

MATLAB EXPO 2019

Software Development Practices within MATLAB

David Sampson



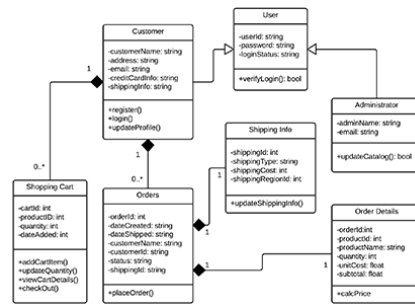
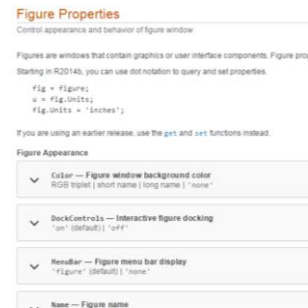
I want to help you...

Design

Architect

Implement

Share



...your MATLAB tools

What are your software development concerns?

- Accuracy
- Execution performance
- Development time
- Cost
- Compatibility
- Documentation
- Reusability
- Effective testing
- Integration
- Ease of collaboration
- Legacy code
- Liability
- Maintainability
- Model risk
- Robustness
- Developer expertise
- Software stack complexity
- ...?

Agenda

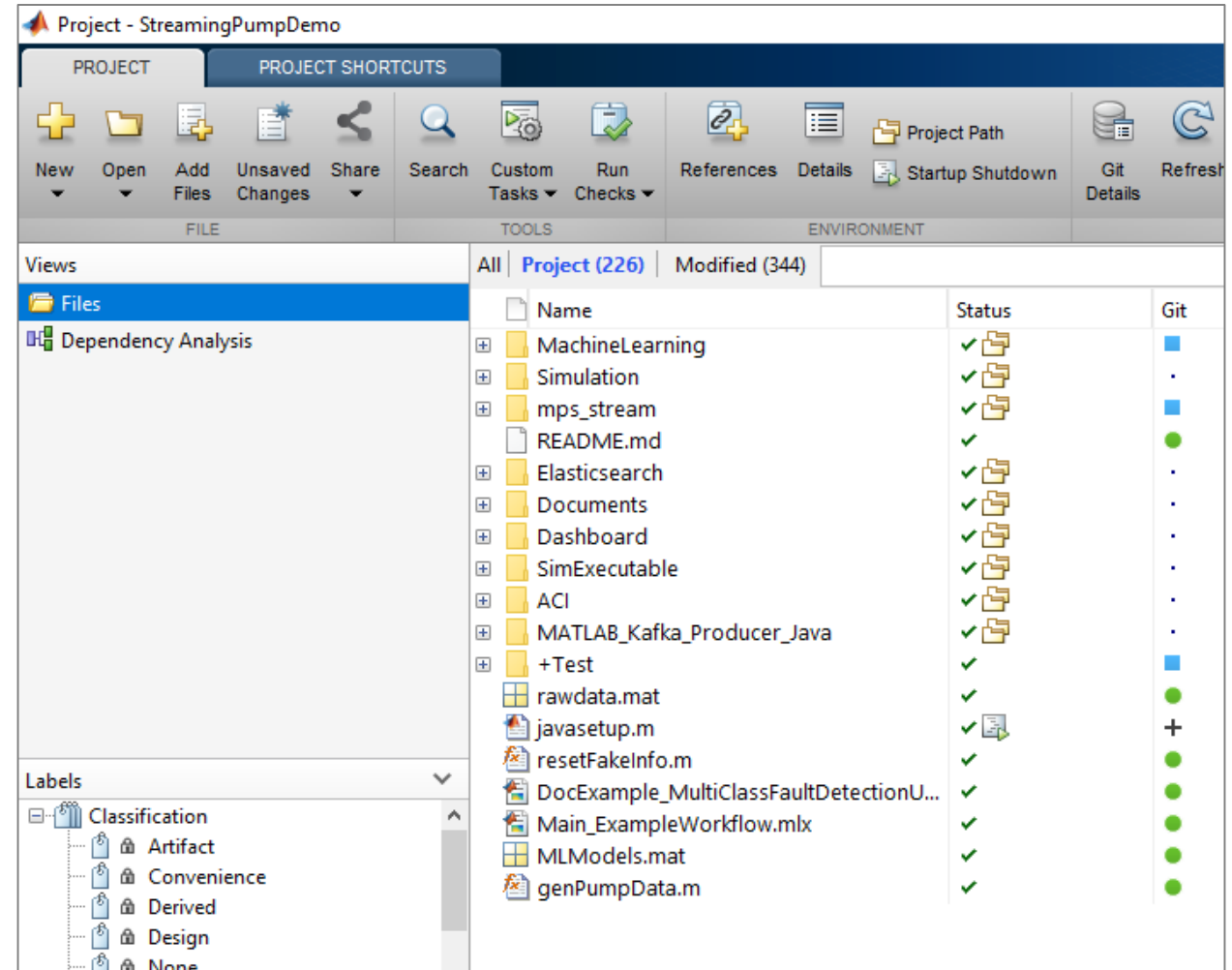
- MATLAB Projects
- Version control integration
- Language features
- Development environment
- Testing & CI
- Toolbox distribution
- Design patterns



MATLAB Projects

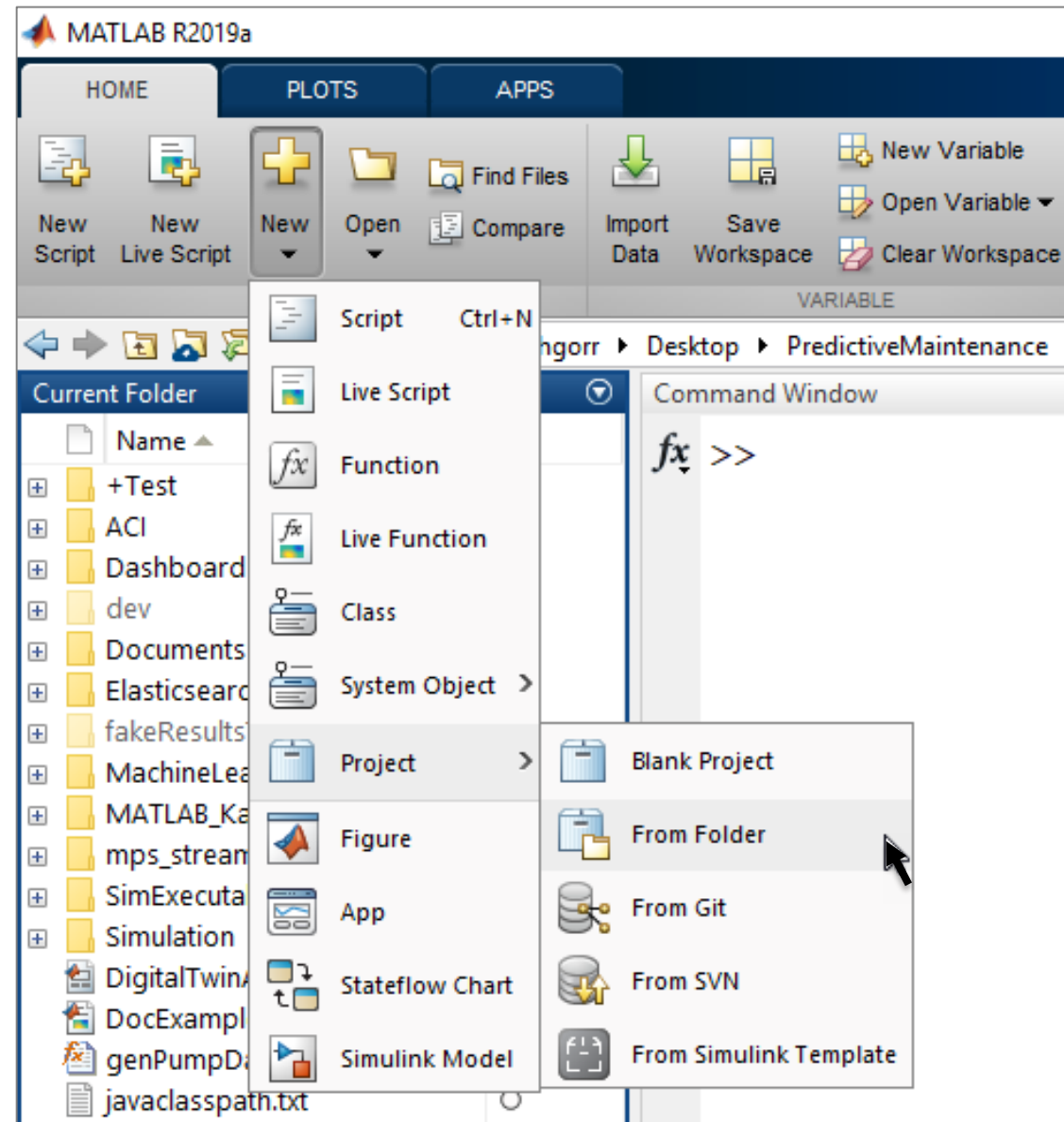
Projects (MATLAB + Simulink Projects)

- Manage your files and path
- Analyze file dependencies
- Function refactoring
- Run startup & shutdown tasks
- Create project shortcuts
- Label and filter files
- Integrate source control



