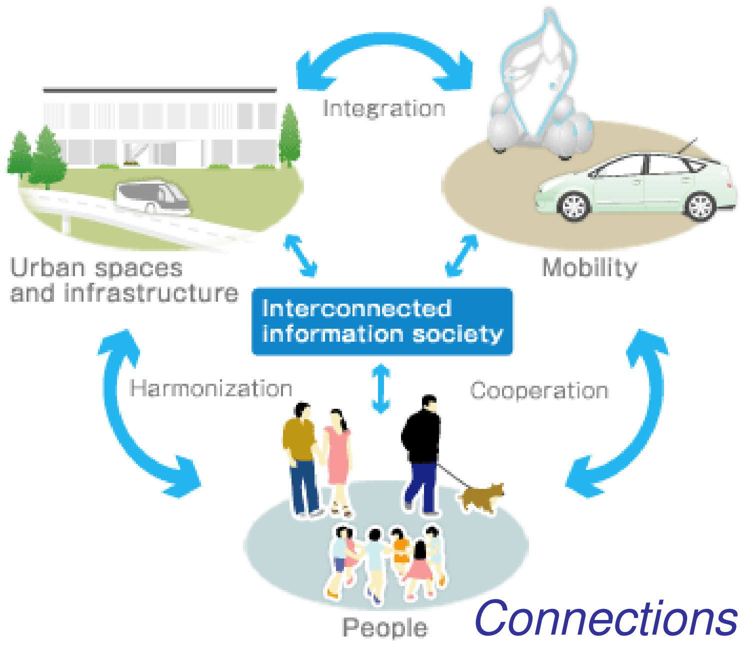
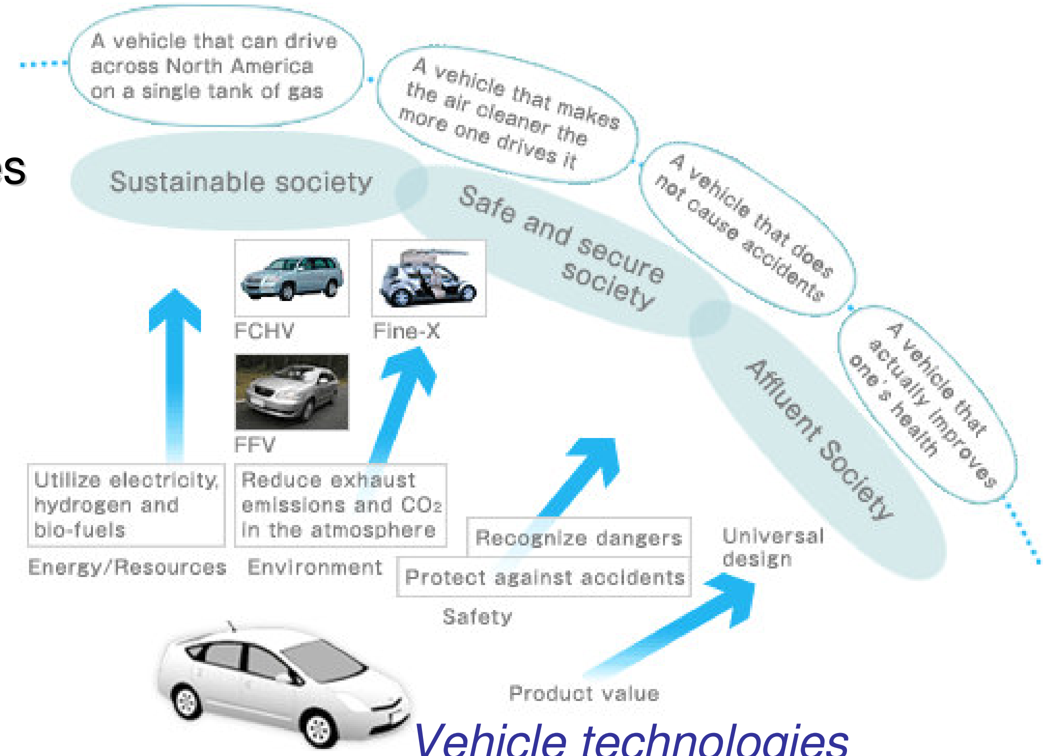
# Toyota's Direction

Our sustainable mobility strategy includes products, partnerships, the urban environment and energy solutions.*


Integration
Urban spaces and infrastructure
Mobility
Interconnected information society
Harmonization
Cooperation
People

*Connections*


A vehicle that can drive across North America on a single tank of gas
A vehicle that makes the air cleaner the more one drives it
A vehicle that does not cause accidents
A vehicle that actually improves one's health
Sustainable society
Safe and secure society
Affluent Society
FCHV
Fine-X
FFV
Utilize electricity, hydrogen and bio-fuels
Reduce exhaust emissions and CO2 in the atmosphere
Recognize dangers
Protect against accidents
Universal design
Energy/Resources
Environment
Safety
Product value

*Vehicle technologies*

*Vehicle-to Vehicle and Infrastructure integration*

*Toyota 2010 North American Environmental Report Highlights

## Presentation Flow

1. Development Pressures → Model-based Development
2. Open Challenge: Verification and Validation (V&V) of In-Vehicle Control Systems
3. V&V Research Directions
4. Discussion

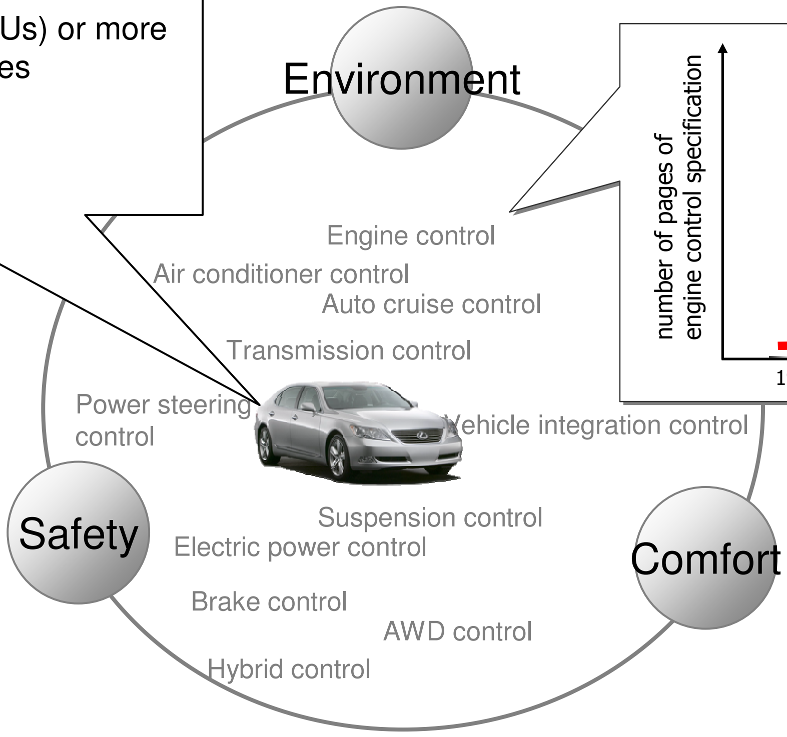
with grateful contributions from:

Koichi Ueda, Hakan Yazarel, Prashant Ramachandra, Derek Caveney, Chrona, UCLA, UC Berkeley CHESS, Carnegie Mellon

