International Journal of Innovative Computing, Information and Control
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 5401–5410

AN EARLY WARNING STRATEGY FOR REAL-TIME BUSINESS PROCESS MONITORING

Bokyoung Kang[1], Dongsoo Kim[2,*] and Suk-Ho Kang[1]

[1]Department of Industrial Engineering
Seoul National University
599 Gwanangno, Gwanak-Gu, Seoul, Korea
{ real369; shkang }@snu.ac.kr

[2]Department of Industrial and Information Systems Engineering
Soongsil University
369 Sangdoro, Dongjak-Gu, Seoul, Korea
*Corresponding author: dskim@ssu.ac.kr

Received March 2011; revised September 2011

Abstract. *This paper proposes a novel approach to real-time business process monitoring incorporating an early warning strategy based on expected gain analysis. First, at each monitoring instant in midcourse, an extended decision-tree algorithm estimates a probability that the ongoing process will accomplish the targeted performance after completion. Second, based on that probability, the expected gains are analyzed for two cases: stopping the running process and executing continuously until completion. If stopping is more profitable, an early warning is generated to notify probable loss after completion. Otherwise, the process is executed until the next instant. This procedure is iterated periodically over entire instants. The proposed method estimates predictors reflecting the real-time progress of a running process, especially current capabilities of accomplishing business goals. Furthermore, by means of the early warning, the monitoring system can proactively prevent probable losses rather than just react to them.*

Keywords: Process monitoring, Business activity monitoring, Real-time system, Early warning, Decision tree

1. **Introduction.**

Business Activity Monitoring (BAM) is defined as an information system providing real-time access to data for process management purposes [1]. Business Process Management Systems support a whole process management lifecycle, including functions such as process design, execution, monitoring and improvement, in which BAM refines business models based on knowledge extracted from historical process logs [2]. Moreover, BAM, as integrated with Business Intelligence (BI), detects sets of events or attributes that lead to risks or opportunities [1,3]. Therefore, BAM can be used for quantitative management in real-time, through monitoring metrics or significant patterns reflecting the status of a process [4].

BAM, to realize its functions, adopts rule-based approaches that define extracted knowledge in the *If* (*conditions*) *Then* (*actions*) form, which means that if predefined conditions are detected, then corresponding actions are taken [5]. However, they have shown some limitations when applied to real-time process monitoring, in which factors are recorded sequentially by order of task execution in the process, not instantly and simultaneously. Those approaches, because of unrecorded factors according to real-time progress, cannot but be reactive rather than proactive [3,6]. After detecting all factors, failures or loses maybe already happened so that it is difficult to operate real-time feedback at the midcourse [7]. Moreover, it is not easy to provide comprehensive indicators implying real-time progress [8].

To alleviate these problems, we propose for real-time business process monitoring an early warning strategy entailing analysis of expected gains and losses. In each monitoring period, an extended decision-tree algorithm estimates the probability that the ongoing process instance will accomplish a final result. Then, the expected gains and losses are analyzed for two decision-making alternatives, stopping the running process, and executing continuously until completion. If the expected gain by stopping is larger, continuous execution might incur losses, and so an early warning of such probable losses is generated. Otherwise, the process keeps running and the above procedure is iterated periodically in the next monitoring period.

The rest of this paper is organized as follows. Section 2 provides details on the existing process monitoring approaches. Section 3 introduces an early warning decision problem for real-time business process monitoring. Section 4 formulates an extended decision-tree algorithm for real-time estimation of target result probability. Section 5 explains early warning condition estimation based on analysis of the expected gain and loss. In Section 6, experiments with two example scenarios are discussed, and the results are evaluated. Our approach can be extended to cope with various types of process structure, which is described in Section 7. Section 8, finally, draws conclusions and suggests future work.

2. Existing Process Monitoring Approaches.

Inductive Data Mining (IDM) techniques investigate historical logs to extract relations among instance attributes significantly associated with the final process performance [9]. Such extracted knowledge is defined as If-Then rule used to monitor the current executions, which is called a rule-based approach [10]. One of the widely used data mining techniques to analyze historical logs is decision tree. Although arguably less accurate than other techniques, it provides the most interpretable rules and describes correlations between attributes clearly [11]. This facilitates and enhances industrial managers' understanding of the status of the applied process [12]. Grigori et al. introduced a BI-based process monitoring approach employing decision tree for use in managing process execution quality [3]. Peddabachigari et al. integrated decision tree with support vector machine extract intrusion patterns in network transactions [13]. Wetzstein et al. introduced Service Level Agreement (SLA) monitoring methods [14,15]. To extract SLA rules, key performance indicators (KPIs) of the process and Quality of Service metrics are analyzed by decision tree. Lim and Lee integrated the decision tree with association rule mining to analyze the relations of KPIs to the process performance [12].

