http://dx.doi.org/10.7236/JIIBC.2013.13.4.229
JIIBC 2013-4-30

Active SLA 기반 서비스 수준 협약의 자동화

Automation of Service Level Agreement based on Active SLA

김상락*, 강만모**, 배재학***

Sang-Rak Kim, Man-Mo Kang, Jae-HakJ.Bae

요 약 최근 SOA와 클라우드 컴퓨팅 기반의 IT 서비스에 대한 수요가 증가하면서 서비스 당사자들 간에 SLA (Service Level Agreement, 서비스 수준 협약)에 대한 관심이 증대되고 있다. 통상 SLA는 자연어로 작성된 종이 계약서이다. 상업용 SLA 관리 툴에서 사용하는 SLA는 절차적 언어를 사용하여 계약서 내용을 애플리케이션 안에 암시적으로 구현된다. 이는 SLA 자동화 작업을 어렵게 한다. 또한 계약시스템에 대한 유지관리를 어렵게 하고 새로운 계약 요구사항을 적용하기 위해서는 소스 코드에 대한 광범위한 수정 작업이 뒤따른다. SLA 유지관리 과정에서 발생하는 문제의 근본 원인은 동일한 SLA가 문서형과 실행형으로 이원화 되어 있다는 것이다. 본 논문에서는 이러한 현행 SLA 관리의 문제점을 개선하기 위해서 능동형 SLA(ASLA : Active Service Level Agreement)를 기반으로 하는 능동형 SLM(ASLM : Active Service Level Management) 시스템을 제안한다. 이 시스템에서는 이원화된 SLA 처리 및 관리 과정이 능동형 SLA(ASLA)의 도입으로 일원화될 수 있음을 보였다.

Abstract As demand for IT services increase, which are based on SOA and cloud computing, service level agreements (SLAs) have received more attention in the parties concerned. An SLA is usually a paper contract written in natural language. SLA management tools which are commercially available, implement SLAs implicitly in the application with a procedural language. This makes automation of SLA management difficult. It is also laborious to maintain contract management systems because changes in a contract give rise to extensive modifications in the source code. We see the source of the trouble is the existence of documentary SLAs (paper contracts) and corresponding executable SLAs (contracts coded in the procedural language). In this paper, to resolve the current SLA management problems we propose an active SLM (Active Service Level Management) system, which is based on the active SLA (Active Service Level Agreement). In the proposed system, the separated management and processing of dual SLAs can be unified into a single process with the introduction of active SLAs (ASLAs).

Key Words : ASLA(Active Service Level Agreement), ASLM(Active Service Level Management), Business Rules, Intelligent Service, Knowledge Representation.

Ⅰ. 서 론

최근 SOA(Service Oriented Architecture)와 클라우드 컴퓨팅(Cloud Computing) 기반의 IT 서비스가 증가함에 따라 정보처리 분야에서 SLA[1] 자동화가 주목을 받고 있다. IT 서비스의 품질을 보장하면서 안정된 서비스

*정회원, 비케이앤씨 대표
**정회원, 울산대학교 전기공학부
***정회원, 울산대학교 전기공학부(교신저자)
접수일자 : 2013년 5월 17일, 수정완료 : 2013년 6월 29일
게재확정일자 : 2013년 8월 16일

Received: 17 May, 2013 / Revised: 29 June, 2013
Accepted: 16 August, 2013
***Corresponding Author: jhjbae@ulsan.ac.kr
School of Electrical Engineering, University of Ulsan

를 고객에게 제공하기 위해서는 SLA에 기초한 IT 서비스 관리(ITSM, Information Technology Service Management)가 필요하다. ITSM은 IT 서비스 가용성 및 장애관리, IT 기반자원관리, IT 서비스 변경 및 구성관리 등으로 구성된다.

SLA는 서비스 공급자가 서비스를 제공하는 동안 서비스의 품질(QoS, Quality of Service)을 보장하기 위하여 약속하는 이행 기준을 설명하는 문서이다. 오늘날 이것은 대부분 자연언어로 작성되어 있다. 문서로 작성된 SLA는 IT 서비스 프로비저닝(Provisioning)과 모니터링 등의 작업을 자동화하기가 어려워 많은 비용이 들고 처리 속도 또한 느린 단점이 있다. 이러한 이유로 서비스 공급자가 수천 건의 계약서를 작성하고 실행, 모니터링해야 하는 오늘날의 업무환경에는 적합하지 않다.

