AUTOMATION IN MODEL SHARING AREA FOR ENGINE ECU PROJECTS

BOSCH

Model Sharing Automation
Problem Statement

- Model based development approach is widely used across Automotive OEMs and suppliers to exchange software modules. In Bosch we receive SIMULINK models from OEM and need to generate code and validate software against the OEM models.
- Challenges during Model Sharing
  - Adaptation of the Models for Code generation
  - Validation of the generated code against the OEM Model

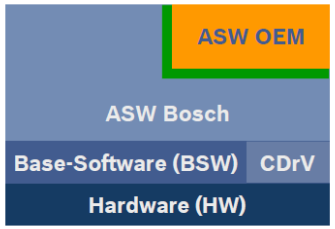
- Our Automation tool chain based on MATLAB and Jenkins will address above problems to perform the code generation and validation in an efficient way

Model Sharing Automation

SW Sharing: Code sharing and Model sharing

➢ “Software Sharing” is the notion that is typically being used to describe several business models by which OEM’s, automotive suppliers or software engineering companies request or allow the use of software modules. (Source: C/IPL)

Base Model – Code sharing

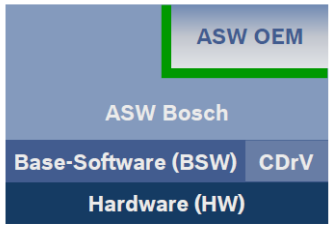


Characteristics

SW sharing – base model

- OEM delivers SW in a RB system
- Integration done by RB
- Typically based on RB architecture

CommDev – Model Sharing

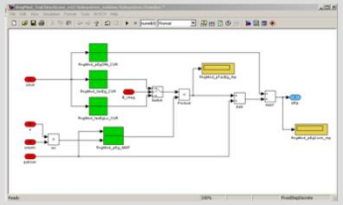


Characteristics

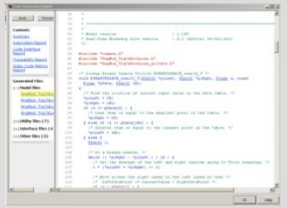
SW sharing – CommDev

- OEM delivers Spec/Models
- Functional responsibility at OEM side
- Adapt different name spaces by CIL

The **OEM** focuses on **Physical Model** and Simulation
A model is delivered



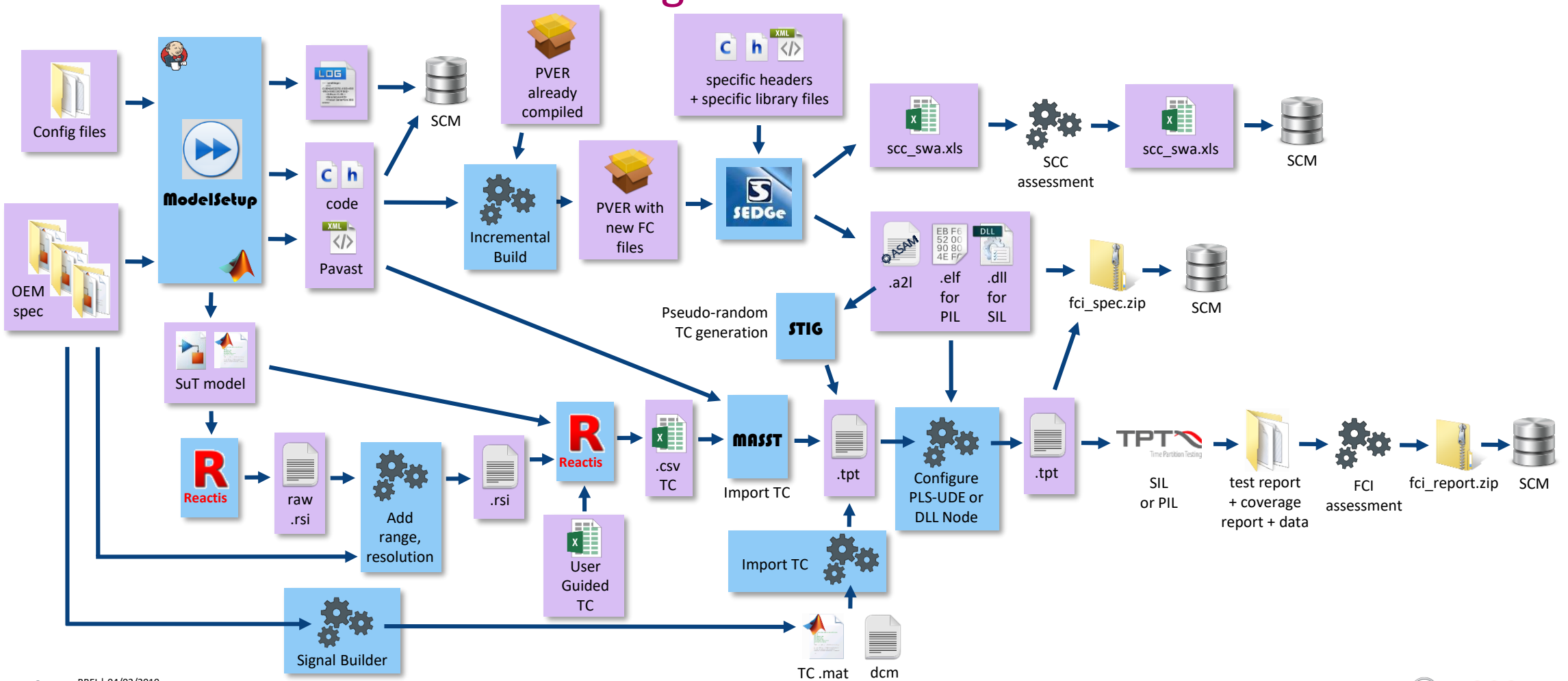
The **supplier** is in charge of the **Industrialisation** of the generated code based on the model



