



Enhanced Data Dictionary

for model based automotive production software development

Todd Nordby, Technical Specialist

Agenda

- About
- Background and context
- Our organization's workflow and motivation
- Solution
- Status
- Summary

About NAVISTAR

- **Major manufacturer** of commercial trucks, buses and defense vehicles
- **\$11 billion** in revenue in 2019
- **Year-over-year market share increase** in the last five years
- **Customer-centric** DNA and industry-leading focus on customer uptime
- **Global alliance with TRATON Group** speeding technology innovation and further cost improvement



Renewed and expanded entire vehicle portfolio in last three years



On Highway



Severe Service



Medium



Bus



About Global Product Development

- Global Product Development (GPD) is an engineering organization within Navistar
- Controls & Software (C&S) is a GPD group responsible for designing and implementing engine and vehicle electrical and electronics components
 - engine and vehicle control applications
 - embedded software
 - electronic component integration
- Model Based Design (MBD) within Controls & Software
 - MBD used extensively for engine and vehicle control applications
 - Algorithm development
 - Production intent software code generation

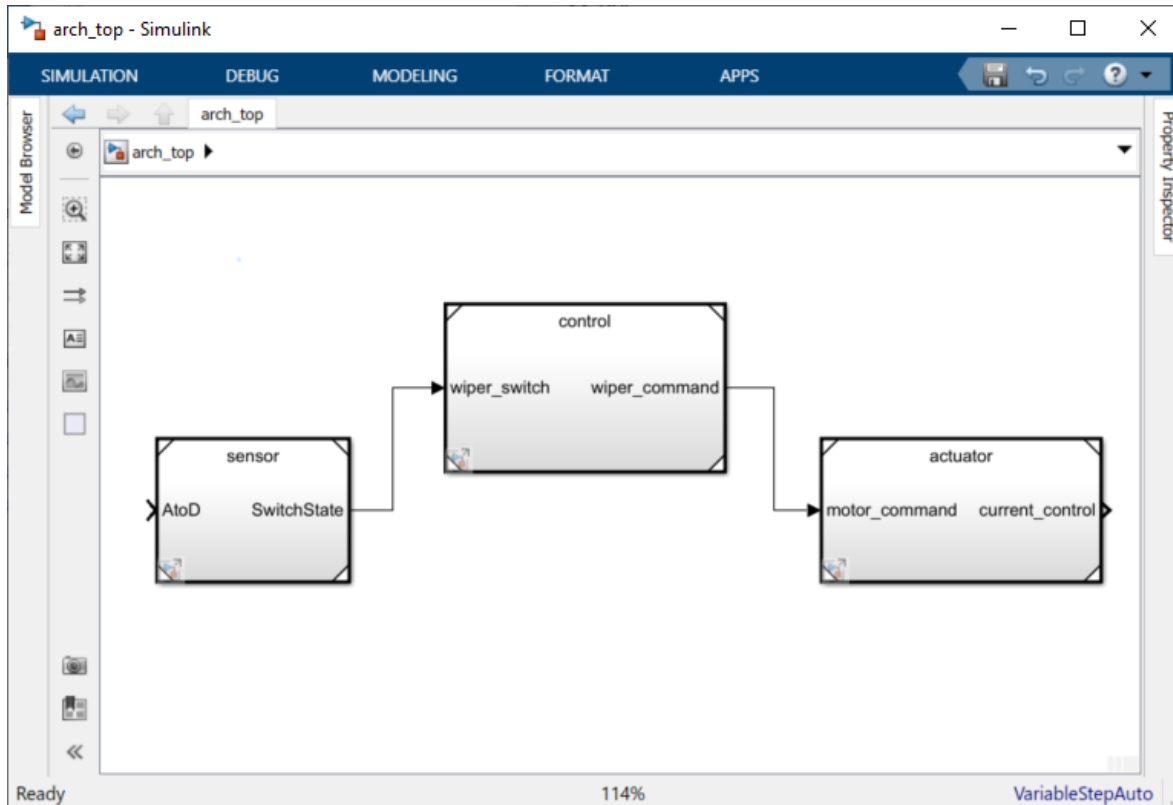
Agenda

- About
- **Background and context**
 - Model Based Design (MBD) and Components
 - Data objects
- Our organization's workflow and motivation
- Solution
- Status
- Summary

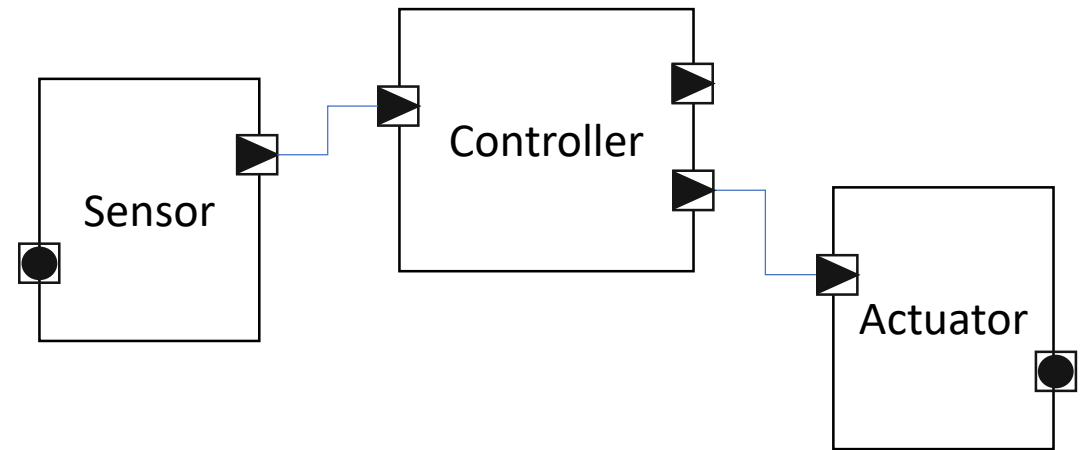
Component Based Development and Architecture

