15th Triennial World Congress, Barcelona, Spain

MODEL-BASED MONITORING OF HUGE FINANCIAL DATABASES

Janusz Milek [*,**] **Franta Kraus** [*] **Daniel Lenz Boris Rankov**

[*] *Automatic Control Laboratory, ETH Zürich, CH-8092 Zürich, Switzerland*
[**] *Predict AG, CH-4153 Reinach, Switzerland*

Abstract: Huge financial databases may contain terabytes of data and have truly industrial dimensions. Since the quality of conclusions drawn using the data depends primarily on the information quality, the data has to be to monitored using appropriate methods. This paper discusses statistical model-based methods, including process monitoring approaches, applied with respect to the aggregated data, as well as data-mining methods, operating at the level of the raw data. Both approaches utilize data redundancy to build statistical models.

Keywords: databases, monitoring, redundancy, model, probability density function

1. INTRODUCTION

Financial databases which belong to banks, insurance, or telecommunications companies, contain terabytes of data. The stored information can be invaluable, *e.g.*, for analytical customer relationship management, where customer-level models can be estimated using data mining methods (Weiss and Indurkhya, 1998). Since information quality is one of the most important factors determining quality of conclusions drawn using the data, it is necessary to monitor or even control the information quality (Block *et al.*, 2000; Milek *et al.*, 2001). Due to the truly industrial scale, high relevance, and hierarchical structure, huge databases can be treated similarly to industrial processes. This analogy help to arrive at useful monitoring approaches. Moreover, it can be expected that modern statistical process monitoring methods can be particularly useful to monitor databases. However, the data in a financial database are influenced by the following factors: (i) customer behavior, (ii) market, (iii) seasonal variations, (iv) data quality issues. Hence, it can be argued that monitoring of a database at a customer-level resembles monitoring of a chemical reactor at a level of single molecules. This calls for new monitoring methods, unprecedented in the classical process monitoring and acting at the micro-economic scale. On the other hand, aggregation of the customer-level data for whole customer segments enables getting the macro-economic effects, which can be monitored using the classical methods.

Statistical monitoring methods

Statistical database monitoring methods comprise ideas belonging to econometrics, process monitoring, and data mining (Wang, 1999). The main assumption of the monitoring is that certain statistical data properties do not depend on time (Makridiakis *et al.*, 1998). Usually, the monitoring procedure comprises two steps: (i) the selected statistical properties are estimated from available reference data and constitute the model, and (ii) the model validates new data. The most general statistical data description can be given in the form of multivariate probability density functions (*pdf*) which can be used to classify data samples. However, the later described pdf estimation for raw record-level data is very difficult due to the curse of dimensionality. This problems can be reduced using suboptimal methods, also described in the forthcoming sections. The first approach is to decrease the number of variables in the estimated pdf. Another possibility is to aggregate customer-level data in order to suppress individual variations. The dimensionality reduction makes pdf estimation task more feasible but is offset by an accuracy loss. Simple pdf-related statistical tests for univariate pdf may involve its estimation, analysis of peaks in the estimated distributions, or outlier detection. Advanced tests may utilize logistic regression-type tests (Weiss and Indurkhya, 1998), hidden Markov models (Elliott *et al.*, 1995), clustering techniques, or time series analysis of multivariate histograms.

2. DENSITY ESTIMATION

The probability density function is the most complete description of a random variable. Hence, knowledge of the 'true' pdf would be particularly useful for the data quality monitoring. Unfortunately the pdf of real data is almost never *a priori* known and must be estimated from the data, *e.g.*, using the described here kernel density estimation method.

2.1 *Basic idea of kernel density estimation*

The density estimator has the following form

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - X_i}{h}),$$

where h is the window width and $K(x)$ a kernel function. The kernel satisfies the following conditions:

$$K(x) > 0 \text{ for all } x, \int_{-\infty}^{\infty} K(x)dx = 1$$
$$\int_{-\infty}^{\infty} K(x)d\ x = 0, \int_{-\infty}^{\infty} x^2 K(x)dx = \sigma_K^2 > 0.$$

The kernel estimator can be regarded as convolution of the observations, represented as Dirac pulses, with the kernel function. Figure 1 shows an example of estimated pdf. The underlying pdf is a uniform distribution $U[0,1]$ and the used kernel is Gaussian. The

