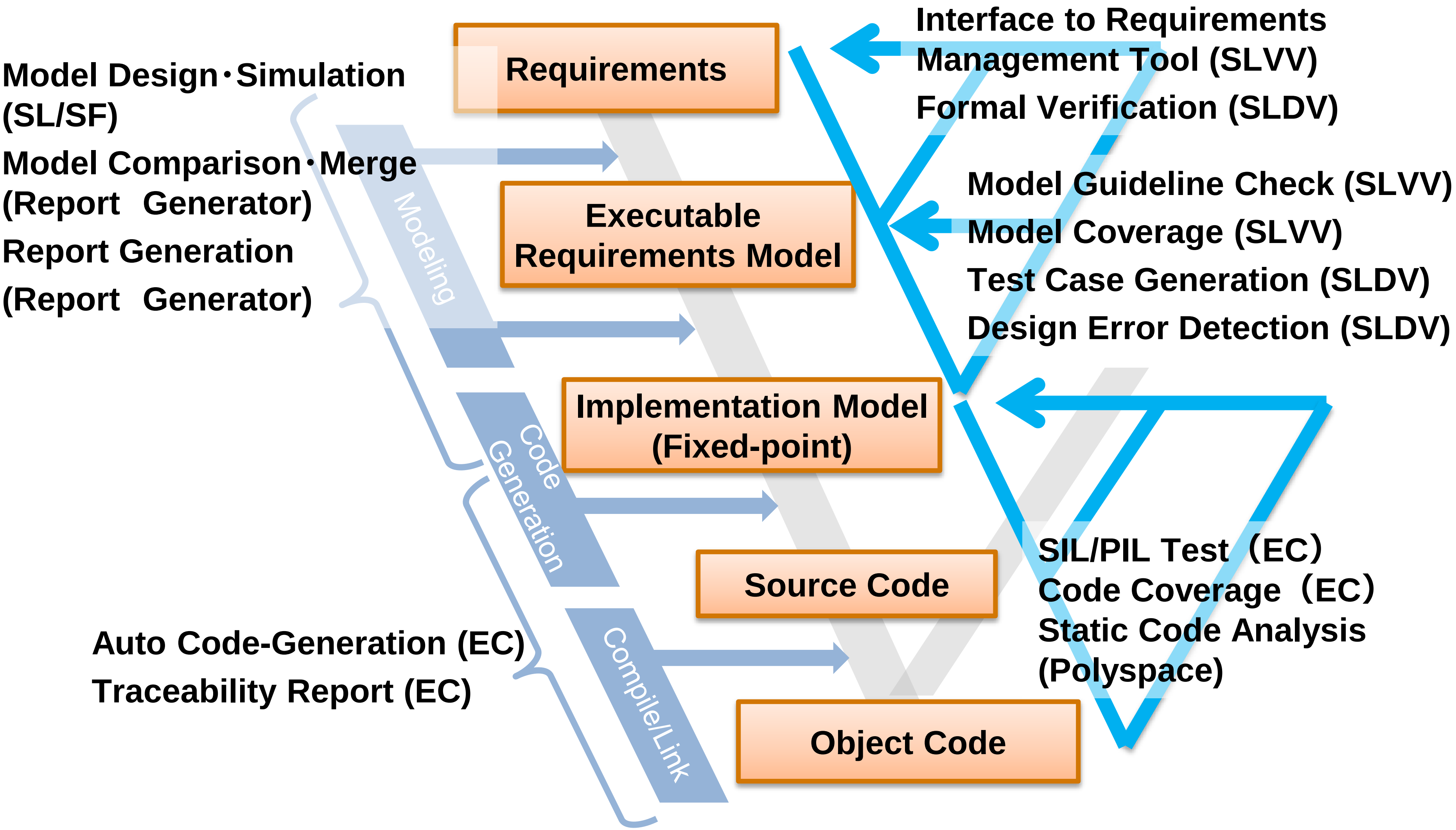
Verification and Validation Solutions for High Integrity Systems

Tiffany Liang
Application Engineer
MathWorks

Recommended Workflow

Detecting errors early in the development cycle



Model Design・Simulation (SL/SF)

Model Comparison・Merge (Report Generator)

Report Generation (Report Generator)

Requirements

Executable Requirements Model

Implementation Model (Fixed-point)

Source Code

Object Code

Interface to Requirements Management Tool (SLVV)

Formal Verification (SLDV)

Model Guideline Check (SLVV)

Model Coverage (SLVV)

Test Case Generation (SLDV)

Design Error Detection (SLDV)

SIL/PIL Test（EC）

Code Coverage（EC）

Static Code Analysis (Polyspace)

Auto Code-Generation (EC)

Traceability Report (EC)

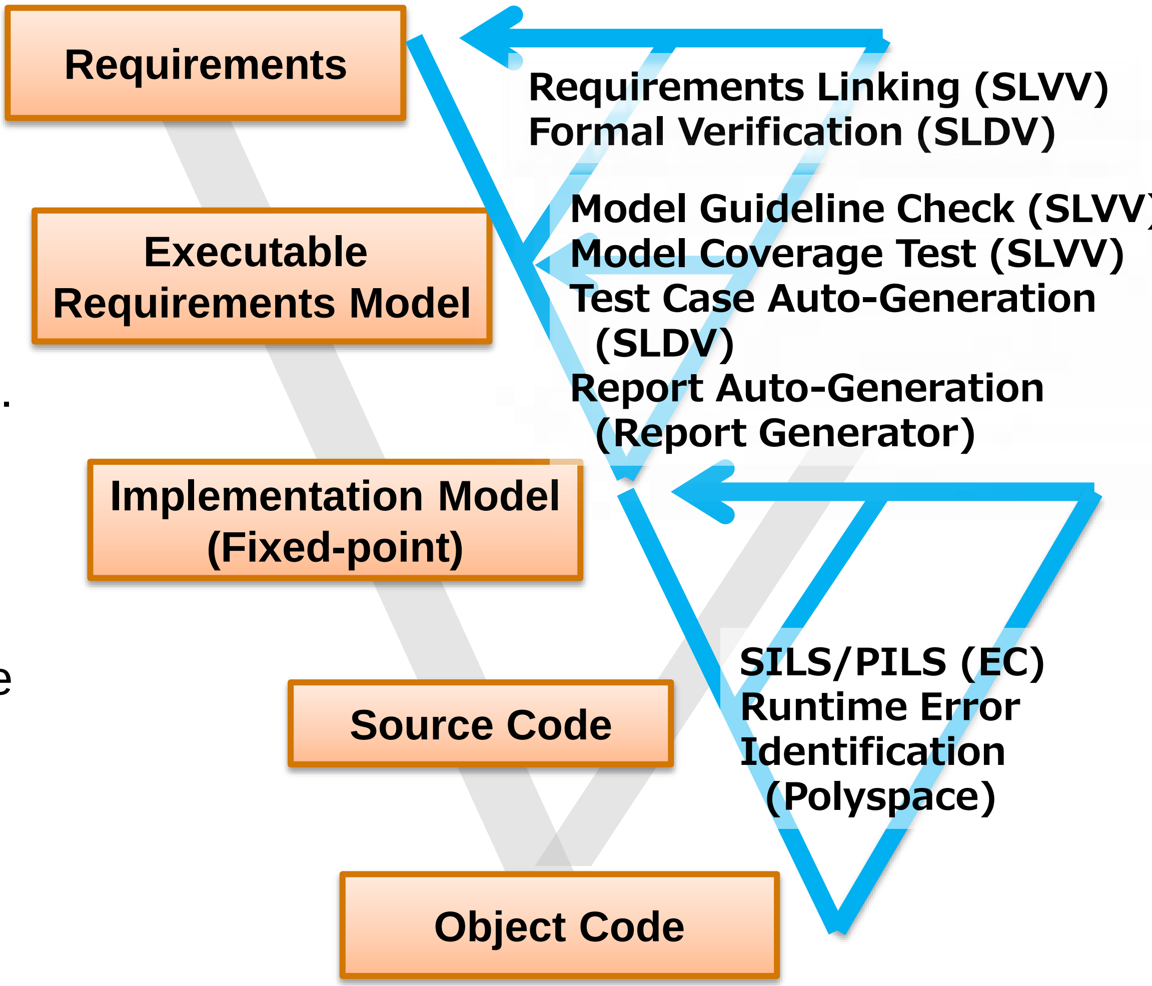
Configuration Management (Simulink Project)

MathWorks benefits

Early verification and Validation

・Able to form small V-loops
・Able to detect errors early in the development cycle

- Model ⇔ Code consistency allows for Simulink simulation results to be considered “truth”.
- Early model verification is possible due to the ability to investigate floating-point models
- Large team development made easy through highly customizable tool chain
- Errors in object code detected easily through synchronization between simulations and SILS/PILS


Requirements
Executable Requirements Model
Implementation Model (Fixed-point)
Source Code
Object Code

Requirements Linking (SLVV)
Formal Verification (SLDV)

Model Guideline Check (SLVV
Model Coverage Test (SLVV)
Test Case Auto-Generation (SLDV)
Report Auto-Generation (Report Generator)

SILS/PILS (EC)
Runtime Error Identification (Polyspace)

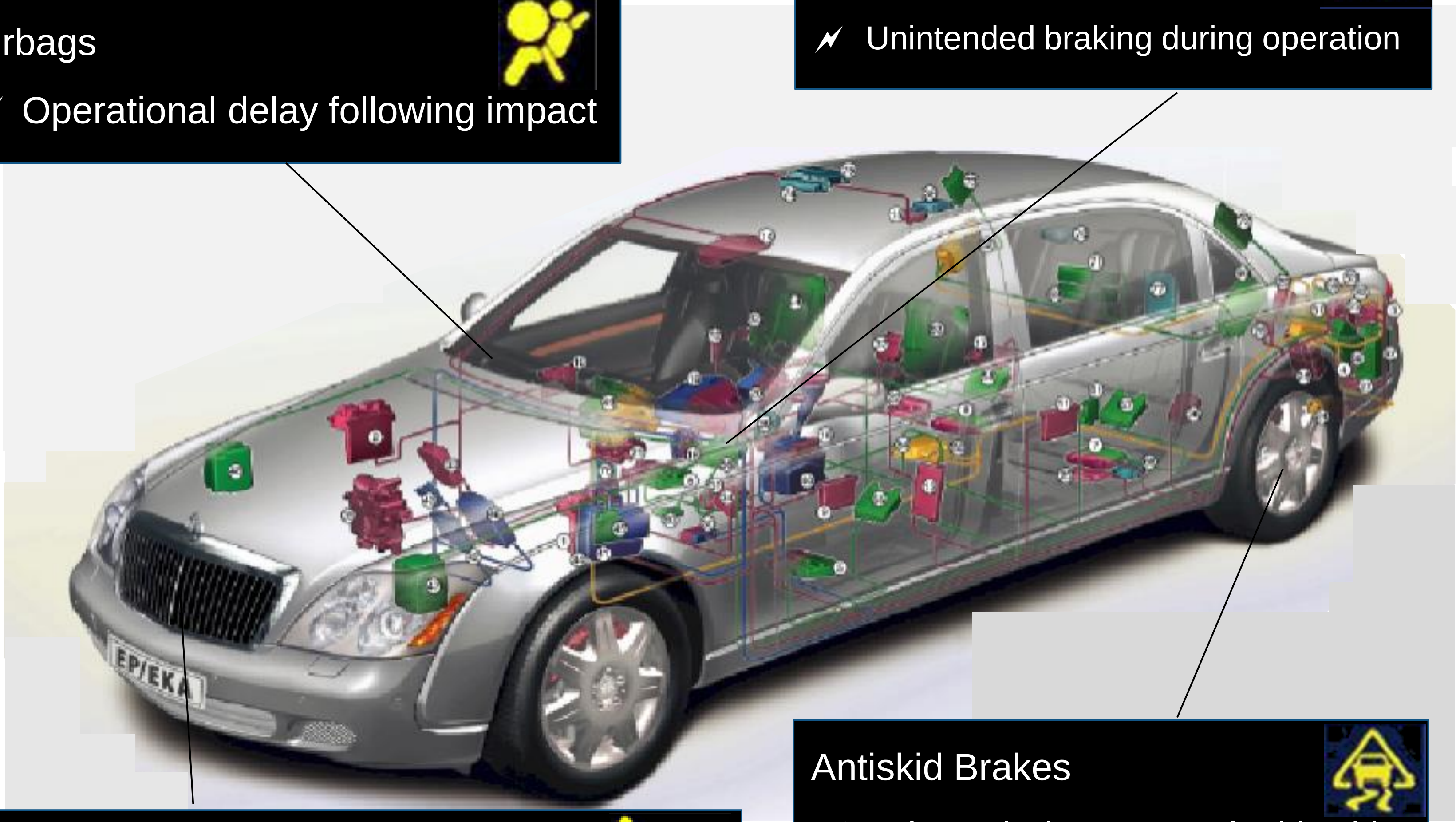
Examples of High Reliability Applications

