Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 4, pp.49－58, December 2008.

의사결정나무를 이용한 비즈니스 프로세스의 실시간 위험 수준 측정

강복영* · 조남욱**† · 김훈태*** · 강석호*

*서울대학교 산업공학과
**서울산업대학교 산업정보시스템공학과
***대진대학교 산업시스템공학과

Real-time Risk Measurement of Business Process Using Decision Tree

Bokyoung Kang* · Nam Wook Cho**† · Hoontae Kim*** · Suk-Ho Kang*

*Department of Industrial Engineering, Seoul National University
**Department of Industrial and Information System Engineering, Seoul National University of Technology
***Department of Industrial and Systems Engineering, Daejin University

This paper proposes a methodology to measure the risk level in real-time for Business Activity Monitoring (BAM). A decision-tree methodology was employed to analyze the effect of process attributes on the result of the process execution. In the course of process execution, the level of risk is monitored in real-time, and an early warning can be issued depending on the change of the risk level. An algorithm for estimating the risk of ongoing processes in real-time was formulated. Comparison experiments were conducted to demonstrate the effectiveness of our method. The proposed method detects the risks of business processes more precisely and even earlier than existing approaches.

Keywords : Business Activity Monitoring, Risk Measurement, Decision Tree, Complex Event Processing

1. 서 론

급변하는 비즈니스 환경의 변화 속도에 적응하고 경쟁력을 창출하기 위해 현대의 기업들이 지향하고 있는 패러다임은 실시간 기업(Real Time Enterprise; RTE)이다. 경영 환경의 변화에 적응하기 자신들이 보유하고 있는 자산이자 곧 경쟁력의 밑거름이 되는 비즈니스 프로세스를 체계적으로 관리하기 위한 도구로 비즈니스 프로세스 관리시스템(Business Process Management System ; BPMS)을 도입하게 되었으며, 동시에 프로세스 관리의 수준 향상을 위해서는 비즈니스 프로세스에 대한 실시간 모니터링의 중요함을 인식하게 됨에 따라 비즈니스 활동 모니터링(Business Activity Monitoring, BAM)에 대한 수요가 증가하였다[1, 12].

BAM이란 프로세스를 구성하는 요소들의 상태와 변화를 관측하여 무슨 일이 일어나고 있는지를 사용자에게 보여주는 시스템이다[3, 8]. 기업의 업무 프로세스의 활동에서 발생하는 변화, 기회, 위협 및 다양한 트랜잭

논문접수일 : 2008년 05월 30일 논문수정일 : 2008년 08월 23일 게재확정일 : 2008년 10월 02일
† 교신저자 nwcho@snut.ac.kr
※ 본 연구는 지식경제부 및 정보통신연구진흥원의 IT 핵심기술개발사업 일환으로 수행하였음(2008-S-018-01, u-City Service용 개방형 SW 플랫폼 개발).

션 등을 이벤트라고 정의하면, 이 이벤트를 빠르고 정확하게 감지하여, 내부 프로세스의 상태를 투명하게 모니터링 함으로써 빠른 대응을 가능하게 하는 것이 BAM의 핵심적인 기능이다.

이벤트 처리란 프로세스에서 발생하는 이벤트를 관측하고 분석하여 그에 따른 적절한 행동을 실행하기 위한 기술이다. 초기의 이벤트 처리 연구는 프로세스 진행 과정에서 발생하는 모든 이벤트에 순서를 부여하고 상관관계를 밝히는 것을 목적으로 하였다[11, 13, 15, 16, 20]. 이와 같은 이벤트 처리 기술이 BAM과 접목하여, 프로세스가 진행되면서 발생하는 이벤트들의 상관관계가 결과에 미치는 영향을 분석하여 효과적인 모니터링을 하기 위한 기술이 개발되었고[14], 이벤트 처리 기반 모니터링 시스템이 제안되었다[1, 17]. 일반적인 이벤트 처리 기반 모니터링 시스템은 룰 추출을 이용한 접근 방식을 사용하였다. 룰 기반 시스템은 과거 프로세스 정보를 바탕으로 프로세스의 결과가 비정상적으로 종료되게 하는 결정적인 요인들을 추출하며, 룰은 'If (condition)-Then (action)'으로 표현한다[2, 4-6, 10]. 만약 비정상적인 결과를 초래하는 원인이 등장하면 이를 감지하고 경보를 내린다. 룰 기반 경보 시스템의 성능은 'If-Then' 룰의 정확도에 따라 결정된다[9]. 또는 결정적인 위험 요인들과의 유사도 측정으로 경보 여부를 결정할 수 있다[1, 19]. 룰 기반 시스템은 룰의 수정과 추가가 용이하여 시스템의 유지보수가 간단하다는 장점이 있다. 하지만 프로세스의 진행 단계에 따라 실시간으로 위험 수준을 표현하기 힘들다는 한계점이 있다. 일반적인 룰 기반 시스템에서 측정되는 위험 수준은 프로세스가 진행된 시점까지 룰의 조건에 해당하는 이벤트가 발생하지 않으면 위험 수준을 측정할 수 없으며 해당 이벤트가 발생해야만 룰의 조건의 충족 여부가 판단이 가능하다. 이때 만족되지 않으면 0%, 만족되면 100%가 되어 경보를 내린다. 따라서 기존에 제안된 룰 기반 시스템[1, 17]은 위험에 대한 조기 경보(early warning)가 힘들 수 있다. 즉, 사전에 정의된 룰이 감지되고 난 뒤에 경보를 발생시키는 원리에 의해 이미 프로세스의 비정상적인 결과를 초래하는 원인이 발생하고 난 뒤에야 경보를 발생시켜 실시간 모니터링의 효율이 떨어질 우려가 있다.

