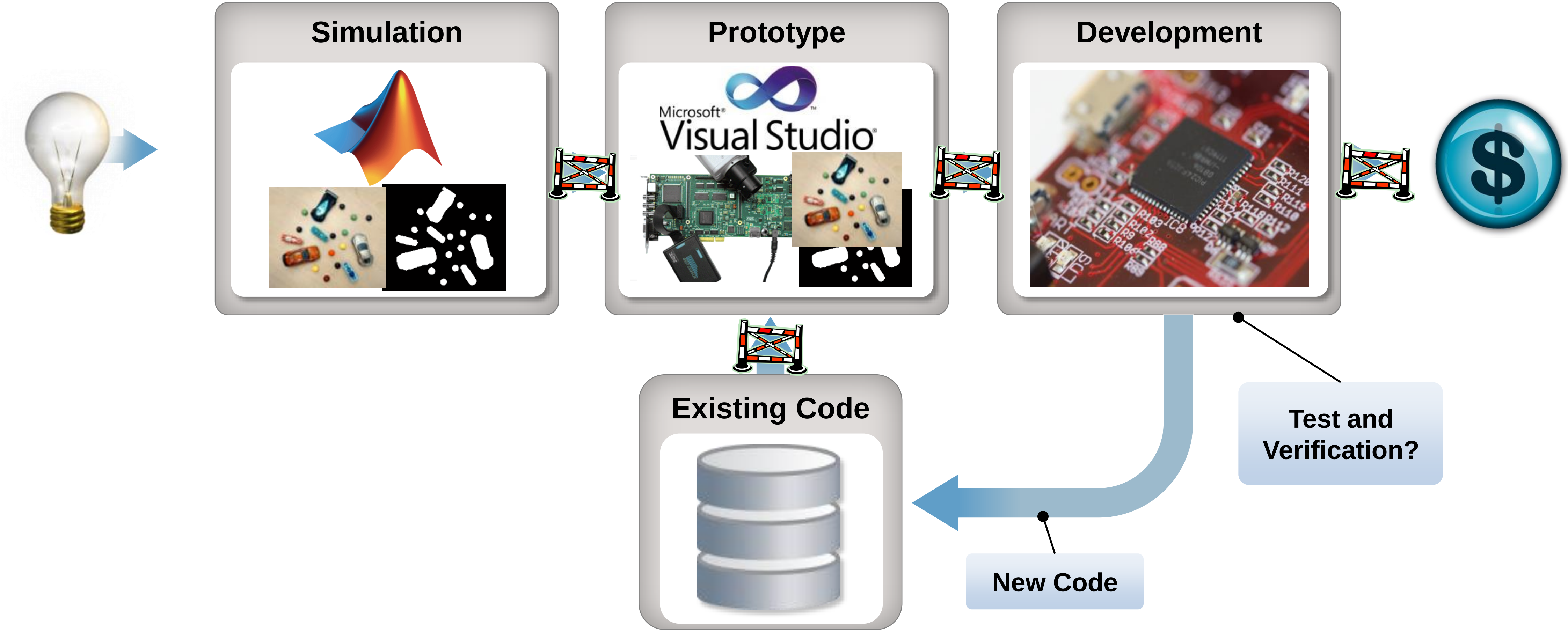
MATLAB for Software Development and System Design