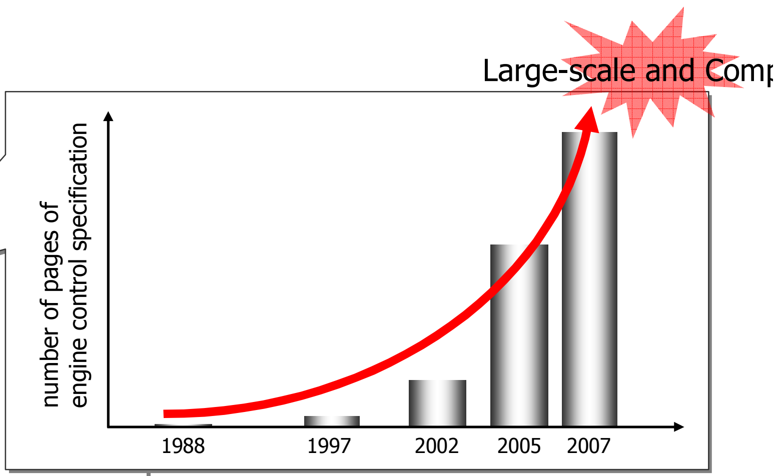
## Product Direction

LEXUS LS460 …
• 100 Electronic Control Units (ECUs) or more
• Program size of seven million lines

Nikkei business (September, 2006)

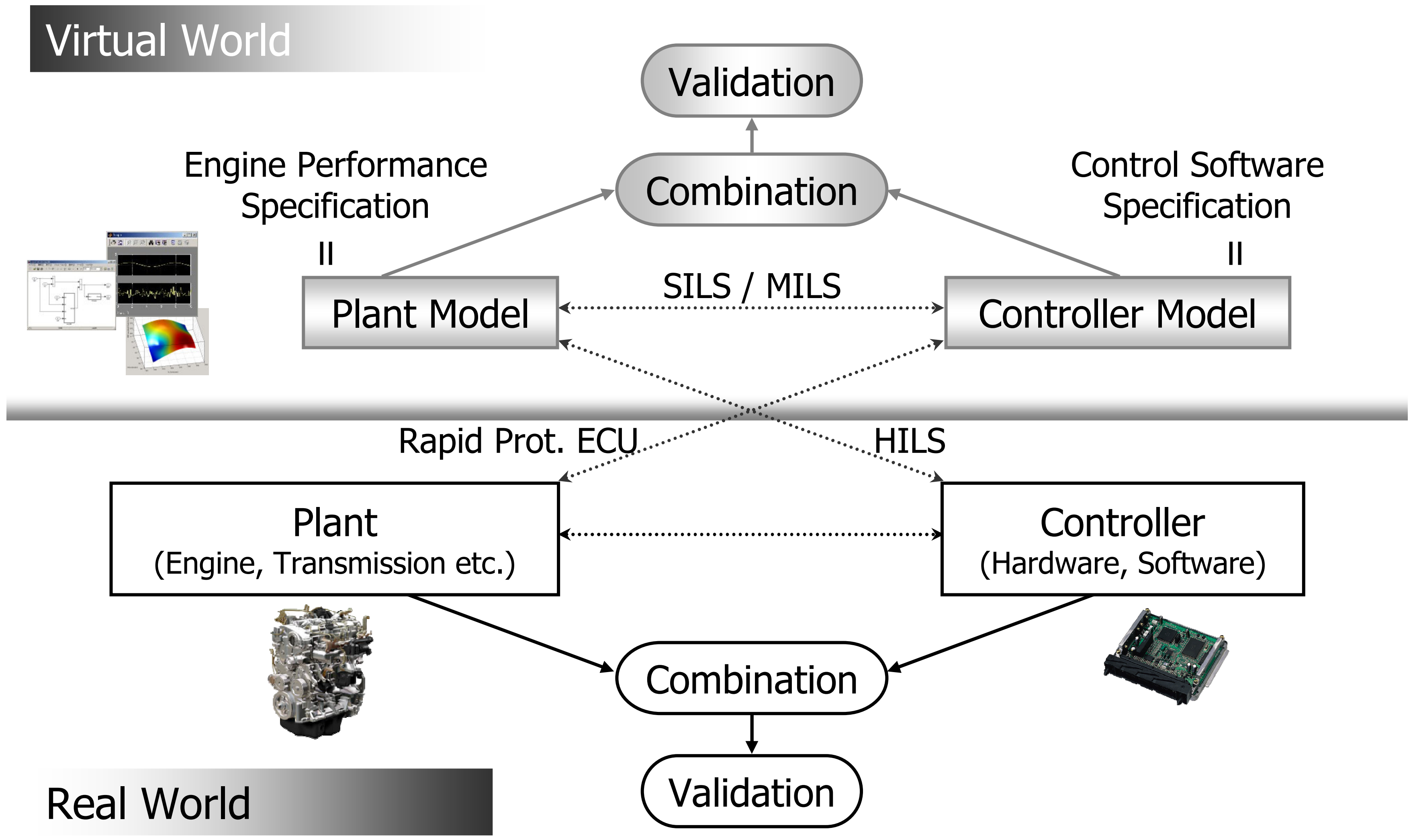


In-vehicle control systems



Product variation, control, integration, and complexity are accelerating in order to improve vehicle performance, address sustainability, and provide new features.

