Public (approved by ASML TPB)

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

Public

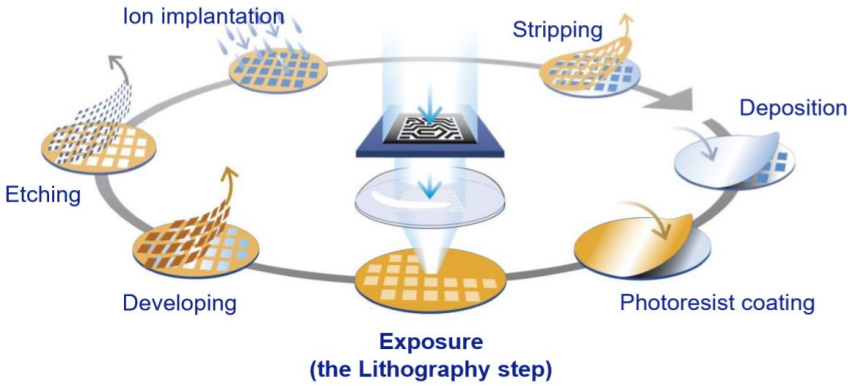
Motar, an ICT Group company

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.

Public

ASML: enables semiconductor manufacturing

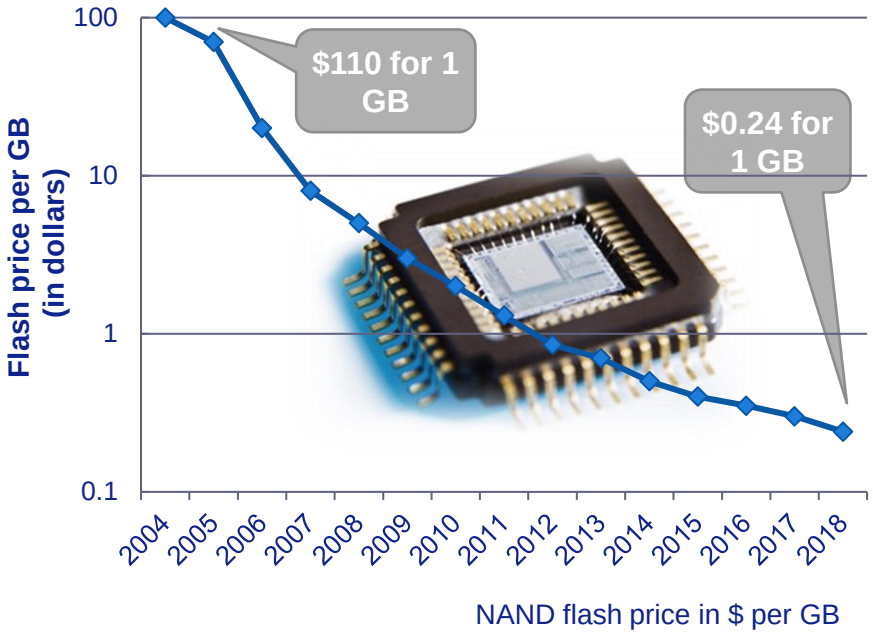


ASML provides wafer scanners for semiconductor manufacturers

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”



© www.dramexchange.com; Q1, 2018.

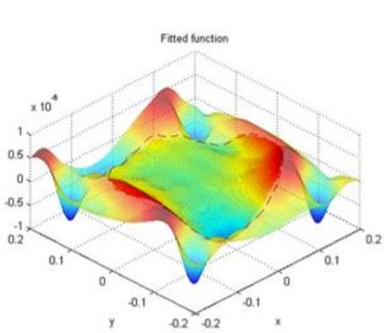
Our customers expect ‘Moore’:

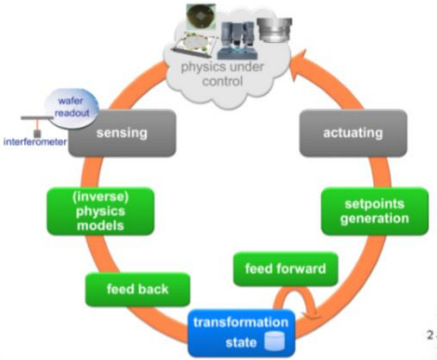
- More transistors per cm2
- More wafers per day
- More machine availability