Electronic Parking Brake

↯ Unintended braking during operation

Airbags

↯ Operational delay following impact



Vehicle-to-vehicle distance control

↯ Insufficient deceleration within required time

Antiskid Brakes

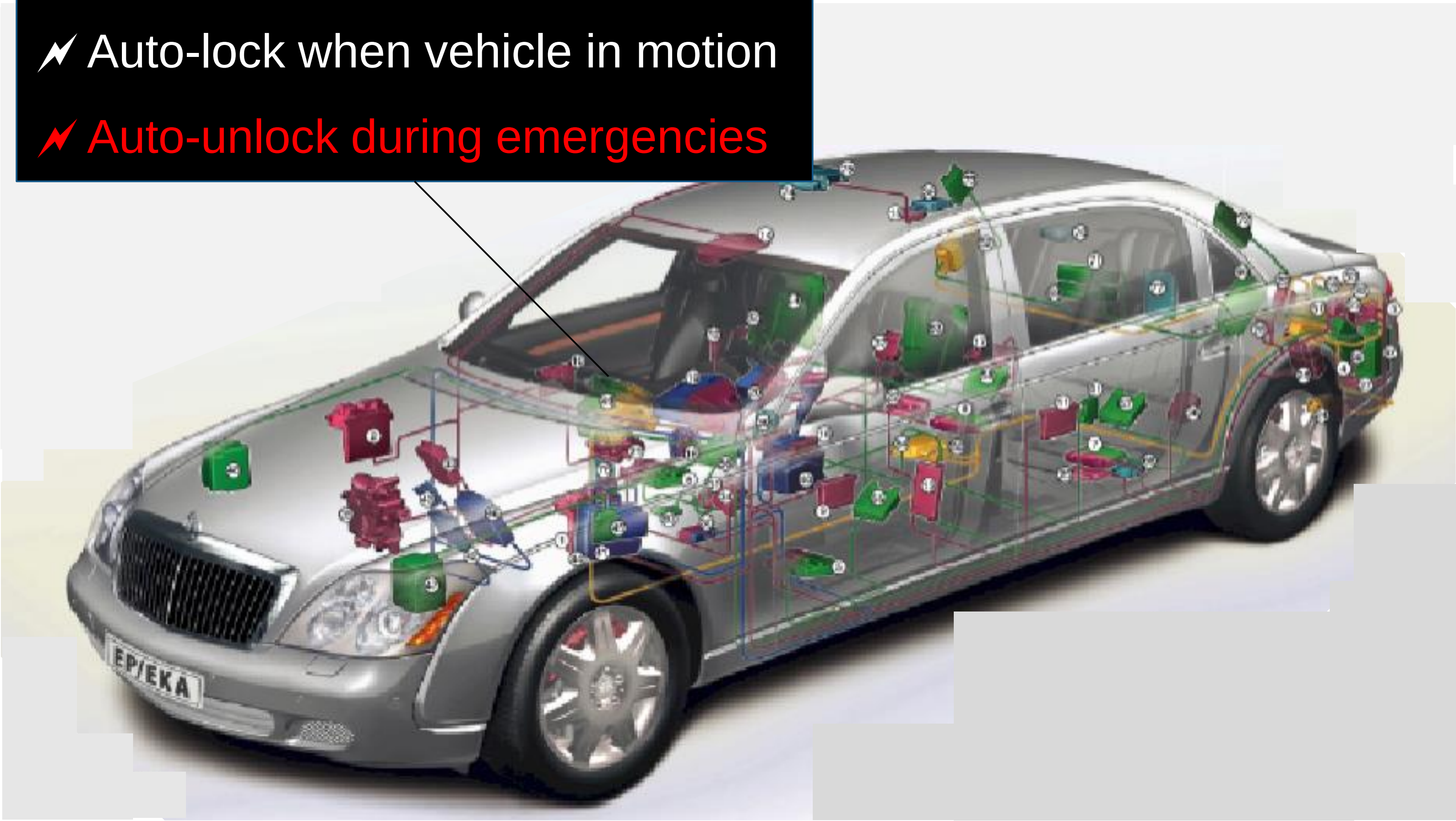
↯ Unintended asymmetrical braking

Source: K. Grimm: Software-Technologie in der Automoboilindustrie. 19. STEV-OESTERREICH-FACHTAGUNG IT-/Softwarequalitätsmanagment in der Praxis, Wien, Austria, May 2004

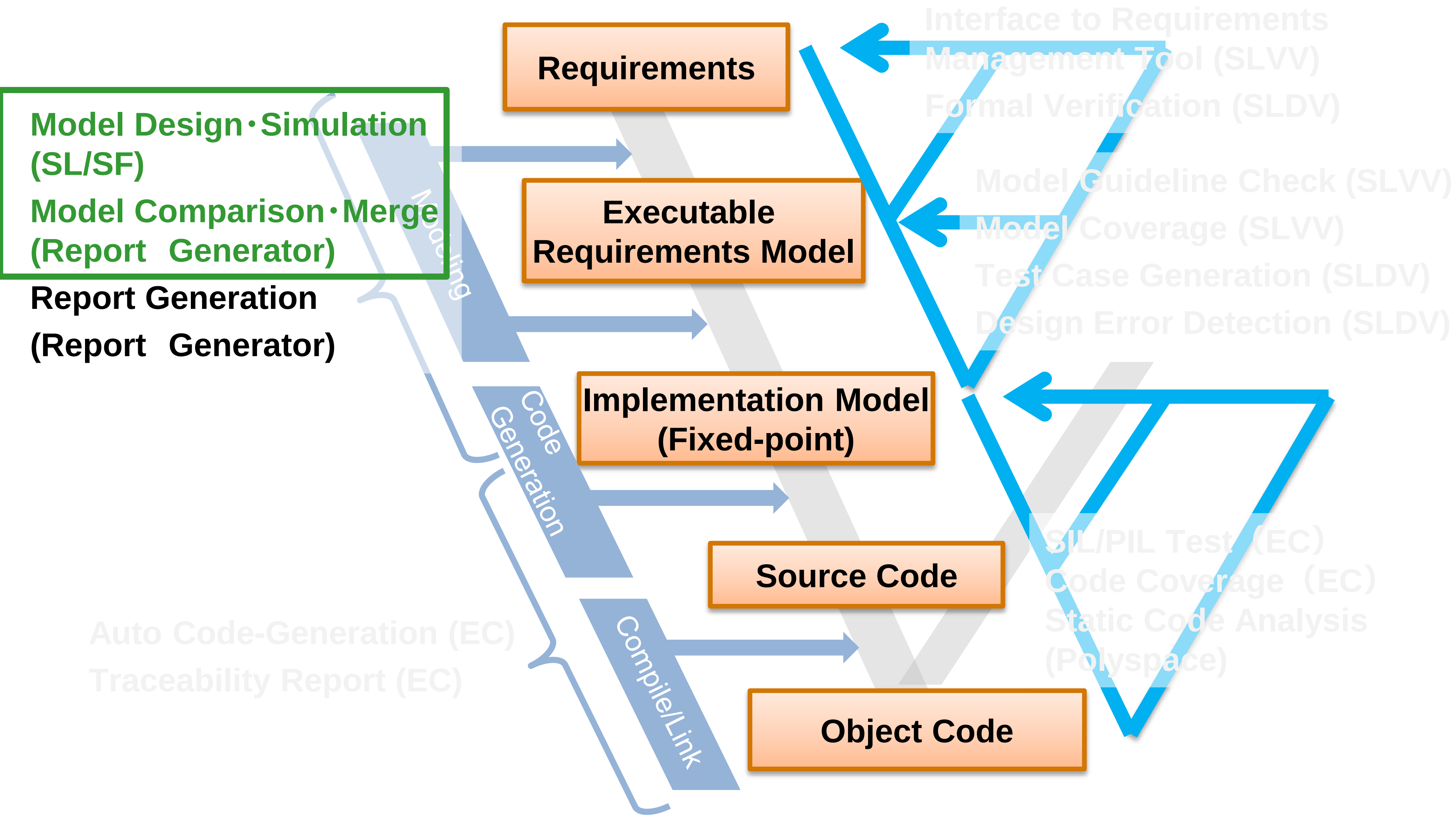
Example: Door Lock Control System

Door Lock Control

- Auto-lock when vehicle in motion
- Auto-unlock during emergencies



Our First Topic



Model Design・Simulation (SL/SF)

Model Comparison・Merge (Report Generator)

Report Generation (Report Generator)

Interface to Requirements Management Tool (SLVV)

Formal Verification (SLDV)

Model Guideline Check (SLVV)

Model Coverage (SLVV)

Test Case Generation (SLDV)

Design Error Detection (SLDV)

SIL/PIL Test (EC)

Code Coverage (EC)

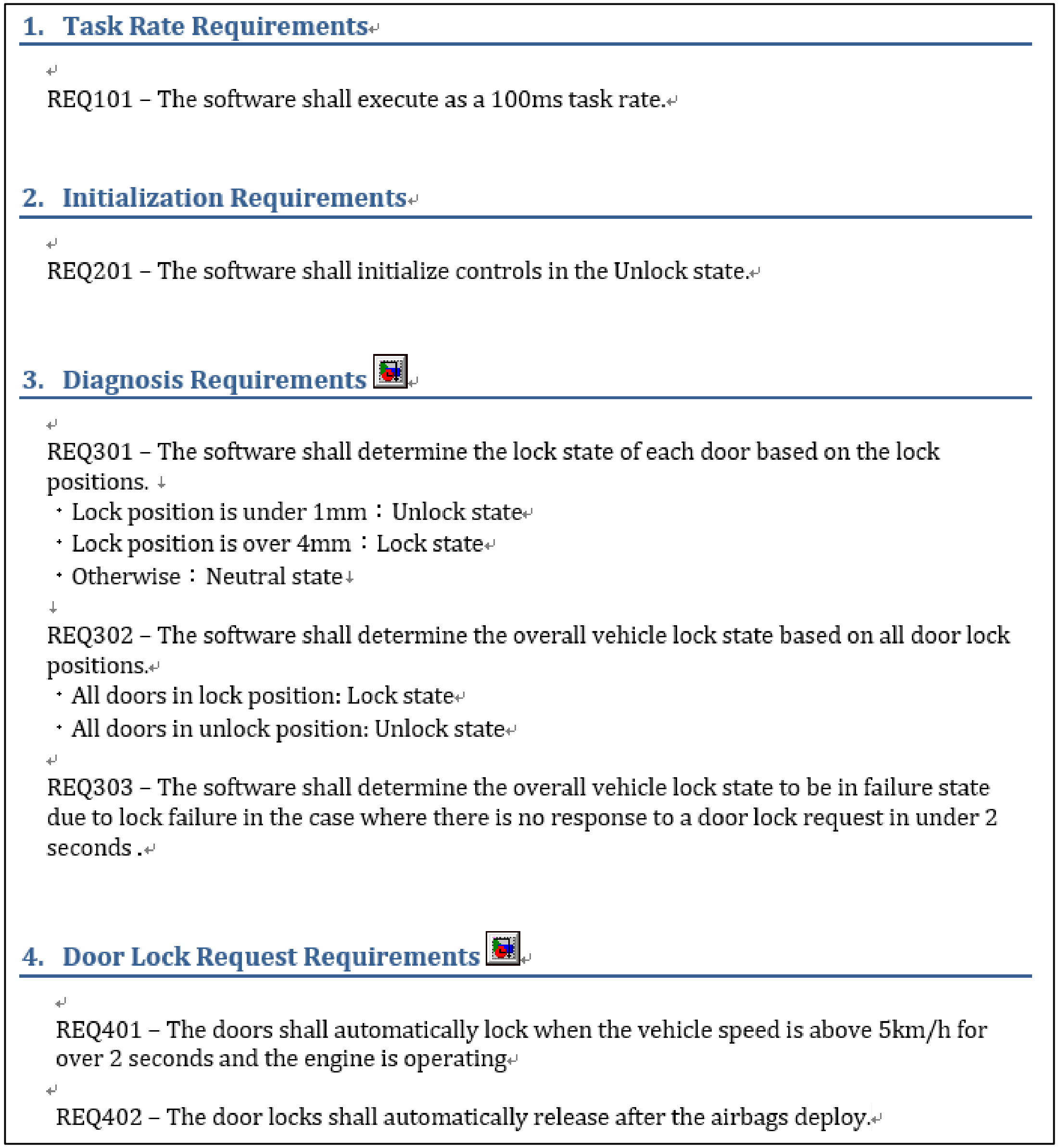
Static Code Analysis (Polyspace)

Auto Code-Generation (EC)

Traceability Report (EC)

Configuration Management (Simulink Project)

Door Lock Control Software Requirements



1. Task Rate Requirements

REQ101 – The software shall execute as a 100ms task rate.

2. Initialization Requirements

REQ201 – The software shall initialize controls in the Unlock state.

3. Diagnosis Requirements

REQ301 – The software shall determine the lock state of each door based on the lock positions.

- Lock position is under 1mm ： Unlock state
- Lock position is over 4mm ： Lock state
- Otherwise ： Neutral state

REQ302 – The software shall determine the overall vehicle lock state based on all door lock positions.

- All doors in lock position: Lock state
- All doors in unlock position: Unlock state

REQ303 – The software shall determine the overall vehicle lock state to be in failure state due to lock failure in the case where there is no response to a door lock request in under 2 seconds .

