MATLAB for Software Development and System Design

Andy Thé
Technical Marketing
Image Processing Applications
Typical Development Workflow
MATLAB for Rapid Simulation
Moving from Simulation to Prototype and Development

- Algorithm and application development often use different groups, toolset, languages
- Long development cycle with many opportunities for errors
Development Workflow with MATLAB

Simulate & Prototype

Visualize
Experiment
Test

...
Demo 1: Visualizing, Prototyping, and Testing from Visual Studio with a Gaussian Filter
MATLAB - Framework for C/C++ Development

PROTOTYPING, TESTING, and VERIFICATION
Summary Demo 1: Visualizing, Verifying, and Prototyping

- Create a powerful C/C++ development & test harnesses
  - Simulate I/O
  - Visualize your data (printf on steroids)
  - Test & Validate to downstream minimize errors

- Access MATLAB functions to quickly prototype

- Technologies include:
  - Visual Studio or Eclipse
  - MATLAB Engine
  - Unit Testing Framework
Development Workflow with MATLAB
MATLAB for System Design

- Simulate with existing code to maximize reuse
- Prototype with proven code to lower risk
Demo 2: Integrate, Simulate, and Prototype with existing C code with MATLAB
Summary Demo 2: System design by leveraging existing code

- Replace MATLAB functions with your custom C code
- Create prototypes with proven code
- Maximize reuse and minimize risk
  - Generate code that leverages existing C code
- Technologies include:
  - LoadLibrary
  - coder.ExternalDependency
  - MEX
Reduce Overall Development Time
(Improve Quality, Reduce Re-spins, etc.)

Time spent in project phases

With MathWorks Tools

Without MathWorks Tools

- Requirements
- System Design
- Implementation
- Integration
- Testing
Using MATLAB for software development enables you to...

- Simulate I/O, Visualize, and Test C/C++ code
- Rapidly prototypes functionality
- Generate C Code to seed development
- Integrate proven code into MATLAB to seed next generation designs
- Deploy designs with more confidence and less time
Thank you

MATLAB for Software Development and System Design
Agenda

- Introduction to MATLAB Overview
- MATLAB for Development & Design
- Simulink for Development & Design
- User Stories
- Summary and Wrap-up
The leading environment for modeling, simulating, and implementing communications systems and semiconductors

- Foundation for Model-Based Design
- Digital, analog, and mixed-signal systems, with floating- and fixed-point support
- Algorithm development, system-level design, implementation, and test and verification
- Optimized code generation for FPGAs and DSPs
- Blocksets for signal processing, communications, video and image processing, and RF
- Open architecture with links to third-party modeling tools, IDEs, and test systems
Simulink Key Features

- MATLAB® integration
- Hierarchical, component-based modeling
- Custom code integration and management
- Custom board integration
- Extensive library of predefined blocks
- Application-specific libraries available
- Open Application Program Interface (API)
Simulink for Software Development and System Design

- Are you working on a complex embedded system?

- Do you want to integrate existing code into your designs?
  - ARM, x86/x64, DSP, HDL, GPU...

- Do you have challenges with build, test, and validation of algorithms?

- Do you want to rapidly prototype algorithms on hardware?

- Do you practice Agile Software Development?
Simulink Enables:

- A single common graphical environment for capturing, executing ideas, and collaboration.

- Rapid prototyping environment from MATLAB - Hardware In the Loop (HIL) testing on embedded processors
  - i.e. ARM, x86, FPGA, DSP…

- Common environment for development and testing of the entire design, reduce transition introduced defects
  - “build a little / test a little” workflows
    - i.e. Vision + Camera + Controls + Signal Processing + etc.
Simulink enables:

- Integration of existing code directly into the prototype
  - i.e. Blocks with ARM, FPGA, or DSP code

- Dynamic/executable models allow for rapid evaluation of requirements changes

- Code generation for rapid prototyping and software deliveries
Using Simulink for Embedded Design

- Analyze trade-offs
- Rapidly prototype ideas
- Integrate Existing Code
- Generate C-code
- Implicit timing and concurrency
- Continuous verification
- Target embedded devices

MathWorks®
Model-Based Design & Agile Development
The Value of Model-Based Design

Model-Based Design
- Executable specification
- Design with simulation
- Implementation through code generation
- Continuous test and verification

Innovation
- Rapid design iterations
- “What-if” studies
- Unique features and differentiators

Quality
- Reduce design errors
- Minimize hand coding errors
- Unambiguous communication internally and externally

