

A Best Practice Guide to Resource Forecasting for the Apache Webserver

Günther A. Hoffmann*, Kishor S. Trivedi
Duke University, Durham, NC 27708, USA
Department of Electrical & Computer Engineering
[gunho|kst]@ee.duke.edu

Mirosław Malek
*Humboldt University Berlin, Germany
Computer Architecture and Communication Group
[gunho|malek]@informatik.hu-berlin.de

Abstract

Recently, measurement based studies of software systems proliferated, reflecting an increasingly empirical focus on system availability, reliability, aging and fault tolerance. However, it is a non-trivial, error-prone, arduous, and time-consuming task even for experienced system administrators and statistical analysts to know what a reasonable set of steps should include to model and successfully predict performance variables or system failures of a complex software system. Reported results are fragmented and focus on applying statistical regression techniques to captured numerical system data. In this paper, we propose a best practice guide for building empirical models based on our experience with forecasting Apache web server performance variables and forecasting call availability of a real world telecommunication system. To substantiate the presented guide and to demonstrate our approach step-by-step we model and predict the response time and the amount of free physical memory of an Apache web server system. Additionally, we present concrete results for a) variable selection where we cross benchmark three procedures, b) empirical model building where we cross benchmark four techniques and c) sensitivity analysis. This best practice guide intends to assist in configuring modeling approaches systematically for best estimation and prediction results.

1 Introduction

Over the past decades software systems have grown in complexity up to the point where their behavior is in parts unpredictable. Testing, rigorous analysis and fault injection can become too rigorous and thus impractical to scale up to enterprise systems ([12],[32]). Also system patches and updates can transform a tested system into a new system. Additionally unpredictable industrial environments make it difficult if not impossible to apply the rigor we find in traditional approaches [1]. This has fostered an empirical perspective of looking at complex software systems which is reflected in numerous *measurement based* studies.

However, modeling procedures detailed in these studies mainly focus on the statistical regression part. Issues in a) the type of data to use, b) the type of modeling technique to use, c) the type of parameter optimization technique to apply and d) the type of variable selection procedure to employ are frequently treated with low priority. Piecing the abundance of procedures in these respective areas together sometimes seems more like an art than sci-

ence. However, we believe there is much to be gained by integrating these disparate aspects into one coherent approach. This best practise guide is based on our experience we have gained when investigating these topics:

- (a) complexity reduction, showing that selecting the most predictive variables contributes more to model quality than selecting a particular linear or nonlinear modeling technique ([24], [26]),
- (b) information gain of using numerical vs. categorical data ([24]), finding that including log file data into the modeling process counter intuitively degrades model quality for prediction of call availability of a telecommunication system,
- (c) data-based empirical modeling of complex software systems ([25]), cross benchmarking five linear and nonlinear modeling techniques, finding nonlinear approaches to be consistently superior than linear approaches, however, not always significantly.

In this paper, we integrate these steps into one coherent approach. To help guide the practitioner in reaching a feasible modeling and forecasting solution (vs. finding the optimal solution) we summarize our experience gained so far in a *best practice guide* and substantiate these steps in two examples. Firstly, we model and forecast the *response time* of an Apache web server in the short (5 minutes ahead) and long term (48 hours ahead). Secondly, we model and forecast the amount of *free physical memory* of the Apache web server in the short term.

The paper is organized as follows: In Section 2 we review related work in empirical modeling and variable selection. In Section 3 we introduce a basic terminology. In Section 4 we introduce our proposed best practice guide, which we put to the test in Section 5. Before we briefly discuss our findings in Section 7 we present the results from the empirical part of the paper in Section 6.

2 Related Work

The primary goal of empirical studies of complex software systems is to gain better understanding of the way the system works. Based on observations about its past behavior statistical models are derived to capture these dynamics and to make forecasts about its future behavior. This approach is in fact similar to long standing approaches in financial engineering, gene expression analysis or physics ([50]). The methods include in some form or the other (see Figure 1)

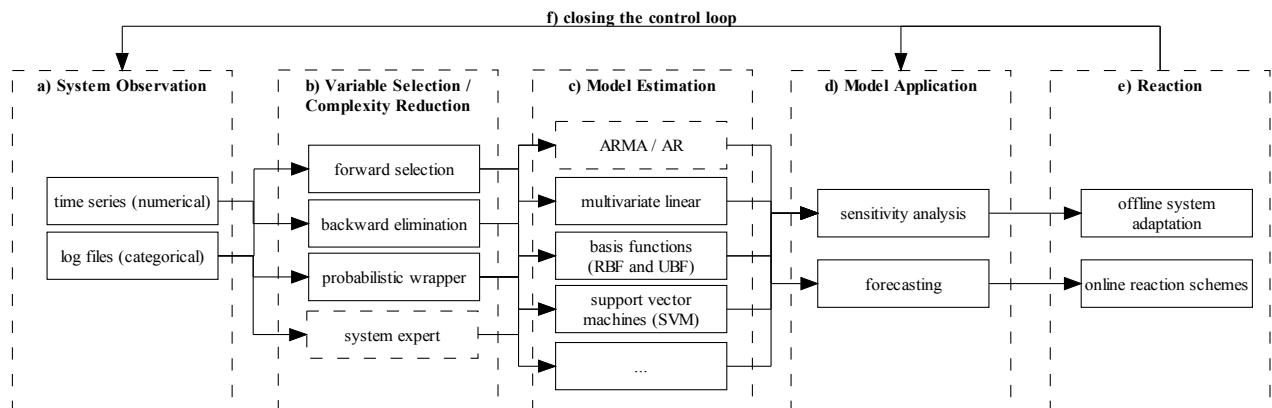


Figure 1: Building blocks for modeling and forecasting performance variables and critical events in complex software systems. System observations (a) could include numerical time series data and/or categorical log files. The variable selection process is typically handled implicitly by system expert's ad hoc theories or gut feeling, rigorous procedures are applied infrequently (b). In recent studies a lot of attention has been given to the model estimation process (c). Univariate and multivariate linear regression techniques have been at the center of attention. Some nonlinear regression techniques such as universal basis functions or support vector machines have been applied as well. While forecasting has received a substantial amount of attention, sensitivity analysis of system models has been largely marginalised (d). Closing the control loop (f) is still in its infancy. Choosing the right reaction scheme (e) as a function of quality of service and cost is nontrivial.

12th IEEE International Symposium Pacific Rim Dependable Computing (PRDC'06), University of California, Riverside, USA, Dec. 18-20, 2006

- *gathering data* describing the timely evolution of the system, this can include *numerical time series data* or *categorical log files* (Figure 1a)
- *estimating statistical models* based on these observations, this is sometimes synonymously called *regression* or *function approximation* (Figure 1c)
- *forecasting future events* or realizations of specific variables, this includes performance variables (e.g. memory or response times) or significant events (e.g. failures) (Figure 1d)
- *closing the control loop* by implementing some reaction scheme to help self-manage the system (e.g. failover, load lowering, rejuvenation and others) (Figure 1f)

The idea is, that once a model is in place, it can predict future events in advance which would allow for the triggering of preventive measures. Three aspects that are important in the overall concept have been largely ignored in to date studies, these are

- *complexity reduction*, synonymously also called *variable selection* and (Figure 1b)
- sensitivity analysis (Figure 1d)
- choice of reaction schemes (Figure 1e)

We revisit these building blocks briefly in the following sections.