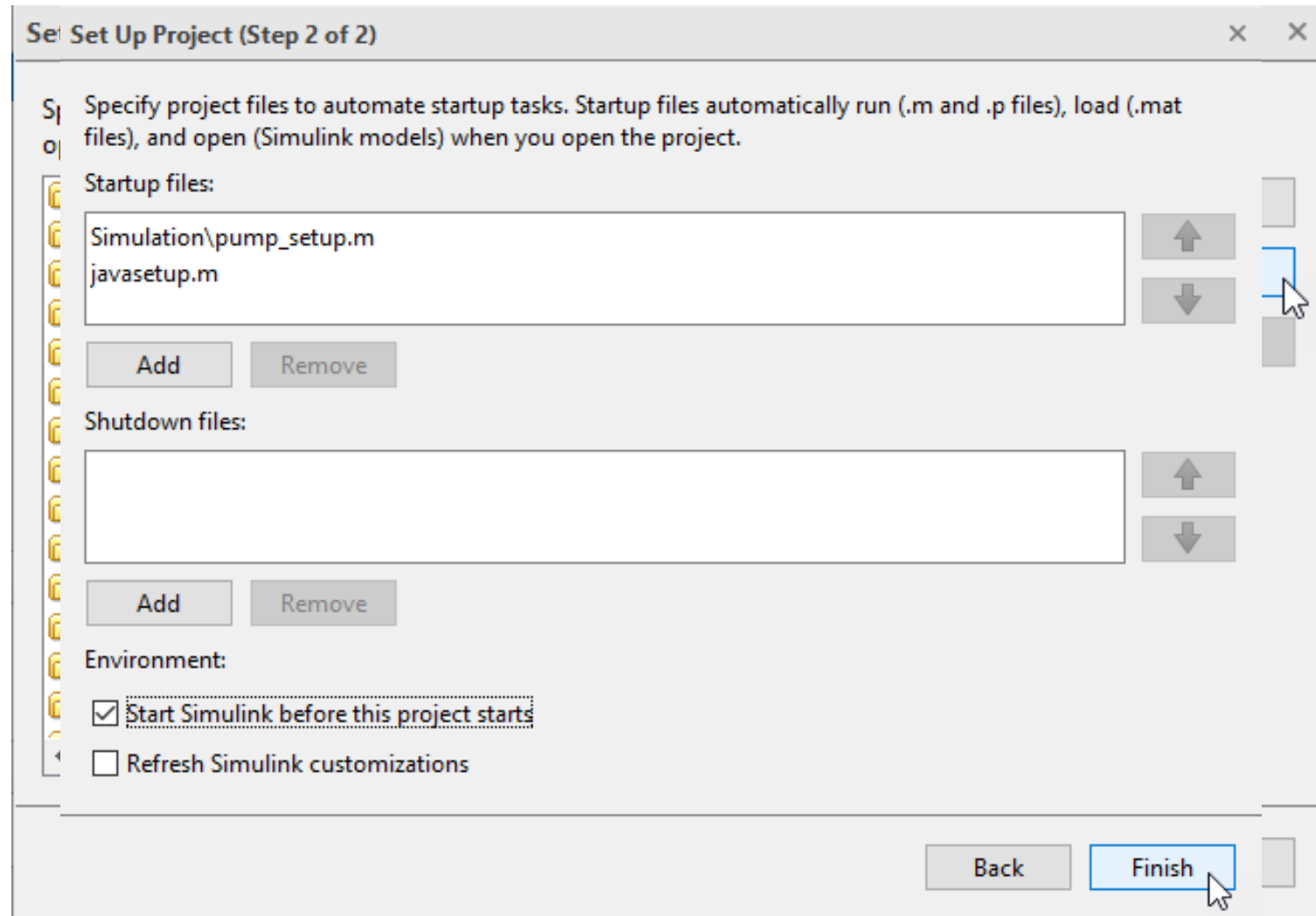
Managing your code with Projects

1. Create project



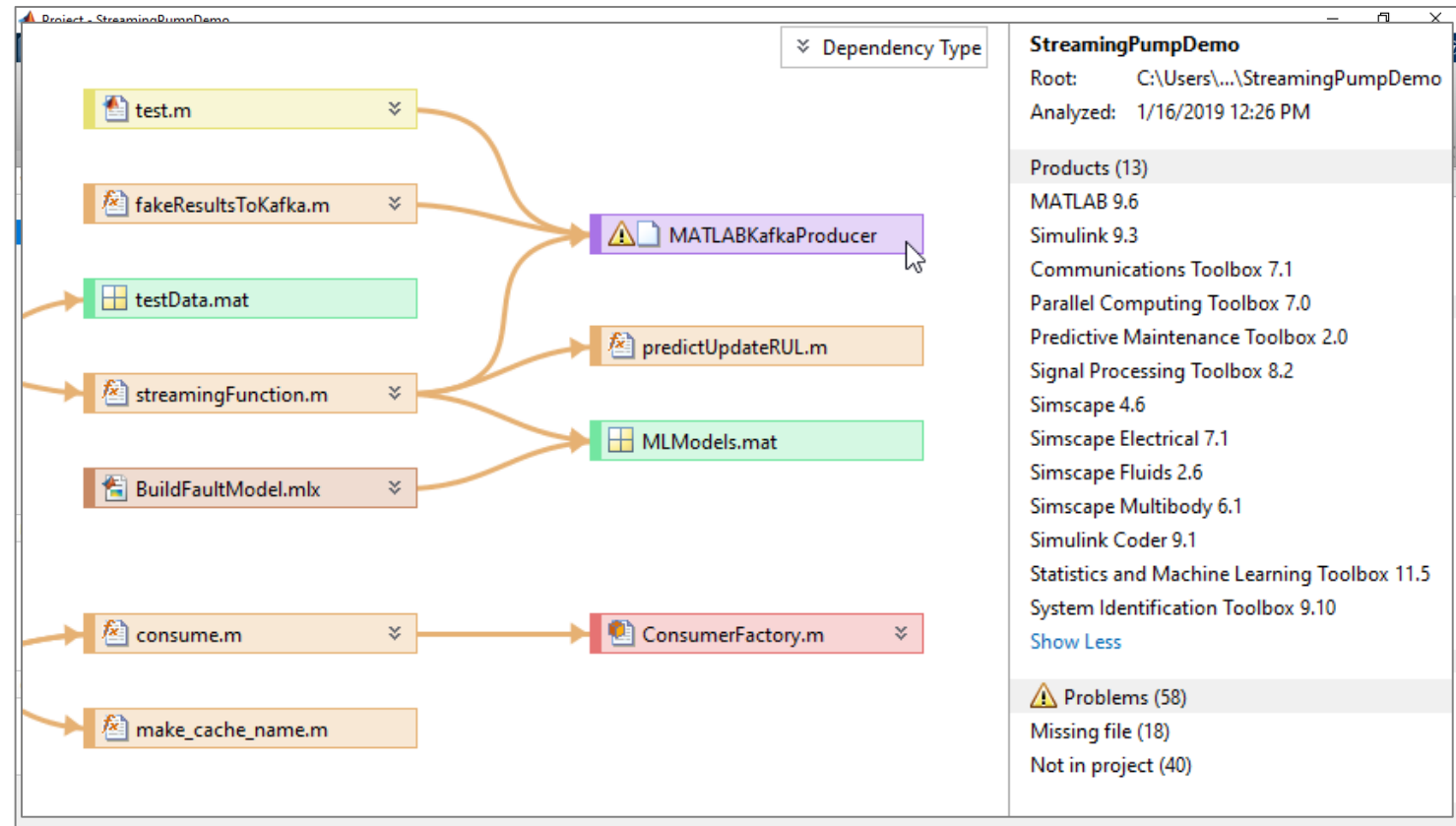
Managing your code with Projects

1. Create project
2. Set path and startup tasks



Managing your code with Projects

1. Create project
2. Set path and startup tasks
3. Explore dependencies



Managing your code with Projects

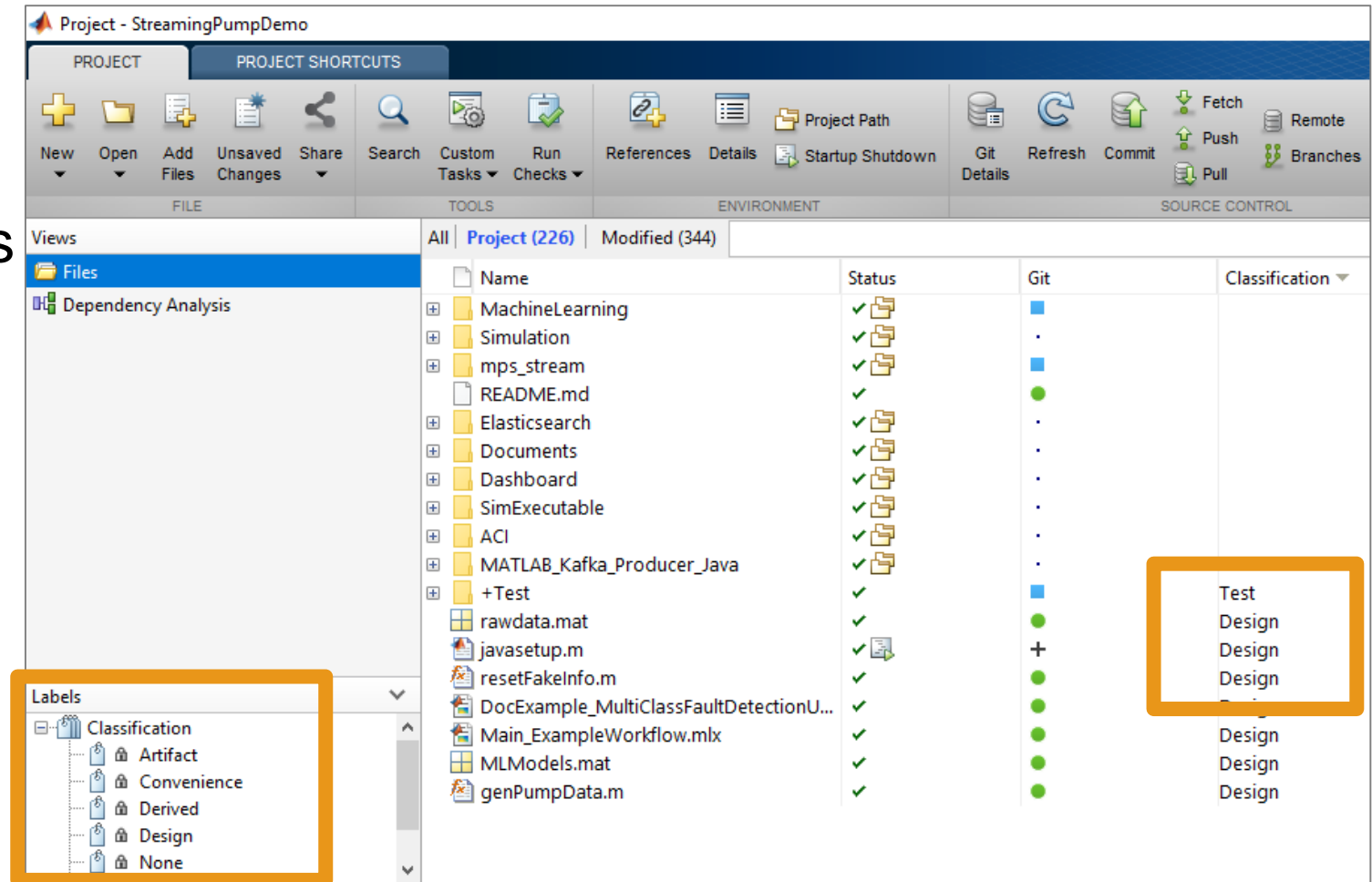
1. Create project
2. Set path and startup tasks
3. Explore dependencies
4. Label files