# Model-Based Development: Basic Tooling



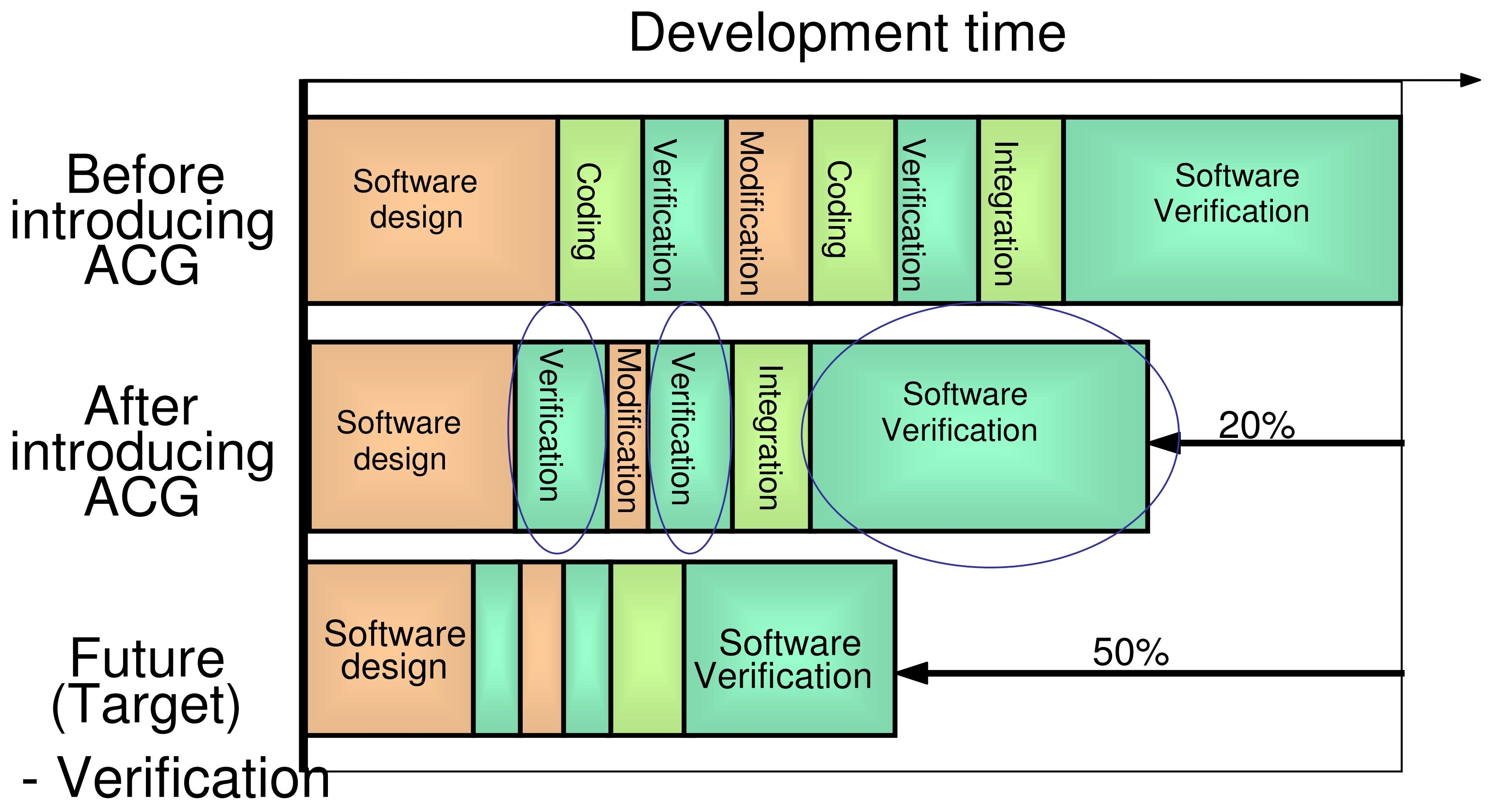
TOYOTA
TOYOTA MOTOR ENGINEERING &
MANUFACTURING NORTH AMERICA

## MBD Focus areas

- Process and Information Management
- Plant Modeling
- Model-based control design
- Calibration
- *Verification and Validation*

# MBD Focus: Verification and Validation

ACG: Automatic code generation



TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

# Verification and Validation: Strategy

## 1. V&V process

(ex. Hierarchical verification, Req. spec)

## 2. Verification with advanced techniques

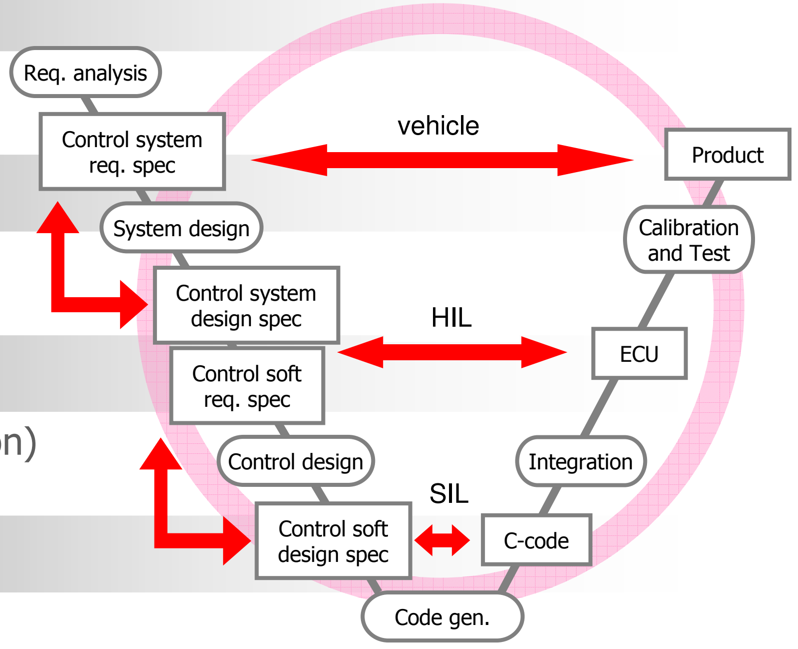
(ex. Automated test generation, Formal methods)

## 3. Efficient Validation including experiment

(ex. Software structural analysis, Defect cause exploration)

## 4. V&V environment

(ex. Integrated V&V environment, Data Base)



Renew development process by applying advanced V&V technologies to improve:
➔ Quality - no defects allowed in product
➔ Efficiency - minimized development cost

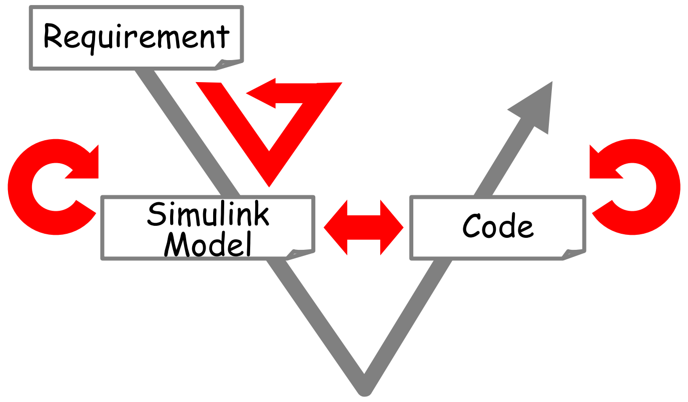
TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

# V & V Application: Model (vs. Code) Coverage

**Technology Goal**

Show satisfied and unsatisfied paths unambiguously

**Motivating Example**



Coverage measurement is required at each confirmation stage.
i.e. Which paths are validated/verified by test cases?

## V & V Application: Test Generation

**Technology Goal**

Structural coverage based test generation

**Motivation**

- %100 MCDC (Modified Condition Decision Coverage) is required for new logic to make sure every part of the code is tested and there is no dead-code.
- Generating test cases for %100 path coverage for main logic of legacy code.
- Equivalence checking of Simulink models and corresponding C-code

## V & V Application: Test Generation

**Current analysis / tool capability**

Commercial test generation tools are available, but they need laborious manual efforts to reach %100 MCDC.

Challenges are:

- %100 MCDC
- %100 path coverage (needed for some logic portions)
- Look-up table coverage
- Lots of nonlinear arithmetic operations
- Logic with counters, timers, integrators

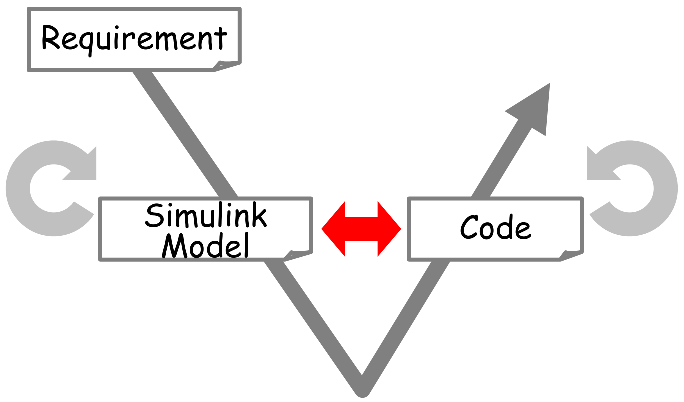
# V & V Research: Symbolic Equivalence Checking

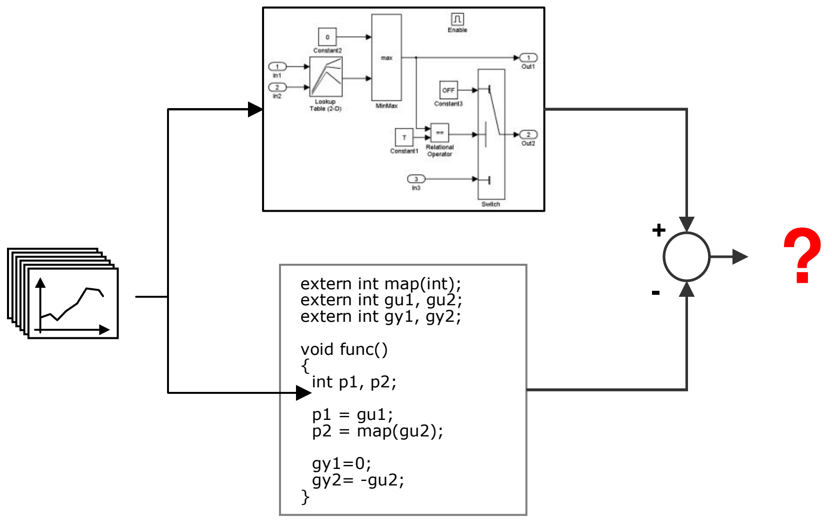
## Technology Goal

Automatic and compositional symbolic equivalence checking of C code against corresponding Simulink models

