

# MATLAB EXPO 2019

Sviluppo di un sistema di gestione  
delle batterie con Simulink

Maurizio Dalbard  
Aldo Caraceto

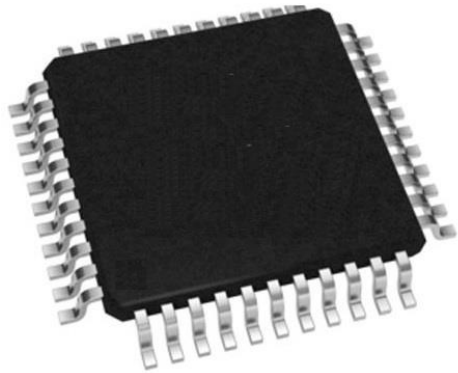






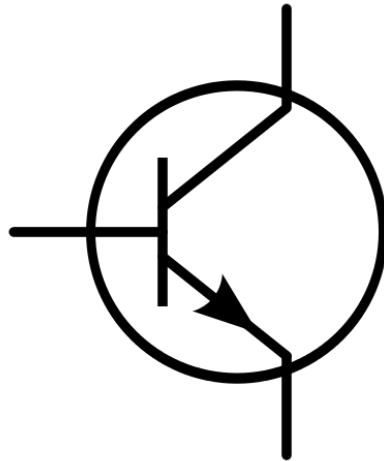
# Battery System

## Battery Management System (BMS)



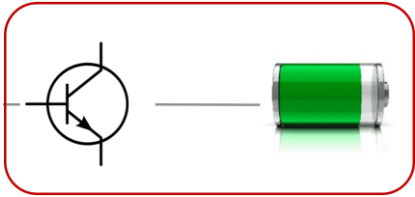
- Software Architecture
- SW Module Design
- Test, Verification & Validation
- BMS Integration Testing

## Battery and Power Electronics



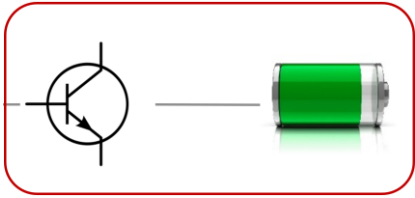
- Equivalent Circuit for Battery Cell
- Parameter Estimation
- Battery Module Configuration
- Charging & Balancing

# Battery and Power Electronics

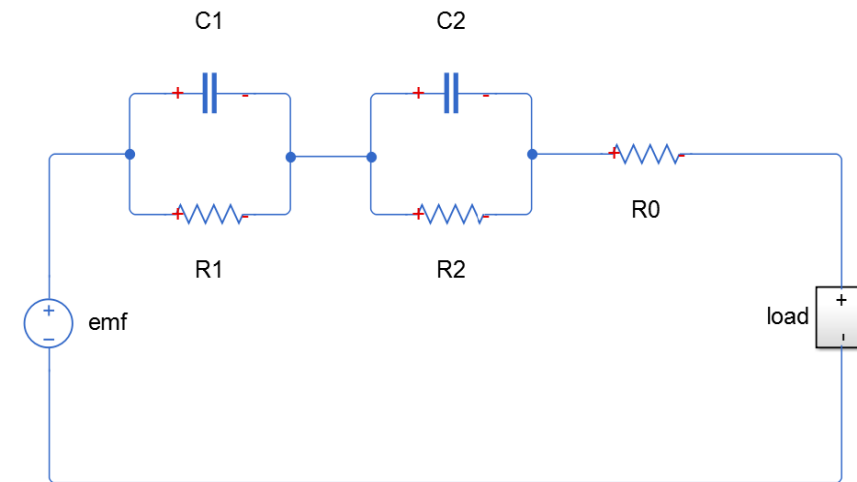


- Equivalent Circuit for Battery Cell
- Parameter Estimation
- Battery Module Configuration
- Charging & Balancing

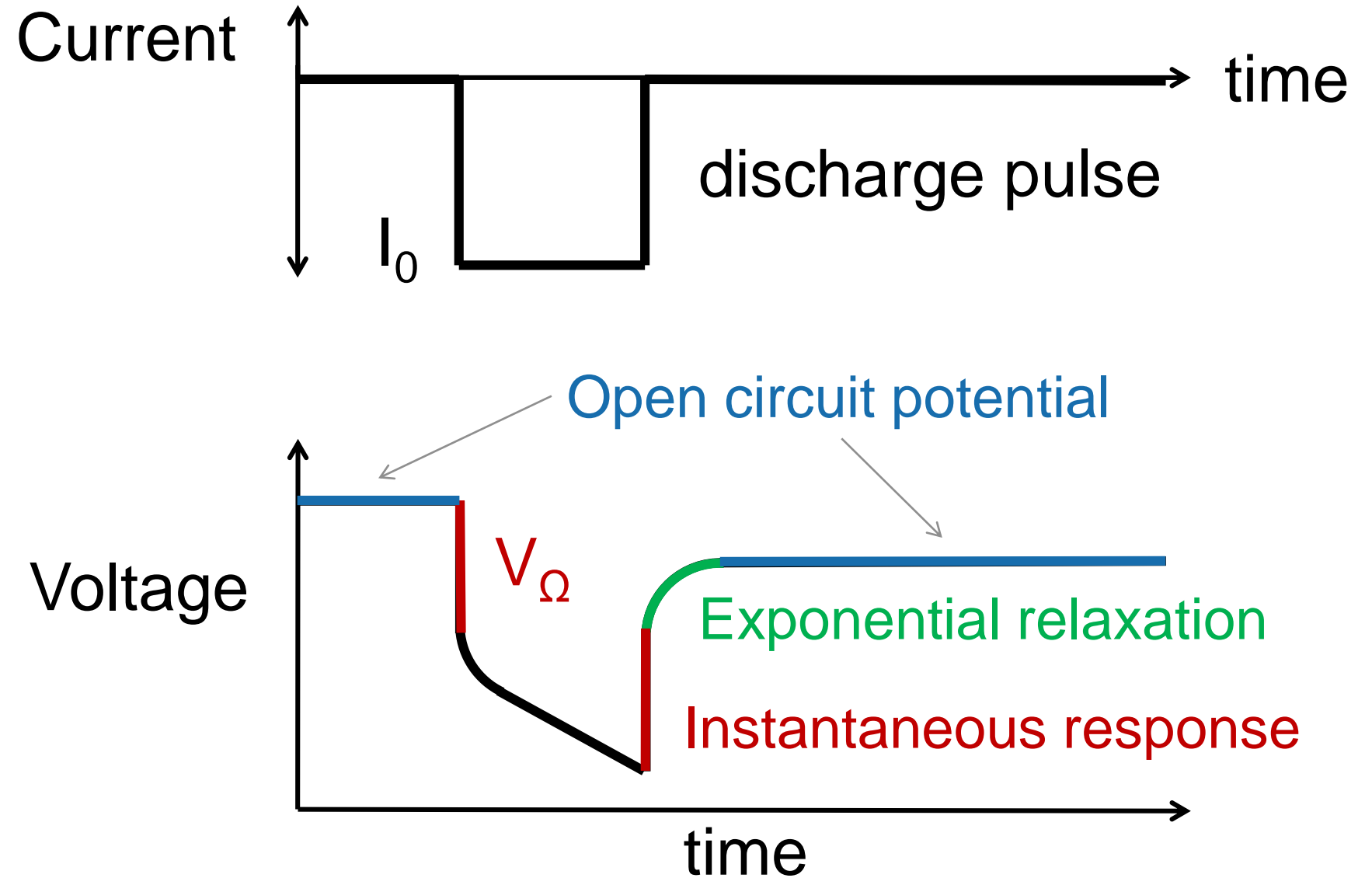
# Battery and Power Electronics



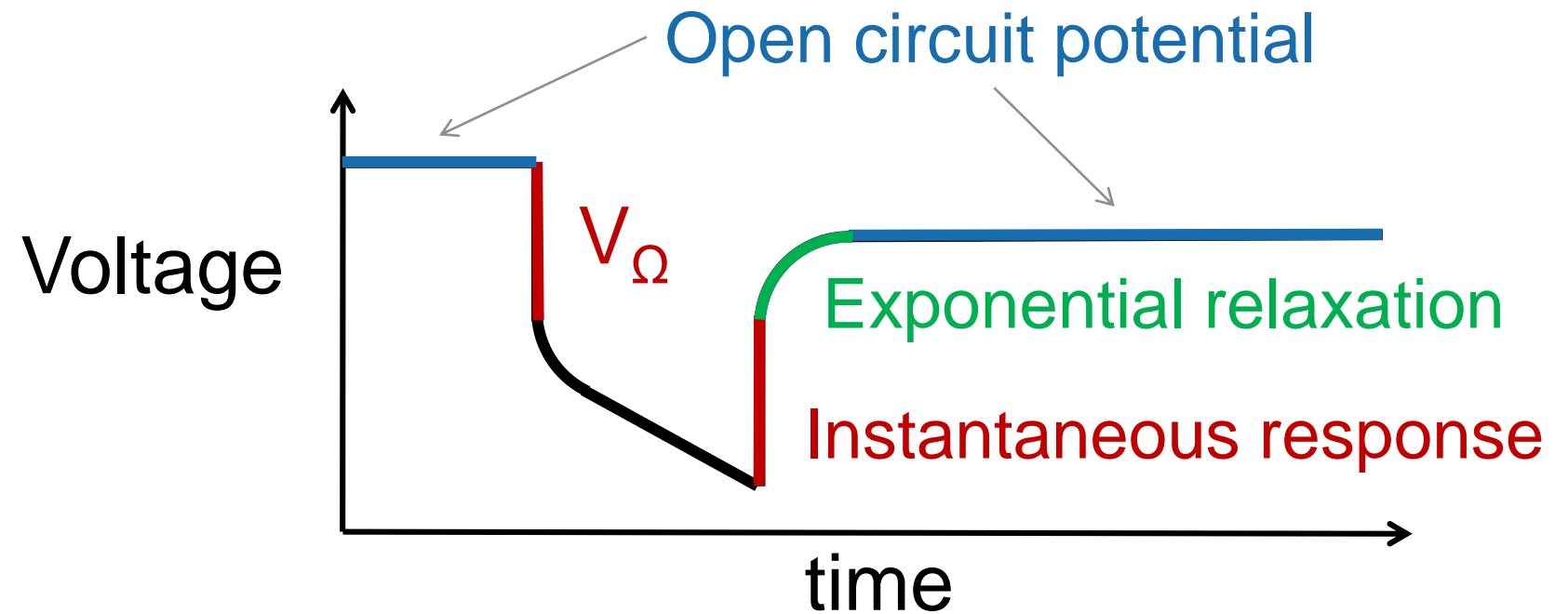
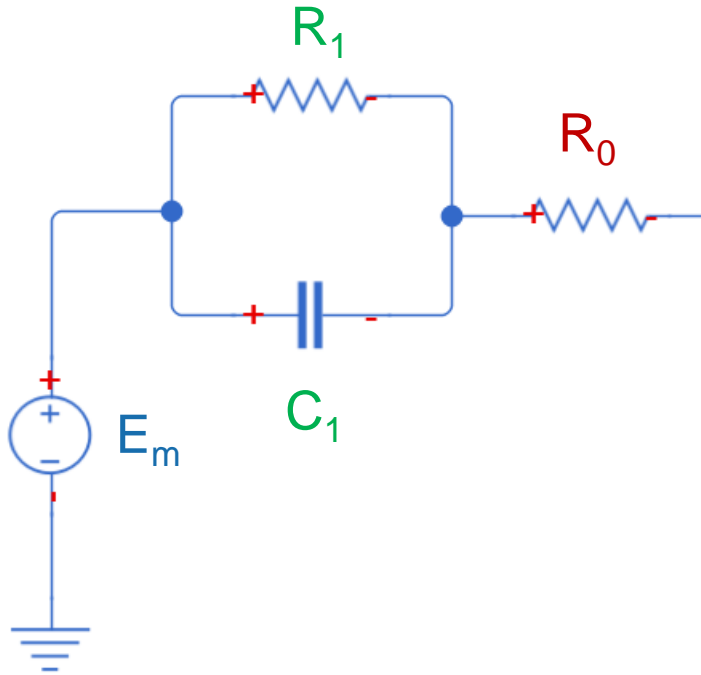
- **Equivalent Circuit for Battery Cell**
- Parameter Estimation
- Battery Module Configuration
- Charging & Balancing



# Pulse Discharge

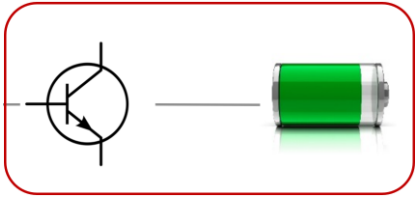


# Equivalent Circuit

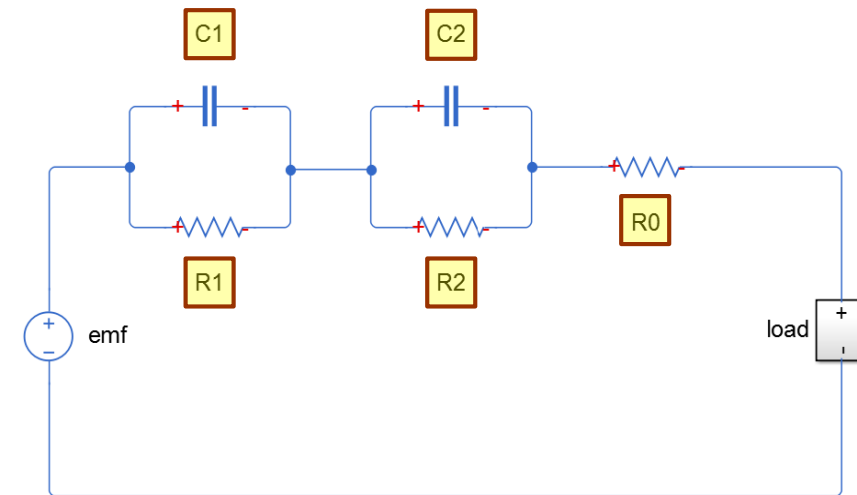




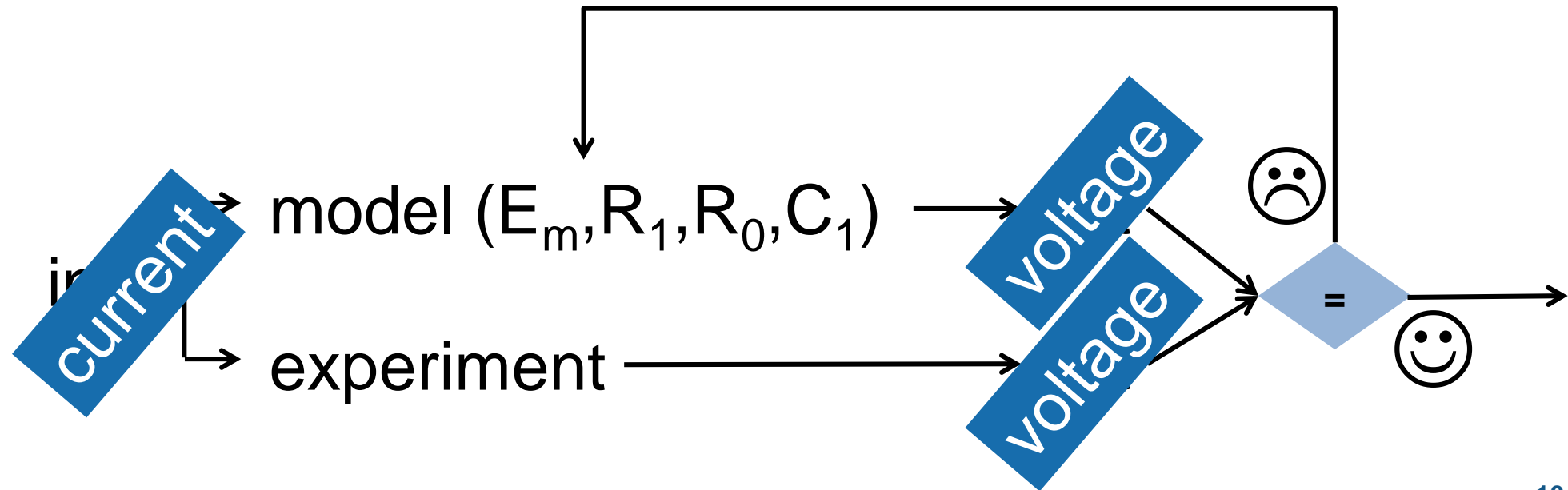
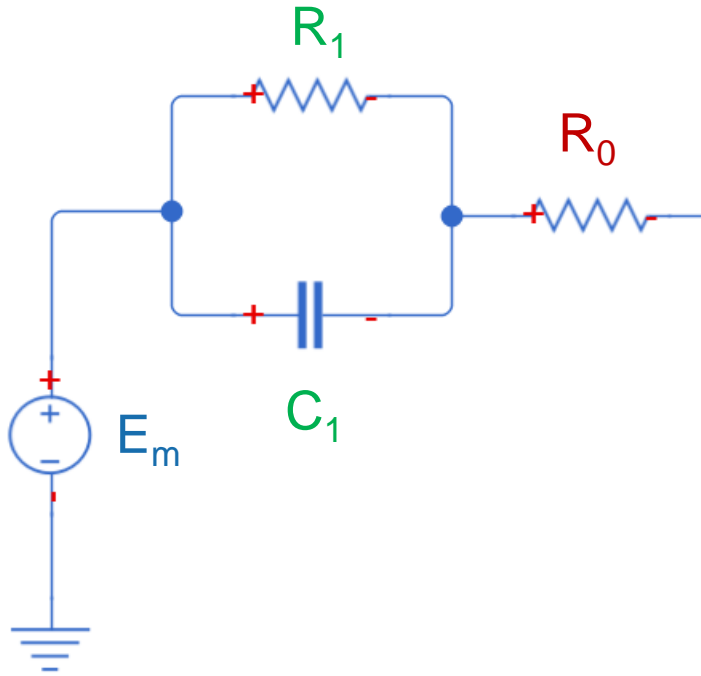
# Battery and Power Electronics



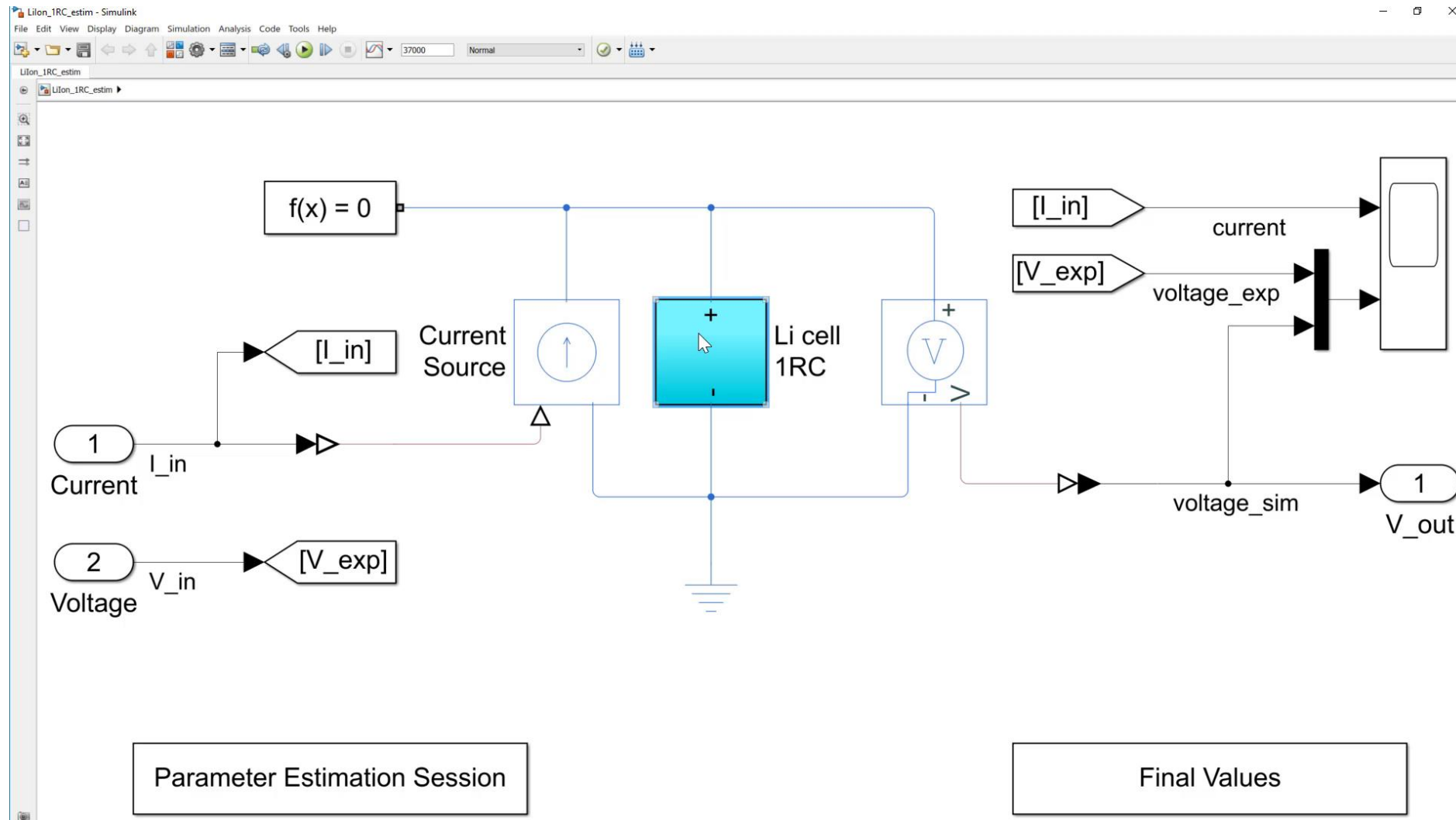
- Equivalent Circuit for Battery Cell
- **Parameter Estimation**
- Battery Module Configuration
- Charging & Balancing



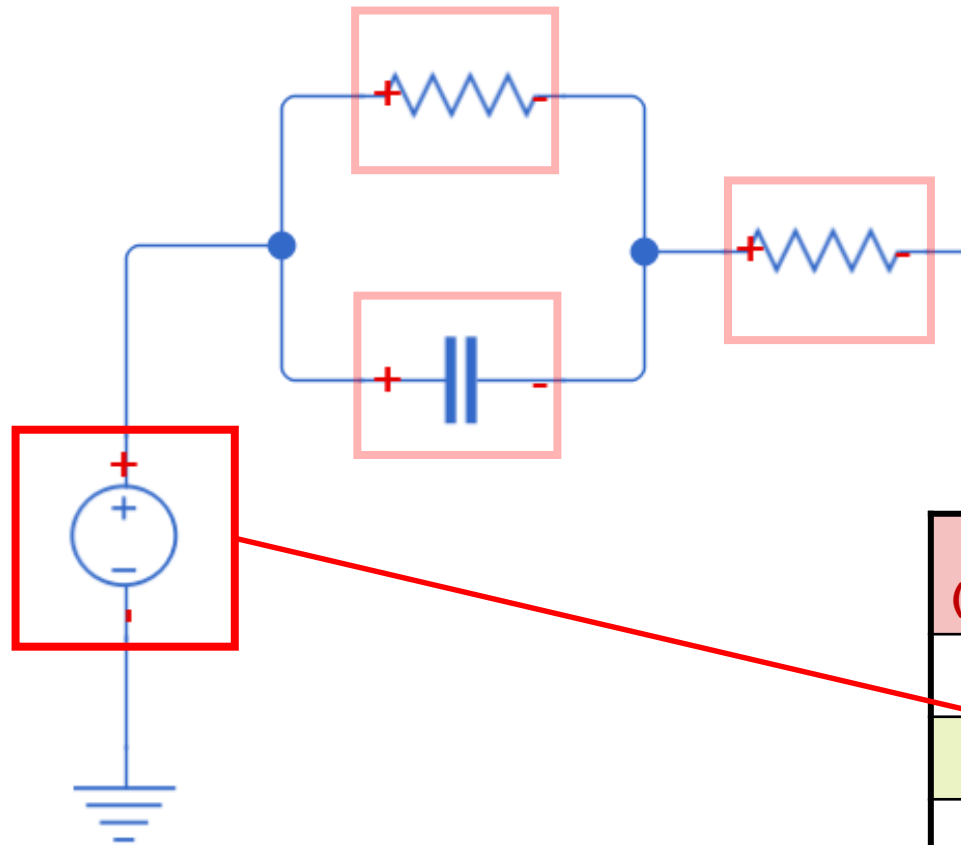
# Parameter Estimation



# Estimate Parameters

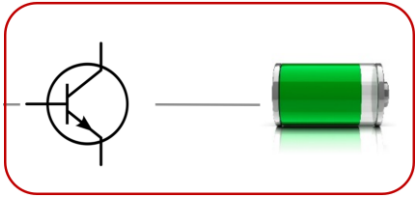


# Look-up Tables



Em (Volts)	SOC 1	SOC 0.9	SOC 0.8	...	SOC 0
5°C					
20°C					
40°C					

# Battery and Power Electronics

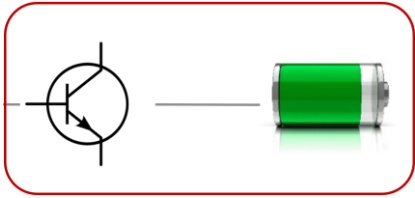


- Equivalent Circuit for Battery Cell
- Parameter Estimation
- **Battery Module Configuration**
- Charging & Balancing