Identify and run tests

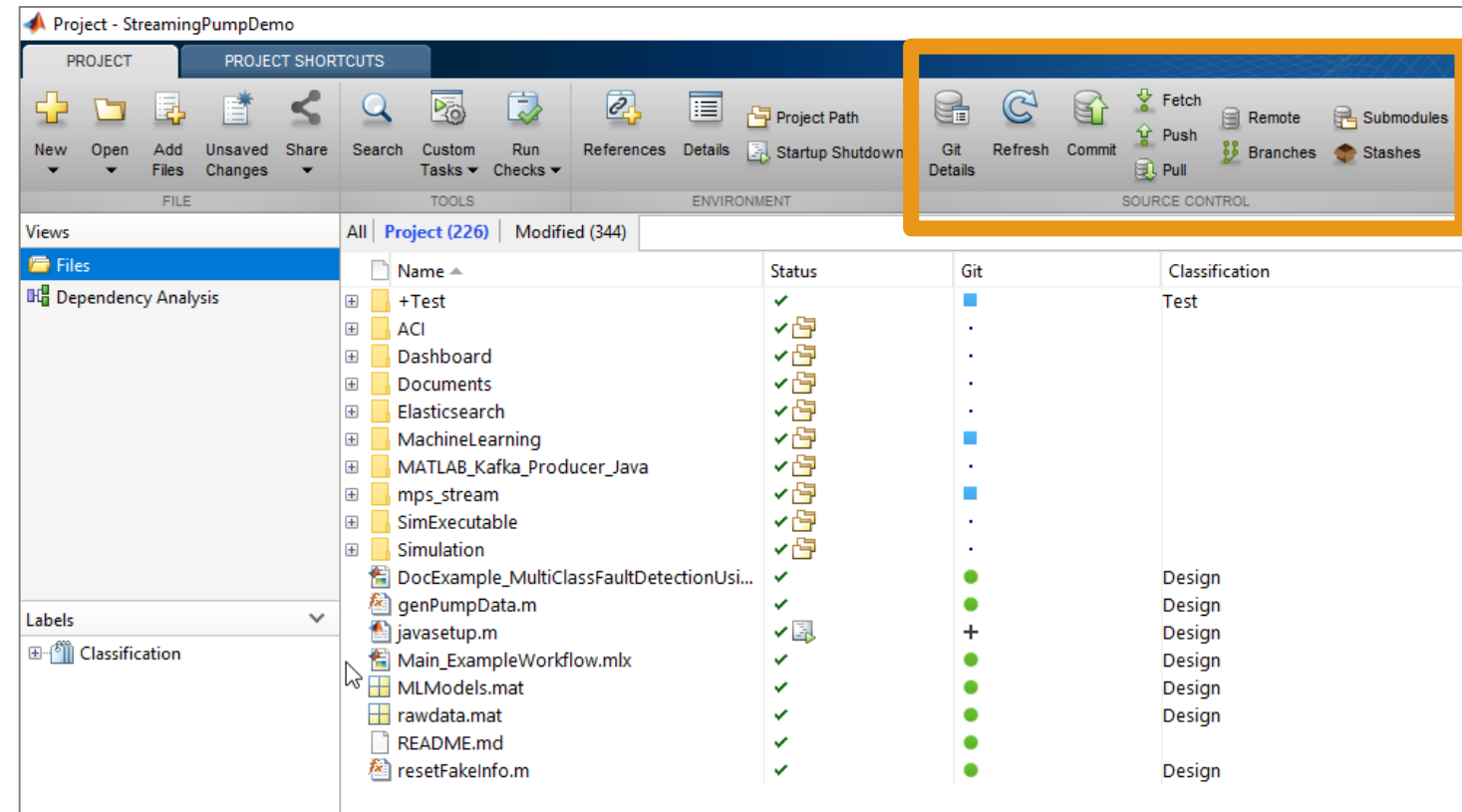
Managing your code with Projects

1. Create project
2. Set path and startup tasks
3. Explore dependencies
4. Label files



Managing your code with Projects

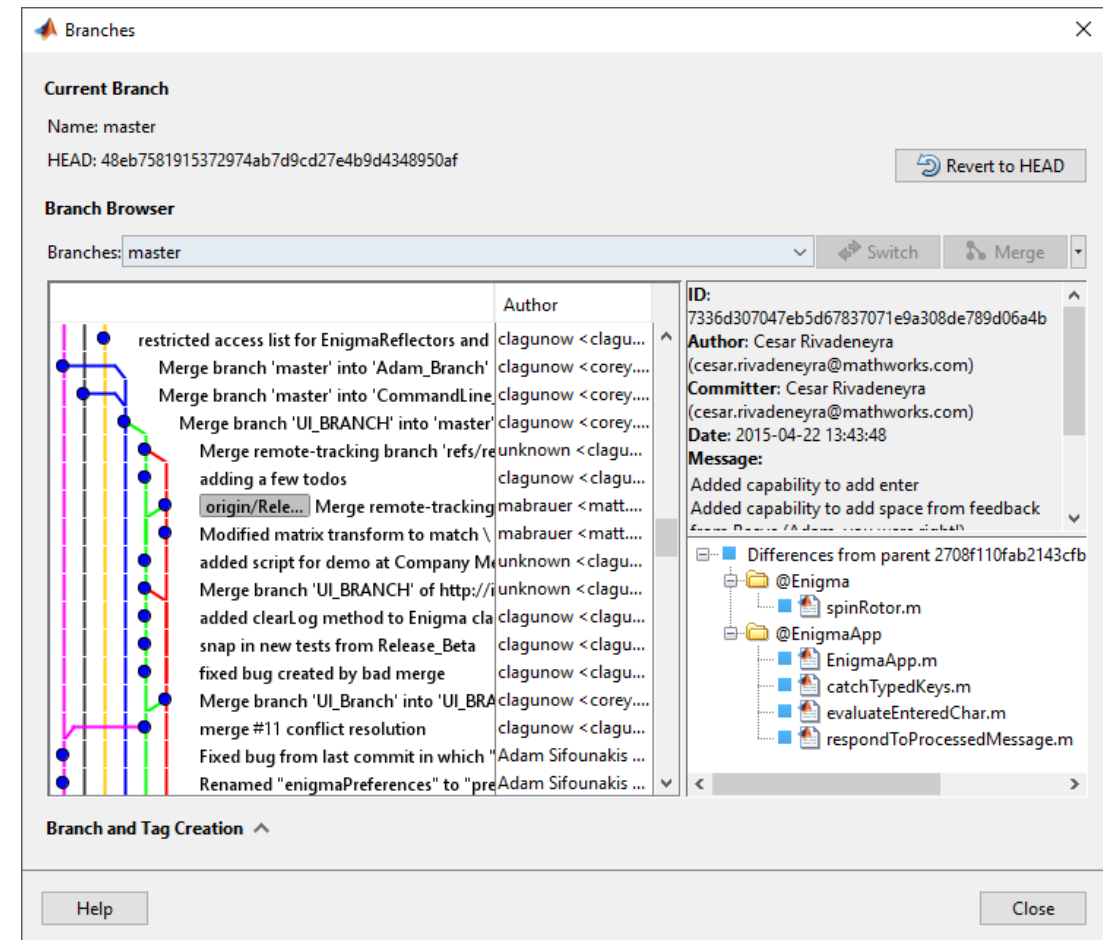
1. Create project
2. Set path and startup tasks
3. Explore dependencies
4. Label files
5. Integrate source control



Version control

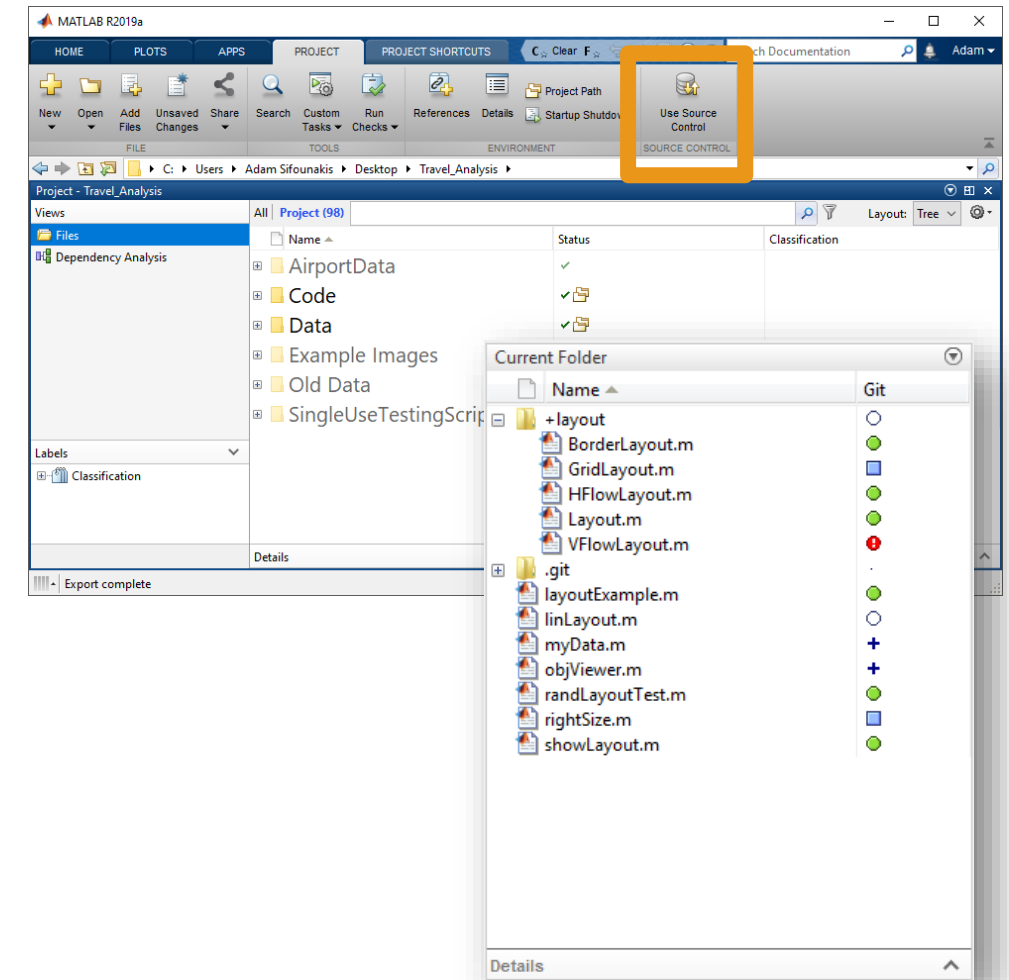
Version control

- Maintain backups, history, and ability to restore
- Track changes and responsibility
- Simplify reconciling conflicting changes
- Generate discussion
- Save you from yourself



Version control integration

- Manage your code from within the MATLAB Desktop
- Git integrated into:
 - Projects
 - Current Folder browser
- Use Comparison Tool to view and merge changes between revisions



