

# 10 Best Practices for Deploying AUTOSAR Using Simulink

By David Jaffry and Holly Keener

## Contents

Glossary .....	2
Introduction .....	3
Overview of Simulink Support for AUTOSAR .....	5
Best Practices for Adopting AUTOSAR .....	7
Learn More .....	18

## Glossary

A discussion of AUTOSAR deployment involves many acronyms. Here is a short list of terms used in this guide.

Acronym	Meaning
AAT	AUTOSAR authoring tool
API	Application programming interface
ARXML	AUTOSAR XML
AUTOSAR	AUTomotive Open System ARchitecture
BSW	Basic software
CRL	Code Replacement Library (feature of Embedded Coder)
ECU	Electronic control unit
IDE	Integrated development environment
OEM	Original equipment manufacturer
PIL	Processor-in-the-loop
RTE	Run-time environment
SIL	Software-in-the-loop
SWC	Software component
V&V	Verification and validation
XML	eXtensible Markup Language

## Introduction

### AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of vehicle manufacturers, suppliers, and other companies from the electronics, semiconductor, and software industry. The AUTOSAR standard is designed to enable software standardization, reuse, and interoperability.

The AUTOSAR standard provides two platforms to support the current and the next generation of automotive ECUs. The first is the Classic Platform, used for traditional applications such as powertrain, chassis, body and interior electronics, and so forth. The second is the Adaptive Platform, used for new applications such as highly automated driving, Car-to-X, software updates over the air, or vehicles as part of the Internet of Things. The Foundation AUTOSAR standard is introduced to enforce interoperability between the AUTOSAR platforms.

### MathWorks Support

MathWorks is an AUTOSAR Premium Member and actively participates in the development of the standard with focus on how to use Model-Based Design with an AUTOSAR development process for automotive electronic control units (ECUs). Using Simulink® and AUTOSAR Blockset (Figure 1), you can:

- Model AUTOSAR Classic and Adaptive software components created in Simulink or imported from ARXML component descriptions.
- Refine AUTOSAR component configuration and create algorithmic model content.
- Create AUTOSAR architecture models in Simulink (requires System Composer™).
- Simulate AUTOSAR component interactions at the system level using Simulink library blocks, including Basic Software services such as NVRAM Manager and Diagnostic Event Manager.
- Generate ARXML component descriptions and algorithmic C/C++ code for testing in Simulink or integration into the AUTOSAR Runtime Environment (RTE) (with Embedded Coder®).

This guide summarizes recommended best practices for AUTOSAR deployment with Simulink based on substantial experience deploying AUTOSAR with Model-Based Design. The result is a process for a systematic approach to enterprise-wide AUTOSAR development, designed to maximize benefits while minimizing the costs, risks, and disruptions associated with achieving those benefits. These recommendations are based on practical experience in guiding automotive OEM and supplier customers in their model-based AUTOSAR deployment with MATLAB® and Simulink.

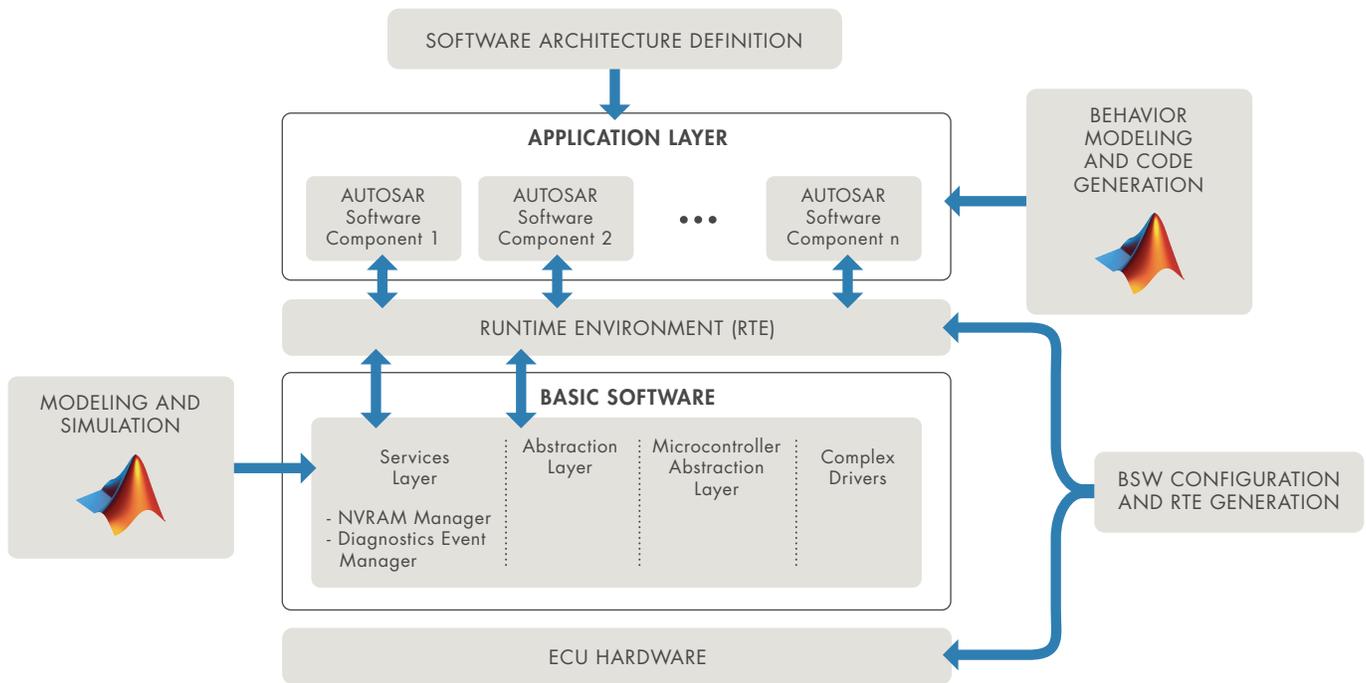


Figure 1. Classic AUTOSAR layered architecture.

