



Detecting Time Correlations in Time-Series Data Streams

Mehmet Sayal
Intelligent Enterprise Technologies Laboratory
HP Laboratories Palo Alto
HPL-2004-103
June 9, 2004*

E-mail: mehmet.sayal@hp.com

time-series, data
mining,
correlation, change
detection,
aggregation

In this paper, a novel method for analyzing time-series data and extracting time-correlations among multiple time-series data streams is described. The time-correlations tell us the relationships and dependencies among time-series data streams. Reusable time-correlation rules can be fed into various analysis tools, such as forecasting or simulation tools, for further analysis. Statistical techniques and aggregation functions are applied in order to reduce the search space. The method proposed in this paper can be used for detecting time-correlations both between a pair of time-series data streams, and among multiple time-series data streams. The generated rules tell us how the changes in the values of one set of time-series data streams influence the values in another set of time-series data streams. Those rules can be stored digitally and fed into various data analysis tools, such as simulation, forecasting, impact analysis, etc., for further analysis of the data.

Detecting Time Correlations in Time-Series Data Streams

Mehmet Sayal
Hewlett-Packard Labs
1501 Page Mill Road
Palo Alto, CA 94043, USA
+1-650-857-4497
mehmet.sayal@hp.com

ABSTRACT

In this paper, a novel method for analyzing time-series data and extracting time-correlations among multiple time-series data streams is described. The time-correlations tell us the relationships and dependencies among time-series data streams. Reusable time-correlation rules can be fed into various analysis tools, such as forecasting or simulation tools, for further analysis. Statistical techniques and aggregation functions are applied in order to reduce the search space. The method proposed in this paper can be used for detecting time-correlations both between a pair of time-series data streams, and among multiple time-series data streams. The generated rules tell us how the changes in the values of one set of time-series data streams influence the values in another set of time-series data streams. Those rules can be stored digitally and fed into various data analysis tools, such as simulation, forecasting, impact analysis, etc., for further analysis of the data.

Keywords

Time-series, data mining, correlation, change detection, aggregation

1. INTRODUCTION

In this paper, a novel method for analyzing time-series data and extracting time-correlations (time delayed cause-effect relationships) among multiple time-series data streams is described.

A *time-series* [8] is formally defined as a sequence of values that are recorded in one of the following manners:

- at equal time intervals
- at random points in time by also recording the time of measurement or recording

The proposed method uses the second definition of time-series data above in order to provide more generic algorithms that can be applied to both periodically recorded time-series data and time-series data that is recorded with random time intervals.

Time-correlations are defined as the relationships among multiple time-series data streams such that a change in the values of one set of time-series data streams triggers a change in the values of another set of time-series data streams. For simplicity, we first

describe our method for detecting time-correlations for a pair of time-series data streams. Then, we describe how a set of time-series data streams can be merged into one compact time-series data stream. As a result of the merging operation, we reduce the problem of finding time-correlations between two sets of time-series data streams into a simpler problem of finding time-correlations between two individual time-series data streams. We also apply data aggregation and change point detection techniques in order to reduce the search space and convert continuous data streams into discrete sequences of change points. Comparison of discrete sequences is much easier and faster than comparing continuous numeric data streams.

When looking for time-correlations, it is necessary to consider a few factors that may determine the way in which one time-series may affect another one. Figure 1 below shows a few time-series examples among which certain relationships exist. Please note that this is just an example and the example relationships discussed here may or may not apply to all business cases. As the first example, server performance has a *time-delayed* impact on database performance that can be easily observed in the figure. This relationship is easy to observe on the figure because the amount of impact on database performance is the same as the amount of change in server performance. As another example, the number of orders received has an immediate impact (i.e., no time delay) on average time per activity (average processing time for each activity in the organization that is related to handling of received orders). It can also be observed in the figure that a large change in number of orders causes a smaller impact in magnitude on average time per activity. That means, although average time per activity is affected by number of orders, its *sensitivity* to changes in number of orders is low. As a final example, let us consider the relationship between the average time per activity and percentage of on-time order delivery. The average time per activity has an immediate affect on percentage of on-time order delivery in the opposite *direction*. That means, when there is an increase in average time per activity, a decrease occurs in percentage of on-time order delivery, and vice versa. In summary, the main factors that we consider while analyzing the relationships among time-series data streams are *time-delay*, *sensitivity*, and *direction*.

