Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

IMPROVED PERFORMANCE MODELS FOR WEB-BASED SOFTWARE SYSTEMS

WEBALAPÚ SZOFTVERRENDSZEREK TOVÁBBFEJLESZTETT TELJESÍTMÉNYMODELLJEI

Ph.D. Thesis Booklet

Ágnes Bogárdi-Mészöly

Advisors:
Tihamér Levendovszky Ph.D.
Hassan Charaf Ph.D.

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
DEPARTMENT OF AUTOMATION AND APPLIED INFORMATICS

Budapest, 2009.

Ph.D. Thesis Booklet

Ágnes Bogárdi-Mészöly

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

1111 Budapest, Goldmann György tér 3.

e-mail: agi@aut.bme.hu
tel: +36(1)4632486
fax: +36(1)4632871

Advisors:
Tihamér Levendovszky Ph.D.
Hassan Charaf Ph.D.

1 Preliminaries and Objectives

Web-based software systems provide users with the opportunity to save time and money, and improve the way to interact with clients, suppliers and business partners. The performance of web-based software systems is one of the most important and complicated consideration, because they face a large number of users, and they must provide high-availability services with low response time, while they guarantee a certain throughput level.

Workload characterization [Ferrari 1972] is a modeling process to reproduce the user behavior. The customers use different navigation patterns, different workload for E-Commerce systems. One approach analyzes the Customer Behavior Model Graph [Menascé 2000], while another is using Markov Chains [Kijima 1997]. Further methods use mathematical models like Phase-type distributions or Quasi Birth Death processes [Bobbio 2004].

Performance metrics are influenced by many factors. Several papers have investigated various configurable parameters, and the way in which they affect the performance of web-based software systems [Sopitkamol 2005]. Statistical methods and hypothesis tests [Trivedi 1982] are used to retrieve factors influencing the performance. Nowadays the Microsoft .NET environment [Platt 2003] became one of the most prominent technologies of web-based software systems. Through the settings of the application server, the performance can be affected in several ways [Meier 2004] [Wienholt 2003].

The performance metrics can be predicted at early stages of the development process with the help of a properly designed performance model and an appropriate evaluation algorithm. In the past few years several methods have been proposed to address this issue. Several of them is based on queueing networks or extended versions of queueing networks [Jain 1991] [Kleinrock 1975]. By solving the queueing model using analytical and simulation solutions, performance metrics can be predicted. Another group is using Petri-nets or generalized stochastic Petri-nets [Bernardi 2002] which can represent blocking and synchronization aspects much more than queueing networks. As the third kind of the approaches, the stochastic extension of process algebras, such as TIPP (Time Processes and Performability Evaluation) [Herzog 2000], EMPA (Extended Markovian Process Algebra) [Bernardo 1998], and PEPA (Performance Evaluation Process Algebra) [Gilmore 1994] can be mentioned.

Web-based software systems access some resources while executing the requests of the clients. Typically several requests arrive at the same time, thus, competitive situation is established for the resources. For modeling such situations queueing model-based approaches are widely used. Queueing networks have also been proposed to model web-based software systems [Menascé 2001] [Urgaonkar 2005].

The objective of my research was to establish performance models as well as evaluation methodologies, which are suitable for performance prediction of web-based software systems. Taking these aspects into account, my goals were the following:

- *Identify and investigate the dominant performance factors considering the response time and throughput performance metrics.*
- *Establish and investigate evaluation and prediction techniques applying the dominant performance factors considering the response time and throughput performance metrics.*

- *Provide and verify multi-tier queueing network models to model web-based software systems.*
- *Apply the methods to industrial applications, and demonstrate the feasibility of practical application.*

2 Methodological Summary

The starting points of my work were the already existing methods, models, algorithms. Firstly, I elaborated the related work, then, I collected the most important models, algorithms. This could help to understand the different approaches, and formed a basis for further research and novel approaches.

The requirements above had determined the direction of my research including the individual steps. The methods of the theoretical research are determined by the actual tasks, the already existing subsolutions, and the shortcomings of the existing solutions.

The whole research that facilitated to improve the present performance models can be classified into three groups:

- *Performance prediction and performance factor identification.*
- *Modeling the thread pool.*
- *Modeling the queue limit.*

I have analyzed the present performance models and evaluation algorithms to predict performance metrics of web-based software systems in ASP.NET environment and to provide a comparative base for proposed models and algorithms. I have identified and investigated novel performance factors. I have modeled these identified factors to improve performance models.

It holds for the whole research method that all of the expected results took aim at practically applicable solutions. The theoretical results of the research have been applied to an industrial application and have been realized in different modules of a developed web-based software system.

3 Novel Scientific Results

The results of my research work are summarized in three theses. The first thesis introduces and verifies queueing network models and evaluation algorithms to model multi-tier ASP.NET web-based software systems. I have shown that these models and algorithms can be used for performance prediction in ASP.NET environments. In my thesis work, I have proven the validity of these models and algorithms as well as the correctness of the performance predictions with performance measurements. This thesis also provides algorithms to identify and investigate the dominant factors considering the response time and throughput performance metrics. The identified performance factors must be modeled to improve performance models.

The second and third theses deal with the establishment and investigation of the evaluation and prediction techniques applying the identified performance factors. The second thesis proposes a novel algorithm which models the behavior of the thread pool. The third thesis contains novel models and algorithms for modeling the queue limit. I have investigated the computational complexity as well as the limit of the response time and throughput sequences provided by the proposed models and algorithms. I have shown that the proposed models and algorithms can be applied to performance prediction in ASP.NET environment. I have proven the validity of the proposed models and algorithms as well as the correctness of the performance prediction with performance measurements.

The rest of this work is organized as follows. Firstly, the main contribution of each thesis is described. Secondly, reference to the appropriate chapter of the dissertation followed by my publications related to the specific thesis is presented. Finally, the detailed description of the thesis is provided.

Thesis I
Performance Prediction and Performance Factor Identification

I have provided queueing network models and evaluation algorithms to model multi-tier ASP.NET web-based software systems. I have demonstrated that these models and algorithms can be applied to performance prediction of web-based software systems in ASP.NET environment. I have validated these models and algorithms and verified the correctness of the performance prediction with performance measurements. I have analyzed the computational complexity as well as the limit of the response time and throughput sequences computed by these models and algorithms in order to provide a comparative base for proposed models and algorithms.

