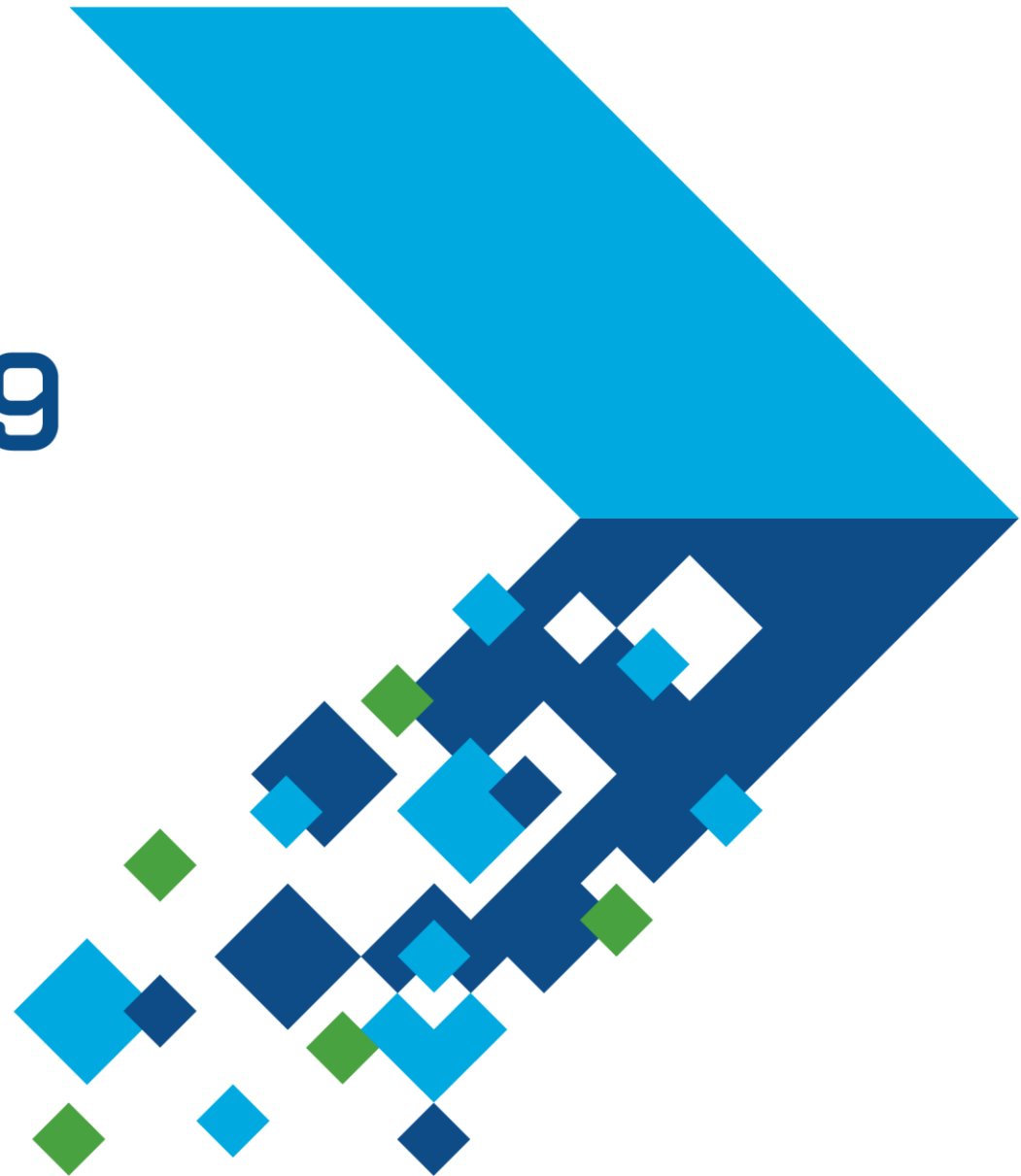


MATLAB EXPO 2019

“软件定义一切”的机遇与挑战

MathWorks 宋胜凯



“软件” 无处不在



The USS Makin Island.



The Alenia Aermacchi M-346.



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



Airmanics co-founders Marko Thaler and Zoran Bjelić with the R5 MSN1 prototype after its first flight.



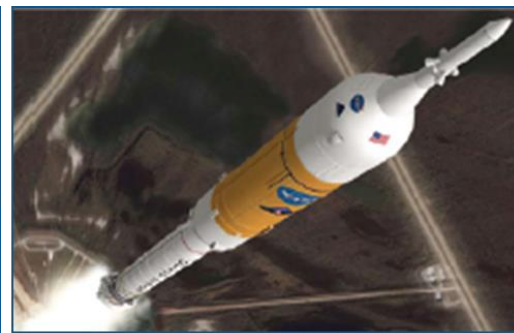
Airbus A380, the world's largest commercial aircraft.



RMSV-designed Unmanned Combat Ground Vehicle.



HAWK surface-to-air missile launch.



NASA's Ares I rocket.



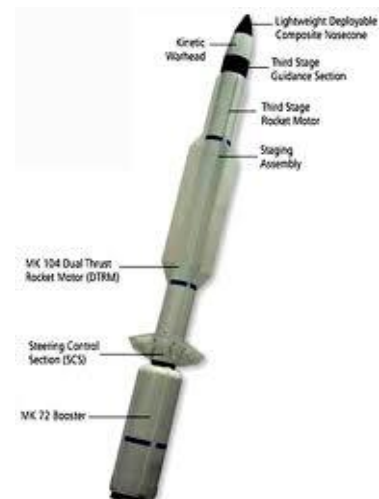
The IRIS observatory.



Artist's rendition of Mars rover. Graphics courtesy of NASA/JPL/Cornell.

MBD无处不在

软件规模庞大并持续增长



Aegis missile system



- SBIRS 卫星 ~25,000 行代码
- Aegis 导弹 ~400万行代码
- 787 梦想客机 ~ 650万行代码
- GM Volt ~ 1000万行代码

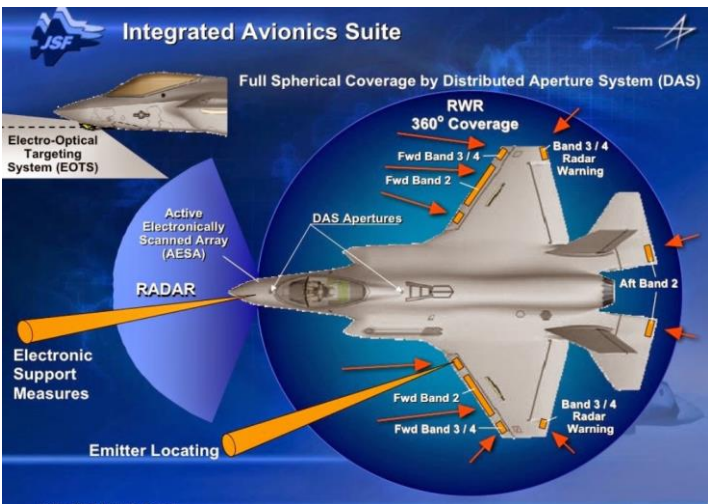


软件规模持续增长!

