MathWorks
AUTOMOTIVE CONFERENCE 2023
Europe

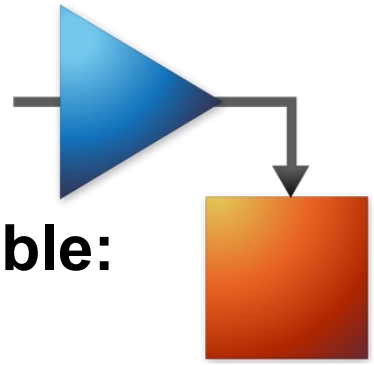
Streamline Automotive SPICE® Compliance Using Model-Based Design

Mohammad Abu-Alqumsan,
MathWorks

Marc Segelken,
MathWorks

MathWorks®

Key Takeaways

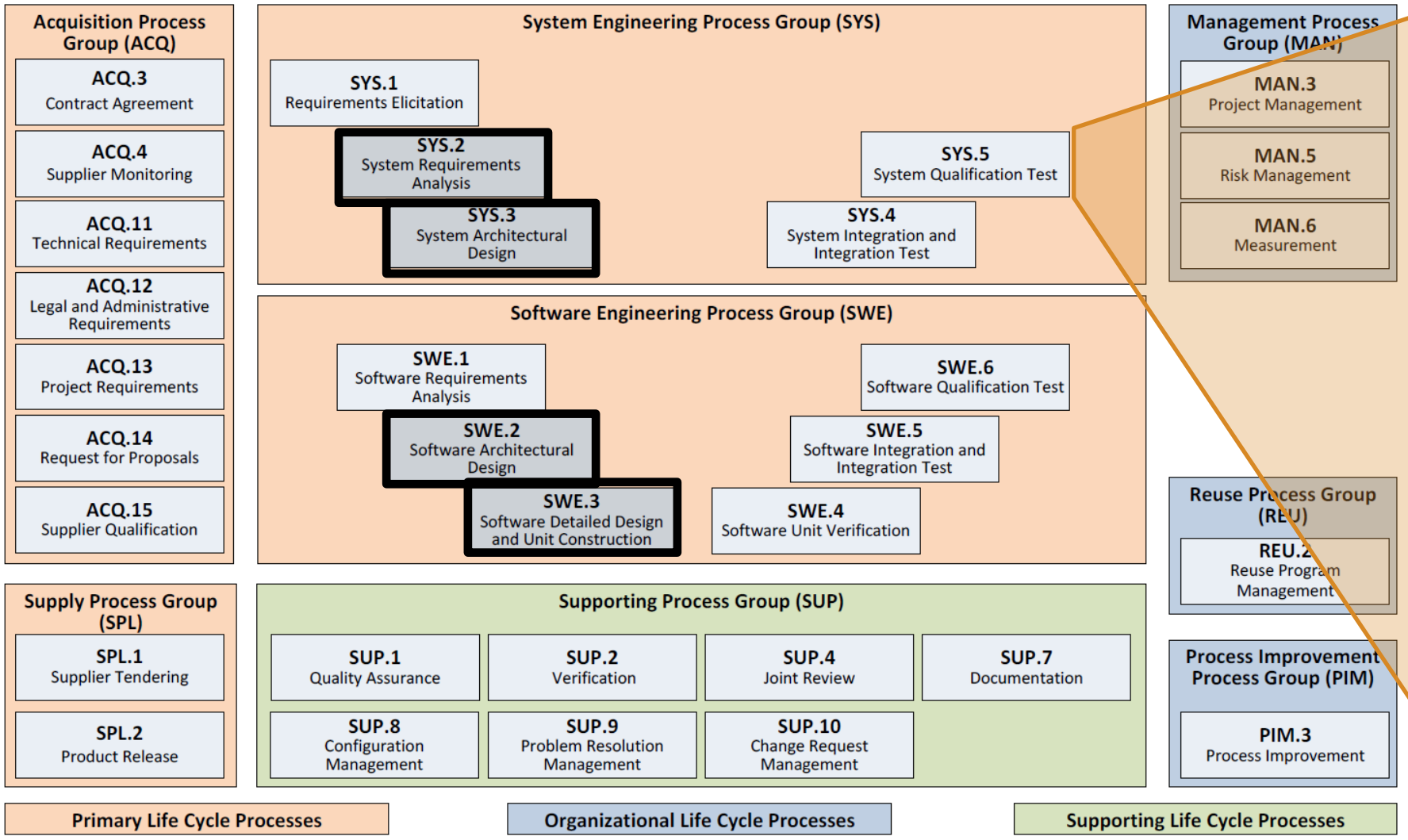


Model-Based Design and Model-Based Systems Engineering enable:

1. **Fast development and realization** of **system and software** architecture and design
2. **Early testing** to detect errors in designs and their realization
3. **Fast and efficient iterations**

Develop **high quality products** following an **efficient Automotive SPICE® compliant process**

Automotive SPICE® – Reference Model



ID:
SYS.5

Name:
System Qualification Test

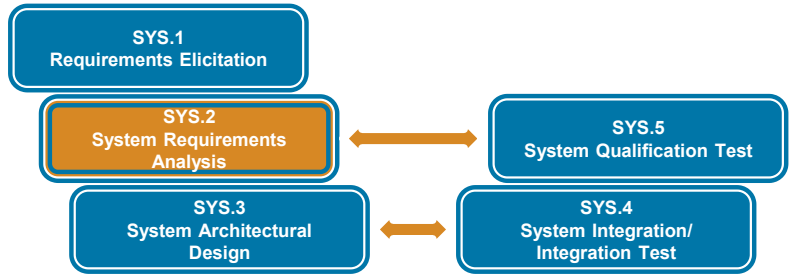
Purpose:

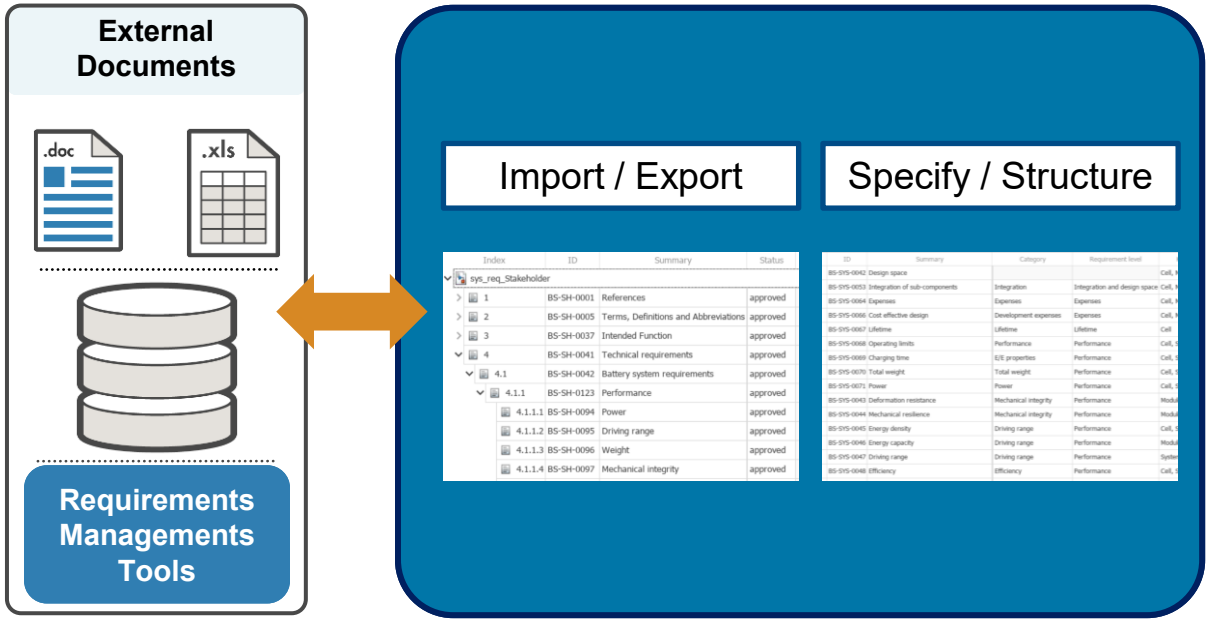
Outcomes:

Base Practices:

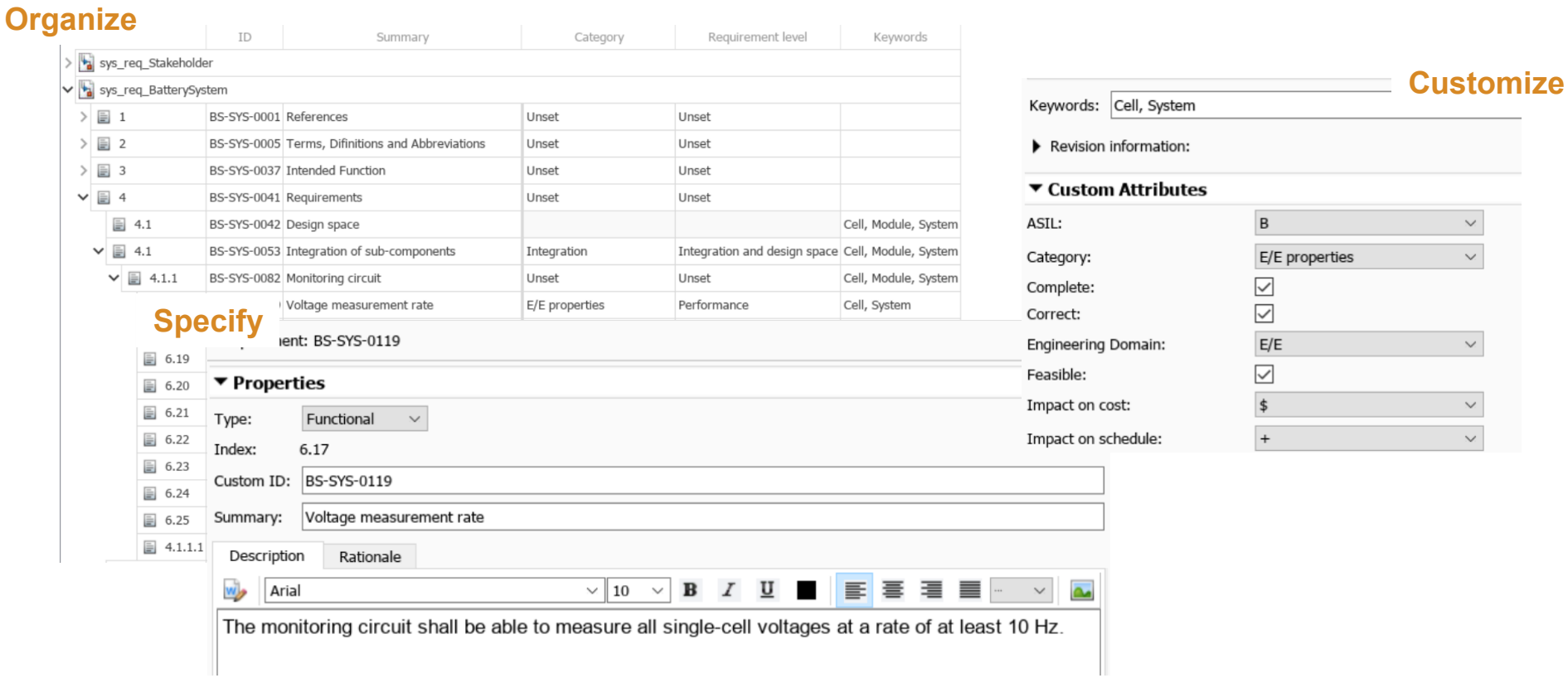
Output Work Products:

SYS.2 System Requirements Analysis

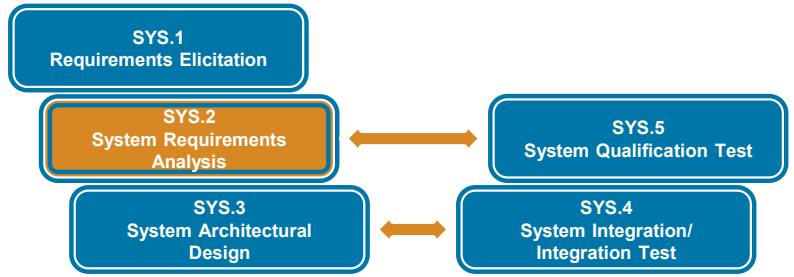


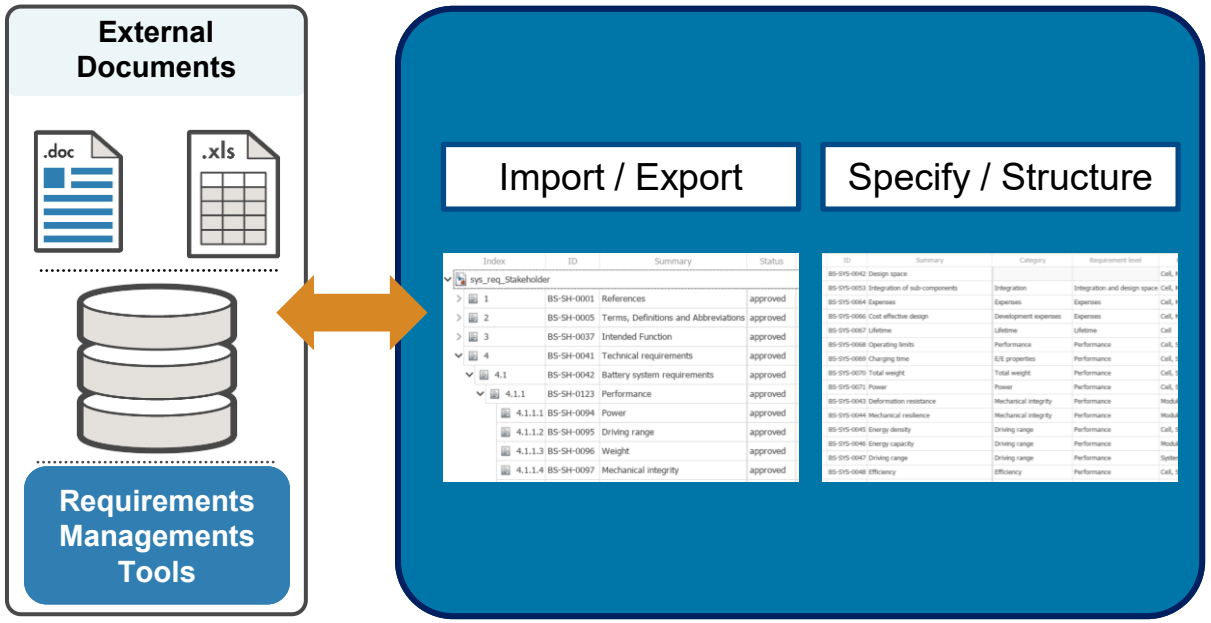


Organize, Specify and Customize Requirements with Requirements Toolbox

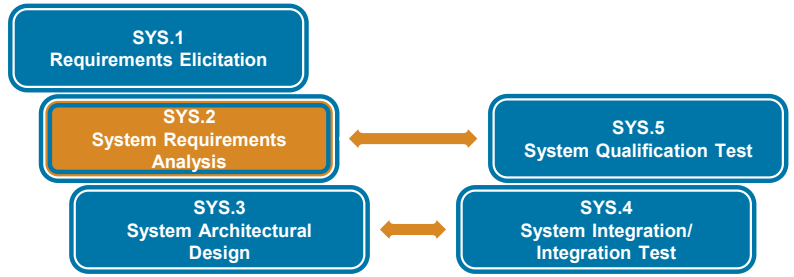


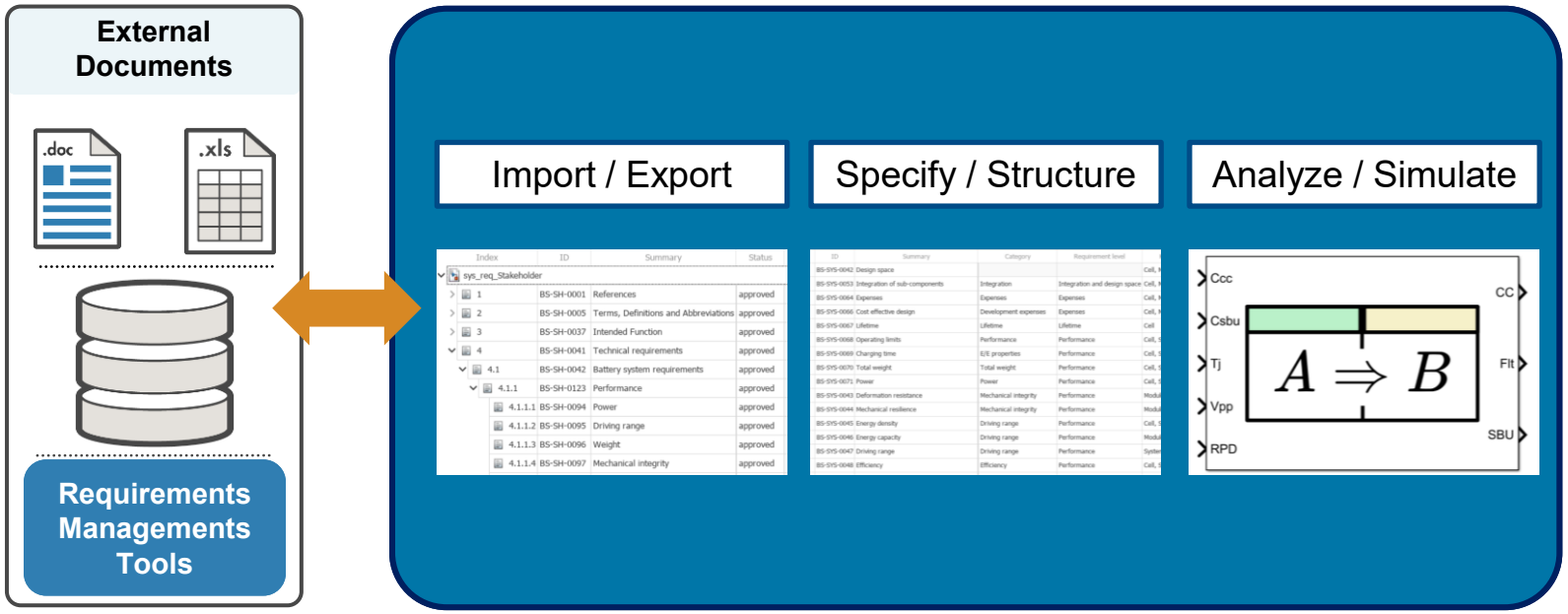
SYS.2 System Requirements Analysis





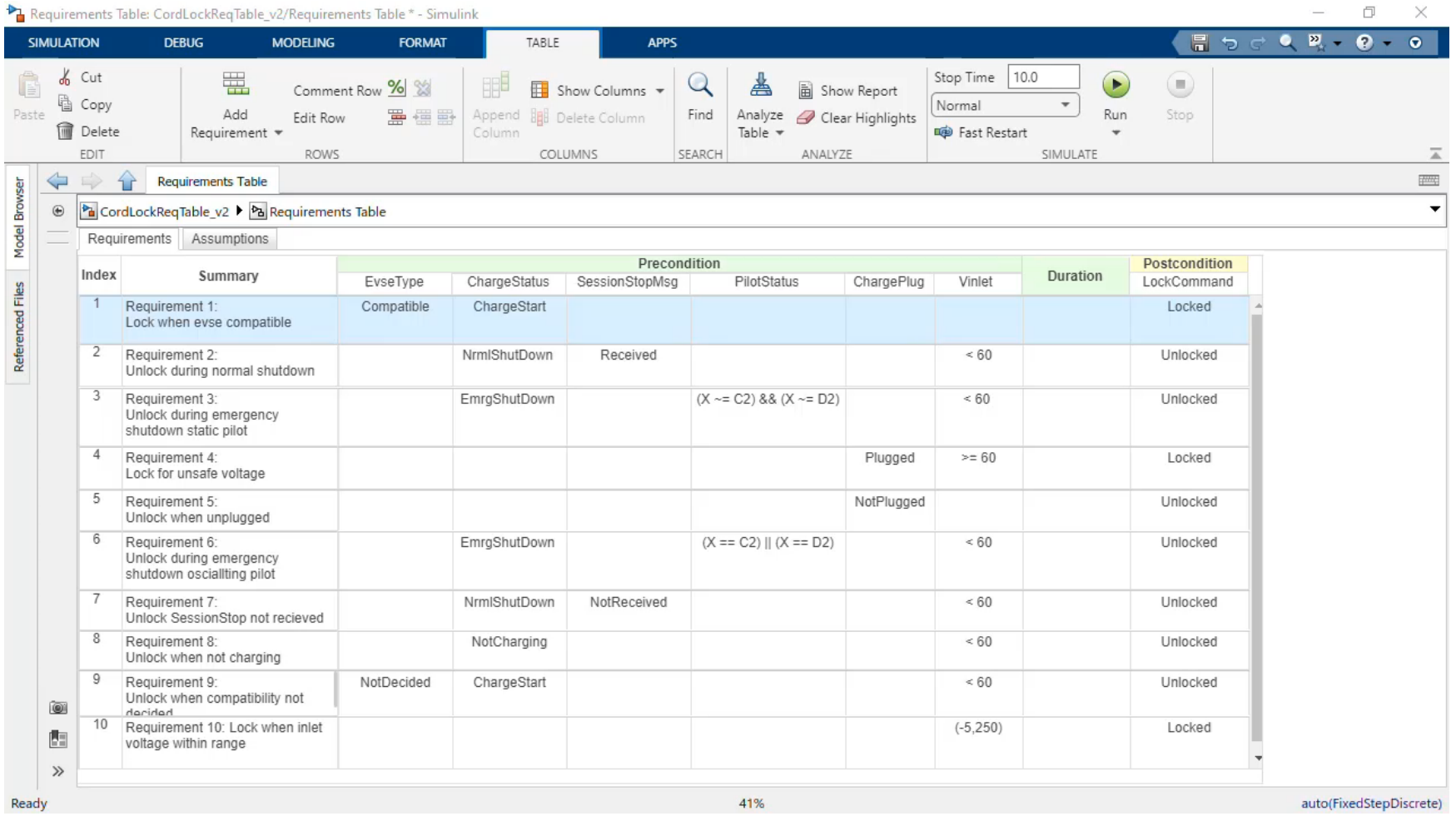
SYS.2 System Requirements Analysis





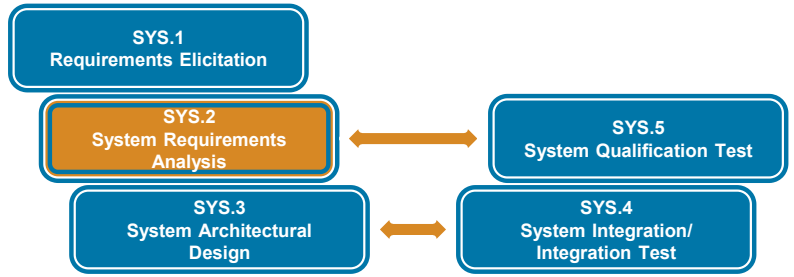
Analyze Logical Requirements with the Requirements Table

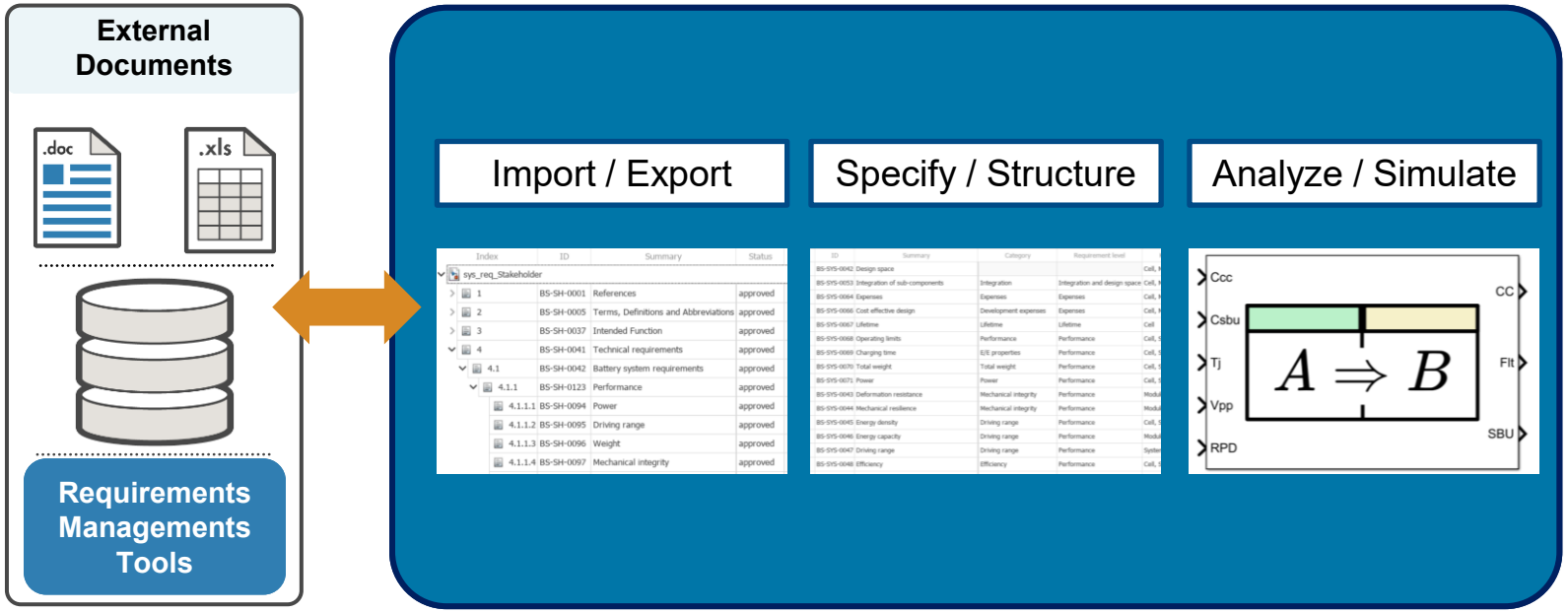
R2022a



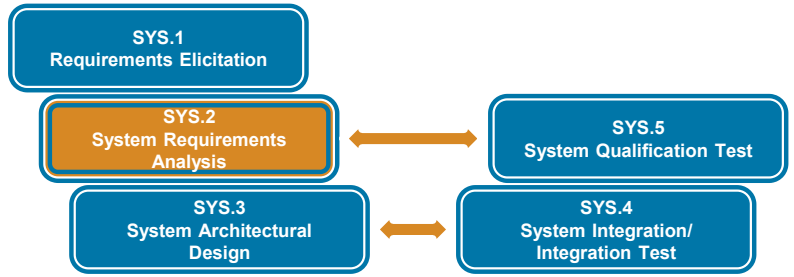
Index	Summary	Precondition						Duration	Postcondition
		EvseType	ChargeStatus	SessionStopMsg	PilotStatus	ChargePlug	VinLet		LockCommand
1	Requirement 1: Lock when evse compatible	Compatible	ChargeStart						Locked
2	Requirement 2: Unlock during normal shutdown		NrmlShutDown	Received			< 60		Unlocked
3	Requirement 3: Unlock during emergency shutdown static pilot		EmrgShutDown		(X ~= C2) && (X ~= D2)		< 60		Unlocked
4	Requirement 4: Lock for unsafe voltage					Plugged	>= 60		Locked
5	Requirement 5: Unlock when unplugged					NotPlugged			Unlocked
6	Requirement 6: Unlock during emergency shutdown oscialliting pilot		EmrgShutDown		(X == C2) || (X == D2)		< 60		Unlocked
7	Requirement 7: Unlock SessionStop not recieved		NrmlShutDown	NotReceived			< 60		Unlocked
8	Requirement 8: Unlock when not charging		NotCharging				< 60		Unlocked
9	Requirement 9: Unlock when compatibility not decided	NotDecided	ChargeStart				< 60		Unlocked
10	Requirement 10: Lock when inlet voltage within range						(-5,250)		Locked