OEM Motivation

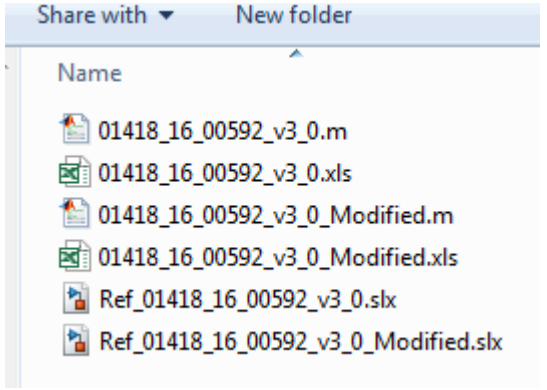
Model Sharing Automation

Workflow: Code and Test case generation

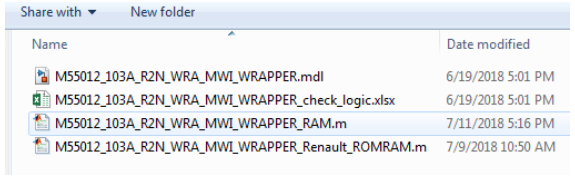


Model Sharing Automation
OEM Inputs

- Simulink Model (*.mdl or *.slx)
- Calibration file (*.m) – Calibration information
- Data Dictionary (*.xls/*.xlsx or *.xml or *.m)
  - Interface Name
  - Data Type
  - Range
  - Resolution
  - Unit
  - Dimension