## Motivating Example

Equivalence Checking of Code against Models





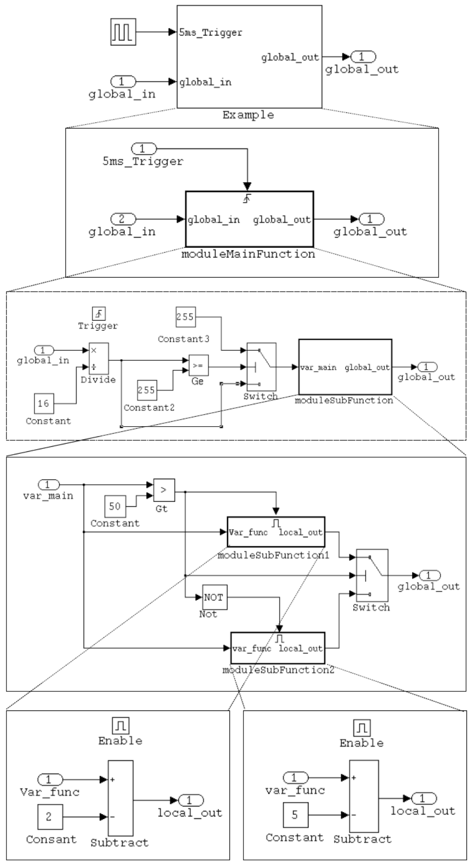
```
extern int map(int);
extern int gu1, gu2;
extern int gy1, gy2;

void func()
{
  int p1, p2;

  p1 = gu1;
  p2 = map(gu2);

  gy1=0;
  gy2= -gu2;
}
```

Ken Butts, TEMA Toyota Technical Center
Powertrain Control – Model based Development

TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

# Equivalence Checking for  Simulink Models and C code

Scenario 1: Code Generation



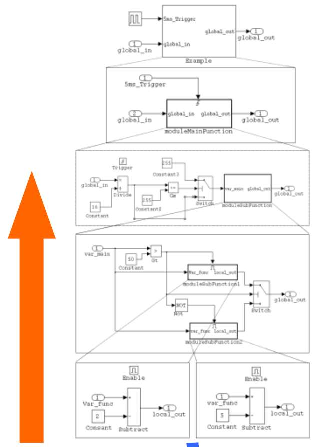
? ≡

```
int global_var;

int moduleSubFunction2(int var_func)
{
  int local_out;
  local_out = var_func - 5;
  return local_out;
}

int moduleSubFunction1(int var_func)
{
  int local_out;
  local_out = var_func - 2;
  return local_out;
}

void moduleSubFunction(int var_main)
{
  int local_in;
  local_in = var_main;
  if (local_in > 50) {
    global_var = moduleSubFunction1(local_in);
  }
  else {
    global_var = moduleSubFunction2(local_in);
  }
}

void moduleMainFunction(void)
{
  int local_main;
  local_main = global_var ≫ 4;
  if (local_main > 255)
    local_main = 255;
  moduleSubFunction(local_main);
}
```

Scenario 2: Legacy Code Modeling

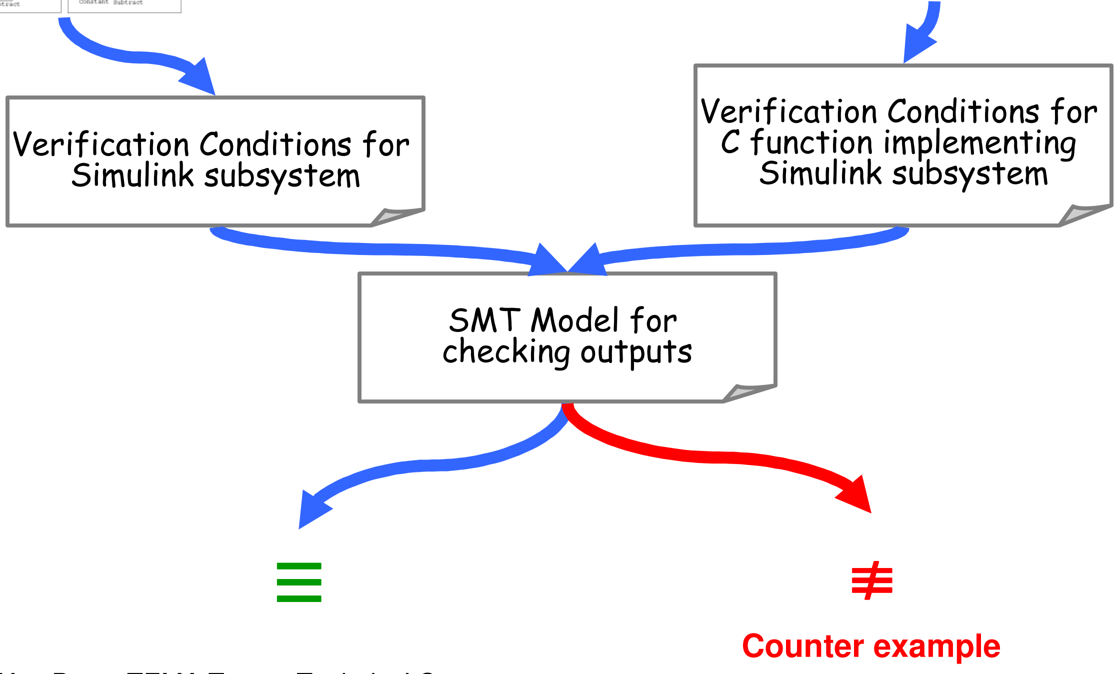
# Compositional Equivalence Checking for Simulink Models and C code (UCLA)

* Saha, Majumdar



```
int global_var;

int moduleSubFunction2(int var_func)
{
  int local_out;
  local_out = var_func - 5;
  return local_out;
}

int moduleSubFunction1(int var_func)
{
  int local_out;
  local_out = var_func - 2;
  return local_out;
}

void moduleSubFunction(int var_main)
{
  int local_in;
  local_in = var_main;
  if (local_in > 50) {
    global_var = moduleSubFunction1(local_in);
  }
  else {
    global_var = moduleSubFunction2(local_in);
  }
}

void moduleMainFunction(void)
{
  int local_main;
  local_main = global_var ≫ 4;
  if (local_main > 255)
    local_main = 255;
  moduleSubFunction(local_main);
}
```

- Process repeated bottom-up on function-call tree
- Function calls in upper hierarchy are treated as un-interpreted functions since they are assumed already verified down the hierarchy


Verification Conditions for Simulink subsystem
Verification Conditions for C function implementing Simulink subsystem
SMT Model for checking outputs
≡
≢
Counter example

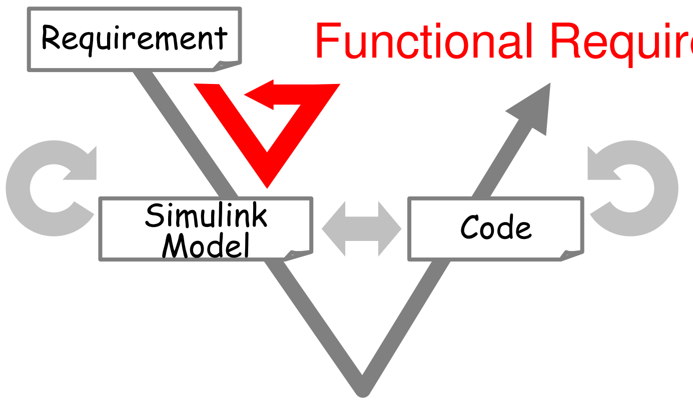
TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

## V & V Research: Control Design Specification Validation
## i.e. Prove model is consistent with requirements

**Technology Goal**

Prove properties using formal methods

**Motivating Example**



Functional Requirement Checking for Models

## V & V Research: Property Proving

**Current analysis / tool capability**

Commercial model-based tools for property proving are available, but very limited in their application:

Technical challenges are…
- Scalability
- Floating-point and Nonlinear mathematics
- Look-up tables
- Logic with counters, timers, integrators

## V & V Research: Real-time Software Checking

Technology Goal

Improve engine control software real-time characteristic robustness to software changes (Chrona TDL)

➔

- Real-time software evaluation with Software-in-the-Loop (SIL) (Chrona Validator)

➔

- Real-time execution time estimation (CHESS GameTime)

➔

- “Predictable” computing (future work ?)
  - Today’s computer architects are not considering real-time control !