I have proposed statistical methods for identifying and investigating novel performance factors. I have proven with statistical methods, that the thread pool attributes and the queue limit parameters have a considerable effect on the performance, in other words, they are performance factors. I have shown with statistical methods that the response time performance metric tends to a normal distribution. I have determined the parameters of the distribution by maximum likelihood estimation with successive approximation.

Thesis I is contained by Chapter 4 of the dissertation. Related publications: [1] [2] [3] [7] [8] [9] [10] [12] [13] [15] [16] [18] [19] [20] [21] [34].

Definition 3.1. *The base queueing model is defined for multi-tier information systems [Urgaonkar 2005], which are modeled as a network of M queues $Q_1, ..., Q_M$ illustrated in Fig. 1. Each queue represents an application tier. S_m denotes the service time of a request at Q_m ($1 \leq m \leq M$). A request can take multiple visits to each queue during its overall execution, thus, there are transitions from each queue to its successor and its predecessor, as well. Namely, a request from queue Q_m either returns to Q_{m-1} with a certain probability p_m, or proceeds to Q_{m+1} with the probability $1 - p_m$. There are only two exceptions: the last queue Q_M, where all the requests return to the previous queue ($p_M = 1$) and the first queue Q_1, where the transition to the preceding queue denotes the completion of a request.*

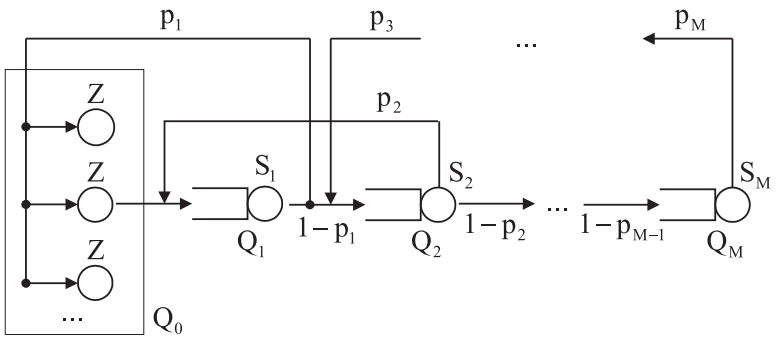

Figure 1: Modeling a multi-tier information system using a queueing network

Internet workloads are usually session-based. The model can handle session-based workloads as an infinite server queueing system Q_0 that feeds the network of queues and forms the closed queueing network depicted in Fig. 1. Each active session is in accordance with occupying one server in Q_0. The time spent at Q_0 corresponds to the user think time Z.

Incoming sessions of a web application can be classified into multiple classes. An enhancement of the base queueing model [Urgaonkar 2005] can handle multiple session classes.

Definition 3.2. *The Mean-Value Analysis (MVA) [Reiser 1980] for the base queueing model is defined by Algorithm 3.1, and the associated notations are in Table 1.*

Algorithm 3.1 Pseudo code of the MVA algorithm

1: **for all** $m = 1$ to M **do**
2: $L_m = 0$
3: **for all** $n = 1$ to N **do**
4: **for all** $m = 1$ to M **do**
5: $R_m = V_m \cdot S_m \cdot (1 + L_m)$
6: $R = \sum_{m=1}^{M} R_m$
7: $\tau = n/(Z + R)$
8: **for all** $m = 1$ to M **do**
9: $L_m = \tau \cdot R_m$

If multiple session classes are handled, the time and space complexities of MVA are proportional to the number of feasible populations, and this number rapidly grows for relatively few classes and jobs per class. Thus, it may be worth using an approximate MVA algorithm or a set of two-sided bounds.

The pseudo code of the applied approximate MVA algorithm [Sinclair 2005] is as follows (see notations in Table 1).

Algorithm 3.2 Pseudo code of the applied approximate MVA algorithm

1: **for all** m,c **do**
2: $L_{m,c} = \frac{N_c}{M}$
3: **repeat**
4: $E_{m,c} = \frac{N_c - 1}{N_c} L_{m,c} + \sum_{\substack{i=1 \\ i \neq c}}^{C} L_{m,i}$
5: Compute $R_{m,c}$ and τ_c using $E_{m,c}$
6: Compute $newL_{m,c}$ using τ_c and $R_{m,c}$
7: **until** $\frac{newL_{m,c} - oldL_{m,c}}{oldL_{m,c}} < \delta$

A system without a bottleneck device is called a balanced system, in other words, the total service time demands are equal in all queues. The bounds referred to as balanced job bounds are based on the fact that a balanced system has a better performance than a similar unbalanced system [Zahorjan 1982]. The balanced job bounds are very tight, the upper and lower bounds are very close to each other as well as to the real performance.

Definition 3.3. *Balanced job bounds are defined by Equations 3.1 and 3.2, and the associated notations are in Table 1.*

$$\begin{aligned} &\max\left\{ND_{\max} - Z, D + (N-1)D_{avg}\frac{D}{D+Z}\right\} \leq \\ &\leq R \leq D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z} \end{aligned} \tag{3.1}$$

$$\frac{N}{Z + D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z}} \leq \tau \leq \min\left\{\frac{1}{D_{\max}}, \frac{N}{Z + D + (N-1)D_{avg}\frac{D}{D+Z}}\right\} \tag{3.2}$$

Table 1: Notations of evaluation algorithms

Notation	Meaning	Notation	Meaning
C	number of classes	N	number of customers
D	total service demands	N_c	number of customers at class c
D_{avg}	average service demand	R	response time
D_m	service demand for Q_m	R_m	response time for Q_m
D_{max}	maximum service demand	$R_{m,c}$	response time for Q_m at class c
δ	maximum allowable error	S_m	service time for Q_m
$E_{m,c}$	expected queue length of Q_m at class c	τ	throughput
L_m	queue length of Q_m	τ_c	throughput at class c
$L_{m,c}$	queue length of Q_m at class c	V_m	visit number for Q_m
M	number of tiers	Z	user think time

It is worth decomposing a web-based software system to multiple tiers, and modeling it according to Fig. 1, because the service time of each tier is independent and additive, and can be very different.

Subthesis I.1

I have shown that the base queueing model (Definition 3.1) and the MVA evaluation algorithm (Definition 3.2), the approximate MVA (Algorithm 3.2), the balanced job bounds (Definition 3.3) can be used to model multi-tier ASP.NET web-based software systems. I have demonstrated that these model and algorithms can be applied to performance prediction of web-based software systems in ASP.NET environment, they can predict the response time, throughput and tier utilization performance metrics. I have validated these model and algorithms and verified the correctness of the performance prediction with performance measurements.

