Challenge
Scaling agile model driven development of AUTOSAR embedded software. Lift the abstraction level of in-house development. Create reliable, automated test.

Solution
In-house model driven engineering (MDE)
Agile development enabled by plant models
Domain specific languages (DSL)
Scaling scripts? Scaling model architectures?

Results
- Increased development speed, but delayed testing
- Investments on plant models pay off!
- Physical modelling languages facilitates modelling and reuse, but still unclear scaleability (for virtual verification on system level)
About Volvo Car Group

- Mechanics (Mechatronics)
- Power Electronics
- Software

A trinity for sustainable automotive business!?

Volvo Car Group – Green, safe, premium cars!
Climat change…
Exponential increase of software in cars…
Modern cars are complex distributed software in moving, safety critical, (high voltage) mechatronics

About me

Technical Expert, Electric Propulsion Systems
Volvo Cars

- Quantum Physicist
- Teacher
- Technical Expert
- Continuous Integration
- Research collaboration
- Sw developer & change driver
at Volvo Cars -2013
Background
From requirement engineering to in-house software development!

From requirement engineering (with requirement as core object)

- Requirements (wishes, ideas, …)
- System design (architecture attempt)
- Detailed requirements (solution attempt, in word)
- Send to supplier
  - Wait...
- Test if it works (as you want).
Background
From requirement engineering to in-house software development!

... to in-house development, but still struggling with slow feedback

Lost knowledge if supplier is replaced!

ECU & SW
Supplier (Tier1)

Research

Development

Test vehicle

Virtual test (MIL/HIL)

Feedback!
Background
From requirement engineering to in-house software development!

... and aiming for Continuous Integration with control models as the core objects for development.

- High level requirements
- Executable model as core object!
- MIL, HIL test, integration
- Architecture (moderating role)
- Test vehicles
- ECU supplier
Model Driven Engineering

Approaching MDE

- Define modelling domains, build competence and knowledge
- Reach platform independent development (utilizing AUTOSAR)
- Combine “local” MDE with Virtual System Integration

1. System models
   - Power supply, Network,…
   - Realistic system test results, early

2. Platform models
   - Platform independence?
   - Supplier(s)

3. Plant & Environment models
   - Controller ECU
   - Device
   - Realistic functional test results, early
2 aspects of Model based/driven engineering

Early virtual integration and Development by Domain Experts

- High level Domain Specific [programming] Languages (DSL), like Simulink or Stateflow, will allow domain experts to develop code. A common way to start using MDE.

- Virtual integration requires Plant models and System models. Enable closed loop simulation and Model Driven Development

This require investments in tools and competence

The goal is to combine these techniques (and to add real continuous integration).

Note: Plant models are primarily useful where closed loops (controller-device/env.) exists.
The Model Driven short circuit

Re-invent the V-model!

Continuous Deployment short loop

High level reqs.

System

ECU

System design tool (database)

Architecture
Simulink

Simulink & Simscape

SW Design

Unit test

MIL (SIL)

RIG

CAR test

Vehicle integration

Developed by Tier1 from requirements

HIL short loop 24h

Plant models

System MIL
... and in real life

System design tool (database)

ECU ready!

Simulink & Simscape

Plant models

HIL (continuous ECU integration possible)

SW Component

SW Design

System

High level reqs.

Assumptions verified

HW Assumptions made

Vehicle integrations

Vehicle ready!

CAR test

RIG

Vehicle integrations

ECU ready!

ECU ready!

System

SW Component
Agile development

Pay to learn early
Gain knowledge early in projects to avoid workload peaks near deadlines!
Learn early by making plant models!

Model Driven Development – Create virtual verification environments!
Although the mechanics, ECU-platform and other mechatronics are not yet delivered, plant models can be designed – using the best knowledge – and used for early continuous virtual integration.

Some assumptions will be proven faulty, but we learn much faster (compared to not use MDE) (Ulf Eliasson et. al, to be published in MODELS 2014)
Scaling Model design

If we want to model more of the system—mechanics, electronics, fluid mechanics,…, then a single DSL (like standard Simulink) may not be sufficient.

A promising solution is to use different Domain Specific Languages (DSL) specialized for the different domains. Electric Propulsion at VCG has tried out Simscape now for 3 years. Other sections are using other languages.