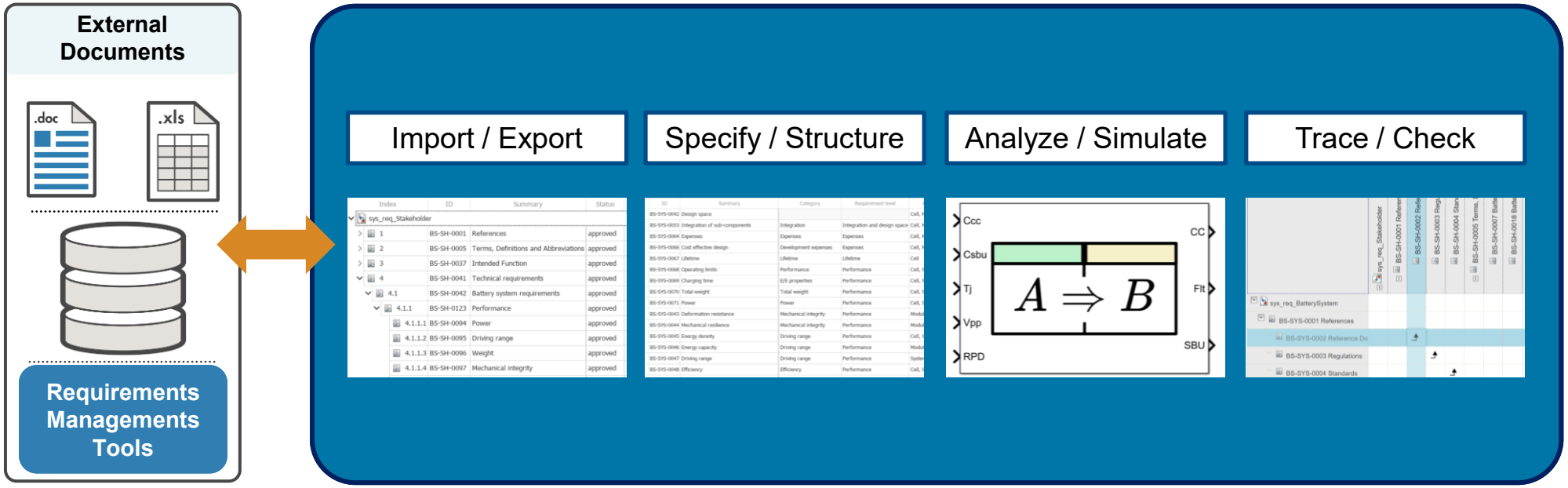
SYS.2 System Requirements Analysis





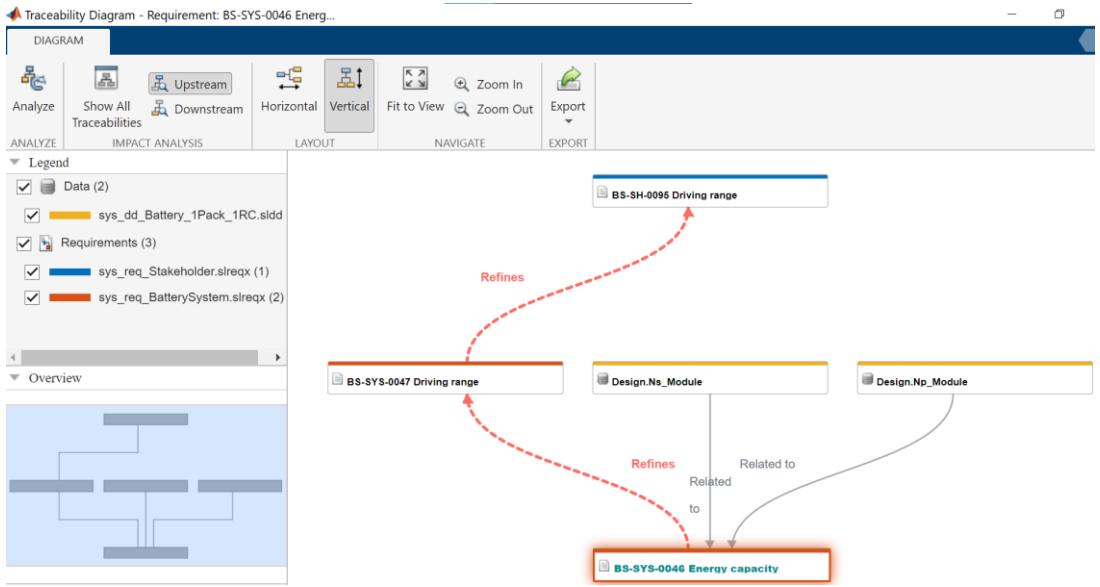
SYS.2 System Requirements Analysis



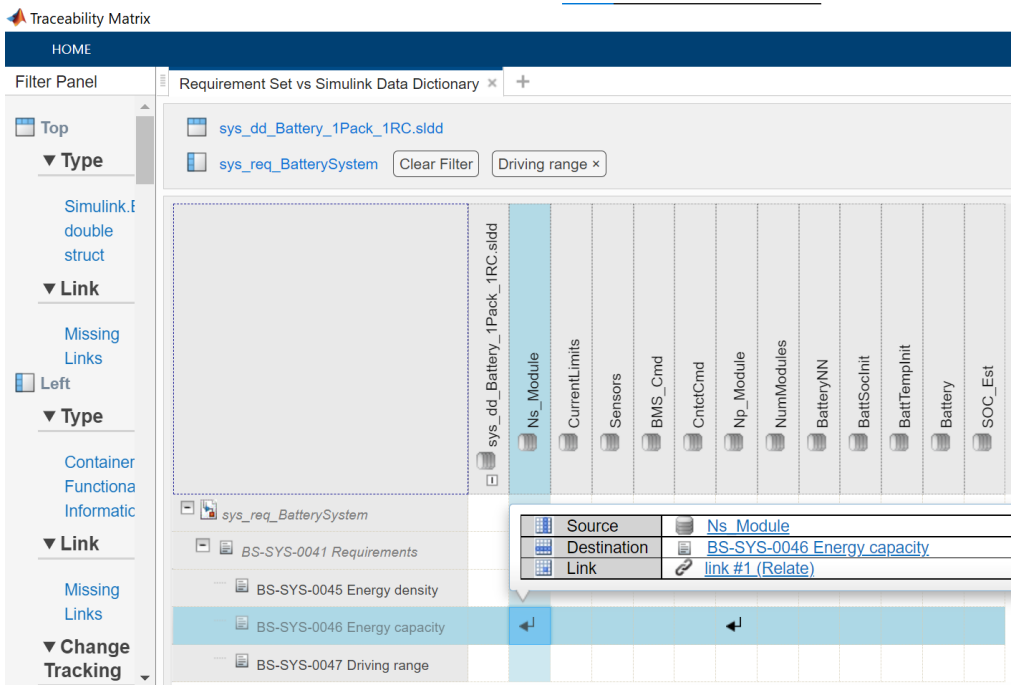


Use Traceability Diagrams and Matrixes to Check for Consistency and Completeness

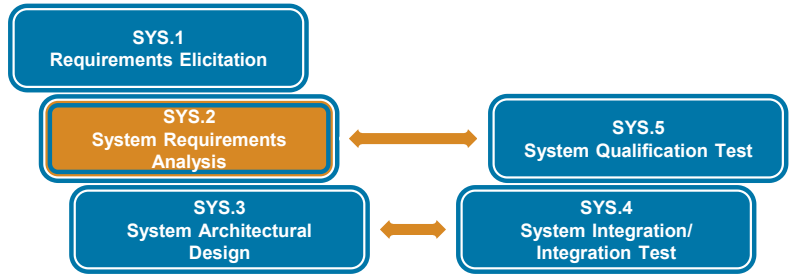
Traceability Diagrams



Traceability Matrix



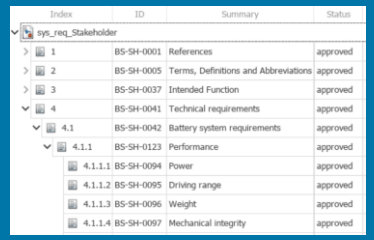
SYS.2 System Requirements Analysis



External Documents

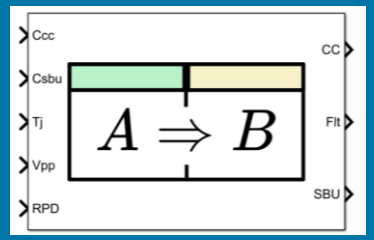
Requirements Managements Tools

Import / Export

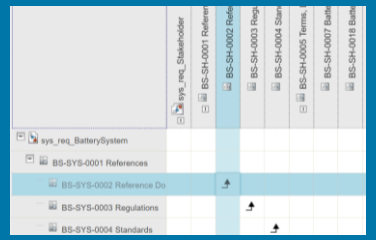


Specify / Structure

Analyze / Simulate



Trace / Check



Output Work Products

System requirements

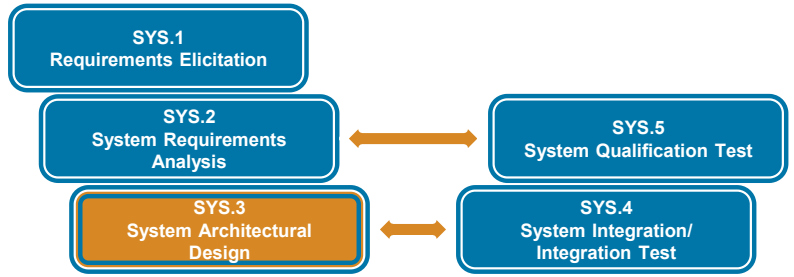
Interfaces

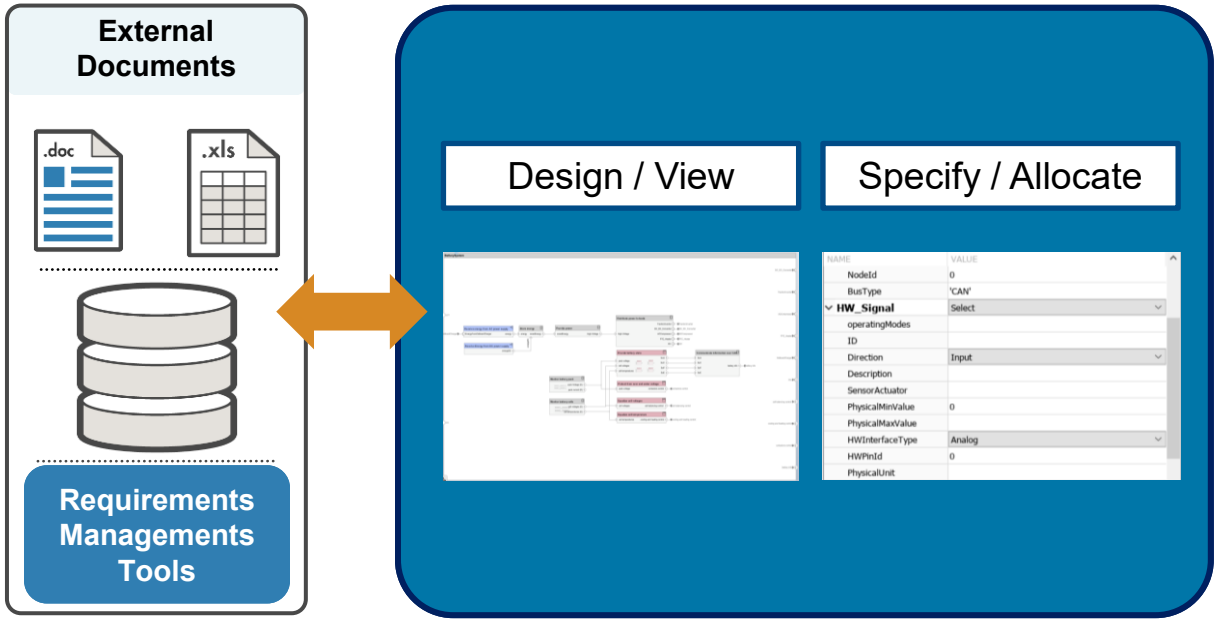
Traceability

Analysis report

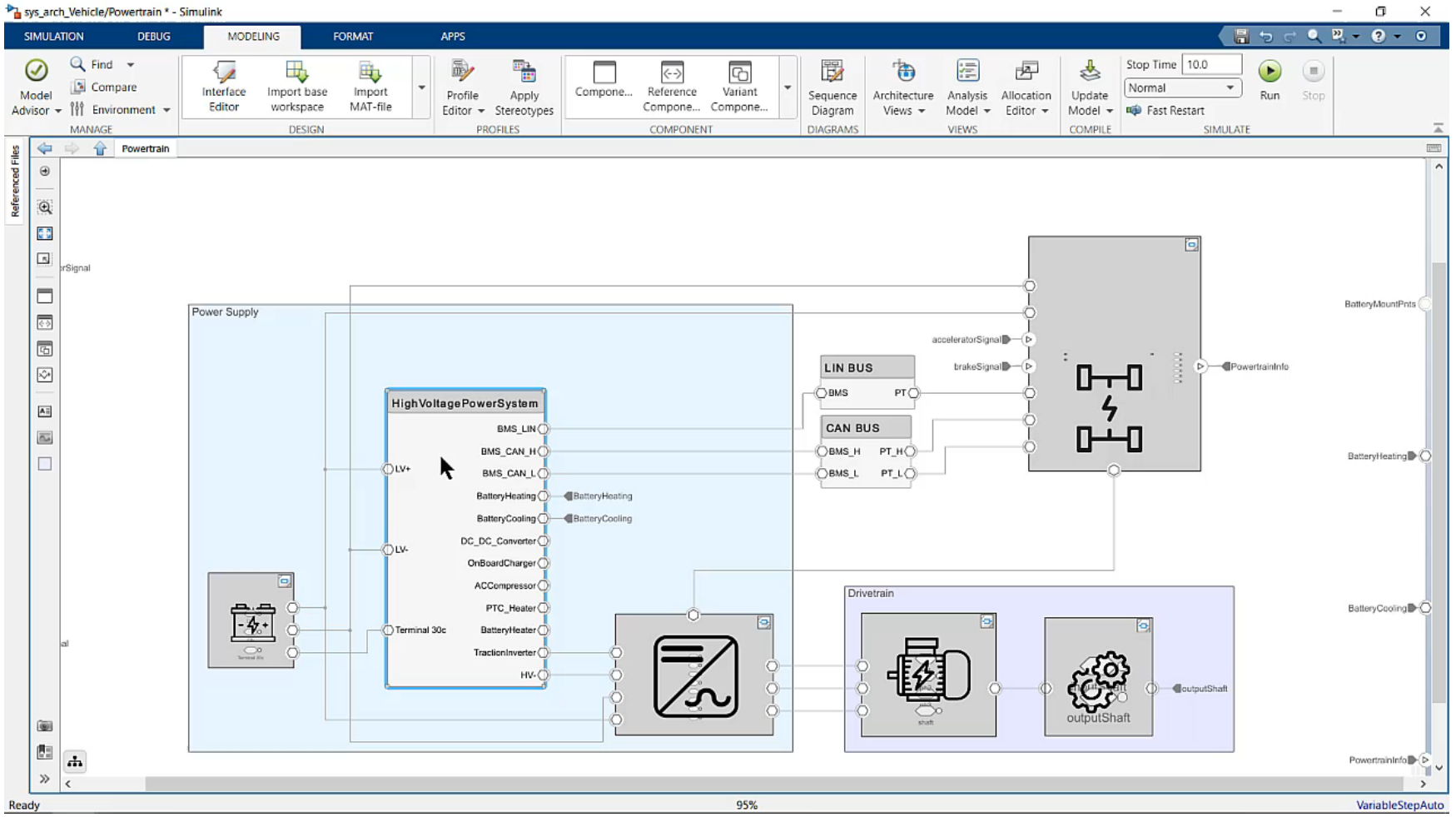