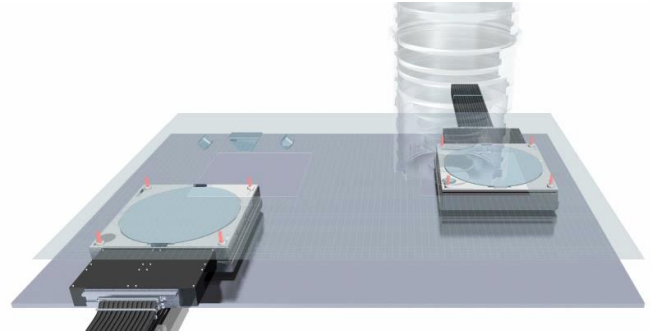
Increasing complexity of ASML machines

Imperfections

Variations







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

ASML

Reticle Stage

Projection optics

Reticle Handler

Light Source, Dose control and illumination

Metrology

Environmental control

Wafer Handler

System Dynamics

Measurement sensors and cameras

Wafer stage

Mechatronics
Sensors and metrology (physics)
Other domains

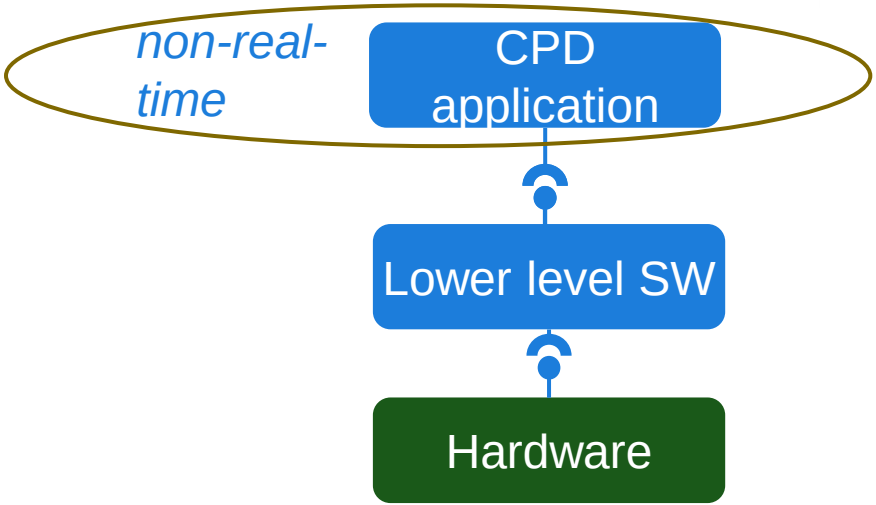
Public

Calibration, Performance and Diagnostics software

Software that is used to:

***C**alibrate machine settings or*
Measure the ***P**erformance or*
***D**iagnose* 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

