iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** V **Month of publication:** May 2019

DOI: https://doi.org/10.22214/ijraset.2019.5205

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue V, May 2019- Available at www.ijraset.com

Model Inspection Tool for Verification of MATLAB Model Guidelines in Automotive Software Development

Sreeraj S
Software Engineer, Continental Automotive Components Pvt Ltd, India

Abstract: ***Model based development and auto code generation is now the widely accepted approach in the field of automotive software development. In order to make model perfectly readable, for easy maintenance and to ease reusing, testing, expanding and exchanging the models between suppliers and OEMs, certain modelling standards should be maintained among the companies.***
This is the purpose of MathWorks Automotive Advisory Board (MAAB) guidelines. The automotive companies have agreed to follow these guidelines while developing the feature models. Before releasing or delivering any MATLAB model, it is necessary to verify that any MAAB guideline is violated or not. For a model having lot of subsystems, ports, line connections and levels, manual checking of any violation is time consuming and there is higher chance of missing the simple errors. This paper explains the development of a Graphical User Interface (GUI) tool which takes the MATLAB model as input and checks for any violation of MAAB guidelines.
The tool is named as 'Model Inspection Tool' and is developed using MATLAB m-script. This tool checks for only some selected guidelines from the MAAB document and not for every point given in it. Thereby developer can ensure the delivery of MAAB guideline compliant models.
Keywords: MAAB guidelines, get_param, set_param, find_system, MATLAB GUIDE

I. INTRODUCTION

For over 15 years, the world's leading automotive companies have employed Model-Based Design with MATLAB, Simulink, and Stateflow to solve development and testing challenges in the areas of engineering analysis, modelling and simulation, rapid prototyping, automatic code generation, and system verification and validation[1]. The efficiency and quality of the software or the C code are dependent on quality of the model. Modelling the functionality of an automotive feature using Simulink/Stateflow does not ensure the production of high quality code. For this reason, in 1998, a group of companies which involved FORD, Daimler Benz and Toyota jointly defined a set of modelling guidelines. MathWorks hoisted the guidelines. This is called MathWorks Automotive Advisory Board (MAAB) guidelines. Now many automotive OEMs and suppliers are involved in MAAB. Also, each OEM maintains its own modelling standards and it is informed the developers or the suppliers who develop the model.

If the model is complex and bigger in size, it is difficult for the developer to sit and manually check for any violation of these guidelines or standards. Also, it is a boring task. But model must be thoroughly checked of guideline compliance before its delivery because even a small violation can create a major setback to the supplier or developer. So, this paper deals with the development of a tool which automates the checking of modelling guidelines in a given MATLAB model. This tool is developed using MATLAB m-script. This is a Graphical User Interface (GUI) which takes the MATLAB model as input and publish an excel report which shows the violated guideline along with its path and place. As a first phase this tool will check only some 24 selected guidelines. The user can either check all the points in one execution or can select only the required guidelines and do the checking.

In the upcoming section, categories and examples of MAAB guidelines are briefed. In section 3, the development approach of the Model Inspection Tool is explained. The MATLAB functions and approach used for checking each guideline is explained in that section.

II. EXAMPLES OF MAAB GUIDELINES

In this section various categories of modelling guidelines are discussed[1]. Different categories and its goals are mentioned in Table 1. An example of a wrong modelling practice and its right method is shown in Fig 1 and Fig 2 respectively. Examples used here are guidelines for *arithmetical problems* and *model layout.* Below given are some guidelines which can be applied to achieve a better readability.

©IJRASET: All Rights are Reserved

A. Inport blocks should be placed on the left hand of a model and Outport blocks to be placed on right hand side.
B. Blocks or names of the blocks should not hide by other blocks.
C. Signal lines should not cross and signal flow should be from left to right.
D. Do not use number directly inside a model. Instead define constant variables for the numbers and then use the variables in the model.