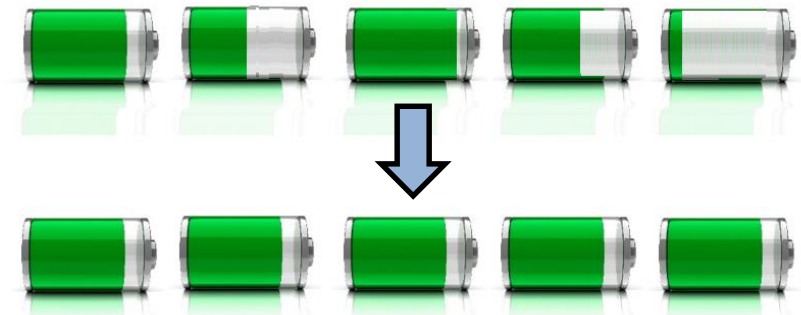




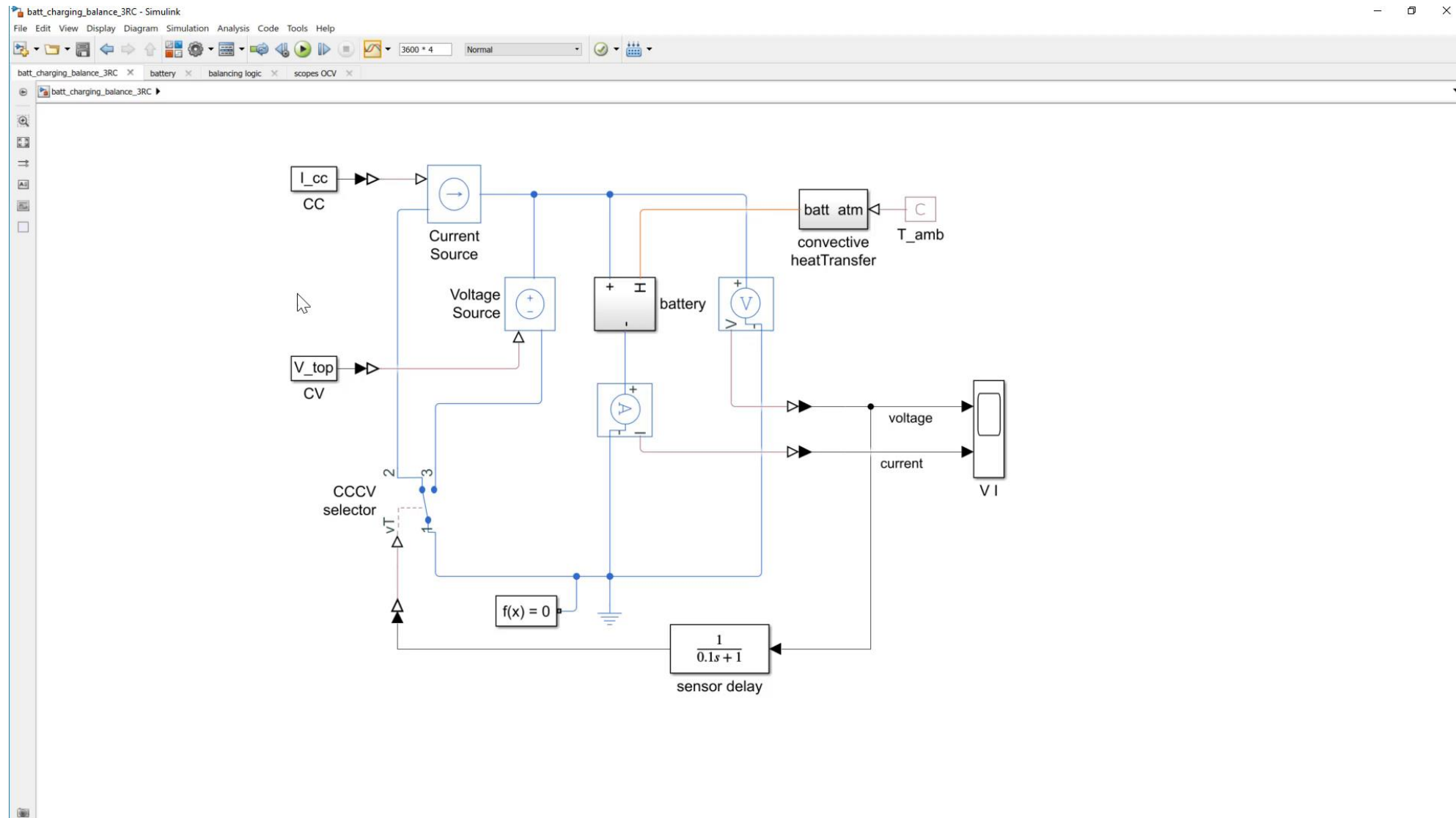
# Battery and Power Electronics



- Equivalent Circuit for Battery Cell
- Parameter Estimation
- Battery Pack Configurations
- **Charging & Balancing**

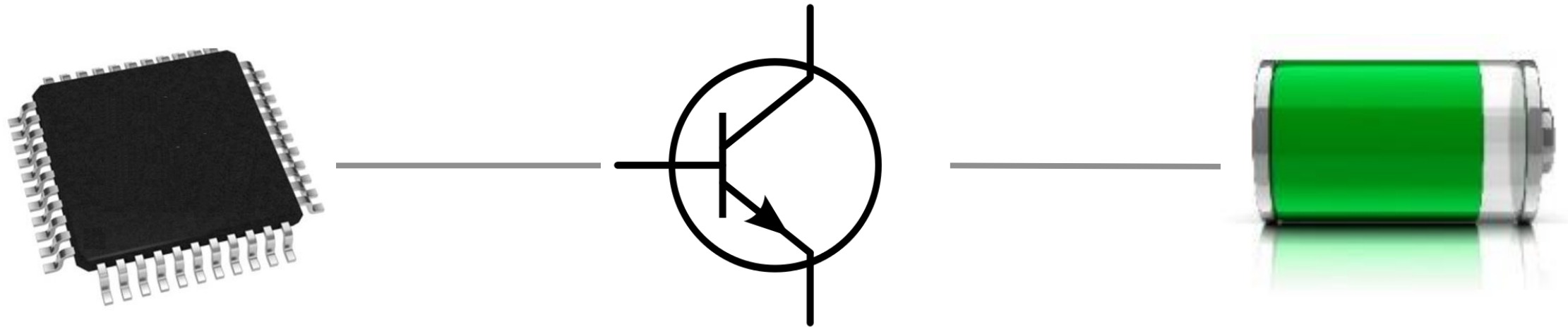


# Charging & Balancing

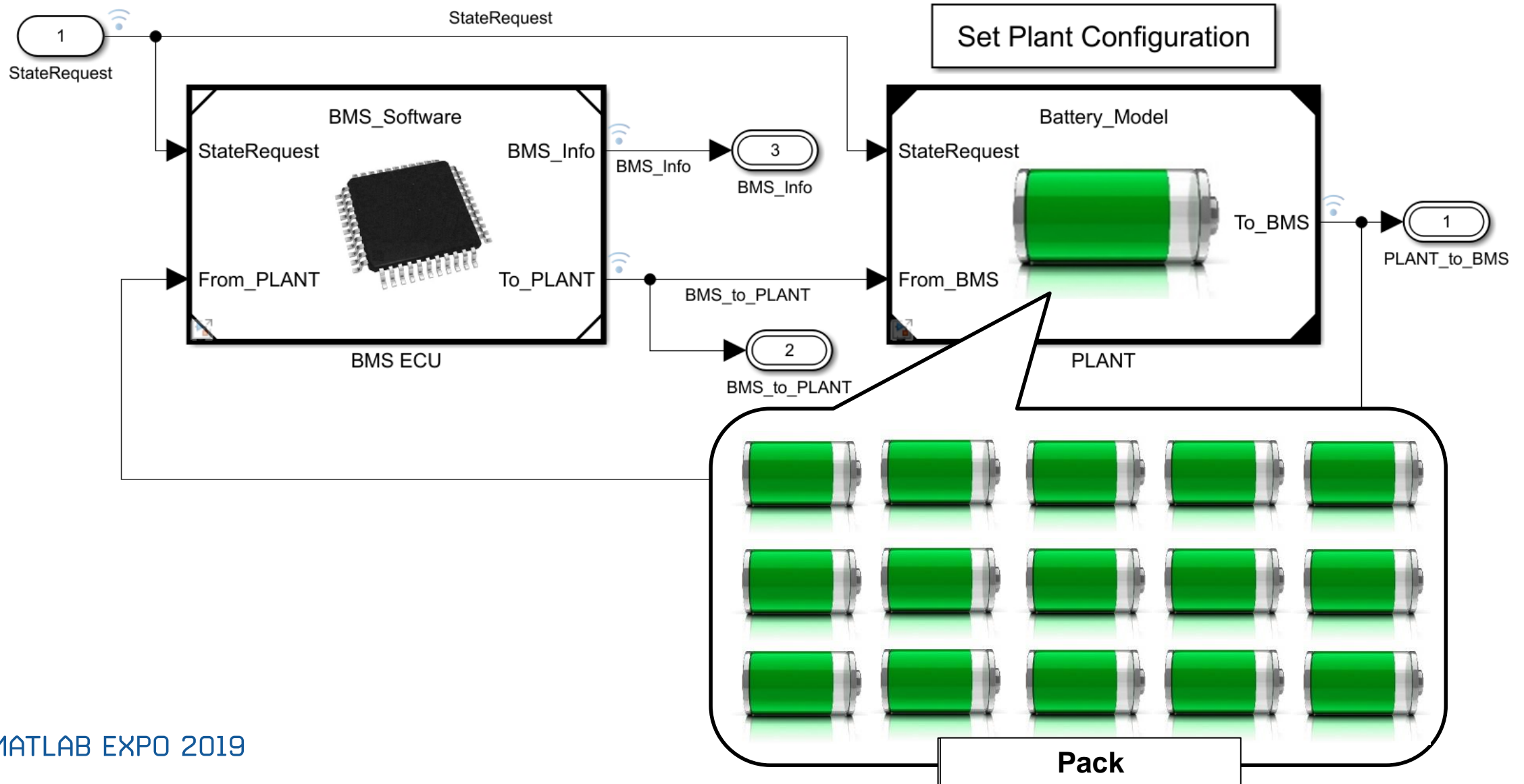




# Battery Management System

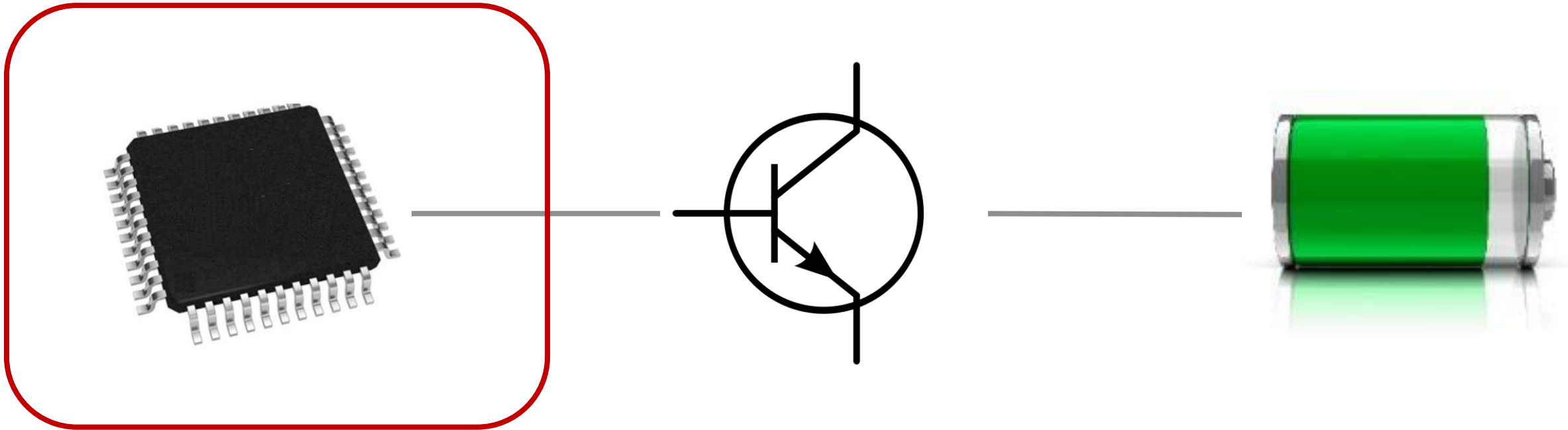


# Battery Management System – Professional Simulink Model



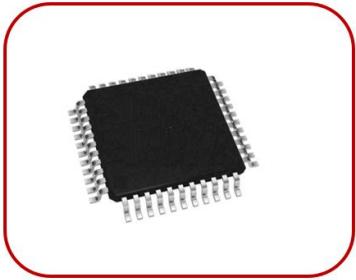
# Battery Management System

## Battery Management System (BMS)

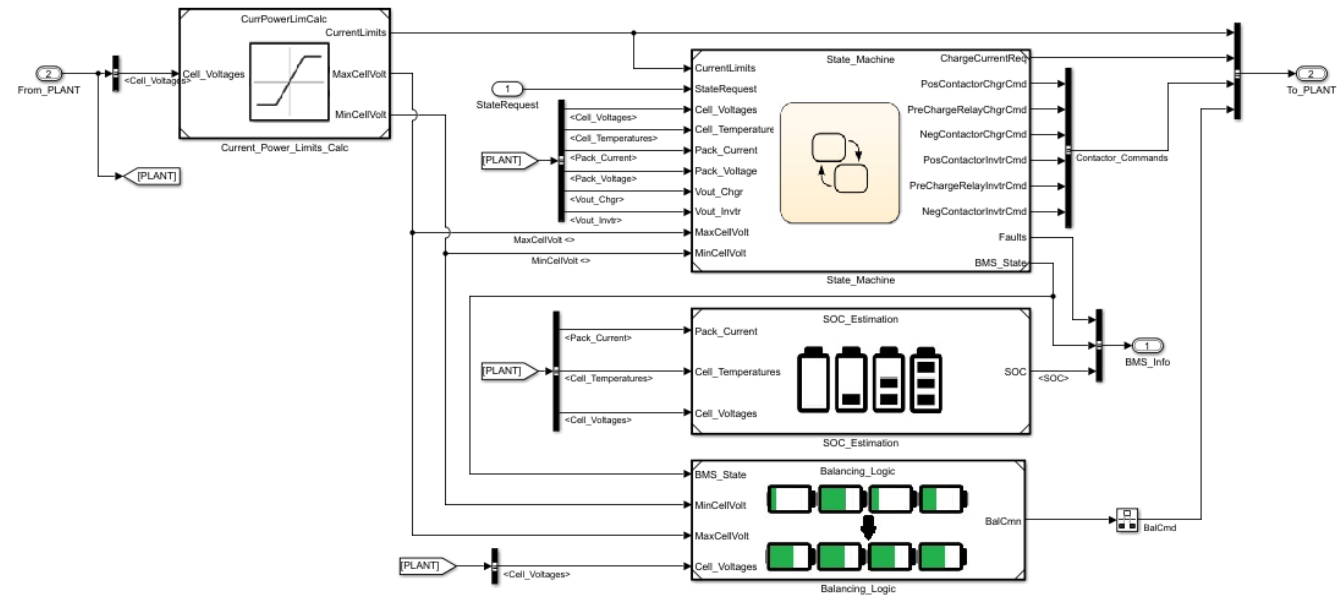


- Software Architecture
- SW Module Design
- Test, Verification & Validation
- BMS Integration Testing

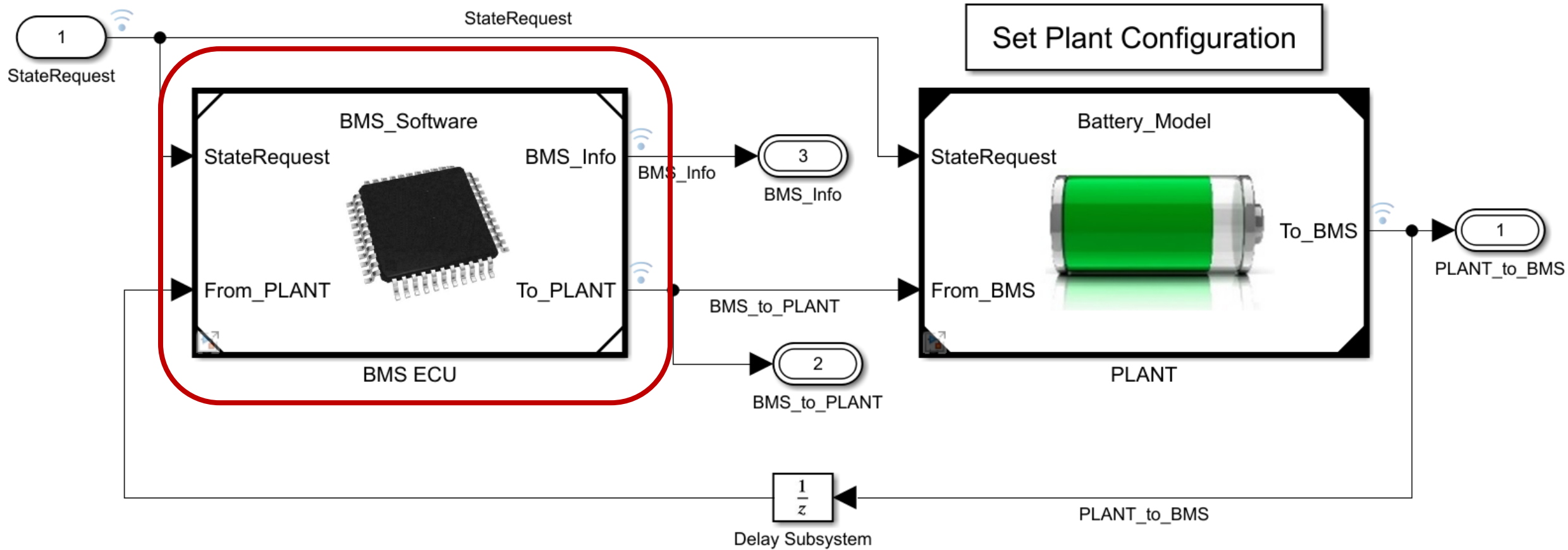
# Battery Management System (BMS)



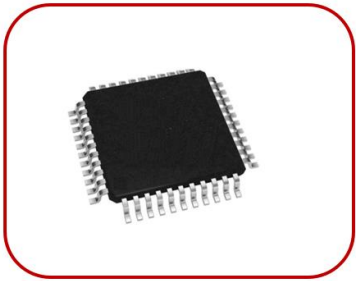
- **Software Architecture**
- **SW Module Design**
- **Test, Verification & Validation**
- **BMS Integration Testing**



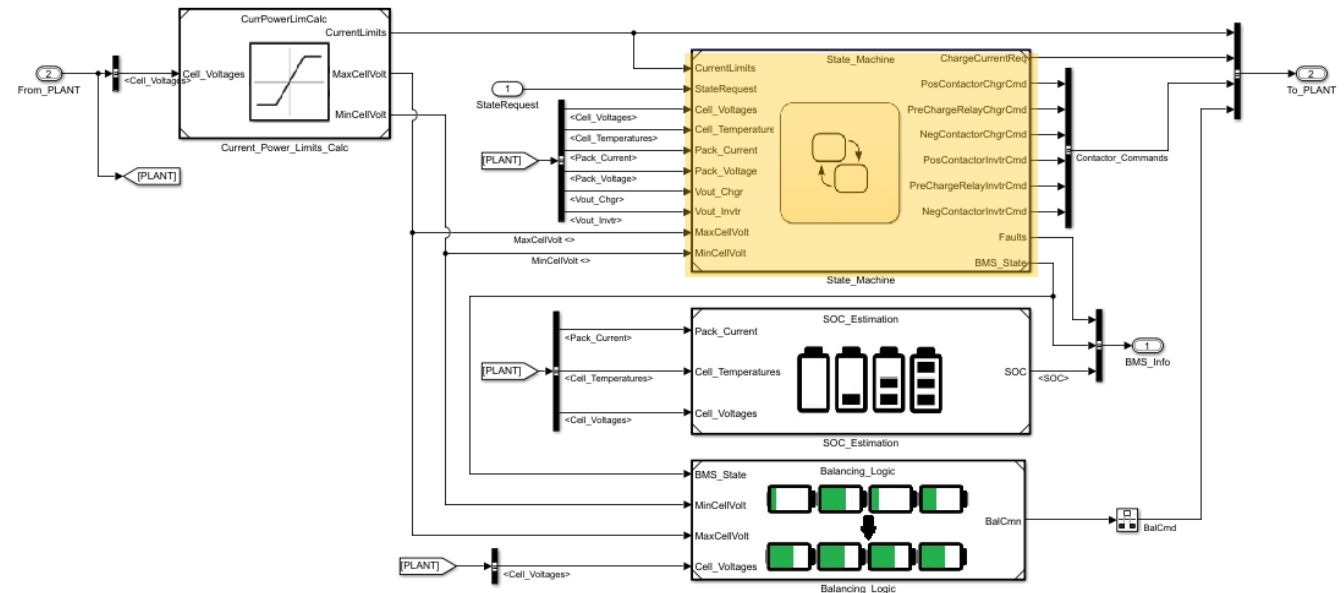
# Battery Management System – Professional Simulink Model



# Battery Management System (BMS)



- Software Architecture
- SW Module Design
- Test, Verification & Validation
- BMS Integration Testing