I have analyzed the computational complexity of the MVA algorithm as well as the limit of the response time and throughput sequences computed by the base queueing model and the MVA algorithm in order to provide a comparative base for proposed models and algorithms.

If the Steps 5, 6, 7, and 9 of the recursive MVA (Algorithm 3.2) are substituted into each other, the following equations can be derived for response time:

$$R(n) = \sum_{m=1}^{M} D_m + \frac{(n-1)\sum_{m=1}^{M} D_m R_m(n-1)}{Z + R(n-1)}, \tag{3.3}$$

for throughput:

$$\tau(n) = \frac{n}{Z + (D_1 + D_2 + ... + D_M) + \sum_{i=2}^{n} \left\{ (D_1^i + D_2^i + ... + D_M^i) \prod_{j=1}^{i-1} \tau(n-j) \right\}}. \tag{3.4}$$

Subthesis I.2

I have proven that the computational complexity of the MVA algorithm (Definition 3.2), which can evaluate the base queueing model (Definition 3.1), is $\Theta(N \cdot M)$, where N is the number of customers and M is the number of tiers. I have shown that the response time sequence (Equation 3.3) provided by the base queueing model and the MVA algorithm converges to infinity (if $n \to \infty$, where n is the number of customers). I have proven that the throughput sequence (Equation 3.4) computed by the base queueing model and the MVA algorithm converges to $1/D_{\max}$ (if $n \to \infty$, n is the number of customers), where $D_{\max}$ is the maximum value of service demands.

After analyzing the present performance models and evaluation algorithms, I have identified and investigated novel performance factors to improve performance models. I have proposed statistical methods regarding the methodology of examinations.

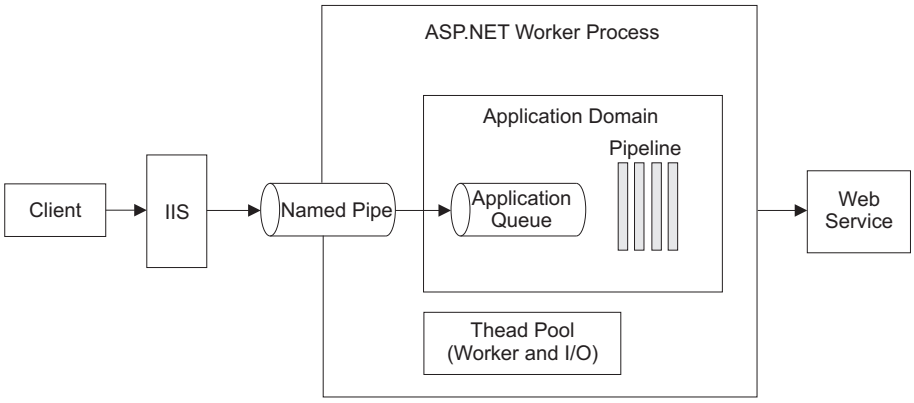

Figure 2: The architecture of ASP.NET environment

Using a thread pool, when a request arrives, the application adds it to an incoming queue [Carmona 2002] (Fig. 2). A group of threads retrieves requests from this queue and processes them. As each thread is freed, another request is executed from the queue.

From the IIS, the accepted HTTP connections are placed into a named pipe. This is a global queue between IIS and ASP.NET, where the requests are posted from native code to the managed thread pool. The global queue is managed by the process that runs ASP.NET, its limit is set by the *requestQueueLimit* property. When the "Requests Current" counter – which includes the requests that are queued, being executed, or waiting to be written to the client – reaches this limit, the requests are rejected [Marquardt 2003].

From the named pipe, the requests are placed into an application queue, also known as a virtual directory queue. Each virtual directory has a queue that is used to maintain the availability of worker and I/O threads. The number of requests in these queues increases if the number of available worker and I/O threads falls below the limit specified by *minFreeThreads* property. The application queue limit is configured by the *appRequestQueueLimit* property. When the limit is exceeded, the requests are rejected.

The *maxWorkerThreads* attribute means the maximum number of worker threads, the *maxIOThreads* parameter is the maximum number of I/O threads in the .NET thread pool. The *minFreeThreads* attribute limits the number of concurrent requests, because all incoming requests will be queued if the number of available threads in the thread pool falls below the value for this setting. The *minLocalRequestFreeThreads* parameter is similar to *minFreeThreads*, but it is related to the requests from the local host. These two attributes can be used to prevent deadlocks by ensuring that a thread is available to handle callbacks from pending asynchronous requests.

Subthesis I.3

I have shown that the chi square test of independence can be applied to performance factor identification. I have proven that the thread pool attributes: maxWorkerThreads, maxIOThreads, minFreeThreads, minLocalRequestFreeThreads as well as the queue limit parameters: requestQueueLimit, appRequestQueueLimit are performance factors. I have shown with statistical methods that the response time performance metric tends to a normal distribution if the probability variables are the thread pool performance factors (namely, maxWorkerThreads, maxIOThreads, minFreeThreads and minLocalRequestFreeThreads). I have determined the parameters of the distribution by maximum likelihood estimation with successive approximation.

The identified performance factors (thread pool and queue limit) must be modeled to improve performance models of web-based software systems.

Thesis II
Modeling the Thread Pool

I have proposed a novel algorithm to model the thread pool performance factor. I have analyzed the computational complexity of the proposed algorithm as well as the limit of the response time and throughput sequences computed by the novel algorithm. I have demonstrated that the proposed algorithm can be applied to performance prediction of web-based systems in ASP.NET environment, the response time and throughput performance metrics can be predicted. I have validated the novel algorithm and verified the correctness of the performance prediction with performance measurements. I have introduced that the average absolute error of the novel algorithm is less than the average absolute error of the original algorithm. I have examined the error histograms of the original and proposed algorithms.

Thesis II is contained by Chapter 5 of the dissertation. Related publications: [4] [5] [6] [22] [23] [24] [25] [26] [27].

Due to the behavior of the thread pool, the MVA evaluation algorithm (Definition 3.2) has been improved.

Definition 3.4. *The extended MVA with thread pool (MVA-TP) is defined by Algorithm 3.3, where the index CPU is related to the CPU tier index and the index I/O corresponds to an I/O tier index from* $1 \leq m \leq M$.

