Model-Based Verification of Automotive Controllers

Rance Cleaveland, PhD

Professor of Computer Science, University of Maryland

and

Executive & Scientific Director, Fraunhofer USA Center for Experimental Software Engineering (CESE)

Fraunhofer USA

Center for Experimental Software Engineering

UNIVERSITY OF MARYLAND

This Talk

- Model-based validation
  - … of automotive software product lines
  - … using instrumentation-based verification
- Talk structure
  - Modeling in automotive software development
  - Instrumentation-based verification
  - Product lines
  - An approach to product-line validation
  - Conclusions

Automotive Software

- Driver of innovation

  90% of new feature content based on software [GM]

- Rising cost

  20% of vehicle cost [Conti], 50% for hybrids [Toyota]

- Warranty, liability, quality

  High-profile recalls in Germany, Japan, US

Fraunhofer USA
Center for Experimental Software Engineering

UNIVERSITY OF MARYLAND

A Grand Challenge

- Ensure high quality of automotive software
  - ... preserving time to market
  - … at reasonable cost
- Key approach: *Model-Based Development* (MBD)

Fraunhofer USA
Center for Experimental Software Engineering

UNIVERSITY OF MARYLAND

Traditional Software Development

?

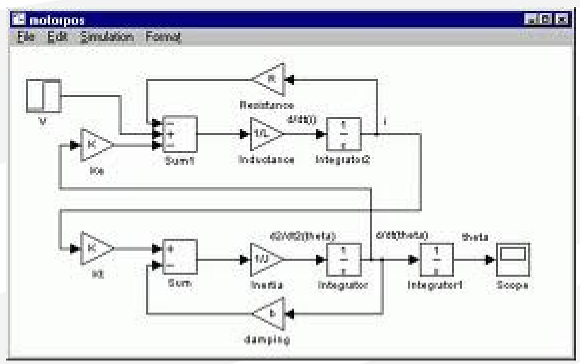
Requirements / specs / designs / test plans / etc.

Source code

Model-Based Development

Use models (MATLAB® / Simulink®) as designs / specs

?



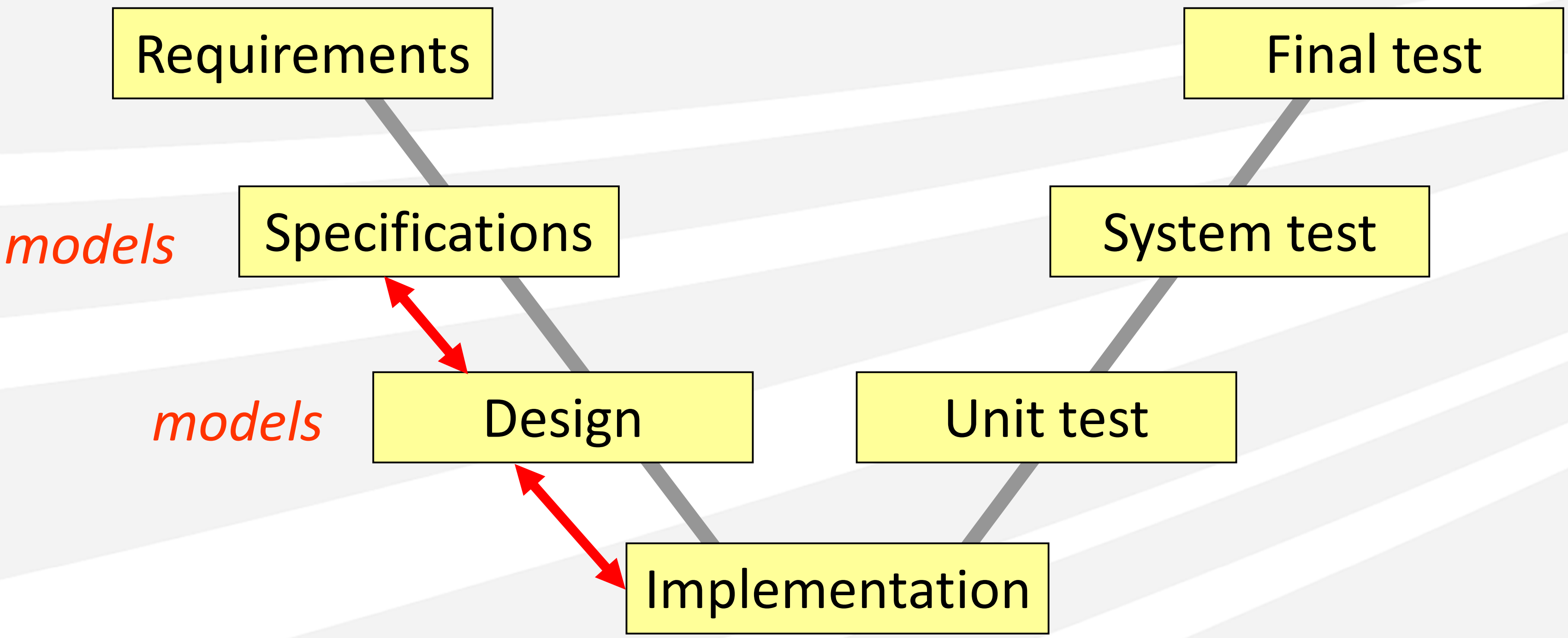
?

Requirements / test plans / etc.

Design / spec

Source code

Model-Based Development (cont.)



Main motivation: autocode! Also:

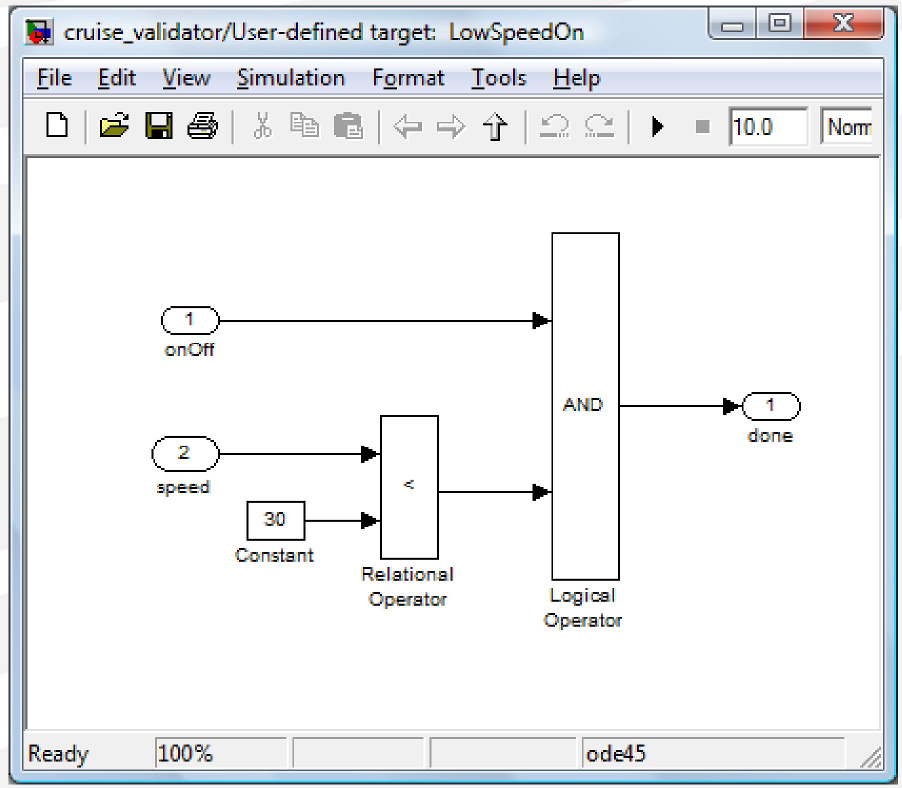
- Models support V&V, testing, communication among engineers
- Models can be managed electronically

