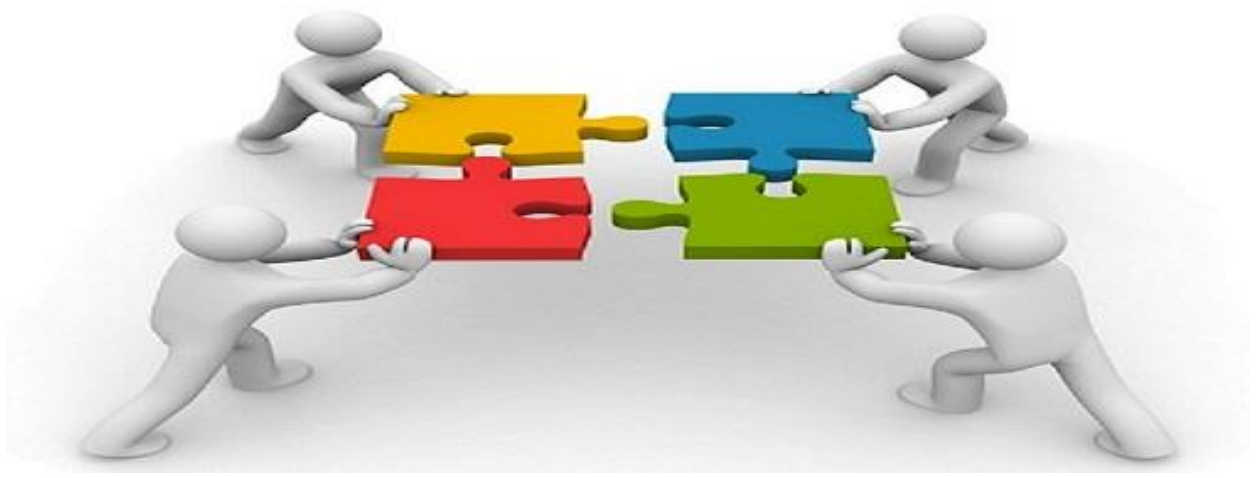


Version [1.0]

[12/10/2014]