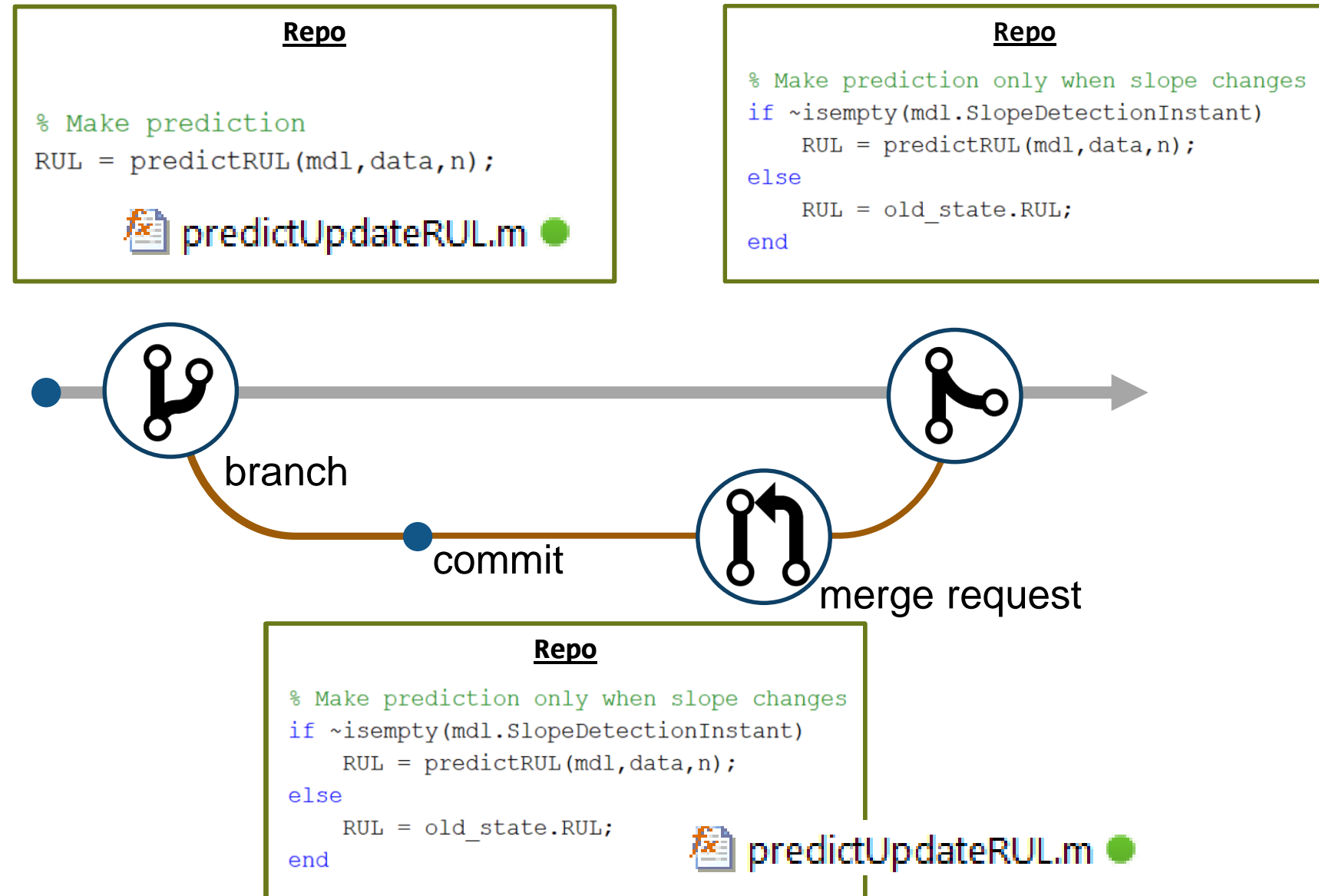
Co-authoring workflows

Creating a repo:

- Initialize
- Add
- Clone

Making changes:

- Commit
- Push
- Branch
- Merge



Implementation

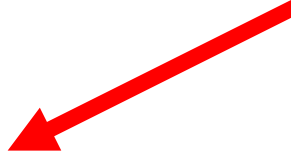
Considerations when writing better, robust, and portable code

- Input validation
- Error handling
- Writing faster code using the MATLAB Profiler
- Writing code faster using the Live Editor
- Refactoring code to reduce complexity
- Writing code that works on all operating systems

Unactionable errors

```
>> y = myfunc( 1:5 )
```

```
Index exceeds matrix dimensions.
```



```
Error in mypkg1.mypkg1a.mypkg1ab.myfunc1 (line 9)
```

```
y(idx) = u(idx)*log(u_hat(idx))+(1-u(idx))*log(1-u_hat(idx));
```

```
Error in mypkg2.mypkg2a.myfunc2 (line 5)
```

```
y = mypkg1.mypkg1a.mypkg1ab.myfunc1( myVar1 .* myVar2 );
```

```
Error in mypkg3.mypkg3a.myfunc3>@(x)mypkg2.mypkg2a.myfunc2(x) (line 4)
```

```
y = arrayfun( @(x) mypkg2.mypkg2a.myfunc2( x ), myVar );
```

```
Error in mypkg3.mypkg3a.myfunc3 (line 4)
```

```
y = arrayfun( @(x) mypkg2.mypkg2a.myfunc2( x ), myVar );
```

```
Error in myfunc (line 10)
```



Validating inputs

- `validateattributes`
- `assert`
- `isempty`, `isnan`, `isfinite`, ...
- `narginchk`
- `inputParser`
- Property validation for classes

```

1 function y = myfunc( x )
2
3 % Validate inputs
4 validateattributes(x, 'double', {'size', [1 3], 'increasing'});
5

```

```

>> myfunc( 1:5 )
Error using myfunc (line 4)
Expected input to be of size 1x3, but it is of size
1x5.

```

```

>> myfunc( [2 3 1] )
Error using myfunc (line 4)
Expected input to be increasing valued.

```

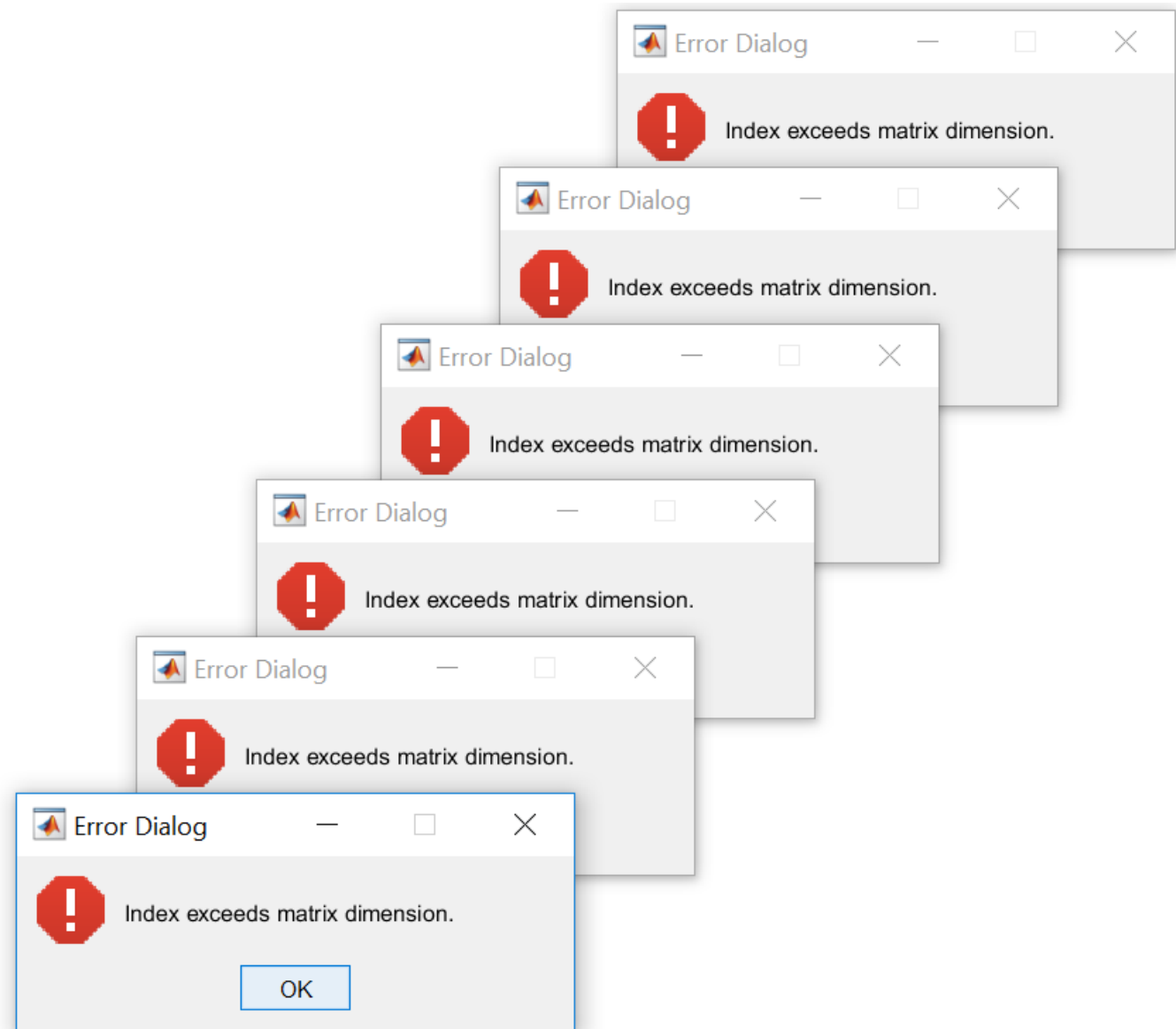
```

classdef ValidatorFunction
    properties
        Data(:,1) double {mustBePositive, mustBeFinite} = [1 2 3]
        Interp {mustBeMember(Interp,{'linear','spline'})} = 'linear'
    end
end

```

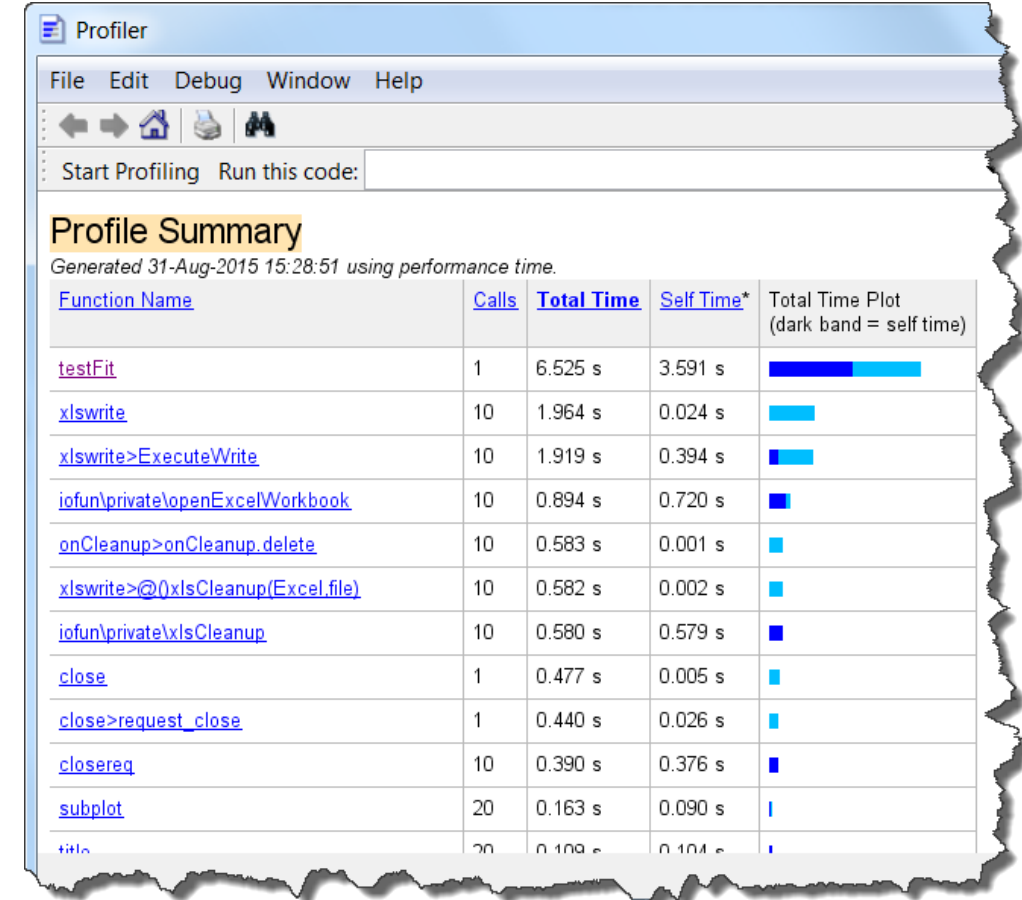
Handling errors more elegantly

- `error` **and** `warning`
 - Use identifiers
- `try/catch`
- `MException`
- `errordlg` **and** `warndlg`