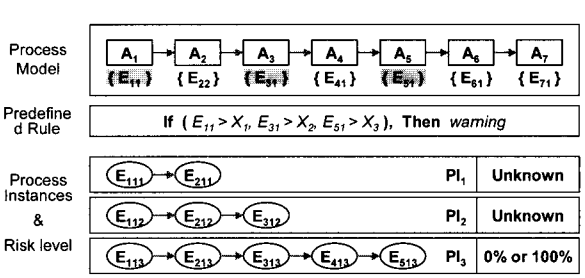
따라서 본 논문은 기존의 접근 방법들의 한계점을 해결하기 위해 비즈니스 프로세스의 실시간 위험 수준 측정 방법론을 제안한다. 실시간으로 진행 중인 프로세스의 위험 수준을 매 관측 시점마다 0~100% 구체적인 수치로 측정하기 위한 방안을 제시한다. 이를 위해 의사결정나무를 이용한 실시간 위험 수준 측정 알고리즘을 제안하였다. 제안된 알고리즘을 적용함으로써 BAM의 사용자는 실시간으로 진행 중인 비즈니스 프로세스의 위험 수준을 관찰할 수 있다. 이 위험 수준의 변화를 살펴 한계값을 초과하는 경우 경보를 내림으로써 기존의 룰 기반 시스템이 경보를 내릴 수 있는 시점보다 빠르게 위험 요소의 발생을 예측하고 조기 경보를 내릴 수 있다.

2. 비즈니스 활동 모니터링

본 논문에서는 프로세스 모델이 BPMS 환경 내에서 정의되고 실행된다고 가정한다. 프로세스 모델이란 프로세스의 구성 요소들을 정의하고, 프로세스의 목적을 달성하기 위하여 구성 요소들이 어떤 순서로 어떻게 실행되어야 하는지를 명시하기 위한 모델이다[18]. 일반적으로 프로세스 모델은 활동(Activity)의 흐름으로 표현된다. 활동이 실행되면 그에 해당하는 이벤트가 발생하고, 이 발생한 이벤트는 프로세스 또는 활동의 상태를 나타내는 속성값의 정보를 보유한다. 정의된 프로세스 모델이 실행됨으로써, 프로세스 인스턴스가 생성된다. 프로세스 인스턴스는 프로세스 모델에 의해 생성된 활동들의 흐름 또는 이벤트들의 흐름이라 정의된다. 본 논문에서는 프로세스 인스턴스를 프로세스 모델이 실행되면서 발생하는 이벤트의 집합으로 정의한다. 본 논문에서 사용하는 표현들을 다음과 같다.

- PI_k : k번째 프로세스 인스턴스($1 \leq k \leq n$)
- A_i : i번째 활동($1 \leq i \leq l$)
- E_{ijk} : k번째 프로세스 인스턴스의 i번째 활동에 의해 발생한 j번째 이벤트의 속성 값($1 \leq j \leq m$)

다음 <그림 1>의 예제 프로세스를 통해 프로세스 모델과 그에 의해 생성된 프로세스 인스턴스들이 갖는 특성을 살펴보자. 예제 프로세스는 7개의 활동으로 구성되어 있으며($l = 7$), 각 활동은 하나의 이벤트를 발생시킨다고 가정하였다(m = 1). 또한 프로세스 인스턴스들을 기존의 룰 기반 시스템 환경 하에서 모니터링하는 상황을 가정한다.

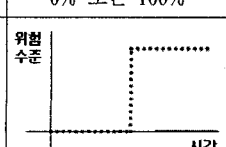
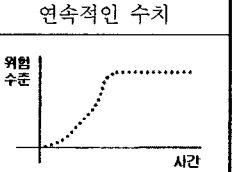


<그림 1> 프로세스 모델과 프로세스 인스턴스

<그림 1>의 프로세스 인스턴스는 모두 진행 중이며 따라서 제한된 정보만을 보유하고 있다. 또한 진행정도에 따라 이벤트 정보의 차이를 보이게 된다. PI_1은 $\{E_{111}, E_{211}\}$의 이벤트가 발생하였고 그 속성 값을 보유하고 있다. PI_2는 추가로 $\{E_{312}\}$의 이벤트 속성값을 보유한다. PI_3은 $\{E_{412}, E_{513}\}$의 이벤트 속성값을 추가로 보유한 것으로 관측되었다. 만약 사전에 정의된 룰, 'If($E_{11} > X_1$, $E_{31} > X_2$, $E_{51} > X_3$), Then warning',을 이용하여 모니터링한다면 실시간으로 진행 중인 프로세스 인스턴스는 그 진행정도에 따라 제한된 이벤트 속성값만을 보유하고 있으므로, 룰 기반 시스템에서는 위험 수준의 평가와 경보 여부의 결정이 힘들 수 있다. 즉, PI_1과 PI_2가 각각의 진행 시점까지 관측된 속성값으로는 룰의 조건을 만족시키는지의 여부를 평가할 수 없다. 따라서 각 관측 시점의 제한된 정보로는 위험 수준의 평가 및 경보 여부 결정이 불가능하다. 이때의 위험 수준을 'unknown' 또는 0%라 할 수 있다. PI_3의 경우, 관측된 속성값들을 통해 룰의 조건에 명시된 이벤트의 조건을 만족시키는 지를 판단할 수 있다. 하지만, 룰의 조건이 프로세스 진행 상태에서 후반부에 발생하는 이벤트 속성값의 판단을 필요로 한다면 경보가 더욱 무의미해지는 상황이 발생할 수 있다.

따라서 본 연구에서는 매 관측 시점마다 보유하고 있는 제한된 이벤트 정보만을 바탕으로 프로세스 인스턴스의 위험 수준을 의미 있는 수치로 측정하는 알고리즘을 제안한다. 제안된 알고리즘은 의사 결정나무를 이용한 Grigori *et al.*[9]의 연구가 가지는 룰 기반 시스템의 문제점을 극복할 수 있다. 즉, 위험 수준의 변화를 살펴 기존의 BAM 시스템보다 빠르게 위험 수준의 상승을 감지하고 경보의 여부를 결정하게 된다. 기존 시스템과 본 논문이 제안하는 실시간 조기 경보 비즈니스 활동 모니터링 시스템의 차이를 <표 1>에 요약하였다.