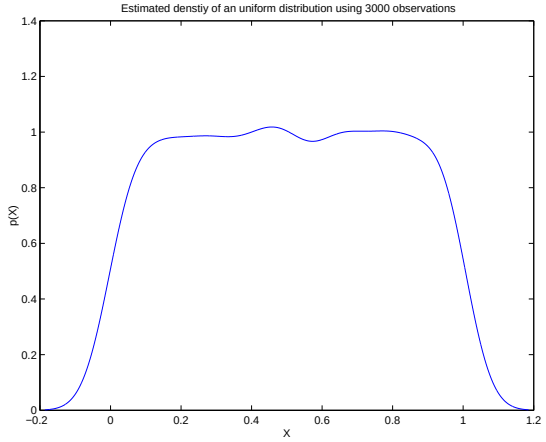

Fig. 1. Example of an estimated univariate probability density

kernel function determines the shape of the individual bumps placed at the observations which are summed up giving to the estimated density function $\hat{f}(\cdot)$.

2.2 *Multivariate kernel density estimation*

It is assumed that $\mathbf{X}_1, ..., \mathbf{X}_n$ is a given multivariate data set in an in d-dimensional space whose underlying pdf is to be estimated. Analogous to the univariate case, the multivariate kernel density estimator with kernel K and window width h is defined by

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K(\frac{\mathbf{x} - \mathbf{X}_i}{h})$$

where $K(\mathbf{x})$ is defined for d-dimensional $\mathbf{x}$, satisfying

$$\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1.$$

Usually the kernel function $K(\cdot)$ will be a radially symmetric unimodal probability density function, for example the standard multivariate normal density function

$$K(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}}\exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x})$$

or the multivariate Epanechnikov kernel

$$K(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\mathbf{x}^T\mathbf{x}) & \text{if } \mathbf{x}^T\mathbf{x} < \mathbf{1} \\ 0 & \text{otherwise} \end{cases}$$

where c_d is the volume of the unit d-dimensional sphere: $c_1 = 2$, $c_2 = \pi$, $c_1 = \frac{4\pi}{3}$, etc.

3. DATASPHERE METHOD

In case of multivariate pdf estimation two major problems arise: (i) visualization is almost impossible for $d > 2$, and, (ii) the number of partitions k increases exponentially with d. The DataSphere partitioning technique (Johnson and Dasu, 2000) allows to cluster the data in a efficient way. While a kernel method made out of a d-dimensional data requires $O(k^d)$ buckets the DataSphere creates only $O(d)$ buckets. Therefore they can be used for monitoring huge databases. The DataSphere-technique can be applied on aggregated data as well as on raw data. The fundamental idea is to partition the data into homogeneous sections. The sections are defined using distance layers to capture distance information, and pyramids to capture directional information.

3.1 *Distance Layers*

The first important data characteristics, the distance from the origin, is captured by the distance layers. For a set of data $X_i = (x_{i1}, x_{i2}, ..., x_{id}),\ i = 1, \ldots, N$ in a d-dimensional space, the distance layers can be computed by performing the following steps (Johnson and Dasu, 2000):

- Define a center of the data (a multivariate mean, componentwise median or trimmed mean in order to make the center more robust against outliers).
- Center and normalize the data, obtaining

  $$Y_i = (\frac{x_{i1} - \bar{x}_1}{\sigma_1}, ..., \frac{x_{id} - \bar{x}_d}{\sigma_d}) = (y_{i1}, ..., y_{id})$$

  where $\bar{x}_j$ and σ_j are the mean and standard deviation respectively of the j^{th} component.

- Compute the distance d_i from the center $\forall i$:

$$d_i = \sqrt{\sum_{j=1}^{d} (\frac{x_{ij} - \bar{x}_j}{\sigma_j})^2}$$

- Sort the data points by distance and define the layer boundaries to be distance quantiles. Therefore there is roughly the same number of data points in each layer.

3.2 *Directional Pyramids*

The second data characteristics, the distance from the origin, is captured by directional pyramids. A pyramid determines the direction of maximum variation. A d-dimensional data set is partitioned into $2d$ pyramids $P_i^{\pm}$:

- $p \in P_i^+$, if $|y_i| > |y_j|$ and $y_i > 0$, where $j = 1,...,d$, $j \neq i$
- $p \in P_i^-$, if $|y_i| > |y_j|$ and $y_i < 0$, where $j = 1,...,d$, $j \neq i$

This means simply searching for the largest component in the observation *(direction of maximum variation)* and determining whether it belongs to the positive or the negative pyramid. Figure 2 shows the complete partition of the data space. This technique may also be applied to aggregated data and can easily be updated with new data. A finer partition may be obtained by subdividing the pyramids into hyperpyramids. The reader is referred to (Johnson and Dasu, 2000) for further details.

