Control and Cybernetics

vol. **42** (2013) No. 1

Creating a knowledge database on system dependability and resilience*

by

Marcin Kubacki and Janusz Sosnowski

Institute of Computer Science, Warsaw University of Technology,
ul. Nowowiejska 15/19, 00-665 Warszawa, Poland
M.Kubacki@stud.elka.pw.edu.pl, J.Sosnowski@ii.pw.edu.pl

Abstract: The paper deals with the problem of creating a knowledge database on system dependability and resilience, created on the basis of available system and application logs. Special tools to collect and analyse these data from many systems have been developed. Taking into account a wide spectrum of various logs we explore them locally and globally. This allowed for identification of characteristics of normal operation and anomalous behaviour. A lot of attention is paid to the problem of selecting measures to identify symptoms characterising system operation and their usefulness in dependability and resilience evaluation or prediction. The concepts presented are illustrated with experience gained during monitoring of real systems.

Keywords: dependability, data mining, event and performance logs, resilience

1. Introduction

High dependability and system resilience are the features gaining recently much attention in various systems (e.g., Simache and Kaaniche, 2005; Stoicescu et al., 2011; Ye, 2008). They gave a significant impact on developing on-line system monitoring at different levels, starting from the micro architectural to application level (e.g., Chen et al., 2012; John, 2006; Li et al., 2012; Magalhaes and Silva, 2011). Most systems comprise some standard monitoring tools targeted at various events or performance parameters; which results in multitude of dedicated logs often with inherent unstructured data. They can be useful in characterizing system operational profiles, detection and predicting their anomalies, although this is a complex task (see, e.g. Chandola et al., 2009; Li et al., 2011; Salfiner et al., 2010). In the literature, there are many publications devoted to system log analysis. Usually they deal either with event (e.g. Chen et al., 2012; Cinque et al., 2009; Naggapan and Vouk, 2010; Simache and Kaaniche, 2005)

*Submitted: July 2012; Accepted: February 2013

or performance logs (e.g., Gmach et al., 2007; Hoffman et al., 2007; Ye, 2008) and relate to specific systems or problems, so the presented results differ significantly. The methods of analysis, proposed there (coarse-grained) mostly focus on some failure patterns and specific known target event types. In practice, we face also the problem of extracting unknown features describing normal and anomalous operation. This confirms the need of deeper (fine-grained) studies of computer systems used, taking into account their operational features.

We have launched a project of monitoring computers and servers used in our Institute. As opposed to classical approaches, we take into account simultaneously event, performance and other logs, moreover, we analyse them in different time perspectives and use some data aggregation schemes. On the bases of our previous experience (Król and Sosnowski, 2009; Latosiński and Sosnowski, 2012; Sosnowski et al., 2006, 2010, 2012) with selected system availability problems (downtimes, performance losses), we have elaborated a generalized and wide scope methodology taking into account a large space of logged data and new measures. In particular, we deal with the morphology and various dependencies of logged events as well as the correlations between various log data dimensions (neglected in the literature) and a wide spectrum of problems. The developed analytical methodology is supported with new tools, combined with available data mining and statistical modules. This approach facilitates tracking of symptoms of various anomalies and creating the relevant knowledge database.

Section 2 presents the scope of system monitoring. Sections 3 and 4 deal with event and performance logs, respectively, and describe the methodology of system operation characterisation using the developed tools. Section 5 discusses integration of developed tools with the created database. Final conclusions are given in Section 6.

2. The scope of monitoring

Dependability and resilience have become important features of many computer systems. Dependability relates to such attributes as reliability, availability, safety, security, etc. Rresilient systems assure correct services despite environmental changes, application modifications, changes of workload profiles, etc. Various monitoring techniques can identify symptoms of failures, anomalies, and operational threats which need some preventive actions (Cinque et al., 2009; John, 2006; Li et al., 2012; Oliner and Stearley, 2007). System behaviour can be observed from different perspectives, e.g. user, administrator, operating system, application, etc. In the first two cases, we deal with some subjective exploitation and evaluation remarks. Unfortunately, in practice they are rarely logged or registered, and quite often they lead directly to some actions to eliminate the problem, etc. The remaining observation perspectives are usually supported with various tools, particularly for registering the appearing events or the pre-programmed performance measures.

Events relate to some specific and well defined situations, which are detected in the system (Naggapan and Vouk, 2010; Sosnowski and Poleszak, 2006), and

provide some general view on system operation. Computer systems are instrumented to provide various event logs on their operation. These logs comprise huge amounts of data describing the status of system components, operational changes related to initiation or termination of services, program updates, configuration modifications, execution errors, etc. The formats of registered events have some loosely defined general scope (Sosnowski et al., 2006, 2012; Vaarandi, 2003); in particular, we can distinguish various data fields containing specific information in textual or numerical form with some specific brackets, etc. Some fields can be considered to be parameters. In particular, we have the time stamp (the time of event registering), name of the event source (e.g. disc, application program, process identifier - PID, etc.), text describing the related event problem, severity of the problem, etc. Events generated by different sources can be stored in common or different log files (e.g. security events specifying authorisation problems, user login and logout events, events generated by applications, etc.). The included texts (usually unstructured) can be very general, of low information value, or more substantive. Nevertheless, their meaning can be better interpreted after gathering some practical experience from many computers and within a longer time perspective.

In most computer systems, various data on performance can be collected in appropriate counters and according to some sampling policy, e.g. in 1-minute periods (John, 2006; Król and Sosnowski, 2009; Ye, 2008). These counters are correlated with performance objects such as processor, physical memory, cache, physical or logical discs, network interfaces, server of service programs (e.g. web services), I/O devices, etc. For each object, many counters (variables) are defined characterising its operational state, usage, activities, abnormal behaviour, performance properties, etc. Special counters related to developed applications can also be added. All these counters provide data useful for evaluating system dependability, predicting threats to undertake appropriate corrective actions, etc. The list of counters, which can be configured, is very long. In the Windows-based systems counters can be programmed for hundreds of variables (Ye, 2008). Using these counters involves some CPU overhead, hence an important issue is to select a representative set of measures (variables).