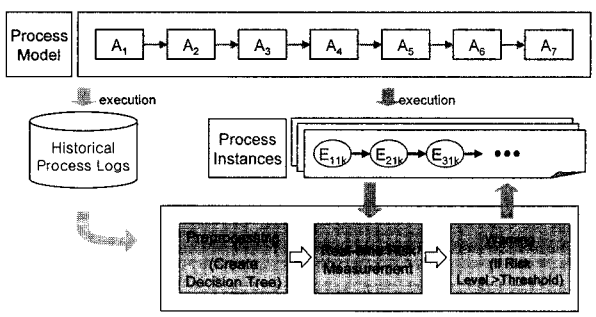
<표 1> 룰 기반 경보 BAM과 실시간 조기 경보 BAM의 비교

	룰 기반 경보 BAM	실시간 조기 경보 BAM
위험 수준 측정 시점	결정적인 위험 요소가 등장하였을 때만 가능	이벤트 속성값이 입력되는 매 시점
경보 발령 시점	결정적인 위험 요소가 등장하였을 때만 가능	룰 기반의 경보 BAM이 가능한 시점보다 빠르게 감지 가능
위험 수준 측정 수치	unknown, 0% 또는 100%	0%부터 100%까지 연속적인 수치
경보 대상 프로세스 인스턴스의 위험 수준 변화		

3. 실시간 조기 경보 비즈니스 활동 모니터링

3.1 시스템 아키텍처

본 절에서는 프로세스 모델로부터 실행된 과거 프로세스 로그(Historical Process Logs)를 바탕으로 의사결정나무를 생성한다. 또한 의사결정나무를 이용한 위험 수준 측정 알고리즘을 제시하여 실시간으로 진행 중인 프로세스 인스턴스들의 전 관측 시점에 걸친 위험 수준을 구체적인 수치로 측정한다. 위험 수준의 변화를 관측하여 한계값을 초과하는 프로세스 인스턴스에 대해서는 경보를 발생시킨다. 경보 발령의 근거가 되는 한계값은 프로세스 인스턴스의 속성값의 분포에 의존적이다. 따라서 실험적으로 설정하여야 한다. 실시간 조기 경보 BAM 아키텍처는 다음 <그림 2>와 같다.



<그림 2> 실시간 조기 경보 BAM의 아키텍처

3.2. 프로세스 인스턴스의 실시간 위험 수준 측정

3.2.1 과거 프로세스 로그의 전처리

알고리즘의 선행 단계로, 먼저 과거 프로세스 로그로부터 의사결정나무를 생성한다. 의사결정나무를 생성하기 위해서는 해당 프로세스의 과거 실행 데이터가 필요하다.

<표 2> 과거 프로세스 로그 테이블

Process Instance	A_1	A_2	A_3	A_4	A_5	A_6	A_7	결과
PI_1	E_{111}	E_{211}	E_{311}	E_{411}	E_{511}	E_{611}	E_{711}	정상/ 비정상
…	…	…	…	…	…	…	…	…
PI_n	E_{11n}	E_{21n}	E_{31n}	E_{41n}	E_{51n}	E_{61n}	E_{71n}	정상/ 비정상

<그림 1>에 제시된 예제 프로세스의 과거 프로세스 로그는 <표 2>의 형태로 n개의 프로세스 인스턴스들과

그 속성 값들로 구성되어 있다.

각 프로세스 인스턴스(PI_k)는 이벤트 속성값(E_{ijk})과 결과 정보(정상/비정상)를 보유한다. 정상적으로 종료되었을 경우 '정상', 비정상적으로 종료되었을 경우 '비정상'의 값을 갖는다. 과거 프로세스 로그를 테이블의 형태로 입력받아 의사결정나무를 생성할 수 있다[9]. 즉, 과거 프로세스 로그는 의사결정나무의 훈련집합(training set)이 된다. 프로세스 인스턴스들의 결과는 목표(target), 이벤트의 속성값들을 특성(feature)에 해당한다. 의사결정나무의 각 노드(node)에 들어가게 될 특성값의 쿼리(query)문의 선택은 엔트로피 불순도(entropy impurity)를 따른다. 엔트로피 불순도란 각 노드에서의 불순도를 엔트로피지수를 통해 측정하고, 분기 이후 엔트로피 지수의 감소가 가장 큰 쿼리문을 분기 기준으로 삼는 방법이다[7, 21]. 루트노드(root node)에서 단말노드(leaf node)까지 상의하달(top-down) 방식으로 분기되면서 최종 단말노드에 도달하면, 하나의 가지에 해당하는 특성의 분기 조건들 즉, 프로세스 인스턴스의 이벤트의 속성값들의 조건에 해당되는 프로세스 인스턴스의 위험수준을 알 수 있다[9]. 의사결정나무를 구축하게 되면 단말노드는 루트노드로부터 해당 노드까지의 분지조건(branching condition)들의 교집합에 해당하는 과거 프로세스 인스턴스들의 결과 정보를 보유하고 있다. 이를 이용하여 프로세스 인스턴스의 위험 수준을 정의한다.

- x_{normal} : 단말노드의 정상 종료 프로세스 인스턴스의 수
- $x_{abnormal}$: 단말노드의 비정상 종료 프로세스 인스턴스의 수
- 위험 수준 = $\frac{x_{abnormal}}{x_{abnormal}+x_{normal}}$, (단말 노드에 이르는 가지의 속성값 조건들을 만족하는 프로세스 인스턴스의 위험 수준)

위험 수준을 이와 같이 정의했을 때, 의사결정나무의 생성은 단말노드에서 위험 수준이 80% 이상 또는 20% 미만일 때까지 생성하였다. 즉, 의사결정나무의 분류 정확도가 80% 이상일 때와 동일하다.