2.1 Modeling Techniques

A number of modeling techniques have been applied to software systems. Including analytical models, statistical approaches, expert systems and fractal modeling. In the following sections we will briefly review some of these techniques and will also review some of their applications.

Analytical

[19] present an analytical model of a software system which serves transactions. Due to aging, the service rate of the system in question decreases over time and the software itself experiences hangs and crashes which re-

sults in unavailability. A number of approaches are built on Markov processes to compute models of the system in question. [27] present a continuous time Markov chain model for a long running server-client type telecommunication system. They express downtime and the cost induced by downtime in terms of the models parameters. [13] and [14] relax the assumptions made in [27] of exponentially distributed time independent transition rates (sojourn time) and build a semi-Markov model. This way they find a closed form expression for the optimal rejuvenation time. [17] and [5] develop stochastic Petri Nets by allowing the rejuvenation trigger to start in a robust state. This way they can calculate the rejuvenation trigger numerically. Another frequently cited approach is to build fault trees. This method implies that a complete enumeration of all possible system errors has to be conducted ([47]). If an error occurs it can be associated with its root cause. The advantage of this method is that an error can be traced back to its root cause. However, in large real systems this method is impractical due to prohibitive complexity. One of the major challenges with analytical approaches is that they do not track changes in system dynamics. Rather the patterns of changes in the system dynamics are left open to be interpreted by the operator. An improvement on this type of fault detection is to take changes into account and use these residuals to infer faulty components as presented in [15] and [38]. However, one fundamental problem with analytical approaches remains which is that they become quickly impractical when confronted with the degrees of freedom of software systems in use.

Heuristic and Linear Approaches

To get a grip on the complexity of real systems, heuristics and data-driven modeling approaches have been proposed by a number of authors. [2] propose the dispersion frame technique to distinguish between transient and intermittent errors and predict the occurrence of intermittent errors. However, the applied traditional statistical

methods are still too rigid to fit distributions where the data was not identically distributed. Other authors suggest heuristics to rejuvenate software components regularly at certain times ([46]). This time-triggered approach basically reflects the assumption of a linear model describing software aging and proposes that regular intervention will increase the system's availability. This method can yield increased system availability of systems with static characteristics. However, it is difficult to employ this approach with systems changing their characteristics more dynamically (see [28]).

Literature on applying linear modeling techniques to software systems has been dominated by approaches based on a single or a limited number of variables. Most models are either based on observations of workload, time, memory or file tables. [18] propose a time-based model for detection of software aging. [48] propose a workload-based model for prediction of resource exhaustion in operating systems such as Unix. [31] collect data from a web server which they expose to an artificial workload. They build time series ARMA (autoregressive moving average) models to detect aging and estimate resource exhaustion times. [9] propose their Pinpoint system to monitor network traffic. They include a cluster-based analysis technique to correlate failures with components and determine which component is most likely to be in a faulty state.

The explicit modeling of an adequate point in time for rejuvenation, which would optimize system availability and at the same time conserve system resources, would allow adapting to changing system dynamics ([13]). This approach would further allow an event-driven initialization of preventive measures. To achieve this goal data-driven methods have been applied to model complex computer systems. [31] propose linear autoregressive moving average (ARMA) models to predict resource exhaustion times in the Apache web server running in a Linux environment based on measured time series of system variables. They plan to use the prediction of resource exhaustion to time-optimize the rejuvenation of software components. They compare their approach to a number of linear regression models described in [8]. The authors indicate that they found nonlinear data relationships and therefore suggest the use of nonlinear modeling techniques. However, they give no further information on error rates to make their approach comparable.

Expert Systems

Another modeling approach is taken by ([51], [52]). In the *Timeweaver* project the author predicts failures in telecommunication equipment. The author identifies log files as a source of system state description. In his case study he uses genetic algorithms to evolve rules based on log files which have been collected during the runtime of a telecommunication system. The rules found, showed clear predictive capabilities. Another noteworthy result is presented by [41]. The authors analyze log files as a po-

tential resource for system state description and calculate the entropy of log files.

Fractal Modeling

Chaotic behavior in data selected from servers used in regular office environments such as file and print servers is reported by [43]. The authors collected a set of data consisting of two variables: *unused memory pages* and *free swap space*. The authors calculate the Hölder exponent ([37]) of the data, which is an important mathematical tool for studying chaotic behavior in other disciplines such as medical sciences, signal processing and economics. They report multi-fractal behavior in both time series and higher self similarity for higher workloads. The fractal approach is further developed by [11] and is turned into a predictive system in [44]. The authors report an algorithm with which they successfully predicted upcoming system crashes in 80% of their experiments. However, they do not give other metrics such as residual errors, area under curve (AUC), precision, recall or F-Measure, to make their approach comparable.

Nonlinear Probabilistic Modeling

Another approach to statistical analysis is nonlinear, non-parametric data analysis ([40], [4]). This modeling technique addresses some of the shortcomings of classical analytical techniques, in particular:

- the need to model nonlinear dependencies among system variables,
- to cope with incomplete and noisy data,
- to handle changing dynamics
- to cope with inconclusive data
- the need to retrofit a system

Especially the need to retrofit a system (i.e. take the data available at runtime without further interfacing the system) has become an important aspect given that most legacy systems are difficult to interface after they are deployed. Most large software systems have some form of existing fault detection during runtime. This is typically implemented in the form of limit checking of some parameters observed in real time which are compared against some preset value. If this predetermined health corridor is left the system will give an alarm. The installed sensors have typically been implemented during the development of the source code and tend to be difficult to change later on, not to mention the implementation of new sensors if the system is in production state. In some software systems counters are special implementations of sensors and are used synonymously in this context. Additionally, the implementation of sensors in real systems can become prohibitively expensive in terms of system load which further complicates their implementation, i.e. data must be stored for online or offline retrieval. The load generated by monitoring, storage and retrieval operations can easily grow prohibitively large.

Probabilistic models predicting resource demand of an algorithm based on observations of its historic behavior are developed in [23]. A probabilistic analysis technique has been developed by [16]. The author introduces

Stochastic Time Demand Analysis for determining a lower bound on the rate at which deadlines are met in fixed-priority systems. An example of a probabilistic discrimination of transient and permanent faults is given by [39]. The authors present a procedure based on applying Bayesian inference to observed events to diagnose a system at run time. They report to have derived an optimal procedure for discrimination of transient and permanent faults. [29] report the application of a number of nonlinear learning algorithms to model and predict run-specific resource usage as a function of input parameters supplied to the models at run time. The author applies his models to optimize resource management by estimating resource requirements before a scheduling decision is made. [48] propose a strategy to estimate the rate of exhaustion of operating system resources as a function of time and system workload. They construct a semi-Markov reward model. [49] predict response times of an M/G/1 processor-sharing queue. A similar approach is considered by [36]. The authors propose using nonlinear regression techniques to predict the optimal point in time for rejuvenation. Data obtained from a software system by measurement is subjected to statistical analysis by [6]. The authors model computation times and extend their approach to provide probabilistic scheduling in [7]. Statistical analysis of noncritical computing systems has been carried out by [45]. The authors analyze run-time traces and conclude that they can automatically extract temporal properties of applications by tracing a reasonably small set of system events. Prediction-based software availability enhancement is proposed in [35]. The authors present modeling approaches based on nonlinear regression techniques and Markov chains to predict upcoming failures in a software system. In fact model quality was shown to be more sensitive to the right variable mix than to a particular linear or nonlinear regression technique.

