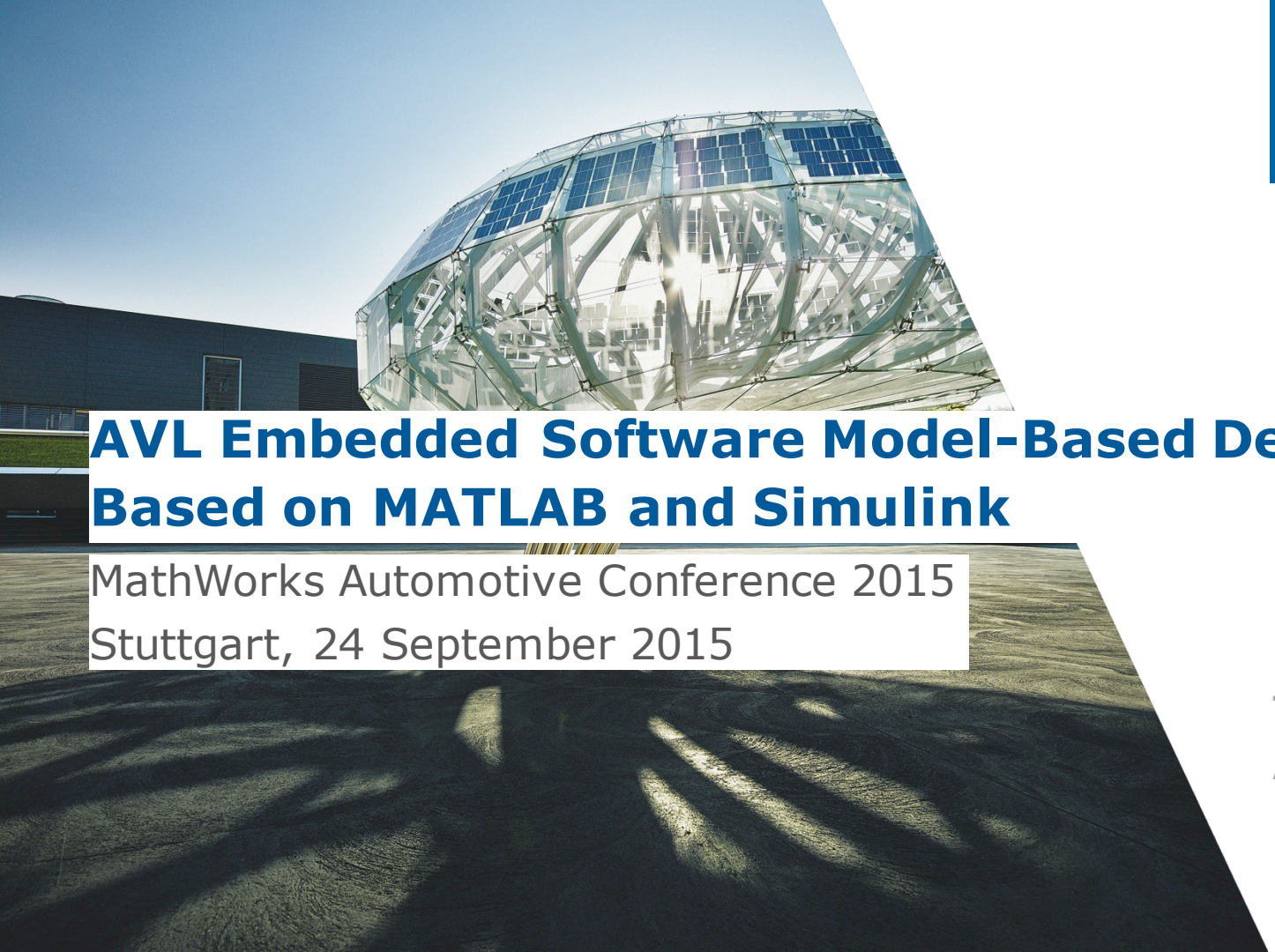


AVL Embedded Software Model-Based Design Platform Based on MATLAB and Simulink

MathWorks Automotive Conference 2015

Stuttgart, 24 September 2015

Thierry Dalon

AVL Software and Functions GmbH

Outline

1. Company use-cases introduction
2. In-house tool platform overview
3. Technical challenges and innovative solutions

Overview AVL Powertrain Controls Business Fields


Passenger Cars


2-Wheelers


Racing


Construction


Agriculture


Commercial Vehicle


Locomotive

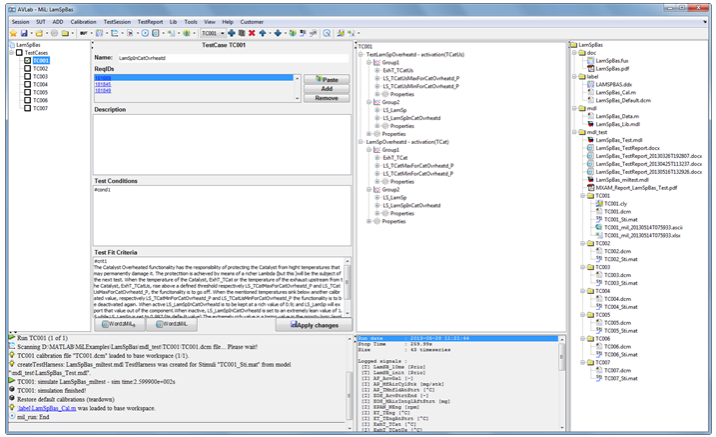

Marine

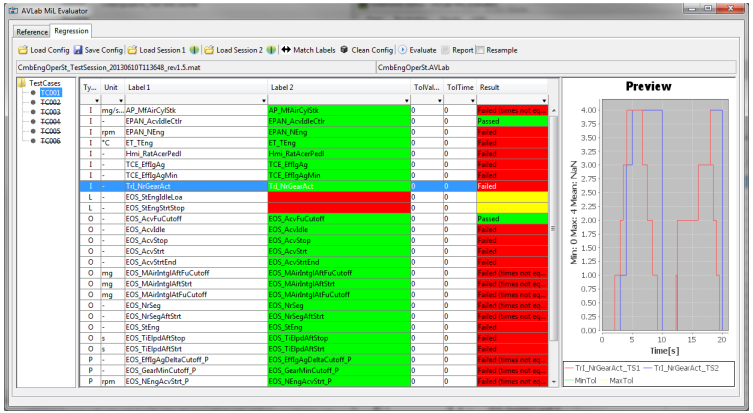

Power Plants

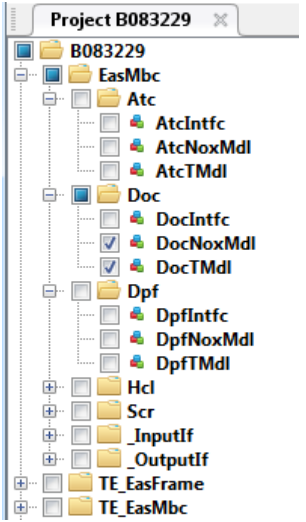
Software and function development

What is “AVLab”?

AVLab (~AVL+MATLAB) describes the tool platform developed at AVL PTE Controls for supporting Model-Based Embedded Software Development in MATLAB and Simulink.

