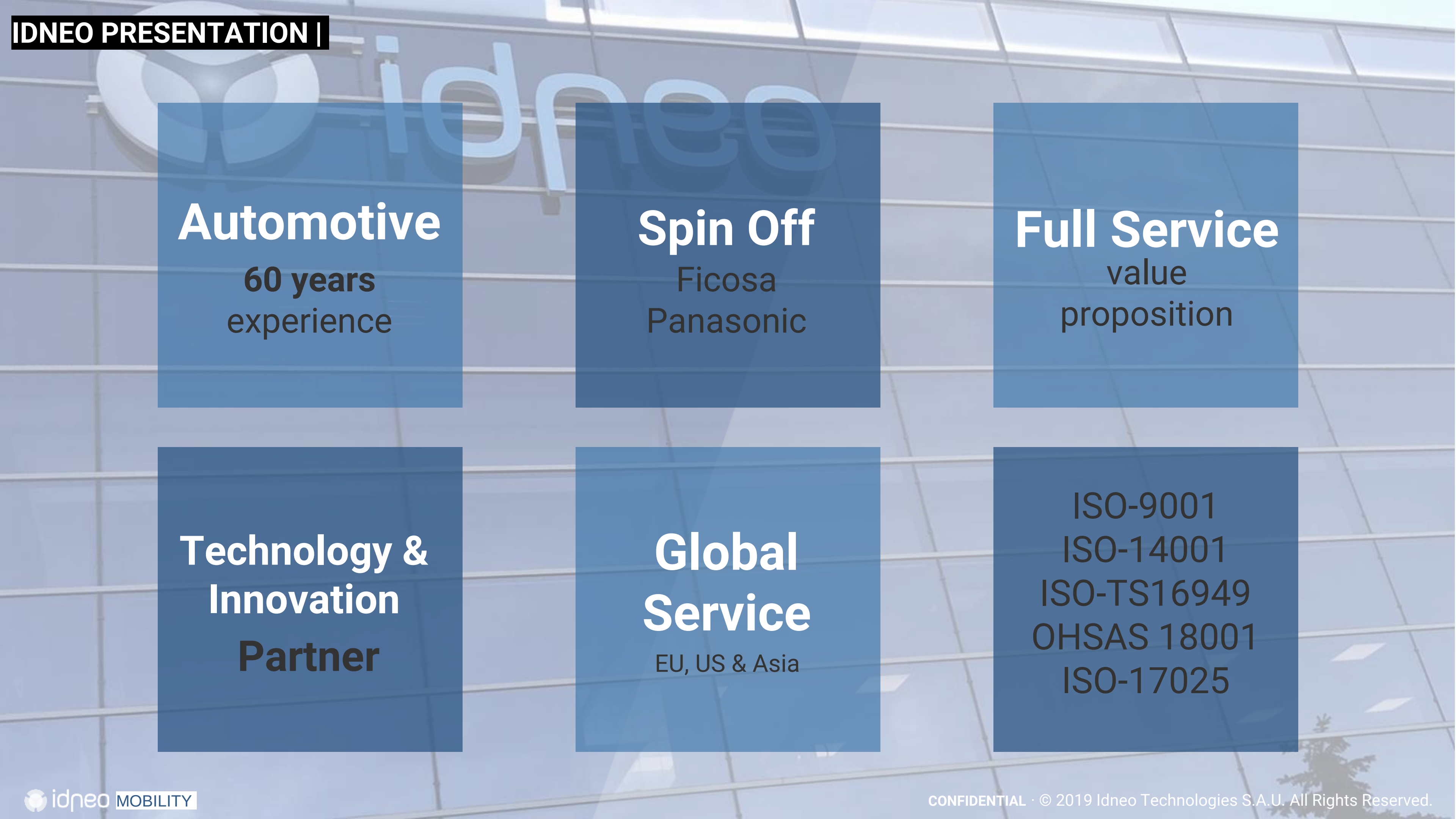
Agile approach to development and validation of AUTOSAR components

Dr. J. Albesa and Dr. E. Valencia
IDNEO Technologies

MATLAB EXPO 2019

Agenda

1. IDNEO introduction
2. Automotive software development process
3. Towards agile practices
4. Tools and methods
5. Projects and products
6. Conclusions



Automotive
60 years
experience

Spin Off
Ficosa
Panasonic

Full Service
value
proposition

Technology & Innovation
Partner

Global Service
EU, US & Asia

ISO-9001
ISO-14001
ISO-TS16949
OHSAS 18001
ISO-17025

Engineering By IDNEO is to timely adapt with critical thinking, method and processes to an ever-changing environment while continuously delivering value to our customers at sustainable pace