Algorithm 3.3 Pseudo code of the MVA-TP

1: **for all** $m = 1$ to M **do**
2: $L_m = 0$
3: **for all** $n = 1$ to N **do**
4: **for all** $m = 1$ to M **do**
5: $R_m = V_m \cdot S_m \cdot (1 + L_m)$
6: $R = \sum_{m=1}^{M} R_m$
7: $\tau = n/(Z + R)$
8: **for all** $m = 1$ to M **do**
9: **if** *m is an I/O tier* **then**
10: $L_{I/O} = \tau \cdot R_{I/O} - \frac{S_{CPU}}{S_{I/O}} \cdot L_{CPU}$
11: **if** $L_{I/O} < 0$ **then**
12: $L_{I/O} = 0$
13: **for all** $m = 1$ to M **do**
14: **if** *if m is a CPU tier* **then**
15: $L_m = \tau \cdot R_m$

Subthesis II.1

I have proposed a novel algorithm (Definition 3.4) to model the thread pool. I have shown that the proposed MVA-TP can evaluate the base queueing model (Definition 3.1). I have proven that the computational complexity of the novel MVA-TP is $\Theta(N \cdot M)$, where N is the number of customers and M is the number of tiers.

This extension does not increase the complexity of the evaluation algorithm, because the computational complexity of the original algorithm is the same (see Subthesis I.2).

Subthesis II.2

I have proven that the predicted throughput sequence provided by the novel MVA-TP (Definition 3.4) converges to $1/D_{\max}$ (if $n \to \infty$, n is the number of customers), where $D_{\max}$ is the maximum value of service demands. I have shown that the predicted response time sequence converges to infinity (if $n \to \infty$, where n is the number of customers). I have proven that the difference between the throughput and response time sequences computed by the novel MVA-TP and the original MVA (Definition 3.2) converges to zero (if $n \to \infty$, where n is the number of customers).

The limits of the response time and the throughput sequences computed by the proposed MVA-TP are the same as the limits of the performance metrics computed by the original MVA (see Subthesis I.2).

Consider $\lim_{n\to\infty} a_n = A$, the definition of the limit is as follows: $\forall \varepsilon > 0$ ($\varepsilon \in R$) $\exists N(\varepsilon) \in N$ that $|a_n - A| < \varepsilon$ if $n > N(\varepsilon)$. According to the definition of the limit, a common threshold index can be found for each arbitrary chosen ε: $\max(N_1(\varepsilon), N_2(\varepsilon))$, where $N_1(\varepsilon)$ is for response time difference between the proposed and original algorithms and $N_2(\varepsilon)$ is for throughput difference. If the number of customers is greater than the threshold index, the proposed MVA-TP and the original MVA can be used, as well. If the number of customers is less than the threshold index, the novel MVA-TP has to be applied.

I have analyzed the error to verify the correctness of the performance prediction with the proposed MVA-TP. Two methods have been applied: the average absolute error function and the error histogram.

Definition 3.5. *The average absolute error function is defined by Equation 3.5, where the index alg corresponds to the values provided by the algorithm, the index obs is related to the observed values, R is the response time, and N is the number of measurements.*

$$error_{alg-obs} = (\sum_{i=1}^{N} |R_{alg_i} - R_{obs_i}|)/N. \tag{3.5}$$

Subthesis II.3

I have demonstrated that the proposed MVA-TP (Definition 3.4) can be applied to performance prediction of web-based systems in ASP.NET environment, this proposed algorithm predicts the performance metrics more accurate than the original MVA (Definition 3.2). I have validated the novel MVA-TP and verified the correctness of the performance prediction with the proposed MVA-TP with performance measurements. I have introduced that the average absolute error (Definition 3.5) of the novel MVA-TP is less than the average absolute error of the original MVA. I have examined the error histograms of the proposed MVA-TP and original MVA. The results have shown that the improved MVA-TP predicts performance metrics much more correctly than the original MVA.

Thesis III
Modeling the Queue Limit

I have provided novel models and algorithms to model the queue limit performance factor. I have examined the computational complexity of the proposed algorithms as well as the limit of the response time and throughput sequences provided by the novel models and algorithms. I have shown that the proposed models and algorithms can be used for performance prediction of web-based systems in ASP.NET environment, the response time and throughput performance metrics can be predicted. I have validated the novel models and algorithms and verified the correctness of the performance prediction with performance measurements. I have demonstrated that the average absolute error of the novel models and algorithms is less than the average absolute error of the original model and algorithm. I have analyzed the error histograms of the original and proposed models and algorithms.

Thesis III is contained by Chapter 6 of the dissertation. Related publications: [28] [29] [30] [31] [32].

By taking the queue limit into consideration, the base queueing model (Definition 3.1) and the MVA evaluation algorithm (Definition 3.2) can be effectively enhanced.

Definition 3.6. *The extended queueing model with global queue limit (QM-GQL) is defined by Fig. 3, where the Q_{drop} is an infinite server queueing system, the Z_{drop} is the time spent at Q_{drop}, the GQL is the global queue limit, which corresponds to the requestQueueLimit parameter in ASP.NET environment. If the GQL is less than the queued requests sum $\sum_{m=1}^{M} L_m$, the next requests proceed to Q_{drop}. Requests from Q_{drop} proceed back to Q_0, namely, these requests are reissued.*

Definition 3.7. *The extended MVA with global queue limit (MVA-GQL) is defined by Algorithm 3.4, where the Z_{drop} is the time spent at Q_{drop}, the GQL is the global queue limit, which corresponds to the requestQueueLimit parameter in ASP.NET environment.*