CPU_Plugin Generator

User Manual

Presented By: Nishanth Prakash

University of Arizona

Tucson, AZ-85719

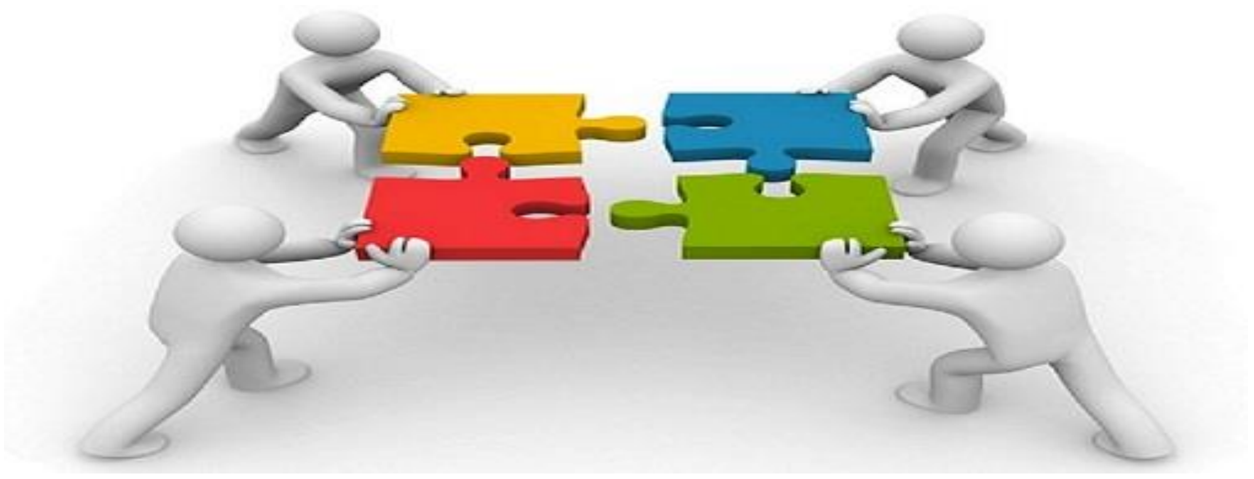
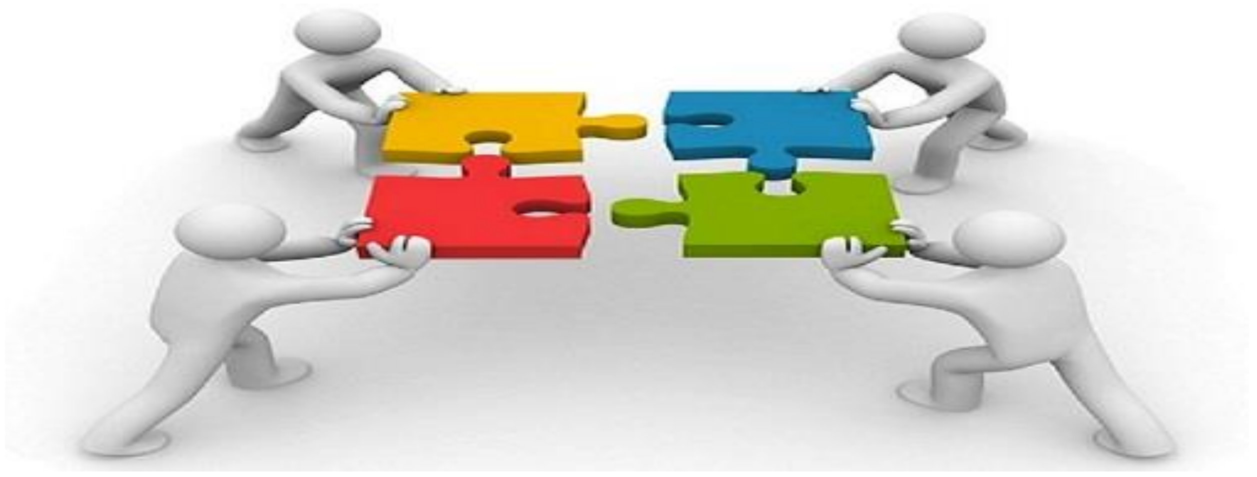


Table of Contents

Table of Figures

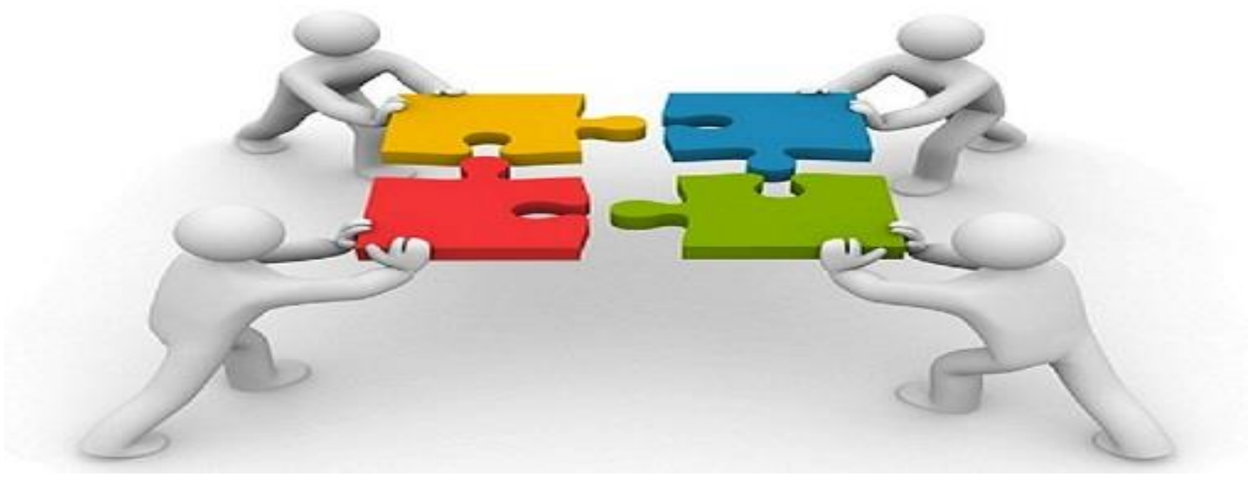


1.0 General Information

- Nagios plugins are standalone extensions to Nagios Core that provide low-level intelligence on how to monitor anything and everything with Nagios Core.
- Plugins operate as standalone applications, but are generally designed to be executed by Nagios Core.
- Plugins process command-line arguments, go about the business of performing a specific type of check, and then return the results to Nagios Core for further processing. These plugins are generated by the CPU_Plugin Generator tool.