최근 문서형 SLA의 문제점을 보완하기 위해서 나온 서비스 수준 관리 툴 들은(IBM Tivoli, HP OpenView, CA Unicenter, BMC Patrol, Remedy SLAs 등) 계약 규칙을 Java 와 C++과 같은 표준 프로그래밍 언어를 사용하여 애플리케이션 로직에 직접 인코딩한다. 이것은 절차적 제어 흐름을 완전하게 구현해야 하고 비즈니스 로직, 데이터 접근과 계산 로직들을 함께 구현해야 한다. 이것들의 문제점은 계약규칙들이 애플리케이션 내에 암시적으로 구현되어 있어 SLA에 대한 유지 및 관리를 어렵게 하고 새로운 요구사항에 대해서는 광범위한 재구현의 노력이 뒤따라야 한다.

본 논문에서는 이러한 문제점들을 해결하기 위하여 능동형 SLA(ASLA, Active SLA)를 소개하고 이를 기반으로 하는 능동형 SLM 관리시스템을 제안한다. ASLA는 자연어를 제한된 규칙언어로 변환하는 도구를 내장하고 있고, 능동문서 기반으로 유지 보수성, 확장성, 자동화 측면에서 SLA를 효율적으로 관리할 수 있으며 비즈니스 프로세스와의 통합이 용이하다. ASLA는 기존 SLA 연구에서 보이는 단점을 보완하면서 SLA 작성 단계부터 실행까지의 SLA 전체 프로세스를 관리하는데 성능이 우수함을 보인다[2][3]. 이 시스템은 유연하고 확장성이 뛰어난 능동문서 기반으로 구성되어 있다. 능동문서는 문서양식과 문서처리 절차를 내장하고 있는 실행가능한 문서이다.

본 논문에서 제안하는 ASLA의 장점은 다음과 같다. ① SLA 계약 단계에서 작성된 자연어 계약 내용을 프롤로그 기반 언어로 변환하여 컴퓨터가 실행 가능하게 한다. ② SLA 계약 내용대로 시스템이 실제 운영되는지를 실시간으로 확인하여 계약 내용과 다를 때 단계적으로 업무 담당자에게 알려주는 이벤트를 발생시킨다. ③ 계약 변경에 따른 서비스 내용을 빠른 시간 내에 변경이 가능하고 변경에 대한 위험이 적다. ④ 서비스 계약 변경에 따른 비용부담이 적다. ⑤ 비즈니스 프로세스와 연동을 통해 서비스 수준 협약 관리 업무 자동화를 실현한다.

II. 관련 연구

1. RBSLA 프로젝트

RBSLA(Rule-based Service Level Agreements)[4-6]는 전자계약, SLA 관리 업무를 효율적으로 지원하기 위한 규칙기반 서비스수준관리 프레임워크이다. 이것은 5개 계층으로 구성되어 있다. 외부시스템(External System), 정적실행(Static Execution), 지식표현(Knowledge Representation), 동적 업무 및 계약로직(Dynamic Business/Contract Logic), 관리 및 통제(Management/Control) 등이다. 지식표현 계층의 ContractLog는 지식표현의 기본이 되는 형식 논리에 대한 이론적인 체계를 갖추고 있는 논리 기반 프레임워크이다. 이것은 RBSLM(Rule-based Service Level Management)시스템의 핵심모듈이다. RBSLA는 외부프로그램과의 통합이 용이하고 SLA 자동화에 필요한 다양한 형식 언어를 모듈로 만들어서 사용하고 있다. 여기에는 SLA 조항을 RBSLA로 자동 변환하는 도구가 없으며, 계약 내용 처리에 중심을 두고 있어 SLA 전체 프로세스를 관리하는데 부족함을 보인다.

2. SLA@SOI 프로젝트

SLA@SOI는 NESSI의 전략 프로젝트 중의 하나이다. 3년(2008년~2011년) 동안 SAP, Intel, XLAB, UDO 등 총 11개 기관이 참여하였다[7]. SLA@SOI는 SLA가 지원되는 서비스 기반 인프라(Service Oriented Infrastructure) 구축을 위한 구조 및 방법제시를 목표로 하고 있다. SOI는 IT 인프라를 서비스로서 설명하기 위한 체계, 즉 SOA가 원활히 구현되는 데 기반이 되는 인프라의 민첩성을 확보하기 위한 개념의 집합이다. SLA@SOI는 상용 프로그램 언어인 자바로 구현되었으며 소스 전체를 오픈소스로 공개하고 있다. 이 기술에는 SLA 조항이 애플리케이션 내에 하드코딩 되어 유지보수성과 자동화 측면에서 단점을 보인다.