4. Door Lock Request Requirements

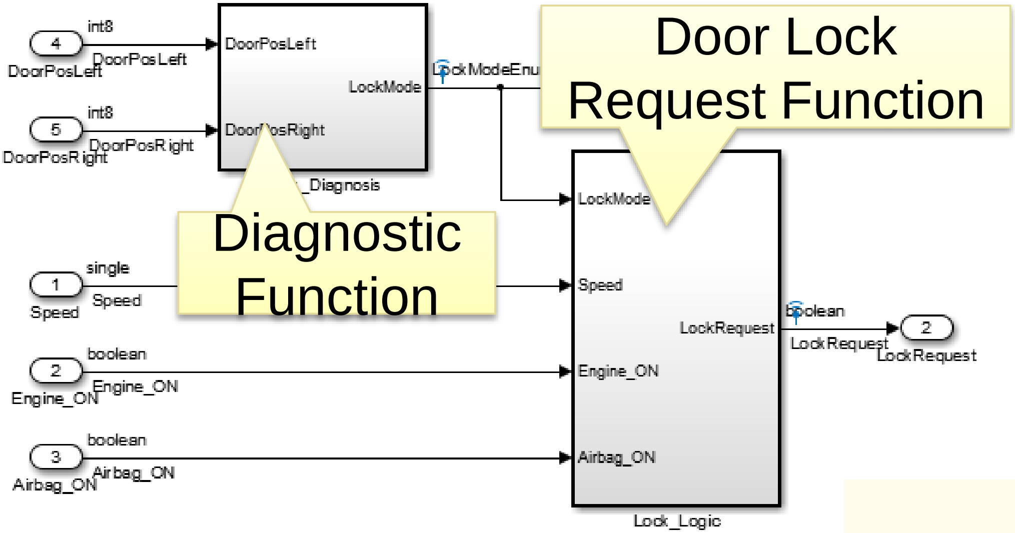
REQ401 – The doors shall automatically lock when the vehicle speed is above 5km/h for over 2 seconds and the engine is operating

REQ402 – The door locks shall automatically release after the airbags deploy.

Door Lock Model

Simulink / Stateflow

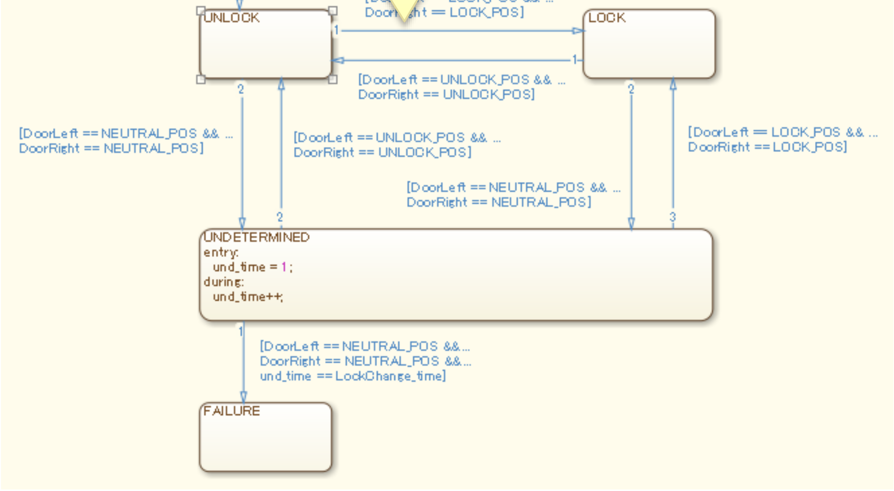
Increased Readability / Productivity through Graphical Modeling



Door Lock Request Function

Diagnostic Function

Diagnostic Function State Transition Diagram

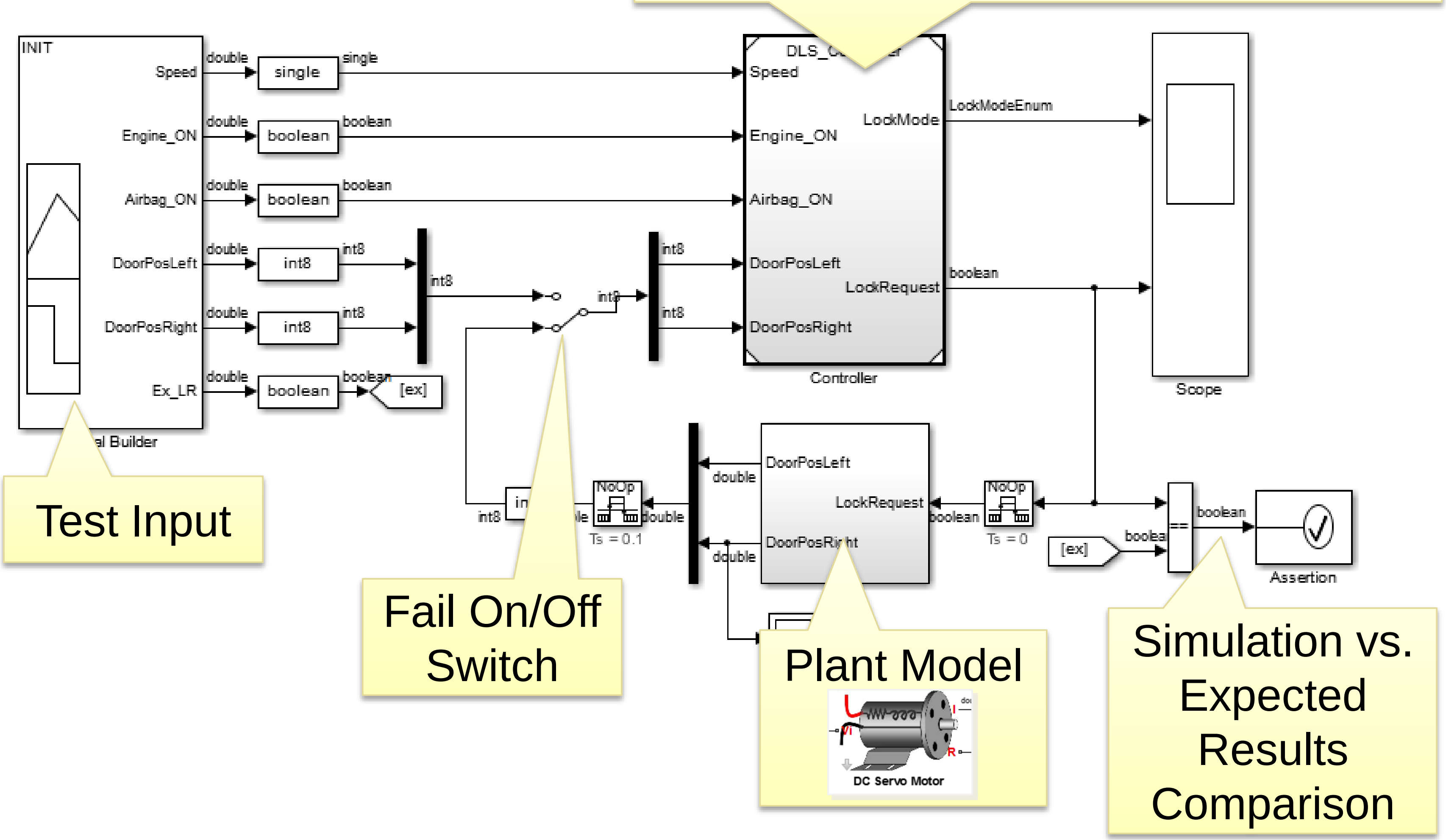


UNLOCK
LOCK
[Do... == LOCK_POS && ...
Door...ht == LOCK_POS]
1
1
[DoorLeft == UNLOCK_POS && ...
DoorRight == UNLOCK_POS]
2
[DoorLeft == NEUTRAL_POS && ...
DoorRight == NEUTRAL_POS]
[DoorLeft == UNLOCK_POS && ...
DoorRight == UNLOCK_POS]
2
[DoorLeft == LOCK_POS && ...
DoorRight == LOCK_POS]
[DoorLeft == NEUTRAL_POS && ...
DoorRight == NEUTRAL_POS]
2
3
UNDETERMINED
entry:
und_time = 1;
during:
und_time++;
1
[DoorLeft == NEUTRAL_POS && ...
DoorRight == NEUTRAL_POS && ...
und_time == LockChange_time]
FAILURE

Door Lock Test Model

Simulink / Simscape

Able to execute various tests using the control model



Model Block used to call control model

Test Input

Fail On/Off Switch

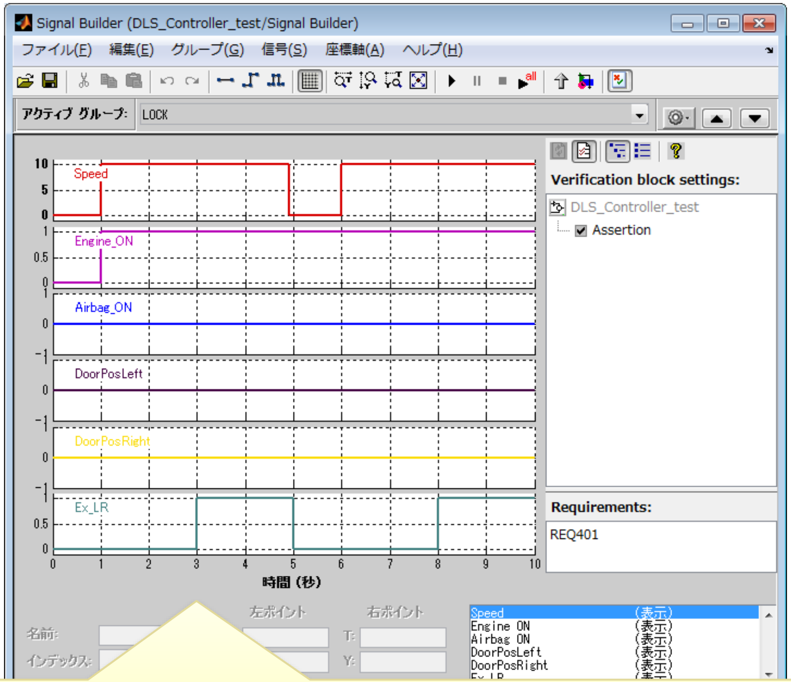
Plant Model

Simulation vs. Expected Results Comparison

Requirements & Logic Testing through Simulation

Simulink / Stateflow

- Early verification of entire system incl. plant behavior
- Investigation of failure/anomaly modes (difficult on H/W)



Signal Builder (DLS_Controller_test/Signal Builder)
ファイル(F) 編集(E) グループ(G) 信号(S) 座標軸(A) ヘルプ(H)
アクティブ グループ: LOCK
Speed
Engine_ON
Airbag_ON
DoorPosLeft
DoorPosRight
Ex_LR
10
5
0
1
0.5
-1
0 1 2 3 4 5 6 7 8 9 10
時間 (秒)
左ポイント
右ポイント
名前:
インデックス:
T:
Y:

Verification block settings:

DLS_Controller_test

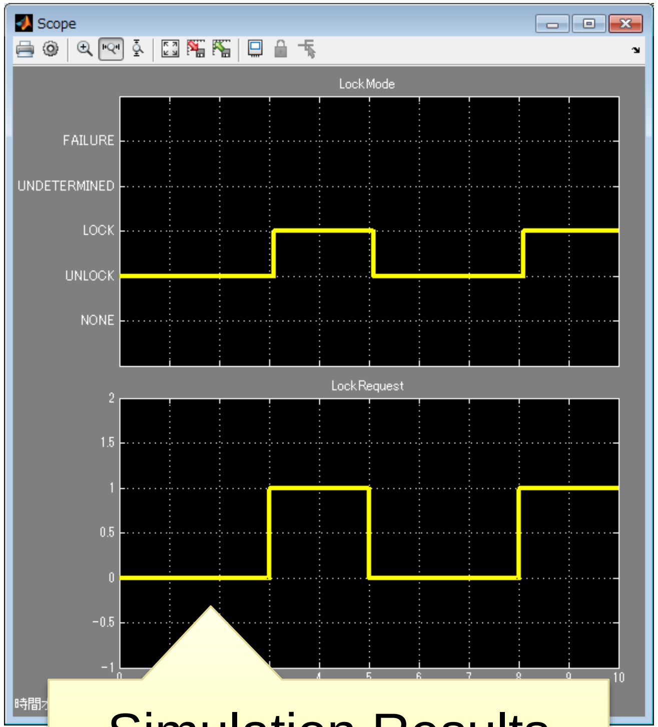
☑ Assertion

Requirements:

REQ401

Speed	(表示)
Engine ON	(表示)
Airbag ON	(表示)
DoorPosLeft	(表示)
DoorPosRight	(表示)
Ex LR	(表示)

Test data definition in Signal Builder

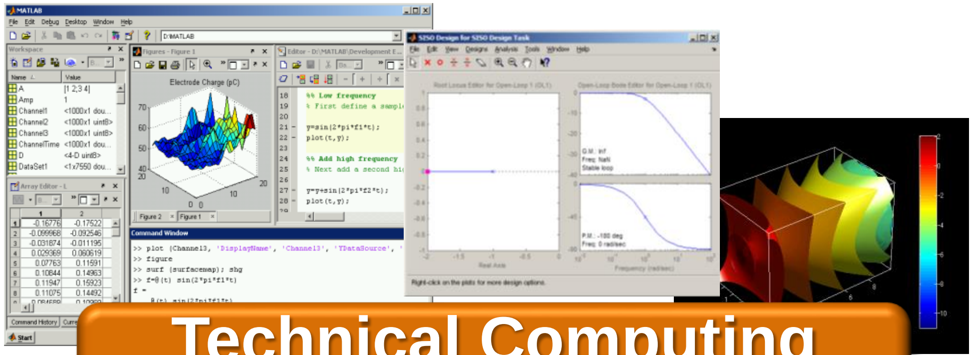


Simulation Results

MATLAB/Simulink Products

MATLAB

- Easy data processing
- Concise programming language
- Abundant mathematical functions・file I/O
- 2-D/3-D visualization functionality



Name	Value
A	[1 2;3 4]
Amp	1
Channel1	<1000x1 dou...
Channel2	<1000x1 uint8>
Channel3	<1000x1 uint8>
ChannelTime	<1000x1 dou...
D	<4-D uint8>
DataSet1	<1x7550 dou...