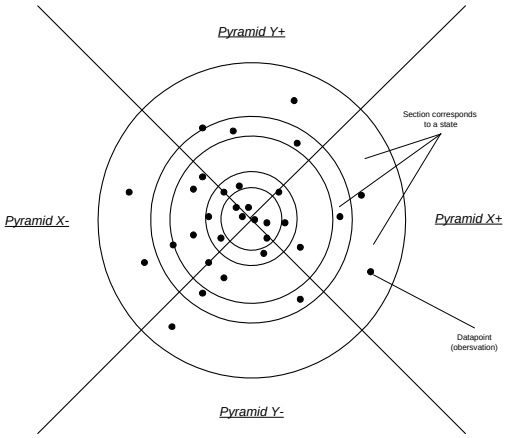

Fig. 2. Data Sectioning with Pyramids and Distance Layers

3.3 *Transition Analysis*

The DataSphere itself can be regarded as a stochastic process $\mathbf{X}[.]$ with the s-dimensional random variable $\mathbf{X}_i$, where s is the number of the states in the DataSphere, *i.e.*, number of layers times the number of pyramids. A data point may move from state

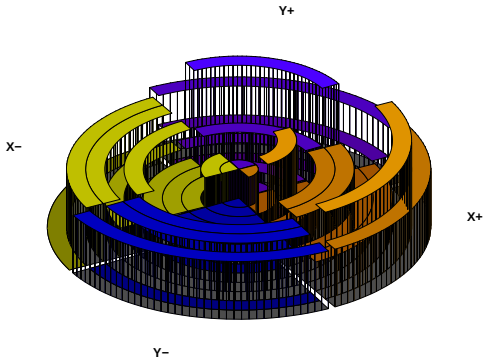

Fig. 3. A typical DataSphere representation, where the height indicates the number of points insight each region

to state over time. The stochastic process $\mathbf{X}[.] = \{...,\mathbf{X}_n,\mathbf{X}_{n+1},...,\mathbf{X}_{n+i},...\}$ shall be regarded as time discrete and first order Markov, *i.e.*,

$$\begin{aligned} P(\mathbf{X}_{n+1} = \mathbf{x}_{n+1} \mid \mathbf{X}_n = \mathbf{x}_n, ,..., \mathbf{X}_1 = \mathbf{x}_1) \\ = P(\mathbf{X}_{n+1} = \mathbf{x}_{n+1} \mid \mathbf{X}_n = \mathbf{x}_n). \end{aligned}$$

The movements of the data points over time among these states are summarized using the transition matrices. That is, the $(i,j)^{th}$ element of the transition matrix contains the probability that a data point (representing a single customer) will make a transition from state i to j at time t. The corresponding transition probability matrix is therefore

$$\mathbf{P}_t = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,s} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,s} \\ \vdots & & \ldots & \vdots \\ p_{s,1} & \cdots & \cdots & p_{s,s} \end{bmatrix}.$$

The individual elements $p_{i,j}$ of the transition matrix can be estimated using the sample proportion

$$p_{i,j}(t) = \frac{n_{i,j}(t)}{n_i(t)}$$

where $n_i(t)$ is the number of customers in state i at time t and $n_{i,j}(t)$ is the number of customers that move from state i at time t to state j at time $t+T$.

3.4 *DataSpheres for model-based monitoring*

In this section a simple model is introduced to show the DataSphere technique when applied to monitor large data sets in order to find faulty data. The basic idea is to build a model based on a reference data set. This DataSphere representation is then used as a foundation to construct a first order Markov process which is useful to detect low-likelihood state transitions of single customers.

Building the Model The model is built on a reference data set and consists of the following elements:

- The DataSphere: Layer bounds and number of pyramids
- The componentwise mean vector or the trimmed mean as a more robust centre.
- The componentwise standard deviation vector or as a more robust spreading description the (**M***edian* **A***bsolute* **D***eviation from the* **M***edian*) scale vector.
- The transition probability matrix

$$\mathbf{P}_{t_i} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,d} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,d} \\ : & & \ldots & : \\ p_{d,1} & \cdots & \cdots & p_{d,d} \end{bmatrix}$$

Figure 4 shows how a model may be built: the transition matrices $P_{1,2}, \ldots P_{q-1,q}$ are calculated from two consecutive DataSpheres considering the points that changed. It is not sufficient to regard the changes in a specific state but instead it is also necessary to track the points that move from a state i to another state j.