系统复杂度增加



多功能雷达系统：搜索、跟踪、监视、导引



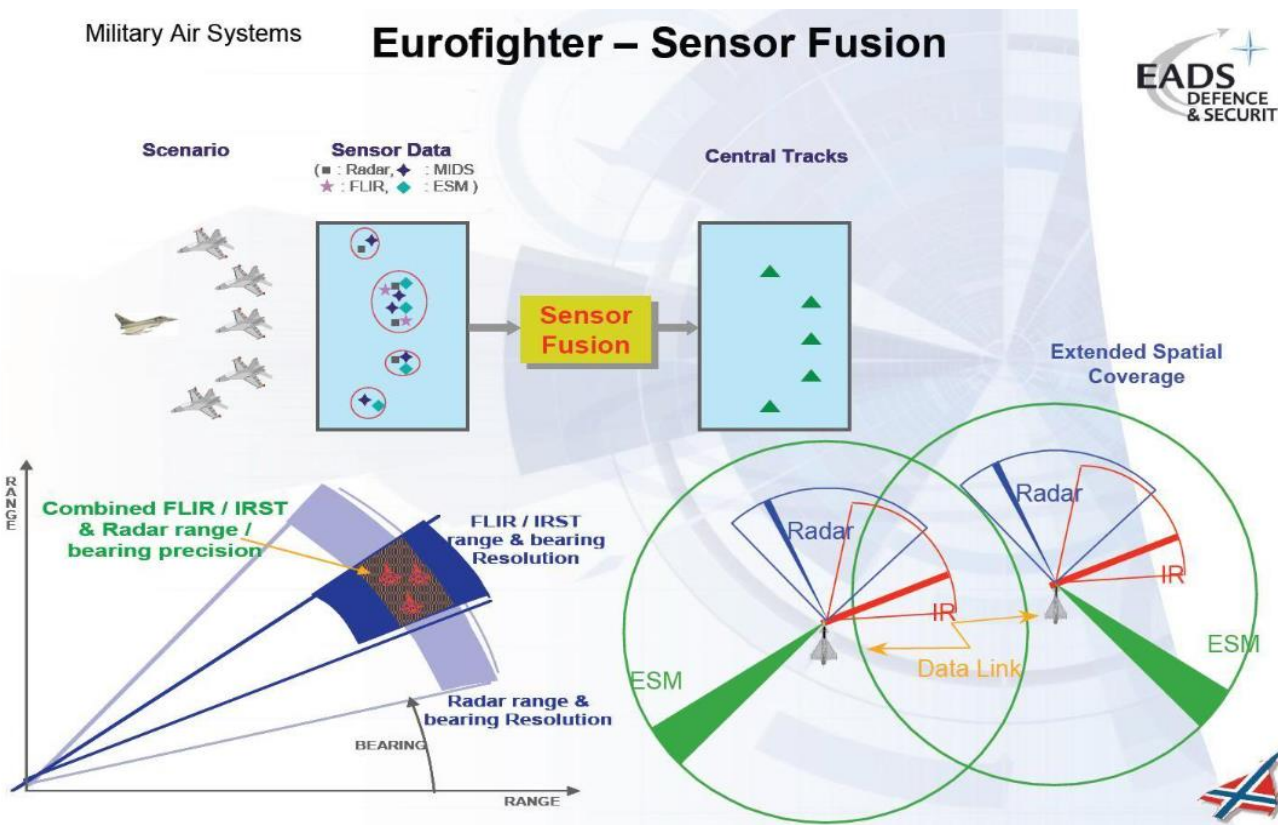
多功能孔径：雷达、电子战、通信...

软件定义、高度综合！

Military Air Systems

Eurofighter – Sensor Fusion

EADS DEFENCE & SECURITY



多传感器系统：雷达、电子支援、光电/红外...

Reutech Radar System

■ 基于模型的舰载海空搜索警戒雷达的系统开发

挑战

- 此舰载雷达的首要目标是它能够在动态变化的环境中，适应较大范围的多重海况。RRS团队必须在航海试验时收集到的数据基础上，对设计进行快速的更新和修正。
- 处理算法复杂，需要多片FPGA协同处理，算法验证将仅能在底层信息交互架构调试完成后，才能进行。

解决方案

- 使用MATLAB和Simulink工具，采用基于模型设计（MBD）的流程，来开发算法、模型的关键单元，继而完成系统级的仿真
- 使用Fixed-Point Designer把浮点数据模型设计自动转化为定点数据模型
- 从模型中直接生成HDL代码
- 在海试时，快速尝试、调整和定点化算法，并自动生成代码

结果

- 10分钟内自动生成**75,000行HDL代码**
- 开发时间减少4171小时（约两个工程师人*年）
- 信号处理模块可复用
- FPGA固件（firmware）高可靠性



The RSR 210N multipurpose 2D radar system.

“若不采用基于模型设计（MBD），要想按时完成本项目将会非常困难。使用HDL Coder自动生成HDL代码，以及将信号处理算法的设计与详细的硬件实现分离开来，这两项能力帮我们节省了两个工程师人*年。”

— Kevin Williams, Reutech Radar System

复杂系统开发

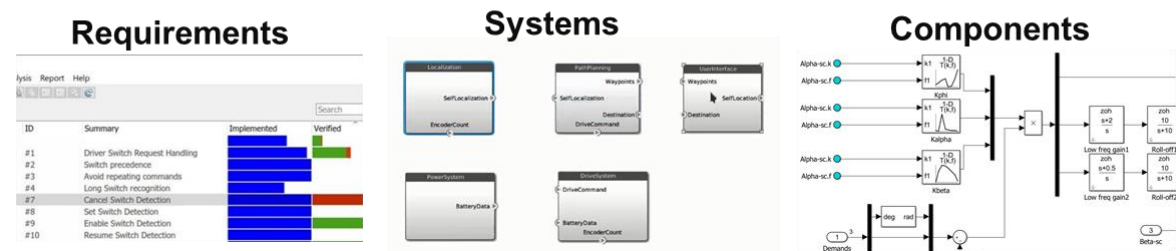
*“The current trend in system design is an increasing level of integration between aircraft functions and the systems that implement them. While there can be considerable value gained when integrating systems with other systems, **the increased complexity yields increased possibilities for errors, particularly with functions that are performed jointly across multiple systems.**”*

[ARP4754]

“Modeling, simulation, and prototyping used during architecture definition can significantly reduce the risk of failure in the finished system. Systems engineers use modeling techniques and simulation on large complex systems to manage the risk of failure to meet system mission and performance requirements.”