In Fig 1, a product block with three inputs is shown. This practise will create problems when the model converted into a fixed-point code using code generator. So, this way of modelling should be avoided. Instead two product blocks with each having two inputs can be used. This guideline comes under the category of *Arithmetical problems.*

Category	Aim/Goal
Model layout	Increase portability, readability and maintainability.
Arithmetical problems	Prevent usual arithmetical problems (e.g. Fixed-point issues, division by zero).
Exception handling	Increase robustness of the model and the code that is generated.
Project specific guidelines	Guidelines which a particular project or a company follows. Naming conventional is one such example.

TABLE I. CATEGORIES OF MAAB GUIDELINES

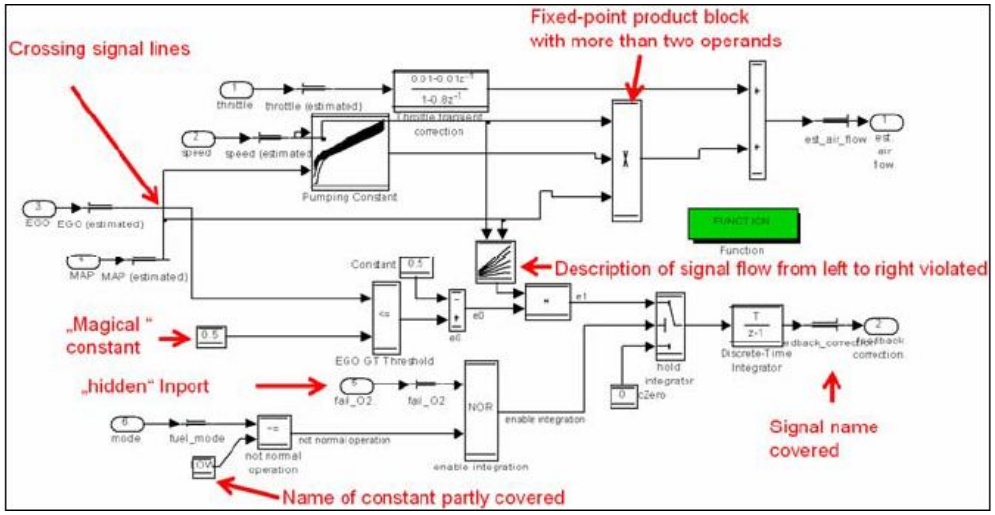


Fig. 1 Examples of wrong practise modelling indicated in red

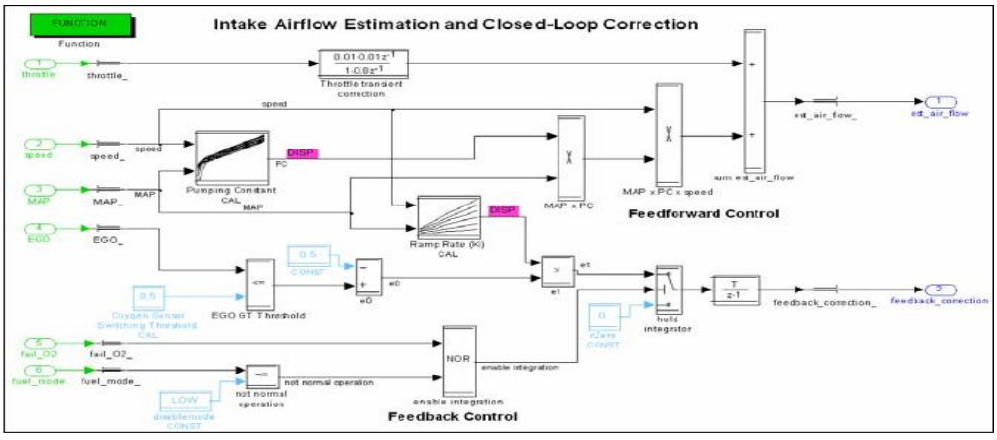


Fig. 2 Violations shown in Fig 1 is corrected as per guidelines

©IJRASET: All Rights are Reserved

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue V, May 2019- Available at www.ijraset.com

III.DESIGN OF MODEL INSPECTION TOOL

The 'Model Inspection Tool' is developed for Ford motors to automate the checking of MAAB guideline violations and to apply some Ford specific guidelines in the model. This tool is developed using Matlab GUIDE application. The front end of the tool is shown in Fig 3.

