# Failure Prediction based on Log Files Using the Cox Proportional Hazard Model

Ilenia Fronza, Alberto Sillitti, Giancarlo Succi, Jelena Vlasenko
Free University of Bolzano - Bozen
Piazza Domenicani - Domenikanerplatz, 3,
I-39100 Bolzano - Bozen, Italy
{Ilenia.Fronza, Alberto.Sillitti, Giancarlo.Succi}@unibz.it, Jelena.Vlasenko@stud-inf.unibz.it

**Abstract—Accurate failure predictions can help in mitigating the impact of computer failures. Resources, applications, and services can be scheduled to limit the impact of failures. However, providing accurate predictions with sufficient lead of time is challenging. Log files track changes of system state. A sequence or a pattern of messages may be used to predict failures. Here we describe an approach to predict failures based on the Cox Proportional Hazards (PH) model that has been applied successfully in various fields of research. We apply our approach to log files collected during approximately 3 months of work in a large Italian company. We compare the performance of the proposed model with Support Vector Machines.**

***Keywords—failure prediction, Cox Proportional Hazard model, logfiles***

## I. Introduction

The impact of failures can be substantial since the recovery process can require unexpected amounts of time and resources [1; 10]. Being able to forecast failures is extremely important, even when failures are inevitable – at least recovery or rescue actions can be taken [25].

Methods for the prediction of system failures based on events (in our case, the log messages) have been proposed in various engineering disciplines. These methods can be classified into design-based methods and data-driven rule-based methods. In a design-based method, the expected event sequence is obtained from the system design and is compared with the observed event sequence [19; 21; 24]. The major disadvantage of these methods is that in many cases events occur randomly and thus there is no system logic design information available.

Data-driven rule based methods do not require system logic design information. These methods are made of two phases: 1) identification of temporal patterns, i.e., sequences of events that frequently occur [16] and 2) development of prediction rules based on these patterns [14].

System log files consist of messages created while the system is running. The information recorded varies from general messages concerning user logins to more critical warnings about program failures [10]. Log file information can be analysed to find causes of failures. Although this type of forensic analysis is valuable, it is also possible to use the information contained in system log files for predicting events.

There have been several approaches for predicting system failures using system log files.

Prediction methods using log files include standard machine learning techniques such as Support Vector Machines, Bayes networks, Hidden Markov Models, and Partially Observable Markov Decision Process [8; 9; 10; 11; 15; 20; 26; 28; 29]. The use of time-series analysis is common among these methods since a system message in isolation has been shown to be insufficient for predicting failure [29]. Unfortunaly, the large amount of information available in system log files makes finding the right pattern(s) difficult [10].

In this paper we introduce a new approach for predicting critical system events based on the Cox PH model. The actualization of such idea will set the path for the development of devices that read logs of running applications and signal the likely crash of such systems.

The Cox PH model has been applied mainly in biomedicine, often for the study of cancer survival [5; 31]. It has also been applied successfully in various fields of research, such as criminology [4], sociology [2], marketing [3]. There are limited uses of the Cox PH model in cybernetic [14] and also an application to software data [27] where, using as input code metrics, failure time data coming from bug report were analysed.

The paper is structured as follows: in Section 2, we present some background; in Section 3, we introduce our approach. In Section 4, we describe our experiments and results. In Section 5 we discuss our results.

## II. Background

*!"#$%& ' ()&*+,$"*

System log files are important for managing computer systems since they provide a history or audit trail of events. In this context, an event is a change in a system status, such as a user login or an application failure. Several studies have proposed different approaches for predicting system failures using system log files [8; 9; 10; 11; 14; 15; 20; 26; 28; 29].

System log files typically are text files that consist of messages sent to the logging service by applications. Applications can send information to the logging service process that stores the messages in a text file in an arrival

order. The logging service is primarily responsible for managing the log file while the message content is largely created by the application. In the dataset used in this study, each log entry consists of eight fields (Table I). The time field is the time when the message was recorded. One field stores the name of the application that was running; the machine sending the message is identified by two fields, server and computer (since the logging service serves multiple computers on different servers). The logged-in user is reported in the UsedName field, and Severity contains the level of severity of the message. In addition, some of the logs (not all), contain also an error name and description of the event that has occurred.