Fraunhofer USA
Center for Experimental Software Engineering

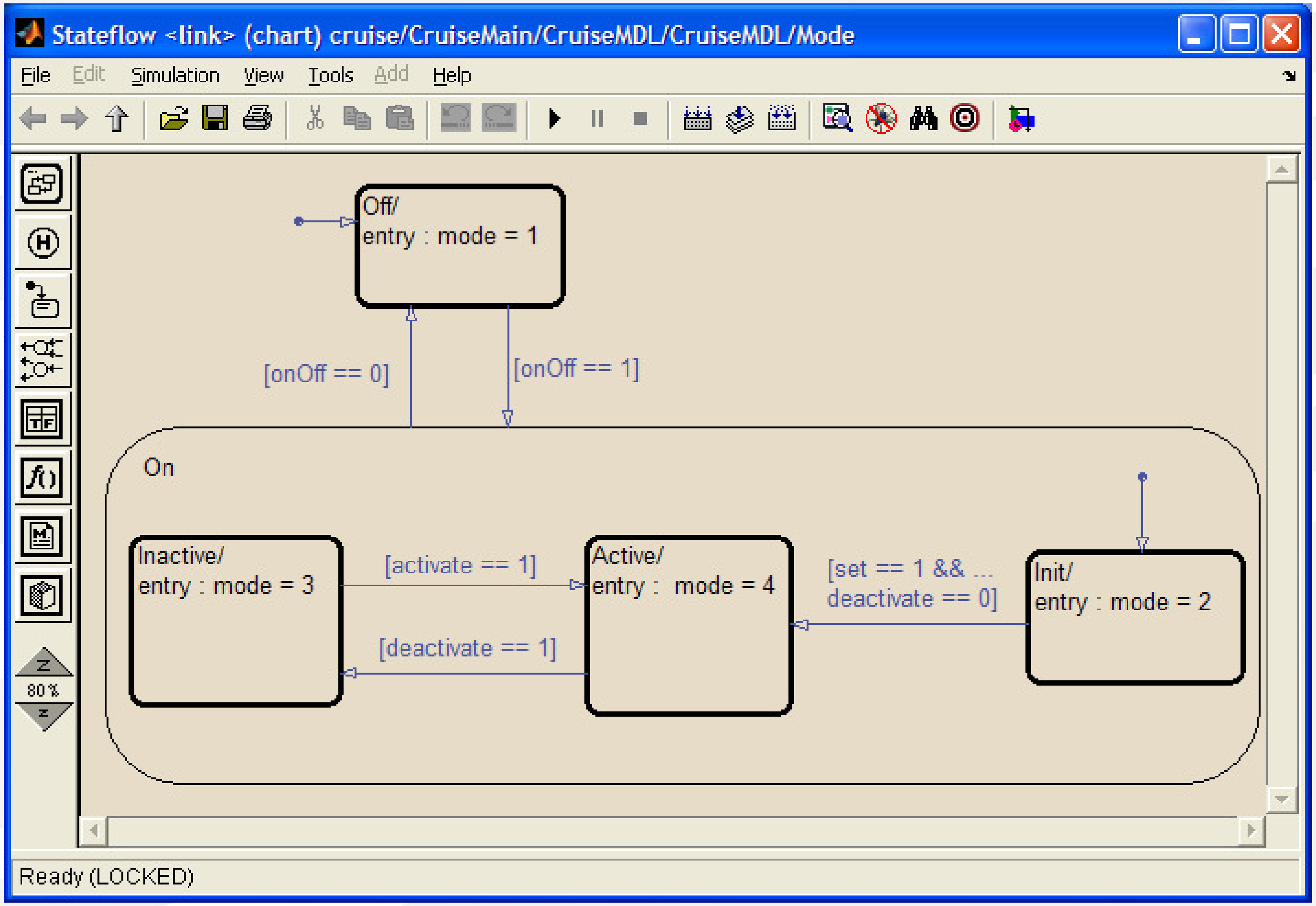
UNIVERSITY OF MARYLAND

Simulink®

- Block-diagram modeling language / simulator of The MathWorks, Inc.
- Hierarchical modeling
- Continuous-time and discrete-time simulation
- Used in MBD of control software



Stateflow®



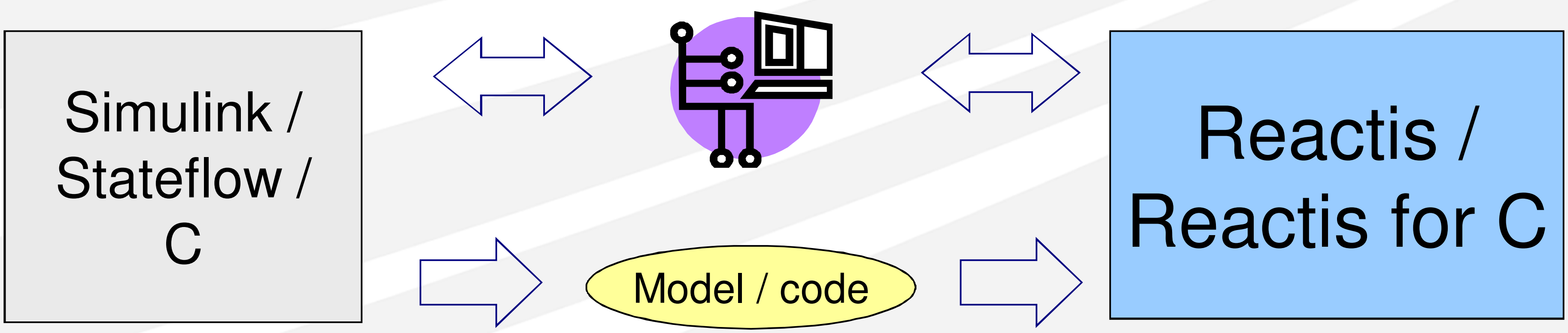
Fraunhofer USA
Center for Experimental Software Engineering

