FROM MULTI-PHYSICAL SYSTEM SIMULATION TO INDUSTRIAL CODE GENERATION

10th October 2023 - MATLAB Expo France
@NEWCAP Event Center – Paris

Rémi FAYOLLE - Anthony MICHEL
About us

Rémi FAYOLLE – Software Engineering Manager

I started at Dassault Systèmes by developing desktop applications on CATIA V6 & 3D Experience platform. Aiming to go back to Grenoble, I joined Symbio as a Software Engineer 7 years ago. I focus on building a strong skilled team, with the right tools and developing the process to be able to answer our fuel cell development projects.

Anthony MICHEL – Applicative Software Manager

After several years within automotive safety SW development, I joined Symbio 3 years ago. My wish was to combine my knowledge of automotive software development to a cleaner mobility. I’m currently leading the industrialization of software development through Model Based Design.
1. Symbio company presentation

2. Development context & system presentation

3. Focus on function development with Model Based Design approach

4. Conclusion
Symbio company presentation
Hydrogen, a key lever
to meet today’s most compelling challenges
A brand new ecosystem under construction

HYDROGEN ECOSYSTEM

PRODUCTION

H2

PUBLIC SUPPORT
Regulation & funding

INFRASCTURE
Transport & Distribution

ECOLOGY
DECARBONATION

ZERO-EMISSION APPLICATIONS

ACCELERATING ZERO-EMISSION MOBILITY
Over 30 years of experience, Engaged builder of the hydrogen ecosystem

Entry in USA

2000

2010

2019

2020

2021

2022

2023

2024 to 2030...

100-500 Syst/Y

1 000-2 500 Syst/Y
Pilot plant in Vénissieux

15k-50k Syst/Y
Saint-Fons Gigafactory

50k-200K Syst/Y
Global footprint
Trusted H2 Partner
Development context & system presentation
H2 Fuel Cell system overview

- Expansion tank
- Hydrogen storage system
- ECU
- Charge air cooler
- Compressor
- Water pump
- Stack, fluid box and humidifier
- DCDC, High Voltage protection
- CAN bus
- Electric Powertrain, incl. battery
- Air filter
- Air intake
- Exhaust
- MT Radiator
- HT Radiator
- Elect. / Electronic heat loads
- Air intake
- Water pump
Our H2Motive range meets all power and durability needs of the automotive market

COMPELLING BENEFITS

- Zero-emission
- Fast refueling: 3-5 min
- Cruising range: up to 1000 km
**H2 Fuel Cell System development expectations**

- **Market hyper activities** request us to demonstrate the benefit of our systems thanks to prototypes & vehicles demonstrator in shortened development time.

- **Product scalability** to develop range of H2 fuel cell systems that meets all power and durability expectations.

- **Software function maturity & quality** to match hydrogen product specificities with automotive tier 1 standards.

![Diagram](image)
H2 Fuel Cell software development expectations

Automotive industries standards as guidelines and roadmap to confirm our Tier 1 position

➢ As influence for quality processes

➢ As a target for modularity

➢ As reference for safety processes
Focus on function development with Model Based Design approach
A system with large components diversity

Multi component system overview:

Fuel Cell System (FCS) ECU is our automotive grade ECU which intend to support various interfaces & communication protocols in order to control:

- Hydrogen Circuit
- Air Circuit
- Power circuit
- Cooling circuit
An intuitive & highly automated toolchain

supporting Software development in automotive normative environment
An intuitive & highly automated toolchain

supporting Software development in automotive normative environment

Purpose of functional design phase is to transform textual SW requirement into functional Simulink model

Textual requirements

Tips & highlights:

- Use template to uniformize development structure
- Use Simulink project to manage easily component initialization

Model design and its folder structure to welcome development workproducts
An intuitive & highly automated toolchain

supporting Software development in automotive normative environment

Purpose of modeling design checks phase is to perform static model analysis. It will prepare model to an easier & proper code generation.

Tips & highlights:
- Modeling Standards for ISO26262, MAAB, ...
- Custom Symbio checks to improve quality, best practice, readability, code customization

Model review report associated to SW Unit

Applicative SW Detailed Design pipeline

Customer Request → Backlog → SW Architecture → SW Detailed Design → SW Application Testing → SW Integration → SW Calibration → HIL Testing → System Testing

Modeling Guidelines Checks
Model Advisor

Simulink
An intuitive & highly automated toolchain
supporting Software development in automotive normative environment

Purpose of unit testing phase is to verify that simulation results are compliant to SW requirements

Tips & highlights:
- Open loop test harnesses creation on SW units
- Simulink test manager to optimize test reuse

SW Unit with functional validation report

Textual requirements

Create manually

Model Advisor

Test vectors database

Modeling Guidelines Checks

Unit MIL Testing

Applicative SW Detailed Design pipeline

Customer Request → Backlog → SW Architecture → SW Detailed Design → SW Application Testing → SW Integration → SW Calibration → HIL Testing → System Testing
An intuitive & highly automated toolchain

supporting Software development in automotive normative environment

Purpose of code generation phase is to transform functional SW Unit model into C-source code.