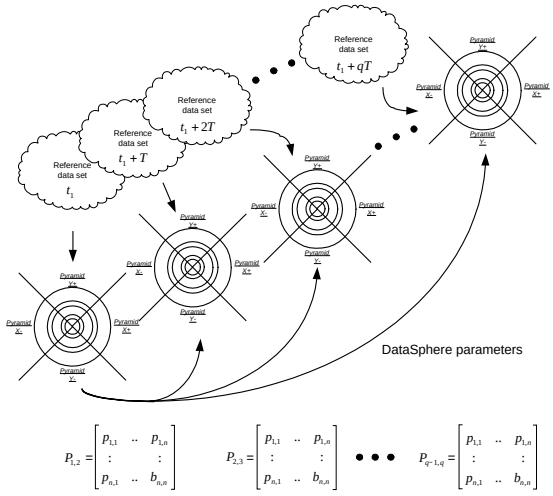

Fig. 4. Building a DS-model

Transition Probabilities in DataSpheres Using the DataSphere representation of data sets a Markov model can be constructed which allows us to anticipate unlikely changes in the customer behaviour. The most likely states of a customer in the next time period are predicted using the transition matrices. Any observed low likelihood transitions are flagged as alerts. Therefore a threshold $p_{i,j,\min}$ for the unlikely transitions must be defined. All transitions with smaller probability than $p_{i,j,\min}$ are flagged for further investigation. Figure 5 shows the procedure. The low likelihood transitions may be allocated to specific data points, finding changes in the model or faulty data. The calculated transition matrices $P_{1,2}, \ldots P_{q-1,q}$ and the DataSphere parameters are used as the model parameters.

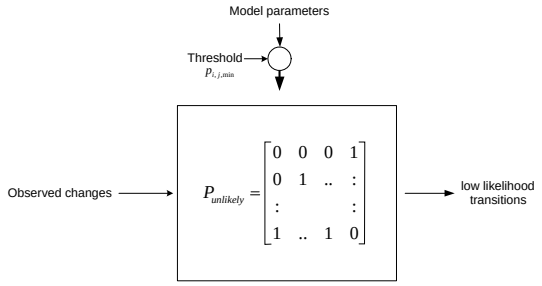

Fig. 5. Tracking of unlikely changes

4. MONITORING AGGREGATED DATA

Aggregated customer-level data exhibit stronger redundancy than the original data. The redundancy is extracted by statistical models which can be identified directly from the data (Ljung, 1998) and used for monitoring purposes, see (Gertler, 1998) and (Busatto, 2000). Two types of redundancy exist: spatial (between variables, segments, partitions, *etc.*) and temporal (in time). Temporal redundancy relates values of one variable for different time instants. The appropriate models are time series or lagged-variable models (AR and ARIMA). Spatial redundancy relates values of several variables for the same time instant, and the corresponding models are multivariate static models like linear (PCA) and nonlinear regression (NNPCA) models (Milek *et al.*, 2000). Both redundancy types can appear when values of several variables are related to each other for different time instants, and can be handled by multivariate time series models (VAR, VARMA, ARX and ARMAX).

4.1 *Data aggregation*

Analysis of the aggregated data (like asset sums, customers counts, service sales) gives an insight into the database contents. These time series have a clear meaning and can be compared to reference controlling data. Additionally, they can be treated as indicators, useful for decision makers for market monitoring and prediction. Moreover, it is often fair to suppose that such time series are slow changing and the relations between variables are almost constant (consider average assets per customer group). The aggregated data are influenced by seasonality, business conditions, and data quality issues. The aggregated data form compact multivariate time series, examples are: (1) number of accounts per customer segment, product type, partition, and time unit, and (2) sum of transactions per customer segment, product, partition, and time unit. Sensible aggregation operations include sum, count, mean, variance, minimum, maximum, histograms; such aggregates can be simply further aggregated (Dasu *et al.*, 2000).