UNIVERSITY OF MARYLAND

Reactis®

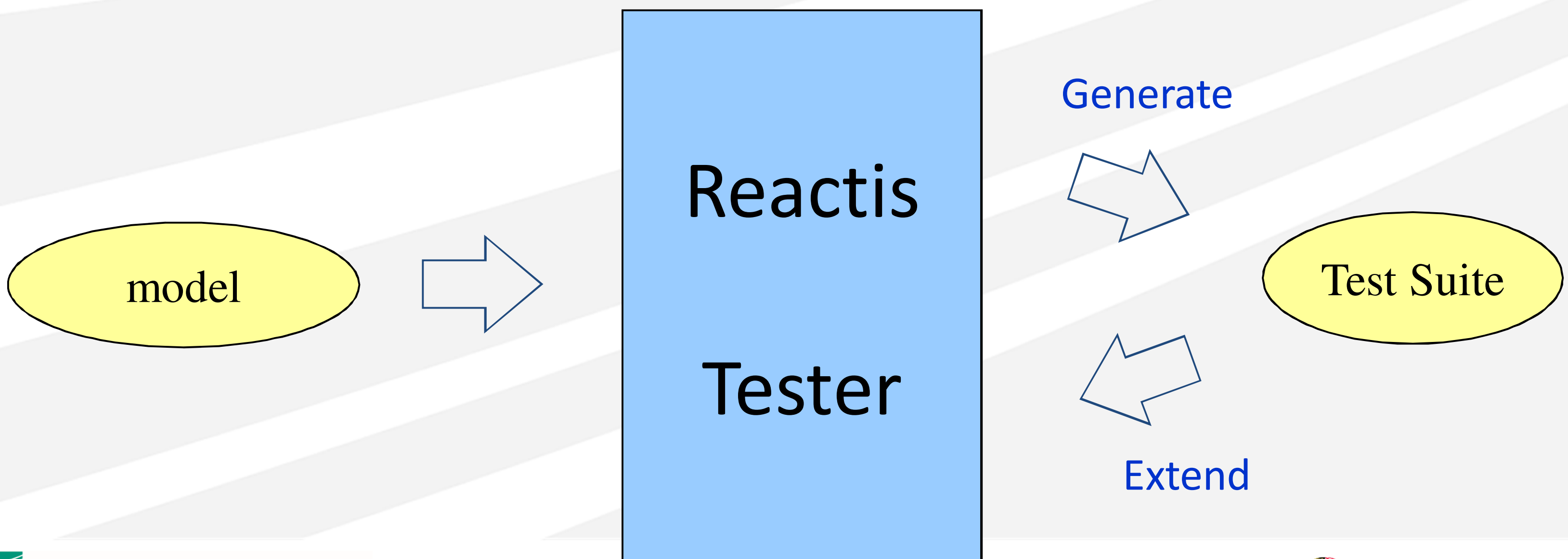
A model-based V&V tool from Reactive Systems, Inc.

Tester	Generate tests from models (also C)
Simulator	Run, fine-tune tests
Validator	Validate models / C



Generating Tests: Guided Simulation

Reactis systematically generates inputs to drive simulation runs to cover model, produce test suites.



Generated Test Data

Reactis Test-Suite Browser: cruise.rst

File View Help

Test 2 (5 steps)

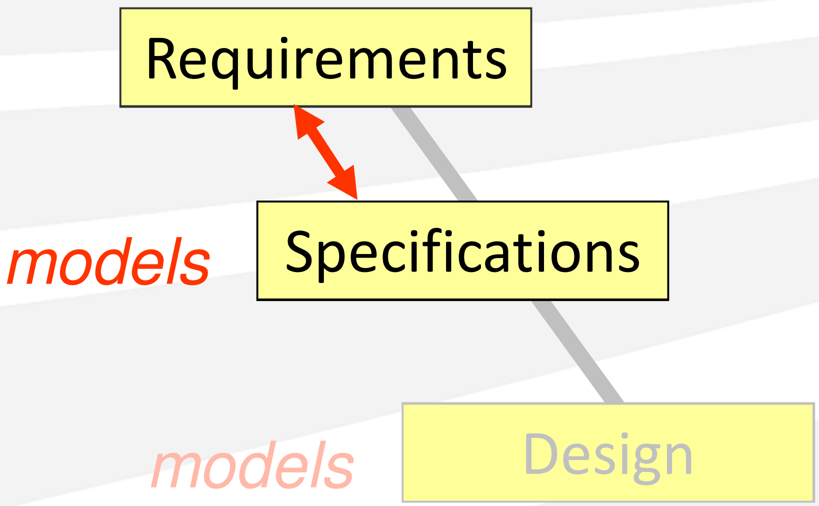
Port	Step 1	Step 2	Step 3	Step 4	Step 5
Inputs					

1: onOff	0.0	1.0	0.0	1.0	1.0
2: accelResume	0.0	1.0	1.0	1.0	1.0
3: cancel	1.0	0.0	0.0	1.0	1.0
4: decelSet	0.0	0.0	1.0	0.0	1.0
5: brake	1.0	1.0	0.0	1.0	0.0
6: gas	1.0	0.0	1.0	0.0	1.0
7: inactiveThrottleDelta	0.1	0.0	0.1	-0.1	0.0
8: drag	-0.0093...	-0.0089...	-0.0094...	-0.0088...	-0.0089...
Outputs					

1: active	0.0	0.0	0.0	0.0	0.0
2: throttleDelta	-0.1	0.0	-0.1	0.0	0.0
___t___	0.0	1.0	2.0	3.0	4.0

Configuration Variable	Value
InitialSpeed	15.79179838897

Ongoing Research



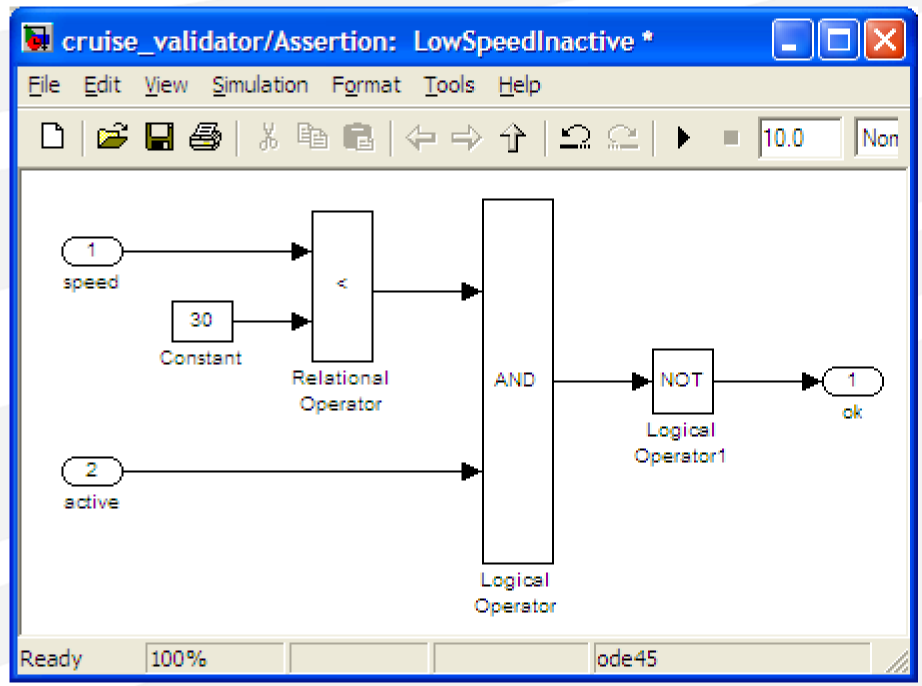
Design-time modeling, requirements verification