1.1 System Overview

- The tool would take a single input of type String. The end user needs to select one of the options from the drop down menu in the Source Attribute List in the Object Inspector Window.
- The user would need to specify the type of program that needs to be generated by the tool.
- For example: If the end user wants the tool to generate a program to count the User Process Counter, the input would be “UserProcessCounter”
- If the end user wants the tool to generate a program to count the CPULoadComparison, the input would be “CPULoadComparison”
- If the end user wants the tool to generate a program to count the Memory_Usage, the input would be “Memory_Usage”
- If the end user wants the tool to generate a program to count the Memory Load, the input would be “MemoryLoad”
- If the end user wants the tool to generate a program to count the Shared Folders, the input would be “Shared_Folders”
- Once selected the end user can run the tool by pressing the “x” mark button on the top.
- The program is generated in a newly created test file which is present in the same folder as the tool.
- The generated program is designed to be used along with the Nagios tool and the NRPE extension. This generated program is treated as an added feature to the Nagios tool.

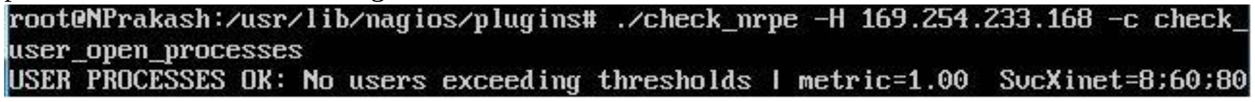


- Nagios tool along with the NRPE (Nagios Remote Plugin Executor) when run the generated code on any computer system, the program would return appropriate results.
- These results can be used by the system administrators to keep a check on the user's activity.

1.2 Functional Description of the Tool

1.2.1 User Process Counter

- When the end user selects the input as UserProcessCounter, the tool would generate a program which is used count the number of open processes in the computer system.
- The generated UserProcessCounter code when run by the Nagios tool along with the NRPE produces the result shown in Figure 1.

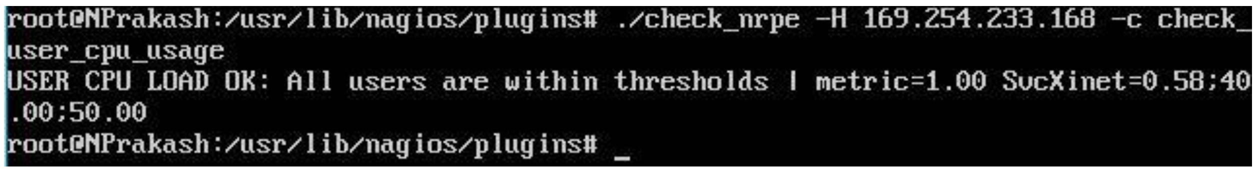


```
root@NPrakash:/usr/lib/nagios/plugins# ./check_nrpe -H 169.254.233.168 -c check_
user_open_processes
USER PROCESSES OK: No users exceeding thresholds | metric=1.00  SvcXinet=8;60;80
```

Figure 1: Output of Generated OpenProcess Code

1.2.2 CPU Usage

- When the end user selects the input as CPULoadComparison, the tool would generate a program which is used to measure the current CPU usage of the computer system.
- The generated CPULoadComparison code when run by the Nagios tool along with the NRPE produces the result shown in Figure 2.