Verification criteria

SYS.3 System Architectural Design

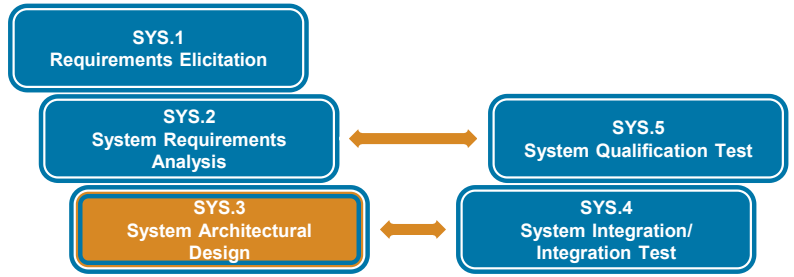


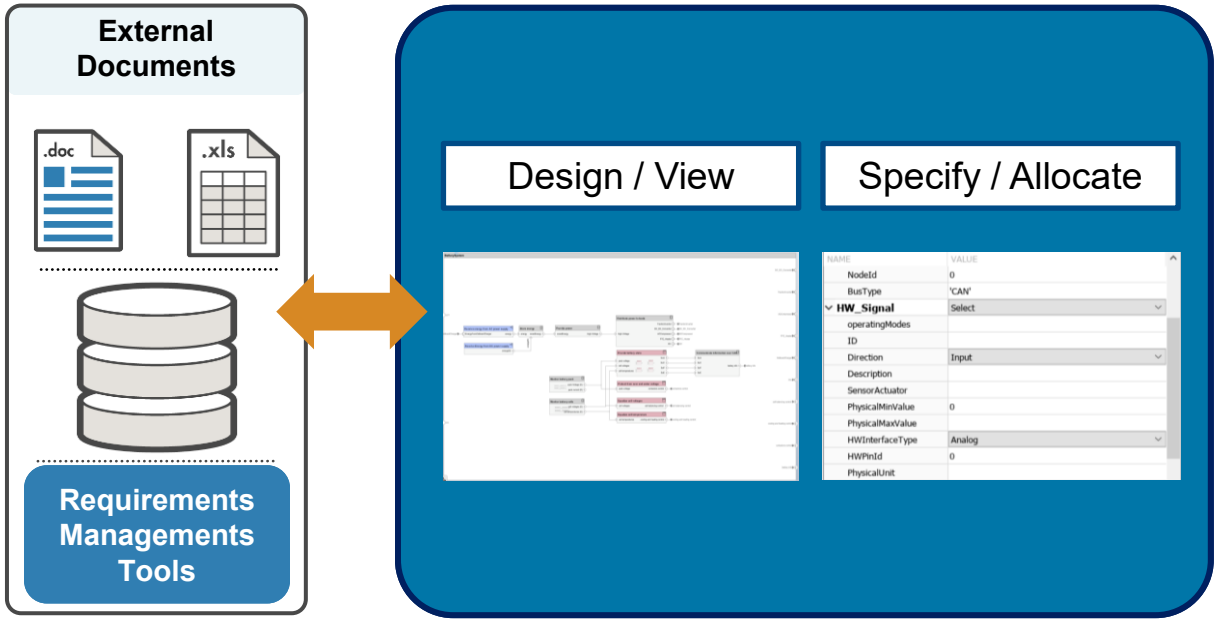


Develop Architectural Design Models with System Composer

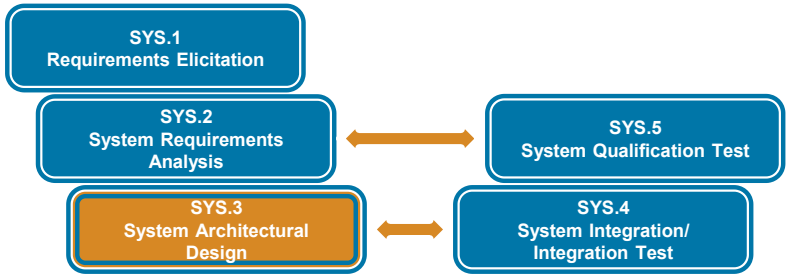


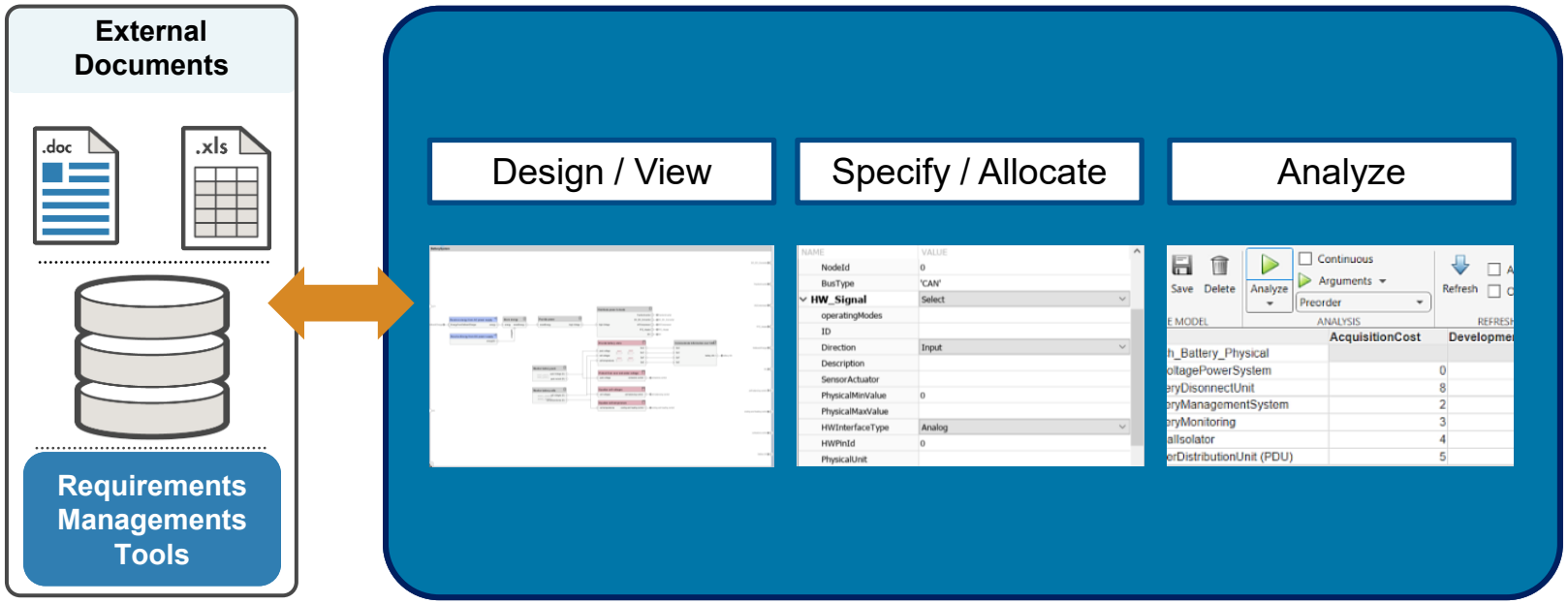
SYS.3 System Architectural Design



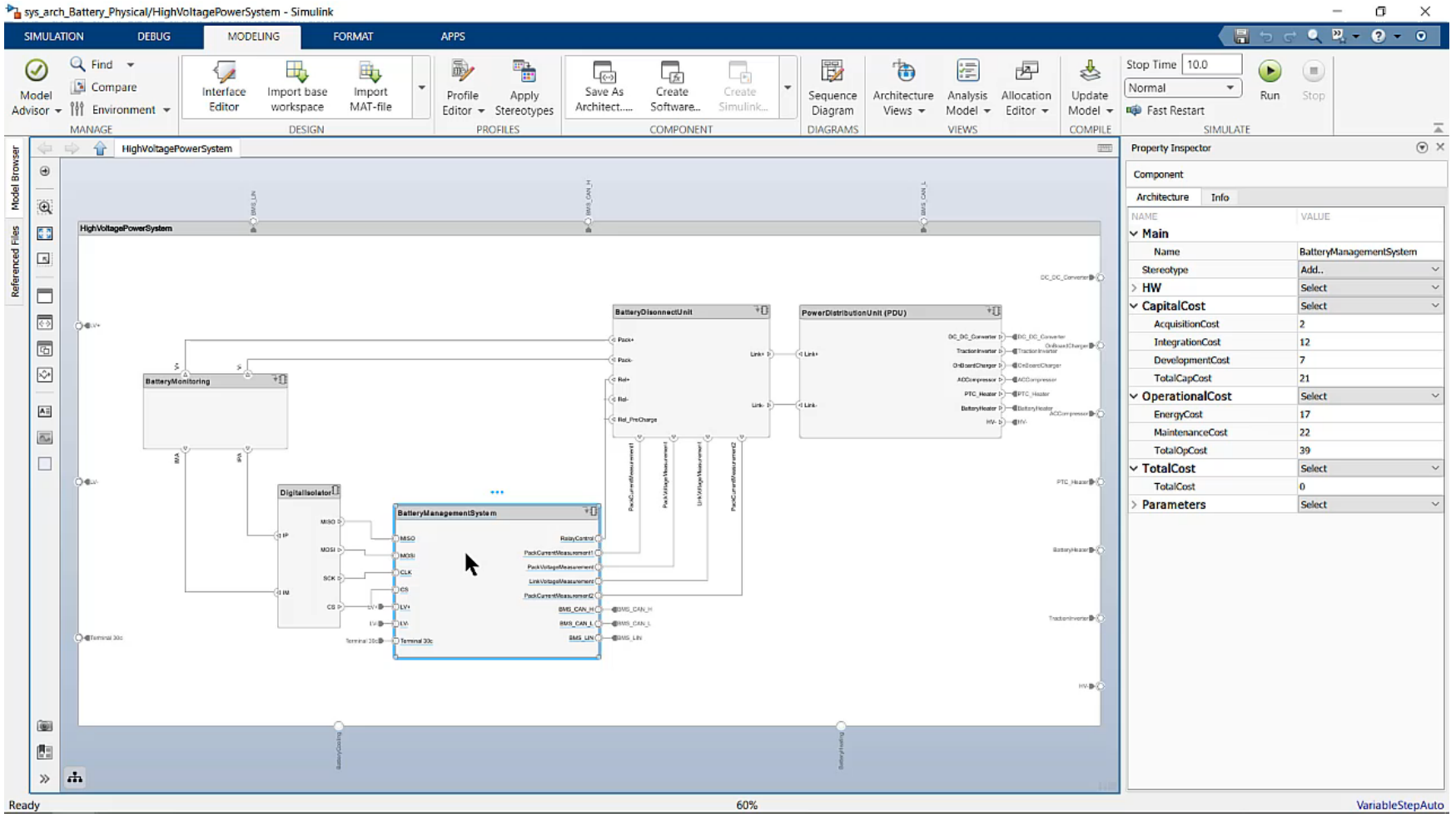


SYS.3 System Architectural Design

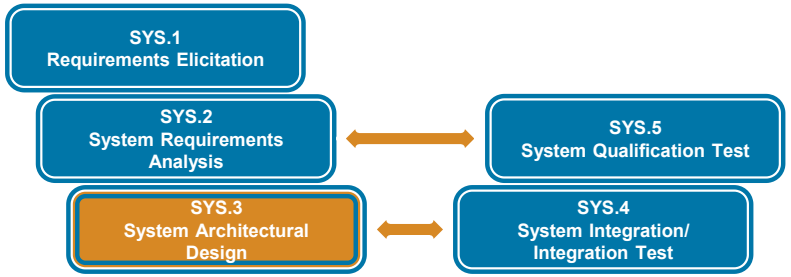


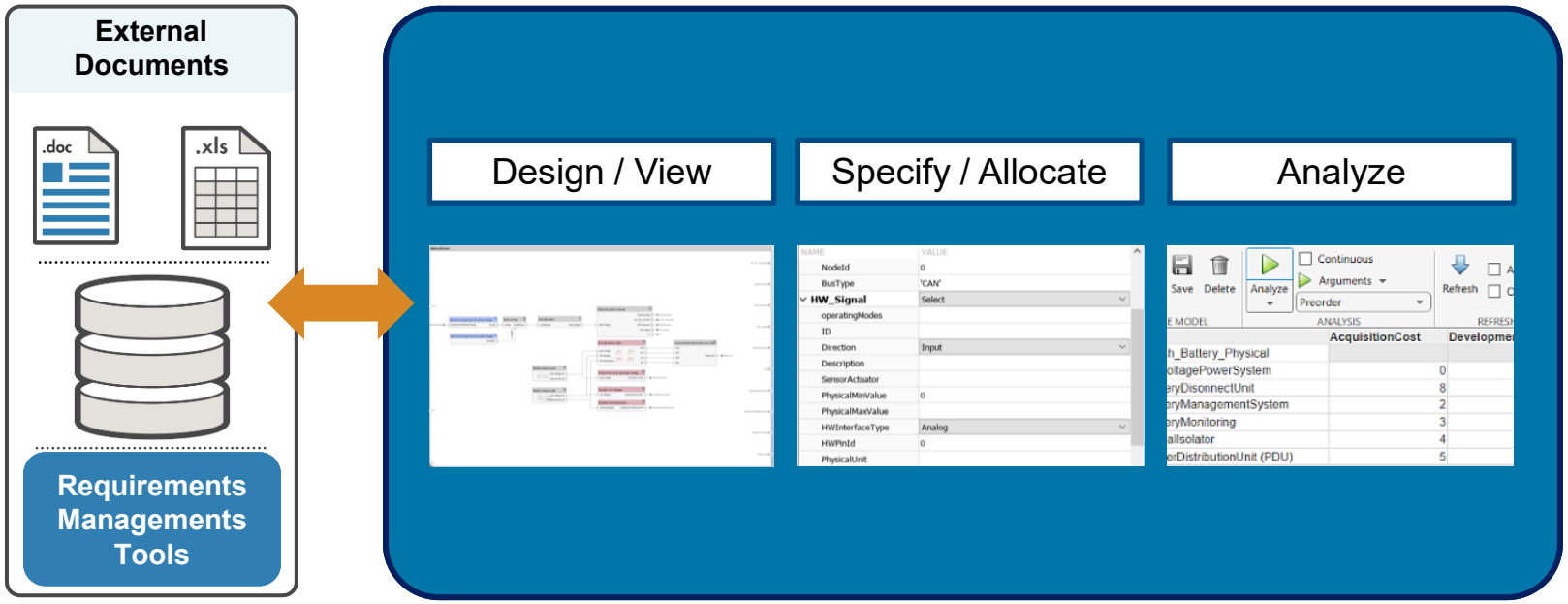


Analyze Architectural Design Models with System Composer

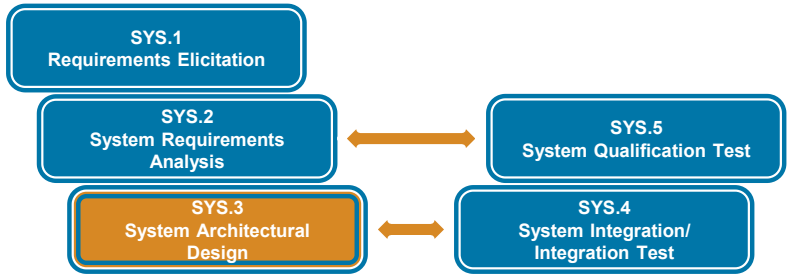


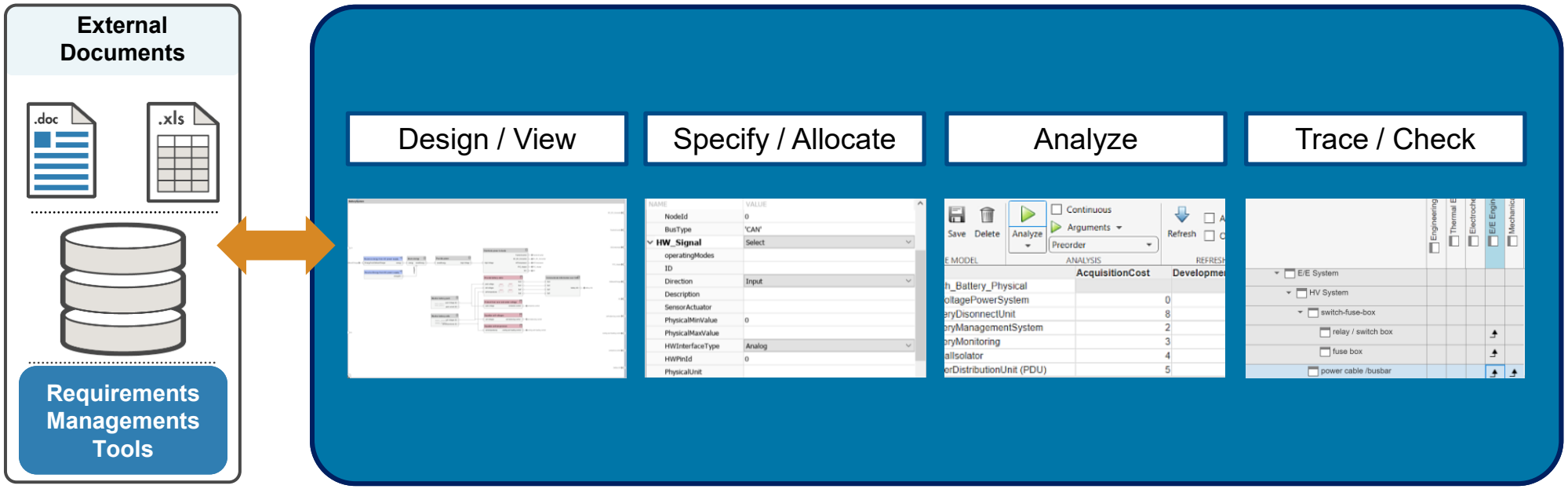
SYS.3 System Architectural Design





SYS.3 System Architectural Design

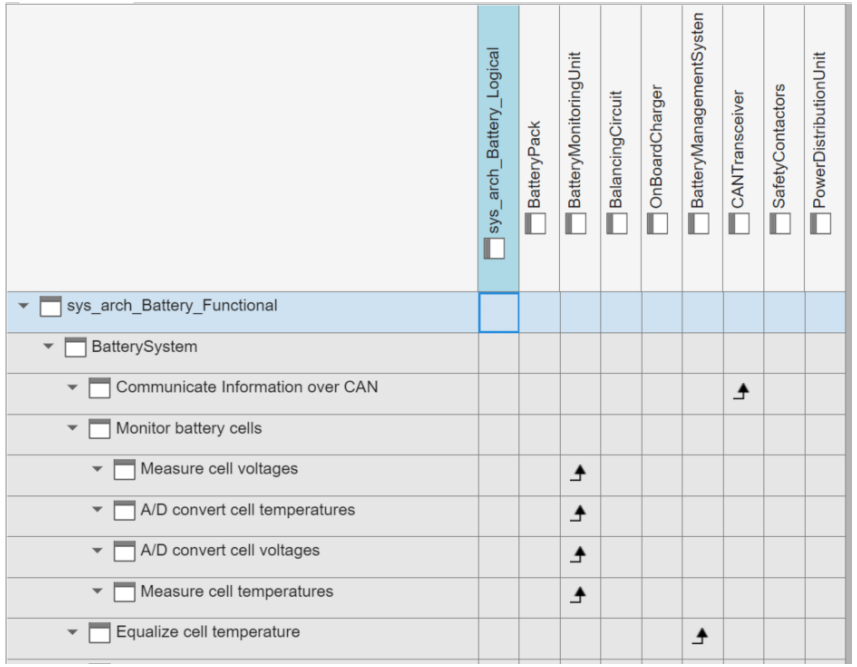




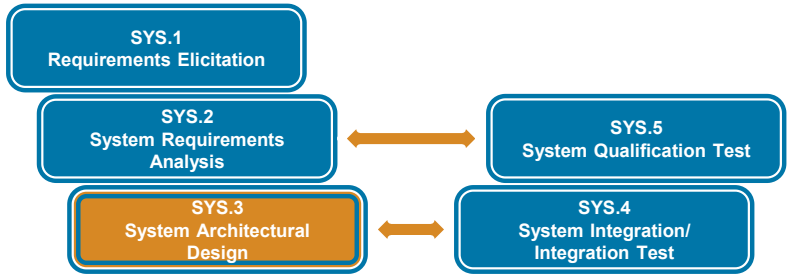
Ensure Consistency with Tool Support for Bidirectional Traceability

