유비쿼터스 환경에서의 상황인식형 자가치유 시스템

윤현상, 박정민, 이은석

성균관대학교 정보통신 공학부 컴퓨터 공학과
경기도 수원시 장안구 천천동 300, 440-746
Tel: +82-32-290-7220, Fax: +82-32-290-7211, E-mail:{ wizehack, jmpark, eslee}@selab.skku.ac.kr

요 약

유비쿼터스환경으로 발전하기 위하여 네트웍을 구성하는 시스템들의 문제나 장애의 자가 진단 및 치유에 대한 요구가 높아지고 있다.

본 논문에서는 이러한 유비쿼터스환경에서 여러 시스템들이 스스로 자신의 오류나 문제를 관찰, 진단하고 치유할 수 있는 자가치유 시스템을 제안한다. 특히, 제안 시스템은 기존의 자가 치유 시스템에서 연구되어왔던 일률적인 치유가 아닌 각각의 시스템의 상황을 인식하고 그것에 맞추어서 치유전략을 달리하는 상황인식 기능과 치유 결과 분석을 통해 치유전략을 자발적으로 개선해 나가는 기능을 제공한다. 구체적인 주요 기능은 다음과 같다: 치유대상들의 의존성 점검을 통한 근원적인 문제 해결, 자가치유를 위한 효율적인 리소스 사용, 벤더의 웹서버를 검색하여 치유정보를 수집, 벤더 의존성의 경감등, 치유프로세스의 개선은 물론 치유를 위한 관리자의 작업 부담을 상당 부분 경감할 수 있게 된다.

제안 시스템은 시뮬레이션을 통하여 그 유효성을 입증하고 있다.

Keywords: ubiquitous Computing; Context-awarenesse; Self-Healing; Agent;

1. 서 론

오늘날 고도로 분산화된 컴퓨팅 환경에서 각종 컴퓨터 시스템의 효율적인 관리를 위해서 숙련된 전문가의 필요성이 증가하고 있다. 하지만 이러한 전문가를 통해 관리를 수행하는 것은 비용 측면에서 뿐만 아니라 인력 수급면에 있어서도 그 한계가 명확히 들어난다[1]. 심지어는 전체 컴퓨터 시스템의 오류 중 약 40%가 관리자의 오류에 의한 것이라 하니[2], 전문 관리자에 의존하는 현재의 시스템 관리 방식은 개선되어야 한다. 가까운 미래의 유비쿼터스 환경에서는 더욱 많은 컴퓨팅 디바이스와 정형화되지 않는 사용방식으로 인하여 이러한 문제는 심각한 결과로 이어질 것이다. 이 문제를 해결하기 위해서 관리자에 의존하지 않고 시스템 혹은 컴퓨팅 디바이스 스스로가 자신에 발생한 문제를 식별하고, 진단하고, 치유하는 자가 치유 시스템(Self-healing system)에 대한 관심이 높다[3]. 그러나 기존의 치유 시스템(Self-healing system)은 리소스 낭비, 로그의 수와 크기 증가, 관리자와 벤더(Vendor)의존성 등등과 같은 문제가 있다.

이러한 문제들에 대처하기 위해서 본 논문에서는 다음과 같은 기능들을 가진 차세대형 치유 시스템(Self-healing system)을 제안한다. 1) 단일 프로세스가 메모리에 상주하여 리소스의 부하를 최소화, 2) 로그 상황인식을 위해서 필터링 과정을 우선적으로 수행, 3) 벤더(Vendor)의 웹서버(Web-server)를 검색하여 필요한 지식을 능동적으로 찾아서 Vendor의 의존성을 줄였다.

위에서 제안한 기능을 이용함으로 해서 상기의 앞서 언급했던 문제점들을 부분적으로 해소하고 있다. 제안 시스템은 실제로 설계, 구현되어 실험을 통하여 그 유효성을 입증하고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구, 3장에서는 제안 시스템, 4장에는 구현 및 평가, 5장에는 결론으로 구성 되었다.

2. 관련연구

자가 치유 시스템은 시스템 결함, 자원 변화 등과 같은 스스로의 변화에 대응하여 자신의 행동을 수정하는 능력을 가진다. Oreizy et. al.[4]은 자가 치유 시스템에 관하여 " 시스템 모니터링(*Monitoring the system)*" , " 변화에 대한 계획(*Planning the changes)*" , " 변화에 대한 기술(*Deploying the change descriptions)*" , " 변화를 실행(*Enacting the changes)*" 프로세스들을 제시 하였다 [5][6][7][8]

이들 프로세스들은 자가 치유를 위한 자가 적응 환경을 만든다. 따라서 관련연구에서는 기존 자가 치유 시스템들의 프로세스들이 어떤 기능을 하는지 알아보고, 장점과 문제점을 알아보고 그들의 진화 레벨을 측정한다.

2.1 자가 치유 시스템 진화모델

우리는 IBM의 자율 컴퓨팅(Autonomic computing) 진화 모델[11] 을 기반으로 상황 인식형 자가 치유(Context-awarable Self-healing) 진화 모델을 표 1과 같이 재구성 하였다. 유비쿼터스 환경에서 지향하는 자가 치유(Self-healing)의 최종 단계는 관리자의 개입 없이 시스템 스스로 정책에 의한 동적 관리를 가능하게 하고, 지속적으로 지식을 성장 가능 하도록 하는 Autonomic 단계이다. 이 자가 치유 진화 모델 (Self-healing Deployment Model)을 기반으로 기존 자가 치유(Self-healing 시스템)의 장단점을 분석하고, 시스템의 레벨을 측정하여 Autonomic 단계를 지향하는 자가 치유 시스템(Self-healing system)을 제안한다.

2.2 IBM 자가 치유 시스템

IBM에서 제안한 자가 치유 시스템(Self-healing system)은 크게 "관찰(Monitoring)", "로그 변환(Translation)", "분석(Analysis)", "진단 및 치유(Diagnosis)", "피드백(Feedback)"의 5단계의 처리 프로세스를 가진다. 운영체제(OS), 데이타베이스(Database)와 같은 다양한 컴포넌트들에 의해서 생성된 로그파일을 Adapter를 사용하여 실시간으로 모니터링 하고, 공통된 로그 형식(common log format)으로 변환한다. 이것을 CBE(Common Base Event)[1]라고 한다. CBE형태로 변환된 로그는 Autonomic Manager에게 전달되고[9], CBE 로그들 간의 연관성을 분석하여 Symptom Service의 도움을 받아 치유방법을 찾는다. 치유방법을 찾았다면 치유방법들 중에 가장 최선의 방법을 찾기 위해 Policy Engine이 사용되고 그 결과 가장 적절한 치유방법이 Resource Manager에게 전달되어 Self-healing의 기능을 수행한다.