2.2 Variable Selection

Given the complexity of industrial software systems the number of observable variables can easily reach an order of magnitude which makes it difficult if not impossible to evaluate the influence of each variable on the predictive quality of the model due to combinatorial explosion. In fact including unfavorable variable combinations can degrade model performance significantly ([3], [20], [22]). This problem is known under a variety of names such as *variable selection*, *dimension reduction* or *feature detection* which we will use synonymously. The problem is finding the smallest subset of input variables which are necessary or suffice to perform a modeling task. This type of problem is one of the most prevalent topics in the machine learning and pattern recognition community, and has been addressed by a number of authors such as [34] and more recently by [21].

In the context of software systems, we have previously applied variable selection techniques to optimize failure prediction in a telecommunication system ([25]) and

to identify the most predictive variables for resource prediction in an Apache web server ([26]).

2.3 Sensitivity Analysis

One important application of an empirical model beyond prediction is sensitivity analysis with respect to the model's input variables. By investigating each variables contribution to the overall quality of the model and by evaluating potentially nonlinear relationships between a variable and the target (e.g. response time or system failure) variable constellations may be identified that yield potential insights into the root cause of a failure or misbehaving of a performance variable. For example, in earlier work ([24]) we identified two distinct variables, the number of semaphores per second and the amount of memory that was allocated by the kernel, which only at a particular parametrization made the system's call availability drop below a threshold value. This provided system engineers with additional information to track down the root cause of this behavior.

3 Terminology

The modeling and forecasting tasks we look at are straightforward. We are given a set of observations $\mathbf{x} = \{x | x = \langle f_1, \dots, f_n \rangle\}$ for which we compute a function Cl that predicts from the observed features $\langle f_1, \dots, f_{n-1} \rangle$ the target variable f_n which is in our case either *response time* or *free physical memory*. Each element $f \in \mathbf{x}$ is a vector of features where we denote $\langle f_1, \dots, f_{n-1} \rangle$ as the input features and f_n as the target variable. Given a previously unseen observation matrix \mathbf{x}_{new} with an unknown target value f_n we obtain $f_n = Cl(\mathbf{x}_{new})$. We calculate the prediction error at time t_3 which is $t_1 + \Delta t_l + \Delta t_p$. The *prediction period* Δt_p occurs some time after the prediction is made and covers the time frame in which the prediction is valid. In our experiments we use $\Delta t_p = 5$ minutes and $\Delta t_l = 60$ minutes. The *lead time* Δt_l defines how far the prediction extends into the future. We predict the scalar value of the target variables at time $t + \Delta t_l$ in the long term with $\Delta t_l = 48$ hours and in the short term with $\Delta t_l = 5$ minutes (i.e. 1 step ahead).

$$\text{lead time} \quad \Delta t_l = t_2 - t_1 \quad (1)$$

$$\text{prediction period} \quad \Delta t_p = t_3 - t_2 \quad (2)$$

The *embedding dimension* Δt_e which is denoted as

$$\text{embedding dimension} \quad \Delta t_e = t_1 - t_0 \quad (3)$$

specifies how far the labeled observations $f \in \mathbf{x}$ extend into the past.

4 Proposed Best Practice Guide

To help guide the practitioner reaching a *feasible* solution we summarize our experience gained, when handling data of an industrial telecommunication system and

the Apache web server and cast them into the following *best practice guide*.

- 1 *Data Preprocessing I*: split your data set into three distinct data sets. Use the first data set to parametrize your model. Use the second data set to validate your model and to prevent over-fitting. Apply the model to the third data set, which has not been presented to the model before. If the residual errors vary significantly from data set to data set their may be a significant change in dynamics hidden in the data.
- 2 *Data Preprocessing II*: if the observed features are not commensurate normalize them to zero mean and unit variance. Experiment with a number of different data preprocessing techniques and embedding dimensions. Even though we found *time series data* the most predictive in earlier experiments ([24], [25]), this step may heavily depend on characteristics of the data at hand.
- 3 *Variable selection I*: rank the observed features by their correlation coefficient with the target, build correlation coefficients for
 - 3.1 time series data
 - 3.2 class labels, such as 'failure' and 'no failure'. Map the class labels to numeric values.
- 4 *Variable selection II*: Reduce the variable set with a technique for complexity reduction, e.g. forward selection, backward elimination ([21]), probabilistic wrapper (PWA) ([26])
- 5 *Variable selection III*: Filter the most important variables
 - 5.1 in conjunction with a linear classifier by forward selection.
 - 5.2 in conjunction with a linear classifier by backward elimination.
 - 5.3 by encouraging experts to contribute their knowledge.
 - 5.4 by using the PWA method particularly when suspecting non-obvious nonlinear interdependence of variables .
- 6 *Model building for classification*: if the distribution of the minority and majority class is unbalanced adjust it to be evenly distributed, e.g. by resampling
- 7 *Model building*: start with simple models, e.g. by using multivariate linear models.
- 8 *Model building*: When suspecting nonlinear dynamics, use the variables found in the previous step as input to nonlinear models, e.g. universal basis functions (UBF) or support vector machines (SVM).
- 9 Experiment with both data types if available, a) numerical time series and b) categorical log file data. We found time series data to be a better predictor than log file data ([24]). However, this may ultimately depend on the quality of the log file data available for the system in question ([2], [41]).
- 10 Calculate model confidence, e.g. by using bootstrapping techniques and verify statistical significance, e.g. by *t*-testing.

5 Experiment: Predicting Resources

To substantiate our proposed guide we use the data set introduced in [30] to model and predict two Apache performance variables: 1) *response time* and 2) *free physical memory*. The authors captured data from a server running Apache and a client connected via Ethernet local area network. The data set contains 102 variables measured continuously once every five minutes over a period of about 165 hours (i.e. 1979 measurements). We enhance that data by including the first differences which doubles the number of variables. Thus, we work with 204 variables in total.

5.1 Forecasting Response Time

Data Preprocessing

Response time is the interval from the time a client sends out the first byte of a request until the client receives the first byte of the reply. In Figure 2a we show a plot of the *response time*. Two characteristics attract our attention. First, there are significant spikes. Second, the response time increases over time to flatten out in a plateau. [30] conclude that the latter is caused by resource exhaustion. By looking at the time between spikes, see Table 1 for the complete list, we find that they occur approximately in multiples of 6 hours.