Requirements ↔ Architecture

Architecture ↔ Architecture



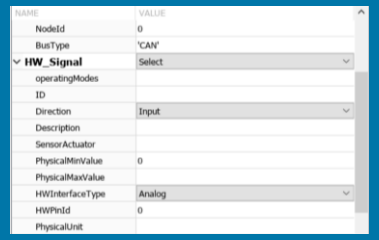
SYS.3 System Architectural Design

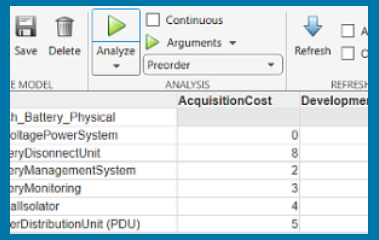


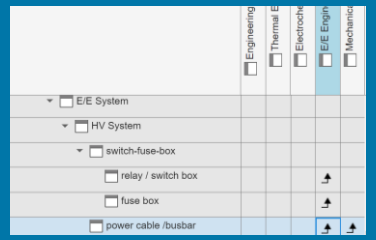
External Documents

Requirements Managements Tools

Design / View | Specify / Allocate | Analyze | Trace / Check







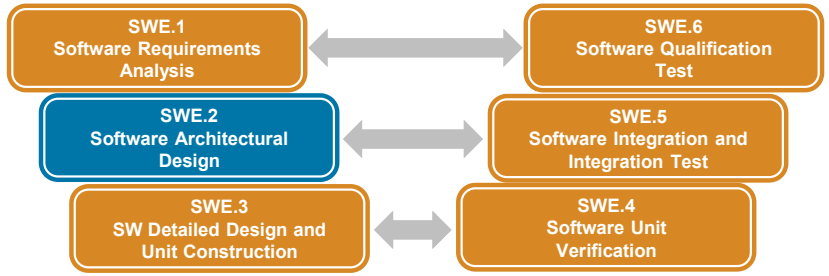
Output Work Products

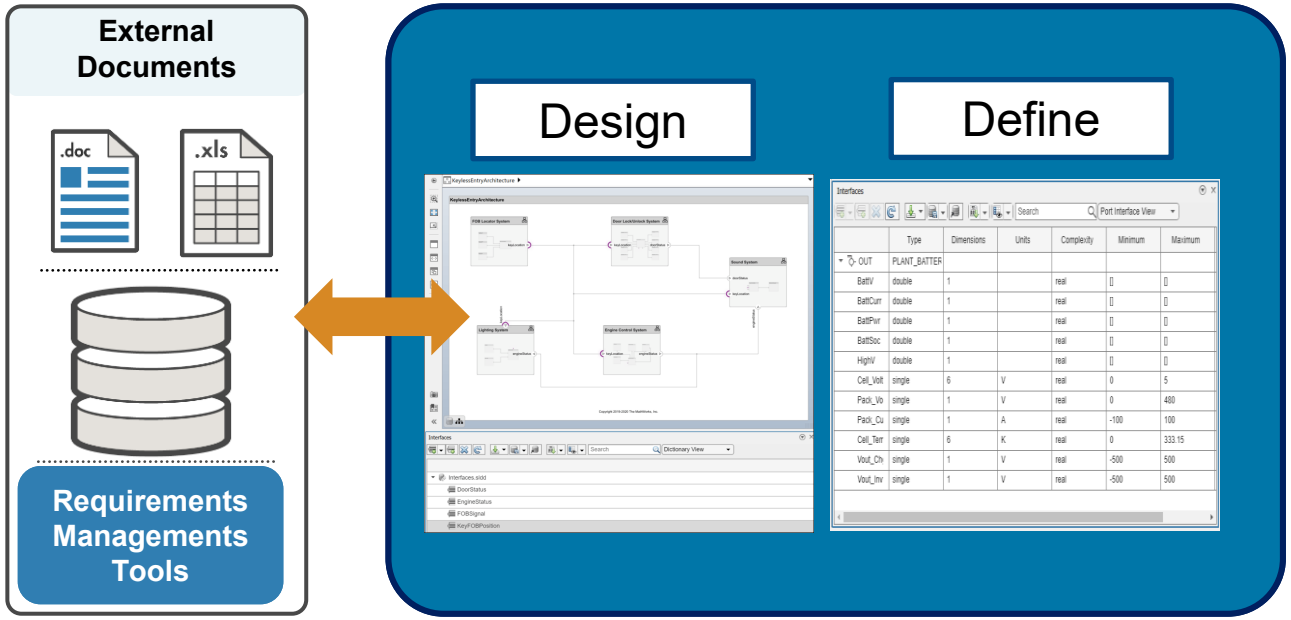
System architecture

Interfaces

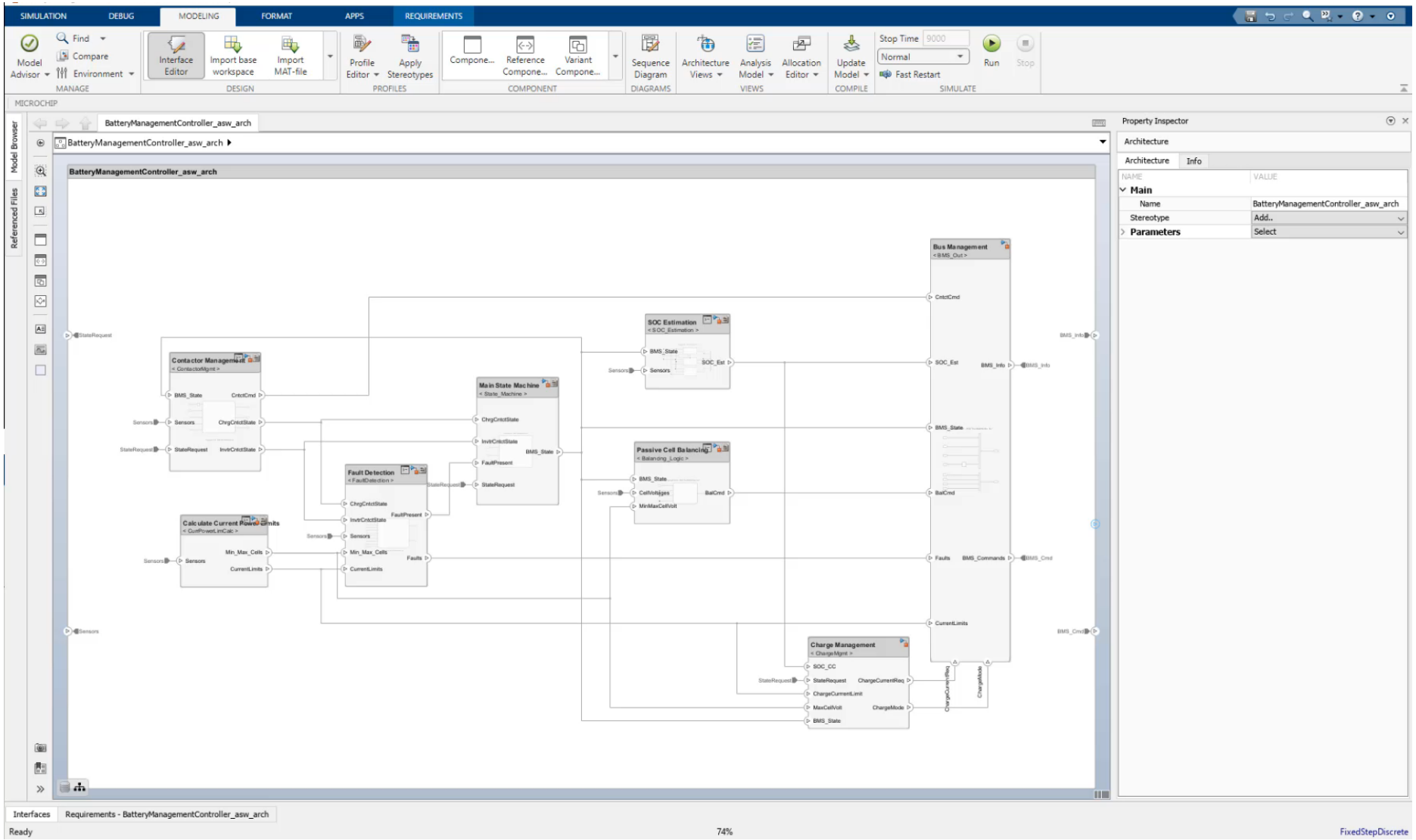
Traceability

SWE.2 Software Architectural Design

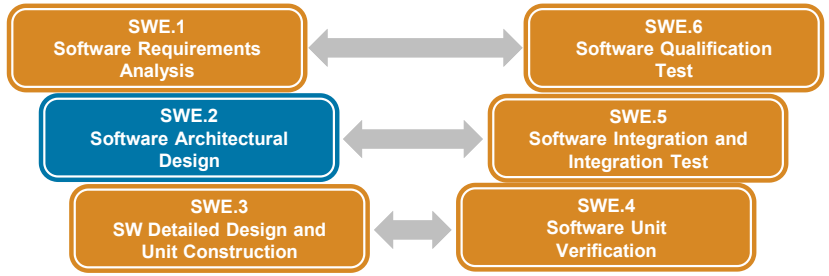


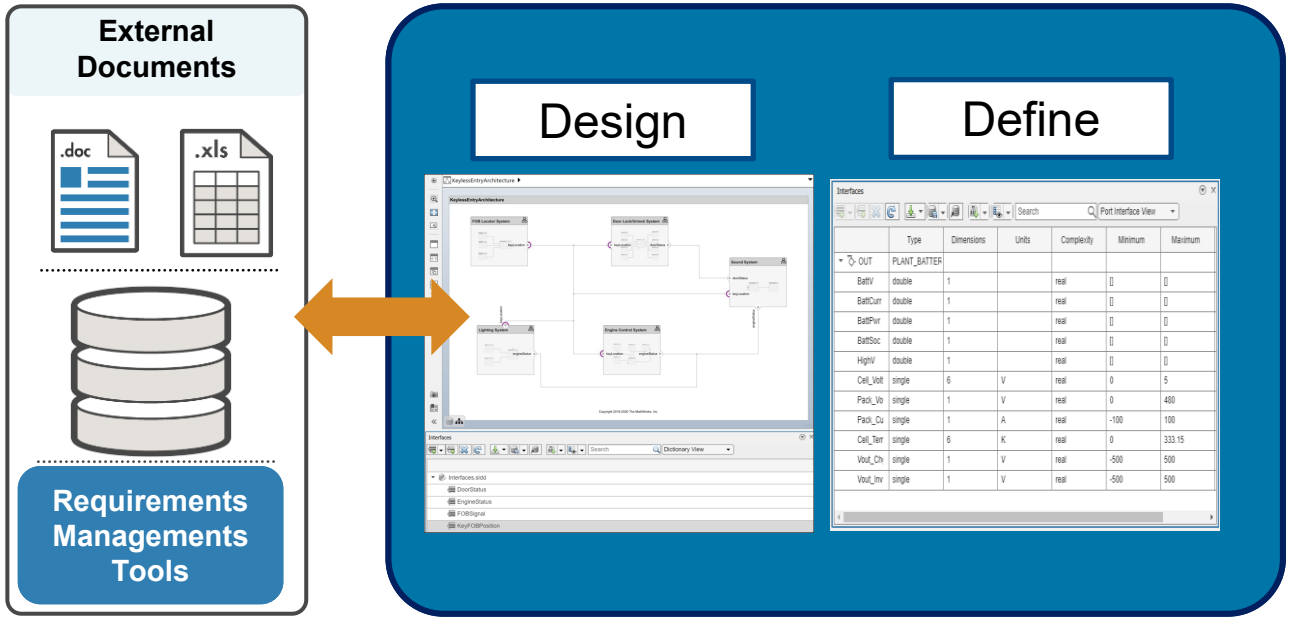


Develop Software Architectural Design

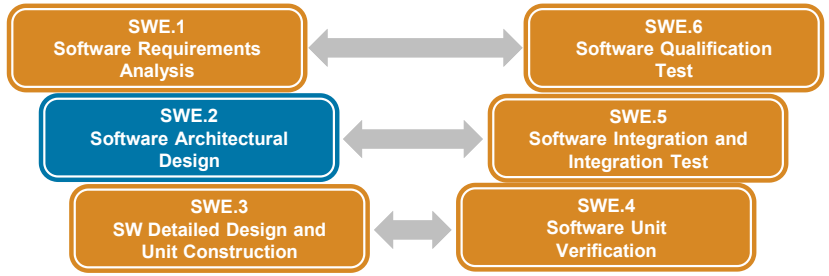


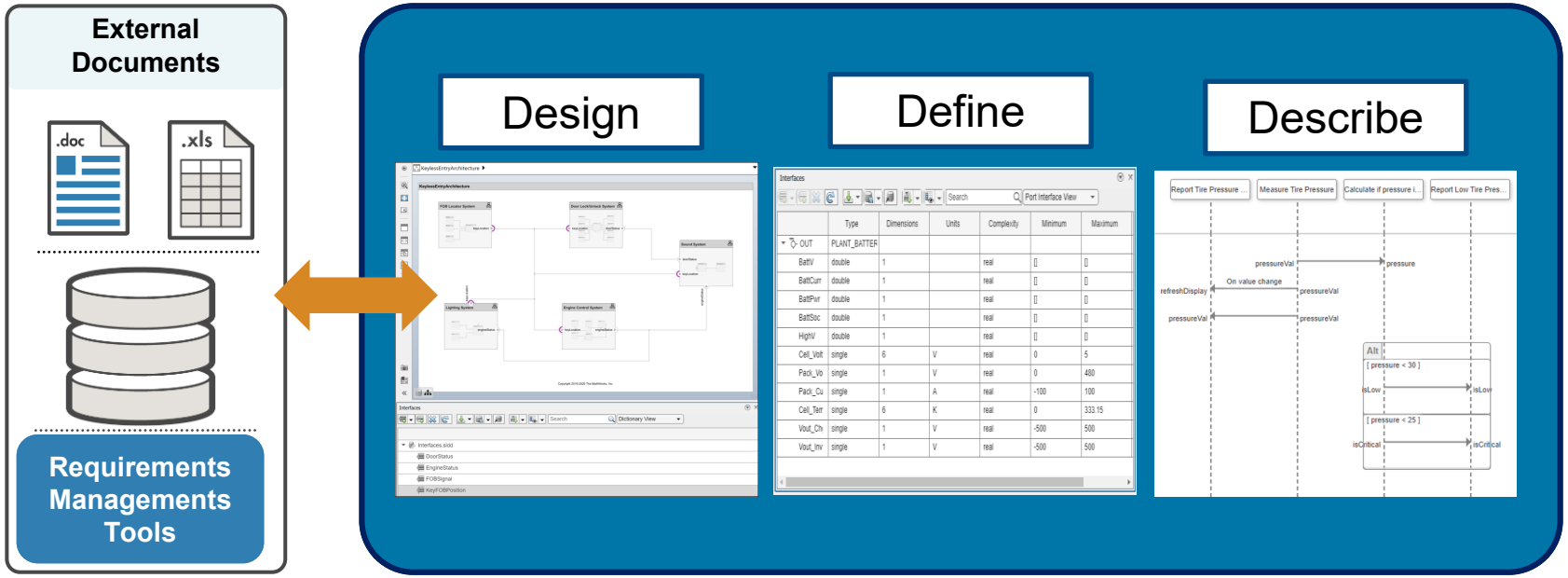
SWE.2 Software Architectural Design



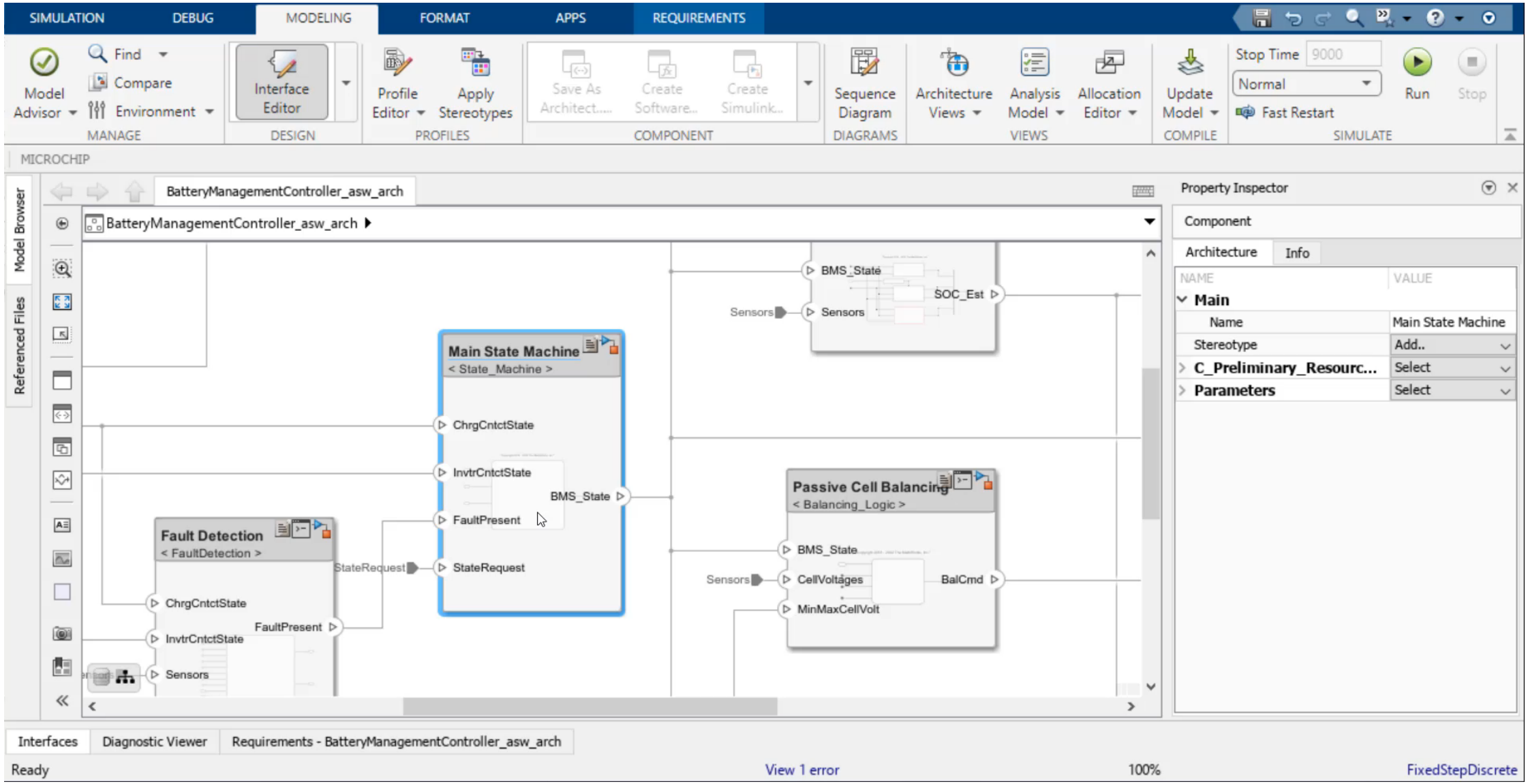


SWE.2 Software Architectural Design

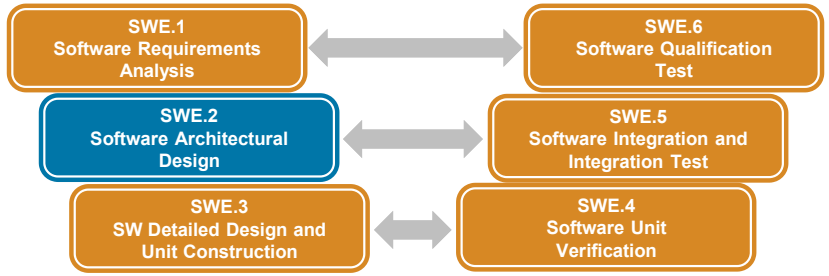


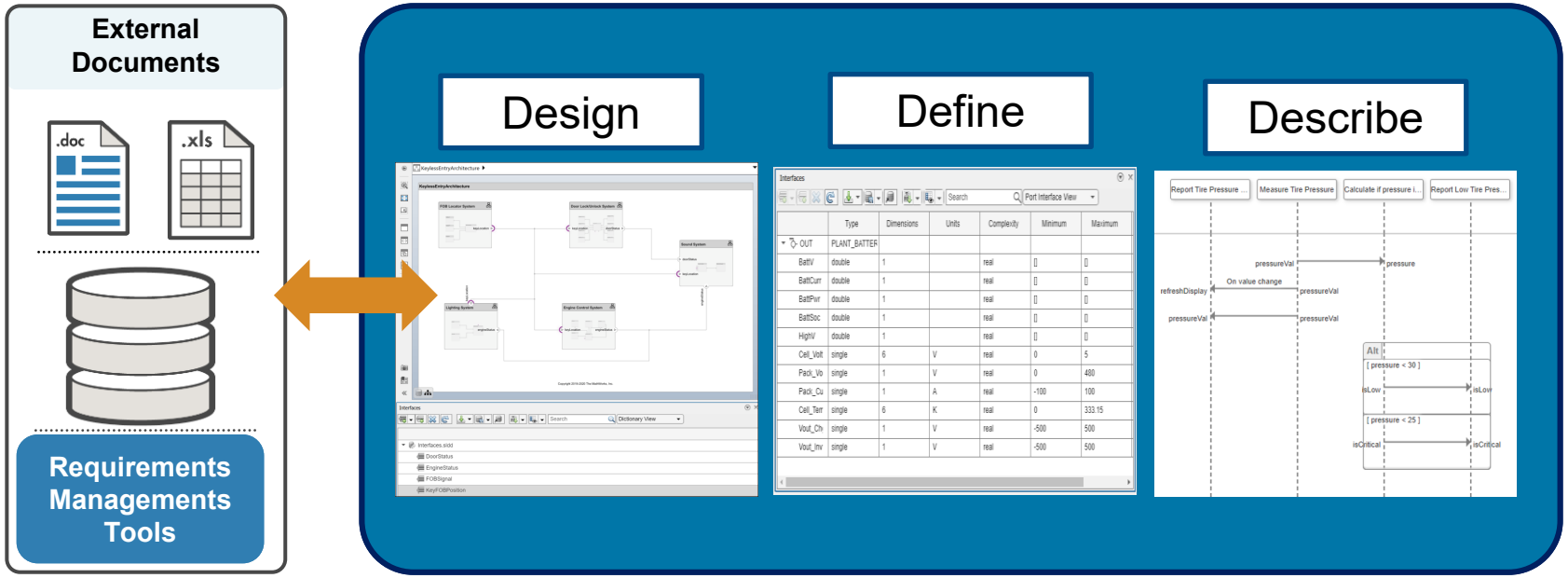


Describe Dynamic Behavior