Cost
- Reduce expensive physical prototypes
- Reduce re-work
- Automate testing

Time-to-market
- Get the product right the first time
## Agenda

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to MATLAB Overview</td>
</tr>
<tr>
<td>MATLAB for Development &amp; Design</td>
</tr>
<tr>
<td>Simulink for Development &amp; Design</td>
</tr>
<tr>
<td>User Stories</td>
</tr>
<tr>
<td>Summary and Wrap-up</td>
</tr>
</tbody>
</table>
Centre for Concepts in Mechatronics Improves Resolution of Agfa Printers Using MathWorks Tools for Model-Based Design

Challenge
Improve image resolution on industrial inkjet printers

Solution
Use MathWorks tools for Model-Based Design to design and automatically generate code for a new control system that enables increased image resolution

Results
- Design iterations completed in minutes
- Functional performance improved by a factor of five
- Hardware-independent solution implemented

“Using MathWorks tools, we can realize our implementations directly, reducing costs and saving up to four weeks of time.”
Arend-Jan Beltman
CCM

Link to user story
Challenge
Develop control systems for a two-armed mobile humanoid robot with 53 degrees of freedom

Solution
Use Model-Based Design with MATLAB and Simulink to model the controllers and plant, generate code for HIL testing and real-time operation, optimize trajectories, and automate sensor calibration

Results
- Programming defects eliminated
- Complex functionality implemented in hours
- Advanced control development by students enabled

“Model-Based Design and automatic code generation enable us to cope with the complexity of Agile Justin’s 53 degrees of freedom. Without Model-Based Design it would have been impossible to build the controllers for such a complex robotic system with hard real-time performance.”

Berthold Bäuml
DLR
Volvo Construction Equipment Streamlines Product Development with a Real-Time, Human-in-the-Loop Simulator

**Challenge**
Evaluate design concepts and parameter values for construction equipment before building physical prototypes

**Solution**
Use Simulink, Simscape, and Simulink Real-Time to model hydraulic, mechanical, and engine systems and perform real-time, operator-in-the-loop simulations

**Results**
- Number of prototypes reduced
- Issues in the field resolved faster
- Controller tuned in simulation

Link to user story

“How was technically impossible for us to build a full-scale hydraulic system model to run in real time without Simulink, Simscape, and Simulink Real-Time. Our simulator enables us to test new concepts for construction equipment, tune parameters, reduce lead times, and minimize issues in the field.”

Jae Yong Lee
Volvo Construction Equipment
<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to MATLAB Overview</td>
</tr>
<tr>
<td>MATLAB for Development &amp; Design</td>
</tr>
<tr>
<td>Simulink for Development &amp; Design</td>
</tr>
<tr>
<td>User Stories</td>
</tr>
<tr>
<td>Summary and Wrap-up</td>
</tr>
</tbody>
</table>
What is in the OpenCV Interface Support Package*?

- Custom MEX build command (mexOpenCV)
  - Links against pre-built OpenCV libraries
  - Accepts all standard MEX flags
- Data type conversions (opencvmex.hpp)
  - Conversions for all commonly used data conversions
- Examples available
  - Normalized cross correlation
  - Geometric transform estimation
  - Background subtraction

* Requires Computer Vision System Toolbox
Things to Consider When Using OpenCV with MATLAB

- Nearly all algorithms are in MATLAB or toolboxes
  - Algorithms not available are often very recent
- MATLAB performance is competitive
  - Sometimes MATLAB is faster, sometimes OpenCV is faster
- Use our build of OpenCV
  - Avoid library and build incompatibilities
  - Save time and effort as we’ve figured out the best build options
Getting Started

- MATLAB Engine – APIs and Documents

- Setting up Visual Studio, Eclipse, or Xcode

- LoadLibrary

- Coder.ceval
Questions?
# MATLAB for Software Development

## Why? How? What?

<table>
<thead>
<tr>
<th>Why?</th>
<th>How?</th>
<th>What?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce costly errors during development</td>
<td>Visualization data directly from Visual Studio (Any IDE)</td>
<td>MATLAB + Visual Studio / Eclipse</td>
</tr>
<tr>
<td>Rapidly prototype missing functionality during development</td>
<td>Call MATLAB functions directly from Visual Studio</td>
<td>MATLAB + IPT + CVST Visual Studio / Eclipse</td>
</tr>
<tr>
<td>Build a testing framework for C/C++ development</td>
<td>Call MATLAB functions / unit tests directly from Visual Studio</td>
<td>MATLAB + Visual Studio / Eclipse</td>
</tr>
<tr>
<td>Leverage existing code bases</td>
<td>Call existing code and libraries from MATLAB</td>
<td>MATLAB Visual Studio / Eclipse</td>
</tr>
<tr>
<td>Target embedded vision systems</td>
<td>Manage algorithms in MATLAB and generate C code</td>
<td>MATLAB Coder, Fixed Point Designer Visual Studio / Eclipse</td>
</tr>
</tbody>
</table>