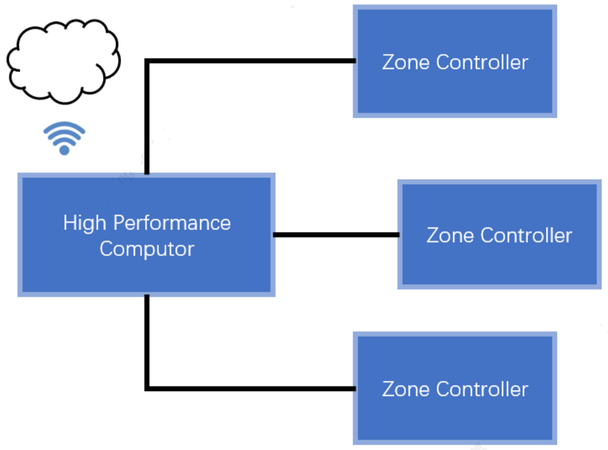
ZEEKR

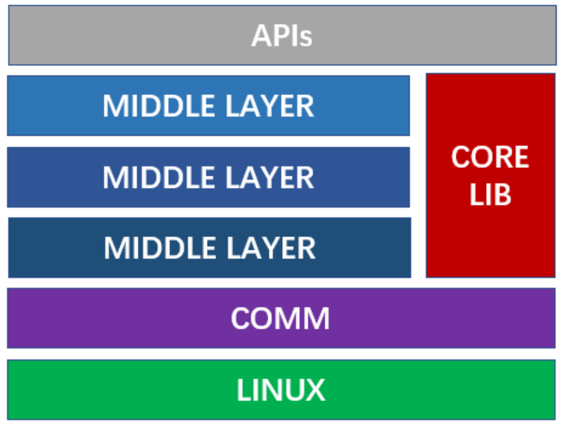
Use Model-Based Design to Develop SOA Application Running on In-vehicle OS

Yiming Luo, ZEEKR TECHNOLOGY LIMITED

MATLAB EXPO

Background





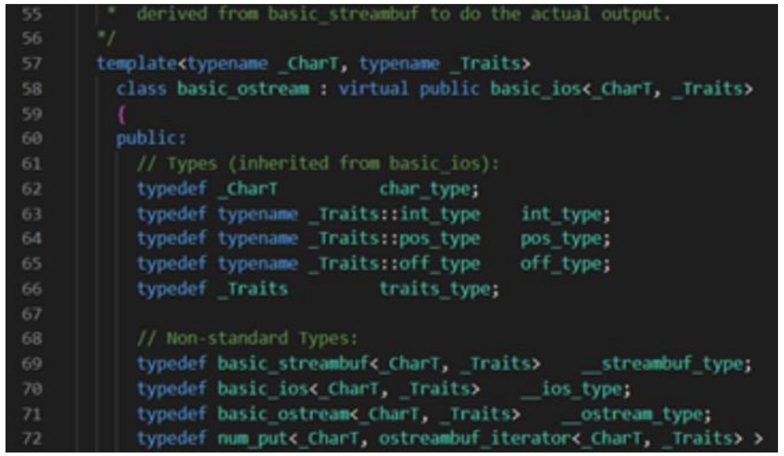
- SOA Trend
  - Software defines a vehicle
  - Growing function demand

- Next EEA
  - Centralized domain architecture
  - Providing hardware to support the software boom

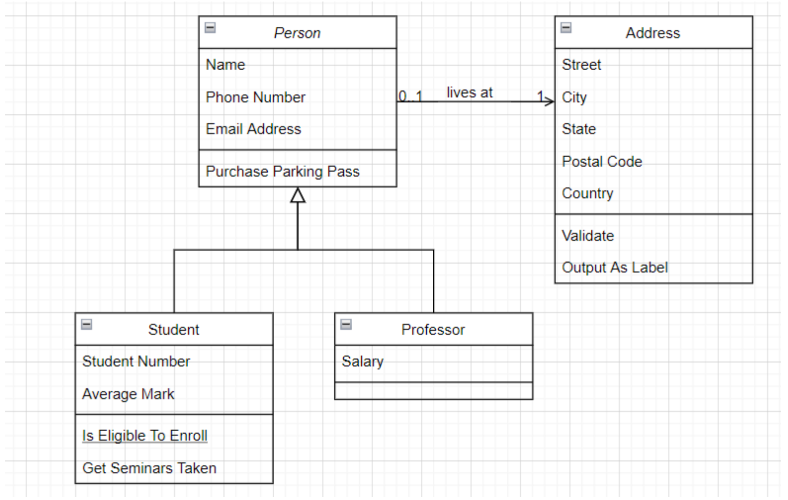
- In-vehicle OS
  - Distributed operating system developed by OEM
  - Support SOA

Challenges

- Coding Difficuty
  - Handwritten C++ language puts forward high requirements not only on the programmer's ability but also on the accompanying tool chain



```
  *  derived from basic_streambuf to do the actual output.
  */
template<typename _CharT, typename _Traits>
  class basic_ostream : virtual public basic_ios<_CharT, _Traits>
  {
  public:
    // Types (inherited from basic_ios):
    typedef _CharT                    char_type;
    typedef typename _Traits::int_type     int_type;
    typedef typename _Traits::pos_type     pos_type;
    typedef typename _Traits::off_type     off_type;
    typedef _Traits                   traits_type;

    // Non-standard Types:
    typedef basic_streambuf<_CharT, _Traits>     __streambuf_type;
    typedef basic_ios<_CharT, _Traits>     __ios_type;
    typedef basic_ostream<_CharT, _Traits>     __ostream_type;
    typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> >
```



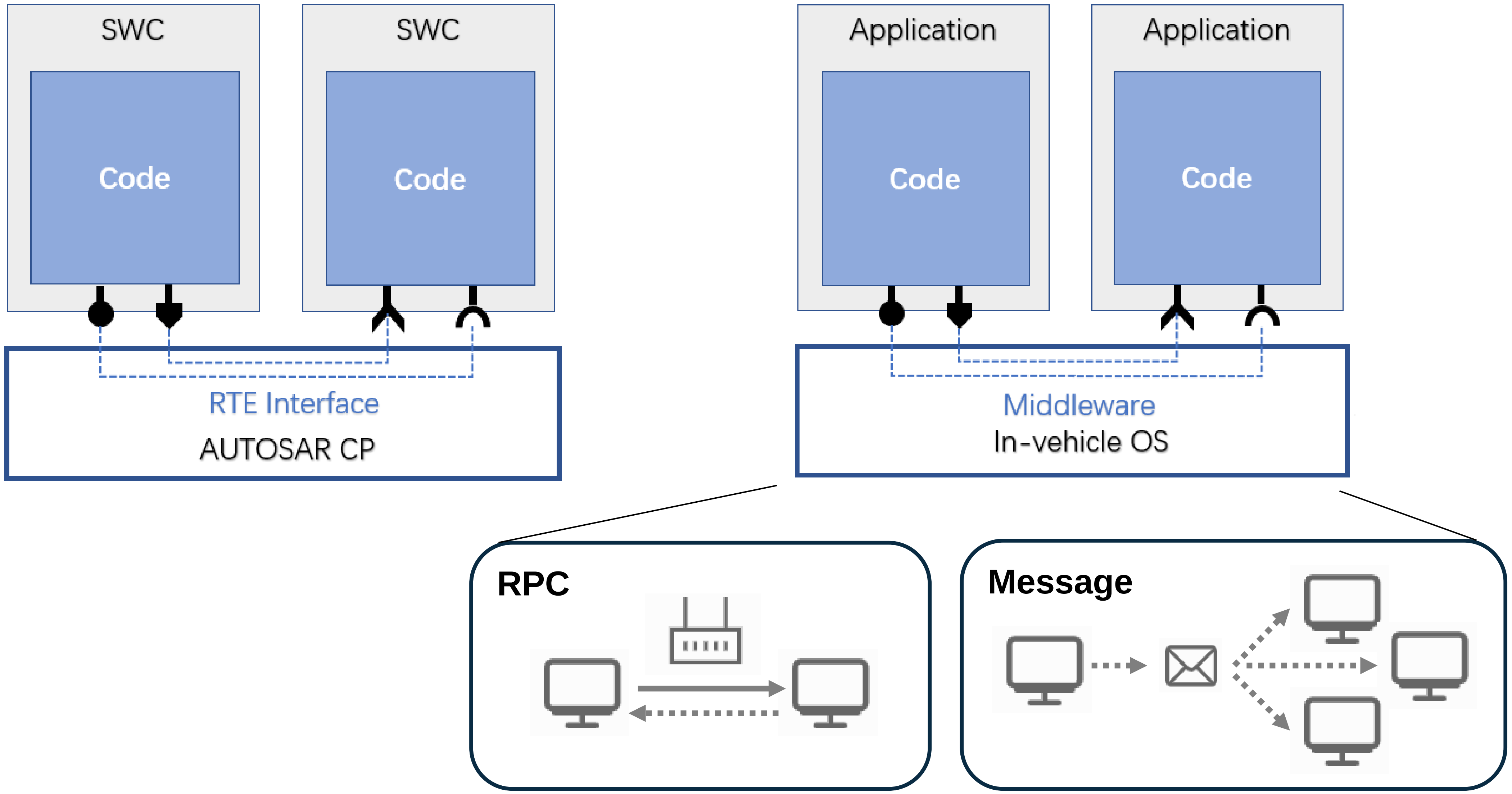
- System Complexity
  - Many software architecture design and management tools on the market are not compatible with non-standard In-vehile OS