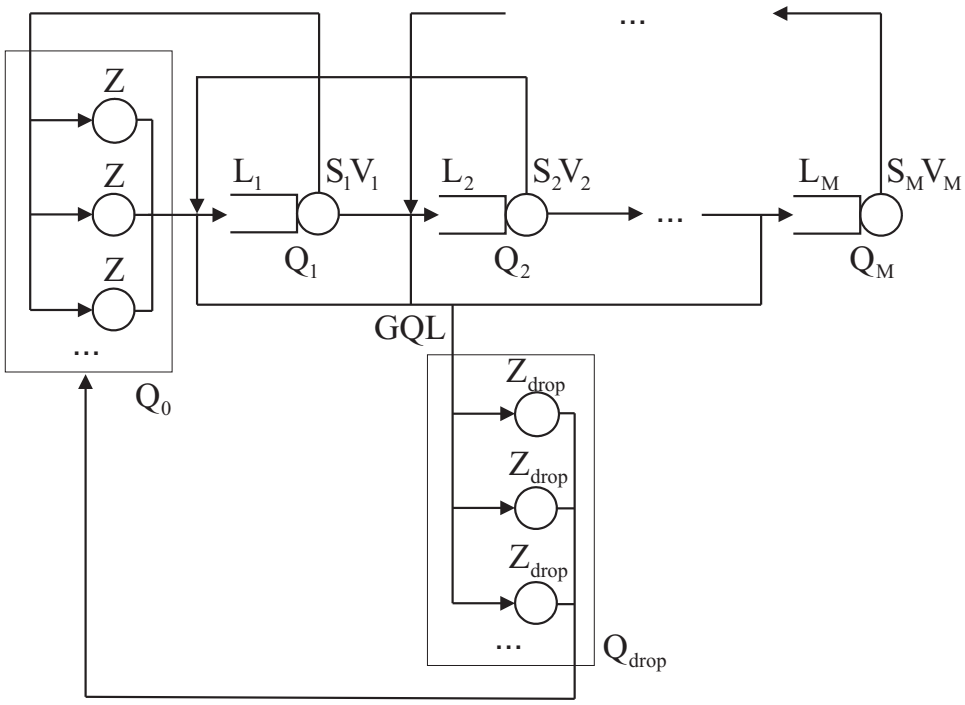

Figure 3: Extended queueing model with global queue limit

Algorithm 3.4 Pseudo code of the MVA-GQL

for all $m = 1$ to M **do**
  $L_m = 0$
$nql = 1$
for all $n = 1$ to N **do**
  for all $m = 1$ to M **do**
    $R_m = V_m \cdot S_m \cdot (1 + L_m)$
  $R = \sum_{m=1}^{M} R_m$
  $\tau = nql/(Z + Z_{drop} + R)$
  for all $m = 1$ to M **do**
    $L_m = \tau \cdot R_m$
  if $\sum_{m=1}^{M} L_m > GQL$ **then**
    for all $m = 1$ to M **do**
      $L_m = oldL_m$
  else
    $nql = nql + 1$
  for all $m = 1$ to M **do**
    $oldL_m = L_m$

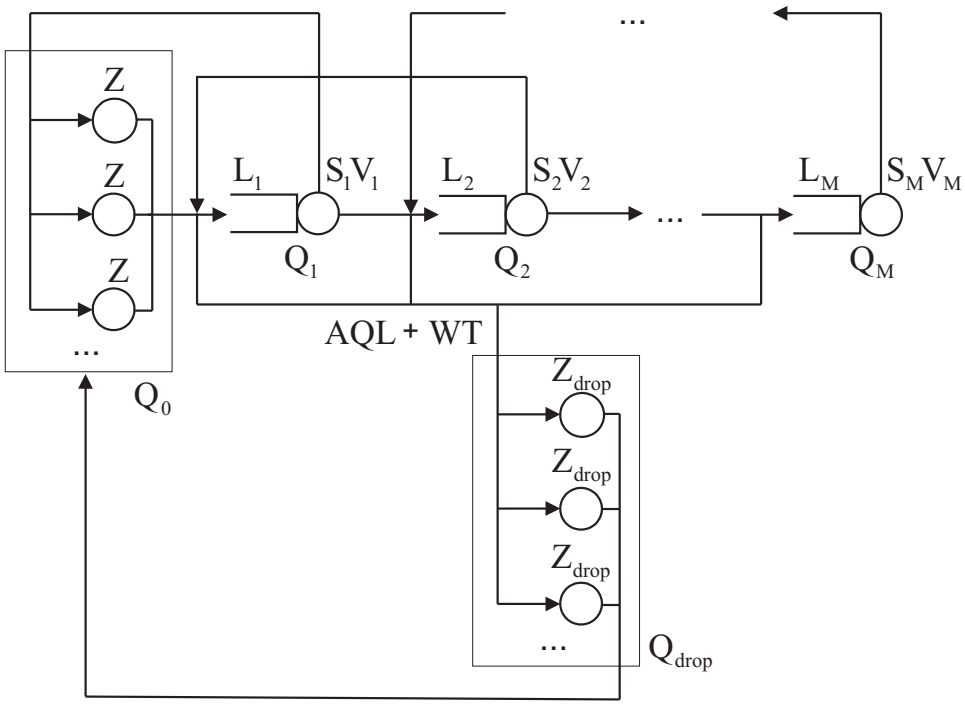

Figure 4: Extended queueing model with application queue limit

Algorithm 3.5 Pseudo code of the MVA-AQL

for all $m = 1$ to M **do**
  $L_m = 0$
$nql = 1$
for all $n = 1$ to N **do**
  for all $m = 1$ to M **do**
    $R_m = V_m \cdot S_m \cdot (1 + L_m)$
  $R = \sum_{m=1}^{M} R_m$
  $\tau = nql/(Z + Z_{drop} + R)$
  for all $m = 1$ to M **do**
    $L_m = \tau \cdot R_m$
  if $\sum_{m=1}^{M} L_m - WT > AQL$ **then**
    for all $m = 1$ to M **do**
      $L_m = oldL_m$
  else
    $nql = nql + 1$
  for all $m = 1$ to M **do**
    $oldL_m = L_m$

Definition 3.8. *The extended queueing model with application queue limit (QM-AQL) is defined by Fig. 4, where the Q_{drop} is an infinite server queueing system, the Z_{drop} is the time spent at Q_{drop}, the AQL is the application queue limit, which corresponds to the appRequestQueueLimit parameter in ASP.NET environment, in addition, the WT is the maximum number of working threads, which equals $maxWorkerThreads + maxIOThreads - minFreeThreads$ in ASP.NET environment. If the AQL+WT is less than the queued requests sum $\sum_{m=1}^{M} L_m$, the next requests proceed to Q_{drop}. Requests from Q_{drop} proceed back to Q_0, namely, these requests are reissued.*

Definition 3.9. *The extended MVA with application queue limit (MVA-AQL) is defined by Algorithm 3.5, where the Z_{drop} is the time spent at Q_{drop}, the AQL is the application queue limit, which corresponds to the appRequestQueueLimit parameter in ASP.NET environment, in addition, the WT is the maximum number of working threads, which equals $maxWorkerThreads + maxIOThreads - minFreeThreads$ in ASP.NET environment.*

Subthesis III.1

I have proposed novel models and algorithms (Definitions 3.6 and 3.7 as well as Definitions 3.8 and 3.9) to model the global and application queue limits. I have shown that the novel MVA-GQL can be applied as an approximation method to the proposed QM-GQL and the MVA-AQL can be used as an approximation method for the QM-AQL. I have proven that the computational complexity of the novel MVA-GQL and MVA-AQL is $\Theta(N \cdot M)$, where N is the number of customers and M is the number of tiers.