Full service partner

Innovation Strategy

Innovation Design

Development

Validation & Certification

Manufacturing

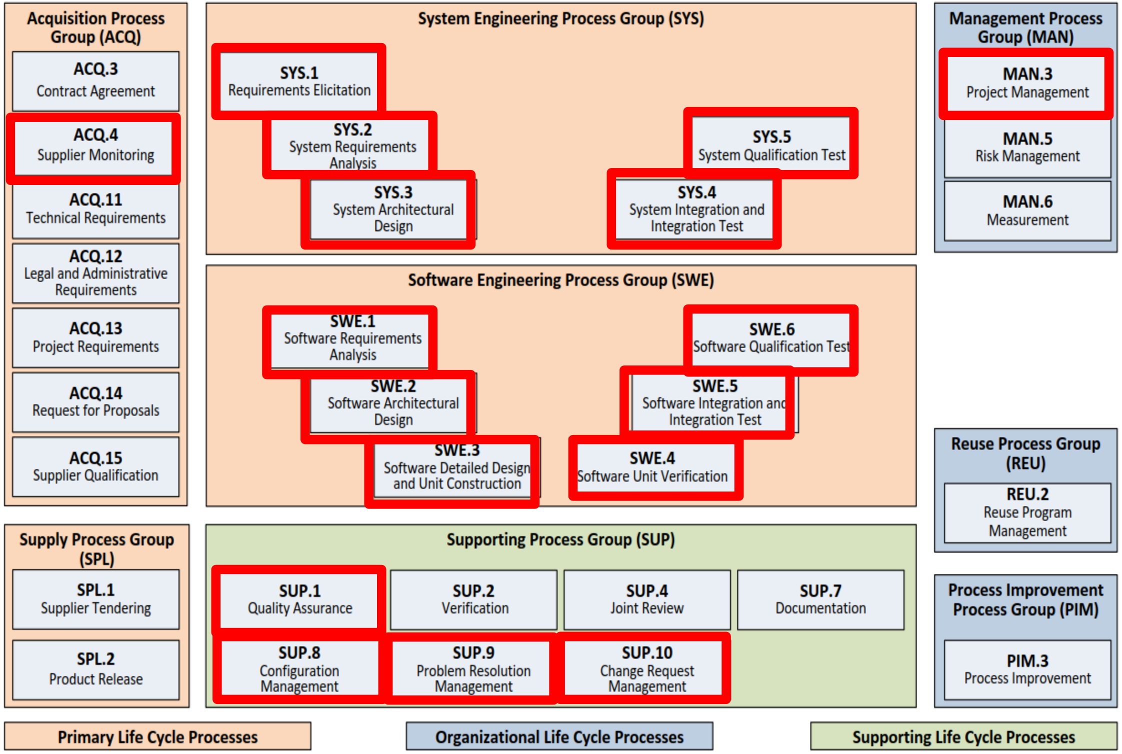
After Sales

idneo

Automotive Software Development Process

A-SPICE Model

🟥 HIS Scope



Acquisition Process Group (ACQ)
- ACQ.3 Contract Agreement
- 🟥 ACQ.4 Supplier Monitoring
- ACQ.11 Technical Requirements
- ACQ.12 Legal and Administrative Requirements
- ACQ.13 Project Requirements
- ACQ.14 Request for Proposals
- ACQ.15 Supplier Qualification

System Engineering Process Group (SYS)
- 🟥 SYS.1 Requirements Elicitation
- 🟥 SYS.2 System Requirements Analysis
- 🟥 SYS.3 System Architectural Design
- 🟥 SYS.4 System Integration and Integration Test
- 🟥 SYS.5 System Qualification Test

Management Process Group (MAN)
- 🟥 MAN.3 Project Management
- MAN.5 Risk Management
- MAN.6 Measurement

Software Engineering Process Group (SWE)
- 🟥 SWE.1 Software Requirements Analysis
- 🟥 SWE.2 Software Architectural Design
- 🟥 SWE.3 Software Detailed Design and Unit Construction
- 🟥 SWE.4 Software Unit Verification
- 🟥 SWE.5 Software Integration and Integration Test
- 🟥 SWE.6 Software Qualification Test

Reuse Process Group (REU)
- REU.2 Reuse Program Management

Supply Process Group (SPL)
- SPL.1 Supplier Tendering
- SPL.2 Product Release

Supporting Process Group (SUP)
- 🟥 SUP.1 Quality Assurance
- SUP.2 Verification
- SUP.4 Joint Review
- SUP.7 Documentation
- 🟥 SUP.8 Configuration Management
- 🟥 SUP.9 Problem Resolution Management
- 🟥 SUP.10 Change Request Management

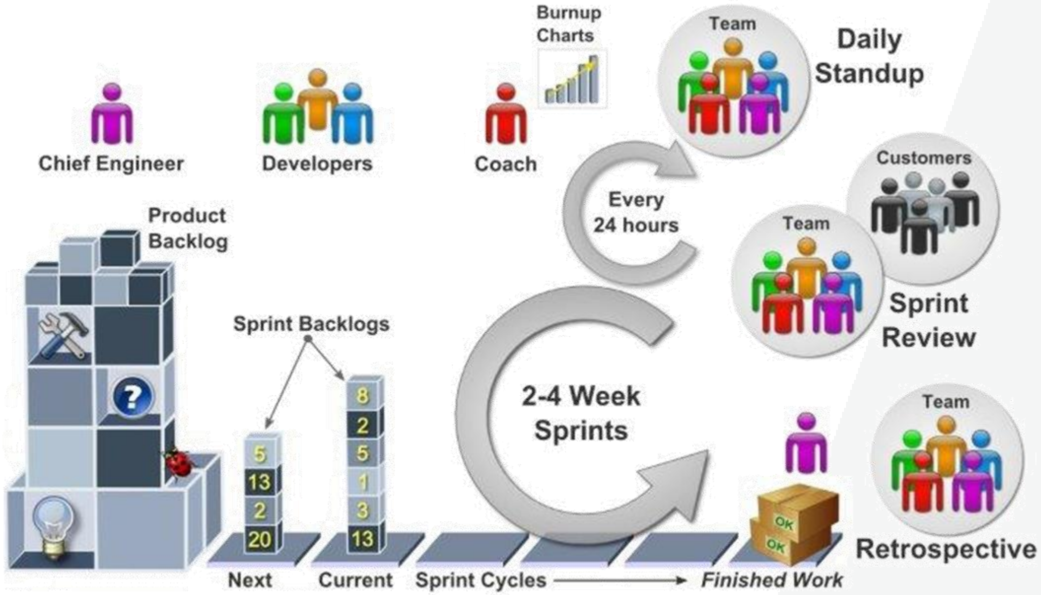
Process Improvement Process Group (PIM)
- PIM.3 Process Improvement

Primary Life Cycle Processes

Organizational Life Cycle Processes

Supporting Life Cycle Processes

Automotive shift towards Agile



IDNEO continuously inspects and adapts our products, processes and deliverables to **Client Needs**

Industry trend to shorter development time and fast design iterations pave the road for agile practices adoption.