	1	2
1	-0.16776	-0.17522
2	-0.099968	-0.092546
3	-0.031874	-0.011195
4	0.029369	0.060619
5	0.07763	0.11591
6	0.10844	0.14963
7	0.11947	0.15923
8	0.11075	0.14492

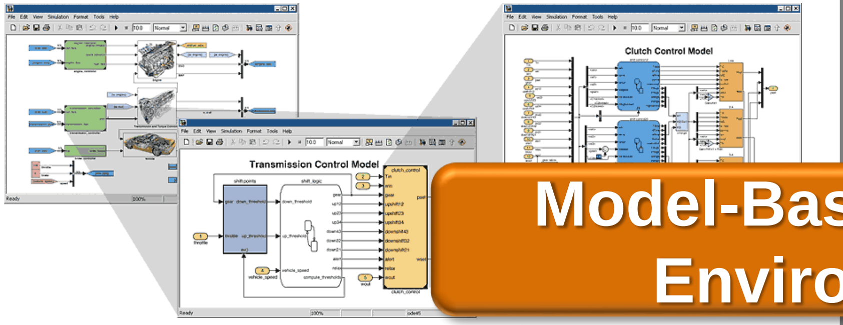
```
%% Low frequency
% First define a sampl
y=sin(2*pi*f1*t);
plot(t,y);
%% Add high frequency
% Next add a second hi
y=y+sin(2*pi*f2*t);
plot(t,y);
```

```
>> plot (Channel3, 'DisplayName', 'Channel3', 'YDataSource', '
>> figure
>> surf (surfacemap); shg
>> f=@(t) sin(2*pi*f1*t)
f =
```

Technical Computing Environment

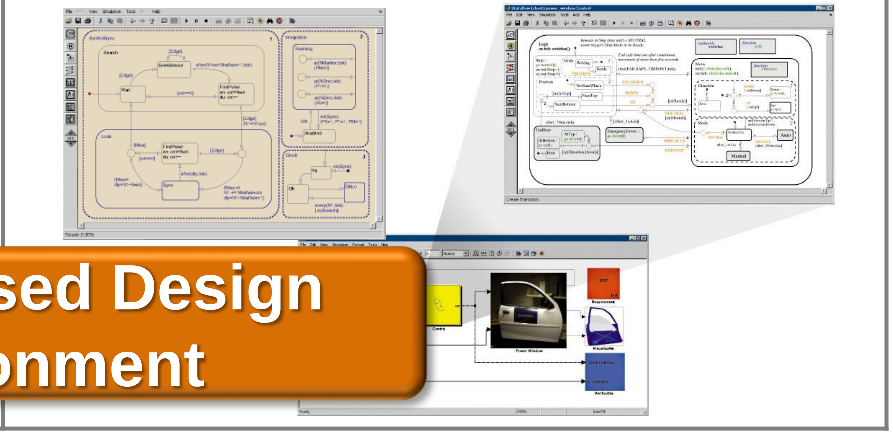
Simulink

- Block diagram modeling
- Abundant block library
- High-precision time simulation


File Edit View Simulation Format Tools Help
Clutch Control Model
Transmission Control Model
Ready
100%

Stateflow

- Flowcharts, State Diagrams, State Transition Tables

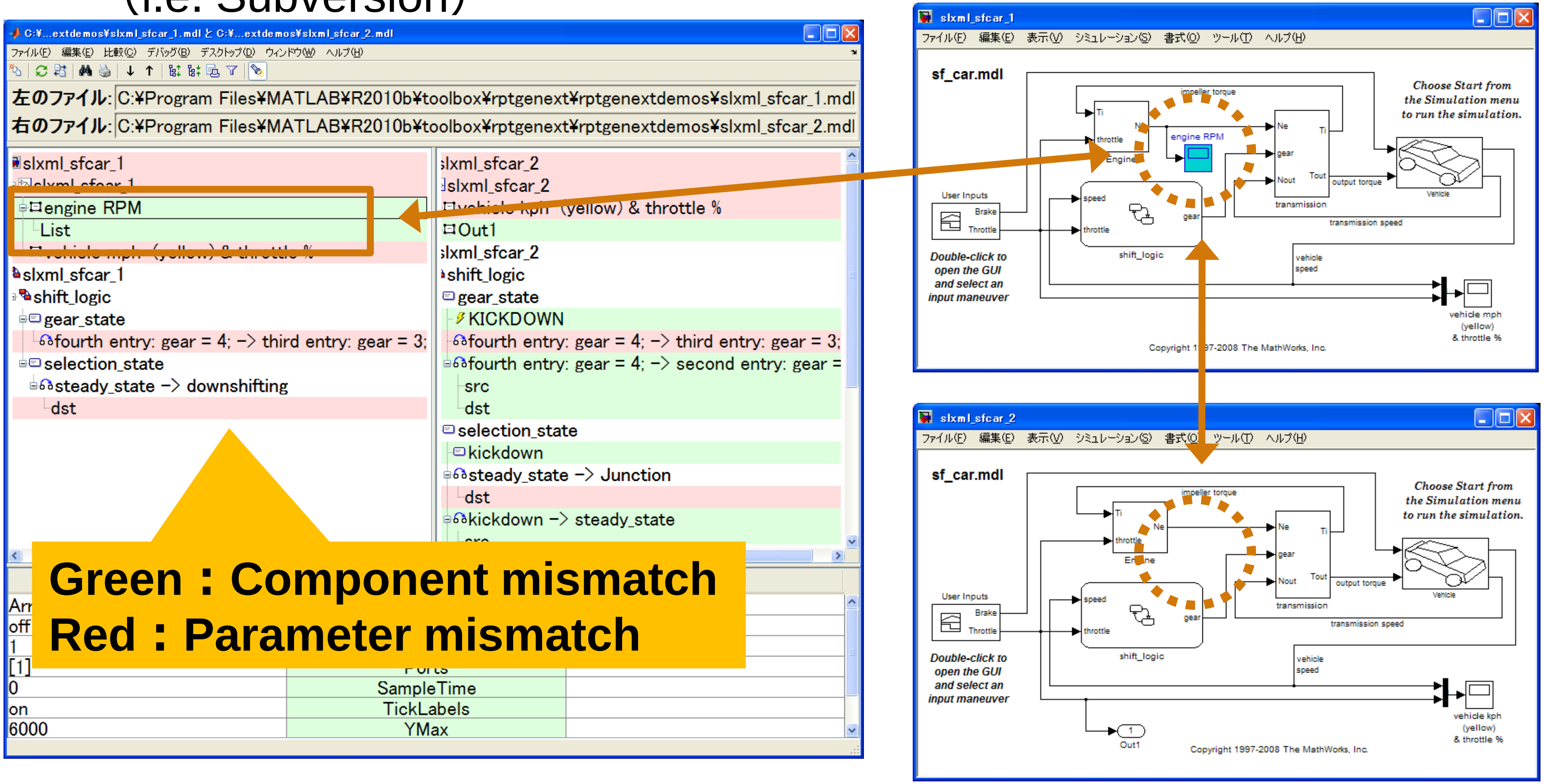


Model-Based Design Environment

Model Difference Comparisons

Simulink Report Generator

- Generate reports on difference comparisons between 2 models
  - Compatible with Simulink Project and version management software （i.e. Subversion）



C:¥...extdemos¥slxml_sfcar_1.mdl と C:¥...extdemos¥slxml_sfcar_2.mdl

ファイル(F) 編集(E) 比較(C) デバッグ(B) デスクトップ(D) ウィンドウ(W) ヘルプ(H)

左のファイル: C:¥Program Files¥MATLAB¥R2010b¥toolbox¥rptgenext¥rptgenextdemos¥slxml_sfcar_1.mdl

右のファイル: C:¥Program Files¥MATLAB¥R2010b¥toolbox¥rptgenext¥rptgenextdemos¥slxml_sfcar_2.mdl

```
slxml_sfcar_1
 slxml_sfcar_1
  engine RPM
   List
  vehicle mph (yellow) & throttle %
slxml_sfcar_1
 shift_logic
  gear_state
   fourth entry: gear = 4; -> third entry: gear = 3;
  selection_state
   steady_state -> downshifting
    dst
```

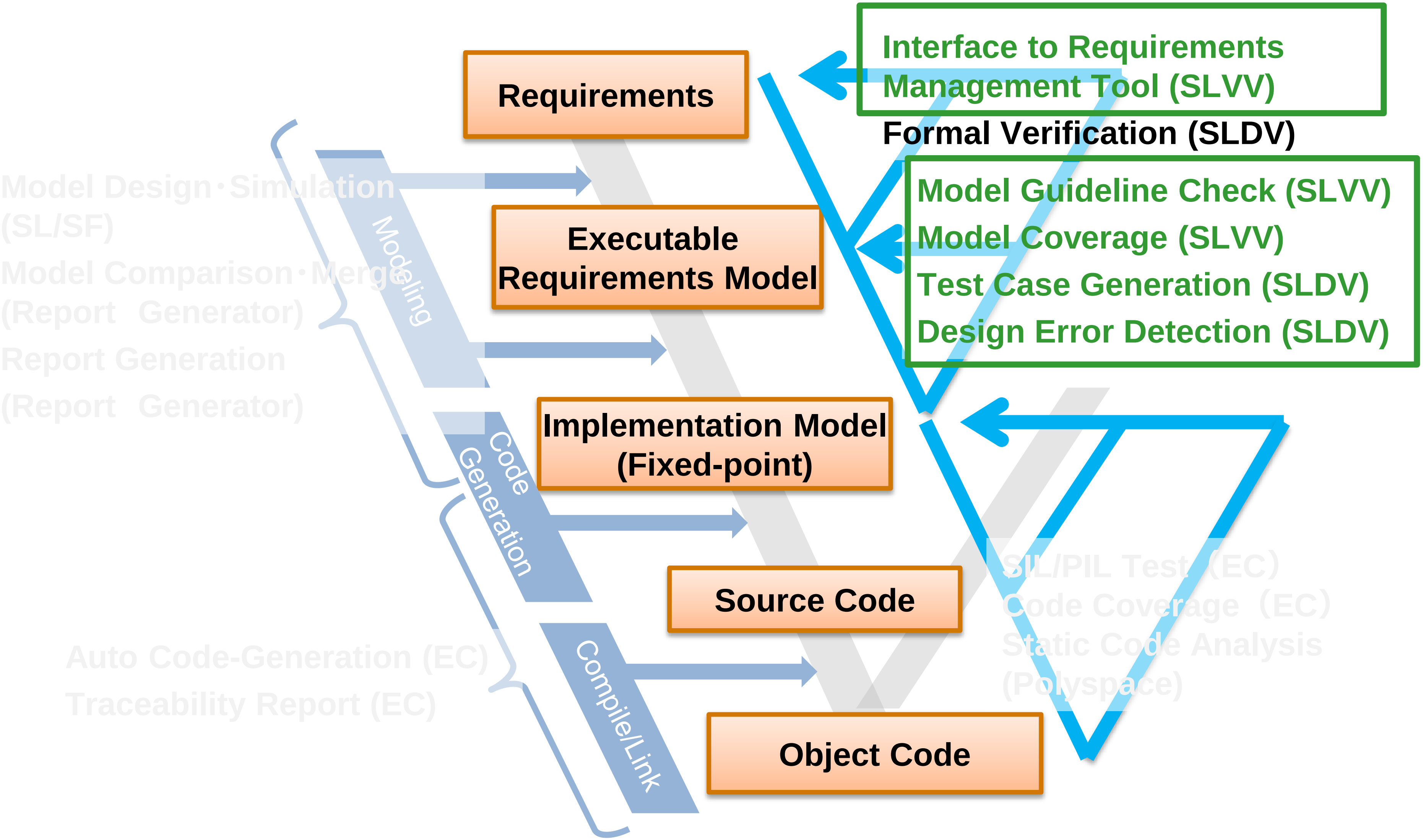
```
slxml_sfcar_2
slxml_sfcar_2
vehicle kph (yellow) & throttle %
Out1
slxml_sfcar_2
shift_logic
gear_state
 KICKDOWN
 fourth entry: gear = 4; -> third entry: gear = 3;
 fourth entry: gear = 4; -> second entry: gear =
  src
  dst
selection_state
 kickdown
 steady_state -> Junction
  dst
 kickdown -> steady_state
  src
```

Arr	
off	
1	
[1]	Ports
0	SampleTime
on	TickLabels
6000	YMax

Green：Component mismatch
Red：Parameter mismatch

slxml_sfcar_1
ファイル(F) 編集(E) 表示(V) シミュレーション(S) 書式(O) ツール(T) ヘルプ(H)
sf_car.mdl
impeller torque
Ti
Ne
throttle
Engine
engine RPM
Ne
Ti
gear
Nout
Tout
transmission
output torque
Vehicle
transmission speed
User Inputs
Brake
Throttle
speed
throttle
gear
shift_logic
vehicle speed
vehicle mph (yellow) & throttle %
Choose Start from the Simulation menu to run the simulation.
Double-click to open the GUI and select an input maneuver
Copyright 1997-2008 The MathWorks, Inc.
slxml_sfcar_2
ファイル(F) 編集(E) 表示(V) シミュレーション(S) 書式(O) ツール(T) ヘルプ(H)
sf_car.mdl
impeller torque
Ti
Ne
throttle
Engine
Ne
Ti
gear
Nout
Tout
transmission
output torque
Vehicle
transmission speed
User Inputs
Brake
Throttle
speed
throttle
gear
shift_logic
vehicle speed
vehicle kph (yellow) & throttle %
1
Out1
Choose Start from the Simulation menu to run the simulation.
Double-click to open the GUI and select an input maneuver
Copyright 1997-2008 The MathWorks, Inc.