The primary goals of system monitoring can be identification of failures, anomalies or their threats, evaluation of various trends (e.g. resource usage, workload profiles), etc. These problems can be analysed from the operating system, application or user perspectives. Here we should also consider possible environment and workload changes which may impact system behaviour. These changes can be considered as normal trends, which should be detected and used in upgrading and adapting the system to new challenges (resilient systems). Recently, high complexity of hardware and software creates various inconsistency and misconfiguration problems, which are nondeterministic and vague in their nature. To deal with these problems we use quite sophisticated tools and explore (locally and globally) all available logs in various observation perspectives. Moreover, we systematically collect the knowledge on anomaly symptoms (referenced to normal system operation) in the monitored space.

3. Analysing event logs

Depending upon the goal of the log analysis (e.g. error detection and diagnosis, finding operational profiles or trends, identifying anomalies), we are looking for different events or their sequences. In many cases, this is reduced to a search for a well specified event category like reboots, errors, warnings (Sosnowski and Poleszak, 2006). In practice, we have an enormous amount of events and insufficient knowledge on their formats and meaning, so some exploration and data mining processing is needed. Systematic analysis is targeted at two aspects: (i) event classification and morphology analysis; and (ii) identification of various characteristic features in different time and space perspectives. In particular, we can try to identify event classes. Many logs give some specification of event type (e.g. warning, error, information) but this is a very coarse classification and usually not precise. More precise classification can be performed with regular expressions by defining related patterns. Here it is worth noting that application logs may have different and unique formats (e.g. in CSV or XML). In the case of some application logs, we can do this manually. For example, in a mail server (Latosinski and Sosnowski, 2012) with the popular *sendmail* program we have identified 52 event classes with the following type distribution: 24 - info, 18 - notice, 2 - debug, 2 - alert, 3 - warning, 3 - error. The programs responsible for access protocols (*pop3, pop3d, pop3ds, imapd, impads*) generated 27 event classes to the *mail.log* and 2 classes to *auth.log*. Syslog and other programs generated 61 different event classes stored in *auth* and *security* logs. It is also useful to analyze service logs (e.g. emails generated in correlation with some events).

To deal with any event logs, we have developed a hierarchical classification algorithm which identifies static and variable (parameter) fields within log lines that can be applied to preprocess log entries with partial generalizations introduced by regular expressions. This classification is supported with event message analysis. For this purpose, we use some text mining techniques (abstracting events with semantic similarities, checking word frequencies, positions and contexts). Having identified parameters (variables) in the logs, we can also recognize their types, e.g. I/O port, file path, web links and include them in the abstracted event patterns (Sosnowski et al., 2012). This technique surpasses the typical log abstraction approaches (Naggapan and Vouk, 2010) restricted to identification of variable data fields and is simpler than clustering approaches (e.g. Fu et al., 2009, 2012; Vaarandi, 2003). When analyzing logs for various computer platforms (Windows, Linux) and systems (laptops, workstations, servers and a cluster system), we have gained some knowledge on event morphology, dependencies and significance. Using this technique in a cluster subsystem composed of 30 nodes (with Linux), we have started with 248 368 syslog raw events (excluding cron) related to 4 months period. By abstracting the time stamp and PID (replaced by wildcard (*)), we have got 117 294 different events. Upon applying to this reduced set our algorithm, we got 12 624 event classes, and by abstracting nodes we got 5 197 classes. Further refinements resulted in

5 589 and 383 event classes, depending upon the algorithm used (Sosnowski et al., 2012). The analysed log comprised 2 502 073 words (some included digits and other special symbols); within this set we have detected 116 213 unique words, among them 981 words, which do not comprise digit characters. The number of different word positions in the message field was up to 25. These numbers reflect the data mining complexity.

For some selected computers running under Windows, we have got about 300 000 events in the application logs which have been classified into 141 event classes. This classification was a little bit easier than for the Linux systems, due to unique specification of event sources and event IDs in logs. Nevertheless, the text mining was not trivial. For example, in a computer with over 44 000 application events we got 1 571 different words (the total of 710 964 words). Quite frequent words were "error" in English, 14 041, and its equivalent in Polish ("błąd") – 11 826, "updating" – 11 337, whereas potentially critical events comprising "failed" and "failure" corresponded to 14 and 3 cases. Logs with "error" word in most cases were not critical, more critical were logs with terms "not able", "unsuccessful". For the system log, we got 739 different words within a total of 447 212 words, which can be correlated with some problems, e.g. related to performance, service, and software actualization. Such semantic analysis is helpful in searching interesting events in a wide search space. It can be enhanced with checking links within messages or service logs.

Obviously, it is difficult to trace logs manually. On the one hand, we have different types of events, sometimes with some parameters of more or less important value. On the other hand, it is important to know the source of event, its location in time and space (e.g. node number). So, in the analysis we are interested in event profiles, various statistics of global or filtered event sets (according to various attributes), raw or preprocessed (normalized) events, abstracted (classified with regular expressions) events, etc. It is reasonable to identify frequent (dominating) events - crossing a specified support threshold (Vaarandi, 2004), critical events (characterizing known problems), emerging events (rarely reoccurring events) and jumping events (new and suspected). For this purpose, we have developed ELogAnalyser and some other supplementary tools. They use special agents installed in the monitored systems, which fill up the associated database. This assures parallel monitoring and analyzing many computers or nodes in a cluster.

Having analysed the morphology of event logs (including event classification - abstraction), we can perform more detailed studies to identify critical, anomalous states or to find the characteristics of the operational profile. This process is supported with appropriate visualization tools. In the developed ELogAnalyser, we have the possibility of visualizing events in various time perspectives, filtered (e.g. based on regular expressions) or aggregated. For example, hour, daily or monthly perspectives show event distribution over the related subsequent time periods (events are summarized within these periods). Aggregated hour views show distribution of events within 24 hours, for each hour we sum all events of this hour for each day of the considered period. Similarly, we can

generate aggregated daily distribution over all week days (summarized events for each week day over the considered period). Aggregated distributions show activity profiles (or trends) during day and night hours, week days (e.g. low activity on weekends) or months.

The generated distributions can be limited to specified event classes (e.g. errors, warnings) or unique events. Moreover, they can be presented for individual computers/ nodes (on single or independent plots) or summarized over a selected group of nodes, etc. For illustration, in Fig. 1 we give some plots related to a cluster subsystem (Sosnowski et al., 2012). Fig. 1a gives the distribution of NTP (Network Time Protocol) synchronization actions within one month for some set (space) of nodes. It shows different clock stability in nodes, for some of them more actions have been activated, 10 days of no activity related to the period of switched-off NTP server. Fig. 1b shows problems with the file system over several nodes. The first and the third high value pulses refer to a disc array problem, the second one is related only to 3 nodes and was a local problem. Figs. 1c and 1d show the aggregated distribution of all errors in individual nodes (weekly perspective) and summarized over all nodes (day hour perspective), respectively (the aggregation process takes results for about 100 days). Here, low activity is visible for weekends and night hours.