Agile in Automotive Software Development?

Customer collaboration over contract negotiation

Responding to change over following a plan

Working software ~~over~~ documentation

People ~~over~~ processes and tools

Millions of units in the field
to be maintained 10+ years!

- Quality and maintainability require adhesion to established and audited processes.

Model-Based Design can help

Model Based Development is a established methodology which helps to agility...

- Abstraction from HW, early error detection, fast prototyping and design iterations, improved communication, ...

...but is not sufficient:

- Tools are required to ease process execution and produce process evidences.
- MathWorks products help from early design phases and design validation, up to coding and code verification (@IDNEO since 2015).

Simulation

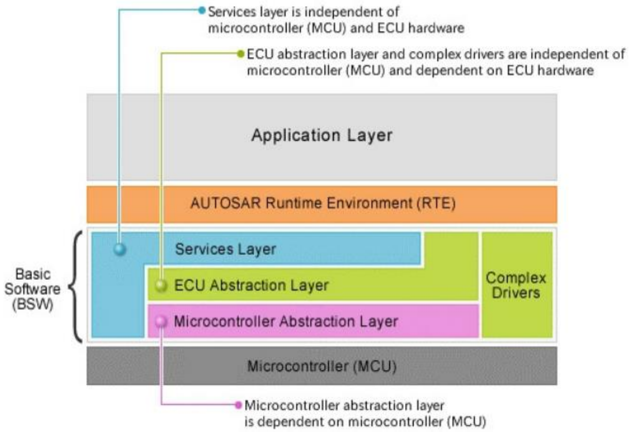
+

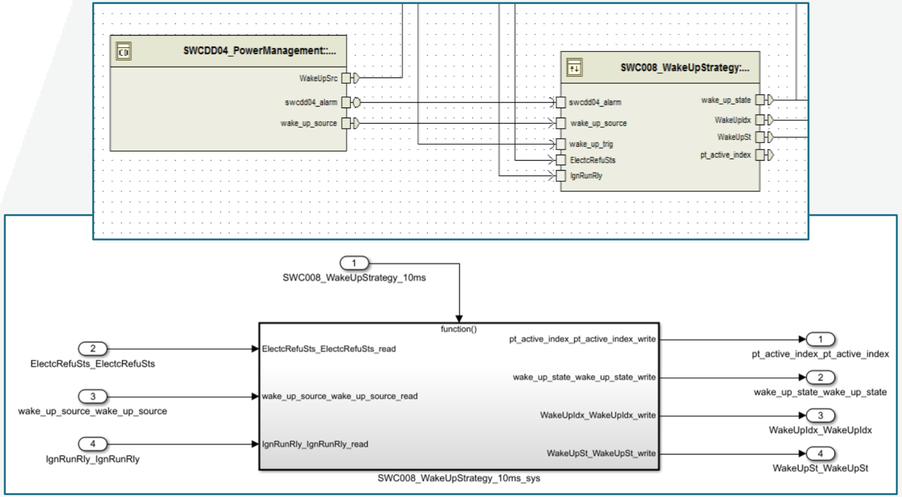
Automation

Design phase

From architecture to design

SW modules defined in architecture using AUTOSAR authoring tool → Module definition (ARXML artifact) imported into Simulink → Model scaffolding automatically generated → Design engineers fill in functional units with required implementation

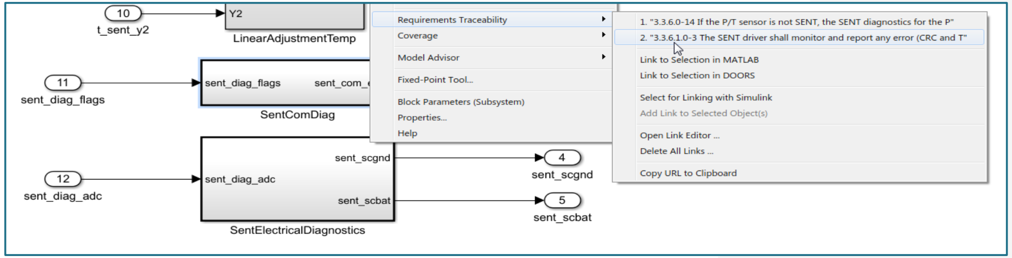

Services layer is independent of microcontroller (MCU) and ECU hardware
ECU abstraction layer and complex drivers are independent of microcontroller (MCU) and dependent on ECU hardware
Application Layer
AUTOSAR Runtime Environment (RTE)
Basic Software (BSW)
Services Layer
ECU Abstraction Layer
Complex Drivers
Microcontroller Abstraction Layer
Microcontroller (MCU)
Microcontroller abstraction layer is dependent on microcontroller (MCU)