- Create components and make connections

Simulink top model with Reference Models



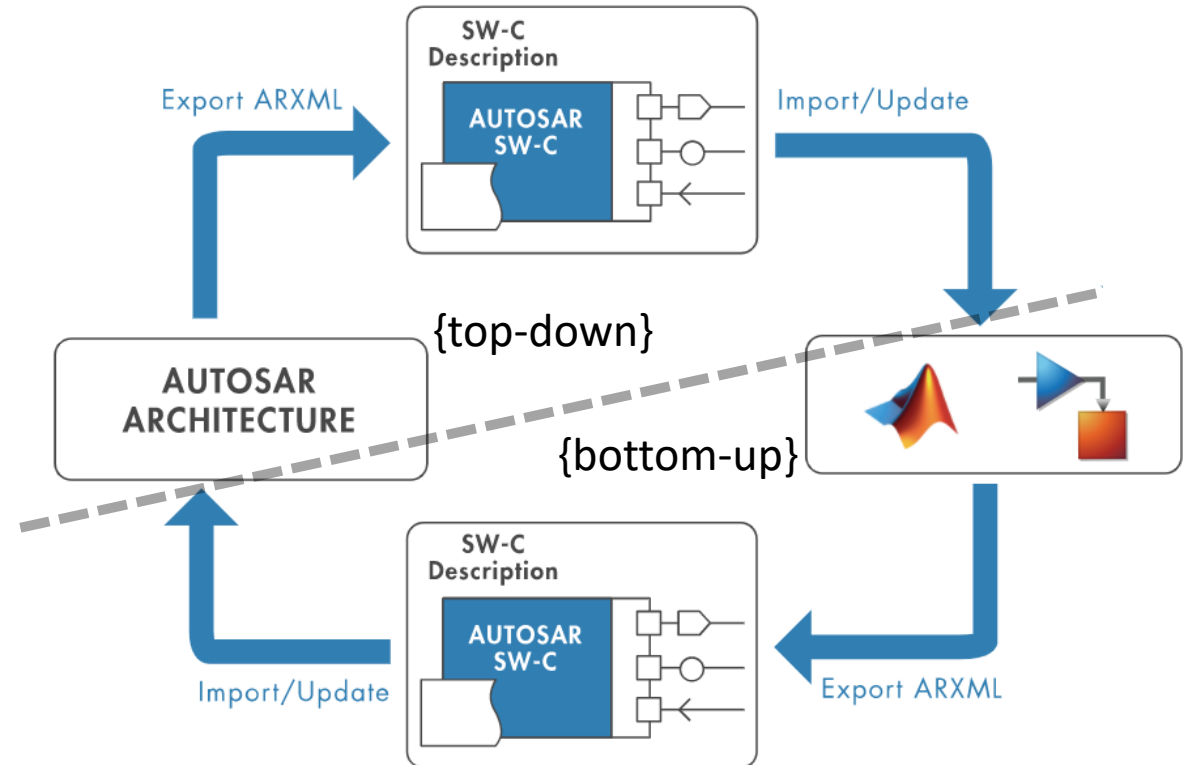
or AUTOSAR Composition with SWCs



Top-down vs. Bottom-Up workflows

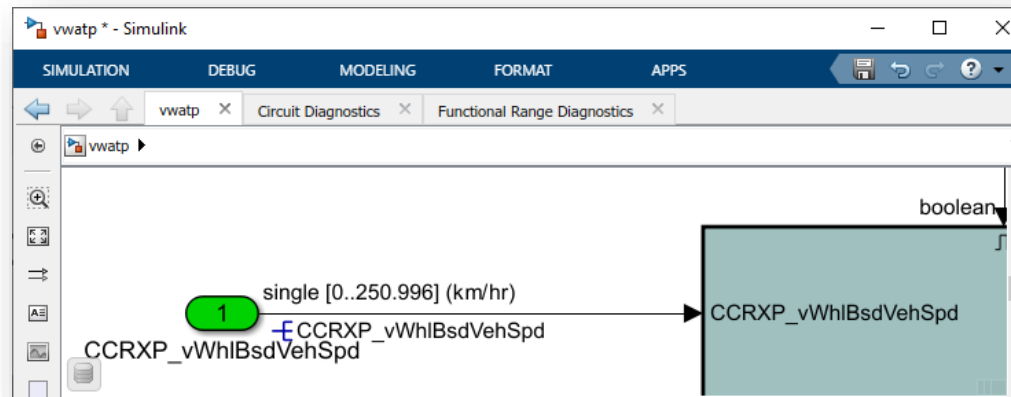
- Top-down
 - Authoring defines components
 - Authoring makes connections
 - Create ARXML
 - Create skeleton models
- Bottom-up
 - Create models independently
 - Generate ARXML
 - Authoring integrates components
 - Authoring connects existing inputs/outputs

Iterating between Simulink model and AUTOSAR Architecture.



Data Management for MBD

- Data objects elaborate graphical designs
 - Define interfaces
 - Describe behavior

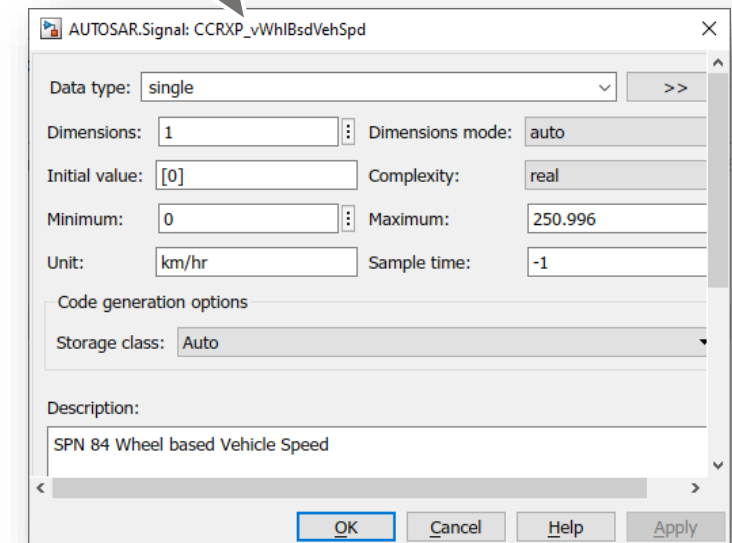


- Control deployment/code generation

```
495 * Inport: '<Root>/CCRXP_flgWhlBsdVehSpdCANErr'  
496 * Inport: '<Root>/CCRXP_vWhlBsdVehSpd'  
497 * MultiPortSwitch: '<S17>/Multiport_Switch1'  
498 * Sum: '<S47>/Add'  
499 */  
500 if (!Rte_IRead_vwatp_Step_CCRXP_flgWhlBsdVehSpdCANErr_CCRXP_flgWhlBsdVehSpd) {  
501     () {  
502     temp_MultiportSwitch_p = ((uint8)VWATP_nrZeroUint8_SC);  
503     temp_MultiportSwitch1_b =  
504     Rte_IRead_vwatp_Step_CCRXP_vWhlBsdVehSpd_CCRXP_vWhlBsdVehSpd();  
505     } else {
```

Where are these Simulink data objects stored?