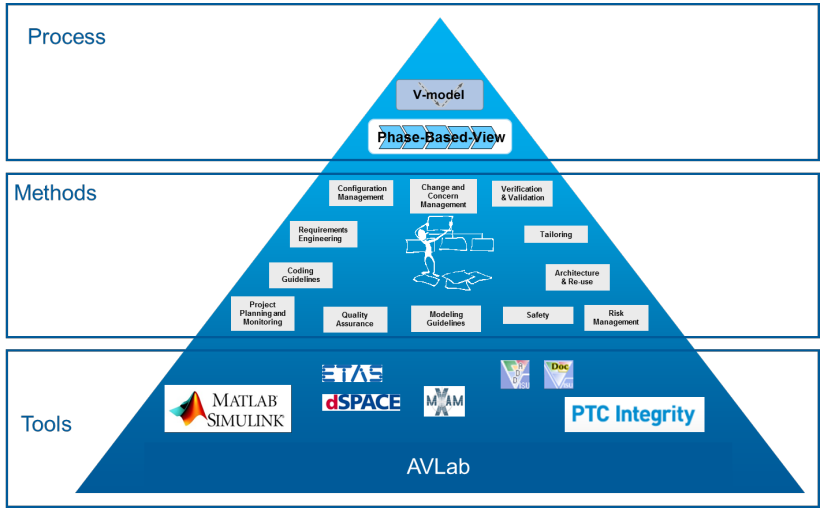




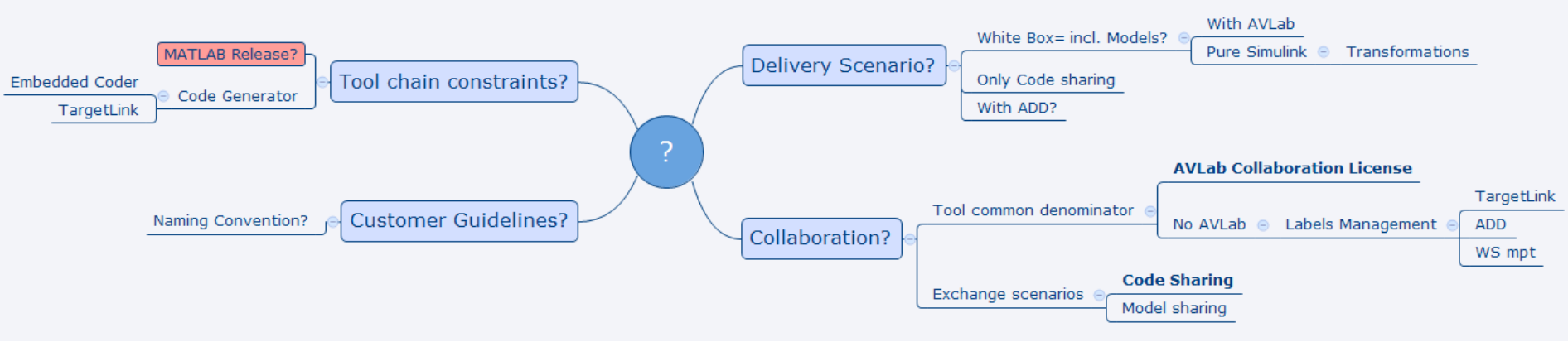


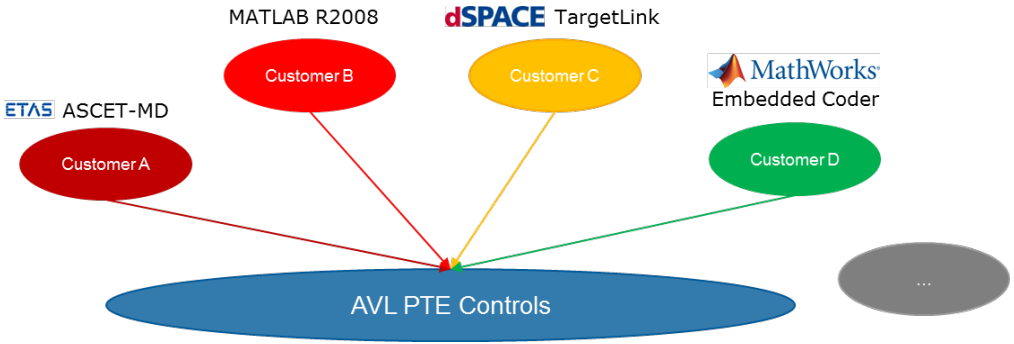
“AVLab” Approach and Philosophy

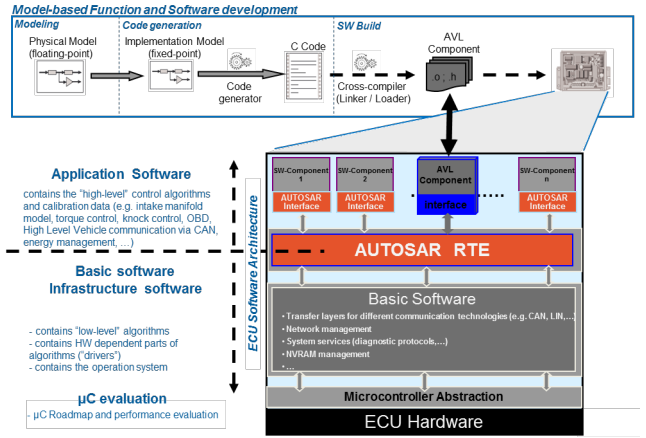
- Do not make, if you can buy state-of-the-art tool covering our needs
- Build toolchain/interfaces between existing tools (glue tools) to have an *integrated seamless toolchain*
- Don’t be dependent but flexible (*open* platform)
- Cover/support all standard tool landscape/customer scenarios with least effort
- Component-based approach
- Support processes/methods
- Standardization
- Proxy for best practices
- Re-use oriented



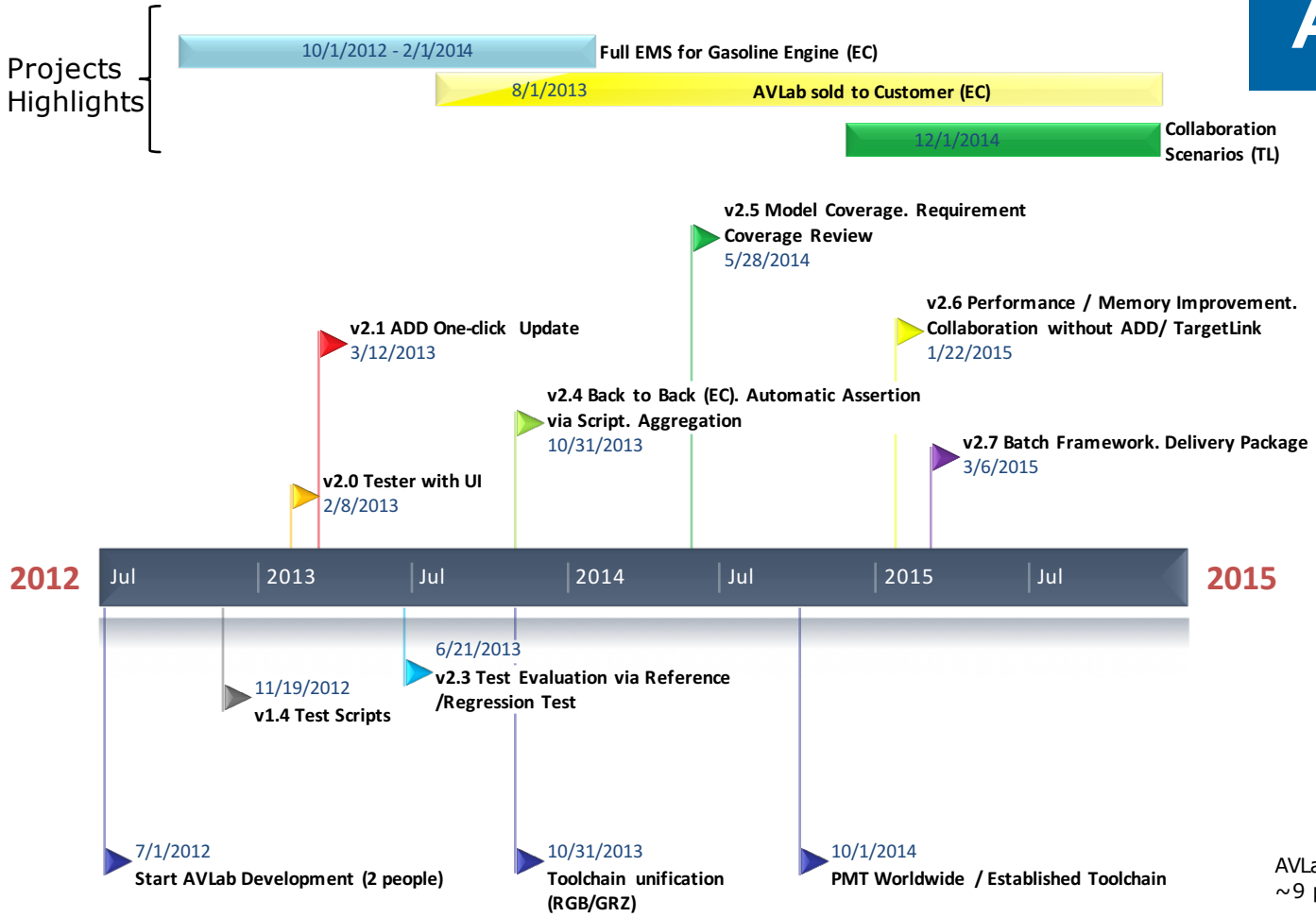
AVL Customer Toolchain Use Cases







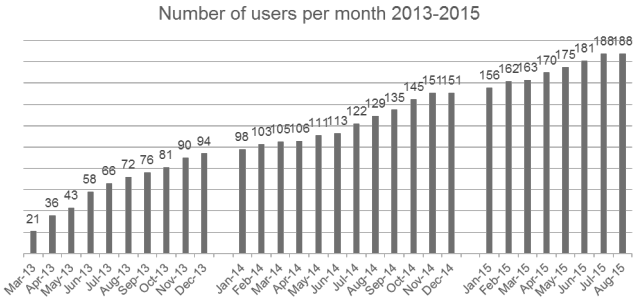
AVLab Timeline

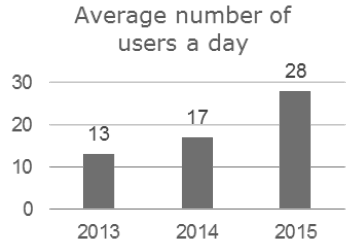


AVLab Team now:
~9 people

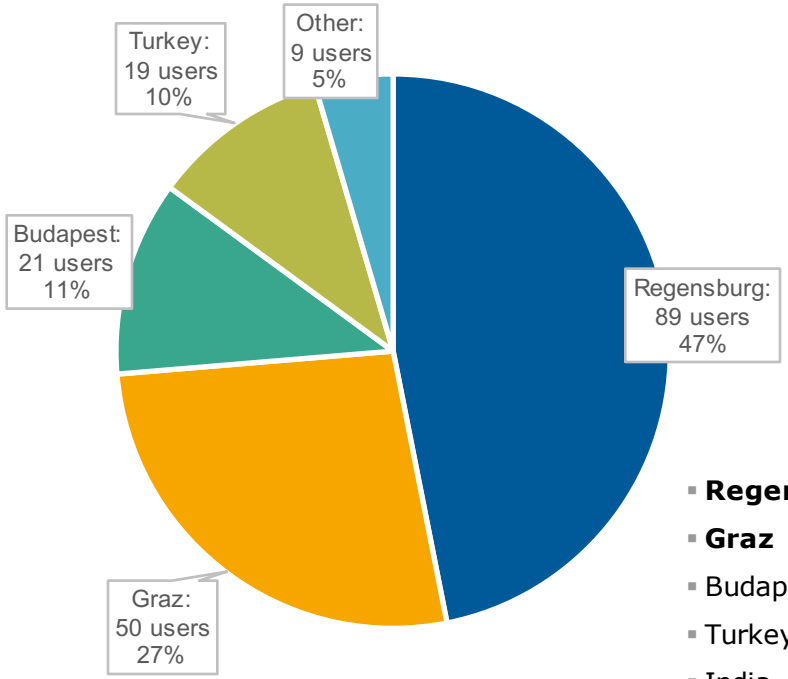
AVLab Users

AVLab has a strong in-house user basis (**188** users).