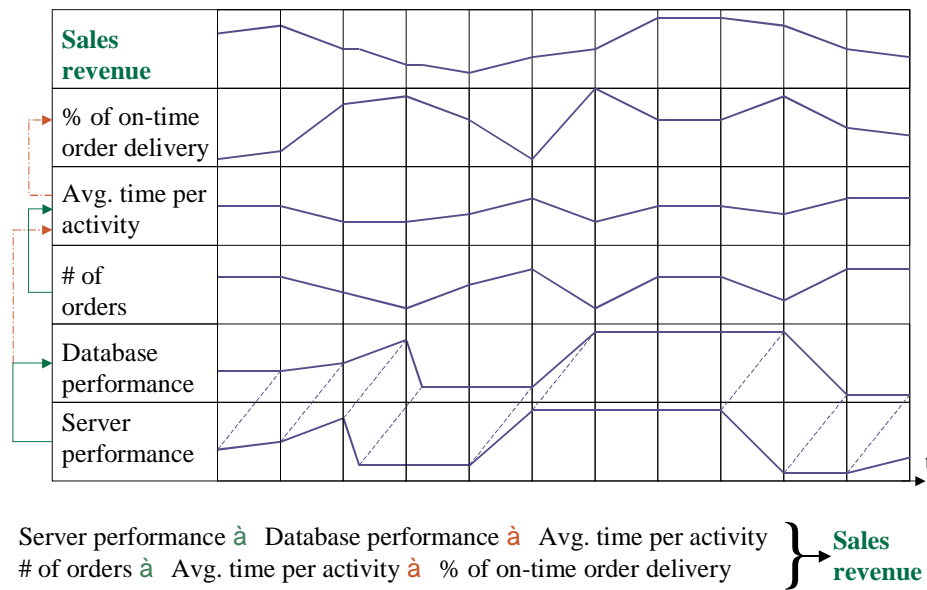


Figure 1. Time-series examples among which certain relationships exist

In addition to those factors, we consider confidence as one of the important factors that need to be considered when detecting the relationships among time-series data streams. Confidence is an indicator of how certain we are about the detected relationship among the time-series data streams.

A prototype has been developed that uses the proposed method in this paper for time-correlation detection among multiple time-series data streams. The prototype is called *Correlation Engine*. It has been developed in Java language with a web-enabled user interface.

The rest of this paper is organized as follows. Section 2 describes the proposed method, explains its main steps, and shows the overall architecture of the Correlation Engine prototype that uses the proposed method for detecting time-correlations among time-series data streams. An example scenario for using the introduced time-correlation detection method in Business Impact Analysis (BIA) is given in section 3. Section 4 summarizes related work in time-series analysis. Section 5 contains the conclusion. Finally, section 6 lists the references.

2. PROPOSED METHOD

2.1 Summary of the Method

The proposed method for detecting time-correlations from time-series data streams consists of the following steps, as shown in Figure 2:

- Summarizing the data at different time granularities (e.g., seconds, minutes, hours, days, weeks, months, years)
- Detecting change points using CUSUM (Cumulative Sum) statistical method [11, 12] in order to both reduce

the search space and convert continuous data stream into discrete data stream which is much easier to manipulate

- Generating correlation rules
 - Merging multiple time-series data to generate *Merged Time-Series Data* (note that because merged time-series data has the same data structure as regular time-series data, the term “Time-Series Data” will be used to refer to both regular and merged time-series data in the rest of this document)
 - Comparing time-series data to generate time correlation rules

These steps are described in more detail later in this paper.

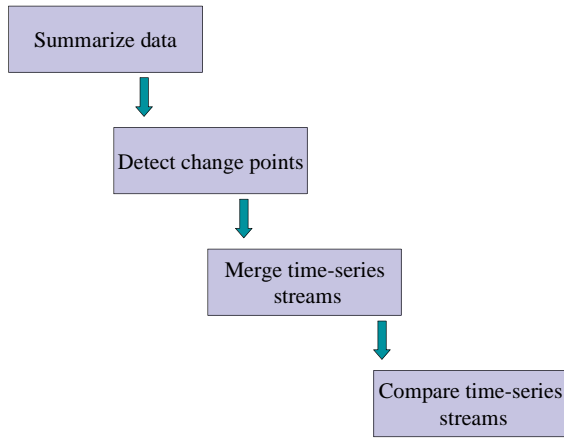


Figure 2. Main steps of the proposed method for extracting time-correlations from time-series data streams

The input of the method is any kind of numeric data stream that is time-stamped, which is called “time-series” data in this paper. The input data can be read from one or more database tables, an XML document, or a flat text file with character delimited data fields. The output is a set of time correlation rules that describe what data object fields are correlated to each other. Each time correlation rule includes information about:

- **Direction:** “Positive” if the change in the value of one time-series data is correlated to a change in the same direction for another time-series data. “Negative” if the change direction is opposite in the two correlated time-series.
- **Sensitivity:** The magnitude of change in data values in two correlated time-series are recorded in order to indicate how sensitive one time-series is to the changes in another time-series.
- **Time Delay:** The time delay for correlated time-series data are recorded in order to explain how much time it takes to see the effect of a change in the value of one time-series as a result in the value of another time-series.
- **Confidence:** Confidence provides an indication of how certain we are about the detected time-correlation among time-series data streams. Confidence is measured as a value between zero and one. A confidence value that is close to one indicates high certainty. Similarly, a confidence value that is close to zero indicates low certainty.