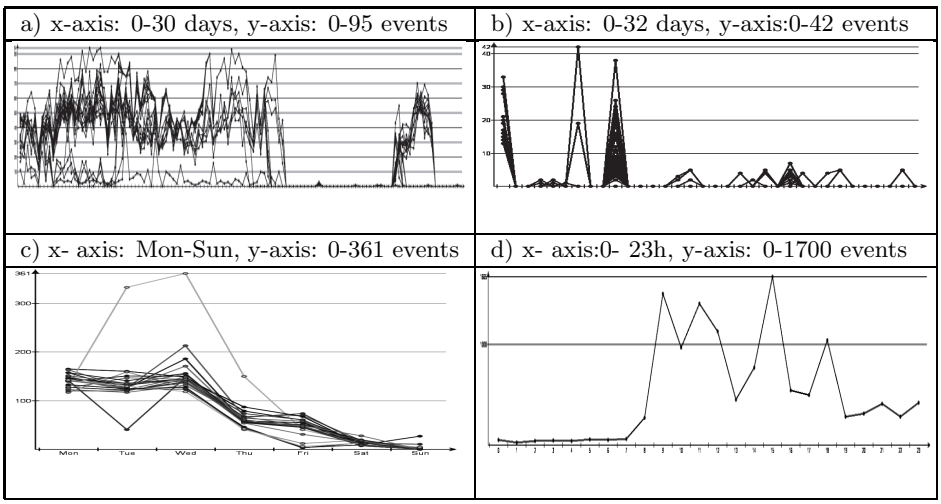

Figure 1. Event distributions: a) synchronization events for individual nodes, b) file system errors, c) weekly profile of all errors for individual nodes, d) hourly profile of errors over all nodes

An important issue is tracing distribution of events in time and space as well as their correlations. We have analyzed distribution of times between events, distribution of the number of registered events in specified time periods. For example, in a laptop we have registered 42 333 (within 2 years) application

events (25 656 errors, 400 warnings, 16 277 specified as information). For 15 810 sampling periods (1s), we have identified at least one event, more than 14 790 periods comprised 1-4 events, 977 comprised 5-15 events, and 97 more than 16 events, the maximum number was 74. Taking into account the fact that on average we have about 20 events per day, which are concentrated within working hours, we get on average 20 events per 4 hours of system operation. Events followed by another event within 2 seconds contributed 31 000 events. Hence, bursts of events can relate to some specific situations, in particular we have got 15 samples with more than 30 events (in one second); such cases are worth deeper analysis and correlations with other logs.

Event distribution and event sequences need special attention. We have analysed distribution of events, their pairs and longer sequences. We have considered event classes specified with the source and ID number. To illustrate the significance of this analysis we give some results for one computer. We have got 42 different sources and 152 different event IDs within the computer application logs, which resulted in 219 different event classes. The total number of event pairs was 42 332, of which only 1 305 were different. Deeper analysis showed 207 pairs with at least 0.5 confidences (the probability that the first element implies the second one). The pair <Application error 1000, error report 1001> had confidence 0.86. There were several pairs with confidence 1.0 related mostly to checking program licences or rollbacks. For many (536) pairs, the support (number of appearance) was only 1, for 242 pairs the support was over 10, within this 2 pairs with support exceeding 10 000 and 30 exceeding 150. Event pairs can be visualized as a graph with nodes related to each event class and edges corresponding to event pairs. On average, each event was followed by one of 5.3 different events. An event was a predecessor of maximally 43 (node fan-out) other events and the successor of maximally 39 events (node fan-in). We have also identified some longer unique event sequences (4-12 events), e.g. related to checking software licences/program originality, successful or unsuccessful updates. These observations confirm some regularities in logs and relatively high event correlations. This feature also appeared in other logs and computers. For example, for the considered computer the number of different event pairs in syslog was only 287. For another computer with lower number of used applications, we have got 191 and 88 different event pairs for applications and system logs, respectively.

Distribution, support and confidence of event pairs are some characteristics of operational profiles. Changes in these profiles, especially appearance of some new event pairs (e.g. caused by some attempts of attacks) or unexpected ones (e.g. comprising restarts), are worth deeper analysis. Appearance of new events or pairs can be attributed to profile changes (e.g. new applications) or to some anomalies. Similarly, we can interpret disappearance of some events, pairs or sequences. Correlations between events of different logs (global perspective) were in general weaker than within single logs (local perspective). However, higher level correlations can also be identified, like those related to different users within the considered computer (correlation with last log giving information on

logins and logouts) or with times of different student courses within student laboratory computers. We have identified some errors mostly triggered by a small group of students e.g. related to pen drive usage (incorrect disconnections or any read errors due to device incompatibility) or unauthorised access tries. Some users were correlated with higher number of errors resulting from their superficial knowledge of applications, etc. An important issue in this analysis is presentation of events in different perspectives with the capability of various filtering and aggregating features assured by the developed tool.

4. Analysing performance logs

When analyzing performance parameters we investigate their behavior in time, explore distribution of their values, check correlations between different parameters, etc. For this purpose, we can use appropriate statistical and data mining tools, in particular the specially developed tool EPLogAnalyser and the IBM SPSS Modeller. Using SPSS we have designed appropriate scenarios of data analysis. We have concentrated on three scenarios: auditing the collected data (DA), finding performance models (PM), and finding time series prediction models (TPA).

For each scenario, we have configured in SPSS an appropriate scheme of data flow and processing (composed of selected standard components and connections between them). The DA scenario allowed us to derive statistical properties of the collected data samples for each performance variable as well as correlations between different variables. Table 1 provides a sample of statistics of some application. They include the following variables: CPU usage in percent (CPU), memory utilization in MBytes (total - MU, for applications - MUA, for swaps - MUS), disc activity (the number of disc read blocks - DiscR, the number of disc written blocks – DiscW), CPU average load (LAv - average length of the process queue), the number of received (NRB) and sent (NSB) bytes in Ethernet network. Usually, we generate statistics (minimal, maximal, average, median, dominant values, standard deviation, standard error of the average value, etc.) of raw data and then perform some refinement. In particular, we can reject samples exceeding some specified multiple of standard deviation or filter improbable values (e.g. due to measurement faults) or delete samples related to some specific states (e.g. system maintenance actions, reconfigurations). The statistics of maximal (Max), average (Av1) and standard deviation (Dev) in Table 1 relate to refined data (after deleting from the raw data samples with values exceeding 5 times the standard deviation). To show the impact of this refinement, in column Av2 we give average values of the raw data and in columns Ex1 and Ex2 the number of samples exceeding 3 and 5 times the standard deviation. The total number of samples for each variable was about 150 000 (sampling rate 1 minute). On the basis of the refined data we have generated Pearson correlation factors (Hill and Lewicki, 2006) between the analyzed variables. They are given in Table 2.