TABLE I. FIELDS OF A LOG MESSAGE IN THE ANALYSED DATASET.

| Field Name | Description |
|---|---|
| Time | Time the message was recorded |
| Application | Running application |
| Server | Machine sending the message |
| Computer | |
| UserName | Name of the logged user |
| Severity | Level of severity of the message |
| Operation | Performed operation |
| Description | Some of the logs (not all) contain also the error name and the Description of the event that has occurred |
| ErrorName | |

## - ./$%0(1%23%4(5$,

Cox PH model (Cox 1972) gives an expression for the hazard at time # for an individual + with a given specification of 6% covariates *1*:

$$h(t|x) = h_0(t)e^{\sum_{i=1}^{n}\beta_i x_i} \quad (1)$$

The Cox model formula says that the hazard at time t is the product of two quantities. The first of them, $h_0(t)$, is called the baseline hazard function and is equal for all individuals; it may be considered as a starting version of the hazard function, prior to considering any of the x´s. Cox PH model focuses on estimating regression coefficients *7*´s leaving the baseline hazard unspecified. *7* is a vector of regression coefficients; in the p < n setting, *7*´s are estimated by maximizing the log partial likelihood, which is given by:

$$l(\beta) = \sum_{i=1}^{n} \delta_i \left\{ x_i\beta - \log\left[ \sum_{j \in R(t_i)} e_i^x \beta \right] \right\} \quad (2)$$

Where $R(t_i)$ is the risk set at time $t_i$, i.e. the set of all individuals who are still under study just prior to time $t_i$.

A parametric survival model is one in which survival time (the outcome) is assumed to follow a known distribution. The Cox PH model is not a fully parametric model; rather it is a semi-parametric model because even if the regression parameters ß´s are known, the distribution of the outcome remains unknown. The Cox PH model is a "robust" model, since the results obtained from it closely approximate the results of the correct parametric model.

The key assumption of the Cox PH model is proportional hazards; this assumption means that the hazard ratio (defined as the hazard for one individual over the hazard for a different individual) is constant over time.

Cox PH model is widely used because of its characteristics: 1) even without specifying $h_0(t)$, it is possible to find the *7*´s, 2) no particular form of probability distribution is assumed for survival times, and 3) it uses more information – the survival times – than the logistic model, which considers a (0,1) outcome and ignores survival times and censoring. Therefore, it is preferred over the logistic model when survival time is available and there is censoring [13].

## 0 2$8(8%:;<$% ""$""%$;#

A contingency table (sometimes called confusion matrix) is a convenient way to tabulate statistics for evaluating the quality of a model. In Table II, TP, FP, TN and FN stand for true/false positive/negative counts, respectively; PP and PN stand for predicted positive/negative; and POS and NEG stand for actual positive/negative.

In this paper we will only consider metrics that can be defined in terms of the counts in a contingency table; this excludes, e.g., metrics that consider model complexity [7]. The most relevant metrics of this type are reported in Table III.

TABLE II. CONTINGENCY TABLE (A.K.A., CONFUSION MATRIX).

| | | Actual Value | |
|---|---|---|---|
| | | Pos | Neg |
| Prediction Outcome | PP | TP | FP |
| | PN | FN | TN |
| PP=Predicted Positive, PN=Predicted Negative, Pos=Actual Positive, Neg=Actual Negative, TP=True Positive, FP=False Positive, TN=True Negative, FN=False Negative | | | |

TABLE III. DERIVATIONS FROM A CONFUSION MATRIX: MOST RELEVANT METRICS FOR CLASSIFICATION PERFORMANCE EVALUATION [17].