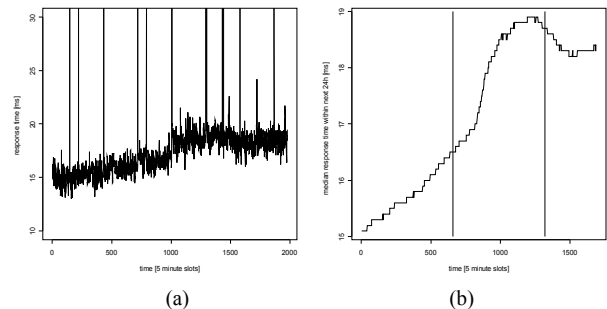


Figure 2: (a) Response time of Apache server with spikes. (b) To filter out spikes and to detect resource exhaustion more easily we calculate the median of the response time for the next 24 hours over a sliding window. Vertical solid lines segment the data into a training set (first segment), a validation set (second segment) and a generalization set (third segment).

# of 5 minute lag intervals	Minutes	~hours
73 and 74	369	6
150	750	13
213	1065	18
286	1430	24

Table 1: Lag times between spikes. The left column shows the number of time slots between spikes measured in 5 minute intervals. This translates into the minutes shown in the middle column. The approximate hours are shown in the right column. Spikes occur roughly at multiples of 6 hours.

However, that does not give us the root cause, but it indicates some areas for further research. Potential explanations include a) cron-jobs in additive 6 hour intervals, b) additive effects of update cycles of many machines affecting the network or c) broadcast messages. More theories concerning the root cause could be developed based

on this finding. However, from here on human experts will have to probe further into the system to find the root cause. Statistical analysis is a potential tool to narrow down the areas to look for a root cause. The increasing response time over time has been explained previously by resource exhaustion ([31]). The question remaining is, can we forecast the response time, such that we could close the control loop and trigger a preventive maintenance scheme based on this forecast. In the next step we cross benchmark three variable selection techniques to filter out the most predictive variables.

Variable Selection

To filter out the spike effect we set $\Delta t_p = 24$ hours. This means that instead of making point predictions we are rather looking at the median *response time* over the next 24 hours, see Figure 2b for a plot. The vertical solid lines split the data into three segments:

1. *training* or *parametrization* set which is used to parameterize the statistical models (first segment),
2. *validation* set which is used to validate the models and to avoid over-fitting (second segment) and
3. a *generalization* or *testing* set which is used to test a model on previously unseen data (third segment).

In this section we cross benchmark three procedures for variables selection:

- forward selection
- backward elimination and
- probabilistic wrapper (PWA)

These procedures have been introduced, described in detail and applied in a number of articles. For an introduction to forward selection and backward elimination see for example [21], for the PWA see [33] or [24],[26] for a more recent application. We investigate short term predictions with $\Delta t_i = 5$ minutes and long term predictions with $\Delta t_i = 2$ days. In Table 2 we report short term results for $\Delta t_p = 24$ hours and $\Delta t_i = 5$ minutes. We report the root-means-square-error (RMSE) for the *parametrization* and the *validation* data set in Table 2. Consistent with our earlier findings ([24],[26]) forward selection and PWA outperform backward elimination by an order of magnitude. PWA shows an RMSE of 0.025 on the validation data set while backward elimination shows an RMSE = 0.336.

	RMSE		
	forward selection	backward elimination	PWA
parametrization	0,012195	0,013896	0,014602
validation	0,046748	0,336004	0,024502

Table 2: Root-mean-square-errors (RMSE) for short term predictions ($\Delta t_p = 24$ hours and $\Delta t_i = 5$ min) of response time for three cross benchmarked variable selection procedures: forward selection, backward elimination and probabilistic wrapper approach (PWA).

Variable Nr.	Variable Name	Variable Nr.	Variable Name
30	NewProcesses	30	NewProcesses
23	DiskWrittenBlocks	81	eth0_recv_pkts
89	eth0_trans_pkts	29	ContextSwitches
65	lo_recv_pkts	18	TimeIdleMode
27	SwapOutCounter	72	lo_trans_bytes
63	si_slab_cache	26	SwapInCounter
59	si_size_256	59	si_size_256
29	ContextSwitches	62	si_size_32
6	CachedMemory	29	ContextSwitches

(a)

(b)

Table 3: The actual variables selected by the PWA procedure for short term (a) and long term (b) prediction of response time. Note that variable names in italic font indicate first differences rather than the absolute value of that variable.

As a result of the variable selection step we choose the variables identified by the PWA variable selection procedure which scored the smallest RMSE on the validation data set. These variables are shown in Table 3. The *number of new processes*, the *number of blocks written to the disk* and *number of Ethernet packets transported* were identified as the top three variables.

In the next step we apply the same three variable selection procedures to a long term prediction task with $\Delta t_p = 24$ h and $\Delta t_i = 2$ days. Results for this setting are shown in Table 4. Forward selection and PWA clearly outperform backward elimination on the parametrization as well as on the validation data set. PWA scored the smallest error with RMSE = 0.01. The resulting set of variables is shown in Table 11.

	RMSE		
	forward selection	backward elimination	PWA
parametrization	0,006096	0,018397	0,005312
validation	0,011146	0,716490	0,010160

Table 4: Root-mean-square-errors (RMSE) for long term predictions ($\Delta t_p = 24$ hours and $\Delta t_i = 2$ days) of response time for three cross benchmarked variable selection procedures. Compare to Table 3, forward selection and PWA clearly outperform backward elimination on the parametrization as well as on the validation data set.

It is interesting to note that the most predictive variables sets identified for short and long term prediction share an intersection of variables which are *NewProcesses*, *eth0_recv_pkts*, *ContextSwitches* and *si_size_256*. In addition to this base set of variables more counters and variables are added as needed by the PWA procedure.

Model Building

In the previous step we have identified the most predictive variables for long term (Table 3b) and short term (Table 3a) predictions of *response time*. We have identified these variables based on cross benchmarking three variable selection techniques (see Tables 2 and 4). We selected the variable set yielding the smallest RMSE on the validation data set. We now turn to model building and forecasting. We use the variable sets identified in the previous step to cross benchmark four modeling techniques for short and long term predictions. The modeling techniques are:

- multivariate linear regression (ML)
- support vector machines (SVM)
- radial basis functions (RBF)
- universal basis functions (UBF)

Again these techniques have previously been described in detail and have been applied to numerous applications. For an introduction to multivariate linear regression (ML) consult any basic text book on statistics, support vector machines (SVM) have been introduced by [10], kernel based nonlinear regression techniques such as SVM and radial basis functions (RBF) have recently been discussed in detail in [42]. Universal basis functions

(UBF) are an extension of RBF and we have previously discussed and applied them in the context of forecasting call availability of an industrial telecommunication system in [25].

In Table 5 we list RMSE values for *short term* prediction ($\Delta t_p = 24$ h, $\Delta t_i = 5$ min) of *response time* for the three data segments parametrization, validation and testing. In Table 6 we list RMSE values for *long term* prediction ($\Delta t_p = 24$ h, $\Delta t_i = 2$ days).

SVM outperform all other modeling techniques for *short term* (Table 5) as well as for *long term* predictions (Table 6). For short term predictions SVM produce a RMSE of 0.012 and 0.007 for long term predictions on the validation data set. However, this performance does not translate into an equally low forecast error on the testing data set. In short term SVM are outperformed on the *testing* (i.e. *generalization*) data set by UBF and by UBF and RBF in the long term. However, in reality at the time of model building we only have the parametrization and the validation data available. The multivariate linear approach is consistently outperformed by the nonlinear contenders SVM, RBF and UBF. Based on error results from the *validation set* we choose the support vector approach (SVM) for forecasting.