Instrumentation-Based Verification

- Model-validation technique supported by Reactis
- Combines assertions in models, testing

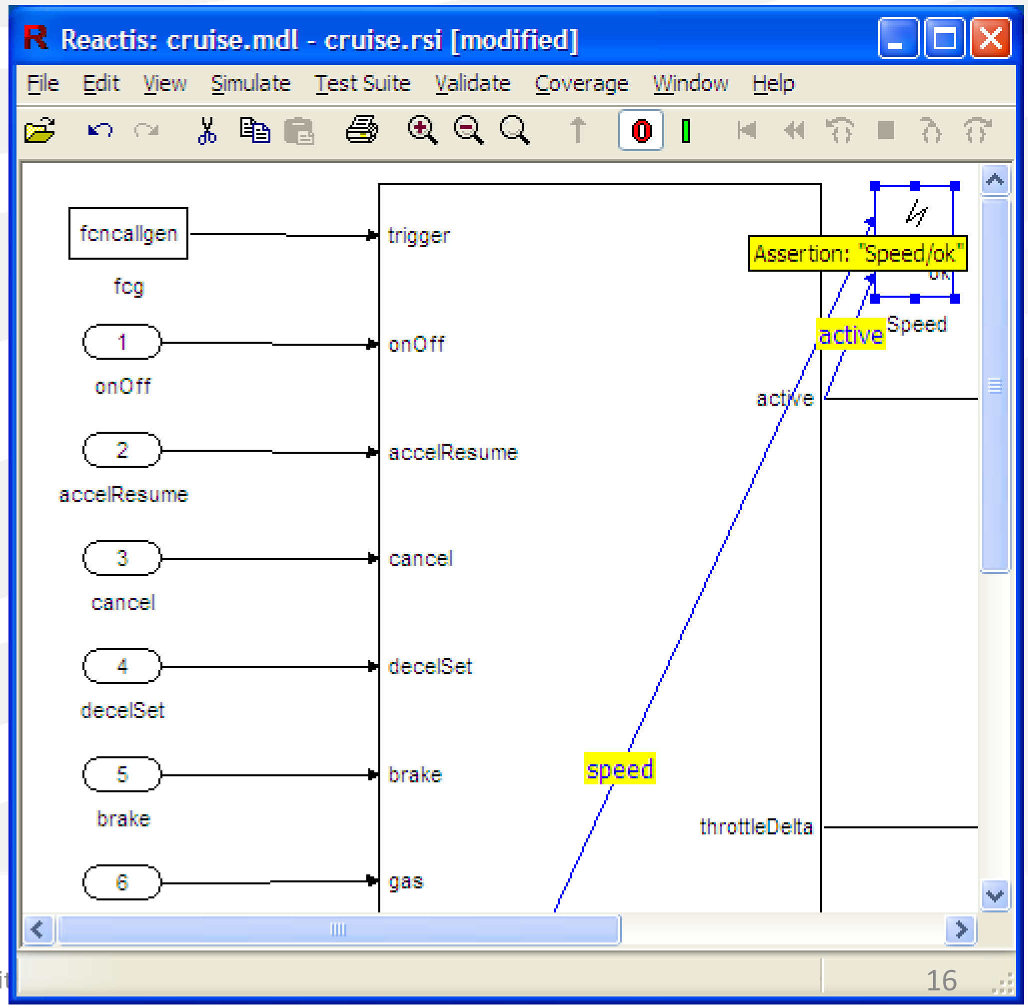
Instrumentation-Based Verification: Requirements

- Automatic verification requires formalized requirements
- IBV: formalize requirements as *monitor models*
- Example
  "If speed is < 30, cruise control must remain inactive"



Instrumentation-Based Verification: Checking Requirements

- Instrument design model with monitors
- Use coverage testing to check for monitor violations
- Reactis:
  - Separates instrumentation, design
  - Automates test generation



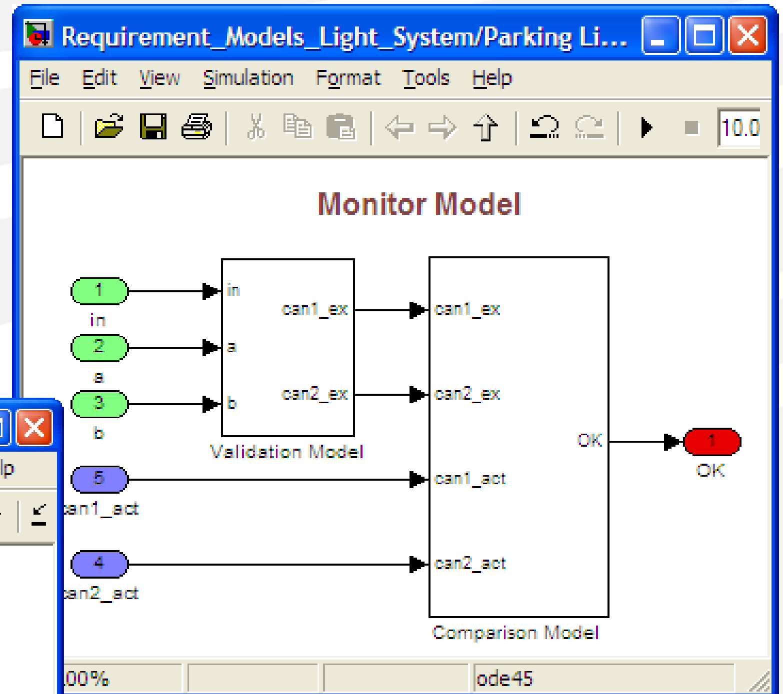
Fraunhofer USA
Center for Experimental Software Engineering

