Changes made:

1. The interpreter generates more programs with the current design
2. Each Atom (except the source and the switch) is associated to a class in the interpreter code.

The Meta model and the activity diagram submitted before, was designed to generate 2 type of programs. (1) Used to measure the CPU\_Usage and (2) count the number of open processes of the computer system. I had to change this after our discussion about the project.

In the previous Meta Model, I had a model generatePlugin which contained another model CPUPluginGenerator. CPUPluginGenerator consisted of source and switch atom and generatePlugin consisted of checkArgs, UserProcessCounter, CPULoadComparison and generateSignal Atom. I used to take 2 inputs from the user to define the critical and the warning value. However, in the current design of the Meta model, only one input is given by the user of type String, as I thought it would not be appropriate for the users to decide the warning and critical values for each of the program that the interpreter generates.

In the current design, I have just 1 model and every atom is within that model and I thought it was more appropriate to have checkArgs, generatePlugin, executePlugin as attributes of each Atom. In the current submission, I associated each atom to a class and hence I have added checkArgs, failtest instead of generateSignal (same functionality), executePlugin as methods to each of the atom in the Meta model. This allowed me to add more functionality to my interpreter. I have added **memory usage**, which would check current usage of memory which is dependent on the number of open processes in the memory, **memory load**, which would check the load of the memory, dependent on the current usage of the memory by the user and **shared folders**, which would count the number of shared folders in the user’s computer. Thus I have modified my previous interpreter Meta model to accommodate more atoms, thus generating 5 different programs each having a specific functionality of its own when executed.

Accordingly, the activity diagram has also changed. The new version of the Meta model and the activity diagram is shown in the paper. In the activity diagram, I had a state to check the arguments the user passes i.e to check whether the warning value is less than the critical value, however, in the new version, since I am not taking the input (warning and critical values) from the user anymore (except the input string which tells the interpreter which program to run), those states have been removed and **memory usage**, **memory load** and **shared folders** are the new states added to the activity diagram. Apart from this, I have also added two more check conditions into my code which is reflected on the activity diagram, first, the interpreter would check the existence of source and second, the interpreter would check the existence of switch. If either of them do not exist, an error is thrown on the GME console by the interpreter.