4.2 *Monitoring via differencing in time*

This method is appropriate for exploiting temporal redundancy and testing if the aggregated variables change slowly, *e.g.*, in time/partition or time/customer segment coordinates. Examples of such variables are record counts, asset sums, number of customers, accounts, etc.. The underlying model is $x(t) = x(t-1) + \varepsilon(t)$, where $\varepsilon(t)$ is small compared to $x(t)$. Figures

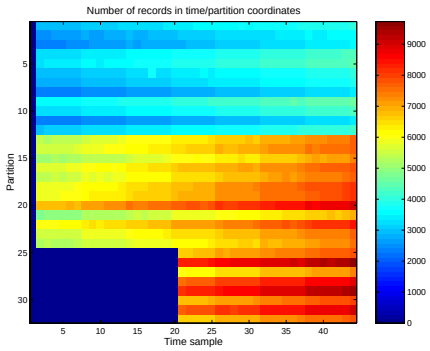

Fig. 6. Number of records in an example table

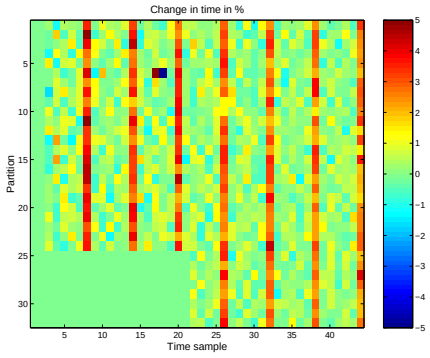

Fig. 7. Differenced number of records in %

6 and 7 shows simulated results: color-coded record count and its relative increase in percent, generated via the following residual generator: $e(t) = 100 * (x(t) - x(t-1))/x(t)$. The monitoring principle is to raise the alarm if bounds on $e(t)$ are violated. The following effects can be seen in Fig.7: (1) increase in number of records every 6 time units, (2) the appearance of new partitions, and (3) a small single outlier, not visible in the record counts.

4.3 *Monitoring of Aggregated Variables via PCA Algorithm*

This method can be used for monitoring an arbitrary number of variables, *e.g.*, aggregated for specified product and customer segments. The aggregated data related to different partitions and time instants are assumed to be approximately located in some hyperspace.

The Principal Component Analysis Model The aggregated, centered, and normalized data are stored in the matrix $X \in \Re^{K \times N}$ such that its columns correspond to N variables and the row $x'(k) \in \Re^N$ to k time sample. The Singular Value Decomposition $X = U\Sigma V'$, where $U \in \Re^{K \times K}$ is an orthogonal matrix, $\Sigma \in \Re^{K \times N}$ is a diagonal matrix $\Sigma = [\, diag(\sigma_i) \mid 0_{K \times (N-K)}]\,, i = 1 \ldots N$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_K \geq 0$, and $V \in \Re^{N \times N}$ is an orthogonal matrix, enables determination of the model. The coefficient matrix Θ is given as the last $N - L$ column vectors v of V, $\Theta = [\, v_{L+1} \cdots v_N]$, where L denotes order of the model. The orthonormal vectors spanning the model hyperspace are the first L column vectors of V, $P = [\, v_1 \cdots v_L]$.

PCA Fault Diagnosis Algorithm Fault diagnosis is performed in the following steps

- Fault detection via evaluation and assessment of the norm of the primary residuals

$$r = \|\Theta x\|_2 \tag{1}$$

- Fault isolation via evaluation and assessment of the norm of the structured residuals

$$r_i = \sqrt{x'\Pi_i(I - P(P'\Pi_i P)^{-1}P')\Pi_i x}, \tag{2}$$

  where Π_i is a diagonal matrix with ones for retained variables and zeros for eliminated variables. Note that structured residuals are sensitive to faults in the retained variables and completely insensitive to faults in the eliminated variables.
- Fault-free reconstruction of the data

$$\hat{x}_i = -(\hat{\Theta}\hat{\Theta}')^{-1}\hat{\Theta}\bar{\Theta}'\bar{x}_i, \tag{3}$$

  where $\hat{x}_i$ is a vector containing the reconstructed variable(s). $\bar{x}_i$ is a vector containing all variables except the variable(s) i to be reconstructed. $\hat{\Theta}$ contains only the column(s) i of Θ and $\bar{\Theta}$ contains the remaining columns.

Application Example This simulated example follows a real application. The processed data contain various customer transactions for a given product and customer segment, aggregated within partitions and time periods. There are 19 variables in 5 partitions, collected during 30 samples, additional 5 variables contain the numbers of the aggregated entries. It is assumed that a simulated fault may corrupt: (i) single variable in one partition, or (ii) single variable in all partitions, or (iii) all variables in one partition. Note that for each case the Π_i matrices in (2) are different. Figure 8 shows the primary residuals and the norm of the structured residuals (2) for an example fault corrupting all variables from the partition 4 (#61-79) for the time sample 9. The fault causes an increase of the norm of the primary residuals (Fig.8), and decrease of the structured residuals (Fig.9) related to partition 4, and is correctly isolated. Hence, the fault-free reconstruction of the corrupted data via (3) is possible.