- workspace (e.g. from .m or .mat files)
- Simulink Data Dictionary

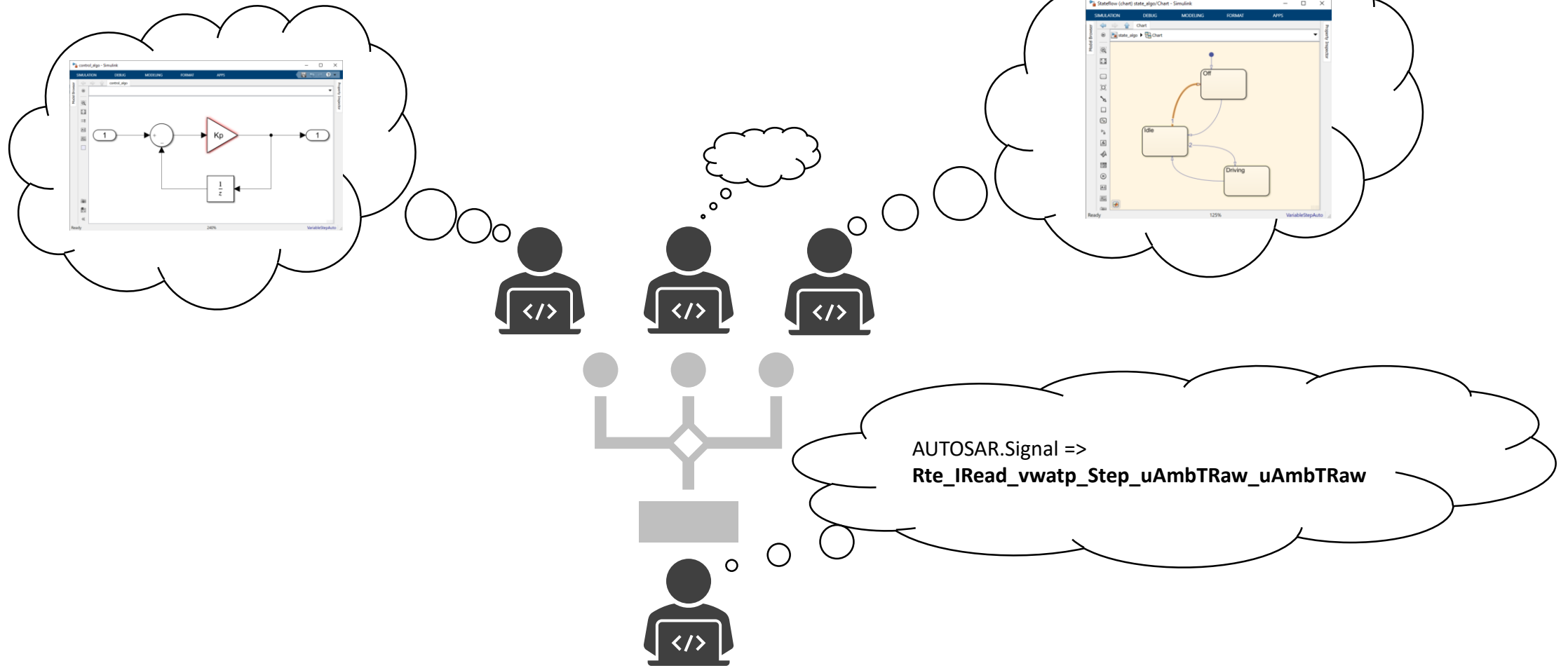


Agenda

- About
- Background and context
- **Our organization's workflow and motivation**
 - Functional/algorithm focus vs software implementation
 - Agile architecture development
 - Production code generation
- Solution
- Status
- Summary

