SOFTWARE PERFORMANCE ENGINERRING

Gouri Sharma*

Meetu Arora*

ABSTRACT

Software performance engineering (SPE) is a systematic, quantitative approach to the cost-effective development of software systems to meet performance requirements .SPE, a software oriented approach, focuses on architecture, design and implementation choices. It helps as to build software that meets performance requirement on time and within budget. There are two important dimensions to software performance timeliness i.e. responsiveness and scalability. SPE is the process of predicting and evaluating whether the software system satisfies performance goals defined by the user .Predicting the performances of a software system requires the availability of a suitable abstract model of the final software. Such model has to provide a suitable description of the run-time behavior of the software system, so that performance can be estimated. One of the key activities in performance testing is to decide what type of load (operational load and background load) or loads to place on the system application. A successful software performance test plan starts with knowing what you want to accomplish Software performance key testing types includes load, reliability, scalability, failover, recovery and stress testing .

*Lecturer, SVIET , Ramnagar, Banur.

INTRODUCTION

Software performance engineering gives a systematic, quantitative approach for constructing cost effective software systems that meet performance objectives. With SPE, we detect problems early in development, and use quantitative methods to support cost-benefit analysis of hardware solutions versus software requirements or design solutions, or a combination of software and hardware solutions.SPE, a software-oriented approach, focuses on architecture, design, and implementation choices.SPE gives you the information you need to build software that meets performance requirements on time and within budget.

Poor performance is one of the main quality-related shortcomings that cause software projects to fail. Thus, the need to address performance concerns early during the software development process is fully acknowledged, and there is a growing interest in the research and software industry communities towards techniques, methods and tools that permit to manage system performance concerns as an integral part of software engineering. Model-based software performance analysis introduces performance concerns in the scope of software modeling, thus allowing the developer to carry on performance analysis throughout the software lifecycle

The SPE process begins early in the software development life cycle and uses quantitative methods to identify satisfactory designs and to eliminate those that are likely to have unacceptable performance before developers invest significant time in implementation. SPE continues through the detailed design, coding and performance and load testing phases to predict and manage the performance of the evolving software as well as monitor and report actual performance versus specifications and predictions. SPE methods encompass: performance data collection; quantitative performance analysis techniques; prediction strategies; management of uncertainties; data presentation and tracking; performance testing, stress and load testing, model verification and validation; critical success factors; and performance design principles, patterns and anti-patterns.

PERFORMANCE OF A 'SYSTEM'

Performance is the degree to which a software system or component meets its objectives for timeliness.

There are two important dimensions to software performance timeliness: *responsiveness* and *scalability*.

Responsiveness is the ability of a system to meet its objectives for response time or throughput. In end-user systems, responsiveness is typically defined from a user perspective.

For example, responsiveness might refer to the amount of time it takes to complete a user task, or the number of transactions that can be processed in a given amount of time.

Scalability is the ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases. Scalability is an increasingly important aspect of today's software systems.

In order to build scalability into your system, you must know where the "knee" of the scalability curve falls for your hardware/software environment. If the "knee" occurs before your target load requirements, you must either reduce the utilization of the bottleneck resource by streamlining the processing, or add additional hardware (e.g., a faster CPU or an extra disk) to remove the bottleneck.

Performance is an indicator of how well a software system or component meets its requirements for timeliness. Timeliness is measured in terms of response time or throughput.

-

- ***Response Time:*** Response time specifies the time taken by the system to respond back with the expected output. For an enterprise application Response time is defined as the time taken to complete a single business transaction. It is usually expressed in seconds.
- ***Throughput:*** Throughput refers to the rate which the system churns expected outputs when the designated input is fed in the system. In other words, for an Enterprise Application, throughput can be defined as the total number of business transactions completed by the application in unit time (per second or per hour).

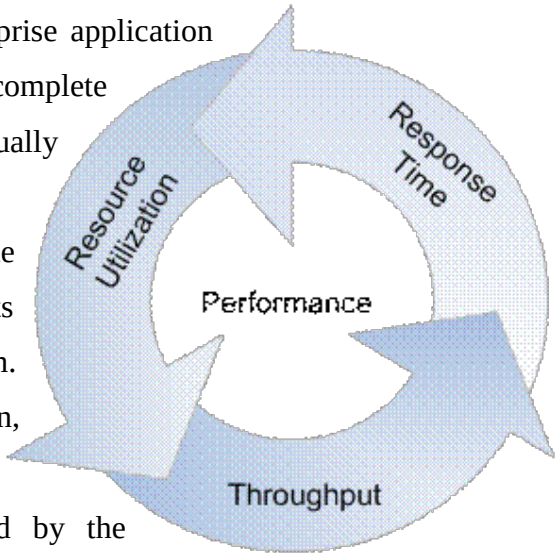

FIGURE- 1

- ***Resource Utilization:*** For any system to consume an input and produce the designated output, certain resources would be required. The amount of resources consumed by the system during processing that request, defines the resource utilization. There can be different resources factored in, such as processor, disk (i/o controller), memory etc. Utilization of 80% is considered an acceptable limit. Normally utilization of 70% warrants ordering of additional hardware.