In practice, auditing the collected data allows us to select the most interest-

Table 1. Statistics of selected performance measures

	Max	Av1	*Av2*	Dev	*Ex1*	*Ex2*
CPU	79.3	8.18	*8.18*	8.986	*1768*	*77*
DiscR	24128	912.5	*2194.333*	3496.629	*0*	*1*
DiscW	32184	7081.68	*7208.801*	4924.037	*4717*	*309*
LAv	2.558	0.23	*0.23*	0.27	*3275*	*274*
MU	1345	1217.405	*1319.27*	125.374	*1*	*84*
MUA	1245.6	860.805	*924.864*	173.488	*102*	*0*
MUS	3032	305.976	*332.686*	131.5	*6*	*285*
NRB	1517559	12419.463	*13810.743*	27986.337	*53*	*354*
NRW	832365	13445.785	*16272.712*	30615.427	*14*	*106*

Table 2. Pearson correlation coefficients for the analyzed performance variables

	CPU	DiscR	DiscW	LAv	MU	MUA	MUS	NRB	NSB
CPU	-	0.530	0.719	0.544	0.260	0.251	-.001	0.887	0.848
DiscR	0.530	-	0.764	0.476	0.192	0.188	0.013	0.249	0.233
DiscW	0.719	0.764	-	0.513	0.229	0.215	0.021	0.600	0.601
LAv	0.544	0.476	0.513	-	0.291	0.286	-.015	0.411	0.398
MU	0.260	0.192	0.229	0.291	-	0.846	0.012	0.201	0.192
MUA	0.251	0.188	0.215	0.286	0.846	-	0.025	0.190	0.179
MUS	-.001	0.013	0.021	-.015	0.012	0.025	-	-.042	-.051
NRB	0.887	0.249	0.600	0.411	0.201	0.190	-.042	-	0.931
NRW	0.848	0.233	0.601	0.398	0.192	0.179	-.051	0.931	-

ing performance parameters and reject those of lower interest, e.g. not showing fluctuations of values. Moreover, we can optimize the set of parameters by reducing those which are highly correlated (e.g. 99%). This is an important issue, because the list of possible variables could be very long (Sosnowski and Król, 2010; Ye, 2008).

The goal of PM scenario is to find a model of predicting the value of a selected performance variable in the function of other variables. Within SPSS (see IBM, 2011), we have many available methods to generate such models for the collected data, e.g. linear regression, decision trees, neural networks, C&R trees (classification and regression trees), KNN (k nearest neighbor). We have concentrated on finding such models for CPU usage. For the considered application, we obtained very good models (based on 50% of raw data – the training sets) with correlation factors in the range from 0.940 to 0.972 and relative errors in the range 0.056-0.117. Getting such results needed appropriate configuration of the generators (selection of appropriate coefficients and constants). It is worth noting that the C&R model is quite intuitive, as it generates a tree which gives some idea on interpreting dependencies among variables. For the considered example, we have got a tree with depth 5, 19 nodes and 8 leaves, on the basis of 30% raw data samples.

For another project, we have found some fluctuation of basic statistics and Pearson coefficients for subsequent months. The derived SPSS prediction model for CPU load average based on data from 6 months showed higher error (0.48) and lower correlation (0.718). This was caused by monthly fluctuations of Pearson correlation coefficients related to CPU usage (0.7-0.8) and disc operations (0.23-0.49). In the previously analyzed system higher stability was observed. Moreover, the CPU usage models are usually more accurate than those of CPU load. Taking into account fluctuations of some variables in time we have constructed also models with an additional time variable. Quite good results were obtained using such variable with periodicity 24 hours, taking into account higher system activity during working hours and days.

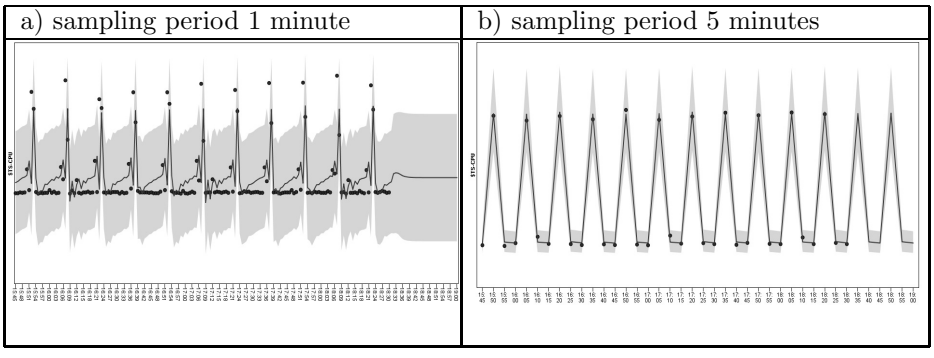

Figure 2. Example of LAv prediction with time series (x-axis range: 18.45-19.00)

TPA modeling is targeted at tracing the behavior of performance parameters in time and predicting their future behavior. This is a good approach to identify some features of normal operation, in particular, various periodic activities related to backups, program updates, etc. However, while using standard SPSS time series functionality we have encountered some prediction accuracy problems, as illustrated in Fig. 2. The predicted values of CPU load average (LAv) relate to the plotted line (derived from 15 days data). The 95% confidence range of predicted values (dashed area in Fig. 2) was consistent with the real values (dark dots), but the prediction of periodicity was problematic. Detection of periodical fluctuations (15 minutes period) depended upon the sampling period: it was skipped for 1 minute sampling period and predicted for 15 minutes sampling period (Fig. 2b – the two final pulses predicted).

Better results were obtained with the models used for checking correlations between variables (specified in Table 2). In this case, we used the CPU load average variable and correlated it with the time variable. For the considered application, the highest correlation coefficient was obtained for the KNN model (0.829), although the relative error was quite high (0.568). The problem with high relative error resulted from some time shift of the predicted and real plots. In fact this is not critical and we have developed another quality measure admitting some time shift (of pulses) and taking into account the integrated values of the real and predicted pulses.