IBM의 Self-healing System은 다양한 경로에서 생성되는 로그를 관리자와 시스템이 쉽게 이해할 수 있도록 Common Log Format으로 로그를 변환하여 시스템 스스로가 문제를 인식하고 해결책을 제시할 수 있도록 하는 장점이 있다. 그러나 Adapter가 자신이 담당한 컴포넌트에서 생성된 로그에 대하여 실시간으로 모니터링 하는 것은 로그가 생성되는 시간적인 간격에 비해 해당 로그에 대한 불필요한 확인이 여러번 이루어지게 된다. 그렇기 때문에 컴포넌트의 수만큼 모니터링 해주는 프로세스가 메모리에 상주하여 리소스의 부하를 주게 된다. 또한 긴급한 처리가 필요한 위급한 상황이 발생하였을 경우, 이것을 적절히 대응할 수 있는 방안 역시 부족하다.

이러한 내용으로 볼때 IBM의 self-healing system은 부분적으로는 level4에 접근하려 하고 있지만, 현재 시스템이 가지고 있는 단점들로 인해 level 2 에서 level 3 정도의 서비스만 온전하게 보장할 수 있다.

2.3 CISCO와 IBM의 Adaptive Services Framework

IBM과 CISCO가 공동으로 연구한 Adaptive Service Framework(ASF)는 크게 "관찰(Monitoring)", "로그 변환(Translation)", "여과(Filtering)", "분석(Analysis)", "진단 및 치유(Diagnosis)", "피드백(Feedback)" 6단계의 처리 프로세스를 가진다. 다양한 컴포넌트에서 나오는 로그를 Adaptor가 모니터링 하여 CBE 형식으로 변환하고, 치유가 필요하다고 판단되는 로그의 상황을 인식하기 위해서 여과 작업을 거쳐 Autonomic Manager에게 전달한다. Autonomic manager는 Symptom Rule에 의해서 적절한 치유 방법을 찾아 해당 컴포넌트에 적용시키고, 컴포넌트에 심각한 문제(critical problem)나 해결하기 힘든 문제가 발생하였을 경우 Call Home Format[2]에 맞게 작성된 message를 SSP(Support Service Provider)/Vender에게 보내어 필요한 해결방법을 찾는다 [10].

ASF는 필터링을 이용하여 로그의 상황을 인식하였고, 기존에 IBM에서 제안했던 Self-healing system의 단점이 였던 Autonomic Manager의 부하를 줄였다. 또한 치유(healing)를 위한 지식이 부족하거나 현재 상황에서 해결할 수 없는 문제의 경우에는 Call Home Format을 이용하여 적절한 대처를 수행할 수 있는 기능을 제공했다.

그러나 변환되기 전의 로그의 사이즈 보다 CBE 형식으로 변환된 로그의 사이즈가 커지고 복잡한 연산으로 인하여 변환 과정 중에 디스크 및 CPU, 메모리 사용량이 증가하는 문제점이 있다. 이것은 유비쿼터스 환경에서 휴대용 디바이스를 사용하게 된다면 리소스사용에 과부하 문제를 야기 시킬 수 있으며 Call Home Format을 이용하는 경우에 있어서 해당 SSP/Vendor에서 지원하지 않으면 해결책을 제시해줄 수 없다는 제약이 있다. 또한 이 방식역시 CBE형식의 로그를 통해서 상황을 분석하기 때문에

---

[1] CBE(Common Base Event): XML로 표현된 log에 대한 공통적인 표현법을 나타낸다.
http://xml.coverpages.org/IBMCommonBaseEventV111.pdf

[2] Call Home Format: IBM&CISCO에서 표준화를 추진하고 있는 Healing System과 SSP/Vendor간의 메시지 전송 규약이다.
http://www.cisco.com/application/pdf/en/us/guest/partners/partners/c644/ccmigration_09186a0080202dc7.pdf

[표 1] 상황 인식형 자가 치유(Context-awarable Self-healing) 진화 모델

Basic Level 1	Managed Level 2	Predictive Level 3	Adaptive Level 4	Autonomic Level 5
Characteristic				
다양한 경로에서 시스템 상태정보 생성	*하나의 경로에서 공통된 타입을 가진 로그의 생성과 모니터링*	*시스템 스스로 로그를 모니터링, 관리자에게 행동을 추천*	*시스템 스스로 지식기반 분석 및 Healing*	*시스템 지식 및 정책과 성능의 동적 관리 및 성장*
Required skill				
로그의 생성경로 관련 지식 *컴포넌트 관련 지식* *관리 노하우*	*공통된 형태의 로그에 대한 지식* *컴포넌트 관련 지식* *관리 노하우*	*관리 대상에 대한 전반적인 이해도* *특정한 경우에서의 관리 노하우*	*관리대상에 대한 전반적인 이해도*	*시스템 정책 및 서비스에 대한 지식*
Administrator's burden				
관리자에대한 기술 의존도가 매우 높음	*Healing이 필요한 정보만 모니터링*	*시스템이 제안하는 Healing 방식을 수랑 또는 선택*	*시스템의 행동을 모니터링* *성능에 대한 임계갑 관리*	*서비스 정책 결정*
Benefit				
Healing을 위한 가장 최소한의 조건 충족	*집중화된 파리가 가능한 생산성 향상*	*관리자의 지식에 대한 의존도가 낮아짐* *빠른 의사결정* *오류 감소*	*관리자 개입 최소화*	*생산적인 작업에 집중*

CBE형식의 로그가 만들어지고 Autonomic Manager에 의해서 분석되기 전에 적절한 조치를 취해야 하는 긴급한 상황의 경우에는 적용될 수 없는 단점이 있다.

