

Preface

The performance analysis of concrete technologies has already been discussed in a multitude of publications and conferences, but the practical application has often been neglected. An engineering procedure was comprehensively discussed for the first time during the “International Workshop on Software and Performance: WOSP 1998” in Santa Fe, NM, in 1998. Teams were formed to examine the integration of performance analysis, in particular into software engineering. Practical experiences from industry and new research approaches were discussed in these teams. Diverse national and international activities, e.g., the foundation of a working group within the German Association of Computer Science, followed.

This book continues the discussion of performance engineering methodologies. On the one hand, it is based on selected and revised contributions to conferences that took place in 2000:

- Second International Workshop on Software and Performance – WOSP 2000, 17 – 20 September 2000 in Ottawa, Canada,
- First German Workshop on Performance Engineering within Software Development, May 17th in Darmstadt, Germany.

On the other hand, further innovative ideas were considered by a separate call for chapters. With this book we would like to illustrate the state of the art, current discussions, and development trends in the area of performance engineering.

In the first section of the book, the relation between software engineering and performance engineering is discussed. In the second section, the use of models, measures, and tools is described. Furthermore, case studies with regard to concrete technologies are discussed in the third section.

The contributions published in this book underline the international importance of this field of research. Twenty contributions were considered from Venezuela, Spain, Cyprus, Germany, Canada, USA, Finland, Sweden, and Austria.

We would like to thank all the authors as well as Springer-Verlag for the good cooperation during the preparation of the manuscript. Furthermore, our thanks are due to Mrs. Dörge for her comprehensive editorial processing.

March 2001

Reiner Dumke
Claus Rautenstrauch
Andreas Schmietendorf
André Scholz

Historical Roots of Performance Engineering

Initially, computer systems performance analyses were primarily carried out because of limited resources. *D. E. Knuth* discussed extensive efficiency analyses of sort and search algorithms in 1973 [5].

Although the performance of modern hardware systems doubles every two years, performance analyses still play a major role in software development. This is because functional complexity and user requirements as well as the complexity of the hardware used and software technologies are increasing all the time.

Performance engineering has many historical roots. The following historical survey sketches selected periods, all of which have contributed to the basis of performance engineering.

The Danish engineer *A. E. Erlang* had developed the mathematical basis for the utilization analysis of telephone networks by 1900. He described the relationships between the functionality of communication systems and their arrival as well as service times with the help of mathematical models. The queuing theory, with which runs in hardware systems can be modeled, is based on these mathematical basics.

A. A. Markov developed a theory for modeling stochastic characteristics of arrival and service processes in 1907. Markov chains, as a special case of Markov processes, have offered the basis for the coverage of IT systems with the help of discrete models for performance analysis since 1950.

Furthermore, *C. A. Petri* developed an important metamodel for the specification of dynamic system characteristics – the Petri net. The basic model has been extended over the years. Time-based Petri nets in particular are a basis for performance-related analyses and simulations of IT systems.

In the 1970s a series of algorithms were developed for the evaluation of these models. Most of them were very time-consuming [2, 8]. Henceforth, research has concentrated on approximate and efficient solution procedures [1].

The operational analysis, which was developed by *J. P. Buzen* and *P. J. Denning*, provided a practicable-analytical-oriented model evaluation [3, 6]. It could be used comprehensively and was integrated into many modeling tools.

Simulation is a further measure for the evaluation of IT systems. The use of conventional programming languages was replaced by the application of specific simulation language and/or environments. The simulation system GPSS was developed back in 1961 by *G. Gordon* in order to facilitate the capacity planning of new systems.

In addition, modeling measurements are also used for the determination of performance characteristics. *J. C. Gibson* proposed one of the first measurement-oriented approaches with the "Gibson instruction mix" at the end of the 1950s[7]. Measurements and benchmarks in particular, which have been used since the mid 1970s, are based on the transfer of a synthetic workload to a hardware system. The whetstone benchmark for the measurement of the floating point performance, which was developed by *H. J. Curnow* und *B. A. Wichmann* [4], as well as the Dhrystone-Benchmark for the determination of the integer performance, which was developed by *R. Weicker* in 1984 [10], are classical benchmarks. Only single system components, e.g., processors, were considered within the first phases of using benchmarks.