However, some of the existing approaches have shown limitations when applied to real-time process monitoring. Such limitations are obvious when attributes associated with final results are recorded sequentially in the order of the execution of relevant tasks. These approaches still operate reactively rather than proactively. Because some conditions of each rule can be identified only at the end of the execution phase, rule-detection methods are difficult to embed at the midcourse [16]. This incurs delay, which in turn causes a monitoring system to remain idle until the last condition is identified [3,6], which means that a reaction is too late [17,18]. As a result, it is difficult to make appropriate corrections from the near-real-time feedback at the midcourse [7]. Therefore, it is necessary to predict probable outcomes before actual eventualities while counteractive actions to prevent them are still possible [19]. Second, there is no comprehensible and sophisticated indicator to show real-time progress [8]. It is necessary for indicators to predict outcomes based on current progress [16]. With such predictors, users can obtain insights into process performance during the initial phase, and thereby can predict the possibility of achieving targeted outcomes [7]. Moreover, these insights serve as proactive guidance for recalibration of practical expectations about outcomes [16].

To resolve above-noted limitations, some researchers suggested prediction models based on general linear equation characteristics [16,19]. However, they are difficult to represent more complicated correlations. IDM-based models using decision tree [4,8] and clustering [20] were suggested to predict outcomes based on partial data. As another approach, formal concept analysis was introduced to construct a data structure visualizing probable process status with binary events [21]. Although they also adopted the warning strategy, there still were some limitations about the determination of the threshold for warning conditions: simple comparison between the estimated value and the user-defined threshold.

3. Problem Statement.

A business process is composed of tasks and the flows between them, the execution of which generates a process instance. During the running of the process instance, tasks are executed in a sequential flow with spending costs. After executing each of the tasks, a *task result* is generated, which is defined as an attribute recorded after task execution. After completion, a performance is evaluated as a *final result*. In our approach, we defined the final result as completion types in terms of *Success/Failure*. Then, a gain or a loss occurs dependently on the final result of the process. Notations of the process model are defined as follows.

- $T = \{t_i\}$ *is a set of tasks executed in order during the implementation of a process.*
- $C = \{c_i\}$ *is a set of costs incurred for executing corresponding tasks.*
- $X = \{x_i\}$ *is a set of task results generated after executing relevant tasks.*
- $Y = \{Success \text{ or } Failure\}$ *is a final result, which is determined as Success or Failure according to whether the final outcome obtains the targeted performance result or not.*
- r_S *and* r_F *are a gain from Success and a loss from Failure, respectively.*

Figure 1 shows the overall procedure of the proposed real-time business process monitoring method. In running the process, task results and the final result are monitored and archived in a treatable data type by an integrated information system, specifically its BAM component. We assumed that a final result of an entire execution is significantly associated with correlations between the task results. Therefore, a decision tree can be trained by using historical process logs, and from the constructed tree, If-Then rules for the correlations can be derived. During run-time, an ongoing process instance is monitored through a set of generated attributes. An extended decision-tree algorithm is run to estimate the probability of success completion. Then, the expected gains and losses are estimated for two decision alternatives, stopping the ongoing process instance and executing continuously the remaining tasks. If stopping is more beneficial than executing, a monitoring system provides an early warning of probable losses to industrial managers. Otherwise, the ongoing instance is executed continuously until the next monitoring period, at which time the above procedure is iterated periodically.

Figure 2 illustrates a monitoring instant i in midcourse defined as the moment when the task results of an ongoing instance are recorded. As i elapses during continuous executions, additional results periodically are recorded. The ongoing instance progresses in executing tasks and incurring costs. At i, t_1, …, t_i are executed by incurring $c_1, \ldots, c_i$, after which the monitoring system can observe x_1, …, x_i recorded up to i. Based on observed attributes until i, $P_i(Success)$ is estimated as the probability that the ongoing process instance can be completed as a *Success*. Then, we determine whether to stop the ongoing instance at i or execute continuously until completion. To evaluate the expected gains