Andy Thé
Technical Marketing
Image Processing Applications

Typical Development Workflow



Simulation

Prototype

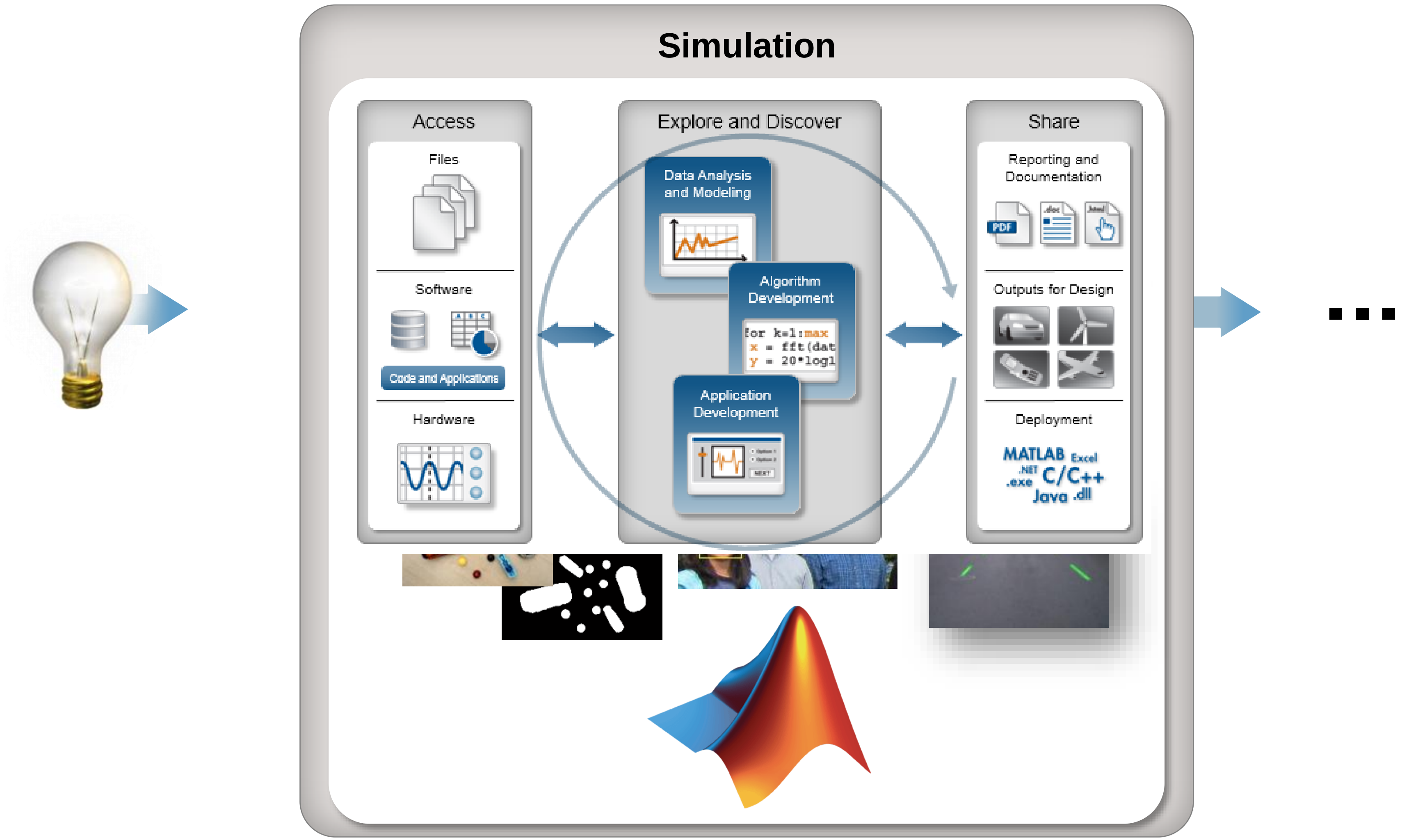
Development

Existing Code

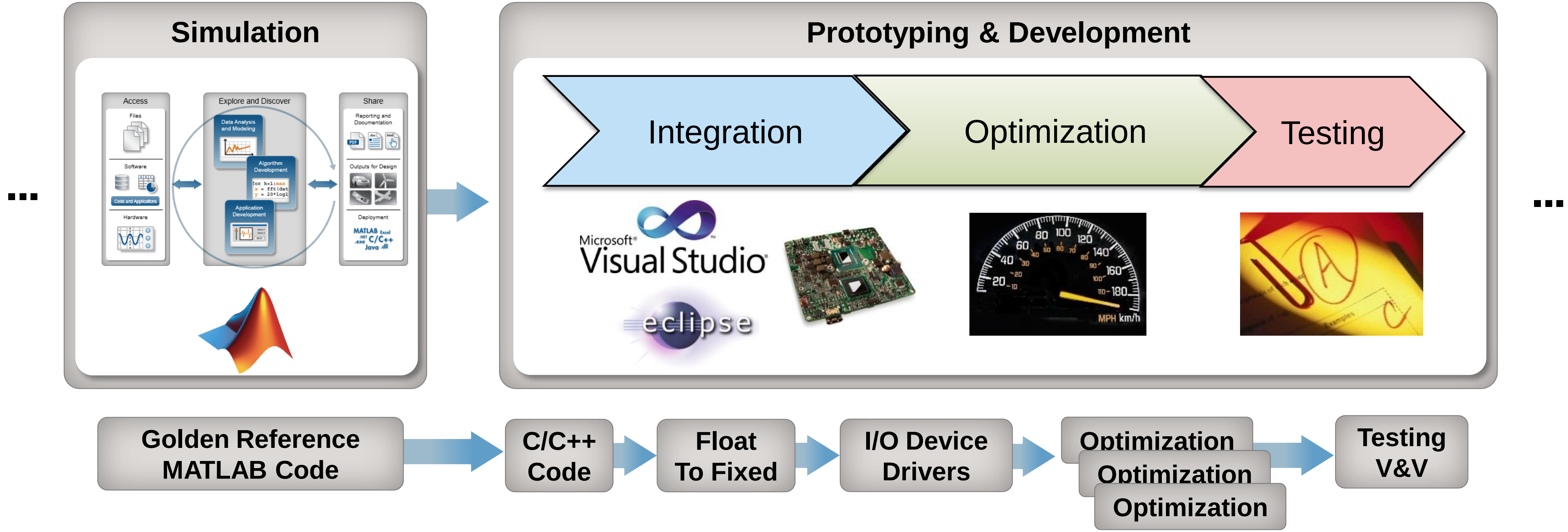
New Code

Test and Verification?

MATLAB for Rapid Simulation



Moving from Simulation to Prototype and Development



...

Simulation

Prototyping & Development

Integration

Optimization

Testing

...