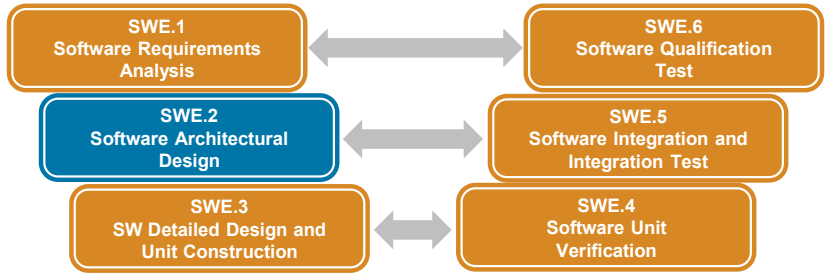


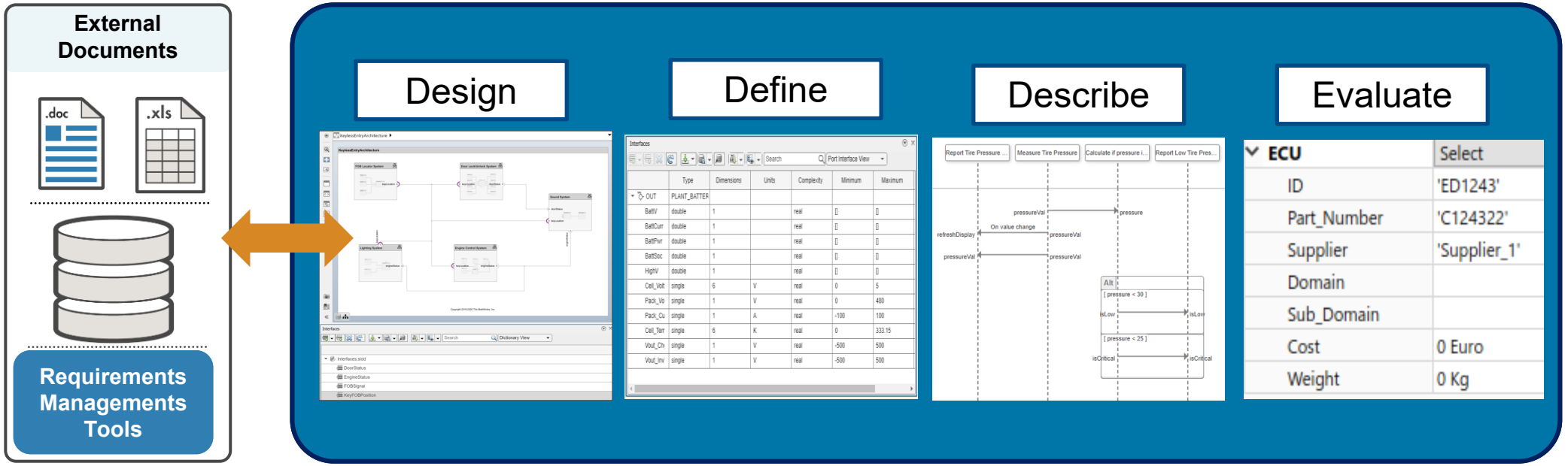
SWE.2 Software Architectural Design



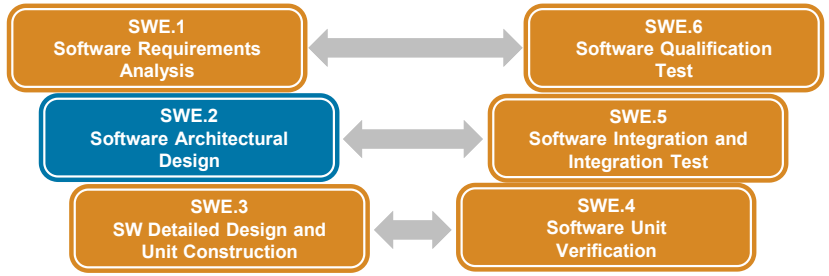


SWE.2 Software Architectural Design





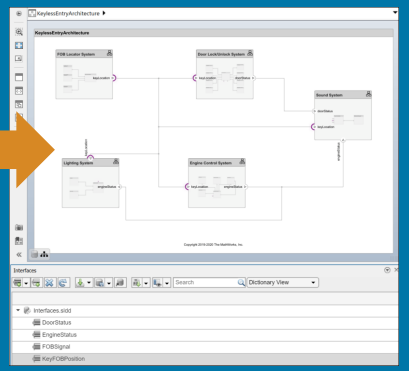
SWE.2 Software Architectural Design



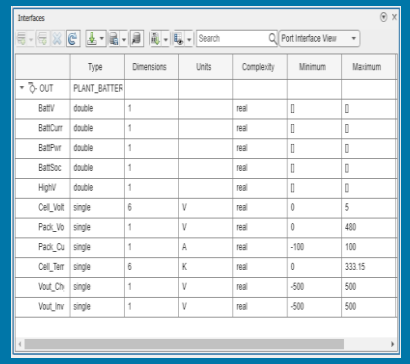
External Documents

Requirements Managements Tools

Design

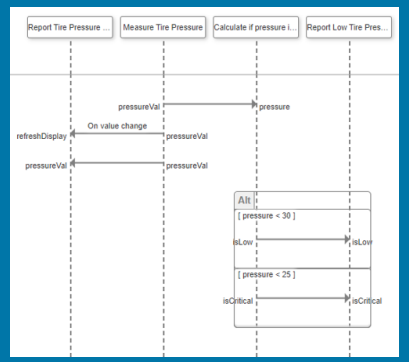


Define



	Type	Dimensions	Units	Complexity	Minimum	Maximum
OUT	PLANT_BATTER					
BattV	double	1		real	[]	[]
BattCurr	double	1		real	[]	[]
BattPwr	double	1		real	[]	[]
BattSoc	double	1		real	[]	[]
HighV	double	1		real	[]	[]
Cell_Volt	single	6	V	real	0	5
Pack_Vo	single	1	V	real	0	480
Pack_Cu	single	1	A	real	-100	100
Cell_Ter	single	6	K	real	0	333.15
Vout_Ch	single	1	V	real	-500	500
Vout_Inv	single	1	V	real	-500	500

Describe



Evaluate

ECU	Select
ID	'ED1243'
Part_Number	'C124322'
Supplier	'Supplier_1'
Domain	
Sub_Domain	
Cost	0 Euro
Weight	0 Kg

