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

This week, Hewlett-Packard (where I am on the board) announced that it is exploring jettisoning its struggling PC business in favor of software and services, an opposite move to Apple’s focus on hardware. It is an interesting and important shift for a company that has been around for 65 years and can claim to have created a personal computer that sits on a desk. In the future, every company will become a software company. The personal computer will give way to the personal computer in the cloud. It is just beginning.
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

1-4 Weeks
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
Physical components

- Vehicle Component
  ![Vehicle Dynamics Diagram]

- Sensor Model
  ![Sensor Model Image]

- Communications Channel
  ![Communications Channel Diagram]

- Motor
  ![Motor Image]
Simscape for physical modeling

Publication-quality diagrams

Simscape modeling language

Models just run
Types of models
Simulink as an Integration Platform
Simulink as an Integration Platform
Simulation Integration: Infrastructure

Data Management  
Solver Technology  
Vehicle Configuration  
Multi-actor Scenarios  
Visualization

Simulink
Simulation Integration: Analyses

- Verification and Validation
- Design Optimization
- Sensitivity Analysis
- Virtual Calibration

- Fuel Economy
- Performance
- Energy Consumption
- Drivability
- Ride & Handling
Scaling up simulations

X 1,000,000’s

Parallel simulations

Simulation Manager

Programmatic test creation

for i = 1:10000
    in(i) = Simulink.SimulationInput('my_model');
    in(i).setVariable('my_var', i);
end
out = parsim(in);
“A typical ECU contains 2000 function components that each are developed by a different person.”
Working at a high-level of abstraction

SOFTWARE COMPONENTS
Component modeling

Reusable components that can be adapted to any software system

Startup and shutdown behavior

Variant management
Types of models
System architecture is the #1 topic

Breakout Topic Requests (2018)

<table>
<thead>
<tr>
<th>Topic</th>
<th>Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling System Architecture</td>
<td>75</td>
</tr>
<tr>
<td>Sensor Fusion and Tracking</td>
<td>64</td>
</tr>
<tr>
<td>Customizing Embedded Coder</td>
<td>56</td>
</tr>
<tr>
<td>Testing Simulink Models</td>
<td>55</td>
</tr>
<tr>
<td>Efficiency of Generated Code</td>
<td>51</td>
</tr>
</tbody>
</table>

Feature Prioritization (2017)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Architecture</td>
<td>173</td>
</tr>
<tr>
<td>Code Generation</td>
<td>167</td>
</tr>
<tr>
<td>Large-scale Modeling</td>
<td>123</td>
</tr>
<tr>
<td>Verification &amp; Validation</td>
<td>106</td>
</tr>
<tr>
<td>Improved UI</td>
<td>103</td>
</tr>
</tbody>
</table>
Systems engineering

Requirements

Systems

Components
Systems engineering

Requirements

Components
Linking top-down and bottom-up workflows

SOFTWARE & SYSTEM ARCHITECTURE
Types of models
Deep solutions

Controls

Signal Processing

Wireless

Vision

Robotics
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

Location

Perception

Planning

Control

SLAM (18a)
Robotics System Toolbox

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**
- MUNGING/LABELING
- FUSION
- DENOISING

**Preprocess**
- BUILD
- BORROW

**Access Models**
- 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**

**Inference performance**
- NVIDIA

**Domain-specific workflow support**
Ground truth labeling apps for:
- Video
- Audio
- Application-specific datastores

**Network training performance**
- NVIDIA GPU Cloud
- Azure

**Deployment support**
- Intel
- ARM

© 2019 The MathWorks, Inc.
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
/* Function: div_s32_flow (int32_T numerator, int32_T denominator)

   Description:
   Divides two 32-bit integers using a custom division algorithm.

   Parameters:
   - numerator: The numerator of the division.
   - denominator: The denominator of the division.

   Returns:
   The quotient of the division.

   Example:
   int32_T quotient;
   int32_T absNumerator;
   int32_T absDenominator;
   int32_T tempAbsQuotient;
   boolean_T quotientNeedNegation;

   if (denominator == 0) {
       quotient = numerator; /* H5: int32_T : MUB_int32_T;
   }
   else {
       absNumerator = numerator < 0 ? static_cast<int32_T>(-numerator) : numerator;
       absDenominator = denominator < 0 ? static_cast<int32_T>(-denominator) : denominator;
       quotientNeedNegation = (numerator < 0) != (denominator < 0);
       tempAbsQuotient = absNumerator / absDenominator;
       if (quotientNeedNegation) {
           tempAbsQuotient = -tempAbsQuotient;
       }
       quotient = tempAbsQuotient * static_cast<int32_T>(tempAbsQuotient) / static_cast<int32_T>(tempAbsQuotient);
   }
   return quotient;
*/

void AutomatedParkingPileupAlgorithm() {
    // Implementation details...
}
```
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
Model-Based Design

- No wrappers
- No data typing
- No data copies

C/C++ Libraries

Hand Code

Internal Libraries

Vendor Libraries

Middleware

ROS

AUTOSAR

DDS
Automated Test and Verification

- **Find bugs**
  - Simulink Design Verifier
  - Polyspace Bug Finder

- **Manage tests**
  - Simulink Test

- **Check & Coverage**
  - Simulink Check

- **Inspect code**
  - Simulink Code Inspector
Online Access for Test and Verification
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