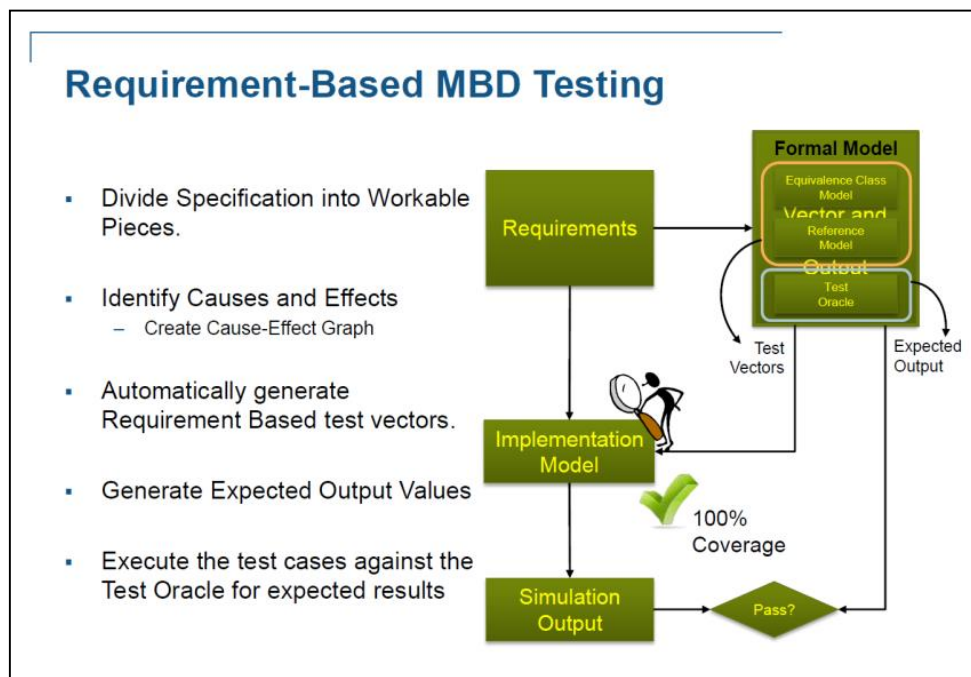
[INCOSE]

仿真(Simulation) & 基于模型设计(MBD)



需求建模和验证

需求模型-基于需求的测试



Lear Delivers Quality Body Control Electronics Faster Using Model-Based Design



Lear automotive body electronic control unit.

Challenge

Design, verify, and implement high-quality automotive body control electronics

Solution

Use Model-Based Design to enable early and continuous verification via simulation, SIL, and HIL testing

Results

- Requirements validated early. Over 95% of issues fixed before implementation, versus 30% previously
- Development time cut by 40%. 700,000 lines of code generated and test cases reused throughout the development cycle
- Zero warranty issues reported

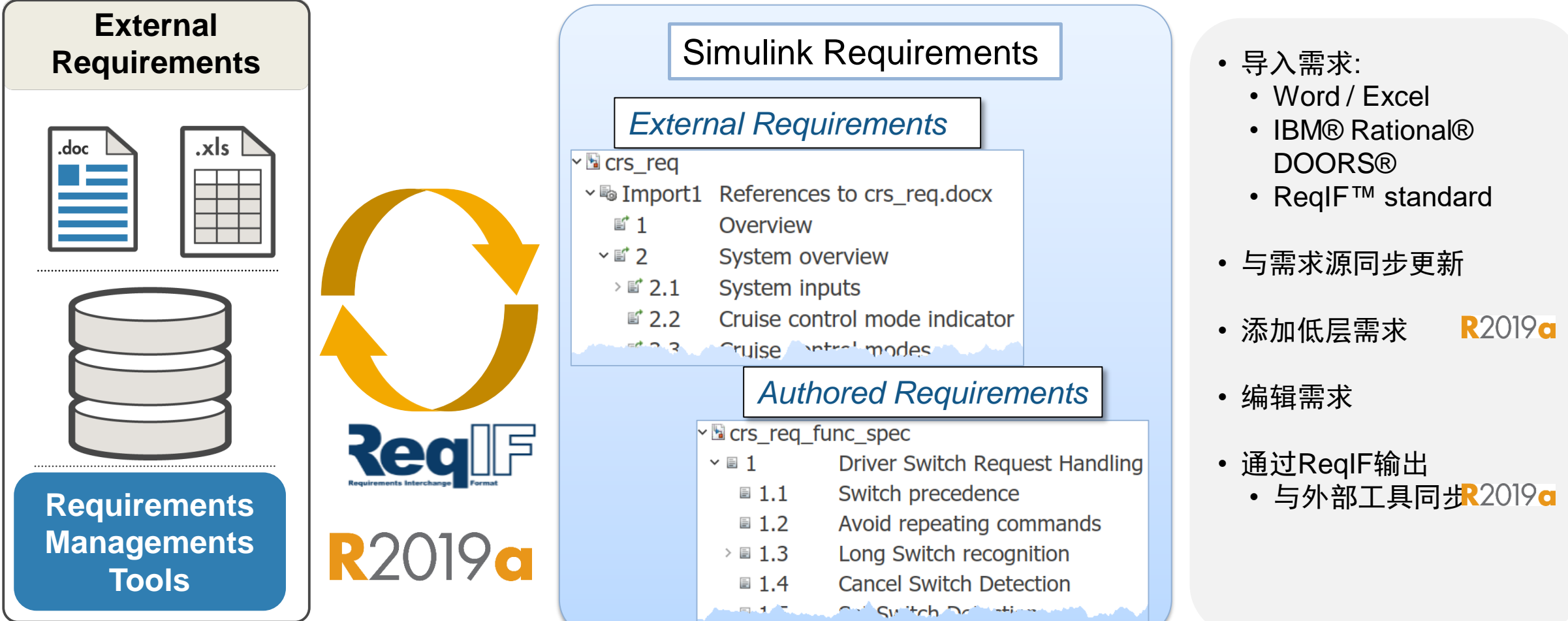
"We adopted Model-Based Design not only to deliver better-quality systems faster, but because we believe it is a smart choice. Recently we won a project that several of our competitors declined to bid on because of its tight time constraints. Using Model-Based Design, we met the original delivery date with no problem."

Jason Bauman
Lear Corporation

[Link to user story](#)

高层抽象模型

需求的关联和追踪



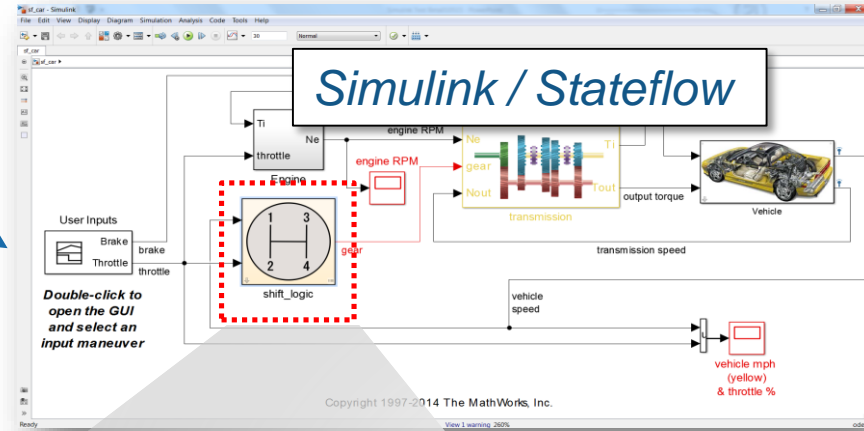
系统实现和需求验证

Requirements

- TransmissionReq
 - 1 Transmission Operating Modes
 - 1.1 Reverse cannot be entered from drive
 - 1.2 Engine only starts in Park