Solution Outline

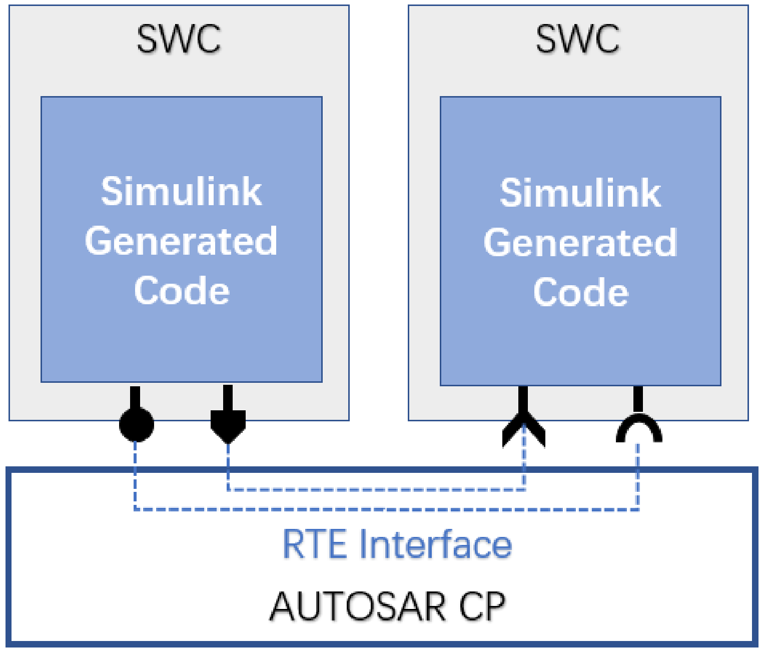
- How to Model the Software Behavior
  - SOA Behavior Modeling
  - SIMULINK New Features
  - Wrapper Code Generator
- How to Maintain Complex Software Clusters
  - Software Architecture Engineering
  - System Composer(Deeply Customized)

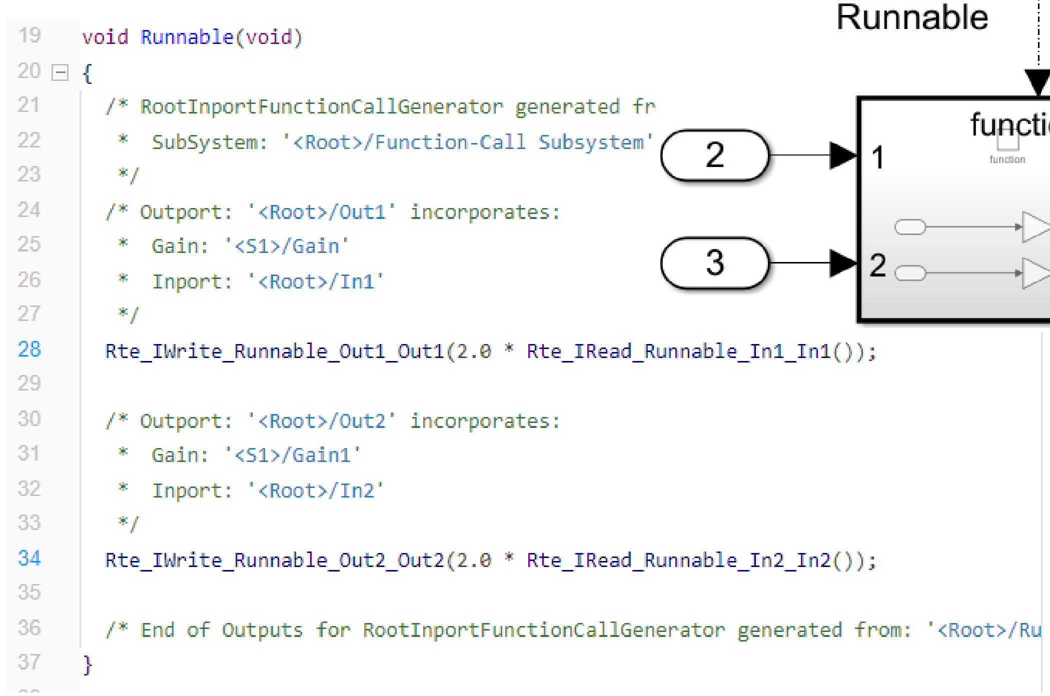
PART I
How to Model the Software Behavior

Comparison Between Different Modeling Environments

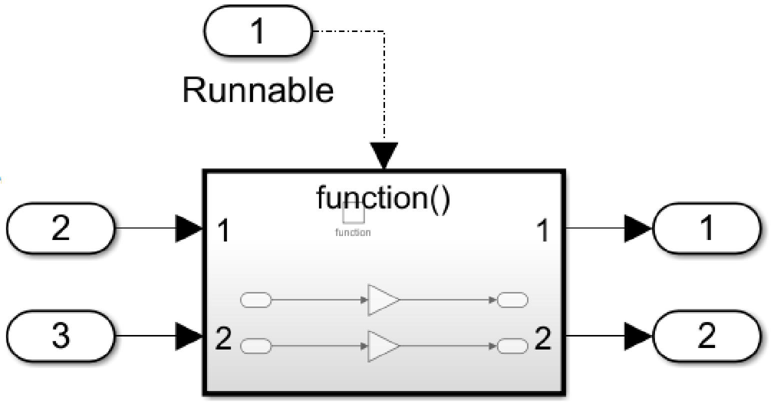


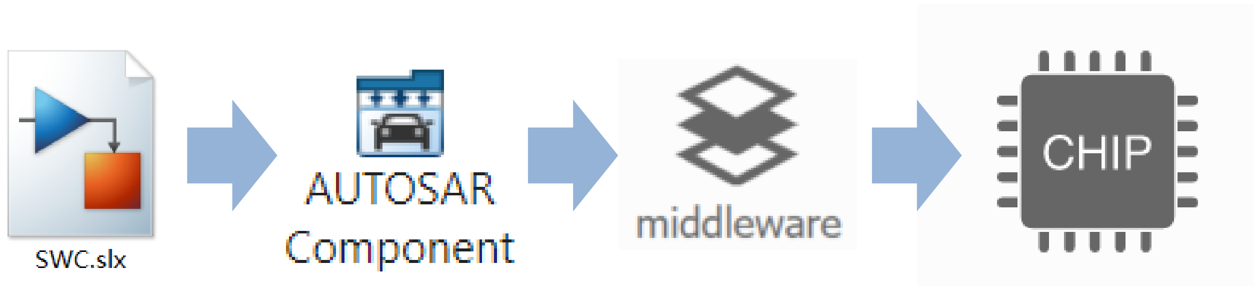
Typical MBD Process on AUTOSAR Workflow



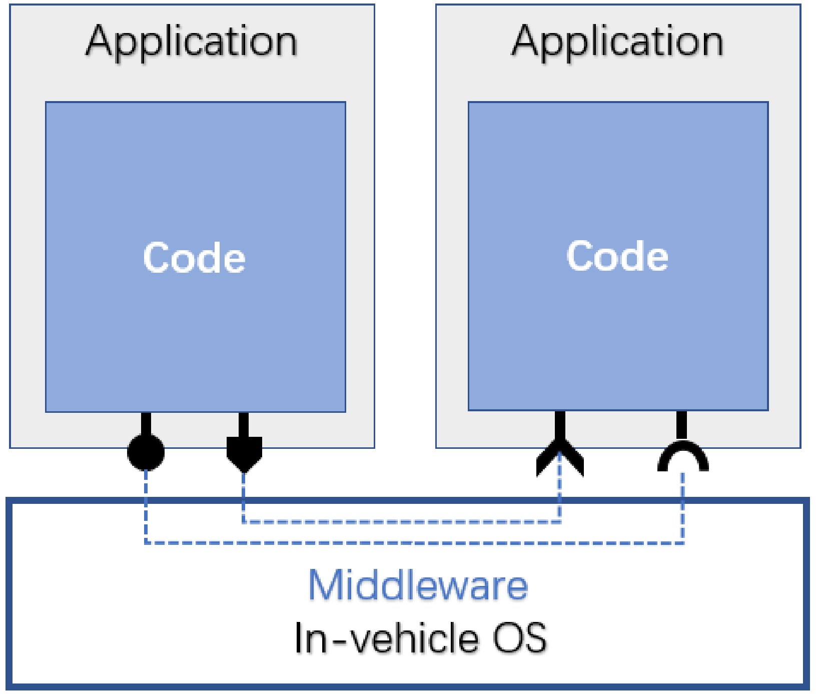


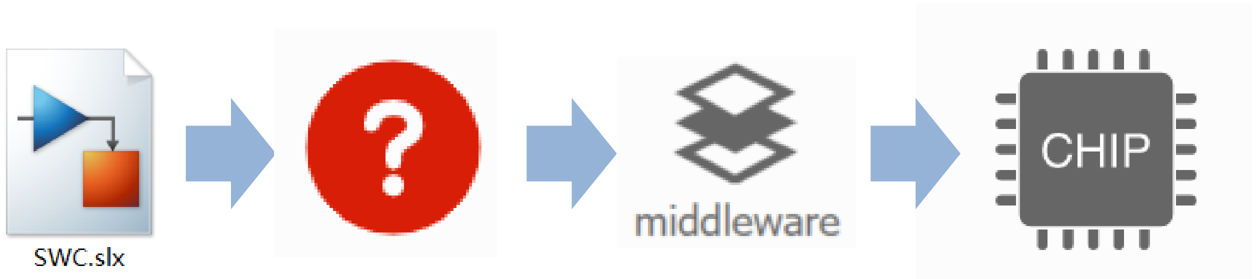
```
void Runnable(void)
{
  /* RootInportFunctionCallGenerator generated fr
   * SubSystem: '<Root>/Function-Call Subsystem'
   */
  /* Outport: '<Root>/Out1' incorporates:
   * Gain: '<S1>/Gain'
   * Inport: '<Root>/In1'
   */
  Rte_IWrite_Runnable_Out1_Out1(2.0 * Rte_IRead_Runnable_In1_In1());

  /* Outport: '<Root>/Out2' incorporates:
   * Gain: '<S1>/Gain1'
   * Inport: '<Root>/In2'
   */
  Rte_IWrite_Runnable_Out2_Out2(2.0 * Rte_IRead_Runnable_In2_In2());

  /* End of Outputs for RootInportFunctionCallGenerator generated from: '<Root>/Ru
}
```