The top 10 best practices for effective AUTOSAR deployment with Simulink are:

1. Determine your strategy for migrating existing Simulink models to AUTOSAR
2. Use one AUTOSAR workflow
3. Select a data management strategy
4. Establish a modeling standard
5. Simulate before you generate code
6. Use production code generation
7. Use Simulink to migrate legacy code to AUTOSAR
8. Automate, automate, automate
9. Plan ahead for ISO 26262
10. Actively plan for migration

# Overview of Simulink Support for AUTOSAR

Let's start with a quick review of how the Simulink product family supports AUTOSAR. You can use Simulink and AUTOSAR Blockset to design, implement, and verify AUTOSAR atomic software components (SWCs). Each AUTOSAR software component can contain one or many entry point functions, which are implemented in AUTOSAR as runnable entities. AUTOSAR Blockset allows a Simulink model to be mapped to AUTOSAR such that Embedded Coder is able to generate AUTOSAR-compliant C/C++ code and AUTOSAR XML (ARXML) files.

Specific aspects of the Simulink model are used to implement AUTOSAR concepts. Once you configure a Simulink model to use the AUTOSAR Target, you can manage the AUTOSAR attributes and map the Simulink model to AUTOSAR through the editor shown in Figure 2:

- Configure *AUTOSAR components*
- Map AUTOSAR elements for code generation using the *Code Mappings Editor*
- Generate readable, compact, and fast C and C++ code from models, which use AUTOSAR blocks, for embedded processors used in mass production using *AUTOSAR Component Designer*

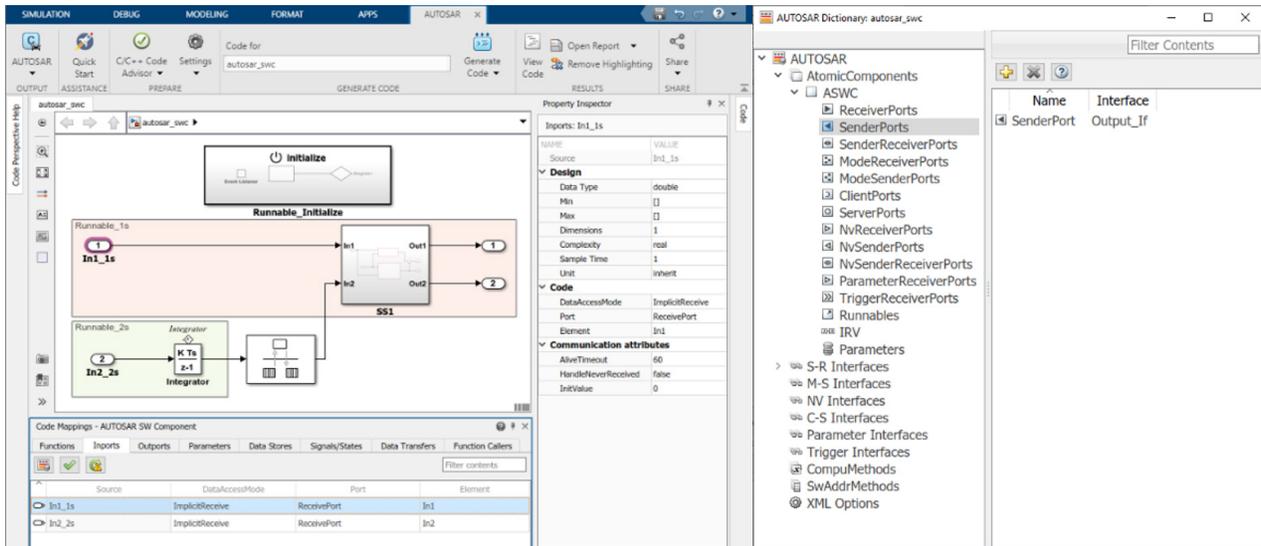


Figure 2. Simulink to AUTOSAR mapping.

You can select from several approaches and workflows when adopting AUTOSAR. Two common approaches, which are depicted in Figure 3, are the top-down approach and the bottom-up approach.

## Top-down approach

You start by designing the architecture of the system in the AUTOSAR authoring tool (AAT), then generate ARXML files and import them into Simulink to generate a shell of a model. You then design the Internal Behavior in Simulink. After validating your software component, you generate C code. The AAT manages all ARXML files.

## Bottom-up approach

You start by using Simulink to create, develop, and validate your software components. If you have System Composer, you can create a software architecture canvas for developing AUTOSAR compositions and components. You can add simulation behavior including Basic Software service components. After developing your software components, generate ARXML component descriptions and algorithmic C or C++ code for testing in Simulink or integration into the AUTOSAR run-time environment. The AAT manages all ARXML files.