# State Machine Logic

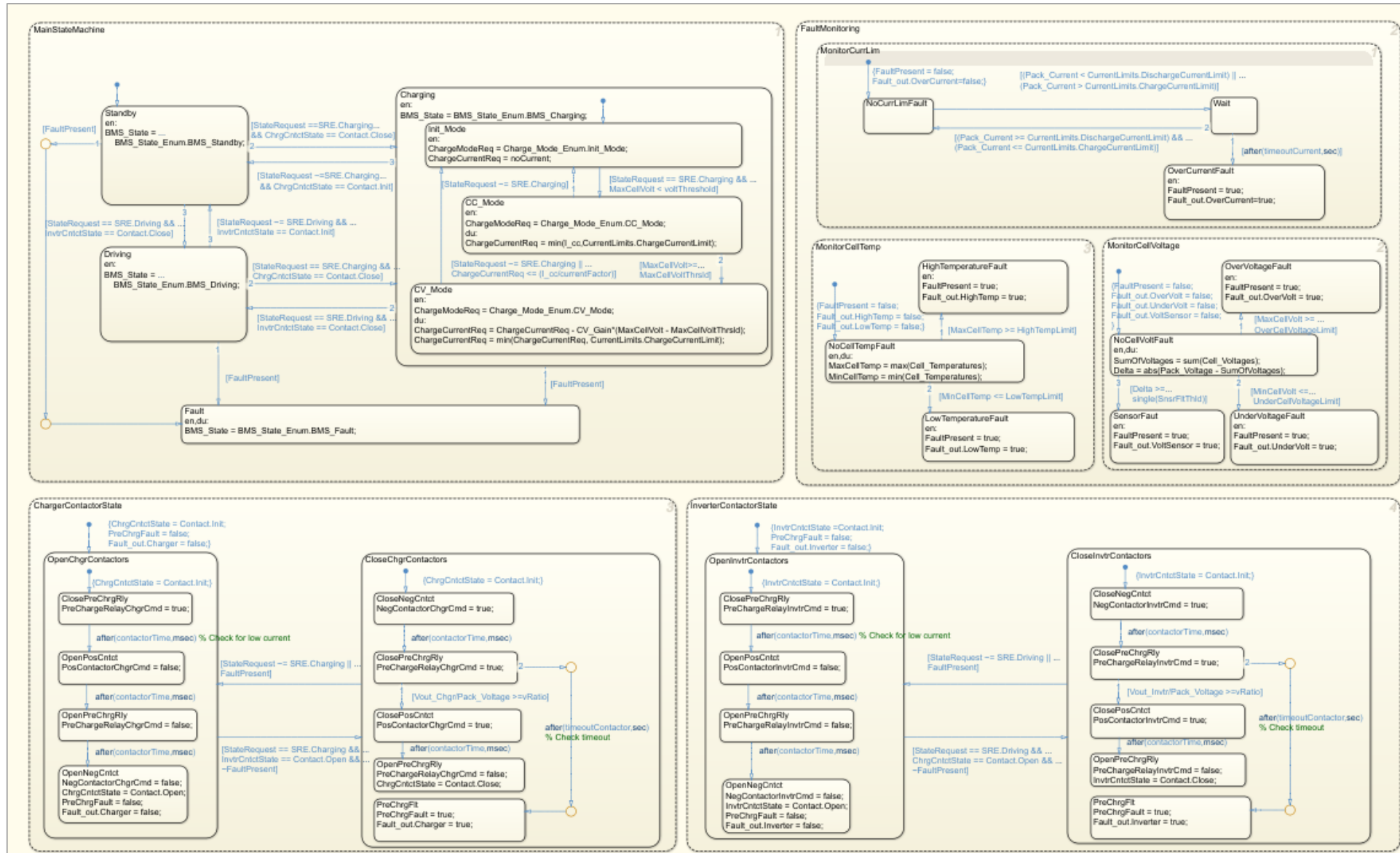
**BMS State  
and  
Charging Mode**

**Fault  
Monitoring**

**Charger  
Contactors Management**

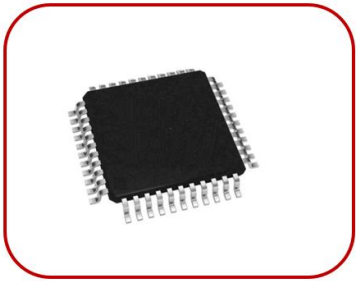
**Inverter  
Contactors Management**

# State Machine Logic in Stateflow

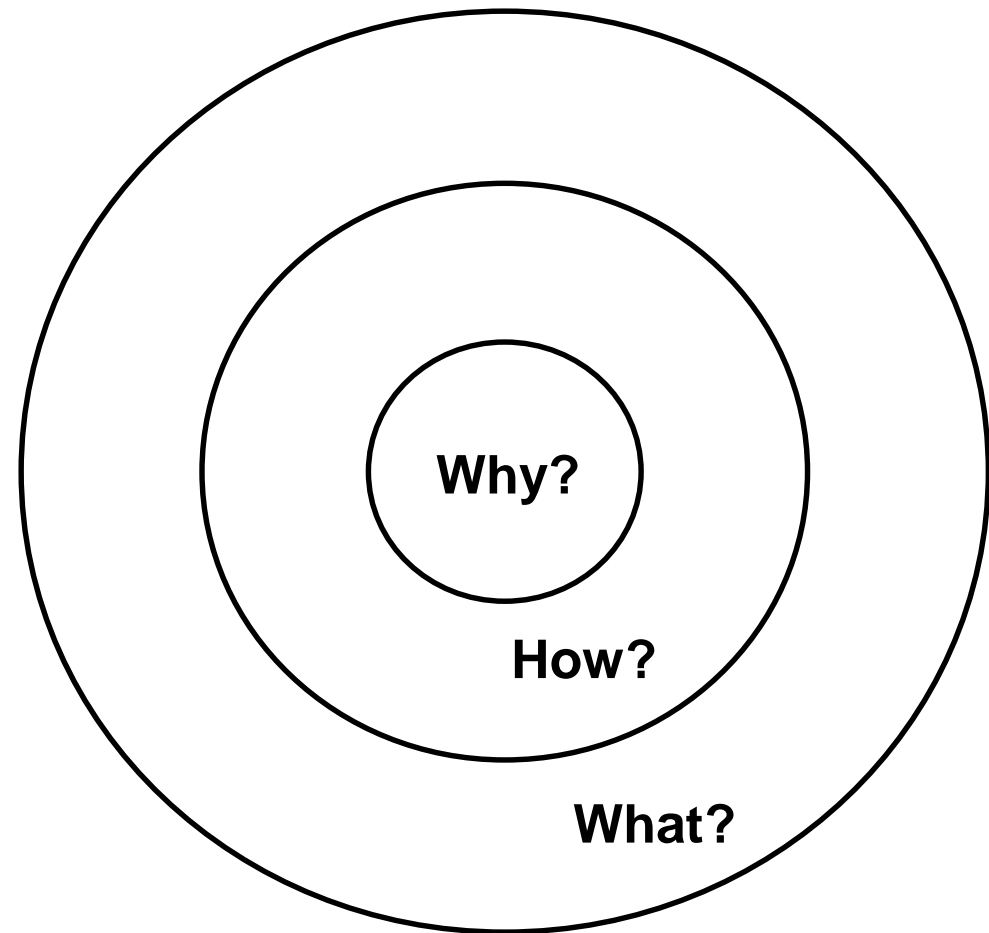




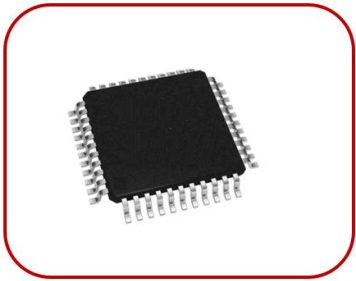
# Battery Management System (BMS)



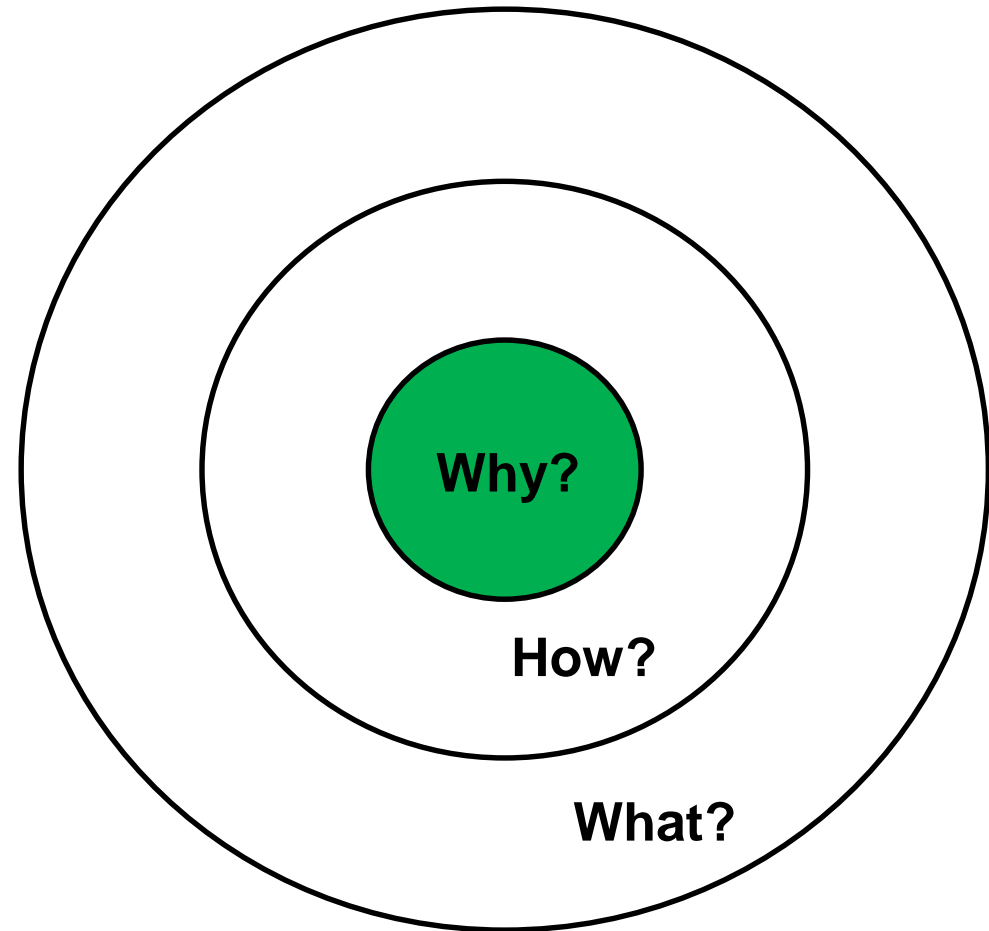
- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Integration Testing



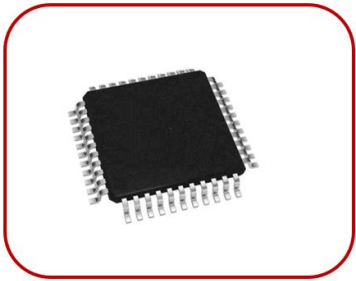
# Battery Management System (BMS)



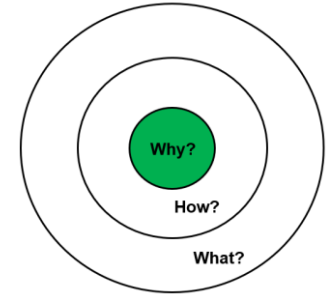
- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Testing



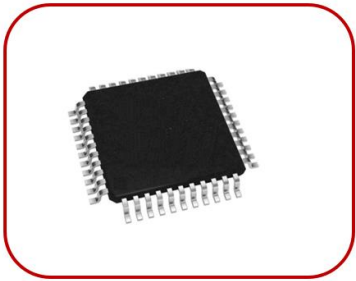
# Battery Management System (BMS)



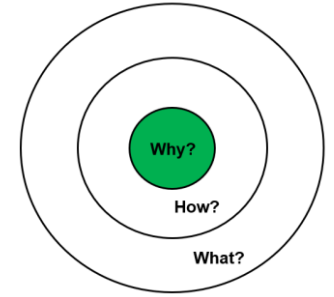
- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Integration Testing



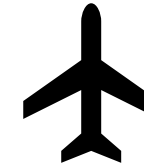
# Battery Management System (BMS)



- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Integration Testing



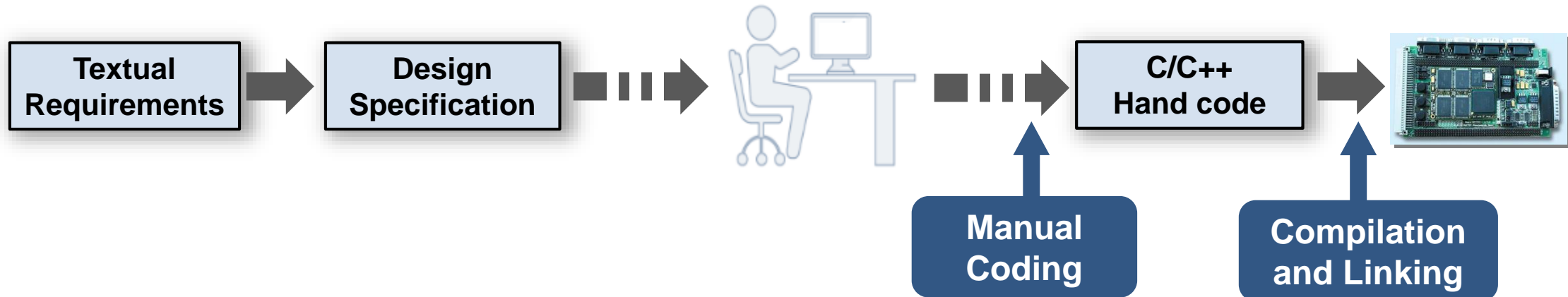
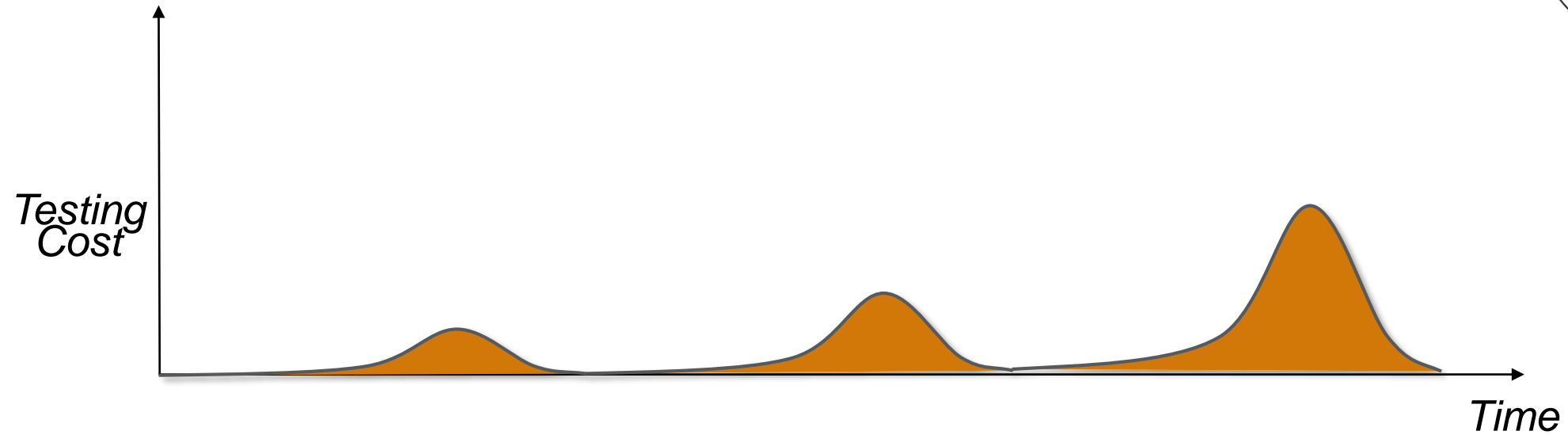
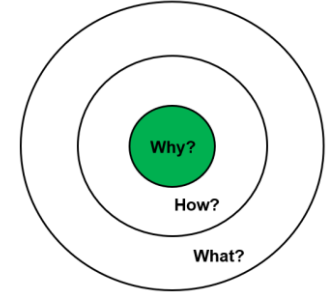
## Safety Critical System



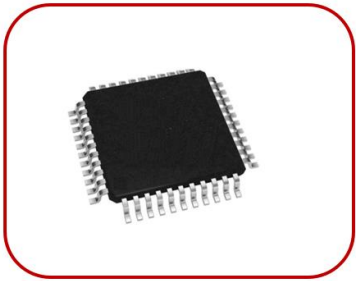
## Functional Safety Certification



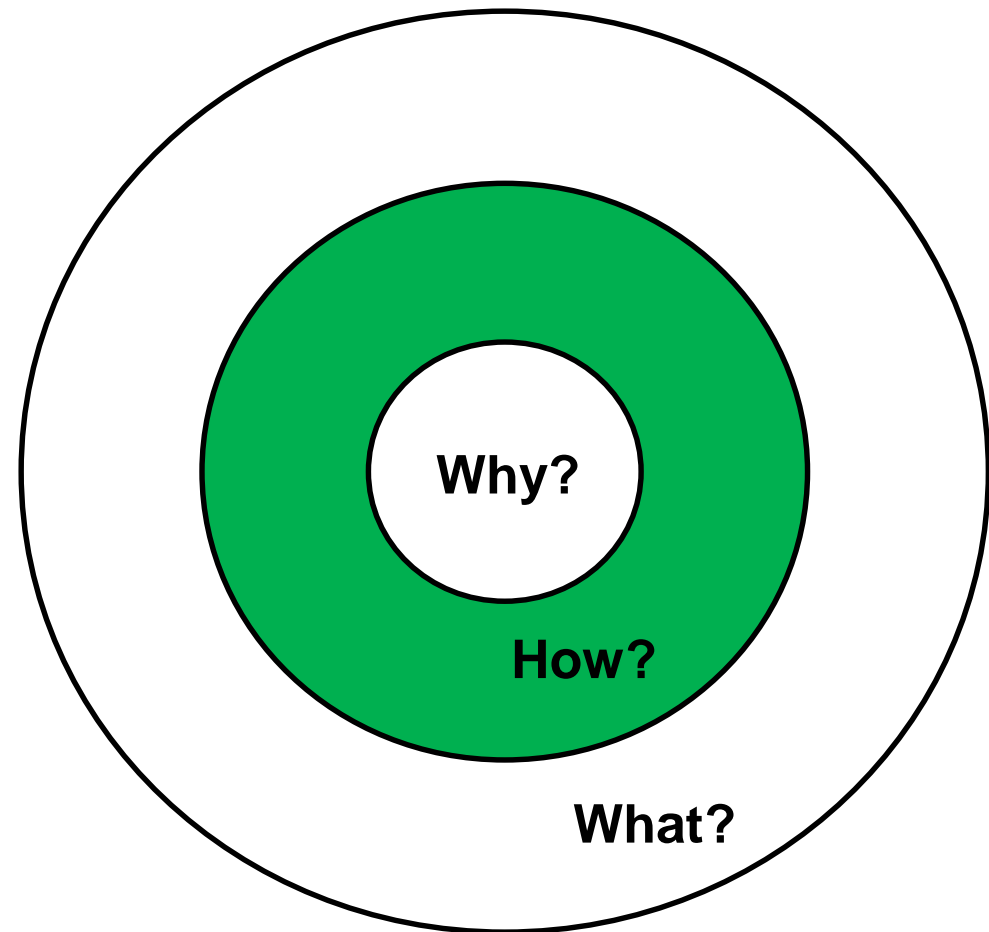
# Traditional Development Process



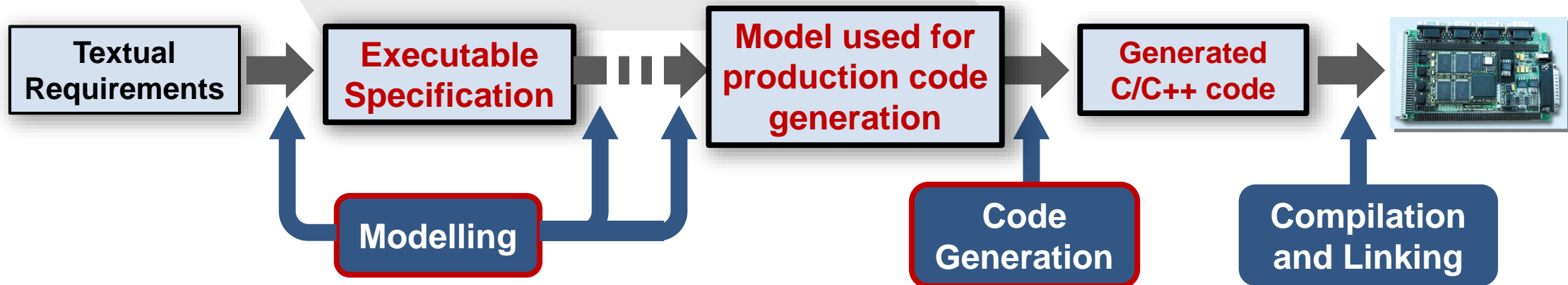
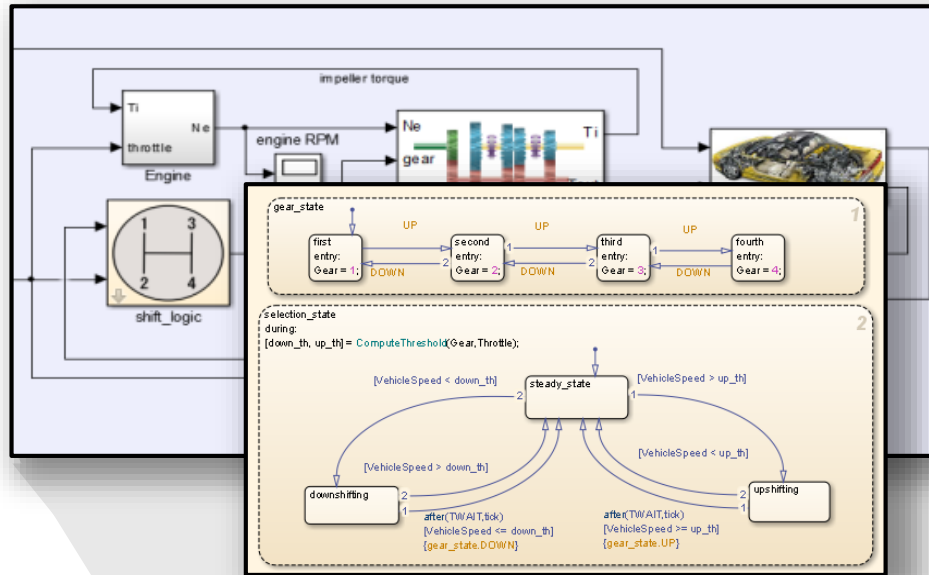
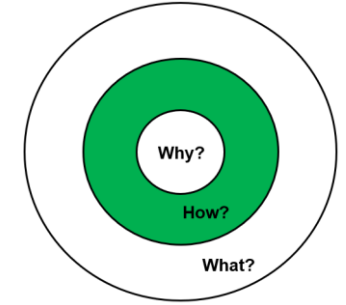
# Battery Management System (BMS)



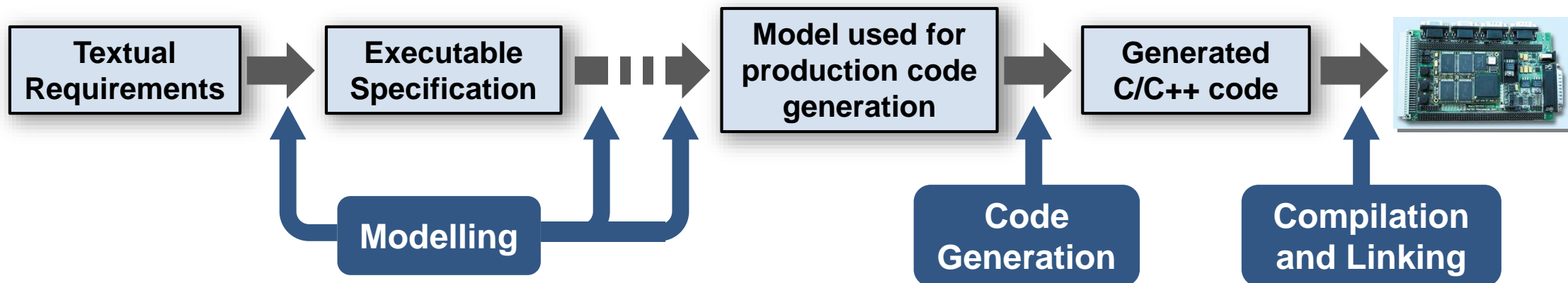
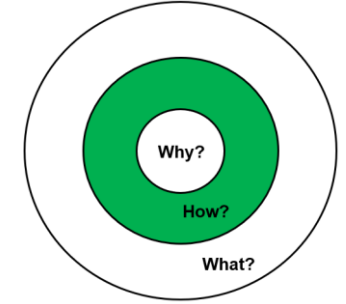
- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Integration Testing