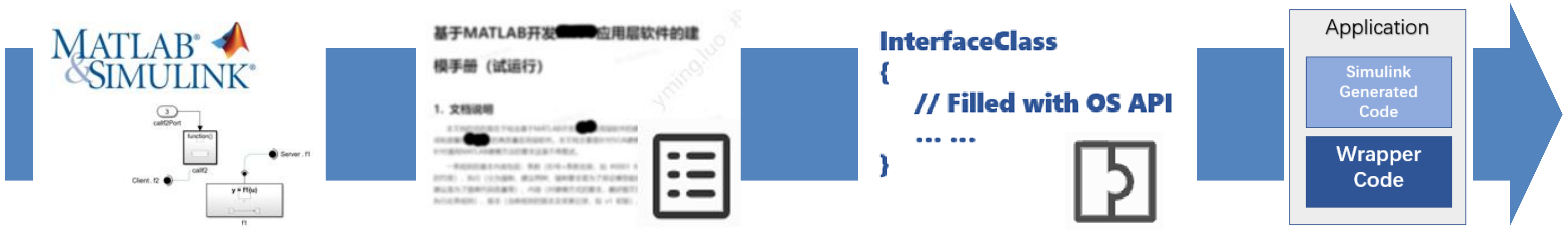


How to Model on In-vehicle OS





How to Model on In-vehicle OS



基于MATLAB开发[redacted]应用层软件的建模手册（试运行）

1. 文档说明

[illegible]

```
InterfaceClass
{
    // Filled with OS API
    ... ...
}
```

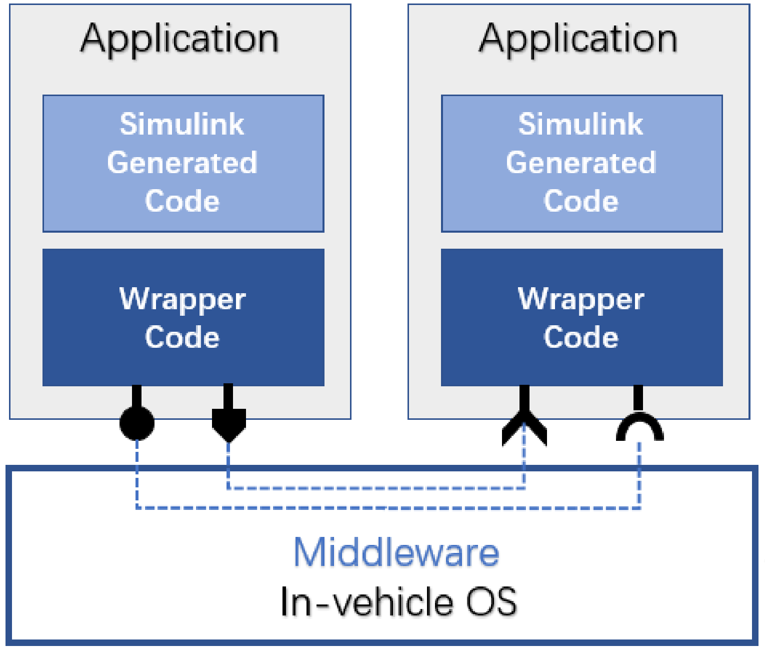
New Features
- Support SOA Behavior

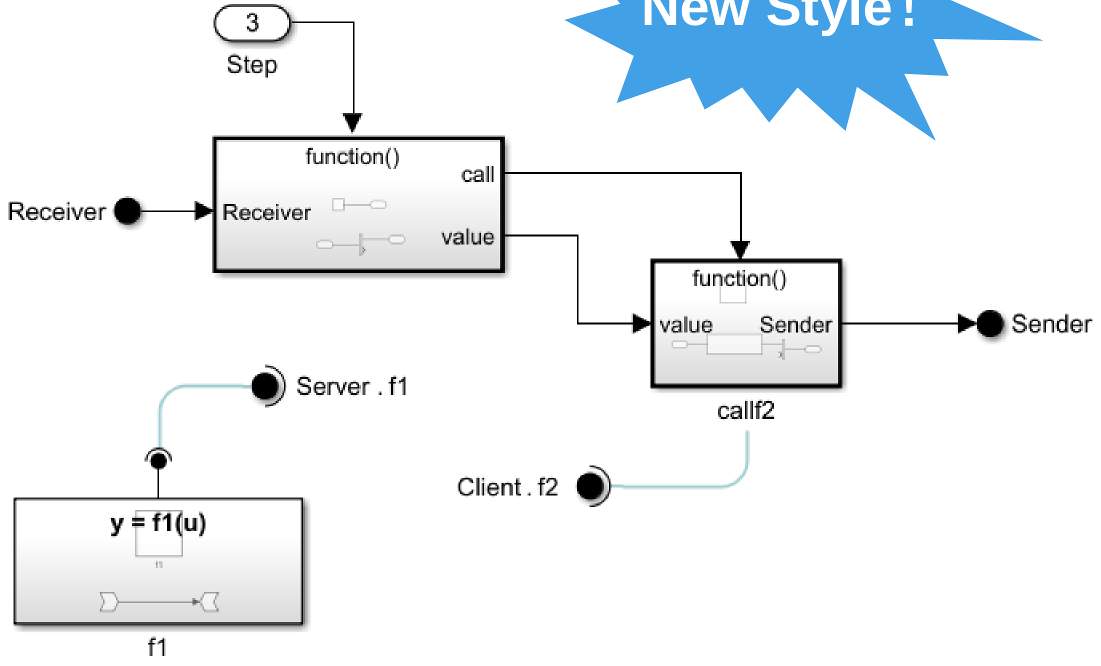
New Modeling Principle
- Specific rules based on practice

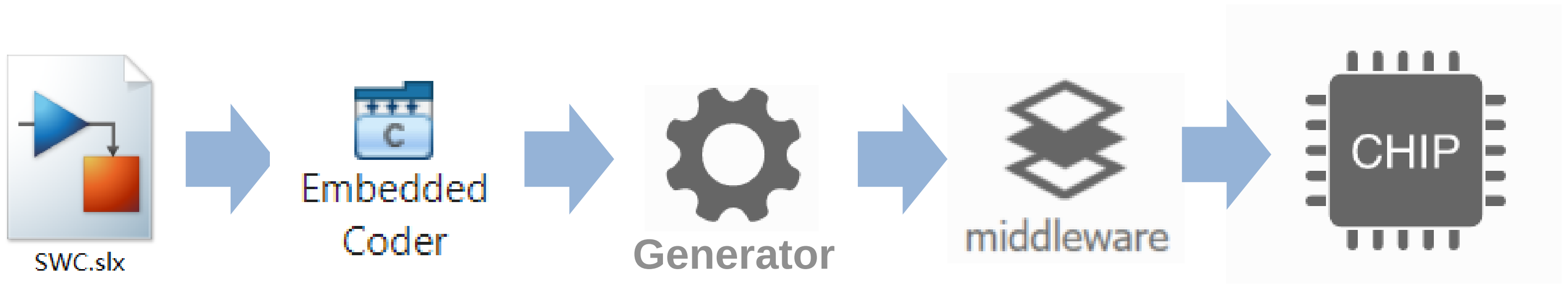
Wrapper Code Generator
- Link Simulink side and OS side