Figure 10 shows such a reconstruction of the variable #63 from partition 4, which enables an assessment of the fault's magnitude and sign.

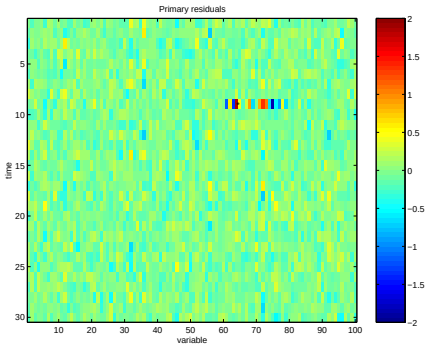

Fig. 8. Primary residuals

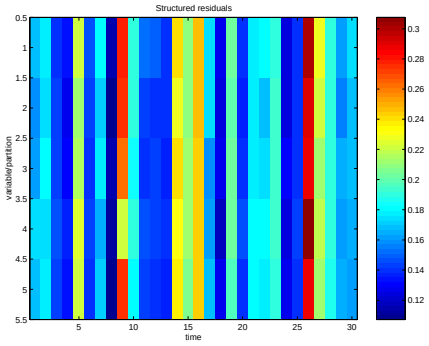

Fig. 9. Structured residuals

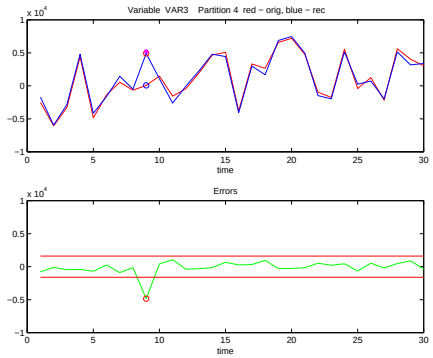

Fig. 10. Fault-free reconstruction (3) of the corrupted data (above) and the residuals (below)

5. CONCLUSIONS

Financial databases can be monitored using statistical, including the well known process monitoring approaches, applied with respect to the aggregated data, as well as data-mining methods, acting at the level of the original data. The discussed approaches exploit data redundancy to build statistical models, and deliver results which are of interest not only for the data quality monitoring but also for business–relevant information extraction.

6. REFERENCES

Block, F., J. Milek and M. Reigrotzki (2000). Datenqualität als basis künftiger business intelligence applikationen. In: *SAS Warehousing 2000*. Zürich. Switzerland.

Busatto, R. (2000). Using time series to assess data quality in telecommunications data warehouses. In: *International Conference on Information Quality, IQ 2000*. Cambridge, MA.. pp. 129–136.

Dasu, T., T. Johnson and E. Koutsofios (2000). Hunting data glitches in massive time series data. In: *International Conference on Information Quality, IQ 2000*. Cambridge, MA.. pp. 190–199.

Elliott, R., L., L. Aggoun and J. Moore (1995). *Hidden Markov Models. Estimation and Control*. Springer Verlag.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker.

Johnson, T. and T. Dasu (2000). Comparing massive high-dimensional data sets. *American Association for Artificial Intelligence*.

Ljung, L. (1998). *System Identification: Theory for the User*. Prentice Hall. 2nd edition.

Makridiakis, S., S. C. Wheelright and R. Hyndman (1998). *Forecasting: Methods and Applications*. John Wiley and Sons Inc. 3rd edition.

Milek, J., M. Reigrotzki, H. Bosch and F. Block (2001). Monitoring and data quality control of financial databases from a process control perspective. In: *International Conference on Information Quality, IQ 2001*. Cambridge, MA. To be presented.

Milek, J., O. Hermann and F. Kraus (2000). Use of hypersurfaces for fault detection, isolation, and reconstruction. In: *SAFEPROCESS 2000*. Budapest. Hungary. pp. 1199–1204.

Wang, X. Z. (1999). *Data Mining and Knowledge Discovery for Process Monitoring and Control*. Springer Verlag.

Weiss, S. M. and N. Indurkhya (1998). *Predictive Data Mining*. Morgan Kaufmann Publishers, Inc.