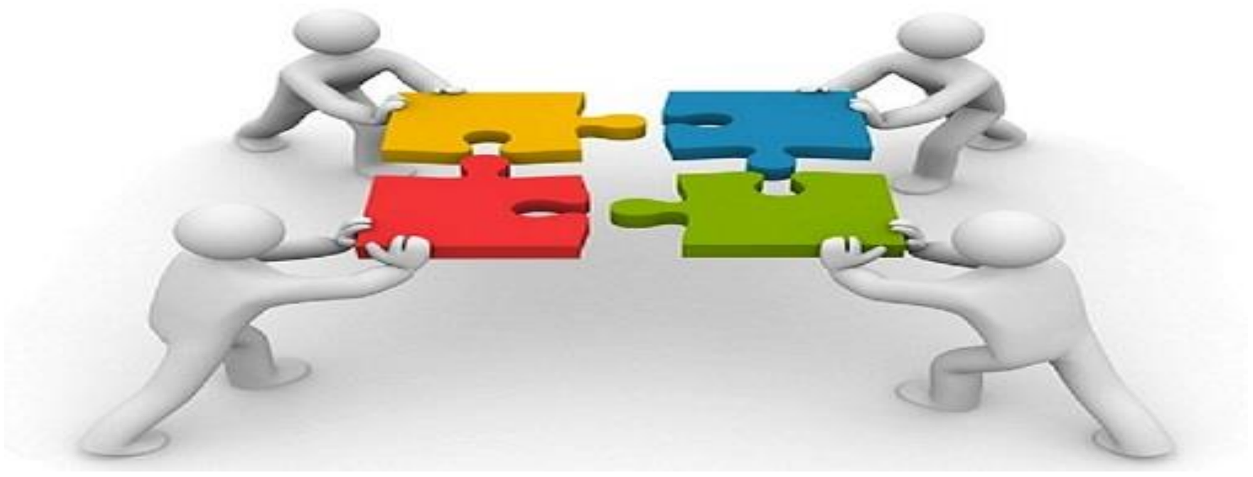


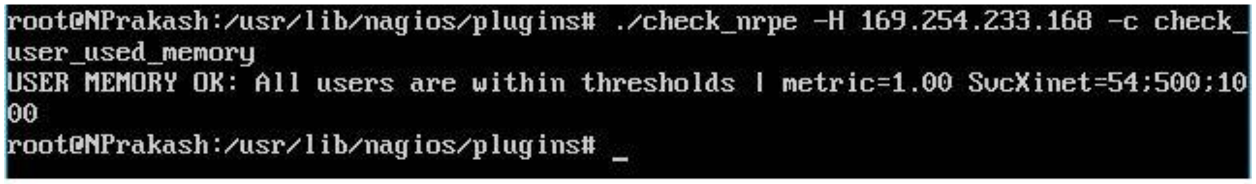
```
root@NPrakash:/usr/lib/nagios/plugins# ./check_nrpe -H 169.254.233.168 -c check_
user_cpu_usage
USER CPU LOAD OK: All users are within thresholds | metric=1.00 SvcXinet=0.58;40
.00;50.00
root@NPrakash:/usr/lib/nagios/plugins# _
```

Figure 2: Output of Generated CPULoadComparison Code

1.2.3 Memory Usage

- When the end user selects the input as Memory_Usage, the tool would generate a program which is used to measure the current Memory usage of the user's computer system.
- The generated Memory_Usage program when run by the Nagios tool along with the NRPE produces the result shown in Figure 3.



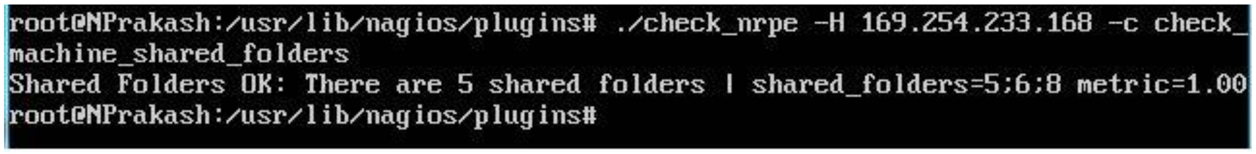


```
root@NPrakash:/usr/lib/nagios/plugins# ./check_nrpe -H 169.254.233.168 -c check_
user_used_memory
USER MEMORY OK: All users are within thresholds | metric=1.00 SvcXinet=54;500;10
00
root@NPrakash:/usr/lib/nagios/plugins# _
```

Figure 3: Output of Generated Memory_Usage Code

1.2.4 Shared Folders

- When the end user selects the input as Shared_Folders, the tool would generate a program which would report the number of shared folders of the computer system
- The generated Shared Folders code when run by the Nagios tool along with the NRPE produces the result shown in Figure 4.