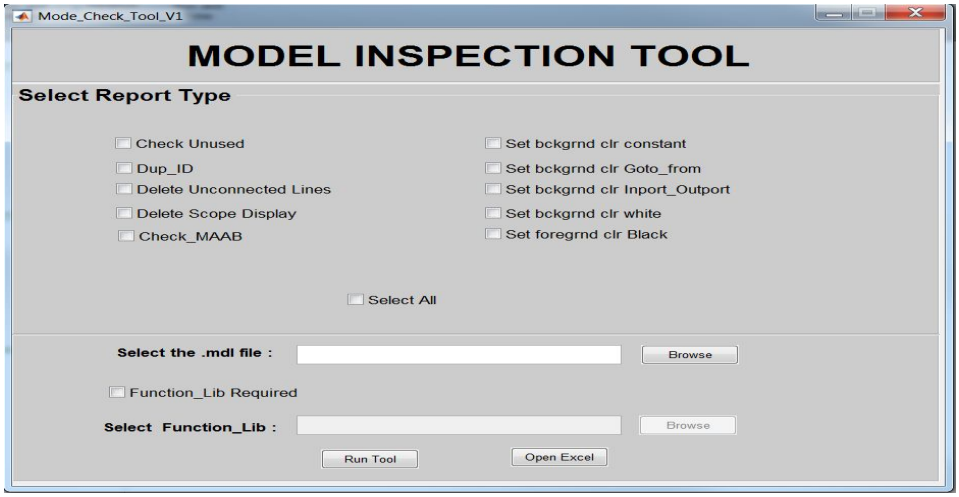


Fig. 3 Front end of Model Inspection Tool

As shown in Fig 3, several actions are provided with check boxes. User can select the actions which he wants to apply on the model. Among the given options, some are only checking and some are modifications on the model. For example, *Dup_ID* will check for the presence of blocks with same Ids and list the names of such blocks in the report. But '*Delete Scope Display*' will delete the scope and display blocks in the mode. User can select '*Select All'* so that all modifications and checks will be done on the model. The option '*Check_MAAB*' will check for the violation of some selected MAAB guidelines in the project.

After selecting the required actions to be done, user should browse and select the model to be checked. Selected model name along with path will be displayed at text box against '*Select the .mdl file*'. Then click on *Run Tool* button to start execution.

After the checking or modifying the model, user can open the excel report by clicking on *Open Excel* button. The excel report will have the detailed report of the model modification or guideline checking in the model. The report will have the name and path of the blocks which violated the guidelines along with the name of guideline that the block violated.

The back end of this graphical user interface is a function having function calls corresponding to each object on the GUI.

For example, *Browse* button will have a corresponding function call which defines the action on pressing it.

Apart from the check box *Check_MAAB*, all other options are guidelines which are specific to Ford. In the following sections, each option on the GUI and its algorithm is briefed.

A. Check Unused

This is a Ford specific guideline[2]. Every Simulink model which are developed in Ford will have an excel workbook called Data dictionary associated with it. Data dictionary will have the details of all the signals, constant variables and tuneable parameters used in the model. Check Unused will check whether all parameters defined in the Data dictionary are used in the model. The MATLAB functions 'find_system' and 'get_param' are used to find all the parameter names used in a model. And the MATLAB function 'xlsread' is used to read all the model parameters defined in excel file named 'Datadictionary.xls'. Then the tool will check whether all the parameter it got from the excel file is available in the model. If not available, such parameters will be listed in the output excel report.

Below shown is an example code to get the parameters used in all the Gain blocks used in the model.

```
gb= find_system(system,'LookUnderMasks','all','FollowLinks','on','BlockType', 'Gain');
gbv = get_param(gb,'Gain');
```

Here variable gbv will have the list of gain value parameters used in all the Gain blocks.