Golden Reference MATLAB Code

C/C++ Code

Float To Fixed

I/O Device Drivers

Optimization

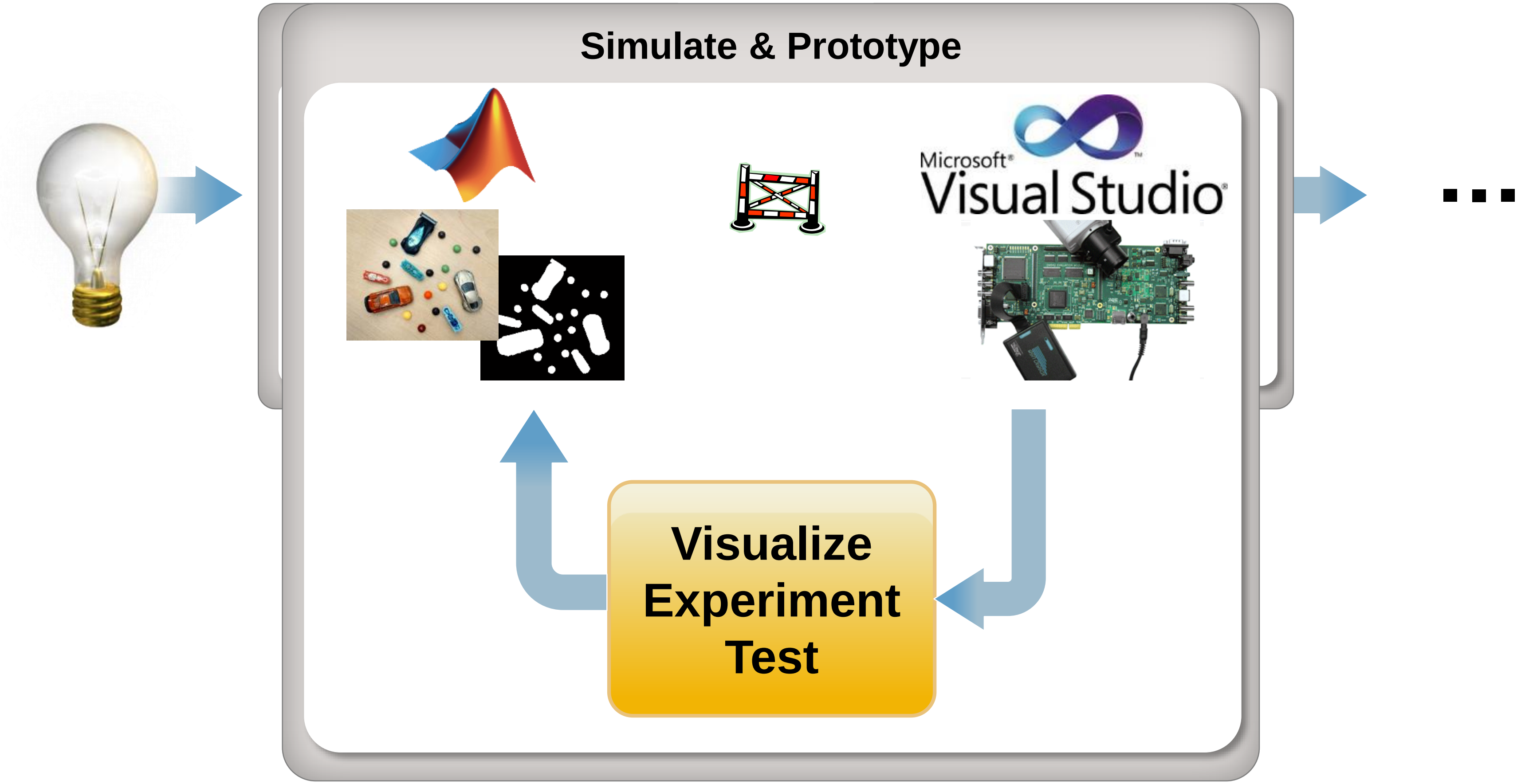
Optimization

Optimization

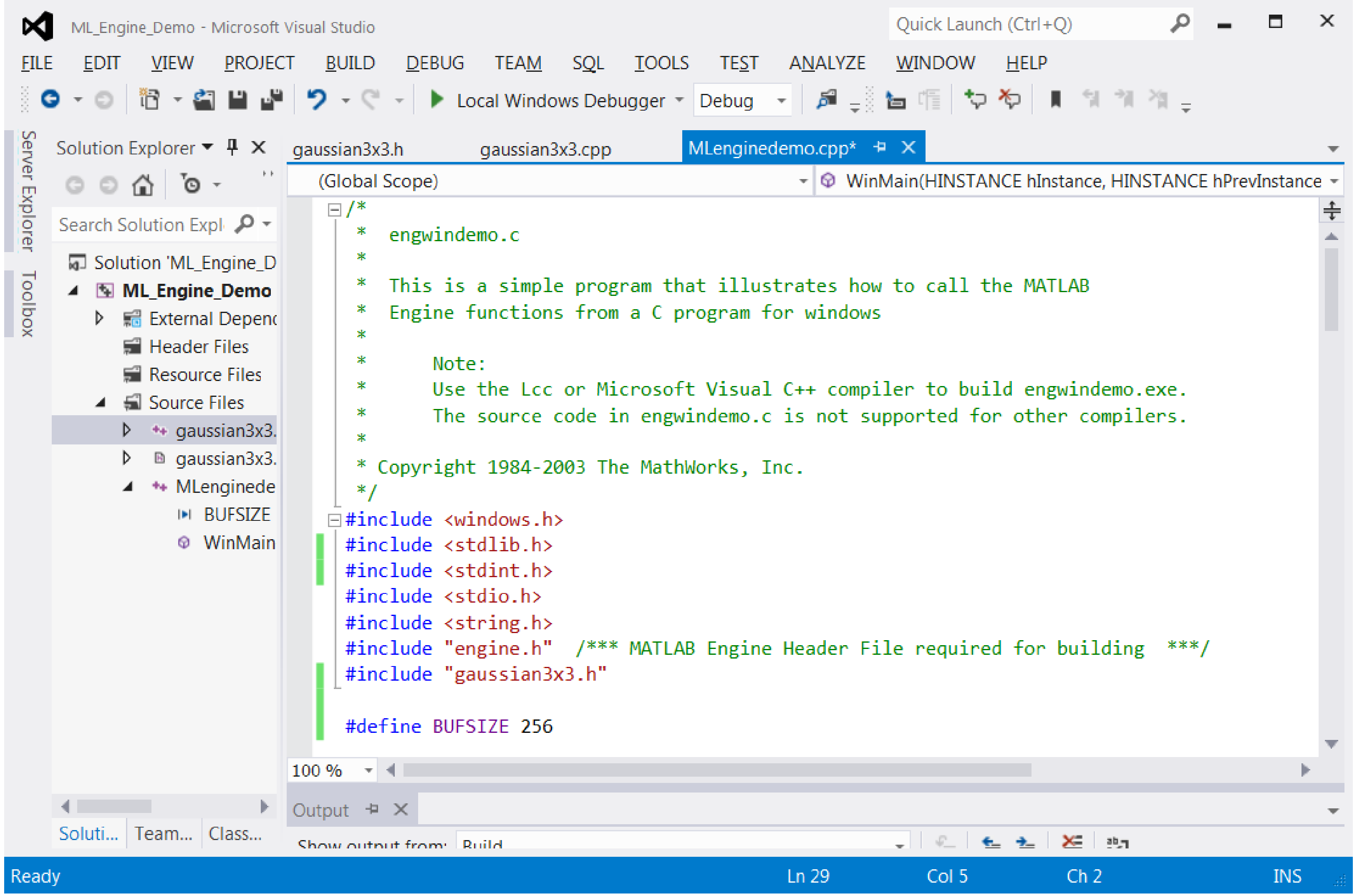
Testing V&V

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

Development Workflow with MATLAB

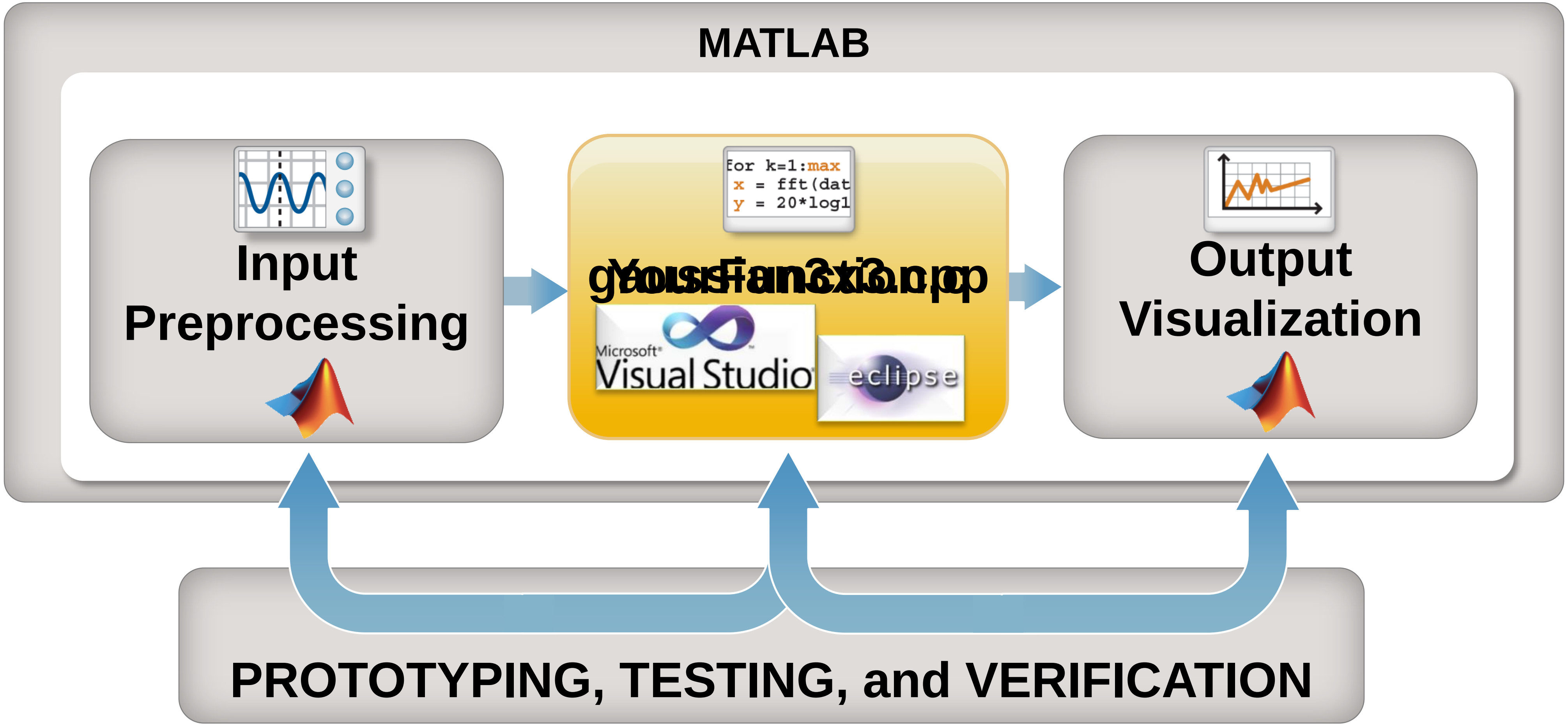


Demo 1: Visualizing, Prototyping, and Testing from Visual Studio with a Gaussian Filter



```
/*
 *  engwindemo.c
 *
 *  This is a simple program that illustrates how to call the MATLAB
 *  Engine functions from a C program for windows
 *
 *      Note:
 *      Use the Lcc or Microsoft Visual C++ compiler to build engwindemo.exe.
 *      The source code in engwindemo.c is not supported for other compilers.
 *
 * Copyright 1984-2003 The MathWorks, Inc.
 */
#include <windows.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include "engine.h"   /*** MATLAB Engine Header File required for building  ***/
#include "gaussian3x3.h"

#define BUFSIZE 256
```