The Next Topic



Requirements

Executable Requirements Model

Implementation Model (Fixed-point)

Source Code

Object Code

Model Design・Simulation (SL/SF)

Model Comparison・Merge (Report Generator)

Report Generation (Report Generator)

Modeling

Code Generation

Compile/Link

Auto Code-Generation (EC)

Traceability Report (EC)

Interface to Requirements Management Tool (SLVV)

Formal Verification (SLDV)

Model Guideline Check (SLVV)

Model Coverage (SLVV)

Test Case Generation (SLDV)

Design Error Detection (SLDV)

SIL/PIL Test (EC)

Code Coverage (EC)

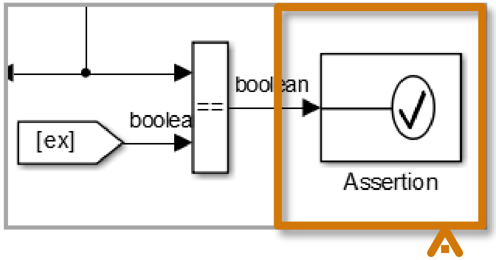
Static Code Analysis (Polyspace)

Configuration Management (Simulink Project)

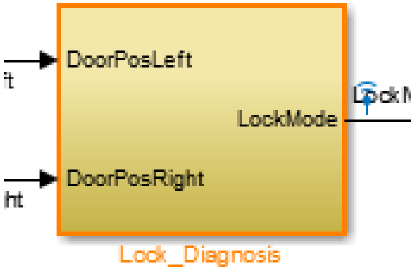
Ensure Traceability
Requirement⇔Model⇔Test

Simulink Verification & Validation

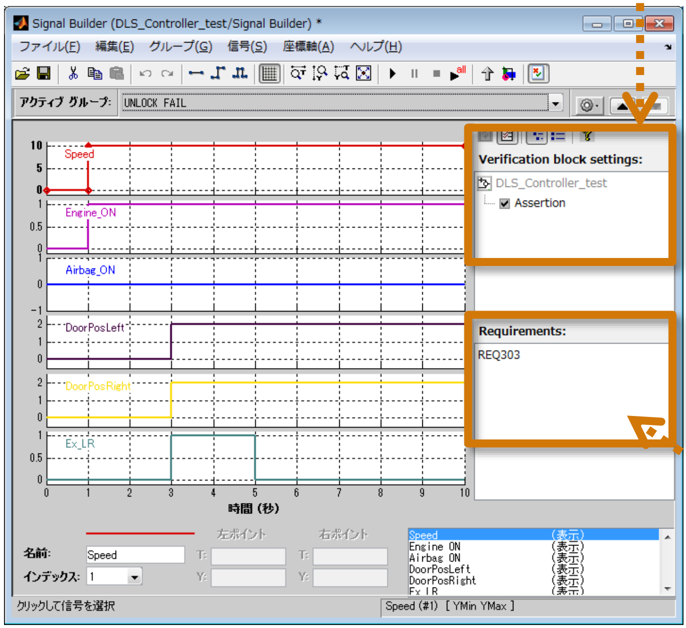
Clarification of effects of requirement changes



What is being checked?



What is being modeled?



アクティブ グループ: UNLOCK FAIL

Verification block settings:

DLS_Controller_test
☑ Assertion

Requirements:

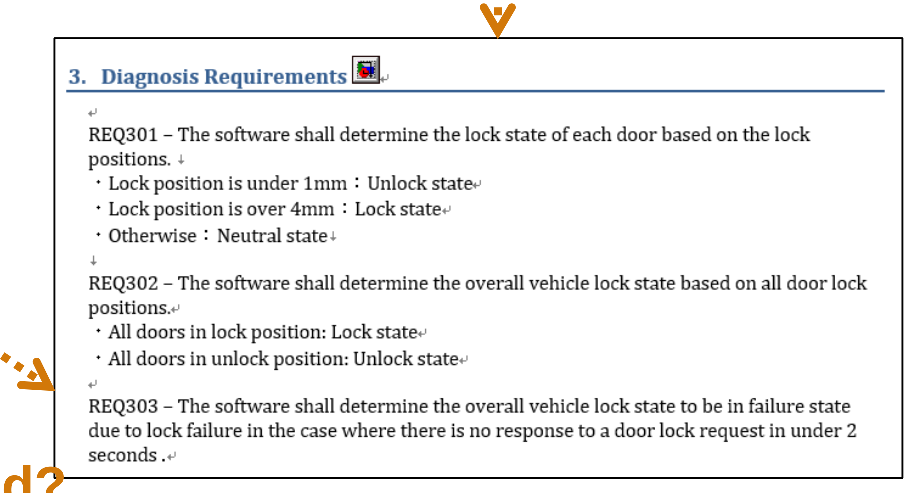
REQ303

名前: Speed

インデックス: 1

Speed (表示)
Engine ON (表示)
Airbag ON (表示)
DoorPosLeft (表示)
DoorPosRight (表示)
Ex LR (表示)

What is being tested?



3. Diagnosis Requirements

REQ301 – The software shall determine the lock state of each door based on the lock positions.

- Lock position is under 1mm：Unlock state
- Lock position is over 4mm：Lock state
- Otherwise：Neutral state

REQ302 – The software shall determine the overall vehicle lock state based on all door lock positions.

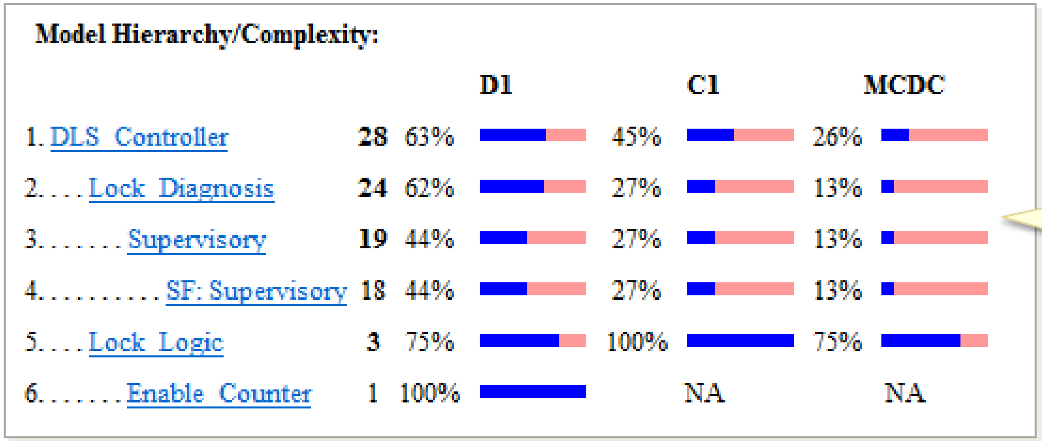
- All doors in lock position: Lock state
- All doors in unlock position: Unlock state

REQ303 – The software shall determine the overall vehicle lock state to be in failure state due to lock failure in the case where there is no response to a door lock request in under 2 seconds .

Model Coverage for Measuring Test Completeness Level

Simulink Verification & Validation

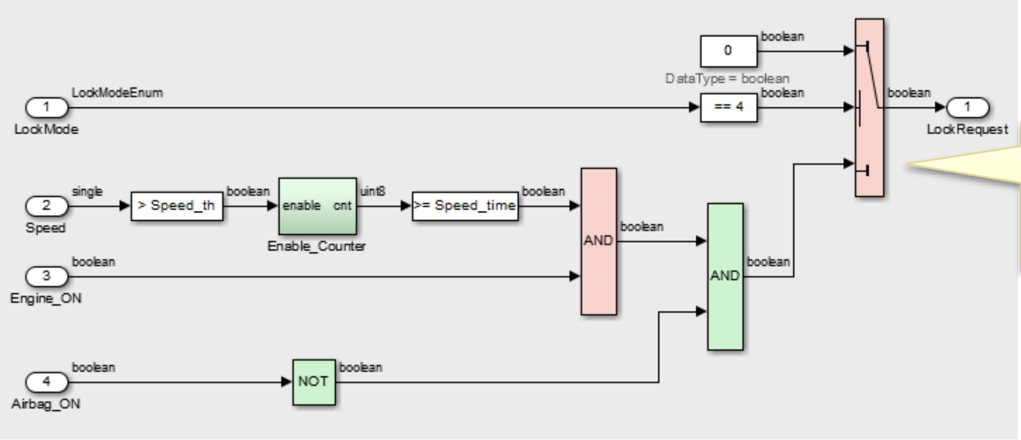
Check for insufficient testing



Model Hierarchy/Complexity:

		D1		C1	MCDC
1. DLS_Controller	28	63%		45%	26%
2. . . . Lock_Diagnosis	24	62%		27%	13%
3. . . . . . . Supervisory	19	44%		27%	13%
4. . . . . SF: Supervisory	18	44%		27%	13%
5. . . . Lock_Logic	3	75%		100%	75%
6. . . . . . . Enable_Counter	1	100%		NA	NA

Cumulative coverage results on multiple tests



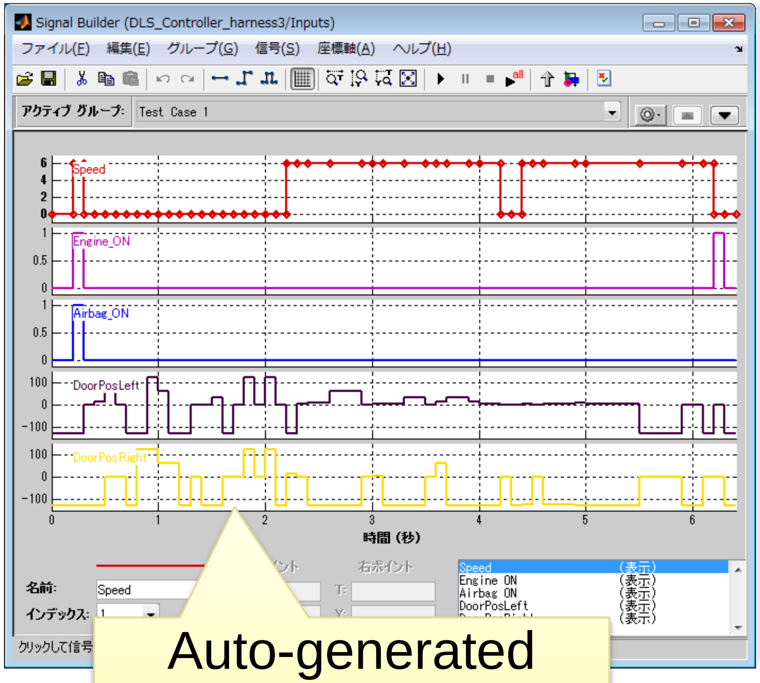
Identify areas of missing coverage

Generate Tests for Full Model Coverage

Simulink Design Verifier

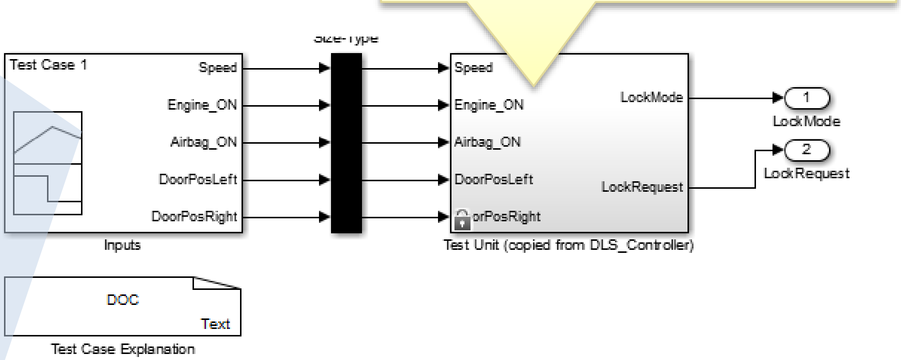
- Automatic test generation
- Suitable for equality tests