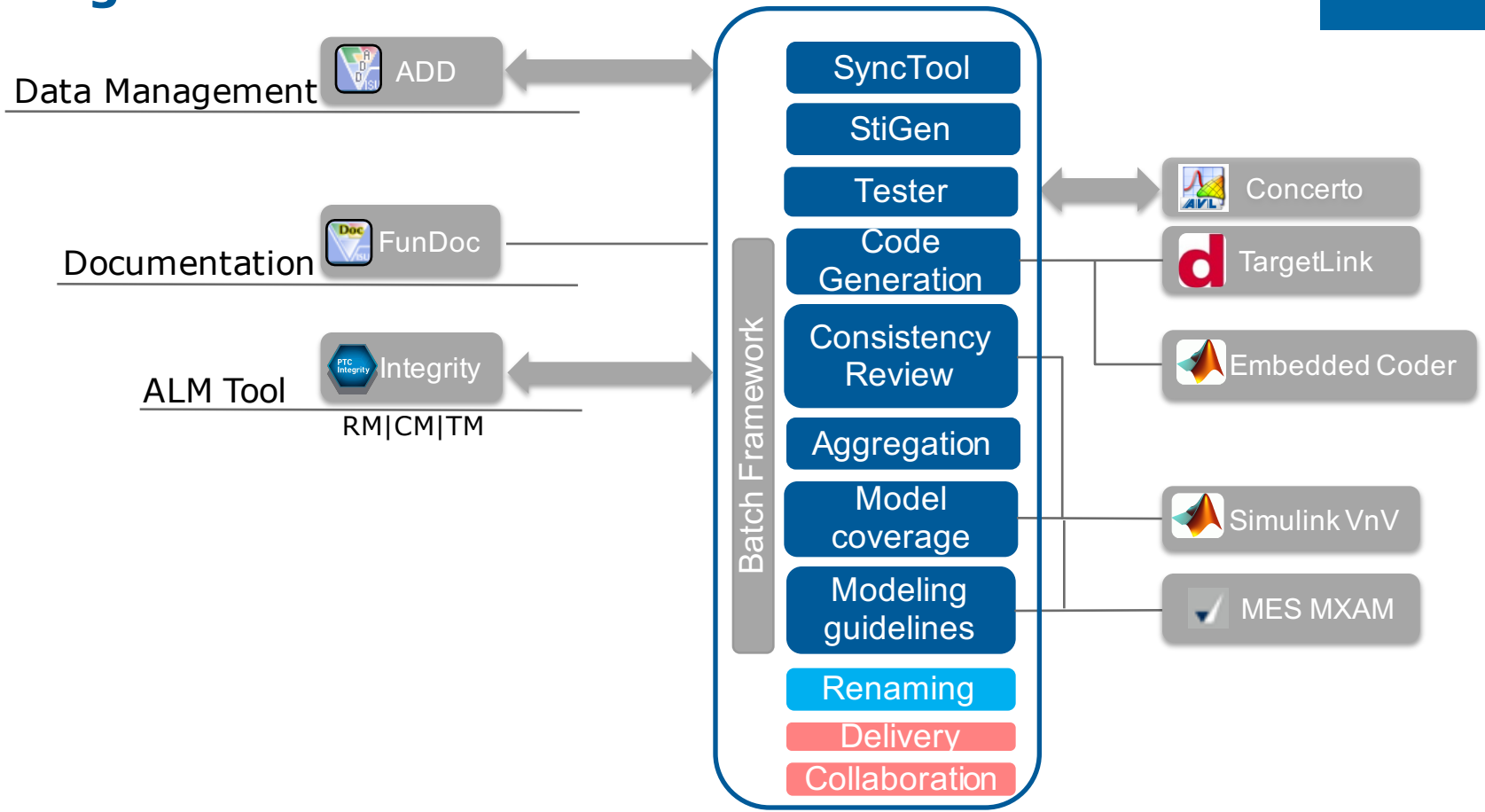


AVLab supports the MBD development process in several AVL PTE affiliates around the world.



- **Regensburg**
- **Graz**
- Budapest
- Turkey
- India
- Sweden
- China
- UK ↗
- US ↗

AVLab Modules and Tool Integrations



AVLab Advantages

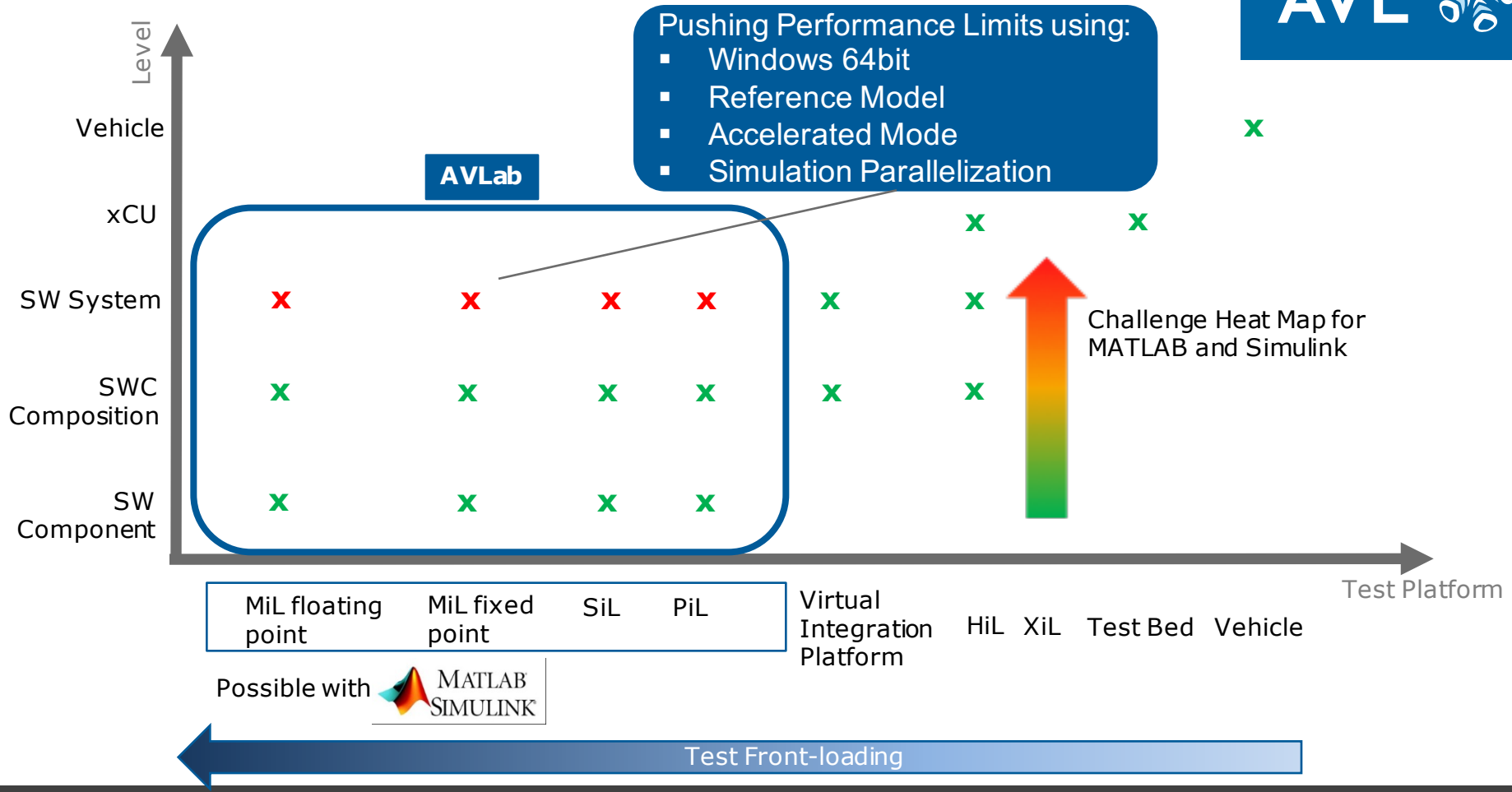
Before AVLab:

- Too many tools, too complex
- Local project specific solutions/scripts (fill missing feature)
- High Cost and time delay because of tools
- Effort to link tools
- Multiplied maintenance and effort by project
- Developer without guidance

With AVLab:

- Shorter development time
- Increased efficiency and productivity (one-click solutions)
- Better quality
- Best practices proxy/levelling up
- Easy re-use
- **One** standard workflow/**one** way (from start to end/continuity)
- **One** platform (tool linked together/seamless toolchain)

Development Levels and Test Platforms



List of Challenges

We will present our current answers to some of following questions/challenges:

- **How to ensure Traceability to the System-Under-Test?**
- How to support Data Management for both Embedded Coder and TargetLink?
- How to handle Calibration Data for a component and Test Cases?
- **How to handle simulation data in a lean way to reduce out of memory issues but still ensure test results consistency?**
- How to push the limits of full ASW System simulation on MATLAB and Simulink?
- **How to support component aggregation in MATLAB and Simulink in a semi-automatic way?**
- **How to ensure consistency between all development artefacts?**
- How to ensure test continuity between different test platform (example MiL/SiL -> HiL)? [re-use and Back-to-Back]
- …

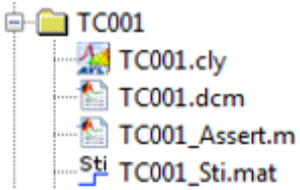
Lean Specification Data management

Test Specification Data are handled in single files.

One directory per Test Case.

Test Cases directory contains

- Specification data (stimuli, calibration file)
- Evaluation data (plot config, assessment script, reference signals)



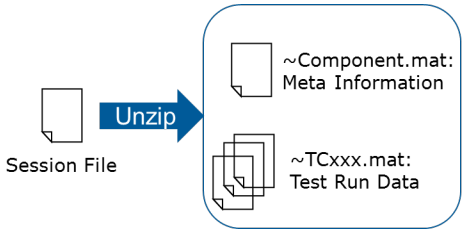
Advantages:

- Allows direct access to information (example Test Case calibration)
- Straightforward re-use of test cases
- Traceability Test Case Item to test case data (source traces in Integrity) with suspect/impact analysis

Lean Modular Simulation Data management

Simulation Data

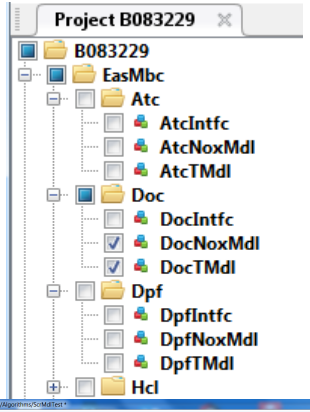
- Simulation Data is split from Configuration Data and saved in a separated file
- Simulation Data belonging to a Test Session are packed/ zipped together for test run consistency
- If the System-Under-Test is unchanged (checksum), Data are merged. Else reset.