OEM Inputs

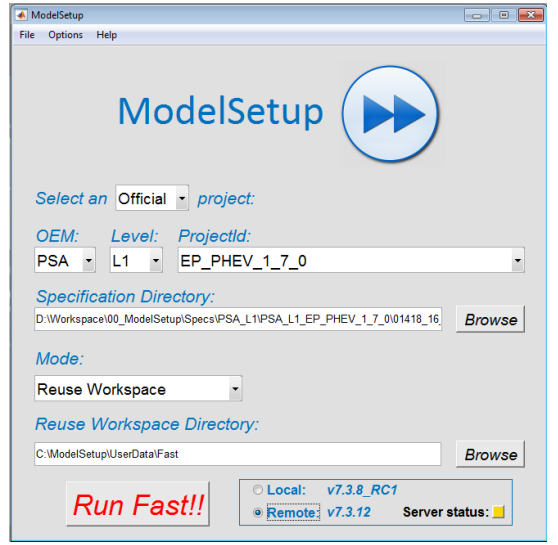


OEM Inputs

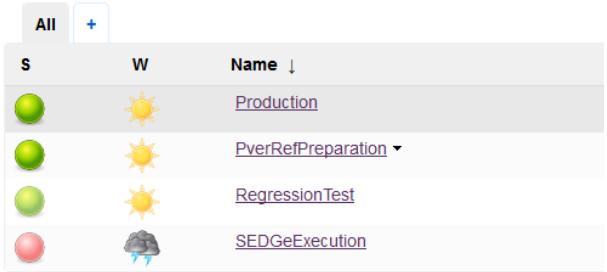
Model Sharing Automation

Model Setup

- Automation tool to generate code and test cases
- Developed using m-Scripts(MATLAB)
- Simulink and Bosch AddOns used in the background
- REACTIS is used to generate Test cases
- Configured to Run on specific Jenkins machines(With MATLAB installation)
- Code generation of multiple specs supported in parallel
- Multiple Jobs can be triggered on Jenkins machine



Model Setup GUI



Model Setup Jobs

Model Sharing Automation

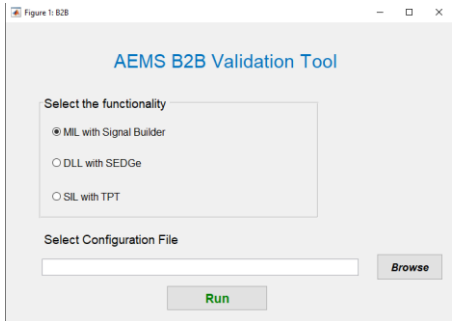
Summary of Model Setup steps

- Creation of Codegen and Test model with proper subsystem structure
- Data Dictionary and Library management(CWO/SLDD, Library replacements)
- Code generation and Post processing(MASST, Add Info to c and Interfaces files)
- Documentation Generation(Pictures, Docu Support, Web view)
- Test Model & Test Case (REACTIS) creation
- Log Creation(html)