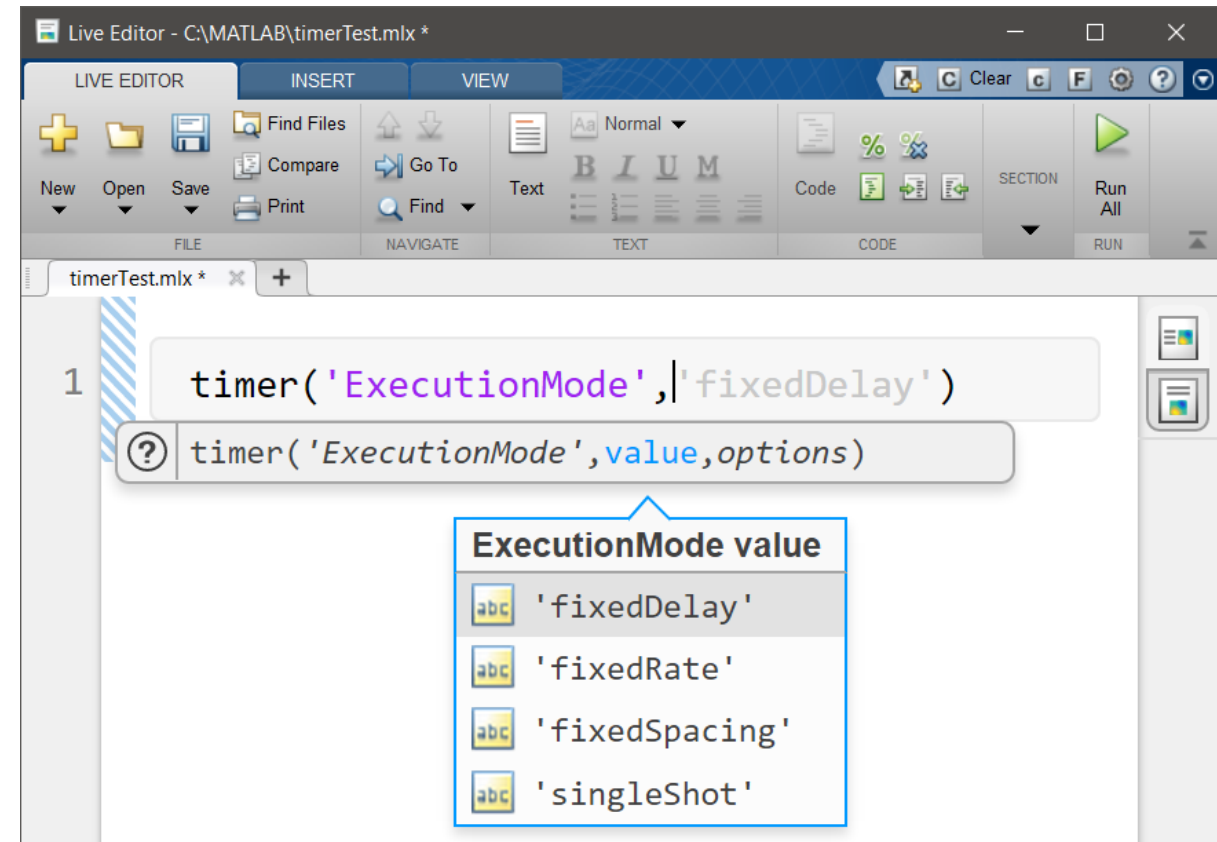
MATLAB Profiler

- Total number of function calls
- Time per function call
- Highlights largest code bottlenecks
- Statement coverage of code



Programming aids in the Live Editor

- Automatically closed parentheses, loops, and conditional blocks
- Context-aware coding guides
 - Automatically suggest function names, variables, or file names
 - List available Name/Value pairs



Quickly and safely refactoring code

- Live Editor shortcuts to refactor blocks of code into functions

The image shows a MATLAB Live Editor window with two tabs: 'MainScript.mlx' and 'myMathFunction.mlx'. In the 'MainScript.mlx' tab, lines 3-7 of code are selected and highlighted in blue. A context menu is open over this selection, with 'Convert to Function' and 'Convert to Local Function' highlighted in orange. An orange arrow points from this menu to the 'myMathFunction.mlx' tab. In the 'myMathFunction.mlx' tab, the code has been refactored into a function. The function signature 'function [z3, zSum] = myMathFunction(x, y)' is shown, with '[z3, zSum]' and 'myMathFunction' highlighted in orange. The body of the function contains the same code as the original script, with 'end' at the bottom.

Calculate my answer:

```
3 z1 = x+y;  
4 z2 = x-y;  
5 z3 = y-x;  
6 z4 = x*y;  
7 zSum = z1 + z2 + z3 + z4;
```

Display answers of interest:

```
8 disp(z3)  
9 disp(zSum)
```

Context Menu:

- Evaluate Selection in Command Window F9
- Open Selection Ctrl+D
- Help on Selection F1
- Copy Output
- Copy All Output
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Comment Ctrl+R
- Uncomment Ctrl+T
- Convert Between Code and Text Ctrl+E
- Change Case Ctrl+Shift+A
- Smart Indent Ctrl+I
- Convert to Function**
- Convert to Local Function**
- Insert Section Break Ctrl+Alt+Enter
- Run Section Ctrl+Enter
- Close All Output

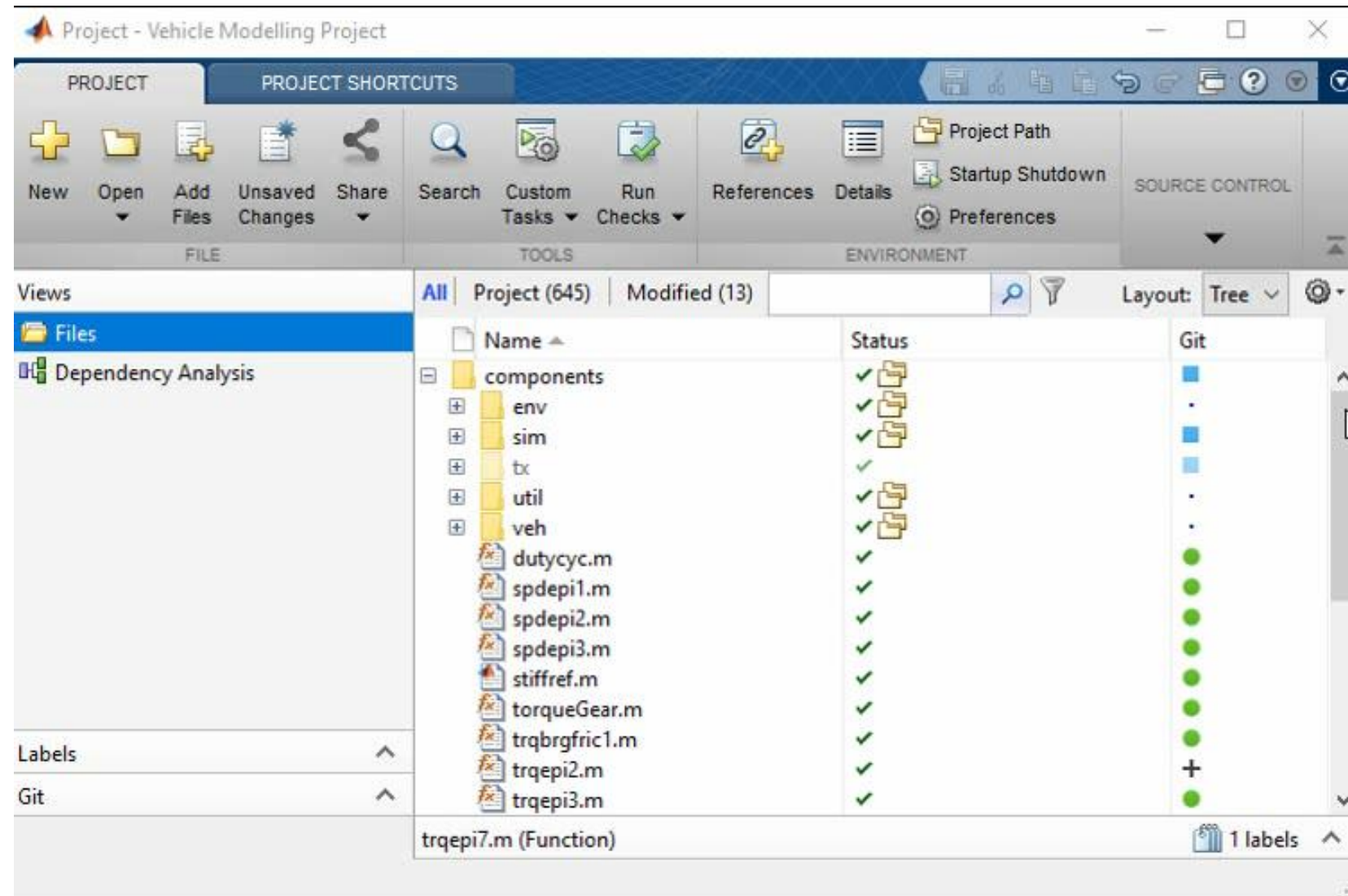
Live Editor - C:\MATLAB\TestFolder\myMathFunction.mlx

MainScript.mlx * myMathFunction.mlx +

```
1 function [z3, zSum] = myMathFunction(x, y)  
2 z1 = x+y;  
3 z2 = x-y;  
4 z3 = y-x;  
5 z4 = x*y;  
6 zSum = z1 + z2 + z3 + z4;  
7 end
```


Quickly and safely refactoring code

- Function refactoring across files in Projects



Simple code quality and complexity assessment – checkcode

- Analyze all warnings and errors in a code

```
>> checkcode standardizeEmployeeInfo
```

```
L 13 (C 14-24): The value assigned here to 'maxDatetime' appears to be unused. Consider replacing it by ~.
```

```
L 80 (C 1-27): The value assigned to variable 'emailsInUsernameFormatParts' might be unused.
```

```
L 116 (C 1-17): The value assigned to variable 'validEmployeeData' might be unused.
```

```
L 118 (C 1-28): The value assigned to variable 'emailsInFirstLastFormatParts' might be unused.
```