Another interesting issue is the behavior of performance measures in time. This can be done in different time perspectives with different granularity and aggregations rules (similarly as for event logs – see Section 3). When looking for specific patterns, we can distinguish some general properties such as average values, fluctuation ranges, and negative or positive pulses, their frequency, time positions, distribution in amplitude, distribution in time, etc. It is worth noting that some patterns relate to specific system activities like backup processes, program updates, monitoring processes, administrator audits (with more or less regular scenarios).

Fig. 3a shows CPU usage (up to 47%) of a complex security monitoring program (SOA – service oriented architecture) within one node of a cluster system with averaged values for hours (lower plot relates to load average with values from 0 to 8). The visible pulses closely relate to the task scheduling (task duration is about 5h). Such characteristic of correct operation was not easy to detect in plots with higher resolution (e.g. 1 minute). To deal with higher resolution data, it is sometimes helpful to use compacted plots. This is illustrated in Fig. 3b, which shows average (horizontal line), maximal and minimal values (specified by the vertical line) of disc write operations within compacted time periods related to 20 samples (each of 1 s). Fig. 3c shows a daily perspective (the period of 12 days) of sent out bytes (samples with average values over 30 minutes) with visible weekend low value and an abnormal peak caused by an uncontrolled burst of emails. In weekly and monthly perspectives, we use average sample values over 2 hours and 1 day, respectively, which allows us to identify some regularities and trends. For instance, the monthly plot in

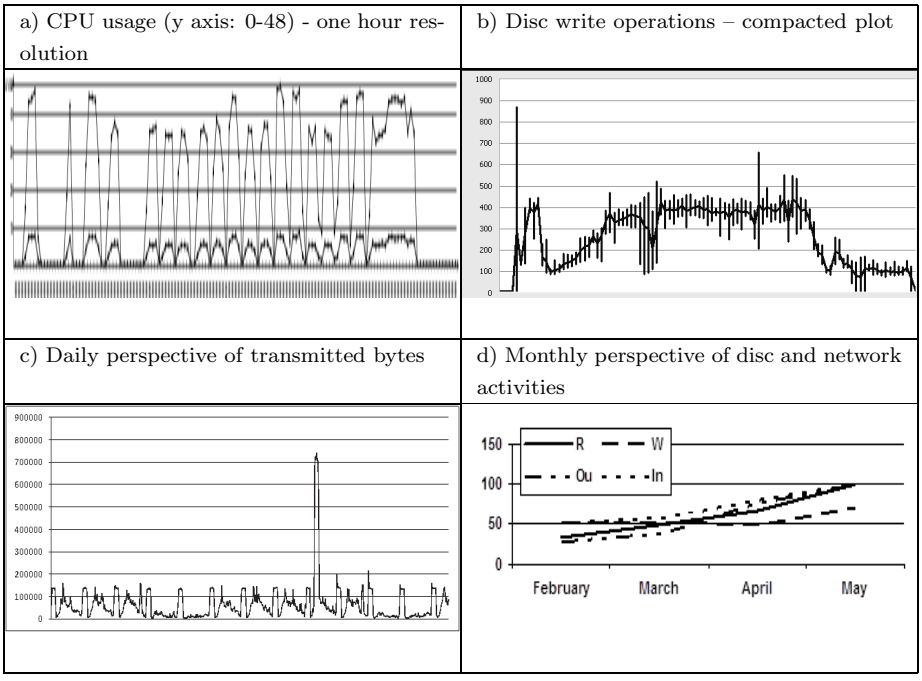

Figure 3. Visualization of performance parameters in different time perspectives

Fig. 3d shows systematically growing activity of a disc (W – write, R – read operations) and interconnections (Ou - sent out, In - received bytes) in a newly installed data repository system (trends of increasing user interest and adding new data).

Depending upon the application benchmark running in the analyzed system we may have different performance characteristics which can change within a single application, e.g. if there are well defined periods of heavy calculations and burst disc operations. We have found that in general it is reasonable to analyze performance time dependencies in correlation with event logs (compare Section 5). By analyzing performance plots, we can try to identify their models and predict their behavior for some future period to check if the real behavior differs significantly (warning situation to trigger more accurate analysis and monitoring) or to identify some lacking patterns (e.g. skipped backups or their time skews).

5. Combining event and performance analysis

In the process of identifying normal system operation features, their trends and anomalous or suspected situations, we use various monitoring schemes involving event logs (console logs, application logs, security logs, etc.) and performance logs. All these mechanisms are usually available on the system platform or are incorporated into applications. In some domains, they can be enhanced with traces of user or environment interaction with the system. Some of these activities relate to backups, maintenance actions (planned and unplanned), system restarts, unauthorized accesses of other users, etc. They could be helpful in revealing system anomalies. A specific anomalous case is non-professional application usage resulting in lowering of its performance, generating excessive memory or processor usage. This can result from low user qualification or his/her unconsciousness of possible problems. Such observations can draw attention of responsible authorities to organize special courses to the users, etc.

In general, depending upon the application, we have different available data, different possibilities of anomaly classes and severities. In the process of analysis, we can use various statistical techniques or advanced data mining and data exploration approaches (e.g., Brandt et al., 2008; Li et al., 2011; Salfiner and Malek, 2007) which can be supplemented with some measures related to information theory, spectral analysis, statistical properties, etc. Taking into account some vagueness of qualification results, it may be also reasonable to use fuzzy sets or rough set theory.

It is important to get long term experience by monitoring event logs for longer periods and on different hardware and software platforms. This monitoring should be correlated with systematic user and administrator observations, their reports on operation anomalies, occurring system crashes, power blackouts, network disconnections, system overloading, or other problems. All these situations should be described and registered in some special repository, so that they can be confronted with collected logs at the time of problem appearance or

in a postponed log analysis. Unfortunately, it is difficult to base on experience published in the literature, which usually is targeted at some unique problems, specific workloads, etc. Moreover, in the available literature (compare Section 1), the authors publish aggregated statistical results, so the important detailed knowledge on problem uncovering or manifestation is not at all available or only partially.