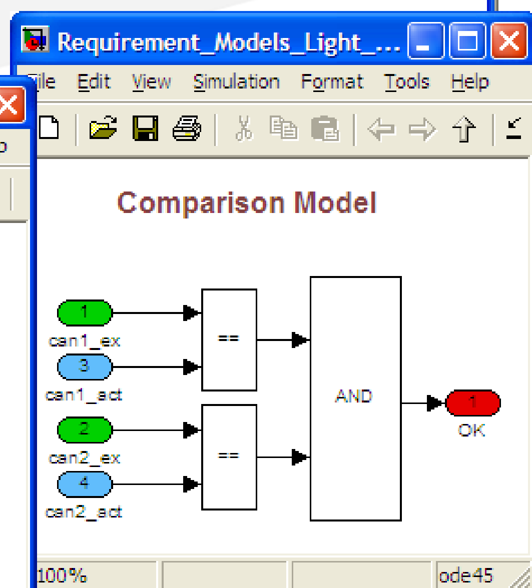
IBV Works

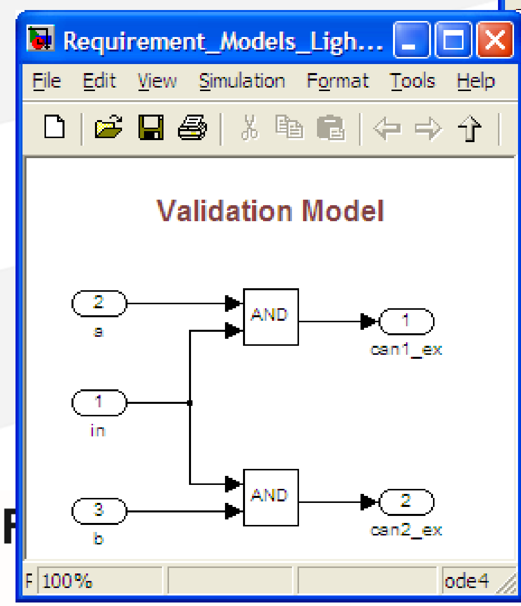
- Three-month case study with Tier-1 automotive supplier on production system
- Artifacts
  - 300-page requirements document
  - Some source code
- Results (intern)
  - 62 requirements for 10 design features formalized as monitor models
  - Requirements checked on feature models
  - 11 inconsistencies in requirements identified
  - Key technical insight: architecture for monitor models

From Requirements to Monitors: A Monitor Model Architecture

"[This] is the complete description of the control of the CAN output signals can1 and can2 produced by Function A. Function A can be activated only with in = 1. The activation takes place when either the CAN bus messages a or b is present…."



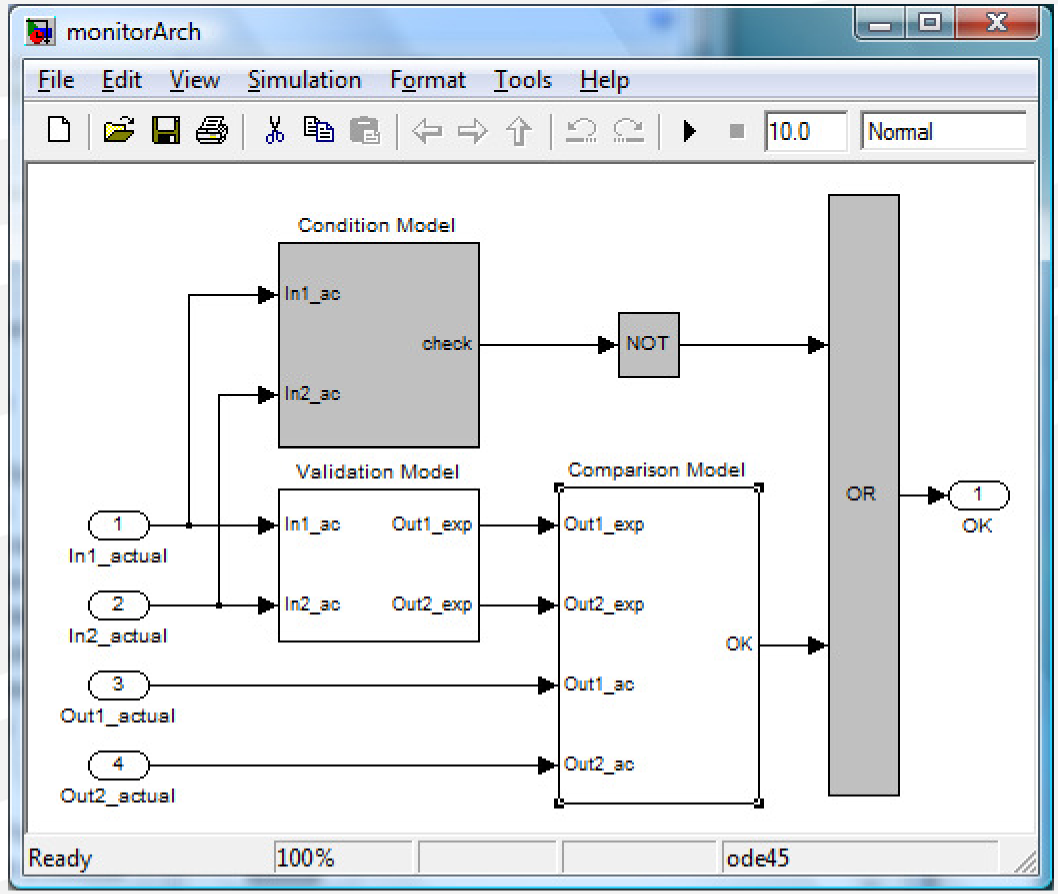




Final Monitor Model Architecture

Need for *conditional requirements*

- Behavior only specified for certain situations
- "If timeout occurs do something"



Software Product Lines

- (From SEI): product line = "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way"
- Key terms
  - Common assets
  - Variation points
  - Variants

SPL in Automotive

- Toyota: 1,800 variants for engine control software
  - Diesel vs. gas vs. hybrid
  - Different emissions regulations
  - Performance profiles for different markets
  - # of cylinders
  - Cruise control?
  - Etc.
- Product lines offer a framework for streamlining development, maintenance
- What about V&V?

Variants in Monitor Modeling

- Fine-grained product-line info often captured at model level
- How can functionality of product-line models be verified?
  - Want to re-use verification effort
  - Some requirements are *universal* (apply to every variant)
  - Others are *variant-specific*

Fraunhofer USA
Center for Experimental Software Engineering

UNIVERSITY OF MARYLAND

Example: Cruise Control

- Product line could include following variants
  - Maximum-speed restriction or not
  - Adaptive or not
  - Manual or automatic transmission
- Sample universal requirement

  If the brake pedal is pressed, the cruise control shall become inactive.
- Sample variant-specific requirement

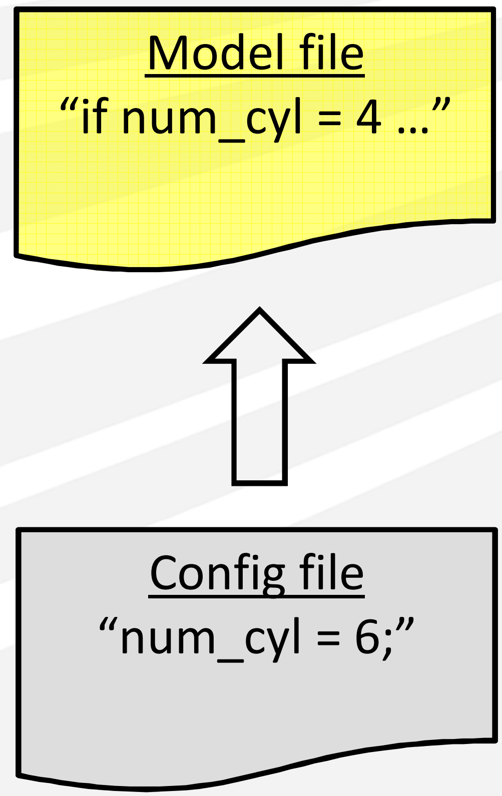
  If the transmission is manual, then the cruise control shall become inactive if the desired speed is inconsistent with the current gear.

