A SURVEY ON PERFORMANCE MODELLING AND SIMULATION OF SAP ENTERPRISE RESOURCE PLANNING SYSTEMS

Manuel Mayer[(a)], Stephan Gradl[(b)], Veronika Schreiber[(c)], Harald Kienegger[(d)], Holger Wittges[(e)] , Helmut Krcmar[(f)]

[(a) (b) (c) (d) (e) (f)] Chair for Information Systems, Technische Universitaet Muenchen,
Boltzmannstr. 3, 85748 Garching, Germany

[(a) (b) (c) (d) (e) (f)] {mayerm, gradl, schreibv, kienegger, wittges, krcmar}@in.tum.de

ABSTRACT
IT industry faces the need of a robust, reliable and scalable architecture to support enterprise-scale IT systems. One of such complex systems are enterprise resource planning (ERP) systems. ERP systems support company-wide processes and are considered as a critical success factor for a reliable business operation. Despite the importance of performance prediction of IT systems shown in literature, there is not much work done in the context of ERP systems. This paper presents the results of an online survey conducted in December 2010. 36 IT decision makers of various industries and company sizes took part in this survey, which contained questions about whether and to which degree performance modelling and simulation is currently implemented in companies, which tools are used to measure the performance of ERP systems, and which requirements are expected.

Keywords: performance measurement, performance modelling, simulation, ERP systems

1. INTRODUCTION

With an increasing number of functionalities added to systems by manufacturers and system developers, modern computer systems are becoming more and more complex. One problem that arises is how to choose the right system and components for a certain problem. According to Risse (2006), the selection process is typically driven by different factors, such as functional requirements, performance demands and economic constraints. While architecture design decision-making involves addressing functional requirements, selecting necessary software (components) in the context of standard software, and considering tradeoffs due to the presence of economic constraints, questions arise about how the system performs, how the system scales if expected workload increases, which components might be potential bottlenecks, and what the system evaluation criteria would be. The objective of (technical) performance evaluation is to give answers on these questions and to present techniques to get the performance values (e.g. Jain 1991; Sauer and Chandy 1981).

The literature shows the importance of performance evaluation of IT (e.g. Robertazzi 2000). However, there are only a few publications regarding performance evaluation of ERP systems (Rolia et al. 2009). Therefore, the aim of this work is to analyze the status quo of ERP performance evaluation and prediction in companies' data centres. For that purpose, essential performance evaluation techniques have been examined. Based on this knowledge, an online survey has been developed. The focus concentrates on the following main questions:

- Which demands do companies have regarding performance evaluation of ERP systems?
- Is performance of ERP systems modelled and predicted in companies, and if so, which methods and tools are used?
- Are companies satisfied with the reliability of current performance evaluation techniques used?

The remainder of this work is as follows: after giving a short overview of performance evaluation techniques in section 2, related work concerning ERP systems is presented in section 3. Section 4 explains the structure of the online survey and gives a summary of the results. Section 5 outlines interesting issues for future work.

2. THEORETICAL BACKGROUND

According to Ferrari (1986), the field of performance evaluation dates back to the thesis of Alan Scherr (1965). Since then the discipline of performance evaluation has been addressed by many different text books, for example by Sauer and Chandy (1981) or Raj Jain (1991). Following Hu and Gorton (1997), the factors *functionality*, *reliability*, *speed*, and *economic efficiency* have to be considered in conjunction with performance. In the majority of cases, the first two factors, functionality and reliability, are addressed by system designers. As a result, the main focus of performance evaluation community is speed and economic efficiency. While speed is often described by response time and throughput, economic efficiency

reflects the need to design and implement a system at the lowest cost.

To evaluate the performance of a system, first it is necessary to define some criteria or metrics. The right selection of the metric depends on the scenario, for example, in a user oriented analysis response time is of more interest than throughput. On the other hand, performance evaluation of networks mostly concentrates on the throughput achieved.

Once the metrics, their relationships and effects on performance parameters are known, it is equally important to select a proper workload in a particular environment. According to Jain (1991), workload types can be divided into single instruction, instruction mixes, kernels, synthetic programs and application benchmarks. Single instruction, instruction mixes and kernels are primarily used for hardware-related performance evaluations. Synthetic programs (also called synthetic benchmarks), such as the Zachmanntest for SAP ERP systems (Boegelsack, Wittges, and Krcmar 2010; Kühnemund 2007), are designed to simulate real workloads. Their only objective is to consume system resources, but often they are too simple to accurately reflect real system issues, such as unrepresentative disk or memory references.

Finally, after choosing the metrics and workload, the right evaluation technique is fundamental to get significant results. Most authors in literature, for example Heidelberger and Lavenberg (1984) or Jain (1991), distinguish between three different techniques for performance evaluation: measurement, analytic modelling and simulation. Ferrari et al. (1983) merge analytic modelling and simulation as one technique, since both require the construction of a system model. The selection of the right technique depends on the design stage of the system.

Compared to other performance techniques, measurement provides the highest accuracy at highest cost, but can only be used after the system has been built. There are different purposes for system measurement: to get information to characterize and model workloads, to validate system models, and to get insights into system behaviour to improve system performance. Typically, a monitoring tool is used for data collection. Snodgrass (1988) defined monitoring as "the extraction of dynamic information concerning a computational process, as that process executes". The main challenge in monitor design is to minimize the observation impact to the performance behaviour of the system, since a monitor requires a certain amount of resources every time it is activated. (Ferrari, Serazzi, and Zeigner 1983)

Following Trivedi, Haverkort, Rindos and Mainkar (1994), analytic modelling can be used at very early design stages but provides only a limited accuracy. Jonkers (1994) divides performance modelling formalisms into two classes: deterministic and probabilistic. In deterministic models, all quantities such as timing parameters are fixed, while probabilistic models allow some degree of uncertainty. The latter enable the solution of models that would otherwise be analytically intractable, due to the assumption of certain time distributions. Well-known representatives of probabilistic models are Markov Chains (Trivedi 1982; Trivedi et al. 1994), Petri Nets (Peterson 1981) and Queuing Networks (Bolch et al. 2006; Kleinrock 1976a; Kleinrock 1976b; Lazowska et al. 1984; Tijms 2003)

Higher accuracy but associated with higher costs can be achieved with simulation. In summary, simulation imitates the operation of a real-world process or system over time (Banks et al. 2004). A simulation can take place at any point in the life-cycle of the product. A major advantage of simulation over analytic modelling is that it can be used to create very detailed, accurate models. The other side of the coin is the fact that detailed models are often time consuming and difficult to design. A wide variety of simulation types exist and can be categorized in stochastic or deterministic and static or dynamic simulation (Jain 1991).

3. RELATED WORK

According to Rolia et al. (2009), there is not much work done in the context of performance evaluation and prediction of ERP systems. Rolia et al. (2009) investigate response time behaviour of a SAP ERP system using Layered Queuing Networks (LQN) models. The workload is composed by a fixed mix of sales and distribution requests. The requests are repeated cyclically 20 times, which corresponds to experiment duration of about 40 minutes. The results presented show that the LQN model offered mean response time predictions within 15% of measured values for a wide range of load levels. Gradl et al. (2010) pursue a similar approach for ERP business process modelling. A case study of an existing production planning process shows how LQN models can be exploited as a performance analysis tool.