3.2.2 의사결정나무를 이용한 실시간 위험 수준 측정 알고리즘

본 절에서는 의사결정나무를 이용한 실시간 위험 수준 측정 알고리즘을 설명한다. 실시간으로 진행 중인 프로세스 인스턴스의 관측된 이벤트 속성값들과 앞 절에서 설명한 의사결정나무를 이용하여 위험 수준을 측정할 수 있다. 의사결정나무의 노드들은 이벤트 속성값의 쿼리문들로 이루어져 있다. 그러므로 루트노드부터 단말 노드까지 상의하달 방식으로 의사결정나무를 탐색할 수 있다. 하지만 구축된 의사결정나무는 종료된 과거 프로세스 이력으로부터 추출한 의사결정나무이므로 진행 중인 프로세스 인스턴스로부터 관측된 정보와 차원(dimension)이 다르다. 과거 프로세스 이력의 차원은 하나의 프로세스 인스턴스가 보유한 총 이벤트 속성 값의 수이다. 하지만 실시간으로 진행 중인 프로세스 인스턴스의 차원은 그 이하다. 따라서 일반적인 의사결정나무의 탐색 방법으로는 진행 중인 프로세스의 위험 수준을 측정하는 것은 불가능하다. 따라서 다음과 같은 방법을 제안한다.

제한된 이벤트 정보, 즉 관측 시점에서 프로세스 인스턴스가 보유한 이벤트 속성값만으로 의사결정나무의 가지를 탐색하면 노드에서 이벤트 값의 조건에 의한 분기를 결정할 때, 다음 2가지 상황 중 하나가 발생할 수 있다.

- **Case I** : 노드의 쿼리문이 관측된 이벤트 정보의 쿼리문인 경우이다. 이때는 관측된 이벤트 속성값을 이용하여 분기를 직접 결정할 수 있다.
- **Case II** : 노드의 쿼리문이 관측된 이벤트 정보의 쿼리문이 아닌 경우이다. 즉, 아직 발생하지 않은 이벤트 정보의 쿼리문인 경우이다. 실시간으로 진행 중이라는 특성으로 현 시점에서는 분기를 결정할 수 없어 의사결정나무의 가지를 더 이상 따라갈 수 없다. 이때, 해당 분기 조건에서 과거 프로세스 이력의 분기 비율로 분기하고 복수개의 가지를 따라 탐색을 계속 진행한다.

- N : 속성값이 관측되지 않아 분기를 결정할 수 없는 노드
- N_y : N의 쿼리를 만족하는 분기 노드
- N_n : N의 쿼리를 만족하지 않는 분기 노드
- $P_y = \frac{n\left(\left\{\begin{matrix}N\text{과 }N\text{의 모든 부모 노드들의}\\ query\text{를 만족하는 인스턴스들}\end{matrix}\right\}\right)}{n\left(\left\{\begin{matrix}N\text{의 모든 부모 노드들의}\\ query\text{를 만족하는 인스턴스들}\end{matrix}\right\}\right)}$, ($N_y$로의 분기 비율)
- $P_n = 1\text{-}P_y$, (N_n으로의 분기 비율)

같은 방식으로 단말노드까지 탐색한다. Case II가 발생하면 복수개의 가지를 탐색하게 되고, 따라서 복수개의 단말노드에 도달할 수 있다.

탐색이 완료되면, 탐색한 모든 단말노드로부터 하의상달(bottom-up) 방식으로 모든 위험 수준을 N_y와 N_n의 비율로 가중합(weighted sum)을 구하면, 실시간으로 진행 중인 프로세스 인스턴스의 위험 수준을 측정할 수 있다. 2분기 의사결정나무의 경우 다음 <표 3>의 알고리즘을 따른다.

<표 3> 실시간 위험 수준 측정 알고리즘

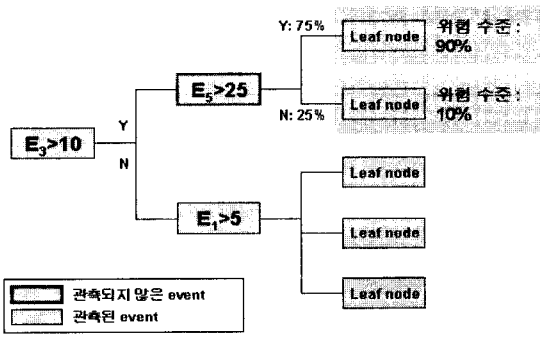
```
1: N = the root node
2: procedure RiskLevel(observed attributes E, node N)
3:     if N is the leaf node
4:         then return(the risk level)
5:     elseif the split of N can be determined
6:         then call RiskLevel(E, the next node after split)
7:     else
8:         return P_y*RiskLevel(E, N_y) + P_n*RiskLevel(E, N_n)
9: end RiskLevel
```

생성한 의사결정나무에서 각 가지에서 노드의 속성값들의 순서는 프로세스 모델에서 발생하는 순서와 다를 수 있다. 의사결정나무에서의 순서는 속성값들이 결과에 미치는 영향에 따라 다양하게 나올 수 있기 때문이다. 이와 같은 순서의 불일치로, 실시간으로 진행 중인 프로세스 인스턴스의 제한된 정보를 바탕으로 의사결정나무를 탐색할 때, 분기의 결정이 불가능한 상황이 발생하는 것이다. 이 알고리즘의 의의는 이러한 정보 부재의 상황에서 과거 프로세스 이력으로부터 추출한 사전 확률을 이용한다는 것이다. 알고리즘에서 정의한 P_y와 P_n은 관측된 프로세스 인스턴스가 앞으로 진행될 수 있는 각 방향으로의 확률을 의미한다. 따라서 본 알고리즘은 과거 정보를 바탕으로 앞으로의 진행을 예측하여 현재의 위험 수준을 측정한다. 동시에 의사결정나무의 정확성을 잃지 않으면서 제한된 정보만으로도 의사결정나무의 탐색이 가능하다.