SWCDD04_PowerManagement:...
WakeUpSrc
swcdd04_alarm
wake_up_source
SWC008_WakeUpStrategy:...
swcdd04_alarm
wake_up_source
wake_up_trig
ElectcRefuSts
IgnRunRly
wake_up_state
WakeUpIdx
WakeUpSt
pt_active_index
1
SWC008_WakeUpStrategy_10ms
function()
2
ElectcRefuSts_ElectcRefuSts
3
wake_up_source_wake_up_source
4
IgnRunRly_IgnRunRly
ElectcRefuSts_ElectcRefuSts_read
wake_up_source_wake_up_source_read
IgnRunRly_IgnRunRly_read
pt_active_index_pt_active_index_write
wake_up_state_wake_up_state_write
WakeUpIdx_WakeUpIdx_write
WakeUpSt_WakeUpSt_write
SWC008_WakeUpStrategy_10ms_sys
1
pt_active_index_pt_active_index
2
wake_up_state_wake_up_state
3
WakeUpIdx_WakeUpIdx
4
WakeUpSt_WakeUpSt

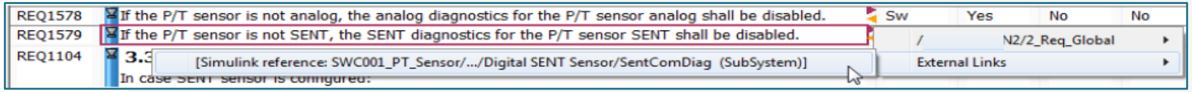
Traceability requirements <-> design

Are we done yet? What is this block meant for?

- Bidirectional traceability between design and requirements helps to answer those questions (+ needs to be demonstrated…).
- Direct linkage between Simulink design blocks and requirements management tool:



1. "3.3.6.0-14 If the P/T sensor is not SENT, the SENT diagnostics for the P"
2. "3.3.6.1.0-3 The SENT driver shall monitor and report any error (CRC and T"



REQ1578 If the P/T sensor is not analog, the analog diagnostics for the P/T sensor analog shall be disabled. Sw Yes No No

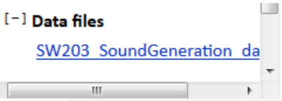
REQ1579 If the P/T sensor is not SENT, the SENT diagnostics for the P/T sensor SENT shall be disabled.

REQ1104 3.3 In case SENT sensor is configured:

[Simulink reference: SWC001_PT_Sensor/.../Digital SENT Sensor/SentComDiag (SubSystem)]

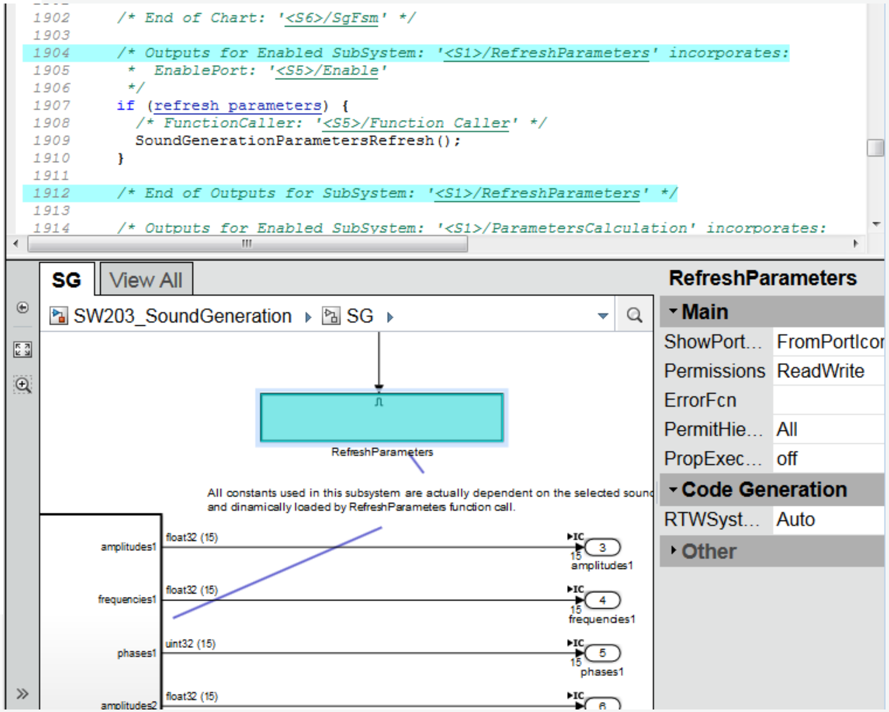
Design reports

- Design reports help at understanding and reviewing design.
- Full-blown HTML reports very useful for development and quality teams:
  - Browsable design, block configuration parameters, traceability design to code, …
- Sometimes too much info for customers:
  - IDNEO custom reports using Report Generator API + DOM API to programmatically create PDF documents.



[-] Data files

SW203_SoundGeneration_da



```
/* End of Chart: '<S6>/SgFsm' */

/* Outputs for Enabled SubSystem: '<S1>/RefreshParameters' incorporates:
 *  EnablePort: '<S5>/Enable'
 */
if (refresh_parameters) {
  /* FunctionCaller: '<S5>/Function Caller' */
  SoundGenerationParametersRefresh();
}

/* End of Outputs for SubSystem: '<S1>/RefreshParameters' */

/* Outputs for Enabled SubSystem: '<S1>/ParametersCalculation' incorporates:
```

All constants used in this subsystem are actually dependent on the selected sound and dinamically loaded by RefreshParameters function call.