Seelig et al. (2008) discuss performance evaluation techniques by comparing the results of one analytic and one simulative approach. The environment used in this work is a SAP Web Service application. The authors conclude that the simulation approach is suitable even in a very early stage of software development, enabling the software architect to identify potential hotspots prior to actually implementing the software components. On the other hand, quantitative models not only validate the simulation results, but also enable the architect to evaluate variants of the modelled component without much additional effort.

An evolutionary model generation for ERP performance simulation is presented by Tertilt et al. (2010). In performance models, some of the components in complex ERP systems are handled as black boxes (e.g. due to intellectual property). To increase the prediction accuracy of these components, an evolutionary algorithm is used. First results of the prototypical implementation showed the feasibility of the depicted approach as long as the data used for modelling was equally distributed. However, if the

measured performance data is unbalanced, the error increases significantly.

Besides the analytic and simulative approaches mentioned, different efforts have been made to measure performance of ERP systems. Jehle (2010) analyzes the performance behaviour of a portal system by SAP. Performance measurements are effected on identically configured test systems, both on native and virtualized environments. The results show that under certain circumstances a virtualized environment even increases the performance of the portal system being studied, namely, if the test workload is moderate and causes excessive Input/Output (I/O) loops. Similarly, Boegelsack, Wittges, and Krcmar (2010) investigate the scalability and performance of a virtualized SAP system on the basis of a quantitative approach and gives recommendations how to configure a SAP system for heavy workload. It is also shown that the average performance of a SAP system increases if a container-based virtualization solution is used, and decreases with a Xen-based virtualization solution.

4. SURVEY PREPARATION AND RESULTS

To understand whether and to which degree performance evaluation is currently implemented in companies, an online survey has been conducted. The main questions mentioned above can be broken down as follows:

- Which demands do companies have concerning performance evaluation of ERP systems?
- Do companies evaluate the performance of ERP systems? If so,
  - Which methods and metrics are used?
  - Which parts or layers of their systems are evaluated?
  - Which tools are used?
- Are companies satisfied with currently used methods?
- How big is the interest in this topic?
- What are the benefits expected?

Based on these questions, the basic structure shown in Figure 1 forms the backbone of the questionnaire.

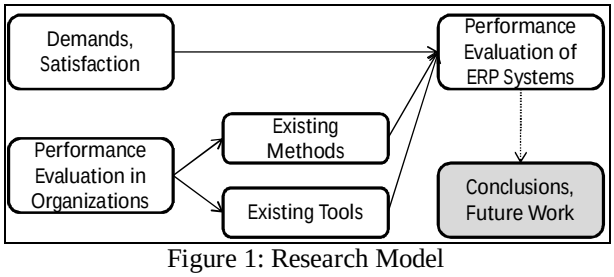


Figure 1: Research Model

4.1. Structure

Besides introduction and conclusion, the final version of the survey contains four blocks of questions concerning performance evaluation of ERP systems. The first one deals with questions about the general performance evaluation of ERP systems in organizations, the second one with questions about the concrete system, and the last two ones contain questions about demographic data of organizations and persons responding to the survey.

4.2. Data Collection and Demographics

The online survey was initiated on two big internet platforms in December 2010 and continued for a 6-weeks period until January 2011. The chosen platforms were the German-speaking SAP User Group (DSAG) and the SAP Developer Network (SDN).

These two platforms were chosen for the survey due to the composition of their members. Both are specialized platforms to SAP ERP systems. We received 36 fully completed and usable questionnaires from 116 participants, which equals to a return rate of 31.03%. This sample is not as big as we had expected, but still sufficient for a reliable evaluation. Due to the limit theorem, the normal distribution is sufficiently exact for a sample of $n > 30$ (e.g. Pal and Sarkar 2005).

The industry most represented was IT services (25%), followed by public administration (19.44%), manufacturing (11.11%), banks, assurances and financial services (11.11%), automotive (8.33%), mechanical engineering and construction (8.33%), and others (16.68%). Companies with less than 2000 employees represented almost 60% of the participating organizations. Most answers were received from IT consultants (33.33%), followed by IT managers (27.78%) and IT project leaders (27.78%). 97% of the participants of the survey are long year veterans regarding ERP systems. It can be expected that these results are based on a high level of experience.

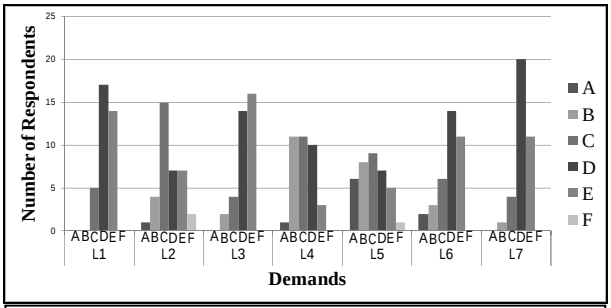
4.3. Results

The survey shows that more than 97% of the interviewed persons are very open-minded to performance evaluation of ERP systems. About two-thirds of the participants believe that performance evaluation is very important. All participants measuring performance use common metrics, namely, response times (100%), capacity utilization (100%), and throughput (37.5%).

4.3.1. Demands, Expectations

The participants were asked about their demands on performance evaluation of ERP systems. The result is shown in Figure 2.

To get a better understanding of which factors are most important for the respondents, the response options A (not important at all) to E (highly important) were given weights from 0 to 4. Result F (no answer given) is also given the weight 0. Subsequently, the percentages have been multiplied by the weight and normalized.



Legend:
L1 Improvement of the system's quality
L2 Flexibility
L3 Identification of critical components of the system
L4 Cost reduction
L5 Gaining of strategic advantages in competition
L6 Reduction of risk
L7 Increase of efficiency
A Not important at all
B Nice to have but not really important
C Important for some systems/parts only
D Important for most systems/parts
E Highly important for every System/part
F No answer given

Figure 2: Demands on Performance Evaluation of ERP Systems

The result showed that L1 (to improve the system quality) and L3 (identification of critical system components) are most important for the respondents, closely followed by the requirement L7, to increase efficiency. Two other demands, which were not included in the poll, but were added by the respondents, are operational safety and system stability.

4.3.2. Performance Evaluation of ERP Systems in Organisations

For the further course of the questionnaire it was important to check if companies already evaluate the performance of their ERP systems (see Figure 3).