How To Do V&V for Product-Line Models?

- Use IBV!
- Result of industrial study
  - Framework for modeling product lines in Simulink
  - Strategy, architecture for variant-specific monitor-models
  - Use of IBV to debug models, find requirements issues

Product-Line Modeling

- Model file defines control functionality
- Configuration file defines parameters
- Some parameters used to define which variant is intended

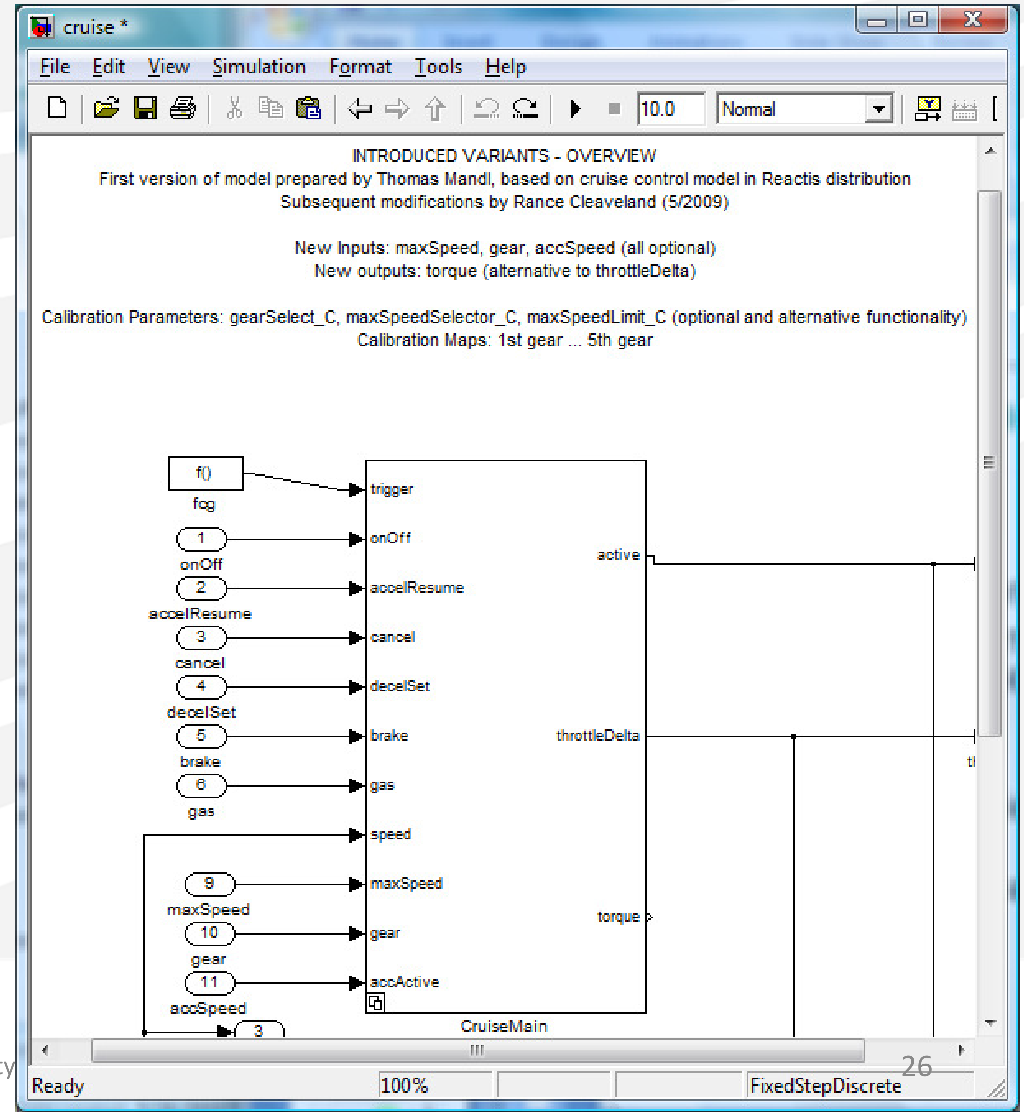


Fraunhofer USA
Center for Experimental Software Engineering

University of Maryland

Pilot Study: Cruise Control

- Simulink model is in Reactis distribution
- Partner adapted it as sample product-line model
- Variants
  - Max-speed limitation
  - Adaptive
  - Manual vs. automatic transmission
  - Output interface