The proposed method can be used for detecting the following types of correlations between time-series data streams:

- **Simple correlation:**
city="Los Angeles" à population="high"
(conf: 100%)
- **Quantified correlation:**
A=5 or A=6 à B > 50 (conf: 75%)

- **Time correlation:**

A increases more than 5% à B will increase more than 10% within 2 days (conf: 80%)

The first two types of correlations above can also be detected by the existing solutions. The proposed method can be used for detecting not only those two types, but also time correlations, which are more difficult to detect. Detection of time correlations provides significant advantages over existing solutions, because in most systems the effect of a change is observed after a certain time delay (not simultaneously), which is very difficult to detect using the existing solutions.

Table-1 below shows an example database table that contains 3 time-series data for the grades of a high school student: Math, Physics, and English.

Table 1. Example database table containing time-series data

Name	Value	Time-stamp
Math	85	January 12, 2002
Physics	93	January 26, 2002
English	74	February 20, 2002
Math	96	March 23, 2002
Physics	81	April 2, 2002
English	65	April 5, 2002
...
Math	97	January 10, 2003
...

Figure 3 below shows the overall architecture of Correlation Engine, which is the prototype implementation of the time-correlation detection method introduced in this paper.

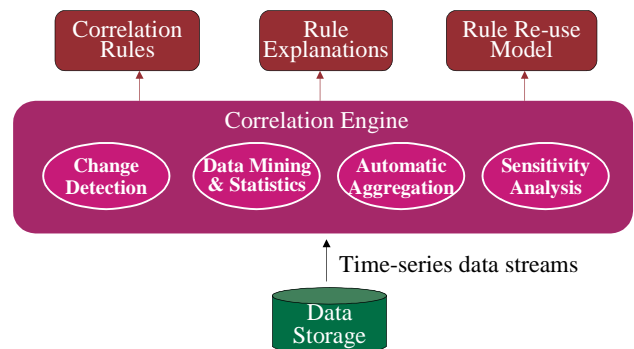


Figure 3. Overall Architecture of Correlation Engine

Correlation Engine receives time-series data streams as its input, and generates correlation rules, textual rule explanations, and stores the generated rules digitally so that they can be reused by

other data analysis tools. Correlation rules fall into one of the three categories explained above: simple, quantified, or time-correlations. Simple and quantified correlation rules may also be detected using conventional data mining techniques, such as classification. The time-correlations are detected by the Correlation Engine using the method introduced in this paper. The simple and quantified correlations can be easily detected by using the method introduced in this paper, but the main focus in this paper is the detection of time-correlation rules, which is much harder than detection of other types of correlation rules.

Correlation Engine applies automatic data aggregation and change detection algorithms in order to reduce the search space. Those two algorithms are described in the following two subsections. Data mining and statistics is used by Correlation Engine for various purposes, such as statistical correlation calculation for a pair of data streams, and change detection (i.e. CUSUM). Sensitivity analysis is embedded inside change detection and correlation rule generation steps in the proposed method. Sensitivity analysis is achieved through recording the change amounts while detecting change points, and comparing those change amounts from different time-series data streams while generating the correlation rules.

Rule explanations that are generated by Correlation Engine are textual explanations of the detected rules. For example, assume that the following database table contains a simplified digital view of a time-correlation rule:

Table 2. Example database table that shows how the generated time-correlation rules can be digitally stored

COL1_NAME	COL2_NAME	TIME_DIFFERENCE	TIME_UNIT	CONFIDENCE	TYPE
Health of Resource X	Response Time of Activity Y		1 hour	0.6	Negative
Response Time of Activity Y	Violations of SLA S1		0 hour	0.7	Positive

then a textual explanation of this rule can be easily generated in the following form:

If **Health of Resource R** decreases more than 5%, then **Duration of Activity A** increases more than 10% on the next day.

The database table above skips the details about sensitivity information (change magnitude) for simplicity of this example, but in the actual implementation Correlation Engine stores such sensitivity information in order to provide detailed information to users about the detected time-correlation rule. The sensitivity information contains the magnitude changes compared to each other for the correlated time-series data streams. Sensitivity analysis is very important because it tells us the amount of expected impact when a variable (i.e., the value of time-series data stream) changes.