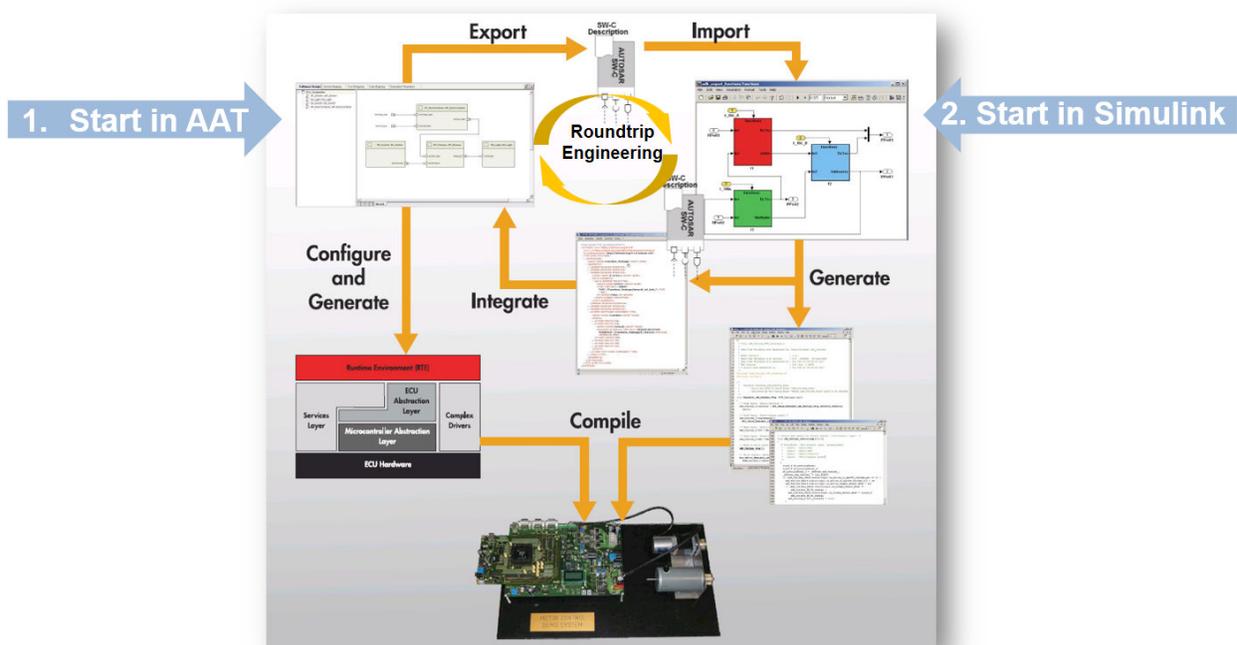


Figure 3. AUTOSAR workflows using Simulink. A top-down approach starts in the AUTOSAR authoring tool; a bottom-up approach starts in Simulink.

# Best Practices for Adopting AUTOSAR

## 1. Determine your strategy for migrating existing Simulink models to AUTOSAR

A common workflow is to use Simulink to design software algorithms prior to the adoption of AUTOSAR (Figure 4). Deciding on a uniform approach for the development team to use in migrating these designs from a non-AUTOSAR approach to AUTOSAR is important to ensure the team can use common processes and tools. The three options for handling the migration of these designs to AUTOSAR are:

- Clean sheet
- Complete migration to AUTOSAR
- One model for AUTOSAR and non-AUTOSAR

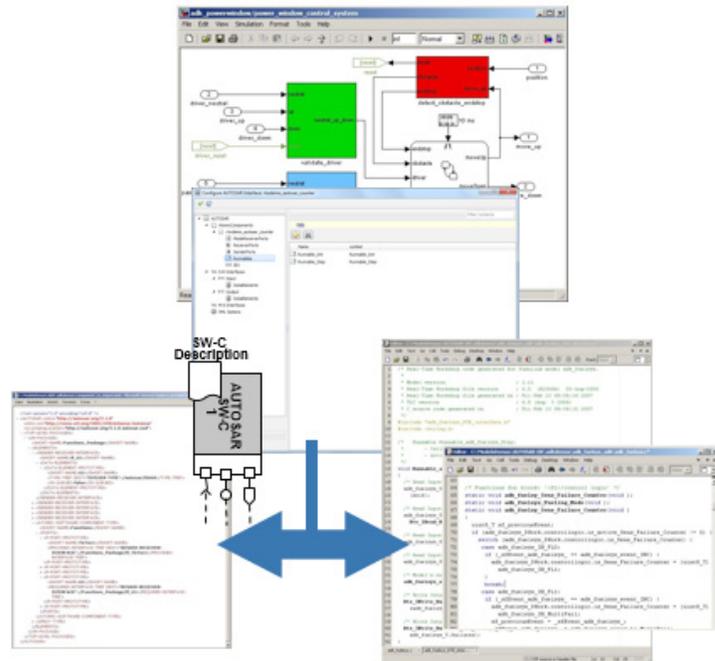


Figure 4. Code generation from Simulink.

### Clean sheet

If you will be substantially redesigning the algorithms, starting with a “clean sheet” is the best approach. You start the design process from requirements, and develop new Simulink models that are well suited for AUTOSAR from the start (Figure 5). This option provides for the most flexibility in getting the most out of AUTOSAR and Model-Based Design because there are no legacy design constraints. However, this option is only practical if there will be minimal reuse of those legacy implementations.

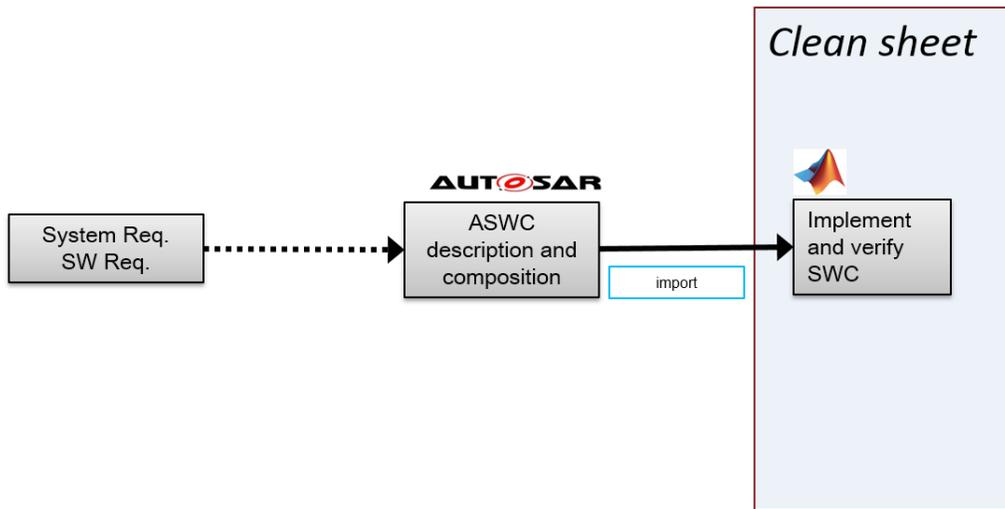


Figure 5. Clean sheet migration.