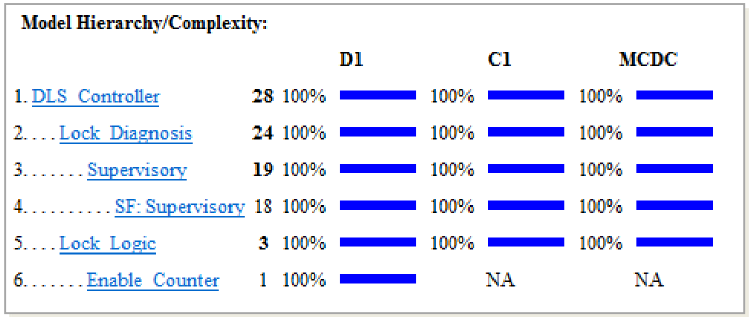
※ Able to generate missing tests based on user-defined tests



Auto-generated Test Data

Test Harness Model





Model Hierarchy/Complexity:

		D1		C1		MCDC	
1. DLS_Controller	**28**	100%	▬	100%	▬	100%	▬
2. . . . Lock_Diagnosis	**24**	100%	▬	100%	▬	100%	▬
3. . . . . . . Supervisory	**19**	100%	▬	100%	▬	100%	▬
4. . . . . SF: Supervisory	18	100%	▬	100%	▬	100%	▬
5. . . . Lock_Logic	**3**	100%	▬	100%	▬	100%	▬
6. . . . . . . Enable_Counter	1	100%	▬	NA		NA	

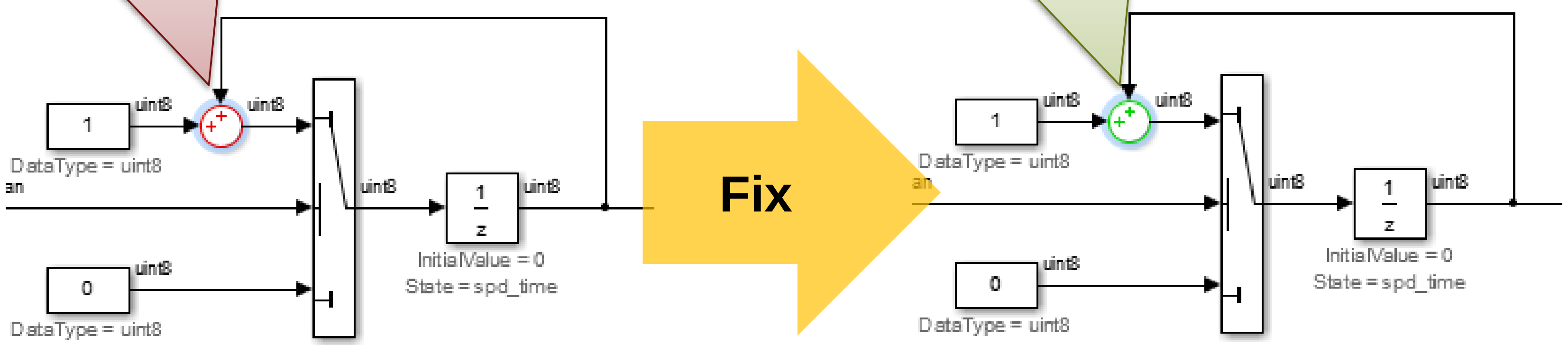
Identification of Software Design Errors

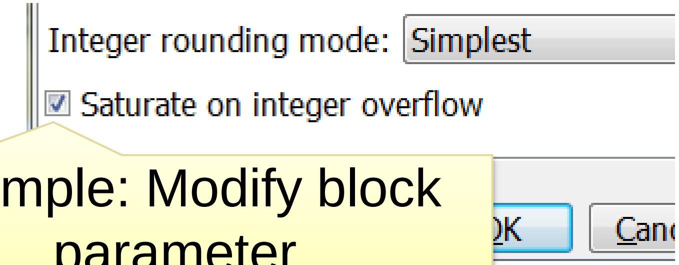
Simulink Design Verifier

- **Check for risks of software design errors prior to implementation**
  Integer overflow, division by zero, range violations, dead logic

Overflow Identified

No risk of overflow





Integer rounding mode: Simplest

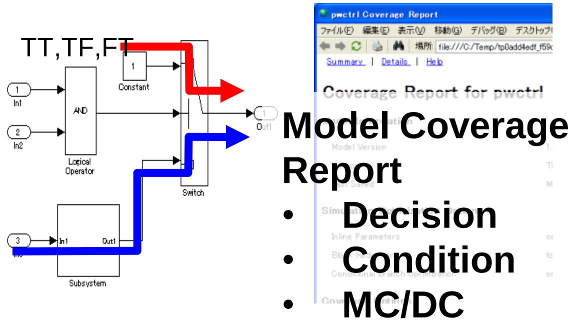
☑ Saturate on integer overflow

Example: Modify block parameter

Model Verification & Validation Products

Simulink Verification and Validation^{TM} (SLVnV)

Measure Model Coverage



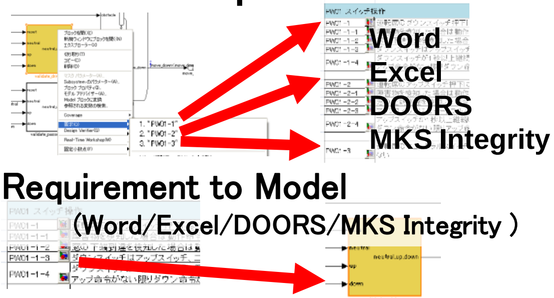
Model Coverage Report

- Decision
- Condition
- MC/DC

Test Data Sufficiency Check

Traceability

Model to Requirement



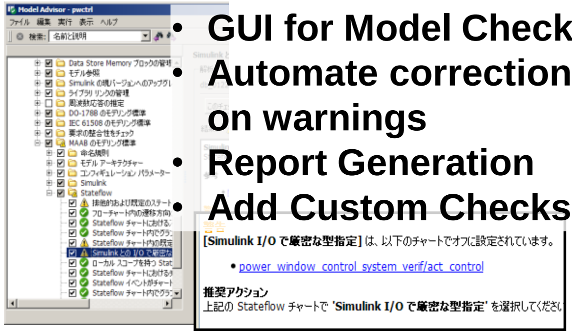
- Word
- Excel
- DOORS
- MKS Integrity

Requirement to Model

(Word/Excel/DOORS/MKS Integrity)

Requirement Sufficiency Check

Model Checker (Model Advisor)



- GUI for Model Checks
- Automate corrections on warnings
- Report Generation
- Add Custom Checks

[Simulink I/O で厳密な型指定] は、以下のチャートでオフに設定されています。

- power_window_control_system_verif/act_control

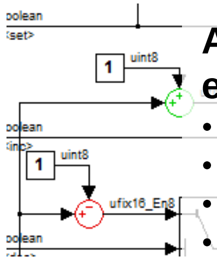
推奨アクション

上記の Stateflow チャートで 'Simulink I/O で厳密な型指定' を選択してくださ

Automate Model Checking

Simulink Design Verifier^{TM} (SLDV)

Design Error Detection

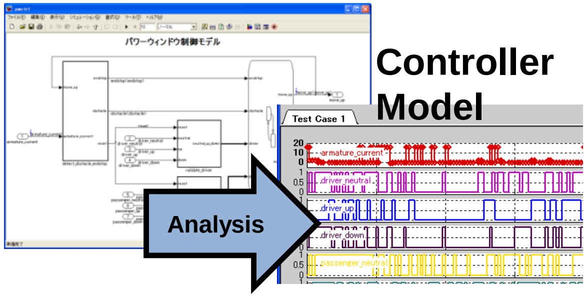


Auto-detect design errors

- Division by zero
- Range overflow
- Deadl Logic
- Saturation overflow
- Out of bounds access

Automate Error Detection

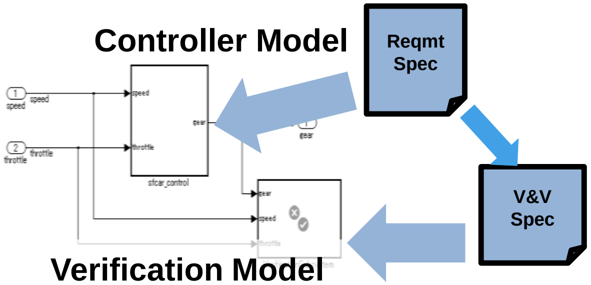
Auto-Generate Test Cases



Controller Model

100% Coverage Test Data

Property Proving (Formal Methods)

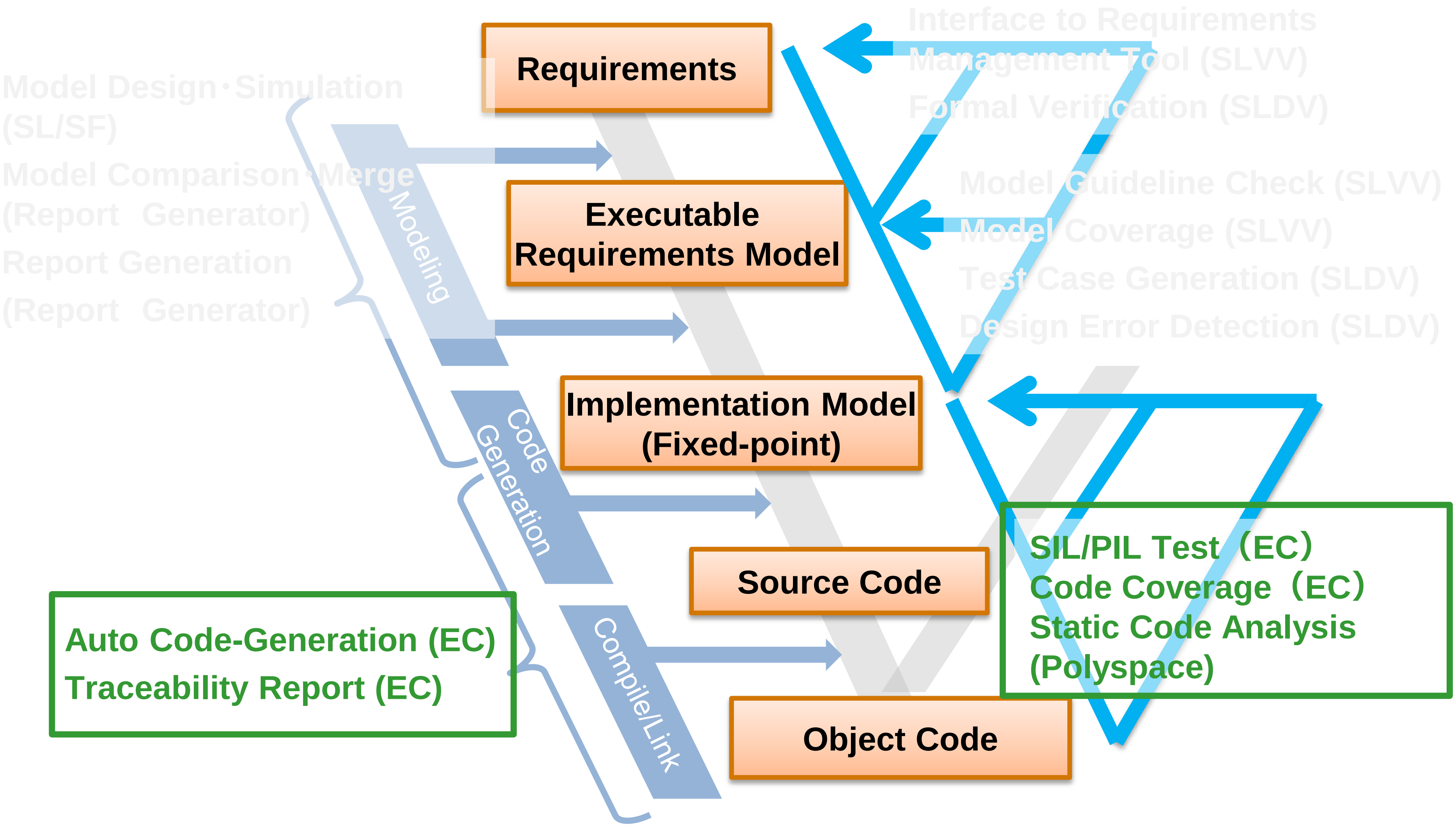


Controller Model

Verification Model

Certify Correct Behavior

The Final Topic



Requirements

Executable Requirements Model

Implementation Model (Fixed-point)

Source Code

Object Code

Model Design·Simulation (SL/SF)

Model Comparison·Merge (Report Generator)

Report Generation (Report Generator)

Interface to Requirements Management Tool (SLVV)

Formal Verification (SLDV)

Model Guideline Check (SLVV)

Model Coverage (SLVV)

Test Case Generation (SLDV)

Design Error Detection (SLDV)

Auto Code-Generation (EC)

Traceability Report (EC)