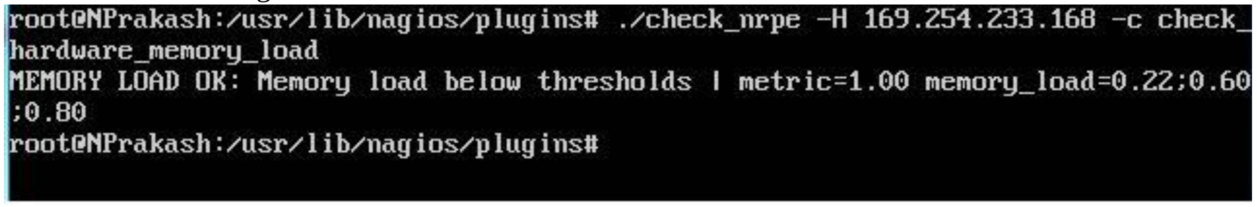


```
root@NPrakash:/usr/lib/nagios/plugins# ./check_nrpe -H 169.254.233.168 -c check_
machine_shared_folders
Shared Folders OK: There are 5 shared folders | shared_folders=5;6;8 metric=1.00
root@NPrakash:/usr/lib/nagios/plugins#
```

Figure 4: Output of Generated Shared Folders Code

1.2.5. Memory Load

- When the end user selects the input as MemoryLoad, the tool would generate a program which would report the current load of the computer system.
- The generated MemoryLoad code when run by the Nagios tool along with the NRPE produces the result shown in Figure 5.

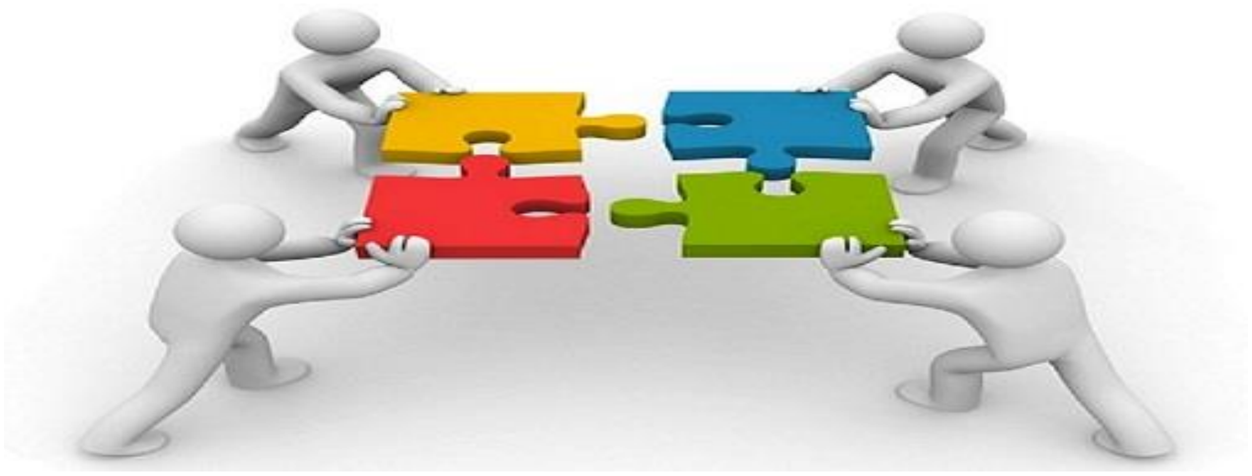


```
root@NPrakash:/usr/lib/nagios/plugins# ./check_nrpe -H 169.254.233.168 -c check_
hardware_memory_load
MEMORY LOAD OK: Memory load below thresholds | metric=1.00 memory_load=0.22;0.60
;0.80
root@NPrakash:/usr/lib/nagios/plugins#
```