MATLAB - Framework for C/C++ Development



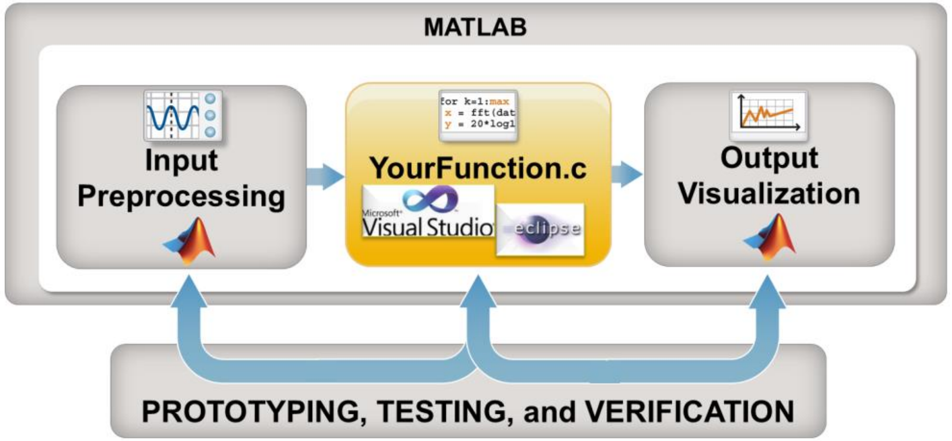
```
for k=1:max
 x = fft(dat
 y = 20*log1
```

Summary Demo 1:
Visualizing, Verifying, and Prototyping

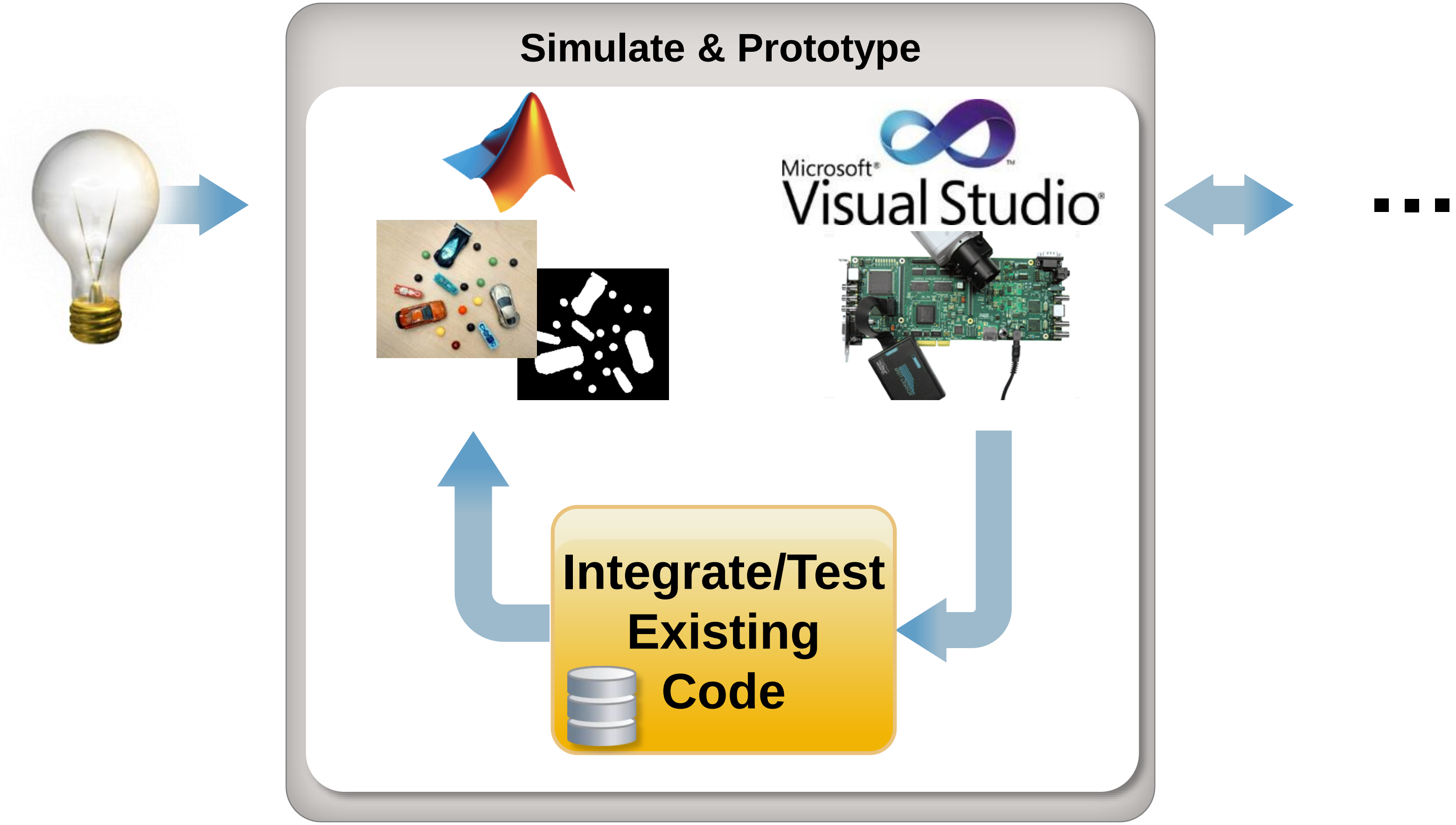
- Create a powerful C/C++ development & test harnesses
  - Simulate I/O
  - Visualize your data (printf on steroids)
  - Test & Validate to downstream minimize errors

- Access MATLAB functions to quickly prototype

- Technologies include:
  - Visual Studio or Eclipse
  - MATLAB Engine
  - Unit Testing Framework