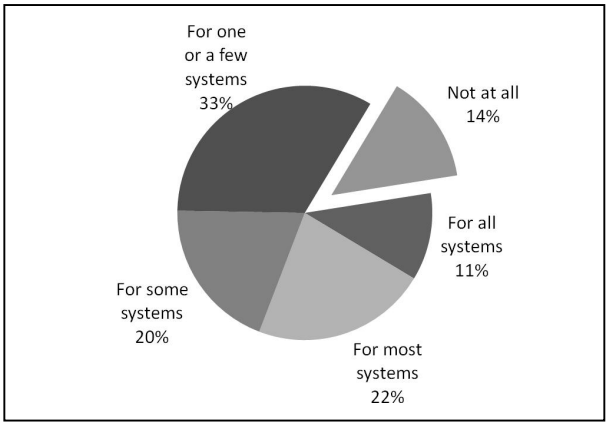


Figure 3: Performance Evaluation of ERP Systems in Organisations

About 86% of participants' companies have already evaluated performance of ERP systems at least once. On the other hand, those who use performance evaluation on a regular basis for most or all of their systems are clearly underrepresented with only about 33% of the participants.

In this context, it would be interesting to see, whether there are dependencies between these results and the company size or IT budgeting. Unfortunately, the sample size does not allow giving statistically relevant statements. The participants that are not evaluating the performance abandon it due to not enough resources (60%) or competencies (40%).

4.3.3. Established Tools

Only half of the participants of the online survey are using tools and methodological approaches to measure performance. All considered companies use inherent tools of the ERP system. The survey did not ask for the reason, but it can be assumed that these ERP integrated tools are used not only to avoid potential compatibility problems and version release dependencies when using third party tools, but also to avoid additional licensing costs.

Another question about tools is shown in Figure 4. It has been asked, how flexible these tools are in respect of application field, functionality and modifications.

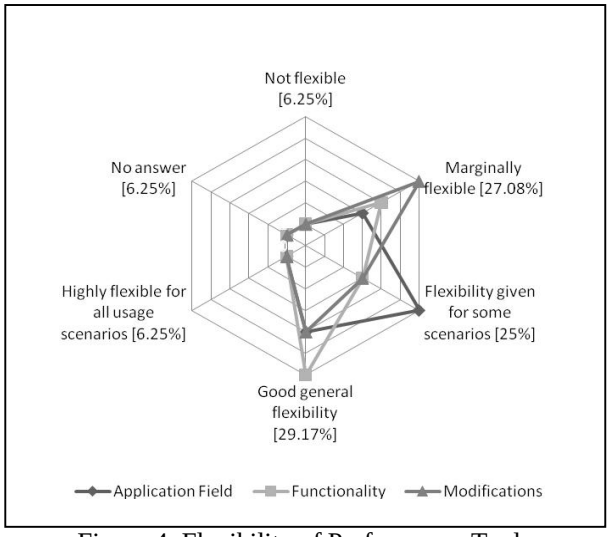


Figure 4: Flexibility of Performance Tools

More than 58% of the respondents are not or only partially satisfied with the flexibility of the product. As we had seen before, all of them are using ERP inherent performance tools, although the given answers indicate that they are not satisfied with the flexibility of their tools. Normally, one would assume that the customer would consider switching to a different tool in such cases. But it seems that the assumption taken above is backed by this result. In addition, the assumption is also supported by the fact that only 12.50% had switched the performance tool in the past.

In conclusion, the respondents use the offered ERP integrated tools and products but are not really satisfied. Interestingly, the major dissatisfaction comes with the suitability for performance measurements of future workload characteristics (e.g. with increased loads).

4.3.4. Established Methods

The next question concentrated on methods used for performance evaluation purposes. Here, multiple answers were possible, because the organizations might use different methods in different phases in the life cycle of an ERP system, or they could use different methods to compare these results and thus get a probably more reliable overview of system's performance.

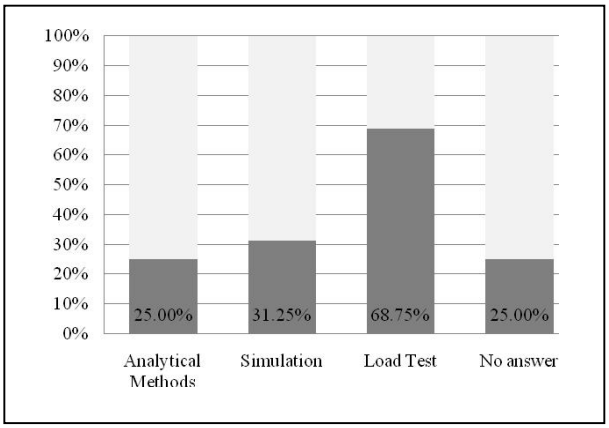


Figure 5: Methods Used for Performance Evaluation

Figure 5 shows that almost 70% of the respondents are using workload tests, 25% use analytical models and 31.25% use simulations. One of the reasons might be the concentration on performance behaviour of the current system configuration, without the intention to further investigate alternative scenarios with different workloads. In addition, we asked the participants about the drawbacks of their currently used methods. Most of them said that on the one hand these methods are not entirely flexible and on the other hand there is no end-to-end view across all layers.

In retrospect, interesting aspects for future data acquisitions are the reasons why there are no further tools other than ERP integrated instruments used or developed, and which functional improvements and additions are required to cover all aspects to entirely evaluate performance of the whole system and software stack.

4.3.5. User Satisfaction

Another important question concerned with the satisfaction of reliability, namely, the robustness of tools versus failures and the reliability of performance results.

As shown in Figure 6, no respondent is absolutely satisfied with reliability. This confirms the already gained knowledge about the weaknesses of currently used tools. Surprisingly, one-third of the respondents, who all measure performance regularly and most likely did not switch the tools in the past, question the reliability. The lack of confidence correlates to the results about functional weaknesses of currently used tools.

Figure 6: User Satisfaction Concerning Performance Evaluation Reliability

5. CONCLUSIONS AND FURTHER RESEARCH

The survey shows that almost all participants (97%) are open-minded to performance evaluation of ERP systems and 86% of them currently measure performance at least for some components or systems. All of them measure the performance in the phase of operation and maintenance. Almost 70% use workload tests to measure performance, 25% are using analytical models, and 33% are using simulations. Performance predictions for increasing workloads are rarely made. This is reasoned by functional weaknesses of the tools used.

Interestingly, only ERP inherent and no third party or own developed tools are used. These integrated tools only partially satisfy the demands of the participants. Although only the minority of participants is satisfied with the reliability of performance evaluation results, almost no one has changed the method or tool used, due to the lack of know-how, resources, and high implementation costs. Comparing these results with well-established performance evaluation techniques in literature, it is assumed that there are a lot of potentials to improve methods and tools in practice.

The advantages seen in performance evaluation of ERP systems include mainly improvements in efficiency (85.29%), risk reduction (76.47%), increased flexibility (50%) and cost reduction (44.12%). Consequently, further research should be done to combine existing performance evaluation techniques in literature with requirements in practice. To identify the demands in detail, additional data acquisitions are necessary. For this purpose and with the assistance of the expert communities, semi-structured interviews are being considered. Based on these findings, aligned methods for performance evaluation of ERP systems can be developed.

ACKNOWLEDGMENTS

The online survey has been conducted via www.2ask.de. For a copy of the survey questionnaire, results, and analysis please contact the authors.

REFERENCES

Dsag - Deutsche Sap Anwendergruppe. Available from: http://www.dsag.de [February 12 2011].

Sdn - Sap Developer Network. Available from: http://www.sdn.sap.com [February 12 2011].