Current benchmarks of vendor-independent organizations, such as the SPEC (Standard Performance Evaluation Cooperation) or TPC (Transaction Processing Council), consider complex software and hardware architectures as well as user behaviors. They are the basis for the comparison of IT systems and provide performance-related data for the creation of performance models. ISO 14756 defined the first international standardized procedure for the evaluation of IT systems with the help of benchmarks in 1999.

C. *Smith* pointed out the problematic of a performance-fixing approach at the end of the software development (fix-it-later). With software performance engineering she proposes that the performance characteristics of IT systems consisting of hardware and software be analyzed during the whole software life cycle [9].

Reiner Dumke, Claus Rautenstrauch, Andreas Schmietendorf, André Scholz

References

1. Agrawal, S.C.: Metamodelling. A Study of Approximations in Queuing Models. Ph.D. Thesis, Purdue University of W. Lafayette, IN, 1983.
2. Basket, F., Brown, J.C., Raikes, W. M.: The Management of a multi-level non-paged memory System. Proceedings of AFIPS, AFIPS Press, 1970, p. 36.
3. Buzen, J.P.: Computational Algorithms for Closed Queuing Networks with Exponential Servers. *Comms. ACM* 16, 1973, 9, 527-531.
4. Curnow, H.J., Wichmann, B.A.: A Synthetic Benchmark. *The Computer Journal*, Volume 19, Oxford University Press, 1976.
5. Knuth, D.E.: *The Art of Computer Programming*. Vol. 3: Sorting and Searching. Addison-Wesley: Reading/MA, 1973.
6. Denning, P.J., Buzen, J.P.: The Operational Analysis of Queuing Network Models. *ACM Computing Surveys* 10, 1978, 3, 225-261.
7. Gibson, J.C.: The Gibson Mix. IBM System Development Division. Poughkeepsie, NY, Technical Report. No. 00.2043.
8. Kleinrock, L.: *Queuing System Volume II: Computer Applications*. Wiley & Sons: New York/NY, 1976.
9. Smith, C.U.: The Evolution of Software Performance Engineering: A Survey. In Proceedings of the Fall Joint Computer Conference, November 2–6, 1986, Dallas, Texas, USA, IEEE-CS.
10. Weicker, R.P.: Dhrystone. A Synthetic Systems Programming Benchmark. *Communications of the ACM* 27, 1984, 10, 1013-1030.

Aspects of Performance Engineering – An Overview

Andreas Schmietendorf and André Scholz

University of Magdeburg, Faculty of Computer Science
P.O. Box 4120, Magdeburg, 39106, Germany
schmiete@ivs.cs.uni-magdeburg.de;
ascholz@iti.cs.uni-magdeburg.de

1 Motivation

The efficient run of business processes depends on the support of IT systems. Delays in these systems can have fatal effects for the business. The following examples clarify the explosive nature of this problem:

The planned development budget for the luggage processing system of the Denver, Colorado airport increased decisively by about 2 billion US\$ because of inadequate performance characteristics. The system was only planned for the United Airlines terminal. However, it was enlarged for all terminals of the airport within the development without considering the effects on the system's workload. The system had to manage more data and functions than any comparable system at any other airport in the world at that time. Beside a faulty project management, inadequate performance characteristics led to a delay in the opening of the airport of 16 months. A loss of 160 000 US\$ per day was recorded.

An IBM information system was used for the evaluation of individual competition results at the Olympic Games in Atlanta, Georgia. The performance characteristics of the system were tested with approximately 150 users in advance. However, more than 1000 people used the system in the productive phase. The system collapsed under this workload. The matches were delayed and IBM suffered deep-cutting image losses, whose immaterial damages are hard to determine. These examples show that the evaluation of the performance characteristics of IT systems is important, especially in high-heterogeneous system environments.