### Complete migration to AUTOSAR

If you need to reuse existing Simulink algorithms but will be abandoning your current software architecture in favor of AUTOSAR, the best approach is to completely convert or migrate your Simulink models to the optimal modeling style for AUTOSAR. You select the ideal Simulink modeling style and feature set for AUTOSAR and then plan to migrate each model to adopt this approach (Figure 6). Where necessary, you should update the Simulink implementation to reflect this modeling style. A best practice is to perform regression testing using simulations prior to and after changing the Simulink model to ensure you have not introduced any change in the functionality. This approach is an effective compromise between reuse of legacy implementations and taking advantage of AUTOSAR with Model-Based Design.

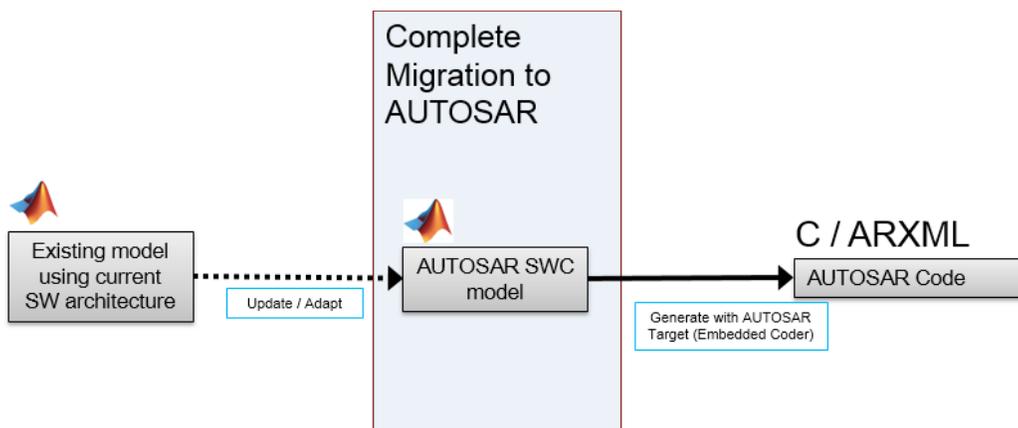


Figure 6. Complete migration to AUTOSAR.

## One model for AUTOSAR and non-AUTOSAR

One of the key advantages of using Simulink and Embedded Coder with AUTOSAR is that you can generate code for both AUTOSAR and non-AUTOSAR targets from a single model (Figure 7). This approach is effective when you will be developing both AUTOSAR and non-AUTOSAR applications for a single implementation in parallel. Similar to the complete migration approach, you will make a one-time change in modeling style and Simulink feature selection in order to ensure the design is compatible with the Simulink modeling styles and constraints required with AUTOSAR. Again in this scenario, regression testing is an important part of the migration process. Working with one implementation for multiple architectures may constrain the extent to which you can adopt AUTOSAR. However, you can make significant gains in efficiency by designing and testing a single model and then deploying it to two target architectures.

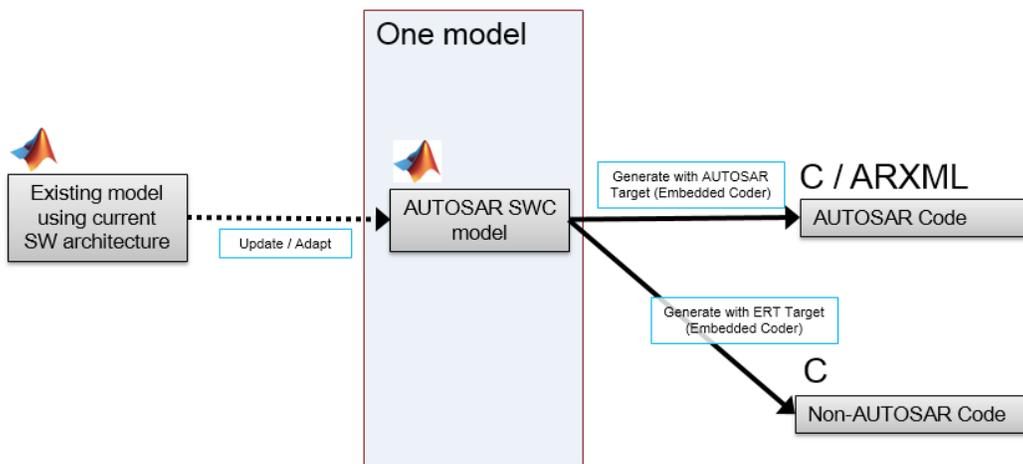


Figure 7. One-model migration.

Regardless of your migration approach, it is important to select the appropriate scope for the migration process, starting with a small pilot project with a representative Simulink model and then scaling up to more complete portions of the system. A typical embedded system spans multiple domains; it is important that its components can be clearly represented in the modeling domains available in Model-Based Design.

## 2. Use one AUTOSAR workflow

The AUTOSAR standard includes a requirement for tool interoperability, which enables efficient top-down and bottom-up workflows (Figure 8). Evaluate the pros and cons of each workflow and then select the one that best meets the goals of your initial AUTOSAR adoption. Introducing different workflows within a single project can result in confusion, rework, and waste due to unnecessary iterations and checks to ensure consistency in the data and the design.

The round trip, in particular, works best with one clear owner of data.

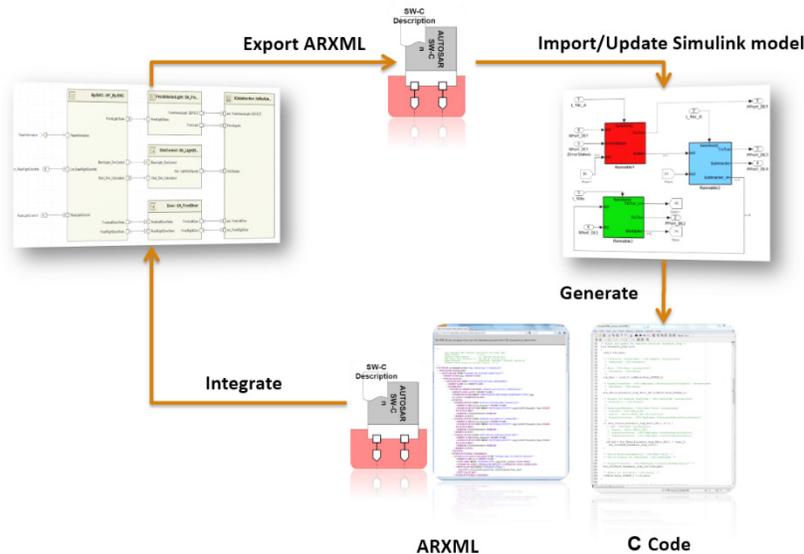


Figure 8. Round-trip AUTOSAR workflow.