These extensions do not increase the complexity of the evaluation algorithm, because the computational complexity of the original algorithm is the same (see Subthesis I.2).

Subthesis III.2

I have shown that the predicted response time sequence provided by the novel models and algorithms (Definitions 3.6 and 3.7 as well as Definitions 3.8 and 3.9) converges to RQL (if $n \to \infty$, n is the number of customers), where the RQL constant value can be seen in Equation 3.6, and QL corresponds that index when $\sum_{m=1}^{M} L_m$ is greater than GQL or AQL+WT at the first time. I have proven that the predicted throughput sequence computed by the novel models and algorithms converges to $\min\{1/D_{\max}, \tau QL\}$ (if $n \to \infty$, n is the number of customers), where $D_{\max}$ is the maximum value of service demands, in addition, τQL constant value can be seen in Equation 3.7, and QL corresponds that index when $\sum_{m=1}^{M} L_m$ is greater than GQL or $AQL + WT$ at the first time.

$$RQL = \sum_{m=1}^{M} D_m + \frac{QL \sum_{m=1}^{M} D_m R_m(QL)}{Z + Z_{drop} + R(QL)} \tag{3.6}$$

$$\tau QL = \frac{QL}{Z + Z_{drop} + \sum_{m=1}^{M} D_m + \sum_{i=2}^{QL} \left(\sum_{m=1}^{M} D_m^i \prod_{j=1}^{i-1} \tau(QL - j) \right)} \tag{3.7}$$

Until all the requests have successfully been served, the response time and the throughput computed by the proposed models and algorithms are the same as the performance metrics computed by the original model and algorithm. When requests are rejected because of exceeding the queue limit, the throughput sequence computed by the proposed models and algorithms converges to a constant value, in most cases this is the same as in case of the original model and algorithm, but in some cases this is another constant value. Furthermore, the limit of the response time sequence provided by the novel models and algorithms is not infinity, it converges to a constant value.

Subthesis III.3

I have demonstrated that the proposed models and algorithms (Definitions 3.6 and 3.7 as well as Definitions 3.8 and 3.9) can be applied to performance prediction of web-based systems in ASP.NET environment, this proposed models and algorithms predicts the performance metrics more accurate than the original base queueing model and MVA (Definitions 3.1 and 3.2). I have validated the novel models and algorithms and verified the correctness of the performance prediction with the proposed models and algorithms with performance measurements. I have introduced that the average absolute error (Definition 3.5) of the novel models and algorithms is less than the average absolute error of the original base queueing model and MVA. I have examined the error histograms of the original and proposed models and algorithms. The results have shown that the novel models and algorithms predict performance metrics much more correctly than the original base queueing model and MVA.

The extended MVA with thread pool and the extended MVA with queue limit are independent and additive improvements, they can be applied together or separately, as well.

4 Application of the Novel Scientific Results

Possibilities of practical application are contained by Chapter 7 of the dissertation. Related publications: [28][29].

The results – performance prediction and performance factor identification, modeling the thread pool, modeling the queue limit – together form improved performance models of web-based software systems. I have successfully applied these techniques to industrial applications like Hungarian Microsoft Educational Portal [MsPortal]. Furthermore, I have developed a web-based software system to demonstrate the practical application of the proposed methods.

The results of this thesis have been realized by a software package, which contains the contributions of this thesis in modular structure. My thesis work includes flowcharts and case studies offering solutions to practical problems.

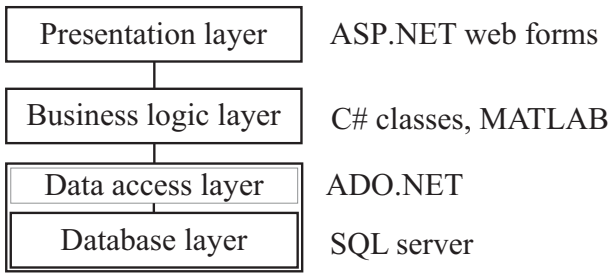

Figure 5: Architecture of the developed web-based software system

The architecture of the developed web-based software system is illustrated in Fig. 5. The proposed methods have been implemented in MATLAB. The MATLAB programs have been invoked and applied in .NET environment.

There are three main parts of the web-based software system as tab layout:

- *Measuring the performance.*

  I have given a methodology described as a process to provide performance measurements of web-based software systems. The whole process cannot be automated. The given steps should be followed to perform a measurement process. I have statistically analyzed the results of performance measurement processes in Thesis I. In addition, I have applied them to validate the present (Thesis I) and the proposed (Thesis II and III) performance models and evaluation algorithms as well as to verify the correctness of the performance prediction. For model parameter estimation and model evaluation, only one measurement or estimation in case of one customer is required.

- *Identifying performance factors.*

  I have provided the performance factor identifying process. I have realized here the proposed statistical methods of Thesis I.

- *Predicting performance metrics, validating performance models and evaluation algorithms, analyzing the error.*

  I have provided the following functions. The performance metrics can be predicted with the original along with the proposed models and algorithms modeling the thread pool and the queue limit. The novel models and algorithms can be validated using results of performance measurements. Error analysis can be performed to demonstrate the accuracy of the proposed models and algorithms. I have realized here the propositions of Thesis I, II, and III.

5 Scientific Publications

International Journals

1 **Á. Bogárdi-Mészöly**, G. Imre, and Charaf H., "Investigating Factors Influencing the Response Time in J2EE Web Applications", WSEAS Transactions on Computers, Vol. 4(2), pp. 179–183, February 2005, ISSN 1109-2750.

2 **Á. Bogárdi-Mészöly**, Z. Szitás, T. Levendovszky, and H. Charaf, "Investigating Factors Influencing the Response Time in ASP.NET Web Applications", Advances in Informatics, Lecture Notes in Computer Science, Vol. 3746, pp. 223–233, ISSN 0302-9743, IF: 0.402, November 2005.

3 **Á. Bogárdi-Mészöly**, T. Levendovszky, and Charaf H., "Models for Predicting the Performance of ASP.NET Web Applications", Periodica Polytechnica Electrical Engineering, Vol. 51(3-4), pp. 111–118, 2007.

4 **Á. Bogárdi-Mészöly**, T. Hashimoto, T. Levendovszky, and H. Charaf, "Thread Pool-Based Improvement of the Mean-Value Analysis Algorithm", Lecture Notes in Electrical Engineering, Vol. 27(2), pp. 1241–1254, ISBN 978-0-387-84813-6, February 2009.