Similarly, parameters used in the Constant, From, GoTo and Saturate blocks are obtained. Then the parameters defined in Data dictionary is obtained in the variable Param as shown below.

```
Param = xlsread('DataDictionary_WWA.xls','A4:A130');
```

©IJRASET: All Rights are Reserved

B. Dup_ID

Ford has insisted to have unique Id number for each and every block used in a Simulink model[2]. This unique Id is given in the description property of a block. A block has several properties and *Description* is one of that. So, enabling this option will check for blocks having duplicate Ids. Using below functions, Description of all the blocks used in a model is obtained in the variable Id.

```
SubSystems = find_system(System,'LookUnderMasks','all');
Id = get_param(SubSystems,'Description');
```

System is the modelId is a cell array having the list of Ids of all the blocks in the model. The tool will check for duplicate Ids in the array. If found, it will publish the name and path of the blocks having similar Ids.

C. Deleted Unconnected Lines

Enabling this option will delete all the unconnected lines present in the model. Using below command the tool gets list of all the lines in the model.

```
Lines=find_system(system,'LookUnderMasks','all','FindAll','on','Type', 'line' ) ;
```

Then the below algorithm is applied to delete the lines which does not have either a source or destination port.

```
if  get( line, 'SrcPortHandle' ) < 0
delete_line( line ) ;
end
if  get( line, 'DstPortHandle') < 0
delete_line( line ) ;
end
```

D. Delete Scope Display

Scope and display blocks are used for analysis purpose. They are not part of the algorithm. So, it is mandatory to delete those blocks before a model delivery[2]. Enabling *Delete Scope Display* option will delete all the scope and display blocks present in a model. Using the MATLAB function '**find_system**', list of all the scope and display blocks will be obtained. Then using the function 'delete_block' all those blocks can be deleted from the model.

E. Check_MAAB

Enabling this option will check for the violation of following MAAB guidelines[3].

1) Prohibited Characters In Block Names[3]

As per MAAB guidelines, some characters are prohibited from using in a block name. Tool checks the presence of prohibited characters in a signal name using the MATLAB function '**regexp**' with the prohibited characters as function arguments. If prohibited characters are found, then the rule is violated. For example , MAAB doesn't allow the usage of '@' in block names. To check whether '@' is used in aparticular block name, tool use the code below.

```
Pos = regexp(Name, '@');
```

Pos will give the position in the name where '@' is used.

2) Basic Block Names Should Be Hide[3]

As per MAAB guidelines,for all basic blocks except Subsystem, Inport and Outport, block names should be hidden.

The tool will do this using following code.

```
set_param(h1,'ShowName','off');
```

h1 is the block name.

3) Subsystem name position should be below the block[3].

This guideline is implemented using following code .

```
set_param(h1,'NamePlacement','normal');
```

h1 is the subsystem name.

4) Inports and Outports should be aligned properly[3].

Using 'find_system' names of all Inport and Outport blocks can be obtained. Then position is found using 'get_param' function with 'Position' as the argument. An example code is shown below.

```
Inport_Pos=get_param(Level1_In,'Position');
```

Inport_Pos has the positions of all the Inport block. If all the position is not same, the guideline is violated and will be published in the output report.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue V, May 2019- Available at www.ijraset.com

5) Diagnostic parameters should not be set as none[3].

MAAB guidelines strongly recommends some diagnostics to be enabled[3]. An enabled diagnostic is set to warning or error. Setting those diagnostic options to *none* is not permitted. Below given is the code for checking whether the setting for Algebraic loop is enabled or not.

```
cs = getConfigSet(system,'Configuration');
Settings=get_param (cs, 'AlgebraicLoopMsg');
```

Using the MATLAB function getConfigSet, model configuration is obtained in the variable cs. Using the argument AlgebraicLoopMsg , the current setting of Algebraic Loop is obtained. If the setting is none, the guideline is violated.

Similarly, using the arguments Artificial Algebraic Loop Msg, Multi Task Rate Trans Msg, SignalInfNan Checking, UniqueDataStoreMsg,UnconnectedInputMsg,UnconnectedInputMsg,UnconnectedLineMsg,RootOutportRequireBusObject, StrictBusMsg, InvalidFcnCallConnMsg and BusObjectLabelMismatch, settings of the other diagnostic parameters can be obtained.

6) Port name should contain only allowed characters[3].

Tool checks the presence of prohibited characters in a port name using the MATLAB function '**regexp**' with the unallowed characters as function arguments. If prohibited characters are found, then the rule is violated.