Once you have selected a workflow, choose the simplest approach for applying the AUTOSAR configuration to the Simulink model. With the top-down workflow, use the built-in functionality in AUTOSAR Blockset to apply the AUTOSAR configuration by importing the ARXML and creating or updating a Simulink model. With the bottom-up workflow, AUTOSAR Blockset provides both graphical and programmatic methods for applying and updating the AUTOSAR configuration. Avoid introducing complex custom approaches for managing and configuring the AUTOSAR properties, which creates a legacy of maintenance of custom solutions. Commercial tools provide strong support for the most important workflows.

Select tools that best support the workflow selected and the AUTOSAR concepts critical to adoption.

### 3. Select a data management strategy

AUTOSAR provides a rich schema for specifying critical design data. How you manage this critical design data is a crucial part of your AUTOSAR adoption strategy. Data can be managed by Simulink, AUTOSAR Authoring Tool, or an external tool. Data can also be defined and managed by different owners and at different levels. Change management is another key part of the data management strategy. It is a best practice to base the data management strategy on the AUTOSAR workflow selected, the ownership strategy for the data, the approach used to manage the legacy data, and the strategy for migrating existing Simulink designs to AUTOSAR.

The AUTOSAR Component Designer app in Simulink (Figure 9) lets you manage the data of AUTOSAR software components. Parameter and variable data are internal and scoped to a component, while system-wide constants are shared across components. The Component Designer app provides a Code Mappings editor and a Property Inspector for managing internal component data. These Simulink tools complement the management of AUTOSAR-specific data in separate tools.

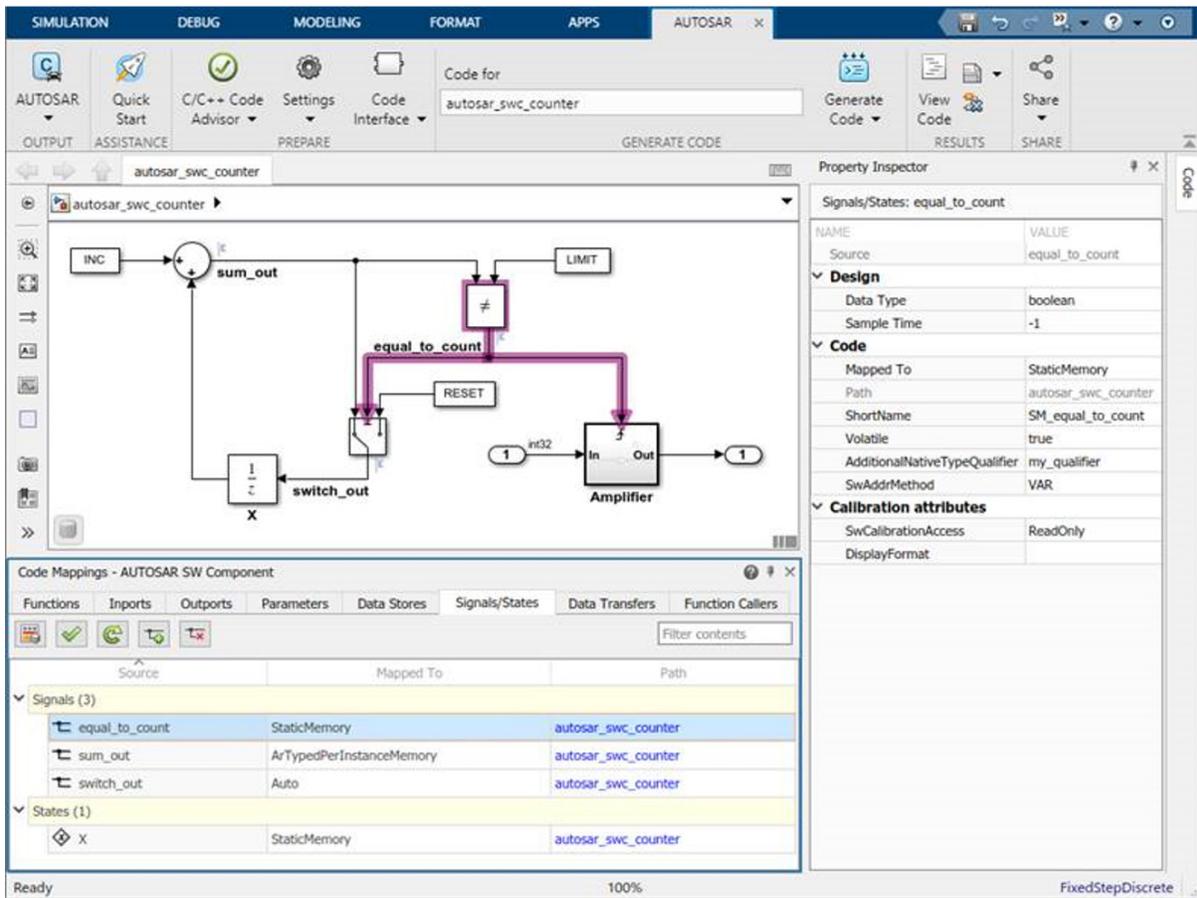


Figure 9. The AUTOSAR Component Designer app.

Furthermore, you can configure component data for *run-time calibration and measurement* to meet your requirements.