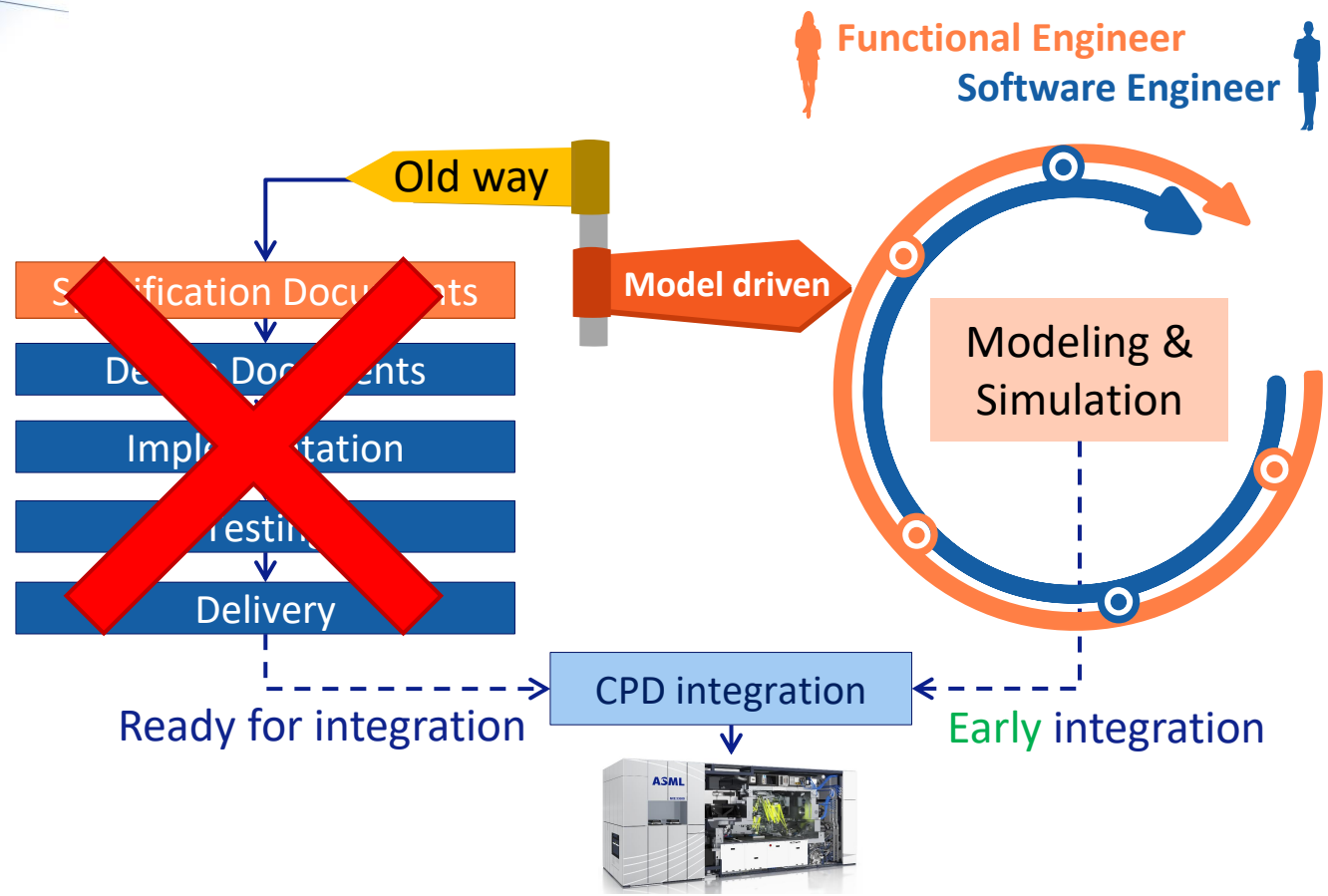
- Domain expert
- Physics or mechatronics or control background
- Often proficient in MATLAB
- Sometimes proficient in Simulink
- Rarely likes to deal with software details

Challenge:
Bridge domain knowledge gap

Software Engineer

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

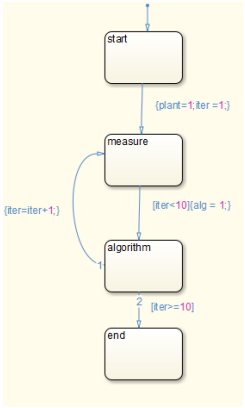
From waterfall to iterative co-development



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

CPD decomposition

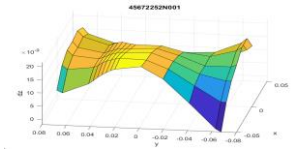
CPD model



Control Sequence

Data

Annotations



MATLAB Algorithms

```
function ye = kalmanf(A,B,C,Q,R,u,t,yv) %#eml
 P = B*Q*B';                    % Initial error covariance
 x = zeros(size(B));            % State initial condition
 ye = zeros(length(t),1);
 errcov = zeros(length(t),1);
for i=1:length(t)
   % Measurement update
   Mn = P*C'/(C*P*C'+R);
   x = x + Mn*(yv(i)-C*x);      % x[n|n]
   P = (eye(size(A))-Mn*C)*P;   % P[n|n]
   % Compute output
   ye(i) = C*x;
   errcov(i) = C*P*C';
   % Time update
   x = A*x + B*u(i);            % x[n+1|n]
   P = A*P*A' + B*Q*B';         % P[n+1|n]
end
```