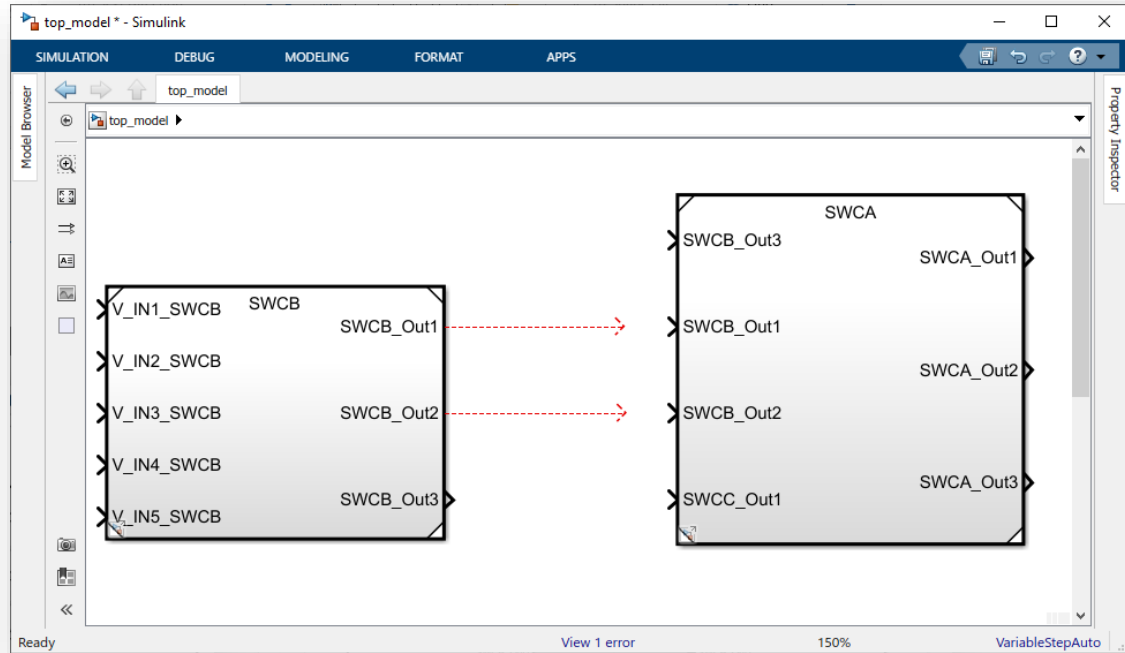
Functional focus

- Functional focus vs. software implementation

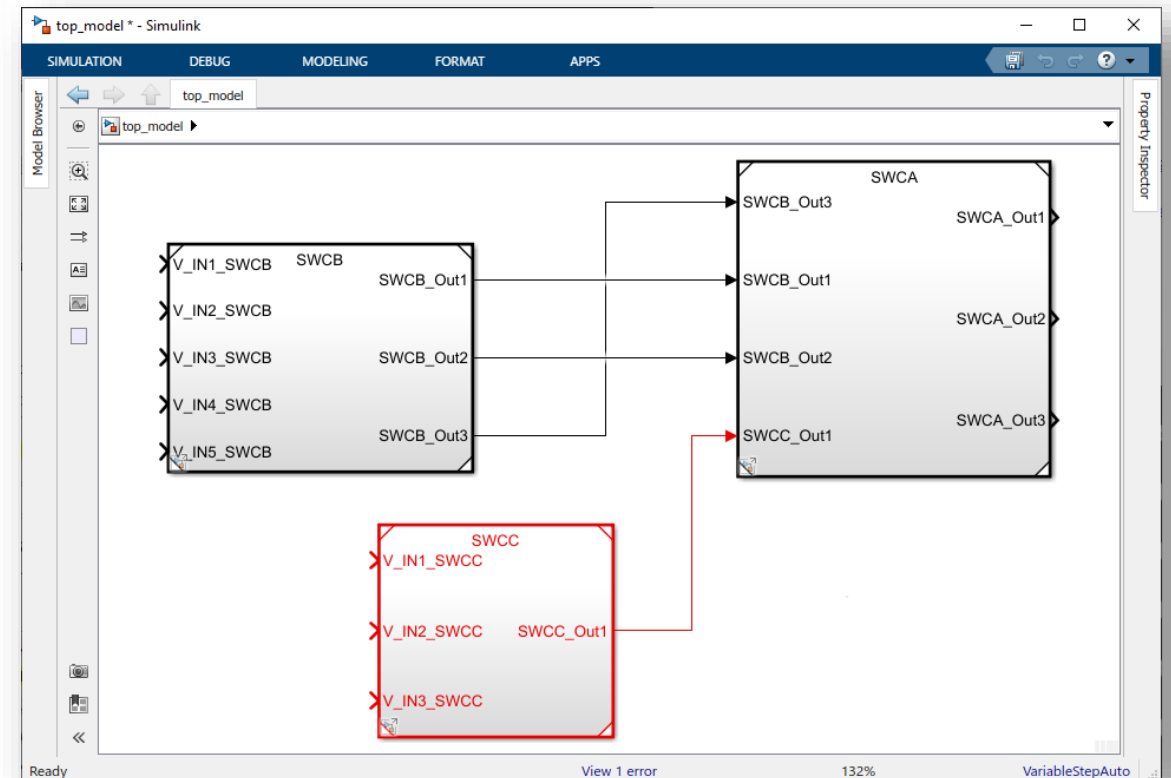


Organic architecture using bottom-up

- Can we facilitate “implicit authoring”?
Ensure I/O Signal names exactly match



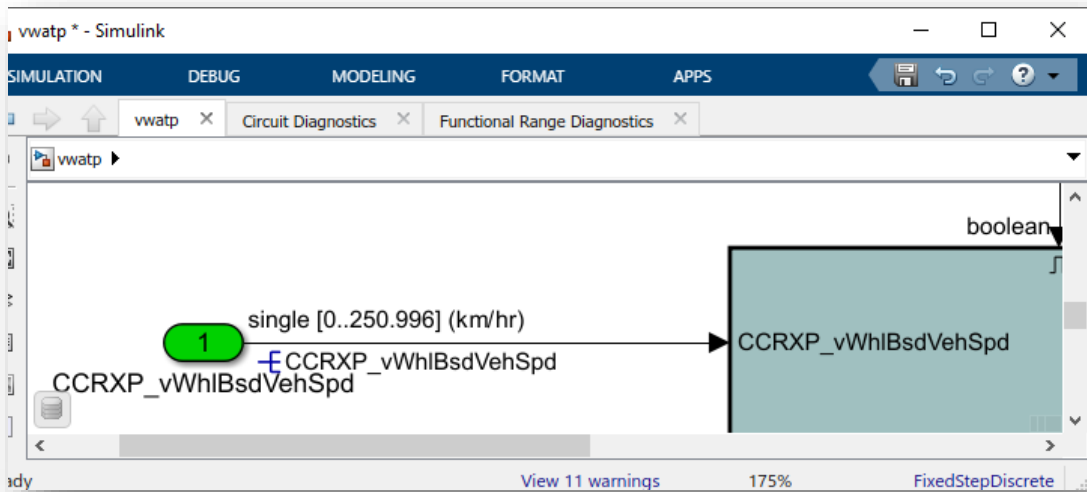
- Can we grow the system organically?
Add components as appropriate



Streamlined production code generation

- Can we go from functionally focused MBD to production code easily?

How easily can we go from here...



to here?

```
495 * Inport: '<Root>/CCRXP_flgWhlBsdVehSpdCANErr'  
496 * Inport: '<Root>/CCRXP_vWhlBsdVehSpd'  
497 * MultiPortSwitch: '<S17>/Multiport_Switch1'  
498 * Sum: '<S47>/Add'  
499 */  
500 if (!Rte_IRead_vwatp_Step_CCRXP_flgWhlBsdVehSpdCANErr_CCRXP_flgWhlBsdVehSpd  
501     ()) {  
502     temp_MultiportSwitch_p = ((uint8)VWATP_nrZeroUint8_SC);  
503     temp_MultiportSwitch1_b =  
504     Rte_IRead_vwatp_Step_CCRXP_vWhlBsdVehSpd_CCRXP_vWhlBsdVehSpd();  
505 } else {
```

Agenda

- About
- Background and context
- Our organization's workflow and motivation
- **Solution**
- Status
- Summary

Simulink Data Dictionary advantages

- Design data management
 - Create and manage data object definitions
 - Specify design data using Model Explorer interface
- Persistent repository
 - Repeated data loading not required
 - Automatically associate design data with model

Simulink Data Dictionary gaps

- Functional focused support (abstraction)
 - Uses data object class not abstract type
 - Includes code generation specifics
- Organic architecture (parallel development and implicit authoring)
 - Doesn't match input/output names explicitly
 - Bottom-up approach possible but integration complicated

Navistar Enhanced Data Dictionary

- Goals
 - Enhance and add functionality
 - User friendly (particularly function developers)
 - Support Controls & Software organization's workflow
- Approach
 - Simulink Data Dictionary as "database"
 - MATLAB Engine API for Java



Data abstraction

- Functional view
 - Inputs
 - Outputs
 - Calibrations
 - etc.

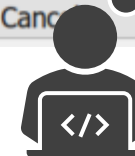
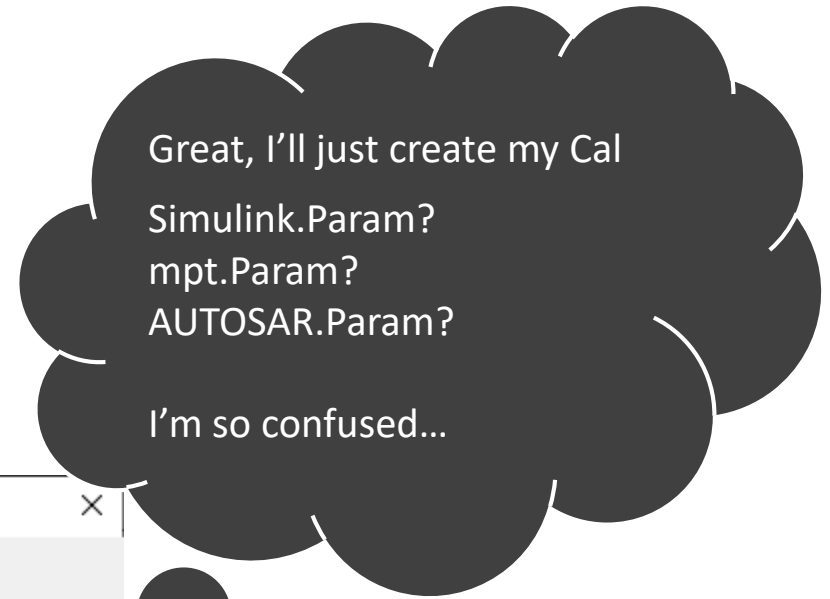
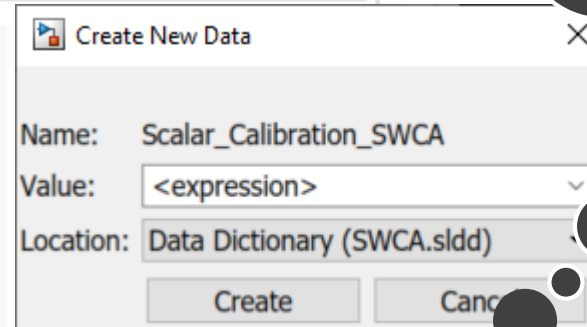
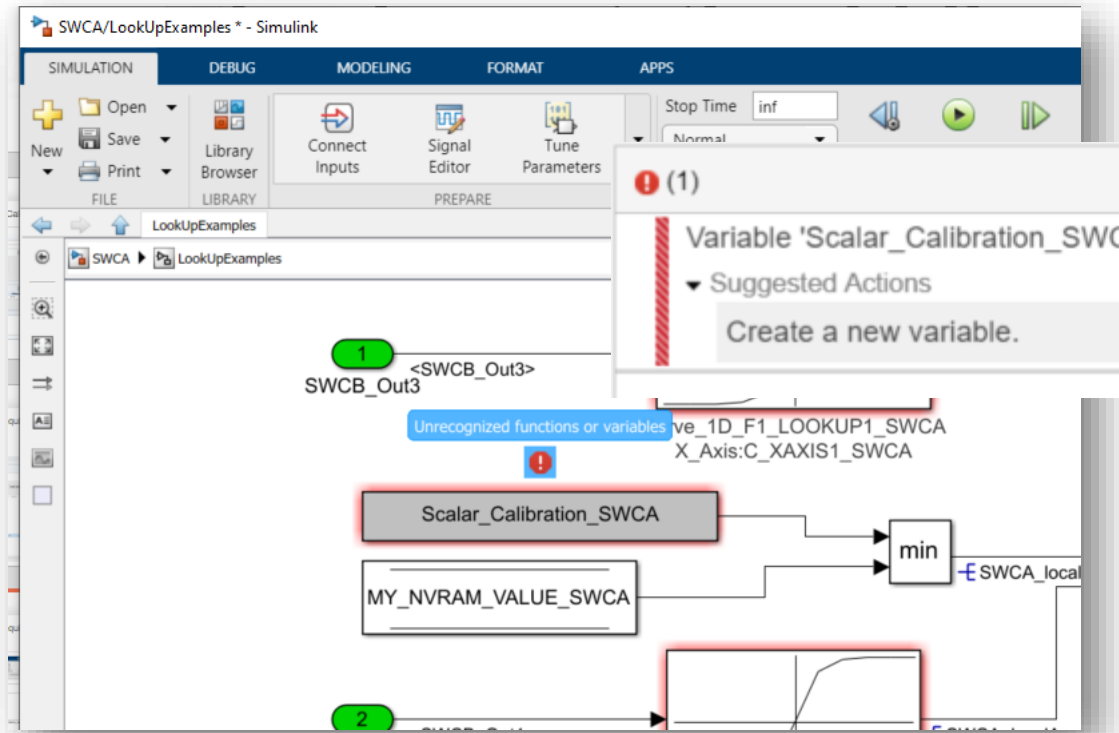
The screenshot displays a software interface with two main panels. The left panel shows a functional block diagram for a component named 'SWCA'. It includes several input and output ports, such as 'SWCB_Out3', 'SWCA_Out1', 'SWCB_Out1', 'SWCC_Out1', and 'SWCB_Out2'. These ports are connected to various internal blocks, including 'Scalar_Calibration_SWCA', 'min', 'x', and 'Map_2D_F2_LOOKUP_SWCA'. The right panel is a 'Data Dictionary' window, which lists data objects in a table format. The table has columns for Name, Description, Base Type, Offset, Slope, Min, Max, Unit, and Initial Value. The first table lists four data objects: SWCB_Out1 (single), SWCB_Out2 (int32), SWCB_Out3 (single), and SWCC_Out1 (single). The second table lists one data object: Scalar_Calibration_SWCA (single) with a value of [0] and a unit of km/hr. Annotations include dashed green and blue lines connecting elements between the functional diagram and the data dictionary, and dashed grey circles highlighting specific elements.