	ML	SVM	UBF	RBF
parametrization	0,012540	0,006885	0,008867	0,008116
validation	0,018170	0,012256	0,014236	0,015242
testing	0,084692	0,073457	0,048701	0,076342

Table 5: Root-mean-square-error (RMSE) for short term prediction ($\Delta t_p = 24$ h, $\Delta t_i = 5$ min) of *response time*. We cross benchmark four modeling techniques: multivariate linear regression, support vector machines, universal and radial basis functions. We report RMSE for the three data segments parametrization, validation and testing. The best model is typically selected by its performance on the validation data set. In this case SVM produce the smallest error with RMSE = 0.012.

	ML	SVM	UBF	RBF
parametrization	0,007714	0,006818	0,007985	0,006663
validation	0,009323	0,006852	0,008147	0,008139
testing	0,428725	0,126498	0,064269	0,135337

Table 6: Root-mean-square-error (RMSE) for long term prediction ($\Delta t_p = 24$ h, $\Delta t_i = 2$ days) of *response time*.

Sensitivity Analysis

Sensitivity analysis is the analysis of how sensitive outcomes are to changes in the assumptions. In the previous two steps we have *first* identified and *secondly* employed particularly predictive variables to build statistical models. But how sensitive is the outcome of our models to each variable? The knowledge of this sensitivity factor can potentially help increase the knowledge about how the system under scrutiny works.

In this step we take each of the four models built in the previous *model building* step (ML, SVM, UBF, RBF) and remove each of the variables found in the *variable selection* step one at a time. We then calculate the change in model error RMSE for the validation and the generalization data set. The percentage change in RMSE we report in Table 10. The upper part in Table 10 shows the data for the validation set, the lower part shows data for the generalization data set. For the validation set we also

report the linear correlation coefficient of each variable with the target value *response time*. It is interesting to note that variables to which the model is highly sensitive to are not necessarily highly correlated with the target. For example, by removing the variable *IO_recv_pkts* from the SVM model, RMSE increases by 23% on the validation data set, however removing the number of new processes increases the RMSE only by 3.5%.

5.2 Free Physical Memory

In this section we use the same methodology described in 5.1. to model the *free physical memory*. Figure 3 shows a plot over time, the variable first decreases then increases again stepwise. This behavior cannot readily be explained by resource exhaustion oder software aging. In the Apache server, *free physical memory* cannot be lower than a preset threshold value. If physical memory approaches the lower limit, the systems frees up memory by paging. Thus, we get an irregular utilisation pattern.

Data Preprocessing

Free physical memory shows similar spikes as *response time* does. However, only few of the spikes coincide, making a mutual root cause less likely. Besides the spikes we observe a rather linear behavior except for some abrupt changes when the server frees physical memory. Our ultimate goal is to forecast future realizations of this variable. We start by looking at the first differences of this variable (Figure 4a) and its autocorrelation function (Figure 4b). However, there is not much autocorrelation in the first differences that could be used for modeling. Autocorrelation depletes at the first step which correlates to a 5 minutes horizon.

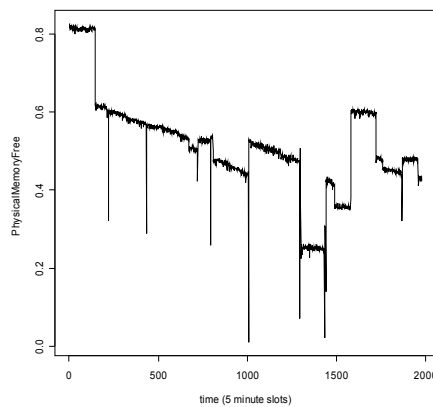


Figure 3: *Free physical memory* (normalized values) for the Apache web server.

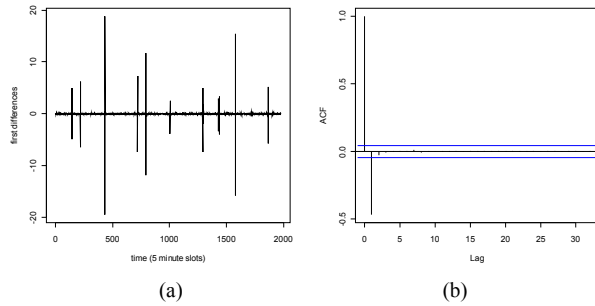


Figure 4: (a) First differences of *free physical memory*. (b) Autocorrelation function of first differences. There is not much autocorrelation information in the first differences that could be used for model building.

Variable Selection

We continue with variable selection and cross benchmark again our three selection procedures for $\Delta t_p = 5$ min, $\Delta t_i = 5$ min, $\Delta t_e = 5$ min. In Table 7 we list the RMSE results for the parametrization and the validation data set for this step.

	RMSE		
	forward selection	backward elimination	PWA
parametrization	0,026648	0,028293	0,026934
validation	0,019938	0,220954	0,018614

Table 7: Variable selection: Root-mean-square-error (RMSE) for $\Delta t_p = 5$ min, $\Delta t_i = 5$ min of free available memory.

Variable Nr.	Variable Name
3	PhysicalMemoryFree at time t-1
15	TimeUserMode
30	NewProcesses
48	si_files_cache

Table 8: The actual variables selected by the PWA procedure.

Based on the lowest RMSE error rate generated on the validation data set we select the variable set produced by the PWA mechanism. Variables are listed in Table 8. Given the autocorrelation at lag $t-1$ not surprisingly the *free physical memory* at $t-1$ is in the variable set.

Model Building

We cross-benchmark again the variable set identified in the previous step with the four modeling techniques introduced earlier: multivariate linear, support vector machines, radial and universal basis functions for $\Delta t_p = 5$ min, $\Delta t_i = 5$ min, $\Delta t_e = 5$ min. Results are shown in Table 9.

	ML	SVM	UBF	RBF	ARMA
parametrization	0,017971	0,017580	0,017638	0,017620	0,02231004
validation	0,026998	0,027021	0,026706	0,026759	0,03517245
testing	0,018513	0,051723	0,017123	0,059156	0,02235754

Table 9: Model building: Root-mean-square-error (RMSE) for for $\Delta t_p = 5$ min, $\Delta t_i = 5$ min of free available memory.

Sensitivity Analysis

In this step we remove one variable at a time from each model (i.e. ML, SVM, UBF, RBF) and calculate the change in the models RMSE for the validation and the generalization data set. In Table 11 we report the models RMSE for each variable removed and the percentage

change in RMSE. All models show that prediction performance is particularly sensitive to two variables which are *PhysicalMemoryFree* at time $t-1$ and *si_files_cache*. Given the high autoregressive correlation, *PhysicalMemoryFree* is an expected result. However, *si_files_cache* shows only a small linear correlation of 0.075 with the target variable, however its removal would increase the models RMSE by 26% for ML models and 15.8% for UBF models. In the generalization data set (Table 11 lower part) this effect becomes even more clear. The removal of *si_files_cache* would increase model errors by 54% for ML and 139% for UBF models.