Deployment
- Deployed just like normal application

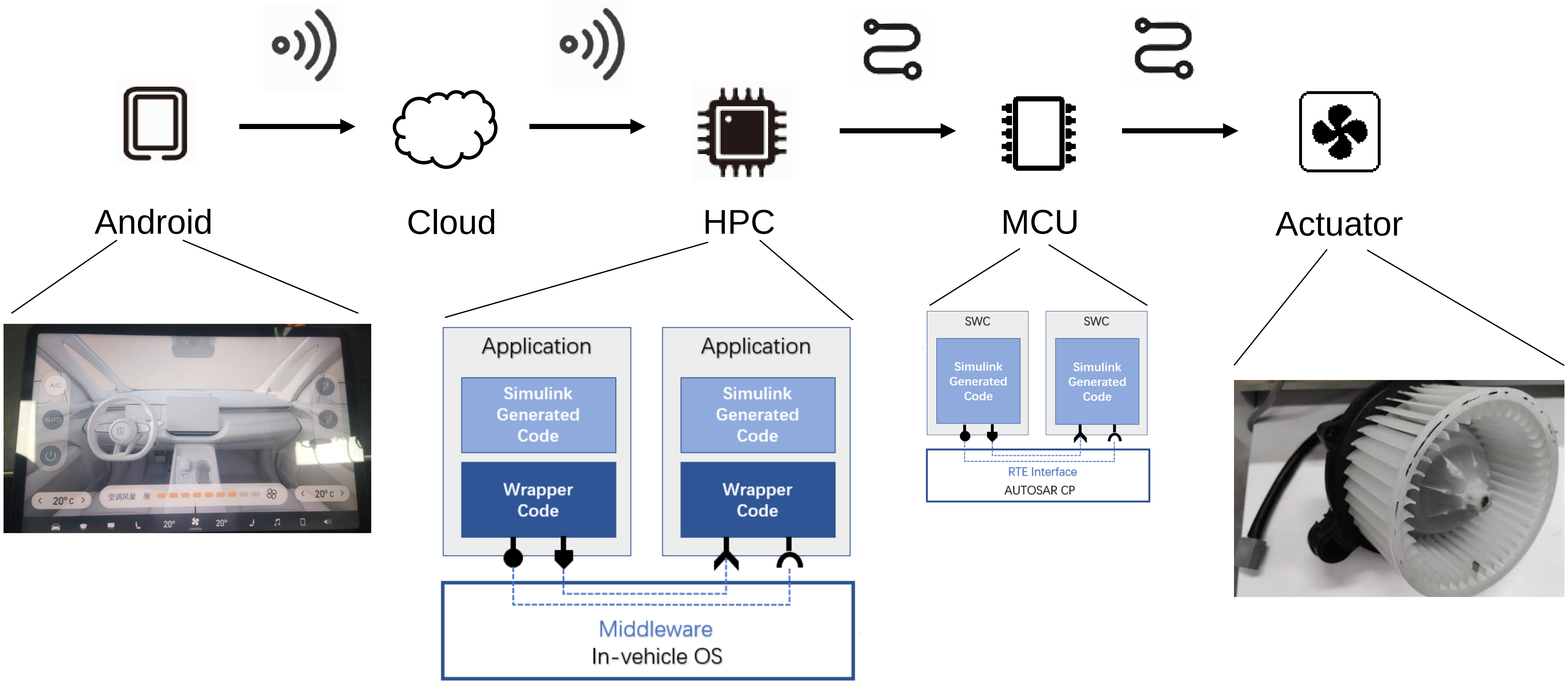
How to Model on In-vehicle OS



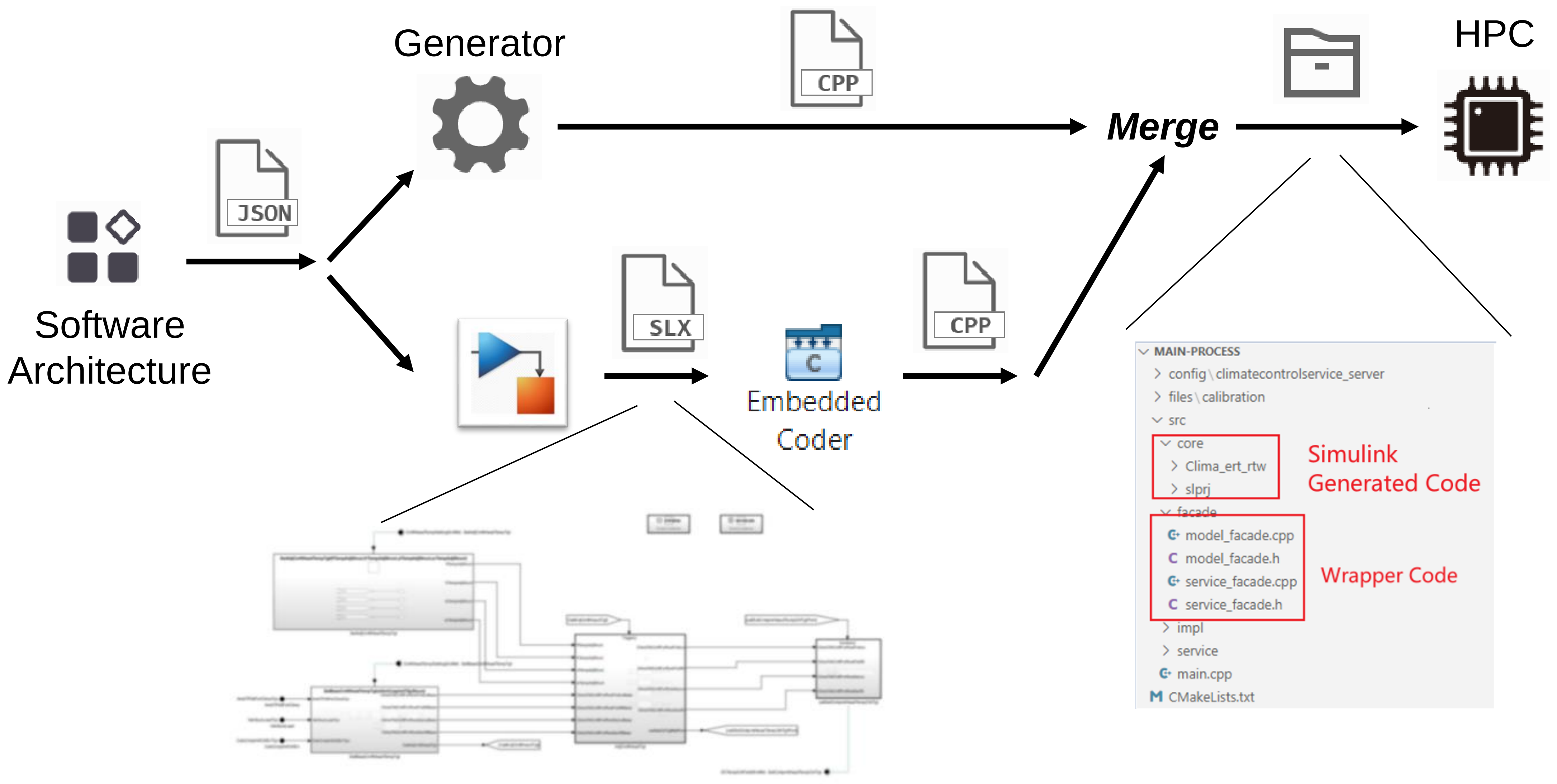


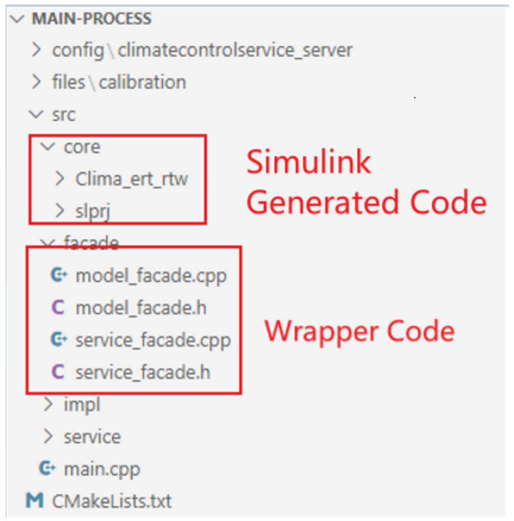


Example in Real Case



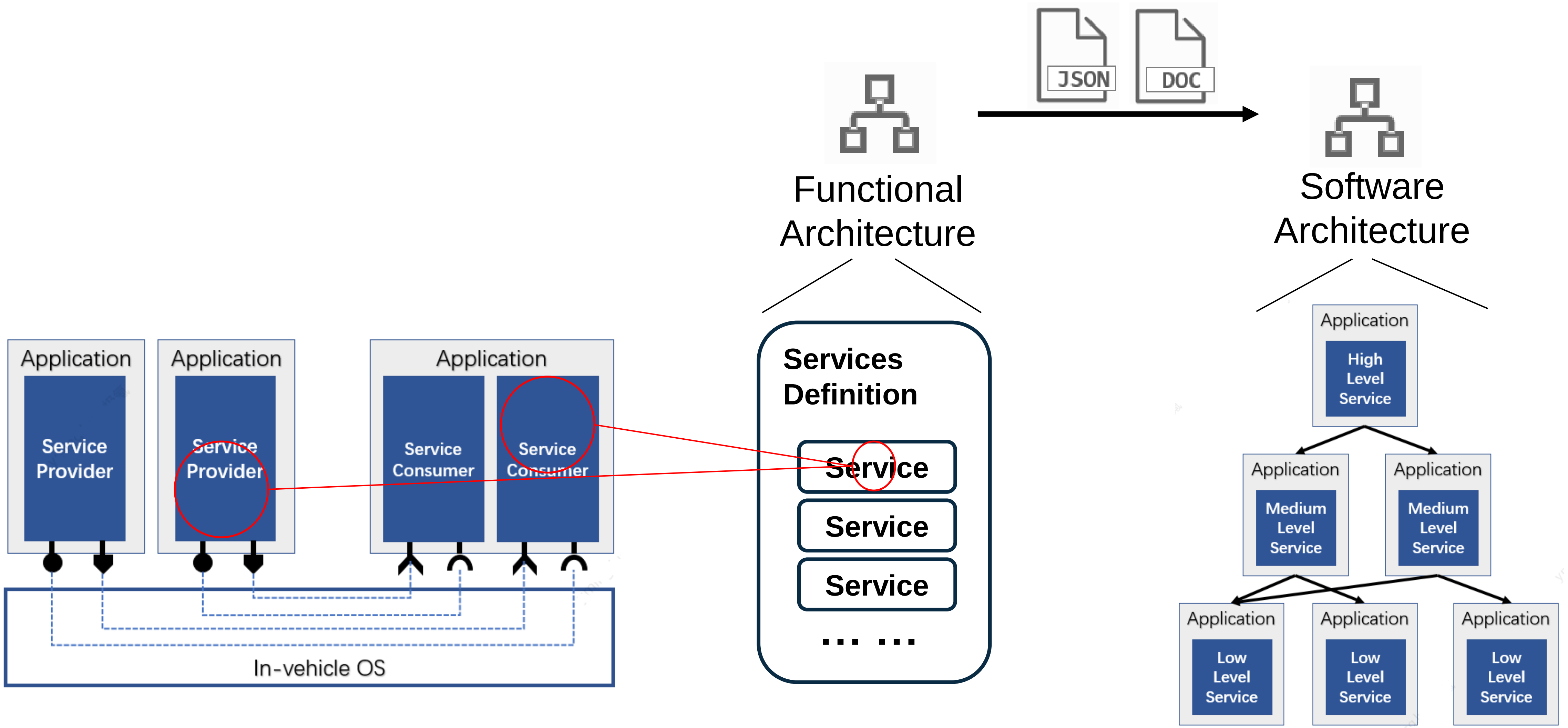
Example in Real Case



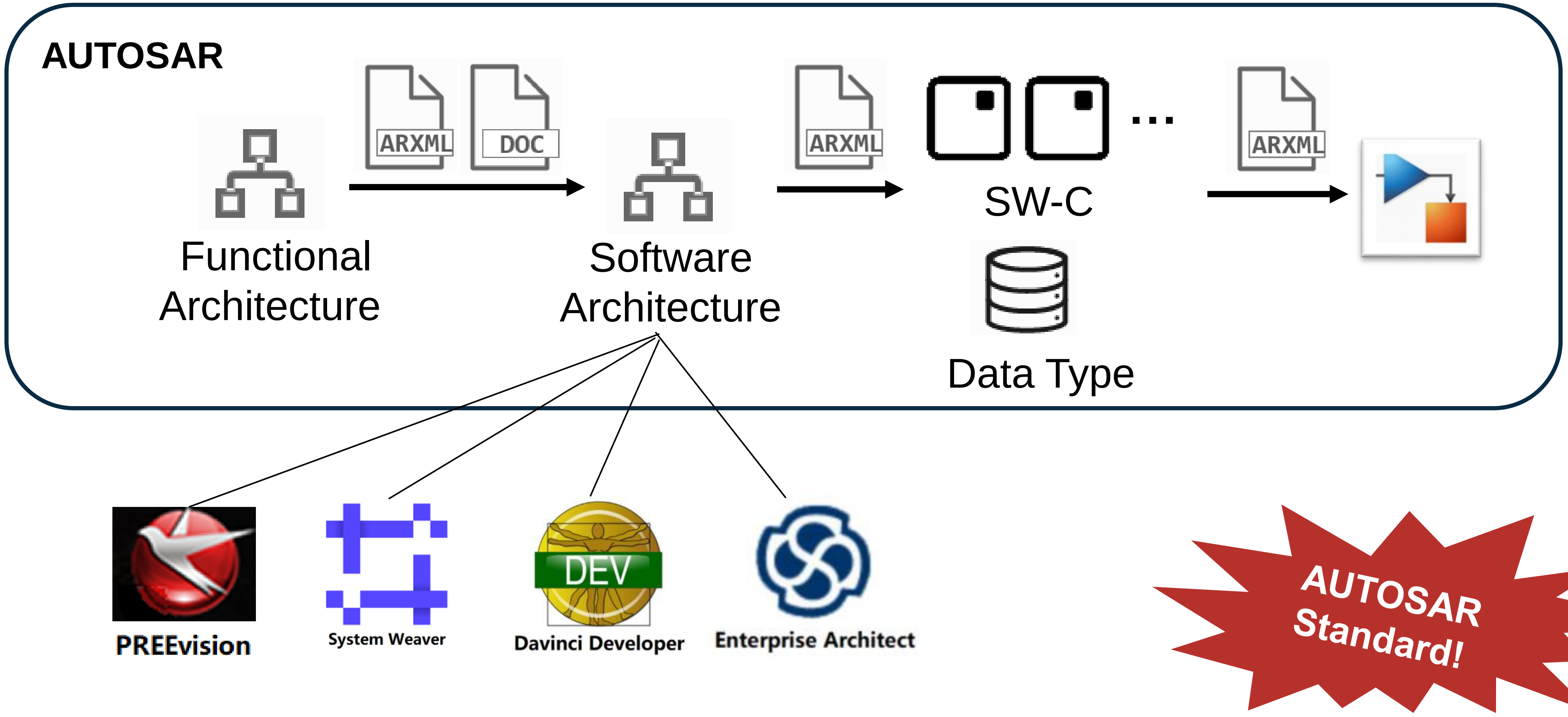

```
MAIN-PROCESS
  > config \ climatecontrolservice_server
  > files \ calibration
  v src
    v core
      > Clima_ert_rtw
      > slprj
    v facade
      model_facade.cpp
      model_facade.h
      service_facade.cpp
      service_facade.h
    > impl
    > service
    main.cpp
  CMakeLists.txt
```