RefreshParameters

Main

ShowPort...	FromPortIcor
Permissions	ReadWrite
ErrorFcn	
PermitHie...	All
PropExec...	off

Code Generation

RTWSyst...	Auto

Other

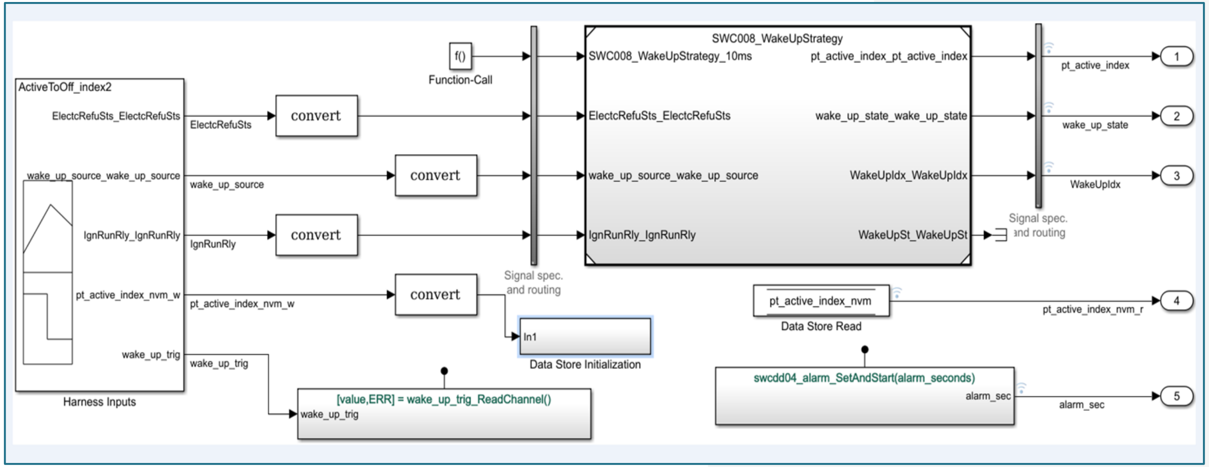
Design validation phase

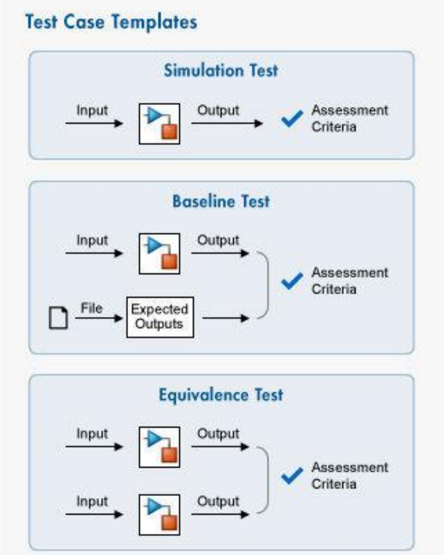
From simulation to design validation

- With Model-Based Design, design and design validation become very coupled processes.
- Design engineer performs exploratory simulations right from the time of model inception.
- Need for testing increases with design maturity:
  - Test suite with controlled (repeatable and maintainable) test cases.
- Simulink Test used to create and execute model test suites.
  - @IDNEO, replaced internal tool that required high development maintenance effort.

Simulink Test

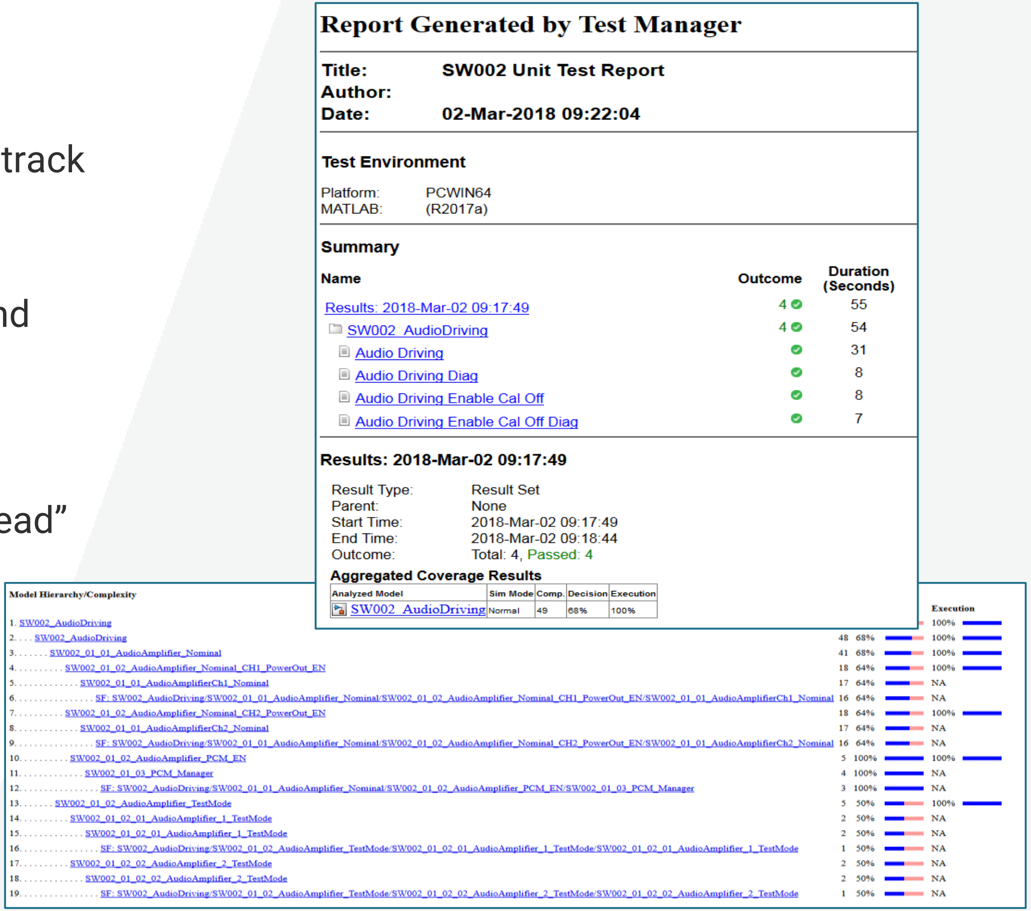
- Test Manager to create test suites and define test cases
- Test harnesses automatically created for the model under test
- Controlled setting of inputs and parameters for each test case
- Model output assessment based on different criteria





Test results and coverage reports

- Native test results report used to keep track of test status and outcomes.
- Cumulative test coverage computed and reported for the complete test suite.
- Coverage helps to assess test suite completeness, as well as to identify “dead” Simulink blocks.
- Justification feature useful for audit purposes.