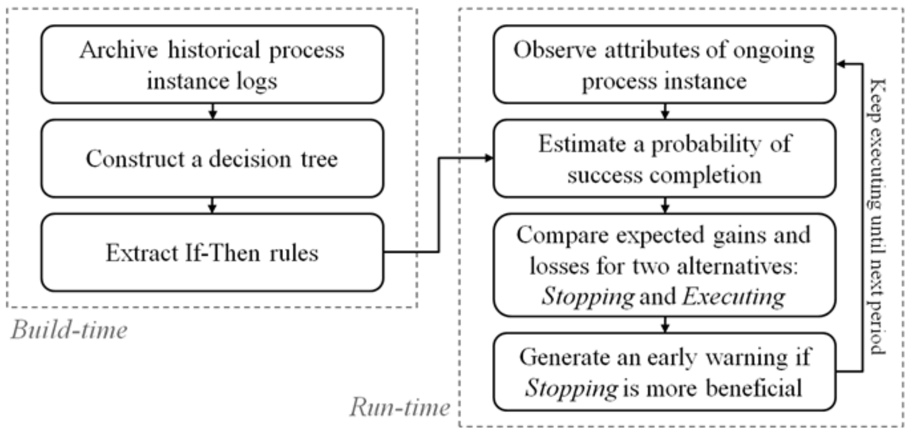

Archive historical process instance logs
Construct a decision tree
Extract If-Then rules
Build-time
Observe attributes of ongoing process instance
Estimate a probability of success completion
Compare expected gains and losses for two alternatives: Stopping and Executing
Generate an early warning if Stopping is more beneficial
Run-time
Keep executing until next period

Figure 1. Overall real-time business process monitoring procedure

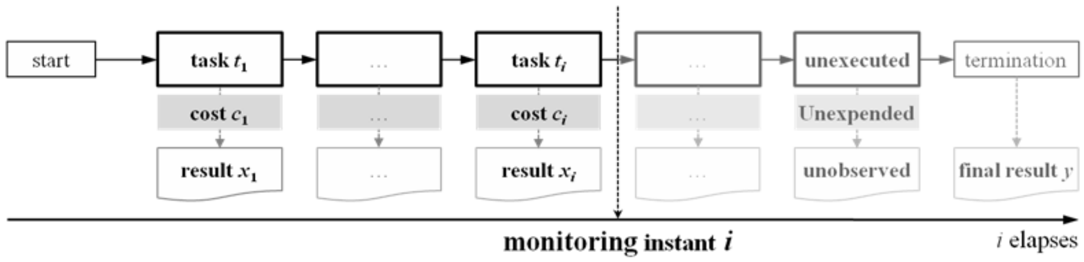


Figure 2. Monitoring instant in real-time process monitoring

and losses for the two decision alternatives, we should consider the estimated probability $P_i(Success)$, the costs that have been incurred already, and the additional gains or losses according to the final result. Finally, we can select the more beneficial alternative and determine whether to generate an early warning or not.

4. $\boldsymbol{P_i(Success)}$ Estimation.

We here present an extended decision-tree algorithm to estimate $P_i(Success)$ based on observed $x_1, \ldots, x_i$ and unobserved $x_{i+1}, \ldots, x_n$. We adopted and extended ideas of the real-time risk-level measurement concept proposed by Kang et al. [8].

Let us suppose that there are historical logs in which each instance is archived as a set of task results and a final result in a form of $\{(x_1, \ldots, x_n), y\}$. The task results and the final result correspond to attributes and the target value in a training data so that the historical logs can be used to train a decision tre. We construct a simple decision tree by J. R. Quinlan [11]. The decision tree is composed of branches extending from one root and ending in leaves. The leaves represent a classified target value satisfying the conditions of the upper nodes, and the other nodes represent the attributes-based conditions, with a branch for each possible outcome. Therefore, each branch can be interpreted as "*If an object satisfies the conditions of all nodes on the branch, then it is classified as the target value of the leaf*".

After deriving m rules, denoted as $rule_1, \ldots, rule_m$, from the constructed decision tree, we specify rule parameters for each $rule_j$ as follows.