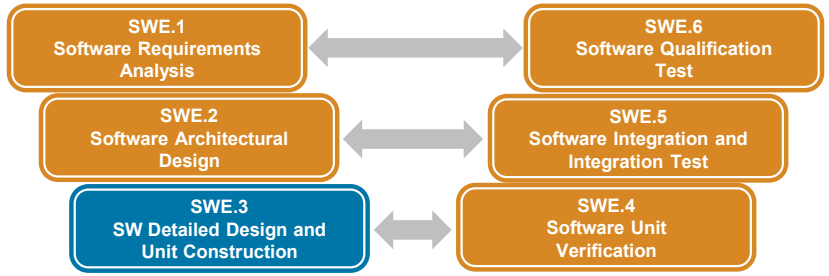
Output Work Products

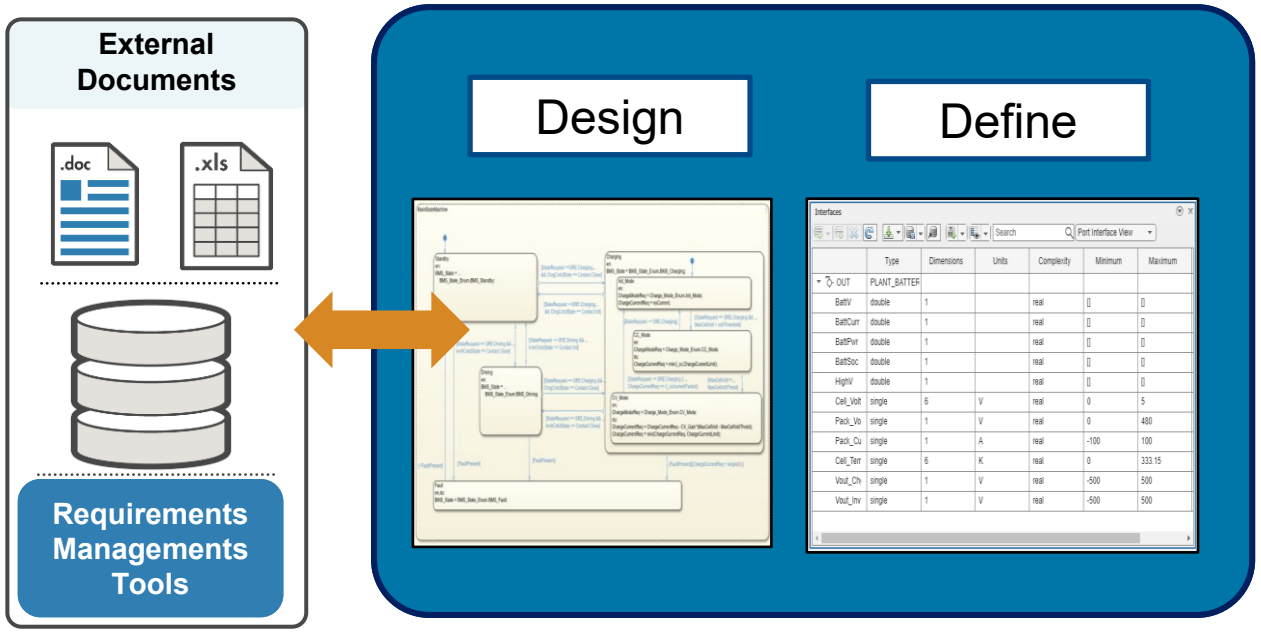
Software architecture

Interfaces

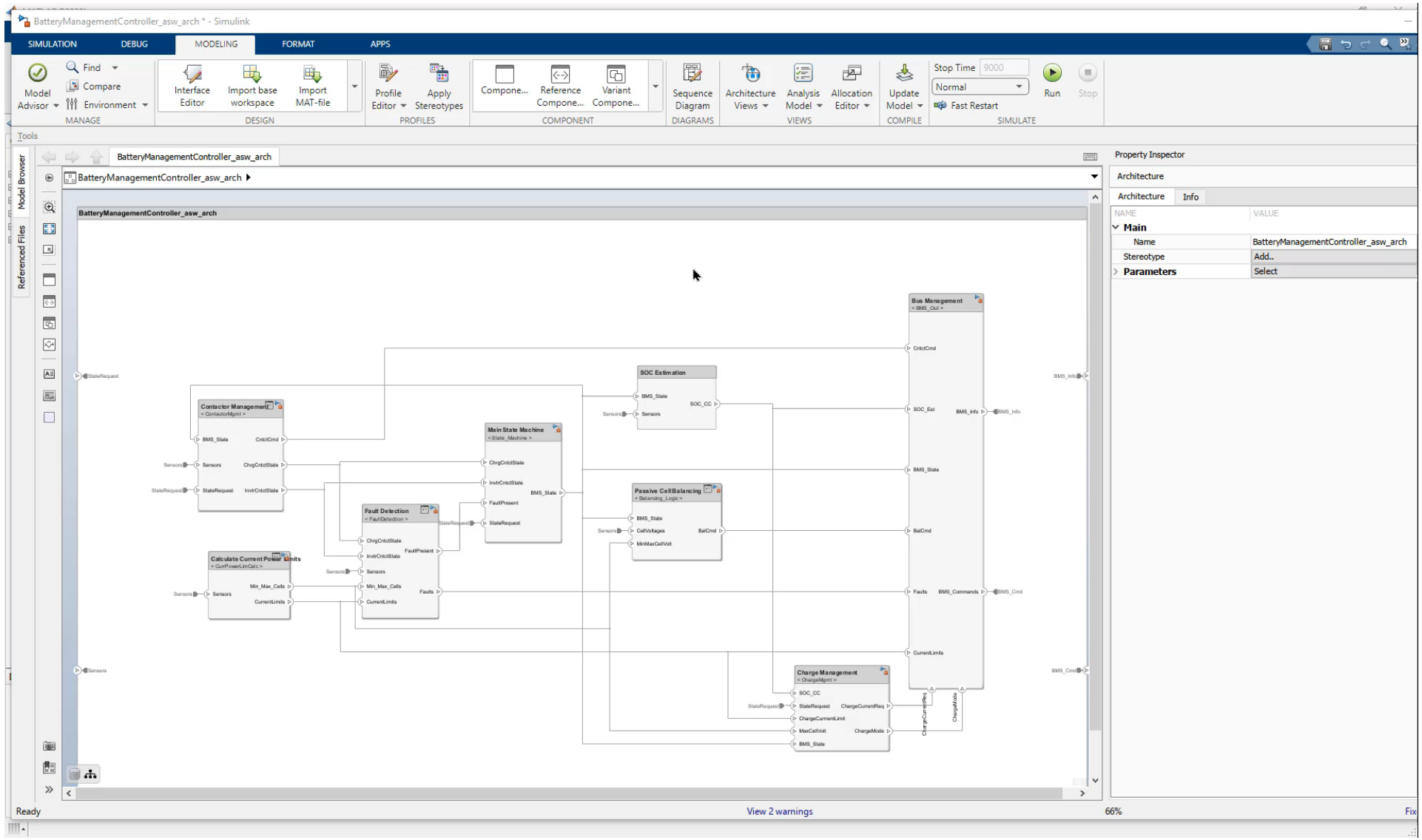
Traceability

SWE.3 SW Detailed Design and Unit Construction

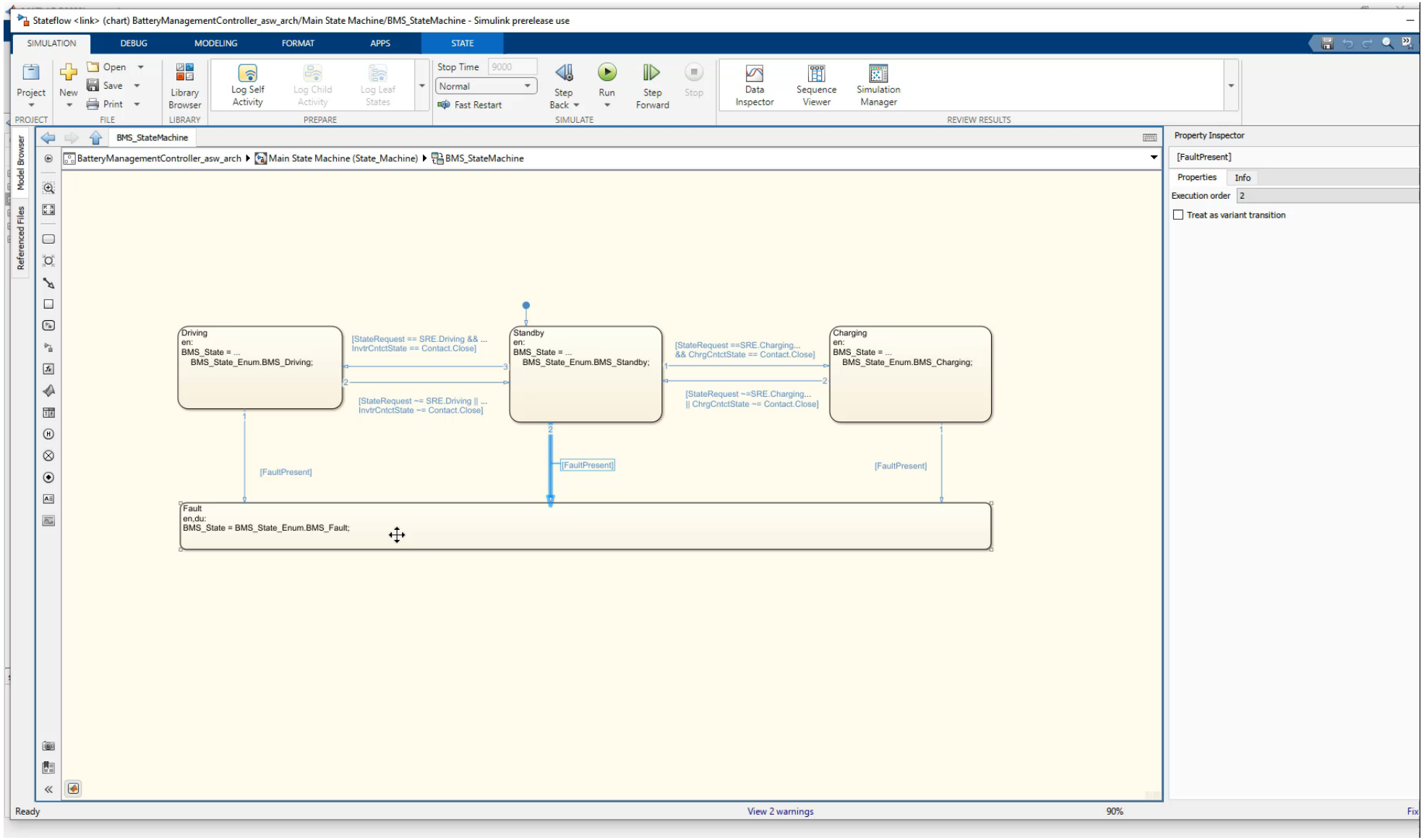




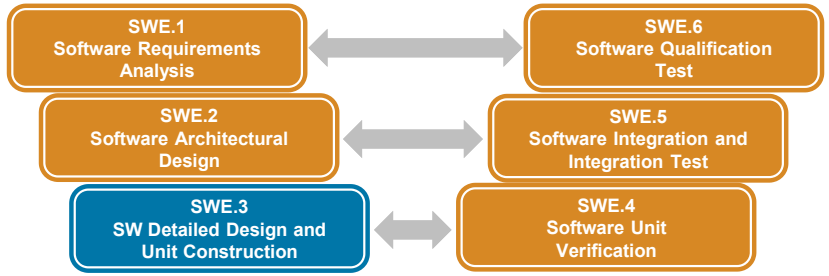
Develop Software Detailed Design

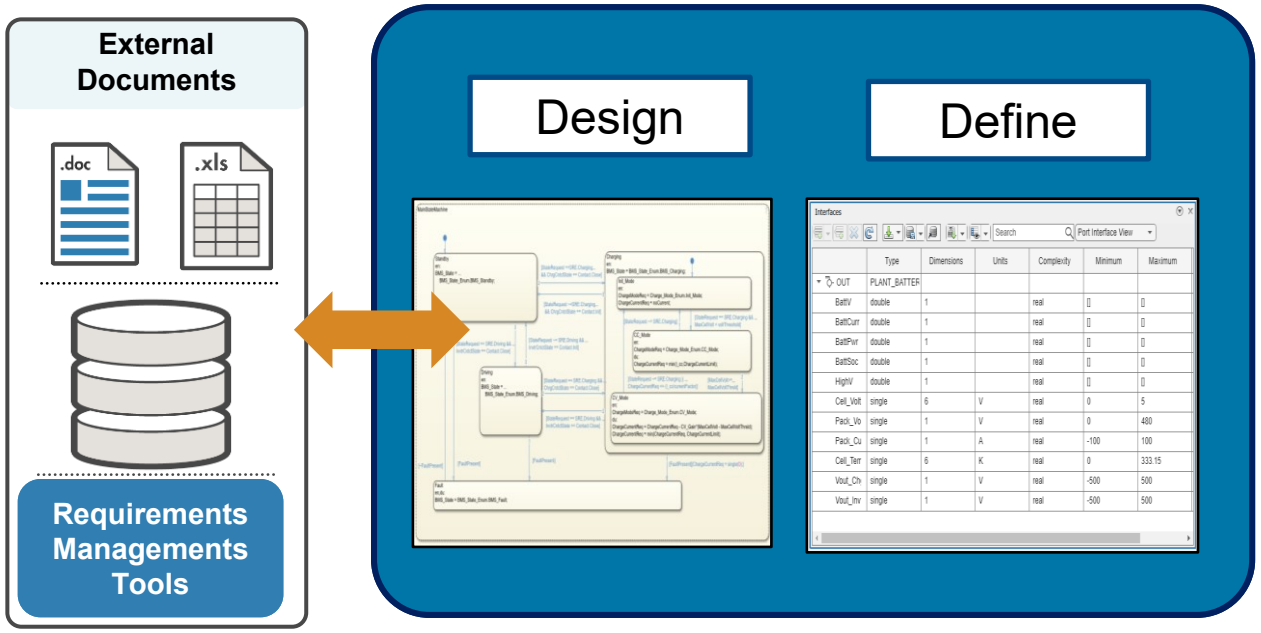


Develop Software Detailed Design

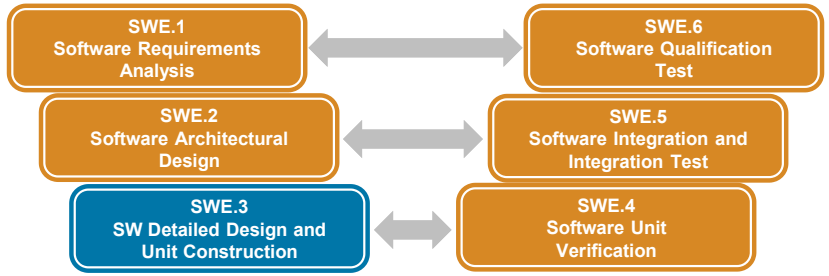


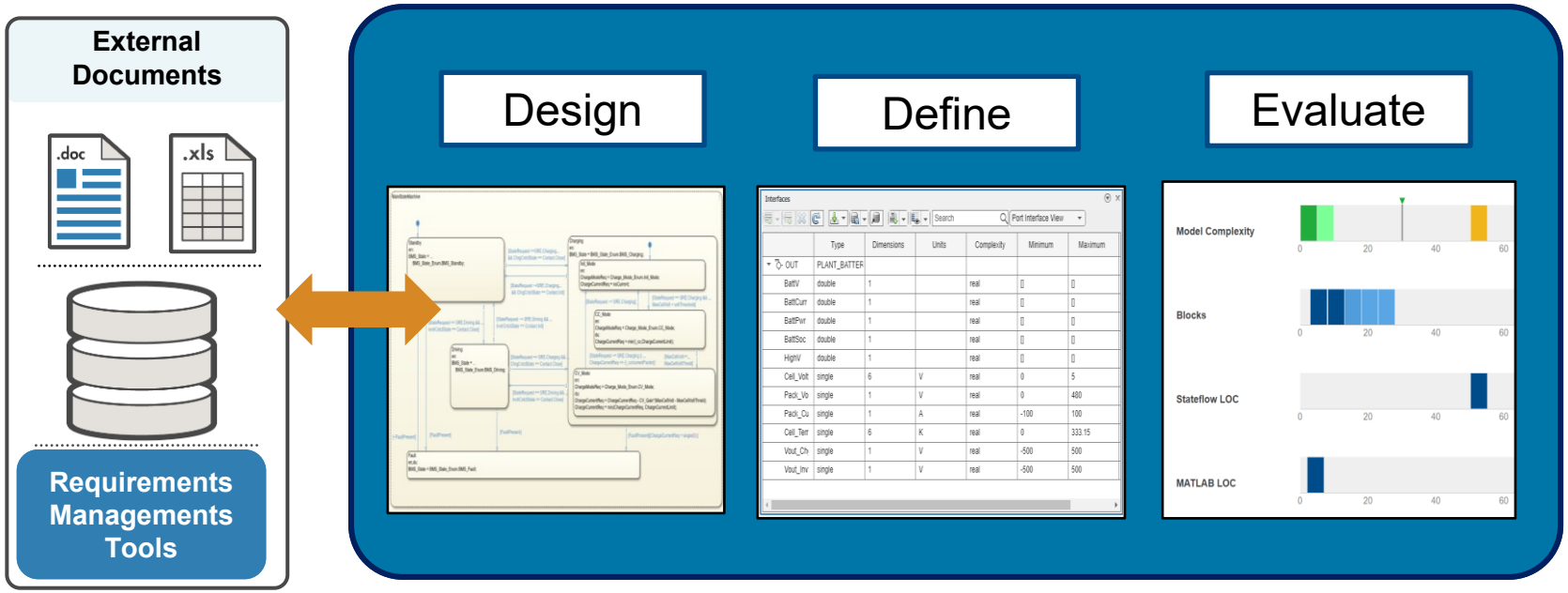
SWE.3 SW Detailed Design and Unit Construction



