Executive Summary

What is performance engineering and why is it a critical part of the early design process?

Written by Sadir Abdul Hadi, Student Number 15068907

Performance is a key quality of software, as a poor one often causes delays, failures, and sometimes the abandonment of whole projects. It's also quite a complex quality: it's affected by the multiple layers upon which a software is built, and relies on every aspect of software engineering from design to coding. Hence, it seems important to study software performance and improve it, and that's what Software Performance Engineering (SPE) is all about. It's the set of software engineering-related activities used throughout the software development cycle to meet the performance requirements. We are going to discover the different aspects of SPE, in order to understand what makes it particularly important in the early phases of the development cycle, and see how early SPE means better software.

Software Performance Engineering is generally conducted under two approaches. The most widely used one is based on measurements, and occurs late in the development design. It relies on running performance tests, and monitoring the software under these tests through instrumentation, which means inserting probes into a system to display the utilization rate of several devices (e.g. Central Processing Unit, the device that runs computer programs), or the number of cycles of code executed. Another performance engineering technique is a model-based one. It consists of creating performance models describing how the system uses resources (such as space and time), as well as the capacity of these resources, with the design and the software architecture to be adjusted accordingly.

Although different in practice, these two techniques share many elements, and the same objective. In fact, several activities form what is called performance engineering, and these can include the identification of factors that may affect the performance, requirements definition and analysis, performance prediction and testing, and, finally, maintenance and evolution. All of these then allow a total

system analysis, which puts the planned software in the context of the final deployed system.

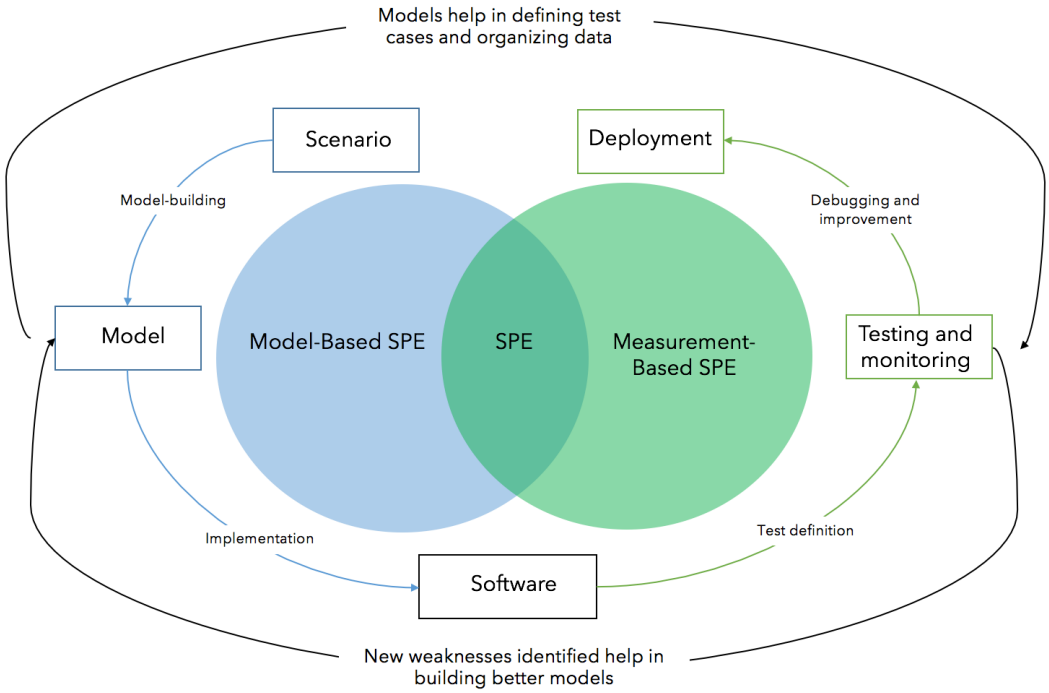
Despite this well-defined set of activities, SPE currently suffers from many weaknesses. Developers often rely on users to do performance testing in the field, and a semantic gap between performance concerns and functional requirements prevent developers from even thinking about performance. Hence, the importance of performance engineering in the early development stages should be stressed on.

Model-based performance engineering can serve as an articulation point between requirements analysis and implementation in the early stages of the development cycle. In fact, performance models are generally created from scenarios which describe the intended use of the system. This makes it is very useful tool compared to having to modify the system to suit the requirements after it has been built, since it costs less time and money. Hence, one of the core advantages of early performance modelling is that it provides an "early warning" of performance issues, and that it draws a clear list of performance concerns that should be taken into consideration.

Reasoning about software performance in the early design process has many other advantages. Firstly, it generally implies the use of performance models mentioned earlier, which are becoming more and more standardized thanks to different tools that exist, such as the Unified Modelling Language (UML), or are being developed. These performance models might include reusable blocks, which can be integrated in a "performance knowledge base" that supports the relationship between the models, and enables developers and designers to search for relevant information. Secondly, having performance in mind early in the development process makes it easier to evaluate the impact of a potential change, and to choose one of many ways in which a new functionality could be implemented.

Early stage performance engineering also complements the late stage one, and if we establish an analogy with other engineering disciplines, it would seem that the

way forward is the convergence of the measurement and modelling approaches. In fact, models allow the integration and the extrapolation of the data collected in the measurement phase. This means that performance engineers can predict future performance behaviors based on past ones, giving even more value to the data collected in the later stages. Furthermore, performance models provide a structure which allows to organize the data and reason about it.



Early-stage SPE helps in having a better late-stage SPE, and vice versa

In conclusion, SPE is very important in the early stages of the development cycle. Not only does it limit the costs and decrease the chances of a project being delayed because of performance issues, but it also makes the development phase more flexible by making new features easier to add. Furthermore, it creates a "Knowledge Base" for performance engineers, which allows them to share their work with others who might need it. However, the novelty that this research paper highlights is that the way forward is the convergence of the model-based and measurement-based approaches, and that neither of these is as important alone as when combined with the other.

References:

Woodside,M. Franks,G. Petriu,D.C. 2007. The Future of Software Performance Engineering. pp.171-187.