Banks, J., Carson, J., Nelson, B. L., Nicol, D., 2004. *Discrete-Event System Simulation*. Englewood Cliffs, NJ, USA:Prentice-Hall.

Boegelsack, A., Wittges, H., Krcmar, H., Year. Scalability and Performance of a Virtualized Sap System. *Proceedings of the 16th American Conference on Information Systems*, Paper 13. August 12-15, Lima, Peru.

Bolch, G., Greiner, S., de Meer, H., Trivedi, K. S., 2006. *Queueing Networks and Markov Chains: Modelling and Performance Evaluation with Computer Science Applications*. Hoboken, NJ, USA:John Wiley and Sons.

Ferrari, D., 1986. Considerations on the Insularity of Performance Evaluation. *IEEE Transactions on Software Engineering* (12:678-683).

Ferrari, D., Serazzi, G., Zeigner, A., 1983. *Measurement and Tuning of Computer Systems*. Englewood Cliffs, NJ, USA:Prentice-Hall.

Gradl, S., Mayer, M., Wittges, H., Krcmar, H., Year. Modelling Erp Business Processes Using Layered Queueing Networks. *Proceedings of the 12th International Conference on Enterprise Information Systems*, 255-260, Funchal, Portugal.

Heidelberger, P., Lavenberg, S. S., 1984. Computer Performance Evaluation Methodology. *IEEE Transactions on Computers* (33:1195-1220).

Hu, L., Gorton, I. "Performance Evaluation for Parallel Systems: A Survey," 9707, University of New South Wales, Sydney, Australia.

Jain, R., 1991. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modelling*. New York, NY, USA:Wiley/Interscience.

Jehle, H., 2010. *Performance Measurement of an Sap Enterprise Portal System in a Virtualized Environment*. University of Technology Munich.

Jonkers, H., Year. Queueing Models of Parallel Applications: The Glamis Methodology. *Proceedings of the 7th International Conference on Computer Performance Evaluation*, 123-138, Secaucus, NJ, USA.

Kleinrock, L., 1976a. *Queueing Systems, Volume 1: Theory*. Hoboken, NJ, USA:John Wiley & Sons.

Kleinrock, L., 1976b. *Queueing Systems, Volume 2: Computer Applications*. Hoboken, NJ, USA:John Wiley & Sons.

Kühnemund, H. "Documentation for Slcs V.2.3.," SAP AG, Linux Lab, Walldorf, Germany.

Lazowska, E. D., Zahorjan, J., Graham, G. S., Sevcik, K. C., 1984. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Englewood Cliffs, NJ, USA:Prentice-Hall.

Pal, N., Sarkar, S., 2005. *Statistics: Concepts and Applications*. New Dehli, India:Prentice-Hall.

Peterson, J. L., 1981. *Petri Net Theory and the Modelling of Systems*. Englewood Cliffs, NJ, USA:Prentice-Hall.

Risse, T., 2006. *Design and Configuration of Distributed Job Processing Systems*. Thesis (PhD). Technische Universität Darmstadt.

Robertazzi, T. G., 2000. *Computer Networks and Systems: Queueing Theory and Performance Evaluation.* New York, New York, USA:Springer.

Rolia, J., Casale, G., Krishnamurthy, D., Dawson, S., Kraft, S., Year. Predictive Modelling of Sap Erp Applications: Challenges and Solutions. *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, 1-9, Pisa, Italy.

Sauer, C. H., Chandy, K. M., 1981. *Computer Systems Performance Modelling*. Englewood Cliffs, NJ, USA:Prentice-Hall.

Scherr, A. L., 1965. *An Analysis of Time-Shared Computer Systems*. Massachusetts Institute of Technology.

Seelig, M., Kluth, S., Porzucek, T., Copaciu, F., Naumann, N., Kühn, S., Year. Comparison of Simulation and Performance Modelling - a Case Study. *Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, 49-56, Washington, DC, USA.

Snodgrass, R., 1988. A Relational Approach to Monitoring Complex Systems. *ACM Transactions on Computer Systems* (6:157-195).

Tertilt, D., Leimeister, S., Gradl, S., Mayer, M., Krcmar, H. "Towards an Evolutionary Model Generation for Erp Performance Simulation," in: *IADIS International Conference on Intelligent Systems and Agents (ISA)*, Freiburg, Germany, 2010.

Tijms, H. C., 2003. *A First Course in Stochastic Models*. Hoboken, NJ, USA:John Wiley & Sons.

Trivedi, K. S., 1982. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Englewood Cliffs, NJ, USA:Prentice-Hall.

Trivedi, K. S., Haverkort, B. R., Rindos, A., Mainkar, V., Year. Techniques and Tools for Reliability and Performance Evaluation: Problems and Perspectives. *Proceedings of the 7th International Conference on Computer Performance Evaluation*, 1-24, Secaucus, NJ, USA.

UNDERSTANDING THE PERFORMANCE BEHAVIOR OF A SAP ERP SYSTEM FOR THE USE OF QUEUING MODELS

Stephan Gradl[(a)], Manuel Mayer[(b)], Alexandru Danciu[(c)], Holger Wittges[(d)], Helmut Krcmar[(e)]

[(a) (b) (c) (d) (e)] Chair for Information Systems, Technische Universitaet Muenchen, Boltzmannstrasse 3, 85748 Garching, Germany

[(a) (b) (c) (d) (e)] {Stephan.Gradl, Manuel.Mayer, Danciu, Holger.Wittges, Krcmar}@in.tum.de

ABSTRACT

ERP systems support the management of a company's resources. As a large number of business-relevant processes are supported by ERP systems, the performance and availability of those systems is crucial for the success of a company (Apache Software Foundation 2010). We analyze the response time of 49350 requests. Furthermore, we interpret the system's internal behavior by fetching and analyzing the statistical data. As results we can show that queuing models can be used for evaluating the performance of SAP ERP systems as the response time behavior follows the assumptions of queuing theory, resulting in nearly constant resource consumption per user interaction task, independent of the number of parallel requests. By analyzing the reasons for these results, important insights into the performance behavior of SAP ERP systems for performance analysis and prediction are achieved.

Keywords: ERP, SAP, Performance, Load Test, Measurement, Analysis

1. INTRODUCTION

ERP systems support the management of a company's resources. As a large number of business-relevant processes are supported by ERP systems, the performance and availability of those systems is crucial for the success of a company (Krcmar 2009). In particular, we focus on the performance analysis of the SAP Enterprise Resource Planning (SAP ERP, formerly SAP R/3) application (Schneider 2008). SAP ERP is an integrated backend application with tens of thousands of installations worldwide designed for tracking and managing business processes in midsize and large enterprises. From a technical perspective, this application is built on top of a software integration platform that provides primitives to control the concurrency offered by application server and database server, the layered use of servers, asynchronous messaging, and priority scheduling for certain types of processing. According to Jain (1991), there are several classical approaches for capacity planning and performance evaluation of computer systems like measurement (benchmarking and stress testing), simulation and analytical modeling. To evaluate the performance using simulation techniques, the system has to be modeled. Performance modeling of an ERP system requires deep knowledge about the structure and its performance behavior. To achieve accurate and significant results using queuing models the system has to follow certain performance criteria (Chen et al. 2008):

- The CPU time per user interaction task has to be independent from overall system utilization.
- The CPU utilization has to increase linearly with the number of concurrent load steps.
- The response time has a constant section that is followed by a linearly increasing section, ending in an exponentially increasing behavior.