However, active performance evaluations are often neglected in industry. The quality factor performance is only analyzed at the end of the software development process [4]. Then performance problems lead to costly tuning measures, the procurement of more efficient hardware, or to a redesign of the software application. Although the performance of new hardware systems is increasing all the time, particularly complex application systems, based on new technologies, e.g., multimedia data warehouse systems or distributed systems, need an explicit analysis of their performance characteristics within the development process.

These statements are also confirmed by Glass. In an extensive analysis, he identifies performance problems as the second most frequent reason for failed software projects [2].

2 Requirements and Aims

The performance characteristics of a system have to be considered within the whole software development process. Performance has to have the same priority as other quality factors such as functionality or maintainability.

However, a practicable development method is necessary to assure sufficient performance characteristics. Extensive and cost-intensive tuning measures, that play a major part within most development projects, can consequently be avoided. The operation of high-critical systems, e.g., complex production planning and control solutions, depends on specific performance characteristics, since inefficient system interactions are comparable with a system breakdown, because all following processes are affected. The system user's work efficiency is impaired by inadequate response times, because of frustration. Additionally, software ergonomic analyses have shown that users, who have to wait longer than five seconds for a system response, initiate new thought processes. Controlled cancellations of the new thought processes and the resumption of the old condition take time and lead to a lower productivity of the user [3, pp.326].

The development method has to determine performance characteristics early within the development process to minimize performance-entailed development risks. A structured performance analysis is necessary. This should be supported by a process model, which has to be integrated within the existing company-specific software development process. Also, the application of this method should not be isolated from the development process, since additional activities, that accompany the software development process, are usually neglected when faced with staff and temporal problems.

The fused models should not become an inefficient complex. An economical application must still be guaranteed. Expenditures should be justifiable in relationship to the total project costs [6].

The size of the expenditures is often based on empirically collected data and knowledge. The deployment of qualified employees, who have a high level of knowledge in the areas of software engineering and performance analysis, is imperative. However, methods and technologies must be shaped with the developer in mind.

3 The Performance Engineering Method

The described set of aims can be reached with the performance engineering method. Performance engineering can be defined as a collection of methods for the support of the performance-oriented software development of application systems throughout the entire software development process to assure an appropriate performance-related product quality. Performance engineering becomes an interface between software engineering and performance management. It is not a relaunch of long-standing performance management methods as were other engineering approaches that had already been successfully used in the area of telecommunication.

Performance engineering analyzes the expected performance characteristics of a software system in its early development phases. In the system analysis, software

developers, customers, and users define performance characteristics as service level objectives beside functional specifications.

The performance has to be determined and quantified by performance metrics. These are specific product metrics that can be derived from different system levels and perspectives. An internal and external perspective is often distinguished. Internal performance metrics refer to operation times, e.g., the number of operations per time unit needed to transfer ratios, like the number of transferred bytes per second, or to the utilization of system resources, like CPU or RAM. External performance metrics reflect the outside behavior of the software system with regard to executed functions. The metrics often refer to response times of concrete application functions and to the throughput. A mix of varied quantified metrics describes the performance characteristic of an IT system.

The negotiations of developers, customers, and users should be based on suitable and justifiable cost-performance-ratios [1]. Performance engineering already provides instruments for this phase, like rules of thumb. Unrealistic developments can be discontinued early on or can be renegotiated.