Advantages:

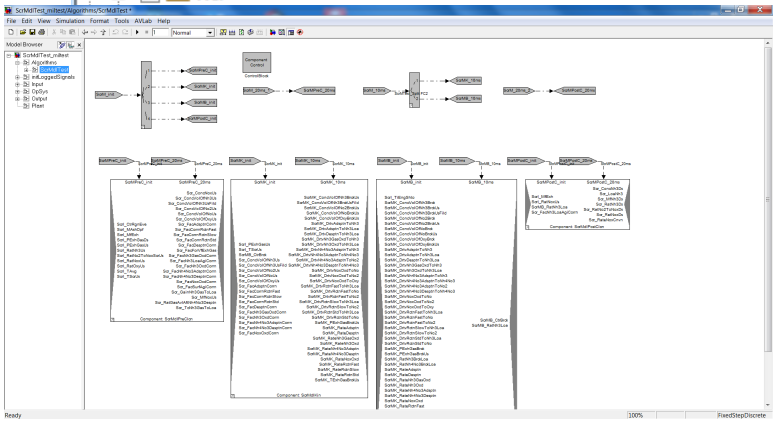
- Ensure consistency
- Lean memory usage for evaluation
  - Only result data from one Test Case is loaded simultaneously.
  - Only the necessary signals are loaded. (mat-file API)
- Supports Simulation Parallelization

Component Aggregation

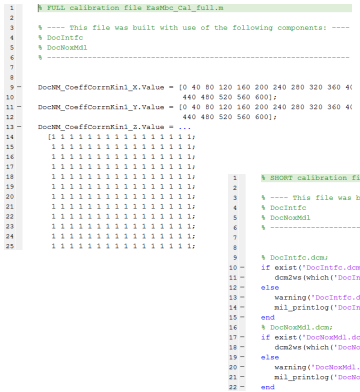


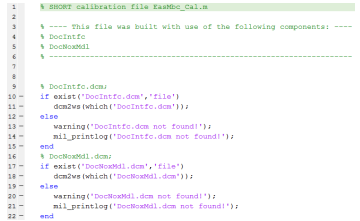
Project Interface to build/aggregate components

- Build/update model (using library or model reference)
- Build data files (flat included or referenced)
  - For TargetLink also .dd files are merged
- Build calibration file (flat included or referenced)



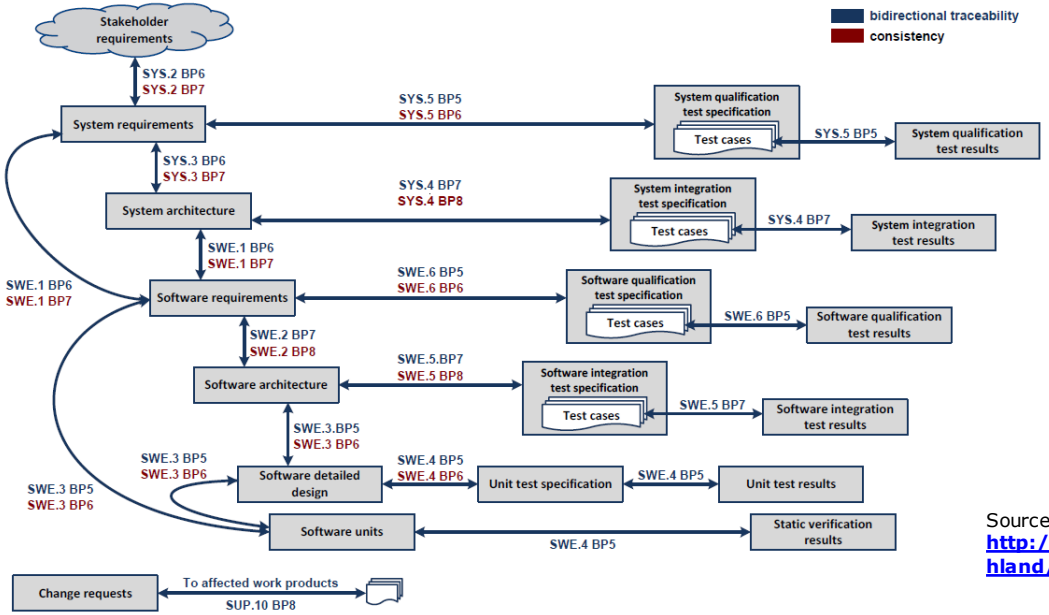
Semi-automatic: scheduling has to be finished manually.





Traceability & Consistency Challenge

In automotive SPICE 3.0 special focus on traceability and consistency.



Source: B. Sechser
http://www.slideshare.net/Polarion_Deutschland/automotive-spice-30-was-ndert-sich

Example: Test Report and Test Results shall reference to the revision of the System-Under-Test.

Traceability System-Under-Test (SUT)

Challenge:

Assure all test work products are traceable to their tested objects.

"What was tested?"

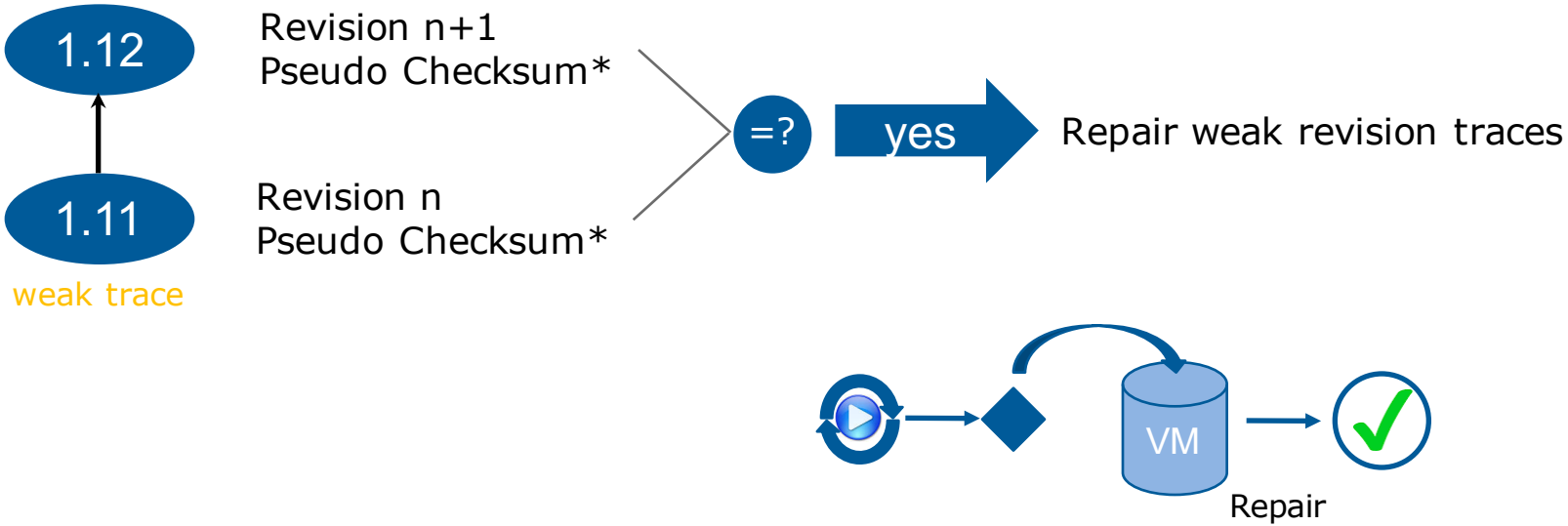
Summary of solutions

1. Display revision as Expanded Keywords Properties (Model, ASCII Files) 
2. Download revisions from Version Management Repository and run test against downloaded unmodified revisions (redirect path to download location)
3. Tracing revision in working Sandbox/on the fly; marking of weak (=modified) revision traces
4. Solution 3 + automatic repair of weak revision traces

Traceability System-Under-Test (Solution 4)

Solution 4: Run against working files and gather on the fly suspect/weak trace information
+ checksum information

Repair Weak revision traces



*Pseudo Checksum=Checksum of artefact without revision information

Artefacts Consistency Check

Goal:

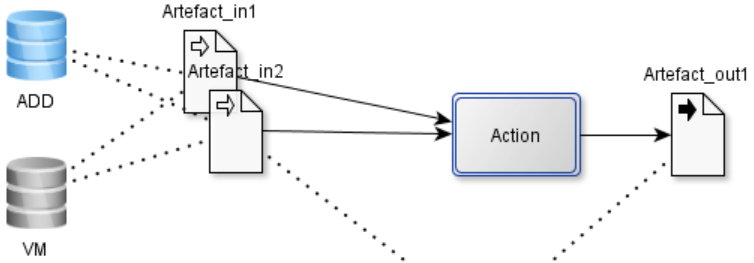
- While looking at a collection of artefacts, be able to check if they are consistent together.

