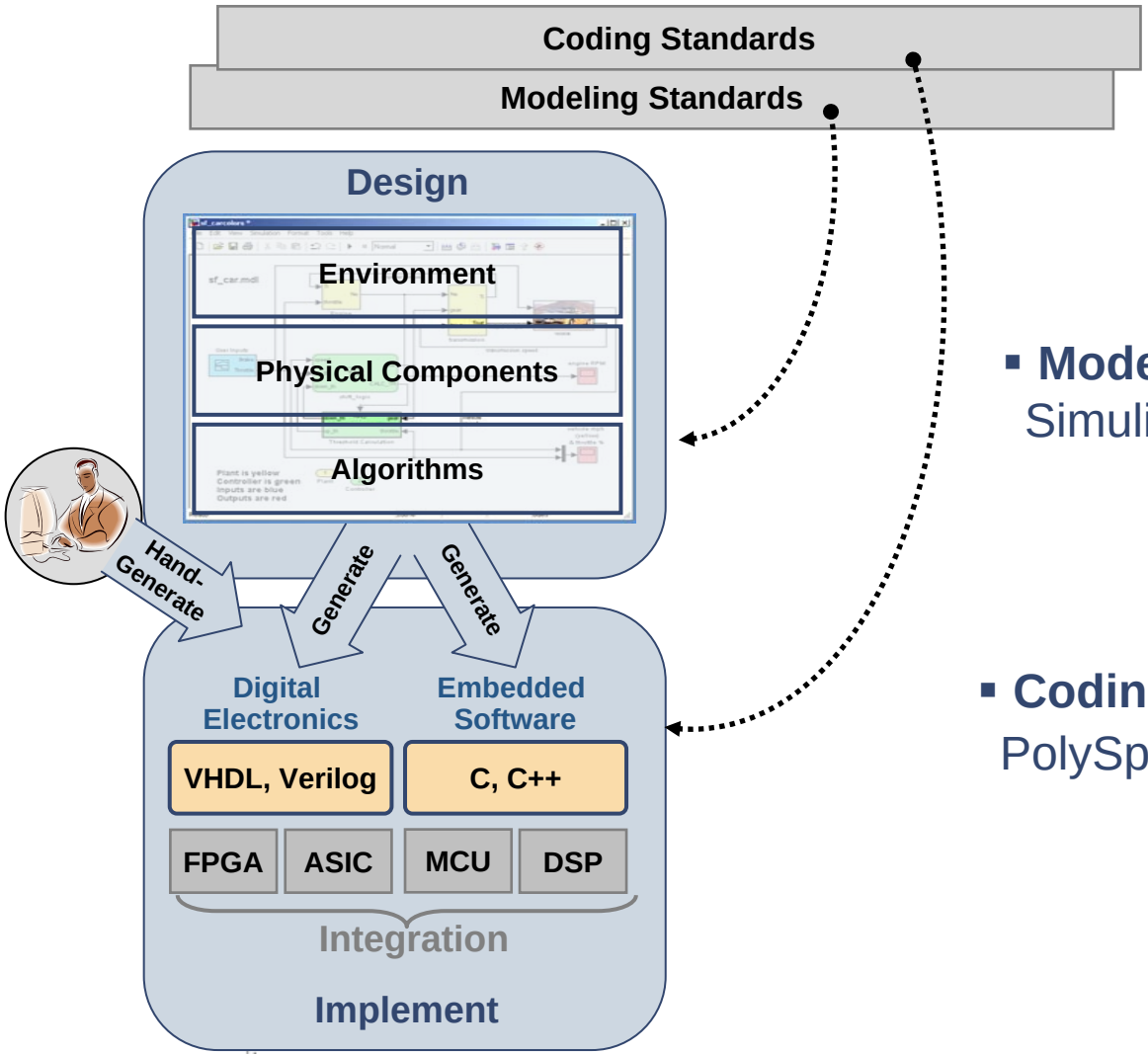


Traceability



- **Tracing Requirements⇔Model**
  Simulink® Verification and Validation™
- **Tracing Model⇔Source Code**
  Real-Time Workshop® Embedded Coder™
- **Tracing Requirements⇔Source Code**
  Simulink Verification and Validation