Report Generated by Test Manager

Title:	SW002 Unit Test Report
Author:	
Date:	02-Mar-2018 09:22:04

Test Environment

Platform:	PCWIN64
MATLAB:	(R2017a)

Summary

Name	Outcome	Duration (Seconds)
Results: 2018-Mar-02 09:17:49	4 ✓	55
SW002_AudioDriving	4 ✓	54
Audio Driving	✓	31
Audio Driving Diag	✓	8
Audio Driving Enable Cal Off	✓	8
Audio Driving Enable Cal Off Diag	✓	7

Results: 2018-Mar-02 09:17:49

Result Type:	Result Set
Parent:	None
Start Time:	2018-Mar-02 09:17:49
End Time:	2018-Mar-02 09:18:44
Outcome:	Total: 4, Passed: 4

Aggregated Coverage Results

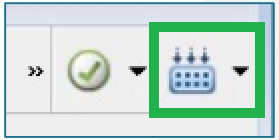
Analyzed Model	Sim Mode	Comp.	Decision	Execution
SW002_AudioDriving	Normal	49	68%	100%

Model Hierarchy/Complexity			Execution
1. SW002_AudioDriving			100%
2. . . . SW002_AudioDriving	48	68%	100%
3. . . . . . . SW002_01_01_AudioAmplifier_Nominal	41	68%	100%
4. . . . . SW002_01_02_AudioAmplifier_Nominal_CH1_PowerOut_EN	18	64%	100%
5. . . . . SW002_01_01_AudioAmplifierCh1_Nominal	17	64%	NA
6. . . . . SF: SW002_AudioDriving/SW002_01_01_AudioAmplifier_Nominal/SW002_01_02_AudioAmplifier_Nominal_CH1_PowerOut_EN/SW002_01_01_AudioAmplifierCh1_Nominal	16	64%	NA
7. . . . . SW002_01_02_AudioAmplifier_Nominal_CH2_PowerOut_EN	18	64%	100%
8. . . . . SW002_01_01_AudioAmplifierCh2_Nominal	17	64%	NA
9. . . . . SF: SW002_AudioDriving/SW002_01_01_AudioAmplifier_Nominal/SW002_01_02_AudioAmplifier_Nominal_CH2_PowerOut_EN/SW002_01_01_AudioAmplifierCh2_Nominal	16	64%	NA
10. . . . . SW002_01_02_AudioAmplifier_PCM_EN	5	100%	100%
11. . . . . SW002_01_03_PCM_Manager	4	100%	NA
12. . . . . SF: SW002_AudioDriving/SW002_01_01_AudioAmplifier_Nominal/SW002_01_02_AudioAmplifier_PCM_EN/SW002_01_03_PCM_Manager	3	100%	NA
13. . . . . . SW002_01_02_AudioAmplifier_TestMode	5	50%	100%
14. . . . . SW002_01_02_01_AudioAmplifier_1_TestMode	2	50%	NA
15. . . . . SW002_01_02_01_AudioAmplifier_1_TestMode	2	50%	NA
16. . . . . SF: SW002_AudioDriving/SW002_01_02_AudioAmplifier_TestMode/SW002_01_02_01_AudioAmplifier_1_TestMode/SW002_01_02_01_AudioAmplifier_1_TestMode	1	50%	NA
17. . . . . SW002_01_02_02_AudioAmplifier_2_TestMode	2	50%	NA
18. . . . . SW002_01_02_02_AudioAmplifier_2_TestMode	2	50%	NA
19. . . . . SF: SW002_AudioDriving/SW002_01_02_AudioAmplifier_TestMode/SW002_01_02_02_AudioAmplifier_2_TestMode/SW002_01_02_02_AudioAmplifier_2_TestMode	1	50%	NA

Code generation and verification

Code generation

- Once design is ready, Embedded Coder and AUTOSAR support package are used for production code generation (C language).
- Interfaces and architectural dependencies automatically in-place.
- Straightforward integration into AUTOSAR platform, it compiles right away.



Code verification

Simulink Test

- Direct reuse of design test suite by running in SIL mode.
- Equivalence test between MIL and SIL.

Polyspace

- Built-in configuration for AGC:
  - Reduction of “noise” level in the analysis.
- ...and, of course, Polyspace auto-generated reports.