Problematic examples/pitfalls:

- Are the test artefacts (e.g. Test Report, Test Results) consistent with the data label definition?
- Was the delivered model tested?
  = Is the test configuration in the delivery consistent with the one used for testing?
- Is the documentation up-to-date = consistent with the delivered model revision?
- Is the Test Report consistent with the deliverables?
- …

Artefacts Consistency Check - tracing

Label Repository



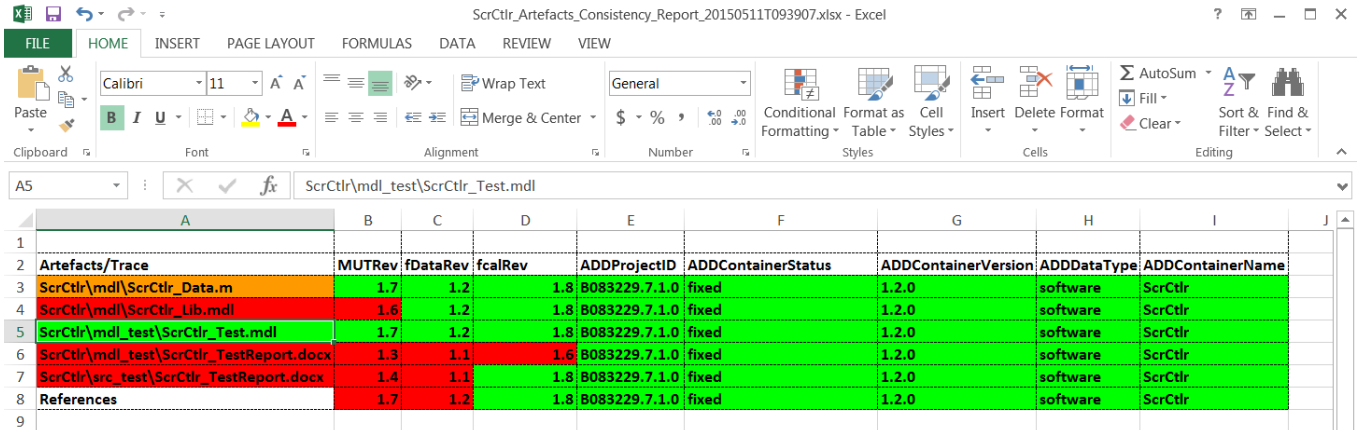
Version Management/
File Repository

Action Examples:

- Generate Documentation
- Write Test Report
- Run Test Session
- ADD Update
- Generate Code
- …

Traces are available as text (in file header) and hidden as file properties.

Artefact Consistency Check/ Matrix - Report



Artefacts/Trace	MUTRev	fDataRev	fcalRev	ADDProjectID	ADDContainerStatus	ADDContainerVersion	ADDDataType	ADDContainerName
ScrCtlr\mdl\ScrCtlr_Data.m	1.7	1.2	1.8	B083229.7.1.0	fixed	1.2.0	software	ScrCtlr
ScrCtlr\mdl\ScrCtlr_Lib.mdl	1.6	1.2	1.8	B083229.7.1.0	fixed	1.2.0	software	ScrCtlr
ScrCtlr\mdl_test\ScrCtlr_Test.mdl	1.7	1.2	1.8	B083229.7.1.0	fixed	1.2.0	software	ScrCtlr
ScrCtlr\mdl_test\ScrCtlr_TestReport.docx	1.3	1.1	1.6	B083229.7.1.0	fixed	1.2.0	software	ScrCtlr
ScrCtlr\src_test\ScrCtlr_TestReport.docx	1.4	1.1	1.8	B083229.7.1.0	fixed	1.2.0	software	ScrCtlr
References	1.7	1.2	1.8	B083229.7.1.0	fixed	1.2.0	software	ScrCtlr

Artefacts/Trace	MUTRev	fDataRev	fcalRev	ADDProjectID	ADDContainerStatus	ADDContainerVersion	ADDDataType	ADDContainerName
ScrMdl\mdl\ScrMdl_Data.m	1.26	1.19	1.23	B083229.7.2.0	*draft*	2.5.0	software	ScrMdl
ScrMdl\mdl\ScrMdl_Lib.mdl	1.26	1.19	1.21	B083229.7.2.0	*draft*	2.5.0	software	ScrMdl
ScrMdl\mdl_test\ScrMdl_Test.mdl	1.23	1.18	1.23	B083229.7.2.0	*draft*	2.5.0	software	ScrMdl
ScrMdl\mdl_test\ScrMdl_TestReport.docx	1.24	1.11	1.19	B083229.7.2.0	*draft*	2.3.0	software	ScrMdl
ScrMdl\src_test\ScrMdl_TestReport.docx		1.11	1.19	B083229.7.2.0	*draft*	2.4.0	software	ScrMdl
References	1.26	1.19	1.23	B083229.7.2.0	*draft*	2.5.0	software	ScrMdl

Outlook / Roadmap

2015: Major Milestones:

- Interface to Integrity Test Management
- Review / Consistency Module

Hot Topics: AUTOSAR, MultiCore Support

Continuity towards HiL, Virtual Integration Platform

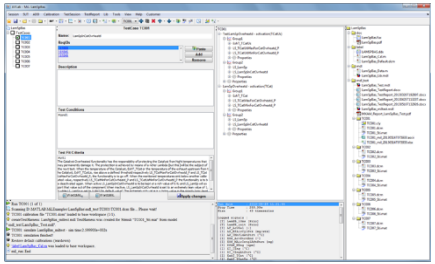
2016: Continuous Integration (Review)

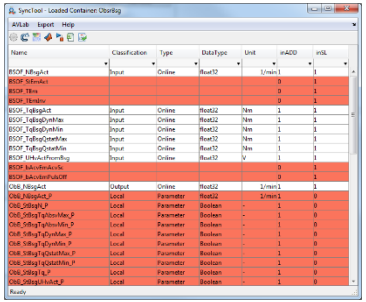
Conclusion

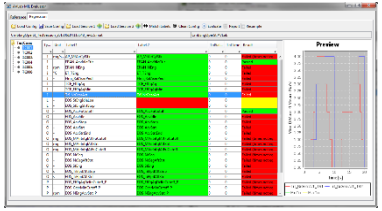
AVL Powertrain Controls uses an open standard internal platform to support and improve its model-based software development process.

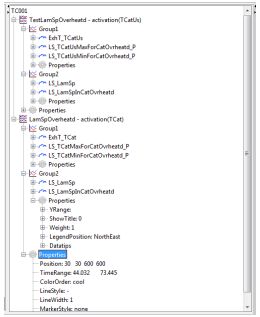
It is purely implemented in MATLAB and Simulink and connects many tools together.

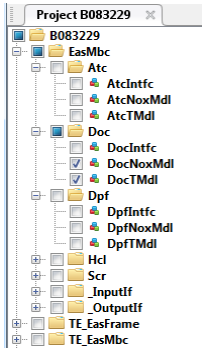
It tries to face many challenges for achieving high quality and provide a seamless toolchain with best practice state-of-the-art solutions to its users.