Name	Description	Base Type	Offset	Slope	Min	Max	Unit	Initial Value	Dimensions
1 SWCB_Out1		single	0	1					1
2 SWCB_Out2		int32	0	0.1					1
3 SWCB_Out3		single	0	1					1
4 SWCC_Out1		single	0	1					1

Name	Value	Description	Base Type	Offset	Slope	Min	Max	Unit	Dimensions
1 Scalar_Calibration_SWCA	[0]		single	0	1	0	1	km/hr	[1,1]

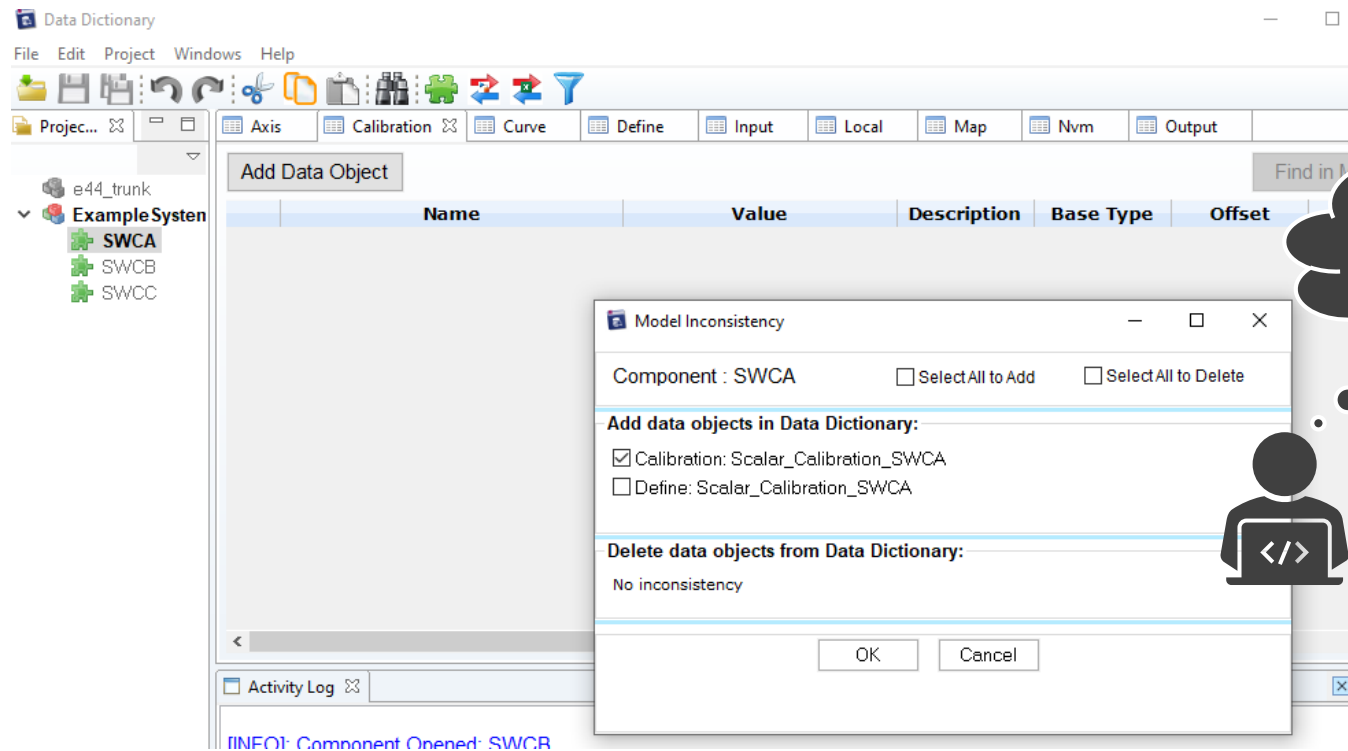
Implementation hiding

- Simulink guides data management (helpful)
- Simulink requires implementation specific knowledge (inconvenient)
- Algorithm developer vs. software engineer