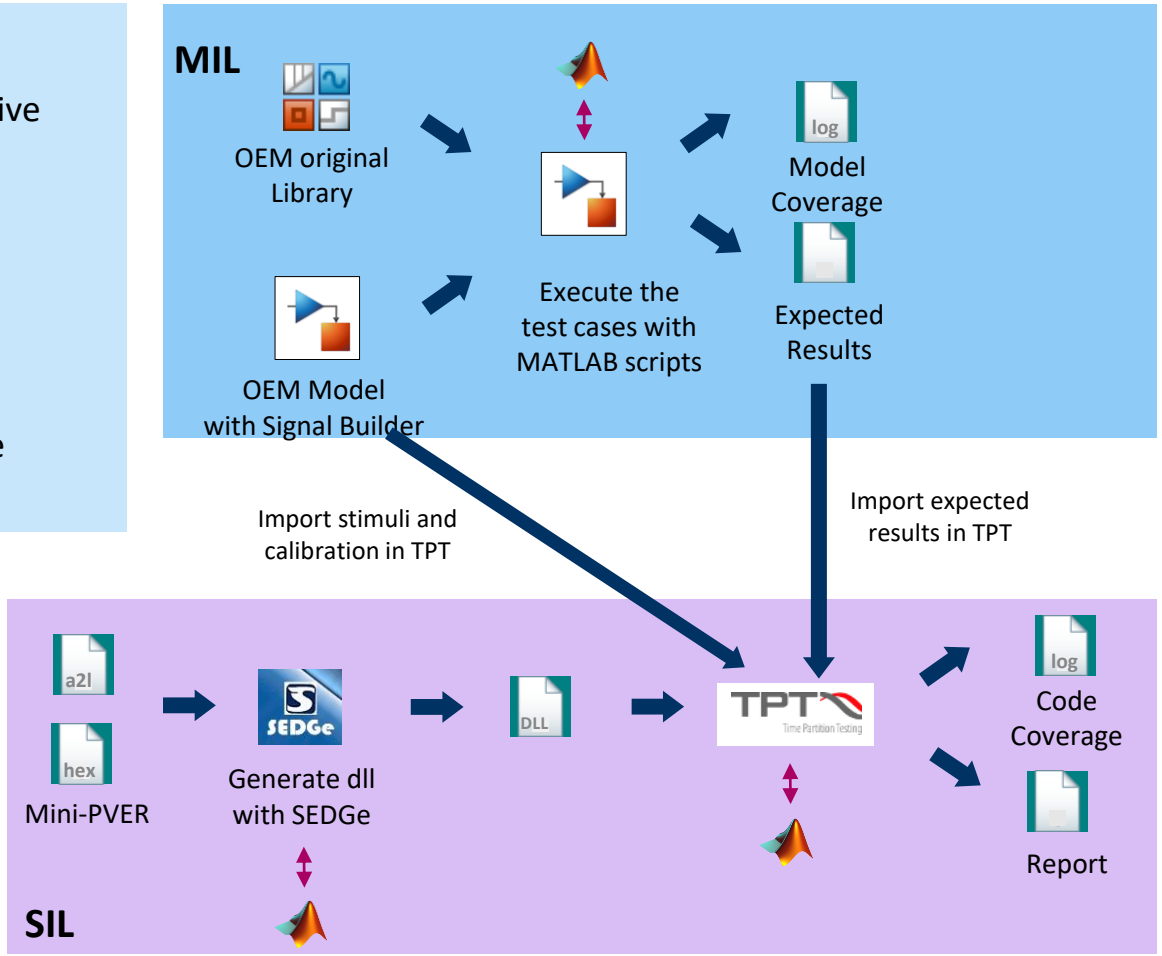
Model Sharing Automation

Back-2-back validation workflow

- Automations achieved
  - Execution of signal builder test cases to generate cumulative model coverage report
  - Export of signal builder test cases and calibration
  - Invocation of DLL generation from MATLAB
  - Import of Test cases and calibrations to TPT
  - Execution of TPT test case to generate report and code coverage

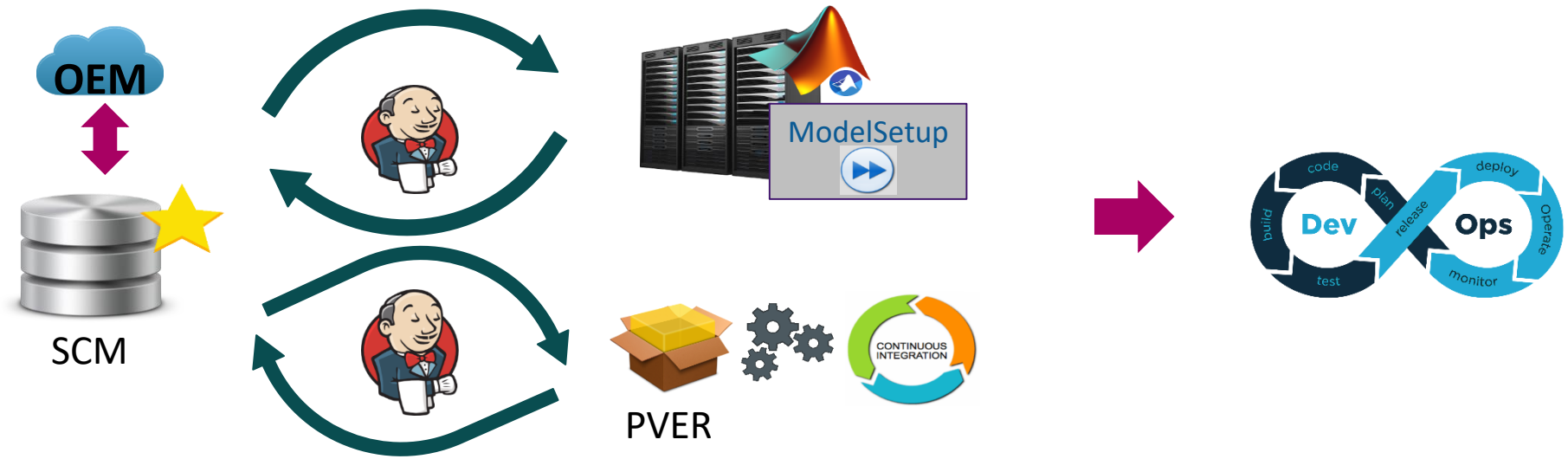


B2B GUI



Model Sharing Automation

Tomorrow's Process



Model Sharing Automation

Demo, Q&A?

Model Sharing Automation
Cloud Words

Guidelines

SCM

Libraries

Implementation

Atlassian Bitbucket

SLDD

MatLab Simulink

git

Specs

Cloud

C.I.

Jenkins

MASST

MatLab Apps

ModelSetup

Tools

Social Coding

Servers

Licences

Post-Processing

AddOn

Automation

BOSCH