프로세스 네트워크에서 발생 가능한 여러 상황들을 과거 데이터가 많이 보유하고 있을수록 이 알고리즘이 실시간으로 측정한 위험 수준의 신뢰도는 높아진다. 과거 프로세스 이력에서 발견되지 않은 새로운 형태의 프로세스 오작동에 대해서는 대처가 힘들다. 하지만 추가적으로 의사결정나무를 다시 학습하여 생성시킴으로써 추후의 대처가 가능하다. 그리고 프로세스의 진행상 종료 시점에 가까워질수록 즉, 보유 데이터가 늘어날수록 측정된 위험 수준은 신뢰도가 높아지는 특성을 갖는다.

아래 <그림 3>과 같이 $\{E_1, E_2, E_3, E_4, E_5, E_6, E_7\}$의 이벤트들의 속성 값으로 구성된 의사결정나무를 생성하였다고 하자. 현재 진행 중인 프로세스 인스턴스의 관측값은 $\{E_1, E_2, E_3\}$ = {5, 10, 20}으로 가정한다.

루트노드의 분기 조건이 '$E_3 > 10$'이고 'Y'로 분기된다(**Case I**). E_5의 노드로 진행된다. 이 노드에서의 조건이 '$E_5 > 25$'이다. 이때, 선행 노드의 조건($E_3 > 10$)을 만족시키는 과거 프로세스 로그를 추출해서 '$E_5 < 25$'의 노드에서의 분기 비율을 찾는다(**Case II**). E_5의 노드 이후, E_4의 노드에서도 같은 방법의 가중합 방식을 수행한다. E_5의 분기이후 하위 노드들에서의 총 위험 수준은 각각 90%와 10%였다. 따라서 실시간으로 진행 중인 프로세스 인스턴스, $\{E_1, E_2, E_3\}$={5, 10, 20}의 위험 수준은 70%(0.75 * 90% + 0.25 * 10%)로 측정된다.

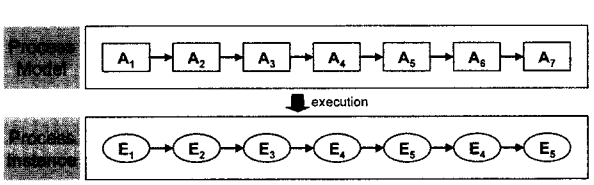


<그림 3> 의사결정나무를 이용한 실시간 위험 수준 측정

4. 실험 및 결과 분석

본 장에서는 룰 기반 경보 BAM과 본 연구에서 제안된 실시간 조기 경보 BAM간의 비교를 위하여 실험을 수행하였다. 실험의 목적은 두 가지이다. 먼저 본 논문이 제안하는 알고리즘을 이용하여 실시간으로 진행 중인 프로세스 인스턴스의 상태를 구체적이고 유의한 수치(0~100%)로 측정할 수 있음을 보이는 것이다. 그리고 동일한 비정상 종료 프로세스 인스턴스에 대해서 기존의 시스템보다 본 논문이 제안하는 시스템이 보다 빠르게 위험 수준의 상승을 감지하고 경보의 시점을 앞당길 수 있음을 보여주고 그 효용성을 입증하고자 한다.

실험을 위해 <그림 4>의 프로세스 모델을 가정하였다. 프로세스 인스턴스는 7개의 활동으로 구성되고 각 활동마다 하나의 이벤트가 발생한다($l = 7$, $m = 1$). 각 이벤트가 순차적으로 발생하며, 이벤트 값이 발생하는 매 시점을 관측시점으로 정의하였다. 따라서 각 프로세스 인스턴스마다 7개의 관측시점이 존재하며 7단계에 걸쳐서 위험 수준의 변화를 살펴볼 수 있다.



<그림 4> 프로세스 모델과 프로세스 인스턴스

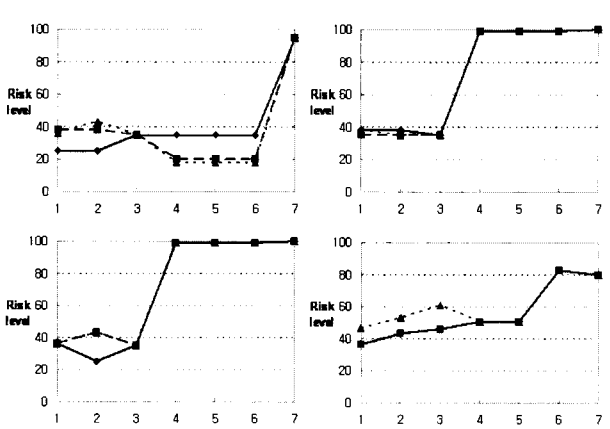
동일한 프로세스 인스턴스를 대상으로 한 룰 기반 기

법과 제안하는 기법의 작동 방식을 비교하기 위해, 실험에 사용한 데이터들은 4가지 비정상 종료 룰(Critical Failure Factors, CFFs)과 4가지 정상 종료 룰(Critical Success Factors, CSFs)을 가정하고 그 룰을 기반으로 생성하였다.

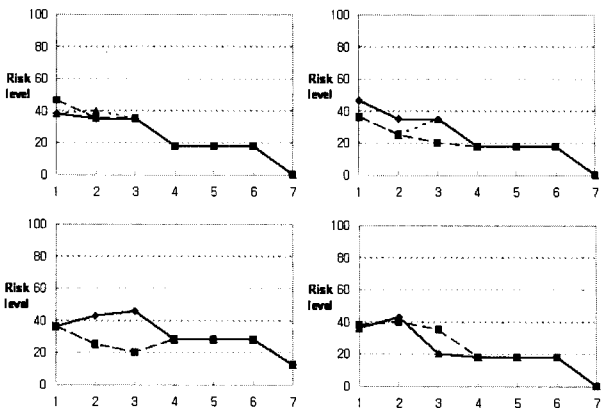
훈련집합으로 총 200개의 과거 프로세스 인스턴스가 의사결정나무의 생성에 사용되었다. 132개의 정상 종료 프로세스 인스턴스와 68개의 비정상 종료 프로세스 인스턴스로 구성되어 있다. 시험집합(test set)에 해당하는 프로세스 인스턴스는 12개의 정상 종료 프로세스 인스턴스, 12개의 비정상 종료 프로세스 인스턴스로 구성되었다. 각 룰마다 3개의 프로세스 인스턴스가 사용되었다. 별첨에 실험 데이터를 첨부하였다.