The digital storage of the generated time-correlation rules in this manner (i.e. in relational database tables) makes it possible to feed those rules easily into various data analysis tools, such as simulation, forecasting, etc. As an example, Figure 4 below shows how the generated time-correlation rules can be reused for simulation.

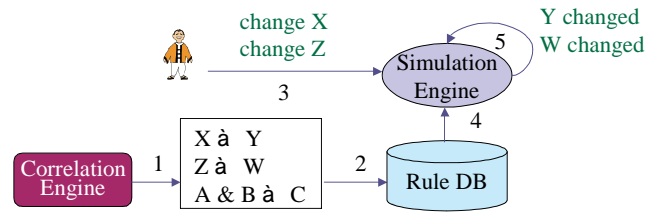


Figure 4. Example of reusing generated time-correlation rules for simulation

Figure shows that variables X and Z affect the variables Y and W respectively. The reuse of those rules in the simulation takes place as a result of the following activities:

1. Time-correlation rules are detected by the Correlation Engine
2. The rules are written into a rule database in a digital format similar to the one shown in Table 2
3. User of the simulation tool makes changes to the variables X and Z
4. The simulation tool retrieves the rules from the rule database in order to assess the expected impact in response to the changes in variables X and Z
5. Simulation tool finds out from the rules that are retrieved from the rule database that variables Y and W are going to be impacted because of the changes in the values of variables X and Z. The time-correlation rules include information about direction, sensitivity, and time-delay of the impact so that the simulation tool can figure exactly when, in which direction, and how much the impact is going to be observed.

2.2 Summarizing the data at different time granularities:

Time-stamped numeric data values (time-series data) have to be summarized for two main reasons:

- o Volume of time-series data is usually very large. Therefore, it is more time efficient to summarize the data before analyzing it. For example, if there are thousands of data records for each minute, it may be more time efficient to summarize the data at minute level (e.g. by taking mean, count, and standard deviation of recorded values), so that the data can be analyzed in a more time efficient manner.
- o Time-stamps may not match each other, which makes it difficult to compare time-stamped data with each other. For example, all exams in Table-1 were recorded on different dates, i.e. different time-stamps. It is not possible to compare the exam scores that have the identical time-stamps, because there is not enough recorded data at each time-stamp value to compare different time-series values. Summarizing the numeric data (e.g. taking the average value for each course) by day wouldn't

be useful either, because all exam scores were recorded on different days. Even summarizing the scores by month is not enough because each month of the year doesn't contain a recorded value for every time-series (i.e., for every course). Consequently, sometimes it may be necessary to summarize data using higher time granularity so that the recorded numeric data can be compared with each other.

If additional time-stamp information is provided, such as the notion of academic calendar year, or business calendar units (such as financial quarter or financial year), then those can be also used as data aggregation attributes.

Figure 5 below shows an example of how data aggregation is done at any particular time granularity level. The aggregation is performed by calculating the sum, count, mean, min, max, and standard deviation of individual data values within each time unit. The figure below shows the mean value calculation, which is equal to sum of values divided by the count of values in within each time unit.

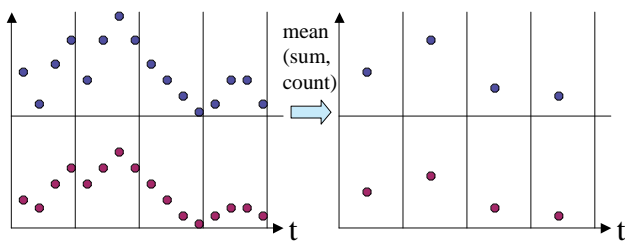


Figure 5. Example for data aggregation that can be done at any time granularity (minutes, hours, days, etc.)

It is necessary to consider the cases in which the effect of a change in one time-series data stream may not be observed exactly at the same time delay all the time. In fact, in most cases the effect is likely to be shifted slightly in the time domain. The experiments showed that the time shift is not very large and 99% of the time, the effect is observed within one time unit difference. In order to capture those cases, the introduced method uses moving window of three time units at any granularity level. For example, aggregation of data values in the "hour" granularity involves the current hour as well as the previous and next hours. As a result, slight shifts in the time domain can be incorporated in the proposed method. Some of the existing methods use complex algorithms for time shifting or time warping in order handle such time shifts [4].

2.3 Detecting change points:

CUSUM (Cumulative Sum) [11, 12] is a well-known statistical method for detecting change points in time-stamped numeric data. *CUSUM* at each data point is calculated as follows:

1. Subtract the mean (or median) of the data off of each data point's value.
2. For each point, add all the mean/median-subtracted points before it.