Implemented
By

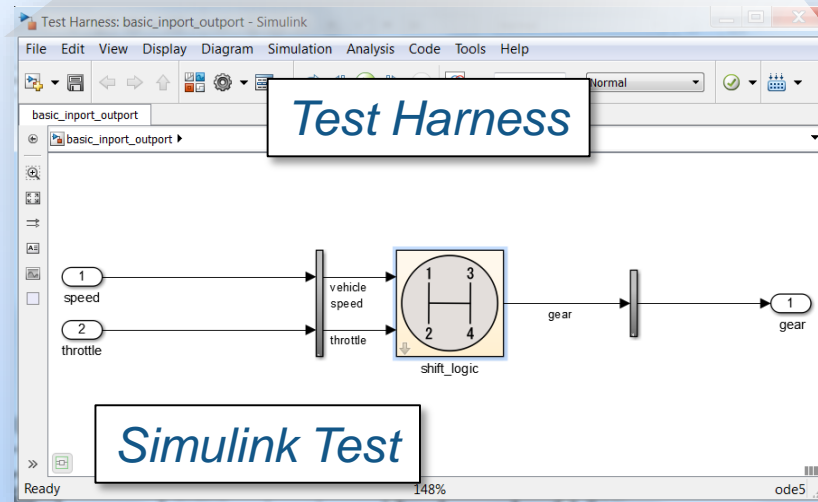
Verified
By



Test Case

Inputs

MAT / Excel file (input)
 Scenario Signal Editor
 Test Sequence



Assessments

MAT / Excel File (baseline)
 Test Assessments

```

function customCriteria
Perform custom criteria
1 test.verifyThat(test.sl
    
```

 MATLAB Unit Test

基于模型的设计过程

- 连续时间/离散时间系统建模
- 状态机建模
- 物理系统建模
- 离散事件系统建模



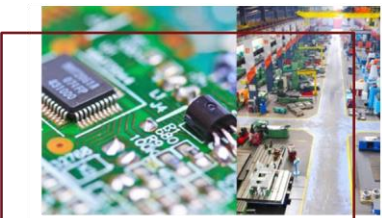
Dynamic Systems



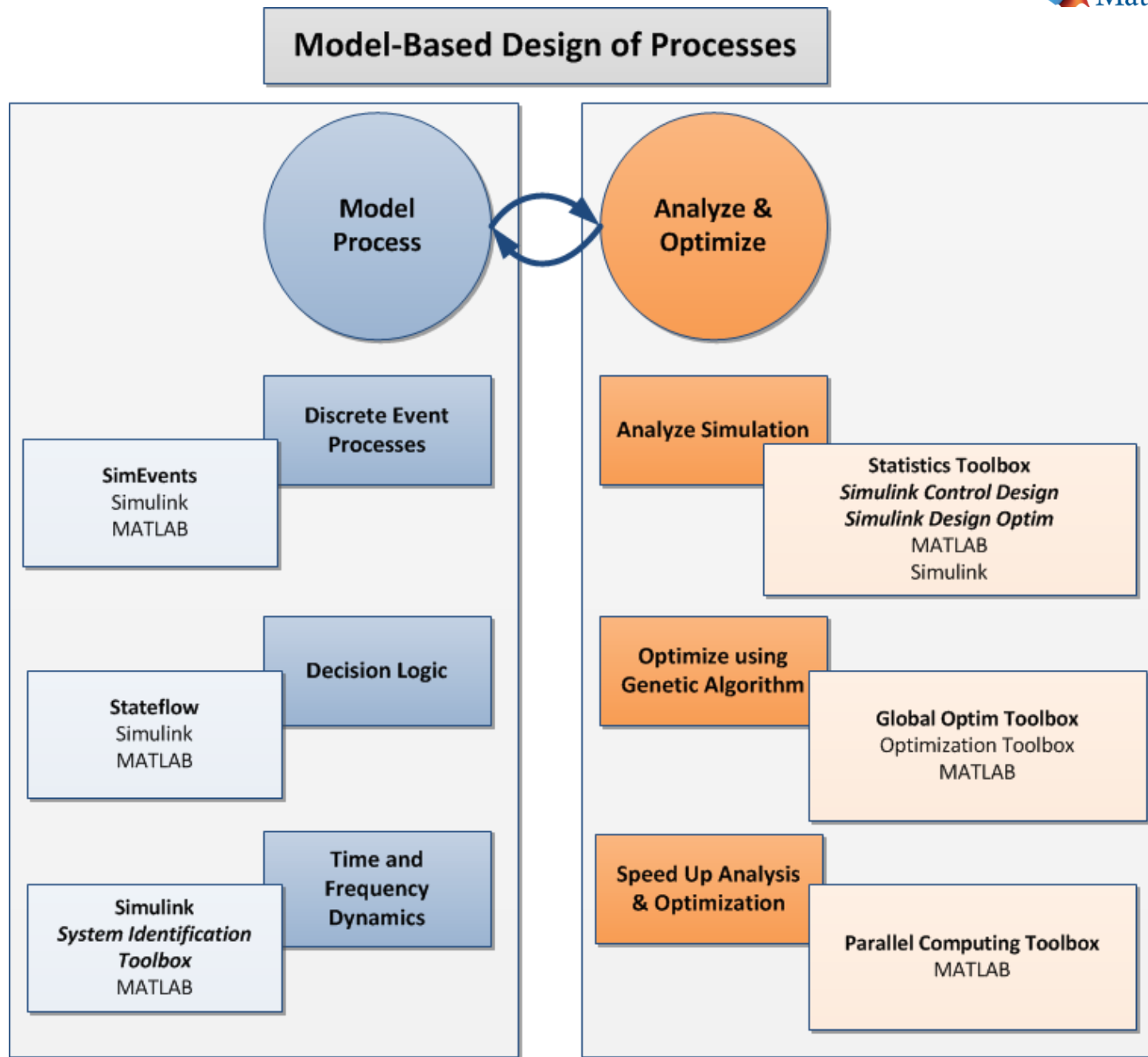
State Machines



Physical Modeling



Discrete-Event Systems



基于模型的设计过程-开发平台的特征

多域建模与多学科建模

Condition Table

Description	DI	DO	D1	D2	D3	D4	D5	D6	D7
Hydraulic system 1 Low pressure (Left Outer Line)	T	T	F	F	-	-	-	-	-
Left Outer actuator position failed	-	-	T	T	-	-	-	-	-
Hydraulic system 2 Low pressure (Inner line)	F	-	F	-	T	-	-	-	-
Left Inner actuator position failed	F	-	F	-	-	T	-	-	-

Action Table

#	Description	Action
1	Default - All ok, do nothing.	Default:
2	Hydraulic System 1 Failure, from off Left Outer Actuator	off_Low1;
3	Left Outer Actuator Failure, Isolate Left Outer Actuator	isolate_Low1;
4	Hydraulic System 2 Failure, from off Left Inner Actuator	off_Low2;
5	Left Inner Actuator Failure, Isolate Left Inner Actuator	isolate_Low2;

```

MATLAB Function
36
37 Bearinghat = atan2(xhat(3),xhat(1));
38
39 % 4 b). Compute observation vector y and linearized measurement matrix M
40 yhat = [Rangehat;
41         Bearinghat];
42 M = [ cos(Bearinghat)           0 sin(Bearinghat)           0
43      -sin(Bearinghat)/Rangehat 0 cos(Bearinghat)/Rangehat 0 ];
44
45 % 4 c). Compute residual (Estimation Error)
46 residual = meas - yhat;
47
48 % 5. Compute Kalman Gain:
49 W = P*M'*inv(M*P*M' + R);
50
51 % 6. Update estimate
52 xhat = xhat + W*residual;
53
54 % 7. Update Covariance Matrix
55 P = (eye(4)-W*M)*P*(eye(4)-W*M)' + W*R*W';
56
57 xhatOut = xhat;
    
```