- McCabe Cyclomatic Complexity

- Measures complexity based on the number of linearly independent paths through a code

```
>> checkcode -cyc standardizeEmployeeInfo
```

```
L 1 (C 14-36): The McCabe cyclomatic complexity of 'standardizeEmployeeInfo' is 13.
```

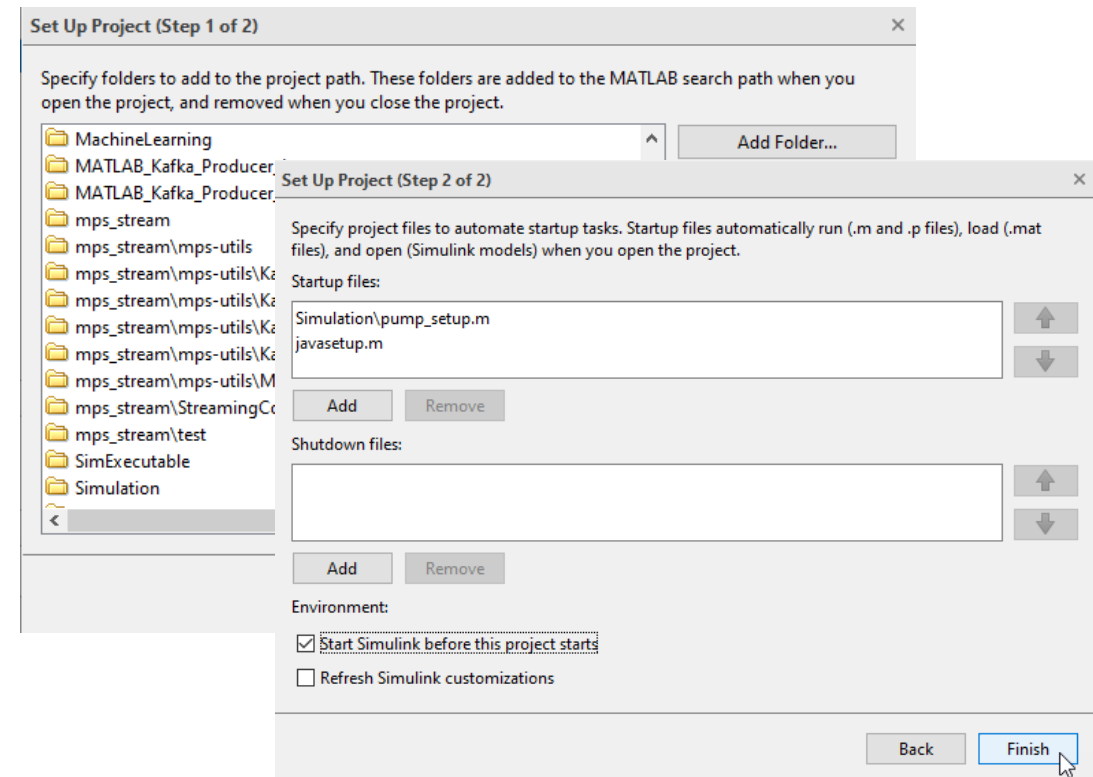
Code that runs everywhere

- Operating System-aware code
 - `fullfile`
 - `ispc`, `ismac`, `isunix`
- More reliable portability with Projects
 - Consistent path management
 - Automated startup/shutdown procedures
 - Built-in file dependency analysis

```
>> fullfile("../", "data", "2019", "April")
```

Windows: `"..\data\2019\April"`

Mac/Linux: `"../data/2019/April"`



Code maintenance

Code Compatibility Report

- Tool to help upgrade code to latest and greatest MATLAB
- Identifies potential compatibility issues
- Hundreds of checks for incompatibilities, errors, and warnings

Web Browser - (3 Errors) Code Compatibility Report

(3 Errors) Code Compatibility Report

Code Compatibility Report [Top](#) [3 Errors](#) [1 Warning](#) [304 Checks](#) [2 Files](#)

Analysis Date: 05-Sep-2017 14:32:08
MATLAB Version: R2017b

Incompatibility and Syntax Errors

Row	Filename	Line	Description	Details
1	classifyBloodPressure.m	18	TREEFIT has been removed. Use fitctree or fitrtree instead.	Details
2	classifyBloodPressure.m	21	TREEDISP has been removed. Use ClassificationTree or RegressionTree VIEW methods instead.	Details
3	classifyBloodPressure.m	24	TREEVAL has been removed. Use ClassificationTree or RegressionTree PREDICT methods instead.	Details

Warnings and Other Recommendations

Row	Filename	Line	Description	Details
1	classifyBloodPressure.m	7	RAND or RANDN with the 'seed', 'state', or 'twister' inputs is not recommended. Use RNG instead.	Details

Link to documentation for updates

Go directly to the line of code

Testing Frameworks

- MATLAB Unit Testing Framework
- Performance Testing Framework
- App Testing Framework

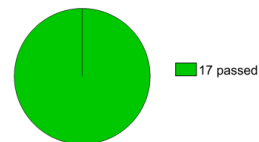
```
results =  
    1×17 TestResult array with properties:  
  
    Name  
    Passed  
    Failed  
    Incomplete  
    Duration  
    Details  
Totals:  
    17 Passed, 0 Failed, 0 Incomplete.  
    1.0937 seconds testing time.
```

MATLAB® Test Report

Timestamp: 04-Jan-2017 13:28:06
Host: AH-SDE
Platform: win64
MATLAB Version: 9.1.0.441655 (R2016b)

Number of Tests: 17
Testing Time: 0.4516 seconds

Overall Result: PASSED



Overview

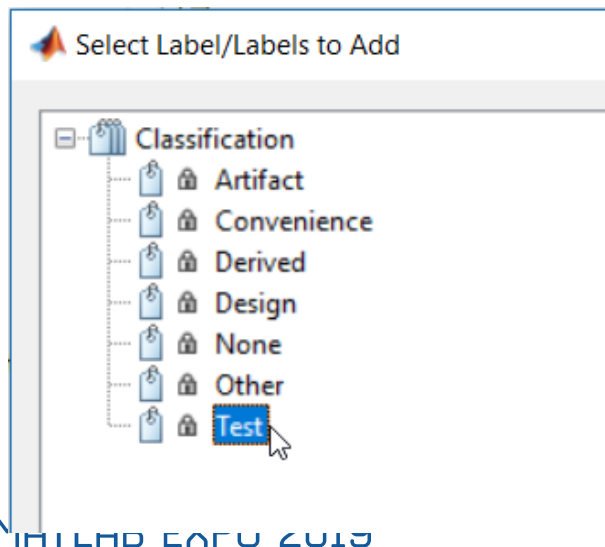
C:\Documents\MATLAB\OOD\Blip\Demos\Extensions\UnitTest\Class\	
BlipTests.BlipSizeLengthTests	0.1409 seconds
BlipTests.BlipSubassignTests	0.1542 seconds
BlipTests.BlipSubrefTests	0.1572 seconds

Details

C:\Documents\MATLAB\OOD\Blip\Demos\Extensions\UnitTest\Class\	
BlipTests.BlipSizeLengthTests	
• scalarBlipSize	
The test passed.	
Duration: 0.0863 seconds	
• vectorBlipSize	
The test passed.	
Duration: 0.0027 seconds	
• scalarBlipLength	
The test passed.	
Duration: 0.0044 seconds	
• vectorBlipLength	
The test passed.	
Duration: 0.0468 seconds	
BlipTests.BlipSubassignTests	
• assignVectorAoAParen	
The test passed.	
Duration: 0.0901 seconds	

MATLAB Unit Testing Framework

- Script-based test
- Function-based test
- Class-based test
- Test integration with Projects



test_Predictions.mlx

Test Pump Fault Model

This includes unit tests for the predictions

Test: Model type

Load the models and ensure they are the right types.

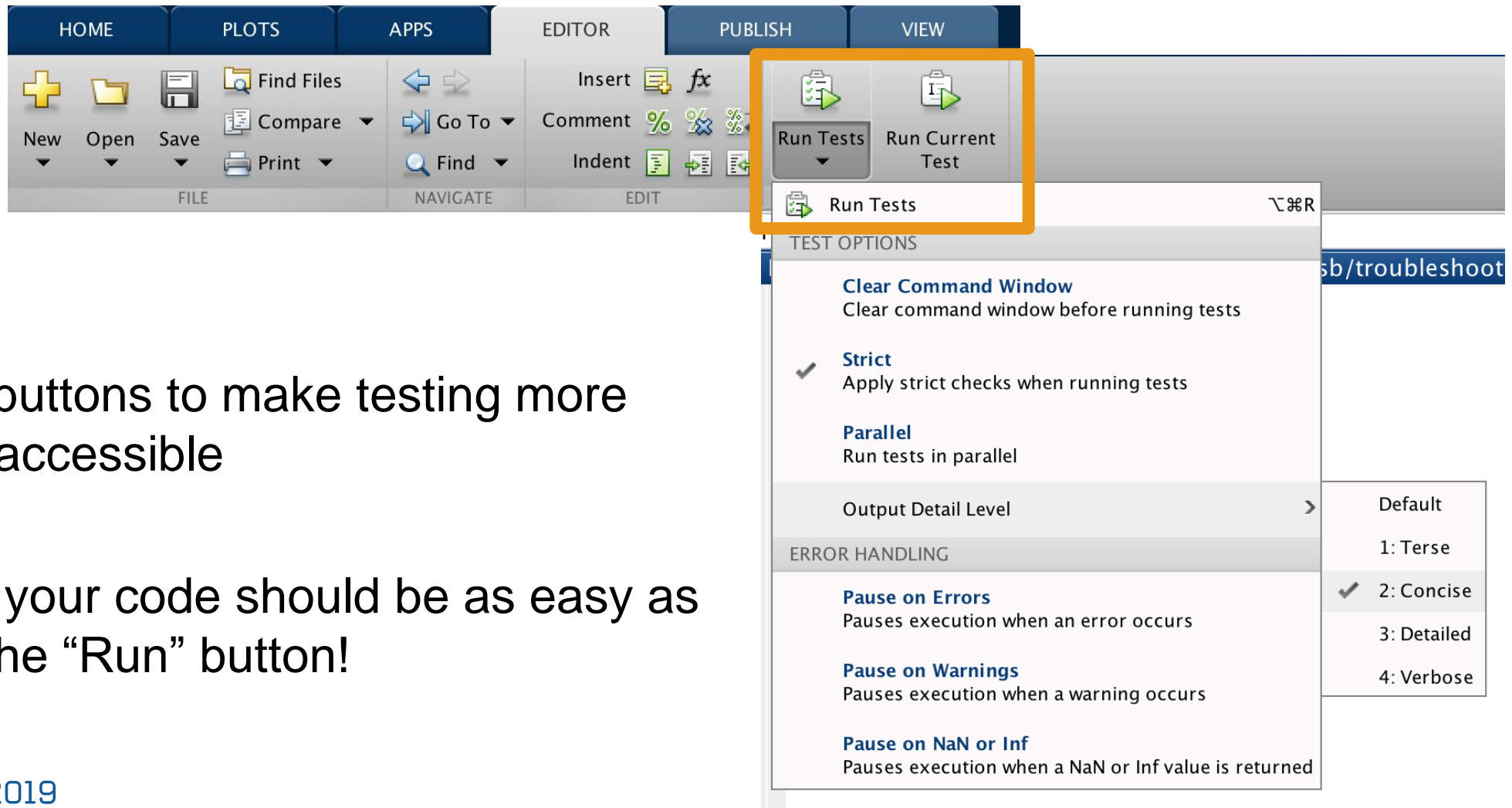
```
1 load MLModels trainedModel
2 mdl = trainedModel.ClassificationEnsemble;
3 assert(isa(mdl, 'classreg.learning.classif.CompactClassificationEnsemble'), ...
4         'Model is not a CompactClassificationEnsemble.')
```

Test: Prediction

Ensure a prediction is returned from the model using predictFcn.

```
5 load MLModels trainedModel
6 load MLData data
7 FaultType = trainedModel.predictFcn(data);
8 assert(length(FaultType) == height(data))
9 assert(iscategorical(FaultType))
```

Editor integration

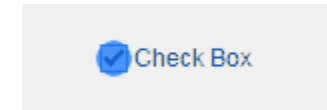


- Added buttons to make testing more readily accessible
- Testing your code should be as easy as hitting the “Run” button!