<그림 5>와 <그림 6>은 각각 실시간으로 진행 중인 12개의 시험집합 프로세스 인스턴스들의 위험 수준 변화를 나타낸다. 본 논문이 제안하는 알고리즘을 통하여 프로세스 인스턴스의 전 관측 시점에 걸쳐 위험 수준을 수치로 측정하였다. 각 그래프는 동일한 룰에 의해 생성된 3개의 프로세스 인스턴스의 위험 수준 변화를 나타낸다.

<그림 5>와 <그림 6>을 통하여 프로세스 인스턴스가 시작 초기에는 어느 정도의 위험 수준을 갖고 있다는 것을 알 수 있다. 보유한 정보가 제한적이라는 특성으로 프로세스 인스턴스의 결과에 대한 불확실성을 내포하고 있는 것이다. 하지만 프로세스 인스턴스가 진행되고 보유 정보량이 늘어나면서 위험 수준은 그 종료의 성격에 따라 일정한 변화 추세를 가진다는 것을 관찰할 수 있다. <그림 5>의 비정상 종료 프로세스 인스턴스의 위험 수준은 종료 시점으로 진행될수록 점차 상승하는 추세를 보이고 있으며, 반대로 <그림 6>의 정상 종료 프로세스 인스턴스의 위험 수준은 점차 하락하는 추세이다.



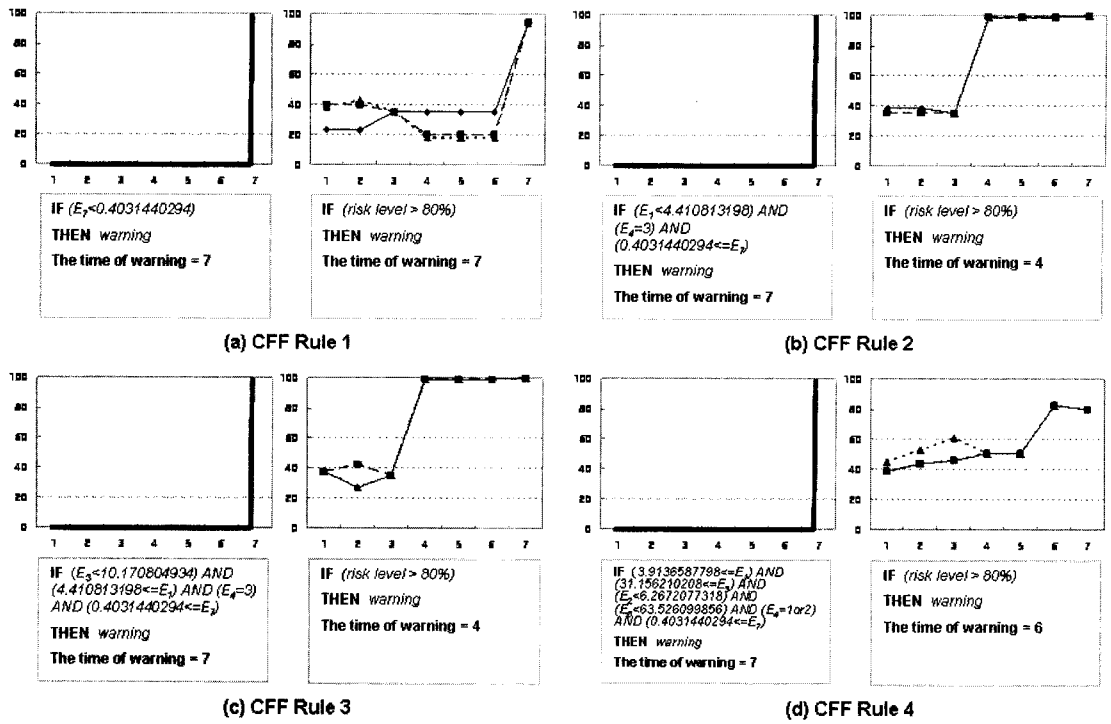
<그림 5> 비정상 종료 프로세스 인스턴스의 위험 수준 변화



<그림 6> 정상 종료 프로세스 인스턴스의 위험 수준 변화

경보 발령의 근거가 되는 한계값(threshold)은 프로세스 의존적이기 때문에 실험을 통해 결정하게 된다. 실험에 사용된 프로세스 인스턴스들의 위험 수준 추이를 살펴 한계값을 설정한다. 실험 결과, 모든 정상 종료 프로세스 인스턴스들의 모든 시점에서의 위험 수준의 최고 값은 46.3%였다. 그리고 모든 비정상 종료 프로세스 인스턴스들의 최종 시점에서의 위험 수준의 최저값은 80%이다. 한계값은 이 두 값의 사이에 위치하도록 정의한다. 이는 정상 종료 프로세스 인스턴스에 대한 경보(false alarm)를 방지하고, 비정상 종료 프로세스 인스턴스에 대해서는 종료되기 전에 경보를 발령하기 위함이다. 따라서 본 실험에서는 경보 발령 한계값을 80%로 정의하였다.