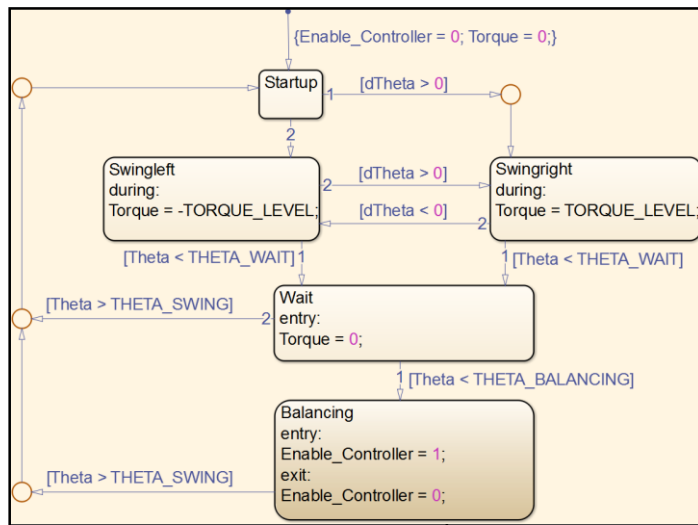
基于模型的设计过程——多种描述方式

```
function [symbols, weights] = gainctrl(rxsig, train)
% 1-tap adaptive equalizer using LMS or RLS algorithm

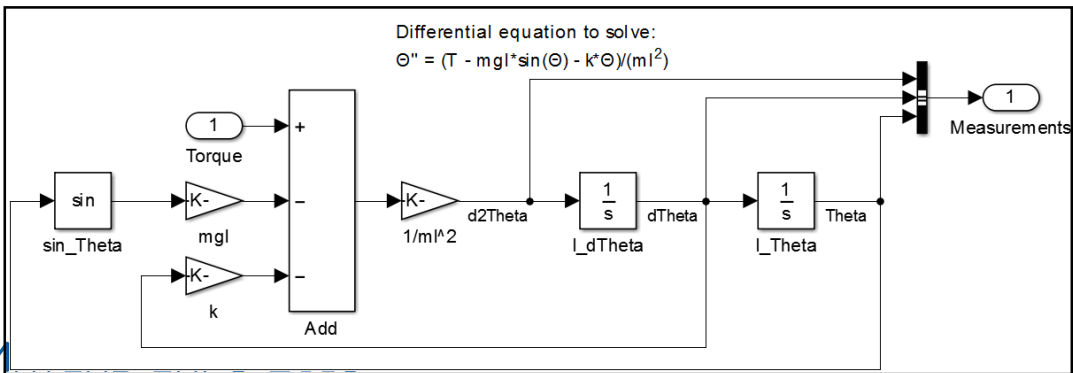
% Equalizer settings
lambda = 0.99;
Delta = 0.1+0i;
weights = 0+0i;

for n = 1:length(rxsig)
    u = rxsig(n); % received sample
    y = conj(weights) * u;
    if n<=length(train)
        d = train(n);
    else
        d = detect(real(y)) + 1j*detect(imag(y));
    end
    % Single-tap RLS
    Delta = 1/(lambda/Delta + u*conj(u));
    G = Delta * u;
    e = d - y; % symbol estimation error
    weights = weights + G*conj(e);
    symbols(n) = y;
end
```