- *n_j is the number of instances in historical logs that satisfy the conditions of $rule_j$.*
- *p_j is the ratio of instances with success completion among n_j instances.*
- *σ_j is the standard deviation of p_j and is defined as $\sqrt{p_j(1-p_j)/n_j}$.*

- $w_{i,j} = \begin{cases} 0, & \textit{if the conditions of rule}_j \textit{ are not satisfied with } x_1, \ldots, x_i \\ 1, & \textit{otherwise} \end{cases}$

At monitoring instant i, the monitoring system can gather only observed $x_1, \ldots, x_i$, because $x_{i+1}, \ldots, x_n$ are not observed yet so that the derived *If-then* rules cannot be followed. However, we can filter out the parts of them that have not been satisfied already with $x_1, \ldots, x_i$. To accomplish this, coefficient $w_{i,j}$ is defined as follows. When identifying the conditions of $rule_j$ at i, the conditions for $x_{i+1}, \ldots, x_n$ are ignored, only those for $x_1, \ldots, x_i$ are considered. If the conditions of $rule_j$ are not satisfied with $x_1, \ldots, x_i$, $w_{i,j}$ is 0, and so $rule_j$ is considered no more after i. Otherwise, $w_{i,j}$ is 1 and, subsequently, re-considered at $i + 1$.

Finally, we estimate the expected value of $P_i(Success)$, $E(P_i(Success))$, and its standard deviation $std(P_i(Success))$ as defined in Equation (1). Each coefficient of p_j represents the probability that the observed instance will satisfy the conditions of $rule_j$ when considering probable progresses after i.

$$\begin{aligned} E\left(P_i\left(Success\right)\right) &= \sum_{j=1}^{m} \left(\frac{w_{i,j} n_j}{\sum_{j=1}^{m} w_{i,j} n_j} p_j \right), \\ std\left(P_i\left(Success\right)\right) &= \sqrt{\sum_{j=1}^{m} \left(\frac{w_{i,j} n_j}{\sum_{j=1}^{m} w_{i,j} n_j} \sigma_j \right)^2} \end{aligned} \tag{1}$$

5. **Expected Gain Estimation for Early Warning.** For an ongoing process instance observed at i, there are two decision alternatives, executing or stopping. By comparing the expected gains of the two alternatives, we can identify which is more beneficial. Then, we can decide whether to generate an early warning or not.

(*Step* 1) *Expected gain from executing*: At i, $c_1, \ldots, c_i$ are already expended, and $c_{i+1}, \ldots, c_n$ additionally will be expended if we decide to execute until completion. Then, instead of incurring remaining costs, we can expect gain r_S after the ongoing instance is completed to *Success*. In summary, the expected gain from executing continuously is calculated as

$$\begin{aligned} \begin{pmatrix} \text{The expected gain} \\ \text{from executing} \end{pmatrix} &= -\begin{pmatrix} \text{Expended cost +} \\ \text{Additional execution cost} \end{pmatrix} + \begin{pmatrix} \text{The expected gain} \\ \text{from Success} \end{pmatrix} \\ &= -\left(\sum_{k=1}^{i} c_k + \sum_{k=i+1}^{n} c_k \right) + P_i\left(Success\right) \times r_S. \end{aligned} \tag{2}$$

(*Step* 2) *Expected gain from stopping*: $c_1, \ldots, c_i$ are already expended but there is no need to incur more costs if we decide to stop at i. Although we cannot expect gain r_S from *Success*, the loss r_F from *Failure* will not occur by early stopping, and so the prevented loss becomes the gain of stopping. Therefore, the expected gain from stopping is calculated as

$$\begin{aligned} \begin{pmatrix} \text{The expected gain} \\ \text{from stopping} \end{pmatrix} &= -\left(\text{Expended cost}\right) + \begin{pmatrix} \text{The expected loss} \\ \text{from Failure} \end{pmatrix} \\ &= -\sum_{k=1}^{i} c_k + \left(1 - P_i\left(Success\right)\right) \times r_F. \end{aligned} \tag{3}$$

(*Step* 3) *Comparison of expected gains from executing and stopping*: If the expected gain from stopping is larger than that from executing, Equation (4) exceeds 0.

$$\left(\begin{array}{l}\text{The expected gain}\\ \text{from stopping}\end{array}\right)-\left(\begin{array}{l}\text{The expected gain}\\ \text{from executing}\end{array}\right)=\sum_{k=i+1}^{n} c_k+r_F-P_i\left(Success\right)\times(r_S+r_F) \tag{4}$$

The above condition can be derived as follows. The right side of Inequality (5) serves as a feasible lower-bound of $P_i(Success)$ for continuous execution. If $P_i(Success)$ satisfies Inequality (5), it can incur the loss after completion so that an early warning is needed.

$$P_i\left(Success\right)<\frac{\sum\limits_{k=i+1}^{n} c_k+r_F}{r_S+r_F} \tag{5}$$