# Complete Model Based Design Workflow

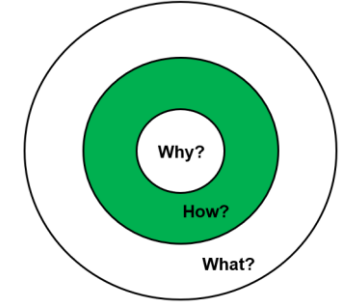


# Complete Model Based Design Workflow

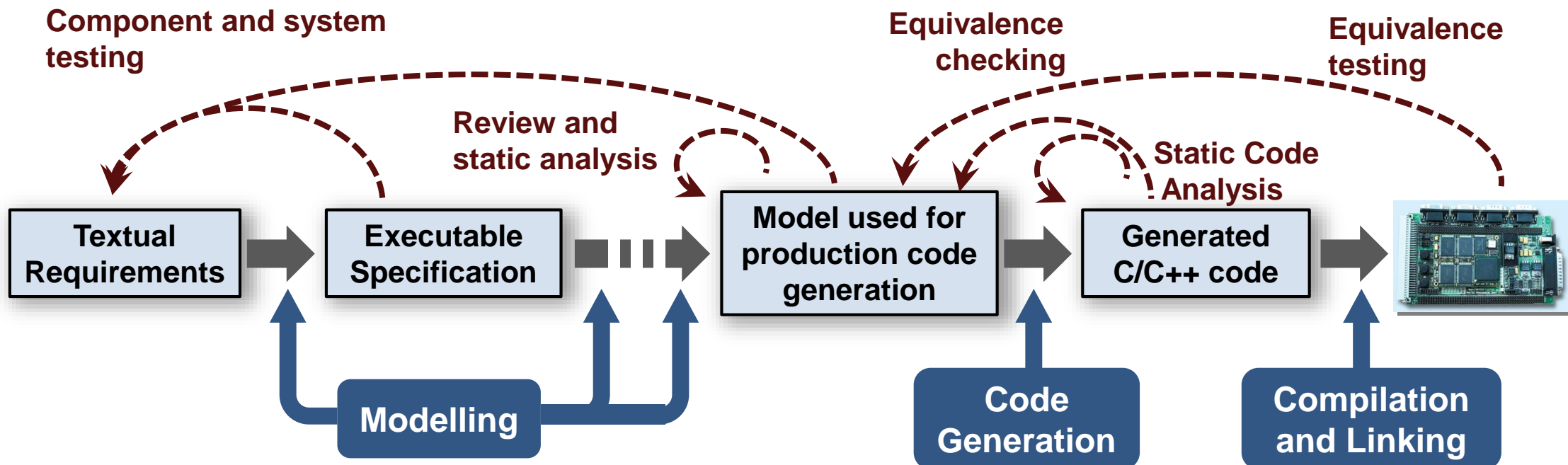




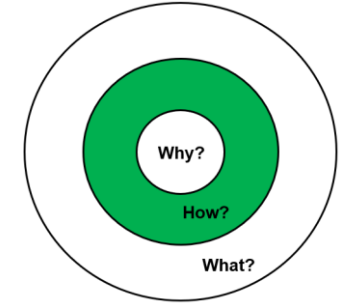
# Complete Model Based Design Workflow



get the complete confidence in your design

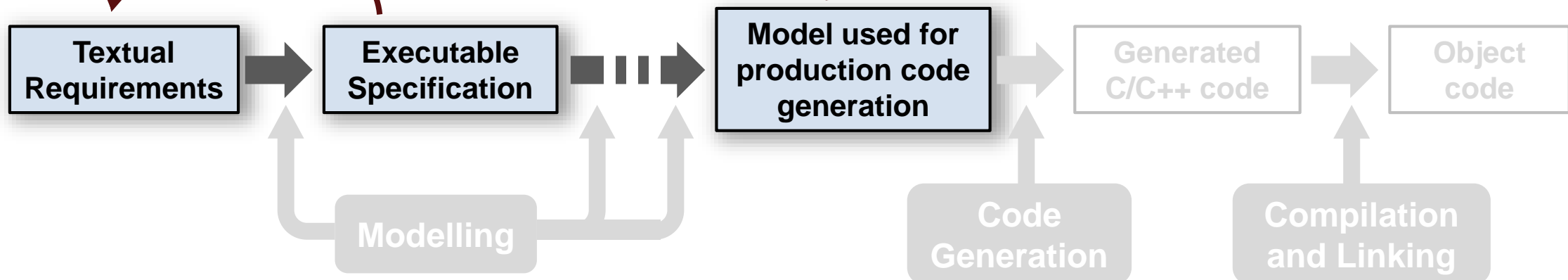


# Model Based Design Verification Workflow



- **Managing Requirements**
- Isolate and test components
- Perform simulation
- Manage and organize tests
- Measure model coverage
- Generate tests for missing coverage

Component and system testing

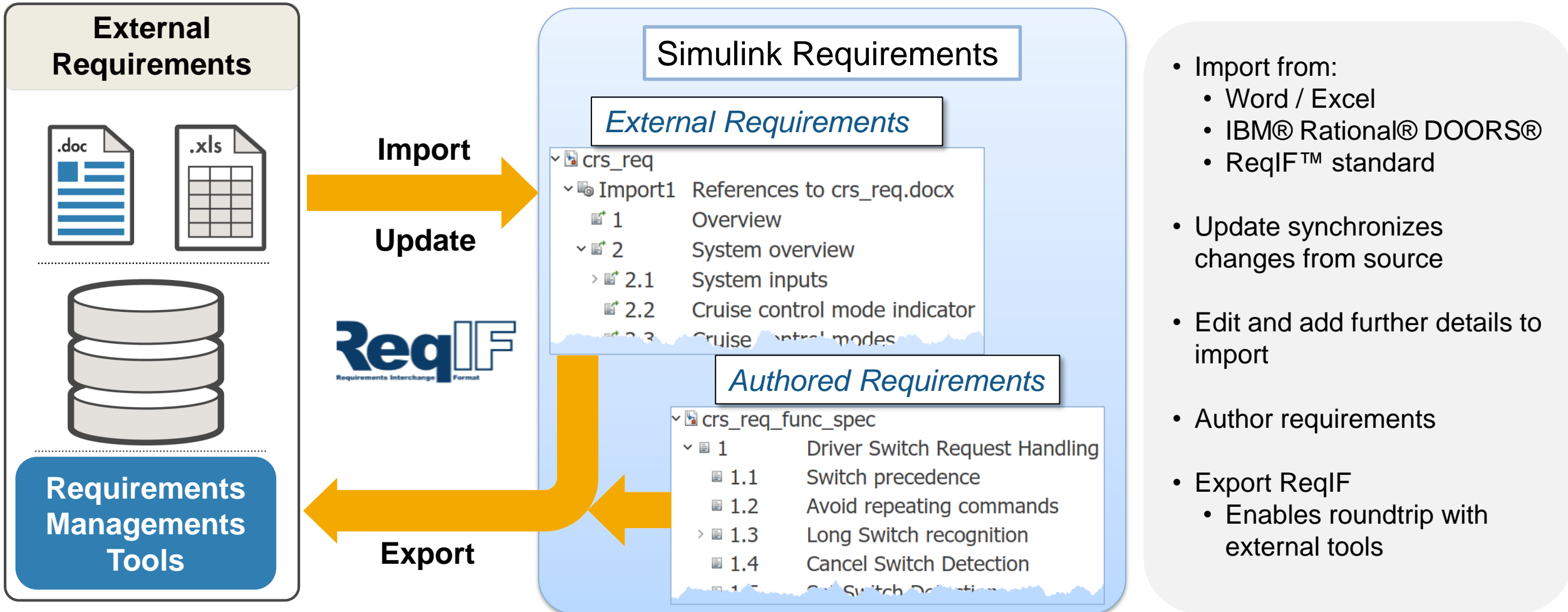


# Why do 71% of Embedded Projects Fail?

## Poor Requirements Management

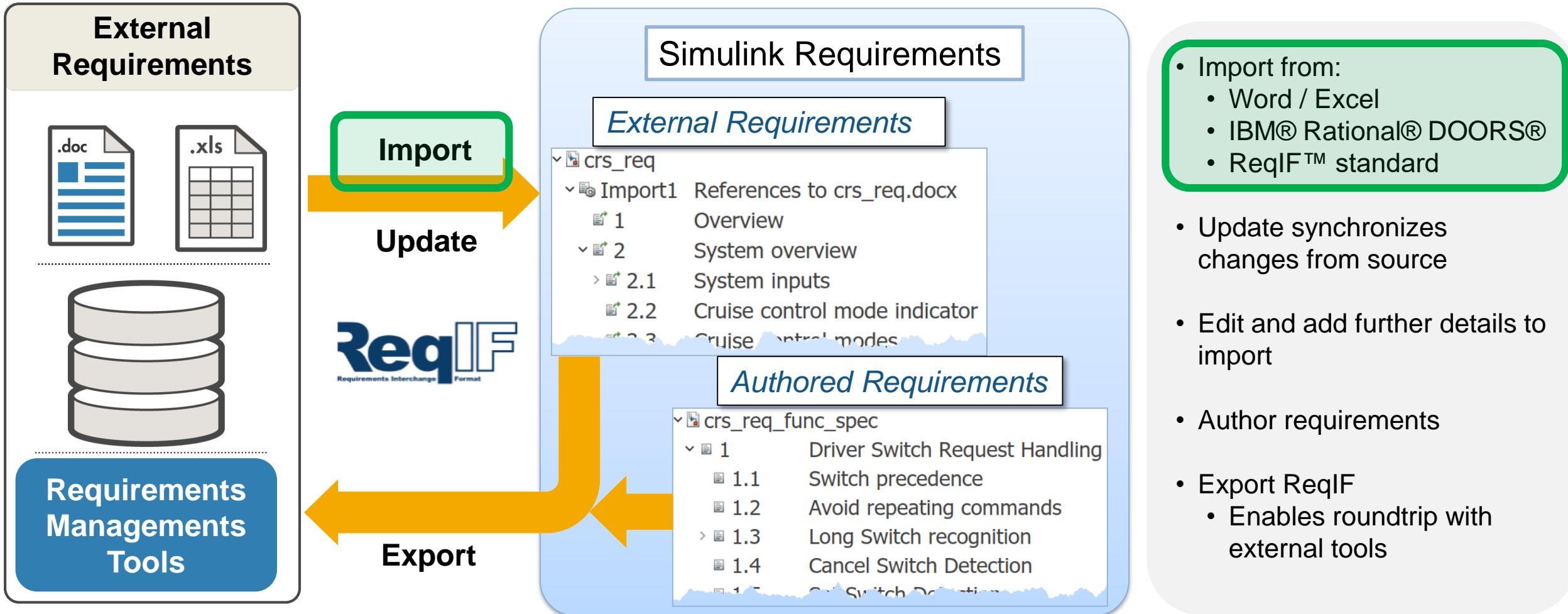
*Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005*

# Integrate with requirements tools and author requirements

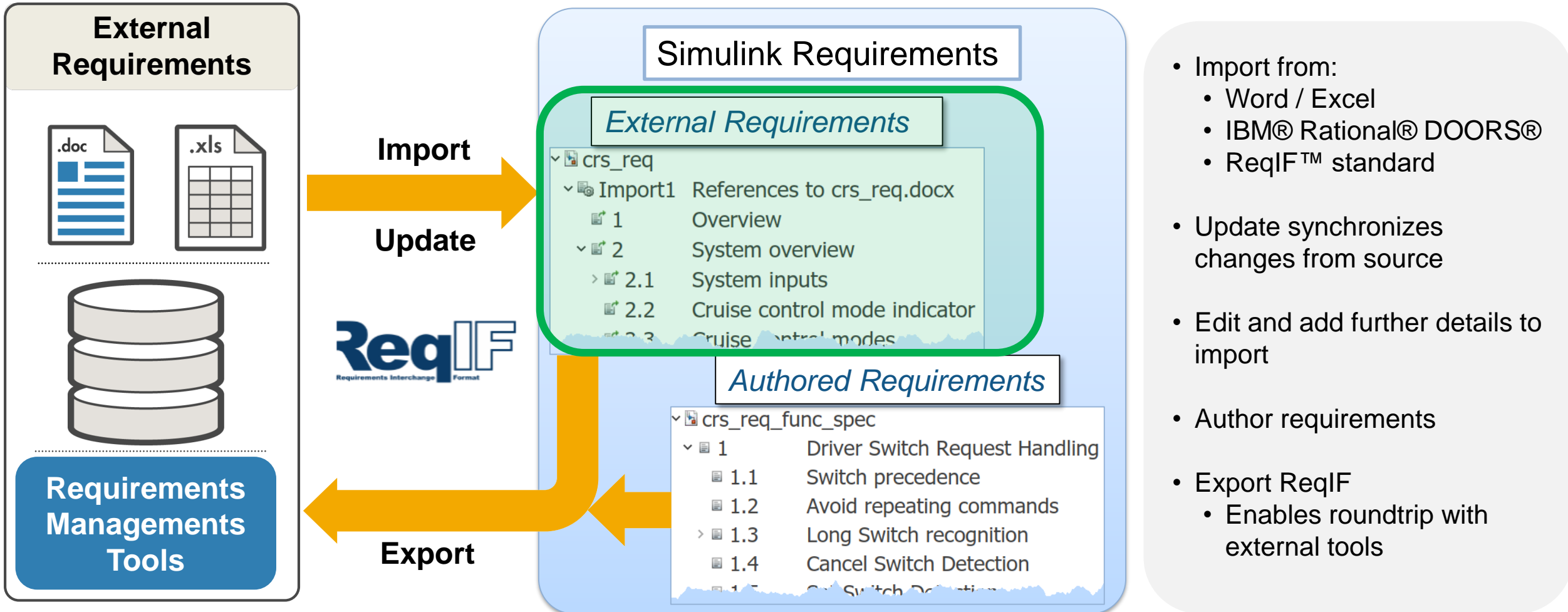




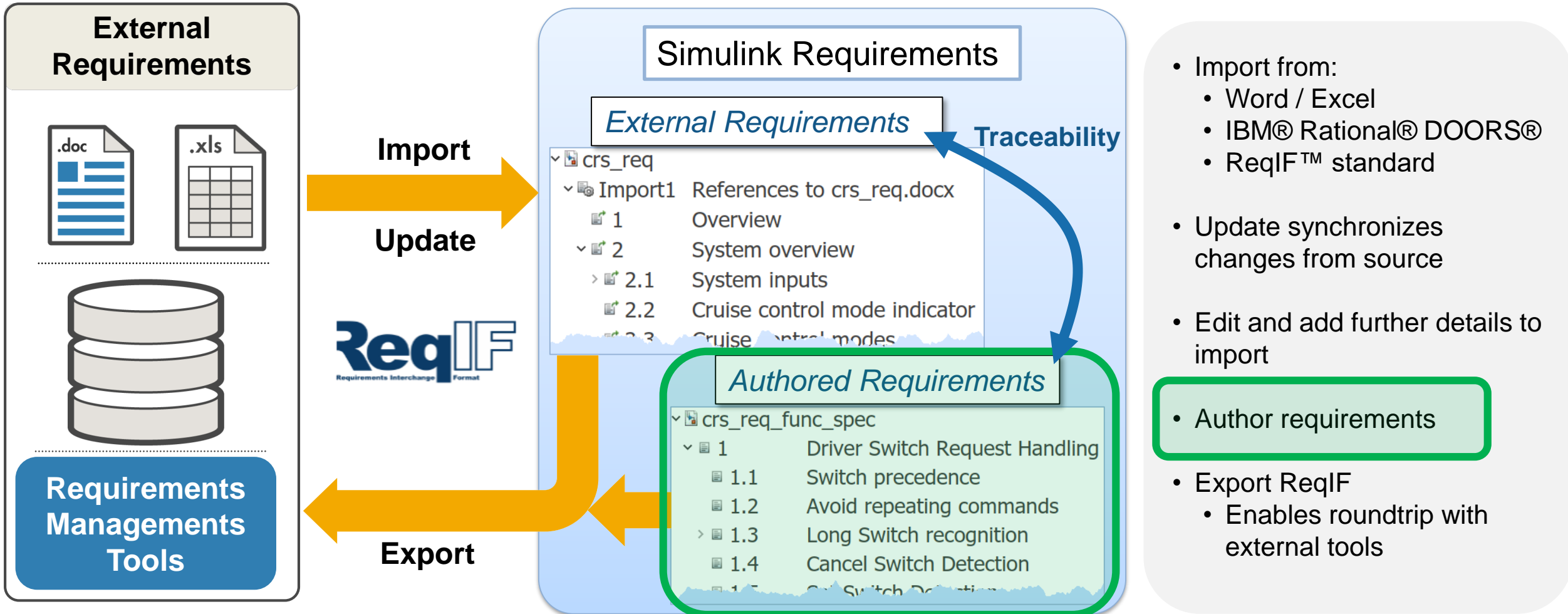
# Integrate with requirements tools and author requirements



# Integrate with requirements tools and author requirements

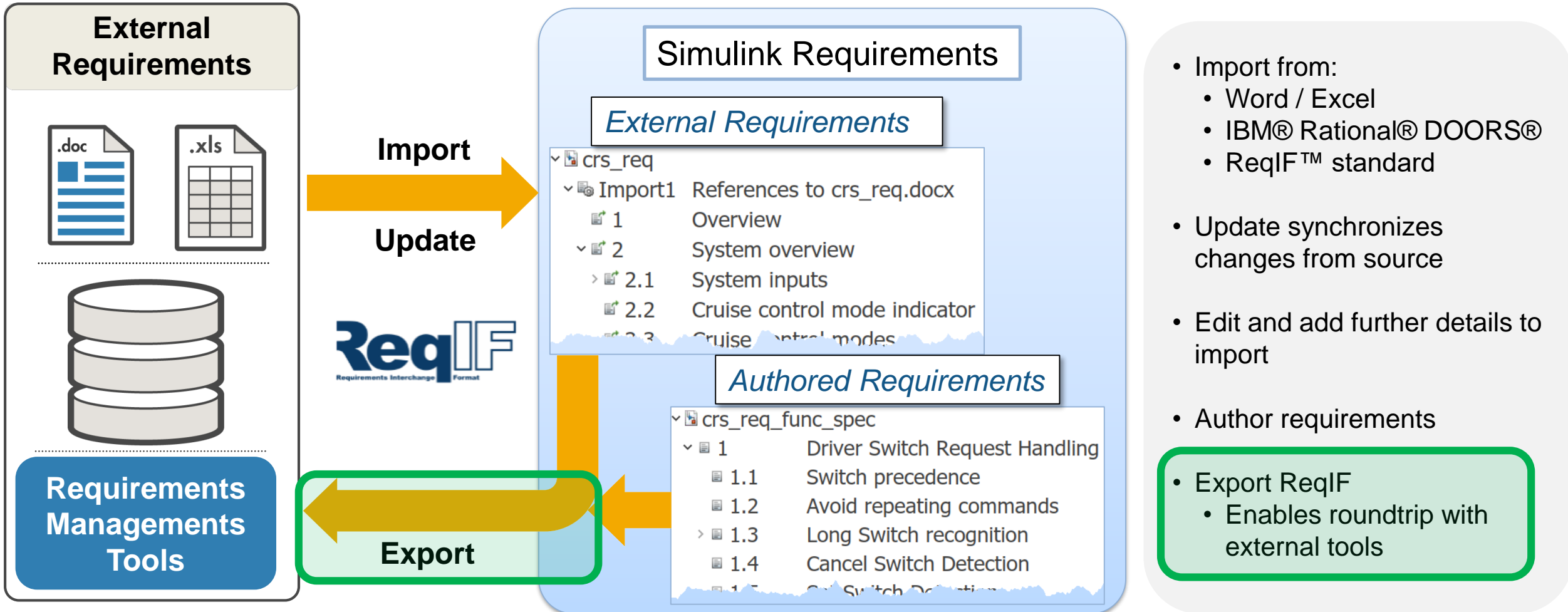


# Integrate with requirements tools and author requirements





# Integrate with requirements tools and author requirements



# Requirements Perspective

## External Requirements



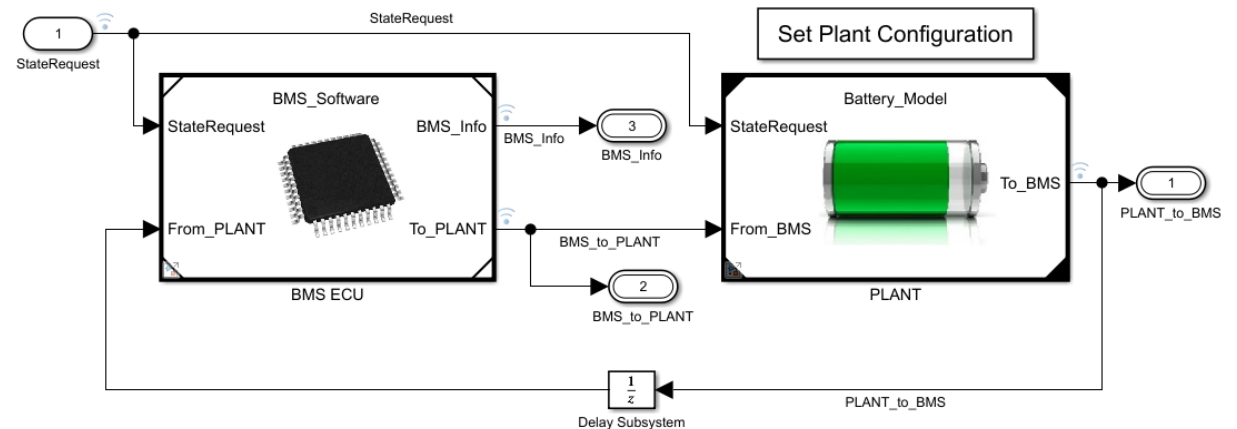
### Requirements Managements Tools

- crs\_req
  - Import1 References to crs\_req.docx
    - 1 Overview
    - 2 System overview
      - 2.1 System inputs
      - 2.2 Cruise control mode indicator
      - 2.3 Cruise control modes

## Authored Requirements

- crs\_req\_func\_spec
  - 1 Driver Switch Request Handling
    - 1.1 Switch precedence
    - 1.2 Avoid repeating commands
    - 1.3 Long Switch recognition
    - 1.4 Cancel Switch Detection
    - 1.5 On Switch Detection

## Design in Simulink



# Traceability: Requirements - Design

## External Requirements



### Requirements Managements Tools

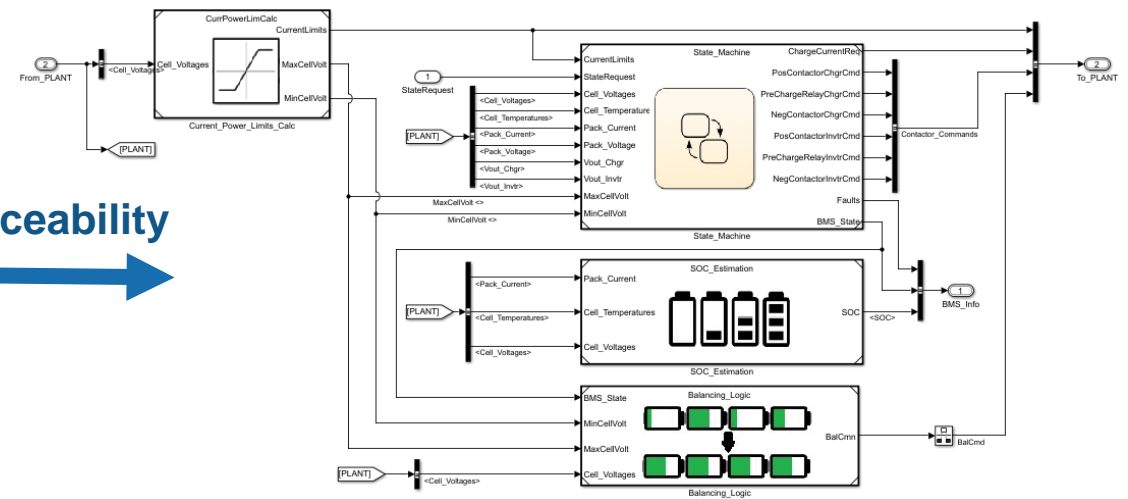
- crs\_req
  - Import1 References to crs\_req.docx
    - 1 Overview
    - 2 System overview
      - 2.1 System inputs
      - 2.2 Cruise control mode indicator
      - 2.3 Cruise control modes

## Authored Requirements

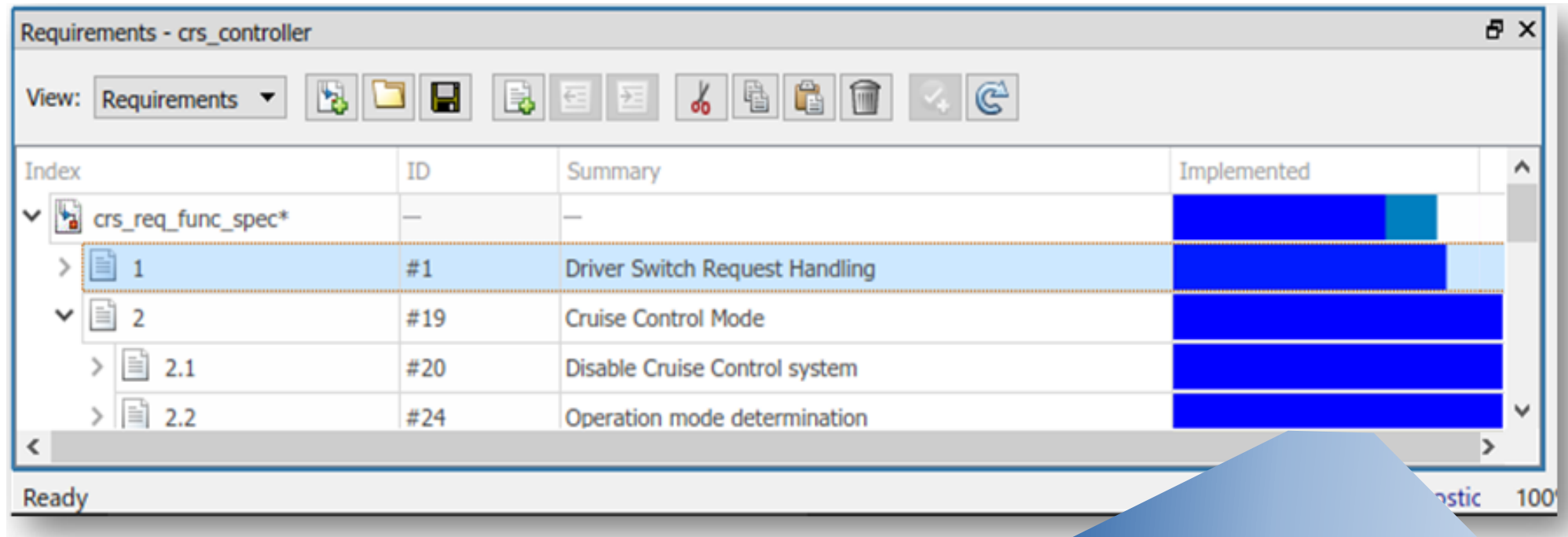
- crs\_req\_func\_spec
  - 1 Driver Switch Request Handling
    - 1.1 Switch precedence
    - 1.2 Avoid repeating commands
    - 1.3 Long Switch recognition
    - 1.4 Cancel Switch Detection
    - 1.5 Switch Detection

## Design in Simulink

Traceability



# Requirements Implementation Status



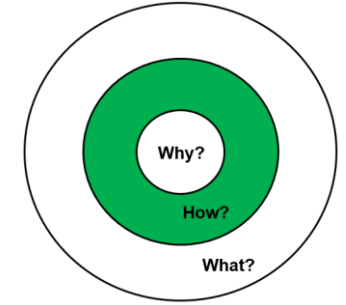
The screenshot shows a software window titled "Requirements - crs\_controller". It features a toolbar with icons for file operations and a table of requirements. The table has four columns: Index, ID, Summary, and Implemented. The "Implemented" column uses horizontal bars to show the status of each requirement. A blue bar indicates the requirement is implemented, a light blue bar indicates it is justified, and a white bar indicates it is missing. The status bar at the bottom shows "Ready" and "100%".