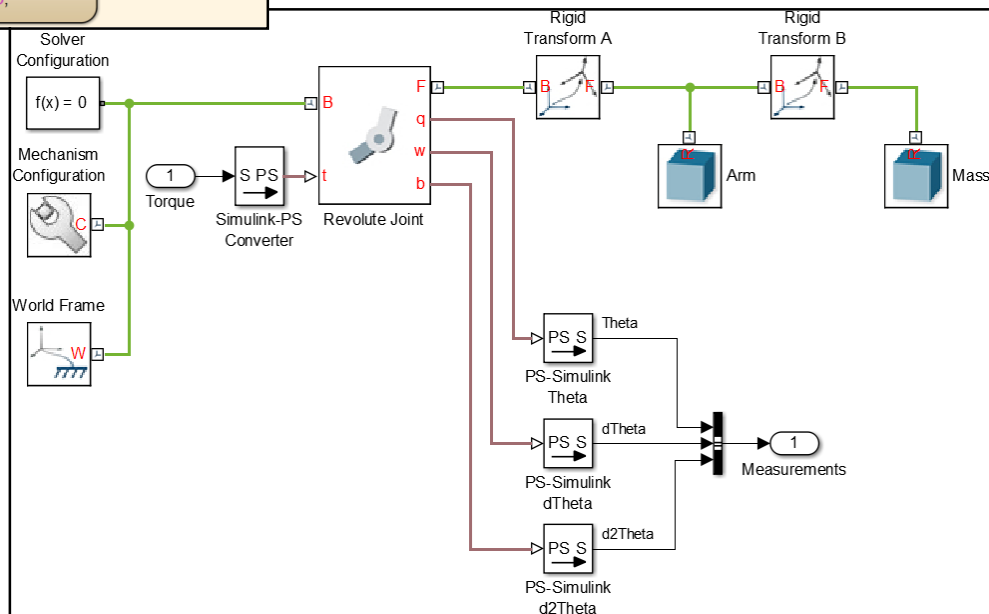
MATLAB



Stateflow



Simulink



Simscape

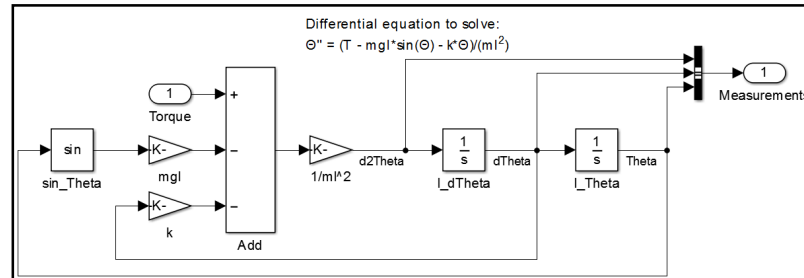
基于模型的设计过程——统一的自动代码生成环境

```
function [symbols, weights] = gainctrl(rxsig, train)
% 1-tap adaptive equalizer using LMS or RLS algorithm

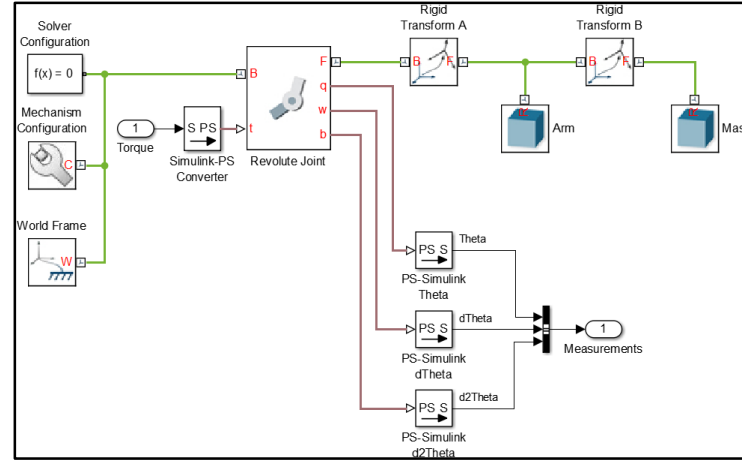
% Equalizer settings
lambda = 0.99;
Delta = 0.1+0i;
weights = 0+0i;

for n = 1:length(rxsig)
    u = rxsig(n); % received sample
    y = conj(weights) * u;
    if n<=length(train)
        d = train(n);
    else
        d = detect(real(y)) + 1j*detect(imag(y));
    end
    % Single-tap RLS
    Delta = 1/(lambda/Delta + u*conj(u));
    G = Delta * u;
    e = d - y; % symbol estimation error
    weights = weights + G*conj(e);
    symbols(n) = y;
end
```

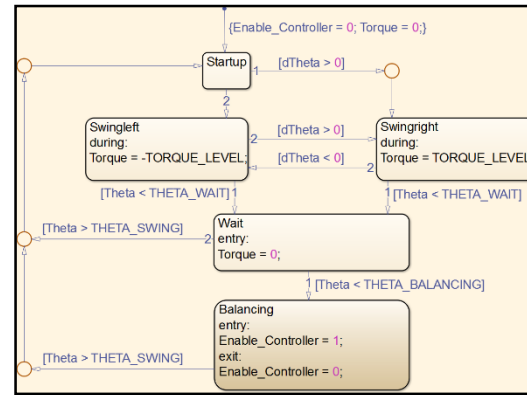
MATLAB



Simulink



Simscape



Stateflow

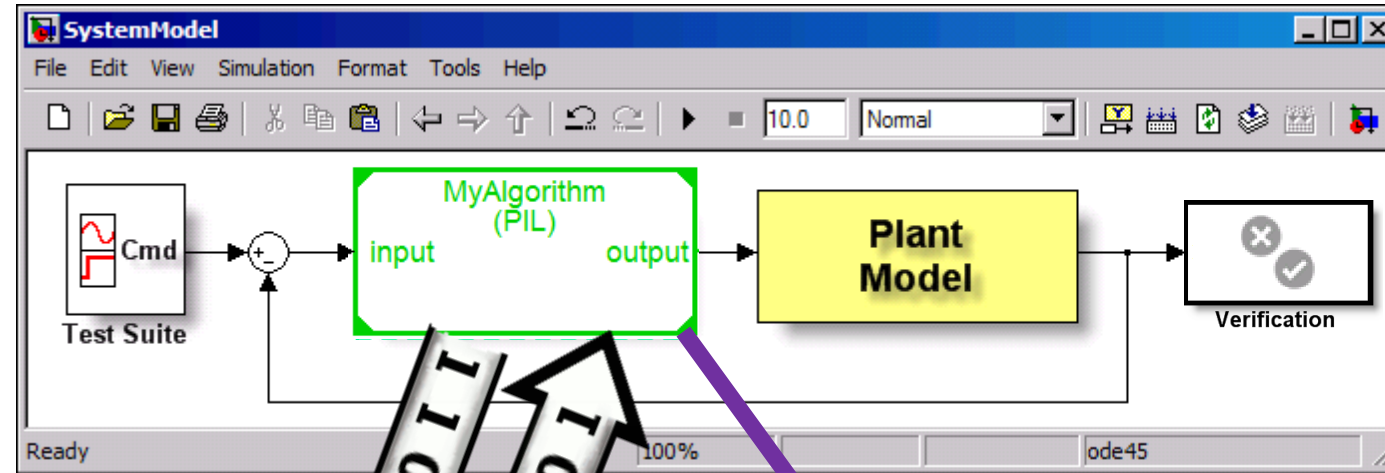
Unified representation

Mathematical engines

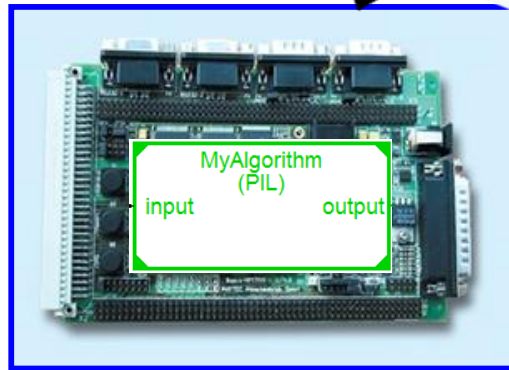
- C Code
- C++ Code
- HDL Code
- PLC Code
- Find design errors
- Test cases
- Fixed-point auto scaling

在环验证 In-Loop Verification

SIL and PIL



Non-Real-Time Synchronization
with Host at Each Time Step

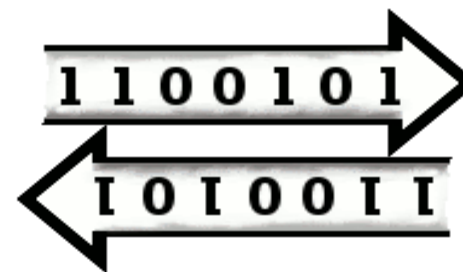
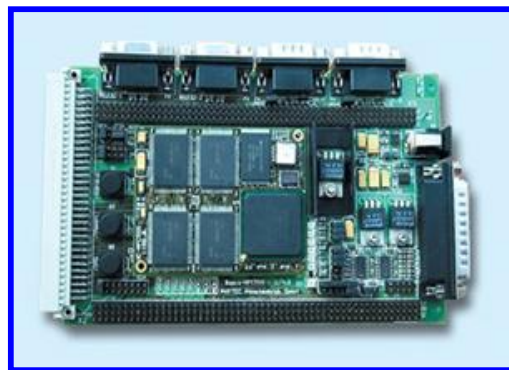
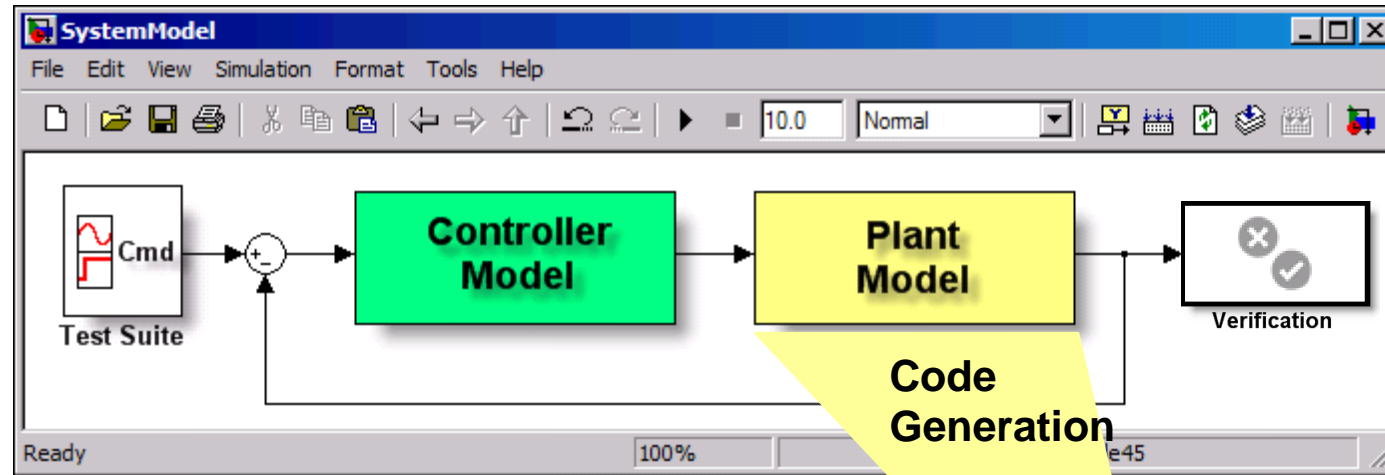