본 논문이 제안하는 시스템의 경보 시점은 기존 룰 기반 시스템의 경보 시점과 같거나 빠를 수 있다. <그림 7>은 한계 값이 80%일 때, 두 시스템의 경보 시점 비교 실험의 결과이다. 실험을 위해 가정한 데이터는 4가지의 CFFs에 의해 생성되었는데, 각 룰에 해당하는 프로세스 인스턴스들의 측정된 위험 수준을 나타내었다. 각 그림의 왼쪽은 기존의 룰 기반 시스템이 프로세스 인스턴스를 실시간으로 관측하였을 때, 측정되는 위험 수준과 그에 따른 경보 시점이다. 오른쪽은 제안하는 시스템에 의한 것이다. 왼쪽의 경우, CFFs의 룰이 모두 E_7의 조건을 포함하고 있다. 이와 같이 프로세스 인스턴스의 진행 후반부에 룰이 정의된 경우, 그 이전 시점까지 진행된 프로세스 인스턴스의 위험은 감지하지 못하는 경우가 발생한다. 하지만 본 논문의 알고리즘으로 측정된 위험 수준은 구체적이고 연속적인 수치로 측정되었으며, 위험을 품고 있는 프로세스 인스턴스에 대해서는 위험 수준이 점차 상승하는 추세로 측정되어 이를 조기에 감지할 수 있다. 4가지 형태의 결과를 살펴보면, 제안하는 시스템은 적어도 기존의 시스템보다는 빠르게



<그림 7> 룰 기반 경보 BAM과 실시간 조기 경보 BAM의 경보 시점 비교

경보를 내릴 수 있다. 단, 한계값을 100%로 설정할 경우, 기존 시스템과 경보 시점은 같게 된다.

5. 결론 및 추후 연구 과제

본 논문에서는 실시간 조기 경보 BAM을 위한 비즈니스 프로세스의 실시간 위험 수준 측정 방법론을 제안하였다. 제안된 방법론은 실시간으로 진행 중인 프로세스 인스턴스의 전 관측 시점에서 위험 수준을 구체적인 수치(0~100%)로 측정하며 측정된 위험 수준의 변화를 모니터링하여 정의된 한계값을 넘어설 경우 경보를 발생한다.

기존의 룰 기반 시스템은 위험 수준을 측정하기 보다는 위험 요소가 등장하였는지를 감지하는 것이 목적이기 때문에 룰의 조건이 등장할 경우에만 이를 감지하고 경보를 내리는 데 반해, 본 논문에서 제안한 방법론은 이진(binary) 형태의 감지 여부가 아닌 연속적인 수치로 프로세스의 위험수준을 측정한다. 이를 위해 실시간으로 진행 중인 프로세스 인스턴스의 제한된 이벤트 속성값 정보만으로 의사결정나무를 탐색하여 위험 수준을 측정하는 방법론을 제시하였다. 의사결정나무의 탐색 과정에서 관측되지 않은 속성값의 처리를 위하여 과거 프로세스 로그로부터 사전 확률을 추출하여 탐색하였다. 측정된 위험 수준은 의사결정나무의 단말노드, 즉 가지에 해당하는 룰의 조건에 따른 결과들의 조합으로 나타난다. 실험을 통하여 기존의 시스템보다 경보의 시점이 빠를 수 있음을 입증하였다.

추후 연구 과제로는 측정된 위험 수준의 정확성과 개량의 문제가 있다. 위험 수준의 신뢰도는 프로세스 인스턴스의 진행 정도가 종료 시점에 가까울수록 높아지는 반면, 상대적으로 적은 정보를 바탕으로 하는 경우(프로세스 인스턴스의 진행 정도가 초반부인 경우)에는 측정된 위험 수준의 값은 신뢰도가 떨어진다. 또한 위험 수준의 신뢰도는 의사결정나무 자체의 정확성에 절대적으로 영향을 받기 때문에 이의 개선이 중요하다. 그리고 프로세스를 구성하는 이벤트들 사이의 연관관계나 각 이벤트의 중요도를 고려해서 평가한다면 더욱 정확하게 위험 수준을 측정할 수 있을 것이다. 의사결정나무를 변형한다거나 다른 데이터마이닝 기법 또는 통계 기법을 적용시켜 그 결과를 비교해볼 수 있다.

또한 프로세스의 흐름을 고려한 위험 수준 측정의 연구가 필요하다. 분기 조건이나 루프, 그리고 분기 이후 정보의 동기화 문제를 고려해야 한다. 그리고 의사결정나무를 이용한 학습을 통해 룰을 찾는 사전 데이터 처리가 필요한 알고리즘이므로, 과거 프로세스 데이터가 추가되는 것에 대한 업데이트의 문제가 있다. 프로세스의 소요 시간에 따라 일정 주기마다 배치형식으로 업데

이트를 해야 한다. 그 적절한 주기에 대한 연구가 필요하다.

참고문헌

[1] 손성호, 정재윤, 조남욱, 강석호; "프로세스 기반 이벤트 분석을 이용한 비즈니스 활동 모니터링", 한국전자거래학회지, 12 : 219-231, 2007.

[2] Abadeh, M. S., Habibi, J., Barzegar, Z., and Sergi, M.; "A parallel genetic local search algorithm for intrusion detection in computer networks," *Engineering Applications of Artificial Intelligence*, 20(8) : 1058-1069, 2007.

[3] Adams, P.; "What's going on? [business activity monitoring]," *Manufacturing Engineer*, 81(6) : 282-283, 2002.

[4] Adi, A., Botzer, D., and Etzion, O.; "Semantic Event Model and Its Implication on Situation Detection," *Proceedings of the 2000 European Conference on Information*, 2000.

[5] Albaghdadi, M., Briley, B., and Evens, M.; "Event storm detection and identification in communication systems," *Reliability Engineering and System Safety*, 91 (5) : 602-613, 2006.

[6] Camci, F. and Chinnam, R. B.; "General support vector representation machine for one-class classification of non-stationary classes," *Pattern Recognition*, 41(10) : 3021-3034, 2008.

[7] Duda, R. O., Hart, P. E. and Stork, D. G.; *Pattern Classification*, Wiley-interscience, 2001.

[8] Golfarelli, M., Rizzi, S., and Cella, I.; "Beyond data warehousing : what's next in business intelligence?," *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, : 1-6, 2004.

[9] Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., and Shan, M-C.; "Business Process Intelligence," *Computers in Industry*, 53 : 321-343, 2004.

[10] Gross, P. N., Gupta, S., Kasier, G. E., Gaurav, S. K., and Parekh, J. J.; "An Active Events Model for Systems Monitoring," *Working Conference on Complex and Dynamic Systems*, 2001.

[11] Lamport, L.; "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, 21(7) : 558-565, 1978.