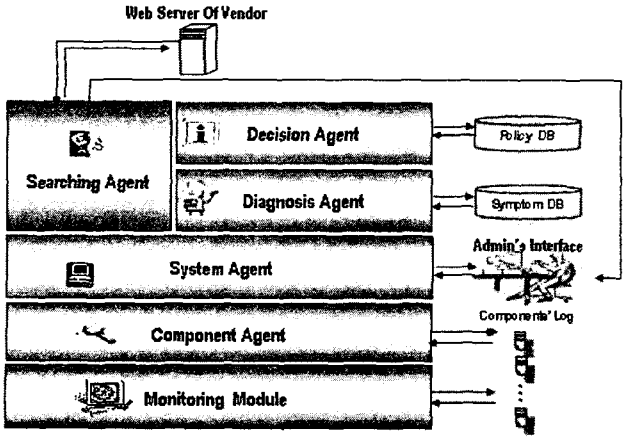
이것을 종합하여 보면 이 시스템은 완전한 Level 4로 보기에는 여전히 무리가 있다.

위에서 언급했던 문제를 해결하고 보다 좋은 성능을 위해 본 논문에서는 다중 에이전트(Multi-Agents)를 이용하였다. 지능을 가진 여러 에이전트(Agent)들이 주변 상황을 인식하고 적절하게 대처함으로써 보다 자율적(Autonomic)인 자가 치유(Self-Healing)을 수행할 수 있다.

3. 제안 시스템

제안 시스템의 구성은 [그림 1]과 같다. 제안 시스템은 다양한 경로의 컴포넌트들로부터 생성되는 로그를 모니터링 하는 Monitoring Module, 생성된 로그를 Filtering하고 CBE형식의 로그로 변환하는 Component Agent, 시스템의 상황을 모니터링하고 치유하는 System Agent, 치료받을 시스템의 상태를 진단하는 Diagnosis Agent, 진단된 결과에 대한 치유방법을 결정하는 Decision Agent, 특정 컴포넌트에 대한 벤더(Vendor)의 웹서버(Web Server)를 검색하여 필요한 정보를 가져오는 Searching Agent, 관리자에게

치유를 위한 정보를 제공하고 스크립트를 작성하여 시스템의 치유를 위한 지식을 학습시키는 Admin Interface로 구성되어 있다.



[그림 1] 제안 시스템 구성도

3.1 Monitoring Module

이 모듈은 각 컴포넌트들의 로그 생성을 실시간으로 모니터링 하기 위해 로그 경로에 대한 정보와 이 정보에 대응하여 로그를 CBE형식으로 변환하는 Component Agent에 대한 경로를 이차원 배열(Array) 내에 문자열(String)값으로 가지고 있다. Monitoring Module은 자신이 가지고 있는 모든 로그의 경로에 순차적으로 접근하여 로그파일의 사이즈가 변경되었는지 확인하고 사이즈 변경이 발생하면 대응되는 Component Agent를 실행시킨다. 다음은 Monitoring Module의 동작에 대한 알고리즘이다.

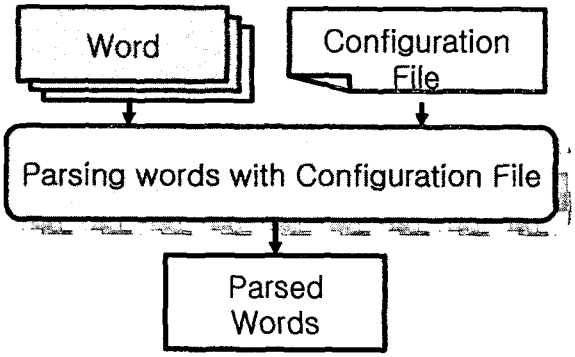
Step1 i번째 로그 경로를 읽는다
Step2 현재 파일 사이즈를
의 파일 사이즈와 비교한다
Step2.1. IF 현재 파일 사이즈와
전 파일 사이즈가 같으면
i를 하나 증가시킨다
ELSE IF 현재 파일 사이즈와
이전의 파일 사이즈가
같지 않으면
i번째 Component Agent
를 실생히킨다.
Step3 Step 1 으로 간다.

3.2 Component Agent

Monitoring Module이 Component Agent을 실행시키면 이 Module은 자가 치유(Self-healing)을 위해 필요한 로그만을 추출하여 CBE형식으로 변환한다. 이 과정을 위해 Agent는 Preprocessing, Parsing, XML Generating. 단계를 거친다. Preprocessing 단계에서는 다음과 같은 행동이 일어난다.

Step1 로그 파일에서 바로 이전에
전처리 과정을 거친 로그의 위치를
찾는다
Step2 다음에 생성된 로그를 읽는다
Step3 "not","unused", "error", "reject".
과 같은 키워드가 있는지 찾는다.
Step3.1 IF (키워드가 존재하면)
그 로그를 하나의
라인으로 분류한다
ELSE IF (키워드가 존재하지
않으면)
Step 4로 간다
Step3.2 하나의 라인으로된 로그를
단어별로 분류한다
Step4 알고리즘 종료

Preprocessing 단계를 거치면 하나의 로그는 작은 단어들로 나뉘게 된다. 이렇게 나눠진 단어들은 Parsing 과정을 통해서 CBE에서 의미하는 속성 값들과 매핑된다. 이 때 로그에 포함되어져 있지 않는 내용이 CBE log를 제작하기 위해 필요한 경우가 있다. 예를 들면 IP address, Host name etc. 이와 같은 값들은 미리 Configuration File에 지정해두고 필요할 때 사용할 수 있게 한다. 다음 [그림 2]는 Parsing 부분을 그림으로 표현한 것이다.

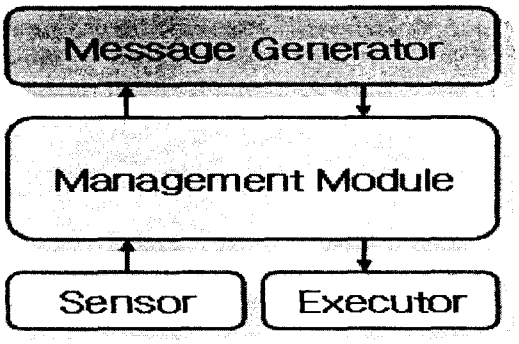


[그림 2] Parsing 과정

Preprocessing과정을 거친 Words는 Configuration File과 함께 IF ~ THEN rule에 의한 Parsing과정을 거치면서 CBE로그에 들어갈 속성값들로 표현된다. 이 속성값들은 XML Generating과정을 거쳐서 XML Schema에 맞는 CBE 로그로 변환된다.

3.3 System Agent

System Agent은 다음 [그림 3]과 같이 여러 모듈로 나뉘어져 있다.