Contact:
Thierry Dalon
thierry.dalon@avl.com
+49 941 63089 - 203

THANK YOU

www.avl.com

Backup Slides

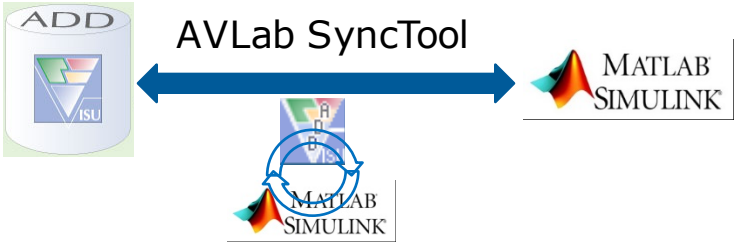
www.avl.com

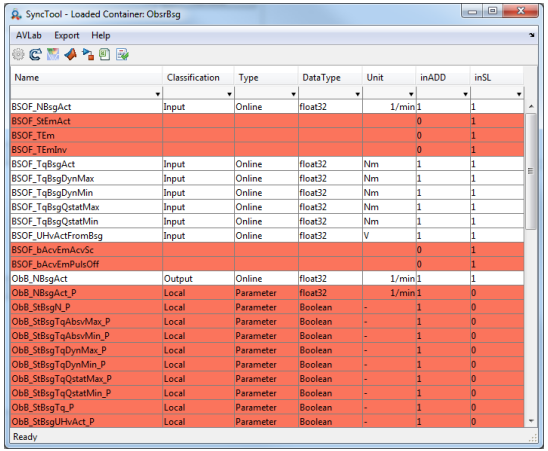
MATLAB/Simulink ADD Interface

Synchronize Tool

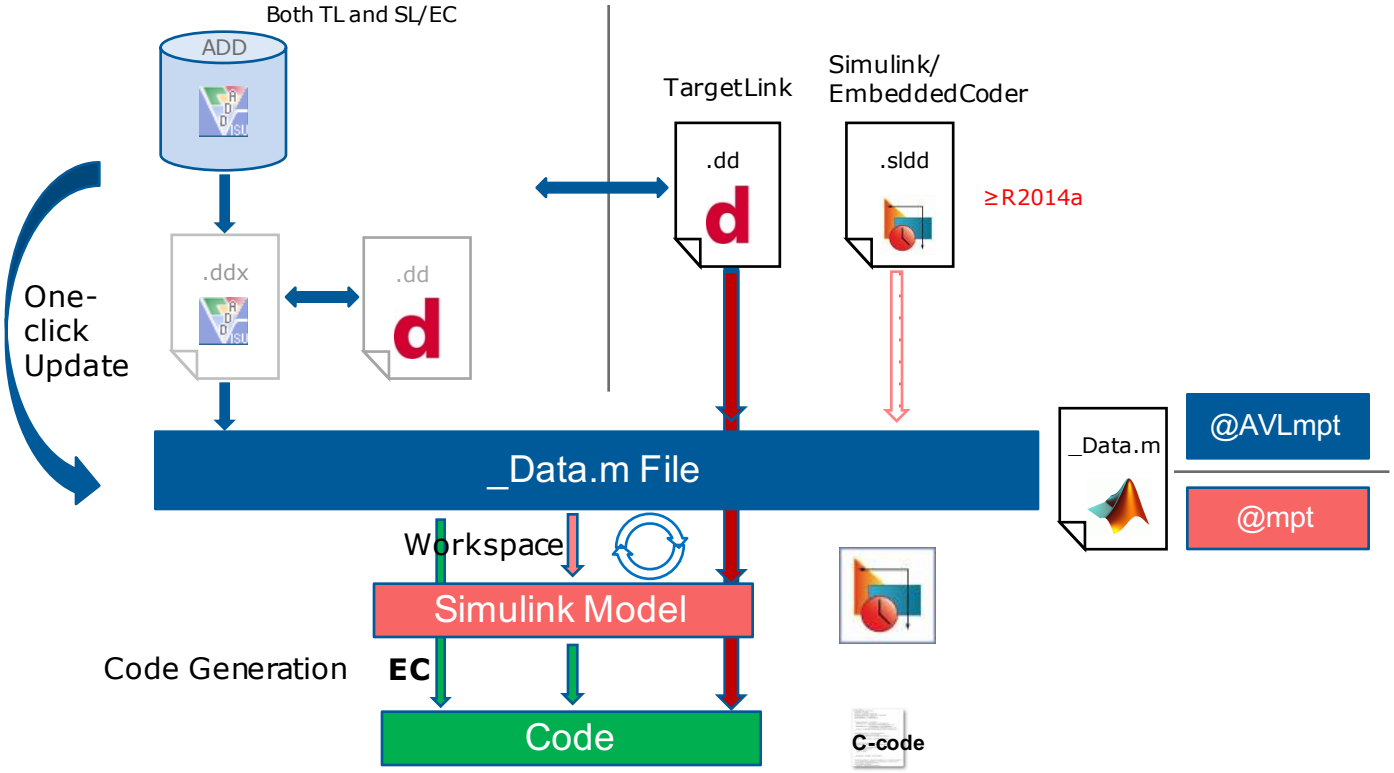
- One-click ADD Update (ADD to MATLAB: data synchronization)
- ADD2Simulink: Synchronization ADD<-> Simulink
- Consistency Check
- Seamless Integration

Simulink Model and MATLAB Workspace are in Sync





Data Management – Labels Dictionary



Editor - D:\Sandboxes\ARE2043\Controls\Functions\Hmi\DrvrReqFild\mdl\D

File Edit Text Go Cell Tools Debug Desktop Window Help

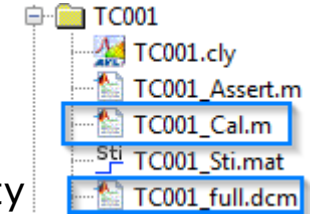
```
% Created by AVLab ddx2m on 29-Jan-2013 16:2
%   from file: D:\Sandboxes\ARE2043\Controls
%   Container name: DrvrReqFild
%   Container status: fixed
%   Version: 1.1.0
%   File created on 04-Dec-2012 10:40:39
%   File last modified on 29-Jan-2013 16:28:
%   Data Type: software
%   Number of Labels: 11

% --------------------------------------------
%  MATLAB file generated by Simulink.saveVar
%  MATLAB version: 7.13.0.564 (R2011b)
% --------------------------------------------


EPAN_NEng = AVLmpt.Signal;
EPAN_NEng.RTWInfo.StorageClass = 'Custom';
EPAN_NEng.RTWInfo.Alias = '';
EPAN_NEng.RTWInfo.Alignment = -1;
EPAN_NEng.RTWInfo.CustomStorageClass = 'VAR_
EPAN_NEng.RTWInfo.CustomAttributes.HeaderFil
EPAN_NEng.RTWInfo.CustomAttributes.Owner = '
EPAN_NEng.RTWInfo.CustomAttributes.Definitio
EPAN_NEng.RTWInfo.CustomAttributes.Persisten
EPAN_NEng.Description = 'Engine Speed (Combu
EPAN_NEng.DataType = 'uint16';
EPAN_NEng.Min = 0;
EPAN_NEng.Max = 65535;
EPAN_NEng.DocUnits = 'rpm';
EPAN_NEng.Dimensions = 1;
EPAN_NEng.DimensionsMode = 'auto';
EPAN_NEng.Complexity = 'real';
EPAN_NEng.SampleTime = -1;
```

Calibration Files Handling

- Default Global Calibration File under **mdl_test**/<compname>.dcm
- Tear-down
- Test Case calibration files:
  - <TCid>_Cal.m (extracted from _Data.m; only .Value)
  - _full.dcm is rewritten after run for documentation and traceability



=> this allows handling of TC calibration variation as Delta in an M-File

```
Run TC001 (1 of 1)
TC001 calibration file TC001_Cal.m loaded to base workspace.
ws2dcm:Workspace was exported to TC001_full.dcm
TestCase full dcm calibration file "D:\MATLAB\MiLExamples\LamSpBas\mdl_test\TC001\TC001_full.dcm" was written.
createTestHarness: LamSpBas_miltest.mdl TestHarness was created for Stimuli "TC001_Sti.mat" from model
"\mdl_test\LamSpBas_Test.mdl".
TC001: simulate LamSpBas_miltest - sim time:2.599900e+002s
TC001: simulation finished!
Restore default calibrations (teardown)
Scanning D:\MATLAB\MiLExamples\LamSpBas\mdl_test\LamSpBas.dcm file... Please wait!
\mdl_test\LamSpBas.dcm was loaded to base workspace.
mil_run: End
```

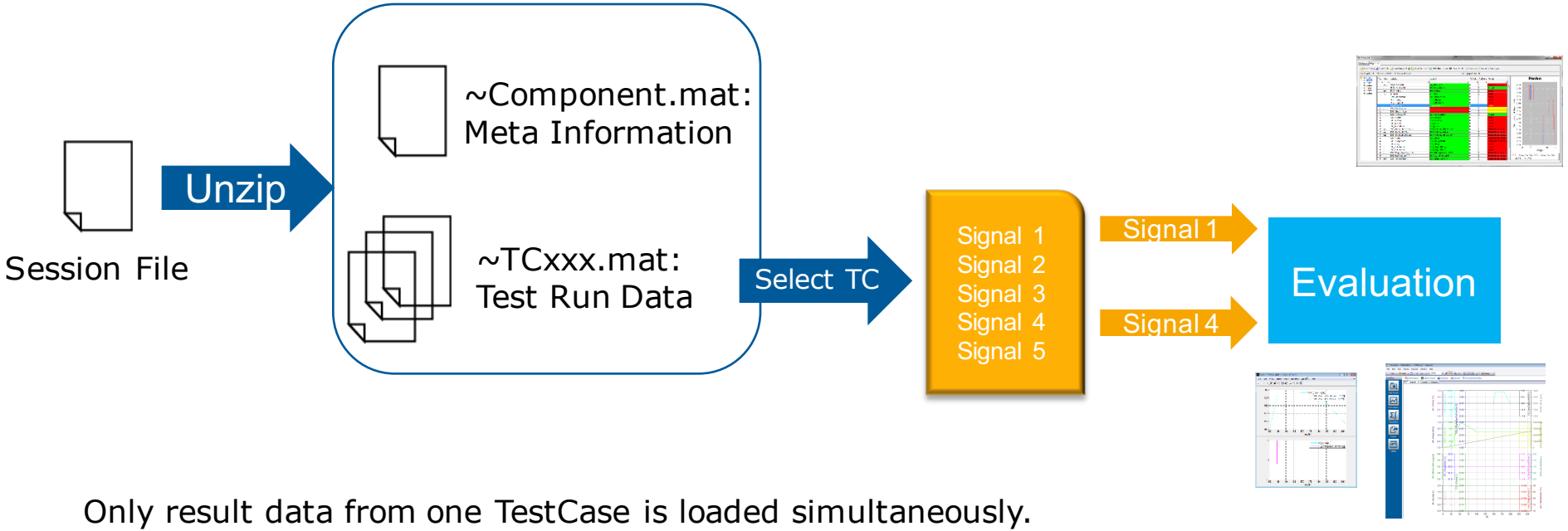
D:\Sandboxes\mkstrain\LamSpBas\mdl_test\TC008\TC008_Cal.m

EDITOR | PUBLISH | VIEW

```
1 -   LS_LamCorrnInCatOvrheatd_Cur.Value = ...
2       [-0.5 -0.37548828125 -0.2509765625 -0.12646484375 -0.001953125 0.12255859375 ...
3        0.2470703125 0.37158203125 0.49609375];
4 -   LS_LamSpInCatOvrheatd_Map.Value = ...
5       [1 1 1 1 1 1;
6        1 1 1 1 1 0.9502;
7        1 1 1 1 0.9502 0.9004;
8        1 1 1 1 0.9004 0.8496;
9        1 1 1 0.9502 0.8496 0.7998;
10       1 1 1 0.9004 0.7998 0.75];
11
```

Ln 1 Col 1

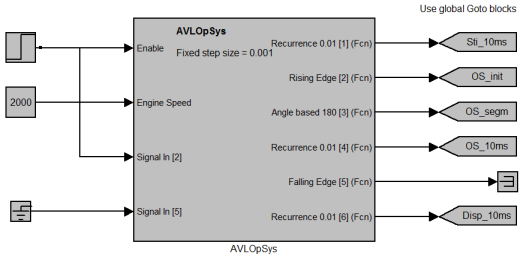
Lean Simulation Data Handling

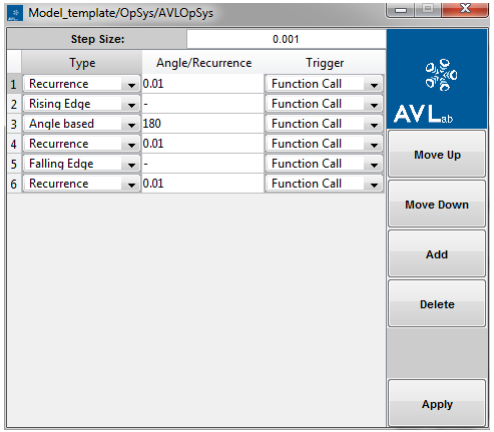


Only result data from one TestCase is loaded simultaneously.
With matfile API, only the necessary signals are loaded.