## V & V Research: Chrona TDL

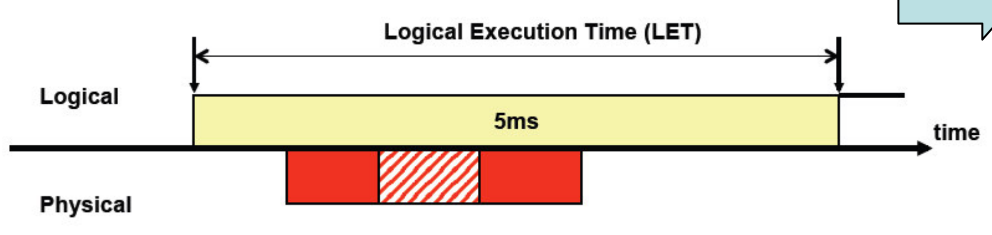
TDL: Timing Definition Language *Pree, Resmerita

Allocate generous computation time budgets (LETs) and use a run-time machine to enforce the timing specification

- Closely related to schedulability analysis
- Requires software program analysis to evaluate variable interactions
- Requires fine grain execution time data

Engine control requires extensions for event-based processes

**Logical Execution Time**



- All inputs are read at the beginning of the LET
- All outputs are updated at the end of the LET
- LET is platform-independent => platform independent behavior
- Schedulability condition: LET ≥ Worst Case Reaction Time of the time-triggered task

Increased robustness of the software: the time profile of a global variable will **not** change as a result of

- Adding/Removing functions in the system (both in the time-triggered and in the event triggered parts)
- Variation of execution times in different runs of the same system due to different dynamic conditions
- Changes of execution times due to porting the software to another ECU

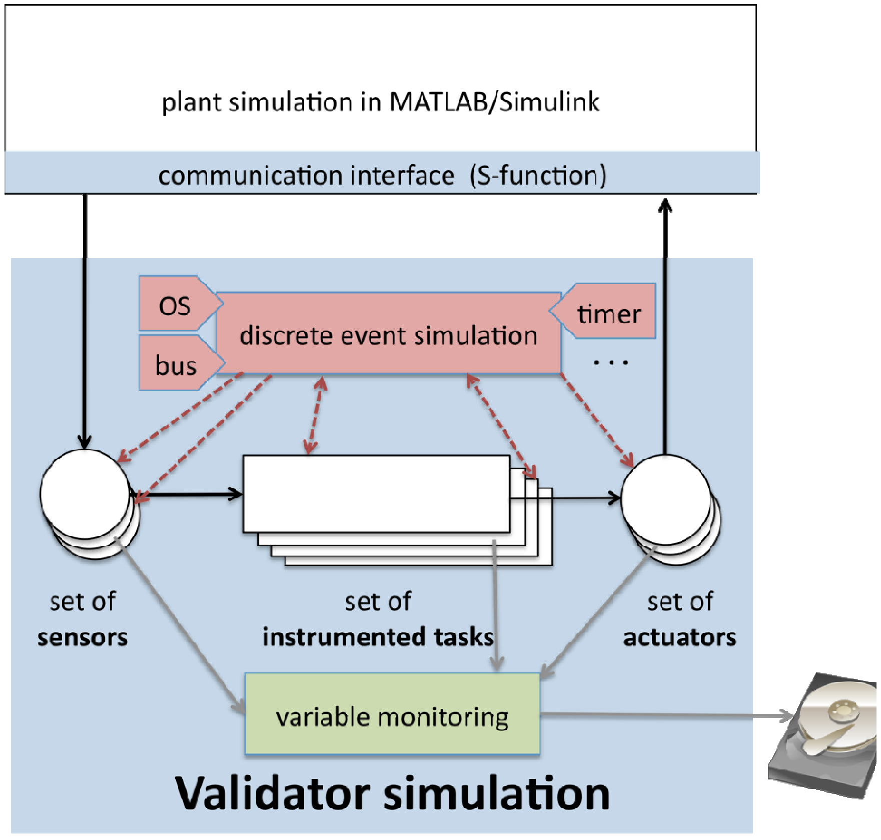
# V & V Research: Chrona Validator

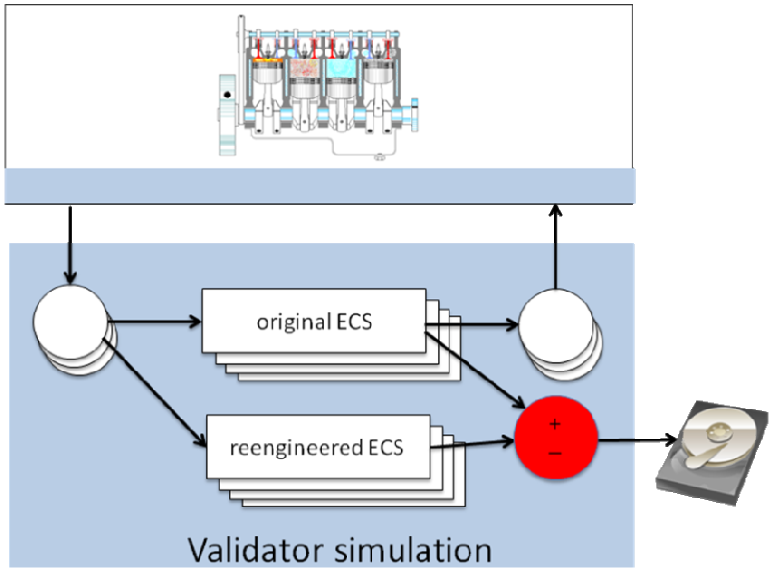
Technology Goal

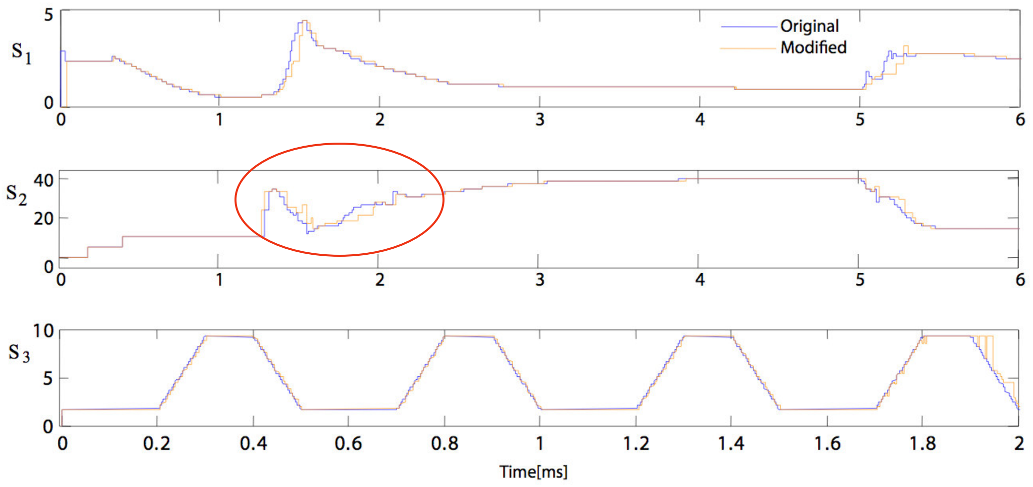
*Resmerita et. al.