[그림 3] System Agent

Sensor는 CPU 정보와 Memory정보 현재 사용중인 프로세스 정보, Job Schedule정보를 모니터링 하여 Management Module에게 전달한다. System Agent내에 있는 Sensor가 Component Agent로 부터 CBE Log를 받으면 Message Generator는 CBE로그에 이용 가능한 Memory정보, Job schedule정보, 현재 사용중인 프로세스 정보, 그리고 관리자 정보(Administrator information)를 추가하여 Diagnosis Agent에게 보낸다. 관리자 정보는 치유(Healing)에 실패하였을 경우 관리자에게 이를 알리는 경로에 관한 정보이다. Management Module은 Sensor로부터 모니터링된 정보의 임계값에 대한 지식을 가지고 위급한 상황에서 즉각적으로 대처한다. 또한 CBE Log와 위에서 언급한 시스템의 상태 정보들의 캡슐화를 위해 모니터링 된 정보들을 Message Generator에게 전달해 준다.

추가적으로 Management Module은 Decision Agent로부터 받은 치유 방법(Healing method)를 받아서 해석하고 Execution Module을 통해 적절한 조치를 취한다.

3.4 Diagnosis Agent

Healing System내에서 System Agent로부터 받은 로그들의 내용을 기반으로 각 로그간의 연관성을 파악하고, 상태를 진단한다. Diagnosis Agent의 동작 알고리즘은 다음과 같다.

Step1 message에서 CBE로그를 추출한다
Step2 CBE 로그에서 correlator component 부분을 읽는다.
Step3 비슷한 시간대에 발생한 연관된 로그를 찾는다.
Step4 연관된 로그를 클러스터링 한다.
Step5 IF~Then rule 기반으로 로그를 분석한다.
Step6 CBE로그 와 함께 받은 시스템 상태 정보와 진단정보를 Decision Agent에게 전달한다.

3.5 Decision Agent

Healing System내에서 Diagnosis Agent로부터 받은 진단정보를 기반으로 치유방법을 결정한다. 이 부분에서 결정되는 치유 방법은 근본적인 치유, 임시적인 치유, 임시적 치유 이후에 근본적인 치유로 결정할 수 있다. 임시적인 치유방법은 근본적인 치유을 위해 선행되어 지거나, 일시적으로 문제를 해결하도록 하는 방법이 있다. 예를들면 네트워크 설정을 끊거나, 임시 메모리를 추가로 할당하는 방법이 그것이다, 근본적인 해결방법은 진단된 결과에 대한 근본적인 치유방법 이다. 예를 들어 재설치, Restart, Rebooting등의 방법이 있다. Decision Agent는 이 방법들을 DB(Database)에 [그림 4]과 같은 테이블의 형태로 저장하였다가 조건에 맞는 치유방법을 찾는다. 그러나 원하는 조건이 존재하지 않을 경우Decision Tree를 이용하여 가장 적절한 치유 방버(Healing Method)를 결정하고 System, Agent가 이를 실행할 수 있는 코드를 보낸다.

위 [그림 4]는 주어진 여러 가지 속성들을 분석하여 최적의 해결방법을 결정하기 위한 테이블이다. " DECISION" Column을 보면 현재 다양한 상황에 놓여있을 때 치유를 위해 " R(Root Solution)", "T(Temporary Solution)", 또는 " TR(first Temporary Solution, second Root Solution)"를 결정하도록 도와준다. "FEEDBACK" Column은 Decision Agent가 보낸 치유를 위한 코드를 실행한 후에 System Management Module로부터 받은 응답을 나타낸다.

Decision Agent는 System Agent로부터 받은 현재 작업스케줄(CURRENT JOB)과 앞으로 수행될 작업 스케줄(FUTURE JOB), 이용 가능한 메모리(AVAILABLE MEMORY)의 패턴과 Decision Agent가 결정한 임시적 치유방법(TEMPORARY SOLUTION)과 근본적 치유방법(ROOT SOLUTION)을 위와 같은 테이블과 비교한다. 만약 같은 패턴이면서 FEEDBBACK이 Positive인 열(row)을 찾으면DECISION Column에 있는 치유방법을 결정한다.

그러나 같은 패턴의 열(row)이 없을 경우 ID3 알고리즘을 이용한 결정 트리(Decision Tree)를 생성하고 이 트리를 통해 가장 적절한 치유방법을 결정한다. ID3 알고리즘은 주어진 패턴에 대한 일반화된 분류규칙을 생성하여, 기존에 가지고 있지

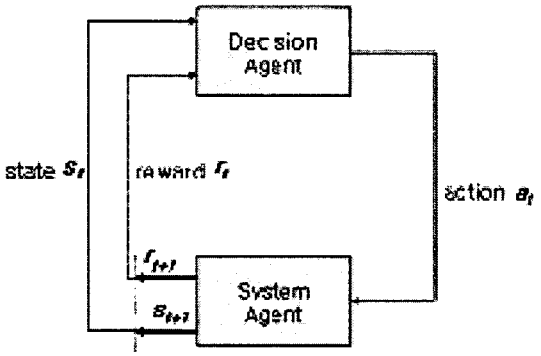
PROBLEM	TEMPRARY SOLUTION	ROOT SOLUTION	CURRENT JOB	FUTURE JOB	AVAILABLE MEMORY	DECISION	FEEDBACK
CONNECT	CHANGE CONFIGURATION	REINSTALL	NULL	NULL	80%	T	POSITIVE
AVAILABLE	NULL	RESTART	NULL	NULL	80%	R	POSITIVE
OVARLOAD	ALLOCATE MEMORY	REBOOT	BACK UP	NULL	30%	TR	POSITIVE

[그림 4] 결정 테이블

않았던 패턴을 분류해 내는데 사용할 수 있는 알고리즘 이다. 이 알고리즘은 다음과 같다.