## 4. Establish a modeling standard

Establish a modeling standard for Simulink and AUTOSAR to ensure uniform adoption across the organization that enables reuse across projects and scales up to large systems and teams. The modeling standard is a best practice for any deployment of Model-Based Design and is also required by the ISO 26262 safety standard.

The modeling standard should be based on the AUTOSAR adoption approach with Simulink, feature set selected, workflow, and data management strategy. Best practices for modeling AUTOSAR software components include:

- Data encapsulation: Within each AUTOSAR software component, encapsulate parameter and variable data. Use Simulink parameters, data stores, signals, and states to model AUTOSAR component internal data. With data encapsulation, instances of a component can be flexibly included in component hierarchies and system-level simulations or used in multiple projects.
- Interface management: Share interfaces and types across AUTOSAR components and projects. With AUTOSAR Blockset, you can predefine interfaces and types for AUTOSAR software components to import or create. For example, your organization can predefine and share:
  - AUTOSAR component port interfaces and their input/output data types
  - AUTOSAR component internal data types (IncludedDataTypeSets)
  - AUTOSAR data types and related elements (AUTOSAR XML packages)
  - AUTOSAR system constants

You can also learn about modeling patterns (Figure 10) for AUTOSAR software components behavior in the [documentation](#).

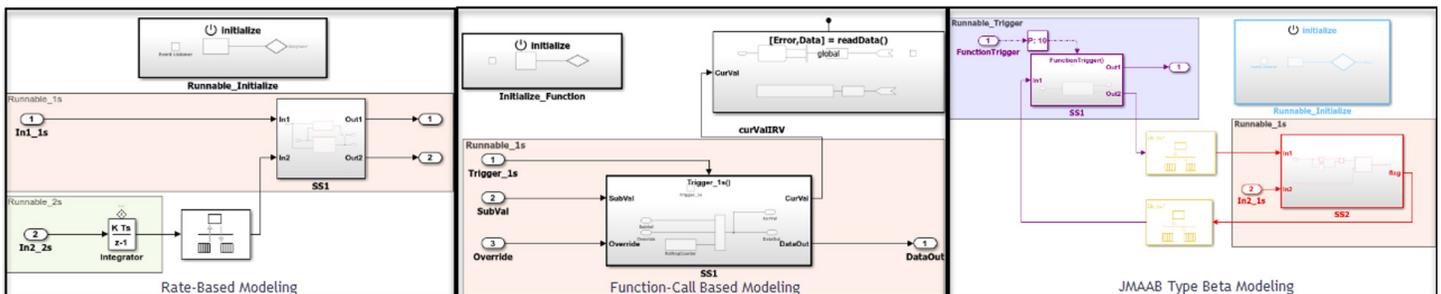


Figure 10. Modeling patterns.

## 5. Simulate before you generate code

With the introduction of AUTOSAR, it is easy to become caught up in the new workflows, the correctness of ARXML files, and the impact on the software. However, the primary reason to invest in designing software components in Simulink is to ensure the implementation meets the requirements long before you generate ARXML and integrate with AAT and RTE generation tools. Simulink ensures that each capability introduced to support the richness of the AUTOSAR standard can also be simulated effectively (Figure 11).

The simulation also enables you to:

- Ensure software component implementation is correct early in the design process.
- Simulate multiple SWCs (AUTOSAR Composition) together with Simulink before code integration.
- Reuse tests for software-in-the-loop (SIL) and processor-in-the-loop (PIL) to verify generated code at the unit level before RTE generation.

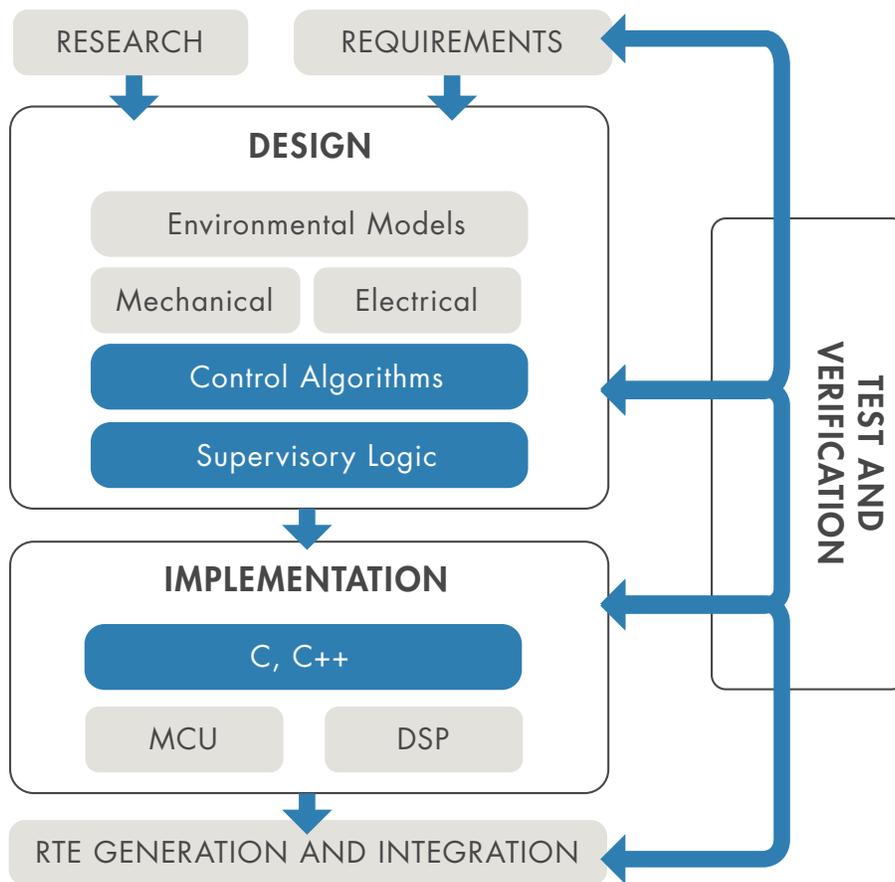


Figure 11. Activities in Model-Based Design.