How to check Modified TDL versus Original ? → Real-time aware SIL

– Pre-emption, scheduling, jitter, (reverse) source level debugging


plant simulation in MATLAB/Simulink
communication interface (S-function)
OS
bus
discrete event simulation
timer
. . .
set of sensors
set of instrumented tasks
set of actuators
variable monitoring
Validator simulation





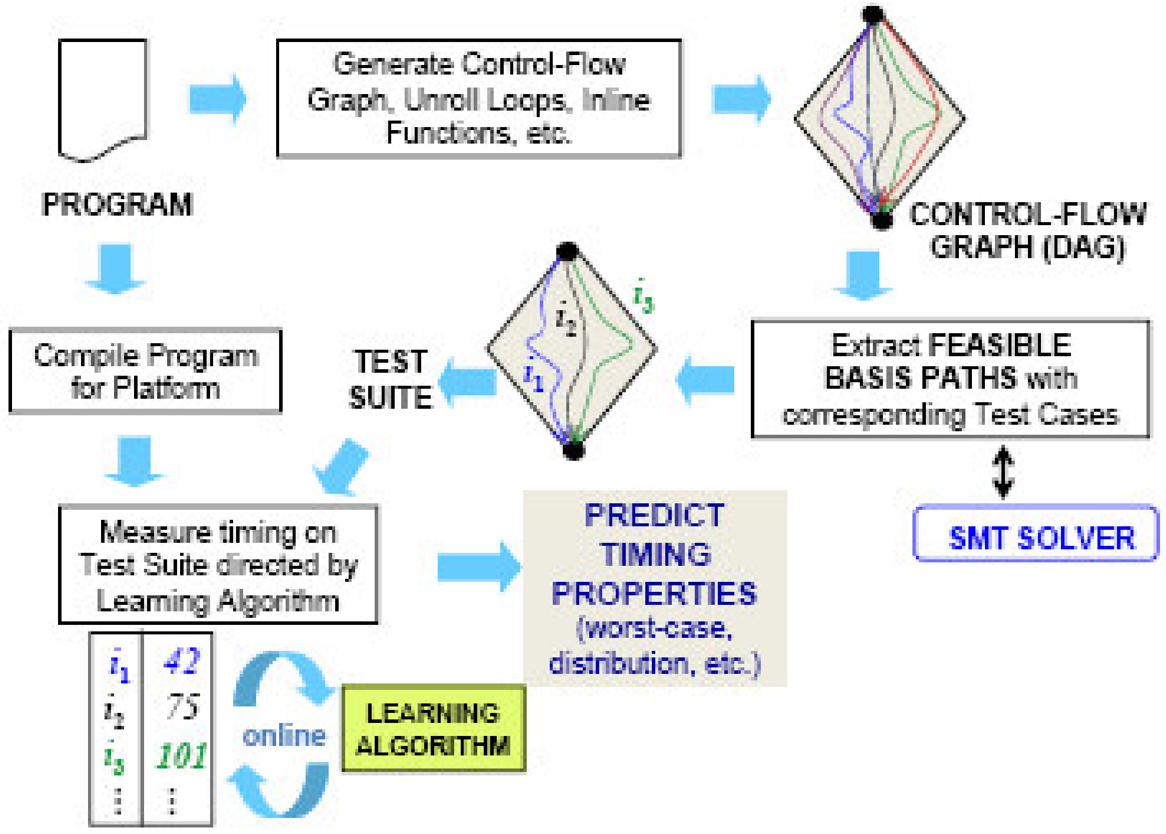
# V & V Research: GameTime in CHESS

*Seshia

Technology Goal

How to get good fine grain execution time estimates for TDL and Validator ?

- Resolve worst-case path and initial state dimensions (and their interactions)
- Measure ‘ feasible basis’ paths and use data to generate a model for all paths



PROGRAM
Generate Control-Flow Graph, Unroll Loops, Inline Functions, etc.
CONTROL-FLOW GRAPH (DAG)
Compile Program for Platform
TEST SUITE
$i_1$
$i_2$
$i_3$
Extract FEASIBLE BASIS PATHS with corresponding Test Cases
SMT SOLVER
Measure timing on Test Suite directed by Learning Algorithm
PREDICT TIMING PROPERTIES (worst-case, distribution, etc.)
$i_1$ 42
$i_2$ 75
$i_3$ 101
online
LEARNING ALGORITHM

TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

# V & V Research: Verifying Complex Systems

(Complex == Cyber Physical Systems) (Carnegie Mellon University)

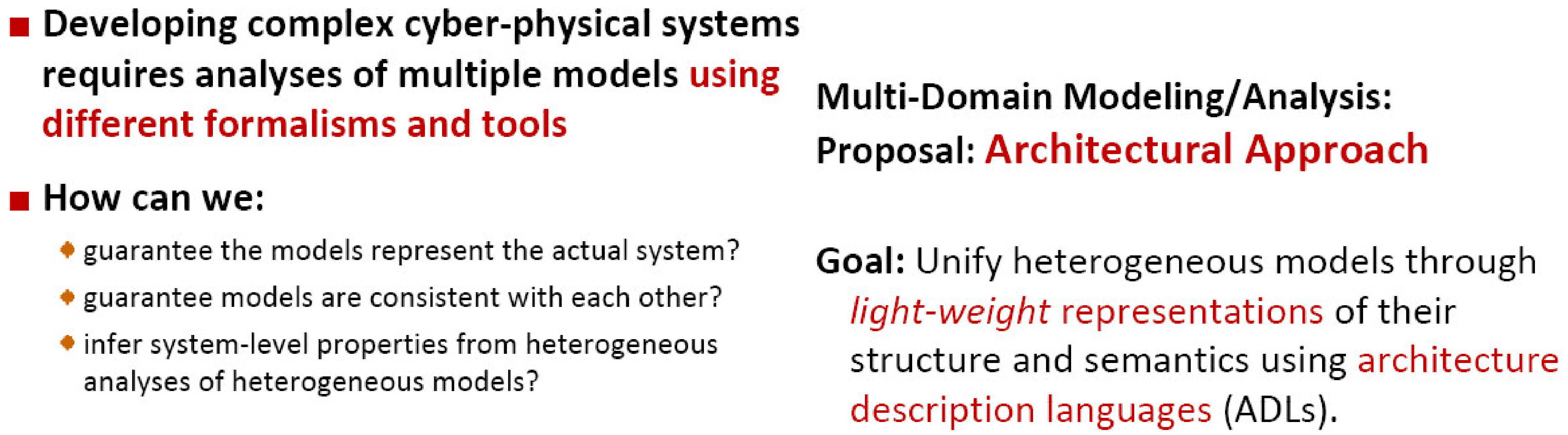
*Krogh, Bhave

**Technology Goal**

Learn how to apply recent V&V results to complex problems

- Architectural and Formal approaches
- Many advancements since MoBIES (circa 2002)

- Funding by the National Science Foundation
- Collision Avoidance application area



- **Developing complex cyber-physical systems requires analyses of multiple models using different formalisms and tools**
- **How can we:**
  - guarantee the models represent the actual system?
  - guarantee models are consistent with each other?
  - infer system-level properties from heterogeneous analyses of heterogeneous models?