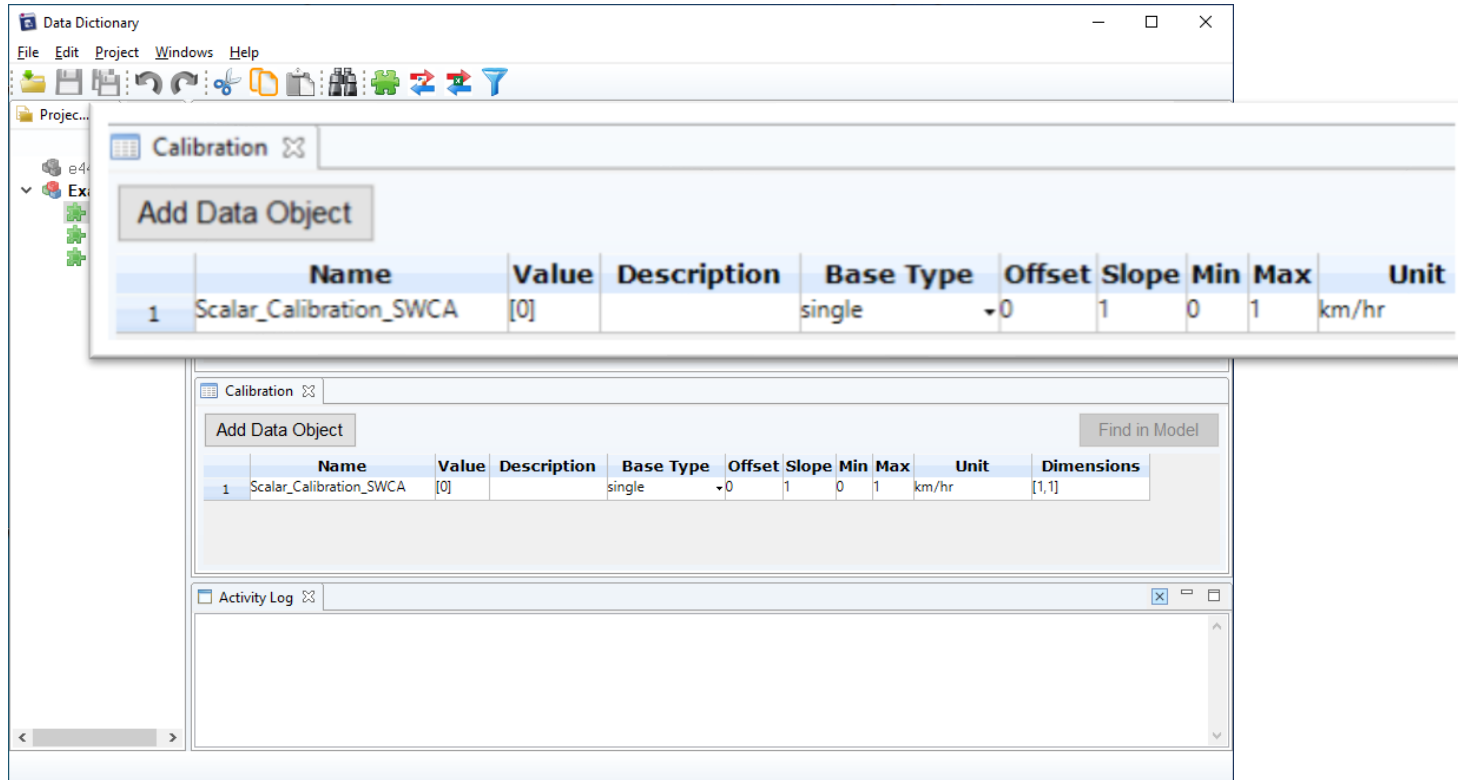
Implementation hiding concept

- User enters functional attributes
 - Min, Max, Units, etc.
- Hide and automate deployment requirements
 - Object Class
 - CoderInfo
 - etc.



Implementation hiding details

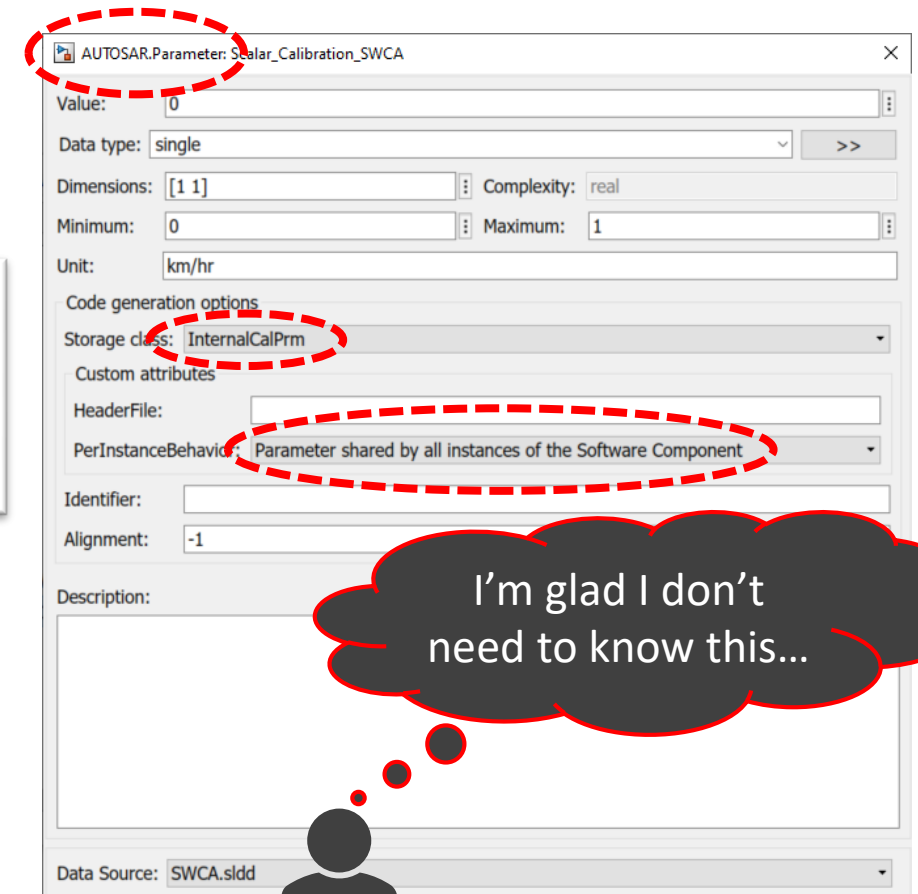
- Hide and automate deployment requirements
- Directly support production code generation



The screenshot shows the 'Data Dictionary' application window. The main area displays a table with the following data:

	Name	Value	Description	Base Type	Offset	Slope	Min	Max	Unit
1	Scalar_Calibration_SWCA	[0]		single	-0	1	0	1	km/hr

Below the table, there is a 'Find in Model' button and an 'Activity Log' section.



The screenshot shows the configuration dialog for the parameter 'AUTOSAR.Parameter: Scalar_Calibration_SWCA'. The dialog includes the following fields and options:

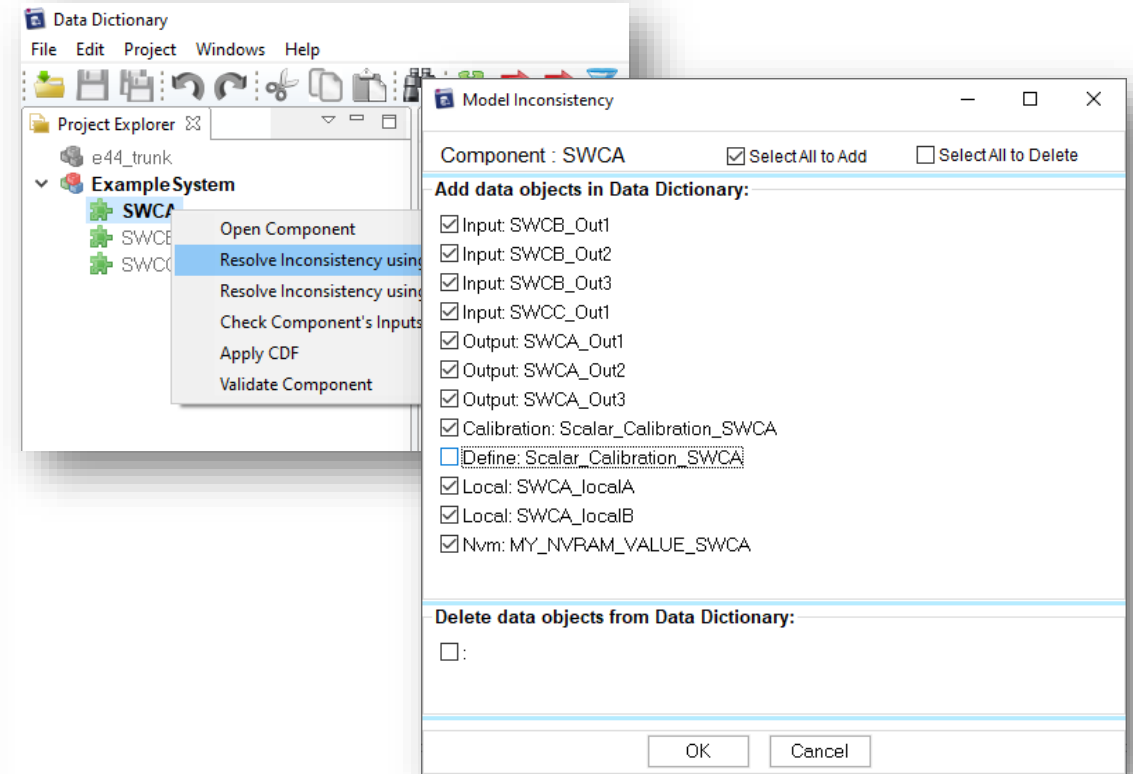
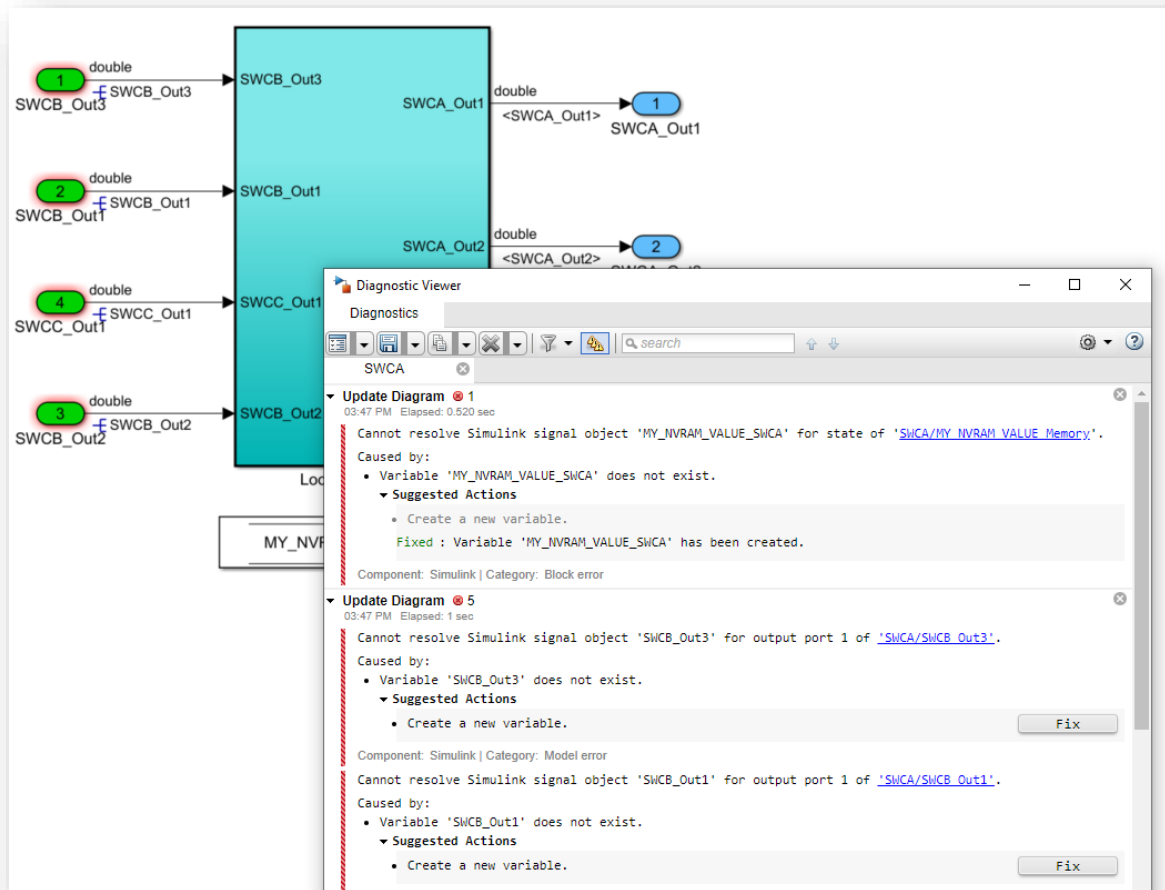
- Value: 0
- Data type: single
- Dimensions: [1 1]
- Complexity: real
- Minimum: 0
- Maximum: 1
- Unit: km/hr
- Code generation options: Storage class: InternalCalPrm
- Custom attributes: HeaderFile: (empty), PerInstanceBehavior: Parameter shared by all instances of the Software Component
- Identifier: (empty)
- Alignment: -1
- Description: (empty)
- Data Source: SWCA.sldd

I'm glad I don't need to know this...



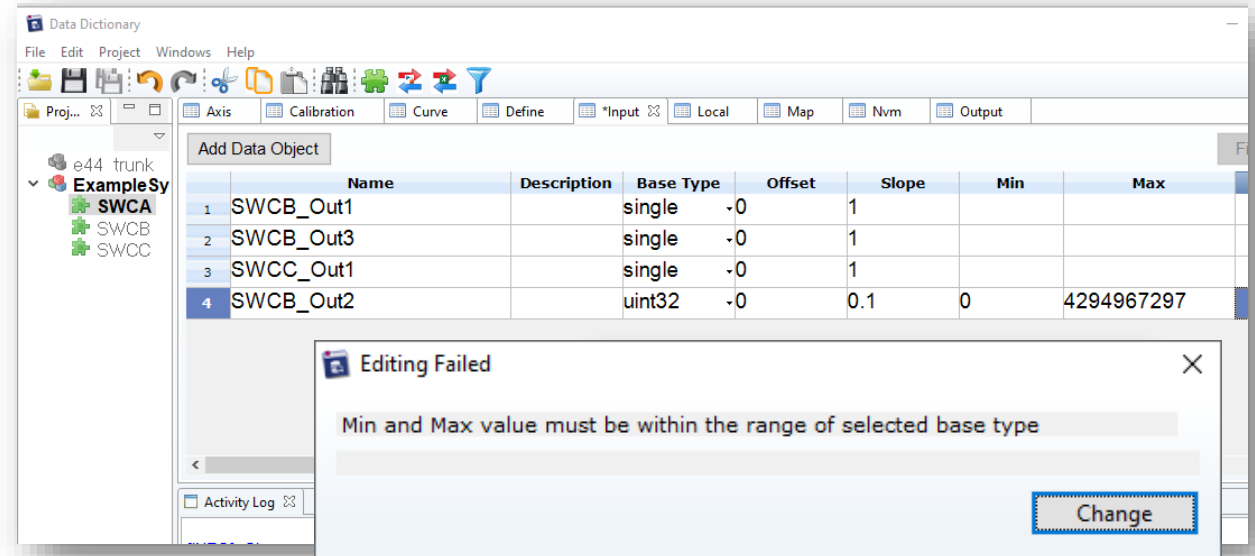
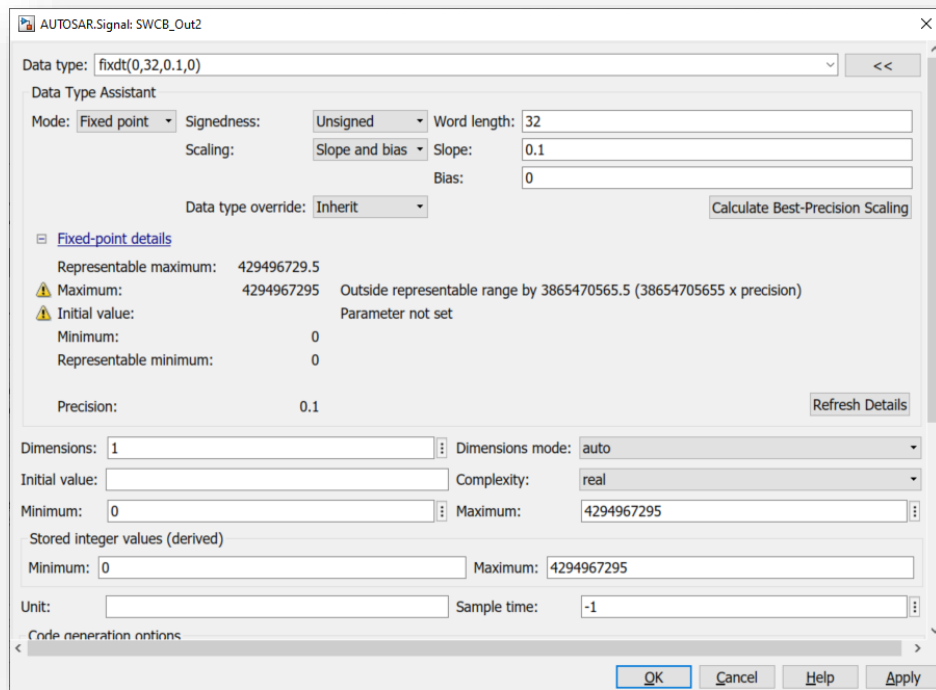
Functional developer assistance