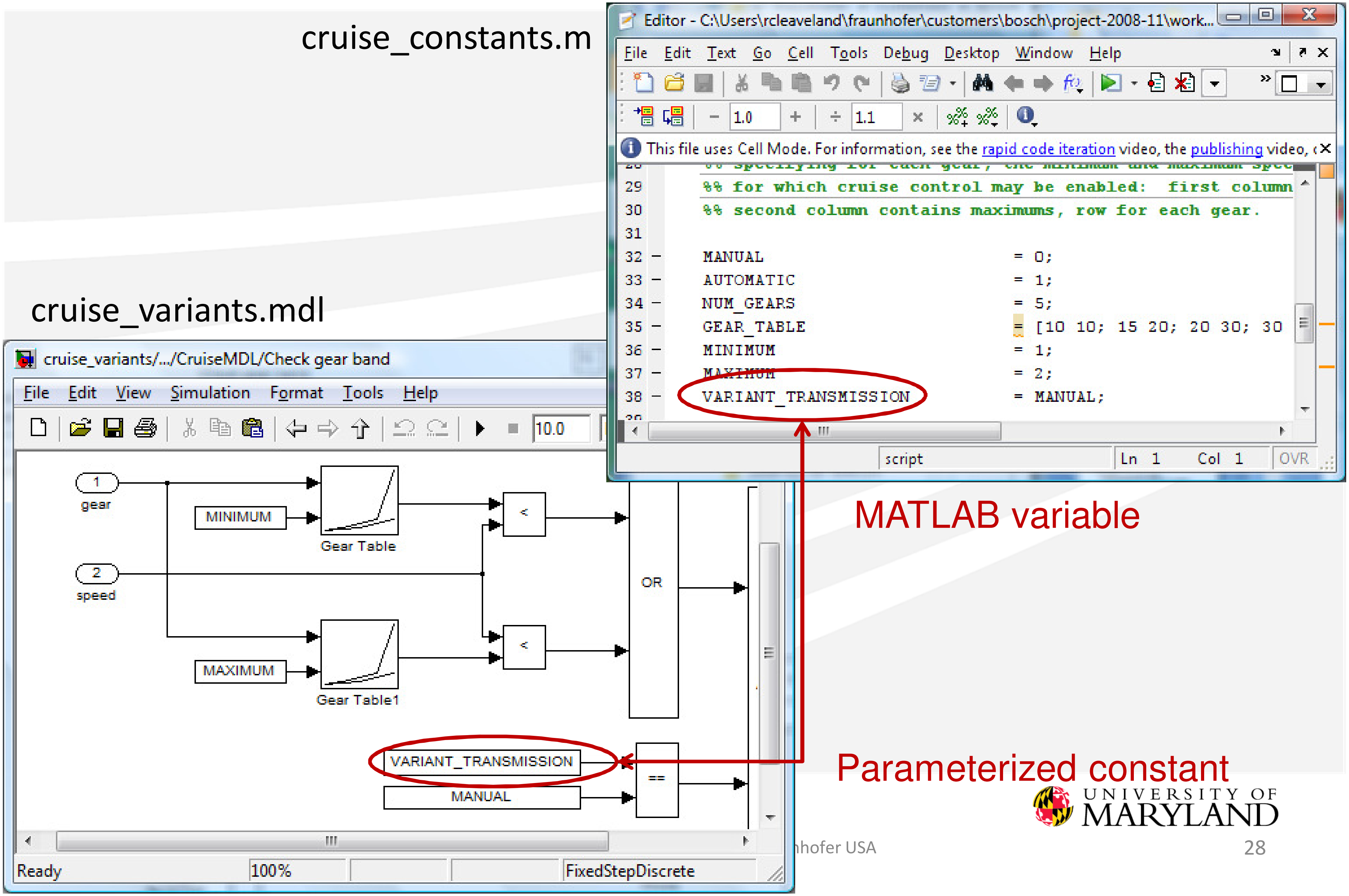


Finalizing Product-Line Model in Simulink / Stateflow

- Program variant selection
  - Introduce parameters into model
  - Define MATLAB variables for use as parameters
- Product line contained in two files
  - cruise_variants.mdl (model)
  - cruise_constants.m (MATLAB variables)

cruise_constants.m

cruise_variants.mdl



MATLAB variable

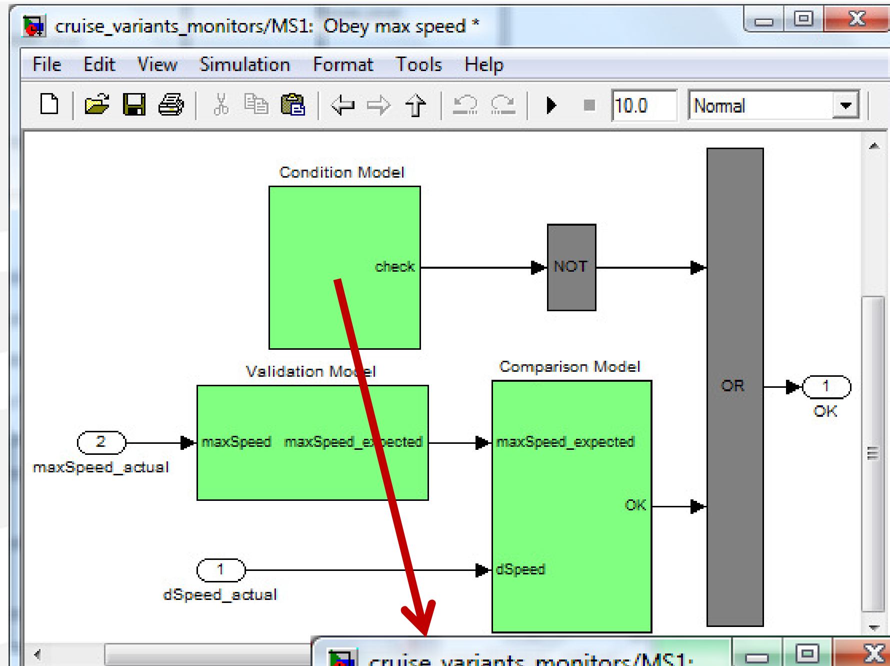
Parameterized constant

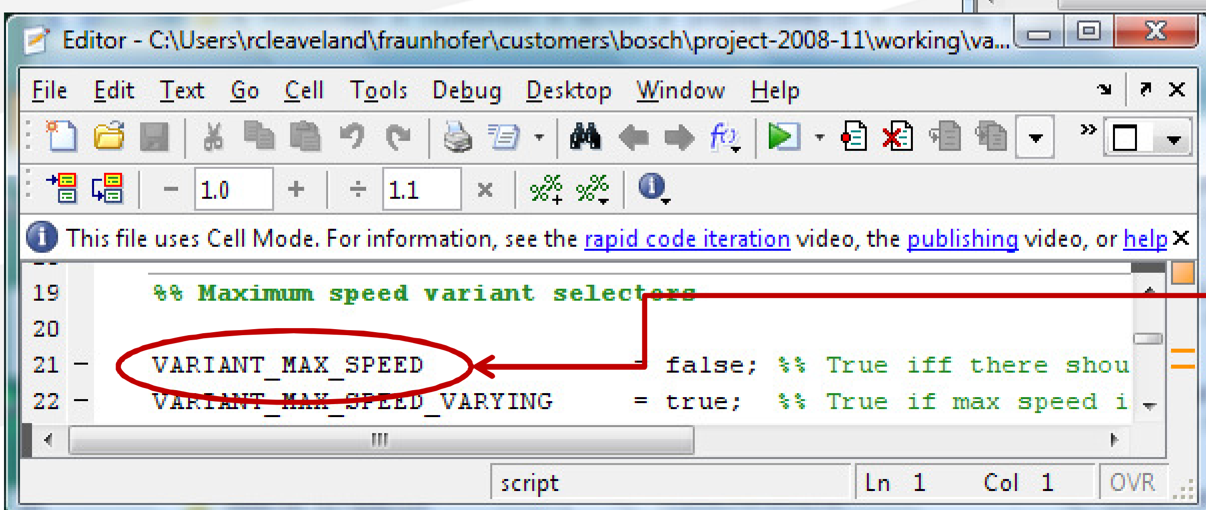
Variant-Specific Monitor Models

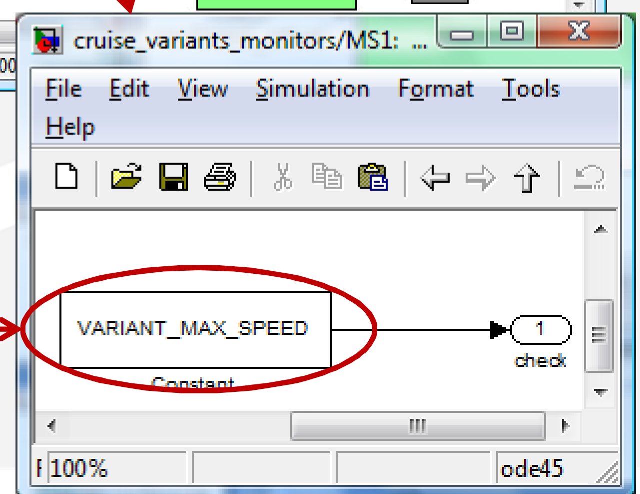
- Idea
  - Configuration files define variant-selection parameters
  - Why not refer to same parameters in monitor models to introduce variant-specificity?
- Pilot study
  - Defined six example variant-specific requirements
  - Translated each into monitor model

Example

[MS1] If the maximum-speed limitation is enabled, the cruise control shall not permit the desired (set) speed to exceed a designated maximum value.