Index	ID	Summary	Implemented
crs_req_func_spec*	—	—	
> 1	#1	Driver Switch Request Handling	
> 2	#19	Cruise Control Mode	
> 2.1	#20	Disable Cruise Control system	
> 2.2	#24	Operation mode determination	

## Implementation Status

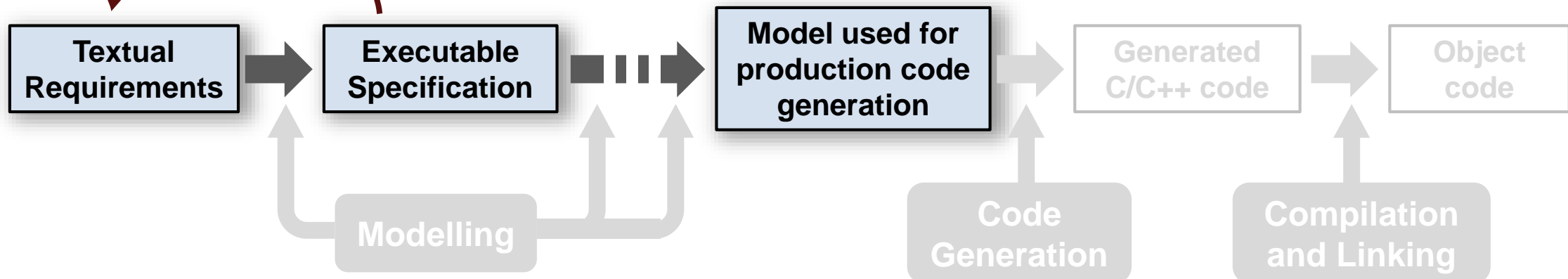
-  Implemented
-  Justified
-  Missing

# Model Based Design Verification Workflow

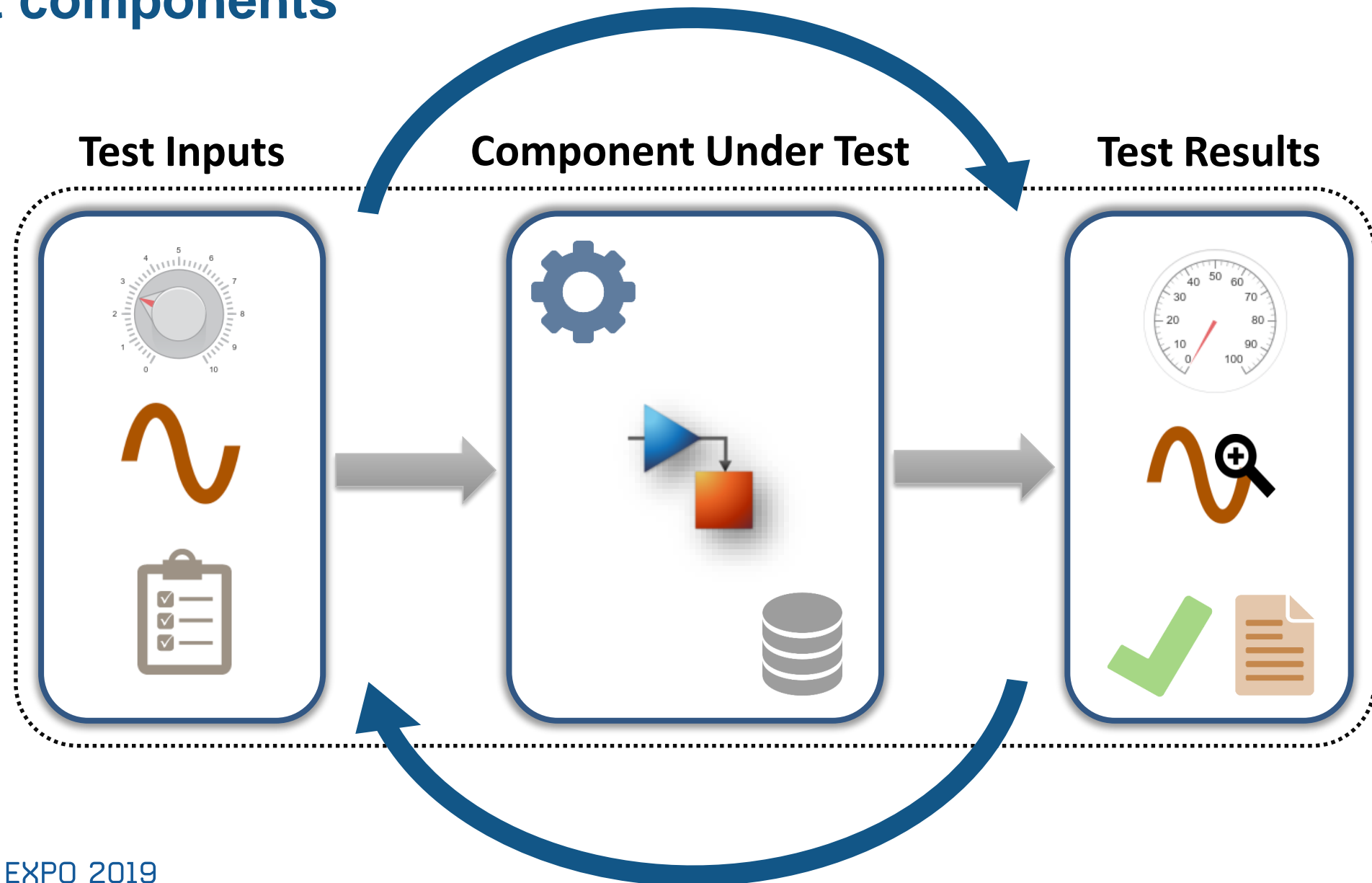


- Managing Requirements
- **Isolate and test components**
- Perform simulation
- Manage and organize tests
- Measure model coverage
- Generate tests for missing coverage

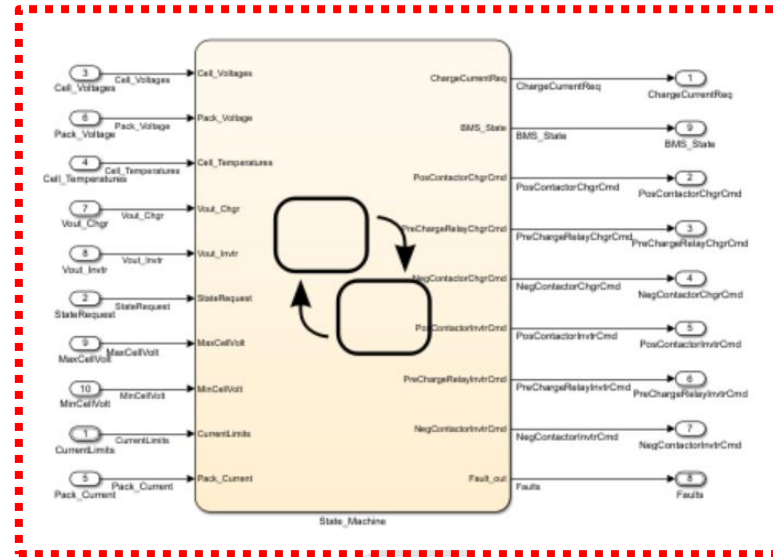
Component and system testing



# Test components

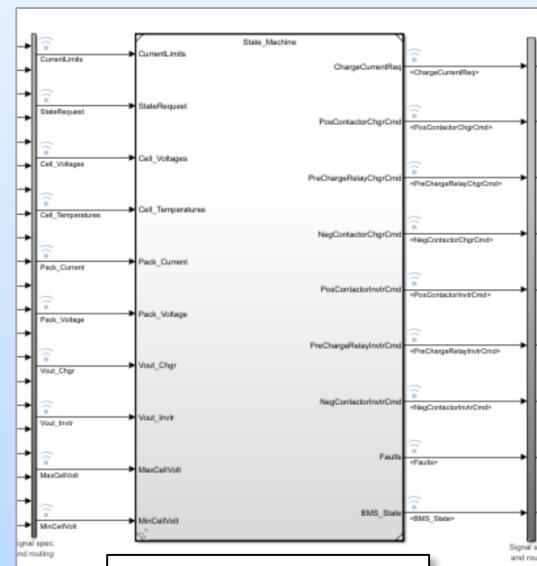
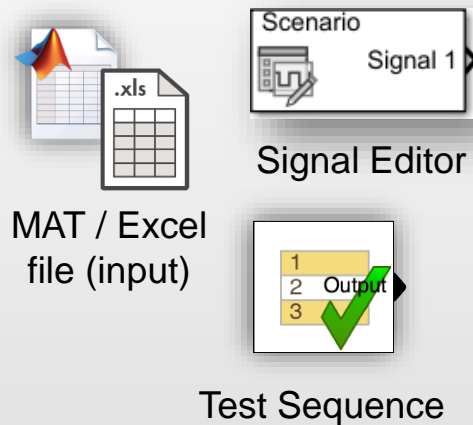


# Create Test Harnesses



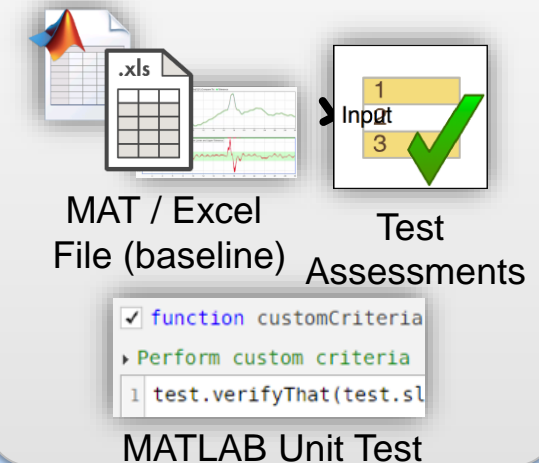
## Test Cases

### Inputs

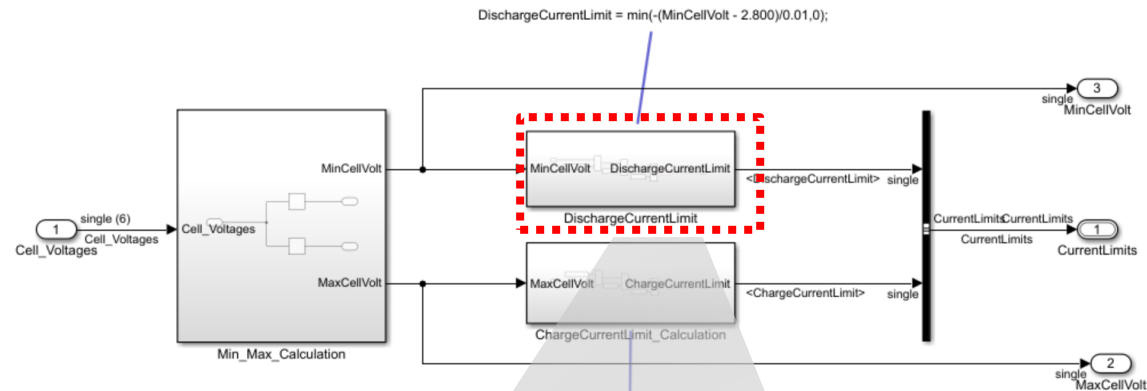


## Test Harness

### Assessments

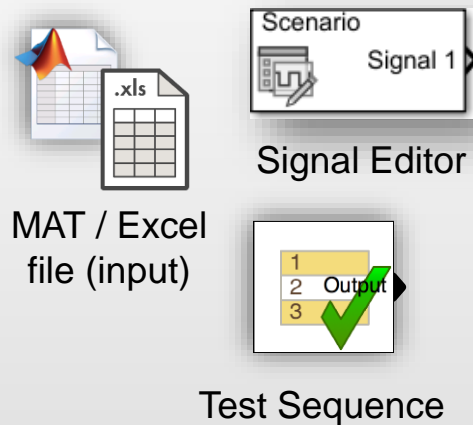


# Create Test Harnesses on Subsystems

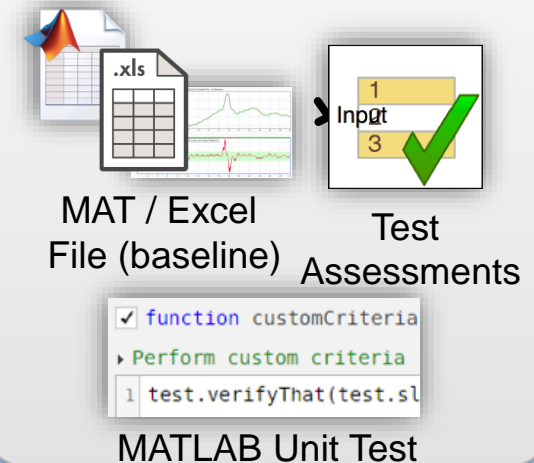


## Test Cases

### Inputs



### Assessments

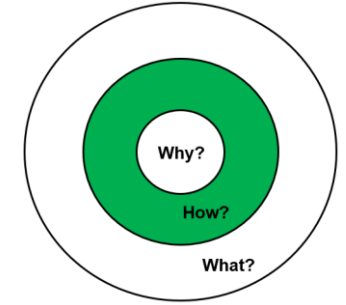


*Test Harness*

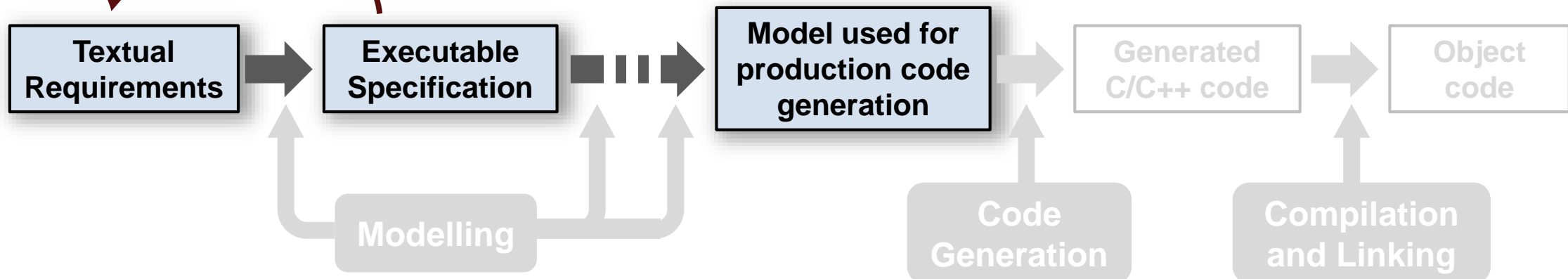


# Model Based Design Verification Workflow

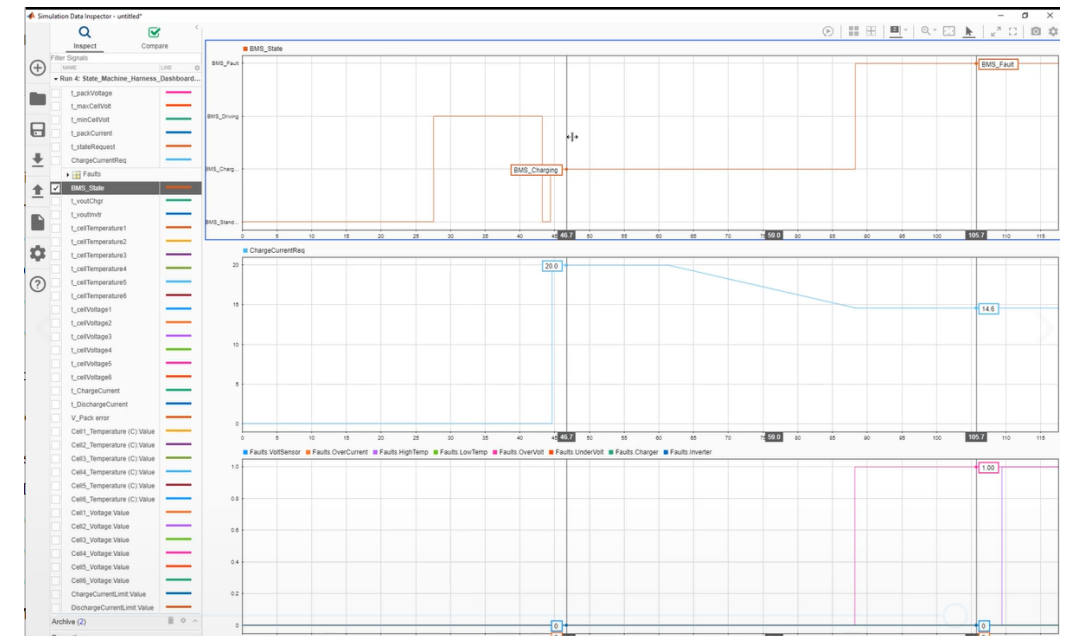
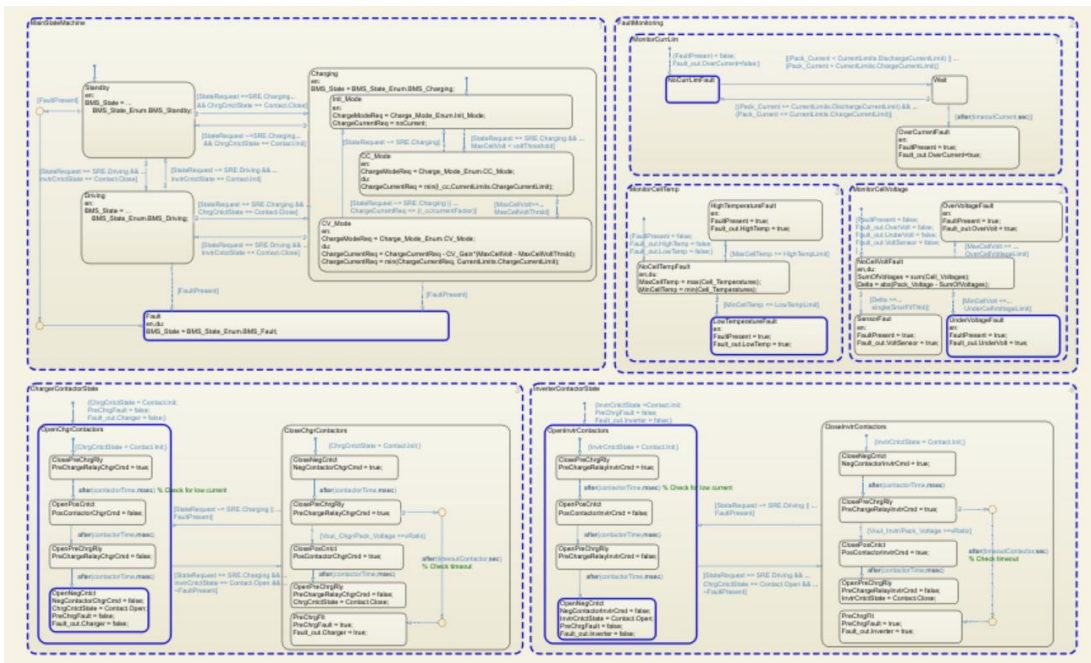
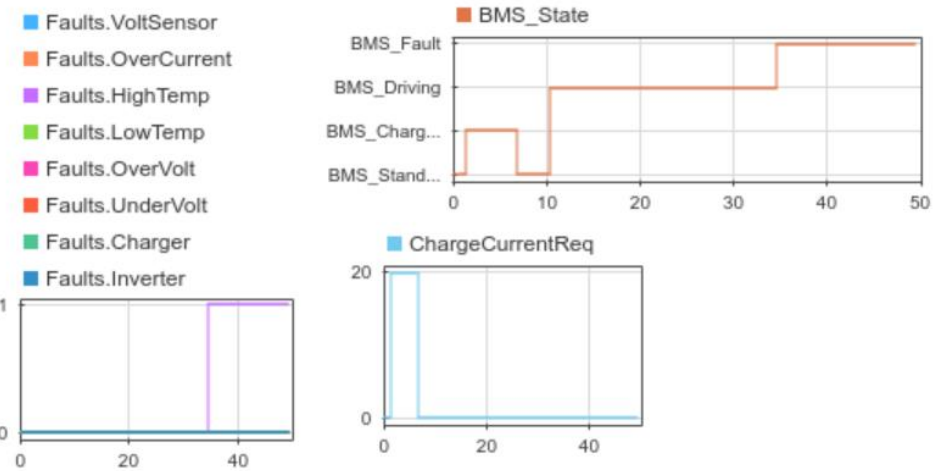
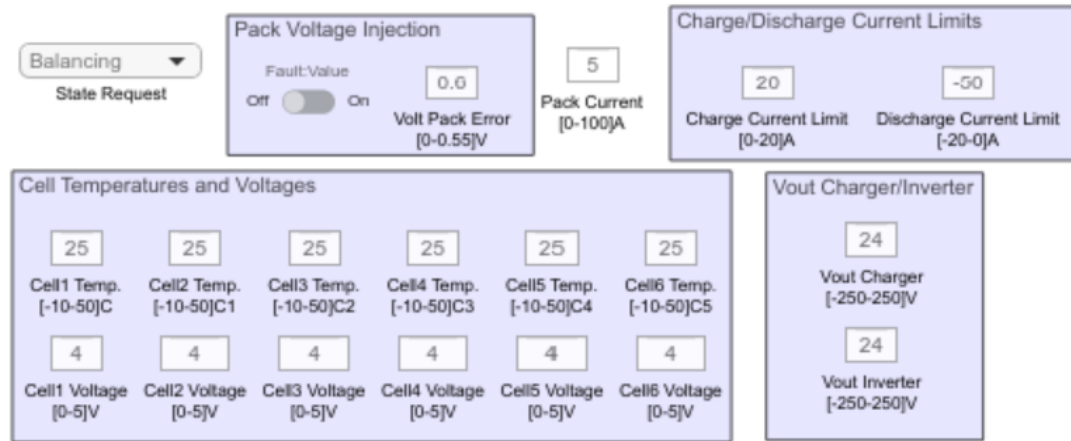
- Managing Requirements
- Isolate and test components
- **Perform simulation**
- Manage and organize tests
- Measure model coverage
- Generate tests for missing coverage