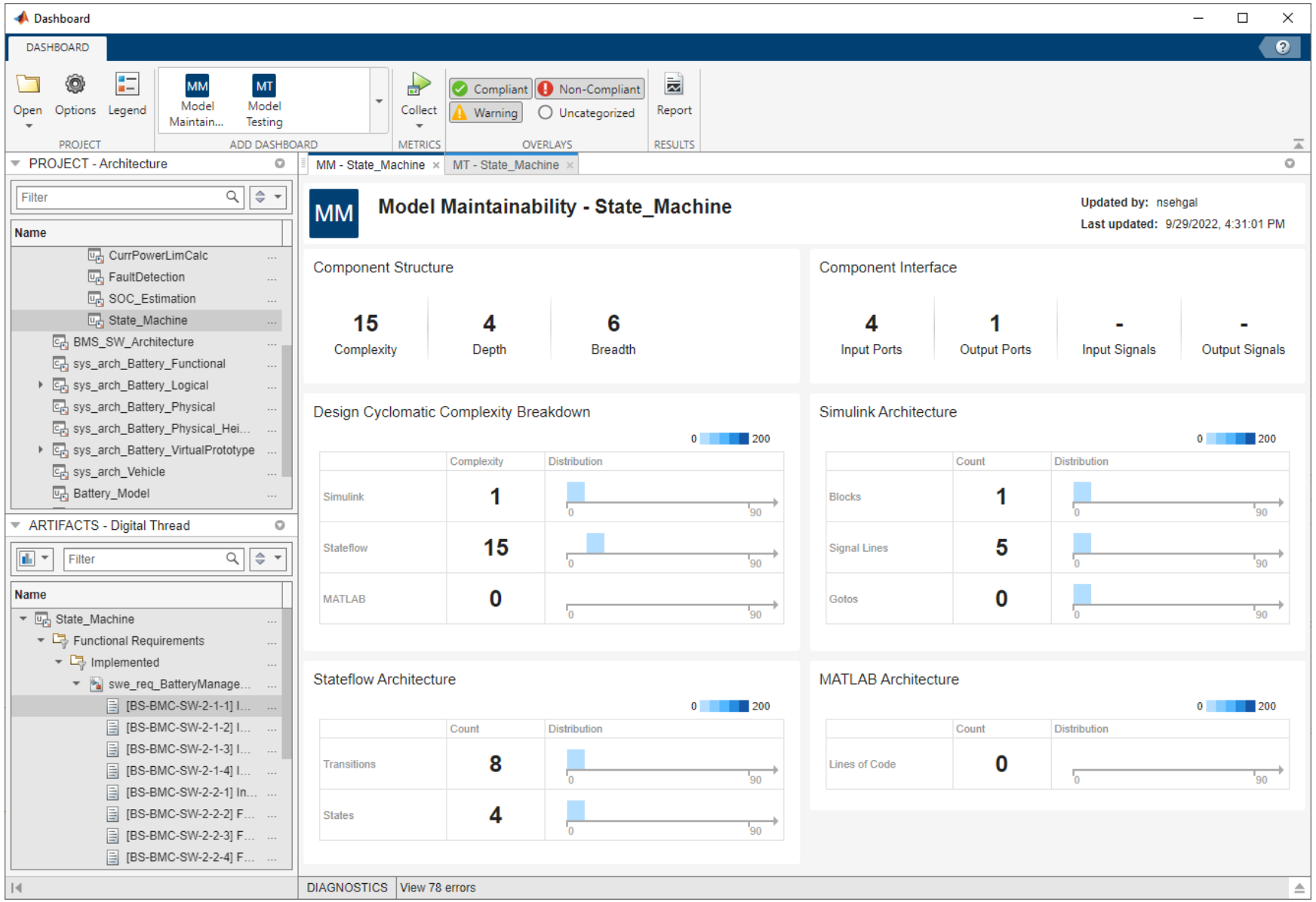


SWE.3 SW Detailed Design and Unit Construction

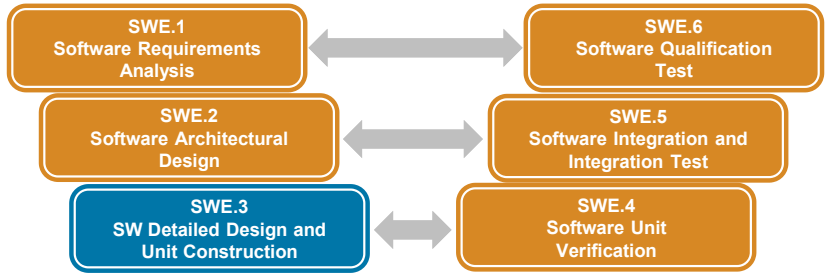


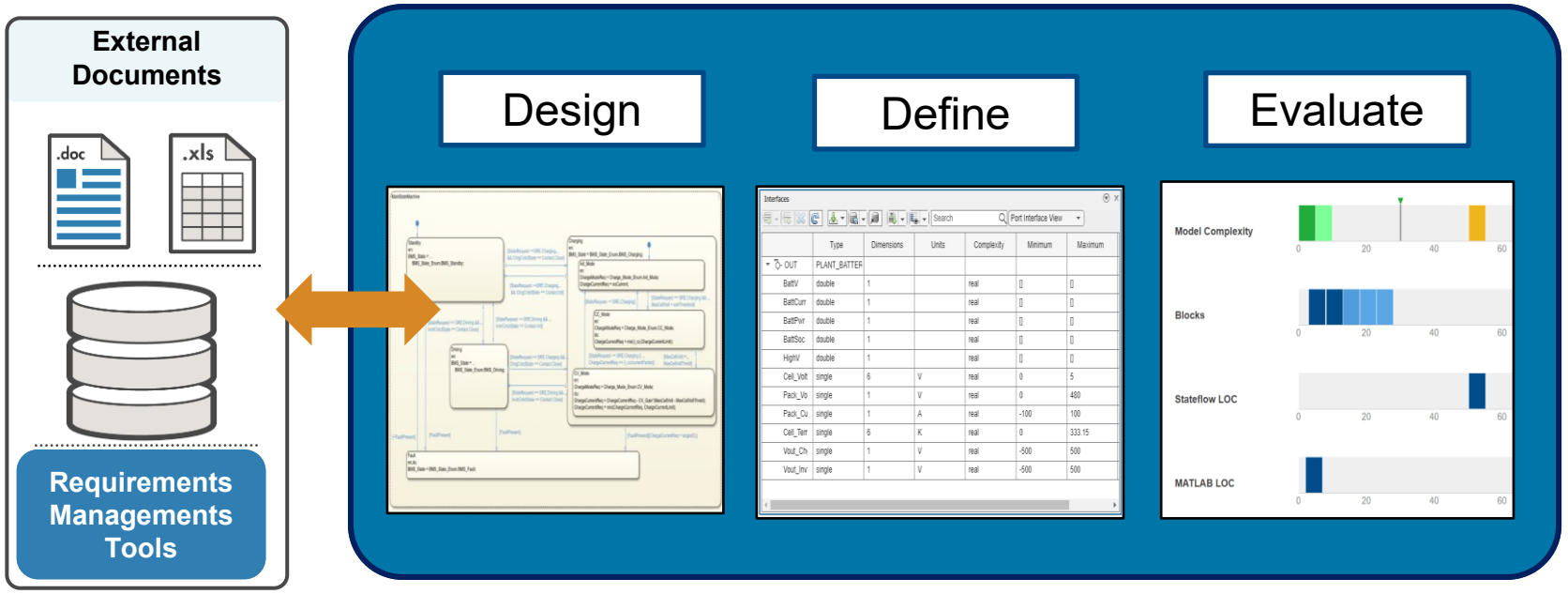


Evaluate Software Detailed Design

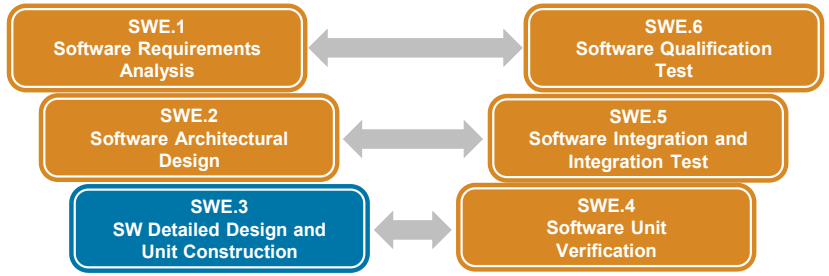


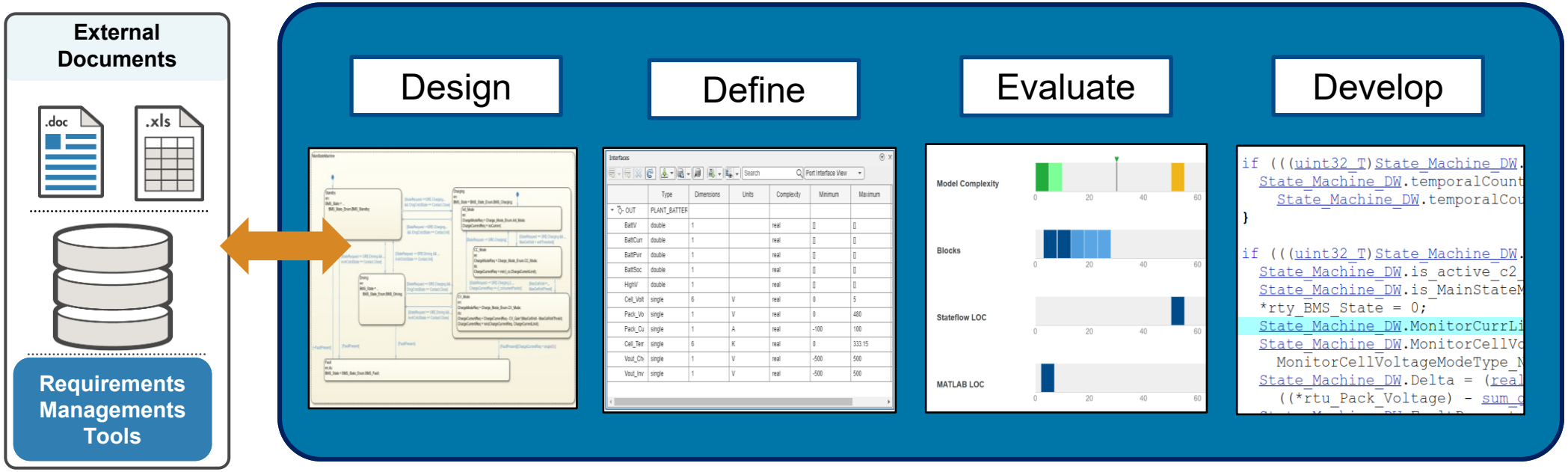
SWE.3 SW Detailed Design and Unit Construction



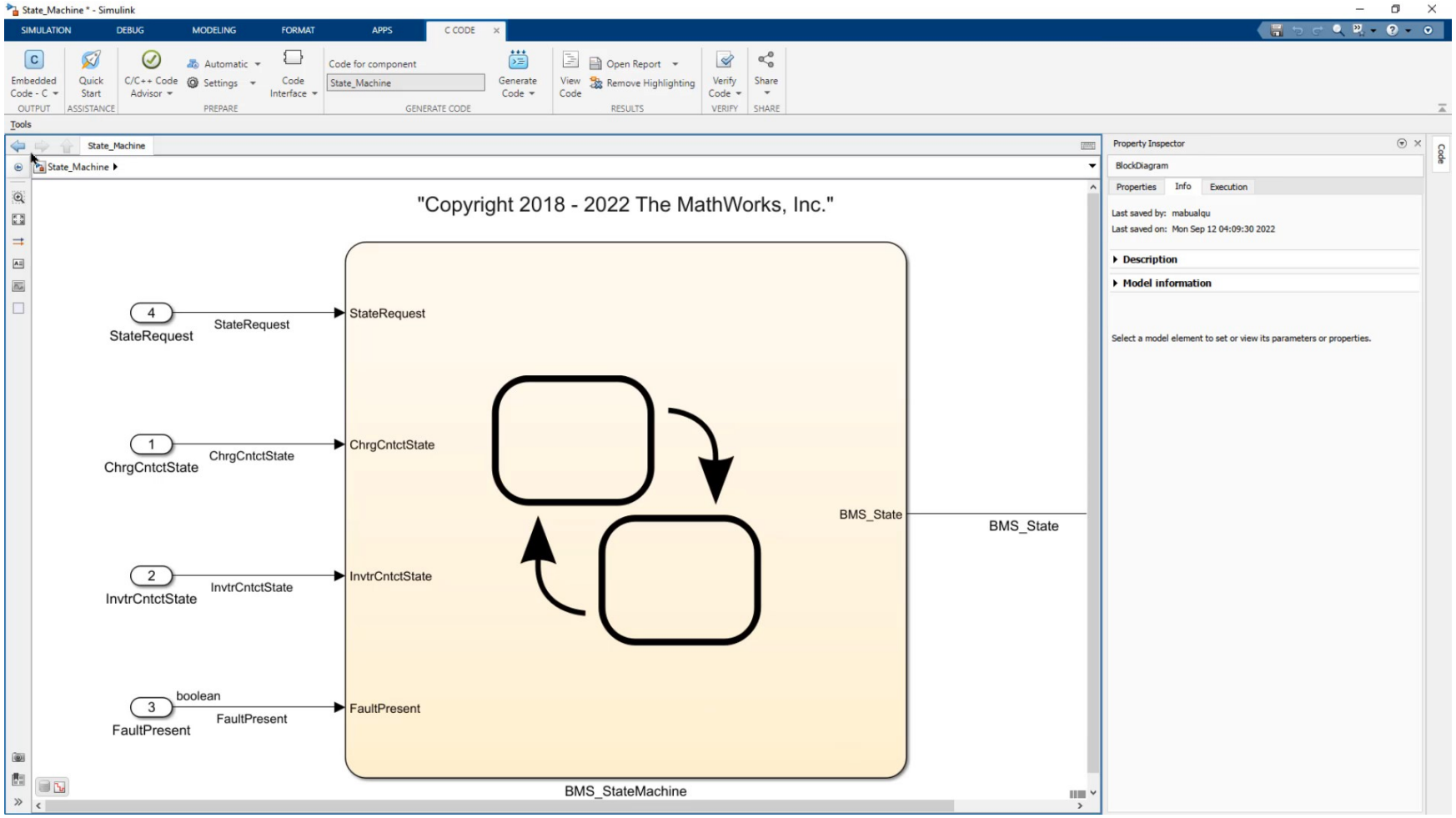


SWE.3 SW Detailed Design and Unit Construction

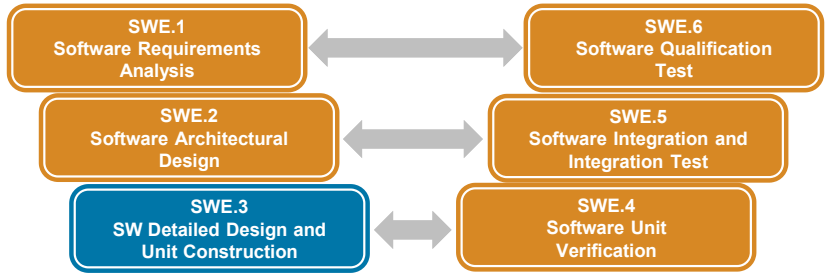




Develop Software Units



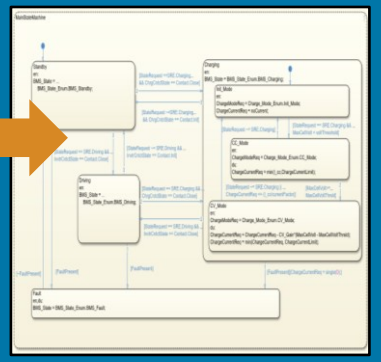
SWE.3 SW Detailed Design and Unit Construction



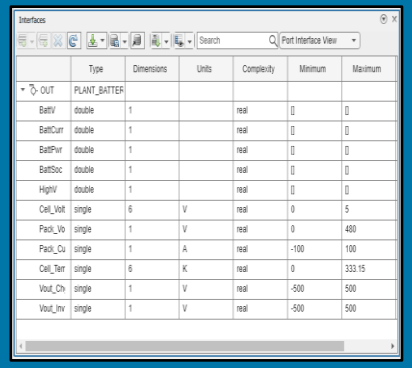
External Documents

Requirements Managements Tools

Design



Define



Evaluate

Develop

```
if (((uint32_T)State_Machine_DW.
  State_Machine_DW.temporalCount
    State_Machine_DW.temporalCou
}

if (((uint32_T)State_Machine_DW.
  State_Machine_DW.is_active_c2_
  State_Machine_DW.is_MainStateM
  *rty_BMS_State = 0;
  State_Machine_DW.MonitorCurrLi
  State_Machine_DW.MonitorCellVo
    MonitorCellVoltageModeType_N
  State_Machine_DW.Delta = (real
    ((*rtu_Pack_Voltage) - sum_c
```

Output Work Products

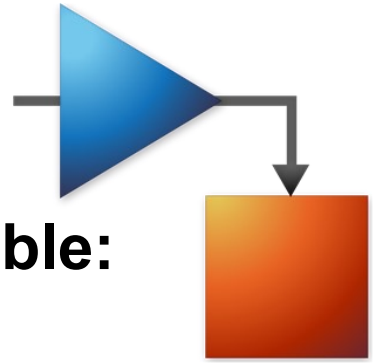
Detailed Design Dynamic Behavior Software Unit

Interfaces

Traceability

Concluding Remarks

Key Takeaways



Model-Based Design and Model-Based Systems Engineering enable:

1. **Fast development and realization** of **system and software** architecture and design
2. **Early testing** to detect errors in designs and their realization
3. **Fast and efficient iterations**

→ Develop **high quality products** following an **efficient Automotive SPICE® compliant process**

IEC Certification Kit – MBD Process for A-SPICE®