(*Step* 4) *Early warning strategy*: Using the expected value and the standard deviation of $P_i(Success)$, we can estimate a probabilistic distribution of $P_i(Success)$ by the normal approximation to the binomial distribution. Then, $P_i(Success)$ has a normal distribution with parameters of Equation (1). Therefore, $P(P_i(Success)<Lower\text{-}bound)$, a probability that satisfies Inequality (5), is derived as

$$P\left(P_i\left(Success\right)<\frac{\sum\limits_{k=i+1}^{n} c_k+r_F}{r_S+r_F}\right)=\int_{-\infty}^{\frac{\sum_{k=i+1}^{n} c_k+r_F}{r_S+r_F}}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(P_i(Success)-\mu)^2/2\sigma^2}dP_i(Success), \tag{6}$$

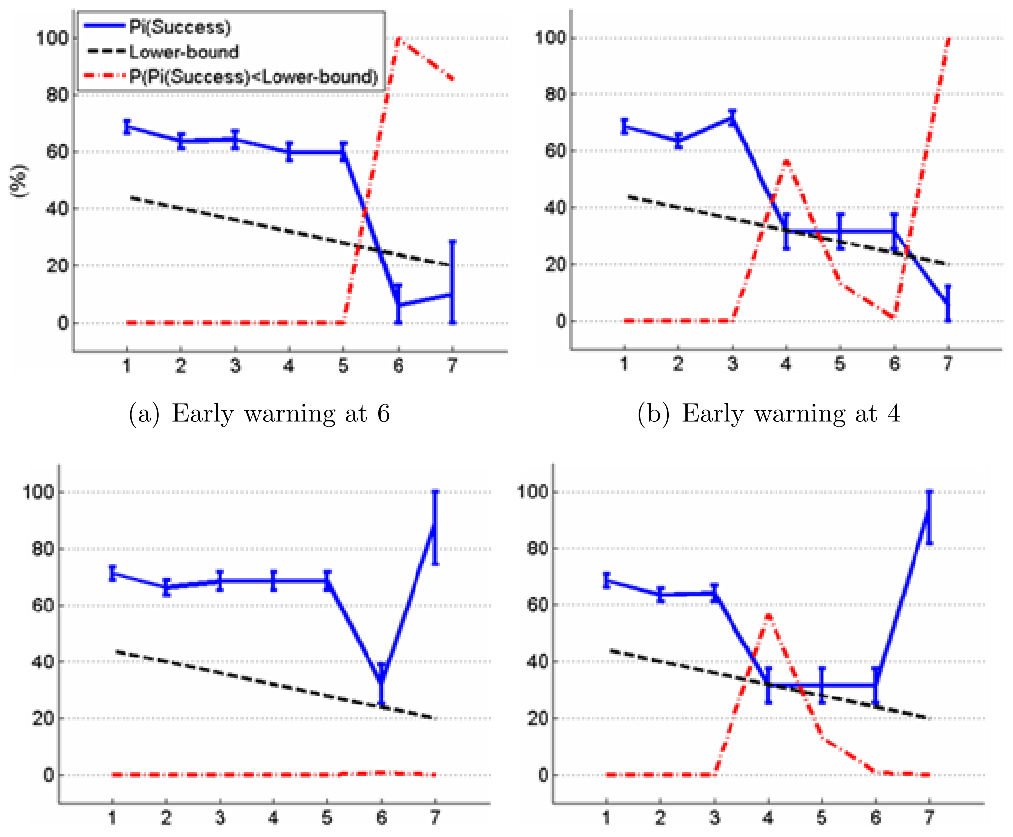
where μ is $E(P_i(Success))$ and σ is $std(P_i(Success))$. The value of Equation (6) indicates the probability that executing continuously incurs a loss. Finally, an early alarm is generated at the i if $P(P_i(Success)<Lower\text{-}bound)$ exceeds the predefined threshold.

6. Experimental Results.

In experiments conducted with two example scenarios, we observed how the proposed method can be applied to real-time business process monitoring by visualizing predictors during the real-time progress of ongoing process instances.