Modeling and Coding Standards

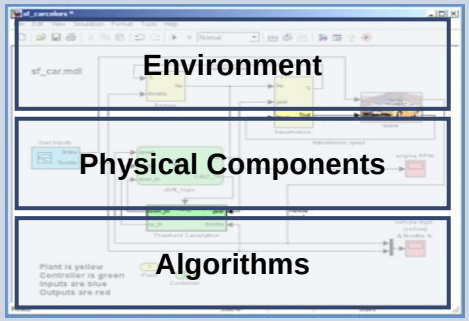


- **Modeling Standards Checking**
  Simulink Verification and Validation
- **Coding Standards Checking**
  PolySpace™ Client™ for C/C++

Early Validation and Robustness Testing

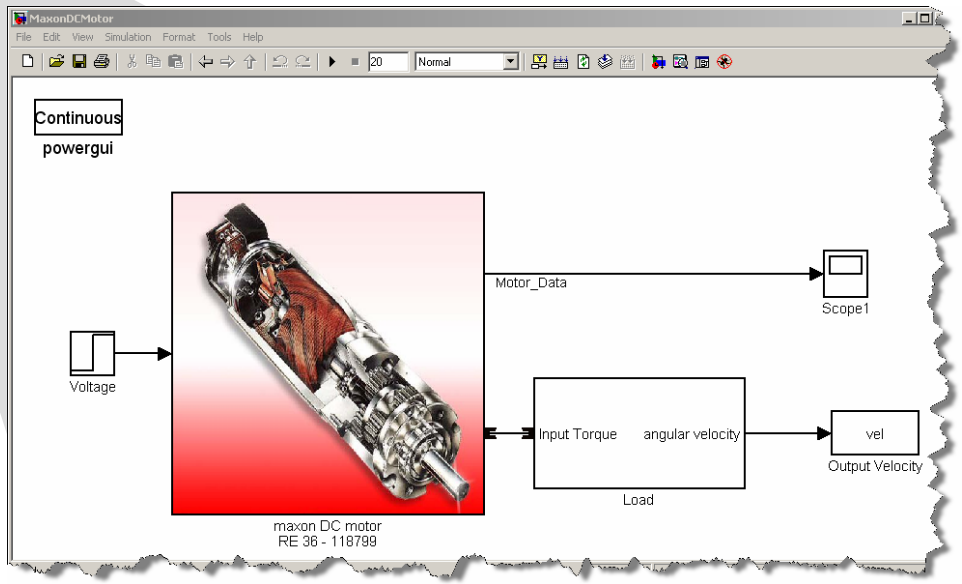
Requirements

Design

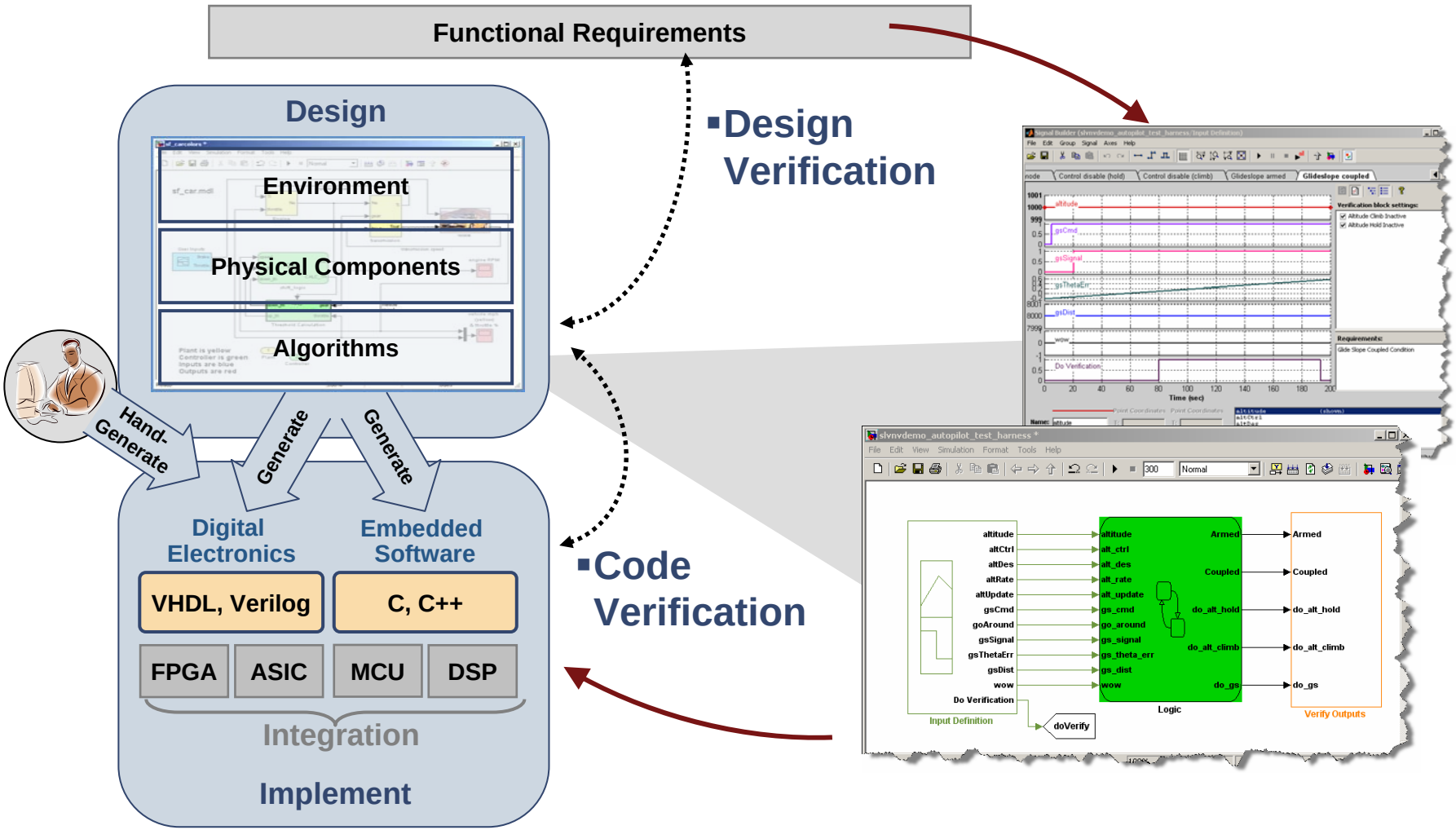


System V&V

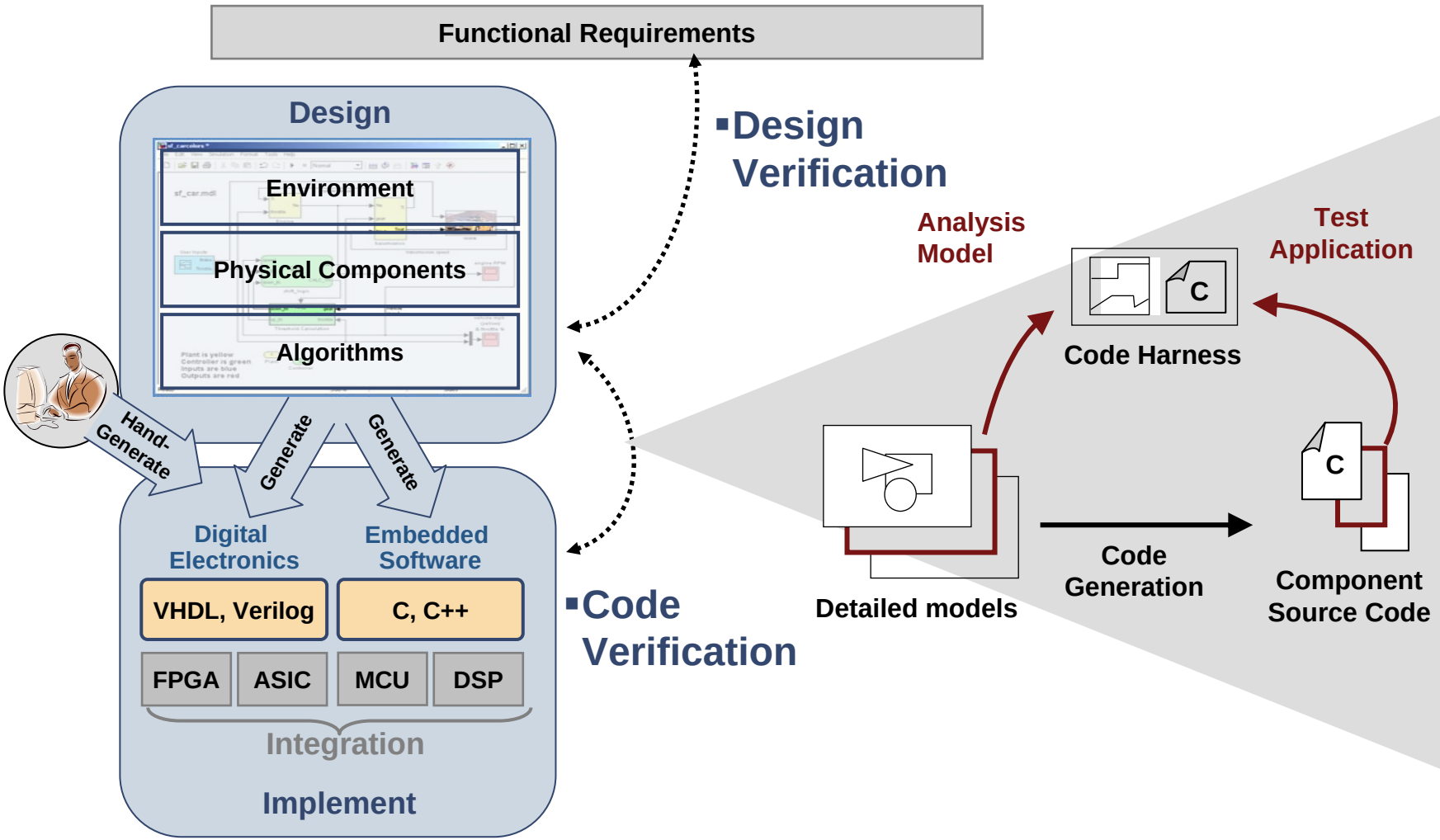
- Requirements Validation
- Robustness Testing
- Modeling Standards Checking