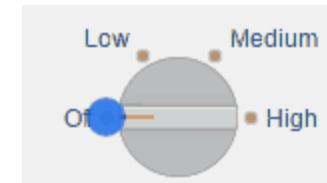
App Testing Framework

- Verify app behavior with tests that programmatically perform gestures on a UI component

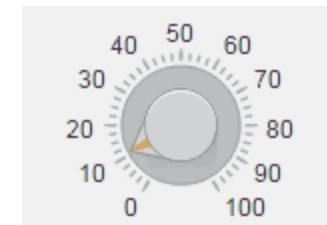
```
testCase.press(myApp.checkbox)
```



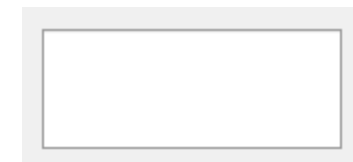
```
testCase.choose(myApp.discreteKnob, "Medium")
```



```
testCase.drag(myApp.continuousKnob, 10, 90)
```

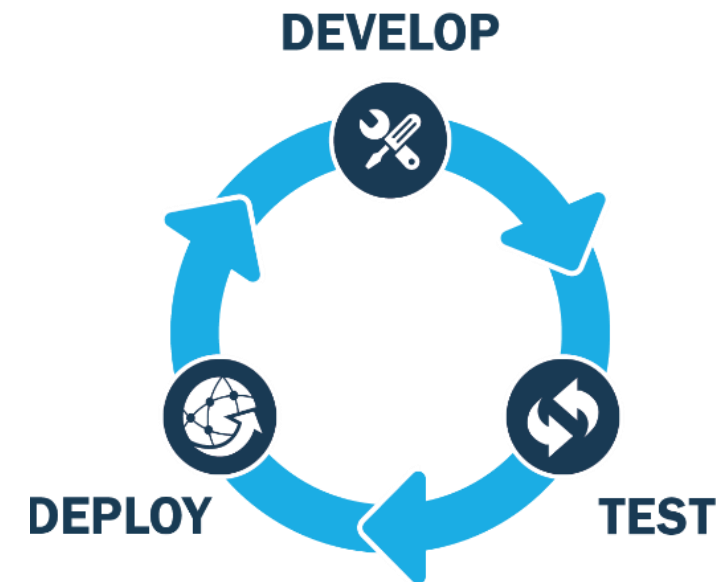


```
testCase.type(myApp.editfield, myTextVar)
```

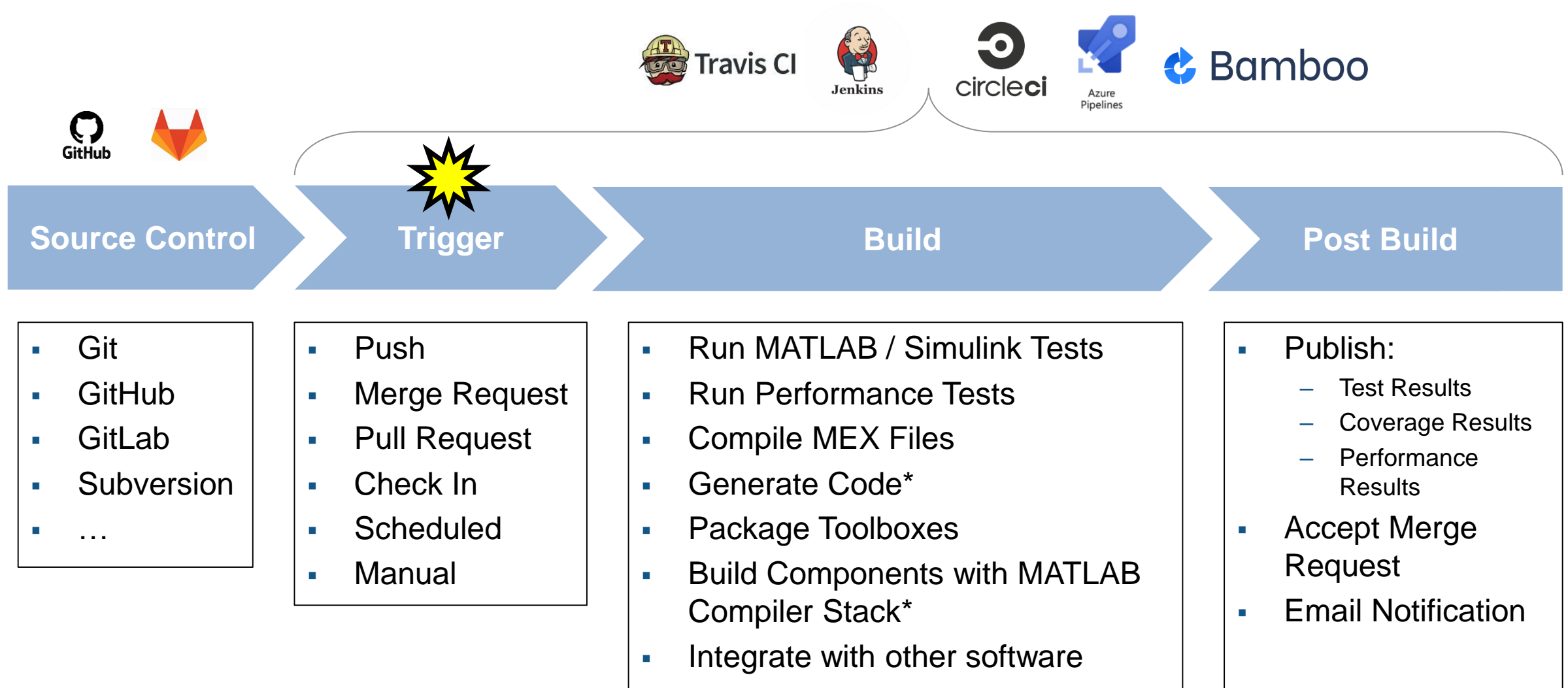


Continuous Integration (CI)

- A system to automate the building, testing, integration, and deployment of code as it is being developed and maintained
- Popular CI systems: Jenkins, Travis, CircleCI , Bamboo, and others...
- Benefits:
 - Detect integration bugs early
 - Allow you to stop bugs from being accepted
 - Track and report testing history
 - Flexible testing schedules and triggers



Continuous Integration workflow



Jenkins plugin



- Easily connect and configure MATLAB with Jenkins
- Schedule automatic code execution and testing:
 - based on time of day
 - whenever new code changes are committed

Plugins Index

Discover the 1000+ community contributed Jenkins plugins to support building, deploying and automating any project.

Browse Find plugins...

Browse categories

- Platforms
- User interface
- Administration
- Source code management
- Build management

New Plugins

- QRebel
- MATLAB**
- MISRA Compliance Report
- Zoom
- CodeBuilder: AWS CodeBuild Cloud Agents

Recently updated

- Mercurial
- VectorCAST Execution
- Klocwork Community
- OverOps Query
- LoadNinja
- QRebel

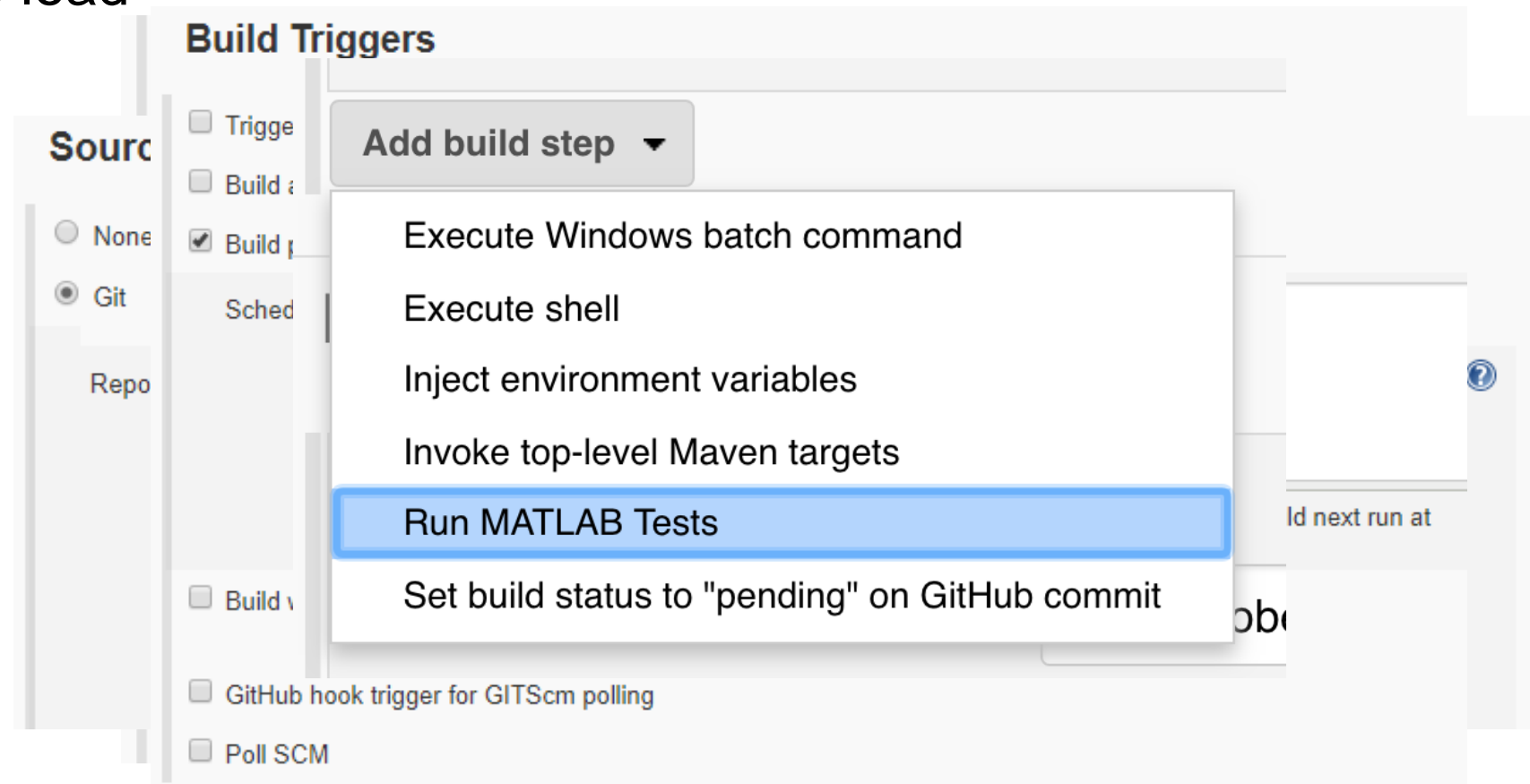
Trending

- jQuery UI
- Lockable Resources
- jQuery
- Analysis Model API
- Warnings Next Generation
- JDK Tool