In the following we describe a case study we performed on an SAP ERP system to analyze the performance behavior of this system when set under heavy parallel. We measure the response time behavior of the system as a black box, and then go a step further and analyze the internal behavior of the ERP system by fetching and interpreting the system's statistical records. Section 2 of this paper provides the research context of this work, while Section 3 provides an overview of the required definitions. In Section 4 we describe how we measure the response time of the system, give a brief overview of the architecture of the system under test and the used benchmark, and point out the method we used to create load. Section 6 then follows with the measurement results, as well as with their interpretation, and the analysis of the statistical records of the ERP system. Finally, in Section 7 we conclude our results and give an overview about the next steps and future work.

2. RELATED WORK

The key literature about performance measurement and analysis of (enterprise) software systems are the books of Jain (1991) and Lilja (2000). These authors describe elaborately the whole process of performance measurement, pointing out what performance is, how it is measured, and which factors affect the performance of a software system. We are basing our work on the

definitions made in these books, and adopt them to the fields of ERP. The importance of performance analysis is pointed out by Menascé (2002). An overview of the factors that determine the performance of an application is given by Bailey (2005) and Hollingsworth (2005). For the performance of an ERP system, we refer to (Schneider-Neureither 2004). In this book, the author explains in detail the effect of the SAP architecture and configuration on its performance, focusing on the solution of concrete operational problems. Although the book is written as an administrator manual, it provides a good overview of the factors affecting the SAP system´s performance. An overview of existing SAP benchmarks is given in (Prior 2003).

A scientific approach for the measurement of ERP performance behavior – in this case focusing on the effects of virtualization - presented by Jehle (2009) and Bögelsack (2010). While Jehle is focusing on the response time behavior using a load test, Bögelsack (2008) is analyzing the system´s internal matters, especially the CPU time, for interpreting its effect on the system performance.

Jin (2007) shows a method for performance prediction of legacy information systems. As the internal architecture of the investigated productive information system is not known, the authors used a method that is based on a black box approach for predicting the technical performance of this legacy information system with historical values. This approach combines benchmarking, production system monitoring, and performance modeling (BMM) by analyzing and correlating the performance values derived from the benchmarks and monitoring. Based on the measurements, a model is created and used for the performance prediction.

In (Rolia et al. 2009), an LQN model for the performance prediction of an SAP ERP system is introduced. In this approach the statistical records provided by the SAP system are used for performance analysis and prediction. In addition, the authors also used CPU values gathered from an SAP tool called saposcol. The workload used is based on a sales and distribution scenario, very similar to the workload that is applied in the SAP SD benchmark. Buffers, both from the applications server and the database, having a significant impact on the overall performance, are not taken into account.

3. DEFINITIONS

For measurement the performance of an application, one first has to define what is understood as performance in the given context, and which metric(s) are considered for the representation of an application's performance. Our understanding of performance is best shown by the following definition, taken from (Schneider-Neureither 2004).

Generally spoken, the performance of a data processing system is its ability to match the requirements in response time and throughput.

As already given by this definition, the most popular metrics for an application's performance are response (or execution) time and throughput. Nevertheless there are other, like the number of accesses to a special resource of energy need. In our context of ESOA, the response time as defined by Nudd (2000) and shown in figure 1 the time from the moment the request is sent (T1) until the time, when the response is completely received (T3) – is the more relevant metric, as the service calls are considered to be short running by time critical (in contrast to a batch job, where in general the throughput is more relevant than the single response time).

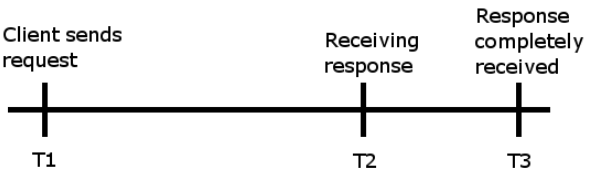


Figure 1: The Structure of the response time (according to Menasce (2002))

Even though it is obvious that the response time and the throughput are connected, in our work we focused on the response time, since the response time is the actual time a user waits while performing a task. On a high dispersion of response times, the throughput could be still eligible, while some high response times are inacceptable (e.g. due to Service Level Agreements).

4. MEASUREMENT

Following (Lilja 2000), the most common benchmark strategy is the fixed-computation approach in which the total time required to execute the benchmark is used as the performance metric. The complementary approach is to fix the amount of time, where the total amount of computation completed in this time period is used as performance metric. The most flexible benchmark strategy is to derive a third dimension from some combination of the execution time and the amount of computation completed within this time. In this way (using this third dimension as performance metric), execution time and computation can be kept variable. The Hierarchical Integration Benchmark (HINT), for instance, uses quality improvements per second as performance metric, defined as a function of the problem being solved by the benchmark program. Table 1 summarizes the strategies that can be used in a benchmark program to exercise the system under test.

For our case study we fixed the amount of computation while measuring the time, needed by the sap ERP System to execute it.

As introduced later on, the benchmark consists of the creation of a material master record in the SAP ERP system. For this task, a WebService in the SAP system has been identified, which has been used for creating load on the system. This service is called in parallel by an own implementation of a Java load generator. For measuring the system behavior, we combine the black

box approach described by Kruse (2009), and the glass box approach used by Malik (2010). In the following section, we illustrate the architecture of the system under test, the benchmark, and the load generator.

Table 1: Benchmark Strategies (based on (Lilja 2000))

Time	Computation	Performance Metric
Variable	Fixed	Execution Time
Fixed	Variable	Consumption completed
Variable	Variable	Third dimension

5. SYSTEM ARCHITECTURE

To provide an understanding of the ERP system architecture shown in figure 2, we derive the system components from the ERP process step-by-step by analyzing the recorded trace and the abstraction of the trace entries. These components are described in detail in (Schneider 2008). The process step of calling a program involves many components of the SAP system (see figure 2).

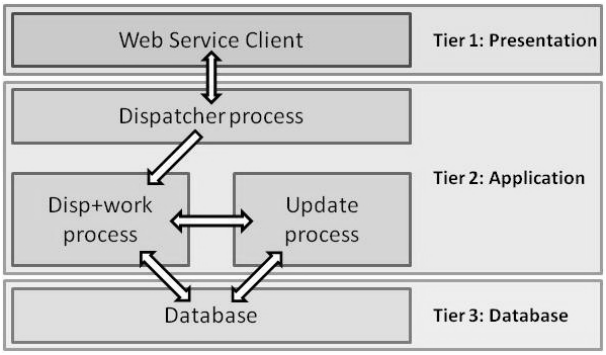


Figure 2: Simplified SAP ERP system architecture

Searching for the program includes access to internal buffers as well as access to the database tables. This access is made by the so called disp+work processes of the SAP system. Such processes are responsible for executing programs, processing user or WebService requests, and accessing the database. Before a request is associated to one of the disp+work processes of the SAP system, a dispatcher process is accessed. The dispatcher process manages all other processes in the SAP system, and his primary task is to assign a user request to a free disp+work process. In our model, we assume the database as a black box.