SIL/PIL Test（EC）

Code Coverage（EC）

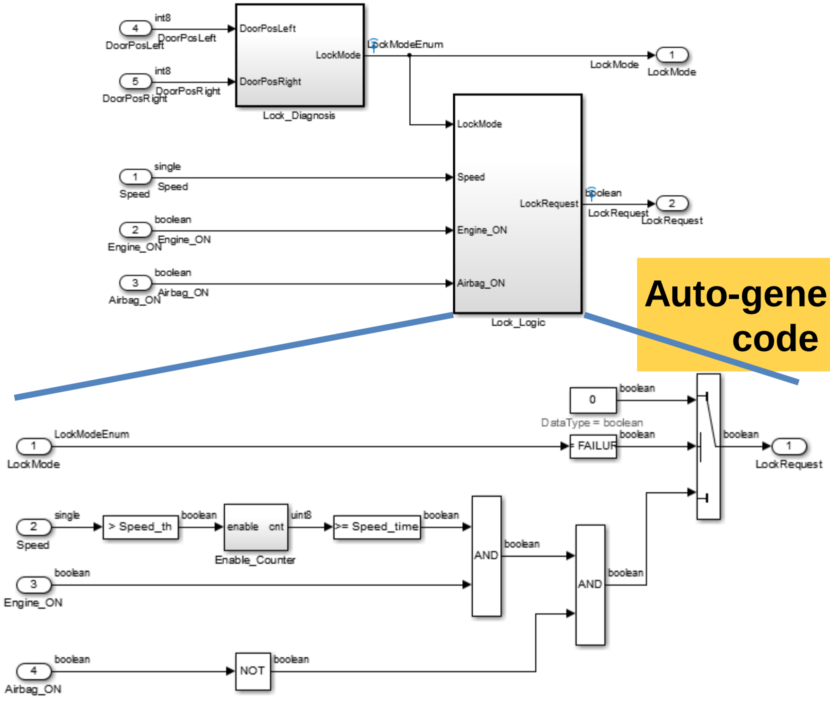
Static Code Analysis (Polyspace)

Configuration Management (Simulink Project)

Generate Code from Controller Model

Embedded Coder

- Auto-generate C-code of high readability/efficiency
- Option settings for variable attributes, function settings, code style, etc.
- Auto-generate scaling for fixed-point design



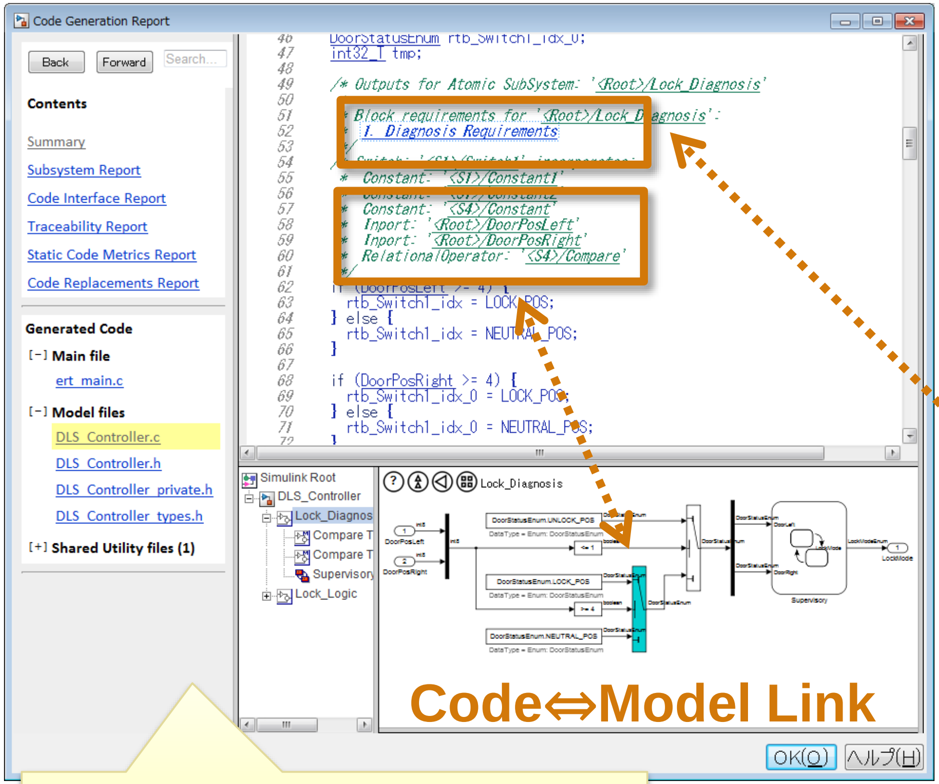
Auto-generate code

```
if (LockMode == FAILURE) {
  LockRequest = FALSE;
} else {
  LockRequest =
   ((spd_time >= Speed_time) &&
   Engine_ON && (!Airbag_ON));
}
```

Ensuring Traceability between Requirements, Models, and Code

Embedded Coder / Simulink Report Generator

- Reflect model specifications in generated code
- Distribute reports with model views (html)

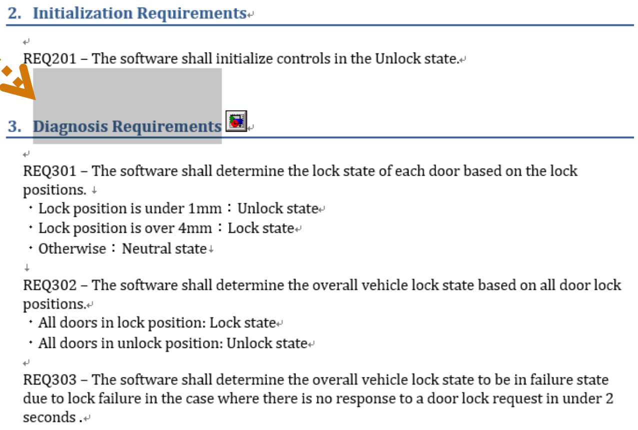


```
  DoorStatusEnum rtb_Switch1_idx_0;
  int32_T tmp;

  /* Outputs for Atomic SubSystem: '<Root>/Lock_Diagnosis'
   *
   * Block requirements for '<Root>/Lock_Diagnosis':
   *  1. Diagnosis Requirements
   */

  /* Switch: '<S1>/Switch1' incorporates:
   *  Constant: '<S1>/Constant1'
   *  Constant: '<S1>/Constant2'
   *  Constant: '<S4>/Constant'
   *  Inport: '<Root>/DoorPosLeft'
   *  Inport: '<Root>/DoorPosRight'
   *  RelationalOperator: '<S4>/Compare'
   */
  if (DoorPosLeft >= 4) {
    rtb_Switch1_idx = LOCK_POS;
  } else {
    rtb_Switch1_idx = NEUTRAL_POS;
  }

  if (DoorPosRight >= 4) {
    rtb_Switch1_idx_0 = LOCK_POS;
  } else {
    rtb_Switch1_idx_0 = NEUTRAL_POS;
  }
```

Code⇔Model Link

Code Generation Report

Code⇔Document Link



2. Initialization Requirements

REQ201 – The software shall initialize controls in the Unlock state.

3. Diagnosis Requirements

REQ301 – The software shall determine the lock state of each door based on the lock positions.
- Lock position is under 1mm：Unlock state
- Lock position is over 4mm：Lock state
- Otherwise：Neutral state

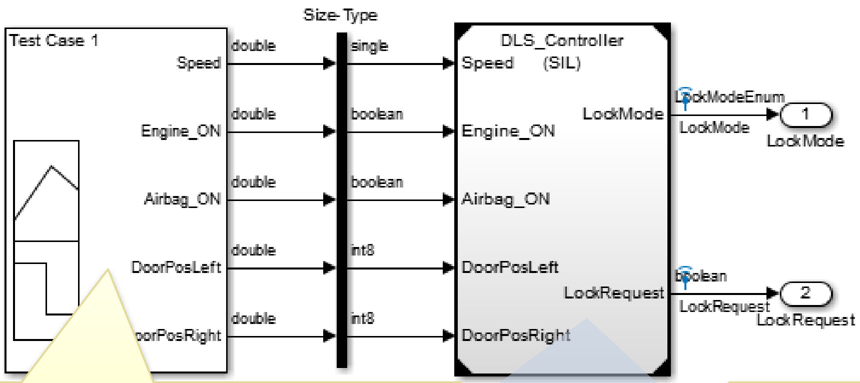
REQ302 – The software shall determine the overall vehicle lock state based on all door lock positions.
- All doors in lock position: Lock state
- All doors in unlock position: Unlock state

REQ303 – The software shall determine the overall vehicle lock state to be in failure state due to lock failure in the case where there is no response to a door lock request in under 2 seconds .

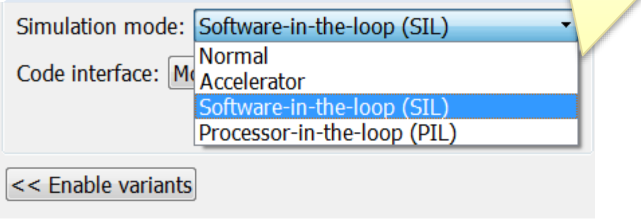
Model⇔Code Equality Checks （SIL/PIL, Back 2 Back Test）

Embedded Coder

Efficient testing by reuse of model verification test data



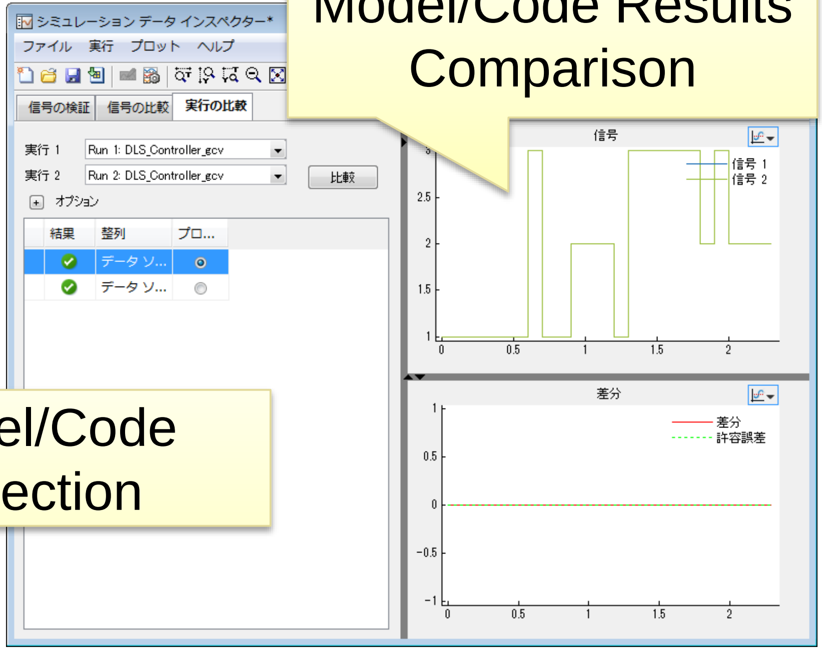
Existing data/SLDV generated test data



Simulation mode: Software-in-the-loop (SIL)

Code interface: Mo

Model/Code Selection



実行 1 Run 1: DLS_Controller_gcv

実行 2 Run 2: DLS_Controller_gcv

Model/Code Results Comparison

※ Test automation through Simulink Test.