3. The resulting values are the Cumulative Summary (*CUSUM*) for each point.

The *CUSUM* test is useful in picking out general trends from random noise as noise will tend to cancel out in the long-run (there are just as many positive and negative values of true noise) but a trend will show up as a gradual departure from zero in the *CUSUM*. Therefore, *CUSUM* can be used for detecting not only sharp changes, but also gradual but consistent changes in numeric data values over the course of time. It is important to keep in mind that the *CUSUM* is not the cumulative sum of the data values; instead it is the cumulative sum of differences between the values and the average. *CUSUM* is a simple, but effective method for detecting change points in time-series data.

Once *CUSUM* value for every data point is calculated, the calculated *CUSUM* values are compared with upper and lower thresholds to determine which data points can be marked as change points. The data points for which the *CUSUM* value is above the upper threshold or below the lower threshold are marked as change points. The thresholds can be determined using standard deviation (i.e. a fraction or factor of standard deviation) or set to two constant values. In most cases thresholds are set using the standard deviation. It is easy to calculate a moving mean or standard deviation using a moving window. Therefore, we can safely assume that standard deviation can easily be calculated on any time-series data.

In order to establish control limits in *CUSUM* plots, Barnhard [3] proposed the so-called *V-mask*, which is plotted after the last data sample (on the right). The *V-mask* can be thought of as the upper and lower control limits for the cumulative sums. However, rather than being parallel to the center line; these V-shaped lines converge at a particular angle to the right, producing the appearance of a V-shape rotated on its side. If the line representing the cumulative sum crosses either one of the two lines, the change point is detected.

Detected change points are marked with one of the labels, indicating the direction of change that is detected:

- Down: trend of data values change from up or straight to down
- Up: trend of data values change from down or straight to up

In addition, amount of the change is recorded for each change point. This amount of change is used for sensitivity analysis.

2.4 Generating Time-Correlation Rules:

When trying to find out time correlations among multiple time-series data streams, the proposed method first reduces many-to-one and many-to-many time-series comparisons into that of pairwise (one-to-one) time-series comparison. Then, the problem of comparing multiple time-series data streams can be tackled efficiently and easily. In order to explain the proposed reduction and comparison steps of the proposed method, it is first necessary to explain what is meant by one-to-one, many-to-one, and many-to-many time-series comparisons:

- **One-to-one:** comparison of two time-series data streams with each other. This is the simplest form of time-series comparison. The purpose is to find out if there exists a

time correlation between two time-series. For example, if A and B identify two time-series data streams, one-to-one comparison tries to find out if changes in data values of A have any time delayed impact on changes in data values of B. The comparison is denoted $A \hat{=} B$.

- **Many-to-one:** comparison of multiple time-series data streams with a single time-series data stream. For example, if A, B and C identify three time-series data streams, many-to-one comparison tries to find out if changes in data values of A and B collectively have a time delayed impact on changes in data values of C. This comparison is denoted $A*B \hat{=} C$.
- **Many-to-many:** comparison of multiple time-series data streams with multiple time-series data streams. For example, if A, B, C and D identify four time-series data streams, many-to-many comparison tries to find out if changes in data values of A and B collectively have a time delayed impact on changes in data values of C and D. This comparison is denoted $A*B \hat{=} C*D$.

Usually, many-to-many comparisons do not have any practical use because their results can be easily derived from results of many-to-one comparisons. However, the proposed method is also capable of reduce many-to-many comparisons into one-to-one comparisons, if necessary.

The proposed method first describes how to reduce many-to-one and many-to-many time-series comparisons into one-to-one time-series comparison. Then, it explains how to perform one-to-one time-series comparison in order to extract time correlation rules, which is applicable to any combination of time-series comparisons as a result of the reduction step.

2.4.1 Reducing Multiple Time-series Data Streams into One:

The purpose of reducing multiple time-series data streams into one by merging them is to be able to compare multiple time-series data streams with each other in one shot. This yields performance improvement because the merged time-series data streams can be reused, similar to the way query results can be reused in database management systems in order to provide performance improvement. For example, after merging two time-series data streams A and B, the merged time-series data stream can be stored in order to generate higher order merged time-series data streams, such as $A*B*C$, where the symbol "*" is used for indicating the merger operation.

The proposed method uses two different methods for merging time-series data streams: convolution and sum.

Convolution is a well-known technique that can be used for merging multiple time-series data streams into a single time-series data stream. Convolution is a computational method in which an integral expresses the amount of overlap of one function $g(x)$ as it is shifted over another function $f(x)$. Convolution may therefore blend one function with another. Convolution of two functions $f(x)$ and $g(x)$ over a finite range is given by the equation:

$$f * g \equiv \int_0^t f(\tau)g(t-\tau)d\tau$$

where $f * g$ denotes the convolution of f and g .