7) For signals in a model, the property *'Show propagated Signal'* must be enabled[3].

MAAB guidelines insist that signal names of propagated signals should be displayed from source to destination of that signal. The tool enables this property of the signal is by following code.

```
signalLines = find_system(system,'FindAll','on','type','line');
set(signalLines,'signalPropagation','on');
```

Variable signallines gets all the signal handles of the model. Then using 'set' function, set the line property *signalpropagation* to *on*.

8) Signal name should not contain prohibited characters[3].

Tool checks the presence of prohibited characters in a signal name using the MATLAB function '**regexp**' with the prohibited characters as function arguments. If prohibited characters are found, then the rule is violated.

9) Subsystem name should not contain prohibited characters[3].

Tool checks the presence of prohibited characters in a subsystem name using the MATLAB function '**regexp**' with the prohibited characters as function arguments. If prohibited characters are found, then the rule is violated.

10) Model appearance setting should be as per following the MAAB guidelines[3].

The tool will set the model appearance using the code below. The argument corresponding to all parameters and its settings are shown in the code. 'set_param' is the MATLAB function used.

```
set_param(system,'ModelBrowserVisibility','off','ScreenColor','White','StatusBar','on','ToolBar','on','ZoomFactor','FitSystemToView
','ExecutionContextIcon','off','LibraryLinkDisplay','none','ShowPortDataTypes','off','ShowLineDimensions','off','ShowStorageClass',
'off','ShowTestPointIcons','on','ShowViewerIcons','on','WideLines','on');
```

F. Set bckgrnd clr constant

Enabling this option will set the background colour of the constant blocks in the model to the colour specified by the company. This is a Ford specified guideline[2]. The tool changes the colour of constant blocks in the model to light blue using the below code.

```
 Constants = find_system(system,'BlockType','Constant');
Colour= set_param( Constants,'BackgroundColor','lightblue');
```

G. Set Bckground clr Goto_from

By enabling this option the tool will set the colour of all 'Go to' blocks to cyan and 'From' blocks to magenta using the code below.

```
GoTos = find_system(system, 'BlockType','Goto');
set_param(GoTos,'BackgroundColor', 'cyan');
Froms = find_system(system, 'BlockType','From');
set_param(Froms,'BackgroundColor', 'magenta');
```

H. Set bckgrnd clr Inports_Outports

By enabling this option the tool will set the colour of all 'Inport' blocks to red and 'Outport' blocks to green using the code below.

```
Inports = find_system(system,'BlockType','Inport');
Colour= set_param( Inports,'BackgroundColor','red');
Outports = find_system(system,'BlockType', 'Outport');
```

©IJRASET: All Rights are Reserved

Colour= set_param(Outports,'BackgroundColor','green');

I. Set Bckgrnd Clr White

By enabling this option, the tool will set the background colour of the model to white using the code below.

```
Systems = find_system(system);
set_param(Systems,'BackgroundColor','white');
```

Here system used as the argument of 'find_system' is the model name with extension.

J. Set foregrnd clr Black

By enabling this option, the tool will set the foreground colour of all the blocks in the model to black using the code below.

```
Systems = find_system(system);
set_param(Systems,'ForegroundColor','black');
```

Here the variable system used as the argument of 'find_system' is the model name with extension.

IV. CONCLUSIONS

Embedded systems continue to become larger and more complex. At the same time, design teams are becoming more dispersed, both geographically and in terms of team members' skills and experience. In this challenging development environment, implementing modelling guidelines to ensure modelling consistency is vital. Modelling guidelines establish a homogenous approach within the design team, making it easier to reuse the models for new projects.