PART II
How to Maintain Complex Software Clusters

What does Software Architecture do

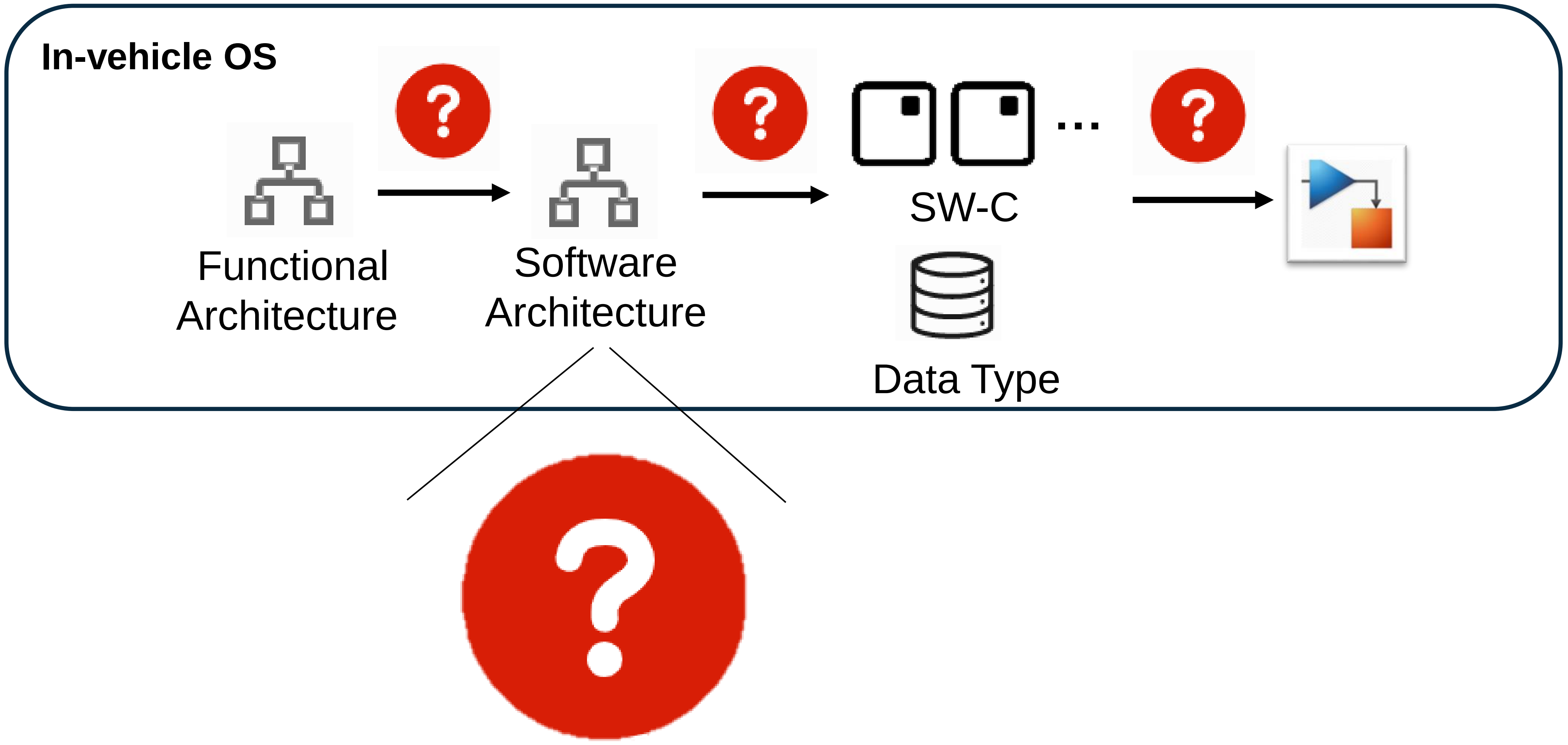


Software Management Dilemma

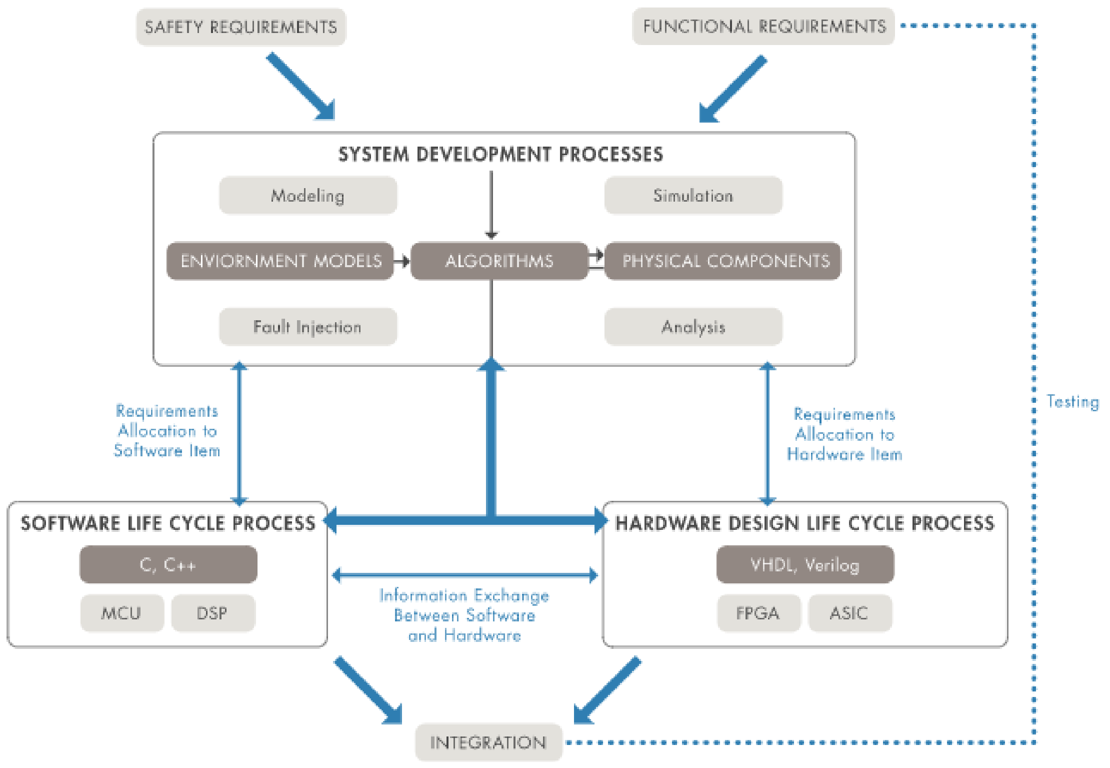


AUTOSAR Standard!

Software Management Dilemma



Model-Based Systems Engineering

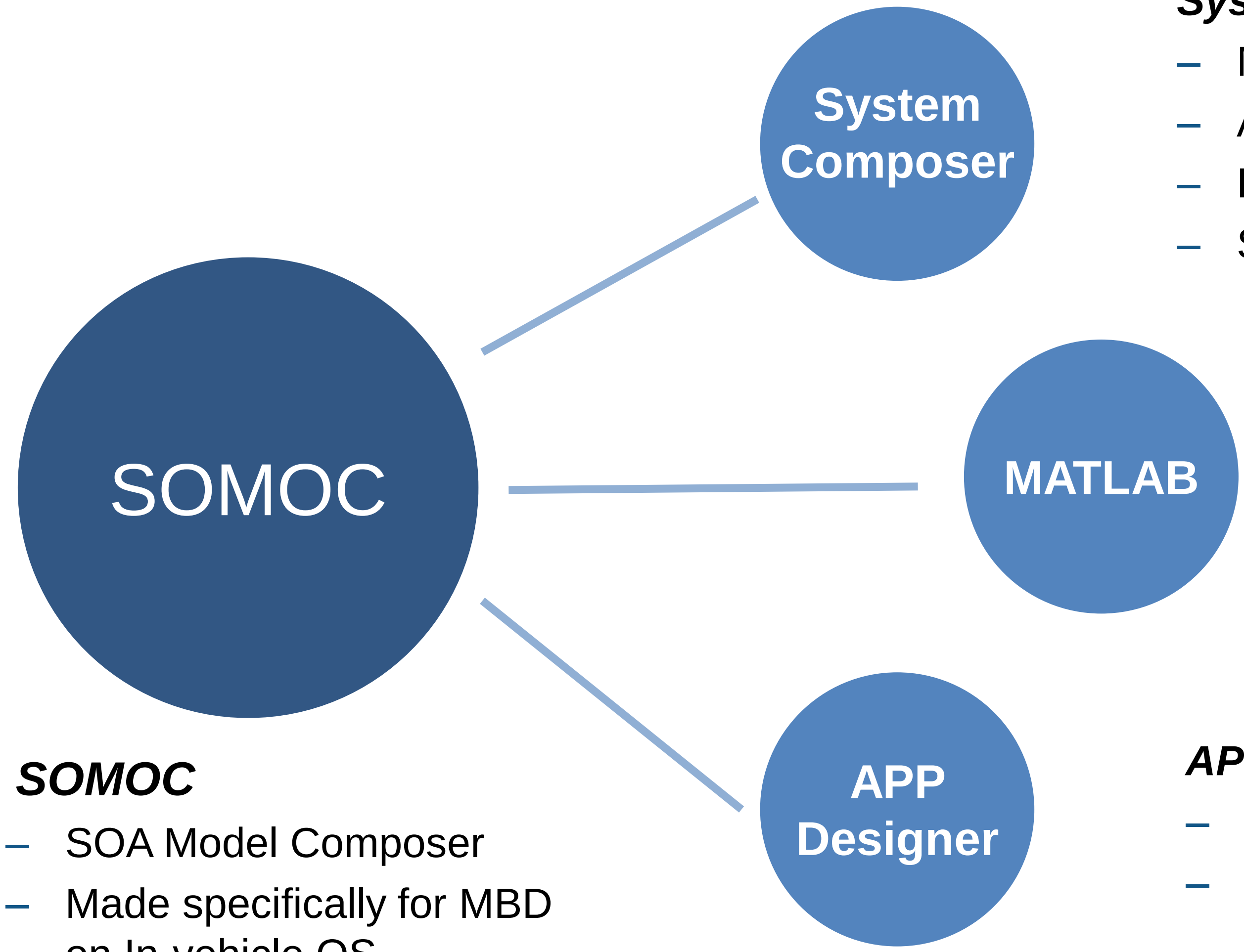


SAFETY REQUIREMENTS
FUNCTIONAL REQUIREMENTS
SYSTEM DEVELOPMENT PROCESSES
Modeling
Simulation
ENVIORNMENT MODELS
ALGORITHMS
PHYSICAL COMPONENTS
Fault Injection
Analysis
Requirements Allocation to Software Item
Requirements Allocation to Hardware Item
Testing
SOFTWARE LIFE CYCLE PROCESS
C, C++
MCU
DSP
HARDWARE DESIGN LIFE CYCLE PROCESS
VHDL, Verilog
FPGA
ASIC
Information Exchange Between Software and Hardware
INTEGRATION

- MBSE
  - Engineers use model-based systems engineering (MBSE) to manage system complexity, improve communication, and produce optimized systems.
  - MATLAB®, Simulink®, and System Composer™ together create a single environment for creating descriptive architecture models that seamlessly bridge into detailed implementation models.

https://www.mathworks.com/solutions/model-based-systems-engineering.html

Build Our Own Software Architecture Tool



System Composer

- Model-based systems engineering
- Architectural design elements
- Powerful visualization
- Support for customization

MATLAB

- Powerful programming language
- Support object oriented programming
- Rich product API

