**Adaptive Monitoring for Continuous Performance Model Integration**

# Thesis statement

# In this section, we will introduce our contribution to the subject of adaptive monitoring for performance model integration as well as the involved scientific challenges that must be solved in order to achieve the goals of our contribution.

# Contribution

# The contribution of this thesis is an extension of the contribution presented in the approach CIPM. Precisely, we focus in our contribution on the first and the second activity of the CIPM approach.

# In order to contribute to the subject of adaptive monitoring for performance model integration, we’ve created an approach that takes into account two properties. The first property is the adaptive monitoring which means that the probes which are responsible for logging the monitoring data are capable to be activated and deactivated during the monitoring phase. The meaning is that once we have collected enough monitoring data of a probe, the probe must be deactivated because more monitoring data is not necessary and can influence the performance of the monitoring. Furthermore, we adapted our approach to the fine-grained monitoring. This is required, when the used performance model needs performance model parameters (like Resource Demands RDs, loop execution number, the probability of selecting a branch). Therefore, we adapted our approach to monitor the response time of computation, the number of execution of loops, the executions of branches and the parameters of a method calls.

# The second property is the monitoring for continuous integration of performance model. The continuous integration of performance model in this context means the incremental update of the performance model in each iteration. The incremental update of the performance model includes the possibility to keep first of all the status of the performance model in the previous iteration and then update it with the information provided within the current iteration. The opposite of that will be to extract the whole performance model in each iteration which will lead to two issues. The first issue is that the extracted performance model in an iteration will not be saved in the next iteration which will cause the second issue. Since the iterations do not consider the modification done in the previous ones, the whole system will have to monitored in every iteration in order to provide the required monitoring data for the new extracted performance model. Hence, in the case of huge Enterprise Applications EAs monitoring the whole system in every iteration will cause a monitoring overhead which is unfeasible and that is the second issue.

# To guarantee the monitoring for performance model integration, we’ve made the following decisions:

# We used the palladio performance model as a performance model

# We used Vitruvius to support the model-driven development in our approach

# We used the coevolution approach for the continuous creation of performance model.

# We’ve created an instrumentation model to persist the generated probes

# Scientific challenges

# in this thesis, we will treat the following challenges:

# *what are the steps necessary to generate the probes and instrument automatically the source code for monitoring?*

# In order to monitor the source code of a system, the system must be firstly instrumented using the needed probes. The instrumentation in our approach is done automatically based of the generated probes during the system development. Moreover, the processes of probes generation and the source code instrumentation can be executed independently. Therefore, we need to define the necessary steps to execute these processes and how they change information between them.

# *what are the probes that must be generated for the monitoring?*

# The definition of probes for monitoring is based on the monitoring data needed by the performance model. Since we used Palladio Performance Model, we have to define probes that provide monitoring data that can be used by the Palladio Performance Model.

# *What are the models required to support the iterative and adaptive monitoring?*

# In order to enable the iterative and the adaptive monitoring for continuous performance model integration. As mentioned before, we extend the coevolution approach which uses models like PCM, Java and Correspondence Model. Therefore, we need to use the information provided by the existing models and define new models for persisting the information that are not provided by the existing model.

# *What are the alternatives of the source code instrumentation for monitoring?*

# The best way to instrument the source code of a system is to use Aspect oriented Programming (AOP) because it separates the instrumentation logic from the business logic which simplify the application maintenance. However, AOP is based on annotations which can not be used for all kind of probes. For example, we can not use AOP annotation to monitor the number of executions of a loop. Therefore, we need to find alternatives that enable the source code instrumentation without changing the original source code of the system. Otherwise, mixing the original source code with instrumentation code will make the readability of the source code infeasible for the user.

# *What level of incremented update of performance model can we reach?*

# Here, we define two levels of incrementation. In the first level, the performance model will be updated based on the changed services. In the second level, it will be updated based on the changed parts of the service. The difference between the first and the second level is that in the first level if a service has changed the SEFF of the service will newly recreated. Therefore, the old elements of this SEFF will be lost with their parameters. In the second level, the unchanged old elements in SEFF will be kept, the new elements will be added and the deleted elements will be deleted. The Coevolution approach has reached the first level. Thus, we will extend to reach the second level of incrementation.