Step0 FEEDBACK이 Positive인 row들을 table로 재 정렬한다.
Step1 주어진table에서 근(root)노드가 될 만한 특징을 결정 한다.
Step2 근 노드를 부모노드 P로 한다.
Step3 부모 노드의 엔트로피(Entropy)를 다음과 같이 계산한다.
E(P)는 Entropy, c는 분류된 클래스의수, Np는 전체 패턴들의 개수 Nl은 집합 cl 의 개수이다.
Step4 부모노드의 패턴들을 분류하는 특징을 선택한다.
Step4.1 근 노드부터 부모노드에 도달하기까지 사용되지 않은 모든 특징들 에 대하여 각각의 특징을 사용한 경우의 엔트로피 감소량을 구한다.
Step4.2 Step4.1에서 구한 엔트로피의 차이가 가장 큰 값을 갖는 특징을 선택한다.
Step4.3 부모노드에 이 특징에 대한 분류를 적용하여 자식노드들을 첨가한다.
Step5 현재 결정트리의 각 단말 노드를 대상으로 그 노드의 패턴들이 속하는 클래스가 두개 이상인지 검사한다.
Step5.1 만약 모든 단말 노드가 한 개 클래스의 패턴들만 포함하면 알고리즘은 종료된다.
아니면 패턴들의 소속 클래스가 두 개 이상인 모든 단말 노드들에 대하여 각 단말 노드를 새로운 부모 노드 P로 두고 단계 2로 간다.

이 알고리즘에서" 특징" 은 FEEDBACK, 현재 작업스케줄(CURRENT JOB)과 앞으로 수행될 작업 스케줄(FUTURE JOB), 이용 가능한 메모리 (AVAILABLE MEMORY)이다. Decision Agent는 이 특징들의 엔트로피 감소량을 구하고 각각의 값을 비교함으로써 결정 트리(Decision Tree)를 확장하여 최종적인 결정을 내리게 된다.
이 알고리즘을 통하여 최종적인 결정이 나오게 되면. Decision Agent는 치유를 위한 스크립트와 코드를 System Agent에게 배포한다. 이 때 유일한 하나의 치유방법만을 보내게 되면 이 방법이 실패하였을경우 시스템에 심각한 위험이 따르는 경우가 있으므로, 여러 가지 차선책을 우선순위를 정해서 함께 보낸다.
System Agent는 이 스크립트를 실행하여 시스템을 치유하고 결과를 Decision Agent에게 피드백(Feedback)한다. Decision Agent는 System Agent로부터 받은 피드백(Feedback)을 기반으로 강화학습(reinforcement learning)을 하며 자신의 지식을 확장시켜 나간다.
다음 [그림 5]는 본 논문에서 강화학습(reinforce-ement learning) [12]을 적용한 예이다.



[그림 5] 강화 학습

Decision Agent로부터 받은 메시지를 실행시킨 System Agent는 결과에 따라 긍정적(Positive)또는 부정적(Negative) 메시지(message)를 피드백(Feed-back)하게 되고 이 결과들은 Decision Agent의 치유 테이블에 다시 학습되어진다. 이것을 위해서 Decision Agent은 System Agemt에게 치유(Healing)가 되었는지를 검증하기위한 방법을 같이 보낸다.

3.6 Searching Agent

만약 어떤 컴포넌트에 새로운 버그가 발견되었고 이것에 대한 패치를 벤더(Vendor)를 통하여 제공받아야 한다면, 이 버그로 인해 이벤트(Event)가 발생하고 로그(Log)가 생성 되었을때 Diagnosis Agent는 진단을 하지 못할 수도 있다. 이러한 경우 Searching Agent은 "google"과 같은 검색엔진을 이용하여 CBE Log에 명시된 Vendor의 Patch와 관련된 웹 사이트의 주소를 아래그림과 같이 찾는다 위의 [그림 6]과 같이 Searching Agnet는 검색엔진의 url 패턴을 인식하고 있다가 원하는 정보에 맞는 url을 작성하고 검색엔진의 가장 첫 번째 검색

```
http://www.google.co.kr/search?hl=ko&ie=UTF-8&newwindow=1&q=sendmail+patch+site%3Aredhat.com&lr=
http://www.google.co.kr/search?hl=ko&ie=UTF-8&newwindow=1&q=apache+patch+site%3Aapache.org&lr=
http://www.google.co.kr/search?hl=ko&ie=UTF-8&newwindow=1&q=wuftp+patch+site%3Awu-ftpd.org&lr=
```

[그림 6] URL 패턴

사이트 url에 접속하여, 해당 페이지로부터 패치 파일(Patch File)의 링크를 찾는다. 링크의 패치 파일(Patch File)의 확장자가 " *.rpm" 이나 " *.pkg" 같은 정형화된 설치형식이 존재하는 파일이면 Searching Agent는 스크립트를 제작하여 Decision Agent에게 넘겨준다. 그러나 확장자가 " *.src" 와 같이 설치방법이 정형화 되어져 있지 않은 파일은 해당 웹 페이지 정보를 Admin Interface에게 넘겨준다.

만약 Searching Agent가 벤더(Vender)에서 제공하는 패치 파일(Patch File)의 url을 찾지 못한다면 Vender의 게시판(Board)이나 FAQ정보를 찾아서 Admin Interface에게 넘겨준다.

3.7 Administrator Interface

Searching Agent가 보내주는 정보를 볼 수 있는 웹브라우징 인터페이스와 다운로드한 패치파일을 설치할 수 있는 인터페이스를 제공한다. 사용자가 위의 인터페이스를 통해 작업한 내용은 스크립트로 작성되어지고, System Agent는 이 스크립트를 실행하여 결과를 피드백(Feedback)한다.

4. 구현 및 평가

본 논문에서 제안한 시스템은 Linux kernel 2.4 에서 Java SDK기반으로 프로토타입을 구현하였다. Management Server는 Oracle 9i를 이용하였고, 자가치유(Self-Healing)를 테스트(test)하기 위한 로그는 RedHat Linux 2.4 의 /var/log/message에 있는 로그를 이용하였다.