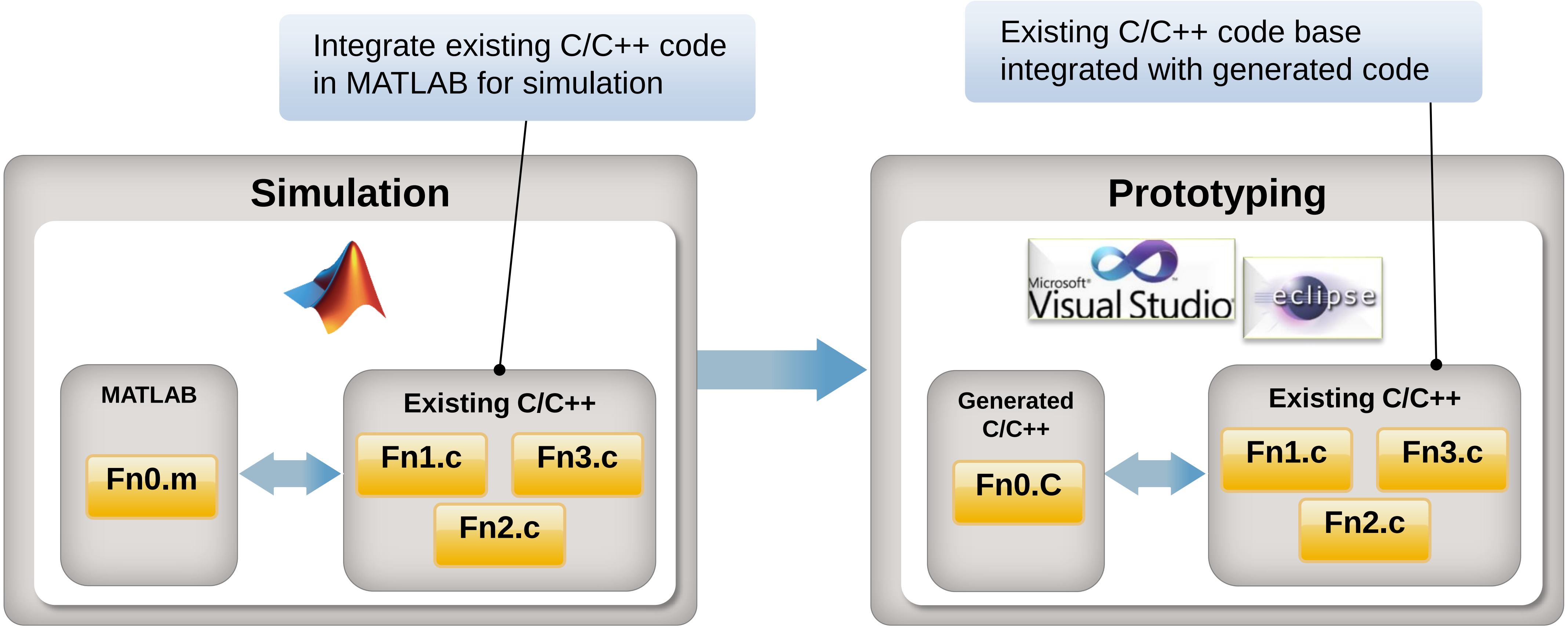
Development Workflow with MATLAB



Simulate & Prototype

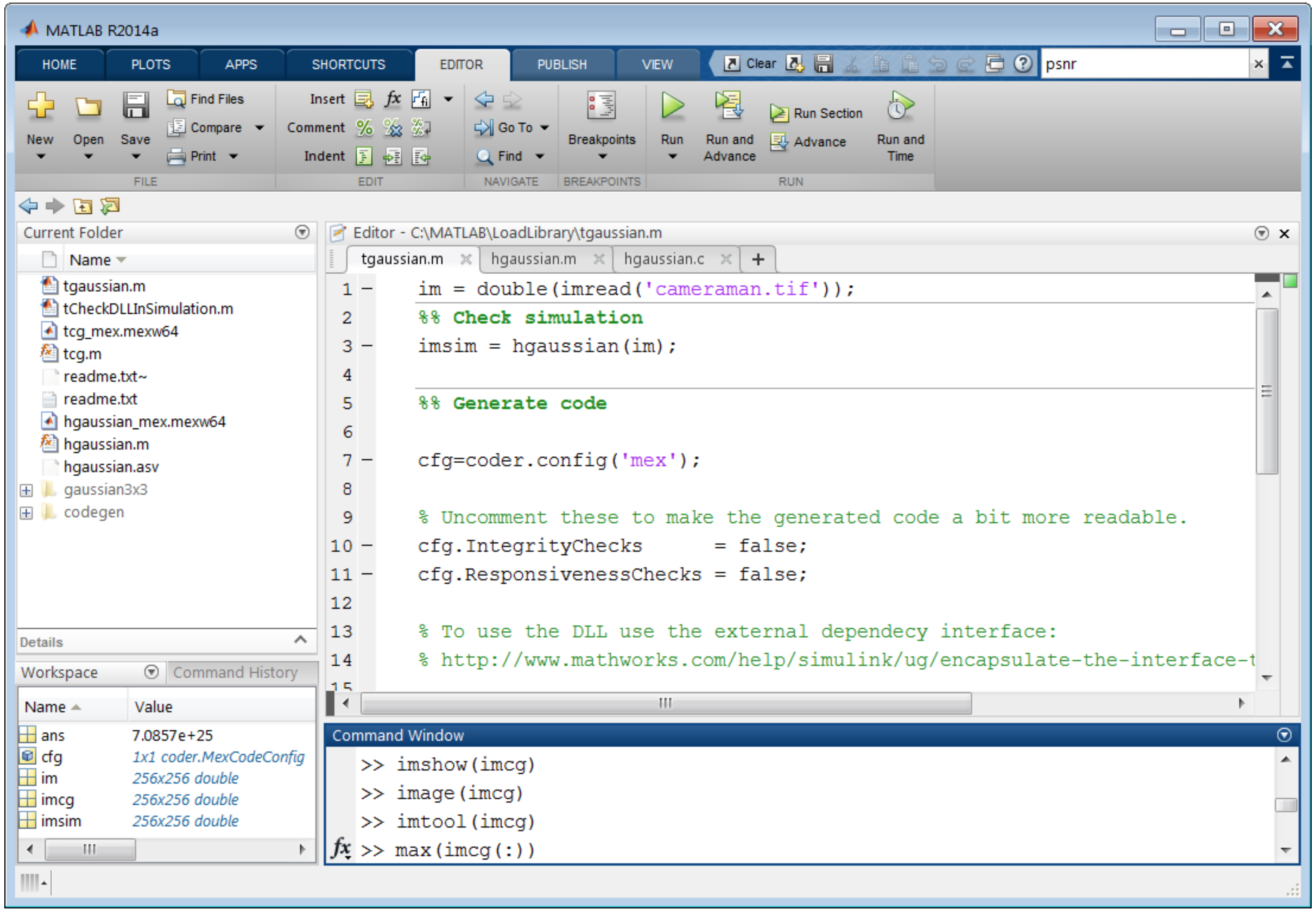
Integrate/Test Existing Code

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



- tgaussian.m
- tCheckDLLInSimulation.m
- tcg_mex.mexw64
- tcg.m
- readme.txt~
- readme.txt
- hgaussian_mex.mexw64
- hgaussian.m
- hgaussian.asv
- gaussian3x3
- codegen

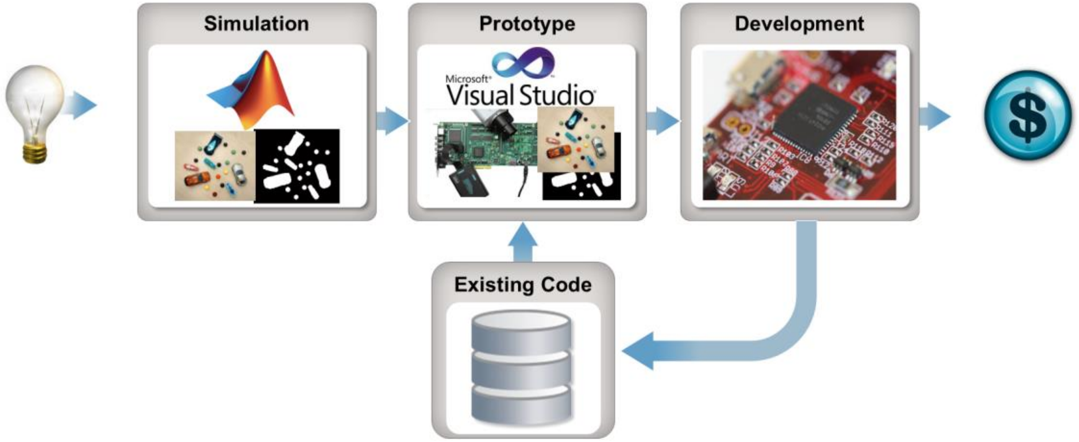
```
im = double(imread('cameraman.tif'));
%% Check simulation
imsim = hgaussian(im);

%% Generate code

cfg=coder.config('mex');

% Uncomment these to make the generated code a bit more readable.
cfg.IntegrityChecks      = false;
cfg.ResponsivenessChecks = false;

% To use the DLL use the external dependecy interface:
% http://www.mathworks.com/help/simulink/ug/encapsulate-the-interface-t
```

Name	Value
ans	7.0857e+25
cfg	*1x1 coder.MexCodeConfig*
im	*256x256 double*
imcg	*256x256 double*
imsim	*256x256 double*

```
>> imshow(imcg)
>> image(imcg)
>> imtool(imcg)
>> max(imcg(:))
```