5 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "An Improved Algorithm for Performance Prediction of Multi-Tier Information Systems", Performance Evaluation, Elsevier, under revision.

6 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Improved Performance Model for Web-Based Software Systems", WSEAS Transactions, under review.

Hungarian Journals

7 **Á. Bogárdi-Mészöly**, G. Imre, and T. Levendovszky, "ASP.NET és Java EE webalkalmazások teljesítménypredikciója sorbanállási modell segítségével", Magyar Távközlés, Vol. 2006/3.

Conference Proceedings

8 **Á. Bogárdi-Mészöly** and H. Charaf, "Comparison of Portal Building Techniques", microCAD 2004 International Scientific Conference, Miskolc, Hungary, March 18-19, 2004, pp. 41–46.

9 **Á. Bogárdi-Mészöly**, G. Imre, T. Levendovszky, and H. Charaf, "Determining the Distribution of the Response Time in J2EE Web Applications", microCAD 2005 International Scientific Conference, Miskolc, Hungary, March 10-11, 2005, pp. 33–38.

10 **Á. Bogárdi-Mészöly**, G. Imre, T. Levendovszky, and H. Charaf, "Investigating the Response Time in J2EE Web Applications", 5th International Conference of PhD Students, Miskolc, Hungary, August 14-20, 2005, pp. 19–24.

11 G. Imre, **Á. Bogárdi-Mészöly**, and H. Charaf, "Monitoring the Performance of J2EE Web Applications", 5th International Conference of PhD Students, Miskolc, Hungary, August 14-20, 2005, pp. 13–18.

12 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Extending the Performance Models of Web Applications with Queueing Algorithm", 6th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, November 18-19, 2005, pp. 719–730.

13 **Á. Bogárdi-Mészöly**, Z. Szitás, T. Levendovszky, and H. Charaf, "Balanced Job Bounds Calculation for Approximating ASP.NET Performance Factors", 4th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence (SAMI 2006), Herl'any, Slovakia, January 20-21, 2006, pp. 152–163.

14 G. Imre, **Á. Bogárdi-Mészöly**, and H. Charaf, "Measuring and Modelling the Effect of Application Server Tuning Parameters on Performance", 4th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence (SAMI 2006), Herl'any, Slovakia, January 20-21, 2006, pp. 471–482.

15 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Using Queueing Model in Predicting the Response Time of ASP.NET Web Applications", IASTED International Conference on Software Engineering, Innsbruck, Austria, February 14-16, 2006, ACTA Press, IEEE Catalog Number 07EX1642C, ISBN 1-4244-0865-2, pp. 252–257.

16 **Á. Bogárdi-Mészöly**, G. Imre, T. Levendovszky, and H. Charaf, "Handling Multiple Session Classes in ASP.NET Environment", microCAD 2006 International Scientific Conference, Miskolc, Hungary, March 16-17, 2006, pp. 31–36.

17 G. Imre, **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Performance Modelling of a J2EE Web Application Considering Application Server Tuning Parameters", microCAD 2006 International Scientific Conference, Miskolc, Hungary, March 16-17, 2006, pp. 115–121.

18 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Predicting the Performance of ASP.NET Web-Based Information Systems with Optimized Algorithms", 9th International Conference Information Systems Implementation and Modelling (ISIM 2006), Přerov, Czech Republic, April 24-27, 2006, pp. 167–174.

19 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Handling Session Classes for Predicting ASP.NET Performance Metrics", 4th International Conference in Central Europe on .NET Technologies, Plzen, Czech Republic, May 29 - June 1, 2006, ISBN 80-86943-11-9, pp. 1–8.

20 **Á. Bogárdi-Mészöly**, "Analytical Model for Multi-tier ASP.NET Web Applications", Automation and Applied Computer Science Workshop (AACS 2006), Budapest, Hungary, June 30, 2006, pp. 109–120.

21 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Performance Factors in ASP.NET Web Applications with Limited Queue Models", 10th IEEE International Conference on Intelligent Engineering Systems (INES 2006), London, United Kingdom, June 26-28, 2006, IEEE Catalog Number 06EX1430, ISBN 1-4244-9708-8, pp. 253–257.

22 **Á. Bogárdi-Mészöly**, "An Enhanced Performance Evaluation Algorithm for Enterprise Architectures", Automation and Applied Computer Science Workshop (AACS 2007), Budapest, Hungary, June 29, 2007, pp. 81–92.

23 **Á. Bogárdi-Mészöly**, T. Hashimoto, T. Levendovszky, and H. Charaf, "Improved Evaluation Algorithm for Performance Prediction with Error Analysis", 11th IEEE International Conference on Intelligent Engineering Systems (INES 2007), Budapest, Hungary, June 29 - July 1, 2007, IEEE Catalog Number 07EX1751C, ISBN 1-4244-1148-3, pp. 301–306.

24 **Á. Bogárdi-Mészöly**, T. Levendovszky, and H. Charaf, "Extending the Mean-Value Analysis Algorithm According to the Thread Pool Investigation", 5th IEEE International Conference on Industrial Informatics (INDIN 2007), Vienna, Austria, July 23-27, 2007, IEEE Catalog Number 07EX1642C, ISBN 1-4244-0865-2, ISSN 1935-4576, pp. 731–736.

25 **Á. Bogárdi-Mészöly**, T. Levendovszky, H. Charaf, and Á. Szeghegyi, "Effect Analysis of an Improved Performance Evaluation Algorithm", 6th International Symposium on Applied Machine Intelligence and Informatics (SAMI 2008), Herl'any, Slovakia, January 21-22, 2008, IEEE Catalog Number CFP0808E-CDR, ISBN 978-1-4244-2106-0, pp. 125–130.

26 **Á. Bogárdi-Mészöly**, T. Levendovszky, and Á. Szeghegyi, "Analyzing the Convergence of an Enhanced Performance Evaluation Algorithm", 12th IEEE International Conference on Intelligent Engineering Systems (INES 2008), Miami, Florida, USA, February 25-29, 2008, IEEE Catalog Number CFP08IES-CDR, ISBN 978-1-4244-2083-4, pp. 185–190.

27 **Á. Bogárdi-Mészöly**, T. Levendovszky, H. Charaf, and Á. Szeghegyi, "Convergence and Limit of Mean-Value Analysis Algorithms", 12th WSEAS International Conference on Computers (CSCC 2008), New Aspects of Computers, Heraklion, Greece, July 23-25, 2008, ISBN 978-960-6766-85-5, ISSN 1790-5109, pp. 601–606.