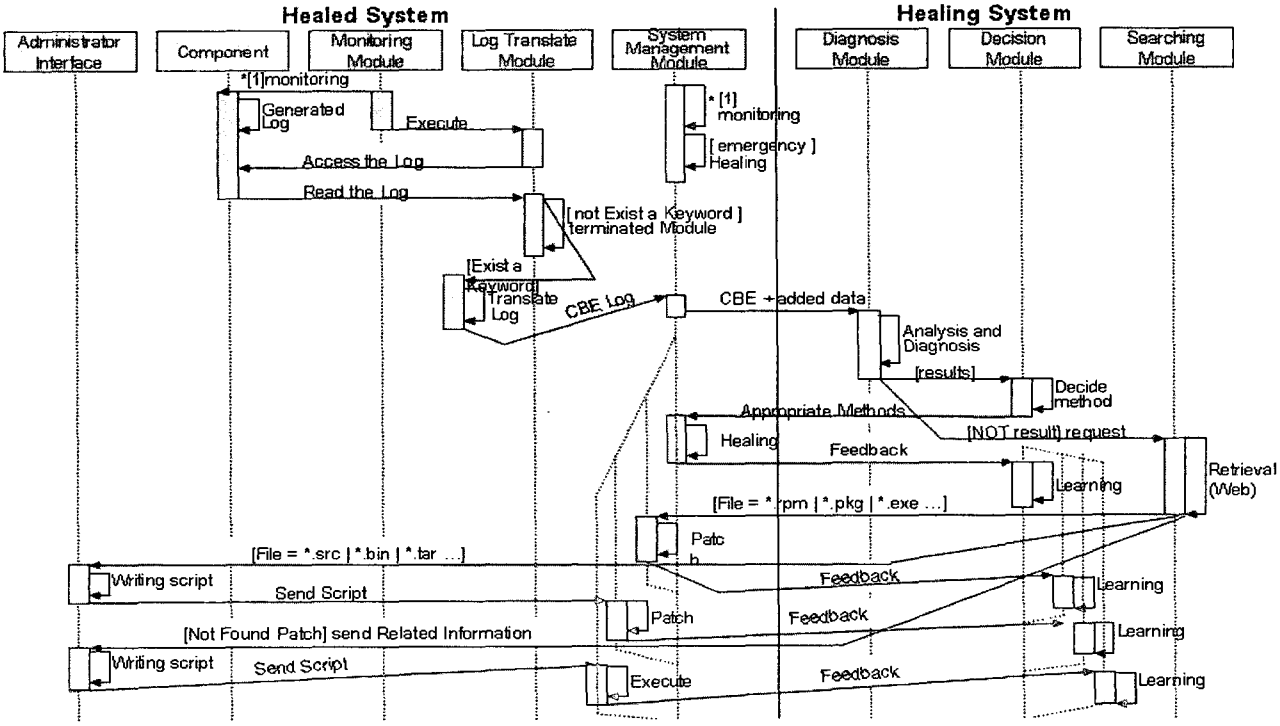
4.1 구현

다음 [그림 7]은 본 논문에서 수행하는 자가 치유 시스템(Self-healing System)의 치유과정에 대한 시퀀스 다이어그램(Sequence Diagram)이다.

Monitoring Module은 시스템 내에서 컴포넌트들의 로그가 발생되는 경로를 실시간으로 모니터링을 하다가 [그림 8] 같은 새로운 로그가 발생하면 Component Agent를 실행시킨다.

Component Agent는 생성된 로그의 키워드(Key word)를 찾아서 Healing을 위해 사용되어질 로그인지 판단한다. 만약 원하는 키워드(Key word)를 찾지 못했을 경우 Agent는 종료된다. [그림 8]에서 나온 로그는 " rejecting"이라는 지정된 키워드를 가지고 있다. 이 경우 Agent는 로그를 [그림 9]과 같은 CBE 형식의 로그로 변환하여 System Agent에게 전송한다.

Searching Agent는 관련된 Url을 자신의 데이타베이스 테이블(Database table)에 미리 저장하고 있다가 해당 벤더(Vendor)의 웹서버(Webserver)에서 필요로 하는 정보를 검색한다. 검색된 정보들은 치료가 필요한 시스템을 관리하는 관리자의 인터페이스 화면(Administrator Interface)으로 보내지고



[그림 7] 메시지 시퀀스 다이어그램

또한 본 논문에서 사용한 DBMS(DataBase

```
Aug 31 13:53:19 selab.skku.ac.kr sendmail[13543]: NAA13543: SYSERR(root): collect: I/O error on connection from [220
```

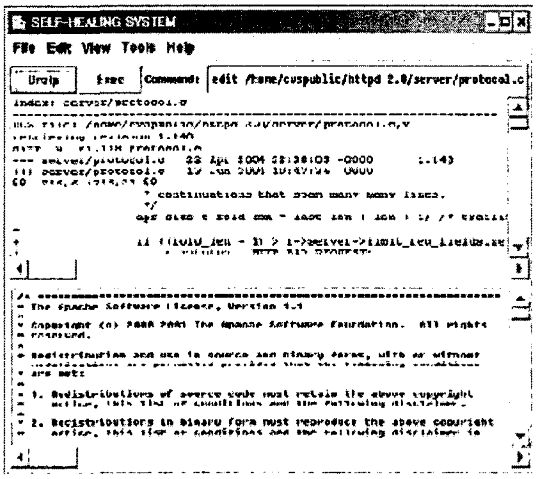
[그림 8]. 생성된 로그


```
<CommonBaseEvent creationTime="replace with our message text" globalInstanceId="NFF58C680F4211D88000FF6BABC2170
        <situation categoryName="ComponentId:&#x9; Application Server;ProcessId:&#x9; 2676;ThreadId:&#x9; 6a61d:
                <situationType xsi:type="AvailableSituation" reasoningScope="replace with our message text" opei
        </situation>
        <situation categoryName="ComponentId:&#x9; Application Server;ProcessId:&#x9; 2676;ThreadId:&#x9; 6a61d:
                <situationType xsi:type="AvailableSituation" reasoningScope="replace with our message text" opei
        </situation>
        <situation categoryName="ComponentId:&#x9; Application Server;ProcessId:&#x9; 2676;ThreadId:&#x9; 6a61d:
                <situationType xsi:type="AvailableSituation" reasoningScope="replace with our message text" opei
        </situation>
</CommonBaseEvent>
```


[그림 9] CBE 형식으로 변환된 로그

System Agent는 평상시에는 시스템의 상태와 성능을 지속적으로 모니터링 하다가 모니터링된 결과가 매우 위급한 상황일 경우 재 부팅과 같은 즉각적인 조치를 취한다. 그러나 Component Agent로부터 CBE로그를 받으면, System Agent는 현재 시스템의 메모리정보와 예약 작업 관리자 정보와 같은 시스템 상태 정보(System State Information)를 CBE 로그와 함께 Diagnosis Agent에게 전송한다.

Diagnosis Agent는 System Agent로부터 받은 메세지(Message)중 CBE부분을 분석하여 Log간의 연관성을 찾고 상태를 진단한다. 진단된 결과와 기존에 System Agent로부터 받은 시스템 상태 정보(System State Information)는 다시 가공되어져서 Decision Agent에게 전달된다. 만약 Diagnosis Agent 가 현재 상태를 진단할 수 없을 경우 Searching Agent에게 관련된 벤더(Vendor)의 웹서버(Webserver)에서 필요로 하는 정보를 검색해줄 것을 요구한다.