Prerequisite: Standard Component Structure

Model Template with Operating System

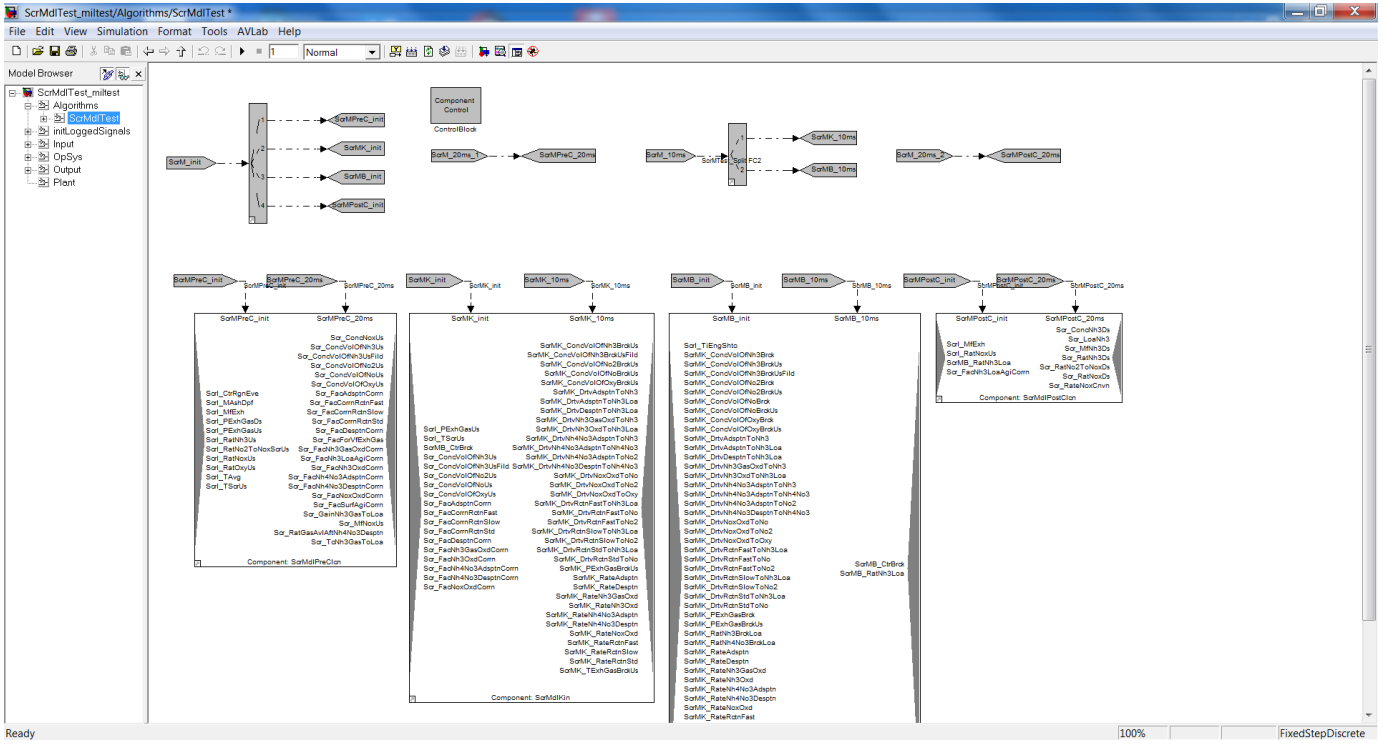




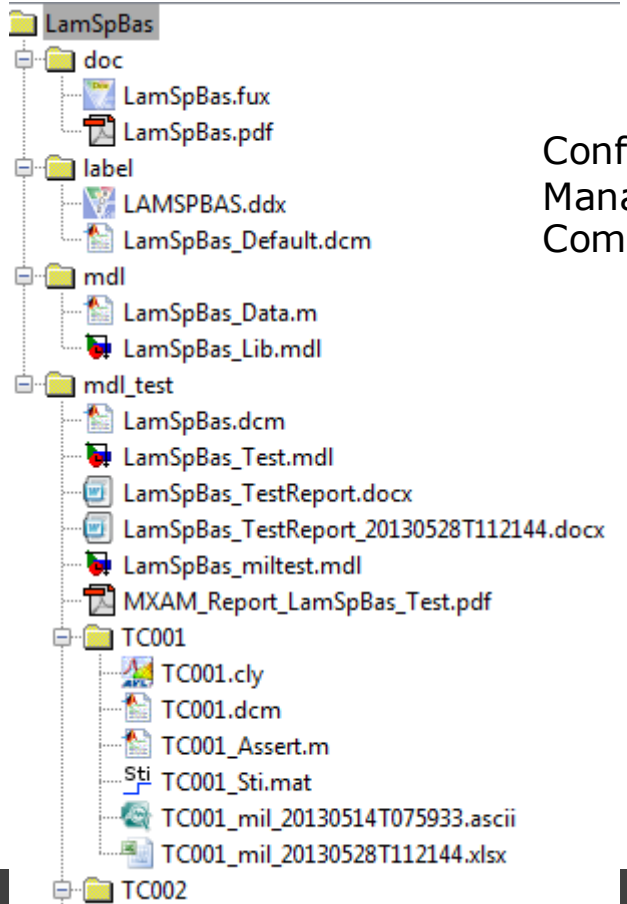
Support Closed-loop Test with Plant Model

Component Aggregation (Model)

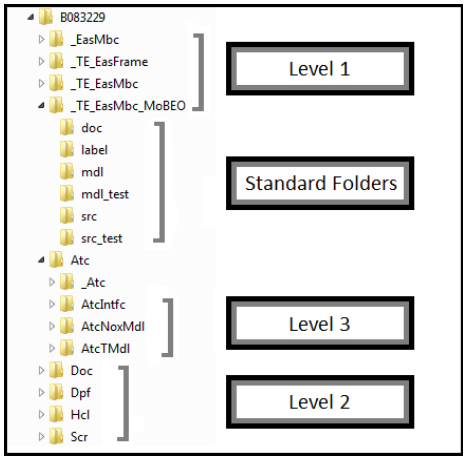
Scheduling is Semi-automatically created at aggregation



Standard Component File Structure

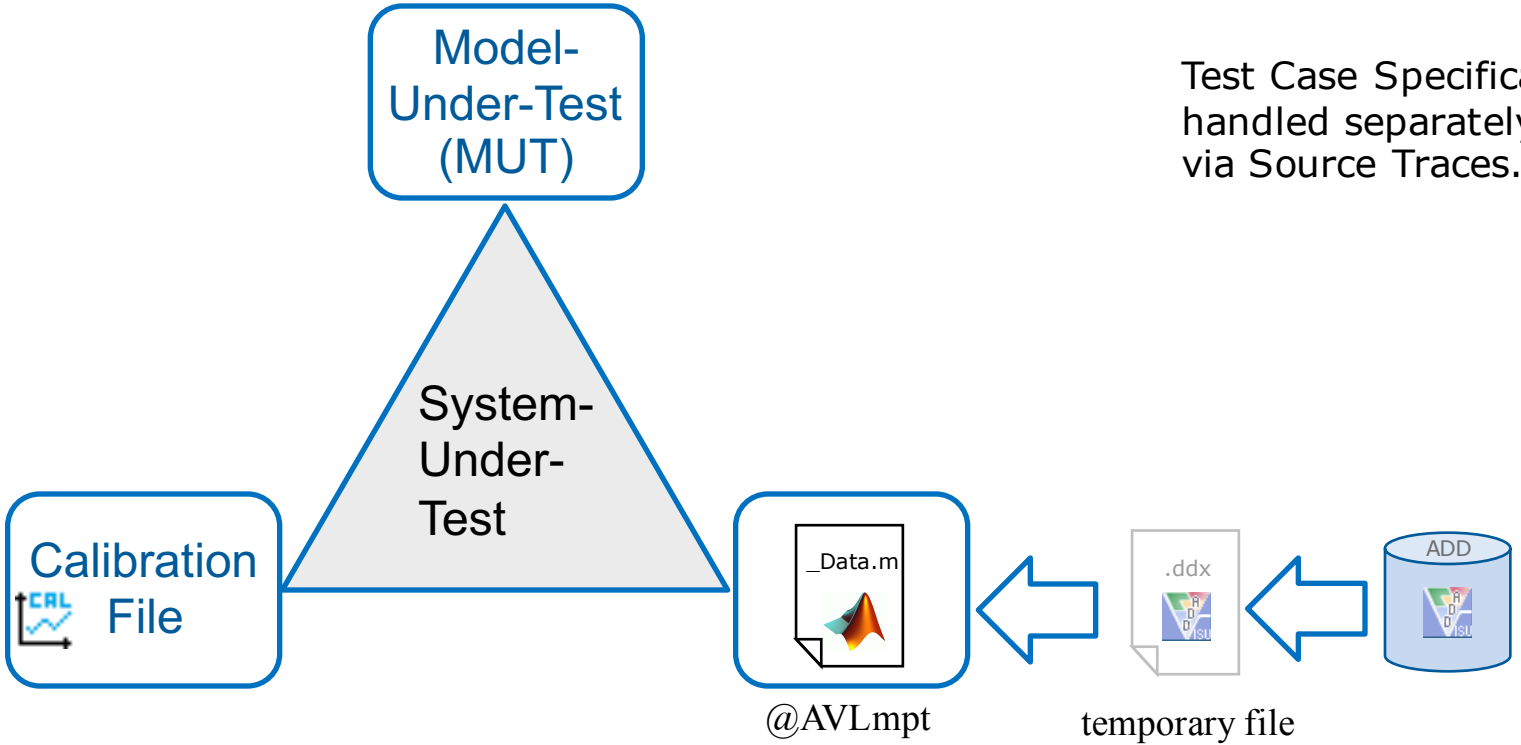


Configuration Management Plan at Component Level



CM Plan at Project/ ASW Sys Level with 3 levels architecture.
Level 1= Aggregation Level

System-Under-Test Definition

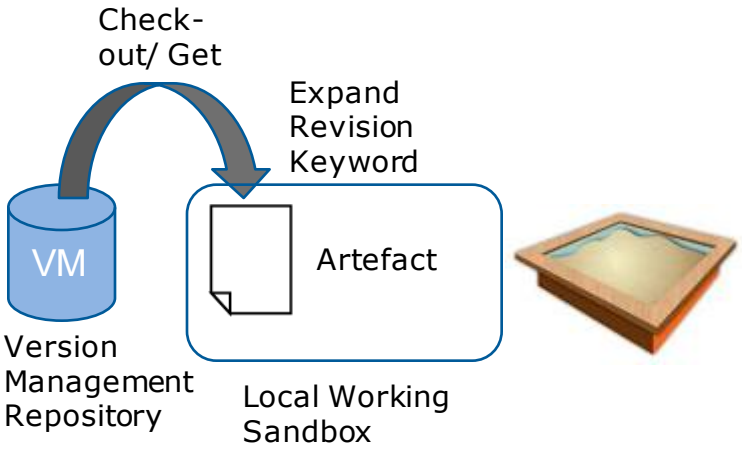


Test Case Specification files are handled separately and traced via Source Traces.