## 6. Use production code generation

With the AUTOSAR standard, tools are used to generate the AUTOSAR Basic Software (BSW), the RTE, the ARXML files, and more, so the question of whether or not to generate the code for the software component becomes clear. It is a best practice to use production code generation to implement the software components. Based on the complexity of the APIs, as shown in Figure 12, manually coding AUTOSAR is clearly painful. Using AUTOSAR Blockset with Embedded Coder will also help produce an implementation that closely approximates the design behavior in Simulink, which helps you make the most of your simulation and verification activities.

```

void SeatBeltReminder_Logic(void)
{
  boolean_T TmpSignalConversionAtSeatBelt_a;
  uint16_T TmpSignalConversionAtSpeedOutpo;
  TmpSignalConversionAtSeatBelt_a =
    Rte_IrvIRead_SBR_Logic_SeatBeltfasten_debounced();
  TmpSignalConversionAtSpeedOutpo = Rte_IrvIRead_SBR_L
  if ((uint32_T)Seat_Belt_Reminder_AUTOSA_Dwork.tempora
    Seat_Belt_Reminder_AUTOSA_Dwork.temporalCounter_i1
    ((int32_T)Seat_Belt_Reminder_AUTOSA_Dwork.tempora
  }
  switch
  (Seat_Belt_Reminder_AUTOSA_Dwork.bitsForTID2.is_c2_Seat_Belt_Reminder_AUTOSA)
  {
  case Seat_Belt_Reminder_IN_KEY_CRANK:
    if ((int32_T)Rte_IRead_SBR_Logic_KEY_KEY() == 1) {
      Seat_Belt_Reminder_AUTOSA_Dwork.bitsForTID2.is_c2_Seat_Belt_Reminder_AUTOSA
      = Seat_Belt_Reminder_AU_IN_KEY_ON;
      .....
    }
  case Seat_Belt_Reminder_A_IN_KEY_OFF:
    if ((int32_T)Rte_IRead_SBR_Logic_KEY_KEY() == 1
    .....
    }
    break;
  }
  .....
  Rte_IWrite_SBR_Logic_SeatBeltIcon SeatBeltIcon
  (Seat_Belt_Reminder_AUTOSAR_B.SeatBeltIcon);
}

```

```

</RUNNABLE-ENTITY>
<RUNNABLE-ENTITY UUID="08f71b09-4c5d-52bb-f537-04e3170d9e5d">
  <SHORT-NAME>SeatBeltReminder_Logic</SHORT-NAME>
  <MINIMUM-START-INTERVAL>0</MINIMUM-START-INTERVAL>
  <CAN-BE-INVOKED-CONCURRENTLY>>false</CAN-BE-INVOKED-CONCURRENTLY>

```

```

<SENDER-RECEIVER-INTERFACE UUID="a0f3e55a-368d-5ac3-825d-a92f04518baf">
  <SHORT-NAME>SeatBeltIcon</SHORT-NAME>
  <IS-SERVICE>>false</IS-SERVICE>
  <DATA-ELEMENTS>

```

Figure 12. Generated code using the AUTOSAR API.

With Embedded Coder, you can generate C/C++ code for production from Simulink. The code can be AUTOSAR or non-AUTOSAR compliant.

## 7. Use Simulink to migrate legacy code to AUTOSAR

Non-AUTOSAR source code (legacy code) that has been fully validated and widely used is commonly reused as part of the initial AUTOSAR adoption. You will need to make some modifications to this code to ensure that it is AUTOSAR compliant. In addition to automatically generating code, it is a best practice to automatically generate AUTOSAR-compliant wrappers for the legacy code. You can bring this code into Simulink using a C S-Function block or C Caller block so that you can then map the model to AUTOSAR. With this approach, you can integrate C or C++ code into Simulink for simulation and production code generation.

Integrating your code into Simulink lets you:

- Create test harnesses
- Visualize and analyze output
- Achieve model coverage including the integrated legacy code

For production code generation, Embedded Coder will generate the necessary AUTOSAR RTE API access points and runnable entities as a wrapper around the legacy code. Figure 13 shows an example of the resulting generated code.

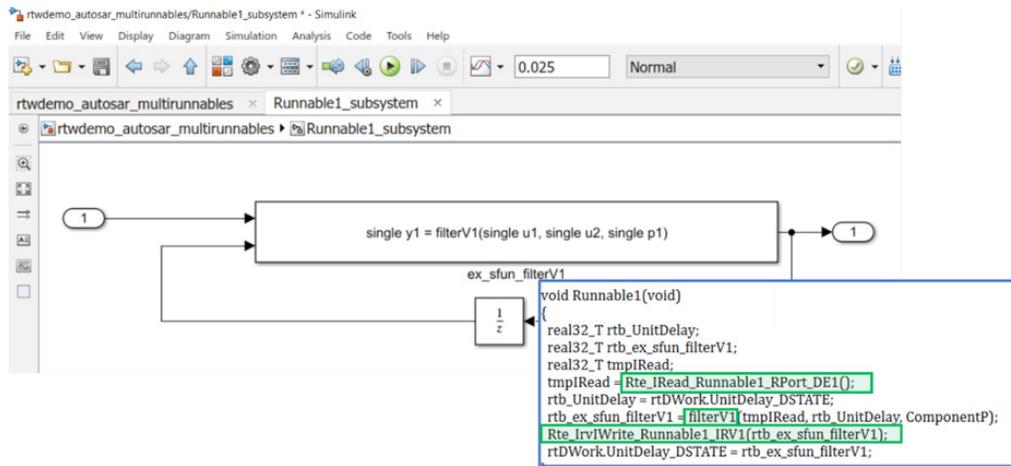


Figure 13. Integrating legacy code using the AUTOSAR API.

