Model-driven production software development for calibration

Using MATLAB, Simulink and Stateflow

Wouter van Heijningen and Koen van Wijk

ASML SW architect
ICT Motar

May 2021
Outline

• Introduction to Motar and ASML
• Calibration application development and challenges
• Model driven development approach
• Multidisciplinary collaboration
• Roadmap
• Conclusion
Motar, an ICT Group company

Motar is a lowcode platform, based on a MATLAB Simulink Stateflow environment.

We help companies to reduce the gap between prototype and production, which enables fast, flexible and model-based development integrated with large existing software. The models introduce a common language and a single source of truth.
ASML: enables semiconductor manufacturing

ASML provides wafer scanners for semiconductor manufacturers

Major product: Wafer Scanner
Moore’s law makes chips cheaper, smaller, faster

“The amount of transistors per given area doubles every 2 years at similar cost”

Our customers expect ‘Moore’:
- More transistors per cm²
- More wafers per day
- More machine availability
Increasing complexity of ASML machines

Many calibration applications needed to:
(1) achieve sub-nanometer precision…

…and (2) increasing scanner productivity to keep further shrink affordable
Calibrations are needed in many parts of the scanner.

- Reticle Stage
- Projection optics
- Reticle Handler
- Light Source, Dose control and illumination
- Environmental control
- Metrology
- Wafer Handler
- Measurement sensors and cameras
- Wafer stage
- System Dynamics
- Environmental control

Other domains

- Mechatronics
- Sensors and metrology (physics)

Public
Calibration, Performance and Diagnostics software

Software that is used to:

- Calibrate machine settings or
- Measure the Performance or
- Diagnose the machine

is called a CPD application

Properties:
- Several CPD applications per domain / subsystem
- Execute a series of actions though software in lower layers
- CPDs are not embedded software
- CPDs on machine’s central host, no direct hardware access
The developers of a CPD application

**Functional Engineer**
- Domain expert
- Physics or mechatronics or control background
- Often proficient in MATLAB
- Sometimes proficient in Simulink
- Rarely likes to deal with software details

**Software Engineer**
- Sometimes has domain knowledge
- Computer science or electronics background
- Mostly new to MATLAB
- Mostly new to Simulink
- The expert on software details

Challenge: Bridge domain knowledge gap
From waterfall to iterative co-development

Old way

Specification Documents
Design Documents
Implementation
Testing
Delivery

Model driven

Modeling & Simulation

CPD integration

Early integration

Functional Engineer
Software Engineer

Both engineers use MATLAB, Simulink and Stateflow to contribute to the model

Ready for integration
CPD decomposition

CPD model

Data

Annotations

Control Sequence

MATLAB Algorithms

Actions

ASML-specific add-ons to interact with the Scanner using Simulink S-functions

Engineers start with a template model

Single source of truth

Model = Documentation
Increasing maturity of single source of truth CPD

Convert using MATLAB/Simulink Coder

Model simulation
- Development environment
- CPD model
- Target system simulator

Rapid productizing
- Development environment
- CPD model
- Remote network connection
- Reuse & enhance
- Generated CPD from model (C++ code)

Fully integrated
- Target system
- Lower level software

Gradually increase application maturity and quality

Feasibility

Shipment
Demonstration: Rapid productizing
Model driven co-development of CPD applications

**Long term benefits:**
- Functional and SW engineer can read and contribute to the CPD model
- Less need for domain knowledge by software engineer
- Early feedback by rapid productizing and continuous integration facilities
- Reduced documentation effort
- Reduced development lead time

**Short term struggles:**
- New tool and way-of-working → learning curve
- Innovation vs. delivery pressure
Co-development: both parties can read/edit the model
Who does what can be different per domain

Functional engineer’s tasks

- Functional requirements
- Algorithms

Software engineer’s tasks

- Verifying selected interfaces with drivers
- Implement changes to real-time SW
- Integration in target framework
- Delivery to SW archive

Flexibility to assign

Functional engineer’s tasks

- Controlling sequence
  - Interfacing with lower-level software
    - (Automatic) Test cases and coverage measurement
      - Maturing and code generation
      - ...

Software engineer’s tasks

Agree responsibilities upfront
CPD development roadmap
Looking multiple years ahead

- Gradual rollout model-driven CPD development in ASML
  - Working development environment
  - 100+ engineers developing applications
  - Several applications delivered
- Expand application of MDE
- Model Template improvements
- Facilitate to convert legacy CPD to Simulink
- Support dynamic memory allocation
- Collaboration with MathWorks®

Requirements and document management
- Automated deployment
- Improve manuals and trainings
- Extend static model checking

Long term: increased usability and functionality empowers the functional engineer to create CPD applications with decreasing effort spent by the software engineer.
Conclusion / Take aways

1. At ASML, functional and software engineers create CPD applications together in a **common language**: MATLAB, Simulink, Stateflow

2. We gradually mature an application using a ‘**single source of truth**’ model including documentation

3. Providing the model development environment **direct remote access** to real machines enables early risk mitigation

4. Integration of MathWorks tooling and generated code within the ASML environment has been successfully made by Motar.