After the SAP system got the information which program has to be executed, it loads a compiled version of the program from the database and executes it. Sometimes such compiled programs are held in the internal buffers of the SAP system to avoid database accesses.

After the request is processed, the data should be saved to the database. This is done by the disp+work process(es) together with a process called update process. This process receives data and stores it in corresponding database tables.

Simultaneously, a lock on a central table is established, which may be described as a little repository of all available material master records (MMR) within the system. This lock is not set by the disp+work process itself; it triggers a so-called enqueue process. The only task of the enqueue process is to set locks on any tables in the SAP system, and to manage such locks. After the lock was set successfully, the disp+work process can store the data into the central MMR repository.

5.1. Benchmark

For the load run processed in this case study, the Production Planning Integration Case Study (Weidner 2006) has been used. In detail Web Services creating different kinds of material master records, bill of materials, routing, etc. has been chosen. As shown in table 2, these Web Services have an average complexity with read, insert and update statements to the database. Each execution of the case study does not depend on another one and therefore can be executed anytime using the SAP Web Service Interface.

As already introduced, the programs in a SAP ERP System are running on an infrastructure containing dispatching, locking, buffering and database access mechanisms (Schneider 2008). In order to understand the performance of a SAP ERP system, it is necessary to use a workload that uses these components. This leads to different kinds of database queries that characterize the Web Service calls from a technical side.

Table 2: Database accesses for material creation

	DBRows	Read to Buffer	ReqTime
Direct Read	6	222	5134
Sequ. Read	2754	176	6352196
Insert	122	0	12633
Update	1	20	686

Table 2 shows different kinds of database accesses. As the labels Insert and Update are self-explanatory, “Direct reads” are always in the form of “select single” and fetch exactly one row from the database, while queries in the form of “select * from...” are named Sequential reads. The column DBRows shows the number of rows that are fetched directly from the database without being served by the buffers, while the column read2Buffer shows the number of requests that could be served from caches. The column ReqTime contains the time requests took. This entry does not include requests time served by the caches, since these times can be neglected, according to the technical documentation of the Transaction STAD.

In addition, the workload emulates an existing business process, affecting the already mentioned key components (buffers, locks, database accesses, etc) on the technical layer.

For the load test, an ERP installation with an application server and a database server, both hosted on a physical server with 16 GB of Ram and 4 Cores running at 2.8 GHz, has been used. The application

server and the database were provided in virtual containers using SUN Solaris Zones. The application server was configured with 30 work processes and the database (MaxDB, version 7.7) with up to 150 parallel connections. For the case study, we used an SAP system with customizing and data of the SAP International Demo and Education System (IDES).

The underlying database for this installation, containing one IDES client, has a size of 220 GB and uses Unicode. The database data files are provided on two internal SAS Discs configured in a performance raid with raid level 0.

5.2. Load Generator

Load generation is done by the own implemented Java application Load Generator. The Load Generator uses threads for parallelization, and the Axis2 framework (Apache Software Foundation 2010) for calling the web service.

A load run is conducted by a stepwise increasing number of parallel service calls. To minimize the amount of overhead, we initialize the payload once, cloning the object tree for each call, and changing the material number. The load procedure calls the service i times in parallel for every step i in the load test (where i = 1 to n, n = the number of maximum parallelism set for the load test), then waits for all results to be stored, and finally increases i by a given step size (step size 1 in our case). Doing this, it is assured that every sequence, which is a load unit of a certain number (i) of parallel requests, is not affected by the request before. For this purpose, a configurable wait time after each sequence has been implemented, too. This short time frame between the sequences is used to fetch the later discussed statistical values from the SAP system, in order to keep these values available and the amount of data, which has to be transferred, as small as possible.

This results in a response time distribution matrix containing the response times for all (i) sequences, and a total number of calls of m_calls=(n*(n+1))/2.

Experiments showed that the initialization of caches on first requests results in non proportional and unpredictable long runtimes. To avoid these “cold start” effects, we perform a settling phase of three times forty requests before starting the measurement. In this way we assure that all caches are initialized, as can be seen by the cache hit statistics provided by the SAP system.

6. RESULTS AND ANALYSIS

In this section we provide two views of the system under test. At first, we interpret the response time behavior, seeing the system as a black box (cf Ludewig (2007)). Afterwards, we switch to the glass box view, taking a deeper look at the system’s internal behavior, analyzing the statistical data, and providing an illustration of where the presented response time behavior originates from.

6.1. Measurement

Figure 3 shows the response time results for five load tests running from one to 140 parallel requests. The diagram contains 49350 response time results, resulting in an easily recognizable behavioral pattern. Using figure 4, we will explain this pattern in the following.

Taking a close look at the response time diagram, one can see that the pattern can be split up in three parts. The first part (marked as 1) is where the number of parallel requests is smaller than the number of available work processes on the ERP system. All requests can be handled by the system in parallel. An increasing number of parallel requests slows down the response times for all requests, as can be seen by the difference between 1.1 (one request) and 1.2 (30 parallel requests). Nevertheless, this slowdown is affecting all requests evenly.

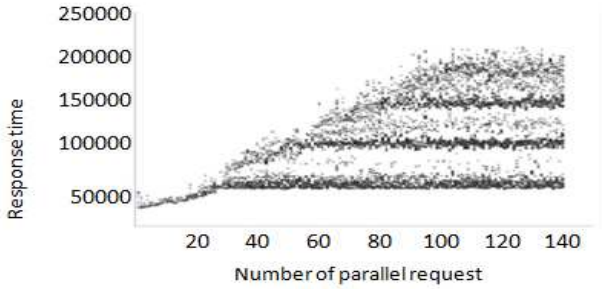


Figure 3: Response time diagram for five load tests

When the number of parallel processes exceeds the number of available work processes, the message queuing used by the ERP system for load balancing becomes visible. The first block of n requests is processed in parallel (where n is the number of available work processes) in a constant time (2), independent of the overall number of requests. This is comprehensible, as the surplus of the requests stays in the queue and thus is not consuming any relevant resources.

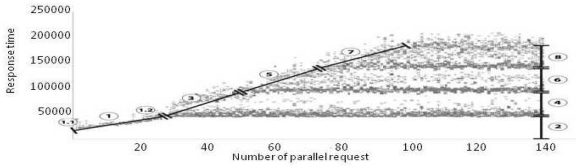


Figure 4: Response time diagram analysis

The surplus of requests is processed after work processes finish the first request. A repeating pattern is recognizable, as the surplus of requests (6) is processed as the requests in section 1. This continues (3 and 7, 4 and 8), up to approximately 100 parallel requests. From this point on, the maximum response time stays constant – due to a timeout of all longer running requests.

As a result we can say that the system’s capacity is at about 100 parallel requests. Passing this limit will result in requests not being successfully answered. But

whatever excessive load generated, at least these 100 requests will be successfully processed.