Basic Issues in SPE

In most software development projects, only the functional requirements are taken into account during the development phase. Performance is not taken into account until the system is ready to go into production. Many organizations have used a "fix-it-later" approach to performance. This approach advocated concentrating on correctness and deferring consideration of performance until the testing phase. Performance problems detected then, were corrected by adding additional hardware, tuning the software (usually in a crisis-mode), or both.

Because it is based on several performance myths, this approach can be dangerous. These myths include:

- **Performance problems are rare:** The reality is that the number, size and complexity of systems has increased dramatically and today's developers are less expert at dealing with performance than their predecessors. As a result, performance problems are all too common.
- **Hardware is fast and inexpensive:** The reality is that processor speeds have increased dramatically, but networks are far slower. Furthermore, software threading issues cause performance problems despite the availability of hardware resources. No one has an unlimited hardware budget, and some software may require more resources than the hardware technology can provide.
- **Responsive software costs too much to build:** This is no longer true thanks to SPE methods and tools. In fact, the "fix-it-later" approach is likely to have higher costs.
- **You can tune it later:** This myth is based on the erroneous assumption that performance problems are due to inefficient coding rather than fundamental architectural or design problems. Re-doing a design late in the process is very expensive.

  Unfortunately, fixing the problem at such a late stage in the development may be too costly since major software rewrites and architectural changes may be required.