6 Results

In this section, we summarize our results. In Section 3 we have introduced a terminology and a formal task description. Based on earlier work we have detailed a best practise guide for empirical modeling of complex software systems in Section 4. In this guide we focus on variable selection and sensitivity analysis, two aspects that have received little attention in literature. In Section 5 we have put our guide to the test by modeling and predicting two performance variables of the Apache web server *response time* and *freePhysicalMemory*.

6.1 Best Practice Guide

It is a complicated, arduous, and time-consuming task for even experienced system administrators and statistical analysts to know what a reasonable set of steps should comprise to model and successfully predict performance variables or system failures of a complex digital system. Thus, this first and by no means complete attempt of a best practice guide intends to assist in configuring modeling approaches systematically for best prediction results. In three examples we detail aspects of our guide focusing on variable selection (i.e. *complexity reduction*), modeling and sensitivity analysis.

6.2 Variable Selection

The objective of variable selection is threefold: 1) improve the prediction result of the underlying model, 2) provide faster and more resource effective (e.g. time, cost) models and c) provide a better understanding of the underlying process that generated the data ([21]). Regarding the *first* aspect we have cross benchmarked three popular variable selection procedures (forward selection, backward elimination and probabilistic wrapper) for three distinct applications (Apache *response time* short and long term prediction and short term *free physical memory* prediction). Our results suggest that forward selection and the probabilistic wrapper approach (PWA) significantly and consistently outperform the backward elimination procedure by an order of magnitude. This is due to the fact that that backward elimination starts out with *all* available variables and then reduces this set to minimize the models error. This procedure is easily affected by the curse of dimensionality. This result is also a strong indicator that variable selection should be pre-

Sensitivity Analysis

Variable Nr.	Variable Name	RMSE on validation data				RMSE change in %				Correlation coefficient
		ML	SVM	UBF	RBF	ML	SVM	UBF	RBF	
	all listed variables	0,018170	0,012256	0,014236	0,015242					
30	NewProcesses	0,019426	0,012683	0,014612	0,015545	6,91%	3,48%	2,64%	1,99%	0,956424
23	DiskWrittenBlocks	0,019316	0,012656	0,014634	0,015539	6,30%	3,26%	2,79%	1,94%	0,955246
89	eth0_trans_pkts	0,018384	0,012679	0,016104	0,016512	1,18%	3,45%	13,12%	8,33%	0,953614
65	lo_recv_pkts	0,020455	0,015131	0,016818	0,018063	12,57%	23,46%	18,14%	18,51%	0,937351
27	SwapOutCounter	0,021763	0,012855	0,015363	0,017086	19,77%	4,89%	7,92%	12,10%	0,755900
63	si_slab_cache	0,018176	0,012288	0,014059	0,015207	0,03%	0,26%	-1,25%	-0,23%	0,682239
59	si_size_256	0,019173	0,013181	0,012582	0,015144	5,52%	7,55%	-11,62%	-0,64%	0,458101
29	ContextSwitches	0,018207	0,011776	0,014161	0,015234	0,20%	-3,92%	-0,53%	-0,05%	0,220309
6	CachedMemory	0,018175	0,011837	0,014025	0,014977	0,03%	-3,42%	-1,48%	-1,74%	0,058488
	adjusted set	0,018170	0,011238	0,013654	0,015240	0,00%	-8,31%	-4,09%	-0,02%	

Sensitivity Analysis

Variable Nr.	Variable Name	RMSE on unseen test data				RMSE change in %			
		ML	SVM	UBF	RBF	ML	SVM	UBF	RBF
	all listed variables	0,084692	0,073457	0,048701	0,076342				
30	NewProcesses	0,069045	0,068437	0,091184	0,108526	-18,47%	-6,83%	87,23%	42,16%
23	DiskWrittenBlocks	0,153435	0,070006	0,092068	0,105964	81,17%	-4,70%	89,05%	38,80%
89	eth0_trans_pkts	0,231876	0,076665	0,207850	0,085982	173,79%	4,37%	326,79%	12,63%
65	lo_recv_pkts	0,040331	0,079400	0,051371	0,021927	-52,38%	8,09%	5,48%	-71,28%
27	SwapOutCounter	0,270449	0,046728	0,097349	0,160244	219,33%	-36,39%	99,89%	109,90%
63	si_slab_cache	0,091150	0,068403	0,028253	0,053292	7,63%	-6,88%	-41,99%	-30,19%
59	si_size_256	0,061327	0,082852	0,116621	0,023679	-27,59%	12,79%	139,46%	-68,98%
29	ContextSwitches	0,084956	0,065106	0,040642	0,085192	0,31%	-11,37%	-16,55%	-11,59%
6	CachedMemory	0,083107	0,088105	0,042010	0,081243	-1,87%	19,94%	-13,74%	6,42%
	adjusted set	0,084692	0,089188	0,028132	0,035020	0,00%	21,42%	-42,24%	-54,13%

Table 10: Results for sensitivity analysis for short term *response time* forecasting. The upper part shows the data for the validation set, the lower part shows data for the generalization data set. For the validation set we also report the linear correlation coefficient of each variable with the target value *response time*. The left column lists all variables identified in the variable selection step. The next column lists RMSE values when the variable in the corresponding row has been removed from the model. The following column lists the percentage changes in model error for this case. Clearly *IO_recv_pkts* is the one variable which is most influential in prediction. Its removal increases model error by 23% for SVM models. On the generalization data set (displayed in the lower part) its removal would still increase the SVM's model error by 8%. See text for further explanation.

Variable Nr.	Variable Name	RMSE on validation data				RMSE change in %				Correlation coefficient
		ML	SVM	UBF	RBF	ML	SVM	UBF	RBF	
	all listed variables	0,026998	0,027021	0,026706	0,026759					
3	PhysicalMemoryFree at t-1	0,057083	0,050432	0,045837	0,041312	111,44%	86,64%	71,63%	54,39%	
15	TimeUserMode	0,027136	0,027309	0,027022	0,026967	0,51%	1,06%	1,18%	0,78%	
30	NewProcesses	0,027927	0,026295	0,027616	0,027589	3,44%	-2,69%	3,41%	3,10%	
48	si_files_cache	0,034094	0,027987	0,030931	0,029154	26,28%	3,57%	15,82%	8,95%	
	adjusted set	0,026998	0,026295	0,026706	0,026759	0,00%	-2,69%	0,00%	0,00%	

Variable Nr.	Variable Name	RMSE on unseen test data				RMSE change in %			
		ML	SVM	UBF	RBF	ML	SVM	UBF	RBF
	all listed variables	0,018513	0,051723	0,017123	0,059156				
3	PhysicalMemoryFree at t-1	0,182444	0,135516	0,370749	0,184623	885,47%	162,00%	2065,19%	212,09%
15	TimeUserMode	0,015504	0,019340	0,016559	0,015847	-16,26%	-62,61%	-3,30%	-73,21%
30	NewProcesses	0,018822	0,047455	0,023734	0,088290	1,67%	-8,25%	38,61%	49,25%
48	si_files_cache	0,028621	0,063636	0,040992	0,221926	54,60%	23,03%	139,39%	275,15%
	adjusted set	0,018513	0,047455	0,017123	0,059156	0,00%	-8,25%	0,00%	0,00%

Table 11: Sensitivity analysis for Apache's performance variable *freePhysicalMemory*. The upper table shows data for the validation data set, the lower table covers the generalization data set. For the validation data set we also show the linear correlation coefficient of each variable for the target variable *freePhysicalMemory*. The left column shows all variables identified in the variable selection step. The next column shows RMSE values when this variable is removed from the model. The next column lists changes to the models performance in this case.