The response time of a single request though is quite unpredictable.

6.2. Glass Box View

To analyze the response time in a glass box view, the relevant components addressed in section 1, and the time slots they use, have to be introduced. The response time in the SAP ERP ABAP stack is quite complex and consists of the following components:

- wait time
- roll in time
- load/generation time
- database time
- enqueue time
- roll out time
- rolled out time
- time in workprocess

As illustrated in figure 1, each request that arrives at the system has to be assigned to a work process in order to get processed. This assignment is done by the dispatcher process. The time for this step is called wait or queue time. The overall time the program is processed by the work process is called time in work process.

As shown in figure 5, this time consists of several components. The first processing step in the work process is to load the process context in its memory. This is called roll in time. If a program calls a remote service, the time, while it has to wait for the response, is also assigned to the roll in time. Then the program has to be loaded into the process memory or generated (compiled) from source code, if it is called the first time. While the request is processed, the work process fetches data from the database. This is aggregated into the metric database time.

The SAP ERP kernel has its own mechanism to control concurrent access to database objects, the so called enqueue process. If a resource is busy, then a work process has to wait until it can be obtained. This slot is called enqueue time. As soon as the work process is finished, the information has to be unloaded from the process memory into the systems shared memory, which is measured by the roll out time. If a request consists of more than 1 work process calls, then the time between the end of the fist call and the beginning of the next call is registered as rolled out time.

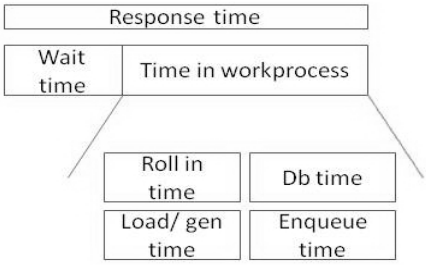


Figure 5: Response time components

These dependencies are shown in figure 3 (response time). In this illustration, the Time in work process consists of the components roll in time, load/generation time, database time, and enqueue time. The roll out time is part of the rolled out time and is not part of the response time, as the response is sent to the client before, in order to reduce the system response time. Different other metrics are not collected, but calculated by the kernel.

The CPU time is a subset of the response time and cannot be assigned to a special component, but is returned by the operating system timer. In the case, of UNIX, the timer works with 100 Hz and consequently the CPU time is always a multiple of 10ms. To analyze the system behavior, the introduced metrics have to be measured. This is done by the SAP ERP kernel completely independent of any ABAP application. The kernel logs a set of performance metrics, like the components of the response time, the program and user name, response size and other important information for performance analysis of the system.

After a request is processed, the work process collects the available information, calculates additional metrics, and stores it in the shared memory. This memory can be accessed by all work processes, and so the performance metrics of the full. As soon as the buffer is full, it is written to binary files on the file system. Every hour a new file named stat is created, while the old one is renamed to stat_<number>. In this way, the statistical records can be accessed as long as the maximum number of stat files is not reached. As soon as this maximum number is exceeded, the oldest file will be deleted. The number of these stat files (and with it the amount of statistical records that are held) is controlled by parameters of the SAP system. For this case study, their values have been increased to hold all the data of a load run. As the logging of statistical records is a standard functionality of the kernel that is always turned on, this method of monitoring can be referred to as non intrusive as mentioned in JAIN, as it does not add an additional load on the system in comparison to a productive usage of the ERP system.

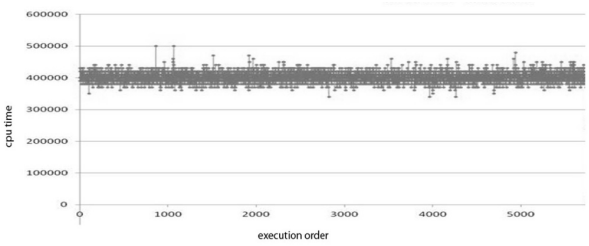


Figure 6: CPU time analysis

The statistical values of each program are displayed using the Transaction STAD in the SAP Gui. To evaluate the big number of these performance values in the presented case study, they have to be exported to a file or a database. For this, several BAPIs exist to deliver the statistical records using RFC function calls. The most complete values are delivered by the BAPI SAPWL_STATREC_DIRECT_READ. As this BAPI is not exported as a web service by the SAP system, it has been copied and modified to meet the requirements for the export as a web service. Thereby, the feature of obtaining the statistical values as soon as the load run is over has been integrated in the presented load generator and analysis tool. To have these performance values easily available, a database has been created to store these values of each load run.

Taking a closer look on the measured values, as discussed in section 5, a clear pattern can be observed. Using the glass box approach, this pattern will be explained by analyzing the process times of each component of the SAP ERP systems kernel. As the system has been warmed up, before the load test has been executed, load/generation time can be neglected, as the ABAP code is already in the corresponding buffer. The same effect can be observed in productive systems, as the size of the buffers is adapted to the expected workload. In addition, the amount of data that has to be rolled in at the beginning of the processing phase is too small to have influence on the systems performance.

The determining factor for the overall response time are the CPU time, total database time, commit time, which is a part of the total db time, and the queuing time. The measured values of these metrics will be introduced in the following to illustrate the performance behavior of the web service used in this case study.

Figure 6 shows the CPU time in nanoseconds according to the timestamp the request has been processed. As the load run has been executed with a growing number of parallel requests, this graph shows that the CPU time, as expected, does not depend on the number of parallel requests. The overall response time is not negatively affected by the CPU time under a growing number of parallel requests.

Figure 7 illustrates the database commit time in nanoseconds; also according to the timestamp the request has been processed. This graph shows a few irregularities, which are created by the log writer mechanism of the underlying database. Besides these irregular values, it does not show any dependencies to the number of parallel requests.

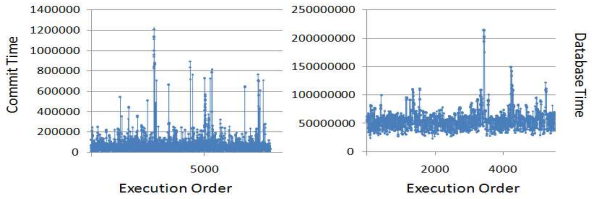


Figure 7: Commit time and database time

The total database time is shown in the right part of figure 7. There is no indication that the performance of the database component does significantly change with the number of parallel requests. As the ERP system has a certain number of work processes, it has been assumed that the database response time does not change significantly with a rising number of parallel requests to the SAP ERP system, as there are not more requests arriving at the database than work processes in the system exist. But even in the starting phase, where less parallel requests had to be served by the system and the database, the performance of the database is in a certain frame.

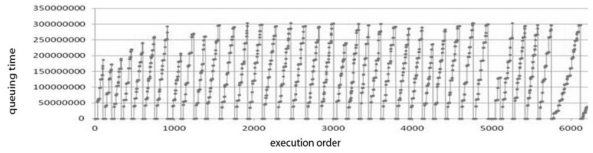


Figure 8: Queuing time

An explanation of the measured values shown in figure 4 is found in figure 8 showing the queuing time in nanoseconds; according to the timestamp the request has been processed.