Having checked the collected logs from many computers, we can state that the number of registered events is quite high even in systems with low activity. In most cases, the system usually operates correctly, so identifying critical or warning situations could be to some extent problematic. Only some events are critical, on the other hand some event sequences should be correlated taking into account performance measures, workload profiles, user sessions, etc. Moreover, we may have various time dependencies, e.g. regular program updates, backups and clock synchronisations, different usage profiles in week days, in day hours, etc. Hence, we have decided to create a common database for event and performance logs to facilitate tracing abnormal or normal situations. Some parameters are stored in an aggregated form: average, maximal, minimal values, difference from the previous sample, etc. This database is used by the developed analysis tools (described in Sections 3 and 4) which are integrated in the EPLogAnalyser system.

The EPLogAnalyzer database consists of 10 tables (Fig. 4). CONFIG table keeps several global settings concerning the tool. All event log files imported into EPLogAnalyzer have a record in FILES table. Log files can be imported either from local disk or remotely from an agent. In the latter case, each file has an ID of an agent, from which data has been imported, and the access path. Each file consists of log entries stored in LOG_ENTRIES table which keeps all of the event log data records and is the second largest table in the database. To speed up "select" statements, each log entry has additional relationship with AGENTS table. AGENTS table contains connection details (port) and information about time of last performance data transfer, current performance data collection timestamp and the measure definition version. In EPLogAnalyzer, a user is able to define the format of the log file as a set of regular expressions. LOG_FORMATS table keeps only basic information like the name of the log format. Specific information for each field in the log file is kept in the LOG_FORMAT_FIELDS. This table defines a set of expressions that are used to parse the text form of the log file and to convert it into a tabular form. Typically, it is either a text specified by a regular expression (e.g. host name, PID, text message) or a date time field (using Qt framework date/time expressions). Each FILE entry must have a corresponding LOG_FORMAT record to properly import the file into EPLogAnalyzer.

EVENTS table comprises events (defined by the user) for filtering data for the analysis. Each event can be associated with only one log format. An event is defined as a set of conditions on each log format field. These conditions are kept in the EVENT_CONDITIONS table. They facilitate tracing various properties like distribution of log data on weekdays or day hours. Each event

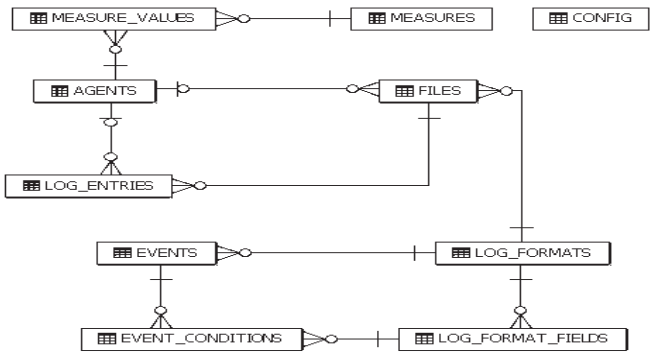

Figure 4. Structure of the EPLogAnalyser database

has exactly one event condition for each log format field (of the associated log format). For a regular expression field of the log entry the condition specifies an expression (pattern) that has to be matched with its contents (e.g. to select a log comprising word "error" or a host name). In the case of date/time fields the specified conditions must be matched for the whole log entry (e.g. month, week day, date range). All conditions defined for a specific record in the EVENT table must be matched in order to treat the particular (selected) log entry as an instance of the user defined event.

From the performance point of view, the most important tables are: MEASURES - which keeps measure definitions and MEASURE_VALUES - which is typically the biggest table in the database. The latter consists of maximum M x S records, where M is the number of measures (parameters) defined and S is the number of samples collected. For example, collecting data every minute gives 24 x 60 = 1440 samples per day for each measure and agent. When data is being retrieved by EPLogAnalyzer, sophisticated SELECT statement transposes MEASURE_VALUES table. To speed up this process, appropriate indexes have been created.

The analytical environment of EPLogAnalyzer is specified by all tables except LOG_ENTRIES and MEASURE_VALUES. In particular, it involves the list of files that have been imported, log formats, events, measures and agents defined by the user. These are typically small tables that are read during application startup and modified only, when the user changes application settings, e.g. adds a new agent or modifies a measure definition. LOG_ENTRIES and MEASURE_VALUES tables are read on demand during the analysis phase, using complex SELECT statements generated dynamically by the EPLogAnalyzer in relevance to the analysis settings. These tables are modified during data import phase, either via file import or by download of data collected by the agent.

The collected information in the database allows us to get better knowledge

on system behaviour and possible anomalies, some of which are characterized by known critical events, whereas others relate to wider contexts. Hence, in our approach we usually specify conditions (AC) of possible anomalies in space of performance measures (e.g. CPU usage) or functional properties (e.g. a suspended service). Then, we check recorded events in a pre-programmed time window (Δ) just before the occurrence of AC. This allows us to identify anomalous symptoms with appropriate level of confidence defined as follows:

$$\text{Confidence}(\text{LE}\rightarrow\text{AC}) = \text{Support}(\text{LE}\rightarrow\text{AC})/\text{Support}(\text{LE})$$

where: LE - event log entry pattern, AC - the specified measurement condition (e.g. CPU usage >x %), Support (LE) - the number of all events LE in the considered time period, Support(LE→AC) - the number of LE events followed by condition AC (the maximal correlation window time Δ is also programmed - typically 1 minute).

Simple events can also be replaced by some sequence of events. Similarly, we can use more complex AC conditions like taking into account duration of excessive parameter value or a burst of such pulses. In this approach, an important issue is specification of event patterns in some universal way (so called event abstraction – skipping of some irrelevant details in the event, e.g. time stamp, process ID). For this purpose we use regular expressions. This process can be supported with event filtering in time and space (to eliminate redundancy); a single problem may generate many events (by various objects) within some short time.

The developed system has been used to collect, explore and analyse logs from various computers in order to create knowledge database on possible problems and their symptoms. Here, we can distinguish known critical problems (CP), suspected situations (SS) and unknown problems (UP). In the case of CP, we know the critical events. Sometimes they are identified by some specific sequences, in more complex situations they should be correlated with some context (e.g. specific configuration of an application, workload and usage profiles). The list of these symptoms and related problems is systematically updated as long as we gain new experience within the monitored systems. In the case of SS and UP situations, we rely on some learning processes combined with monitoring. The main idea is to identify characteristics of normal operation, its correlation with environment and operational profile changes and next selecting suspicious deviations for deeper analysis. These deviations are formulated for various performance measures or event log files. Moreover, in our approach we can study various correlations and identify their level of confidence.