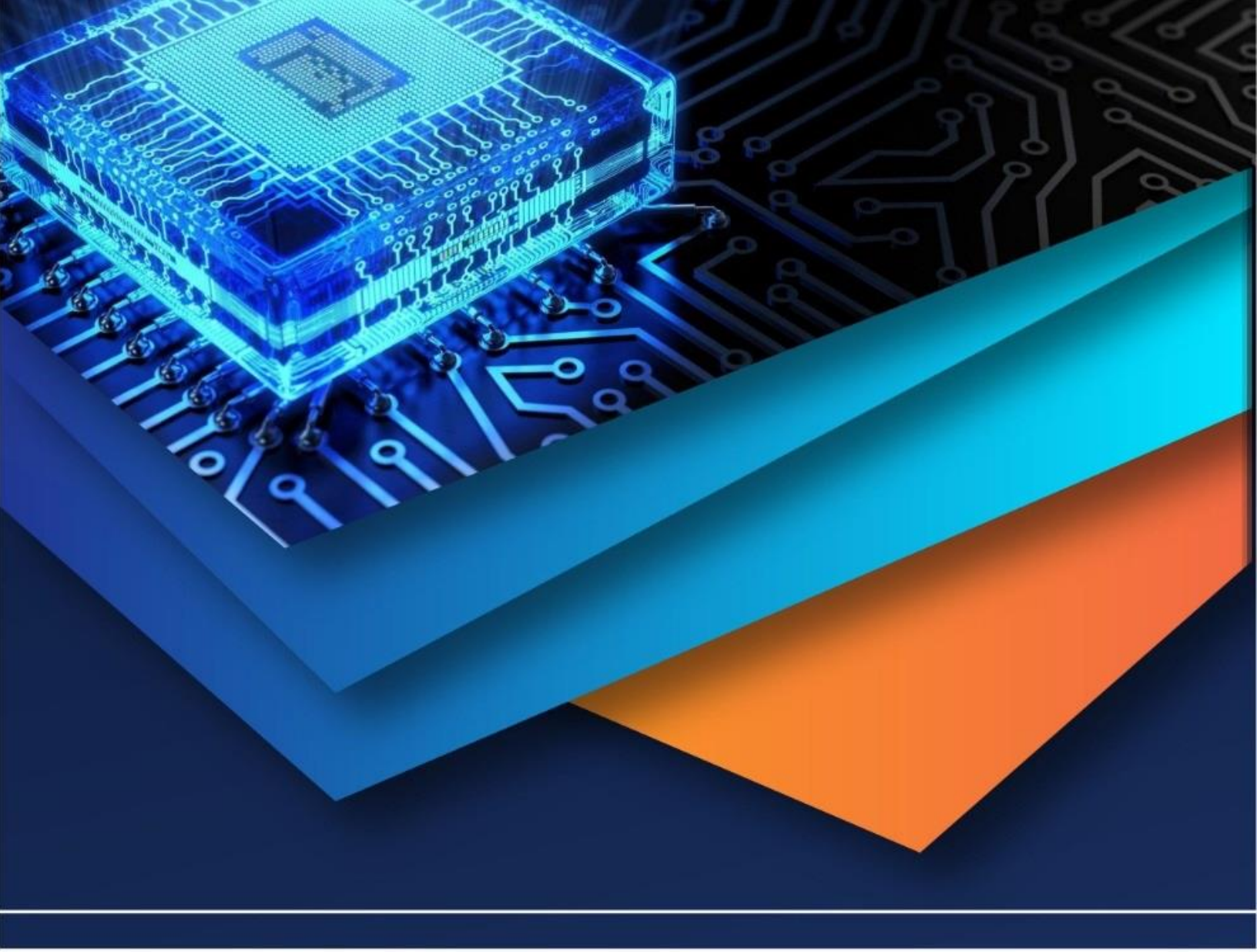
The Model Inspection Tool which is discussed in this paper supports the developer in checking compliancy of the model with MAAB guidelines. The major advantage of the tool is that it automated the process of model review and it produce a report showing the guideline violations in the model. For some guidelines, the tool is designed to correct the model or modify the model.

Adding more MAAB checkpoints on Simulink design and adding checks on Stateflow implementations will be the future work with respect to the tool.

REFERENCES

[1] Stürmer, Ingo & Dziobek, Christian & Pohlheim, Hartmut. (2008). Modelling Guidelines and Model Analysis Tools in Embedded Automotive Software Development. 28-39.
[2] Ford Model-based Development Guidelines
[3] MathWorks Automotive Advisory Board, Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow, Version 2.0", 2007.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)