## 8. Automate, automate, automate

In a typical AUTOSAR workflow, common process steps are executed over and over across the team. Manually configuring and deploying AUTOSAR can be difficult due to:

- The complexity of the standard such as naming conventions
- Iterative work cycles with AUTOSAR
- Complex code APIs and XML file definitions

Using APIs for workflow automation is a best practice. You can use documented MATLAB APIs, such as those shown in Figure 14, to configure SWCs in Simulink. While creating functions to automate these tasks, take the time to ensure that each automated task is well documented and can be verified so that upgrades to new releases can be executed smoothly and some custom solutions can be retired over time.

Stick with the *documented APIs* only to ease future migration with new MATLAB releases.

```
%% Setup AUTOSAR Configuration Programmatically
model = 'autosar_swc_counter';

% Modify AUTOSAR Properties
autosarProps = autosar.api.getAUTOSARProperties (model);
set (autosarProps, 'Input', 'IsService', true);
set (autosarProps, 'XmlOptions', 'ArxmlFilePackaging', 'SingleFile');
```

Figure 14. MATLAB API of the AUTOSAR target.

## 9. Plan ahead for ISO 26262

ISO 26262 is a worldwide standard for the functional safety of road vehicles. Similar to the adoption of AUTOSAR, the need to follow this standard can disrupt both the development process and implementation. It is a best practice to evaluate and plan for the adoption of ISO 26262 while introducing change to adopt AUTOSAR. This approach will ensure the AUTOSAR process and tools can scale to address safety standards such as ISO when the need arrives.

IEC Certification Kit supports ISO 26262 by providing tool qualification for:

- Embedded Coder
- Simulink Check™
- Simulink Coverage™
- Simulink Design Verifier™
- Simulink Requirements™
- Simulink Test™
- Polyspace Bug Finder™
- Polyspace Code Prover™

Artifacts delivered in IEC Certification Kit are certified by TÜV SÜD (Figure 15).

*Learn more about the ISO 26262 standard.*

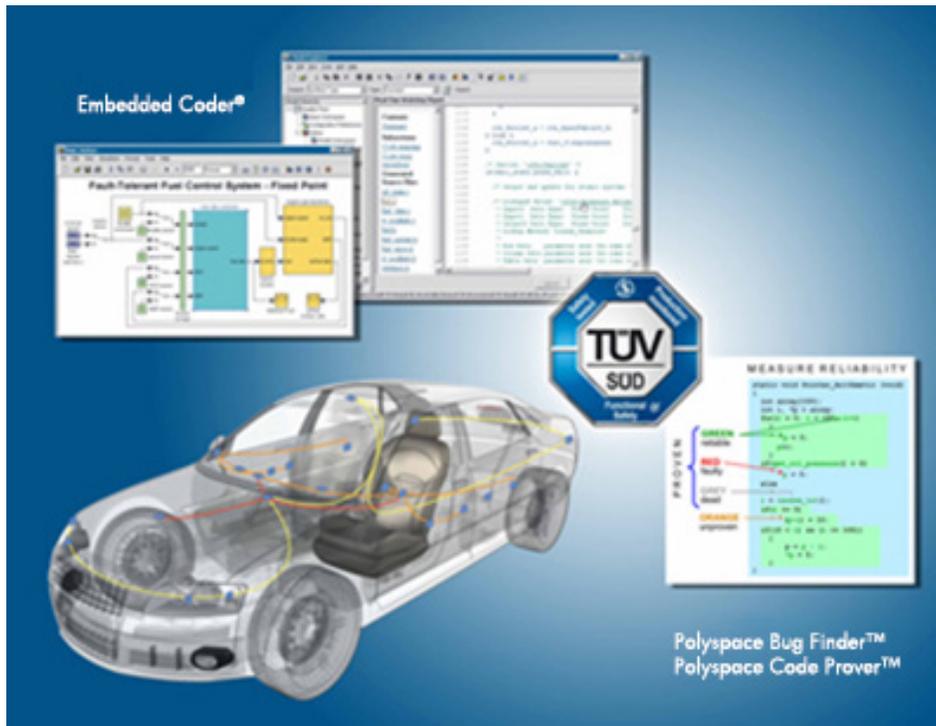


Figure 15. IEC Certification Kit for ISO 26262.

## 10. Actively plan for migration

Relative to the development of embedded systems in the automotive industry, AUTOSAR is still a relatively young standard and is still evolving. MathWorks is expanding tool support for AUTOSAR with each product release. Simulink and Embedded Coder include important new capabilities for AUTOSAR and beyond in each new version made available every six months.

It is a best practice to adopt a strategy for managing new versions of AUTOSAR and Simulink products, including how often to upgrade and what criteria you will use to select new versions.

In general, MathWorks recommends adopting a continuous upgrade philosophy. Continuously performing upgrade activities ensures that the next upgrade is easier than the last upgrade. To ease the adoption of this philosophy, consider participating in prerelease testing and Industry Model Testing, as well as MathWorks seminars, webinars, and conferences.

Read the white paper [MATLAB and Simulink Version Upgrades for Large Organizations](#) to learn more about managing the upgrade process.

## Learn More

Get started with AUTOSAR development with these resources:

- [Documentation](#)
- [Training](#)
- [Consulting](#)

### About the Authors

David Jaffry is a senior consultant engineer in MathWorks Consulting Services. He helps automotive, aerospace, biotech and industrial automation companies with Model-Based Design implementation, verification and validation techniques, and embedded system development. Before coming to MathWorks, his work encompassed software development and code verification for embedded systems. David holds a master degree in computer systems engineering from the École Nationale d'Ingénieurs de Brest.

Holly Keener is a manager for MathWorks Consulting Services based in Michigan. She works with industry-leading companies on a wide range of applications in the automotive, aerospace, defense, communications and medical device industries. She specializes in helping organizations successfully adopt Model-Based Design for developing and verifying embedded systems. Holly has led the way in the application of Model-Based Design for AUTOSAR with automotive OEMs and suppliers throughout the world. Before joining MathWorks, Holly developed embedded systems in the automotive industry. Holly received her B.S. in electrical engineering and M.S.E. in biomedical engineering from the University of Michigan.