An important issue is to distinguish normal and anomalous deviations in the observed performance and event spaces. Most of normal deviations relate to periodic activations of system scripts, program updates, backups, etc., and can be identified using statistics and tracing automatically shapes of plots in EPLogAnalyzer or SPSS (Section 4). Having eliminated normal deviations, we can explore the remaining ones in more detail. In the sequel, we present

some illustrative examples. Observing CPU usage (in per cent) typically, we detect various fluctuations and spikes of its appearance. It is worth tracing their sources. We can select all events registered in some specified time window before the specified level of CPU usage increase or a spike and then check the support measure for various selected events. In most cases this correlates with initiated specific scripts (reported in logs), although it may relate also to some anomalies.

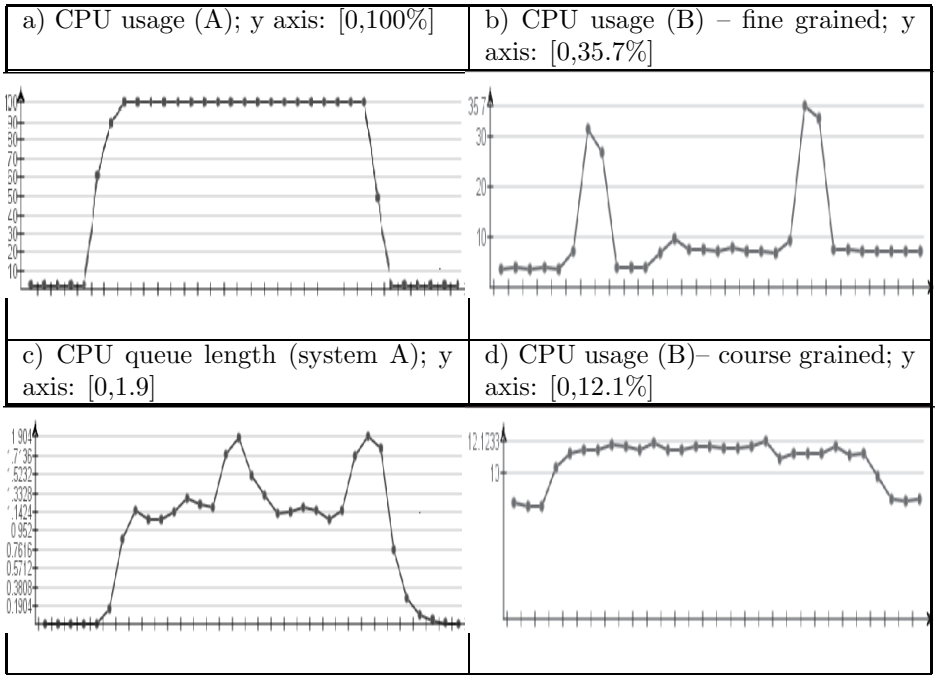

Figure 5. Sample of anomalous plots of selected performance measures for system A and B (x-axis time scale specifies subsequent minutes for a, b, c and hours for d)

For instance, CPU usage in the monitored system A was in the range of few per cent, sometimes rising to 20%, but 100% usage was also observed for about 20 minutes each week. This spike is shown in Fig. 5a. The corresponding average system load (average CPU queue length) is less regular (Fig. 5c). But, correlating this spike with event logs, we have confirmed with 100% confidence that it was invoked by a specific system script (normal deviation). In the case of system B, we have identified average CPU usage 8-12% with some spikes (over 30%) appearing quite frequently - about once per 15 minutes. They were correlated with some scripts capturing screen images. In Fig. 5b (resolution of 1 minute), we see some small increase at the basement of the second spike,

which can be easily skipped by the observer. Looking at this in a wider time perspective, covering 30 hours in Fig. 5d, where each point shows average CPU usage per 1 hour, we see increased average CPU usage from 8 to about 12% for almost 24 hours. This anomaly has been correlated with some sequence of events in the security log: there were 250 000 events related to attempts of logging as an administrator with different user names and passwords corresponding to an external brute force attack, fortunately unsuccessful. Practically, all these events were different (in the field of user name and password), but they have been described with a common generalized event pattern using a regular expression (support confidence 100%).

In general, depending upon the specified condition CA, we can get more than one correlated event pattern. In one of the monitored systems, the average CPU usage was in the range of 8-12% with some spikes of higher values. By specifying CA condition as CPU>30%, we have got 316 different events preceding this condition, which were generalized (with regular expressions) to 34 patterns. For each of these patterns, we have calculated support and confidence values as well as minimal, maximal and average CPU usage levels. For two patterns, we have got 100% support confidence with average 95% CPU usage, for the third pattern the support confidence was 83% (average CPU usage 35%). Actually, they were related to some activities invoked by system scripts. It is interesting that sometimes we got no symptoms of problems in event or performance space, but the problems were identified by the users (e.g. incompatibility of pen drives or remote discs). Such situations should be reported in an additional user log file to improve system logs in the future.

We have faced also the problem of selecting appropriate variables for observation. For instance, in Fig. 5 we have plots of CPU usage and CPU queues for the same time period. The plot of the queue length is less regular and more difficult for analysis, the anomaly of Fig. 5a was sufficient to identify the symptom of the problem. This analysis becomes more complex in multiprocessor and clusters systems, where the correlations between nodes performing the same services must also be taken into account (compare Fig. 1).

6. Conclusions

The main contribution of our research is the developed integrated event and performance log analysis supported by the original database with advanced filtering and visualization functions. This approach takes into account many detailed features which extend the scope of system operation evaluation. When analysing event logs we base on event categorisation (event morphology, identification of parameter fields, etc.) and various statistics, including event sequences. In this process, we use regular expressions and some data mining algorithms. The multitude of possible performance parameters gives the possibility of detailed system tracing. It is important to select the most sensitive and representative variables to avoid system load disturbance. The developed tools with the integrated database facilitate these processes. Moreover, the included wide scope

of features allows tracing anomalies by checking correlations of event and performance logs. Here, we can deal with explicit events or their classes.

An important characteristic of our approach is the fine-grained data analysis taking into account different perspectives (local, global, time, space and problem manifestation). Event sequences and rules (ranked according to their support values), discovered in this way, contribute to the developed knowledge database useful in resolving dependability and resilience problems. The developed methodology has been verified in real systems and facilitated detecting and diagnosing various types of anomalies (HW, SW, administrator, configuration faults, user deficiencies, system downtimes, slowdowns, performance anomalies, HW/SW inconsistencies, external attacks, etc.).