| Name (derivations) | Definition |
|---|---|
| True Positive Rate =.2>?% $;"+#+@+#!?%>$<:,,A | $\frac{TP}{TP+FN}$ |
| False Positive Rate =*2>?%9:,,%(B#A | $\frac{FP}{FP+TN}$ |
| Accuracy | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| True Negative Rate =.C>?% 6$<+9+<+#!A | $\frac{TN}{TN+FP}$ |
| Balance | $1-\frac{\sqrt{(0-FPR)^2+(1-TPR)^2}}{\sqrt{2}}$ |

We use Receiver Operating Characteristics (ROC) space to analyse the classifier performance. False positive rate is plotted against the true positive rate. The graph (Fig. 1) shows

the trade-off benefits (TP) and costs (FP). The points (0,0) and (1,1) represent the training-free classifiers ,D:!"C$):#+@$& and ,D:!"2("+#+@$; the point (0,1) represents the ideal classifier, and (1,0) represents the classifier which gets it all wrong. The ascending diagonal (0,0)–(1,1) represents random training-free behaviour: any point (*6*,*6*) can be obtained by predicting positive with probability *6*& and negative with probability (1–*6*). The upper left triangle contains classifiers that perform better than random, while the lower right triangle contains those performing worse than random. The descending diagonal (0,1)–(1,0) represents classifiers that perform equally well on both classes (.*2*>=1–**2*); left of this line we find classifiers that perform better on the negatives than the positives, to the right performance on the positives dominates [7].

Balance is defined in [17] as the Euclidean distance from the sweet spot FPR= 0, TPR=1 to a pair of (FPR; TPR) . For convenience, we 1) normalize balance by the maximum possible distance across the ROC square ( $\sqrt{2}$ ), and 2) subtract this from 1, i.e. Hence, better and higher balances fall closer to the desired sweet spot of FPR= 0, TPR=1. In particular, classifiers having balance higher than 0.5 fall in the upper left triangle of the ROC space. Thus, we consider 0.5 as a threshold for balance.

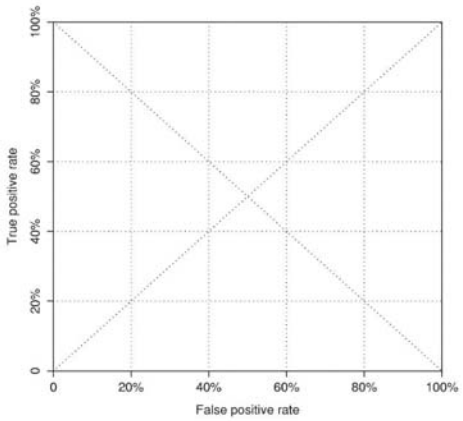


Figure 1. ROC space structure (from [7]).

Ideally, we seek predictors that maximize accuracy, TPR, and TNR. Note that maximizing any one of these does not imply high values for the others. Accuracy is a good measure of a learner's performance when the possible outcomes occur with similar frequencies, and is not suggested when class distributions are uneven [17]. Therefore, this paper will assess its learned predictors using balance, TPR, and FPR and not accuracy.

## III. Approach

#8B<#B8$&(%#/$&:668(:</

We propose a technique to predict the failure of a running software systems using log files. The idea is to develop devices that read logs of running applications and signal the likely crash of such systems. Fig. 2 presents a schematic view of the proposed approach. While the system is running, log data are collected to track the actual execution path. In this work, we look at the running system as a "black box", meaning that we do not have any other information about the system except the log files.

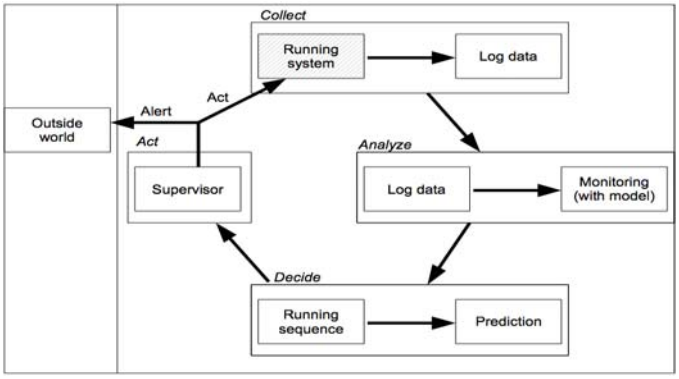


Figure 2. Schema of the proposed approach.