Execution History

- Logged signal results comparison
- Code coverage
- Execution timing

在环验证 In-Loop Verification

HIL, Rapid Prototyping



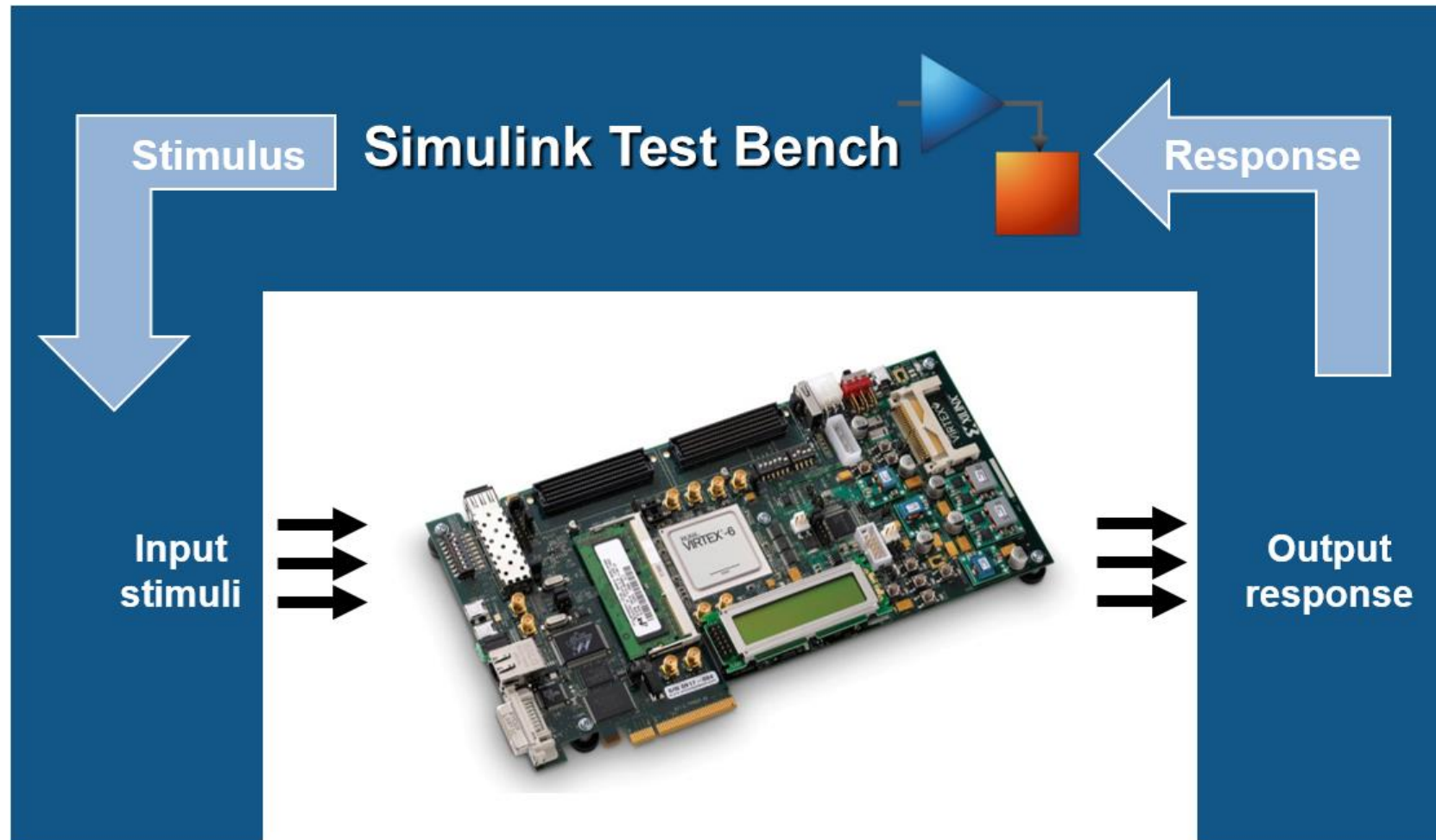
Hard Real-Time Execution

**Logging and
Tuning via Host**



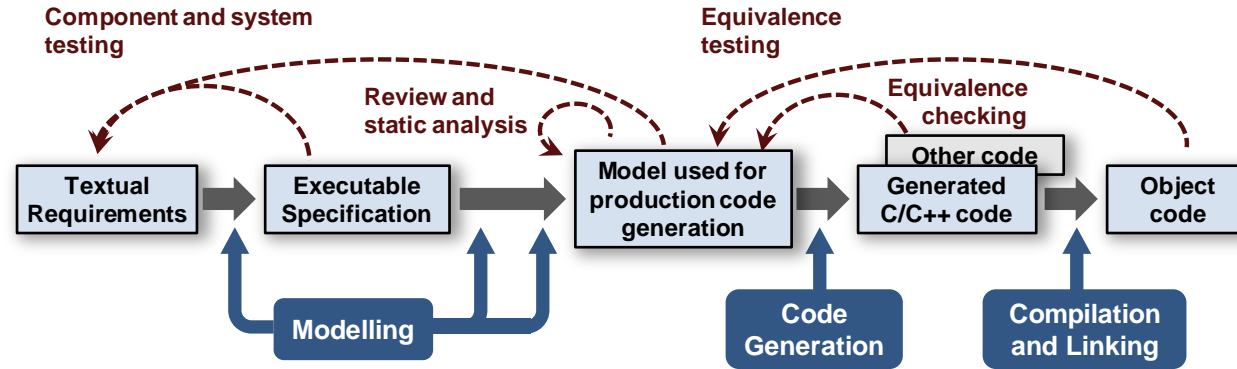
在环验证 In-Loop Verification

FIL, Test Bench Simulation

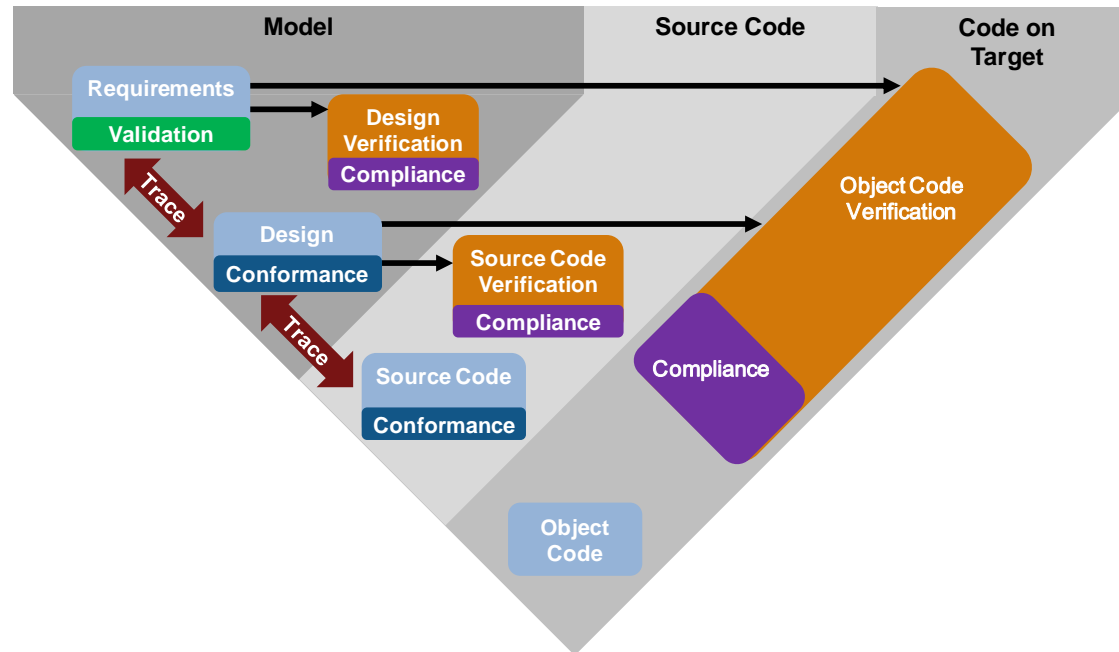


符合认证标准的工作流程

IEC 61508
 ISO 26262
 IEC 62405
 EN 50128



DO-178C
 DO-254
 DO-331
 DO-333



MBD的应用: Lockheed Martin

Fleet Modeling



System Requirements

Modeling Standards

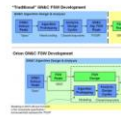
Accelerating NASA GN&C Flight Software Development

By Scott Tamblin and Joel Henry, NASA, and John Rapp, Lockheed Martin
Send e-mail to tom.friedman@nasa.gov

When the guidance, navigation, and control (GN&C) system for the Orion crew vehicle undergoes Critical Design Review (CDR), more than 90% of the flight software will already be developed—a first for NASA on a project of this scope and complexity. This achievement is due in large part to a new development approach using Model-Based Design.

Most NASA GN&C projects follow a traditional process: Domain experts and analysts specify the behavior of the core algorithms in detailed requirements documents. Following CDR, these documents are handed off to the flight software engineers for implementation into the formal flight software. Producing the specification document alone often requires years of effort, and because coding can begin only after the spec is complete, it can be years before there is any code to test.

The design and development of the GN&C flight algorithms for Orion is a partnership between NASA, Lockheed Martin, and other contractors. Model-Based Design has helped these organizations work on both the GN&C algorithm and flight software development concurrently. Simulink models serve as an executable specification from which flight software is automatically generated. As a result, the domain experts—the GN&C analysts—work directly with the executable algorithm models rather than with documents that must then be interpreted by software developers (Figure 1).



Lockheed Martin Develops a Tool for 3-D Antenna Data Visualization

By MICHAEL WOOD, LOCKHEED MARTIN SPACE SYSTEMS COMPANY

Because most spacecraft, once deployed, can never be serviced, it is vital to ensure that their communications systems meet requirements for link availability and quality. Antenna design and analysis are critical to this process.

Engineers use gain pattern analysis to determine how well coverage for the various payloads will be achieved in orbit. This is done by modeling the antenna radiation patterns and the geometry of the spacecraft in which they are installed. The resulting radiation pattern is then used to determine the coverage of the antenna.