[그림 10] 관리자 인터페이스

관리자는 이 정보를 보고 적절한 조치를 취하게 된다. [그림 9]은 관리자 인터페이스이다.

[그림 10]에서 일반적으로 Windows System에 대한 패치(Patch)의 경우는 웹을 통해 패치 화일(Patch File)를 받아서 실행하면 된다. 따라서 툴바의 Exec 버튼만을 눌러주면 된다. Linux/Unix System의 경우는 이진 화일(binary file)을 실행하는 또는 압축을 해제하고 컴파일(Compile) 또는 rpm이나 pkg등의 형태로 설치하면 된다. 이 때에도 위 툴바의 기능을 적절히 활용하면 된다. 만약 추가적인 설정이 필요한 경우라면 메뉴 바를 적절히 활용할 수 있다. 다운받을 패치(Patch)의 내용은 메뉴바 아래에 있는 Web Browsing Board를 통해서 확인할 수 있고, 생성되어지는 스크립트는 [그림 10]의 가장 아래부분인 Text Board를 통해서 확인하거나 직접 수정할 수 있다.

Decision Agent는 Diagnosis Agent로부터 받은 진단정보와 결정을 위한 정보들을 분석하여 현재 상황에 가장 적절한 치유방법을 선택한다. 치유방법이 선택되어지면 Diagnosis Agent는 치유방법들의 우선순위와 실행을 위한 Code및 스크립트, 검증을 위한 스크립트 형태로 작성하여 System Agent에게 보낸다. System Agent는 이 메시지(Message)를 받아서 수행하고, 수행한 결과를 이용하여 Decision Agent를 강화학습(Reinforcement Learning)시킨다.

4.2 시스템 평가

시스템 평가는 로그 모니터링 테스트, 로그 연과및 변환과정에서의 효율성, 관리자 의존성, 긴급한 상황에서의 대처능력의 네가지 관점으로 수행하였다.

로그 모니터링 테스트 기존의 IBM과 Cisco에서 제안한 방식은 컴포넌트에서 생성되는 로그를 각각의 Adaptor가 모니터링 하는 방식을 사용하였다. 이것은 컴포넌트의 수가 Ω 개일 때, 해당 로그를 모니터링하는 프로세스의 수 또한 Ω 개 여야 한다는 문제가 있다. 이러한 문제를 해결하기 위해서 제안시스템의 Monitoring Module기능은 시스템에 존재하는 로그들에 대한 모니터링을 전담하는

하나의 단일 프로세스가 메모리에 상주하는 방식으로 식2)를 적용하여 그림 5와 같이 로그 모니터링 모듈의 효율성을 높였다. 아래는 자가 치유를 위해 메모리에 상주하는 총 프로세스의 수를 기존의 방식과 본 논문에서 제안한 방식으로 나누어서 살펴본 결과이다.

1) 기존 방식

$$P \geq C \text{ and } M=C$$
$$C = \Omega \quad \text{and } M = \Omega$$
$$\therefore H = P+M \qquad (1)$$

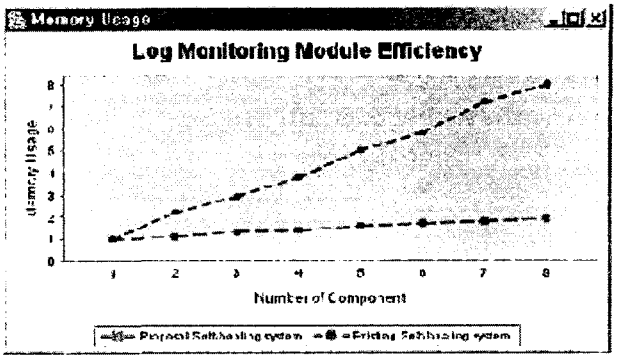
2) 제안 방식

$$P \geq C, \quad M=2$$
$$C = \Omega \quad \text{이고 } M = 2$$
$$\therefore H = P+2 \qquad (2)$$

(P: 메모리에 상주하는 컴포넌트들의 총 프로세스 수)
(C: 컴포넌트들의 수)
(M: 컴포넌트를 모니터링하는 프로세스의 수)
(H: Self-healing을 위해 메모리에 상주한 프로세스의 총 수)

위 식(2) 에서 H = P + 1 이 아니고 H = P + 2 인 이유는 기존의 Monitoring Module의 문제점 완화(fault tolerance)를 위해 예비로 Monitoring Module을 하나 더 추가하였다. [그림 11]은 기존 Self-healing system과 제안시스템의 메모리 사용량을 보여주고 있다.

기존 Self-healing system은 컴포넌트의 수가 증가할 때마다 메모리 사용률이 증가하는 비효율적인 성능을 보였지만, 제안 시스템은 컴포넌트의 수가 증가하여도 메모리의 사용률이 일정한 수준에 있는 효율적인 성능을 보였다.



[그림11] 로그를 모니터링 하는데 사용되는 메모리

로그 여과 및 변환의 효율성 기존의 자가 치유 시스템(Self-healing system)은 컴포넌트에서 생성된 로그를 CBE(Common Based Event)로 변환해서 필터링 하기 때문에 변환과정에서 로그의 수와 크기로 인해 성능이 저하되는 비효율이 존재하고 있었다.

제안된 자가 치유 시스템(Self-healing system)은 컴포넌트에서 생성된 로그 내에서 지정된 " Key word" (예를 들어 Not, Reject, Fail, Error 등등)를 찾고, 해당 " Key word"가 로그 내에 존재하면 치유(healing)가 필요한 로그로 간주하여 먼저 필터링을 과정을 수행하고, CBE로 변환을 한다 이것은 변환과정에서 로그의 수와 크기로 인해 성능이 저하되는 비효율을 개선하는 방안으로 좋은 결과를 가져 왔다.

우리는 500 개의 로그를 대상으로 치유(healing)가 필요한 로그와 그렇지 않은 로그를 분리하여 로그의 수와 크기를 가지고 평가 하였다. 분리한 결과 실제로 치유(healing)를 위해 사용되는 로그는 20% 내외였고, 변환된 로그의 수와 크기가 기존의 방식보다 작아진 것을 알 수 있었다 [표 2].