Tips & highlights:
- Highly customized code generated (class parameter, TLC files)
- Centralized embedded coder configuration

SW Unit code available and standardized between SW developers' laptops
An intuitive & highly automated toolchain
supporting Software development in automotive normative environment

Purpose of static analysis is to prove the absence of run-time errors in generated C source code

Tips & highlights:
- Manage SW unit complexity to make easier orange checks review
- Prepare your auto code generator to match MISRA C-2012 expectations

Static analysis report for MISRA compliancy and runtime error check

Customer Request Backlog SW Architecture SW Detailed Design SW Application Testing SW Integration SW Calibration HIL Testing System Testing

Applicative SW Detailed Design pipeline

Code Generation Unit SIL Testing Unit MIL Testing Modeling Guidelines Checks Simulink Test Simulink Test Embedded Coder Polyspace Code Prover Model Advisor Simulink Polyspace C code

Modeling Guidelines Checks

An intuitive & highly automated toolchain
supporting Software development in automotive normative environment

SW Application Testing

SW Detailed Design

SW Integration

SW Calibration

HIL Testing

System Testing

Polyspace Code Prover

Tips & highlights:
- Manage SW unit complexity to make easier orange checks review
- Prepare your auto code generator to match MISRA C-2012 expectations

Static analysis report for MISRA compliancy and runtime error check
An intuitive & highly automated toolchain
supporting Software development in automotive normative environment

Purpose of SW traceability is to ensure consistency between software requirements, models, generated code and associated test results.

Tips & highlights:
- **Refine requirements** and allocate it to SW units
- Import requirements into Mathworks environment with Simulink requirement

SW Unit with link between requirement, implementation & test artefacts

**Customer Request** → **Backlog** → **SW Architecture** → **SW Detailed Design** → **SW Application Testing** → **SW Integration** → **SW Calibration** → **HIL Testing** → **System Testing**
An intuitive & highly automated toolchain

supporting Software development in automotive normative environment

Purpose of SW Application testing process is to bring & link together SW units to assess functional behavior of a complete SW application.

Tips & highlights:
- Generate automatically SW integration model based on configurations files
- Try to maintain up to date connection between Applicative & Plant model

Confidence level into complete/partial SW application dedicated to project
An intuitive & highly automated toolchain

supporting Software development in automotive normative environment

CI system focus model's designers on function development instead of workproducts creation.

Pipeline report

Module | Analysis status | Metrics
--- | --- | ---
... | ... | ...

CI report generated with CI system

With automated & regular feedback along V-cycle activities
Focus CI system & automation process (1/2)

A partner associated to SW engineers

- What could be automatized?
  - Merge request creation with SW design metrics included
  - Model Advisor, Embedded Coder, Simulink test, Polyspace Code Prover, ...
  - All elements seen before (test reports, Modeling guideline review, static analysis report, ...) except functional design creation and part of SW traceability

Tips & highlights:
- Use GitLab CI to trigger jobs on impacted components only when files are pushed on server
- Use MathWorks docker image
- Produce reports and upload artifacts
Focus CI system & automation process (2/2)

A generated quality overview dashboard, providing **global metrics** on the SW platform.

### SW Components quality status

<table>
<thead>
<tr>
<th>Component</th>
<th>Model Advisor</th>
<th>Unit tests</th>
<th>Code gen.</th>
<th>SQO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reached</td>
</tr>
<tr>
<td>airBackPresCheck</td>
<td>Coverage 100%</td>
<td>Coverage 0%</td>
<td>Code Generation Failed</td>
<td>Code Priority SQO-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bug Finder SQO-9</td>
</tr>
<tr>
<td>airBackPresGuard</td>
<td>Coverage 100%</td>
<td>Unit Tests</td>
<td>Fail Run</td>
<td>Code Generation Failed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bug Finder SQO-4</td>
</tr>
<tr>
<td>airBackPresValvCtrl</td>
<td>Coverage 52%</td>
<td>Coverage 100%</td>
<td>Code Generation Failed</td>
<td>Code Priority SQO-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bug Finder SQO-3</td>
</tr>
<tr>
<td>airDilutionCheck</td>
<td>Coverage 97%</td>
<td>Coverage 100%</td>
<td>Code Generation Failed</td>
<td>Code Priority SQO-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bug Finder SQO-3</td>
</tr>
</tbody>
</table>

### Tips & highlights:

- **Use GitLab CI to trigger** **nightly jobs**
- **Use MathWorks docker image**
- **Produce reports and upload artifacts**

**Daily update** on all metrics

- Automatized on **main branches** for each repo
- Available for all **SW components** and **integrated projects**

_Nightly dashboard overview_
Conclusion
Faster time to market to match with H2 challenges & company roadmap

- Model-Based Design is particularly adapted to design complex systems such as fuel cell system, it allow to:
  - **Reduce need to access real systems** thanks to simulation approach
  - Embedded code is generated automatically from model, which **reduces effort** and **eliminates hand-coding errors**
  - **Stimulate innovation** thanks to possibility to try new ideas

- Model Based Design associated to Continuous Integration system will help to:
  - **Implement incremental workflow**, by testing design, refining, and retesting throughout the development process.
  - **Test and validate continuously** rather than at the end of the process so that many errors are found and corrected before system
  - **Improve system maturity & code quality** by focusing SW developer on function algorithms
And help company to execute a well-defined industrial roadmap

Investing in large state-of-the-art industrial facilities to ensure cost-competitiveness and service OEM’s growing volume needs

**2018-2020**
100-500 systems/year

**2021-2023**
1 000-2 500 systems/year
Fully automated line

**2024-2026**
15k-50k systems/year
Scalable manufacturing - Process reaching lower cycle time
Design to manufacturing, optimized with industrial process

**2027-2030**
50k-200K systems/year
Global production footprint
A leading hydrogen systems innovation partner for fuel cell solutions accompanying our customers …

…to accelerate zero-emission transportation world-wide