[12] Leymann, F., Roller, D., and Schmidt, M-T.; "Web Services and Business Process Management," *IBM Systems Journal*, 41(2), 2002.

[13] Luckham, D. C.; *The Power of Events*, Person Education, 2002.

[14] Luckham, D. C. and Frasca, B.; "Complex Event Processing in Distributed Systems," *Stanford University*, 1998.

[15] Lundberg, A.; "Leverage Complex Event Processing to Improve Operational Performance," *Business Intelligence*, 11(1) : 55-65, 2006.

[16] Mendling, J., Verbeek, H. M. W., van Dongen, B. F., van der Aalst, W. M. P., and Neumann, G.; "Detection and prediction of errorsin EPCs of the SAP reference model," *Data and Knowledge Engineering*, 64(1) : 312-329, 2008.

[17] Perrochon, L., Mann, W., Kasriel, S., and Luckham, D. C.; "Event Mining with Event Processing Networks," *Proceedings of Methodologies for Knowledge Discovery and Data Mining : Third Pacific-Asia Conference*, 1574 : 474-478, 1999.

[18] Rolland, C.; "Modeling the Requirements Engineering Process," *3rd European-Japanese Seminar on Information Modelling and Knowledge Bases*, 1993.

[19] Sharma, A., Pujari, A. K., and Paliwal, K. K.; "Intrusion detection using text processing techniques with a kernel based similarity measure," *Computers and Security*, 26 : 488-495, 2007.

[20] Wang D. and Romagnoli, J. A.; "Robust multi-scale principal components analysis with applications to process monitoring," *Journal of Process Control*, 15(8) : 869-882, 2005.

[21] Mitchell, T. M.; *Machine Learning*, McGraw-Hill, Inc., New York, 1997.

<Appendix A> 훈련집합 프로세스 인스턴스

실험에 사용한 데이터는 <그림 4>의 프로세스 모델을 바탕으로 4개의 CFF 룰과 4개의 CSF 룰을 바탕으로 생성되었다. 아래 표는 훈련집합으로 사용된 200개의 프로세스 인스턴스의 일부이다.

Instances	E_1	E_2	E_3	E_4	E_5	E_6	E_7	Result
PI_1	6	8	46	1	76	83	0.61	정 상
PI_2	8	4	45	1	42	97	0.67	정 상
PI_3	6	4	44	2	50	63	0.91	정 상
PI_4	10	2	44	1	23	93	0.90	정 상
PI_5	7	2	49	1	58	92	0.82	정 상
PI_6	6	7	50	1	42	87	0.58	정 상
PI_7	6	10	44	1	60	100	0.56	정 상
PI_8	8	5	42	1	30	96	0.89	정 상
PI_9	6	8	43	1	45	94	0.67	정 상
PI_{10}	10	8	45	2	90	51	0.45	정 상
PI_{11}	8	7	43	1	86	87	0.54	정 상
PI_{12}	2	9	49	2	50	93	0.53	정 상
PI_{13}	1	4	46	1	59	66	0.59	정 상
PI_{14}	3	8	41	1	42	55	0.55	정 상
PI_{15}	2	4	43	1	14	56	0.42	정 상
PI_{16}	5	10	18	2	37	35	0.29	비정상
PI_{17}	9	0	10	3	57	63	0.35	비정상
PI_{18}	1	1	19	1	60	65	0.34	비정상
PI_{19}	4	4	11	3	57	68	0.37	비정상
PI_{20}	1	6	10	3	65	69	0.34	비정상
PI_{21}	2	6	14	3	74	61	0.39	비정상
PI_{22}	5	0	16	2	31	44	0.34	비정상
PI_{23}	2	2	15	2	22	15	0.30	비정상
PI_{24}	4	4	11	3	57	68	0.37	비정상
PI_{25}	1	6	10	3	65	69	0.34	비정상
PI_{26}	2	6	14	3	74	61	0.39	비정상
PI_{27}	5	0	16	2	31	44	0.34	비정상
PI_{28}	2	2	15	2	22	15	0.30	비정상
PI_{29}	7	5	11	2	95	63	0.38	비정상
PI_{30}	5	6	12	3	57	70	0.59	비정상
…	…	…	…	…	…	…	…	…

<Appendix B> 시험집합 프로세스 인스턴스

아래 표는 시험집합으로 사용된 24개의 프로세스 인스턴스이다.

Instances	E_1	E_2	E_3	E_4	E_5	E_6	E_7	Result
PI_1	2	8	43	2	31	65	0.85	정 상
PI_2	4	9	38	1	67	80	0.79	정 상
PI_3	1	6	21	1	44	77	0.69	정 상
PI_4	4	8	44	1	67	63	0.92	정 상
PI_5	6	7	31	1	71	15	0.50	정 상
PI_6	9	9	10	2	42	41	0.85	정 상
PI_7	5	5	36	3	32	50	0.50	정 상
PI_8	8	8	44	3	46	76	0.85	정 상
PI_9	7	6	31	3	70	63	0.68	정 상
PI_{10}	6	3	30	1	46	61	0.85	정 상
PI_{11}	3	6	26	2	70	21	0.68	정 상
PI_{12}	7	5	31	1	68	40	0.59	정 상
PI_{13}	10	1	18	3	27	98	0.33	비정상
PI_{14}	3	6	19	2	61	49	0.32	비정상
PI_{15}	5	3	8	2	52	31	0.24	비정상
PI_{16}	2	2	6	3	27	45	0.57	비정상
PI_{17}	4	1	43	3	49	39	0.63	비정상
PI_{18}	3	7	21	3	61	49	0.48	비정상
PI_{19}	8	8	8	3	66	24	0.75	비정상
PI_{20}	5	2	5	3	57	70	0.68	비정상
PI_{21}	6	3	6	3	95	63	0.73	비정상
PI_{22}	5	3	33	1	13	41	0.73	비정상
PI_{23}	7	6	47	2	55	36	0.42	비정상
PI_{24}	4	2	36	2	42	23	0.62	비정상

<Appendix C> 훈련집합을 이용하여 생성한 의사결정 나무