The other method used for merging multiple time-series data streams into a single time-series data stream is simple sum of the numeric values in time-series data streams. When calculating the sum, both the positive and negative sums are considered because of the following reason:

Let us assume that two time-series data streams are going to be merged using the sum operation. Assume that there exists an increase in both time-series data streams at the same point in time. Summing up the numeric values from the time-series data streams A and B will strengthen the numeric change at that point of time, as it should be the case. However, consider the case in which there exists an increase in numeric values of time-series A at a point in time, where there exists a decrease in numeric values of B. In that case, sum of the two time-series may decrease the magnitude of each or even neutralize each other completely, which should not be the case. In summary, it is necessary to sum up absolute values, instead of actual numeric values, when merging time-series data streams. The proposed method uses absolute values but keeps the direction information about each change point, which is needed during the comparison of time-series data streams. In general, when merging two or more time-series data streams with any particular method, it is necessary to use the absolute data values and keep the direction information for each change point. Since the merge operation is applied on discrete time-series data streams (after using CUSUM to detect change points), the operation can be performed much faster than merging the actual time-series data streams in their original numeric values.

2.4.2 Comparing Two Time-series:

The proposed method uses statistical correlation to calculate the time correlation between two time-series data streams. The time-series data streams that are compared at this step may correspond to either merged time-series or regular time-series. The statistical correlation between two time-series is calculated as:

$$\text{cor}(x,y) = \frac{\text{cov}(x,y)}{\sigma(x) \sigma(y)}$$

where x and y identify two time-series, and covariance is calculated as

$$\text{cov}(X, Y) = E\{[X - E(X)][Y - E(Y)]\}$$

and $\sigma(x)$ corresponds to the standard deviation of values in time-series x ,

$\sigma(y)$ corresponds to the standard deviation of values in time-series y ,

$E(X)$ and $E(Y)$ correspond to the mean values of time-series data values from x and y .

Time correlation is calculated as follows:

$$\max \{\text{cor}(x_i, y_j)\} \forall i, j \in t; i \neq j,$$

where t corresponds to aggregated time span of the time-series data (e.g. minutes, hours, days, etc.).

Sensitivity is calculated using the following formula:

$$\text{compare change}(x_i, y_j) \text{ where } i, j \in t; i \neq j, |i-j| = d$$

by setting the distance (d) between i and j to that of the maximum statistical correlation found, and comparing the magnitudes of change in the correlated time-series data streams.

In other words, the statistical correlation between aggregated data points with varying time distances are calculated, and the maximum calculated correlation and the corresponding time distance (d) give us the time correlation information between the compared time-series data streams. The time distance for the maximum statistical correlation found between two time-series data streams will be denoted d . The sensitivity is calculated by comparing the change amounts in the correlated time-series data streams using time distance (d) of the calculated maximum statistical correlation. The direction of correlation is also obtained from the calculated statistical correlation.

The most challenging task at this comparison step is to determine the time distance (d) for which the calculated correlation is the highest. An exhaustive search in which all possible time distances are tried and maximum correlation is determined is prohibitive due to performance reasons unless there exists an upper limit for the distance (d) that is set by the user. Otherwise it is necessary to find the distance (d) in a faster way. The proposed method uses sampling in order to first determine which time distances are likely to return a high correlation between the time-series data streams that are being compared. Then, the actual correlation is calculated for a few of those candidate distances, and the one with the highest correlation is selected. The experiments with various kinds of time-series data streams yielded promising results. Although the number of candidate distances considered has a significant effect on the accuracy of the result, the experiments showed that after sampling the data, it is enough to consider at most three or four candidate distances to find the highest correlation distance for 95% of the time.

Once the time correlation is calculated, the confidence can also be calculated easily by comparing the percentage of times the calculated statistical correlation with the time delay (d) of the maximum correlation is higher than a particular threshold. For example, if the proposed method finds out that the time correlation is the highest for a time delay of 3 units, say 3 days ($d = 3$ days), then the confidence is calculated by measuring what percentage of the time x_i and y_j values have a statistical correlation larger than a particular threshold. The threshold can be chosen by the user of this method.

3. EXAMPLE USE SCENARIO

The time-correlation rules generated by the method that is introduced in this paper can be used in various fields. For example, Business Impact Analysis (BIA) tries to find the impact of certain events on the operation of the business. Businesses make use of Workflow Management Systems (WfMS) [10] in order to automate the execution and monitoring of their business processes. A *business process* is a step by step description of activities that are carried out by humans or automated resources (e.g., databases, procurement systems, resource planning software, etc.) in order to achieve a business goal. A *workflow* is the formal definition of a business process that can be executed and monitored using a WfMS. Figure 6 below shows an example workflow definition which is used for handling purchase order (PO) requests that are received from customers.