12th IEEE International Symposium Pacific Rim Dependable Computing (PRDC'06), University of California, Riverside, USA, Dec. 18-20, 2006

ferred over an approach which unreflectedly uses all available variables. Regarding the *third* aspect, we found that when predicting *response time* of the Apache web server, there is a set of basis variables that remains constant, even if the lead time varies from minutes to days. This base set can be further used to understand the underlying processes and for theory building or verification.

6.3 Statistical Modeling

In our experiments we cross benchmarked multivariate linear (ML) and multivariate nonlinear (support vector machines, radial and universal basis functions) data driven modeling techniques. When predicting *response time* the support vector machine (SVM) approach outper-

formed other techniques. When predicting *free physical memory* the universal basis function (UBF) approach turns out to be superior. In all cases the winning margin is far from being an order of magnitude. This result suggests that focusing resources on variable selection *instead* of the type of modeling technique may yield faster and less costly results.

6.4 Sensitivity Analysis

Sensitivity analysis is the analysis of how sensitive outcomes are to changes in the assumptions. We have performed sensitivity analysis for models that predict short term response rates and that predict *free physical memory* of the Apache web server. We removed each variable

from the models, one at a time, and calculated the models change in RMSE. As expected, the RMSE increases for the multivariate linear models with each variable removed. However, this is not the case with nonlinear models. This result suggests that variable selection with a nonlinear wrapper may yield improved model quality. In fact when we rebuilt SVM, UBF and RBF models with this new set of variables we decreased the models error by up to 8% on the validation data set and by up to 54% on the generalization data set.

7 Conclusions and Future Work

This paper proposes a best practice guide for building empirical models for the prediction of performance variables. It explores its feasibility by applying it to forecasting two performance variables of the Apache web server. Recent developments in empirical modeling of complex software systems have addressed the problem from the pragmatic point of view by applying statistical regression techniques. The focus has been on trend analysis, linear and nonlinear regression techniques to sometimes high dimensional input spaces. While the step from trend analysis to multivariate regression techniques in some cases showed a significant improvement, the step from multivariate linear to nonlinear procedures has not always resulted in a notable increase in prediction quality. One reason for this observation may be that multivariate techniques working in high dimensional data spaces easily

suffer from the curse of dimensionality and may over-fit the data. We found that applying first a method that automatically reduces the complexity of the input space (i.e. variable selection) yields a) improved prediction performance and b) a more compact set of variables which can serve as a cornerstone for further system and sensitivity analysis.

The techniques we cross-benchmarked in regression techniques and variable selection are diverse and motivated by various theoretical arguments, but a unifying theoretical framework is lacking. Because of these shortcomings, it is important to have some guidelines before tackling a prediction problem. To that end we have summarized our experience we gained when modeling and predicting performance variables and critical events in the Apache web server and a commercial telecommunication system in this best practice guide. We recommend focusing the modeling efforts on the variable selection step because it yields consistently the largest improvement (one order of magnitude) in prediction performance in our test cases. Results from sensitivity analysis suggest that variable selection with a nonlinear wrapper approach may improve model quality further. Favoring nonlinear over linear multivariate regression techniques consistently improvements, however they may not always be significant. Additionally, the question of optimal reaction schemes has to be addressed in an effort to close the loop (Figure 1). Also, the question of causality inference (i.e. finding root cause) must be investigated.

References

- [1]: Arlat J., Crouzet Y., Karlsson J., *Comparison of Physical and Software- Implemented Fault Injection Techniques*, IEEE Transactions on Computers, pp. 1115-1133, 2003
- [2]: Ascher, H.E. Lin, T.-T.Y. Siewiorek, D.P., *Modification of: error log analysis: statistical modeling and heuristic trend analysis*, IEEE Transactions on Reliability, Vol. 41(4), 1992
- [3]: Baum E., Haussler D., *What size net gives valid generalization?*, Neural computation 1(1):151-160, 1989
- [4]: Bishop C. M., *Neural Networks for Pattern Recognition*, Clarendon Press, London, 1995
- [5]: Bobbio, Garg, Gribaudo, Horvath, Sereno, Telek, *Modeling Software Systems with rejuvenation, restoration and checkpointing through fluid petri nets*, Proc. 8th Int. Workshop on Petri Net and Performance Models (PNPM'99), Zaragoza, Spain, pp. 82-91, 1999
- [6]: Burns A., Edgar S. (2000), Predicting Computation Time for Advanced Processor Architectures, , (), 2000
- [7]: Burns A. And G. Bernat and I. Broster, *A Probabilistic Framework for Schedulability Analysis*, Proceedings of the Third International Embedded Software Conference, EM-SOFT, Lecture Notes in Computer Science, pp. 1-15, 2003
- [8]: V. Castelli, R.E. Harper, P. Heidelberger, S.W. Hunter, K.S. Trivedi, K. Vaidyanathan and W.P. Zeggert, *Proactive Management of Software Aging*, IBM JRD, Vol 45, No. 2, pp. 311-332, 2001
- [9]: Chen Mike , Emre Kiciman, Eugene Fratkin, Armando Fox and Eric Brewer, *Pinpoint: Problem Determination in Large, Dynamic Systems*, Proceedings of International Performance and Dependability Symposium, Washington, DC, 2002
- [10]: C. Cortes and V. N. Vapnik, *Support vector networks*, Machine Learning, 20: 273-297, 1995
- [11]: Crowell Jonathan, Mark Shereshevsky, Bojan Cukic, *Using Fractal Analysis to Model Software Aging*, Lane Department of Computer Science and Electrical Engineering, 2002
- [12]: Edsger Wybe Dijkstra, *Notes On Structured Programming*, Technical Report 70-WSK-03, Technological University Eindhoven, Department of Mathematics (<http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF>), 1970
- [13]: Dohi T., K. Goseva-Popstojanova and K. S. Trivedi, *Analysis of Software Cost Models with Rejuvenation*, Proc. of the IEEE Intl. Symp. on High Assurance Systems Engineering, HASE-2000, 2000
- [14]: Dohi T. , Goseva-Popstojanova Katerina, Trivedi Kishor S., *Statistical Non-Parametric Algorithms to Estimate the Optimal Software Rejuvenation Schedule*, Dept. of Industrial and Systems Engineering, Hiroshima University, Japan and Dept. of Electrical and Computer Engineering, Duke University, USA, 2000
- [15]: Frank, P.M., *Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy - A Survey and Some New Results*, Automatica, Vol. 26, No. 3, pp. 459-474, 1990