After implementing the sequential process model with tasks t_1, …, t_7, a process instance was generated and recorded through 7 task results x_1, …, x_7 comprising 6 continuous and 1 categorical variables, which were observed sequentially. The execution cost was defined as $c_i = 1$ for all i. The gain and loss from the completion types were defined as $r_S = 20$ and $r_F = 5$. Artificial data generated randomly were used as process instances including tasks results and corresponding final results. As training data, 1340 *Success* and 660 *Failure* instances were generated for historical logs to construct a decision tree, and then 9 rules were extracted. As test data, 50 *Success* and 40 *Failure* instances were generated as ongoing instances. For the ongoing instance, the system periodically estimated the predictors, which were visualized with the real-time progress, and an early warning was generated if necessary.

Figure 3 illustrates examples of monitoring results according to completion types. The error-bar line shows the expected value and 95% confidence intervals of $P_i(Success)$. Specific trends appear in $E(P_i(Success))$ after completion types: a decrease if *Failure* and an increase if *Success*. The linearly decreasing dotted-line is a feasible lower-bound of $P_i(Success)$. Since incurred costs grow larger as i elapses, the expected losses from *Failure* become smaller, so that the lower-bound also becomes smaller. And it decreases linearly, because all c_is are the same. The polygon dotted-line is $P(P_i(Success)<Lower\text{-}bound)$, which is the probability of incurring a loss. When $P(P_i(Success)<Lower\text{-}bound)$ exceeds 50% threshold, the early warning is generated. Early warnings can be provided 1 instant



(a) Early warning at 6

(b) Early warning at 4

(c) No warning

(d) False warning at 4

FIGURE 3. Early warning-based real-time monitoring results according to completion types

and 3 instants before *Failure* completion, as in (a) and (b). For *Success* completion in (c), the warning was not needed. However, there was a false warning at instant 4, upon the same threshold, to the instance with *Success* completion, as in (d).

To determine the threshold by comparing losses from missing warnings and false warnings, we conducted the following two example scenarios. 1) Without early warning, all instances are entirely completed with incurring all executing costs so that gains and losses are generated according to completion types. 2) Instead of early warning, early stopping is performed. In a stopping instance, only the executing costs are incurred. In a completed instance, after incurring all executing costs, gains and losses are generated according to the completion types. Finally, a net gain was calculated for each of two monitoring scenarios, using the formula "(*total gains from Success*) – (*total losses from Failure*) – (*total executing costs*)". Then, a relative net gain of early stopping was computed as "(*net gain of early stopping*)/(*net gain of no stopping*)". Figure 4 shows the relative net gain of early stopping with various thresholds from 0% to 100%. With 0%, most of the instances were early stopped, so that the number of false warnings (stopping) increased so that it was too low. With increases to the threshold, it grows larger. Between 60% and 90%, the net gain was 2.59 times as large as that of no stopping. With 100%, there was no early stopping so that two scenarios were the same.

7. Extensions to Various Structures of Business Process.

Though a business process can be defined as diverse types of structure, task results cannot but help being

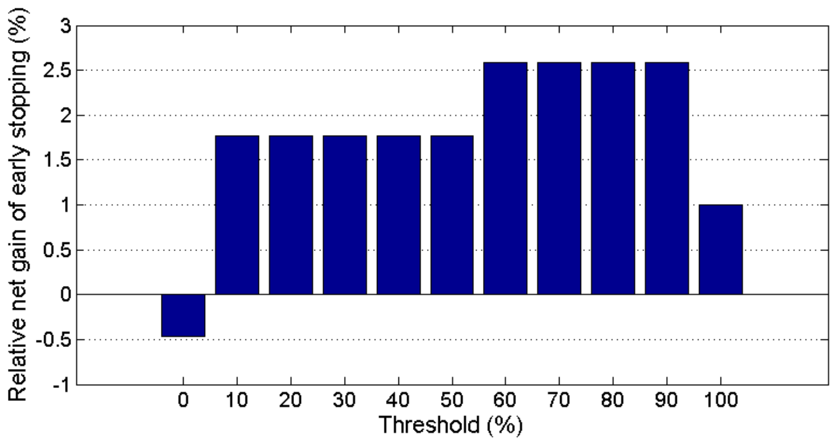


Figure 4. Relative net gain of early stopping

gathered sequentially by the monitoring system. Such structures can be represented by combinations of simple patterns including serial, parallel and cycle process.