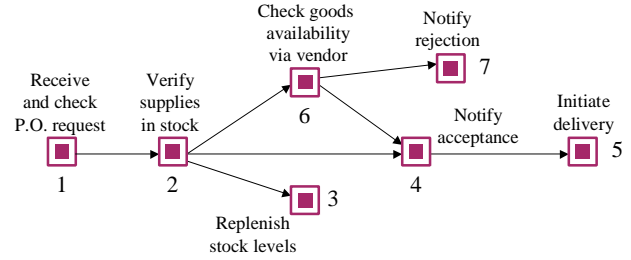


Figure 6. Example workflow definition

The individual activities in this workflow definition are carried out by resources such as humans or automated systems (i.e., IT resources). For example, “Initiate Delivery” activity may be carried out by a human who makes a phone call to internal delivery department of the business in order to initiate the delivery of the ordered products. Similarly, “Notify Acceptance” or “Notify Rejection” activities may be carried out by an e-mail messaging system that is capable of sending e-mail notifications to the customer.

The time-correlation rules are very suitable for use in BIA because the time-correlation rules can tell us the impact of a change in the value of one numeric variable on the value of another numeric variable. Figure 7 below shows an example BIA scenario in which the time-correlation rules that are generated by the introduced method can be used.

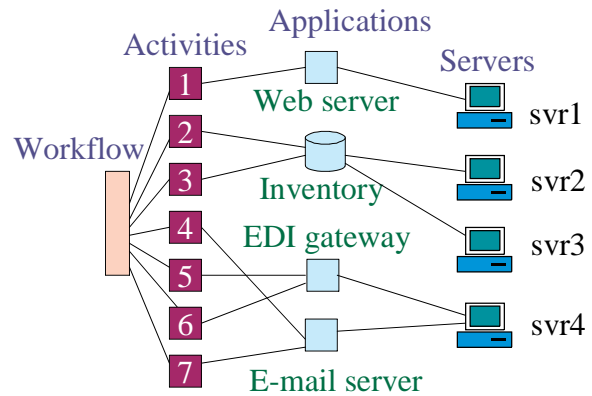


Figure 7. Example Business Impact Analysis (BIA) scenario

Figure 7 shows the overall workflow definition of Figure 6 at the highest level (left-most). The activities that are number between 1 and 7 in Figure 6 are symbolized with their corresponding numbers in Figure 7. Figure 7 indicates that each one of the activities in the workflow by one or more automated resource (human resources are ignored only for simplicity). It can also be observed from the figure that a resource (application) may be in charge of handling one or more activities in the workflow. Each application runs on top of a hardware resource, which is shown as the servers in the figure. Each application may run on one or more servers, and similarly each server may run one or more applications.

Business Impact Analysis on such an architecture requires that we know or find out the impact of change at a lower level entity on the execution of a higher level entity. For example, if server 2

crashes or experiences a performance degradation, then the Inventory application will be affected negatively. That will in turn affect the execution of activities 2 and 3. Eventually, that will affect the overall execution of the workflow and its execution will most probably be delayed or even become impossible.

Time-correlation rules can be used for identifying the dependencies among the performance of any entities in this scenario. For example, the proposed method in this paper can come up with time-correlation rules for any of the following:

- server performance and application performance,
- server performance and activity execution time,
- server performance and workflow execution time,
- application performance and activity execution time,
- application performance and workflow execution time,
- activity execution time and workflow execution time

The generated time-correlation rules tell us impact of any change at a lower level entity on the higher level entities. For example, the following rule, which was used as an example for textual explanation of the generated rules, tells us the impact of a change in the performance of a server on the execution time of an activity in the workflow:

If **Health of Resource R** decreases more than 5%,
then **Duration of Activity A** increases more than 10% on
the next day.

The main advantage of the use of time-correlation rules in BIA is that the rules can be generated for any time-series data streams from different levels in the example scenario architecture shown in Figure 7. That means, we can directly find the impact of a server crash on the overall execution time of the whole workflow without having to know about the actual dependencies or hierarchy of the existing architecture. This is one example that shows the time-correlation detection method introduced in this paper is generic enough to be applied in many different problem domains without requiring domain knowledge.

Similarly, the introduced method can be used in analyzing supply chain transactions, business-to-business (B2B) interactions, procurement systems, web services, etc. As long as transactional or operational log data is available, the introduced method can be applied for detecting time-correlations on such data in order to gain knowledge about how the underlying system works.

