A look to the future with Model-Based Design

Andy Grace
Vice President of Engineering
Design Automation
MathWorks Today

3 million+ users in more than 180 countries

4500+ staff in 31 offices around the world

$1B+ in 2018 revenues with 60% from outside the US

Privately held and profitable every year
Technology Megatrends Driving Automotive

1. Vehicle Electrification
2. Autonomous Driving
3. Connected Vehicles

Software everywhere
Software is reshaping the automotive industry

In the future every company will become a software company

Marc Andreessen
Founder of Netscape,
Renowned Venture capitalist
Software is reshaping the automotive industry

- Augmenting control with machine learning (BMW)
- Trailer backup assist (Ford)
- Autonomous driving (Voyage)
Agile Values

Individuals & Interactions over Process and Tools

Customer Collaboration over Contract Negotiation

Working Software over Comprehensive Documentation

Responding to Change over Following a Plan

“While there is value in the items on the right, we value the items on the left more.”

- The Agile Alliance, 2001
Agile: Values, Principles and Practices

Agile is a mindset defined by values, guided by principles and manifested through many different practices. Agile practitioners select practices based on their needs.

~ Agile Practice Guide (PMI® and Agile Alliance®)
Typical agile development workflow
Models == Understanding
Simulation

Physical Prototyping
Simulation is key to Level 4-5 autonomy

Critical situations are in the long-tail*

Simulation helps achieve this improbable task

*Source: Center for Artificial Intelligence, Saarland University
Model-Based Design

Systematic use of models throughout the development process

Modeling

Simulation

Automation

Coding

Verification

Fast repeatable tests

Fast agile development loops
Types of models

Systems

Software

Physics

Components
Physical components

Vehicle Component

Sensor Model

Communications Channel

Motor
Simscape for physical modeling

Publication-quality diagrams

Simscape modeling language

Models just run
Types of models

Systems

Software

Components

Physics
Simulink as an Integration Platform
Simulink as an Integration Platform
Simulation Integration: Infrastructure

Data Management
Solver Technology
Vehicle Configuration
Multi-actor Scenarios
Visualization
Simulation Integration: Analyses

Verification and Validation
- Fuel Economy

Design Optimization
- Performance

Sensitivity Analysis
- Energy Consumption

Virtual Calibration
- Drivability
- Ride & Handling

• Pure EV (will update graphics)
• Vehicle Dynamics (will update graphics)
• Hybrid EV (will update graphics)
• Automated Driving (will update graphics)
Scaling up simulations

X 1,000,000’s

Parallel simulations

Simulation Manager

Programmatic test creation

```matlab
for i = 1:10000
    in(i) = Simulink.SimulationInput('my_model');
    in(i).setVariable('my_var', i);
end
out = parsim(in);
```

>> scenario = drivingScenario
“A typical ECU contains 2000 function components that each are developed by a different person.”
Working at a high-level of abstraction

SOFTWARE COMPONENTS

MATLAB

Simulink

Stateflow
Component modeling

Reusable components that can be adapted to any software system

Startup and shutdown behavior

Variant management
Types of models

- Systems
- Software
- Physics
- Components
System architecture is the #1 topic

**Breakout Topic Requests (2018)**

- Modeling System Architecture: 75
- Sensor Fusion and Tracking: 64
- Customizing Embedded Coder: 56
- Testing Simulink Models: 55
- Efficiency of Generated Code: 51

**Feature Prioritization (2017)**

- System Architecture: 173
- Code Generation: 167
- Large-scale Modeling: 123
- Verification & Validation: 106
- Improved UI: 103
Systems engineering

Requirements

Systems

Components
Systems engineering

Requirements

Components
Linking top-down and bottom-up workflows
Types of models

Systems

Software

Physics

Components
Deep solutions
Deep solutions

Automotive Products

- Powertrain
- Vehicle
- Automated Driving
- Calibration
Automotive Reference Applications

**Pure EV**

**Lane Keeping Assist**

**Hybrid Powertrain**

**Car Vehicle Dynamics**
Deep solutions for autonomous systems

SLAM (18a)
Robotics System Toolbox

Perception

Localization

Planning

Control

Path Planning (19a)
Automated Driving Toolbox

Semantic Segmentation (17b)
Automated Driving System Toolbox

Adaptive Cruise Control (17a)
Automated Driving System Toolbox
Deep solutions for autonomous systems