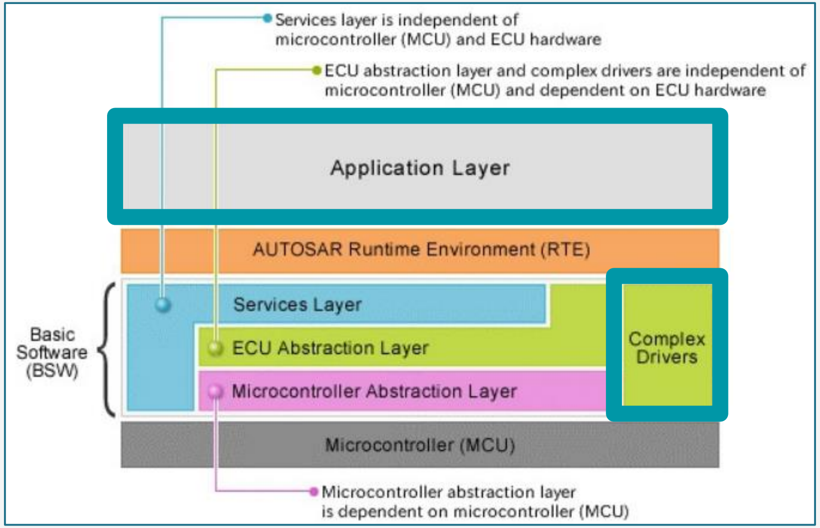
Products and projects

Model-Based Design with MathWorks products used in several projects for different product types...

- AdBlue dosing systems.
- Brake by wire.
- Fuel door lock.
- Engine sound control.
- Water injection for combustion engines.
- Door access module.
- In-vehicle camera monitoring system.

… and different software architecture layers

- High-level application logic.
- Software drivers close to HW.


Services layer is independent of microcontroller (MCU) and ECU hardware
ECU abstraction layer and complex drivers are independent of microcontroller (MCU) and dependent on ECU hardware
Application Layer
AUTOSAR Runtime Environment (RTE)
Basic Software (BSW)
Services Layer
ECU Abstraction Layer
Microcontroller Abstraction Layer
Complex Drivers
Microcontroller (MCU)
Microcontroller abstraction layer is dependent on microcontroller (MCU)

Technical Article available in MathWorks site:

Developing and Testing AUTOSAR Software Components and Complex Device Drivers with Model-Based Design

By Enric Valencia, Ph.D., and Joan Albesa, Ph.D., IDNEO

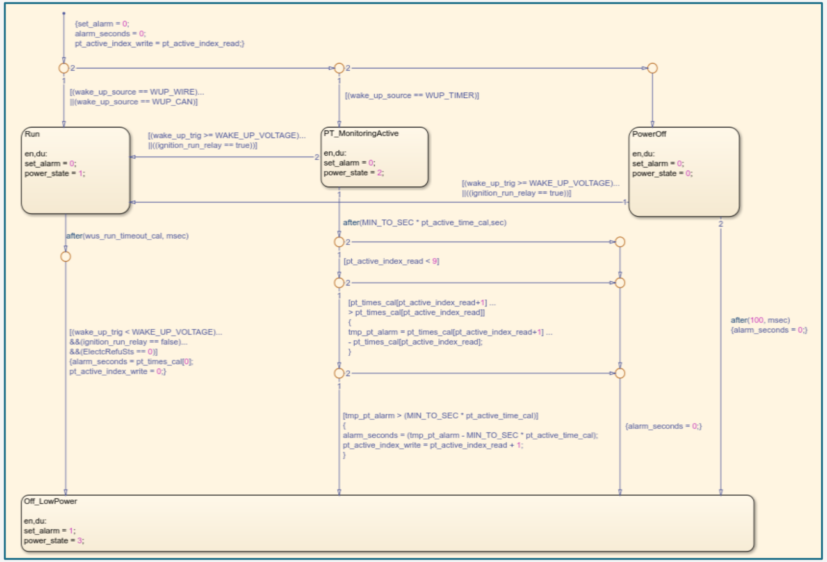
Example 1: Wake-up logic for control unit

Main characteristics:

- “Simple” and small SW component, but in a full AUTOSAR architecture.

Key takeaways:

- Model template auto-generated from ARXML
- Auto-generation avoids manual boilerplate code.


{set_alarm = 0;
alarm_seconds = 0;
pt_active_index_write = pt_active_index_read;}
[(wake_up_source == WUP_WIRE)...
||(wake_up_source == WUP_CAN)]
[(wake_up_source == WUP_TIMER)]
Run
en,du:
set_alarm = 0;
power_state = 1;
[(wake_up_trig >= WAKE_UP_VOLTAGE)...
||((ignition_run_relay == true))]
PT_MonitoringActive
en,du:
set_alarm = 0;
power_state = 2;
PowerOff
en,du:
set_alarm = 0;
power_state = 0;
[(wake_up_trig >= WAKE_UP_VOLTAGE)...
||((ignition_run_relay == true))]
after(MIN_TO_SEC * pt_active_time_cal,sec)
after(wus_run_timeout_cal, msec)
[pt_active_index_read < 9]
[pt_times_call[pt_active_index_read+1] ...
> pt_times_call[pt_active_index_read]]
{
tmp_pt_alarm = pt_times_call[pt_active_index_read+1] ...
- pt_times_call[pt_active_index_read];
}
[(wake_up_trig < WAKE_UP_VOLTAGE)...
&&(ignition_run_relay == false)...
&&(ElectcRefuSts == 0)]
{alarm_seconds = pt_times_call[0];
pt_active_index_write = 0;}
after(100, msec)
{alarm_seconds = 0;}
[tmp_pt_alarm > (MIN_TO_SEC * pt_active_time_cal)]
{
alarm_seconds = (tmp_pt_alarm - MIN_TO_SEC * pt_active_time_cal);
pt_active_index_write = pt_active_index_read + 1;
}
{alarm_seconds = 0;}
Off_LowPower
en,du:
set_alarm = 1;
power_state = 3;