Split Data Definition from Values

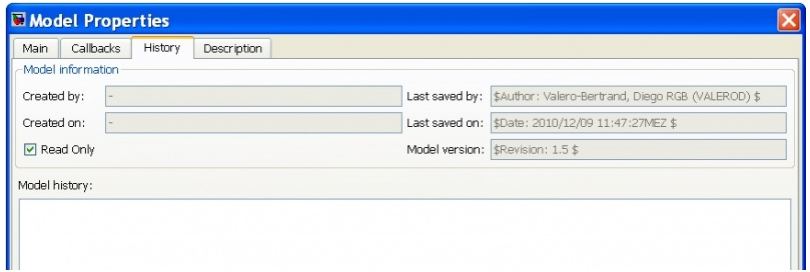
Traceability System-Under-Test (Solution 1)

Display revision information as Expanded Keywords Properties




```
% Copyright (c) AVL Software and Functions GmbH 2011
% $Revision: 1.6 $, $Date: 2015/07/31 09:30:47GMT+00:00 $ by $Author: Dalon, Thierry AVL/DE (DALONT) $
```

ASCII File

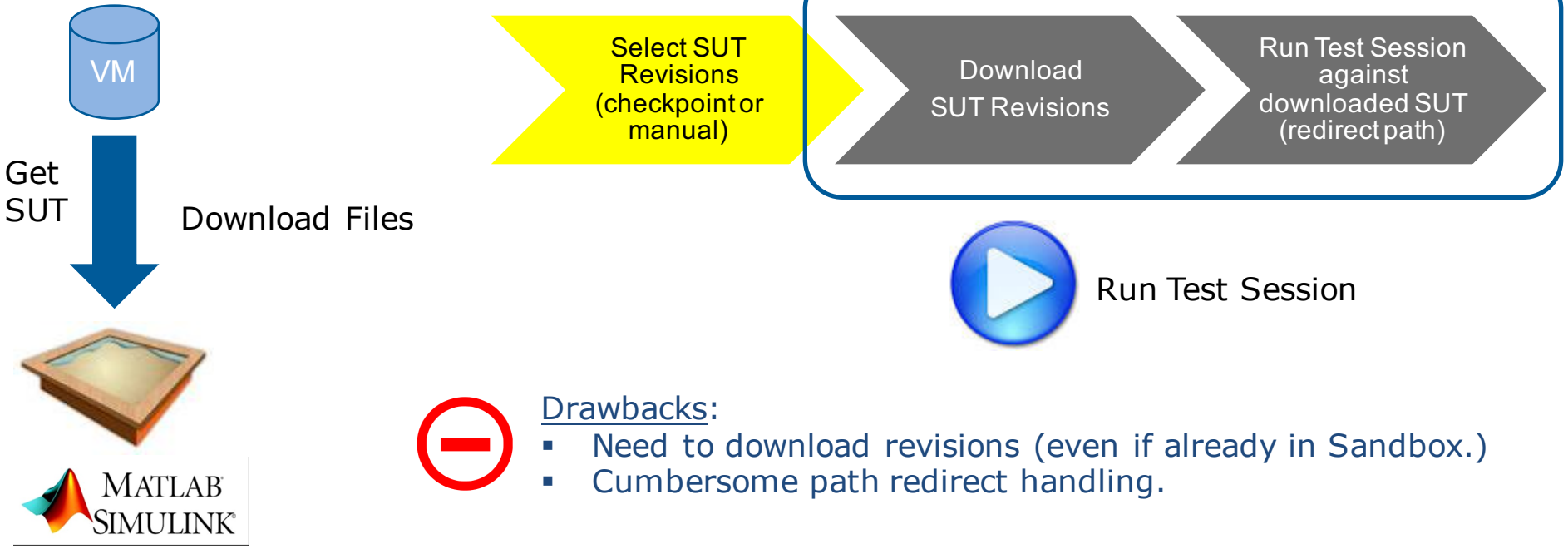


Simulink Model

If file is modified after Check-out/ Get, contained revision information is obsolete. This information can not be used as consistent trace to the VM repository.

Traceability System-Under Test (Solution 2)

Solution 2: Download SUT files and run test session against them without modifying local version in one batch



Drawbacks:

- Need to download revisions (even if already in Sandbox.)
- Cumbersome path redirect handling.

Traceability System-Under-Test (Solution 3)

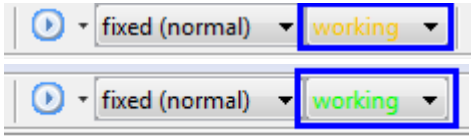
Solution 3: Run against working files and gather on the fly suspect/weak trace information

2.2 Test Environment

Information is necessary to guarantee the reproducibility of test results.

Test evaluation tool			AVLab 2.6.0.1
Simulation environment			MATLAB/Simulink 8.2 (R2013b) AVLib
MKS Revision			working
Run Mode			MiL fixed
ADD			
Container	N/A	Version Status Data type	N/A N/A N/A
Project	N/A		
System under Test			
MUT	LamSpBas_test.mdl		1.28 (modified)
Data File	LamSpBas_Data.m		1.26
Calibration File	LamSpBas.dcm		1.5

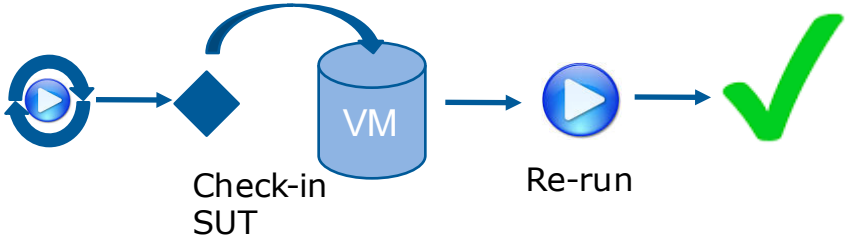
weak trace



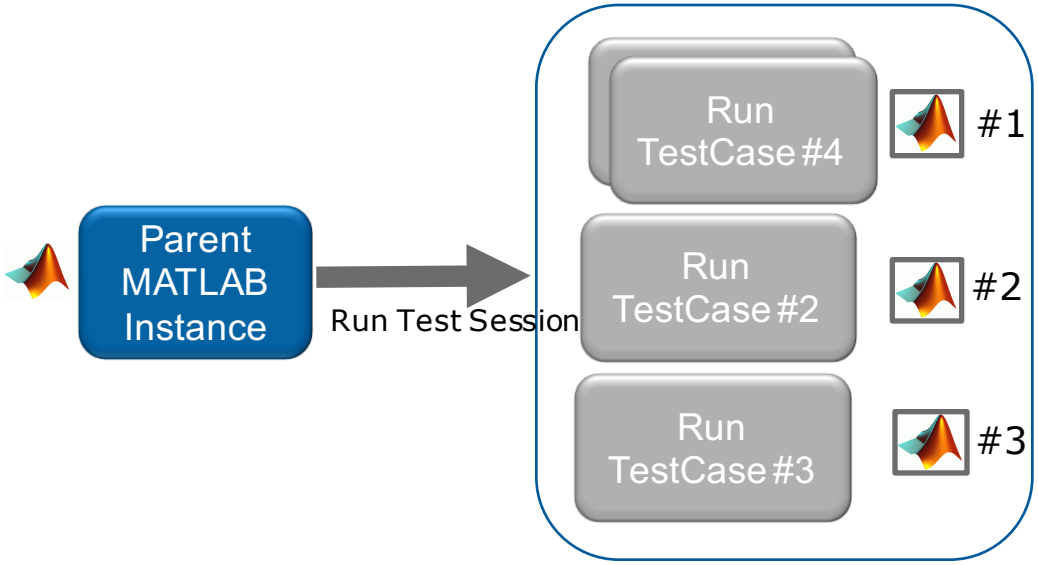
Advantages:

- No need for download and SUT redirect
- Can be checked automatically (at review, checkpoint…)

Drawback: Need to check-in/commit and **re-run** test for green reporting

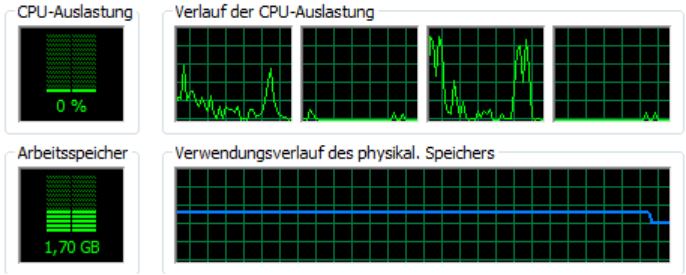


Parallel Execution



Parallel Simulation
Example 3 Children MATLAB Instances

Allocated memory in children MATLAB instances is freed up after every run.



Child
MATLAB
Instance

Parallel Execution - Process