28 **Á. Bogárdi-Mészöly** and T. Levendovszky, "Application of Improved Performance Models", 9th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, Budapest, Hungary, November 6-8, 2008, pp. 205–216.

29 **Á. Bogárdi-Mészöly**, T. Levendovszky, and A. Rövid, "Improved Performance Models and Algorithms", 6th IEEE International Conference on Computational Cybernetics, Stará Lesná (ICCC 2008), Slovakia, November 27-29, 2008, IEEE Catalog Number CFP08575-CDR, ISBN 978-1-4244-2875-5, pp. 157–162.

30 **Á. Bogárdi-Mészöly**, T. Levendovszky, and A. Rövid, "A Novel Algorithm to Model the Queue Limit", 7th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing, Puerto De La Cruz, Canary Islands, Spain, December 15-17, 2008, ISBN 978-960-474-037-6, pp. 81–86.

31 **Á. Bogárdi-Mészöly**, T. Levendovszky, and Á. Szeghegyi, "Improved Performance Models of Web-Based Software Systems", 13th IEEE International Conference on Intelligent Engineering Systems, Barbados, April 16-18, 2009, IEEE Catalog Number CFP09IES-CDR, ISBN 978-1-4244-4113-6, pp. 33–38.

32 **Á. Bogárdi-Mészöly**, T. Levendovszky, and Á. Szeghegyi, "Performance Prediction with Algorithm Modeling the Queue Limit", 13th World Multi-Conference on Systemics, Cybernetics and Informatics), Orlando, Florida, USA, July 10-13, 2009, to be published.

Technical Report

33 G. Imre, and **Á. Bogárdi-Mészöly**, "J2EE Enterprise rendszerek vizsgálata", Research development report, T-Mobile Magyarország Távközlési Rt., November 2004.

34 **Á. Bogárdi-Mészöly**, "ASP.NET webalkalmazások teljesítménypredikciója", Research report, evosoft Hungary Kft., October 2006.

6 Citations

[15] is cited by:

S. Gorappa, "Performance Prediction of Component- and Pattern-Based Middleware for Distributed Systems", Proceedings of the 4th on Middleware Doctoral Symposium, Newport Beach, California, Article No. 14, November 26-30, 2007.

[23] is cited by:

M. Woodside, "The Relationship of Performance Models to Data", Performance Evaluation: Metrics, Models and Benchmarks, Lecture Notes in Computer Science, Vol. 5119, pp. 9–28, Springer Berlin/Heidelberg, ISBN 978-3-540-69813-5, 2008.

7 References

[Bernardi 2002] S. Bernardi, S. Donatelli, and J. Merseguer, "From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models", ACM International Workshop Software and Performance, Rome, Italy, pp. 35–45, 2002.

[Bernardo 1998] M. Bernardo and R. Gorrieri, "A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time", Theoretical Computer Science, Vol. 202, pp. 11–54, 1998.

[Bobbio 2004] A. Bobbio, A. Horváth, and M. Telek, "The Scale Factor: a New Degree of Freedom in Phase-Type Approximation", Elsevier, Performance Evaluation, Vol. 56, pp. 121-144, March 2004.

[Carmona 2002] D. Carmona, "Programming the Thread Pool in the .NET Framework", http://msdn.microsoft.com/en-us/library/ms973903.aspx, .NET Development (General) Technical Articles, Microsoft Spain, 2002.

[Ferrari 1972] D. Ferrari, "Workload Characterization and Selection in Computer Performance Measurement", Computer, Vol. 5(7), pp. 18–24, 1972.

[Gilmore 1994] S. Gilmore and J. Hillston, "The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling", 7th International Conference Modelling Techniques and Tools for Performance Evaluation, pp. 353–368, 1994.

[Herzog 2000] U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle, "Compositional Performance Modelling with the TIPPtool", Performance Evaluation, Vol. 39, pp. 5–35, 2000.

[Jain 1991] R. Jain, "The Art of Computer Systems Performance Analysis", John Wiley and Sons, 1991.

[Kijima 1997] M. Kijima, "Markov Processes for Stochastic Modeling", Chapman & Hall/CRC, 1997.

[Kleinrock 1975] L. Kleinrock, "Theory, Volume 1, Queueing Systems", John Wiley and Sons, 1975.

[Marquardt 2003] T. Marquardt, "ASP.NET Performance Monitoring, and When to Alert Administrators", http://msdn.microsoft.com/en-us/library/ms972959.aspx, ASP.NET Technical Articles, 2003.

[Meier 2004] J.D. Meier, S. Vasireddy, A. Babbar, and A. Mackman, "Improving .NET Application Performance and Scalability (Patters & Practices)", Microsoft Corporation, 2004.

[Menascé 2000] D.A. Menascé and V.A.F. Almeida, "Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning", Prentice Hall, 2000.

[Menascé 2001] D.A. Menascé and V.A.F. Almeida, "Capacity Planning for Web Services: Metrics, Models, and Methods", Prentice Hall PTR, 2001.

[MsPortal] Hungarian Microsoft Educational Portal: http://www.msportal.hu.

[Platt 2003] D.S. Platt, "Introducing Microsoft .NET, Third Edition", Microsoft Press, 2003.

[Reiser 1980] M. Reiser and S.S. Lavenberg, "Mean-Value Analysis of Closed Multichain Queuing Networks", Association for Computing Machinery, Vol. 27, pp. 313–322, 1980.

[Sinclair 2005] B. Sinclair, "Mean-Value Analysis, Computer Systems Performance Handout, 2005.

[Sopitkamol 2005] M. Sopitkamol and D.A. Menascé, "A Method for Evaluating the Impact of Software Configuration Parameters on E-commerce Sites", ACM 5th International Workshop on Software and Performance, Palma, Illes Balears, Spain, pp. 53–64, 2005.

[Trivedi 1982] K.S. Trivedi, "Probability and Statistics with Reliability, Queueing and Computer Science Applications", Prentice-Hall, 1982.

[Urgaonkar 2005] B. Urgaonkar, "Dynamic Resource Management in Internet Hosting Platforms", Ph.D. Dissertation, Massachusetts, 2005.

[Wienholt 2003] N. Wienholt, "Maximizing .NET Performance", Apress, 2003.

[Zahorjan 1982] J. Zahorjan, K.C. Sevcik, D.L. Eager, and B. Galler, "Balanced Job Bound Analysis of Queueing Networks", Communications of the ACM, Vol. 25(2), pp. 134–141, 1982.