**Multi-Domain Modeling/Analysis:**

**Proposal: Architectural Approach**

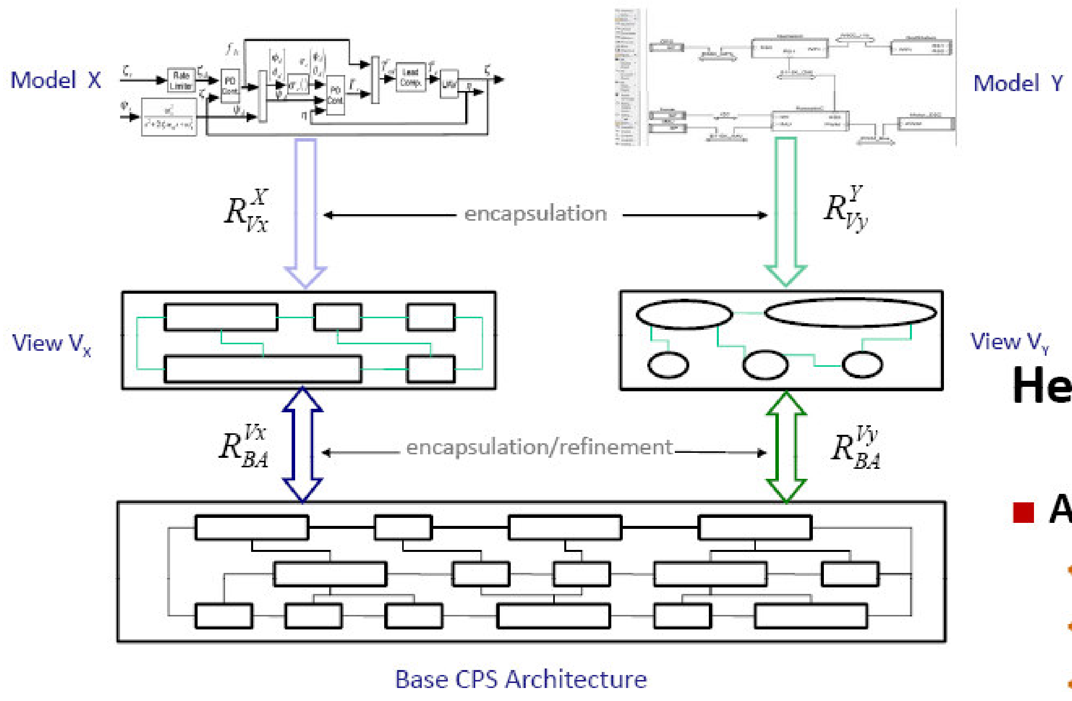
**Goal:** Unify heterogeneous models through *light-weight* representations of their structure and semantics using architecture description languages (ADLs).

## V & V Research: Verifying Complex Systems

(Complex == Cyber Physical Systems) (Carnegie Mellon University)

*Garlan, Krogh, Bhave

### Models as Architectural Views



### Heterogeneous Verification

- **Annotate architectures with**
  - system-level specifications/requirements
  - assumptions underlying models/views
  - guarantees provided by model-based analyses
- **Develop algorithms for**
  - consistency analysis for specifications & assumptions
  - integration of model-based verification results
  - coverage via heterogeneous verification activities

## V & V Research: Verifying Complex Systems

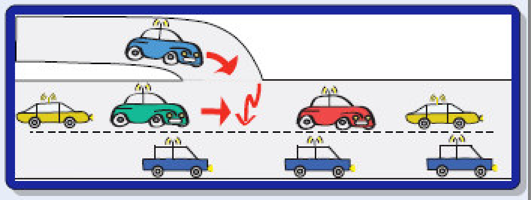
(Complex == Cyber Physical Systems) (Carnegie Mellon University)

Q: I want to verify lots of moving cars A: Distributed hybrid systems Q: How?

*Platzer, Loos

**Challenge (Distributed Hybrid Systems)**

- Continuous dynamics (differential equations)
- Discrete dynamics (control decisions)
- Structural dynamics (remote communication)
- Dimensional dynamics (appearance)



**Quantified Differential Dynamic Logic Qd$\mathcal{L}$:**

**Theorem (Relative Completeness)**

*Qd$\mathcal{L}$ verification sound & complete axiomatisation of distributed hybrid systems relative to quantified differential equations.* ‣ Proof 16p.

**Corollary (Proof-theoretical Alignment)**

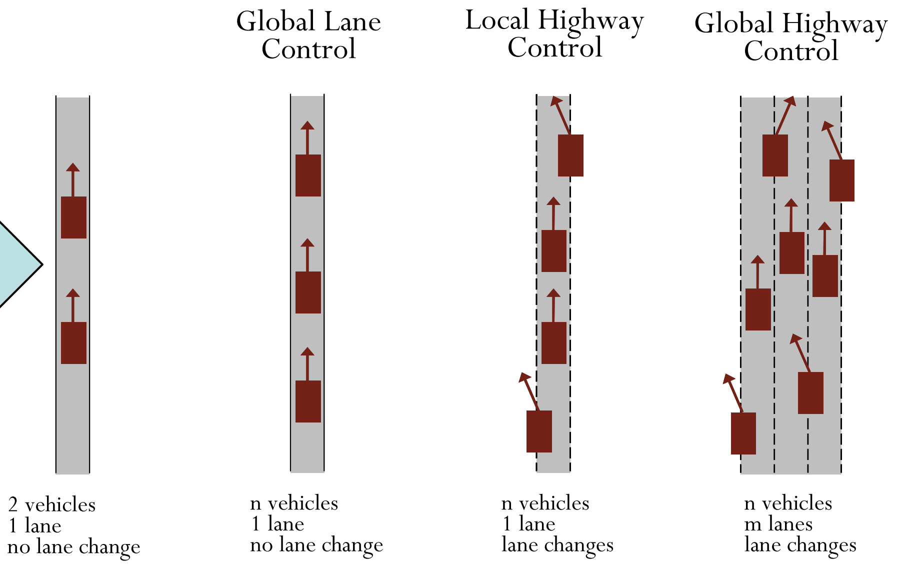
proving distributed hybrid systems = proving dynamical systems!

**Corollary (Yes, we can!)**

distributed hybrid systems can be verified by recursive decomposition

Adaptive Cruise Control:
Hybrid, Dynamic, and Now Formally Verified

### Car Control: Proof Sketch

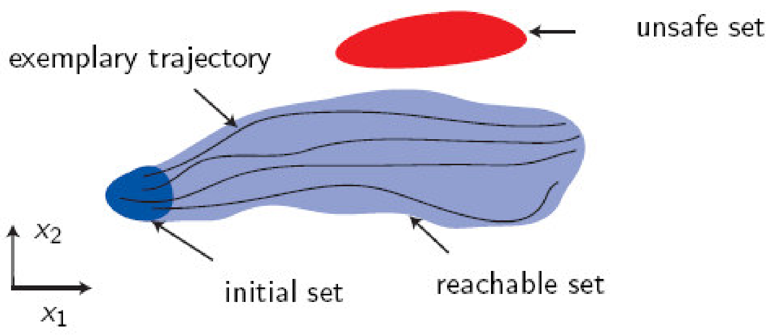


Abstract modeling and formal verification yield control requirements to be satisfied during design and implementation

# V & V Research: Verifying Complex Systems

(Complex == Cyber Physical Systems) (Carnegie Mellon University)

## Safety Verification Using Reachable Sets



- System is safe, if no trajectory enters the unsafe set.

## Nonlinear Sytems with Uncertain Parameters

$$\dot{x} = f(x(t), u(t), p(t)),$$

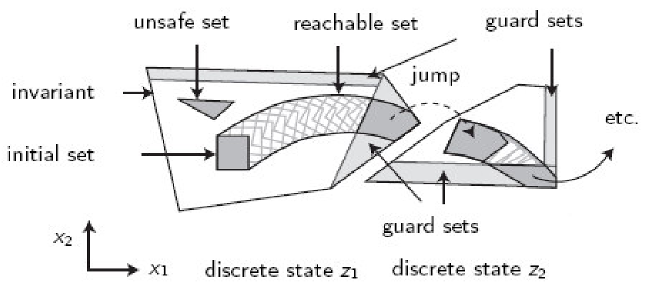
$$x(0) \in \mathcal{X}_0 \subset \mathbb{R}^n, \quad u(t) \in \mathcal{U} \subset \mathbb{R}^m, \quad p(t) \in \mathcal{P} \subset \mathbb{R}^p$$

- Approach is based on linearizing the system dynamics while adding the linearization errors as an additional uncertain input.
- Scalable when using zonotopes, as long as no splitting is involved. For a water tank system [redacted] can be computed in a few minutes.