4. RELATED WORK

Most of the related work concentrated on similarity based pattern querying. The similarity-based approaches use three main data models: similarity model, data representation, and indexing structure. The similarity model defines the semantics of pattern queries. Data representation defines how to store the data for analysis. Indexing structure is used for speeding up the search after the similarity patterns are clustered into groups. For example, Agrawal et al [1] and Faloutsos et al [7] use Euclidean distance as the similarity model, the coefficients of the moving-window DiscreteFourier Transform (DFT) as the data representation, and an R^* -trees as the index structure for speeding up the search. Many of the existing research that applied features that are extracted from the data have used DFT in order to transform the time-series from time domain into frequency

domain [1, 4, 7, 9, 16]. The main problem those approaches was the fact that similarity model was different from the data representation [13]. It is necessary to find a proper transformation between the two models.

In order to consider the fact that sometimes the time-series data streams may have different time domain scales or data value magnitude ranges, etc., the similarity models have been extended to consider various transformations on the data. For example, warping [4, 14, 15, 16], amplitude shifting [5, 9, 15], and allowing time-series segments of different amplitude scales to be considered similar [2, 5, 6, 9].

Based on the observation that humans can ignore small fluctuations and apply smoothing when comparing time-series data visually, Rafiel et al proposed a method based on smoothing and noise reduction using moving averages [14]. Unfortunately, smoothing and noise reduction also result in loss of accuracy due to the modification in the nature of the actual time-series data.

Perng et al [13] suggested a new method that first tries to identify landmarks, i.e., important points in time when certain events happen, and then using six different transformations to confirm similarity of time-series data streams.

The main similarity of the introduced method in this paper to some of the previous work is that we also considered summarizing the continuous time-series data streams in a discrete form. However, we apply a well-known statistical approach for change detection, instead of trying to determine what kind of events should be considered important events that determine the general characteristics of the underlying data. The introduced method also differs in most of the previous research that the similarity model and data representation match each other. Instead of building complex indexing structures to speed up search, the introduced method applies sampling techniques in constant running time to determine few candidates and then analyzes the candidates during the comparison step in linear running time. Instead of smoothing and noise reduction, we apply data aggregation based on actual time units that preserve most of the mid-scale fluctuations in the actual data. The experiments performed during the development of the proposed method and previous research suggest that detecting the change points in the time-series data rather than trying to map the data into various feature sets yields the best results, and that is why change detection is used as the only transformation method in this paper.

5. CONCLUSION

A novel method for detecting time-correlations in time-series data streams is introduced. The method consists of a few main steps in which the original time-series data is summarized using data aggregation; transformed into discrete data using a well-known change detection technique for faster analysis; sampled for fast detection of candidates, and scanned in linear time for confirming the correlation rules. The proposed method takes advantage of certain observations in previous research and during experiments performed for this method. The use of the time-correlation rules in Business Impact Analysis is explained on an example. The generated time-correlation rules can easily be reused for analysis of transactional or operational data in many different domains. Moreover, the generated rules can be converted into textual representations, which is very important for human users to understand the findings.

6. REFERENCES

- [1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, Evanston, IL, October 1993.
- [2] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, Zurich, Switzerland, September 1995.
- [3] G. A. Barnhard, 'Control charts and stochastic processes,' *Journal of Royal Statistical Society*, (B) vol. 21, pp. 239-257, 1959.
- [4] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. MIT Press, 1996.
- [5] K. K. W. Chu and M. H. Wong. Fast time-series searching with scaling and shifting. In *PODS*, 1999.
- [6] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Proceedings of Principles of Data Mining and Knowledge Discovery (PKDD)*, Trondheim, Norway, June 1997.
- [7] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, 1994.
- [8] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.
- [9] D.Q.Goldin and P. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *International Conference on the Principles and Practice of Constraint Programming*, 1995.
- [10] F. Leymann, D. Roller, *Production Workflows*. Prentice-Hall, Englewood Cliffs, 2000.
- [11] E.S. Page, Continuous Inspection Schemes, *Biometrika*, 41, pp.100-114, 1954.
- [12] E.S. Page, An improvement to Wald's approximation for some properties of sequential tests, *Journal of Royal Statistics Society B*, 16, pp.136-139, 1954.
- [13] C-S. Perng, H. Wang, S. Zhang, D. S. Parker, Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases. In *Proceedings of the 16th International Conference of Data Engineering (ICDE)*, San Diego, CA, February 2000.
- [14] D. Rafiel and A. O. Mendelzon. Similarity-based queries for time series data. In *SIGMOD*, 1997.
- [15] H. Shatkay and S. B. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of the 12th International Conference of Data Engineering (ICDE)*, 1996.
- [16] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, 1998.