Component Testing



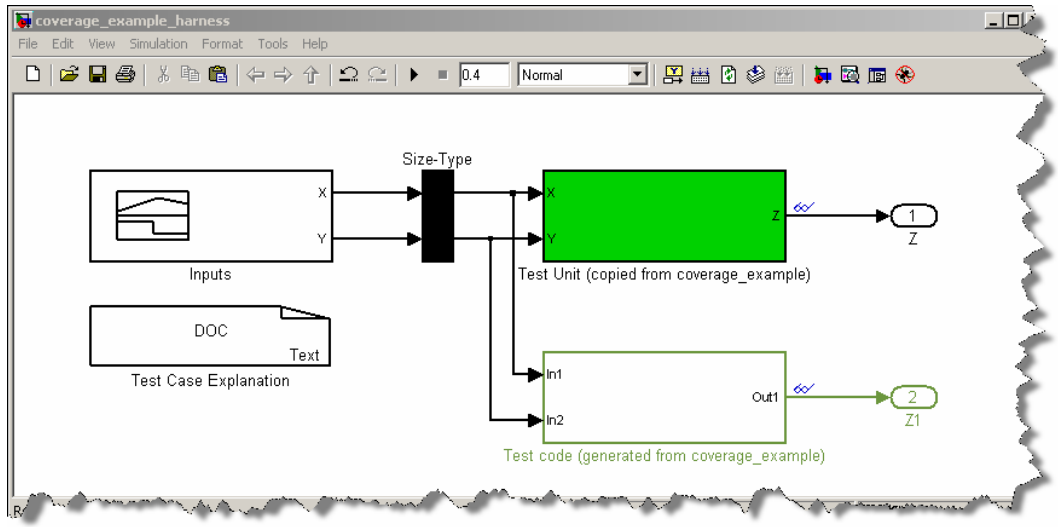
Test Generation Workflow



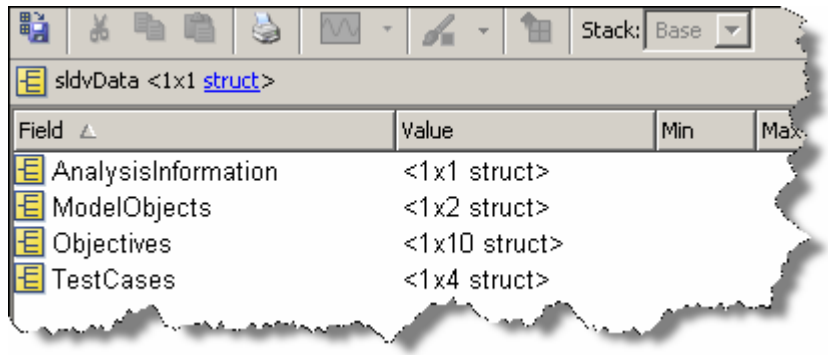
Code Testing with Generated Signals

Simulink

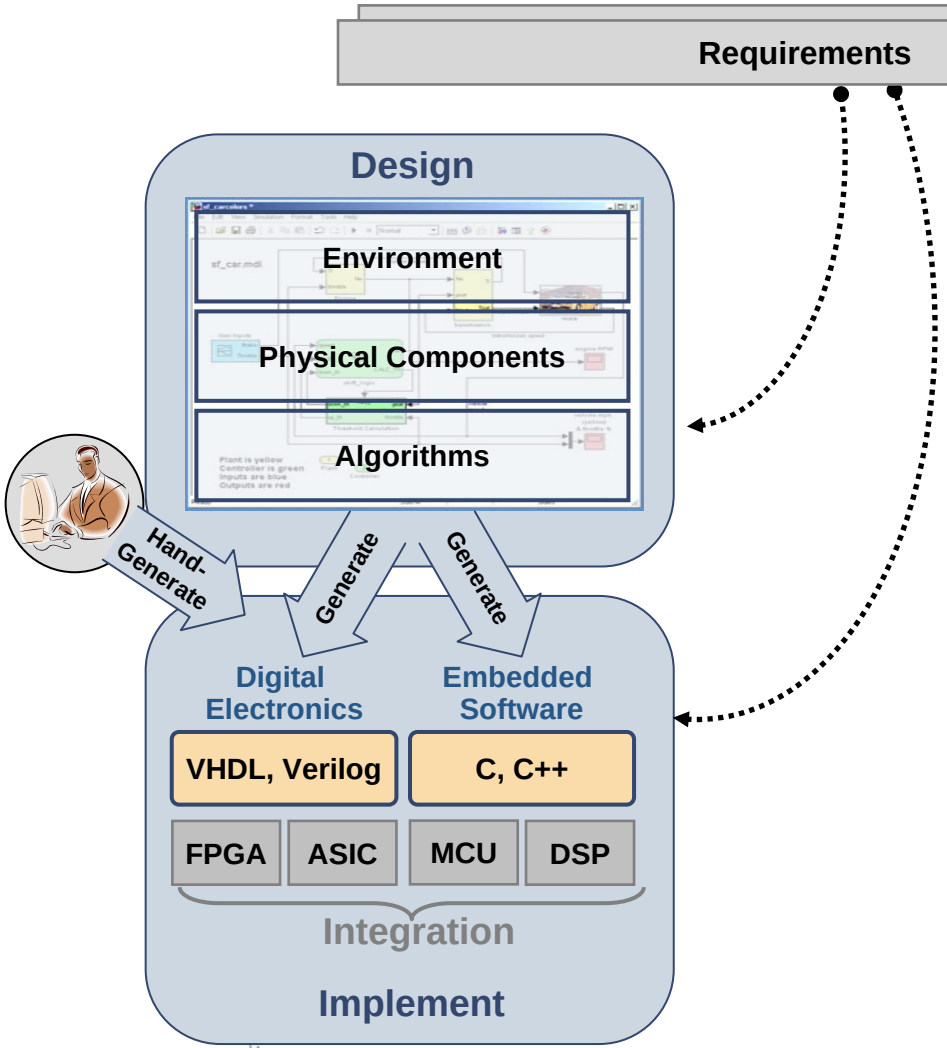
- Software-in-the-loop
  - On the host
- Processor-in-the-loop
  - On the target processor



- Independent code testing environment
  - Generated signals and model outputs are saved as a .mat data file
  - Exported input signals drive code tests
  - Exported model outputs become expectation values for code testing