- [16]: Gardner Mark, *Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems*, PhD Thesis at University of Illinois at Urbana-Champaign, 1999
- [17]: Garg S., A. Puliafito M. Telek and K. S. Trivedi, *Analysis of Software Rejuvenation using Markov Regenerative Stochastic Petri Net*, In Proc. of the Sixth IEEE Intl. Symp. on Software Reliability Engineering, Toulouse, France, 1995
- [18]: Garg S., A. van Moorsel, K. Vaidyanathan and K. S. Trivedi, *A Methodology for Detection and Estimation of Software Aging*, In Proc. of the Ninth IEEE Intl. Symp. on Software Reliability Engineering, Paderborn, Germany, 1998
- [19]: Garg S., Puliofito A., Telek M., Trivedi K., *Analysis of preventive Maintenance in Transaction based Software Systems*, Department of electrical & computer engineering Duke University Durham, 1998
- [20]: Geman, S., Bienenstock, E. and Doursat, R., *Neural Networks and the Bias/Variance Dilemma*, Neural Computation, 4, pp. 1-58, 1992
- [21]: Isabelle Guyon, Andre Elisseeff, *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research 3, 1157-1182, 2003
- [22]: Hochreiter Sepp, Obermayer Klaus, Isabelle Guyon, Steve Gunn, Masoud Nikrav, *Nonlinear Feature Selection with the Potential Support Vector Machine*, Feature extraction, Foundations and Applications, Springer, 2004
- [23]: Hoffmann G. A., *A Radial Basis Function Approach to Modeling Resource Demands in Distributed Computing Systems*, ISMIP - International Symposium on Multi Technology Information Processing, Taiwan, 1996
- [24]: Hoffmann G., *Failure Prediction in Complex Computer Systems: A Probabilistic Approach*, PhD Thesis Humboldt University Berlin, 2005
- [25]: Guenther Hoffmann, Mirosław Malek, *Call Availability Prediction in a Telecommunication System: A Data Driven Empirical Approach*, IEEE Symposium on Reliable Distributed Systems (SRDS 2006), 2006
- [26]: Hoffmann, Trivedi, *Invariant Detection for Apache Server Systems*, Duke University, NC, 2006
- [27]: Huang Y., C. Kintala, N. Kolettis and N. Fulton, *Software Rejuvenation: Analysis, Module and Applications*, In Proc. of the 25th IEEE Intl. Symp. on Fault Tolerant Computing (FTCS-25), Pasadena, CA, 1995
- [28]: Huang, Chung, Wang, Liang, *NT-SwiFT: Software Implemented Fault Tolerance on Windows NT*, Bell Labs, Lucent Technologies and Institute of Information Science, Academia Sinica, Proceedings of the 2nd USENIX Windows NT Symposium, 1998
- [29]: Kapadia, Fortes, Brodley, *Predictive Application-Performance Modeling in a Computational Grid Environment*, Eighth IEEE International Symposium on High Performance Distributed Computing, 1999
- [30]: Lei Li, Kalyanaraman Vaidyanathan, Kishor S. Trivedi, *An Approach for Estimation of Software Aging in a Web Server*, International Symposium on Empirical Software Engineering (ISESE'02), 2002
- [31]: Li L., Vaidyanathan, Trivedi, *An Approach for Estimation of Software Aging in a Web Server*, Department of electrical & computer engineering Duke University Durham, 2002
- [32]: Bev Littlewood, Lorenzo Strigini, *Software reliability and dependability: a roadmap*, Proceedings of the conference on The future of Software engineering, p.175-188, Limerick, Ireland, 2000
- [33]: Liu H., Setiono R., *A Probabilistic Approach to Feature Selection - A Filter Solution*, Proceedings of Machine Learning: ICML'96 - 319-327, Bari, Italy, 1996
- [34]: MacKay D.J.C., *Bayesian interpolation*, Neural Computation 4(3) pg. 415-447, 1992
- [35]: Salfner, Hoffmann, Malek, *Prediction-based Software Availability Enhancement*, Hot Topics Volume of the Springer Lecture Notes in Computer Science (LNCS), 2005
- [36]: Malek, Salfner, Hoffmann, *Self-Rejuvenation - an Effective Way to High Availability*, International Workshop on Self-Star Properties in Complex Information Systems, University of Bologna; Bertinoro (Forl) - Italy, 2005
- [37]: Parker T.S., Chua L.O., *Practical Numerical Algorithms for Chaotic Systems*, Springer Verlag, 1989
- [38]: Patton, R.J., P.M. Frank and R.N. Clark, *Fault Diagnosis in Dynamic Systems: Theory and Applications*, Prentice-Hall, 1989
- [39]: Pizza, Strigini, Bondavalli, Giandomenico, *Optimal Discrimination between Transient and Permanent Faults*, 3rd IEEE High Assurance System Engineering Symposium, 1998
- [40]: Rumelhart, Hinton, Williams, *Learning internal representation by error propagation*, In Rumelhart, McClelland eds. Parallel Distributed Processing Volume I, MIT Press, 1986
- [41]: Salfner F., Tschirpke S., Malek M., *Comprehensive Logfiles for Autonomic Systems*, IEEE Proceedings of IPDPS 2004 (International Parallel and Distributed Processing Symposium), 2004
- [42]: Schoelkopf B., Smola A., *Learning with Kernels*, MIT Press, 2002
- [43]: Shereshevsky Mark, Jonathan Crowell and Bojan Cukic, *Multifractal Description of Resource Exhaustion Data in Real Time Operating System*, Technical Report, West Virginia University, 2001
- [44]: Shereshevsky Mark, Jonathan Crowell, Bojan Cukic, Vijai Gandikota, Yan Liu, *Software Aging and Multifractality of Memory Resources*, International Conference on Dependable Systems and Networks (DSN-2003), San Francisco, 2003
- [45]: Terrasa Andres, Bernat Guillem, *Extracting Temporal Properties from Real-Time Systems by Automatic Tracing Analysis*, Technical University Valencia, Real-time systems research group University of York, 2003
- [46]: Trivedi Kishor S., Vaidyanathan Kalyanaraman and Goseva-Popstojanova Kater, *Modeling and Analysis of Software Aging and Rejuvenation*, Center for Advanced Computing & Communication, Dept. of Electrical & Computer Engineering, Duke University, Durham, NC 27708, USA, 2000
- [47]: Ulerich, N. H., & Powers, G. A., *Online hazard aversion and fault diagnosis in chemical processes: the digraph/fault tree method*, IEEE Transactions on Reliability 37 (2), pp. 171-177., 1988
- [48]: Vaidyanathan K. and K. S. Trivedi, *A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems*, In Proc. of the Tenth IEEE Intl. Symp. on Software Reliability Engineering, Boca Raton, Florida, 1999
- [49]: Ward A., Whitt W., *Predicting Response Times in Processor-Sharing Queues*, Stanford University (Department of Management Science and Engineering), AT&T Labs (Shan-

- non Laboratory), Proceedings of the Fields Institute Conference, 2000
- [50]: Weigend A. S., Gershenfeld N. A., eds., *Time Series Prediction*, Proceedings of the Santa Fe Institute, Vol. XV, 1994
- [51]: Weiss G. M., *Timeweaver: a Genetic Algorithm for Identifying Predictive Patterns in Sequences of Events*, Rutgers University and AT&T Labs, 1999
- [52]: Weiss G. M., *Predicting telecommunication equipment failures from sequences of network alarms*, In W. Kloesgen and J. Zytow, editors, *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, 2001