The monitoring process takes log data as input and, basing on the analysis performed, gives to the supervisor a message indicating the "likely failure" for the running application. The supervisor can act directly on the running system to avoid the predicted failure, or send an alert to the outside world. Possible actions could be to abort the running system, to restart it, to dynamically load components, or to inform the running system if it was a suitably structured autonomic system [18].

- #8B<#B8$&(%#/$&4(;+#(8+;)&28(<$""

EA F+%$;"+(;:,&8$5B<#+(;&(9&#/$&68(G,$%&:;5&5:#:&68$6:8:#+(;

While the system is running, data are <(,,$<#$5 [6; 22; 23] to track the actual execution path. Then, to get from raw logs to temporal event sequences:

1. data are parsed to extract Operation, Time and Severity fields;
2. duplicate rows are deleted together with logs that have missing information in one or several fields Operation, Time stamp, Severity;
3. sequences of activities are extracted: a new sequence starts either if there is a 'Log in' operation or if the day changes.
4. for each sequence, the number of occurrences of each operation is found.

Failures are defined as sequences containing at least one severity "Error".

HA 4(5$,&.8:+;+;)&:;5& ;:,!"+"&(%#/$&8$"B,#"

Following the guidelines of [12; 30; 31], model training includes the following steps:

1. the Schoenfeld test [13] is applied to select the operations satisfying the PH assumption.
2. the Cox PH model is applied and operations that are significantly associated to failures are identified.

3. for each sequence a risk score is evaluated according to the exponential value of a linear combination of the multiplicity of the operation, weighted by the regression coefficients derived from the aforementioned Cox PH model.
4. % is extracted, as the third quartile of risk scores of non failure sequences;
5. The risk score RS is then evaluated for the actual sequence of the running application, as in point 4. A message of "likely failure" about the running application is given as output to the supervisor when RS ≥ m.

## IV. Experiments

*F:#:*

We use log files collected during approximately 3 months of work in an important Italian company that prefers to remain anonymous. Table IV presents some descriptive statistics about the six application in the dataset.

TABLE IV. Descriptive statistics.

| App | Number of sequences | Pos* (%) |
|---|---|---|
| A1 | 12765 | 0.84 |
| A2 | 718 | 15.88 |
| A3 | 60 | 23.33 |
| A4 | 343 | 12.83 |
| A5 | 713 | 4.77 |
| A6 | 8593 | 1.23 |
| Pos = failures (percentage over n) | | |

After the preprocessing phase, sequences were sampled 1000 times using Monte carlo methods to obtainn training sets (60%) and test sets (40%). The proposed method was then applied as explained in Section III.

To compare the performance of the proposed model with SVMs, we applied SVM with linear, polynomial and radial basis kernel. The cost of misclassifying points was 100 in each case, to force the creation of a more accurate model.

- *>$"B,#"*

Table V shows the results of the proposed approach and compares its performance with the best performing SVM.

Our approach shows TNR higher than 0.70 in five cases. SVMs have almost perfect TNR: data are imbalanced, thus SVMs classify almost all instances as negative. Some specificity (i.e., TNR) is lost in our approach to improve the TPR.

TABLE V. Classification performance.