Taking into account the complexity of the problem we must be conscious that the results of analysis may provide false alarms or skip some problems. Hence, an important issue is cooperation with system administrators or users to interpret suspicious situations (interactive and iterative exploration of problems). This experience will enhance the rules of reasoning.

Further research will concentrate on some improvement of generating log schemes (e.g. in applications) and event categorization with advanced data mining algorithms. In developing problem prediction schemes (see, e.g. Li et al., 2011; Salfiner et al., 2010) we will take into account also performance parameters supplemented with user and service reports. This will result in better characterisation of normal as well as abnormal system behaviour.

References

BRANDT J., DEBUSSCHERE B., GENTILE A., MAYO, J. and PEBAY P. (2008) Using probabilistic characterization to reduce runtime faults in HPC systems. *Proc. 8th IEEE Int. Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 759-764.

CHANDOLA V., BAERJEE A. and KUMAR V. (2009) Anomaly detection: A survey. *ACM Computing Surveys,* **41**, 3, 15.1-15.58.

CHEN CH., SINGH N. and YAJNIK M. (2012) Log analytics for dependable enterprise telephony. *Proc. of 9^{th} IEEE European Dependable Computing Conference*. IEEE Computer Society, 94-101.

CINQUE M., COTRONEO D. and PECCHIA A. (2009) A logging approach for effective dependability evaluation of computer systems. *Proc. of 2^{nd} IEEE Int. Conf. on Dependability*. IEEE Computer Society, 105-110.

FU Q., LOU J-G., WANG Y. and LI, J. (2009) Execution anomaly detection in distributed systems through unstructured log analysis. *Proc. of IEEE Conference on Data Mining*. IEEE Computer Society, 149-158.

FU X., REBN R., JIANFENG Z., WEI Z., ZHEN J. and GANG L. (2012) LogMaster: mining event correlations in logs of large-scale cluster systems. *Proc. of IEEE Symposium on Reliable Distributed Systems*. IEEE Computer Society, 71-80.

GMACH D., ROLIA J., CHERKASOVA L. and KEMPER A. (2007) Workload analysis and demand prediction of enterprise data center applications. *Proc. of 10th IEEE Int. Symposium on IISWC*. IEEE Computer Society, 171-180.

HILL T. and LEWICKI P. (2006) *Statistics: Methods and applications, A comprehensive reference for science, industry and data mining.* StatSoft, Inc.

HOFFMANN G. A., TRIVEDI K.S. and MALEK M. (2007) A best practice guide to resources forecasting for the Apache Webserver. *IEEE Transactions on Reliability*, **56**, 4, 615-628.

IBM (2011) IBM SPSS Modeler 14.2, Algorithms guide ftp://ftp.software.ibm.com/software/ analytics/spss/documentation/modeler/14.2/en/ AlgorithmsGuide.pdf

JOHN L. K. and EECKHOUT L. (2006) *Performance Evaluation and Benchmarking.* CRC Press.

KRÓL M. and SOSNOWSKI J. (2009) Multidimensional monitoring of computer systems. *Proc. of IEEE Symp. and Workshops on Ubiquitous, Autonomic and Trusted Computing.* IEEE Computer Society, 68-74.

LATOSIŃSKI P. and SOSNOWSKI J. (2012) Monitoring dependability of a mail server. *Electrical Review*, **88**, 10b, 223-226.

LI X., XUE Y. and MALIN B. (2012) Detecting anomalous user behaviors in Workflow-Driven Web applications. *IEEE Symposium on Reliable Distributed Systems.* IEEE Computer Society, 1-10.

LI Y., ZHENG Z. and LAN Z. (2011) Practical online failure prediction for Blue Gene/P: Period-based vs. Event-driven. *Proc. of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops.* IEEE Computer Society, 259-264.

MAGALHAES J. P. and SILVA L.M. (2011) Adaptive profiling for root-cause analysis of performance anomalies in Web based applications. *Proc. of IEEE International Symposium on Network Computing and Applications.* IEEE Computer Society, 171-178.

NAGGAPAN M. and VOUK M. A. (2010) Abstracting log lines to log event types for mining software system logs. *Proc. of Mining Software Repositories (Co-Located with ICSE 2010).* IEEE Computer Society, 114-117.

OLINER A. and STEARLEY J. (2007) What supercomputers say: A study of five system logs. *Proc. of the IEEE/IFIP Intern. Conference on Dependable Systems and Networks.* IEEE Computer Society, 575-584.

SALFINER F., LENK M. and MALEK M. (2010) A survey of failure prediction methods. *ACM Computing Surveys*, **42**, 3, March, 10.1-10.42.

SALFINER F. and MALEK M. (2007) Using hidden semi-Markov models for effective online failure prediction. *Proc. of 26th IEEE Int. Symposium on Reliable Distributed Systems.* IEEE Computer Society, 161-174.

SIMACHE C. and KAANICHE M. (2005) Availability assessment of SunOS/Solaris Unix systems based on syslog and wtmpx log files; a case study. *Proc. of IEEE PRDC Conference.* IEEE Computer Society, 49-56.

SOSNOWSKI J. and POLESZAK M. (2006) On-line monitoring of computer systems. *Proc. of IEEE DELTA Workshop*. IEEE Computer Society, 327-331.

SOSNOWSKI J. and KRÓL M. (2010) Dependability evaluation based on system monitoring. In: A. Al-Dahoud, ed., *Computer Intelligence and Modern Heuristics*. In-Tech, 331-348.

SOSNOWSKI J., KUBACKI M. and KRAWCZYK H. (2012) Monitoring event logs within a cluster system. In: W. Zamojski et al., eds., *Complex Systems and Dependability. Advances in Intelligent and Soft Computing*, **170**. Springer, 259-271.

STOICESCU M., FABRE J. and ROY M. (2011) Architecting resilient computing systems: overall approach and open issues. In: E. A. Troubitsyna, ed., *Proc. of SERENE 2011 Conference*. **LNCS 6968**, Springer, 48-62.

VAARANDI R. (2003) A data clustering algorithm for mining patterns from event logs. *Proc. of 3rd IEEE Workshop on IP operations and Management*. IEEE Computer Society, 119-126.

VAARANDI R. (2004) A breadth-first algorithm for mining frequent patterns from event logs. *INTELLCOMM 2004,* **LNCS 3283**, Springer, 293-308.

YE N. (2008) *Secure Computer and Network Systems*. John Wiley & Sons, Chichester.