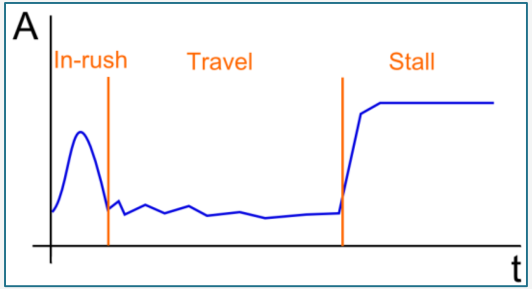
Example 2: Driver for DC motor

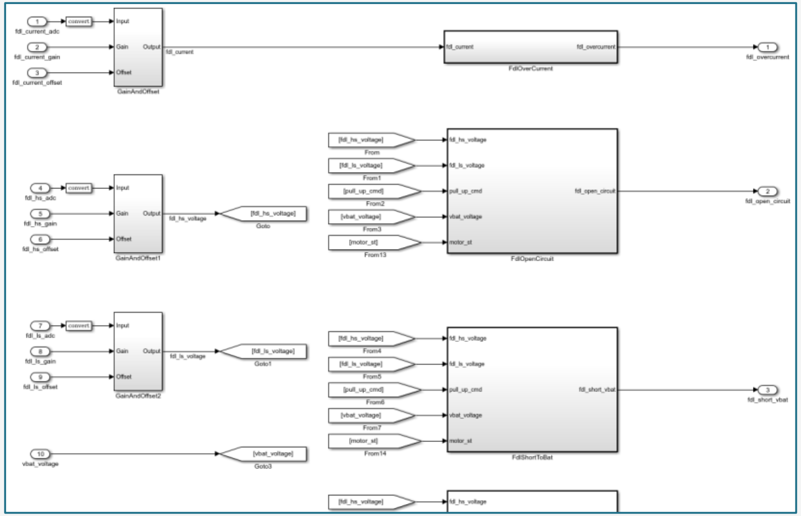
Main characteristics:

- Control of HW actuator (DC motor) with diagnostics.

Key takeaways:

- Simulation of driver nominal behavior.
- Simulation of HW abnormal behavior (diagnostics), difficult to mimic in real system setup.





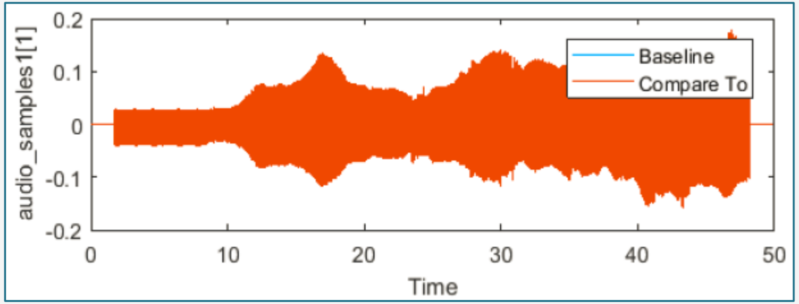
Example 3: Engine sound generation

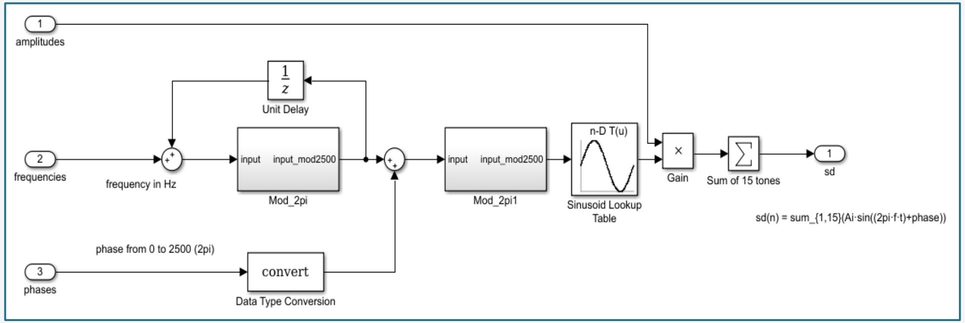
Main characteristics:

- Signal processing algorithms: digital filtering, linear interpolation, ...

Key takeaways:

- Parameter override for each test case.
- Store test output as WAV for audio engineer.
- Baseline recording in Simulink Test.





Conclusions

Agile practices in automotive SW development

Model-Based Design contributes to agile by means of common language (model) and faster development time (simulation).

Automation capabilities of MathWorks tools enable agile in practice without compromising quality standards.

Quality enhancement

Early error detection from design phase.

Consistent model validation.

Improved code quality.

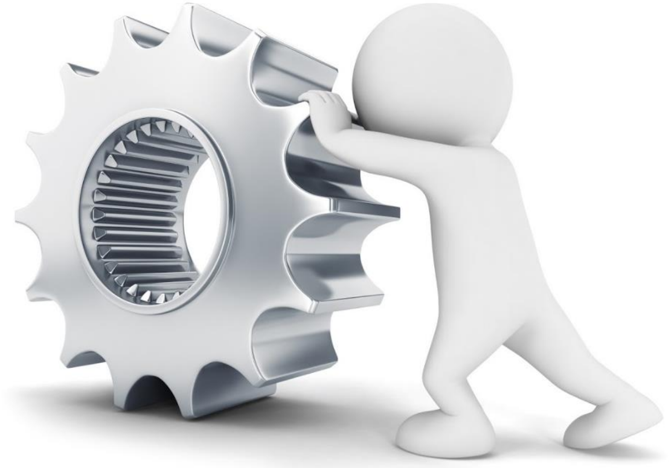
Maintainability

Better readability (model vs code).

Enhanced traceability.

Efficiency gain

A single tool-chain/team,
many different application scenarios



WHAT’s your DREAM?

www.idneo.com