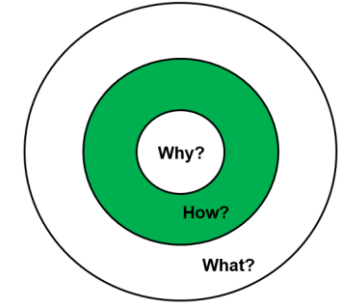
Component and system testing



# Test Ad-hoc with Dashboards

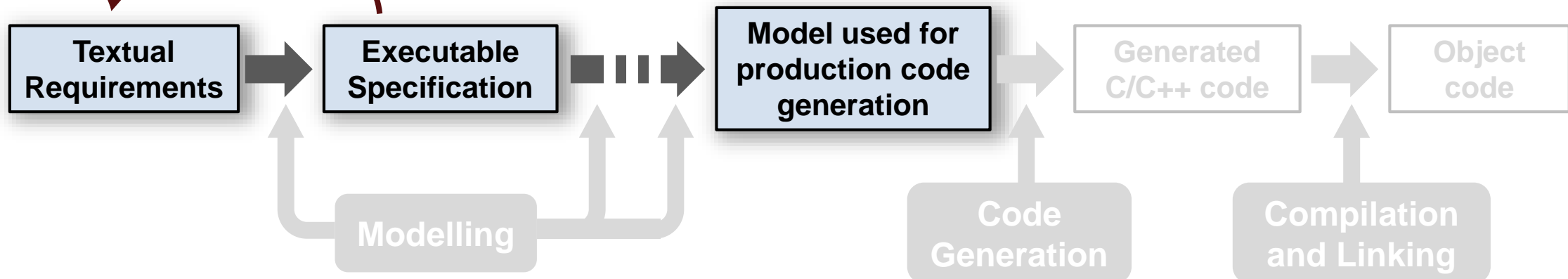


# Model Based Design Verification Workflow

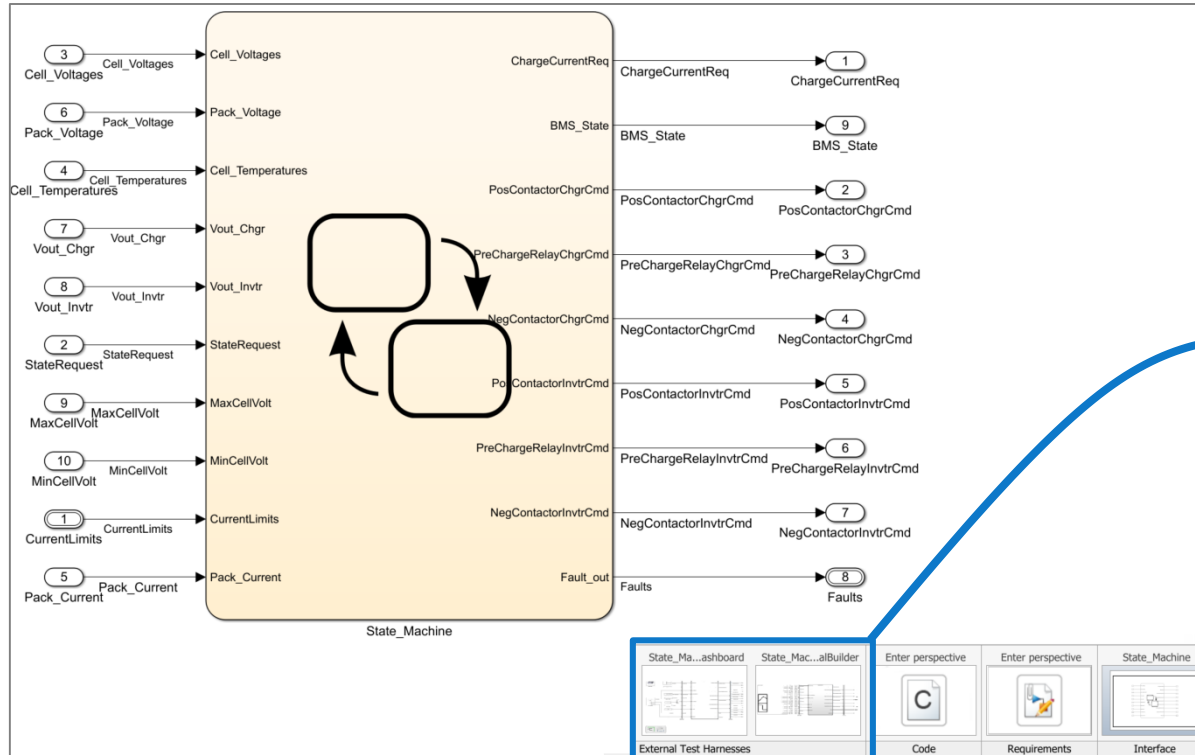


- Managing Requirements
- Isolate and test components
- Perform simulation
- **Manage and organize tests**
- Measure model coverage
- Generate tests for missing coverage

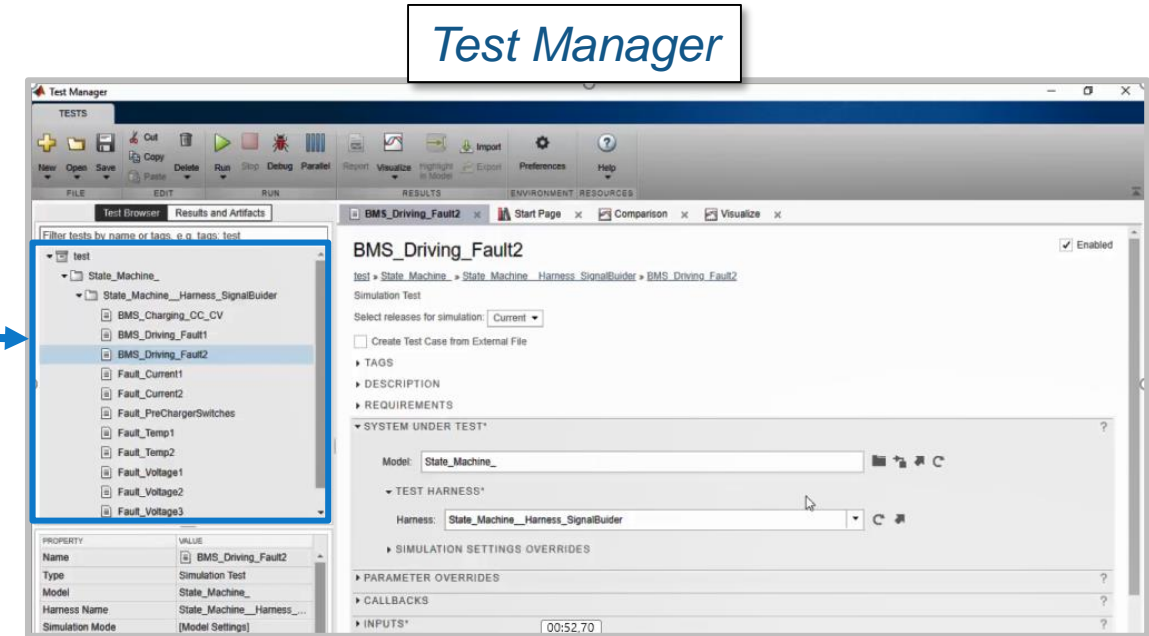
Component and system testing



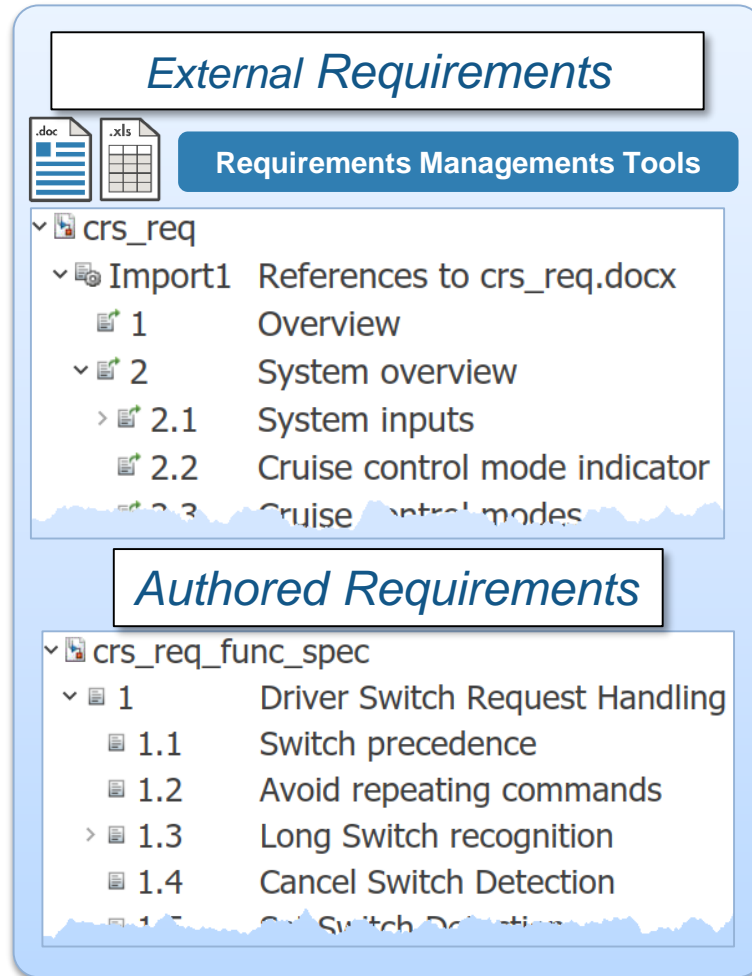
# Test Harness Import in a Centralized Environment



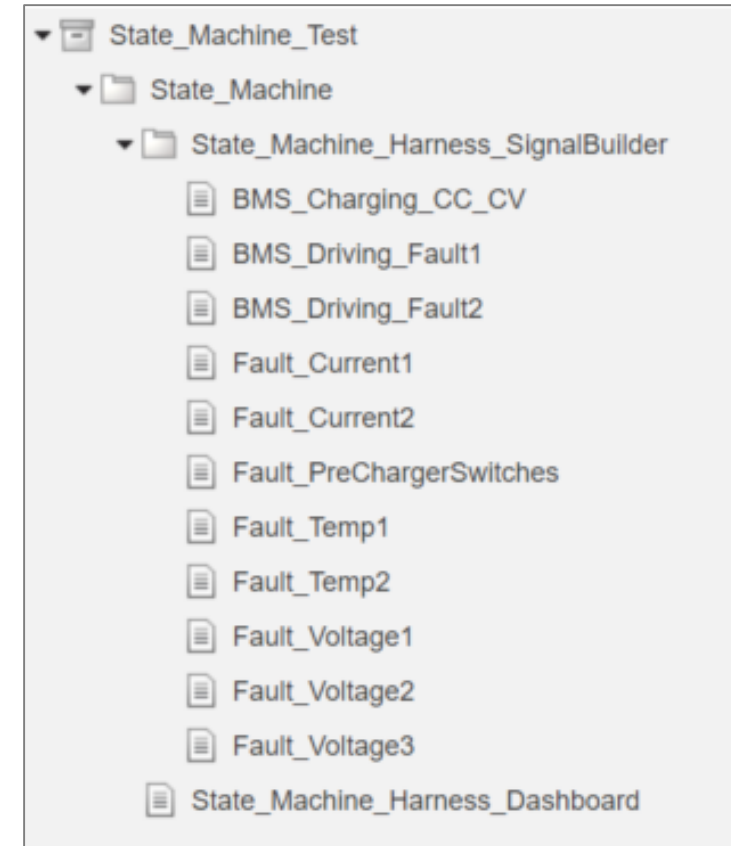
Test Harnesses



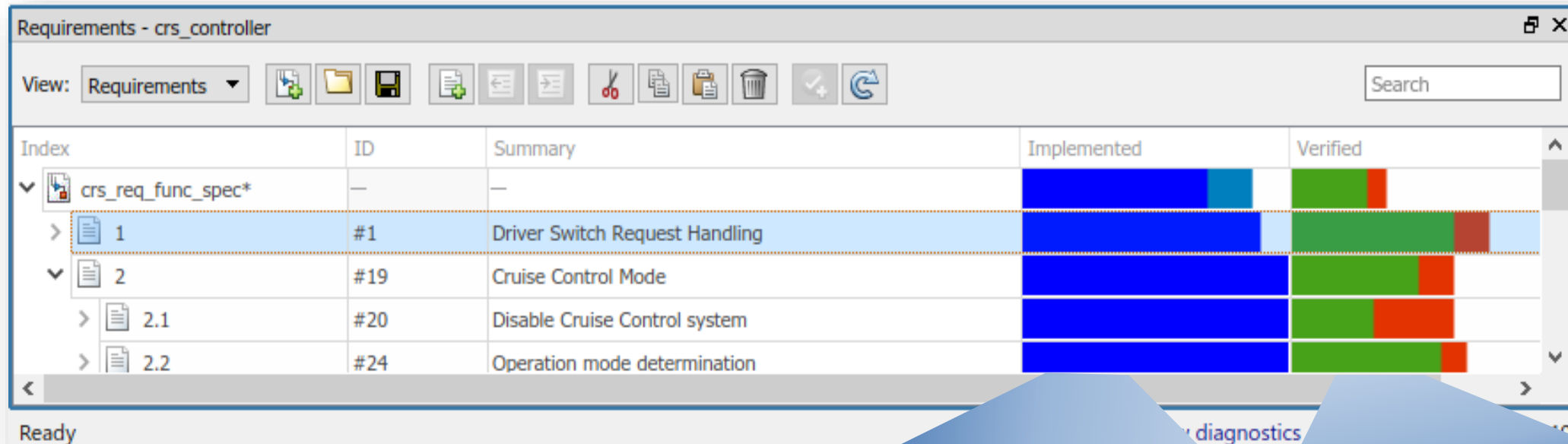
# Traceability: Requirements - Tests



Traceability



# Requirements Verification Status







The screenshot shows the 'Requirements - crs\_controller' window. It features a toolbar with icons for file operations and a search bar. The main table lists requirements with columns for Index, ID, Summary, Implemented, and Verified. The 'Implemented' column uses blue bars to show the percentage of implemented requirements, while the 'Verified' column uses green (Passed), red (Failed), and orange (No Result) bars. The status bar at the bottom indicates 'Ready'.

Index	ID	Summary	Implemented	Verified
crs_req_func_spec*	—	—		
> 1	#1	Driver Switch Request Handling		
> 2	#19	Cruise Control Mode		
> 2.1	#20	Disable Cruise Control system		
> 2.2	#24	Operation mode determination		

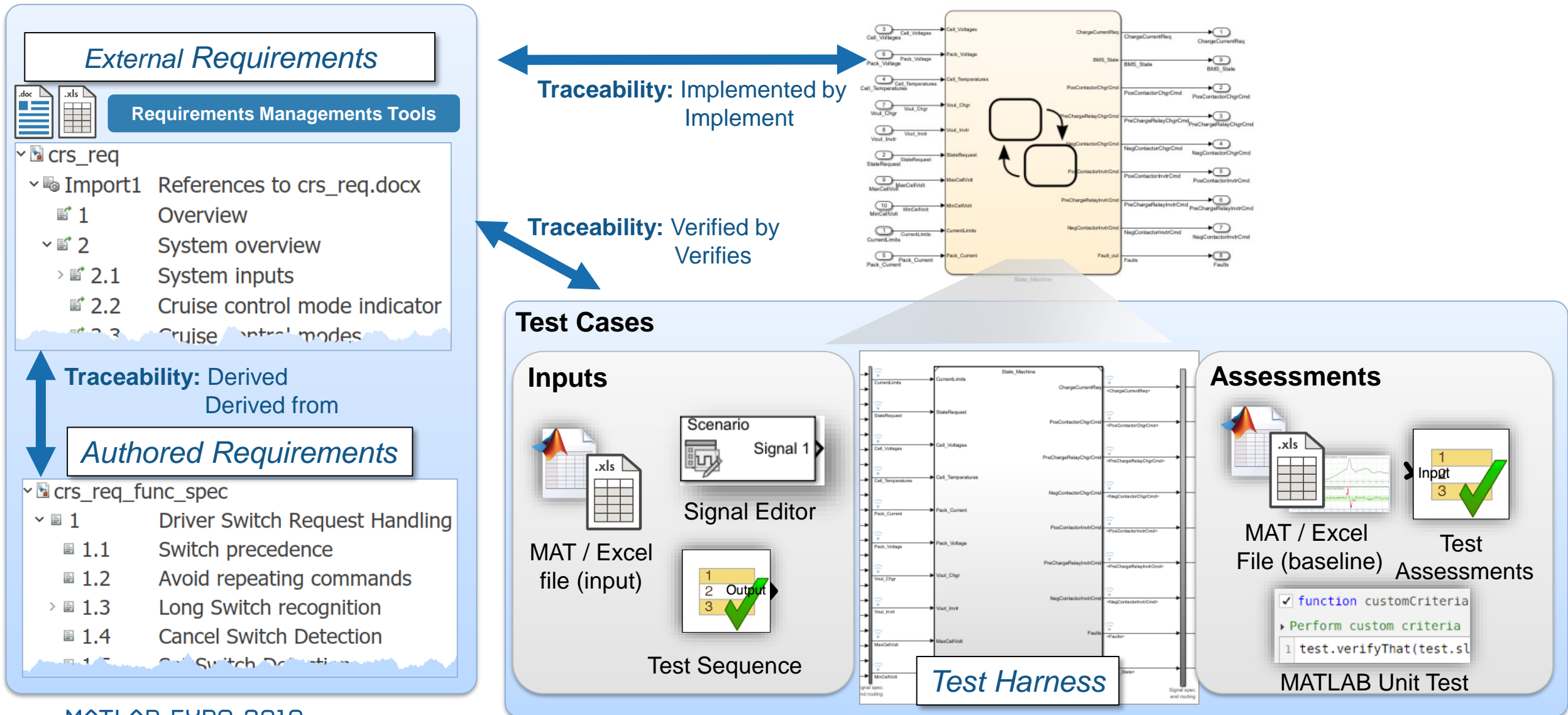
## Implementation Status

-  Implemented
-  Justified
-  Missing

## Verification Status

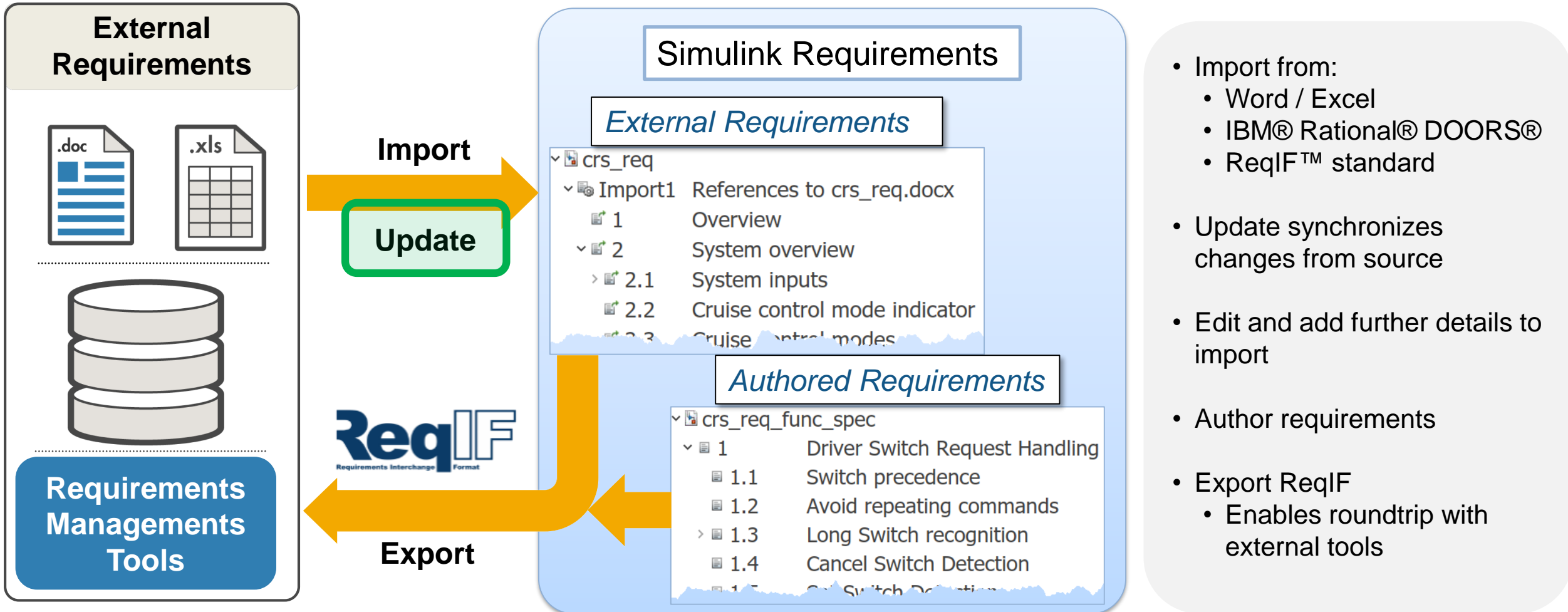
-  Passed
-  Failed
-  No Result
-  Missing

# Requirements Verification with Simulink





# Integrate with requirements tools and author requirements





# Respond to Change – Impact Analysis

## Original Requirement

### 3.2.4 BMS Current Limit Calculation: Charge Current Limit

Calculate the charge current limit following:

ChargeCurrentLimit = **min**((4.500 - MaxCellVolt)/0.01,0);



## Updated Requirement

### 3.2.4 BMS Current Limit Calculation: Charge Current Limit

Calculate the charge current limit following:

ChargeCurrentLimit = **max**((4.500 - MaxCellVolt)/0.01,0);

# Respond to Change – Impact Analysis

## Original Requirement

### 3.2.4 BMS Current Limit Calculation: Charge Current Limit

Calculate the charge current limit following:

ChargeCurrentLimit = **min**((4.500 - MaxCellVolt)/0.01,0);



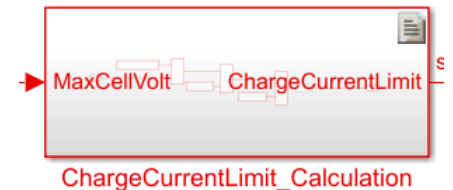
## Updated Requirement

### 3.2.4 BMS Current Limit Calculation: Charge Current Limit

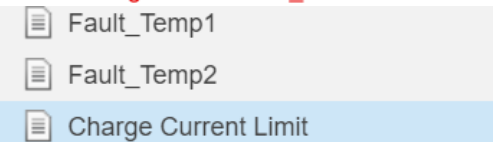
Calculate the charge current limit following:

ChargeCurrentLimit = **max**((4.500 - MaxCellVolt)/0.01,0);

Implemented by



Verified by

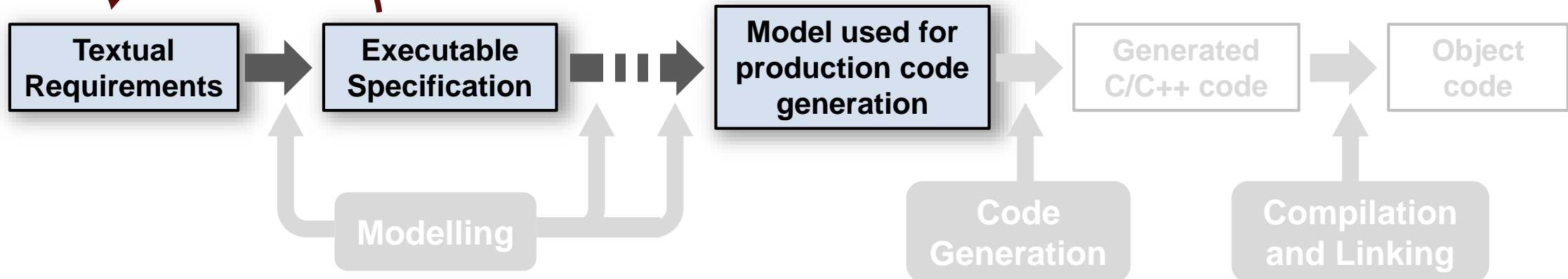


 Issue: Destination Changed.

# Model Based Design Verification Workflow

- Managing Requirements
- Isolate and test components
- Perform simulation
- Manage and organize tests
- **Measure model coverage**
- Generate tests for missing coverage

Component and system testing



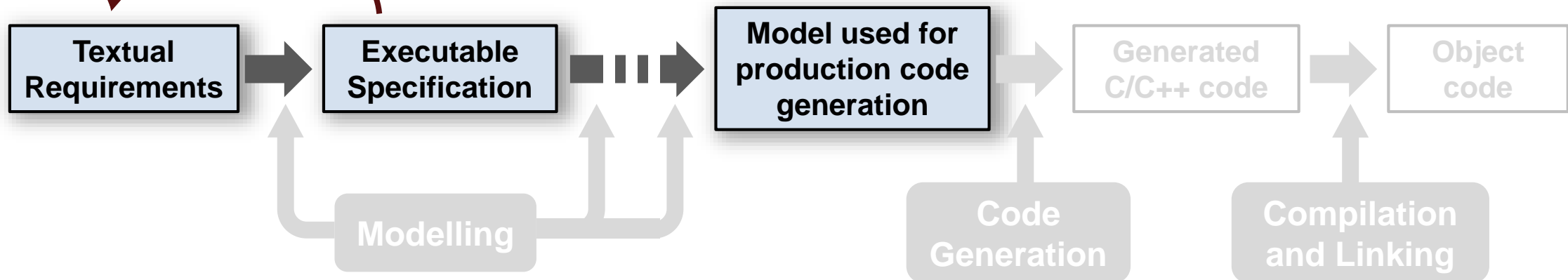
# Why we didn't achieve full model coverage?

