Software Development Practices within MATLAB

Christoph Kammer
Matthias Sommer
What are your software development concerns?

- Accuracy
- Software Speed
- Development Time
- Cost
- Compatibility
- Documentation
- Reusability
- Effective Testing
- Integration

- Ease of Collaboration
- Legacy Code
- Liability
- Maintainability
- Model Risk
- Robustness
- Developer Expertise
- Software Stack Complexity
- …?
Software development practices can help

Treat your software like an asset → reuse it

Developers often spend 4X the effort to maintain vs build software

…but this doesn’t need to be true!

Journal paper: “Faster issue resolution with higher technical quality of software”, Software Quality Journal, 201100
Software development practices can help

- Software development approaches like Agile help improve code quality
- The tools and practices we discuss today support Agile development
## Agenda

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing your code</td>
</tr>
<tr>
<td>Tracking code changes and co-authoring workflows</td>
</tr>
<tr>
<td>Writing better, robust, and portable code</td>
</tr>
<tr>
<td>Testing and maintaining your code</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
How do you currently manage your files and paths?

- One big folder of files?
- Many folders of files?
- Organize your code in packages?
- Manual path management?
Successful collaborative development requires …

- Same source code, tests, documentation, requirements, compiler…
- Consistent, shared environment
- Integration with source control
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
## Agenda

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing your code</td>
</tr>
<tr>
<td>Tracking code changes and co-authoring workflows</td>
</tr>
<tr>
<td>Writing better, robust, and portable code</td>
</tr>
<tr>
<td>Testing and maintaining your code</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
How do you keep track of and share your code as it changes?

- Do you:
  - make copies of your code?
  - e-mail yourself copies of your code?
  - keep a spreadsheet of changes?

- Or do you not keep track of your changes?

There’s a better way!
Source Control

- A system to manage changes to code, documents, etc.

- Benefits of source control:
  - Maintain backups, history, and ability to restore
  - Track changes and responsibility
  - Simplify reconciling conflicting changes
  - Generate discussion
  - Save you from yourself
Source 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
# Agenda

<table>
<thead>
<tr>
<th>Managing your code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking code changes and co-authoring workflows</td>
</tr>
<tr>
<td>Writing better, robust, and portable code</td>
</tr>
<tr>
<td>Testing and maintaining your code</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
What defines “better” code?

- Better organized?
- Smaller?
- Faster?
- More stable?
- More portable?
- Easier to maintain?
- …

YES!
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
- Sharing your code via apps, toolboxes, and deployment
- Integrating with other languages
- And more…
Writing more robust code

>> 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)
Writing more robust code – Validating inputs

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

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

```
>> 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.
```

```matlab
classdef ValidatorFunction
    properties
        Data(:,1) double {mustBePositive, mustBeFinite} = [1 2 3]
        Interp {mustBeMember(Interp,{'linear','spline'})} = 'linear'
    end
end
```
Writing more robust code – Handling errors more elegantly

- **error** and **warning**
  - Use identifiers

- **try/catch**

- **MException**

- **errordlg** and **warndlg**
Writing faster code – MATLAB Profiler

- Total number of function calls
- Time per function call
- Highlights largest code bottlenecks
- Statement coverage of code
Writing code faster – 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
Writing code faster – Quickly and safely refactoring code

- Live Editor shortcuts to refactor blocks of code into functions

```
function [z3, zSum] = myMathFunction(x, y)
z1 = x+y;
z2 = x-y;
z3 = y-x;
z4 = x*y;
zSum = z1 + z2 + z3 + z4;
end
```
Writing code faster – 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

```matlab
>> 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

```matlab
>> checkcode -cyc standardizeEmployeeInfo
L.1 (C 14-36): The McCabe cyclomatic complexity of 'standardizeEmployeeInfo' is 13.
```
Writing more portable code – 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

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

Windows:  ".\data\2019\April"
Mac/Linux:  "/data/2019/April"
```
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
Integrating with other languages – External interfaces

Calling Libraries Written in Another Language

- Java
- Python
- C/C++
- Fortran
- COM components and ActiveX® controls
- RESTful, HTTP, and WSDL web services

Calling MATLAB from Another Language

- Java
- Python
- C/C++
- Fortran
- COM Automation server
## Agenda

<table>
<thead>
<tr>
<th>Managing your code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking code changes and co-authoring workflows</td>
</tr>
<tr>
<td>Writing better, robust, and portable code</td>
</tr>
<tr>
<td>Testing and maintaining your code</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Code Maintenance – The hidden cost of development

- How do you ensure code doesn’t break over time?
- How do you keep new features from breaking existing features?
- How do you maintain confidence that your code is working as expected?
Upgrading to the latest MATLAB – 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

[Link to documentation for updates]

[Go directly to the line of code]
Test early, test often, test automatically

- Reduce risk of code breaking
- Catch problems early
- Improve code quality
- Document expected behaviour

Credit: http://geek-and-poke.com/
Testing Frameworks

*Test your code early and often*

- MATLAB Unit Testing Framework
- Performance Testing Framework
- App Testing Framework
Testing Frameworks – Flexible development

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

**Test Pump Fault Model**
This includes unit tests for the predictions

**Test: Model type**
Load the models and ensure they are the right types.

```matlab
load MLModels trainedModel
dml = trainedModel.ClassificationEnsemble;
assert(isequal(dml,'classreg.learning.classif.CompactClassificationEnsemble'),...
'Model is not a CompactClassificationEnsemble.')
```

**Test: Prediction**
Ensure a prediction is returned from the model using `predictFcn`.

```matlab
load MLModels trainedModel
data = MLData;
FaultType = trainedModel.predictFcn(data);
assert(length(FaultType) == height(data))
assert(iscategorical(FaultType))
```
Testing Frameworks – Easily customize and run existing tests

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

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

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

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

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

testCase.type(myApp.editfield, myTextVar)
```
Automated Testing – 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
Automated Testing – Continuous Integration workflow

Source Control

- Push
- Merge Request
- Pull Request
- Check In
- Periodic
- Manual

Trigger

- Run MATLAB / Simulink Tests
- Run Performance Tests
- Generate Code
- Package Toolboxes

Build

- Publish:
  - Test Results
  - Coverage Results
  - Performance Results
- Accept Merge Request
- Email Notification

Post Build

Continuous Integration System
## Agenda

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing your code</td>
</tr>
<tr>
<td>Tracking code changes and co-authoring workflows</td>
</tr>
<tr>
<td>Writing better, robust, and portable code</td>
</tr>
<tr>
<td>Testing and maintaining your code</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Key Takeaways

- You will save you time, effort, money, and frustration with good software development practices.

- MATLAB provides tools that enable agile software development.

- We’re adding more software development tools and features every release!
MATLAB

is the easiest and most productive environment for engineers and scientists
Related Trainings

- **Writing Robust Code**
  - [MATLAB Programming Techniques](https://nl.mathworks.com/services/training.html)

- **AppDesigner**
  - [Building Interactive Applications in MATLAB](https://nl.mathworks.com/services/training.html)