Actions

Engineers start with a template model

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

Single source of truth

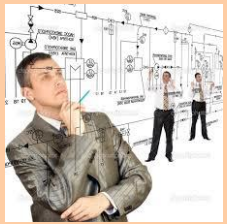
Model = Documentation

Increasing maturity of single source of truth CPD

Convert using MATLAB/Simulink Coder

Model simulation

Development environment



CPD model

MATLAB SIMULINK

Target system simulator

Reuse & enhance

Rapid productizing

Development environment

MATLAB SIMULINK

CPD model

Remote network connection

Target system

Lower level software

Fully integrated

Target system

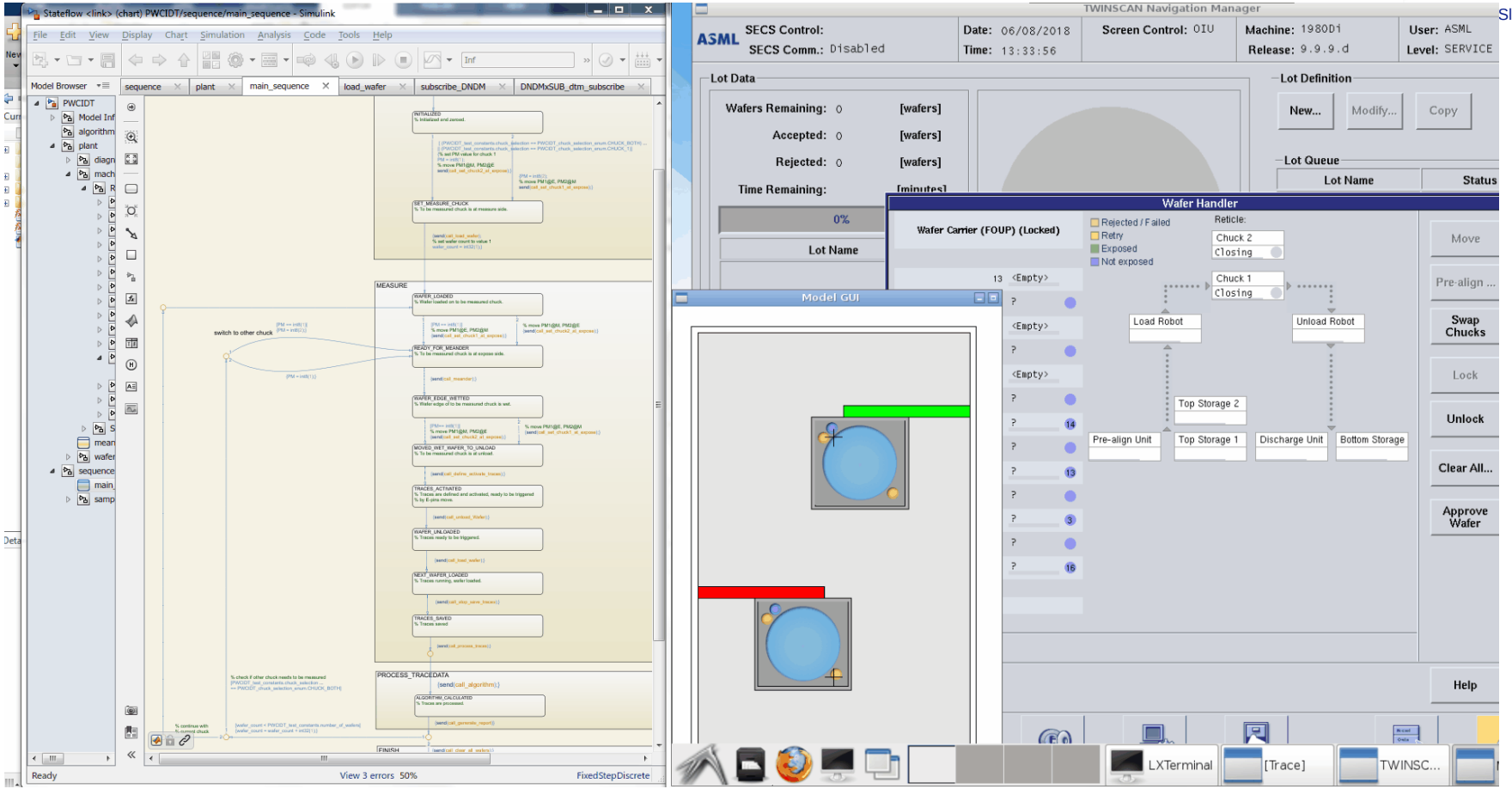
Generated CPD from model (C++ code)

Lower level software

Feasibility → Gradually increase application maturity and quality → Shipment

Public

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 of 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