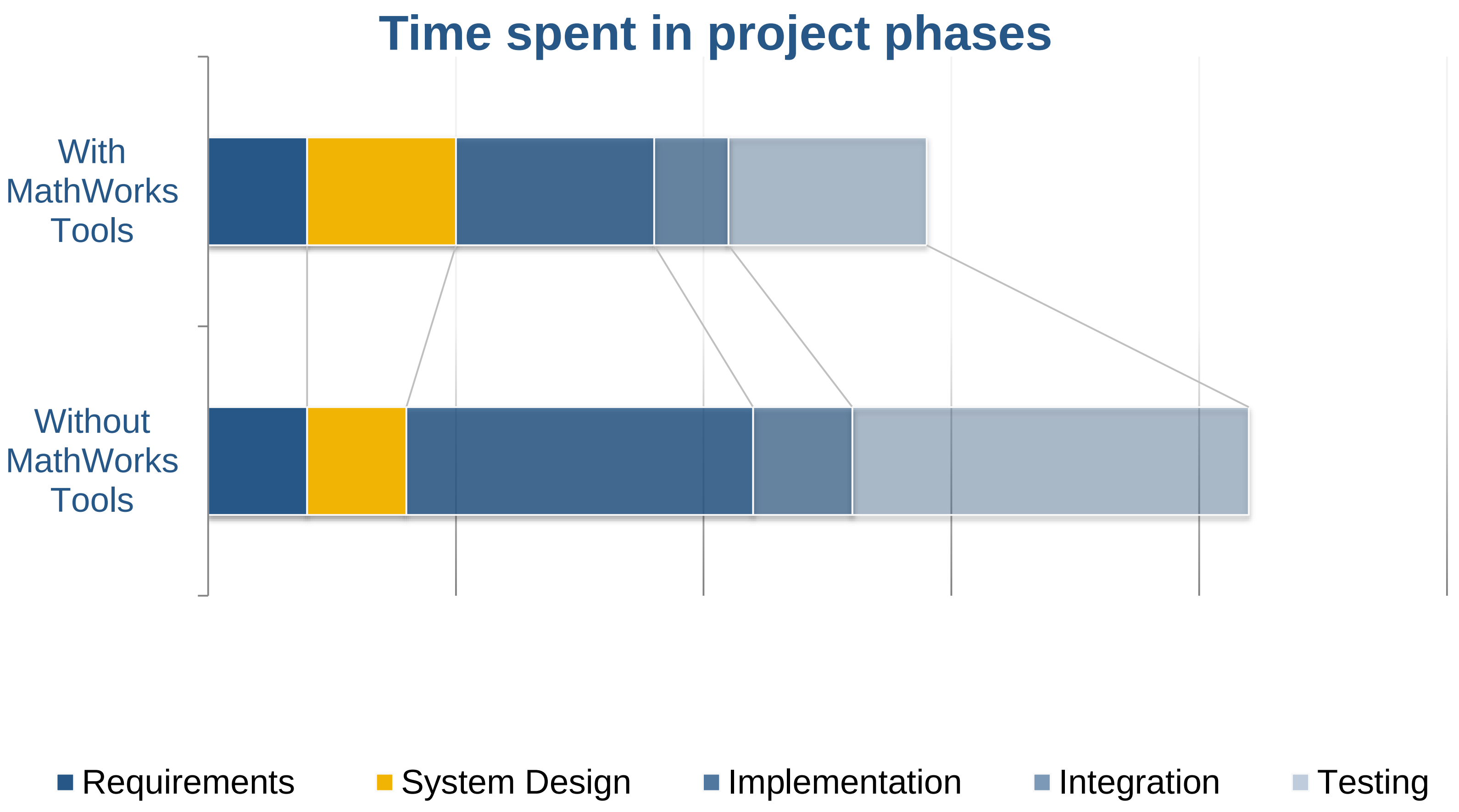
Summary Demo 2:
System design by leveraging existing code

- 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.)



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

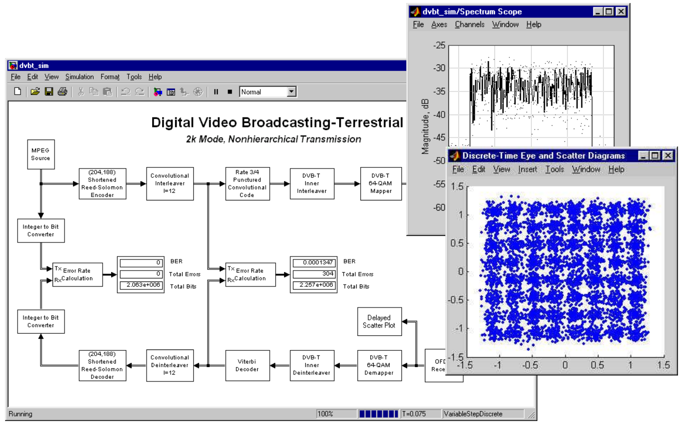
Agenda

- Introduction to MATLAB Overview
- MATLAB for Development & Design
- Simulink for Development & Design
- User Stories
- Summary and Wrap-up

SIMULINK®

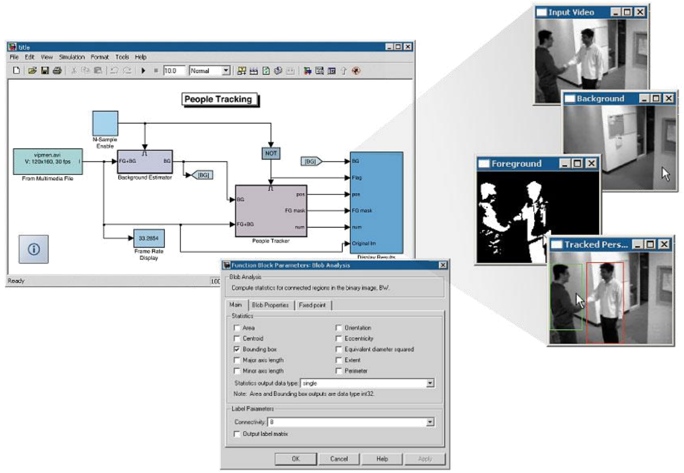
The leading environment for modeling, simulating, and implementing communications systems and semiconductors

- 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


dvbt_sim
File Edit View Simulation Format Tools Help
Normal
Digital Video Broadcasting-Terrestrial
2k Mode, Nonhierarchical Transmission
MPEG Source
(204,188) Shortened Reed-Solomon Encoder
Convolutional Interleaver I=12
Rate 3/4 Punctured Convolutional Code
DVB-T Inner Interleaver
DVB-T 64-QAM Mapper
Integer to Bit Converter
Tx Rx Error Rate Calculation
BER
Total Errors
Total Bits
Integer to Bit Converter
Delayed Scatter Plot
(204,188) Shortened Reed-Solomon Decoder
Convolutional Deinterleaver I=12
Viterbi Decoder
DVB-T Inner Deinterleaver
DVB-T 64-QAM Demapper
OFD Rece
Running
100%
T=0.075
VariableStepDiscrete
dvbt_sim/Spectrum Scope
File Axes Channels Window Help
Magnitude, dB
-25
-30
-35
-40
-45
-50
-55
-60
Discrete-Time Eye and Scatter Diagrams
File Edit View Insert Tools Window Help
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

BER 0, Total Errors 0, Total Bits 2.063e+006

BER 0.0001347, Total Errors 304, Total Bits 2.257e+006



Function Block Parameters: Blob Analysis

Blob Analysis

Compute statistics for connected regions in the binary image, BW.