Proving



- **Proving Design Properties**
  Simulink Design Verifier

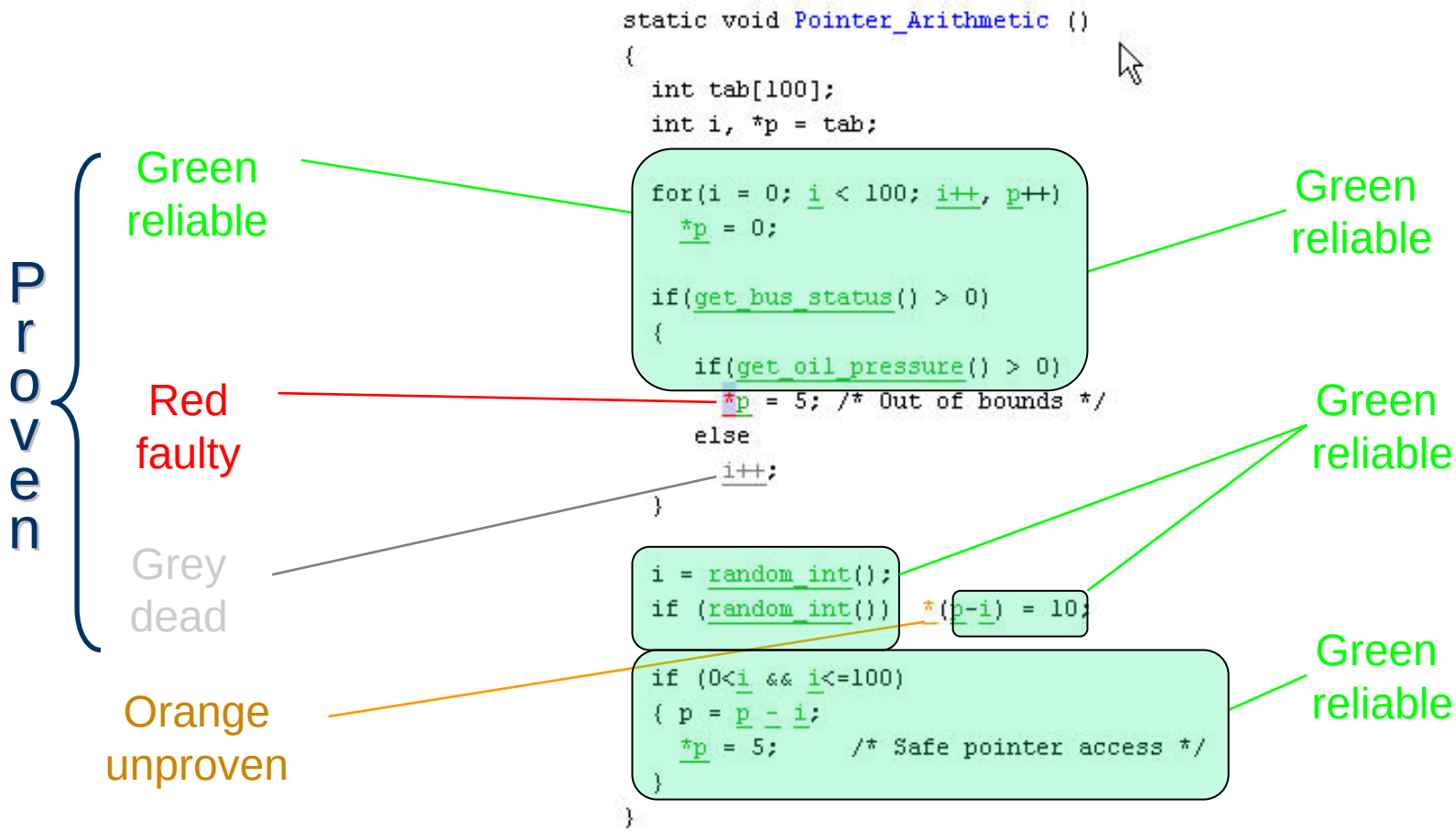
  Prove that design meets the key functional requirements

- **Proving Code Correctness**
  PolySpace™ Server for C/C++

  Prove that code meets non-functional runtime requirements

Code Correctness

Formal method: Abstract Interpretation



Results are proven for all possible executions of the code!!

Summary

- Model-Based Design enables early verification and validation!
- Early verification and validation methods improve and optimize your existing development process.
- Early problem detection significantly reduces time spent debugging – shorter time to resolution

Master Class Invitation

- Methods for Early Verification and Validation
  - Robustness Testing
  - Automatic Test Generation
  - Property Proving