Figure 5 illustrates a course of gathering task results in each of serial, parallel and cycle pattern. In a serial pattern of (a), results are gathered along with sequential executions. In an AND split of (b), both of t_2 and t_3 should be executed after t_1. Then, results can be gathered in one of sequences $x_1 \rightarrow x_2 \rightarrow x_3$ as (b), or $x_1 \rightarrow x_3 \rightarrow x_2$. As such, by the particular temporal sequences, the whole sets are filled gradually. In a cycle pattern of (c), after serial executions of t_1, t_2 and t_3, t_2 and t_3 are re-executed. As pointed out by [3], when tasks are re-executed, only the current logs from p_4 and p_5 are available so

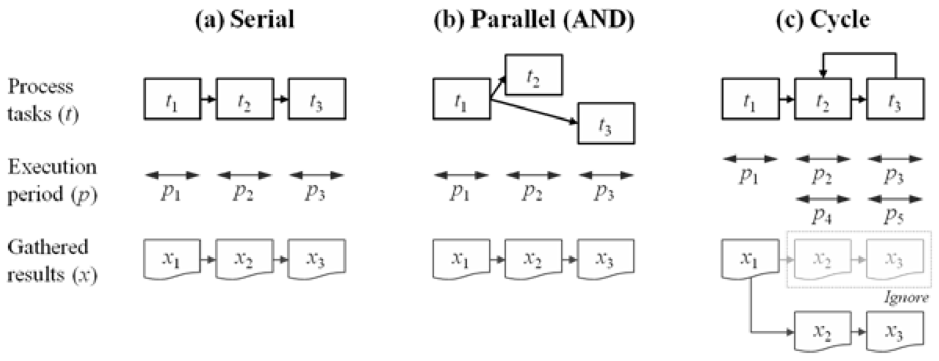


Figure 5. Information gathering in serial, parallel (AND), and cycle processes

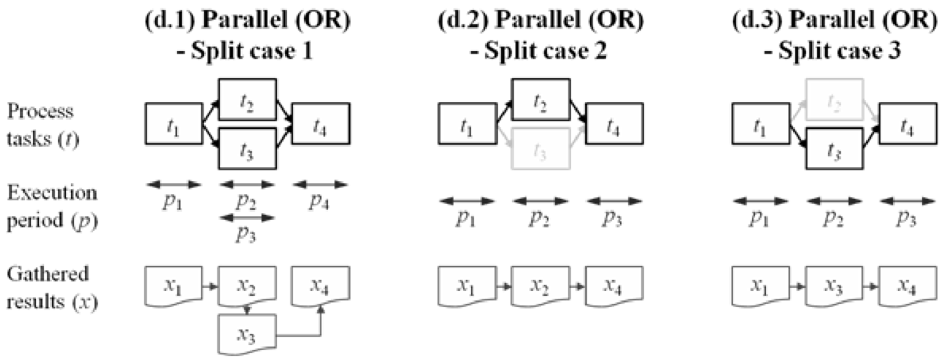

(d.1) Parallel (OR) - Split case 1
(d.2) Parallel (OR) - Split case 2
(d.3) Parallel (OR) - Split case 3
Process tasks (t)
Execution period (p)
Gathered results (x)

Figure 6. Information gathering in parallel (OR) processes

that the previous logs from p_2 and p_3 are ignored. Thus, it also can be considered as sequential gathering. Figure 6 shows three possible gathering orders from an OR split. After t_1, there are three progresses: executing both splits as (d.1), or executing one of them as (d.2) and (d.3). (d.1) is the same as the AND split. (d.2) and (d.3) are the same as the serial pattern, respectively. Therefore, the sequential gathering is also available. In addition, an Exclusive-OR split is the combination of (d.2) and (d.3) so that the sequence is under the same conditions.

From the viewpoint of gathered results until termination, independent execution paths can be extracted from the process structure. Then, for each path, task results are gathered sequentially in midcourse. Therefore, after extracting paths having an independent sequence of information gathering, the proposed approach can be applied to each path in the same way to the serial process so that it can be extended to various process structures.

8. Conclusions.

In this paper, we proposed an early warning strategy using expected gain analysis for real-time business process monitoring. Our approach is valuable in offering information on real-time progress and opportunities for proactive prevention of undesired process performance; as such, it can be used to support the profitable decision-making of industrial managers in real-time. Further work by which the quality of the proposed method may be improved remains. The most urgent is confirming the practicability of the method by applying it to real industry functions such as real-time monitoring of service processes executed in temporal sequences. Moreover, the early warning threshold should be tested experimentally with real data to improve its accuracy and promptness.

Acknowledgment. This work was supported by the Advanced Track of Green Production Processing for Reducing Greenhouse Gas Emission of the KETEP grant funded by Ministry of Knowledge Economy (No. 201119080289).