[표 2]에서 보여진 것처럼 기존시스템과 제안 시스템의 차이는 필터링하는 순서이다. 제안 시스템은 CBE로 변환하기 전에 필터링 작업을

[표 2] 로그 수와 사이즈 비교

Existing Self-healing system					
Created Log	*100*	*200*	*300*	*400*	*500*
Translated Log	*100*	*200*	*300*	*400*	*500*
Filtering Log	*26*	*87*	*90*	*103*	*106*
Translated Log Size	*260,400 byte*	*520,800 byte*	*781,200 byte*	*1,041,600 byte*	*1,302,000 byte*
Proposal Self-healing system					
Created Log	*100*	*200*	*300*	*400*	*500*
Filtering Log	*26*	*87*	*90*	*103*	*106*
Translated Log	*26*	*87*	*90*	*103*	*106*
Translated Log Size	*67,704 byte*	*226,548 byte*	*234,360 byte*	*268,212 byte*	*276,024 byte*

먼저 수행함으로써 CBE로 변환해야 하는 로그의 수와 크기를 감소시켜 시스템의 성능을 높이는 좋은 결과를 가져왔다.

관리자 의존성 제안 시스템은 관리자에 기존의 자율적으로 관리되어지지 않는 시스템에 비해서 관리자 독립적으로 운용되어진다.
우리는 이것을 실험하기 위해 리눅스 시스템에서 운용되어지는 웹 서버와 DBMS간의 연동을 인위적으로 종료시켰다

그 결과 관리자가 이 문제를 인식하고 문제점의 원인을 발견하여 적절한 대처를 수행하는데 대략 2시간 정도 소요되었지만 제안 시스템을 이용하였을 경우 로그를 변환하는 시간과 문제점을 찾는 시간이 거의 실시간으로 이루어졌다.

긴급한 상황에서의 대처능력 제안 시스템은 긴급한 상황에서 기존의 자가치유(Self-Healing) 시스템에 비해 신속하게 대응할 수 있었다. 이것을 실험하기 위해 우리는 fork() 시스템 콜(System Call)을 이용하여 프로세스를 무한이 생성하는 단순한 C 프로그램을 돌려봤다. 그결과 기존의 자가치유 시스템(Self-healing system)이 동작하는 방식으로 상황을 모니터링하고 로그를 생성하여 변환하기전에 이미 시스템은 다운되었지만, 제안 시스템에서 하는 방식으로 위급한 상황에서 대처할 수 있는 정책을 입력한 결과, 본 제안 시스템은 모든 프로세스를 재구동(Restart)하는 방식으로 문제를 간단히 해결하였다.

5. 결론및 향후과제

본 논문에서는 Ubiquitous환경에서 여러 시스템들이 스스로 자신의 오류나 문제를 관찰, 진단하고 치유할 수 있는 자가 치유 시스템(Self-healing system)을 제안하여 기존의 관리자의 지식과 경험에 의한 시스템보다 우수함을 입증하였으며, 기존에 제안되었던 자가 치유 시스템(Self-healing system)보다 더 나은 성능을 보였다.

본 제안시스템의 장점은 다음과 같다. 첫째 즉각적인 조치가 필요한 긴급한 상황에서 시스템이 지능적으로 이를 판단하여 즉각적으로 대처하였다 둘째 Log의 생성을 실시간으로 모니터링하기 위한 모듈을 분리하여 메모리의 성능을 향상시켰다. 셋째 로그를 CBE(Common Base Event)형식으로 변환하기 전에 미리 필터링을 하여 로그 변환을 통한 메모리와 디스크 공간 낭비를 최소화 하였다. 넷째 필요한 지식에 대해 벤더 의 웹서버를 자율적으로 검색함으로써 벤더의 의존성을 줄였다. 다섯째 웹서버로부터 검색된 내용에 대하여 관리자에게 보여주고 이에 대하여 적절한 작업을 수행하고 치유 시스템(Healing System)을 학습시킬 수 있는 인터페이스를 제공함으로써 관리자의 편의성과 치유 시스템(Healing System)의 성장을 제공하였다.

그러나 좀더 다양한 상황에 대한(Context Aware) 신속한 대처가 필요하고, 여전히 벤더와 관리자의 의존성이 존재하고, 문제점을 진단할 때 단순한 규칙 기반이 아닌 정확하면서도 광범위한 추론을 할 수 있는 알고리즘이 필요하다.

참고문헌

[1] http://www.ibm.com/autonomic

[2] IBM: Autonomic Computing: "IBM's Perspective on the State of Information Technology", http://www-1.ibm.com/industries/government/doc/content/resource/ thought/278606109.html.

[3] Jeffrey O. Kephart David M. Chess (2003). IBM Thomas J. Watson Research Center: "The Vision of Autonomic Computing," IEEE Computer Society, January

[4] P. Oreizy et. al. (1999) "An Architecture-Based Approach to Self-Adaptive Software," IEEE Intelligent Systems, Vol. 14, No. 3, May/June 54-62.

[5] Garlan, D. and Schmerl, B. (2002) "Model-based Adaptation for Self-Healing Systems, " Proceedings of the First ACM SIGSOFT Workshop on Self-Healing Systems (WOSS), South Carolina, November 27-32.

[6] G. D. Abowd, R. Allen and D. Garlan(1995) "Formalizing style to understand descriptions of software architecture," ACM Transactions on Software Engineering and Methodology, Vol. 4, No. 4, October 319-364.

[7] D. Batory andS. O'Malley(1992) "The Design and Implementation of Hierarchical Software Systems with Reusable Components," ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 4, October 355-398.

[8] M. Bernardo, P. Ciancarni and L. Donatiello: On the formalization of architectural types with process algebras, Proceedings of the 8th International Symposium on Foundations of Software Engineering, November (2000) 140-148

[9] B. Topol, D. Ogle, D. Pierson, J. Thoensen, J. Sweitzer, M. Chow, M. A. Hoffmann, P. Durham, R. Telford, S. Sheth, T. Studwell (2003) "Automating problem determination: A first step toward self-healing computing system," IBM white paper, October

[10] J. Baekelmans, P. Brittenham, T.Deckers, C.DeLaet, E.Merenda, BA. Miller, D.Ogle, B.Rajaraman, K.Sinclair, J. Sweitzer (2003) "Adaptive Services Framework CISCO white paper," October

[11] A. G. Ganek, T. A. Corbi(2003) "The dawning of the autonomic computing era", IBM SYSTEMS JOURNAL VOL 42, NO 1.