The graph shows a significant dependency to the number of parallel requests. As in the stating phase the queuing time is small, it grows with more requests sent to the system. With 30 work processes configured, the first 30 requests had no or a negligible queuing time, while the remaining requests had to wait until one of the first requests has been finished. Consequently, the queuing time rises with the number of parallel requests until the sequence is finished.

7. LIMITATIONS

This case study does not regard the usage of different technical users on the SAP system. However, in a well configured productive SAP system, the shared memory that is intended to hold the user contexts is big enough to hold all data in shared memory without the need to put some data into a slower memory area. Therefore, the simplification of reducing the amount of different users does not reduce the significance and validity of the case study's results.

In addition, the impact of a suboptimal system configuration to system response times is not part of this work. Using a non optimal resource distribution,

several system components may respond differently compared to the response times and components' behavior in general measured in this case study. For instance, in some special cases the work process is assigned to a user as long as he is logged into the system. This work aims at understanding the general concept of the response time of an ERP system that is well parameterized, and not at the analysis of performance problems.

Equally, the different behaviors of ERP systems that are used for development purposes, so called "DEV" Systems, are not regarded in this case study.

A challenging task when analyzing complex software systems is the decision how many components should be integrated in the analysis. Even in this defined example, the paper demonstrates that a lot of data is gathered from several components in the SAP system and that even the database can be described in a more detailed way. In (Gradl et al. 2010) an eight level architecture was presented to limit the effort of building the architecture of the SAP system. By analyzing the SAP system and its traces, it was discovered that the lowest level is the response time level of the database. As the SAP system does not provide more detailed information about the database, the database has been treated as black box. Response time values of the database are derived directly from the SAP system.

8. CONCLUSION

On the first view, the response time behavior of the analyzed service seems to be complex. But on a second view it reveals a quite predictable behavior. Using a queuing mechanism, the SAP system processes several requests in parallel, and at the same time it regulates the resource consumption by limiting the number of parallel processed requests to the number of work processes available. This leads to a stable environment not being affected by the amount of parallel requests, and a guaranteed response time for some of the requests.

Taking a look inside the system, it can be seen that the resource consumption behind the messaging layer is quite constant. Considering the queuing, this is not surprising. Only the commit time shows some volatility, caused by the fluctuation of parallel processes accessing the database. This volatility of the commit time also causes the jitter in the response time diagram.

In summary we draw the conclusion, that the dispatching time represents the difference between response times, while the processing time is constant independent of how many parallel requests are processed. While this sounds obvious at first, it is an important validation for performance analysis and prediction.

As next steps, we use the information acquired in this work and performance data gathered to parameterize a layered queuing model based on (Gradl et al. 2010) to predict the performance of a SAP ERP system via simulation (Woodside 2002).

REFERENCES

Apache Software Foundation "Apache Axis2/Java," 2010.

Bailey, D.H., and Snavely, A. "Performance Modeling: Understanding the Present and Predicting the Future," in: *Euro-Par 2005 Parallel Processing*, Springer, Berlin / Heidelberg, 2005, pp. 185-195.

Bögelsack, A., Jehle, H., Wittges, H., Schmidl, J., and Krcmar, H. "An Approach to Simulate Enterprise Resource Planning Systems," in: *6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2008, In conjunction with ICEIS 2008*, U. Ultes-Nitsche, D. Moldt and J.C. Augusto (eds.), INSTICC PRESS, Barcelona, Spain, 2008, pp. 160-169.

Bögelsack, A., Krcmar, H., and Wittges, H. "Performance Overhead of Paravirtualization on an Exemplary ERP System," in: *12th International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, 2010.

Chen, S.G., and Lin, Y.K. "Performance analysis for Enterprise Resource Planning systems," Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on, 2008, pp. 63-67.

Gradl, S., Mayer, M., Wittges, H., and Krcmar, H. "Modeling ERP Business Processes Using Layered Queueing Networks," in: *12th International Conference on Enterprise Information Systems*, Funchal, Portugal, 2010.

Hollingsworth, J.K., Snavely, A., Sbaraglia, S., and Ekanadham, K. "EMPS: An Environment for Memory Performance Studies," in: *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 10 - Volume 11*, IEEE Computer Society, 2005, p. 223.222.

Jain, R. *The art of computer systems performance analysis* Wiley, New York, 1991.

Jehle, H. "Performance-Messung eines Portalsystems in virtualisierter Umgebung am Fallbeispiel SAP," in: *CVLBA Workshop 2009. 3. Workshop des Centers for Very Large Business Applications (CVLBA)*, Arndt, H.-K.; Krcmar, H., Magdeburg, Deutschland, 2009.

Jin, Y., Tang, A., Han, J., and Liu, Y. "Performance Evaluation and Prediction for Legacy Information Systems," in: *ICSE´07*, IEEE, Minneapolis, 2007.

Krcmar, H. *Informationsmanagement* Springer-Verlag New York, Inc., 2009.

Kruse, H.G. *Leistungsbewertung bei Computersystemen*, (1. ed.) Springer-Verlag, Berlin, Heidelberg, 2009.

Lilja, D.J. *Measuring Computer Performance - A practitioner's guide* Cambridge University Press, 2000.

Ludewig, J., and Lichter, H. *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken*, (1. ed.) dpunkt.verlag GmbH, Heidelberg, 2007, p. 618.

Malik, H. "A methodology to support load test analysis," in: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ACM, Cape Town, South Africa, 2010, pp. 421-424.

Menascé, D.A. "Software, performance, or engineering?," in: *Proceedings of the 3rd international workshop on Software and performance*, ACM, Rome, Italy, 2002, pp. 239-242.

Menasce, D.A., and Almeida, V.A.F. *Capacity Planning for Web Services: metrics, models, and methods* Prentice Hall, Upper Saddle River, NJ, Upper Saddle River, NJ, 2002.

Nudd, G.R., Kerbyson, D.J., Papaefstathiou, E., Perry, S.C., Harper, J.S., and Wilcox, D.V. "Pace - A Toolset for the Performance Prediction of Parallel and Distributed Systems," *International Journal of High Performance Computing Applications* (14:3), August 1, 2000 2000, pp 228-251.

Prior, D. "Who Sets the Pace in the SAP Performance 'Olympics'?," *Gartner*), 24.2.2003 2003, p 6.

Rolia, J., Casale, G., Krishnamurthy, D., Dawson, S., and Kraft, S. "Predictive modelling of SAP ERP applications: challenges and solutions," in: *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Pisa, Italy, 2009, pp. 1-9.

Schneider-Neureither, A. *Optimierung von SAP-Systemlandschaften*, (1. ed.) Galileo Press, Bonn, 2004.

Schneider, T. *SAP-Performanceoptimierung*, (5 ed.) Galileo Press, Bonn, 2008, pp. 247-249.

Weidner, S. "Integrations-Fallstudie PP (SAP ECC 5.0)," 2006.

Woodside, M. "Tutorial Introduction to Layered Modeling of Software Performance," 2002.