Main | Blob Properties | Fixed point

Statistics

- □ Area
- □ Centroid
- ☑ Bounding box
- □ Major axis length
- □ Minor axis length
- □ Orientation
- □ Eccentricity
- □ Equivalent diameter squared
- □ Extent
- □ Perimeter

Statistics output data type: single

Note: Area and Bounding box outputs are data type int32.

Label Parameters

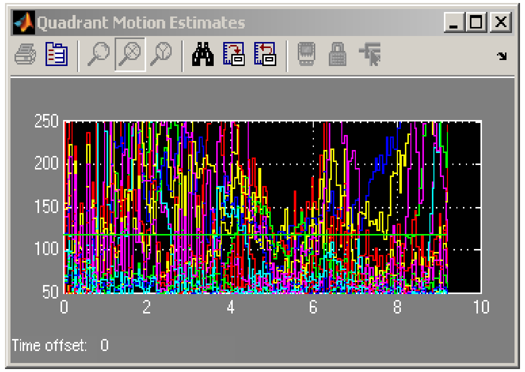
Connectivity: 8

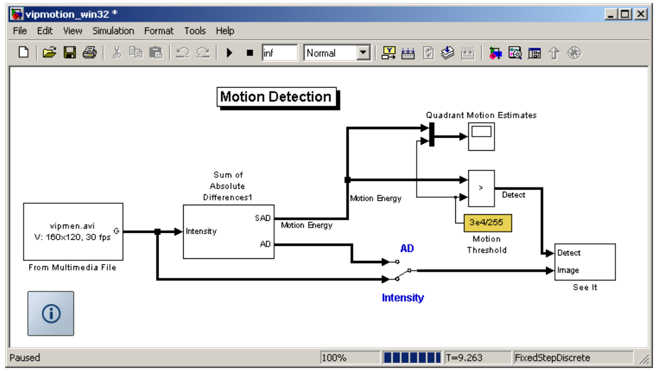
□ Output label matrix

OK | Cancel | Help | Apply

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:

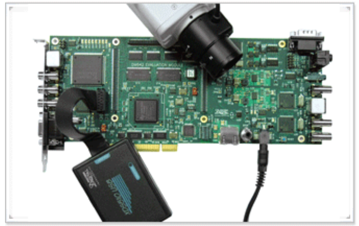
- Integration of existing code directly into the prototype
  - i.e. Blocks with ARM, FPGA, or DSP code

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

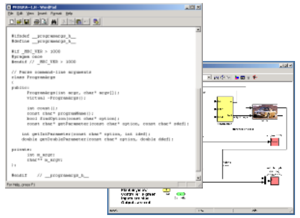
- Code generation for rapid prototyping and software deliveries