| App | Classification Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | TPR (%) | | TNR (%) | | FPR (%) | | Balance | |
| | Cox | SVM | Cox | SVM | Cox | SVM | Cox | SVM |
| A1 | 0.49 (0.25) | 1.00** (0.00) | 0.52 (0.20) | 0.00** (0.00) | 0.51 (0.25) | 0.00** (0.00) | 0.46 (0.03) | 0.29** (0.00) |
| A2 | 0.75 (0.12) | 0.98** (0.01) | 0.97 (0.08) | 0.91** (0.03) | 0.25 (0.12) | 0.02** (0.01) | 0.82 (0.08) | 0.93** (0.02) |
| A3 | 0.72 (0.14) | 0.98* (0.05) | 0.05 (0.09) | 0.04* (0.09) | 0.28 (0.14) | 0.02* (0.05) | 0.29 (0.07) | 0.32* (0.07) |
| A4 | 0.70 (0.12) | 0.96** (0.02) | 0.38 (0.19) | 0.33** (0.11) | 0.31 (0.12) | 0.04** (0.02) | 0.50 (0.11) | 0.52** (0.08)* |
| A5 | 0.93 (0.13) | 0.99* (0.01) | 0.86 (0.11) | 0.73* (0.18) | 0.07 (0.13) | 0.01* (0.01) | 0.88 (0.11) | 0.81* (0.13) |
| A6 | 0.79 (0.16) | 0.99* (0.01) | 0.78 (0.18) | 0.51* (0.12) | 0.21 (0.16) | 0.01* (0.01) | 0.76 (0.14) | 0.65* (0.08) |
| * Linear kernel; ** Radial kernel | | | | | | | | |

Fig. 4 show the ROC space of the proposed approach. The mean values of TPR and FPR of the 1000 experiments are plotted together with their standard deviation. Five points fall in the upper left triangle; thus the corresponding classifiers perform better than random. Three points are left of the descending diagonal (0,1)–(1,0); thus, these three classifiers perform better on the negatives than on the positives. It emerges from the comparison of this plot to Fig. 3 that A1 and A3 have now left the (0,0) point and approach the (0,1) point.

Fig. 5 shows that balance obtained with the proposed approach is higher than 0.50 in four out six cases, as for SVMs. Balance increases in three cases when applying our approach (in return of the above mentioned slight loss in TNR).

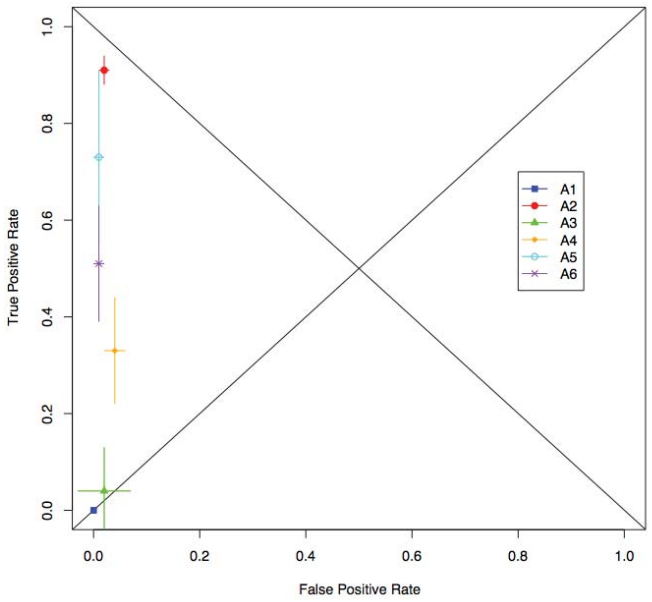


Figure 3. ROC space of the best performing SVMs.

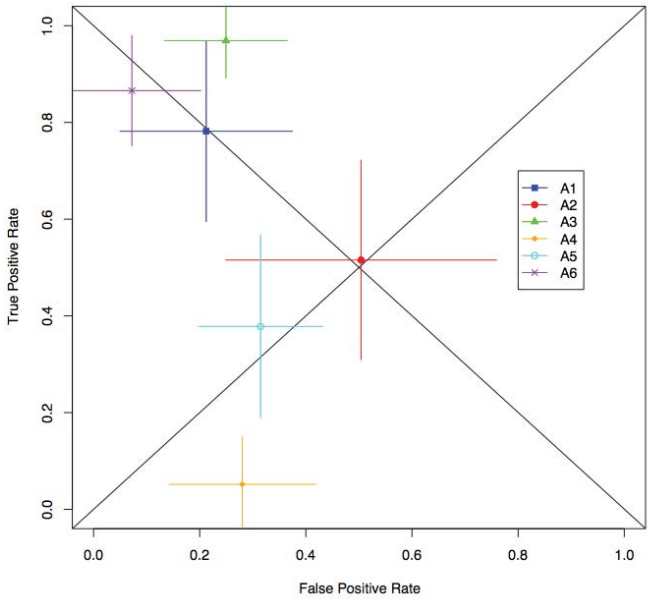


Figure 4. ROC space of the proposed approach based on the Cox PH model.