Traditionally, antenna gain analysis has been done using 2-D plots. This is often done through a 3-D plot. However, this is often done through a 3-D plot.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Using MATLAB graphics and CDR tool, we can visualize and develop 3-D antenna radiation patterns.

Antenna Subsystem Design

A MODEL CHECKING EXAMPLE – SOLVING SUDOKU USING SIMULINK DESIGN VERIFIER

By [Walter A. Storm](#), Lockheed Martin Aeronautics Company

Technology Evaluation

...to understand application of formal methods— an example based on the popular game Sudoku. I ty of this technology as implemented within ent for Model-Based Design. The overarching theme to consider is an analogy of the game to real-world constraint problems.

INTRODUCTION

Sudoku is a logic-based number-placement puzzle. The objective is to populate a 9x9 grid so that each column, row, and 3x3 box contains a single instance of the digits 1-9. The game starts with a partially completed grid, and the solution to the puzzle is the arrangement of digits that meet the single-instance criteria.

The Sudoku grid is analogous to the complex finite state machines (often implemented as hybrid control automata) that are responsible for executing the modes and behaviors of emerging software systems. As the grid is populated, the temporary switching and storing of digits is representative of the various states and modes that the system can enter at any given time. The alteration of the grid is a result of the environment in which the system operates.

The strategy behind using a model checker to solve a Sudoku puzzle is this: formulate a logical proposition that suggests, given an initial state, no cases exist that meet all Sudoku requirements. The resultant counterexample is a solution to the puzzle.

FORMALIZING THE REQUIREMENTS

Our approach to the Sudoku example is to first formalize the requirements of the game as a graphical model in Simulink (Figure 1). This formalization consists of an initial board and an input vector that represents the puzzle's environment—essentially, all the blank spaces to which a digit can be assigned.

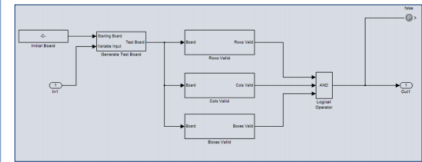
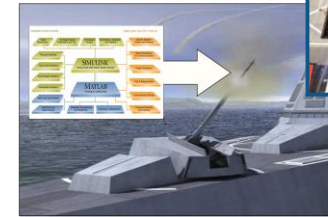


FIGURE 1. MODEL OF SUDOKU REQUIREMENTS.

Page | 2

Design, Verification and Deployment

A User's Experience with Simulink® and Stateflow® Real-Time Embedded Applications



James E. Craft and Bob Risk, Lockheed Martin Missiles and Fire Control

Acceptance and Deployment

System-level Integration and Test

HDL Verification

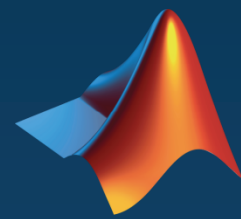
em Design

Subsystem Integration and Test

Production Code Generation

Subsystem Implementation

Simulate early & often



MathWorks®

Accelerating the pace of engineering and science