Using Simulink for Embedded Design

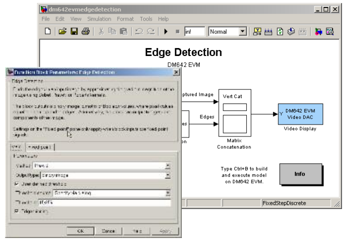
Rapidly prototype ideas

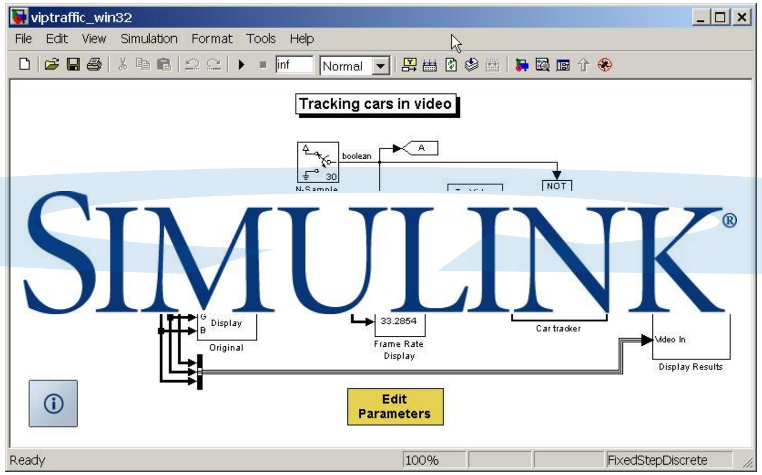


Integrate Existing Code

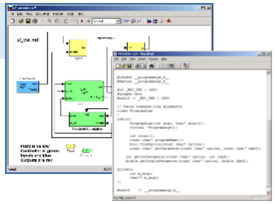


Analyze trade-offs

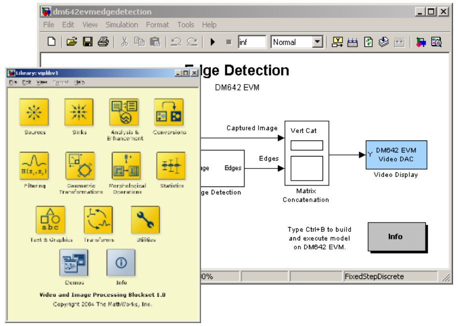




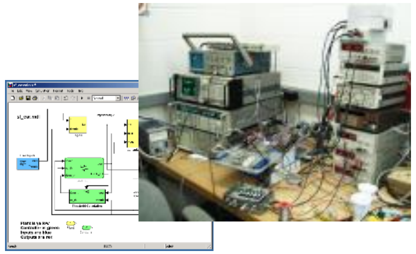
Generate C-code



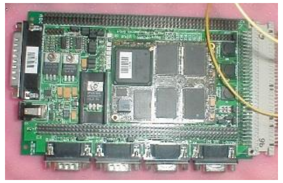
Implicit timing and concurrency



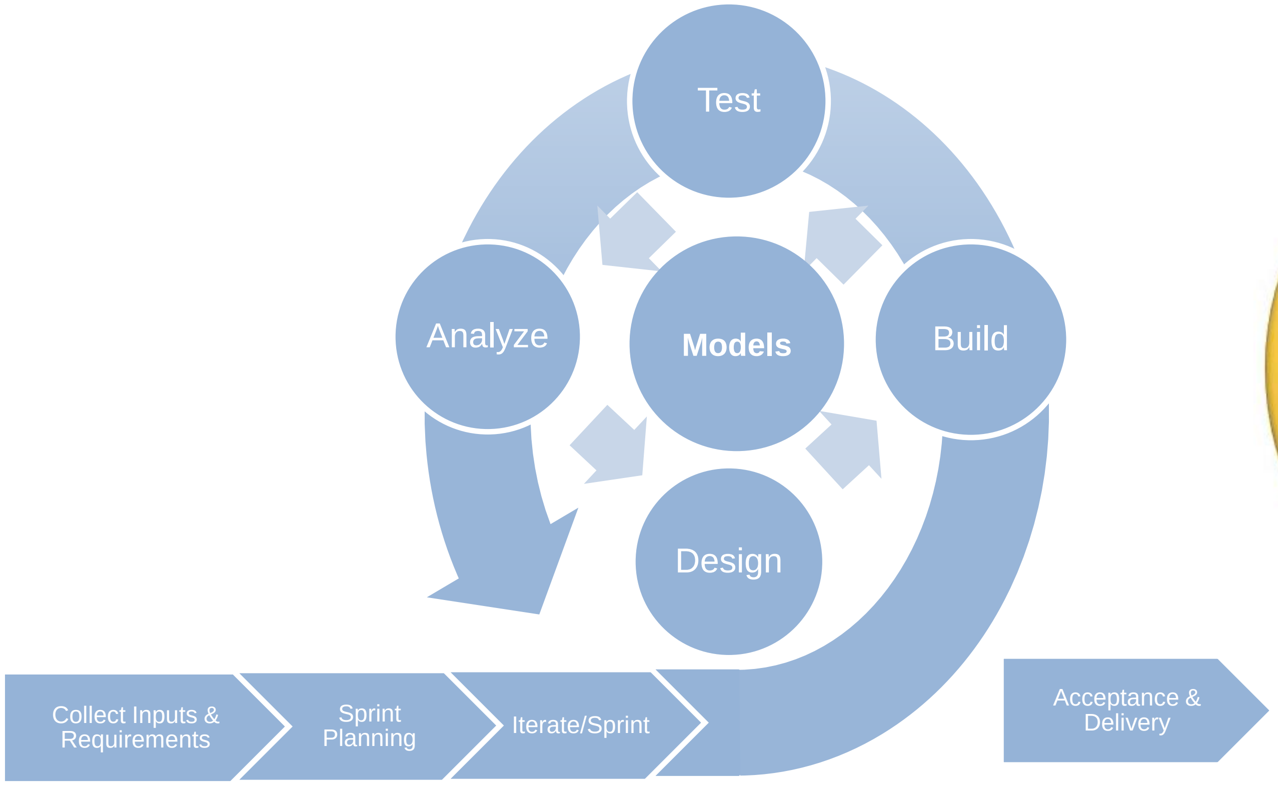
Continuous verification

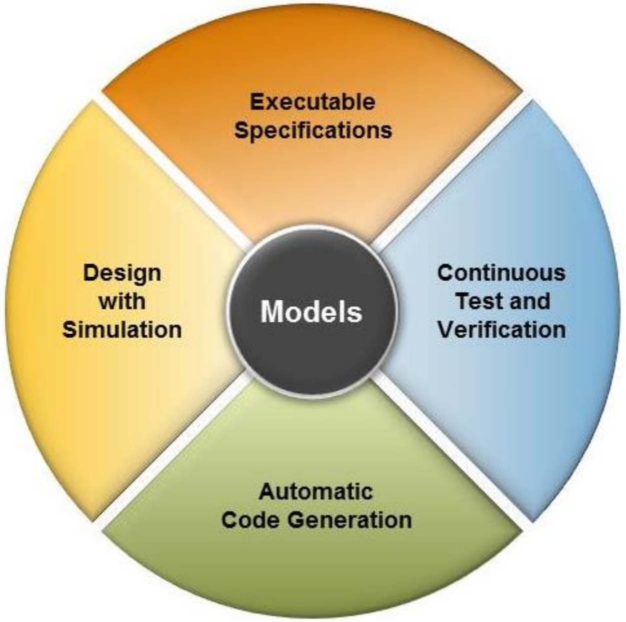


Target embedded devices



Model-Based Design & Agile Development

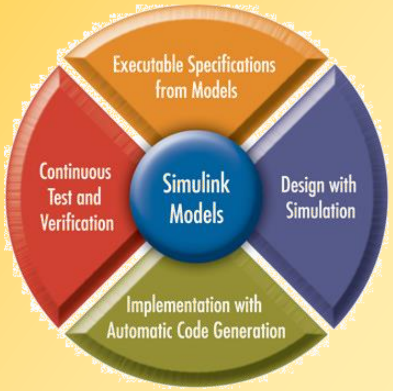



Executable Specifications
Design with Simulation
Models
Continuous Test and Verification
Automatic Code Generation

The Value of Model-Based Design

Model-Based Design

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


Executable Specifications from Models
Design with Simulation
Implementation with Automatic Code Generation
Continuous Test and Verification
Simulink Models

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

Introduction to MATLAB Overview

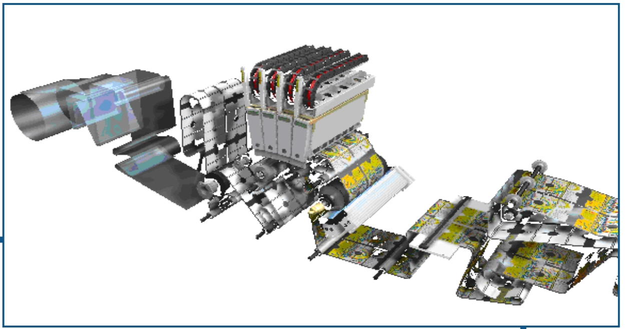
MATLAB for Development & Design

Simulink for Development & Design

User Stories

Summary and Wrap-up

Centre for Concepts in Mechatronics Improves Resolution of Agfa Printers Using MathWorks Tools for Model-Based Design



Substrate path of an industrial printer.

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

DLR Develops Autonomous Humanoid Robot with Model-Based Design

DLR's humanoid robot Agile Justin autonomously performing a complex construction task.

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

Link to user story

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

Volvo Construction Equipment's 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

> **"It 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**

Link to user story

Agenda

- Introduction to MATLAB Overview
- MATLAB for Development & Design
- Simulink for Development & Design
- User Stories
- Summary and Wrap-up

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

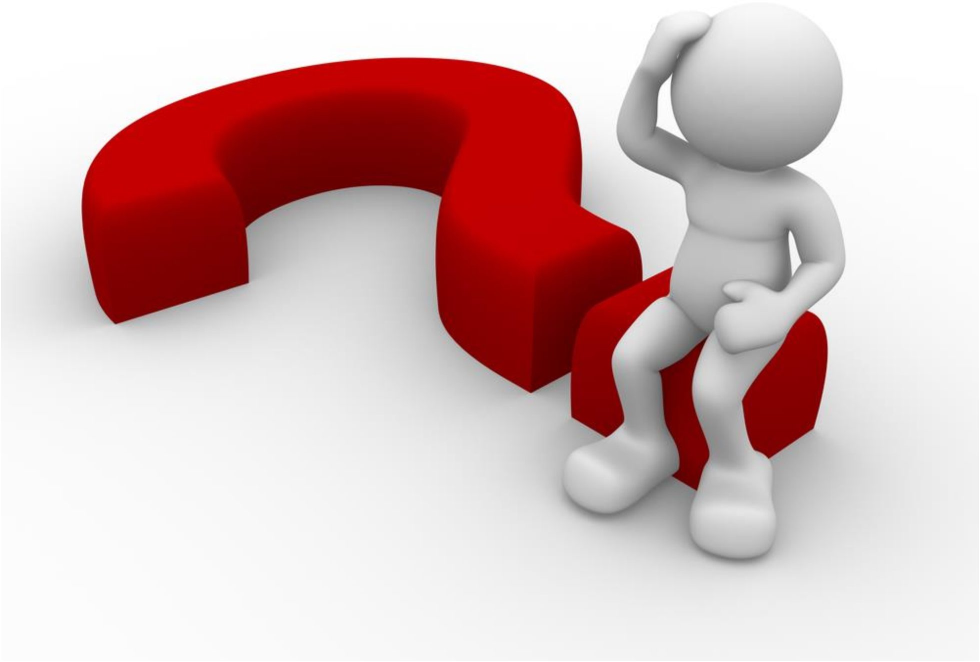
- MATLAB Engine – APIs and Documents
  - http://www.mathworks.com/help/matlab/calling-matlab-engine-from-c-c-and-fortran-programs.html

- Setting up Visual Studio, Eclipse, or Xcode
  - http://www.mathworks.com/help/matlab/matlab_external/compiling-engine-applications-in-an-ide.html

- LoadLibrary
  - http://www.mathworks.com/help/matlab/using-c-shared-library-functions-in-matlab-.html

- Coder.ceval
  - http://www.mathworks.com/help/simulink/slref/coder.ceval.html

Questions?



MATLAB for Software Development
Why? How? What?

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