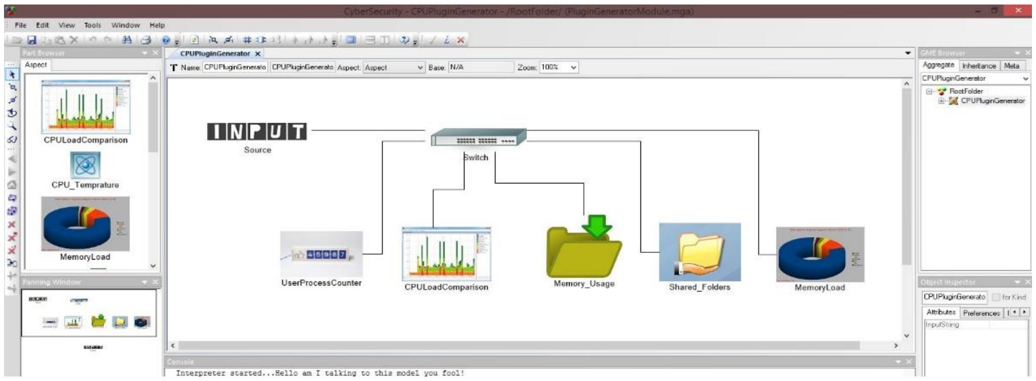
Figure 5: Output of Generated MemoryLoad Code

1.3 Configuring and Running the CPU_Plugin Generator Tool

To run the CPU_Plugin Generator tool, it is important to have to certain softwares installed on your computer.



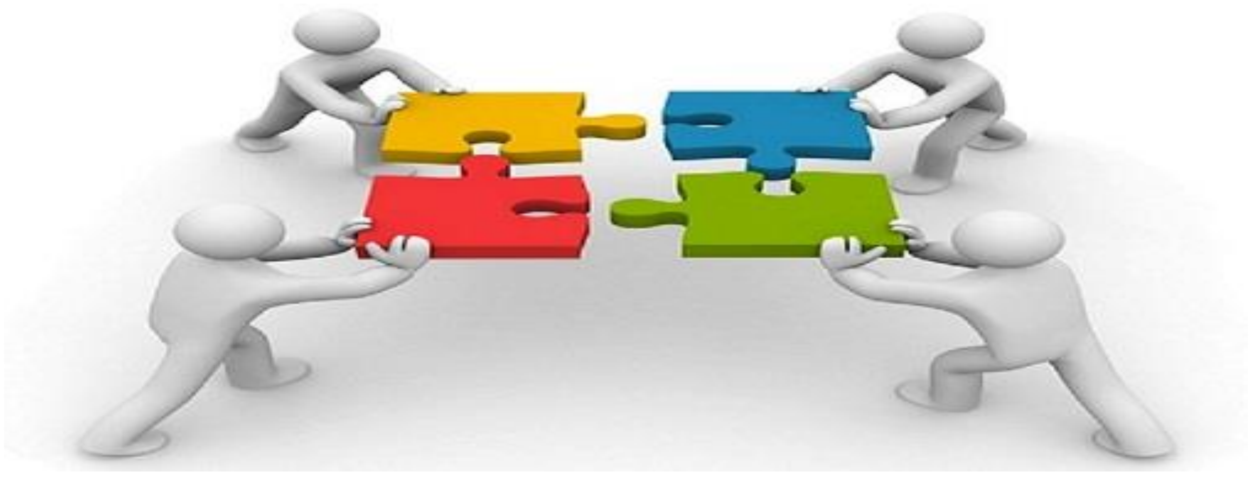
- First, it is important to install Generic Modelling Environment (GME) on your computer. The version for the tool required are **GME-14.10.29**. The Generic Modeling Environment is a configurable toolkit for creating domain-specific modeling and program synthesis environments. The configuration is accomplished through Meta models specifying the modeling paradigm (modeling language) of the application domain.
- Second, you need to have **Microsoft Visual Studio 2012** installed on your computer. This software would run the interpreter code of the CPU_Plugin Generator tool. However you do not need to medal with this, unless you would want to customize the CPU_Plugin Generator tool yourself (explained in the next section).
- After installing all the above softwares, download the .zip file named "CPUPluginGenerator".
- This file contains the PluginGeneratorModule GME project file. Open in this file in your newly installed GME tool.
- On the GME Browser window (On the Right side) expand the Root Folder. CPUPluginGenerator should appear.
- Now, double click on it, CPUPluginGenerator diagram should appear on the GME tool. It should look like Figure 6.