3. 능동문서

능동문서는 그 내부에 선언적 지식을 포함하며 문서 제어와 처리에 대한 자동화를 지원하는 문서이다. 이러한 능동문서를 구성하는 요소는 그림 1과 같이 문서의 구조, 표현, 지식베이스, 규칙 그리고 질의이며, 각각 XML로 일관되게 표현되어 문서자체에 함께 포함되는 서식문서이다[8].

그림 1. 능동문서 모델
Fig. 1. Active Document Model

Ⅲ. 능동형 서비스 수준 협약(ASLA)

ASLA는 Prolog 또는 Prova[9]로 표현된 SLA를 저장, 교환, 그리고 추론할 수 있도록 만든 실행 가능한 규칙 마크업 언어이다. ASLA는 다른 시멘틱 웹 언어와 호환이 가능하다. 이것은 표준 RuleML[10] 기능과 능동문서(Active Document) 기능을 확장하여 만들었다. 이것은 컴퓨터가 실행할 수 있는 Prolog, Prova 등의 논리 언어로 변환 작업을 거쳐야 한다.

서비스 계약 내용을 컴퓨터가 자동 실행하기 위한 ASLA로의 변환 작업은 아래에 기술한 단계의 작업이 필요하다.

단계 1. SLA 조항의 자연어 표현

'웹 서버가 멈추면 서비스 공급자는 웹 서버를 즉시 복구해야 한다'는 의미의 SLA 조항을 다음과 같이 표현할 수 있다.

If a webserver stops then a supplier must recover the webserver immediately.

단계 2. SLA 자연어를 형식 언어로 표현

SLA 자연어를 형식 언어로 표현하는 방법은 수동으로 표현하는 방법과 자동으로 표현하는 방법 등 두 가지가 있다.

(1) 사람에 의한 수동 표현

SLA 조항을 형식 언어로 표현하는 절차는 1) SLA 조항을 단문, 중문, 또는 복문으로 표현 2) 조항의 술어논리 표현 3) 조항의 Prolog 표현 등으로 구성된다[11]. 단계 1의 SLA 조항을 단문으로 분해하면 그림 2와 같이 표현할 수 있다.

```
- A webserver stops
- A supplier must recover the webserver mmediately
```

그림 2. SLA 조항 분해
Fig. 2. SLA Clause Decomposition

위의 문장을 복문 형태로 재구성하면 그림 3과 같이 표현할 수 있다.

```
If(webserver stops) then (A supplier must recover
the webserver immediately)
```

그림 3. SLA 조항 복문 표현
Fig. 3. Representation to Complex sentence for SLA

(2) ACE 도구에 의한 자동 표현

문장 분해와 문장의 재구성 및 각 문장을 술어논리로 표현하는 부분을 시스템을 통해 자동화 하였다[12]. 1) APE Webclient 화면의 입력박스에 ACE(Attempto Controlled English)[13] 문법에 맞는 SLA 문장을 입력한다. 2) APE Parser를 통해 입력된 ACE 문장이 그림 4와 같이 DRS(Discourse Representation Structure)로 표현되어 진다. 이러한 DRS 문장은 [14]논문에서 제안하는 방법으로 Prolog 문장으로 표현할 수 있다.

ACE 표현
If a n:webserver X1 stops then there is a supplier X2 and it is necessary that the supplier X2 recovers the n:webserver X1 immediately.
DRS 표현
[A,B] object(A,webserver,countable,na,eq,1)-1/5 predicate(B,stop,A)-1/6 => [C] object(C,supplier,countable,na,eq,1)-1/9 MUST [D] predicate(D,recover,C,A)-1/11 modifier_adv(D,immediately,pos)-1/16
Prolog 표현
webserver(a). stop(b,a). supplier(c). recover(d,c,a).

그림 4. SLA, ACE, DRS, Prolog 표현
Fig. 4. Representation to SLA, ASLA, DRS, and Prolog

단계 3. Prolog 형식으로 표현

그림 3의 문장을 규칙엔진에서 실행 가능한 Prolog 형식으로 바꾸면 그림 5와 같이 변환할 수 있다.

```
recovery(B, webserver) :- unavailable(webserver).
```

그림 5. SLA Prolog 표현
Fig. 5. Representation to Prolog for SLA

단계 4. ASLA로 변환

ASLA는 요소(Element)로 구성되어 있고 각각의 요소는 <태그>...</태그>의 형식이다. Prolog 규칙을 XML 형식으로 변환하기 위해서는 Prolog에서 사용하는 상수(Constants), 변수(Variables), 구조체(Structures)와 같은 Herbrand Term과 사실(Fact), 규칙(Rule)과 같은 Horn Clause를 XML 요소로 변환하기 위한 명명규칙이 필요하다. 이는 아래 표 1과 같다[8].