Lane Keep Assist
Model Predictive Control

Automatic Emergency Braking
Automated Driving Toolbox
MATLAB Workflow for Deep Learning:

**Access Data**
- Access Data
- Preprocess
  - MUNGING/LABELING
  - FUSION
  - DENOISING
- Access Models
  - BUILD
  - BORROW
- Train
  - FROM SCRATCH
  - TRANSFER
- Deploy

Deep Learning Toolbox
Create, analyze, and train deep learning networks

**Interoperability with open source networks**
- ONNX
- PyTorch
- mxnet
- TensorFlow

**Deep Network Designer App**
- Network training performance

**Inference performance**
- NVIDIA

**Deployment support**
- Intel
- ARM

Domain-specific workflow support
Ground truth labeling apps for:
- Video
- Audio
- application-specific datastores
Artificial Intelligence for your applications

- Application examples

  - Object Detection Using Deep Learning
  - Traffic Sign Detection and Recognition
  - Pedestrian Detection
  - Detecting Cars Using Gaussian Mixture Models
  - Tracking Pedestrians from a Moving Car
  - Waveform Segmentation using Deep Learning
Artificial Intelligence for your applications

- Application examples
- Control design

Reinforcement Learning Toolbox
```c
#include "AutomatedParkingPitAlgorithm.h"
#include "AutomatedParkingPitAlgorithm_private.h"

int32_T div_s32_floor(int32_T numerator, int32_T denominator)
{
    int32_T quotient;
    uint32_T absNumerator;
    uint32_T absDenominator;
    uint32_T tempAbsQuotient;
    boolean_T quotientNeedNegation;
    if (denominator == 0) {
        quotient = numerator > 0 ? MIN_int32_T : MAX_int32_T;
    }
    else {
        absNumerator = numerator < 0 ? static_cast<int32_T>(numerator) + 1 : static_cast<int32_T>(numerator);
        absDenominator = denominator < 0 ? static_cast<int32_T>(denominator) + 1 : static_cast<int32_T>(denominator);
        quotientNeedNegation = (numerator < 0) != (denominator < 0); // Divide by zero handler
        if (quotientNeedNegation) {
            absNumerator = absDenominator;
            if (absNumerator > 0) {
                tempAbsQuotient = absNumerator / absDenominator;
                if (tempAbsQuotient > 0) {
                    return absNumerator / absDenominator;
                } else {
                    return absNumerator / absDenominator;
                }
            }
        }
        if (quotientNeedNegation) {
            return static_cast<int32_T>(tempAbsQuotient) + static_cast<int32_T>(tempAbsQuotient);
        }
    }
    return quotient;
}

void AutomatedParkingPitModelClass::AV_enuInit_real_T(enuArray_T_REAL_T *enuArray,
                                                      int32_T numDimensions)
```
Solutions for **Vision and Deep Learning**

**GPU**  
Fastest

**FPGA / ASIC**  
Lowest Power

**CPU**  
Low Cost
Model-Based Design vs C/C++

- High level of abstraction
- Advanced analysis tools
- Automatic code generation
Model-Based Design

- No wrappers
- No data typing
- No data copies

C/C++ Libraries

COD GENERATION
Model-Based Design

- No wrappers
- No data typing
- No data copies

C/C++ Libraries

Middleware

- ROS
- AUTOSAR
- DDS

Hand Code

Internal Libraries

Vendor Libraries
Modeling

Simulation

Automation

Coding

Verification
Automated Test and Verification

- Find bugs
- Manage tests
- Check & Coverage
- Inspect code

Simulink Design Verifier Polyspace Bug Finder
Simulink Test
Simulink Check Simulink Coverage
Simulink Code Inspector
Online Access for Test and Verification

CONTINUOUS INTEGRATION

AUTHENTICATION

Bug TRACKING

DATA STORAGE

Web browser

Polyspace Server R2019a

Polyspace Access
Model-Based Design

Systematic use of models throughout the development process

Modeling

Simulation

Automation

Coding  Verification

Fast repeatable tests

Fast agile development loops
Who will be successful in the future?

Mechanical-centric

Model-centric

Software-centric

Comprehensive models
Simulation based testing
Generate code and automate verification
Enjoy the conference