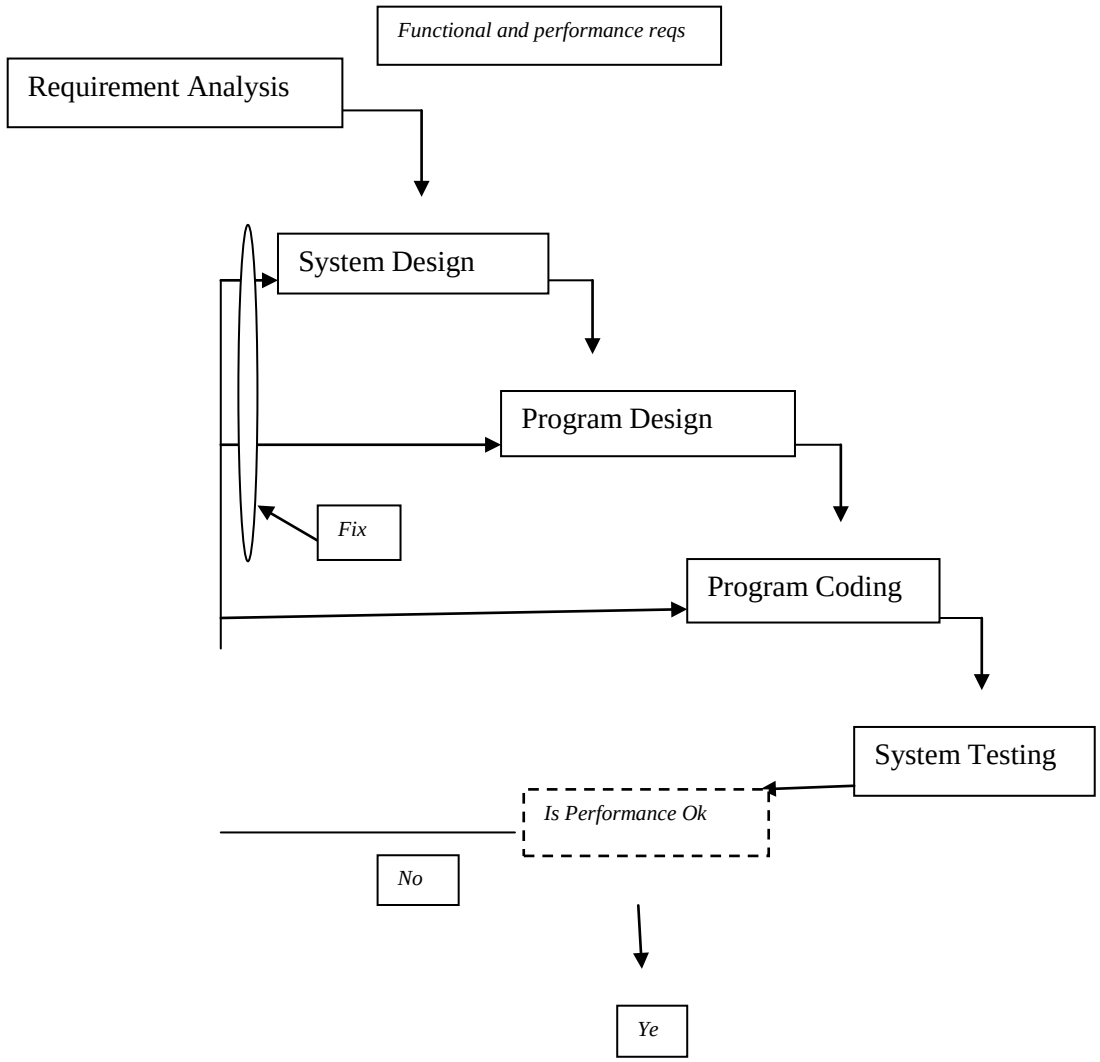

Figure 2 - Common Software Development Approach.

ENGINEERING ALTERNATIVE

The preferred approach is to integrate performance prediction and assessment into all phases of the software development life cycle.SPE provides an engineering approach to performance, avoiding the extremes of performance-driven development and "fix-it-later." SPE uses model predictions to evaluate trade-offs in software functions versus hardware costs. The models assist developers in controlling resource requirements by selecting architecture and design alternatives with acceptable performance characteristics. They aid in tracking performance throughout the development process and prevent problems from surfacing late in the life cycle (typically during performance and stress testing).

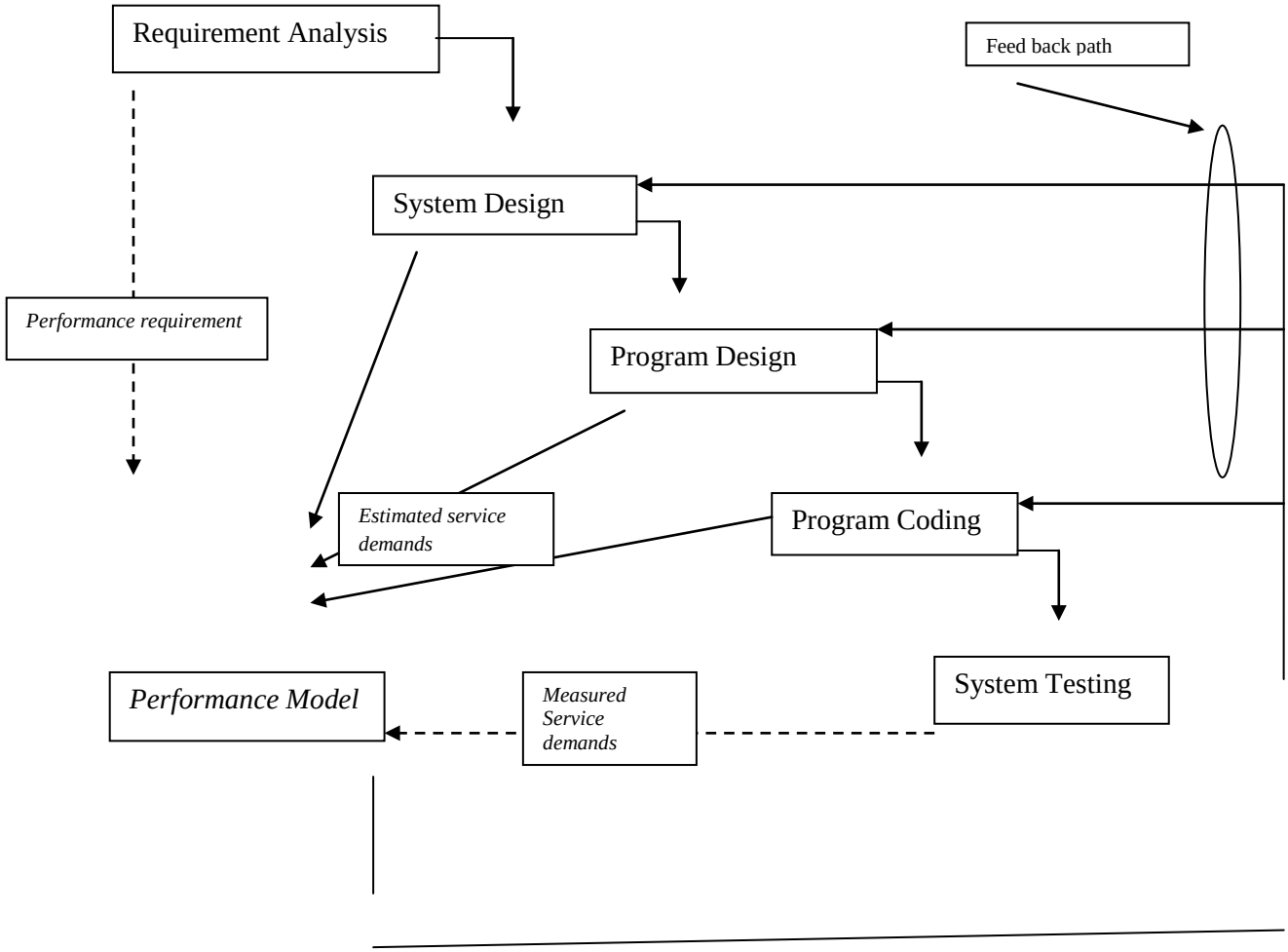

Figure 3 - Integration of software development with performance modeling.