APP Designer

- Customized desktop applications
- Simple and easy to use

SOMOC

- SOA Model Composer
- Made specifically for MBD on In-vehicle OS

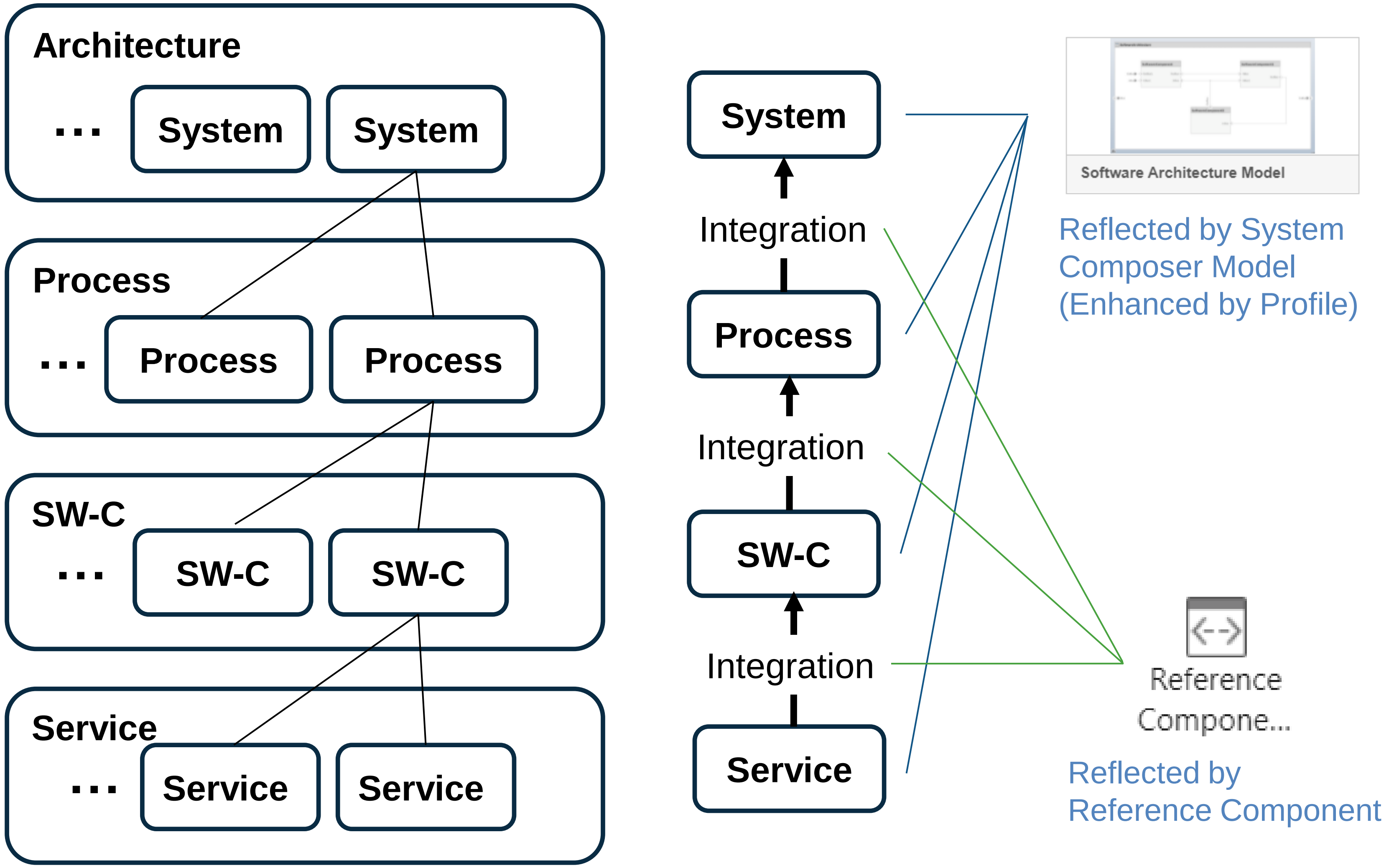
Different Software Architecture Level on SOMOC

Architecture Level
- Whole view of specific system

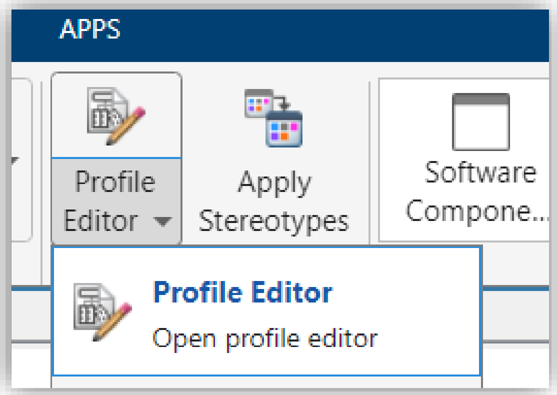
Process Level
- Minimum interagtion unit
- Consists of multiple SW-Cs
- Independent applications
- Communication via middleware

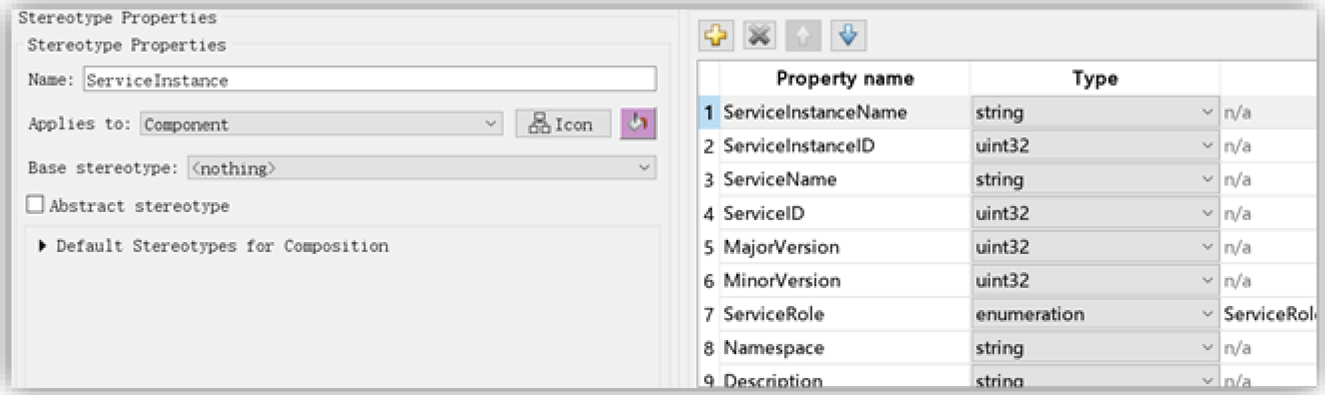
SW-C Level
- Minimum development unit
- Consists of multiple services
- Contains software logic and implementation

Service Level
- Service interface description
- Imported from upstream service design



Enhance System Composer Model by Profile





Stereotype Properties

Stereotype Properties

Name: ServiceInstance

Applies to: Component

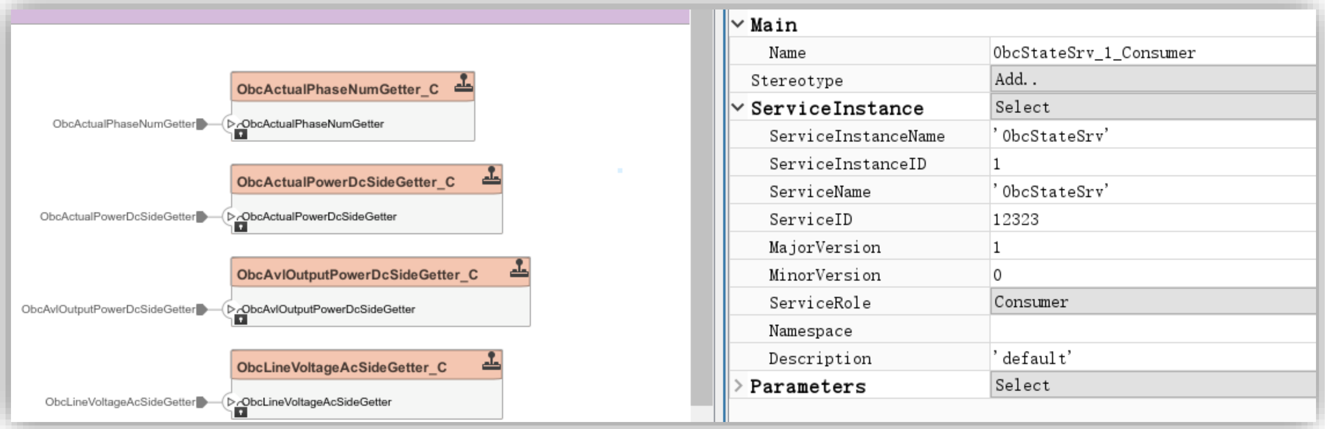
Base stereotype: <nothing>

☐ Abstract stereotype

▸ Default Stereotypes for Composition

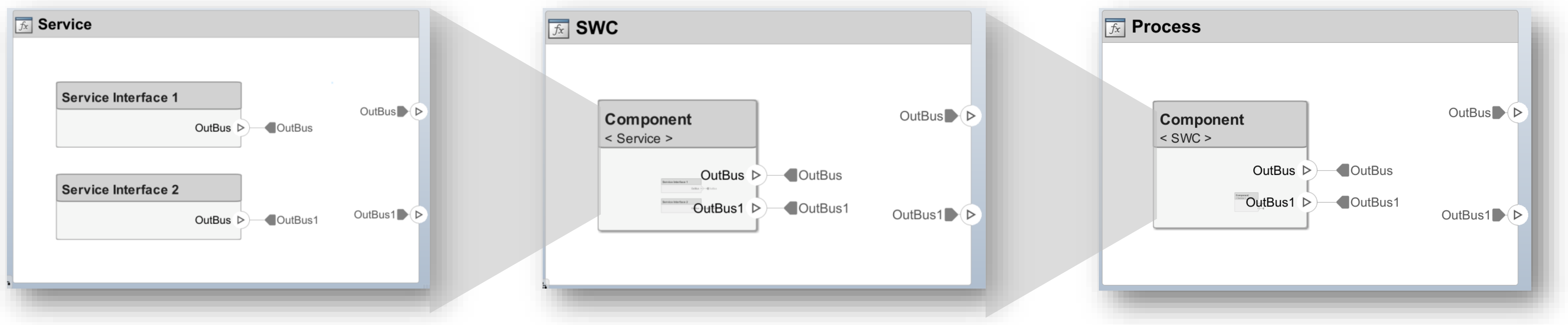
	Property name	Type	
1	ServiceInstanceName	string	n/a
2	ServiceInstanceID	uint32	n/a
3	ServiceName	string	n/a
4	ServiceID	uint32	n/a
5	MajorVersion	uint32	n/a
6	MinorVersion	uint32	n/a
7	ServiceRole	enumeration	ServiceRol
8	Namespace	string	n/a
9	Description	string	n/a

- Profile
  - Give custom property to different architecture elements
  - Enhance system composer to fit specific system requirement


ObcActualPhaseNumGetter_C
ObcActualPhaseNumGetter
ObcActualPhaseNumGetter
ObcActualPowerDcSideGetter_C
ObcActualPowerDcSideGetter
ObcActualPowerDcSideGetter
ObcAvlOutputPowerDcSideGetter_C
ObcAvlOutputPowerDcSideGetter
ObcAvlOutputPowerDcSideGetter
ObcLineVoltageAcSideGetter_C
ObcLineVoltageAcSideGetter
ObcLineVoltageAcSideGetter