- Consolidated model synchronization
All missing/extra data objects vs Simulink 1 type at a time



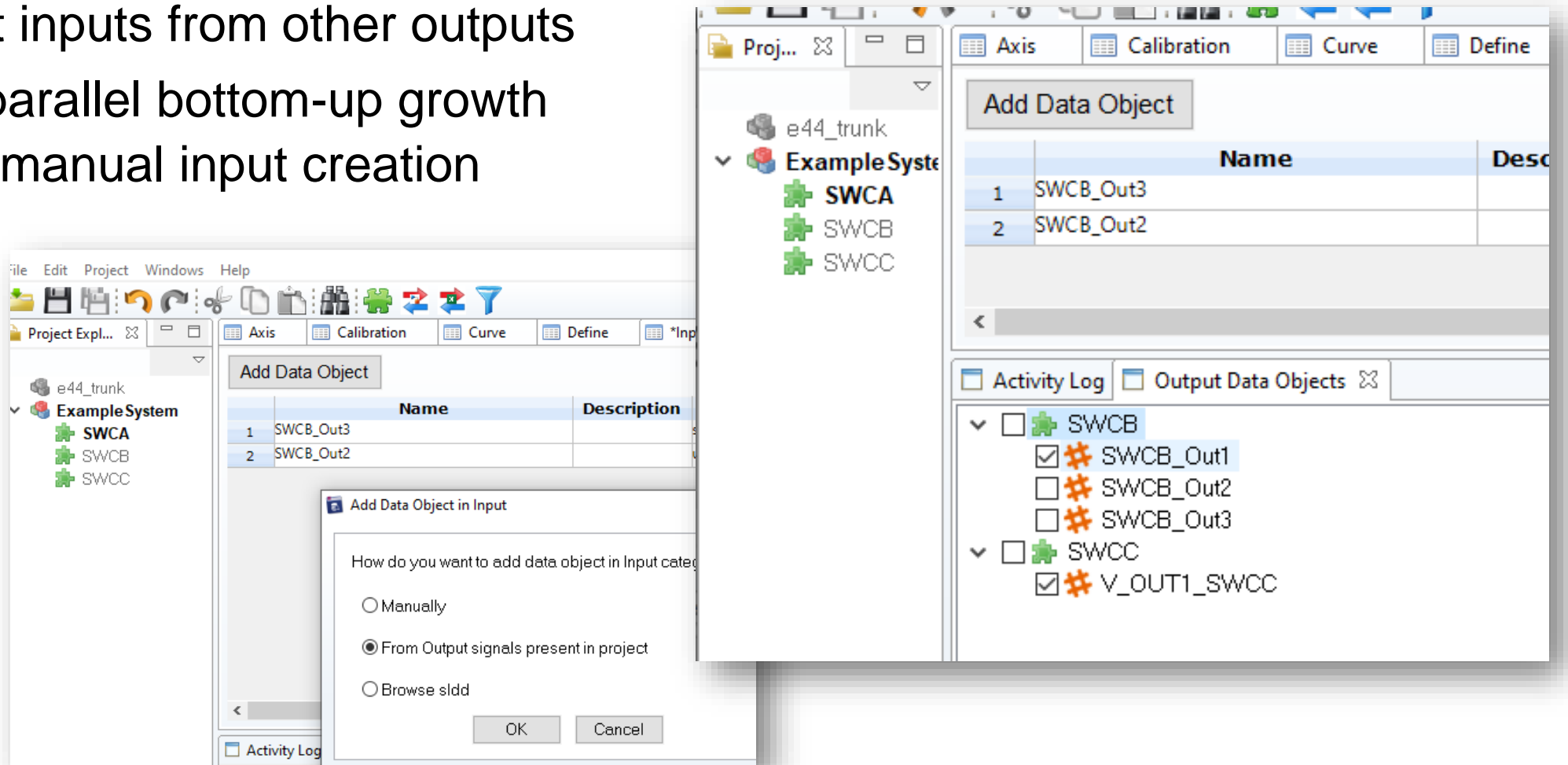
Functional developer assistance

- Typical use case automation
 - Automatically select 'single' datatype on creation (project preference)
 - Fixed-point datatypes automatically defined from slope/offset
 - Boolean datatype automatically sets Min/Max
 - Min/Max checked on data entry



Enable organic & parallel architecture development

- Enable and enforce compatibility
 - Select inputs from other outputs
- Support parallel bottom-up growth
 - Allow manual input creation



Implementation considerations

- Duplicate I/O data objects
 - streamlined/independent component development
 - requires integration reconciliation after component implementation
 - supports “dangling inputs” for resolution later
- Simulink Data Dictionary references
 - complete project and coordination required
 - component integration completed at implementation
 - “dangling inputs” require immediate external component modification

Current status and beyond

- Initial Simulink Data Dictionary based tool launched Q4-2019
- Initial improvements and support for R2020a added Q2-2020
- Ongoing user feedback improvements in progress

- Currently in use on a production intent project
 - Already supported several vehicle intent software releases

- ToDo
 - Incorporate usability feedback
 - Can we take advantage of more built-in Simulink capabilities?
 - Embedded Coder Dictionary
 - Code Mappings Editor

Summary

- Navistar's Controls & Software group uses Model Based Development (MBD)
 - to facilitate embedded application development by 100's of engineers
 - on 4 projects and growing
 - including 2 projects in production
- Navistar's Enhanced Data Dictionary
 - Supports Controls & Software's functional focused organization
 - Provides robust and streamlined component based workflow
 - Supports parallel component development
 - Eases production code generation

THANK YOU



DEFENSE

TRUCKS

BUSES