The Requirements Analysis phase generates performance requirements. The system design, program design, and program coding phases generate estimates of the service demands that can be used as input parameters in SPE studies that in turn generate feedback to the designers as design and development proceeds. At the testing stage, actual service demands can be used for final performance assessment and performance tuning. The goal is that by integrating performance development life cycle, one can guarantee that the resulting system will exhibit the desired performance. Each phase of the software development life cycle generates different kinds of inputs to SPE studies.

IS SPE VIABLE FOR YOUR PROJECTS?

SPE is tried and proven technology – performance experts have always used these techniques for building high-performance software. So the following aspects of a new performance paradigm are now achievable:

- You can deliver that application on time and within budget while meeting your performance requirements - with the first release.
- You can know the hardware capacity requirements for your new application before it is developed.
- Architects know that their architecture will support performance requirements before committing to code.
- Project managers can track the status of performance while the software is being developed.
- Performance specialists have time to run performance tests, load and stress tests without encountering "surprises."
- Risks to achieving performance requirements are identified and addressed early in the process, saving both time and money.
- Software that meets performance requirements is delivered on time and within budget.

SPE is language and platform independent. Performance models are constructed from architectural and design-level information. Thus, SPE works with C++, C# and Java as well as with other object-oriented and non-object-oriented languages. The execution behavior of the software will be different with different languages and platforms. Nevertheless, this is reflected in the resource requirement specifications, not the model structure.

SPE uses deliberately simple models of software processing with the goal of using the simplest possible model that identifies problems with the system architecture, design, or implementation plans. It is relatively easy to construct and solve these models to determine whether the proposed software is likely to meet performance requirements. As the software development process proceeds, we refine the models to more closely represent the performance of the emerging software and re-evaluate performance.

SPE can be easily integrated into the software development process. It has been used with traditional process models, such as the waterfall model. It works especially well with iterative, incremental processes such as the Unified Process. With an iterative, incremental process, you can use SPE techniques to assess and reduce the risk of performance failure at the project outset, and at each subsequent iteration.

Costs and Benefits The cost of using SPE is usually a small percentage of the overall project budget. The level of SPE effort is determined by the amount of performance risk on the project.

For a project with little risk, SPE costs are typically 1% of the total budget. For a high-risk

project, the SPE effort might be as high as 10% of the project budget.Industrial experience indicates that SPE can save far more than it costs by avoiding the costs (both direct and indirect) of performance failure mentioned above.

CONCLUSION

So it is concluded from above discussion that SPE is a cost effective process that detects the problem early in development phase. Major benefits of SPE are improved accuracy, timeliness and confidence in decision making. In most software development projects, only the functional requirements are taken into account during the development phase. Performance is not taken into account until the system is ready to go into production so by using SPE approach performance factor is taken at very early stages. It also helps in improving productivity with reduced cost. So at last we can say that Software performance engineering (SPE) is a method for constructing software systems to satisfy performance requirements. Experience shows that performance of new systems can be orders of magnitude better, with no disruption to delivery schedules, when SPE techniques are systematically applied throughout development.

REFERENCES

1. [Smith and Williams 2003a] C. U. Smith and L. G. Williams,"Ten Best Practices for Software Performance Engineering," *MeasureIT*, Computer Measurement Group online Newsletter, June 2003.
2. [Boehm 1991] B. Boehm, "Software Risk Management:Principles and Practice," I*EEE Software*, vol. 8, no. 1, pp. 32-41, 1991.
3. [Clements and Northrop 1996] P. C. Clements and L.M. Northrop, "Software Architecture: An Executive
4. [Javelin 2002] Javelin Technologies, "Best Practice Definition," Oakville, Ontario, Canada, 2002 (www.javelin-tech.com)
5. [Reifer 2002] D. J. Reifer, Making the Software Business Case: Improvement by the Numbers, Boston,Addison-Wesley, 2002.
6. [Smith 1990] C. U. Smith, Performance Engineering of Software Systems, Reading, MA, Addison-Wesley,1990.
7. [Smith and Williams 2002] C. U. Smith and L. G. Williams,Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software, Boston,MA, Addison-Wesley, 2002.