Flexibility to assign

Software engineer's tasks

- Functional requirements
- Algorithms

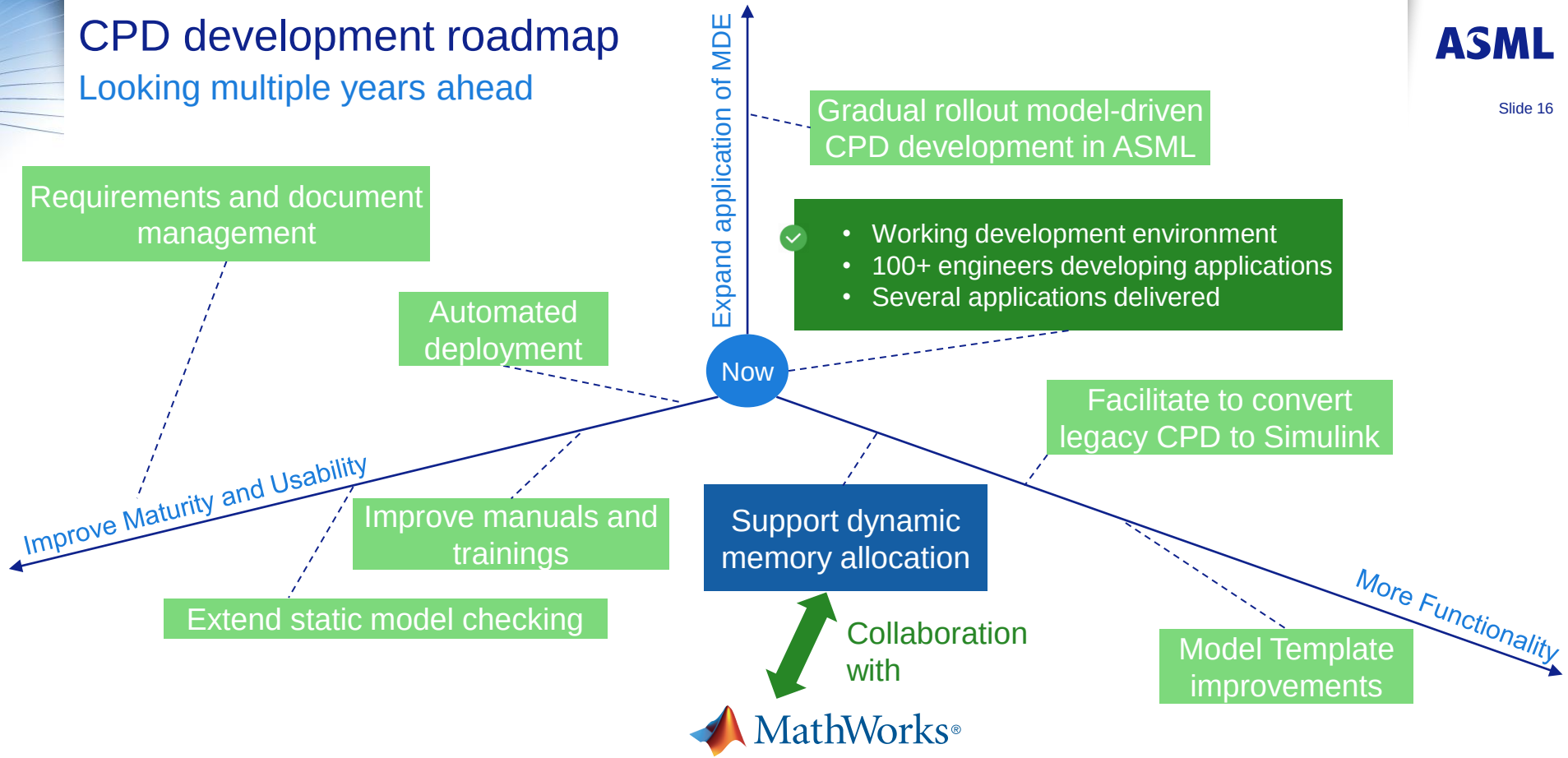
- Controlling sequence
  - Interfacing with lower-level software
    - (Automatic) Test cases and coverage measurement
      - Maturing and code generation
        - …

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

Agree responsibilities upfront

CPD development roadmap

Looking multiple years ahead



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.

Public

ASML