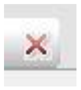
```
Interpreter started...Hello am I talking to this model you fool!
```

Figure 6: CPU_Plugin Generator

Before running the interpreter you need to click on Source. In the object Interpreter panel, you need to select from the drop down menu, the program you want to generate. After selecting it, you are now ready to run the interpreter.



To run the interpreter, you would need to click on the cross mark (As shown in Figure). You would see



"Interpreter started" and "Interpreter Completed" message on the Console window of the GME tool. These messages are important as this would tell whether the interpreter has run correctly or not.

After the interpreter has run successfully, a .txt file named "test" is generated within the same folder that contains the PluginGeneratorModule and the required program is generated in the test file.

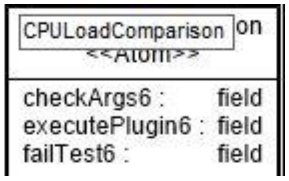
1.4 Customizing the CPU_Plugin Generator Tool

In this section, customizing the tool at the basic level will be covered.

First, in the .zip file provided, find the MetaModel.mga file. Open this file from the GME tool.

On the GME Browser panel, you would find CyberSecurity on the top right window. After expanding this, double click on Paradigm Sheet.

The Meta model of the interpreter design would appear as shown in the Figure 7. You can make several changes to the Meta model.



If you want to change the name of any of the atoms or model, just double on the name of the atom you want to change. A text window with the cursor appears. Now you can change the name of the Atom. However, it is also important to make necessary changes in the interpreter code.

- To make changes to the interpreter code, the user needs to understand the interpreter design which is mentioned in detail in the CPU_Plugin Generator Paper (also contained in the .zip file) and go through the interpreter code (BON2Component1.sln). This is beyond the scope of this user manual and hence left to the interest of the user.
- If you do not need a specific functionality of the tool, the user can select the atom and its required connection and delete it from the Meta model. The user should also comment/ delete that part of the interpreter code. Apart from this, the user needs to click on the Source atom, under the Object inspector window panel, the user needs to remove the name of the Atom deleted from the general

preferences. This step is required as the name of the deleted Atom should not appear on the drop down menu.