REFERENCES

[1] F. Buytendijk and D. Flint, How BAM can turn a business into a real-time enterprise, *Gartner Research, AV-15-4650*, 2002.
[2] W. Jiang, T. Au and K.-L. Tsui, A statistical process control approach to business activity monitoring, *IIE Transactions*, vol.39, no.3, pp.235-249, 2007.
[3] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal and M.-C. Shan, Business process intelligence, *Computers in Industry*, vol.53, no.3, pp.321-343, 2004.
[4] B. Kang, S. K. Lee, Y. Min, S.-H. Kang and N. W. Cho, Real-time process quality control for business activity monitoring, *Proc. of the 2009 International Conference on Computational Science and Its Applications*, Yongin, Korea, pp.237-242, 2009.
[5] M. Niazi, M. Wilson and D. Zowghi, A maturity model for the implementation of software process improvement: An empirical study, *Journal of Systems and Software*, vol.74, no.2, pp.155-172, 2005.
[6] D. Grigori, F. Casati, U. Dayal and M.-C. Shan, Improving business process quality through exception understanding, prediction, and prevention, *Proc. of the 27th Very Large Data Base Endowment Conference*, Roma, Italy, pp.159-168, 2001.
[7] U. S. Rao, S. Kestur and C. Pradhan, Stochastic optimization modeling and quantitative project management, *IEEE Software*, vol.25, no.3, pp.29-36, 2008.
[8] B. Kang, N. W. Cho and S.-H. Kang, Real-time risk measurement for business activity monitoring (BAM), *International Journal of Innovative Computing, Information and Control*, vol.5, no.11(A), pp.3647-3657, 2009.
[9] A. J. Beckett, C. E. R. Wainwright and D. Bance, Implementing an industrial continuous improvement system: A knowledge management case study, *Industrial Management & Data Systems*, vol.100, no.7, pp.330-338, 2000.
[10] C. Y. Ma and X. Z. Wang, Inductive data mining based on genetic programming: Automatic generation of decision trees from data for process historical data analysis, *Computers and Chemical Engineering*, vol.33, no.10, pp.1602-1616, 2009.
[11] J. R. Quinlan, Induction of decision trees, *Machine Learning*, vol.1, no.1, pp.81-106, 1986.

[12] A. H. L. Lim and C.-S. Lee, Processing online analytics with classification and association rule mining, *Knowledge-Based Systems*, vol.23, no.3, pp.248-255, 2010.

[13] S. Peddabachigari, A. Abraham, C. Grosan and J. Thomas, Modeling intrusion detection system using hybrid intelligent systems, *Journal of Network and Computer Applications*, vol.30, no.1, pp.114-132, 2007.

[14] B. Wetzstein, D. Karastoyanova and F. Leymann, Towards management of SLA-aware business processes based on key performance indicators, *Proc. of the 9th Workshop on Business Process Modeling, Development, and Support, Montpellier*, France, 2008.

[15] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar and F. Leymann, Monitoring and analyzing influential factors of business process performance, *Proc. of the 13th IEEE International Enterprise Distributed Object Computing Conference*, Auckland, New Zealand, pp.141-150, 2009.

[16] B. Curtis, G. V. Seshagiri, D. Reifer, I. Hirmanpour and G. Keeni, The cases for quantitative process management, *IEEE Software*, vol.25, no.3, pp.24-28, 2008.

[17] M. Castellanos, N. Salazar, F. Casati, U. Dayal and M.-C. Shan, Predictive business operations management, *International Journal of Computational Science and Engineering*, vol.2, no.5/6, pp.292-301, 2006.

[18] L. A. Rusinov, I. V. Rudakova and V. V. Kurkina, Real time diagnostics of technological processes and field equipment, *Chemometrics and Intelligent Laboratory Systems*, vol.88, no.1, pp.18-25, 2007.

[19] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar and F. Leymann, Runtime prediction of service level agreement violations for composite services, *The 3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing, LNCS*, Stockholm, Sweden, vol.6275, pp.176-186, 2010.

[20] B. Kang, J. Lee, S. Kim, D. Kim and S.-H. Kang, A clustering based incremental faulty-rate estimation algorithm for business process monitoring, *ICIC Express Letters*, vol.5, no.4(B), pp.1261-1266, 2011.

[21] B. Kang, J.-Y. Jung, N. W. Cho and S.-H. Kang, Real-time business process monitoring using formal concept analysis, *Industrial Management & Data System*, vol.111, no.5, pp.652-674, 2011.