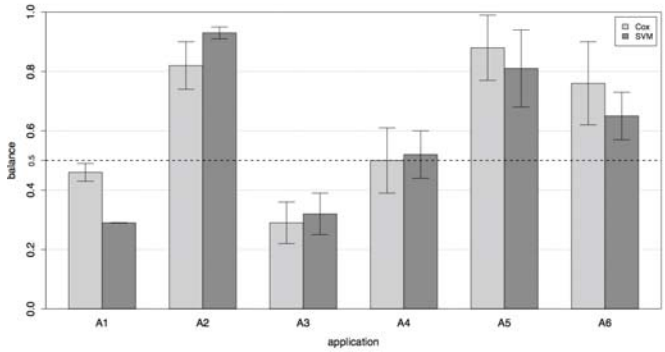


Figure 5. Balance of the proposed approach based on the Cox PH model and of SVMs: a comparison.

## V. Conclusions

In this paper we introduce a new approach for predicting failures of a running system using log files; in particular, our approach is based on the Cox PH model. Each sequence of operations is assigned a risk score evaluated according to the exponential value of a linear combination of the multiplicity of the operation, weighted by the regression coefficients derived from the aforementioned Cox PH model. Classification as failure/non-failure is based on this risk score.

It emerges from the comparison with SVMs performance that the proposed approach increases balance between TPR and FPR in three cases. This result is gained in return of a slight loss in TNR which is almost perfect in SVMs. TNR may be so high because data are imbalanced; thus SVMs tends to classify everything as negative. Our approach improves on TPR inevitably sacrificing some specificity. Moreover, the approach based on Cox model gives good performances in the cases where SVMs give their worst results.

Overall, results from our experiments appear interesting and worth further investigations. Our goal now is to study more in-depth our promising model to determine if we can generalize our results. To this end, we plan to replicate the analysis on more industrial datasets. We will also work on broader comparison with other prediction methods to identify the specialities of our approach.

Another aspect that we will evaluate is the possibility of predicting the occurrence of a failure analysing only an initial portion of a sequence, so that there could be an early estimation of failure, providing additional time to take corrective actions. To this end, Cox PH model could be applied to predict survival and the estimate hazard ratio for the various features of the model may contribute to predict the "most dangerous" failures.

Finally, we are now investigating how we could consider other “black-box” properties or applications to predict failures; candidate properties include memory usage, number of open files, processor usage.

The proposed model could be particularly useful dealing with autonomic systems. Autonomic systems could be instructed to receive signals of likely failures and upon reception of such signals could start a suitable recovery procedure [18].

## Acknowledgments

We acknowledge gratefully the support of the Free University of Bolzano/Bozen, and of the Province of South Tyrol.

## References

[1] Adiga, N., et al. 2002. An Overview of the bluegene/l supercomputer. In *28(<$$5+;)"%(9% B6$8<(%6B#+;)*, 2002.

[2] Agerbo, E. 2007. High income, employment, postgraduate education, and marriage : a suicidal cocktail among psychiatric patients. *8</+@$"%(9%I$;$8:,%2"!</+:#8!*, 64, 12, 2007, 1377-1384.

[3] Barros, C.P. and Machado, L.P. 2010. The length of stay in tourism. *;;:,"%(9%.(B8+"%%>$"$:8</*, 37, 3, 2010, 692-706.

[4] Benda, B. 2005. Gender differences in life-course theory of recidivism: A survival analysis. *J;#$8;:#+(;:,%K(B8;:,%(9%L99$;5$8% ./$8:6!%:;5%0(%6:8:#+@$%08+%+;(,()!*, 49, 3, 2005, 325-342.