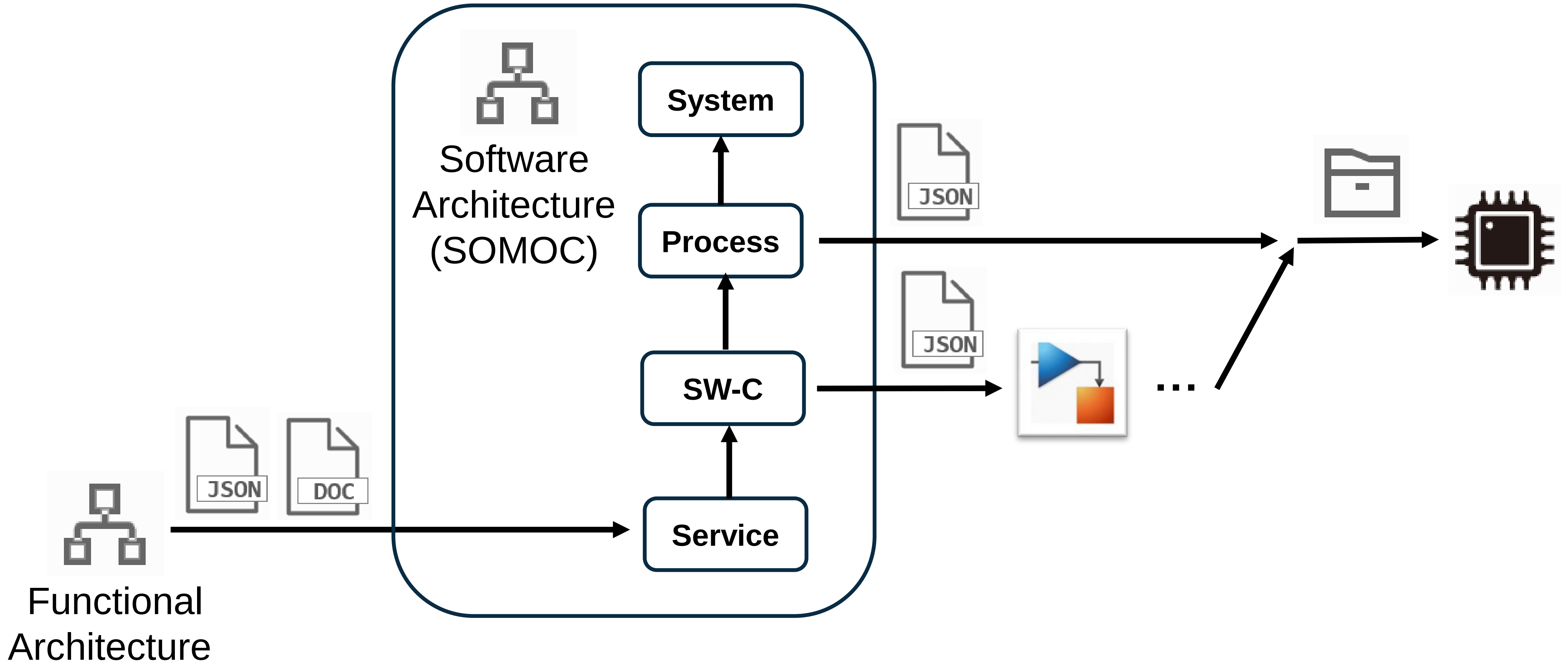
⌄ Main	
Name	ObcStateSrv_1_Consumer
Stereotype	Add..
⌄ ServiceInstance	Select
ServiceInstanceName	'ObcStateSrv'
ServiceInstanceID	1
ServiceName	'ObcStateSrv'
ServiceID	12323
MajorVersion	1
MinorVersion	0
ServiceRole	Consumer
Namespace	
Description	'default'
› Parameters	Select

Link Different Level Elements by Reference Component

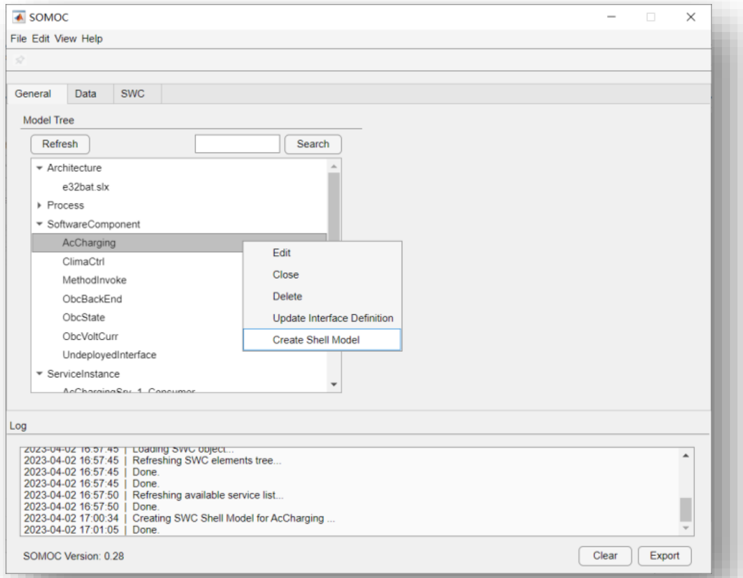


- Reference Component
  - Link to an architectural definition
  - Simulate the operation of integrating low level elements into high level elements
  - System Composer models carry whole information of an software architecture

SOMOC Workflow

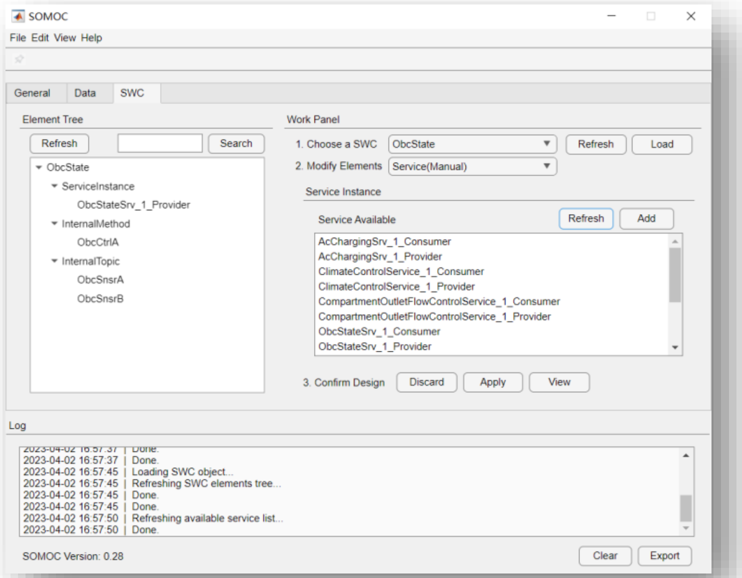


SOMOC GUI



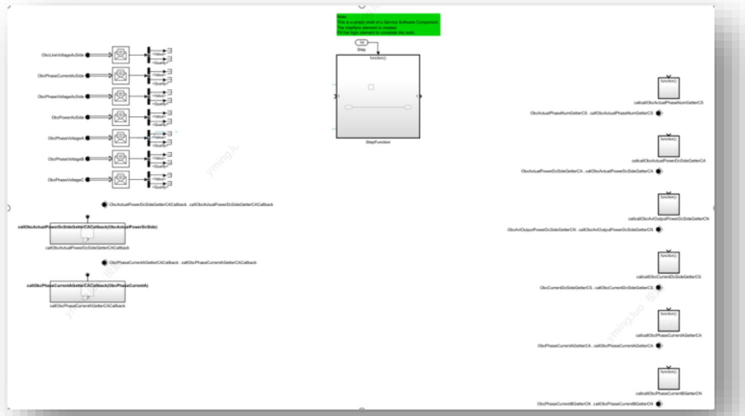
```
2023-04-02 16:57:45 | Loading SWC object...
2023-04-02 16:57:45 | Refreshing SWC elements tree...
2023-04-02 16:57:45 | Done.
2023-04-02 16:57:45 | Done.
2023-04-02 16:57:50 | Refreshing available service list...
2023-04-02 16:57:50 | Done.
2023-04-02 17:00:34 | Creating SWC Shell Model for AcCharging ...
2023-04-02 17:01:05 | Done.
```

SOMOC Version: 0.28



```
2023-04-02 16:57:37 | Done.
2023-04-02 16:57:37 | Done.
2023-04-02 16:57:45 | Loading SWC object...
2023-04-02 16:57:45 | Refreshing SWC elements tree...
2023-04-02 16:57:45 | Done.
2023-04-02 16:57:45 | Done.
2023-04-02 16:57:50 | Refreshing available service list...
2023-04-02 16:57:50 | Done.
```

SOMOC Version: 0.28



Architecture Tree

- Tree view of different level components
- Context menu for different kind of tree nodes

Component Composer Interface

- Simplified operation of a component
- Add or remove elements in a component

Export Shell Model

- Automatically export shell model according to component interface definition
- Transferred downstream for Simulink model detail design

Conclusion

- How to Model the Software Behavior
  - SOA Behavior Modeling
  - SIMULINK New Features
  - Wrapper Code Generator
- How to Maintain Complex Software Clusters
  - Software Architecture Engineering
  - System Composer(Deeply Customized)

MATLAB EXPO

Thank you

MathWorks®