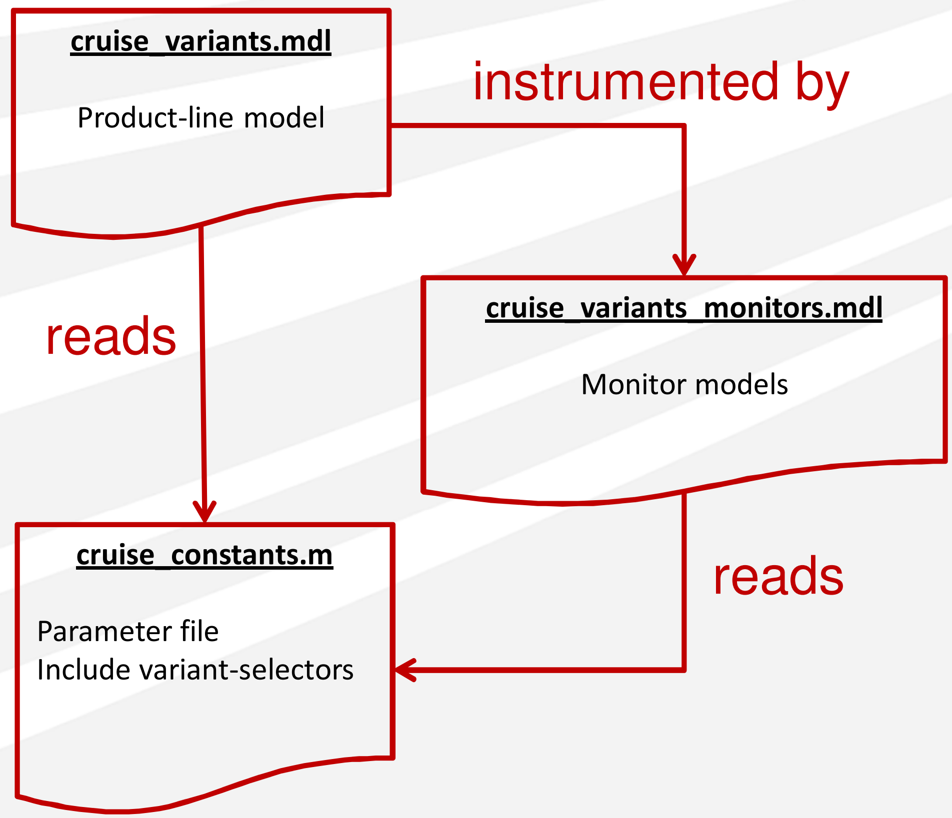




MATLAB variable

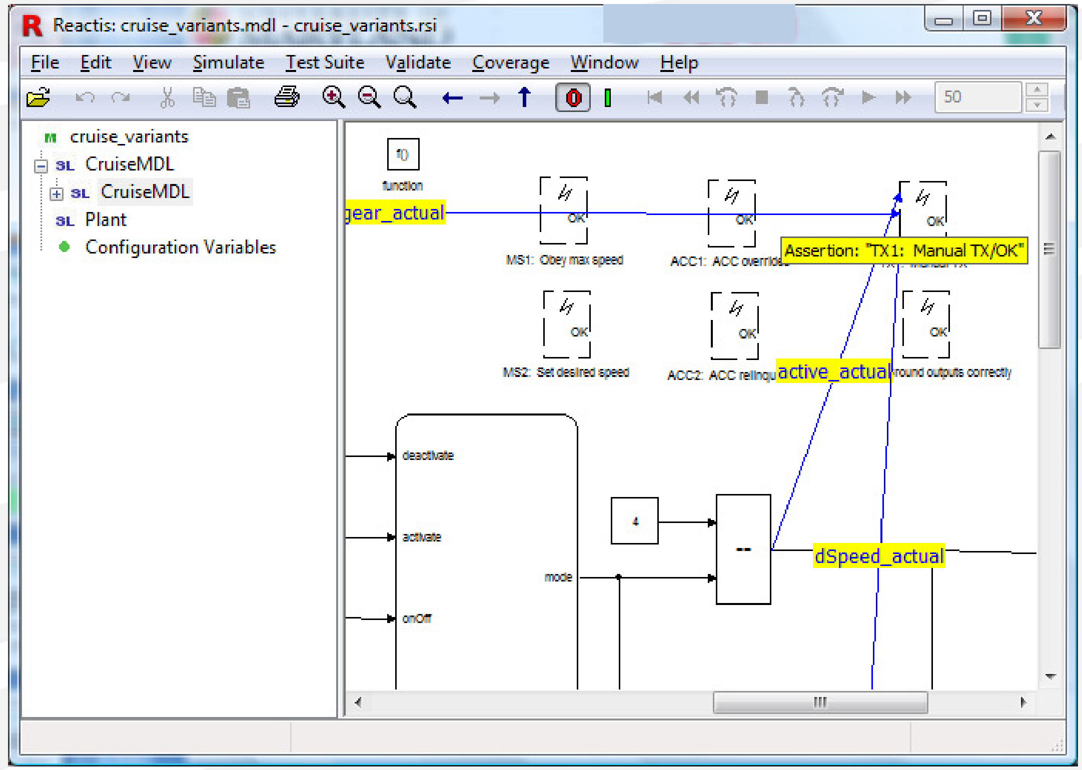
Monitor Model Logistics

- Monitor models stored in single Simulink library file
- Monitor models refer to parameters



Verification

- Product-line model instrumented with monitor models
- Coverage testing used to check for violations
- Reactis® used for both tasks



Verification Results

- Bugs found in product-line model (fixed)
- Bugs found in monitor model (fixed)
- Variant-interaction problem discovered
  - One variant specified maximum speed
  - Other variant specified speed-control by adaptive mechanism

This Talk

Model-based verification of software product lines

– Model product lines in Simulink / Stateflow
– Variant specificity in monitor models
– Instrumentation-based verification
– Variant interactions!

Larger Issues

- Single models vs. parameterized models
  - Typical problem: find parameter settings that ensure satisfaction of requirements
  - Here: parameterize requirements, check consistency of parameterized models vis a vis parameterized requirements
- Parameter interactions
- Requirements are not the always what's required

Thank You!

Rance Cleaveland

University of Maryland / Fraunhofer USA CESE

rance@cs.umd.edu

+1 301-405-8572