[5] Bøvelstad, H.M., Nygård, S., Størvold, H.L., Aldrin, M., Borgan, Ø., Frigessi, A., and Lingjærde, O.C. 2007. Predicting survival from microarray data a comparative study. *-+(+;9(8%:#+<"*, 23, 16, 2080–2087.

[6] Coman, I.D., Sillitti, A., and Succi, G.: A case-study on using an Automated In-process Software Engineering Measurement and Analysis system in an industrial environment. In *28(<$$5+;)"%(9% #/$ % J;#$8;:#+(;:, % 0(;9$8$;<$ % (; % (9#D:8$ % M;)+;$$8+;)%* (Vancouver, Canada, May 16-24 May, 2009).

[7] P.A. Flach. 2003. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In*%28(<$$5+;)"% (9%#/$%HN#/%J;#$8;:#+(;:,%0(;9$8$;<$%(;%4:</+;$%'$:8;+;)*, 2003.

[8] Fu, S. and Xu, C.Z. 2007. Quantifying temporal and spatial fault event correlation for proactive failure management. In *28(<$$5+;)"%(9% !%6("+B%%(;%>$,+:G,$%:;5%F+"#8+GB#$5% !"#$%"*, 2007.

[9] Fu, S. and Xu, C.Z. 2007. Exploring event correlation for failure prediction in coalitions of clusters. In *28(<$$5+;)"&(9&#/$&J;#$8;:#+(;:,&0(;9$8$;<$&(;&3+)/&2$89(8%:;<$&0(%6B#+;)?&C$#D(80+;)?& #(8:)$&:;5& ;:,!"+"* (Reno, NV, USA, Nov. 15-21, 2007).

[10] Fulp, E.W., Fink, G.A., and Haack, J.N. 2008. Predicting computer system failures using support vector machines. In *28(<$$5+;)"&(9&#/$&E"#&0(;9$8$;<$&(;& ;:,!"+"&(9&"!"#$%&,()"*, 2008.

[11] Gross, K.C., Bhardwaj, V., and Bickford, R. 2002. Proactive Detection of Software Aging Mechanisms in Performance Critical Computers. In *28(<$$5+;)"&(9&#/$&HP#/ ;;B:,&C &I(55:85&(9#D:8$&M;)+;$$8+;)&Q(80"/(6*, 2002.

[12] Hao, K., et al. 2009. Predicting prognosis in hepatocellular carcinoma after curative surgery with common clinicopathologic parameters. - *4 0&0:;<$8*, 9, 2009, 398-400.

[13] Kalbfleisch, J.D. and Prentice, R.L. 2002. *./$&"#:#+"#+<:,&:;:,!"+"&(9&9:+,B8$&#+%$&5:#:*. Wiley, 2nd edition.

[14] Li, Z., Zhou, S., Choubey, S., and Sievenpiper, C. 2007. Failure event prediction using the Cox proportional hazard model driven by frequent failure sequences. *JMM&.8:;":<#+(;"*, 39, 3, 2007, 303-315.

[15] Liang, Y., Zhang, Y., Xiong, H., and Sahoo, R. 2007. Failure prediction in ibm bluegene/l event logs. In *28(<$$5+;)"&(9&#/$&J;#$8;:#+(;:,&0(;9$8$;<$&(;&F:#:&4+;+;)*, 2007.

[16] Mannila, H., Toinoven, H., and Verkamo, A.I. 1997. Discovery of frequent episodes in event sequences. *F:#:&4+;+;)&:;5&R;(D,$5)$&F+"<(@$8!*, 1, 1997, pp. 259 – 289.

[17] Menzies, T., Greenwald, J., and Frank, A. Data Mining Static Code Attributes to Learn Defect Predictors. *JMMM&.8:;":<#+(;"&(;&(9#D:8$&M;)+;$$8+;)*, 33, 1, 2007, 2-13.

[18] Müller, H.A., Kienle, H.M., Stege, U. 2009 Autonomic Computing: Now You See It, Now You Don't—Design and Evolution of Autonomic Software Systems. In: De Lucia, A., Ferrucci, F. (eds.): Software Engineering International Summer School Lectures: University of Salerno, LNCS 5413, Springer-Verlag, 32–54.