- A. Missing tests
- B. Missing requirements
- C. Design errors

# Model Based Design Verification Workflow

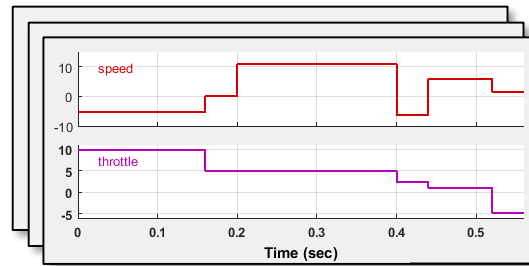
- Managing Requirements
- Isolate and test components
- Perform simulation
- Manage and organize tests
- Measure model coverage
- **Generate tests for missing coverage**

Component and system testing

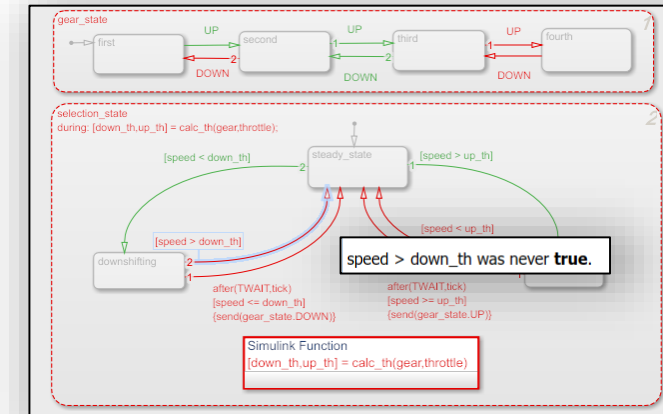
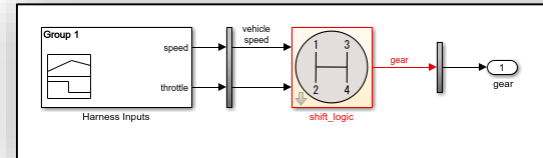


# Addressing Missing Coverage

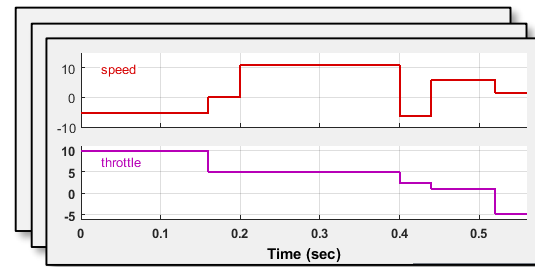
## Partial Coverage



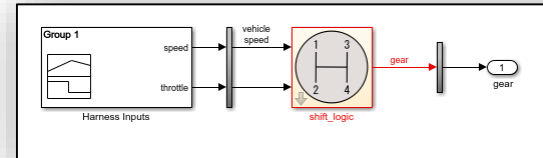
## Test Cases



# Addressing Missing Coverage



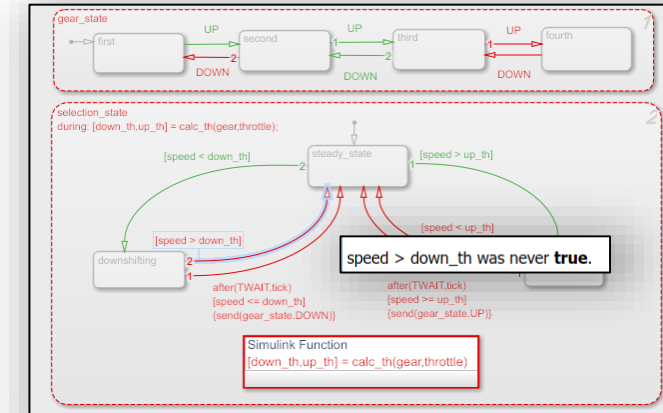
Test Cases



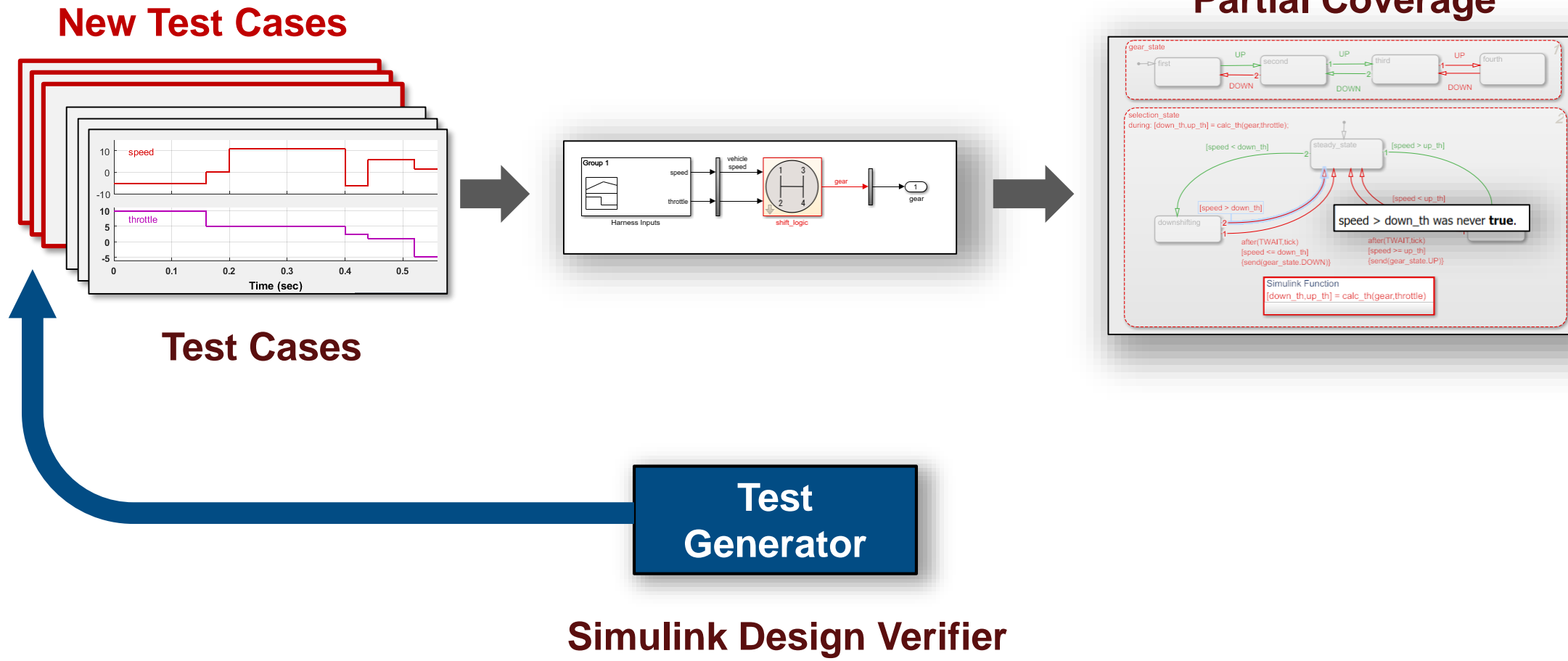
Test  
Generator

Simulink Design Verifier

## Partial Coverage



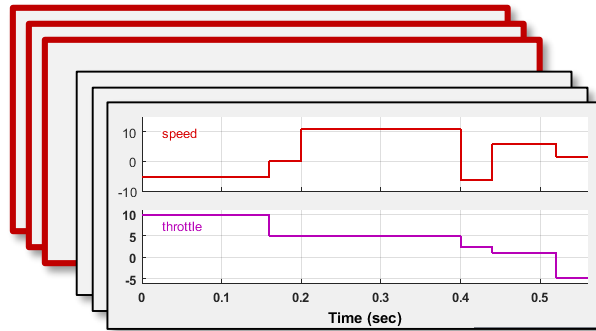
# Addressing Missing Coverage



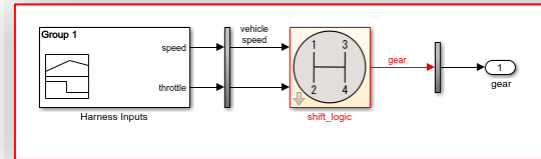


# Addressing Missing Coverage

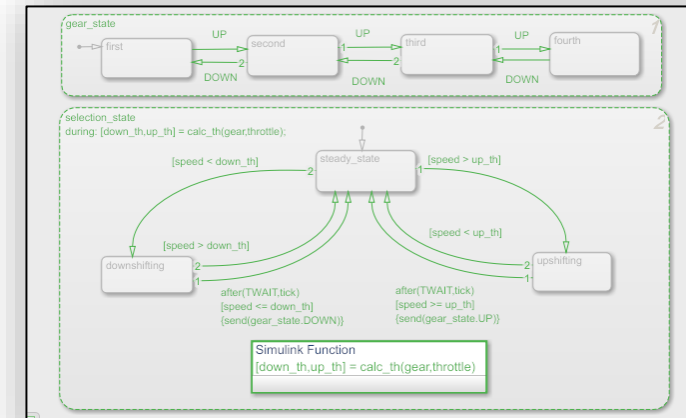
## New Test Cases



## Test Cases



## Full Coverage

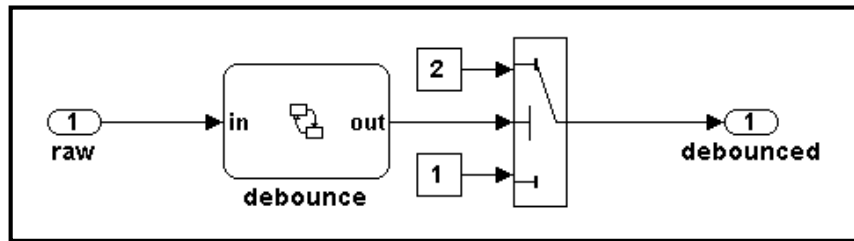


# Why we didn't achieve full model coverage?

- A. Missing tests
- B. Missing requirements
- C. Design errors

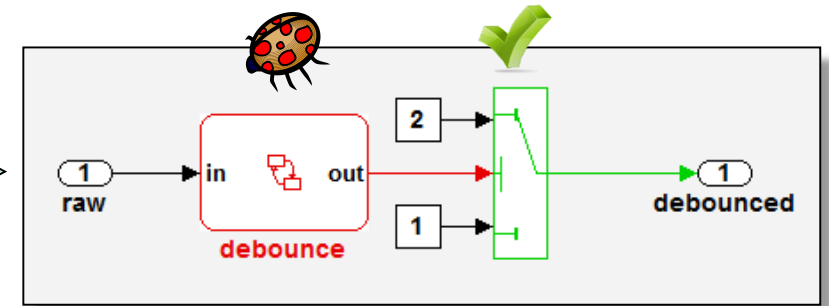
# Detecting Hidden Run-Time Design Errors

Statically, without simulations



**Design Model**

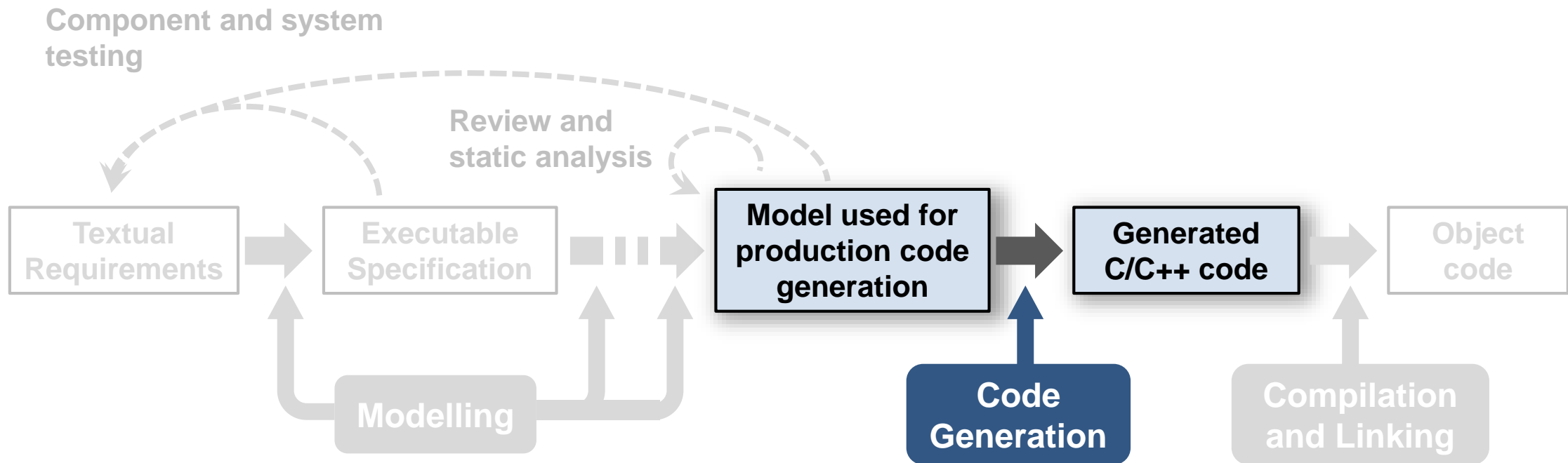
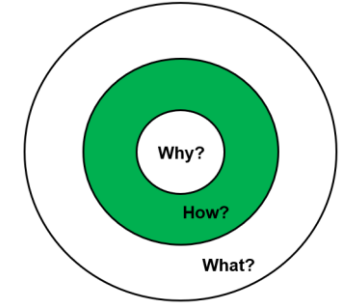
Design error detection



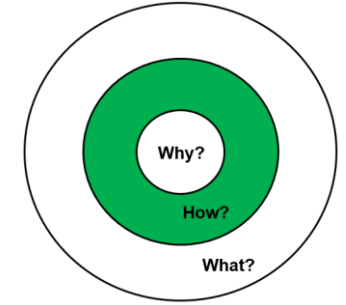
**Highlighted Model**

- Integer overflow
- Division by zero
- Array out-of-bounds
- Range violations
- Dead Logic

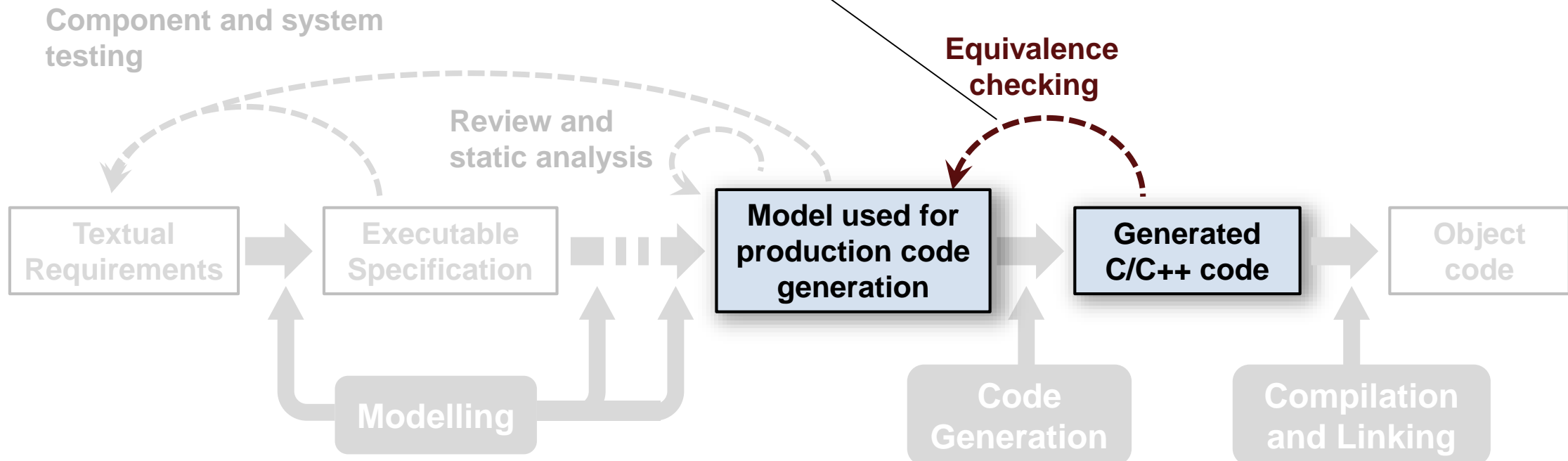
# Model Based Design Verification Workflow



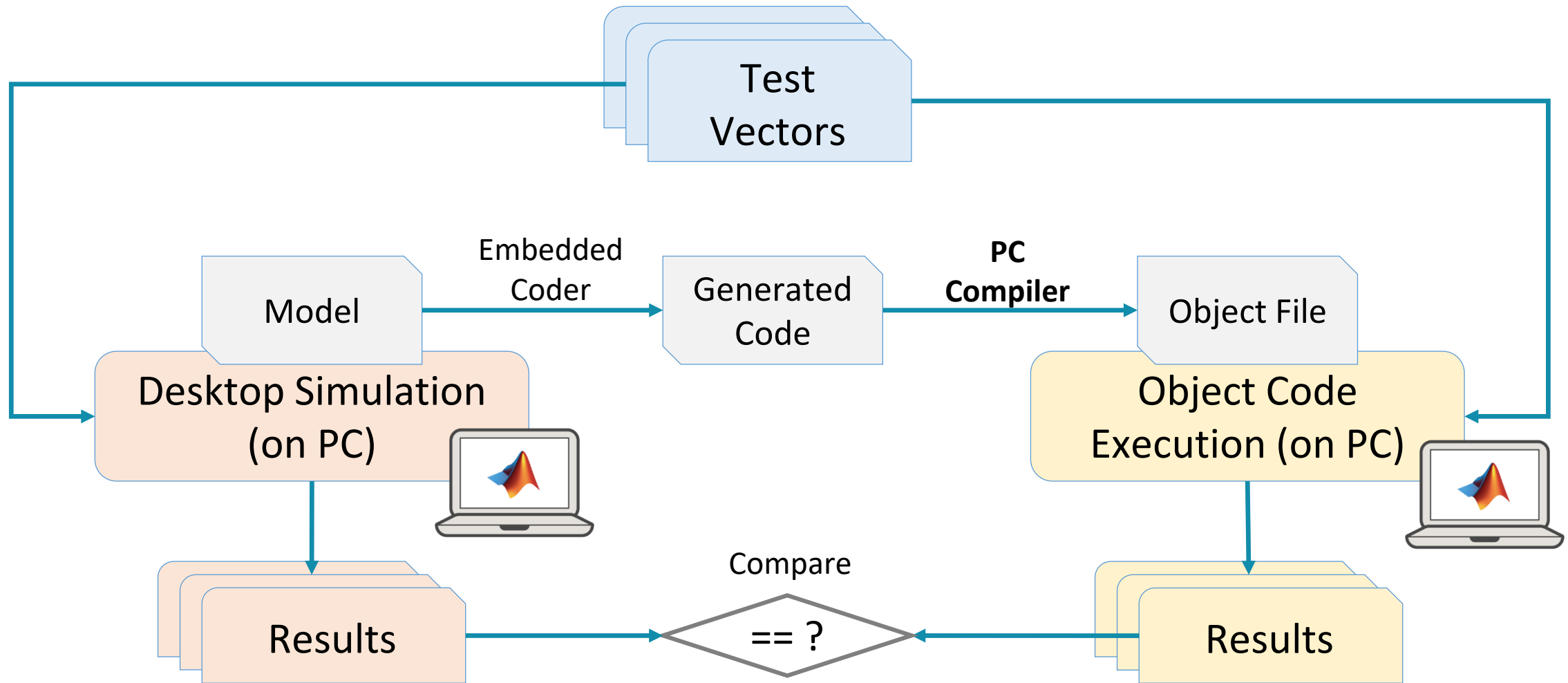
# Model Based Design Verification Workflow



