MATLAB as a Medium for Complete Tool Development

Prepared By,
Anusuya Devi Mani
Sowmyaa Maruthanayagam
Agenda

1. Need for Automation
2. Tool Development
3. Testing & Validation
4. Packaging
5. Conclusion
Advancements and Growth in Automotive Industry

Increased size and complexity of software in vehicle

Moving towards Model Based Development

Need to make the process of development fast

Need to reduce Manual Errors to make it robust

The End Solution is AUTOMATION

We are involved in many developments using Simulink and Stateflow, which involves repeated activities. In order to improve the Quality and Effective Time Utilization we have automated processes.
**Tool Development Phases**

**Command Line**
Application Developed using m-script and Executed via command line

**Graphical UI**
Application developed using GUIDE tool and Executed via a Graphical User Interface in MATLAB

**Stand-Alone App**
Application developed using GUI can be created as Stand-alone using packaging tools

**Tool Development Methods**

- GUIDE Tool Development
- M-Scripting
- Packaging
- Automation
- MATLAB Coding
- Code Analyzer
- Profiler
- MATLAB Compiler
GUIDE – Graphical User Interface Development Environment

- To Develop A Graphical User Interface
- User friendly
- Simplicity of layout
- Automatic Code Generation
- Easy Maintainable
# GUIDE for Automation

<table>
<thead>
<tr>
<th>Without GUIDE</th>
<th>With GUIDE</th>
</tr>
</thead>
</table>
| • Command line execution  
  – With commands | • Graphical User Interface  
  – With Simple Mouse-clicks and Key Presses |
| • Needed Knowledge about syntax for execution | • No syntax knowledge required |
| • Possibility of more Manual errors | • Less chances for Manual errors |
| • No code automations possible | • Programming made easy with automations |
| • Availability of Options for the user is less | • Logos, Mandatory Inputs, Note, Shortcuts, etc. can be added in the GUI |
| • Stand alone application is not possible | • Supports Stand alone application |
| • Not user friendly | • User friendly Interface |
GUIDE Highlights

- Menus & Short cuts
- Property Inspector - Tool Tips
- Automatic Code Generation
- Interfacing
- Multiple GUIs
Menus & Shortcuts

- Menu Editor - To create Menus and their shortcuts
- Shortcuts - Access Menus using Key Press
- Easy access for the User
- No Programming Required
Tool Tips

• Supports the User for usage of the Application/Tool
• Tool tips can be added for every feature in GUIDE
• Tool tips option is present in the property inspector window
• No programming required

NOTE: Property Inspector also has other features like Display properties, Callback functions, Enable/Disable
Once developed GUI is saved, the below are auto generated:

- Main GUI Code
- Function Callback Definitions
- Reduces Developer's Effort
Example:
To generate an excel file with specific properties

```plaintext
objExcel = actxserver('Excel.Application');
objExcel.Workbooks.Open('Excel_app.xls');
objExcel.ActiveWorkbook.Worksheets.Item([sheetName '3']).Delete;
objExcel.ActiveWorkbook.Save;
objExcel.ActiveWorkbook.Close;
```
• File read and write options help users for saving output for future reference in a good format.
• File Read - User can modify their inputs easily in files
• Read/Write Formats - .txt, .xls, .xlsx, .xml, .csv, .bmp, .gif, .jpg, .jpeg

**Example:**
To write data to an Excel Application Outside MATLAB Environment:

```matlab
xlswrite('Excel_app.xls','90','Sum','B3');
```

To Read the above Excel File:

```matlab
xlsread('Excel_app.xls');
```
Multiple GUIs

Multiple GUIs can be interfaced in a single application using GUIDE.

**Example:**
The Multiple_GUI launches 3 different GUIs using UI Controls, where the corresponding m-file is run.
Profiler

- First Step in MATLAB Code Analysis.
- Helps for Performance tuning.
- Gives the function flow and execution time of every line of the developed code.
- Highlights the lines of code and functions consuming more time
- Details on the number of function calls
- Also gives the details of warnings
- Helps greatly in optimization and performance enhancement

For our automated applications, Profiler made us to know about the following

- Time taken for each function and the functions that consumes time
- Number of function call
- Code lines that are executed
- Grey areas of code

RESULT:

- Removed bugs which in turn improved quality
- Optimization of code which enhanced performance
Code Analysis

Code Analyzer Report

- Analyses the code i.e. m-script developed.
- Lists potential errors and problems.
- Suggests opportunities for improvement in code.
- Details on the exact line number and link to it.
- Also provides suggestions to fix the issues.
- Provides instant alerts during programming when set to run during coding.
- Reduces the review effort greatly.

Below improvements were suggested by code analyzer for our application

- The array size that are to be pre-allocated for speed
- Unused variables, functions that are to be removed

RESULT:

- Performance of the tool was improved 4 times
- The reviewers effort was reduced 3 times
How do we share the App/Tool?

Executable using Matlab Compiler:

Some functions are not supported in standalone mode as below:

- Functions that print or report MATLAB code from a function
- Simulink® functions, in general.
- Functions that require a command line.
- clc, home, and savepath will not do anything in deployed mode.
- Tools that allow run-time manipulation of figures

Steps to create stand-alone application using Matlab Compiler:

1. **Open Deploy tool**
2. **Add Main script file & supporting files (scripts and fig files)**
3. **Build the function (invokes 'mbuild')**
4. **Outputs**
   - Readme.txt
   - mccExcludedFiles.log
   - Project.exe
5. **Stand-alone Application Created**

Encrypt script files:

- Converting M files to P-Code i.e. Protected function file.
- P-code is secure and non-readable
- How to convert?
  - `pcode file_name` -> To convert file_name.m to file_name.p
  - `pcode folder_name` -> To convert all the m-files present inside the folder ‘folder_name’ into p-code.
- A simple and easy way to share the application.
Conclusion

- Automation tools developed for applications like MATLAB Model Size measurement, Memory size measurement, etc.,

- MATLAB tools helped us in every phase of development to achieve a quality tool

- Creating such applications via MATLAB
  - Made our development process easier
  - Reduced development effort & time
  - Enhanced quality & performance
  - Easy Maintainability

*MATLAB a Complete Platform for Automation*
Thank You

We Know You Have Them ...!!!