[19] Pandalai, D.N. and Holloway, L.E. 2000. Template languages for fault monitoring of timed discrete event processes. *JMMM&.8:;":<#+(;"&(;& B#(%:#+<&0(;#8(*, 45, 5, 2000, pp. 868 – 882.

[20] Salfner, F. 2008. *M@$;#SG:"$5&*:+,B8$&28$5+<#+(;T& ;&M1#$;5$5&3+55$;&4:80(@&4(5$,& 668(:</*. Berlin, Germany: Dissertation, Verlag.

[21] Sampath, M., Sengupta, R., and Lafortune, S. 1994. Diagnosability of discrete event systems. In *28(<$$5+;)&(9&#/$&EE#/&+;#$8;:#+(;:,&<(;9$8$;<$&(;& ;:,!"+"&:;5 L6#+%+U:#+(;&(9&!"#$%"&F+"<8$#$&M@$;#& !"#$%*s (Sophia, Antipolis, June 15 – 17, 1994).

[22] Sillitti A., Janes A., Succi G., and Vernazza T. 2004. Measures for Mobile Users: an Architecture. *K(B8;:,&(9& !"#$%"& 8</+#$<#B8$*, 50, 7, 393 - 405.

[23] Scotto M., Sillitti A., Succi G., and Vernazza T. 2006. A Non-Invasive Approach to Product Metrics Collection. *K(B8;:,&(9& !"#$%"& 8</+#$<#B8$*, 52, 11, 668-675.

[24] Srinivasan, V.S. and Jafari, M.A. 1993. Fault detection/monitoring using time petri nets. IEEE Transactions on System, Man and Cybernetics, 23, 4, 1993, 1155 – 1162.

[25] Schroeder, B. and Gibson, G.A.Understanding failures in petascale computers. *K(B8;:,&(9&2/!"+<"*, 78, 2007.

[26] Stearley, J. and Oliner, A.J. Bad words: Finding faults in Spirit`s syslogs. In *28(<$$5+;)"&(9&#/$&J;#$8;:#+(;:,& !%6("+B%&(;&0,B"#$8&0(%6B#+;)&:;5&#/$&I8+5* (Lyon, France, May 19-22, 2008).

[27] Wedel, M., Jensen, U., and Göhner, P. 2008. Mining software code repositories and bug databases using survival analysis models. In *28(<$$5+;)"&(9&#/$&H;5& 04SJMMM&+;#$8;:#+(;:,&"!%6("+B%&(;&M%6+8+<:,&"(9#D:8$&$;)+;$$8+;)&:;5&%$:"B8$%$;#&* (Kaiserslautern, Germany, October 09 - 10, 2008).

[28] Xue, Z., Dong, X., Ma, S., and Dong, W. A survey on failure prediction of large-scale server clusters. In *28(<$$5+;)"&(9&#/$&J;#$8;:#+(;:,&0(;9$8$;<$&(;& (9#D:8$&M;)+;$$8+;)?& 8#+9+<+:,&J;#$,,+)$;<$?&C$#D(80+;)?&:;5&2:,,$,VF+"#8+GB#$5&0(%6B#+;)*, 2007.

[29] Yamanishi, K. and Maruyama, Y. Dynamic syslog mining for network failure monitoring. In *28(<$$5+;)"&(9&#/$&J;#$8;:#+(;:,&0(;9$8$;<$&(;R;(D,$5)$&F+"<(@$8!&+;&F:#:&4+;+;)*, 2005, pp. 499-508.

[30] Yanaihara, N., et al. 2006. Unique microRNA molecular profiles in lung cancer diagnosis and prognosis. *0:;<$8&0$,,*, 9, 3, 2006, 189-198.

[31] Yu, S.L. et al. 2008. MicroRNA Signature Predicts Survival and Relapse in Lung Cancer. *0:;<$8&0$,,*, 13, 1, 2008, 48-57.