## Hybrid Sytems

*Althoff

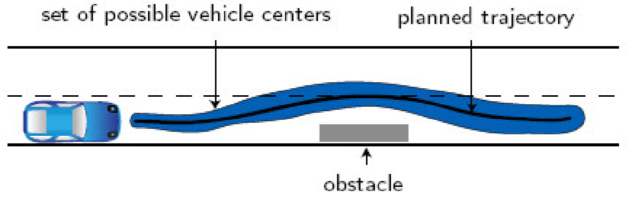
Graphical Description:



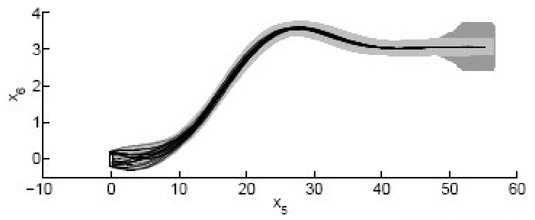
- In addition to continuous systems, the intersection with guard sets is required (seems simple, but it's not).
- Not really scalable; usually [redacted].

## Nonlinear Sytems with Uncertain Parameters

Example: Evasive maneuver of a car.
sketch:



computed set for a lane change maneuver:

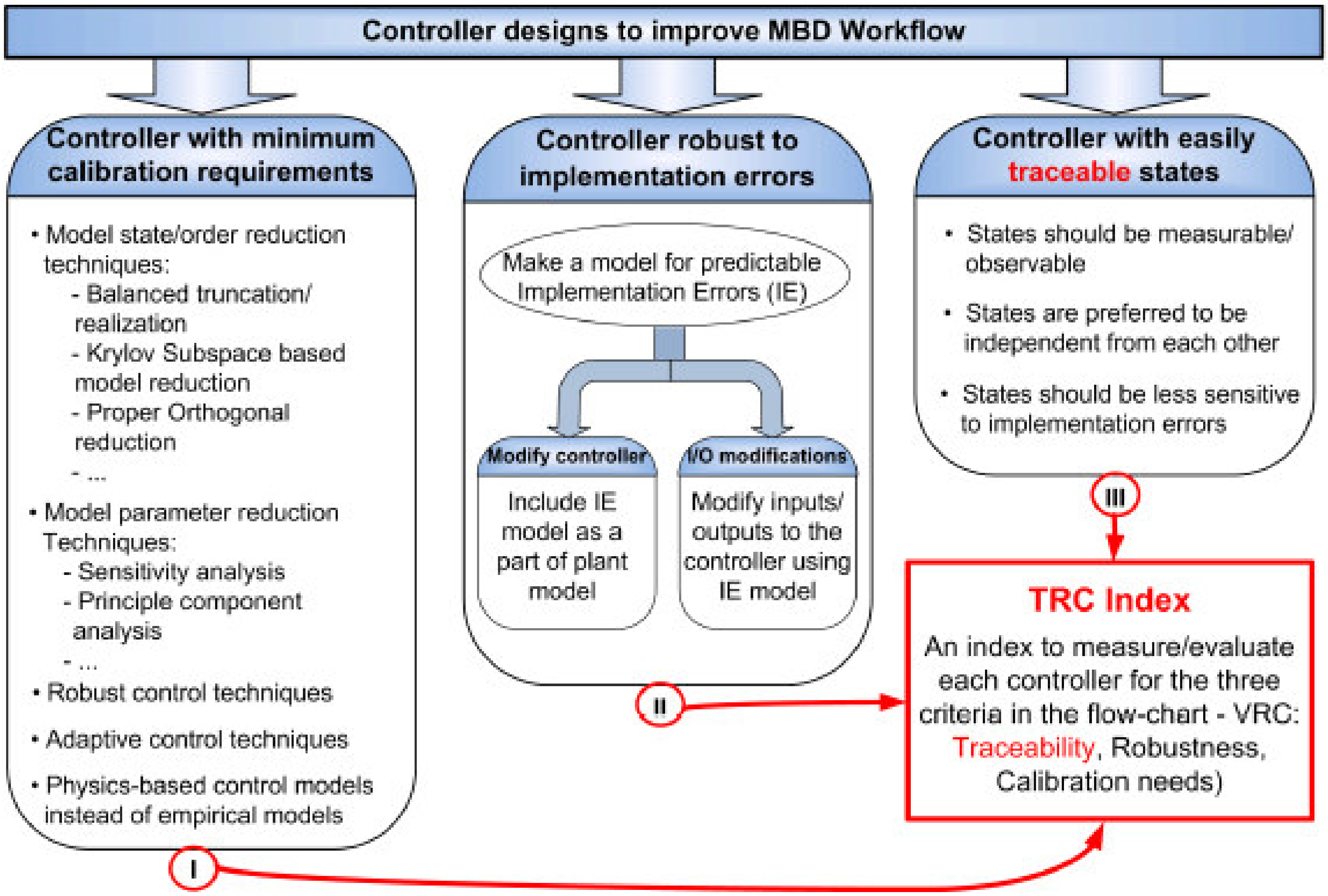


## V & V Research: Verifiable Control Design in CHESS

*Hedrick, Shahbakhti

Technology Goal

How can we consider verification during control design ?


Controller designs to improve MBD Workflow
Controller with minimum calibration requirements
• Model state/order reduction techniques:
- Balanced truncation/ realization
- Krylov Subspace based model reduction
- Proper Orthogonal reduction
- ...
• Model parameter reduction Techniques:
- Sensitivity analysis
- Principle component analysis
- ...
• Robust control techniques
• Adaptive control techniques
• Physics-based control models instead of empirical models
Controller robust to implementation errors
Make a model for predictable Implementation Errors (IE)
Modify controller
Include IE model as a part of plant model
I/O modifications
Modify inputs/ outputs to the controller using IE model
Controller with easily traceable states
• States should be measurable/ observable
• States are preferred to be independent from each other
• States should be less sensitive to implementation errors
I
II
III
TRC Index
An index to measure/evaluate each controller for the three criteria in the flow-chart - VRC: Traceability, Robustness, Calibration needs)

TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

## Model-based V & V: Situation Summary

What we think we can do (soon):

Validation (design confirmation)

- Closed-loop simulation
  - Property assertions
  - Robustness to parameter and scenario variation
- Rapid prototyping

Verification (implementation confirmation)

- Structural
  - Static analysis
  - Test vector generation
  - Visualization and analysis
- Software-in-the-Loop
  - Code to Model Equivalence checking
  - Functional scenarios
  - Fixed point design
- Hardware-in-the-Loop
  - ECU interface and signal conditioning
  - Real-time software confirmation
  - Fault diagnosis and handling

## Model-based V & V: Situation Summary

### What we (will) need:

A *systematic engineering process* to fully explore the operating space

- Deal with system complexity
  - Heterogeneity (hybrid dynamics, wireless networking, dynamic agent scenarios)
  - Hierarchical structure (in-vehicle, vehicle-to-vehicle, vehicle to infrastructure)
  - Scale
  - Floating point design vs.  fixed-point Implementation
- Leverage compositionality / prior knowledge
  - Component to system verification
  - Multiple component providers
  - Control design attributes
- Security – threatening / malicious agents
- Real-time software performance guarantees – predictable computing
- Calibration – accommodate tuning changes at the end of the process.

Thank you for your attention !

We’re trying to hire a researcher to help.