Jenkins plugin configuration



- Locate MATLAB
- Identify repository to load
- Set build triggers
- Add build step



Jenkins plugin reports

- View testing results
- View code coverage
- View testing reports



Sharing your code – The traditional way

- Unzip the zip file
- Find the instructions and release notes
- Decide whether you want the thing
- Remove folders from old versions from the path
- Add folders to the path
- Save the path for next time
- Find the documentation
- Do work




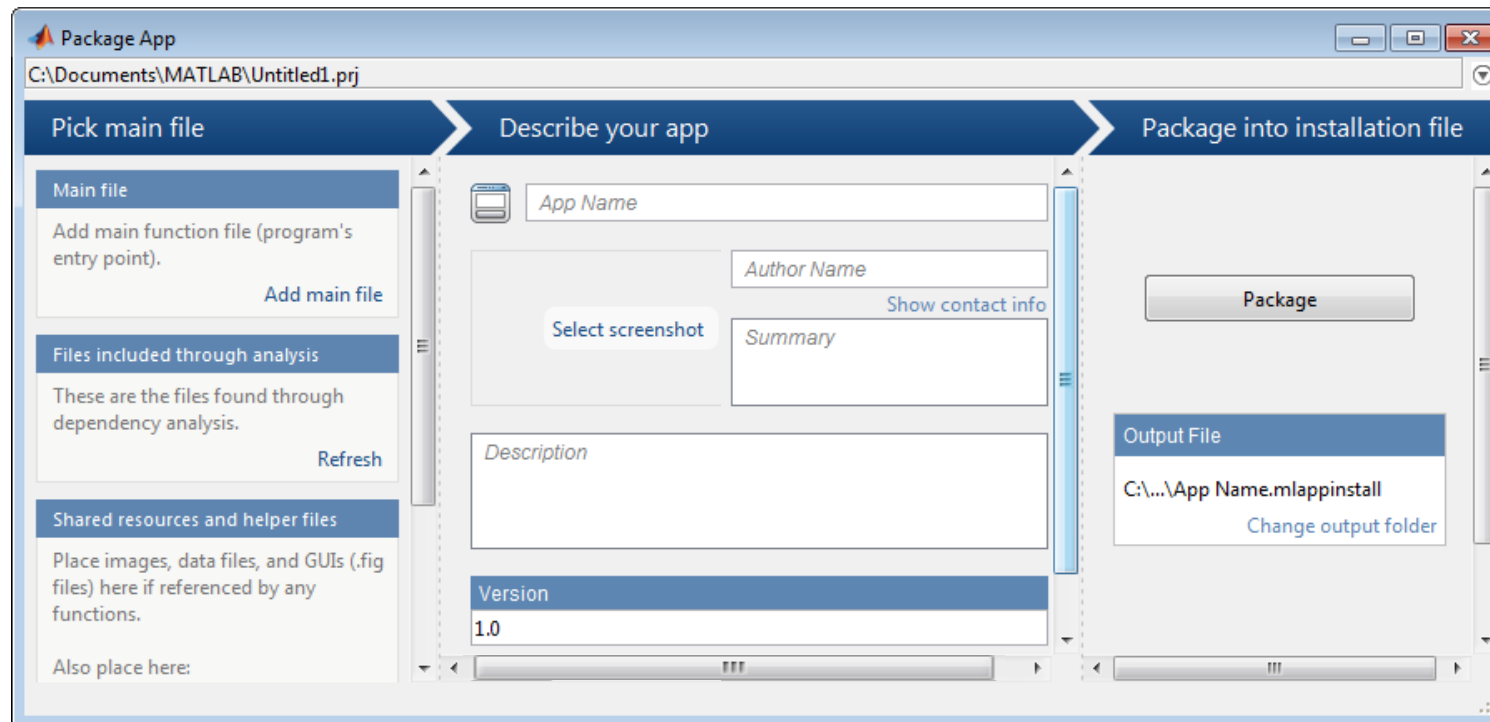
Sharing your code – How should you share code?

It depends on who you are sharing your code with:

- Co-authors → Project
- End-user with MATLAB → Toolbox or App
- End-user without MATLAB → Deployment (application, library, C code ...)

Sharing your code with MATLAB users – Packaging your code

- Toolbox Packaging
 - App Packaging
- 
- Combine files into one installation file
 - Installs in MATLAB Add-Ons or Apps tab
 - Documents required products



Sharing your code outside of MATLAB – Application Deployment

Share your applications as:

- Standalone software
- Web applications
- Language-specific libraries
- Generated code

MATLAB Compiler

MATLAB Compiler

MATLAB Compiler SDK

MATLAB Coder



Design Patterns

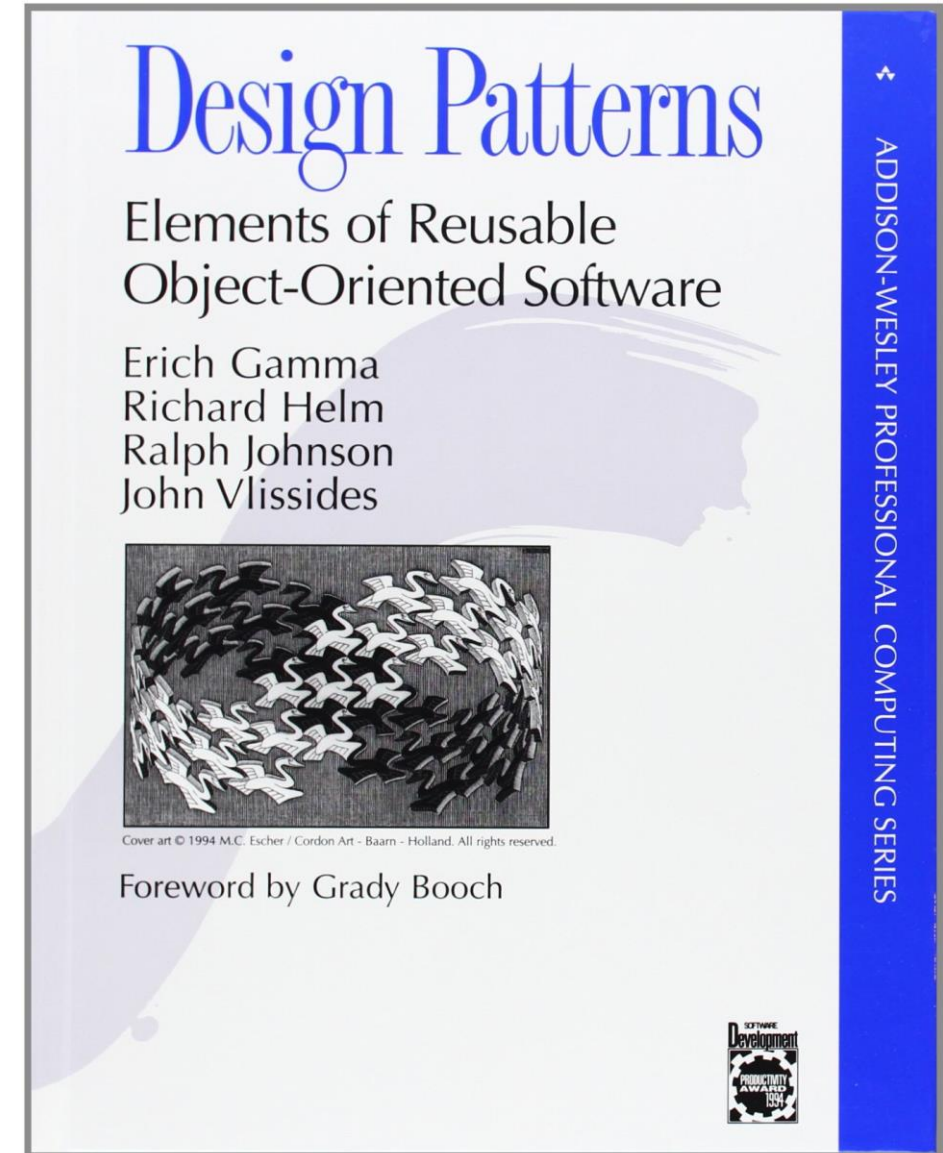
Preface: handles and values

- MATLAB has both value and handle classes
- Everyday MATLAB datatypes exhibit value behaviour
- Handle classes facilitate multiple references to the same object
- MATLAB's copy-on-write optimization limits memory consumption
- MATLAB's reference counter disposes of unused handle objects

Choose handle or value based on the need for multiple references.

Design patterns

- Observer
- Adapter
- Singleton
- Builder
- Memento



Observer pattern

- When an object changes state, how can an arbitrary number of dependent objects react?
- How to avoid making the objects tightly coupled?
- Handle class for subject
- Event(s) on subject, possibly with custom event data
- Observers listen to events on subject
- Example: model with multiple views



Adapter pattern

- How can a class be reused that does not have an interface that a client requires?
- How can classes that have incompatible interfaces work together?
- How can an alternative interface be provided for a class?
- Private property to store an instance of the reused class
- Dependent properties to forward gets and sets
- Wrapper generator using meta.class APIs
- Examples: chart, modified timer, map services



Singleton pattern

- How can it be ensured that a class has only one instance?
 - How can the sole instance of a class be accessed easily?
 - How can a class control its instantiation?
 - How can the number of instances of a class be restricted?
-
- Private constructor
 - Private property to store the object
 - `getInstance` static method
-
- Example: pointer manager



Builder pattern

- How can a class create different representations of a complex object?
- How can a class that includes creating a complex object be simplified?
- MATLAB handle class
- `create* method(s)`
- Example: create unit from database



Memento pattern

- How can the internal state of an object be saved externally so that the object can be restored to this state later?
- Saving to and loading from disk or database is a common case
- `saveobj` instance method
- `loadobj` static method
- `ConstructOnLoad` class attribute
- Examples: renaming a class, removing a property



Closing remarks

Recap

- MATLAB Projects
- Version control integration
- Language features
- Development environment
- Testing & CI
- Toolbox distribution
- Design patterns



Thank you.

Questions?