- **Perform SIL Testing**
- Measure code coverage

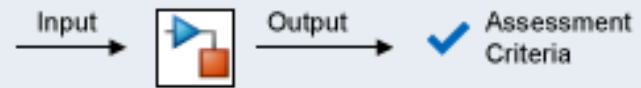


# Software In the Loop (SIL) Testing



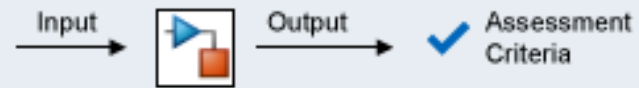
## Test Case Templates

### Simulation Test

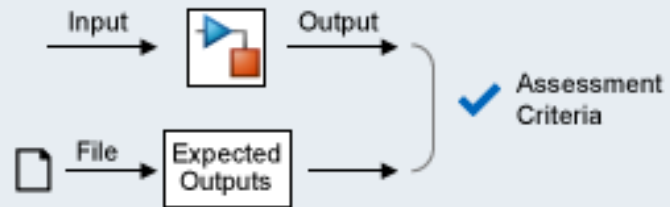


## Test Case Templates

### Simulation Test

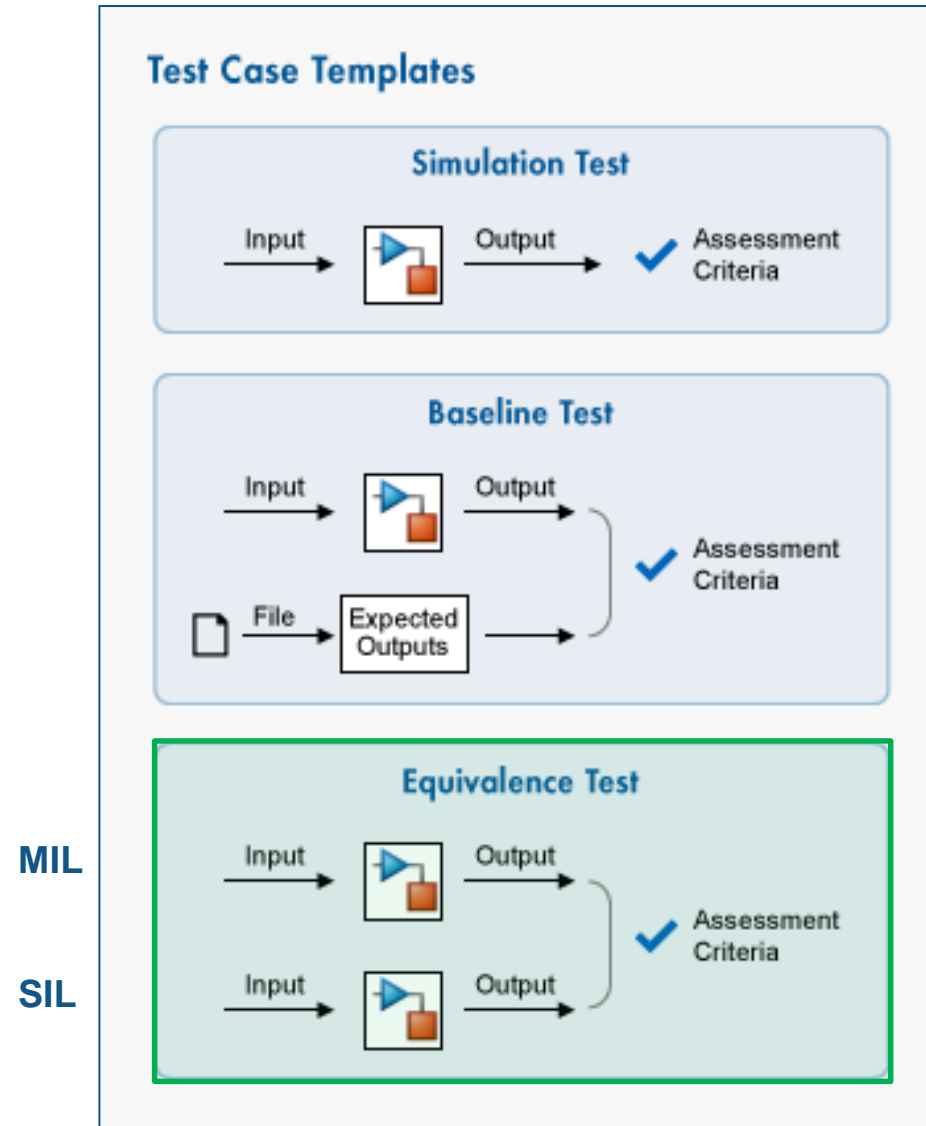


### Baseline Test

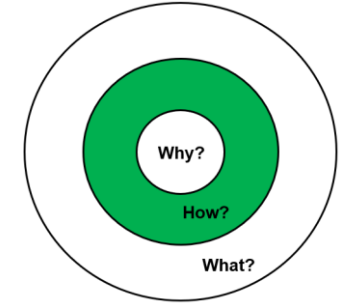




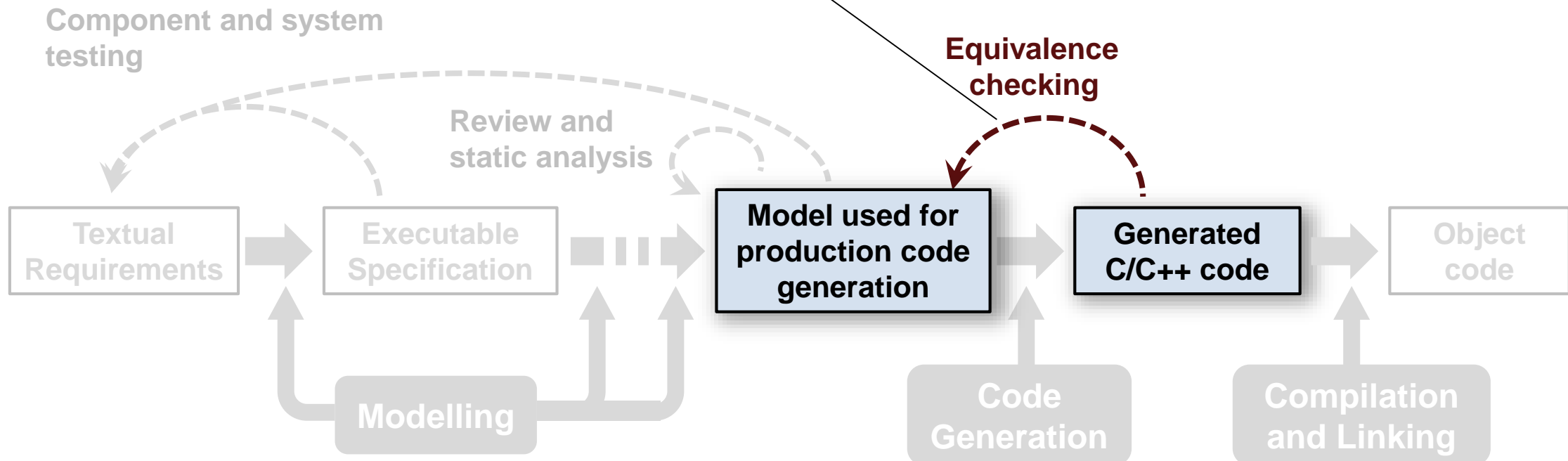
# MIL vs SIL



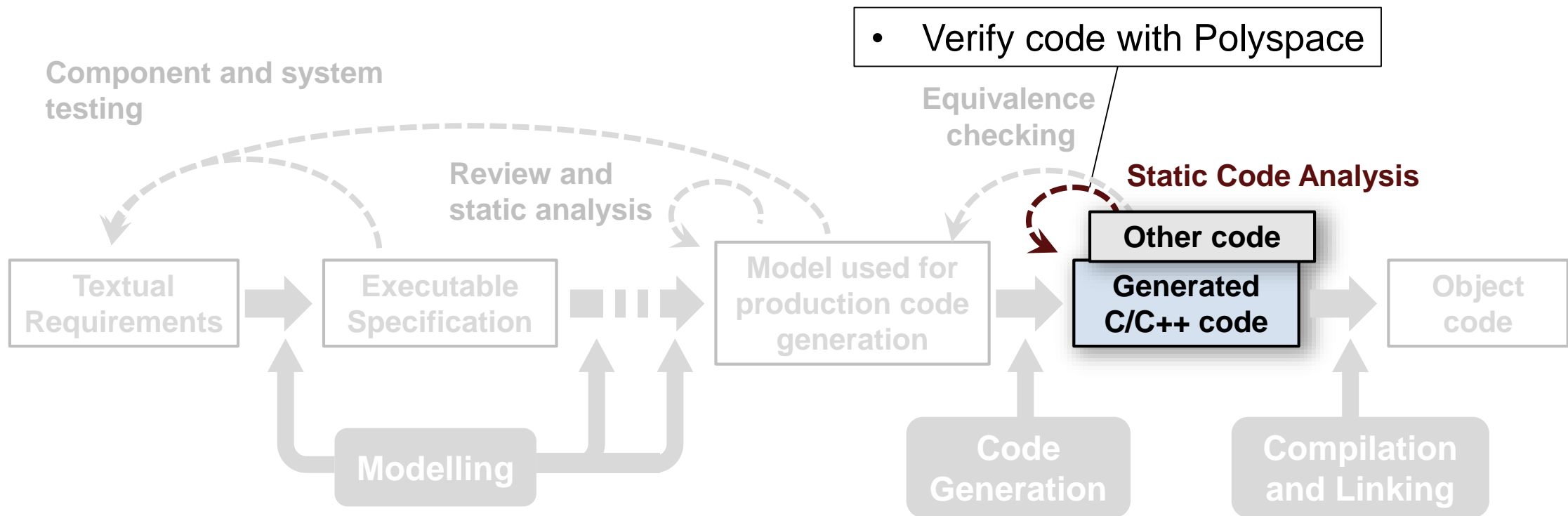
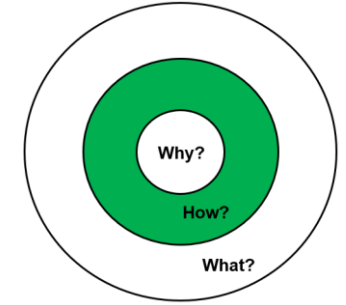
# Model Based Design Verification Workflow



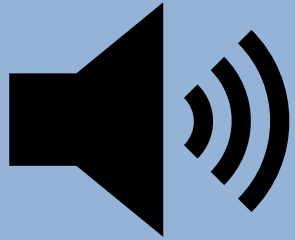
- Perform SIL Testing
- **Measure code coverage**



# Static Code Analysis



# Polyspace Tools



- ✓ C
- ✓ C++
- ✓ Ada

**Method:**

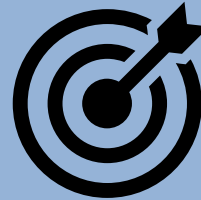
Formal Method based semantic analysis  
→ Abstract Syntax Interpretation



hand-written  
code

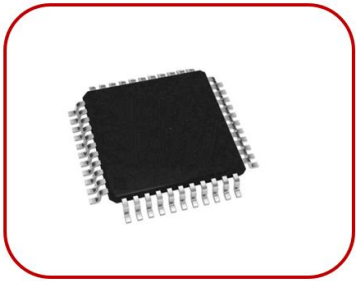


auto-generated  
code

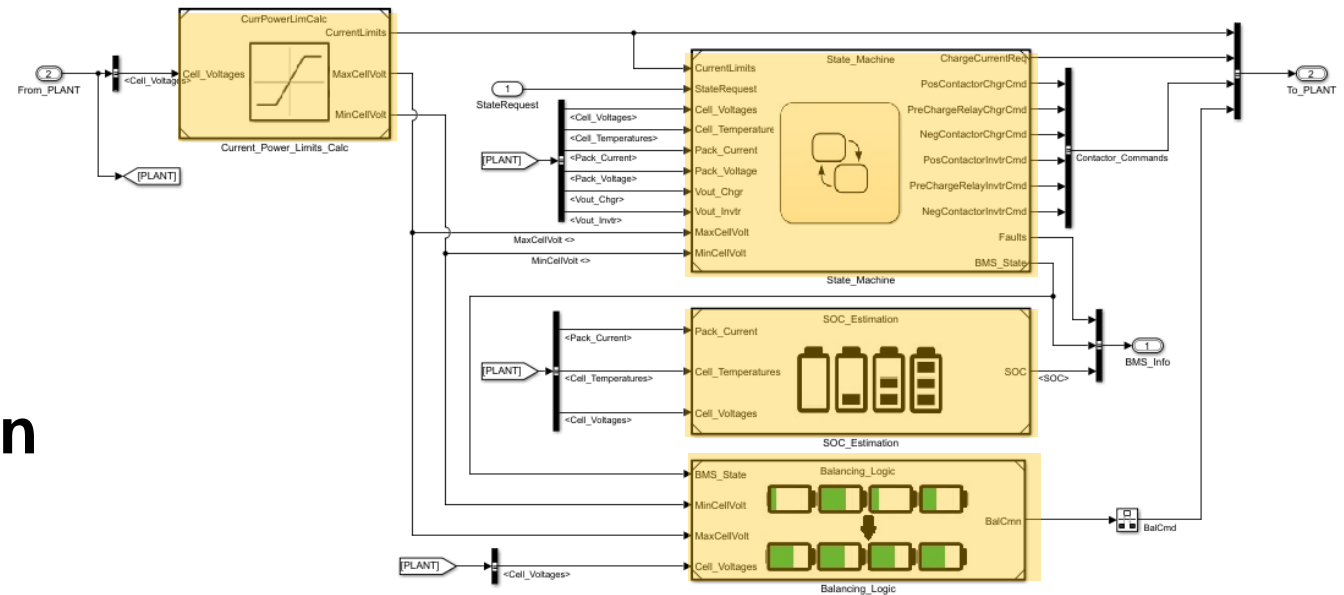
**Goal:**

- ✓ Find Runtime Errors (division by zero, overflows, etc.)
- ✓ Find MISRA violations
- ✓ Provide code metrics
- ✓ Prove absence of Runtime Errors

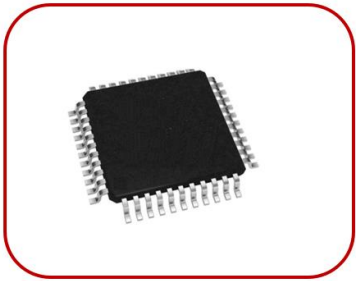
# Battery Management System (BMS)



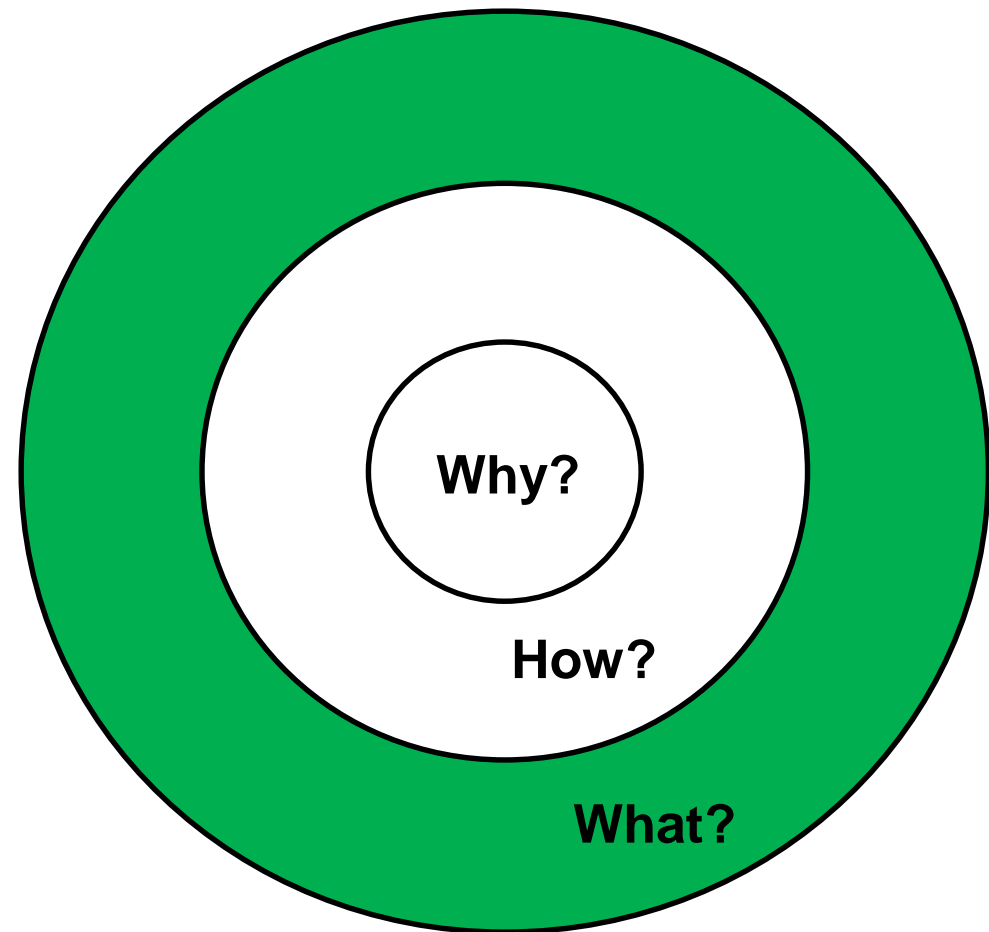
- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Integration Testing



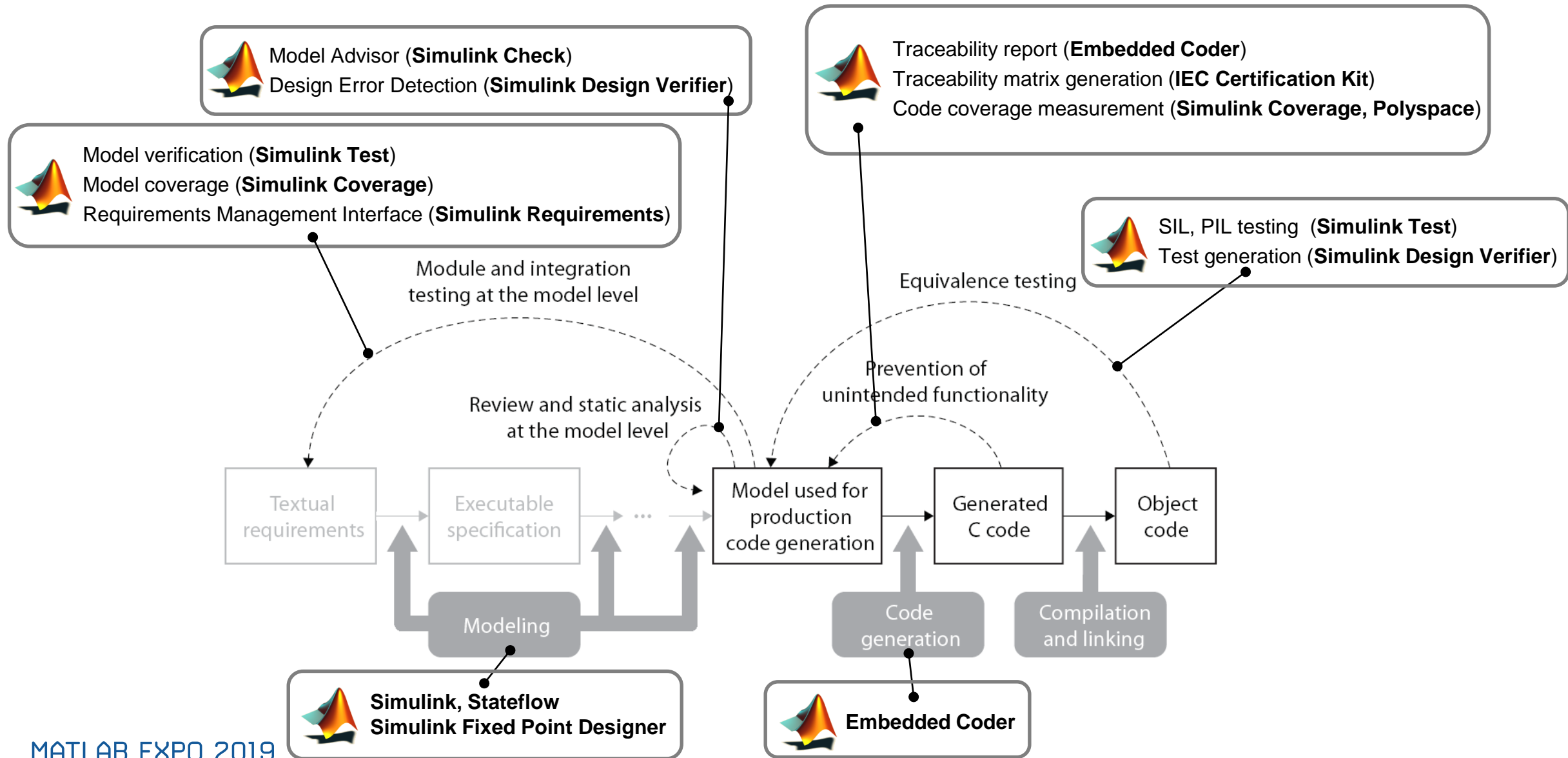
# Battery Management System (BMS)



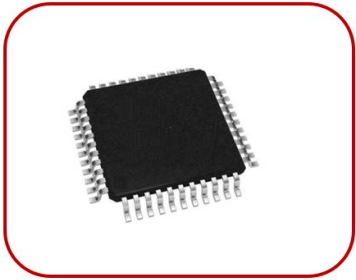
- Software Architecture
- SW Module Design
- **Test, Verification & Validation**
- BMS Integration Testing



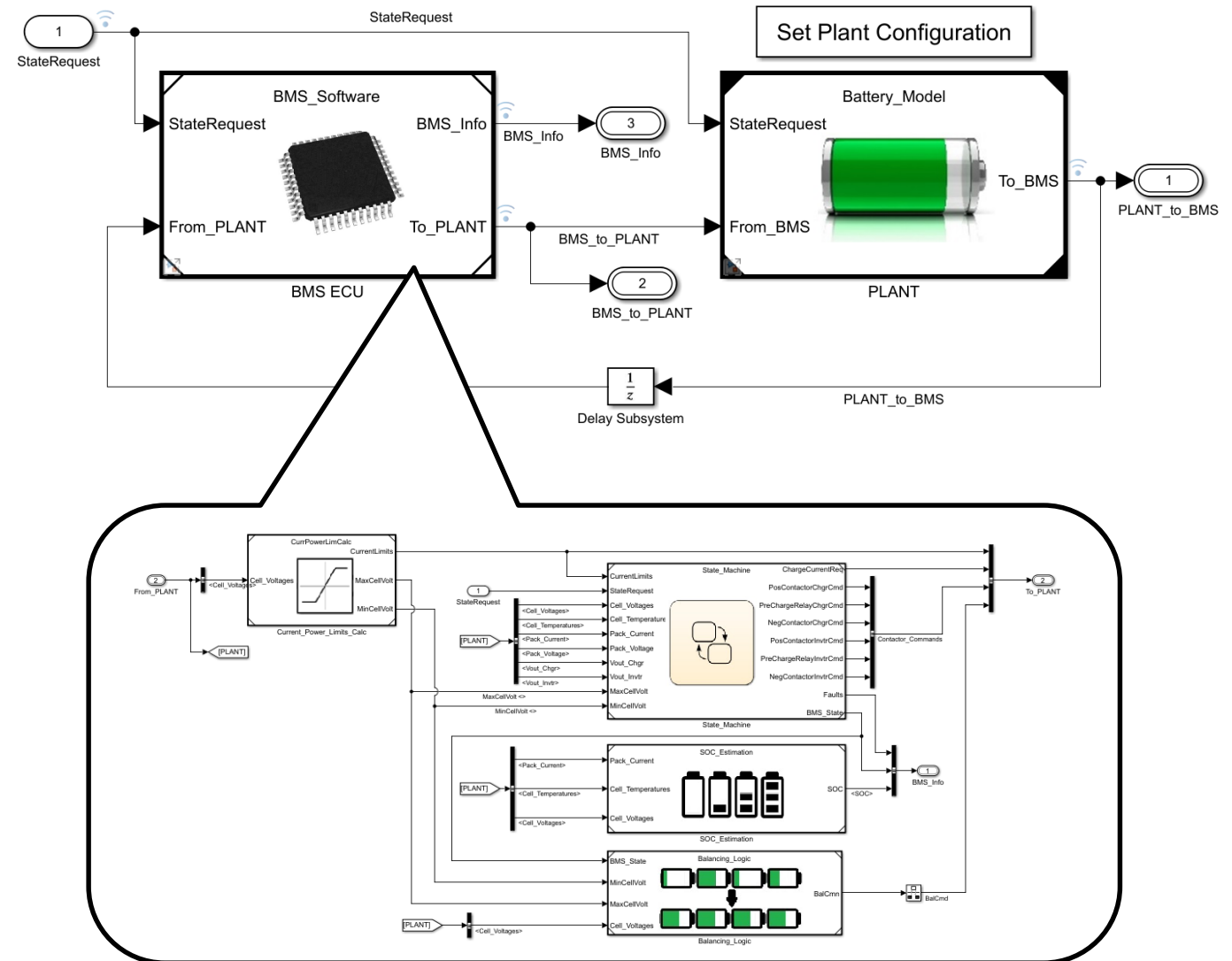
# Model Based Design Verification Workflow



# Battery Management System (BMS)

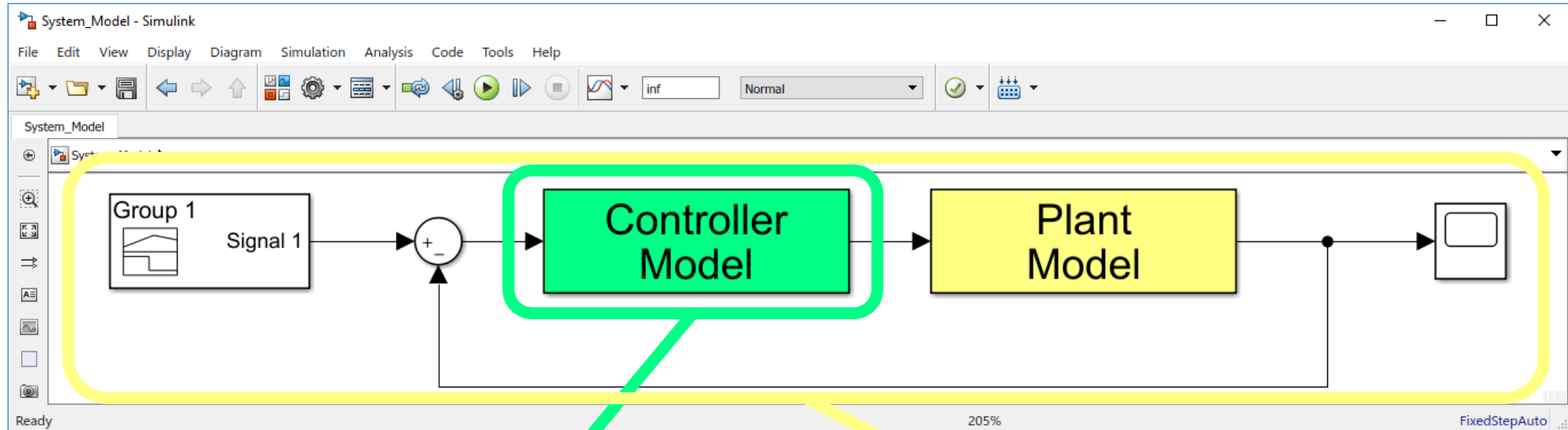


- Software Architecture
- SW Module Design
- Test, Verification & Validation
- **BMS Integration Testing**





# Real-time Simulation and Testing

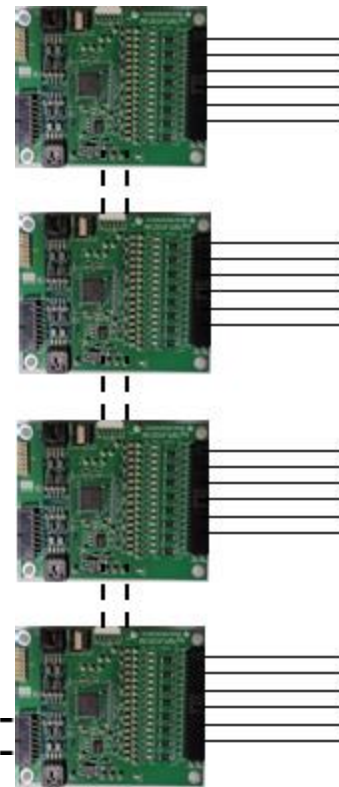
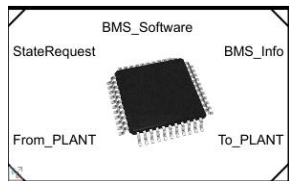


**Embedded Controller Hardware**

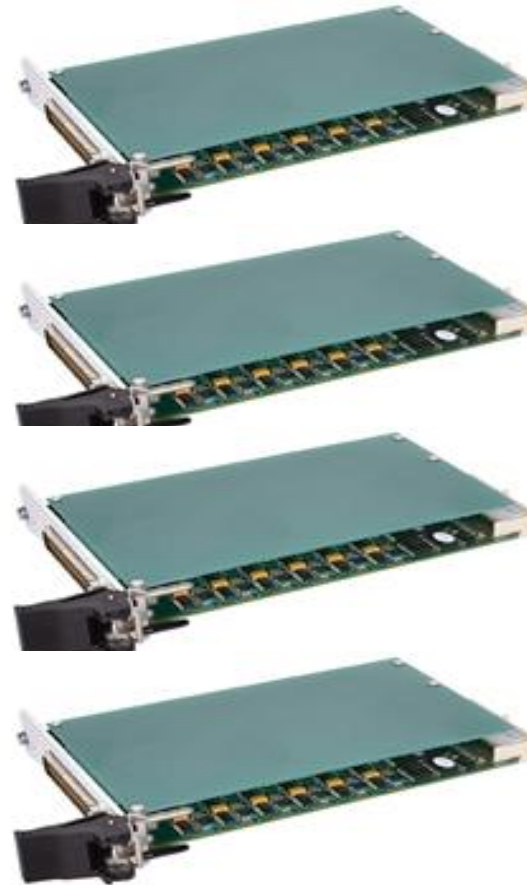
**Target Computer Hardware**



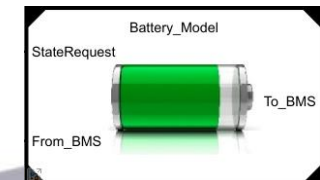
# HIL Testing – Speedgoat Solution



**6-Cell  
Modules**



**IO991-06**



# Q&A