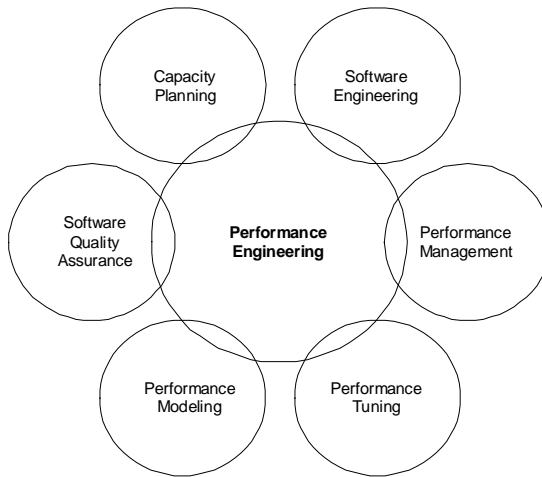


Fig. 1. Use of existing concepts of other disciplines

The quantified performance metrics have to be refined in the system design and implementation phase. Concrete application functions and interfaces can be evaluated at this time. The software development process should allow a cyclic verification of the performance characteristics. These will be analyzed in the respective phase of the software life cycle in the available granularity. In the first phases, estimations have to be used that are often based on rules of thumb. In further development phases, analytical and simulative models can be used. If prototypical implementations of individual program components are already available, measurements can be executed. The quantified metrics should be continuously compared with the required performance characteristics. Deviations lead to an immediate decision process. Performance engineering uses existing methods and concepts from the areas of performance management, performance modeling, software engineering, capacity planning, and performance tuning, as well as software quality assurance. It enlarges

and modifies them by performance-related analysis functions, cf. figure 1. However, the performance metrics are only as exact as the basis data of the models. In order to make a concrete and reality-based calculation, the set up of a performance database is imperative. Since a lot of performance data can be collected in the productive operation of a software system by benchmarking and monitoring, it should be assured that these data are stored in the database for further performance engineering tasks in future development projects. The quality of the performance evaluation depends decisively on the PEMM-level (Performance Engineering Maturity Model) of the development process [5].

However, critical software components are analyzed until the end of the implementation phase. The complete test environment is available within the phase of the system test. If prototypical implementations have not been used within the design and implementation phase, the fulfillment of the performance requirements can be verified with the help of load drivers, e.g., by synthetic workload, in the system test phase.

The real production system is available in the system operation phase. Often performance analyses are performed again in pilot installations over a certain time period, since the analysis is now based on real workload conditions. Thereby, an existing system concept can be modified again. The workload of the system often increases with a higher acceptance. That is why reserves should be considered within the system concept. The proposed procedure has to be adapted to domain specific environments.

Because of the scope of the tasks, specialists should support performance engineering. In further development projects, these tasks are handed over step-by-step to the software developers. Their long-term task spectrum widens. The integration can essentially be simplified if software developers come into contact with these principles during their academic education.

References

1. Foltin, E., Schmietendorf, A.: Estimating the cost of carrying out tasks relating to performance engineering. In: Dumke, R., Abran, A.: *New Approaches in Software Measurement*. Lecture Notes in Computer Science LNCS 2006, Springer-Verlag Berlin Heidelberg, 2001.
2. Glass, R.: *Software Runaways. Lessons learned from Massive Software Project Failures*. Prentice Hall: Upper Saddle River/NJ, 1998.
3. Martin, J.: *Design of Man-Computer-Dialogues*, Engelwood Cliffs, NJ: Prentice Hall, Inc., 1973.
4. Rautenstrauch, C., Scholz, A.: Improving the Performance of a Database-based Information System. A Hierarchical Approach to Database Tuning. In Quincy-Bryant, J. (Ed.): *Milleneal Challenges in Management Education, Cybertechnology and Leadership*, San Diego/CA, 1999, pp. 153–159.
5. Schmietendorf, A., Scholz, A., Rautenstrauch, C.: Evaluating the Performance Engineering Process. In: *Proceedings of the Second International Workshop on Software and Performance*. WOSP2000. Ottawa, ON, 2000, ACM, pp. 89-95.
6. Scholz, A., Schmietendorf, A.: A risk-driven Performance Engineering Process Approach and its Evaluation with a Performance Engineering Maturity Model. In Bradley, J.T.; Davies, N.J.: *Proceedings of the 15th Annual UK Performance Engineering Workshop*. Technical Report CSTR-99-007, Research Press Int., Bristol/UK, 1999.

Performance Engineering

State of the Art and Current Trends

Dumke, R.; Rautenstrauch, C.; Schmietendorf, A.;

Scholz, A. (Eds.)

2001, XIV, 349 p., Softcover

ISBN: 978-3-540-42145-0