Local DSLs will allow local organizations to develop faster with better reuse and understanding. The design scales!
DSL example: modelling mechatronics

Mathematical representation

\[ U_0 = V(t) + RI + L \frac{dI}{dt} \]

\[ \frac{dI}{dt} = \frac{1}{L} (U_0 - V(t) - RI) \]

Model representation in Simulink

Model representation in Physical DSL

Graphical representation

Model using c-code

```c
... 
real MyCircuit_dldt(t,l,V,R,L,U0) {
    return (U0 - V - R*I) / L;
}
/* Use with proper ODE solver */
for(t=0, t += dt, t < t_end) {
    ...
```
DSL example: modelling mechatronics

Then add one little component…
**DSL example: modelling mechatronics**

**Mathematical representation**
\[ U_0 = V(t) + RL_0 + LI_0 \]
\[ I_d = f(V(t)) \]
\[ I = I_0 + I_d \]
\[ \frac{dI_0}{dt} = \ldots \]

**Model representation in Simulink**

**Model representation in Physical DSL**

**Graphical representation**

**Controller (PWM)**

**V(t)**

**U_0**

**L**

**R**

**Rewrite completely**

> 2x no of blocks

**Rewrite completely**

> 2x lines of code

**Model using c-code**

```c
... 
real MyCircuit_dldt(t,I,V,R,L,U0) {
    ... 
    return (U0 - V - RI) / L;
    ...
}
/* Use with proper ODE solver */
for(t=0, t += dt, t < t_end) {
    ...
```
A use case: **developing dog clutch control software**

**Results:**
Although the first versions of the clutch model had numerous faulty assumptions, these were easy to correct – since the developers now understood the system!
The Importance of Testing

Some people believe that a graphical, high level control model, looking sort of like an executable specification, is so descriptive, so simple that no unit test or regression test is required.

Although complexity is hidden from the user in Simulink, it will be revealed in the generated code.

Unit Test, (functional) Regression Test and coverage metrics are absolutely essential!
Note about testing: use the same language!

We know from experience that developers prefer to conduct testing using the same language as they use for development.

Thus, if we develop control code using Simulink, we should develop test cases using Simulink – or in a tool integrated in Simulink with syntax familiar to Simulink users.

This turns out to be tricky…

**Formal verification looks very promising!**
However, Design Verifier (the dedicated toolbox for Simulink) should be integrated in the base product and used by developers. Why not fully integrate testing in the modelling language?

**Test vector design and regression test automation**
Tools for "Signal builder like" testing, but with assert handling
- yes/no returned for each test case!
- efficient test suite management, parameter set & test execution
- fast simulation (minimize compile time, parallel computing, cloud simulation, …) in normal mode (enabling coverage metrics)

**Static analysis**
Important, but not facilitated yet! Why not integrate this also? The language should help, not trust, the designer…
Scaling Model Driven Engineering! Full product simulation – in Simulink
Appealing idea, but comes with considerable challenges:
• Multi domain architecture (combine hw and sw architectures)
• Multi domain modelling (integrating electronics, cooling, data, mechanics…)
• Multiple DSLs has to be integrated (in a base simulink model)
Looking forward – approaching product based development

The virtual product
Automotive development is usually **project based**: massive parallel development and big bang integrations. Early system test is difficult, if possible at all.

Using large and partly auto-created Simulink models we can cheat and create a virtual product, which is ahead of the main project.

With an existing product (although virtual) new features can be faster(!) and distributed over time to avoid interference.
… and in real life

- When available – frequent (continuous) integration **on target**, with high coverage regression test is **essential**!
- Virtual test is for early learning, real test is for real knowledge! (Thus, HIL must be extended with real mechatronics…)
- MDE can be successfully combined with Continuous Integration, but tailor-made tools must be developed (**in-house scripts**)!
Conclusions

• Volvo Car Group has successfully combined Model Driven Engineering with methods of Agile Software development.
• Use cases demonstrate that MDE is an enabler for Agile in complex mechatronic systems.
• Continuous Integration (target builds, regression test) is working well, but suitable tools for testing and test automation in matlab/simulink are still missing.

• Physical Modelling using DSL (Simscape) has been used for several years in Hybrid system development.
• Physical Modelling is useful for readable, reusable and scalable modelling of physical systems, but challenging for large scale MIL testing (due to increased complexity and code generation issues)

• Current Challenge #1: The complete system MIL simulation; the virtual product testing, reaching Virtual Continuous Integration.
• Current Challenge #2: Enable platform independent and uniform-ish modelling in the R&D organization