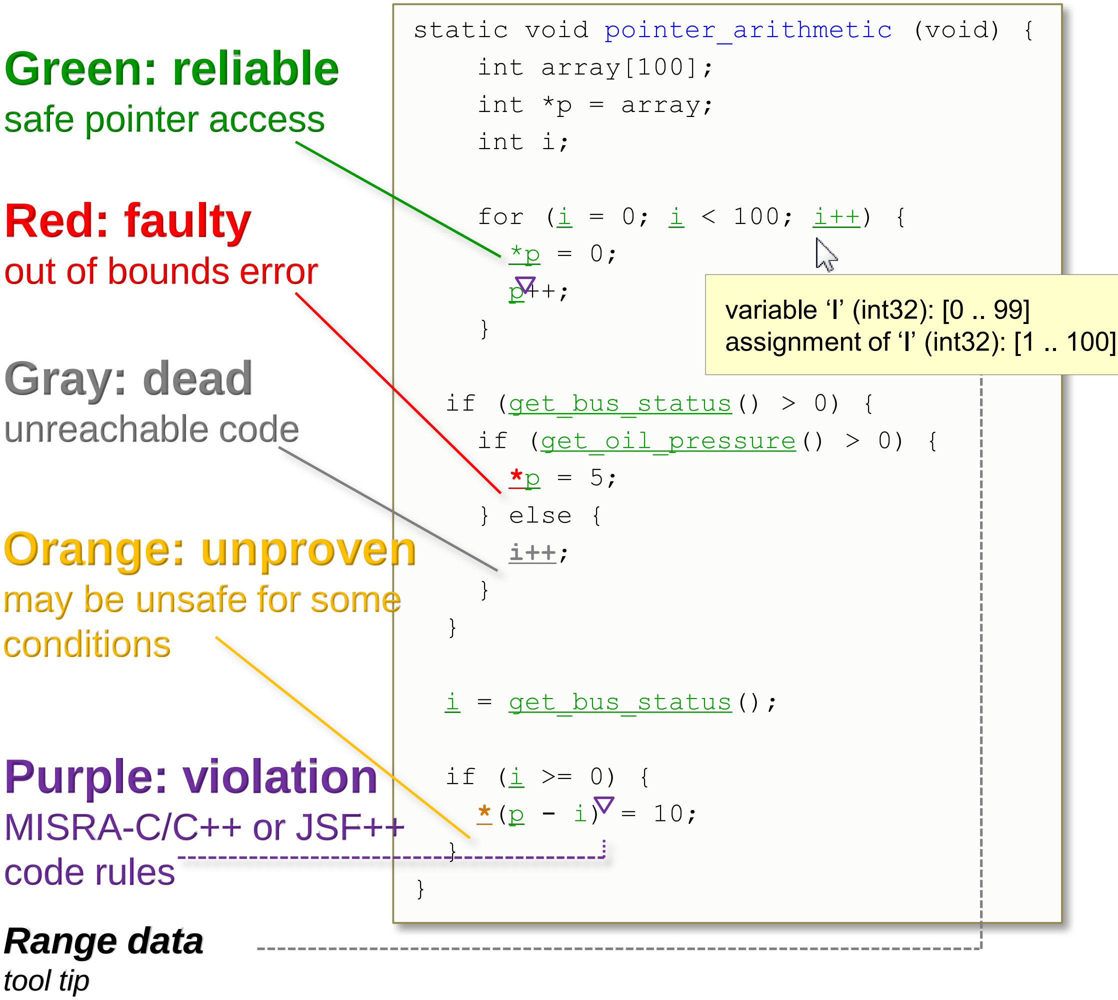
Tool Chain Example: Product List

Product	Functionality	Usage
Simulink	Modeling: Controller Block	Modeling Module/Integration Test
Stateflow	Modeling: State Transitions, Flow Charts	Modeling
Fixed-Point Designer	Modeling: Fixed-Point Processing	Modeling
Simulink Verification and Validation	Model Coverage Requirements Interface Model Advisor	Module/Integration Test Review and Static Analysis
Simulink Design Verifier	Property Proving Test Generation Design Error Detection	Review and Static Analysis
Embedded Coder	Code Generation PIL Test/CGV Bullseye/LDRA Integration Traceability Report	Code Generation Equality Testing Code Coverage Measurement
IEC Certification Kit	Traceability Matrix Generation Templates for Certification	ISO26262 Support
Simulink Report Generator	Report Editing and Generation	Report Generation Model Comparison/Merge

Proving Source Code Correctness

Polyspace Code Prover: Static Code Verification

- **Quality**
  - Prove absence of runtime errors (RTEs)
  - Measure, Improve, Manage
- **Usage**
  - No need to compile, execute, or generate test cases
  - Supports：C/C++/Ada
- **Process**
  - Early detection of RTEs
  - Analyze both hand-code and auto-generated code
  - Measure code reliability



Green: reliable
safe pointer access

Red: faulty
out of bounds error

Gray: dead
unreachable code

Orange: unproven
may be unsafe for some conditions

Purple: violation
MISRA-C/C++ or JSF++ code rules

Range data
tool tip

```
static void pointer_arithmetic (void) {
  int array[100];
  int *p = array;
  int i;

  for (i = 0; i < 100; i++) {
   *p = 0;
   p++;
  }

 if (get_bus_status() > 0) {
  if (get_oil_pressure() > 0) {
   *p = 5;
  } else {
   i++;
  }
 }

 i = get_bus_status();

 if (i >= 0) {
  *(p - i) = 10;
 }
}
```

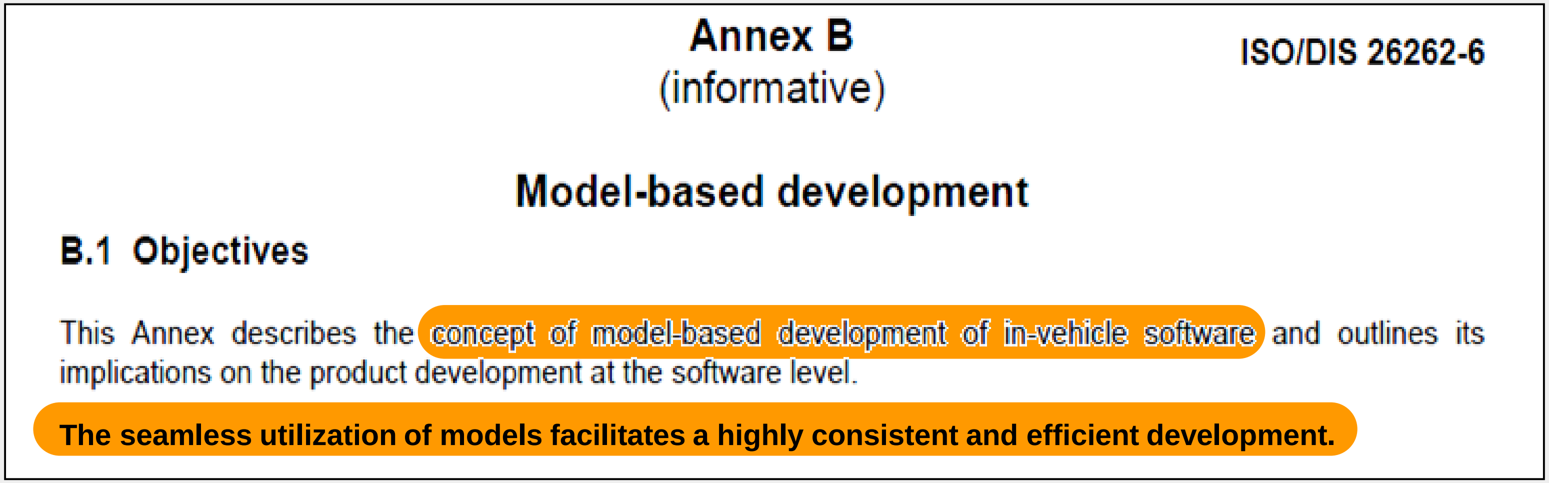
variable ‘I’ (int32): [0 .. 99]
assignment of ‘I’ (int32): [1 .. 100]

Analyze all executable paths to detect errors and prove the absence of errors

ISO26262 Functional Safety Standard

- Functional safety standard for automotive equipment
- Based on IEC61508
- Description of purpose and requirements for development
  - Activities for development process （Software safety life cycle）
  - Development and verification tools （Tool qualification）
- Description of new software engineering concepts
  - **Model-based development**
  - **Early verification and validity checks**
  - **Automatic code generation**

Model-Based Design Benefits（ISO26262 excerpt）



ISO/DIS 26262-6

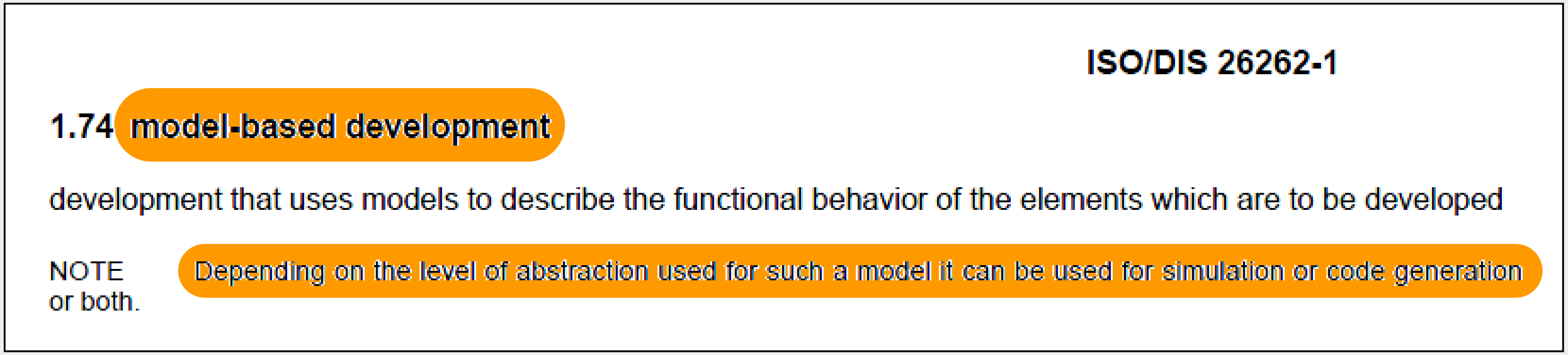
Annex B
(informative)

Model-based development

B.1 Objectives

This Annex describes the concept of model-based development of in-vehicle software and outlines its implications on the product development at the software level.

The seamless utilization of models facilitates a highly consistent and efficient development.



ISO/DIS 26262-1

1.74 model-based development

development that uses models to describe the functional behavior of the elements which are to be developed

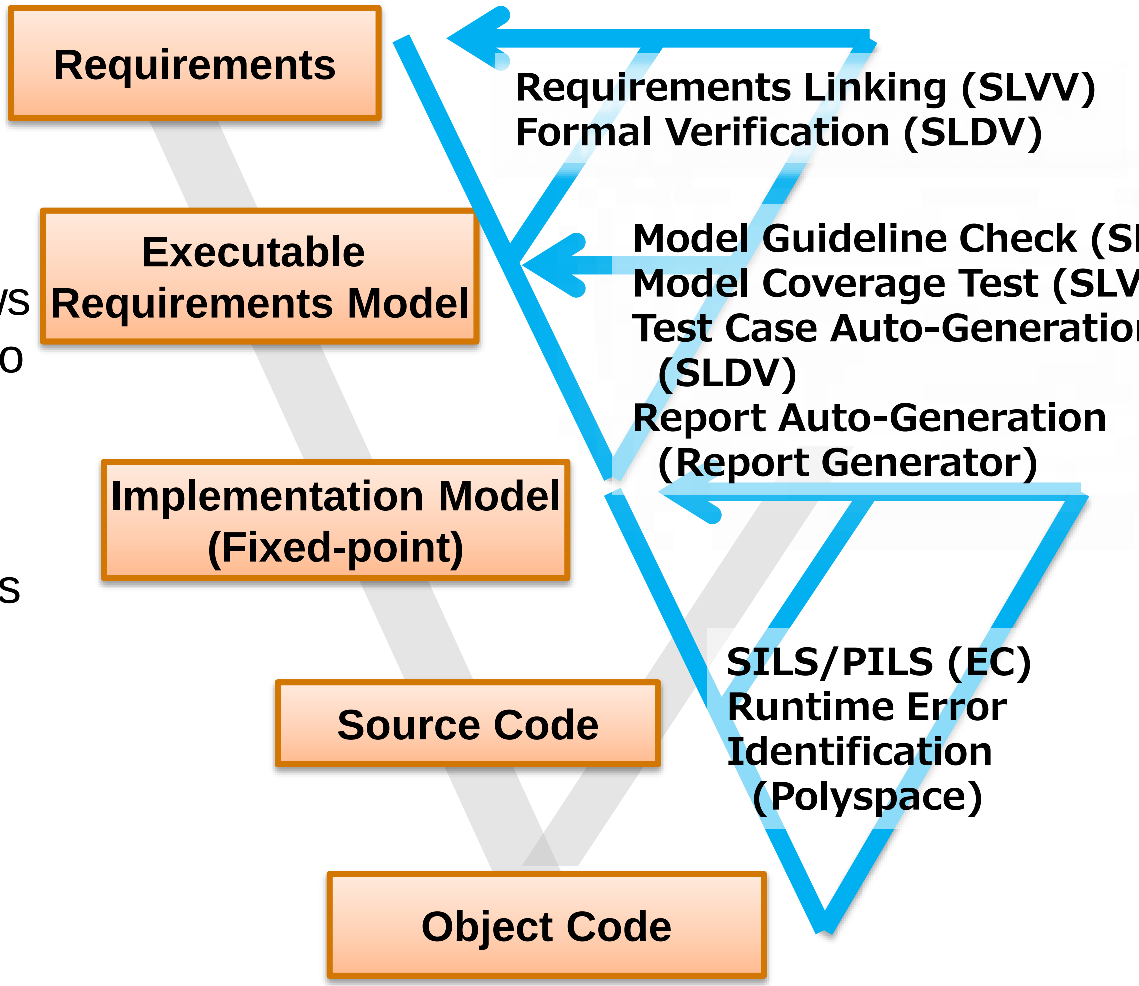
NOTE Depending on the level of abstraction used for such a model it can be used for simulation or code generation or both.

MathWorks Solution: Summary

Using Models to Detect Errors Early and Increase Efficiency

・Able to form small V-loops
・Able to detect errors early in the development cycle

- Mode⇔Code consistency allows for Simulink simulation results to be considered “truth”.
- Early model verification is possible due to the ability to investigate floating-point models
- Large team development made easy through highly customizable tool chain
- Errors in object code detected easily through synchronization between simulations and SILS/PILS



Requirements

Executable Requirements Model

Implementation Model (Fixed-point)

Source Code

Object Code

Requirements Linking (SLVV)
Formal Verification (SLDV)

Model Guideline Check (SI
Model Coverage Test (SLV
Test Case Auto-Generatio
(SLDV)
Report Auto-Generation
(Report Generator)

SILS/PILS (EC)
Runtime Error Identification
(Polyspace)