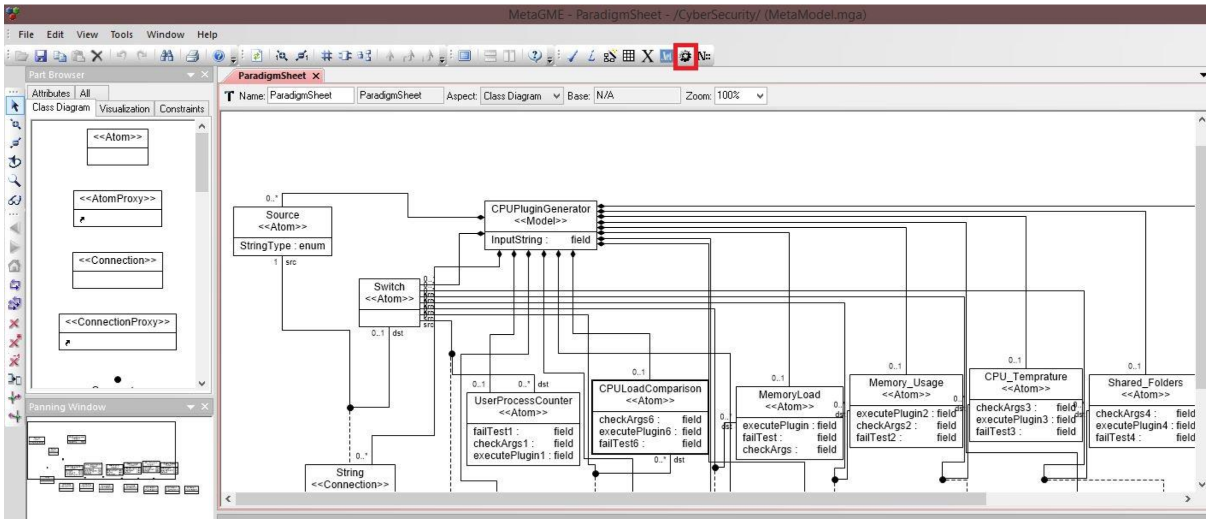


Figure 7: Displaying the Button to Register the MetaModel

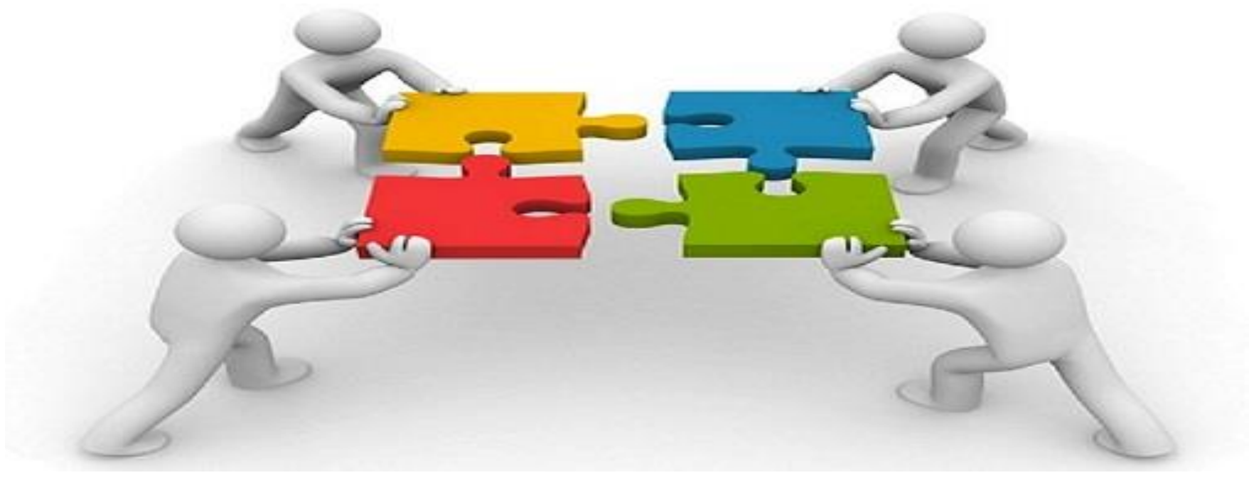
- After all the changes, you need to click on save button. Now the user needs to register the Meta model. This can be done by clicking the MetaGMEInterpreter button (Shown in the red box) as shown in Figure7. The user should also make sure that no error is displayed on the Console of the GME tool.
- Now, a new window appears, which would ask the user on where to save the .xmp file. After clicking save, a new window appears which asks the user to override the existing .xmp file. After clicking "Yes", the new Meta model of the interpreter is registered. Close the project on the GME tool.
- Now click on File->New Project. Under the Paradigm window select the name of the .xmp file you saved. If you have not changed it, it would say "CyberSecurity" and click "Create new".
- A "rootfolder" would appear on the GME Browser window. Right click on it and select insert model. Now select the name of the Meta model you saved. If you have not changed it, it would be CPUPluginGenerator.
- Now you would find the components of the Meta model in the Part Browser of the GME tool. Drag and drop the components and connect them in the appropriate way. This completely depends on how the user has made changes to the Meta model and the interpreter code.

- After all the above steps, run the interpreter by clicking the cross mark. If there are no errors in the interpreter code and in the Meta model, the "Interpreter Completed" message appears on the Console of the GME tool.
- After running the interpreter, the required code is generated onto the test file which is present in the same folder as the CPUPluginGenerator file. HURRAY! You are finally successful in customizing the tool.

I hope you find CPU_Plugin Generator User Manual useful and efficient.

If you have any questions or comments, please contact me at

nishanthprakash@email.arizona.edu



1.5 Index