표 1. Prolog와 XML요소 간의 명명규칙
Table 1. Naming Rules Between Prolog and XML Elements

Herbrand Term	Element	Horn Clause	Element
원자	<atom>	술어	<relator>
숫자	<number>	관계 기호	<relator>
구조체	<variable>	사실	<hn><relationship>
논리변수	<structure>	규칙	<hn><relationship>

ASLA에서 사용하는 DTD(Document Type Definition)는 능동문서의 ERML[8]의 DTD와 동일하다.

```
<!ELEMENT    RuleSet      (hn*)>
<!ELEMENT    hn           (relationship*)>
<!ELEMENT    relationship (relator,relationship*,
                          (variable|atom|number)*)>
<!ELEMENT    relator      (#PCDATA)>
<!ELEMENT    variable     (#PCDATA)>
<!ELEMENT    atom         (#PCDATA)>
<!ELEMENT    number       (#PCDATA)>
```

그림 6. ERML의 DTD
Fig. 6. DTD of ERML

그림 7은 능동문서의 전체 구조를 나타내고 있고, 그림 8은 능동문서 구조로 표현한 ASLA이다. 이것은 추론 시에 지식정보를 제공하는 KB.xml, 문서의 서식구조와 스타일시트를 포함하는 slaform.xml, 문서의 규칙을 나타내는 erml.xml, 사용자 질의와 원형을 나타내는 query.xml 등을 포함한다.

```
<?xml version="1.0" encoding="euc-kr"?>
<!DOCTYPE activeForm [
    <!ENTITY knowledgebase SYSTEM "KB.xml">
    <!ENTITY form SYSTEM "slaform.xml">
    <!ENTITY rule SYSTEM "erml.xml">
    <!ENTITY query SYSTEM "query.xml">]>
<Active_document>
        &knowledgebase;
        &form;
        &rule;
        &query;
</Active_document>
```

그림 7. 능동문서 전체 구조
Fig. 7. Total Architecture of Active Document

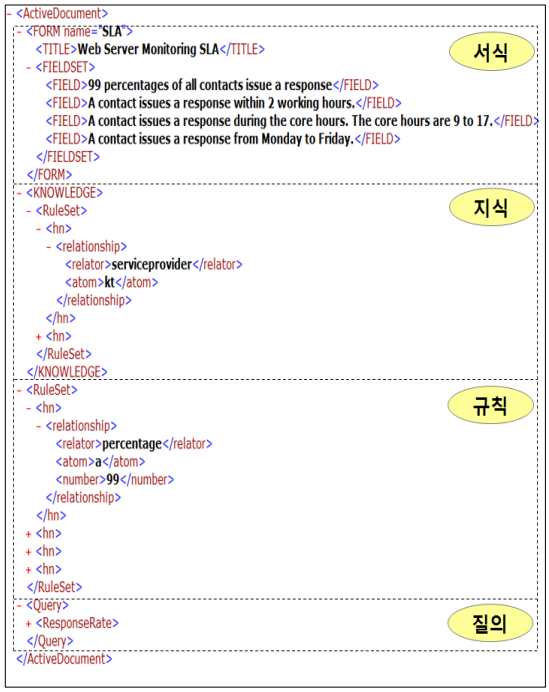


그림 8. 능동형 SLA(activeSLAForm.xml)
Fig. 8. Active SLA

Ⅳ. 능동형 서비스 수준 협약 관리 시스템(ASLM)

1. ASLM 시스템 아키텍처

능동형 SLM(ASLM, Active Service Level Management)는 ASLA를 기반으로 하는 SLA 관리 시스템이다. 이것은 SLA 계약 당사자들이 맺은 서비스 수준 협약을 편리하게 관리할 수 있도록 만든 시스템이다. SLA 조항을 SLA 응용 애플리케이션과 분리시켜 SLA 업무 실행과 그 실행 결과에 대한 피드백을 사람의 개입 없이 컴퓨터에 의해 처리될 수 있도록 자동화하였다.

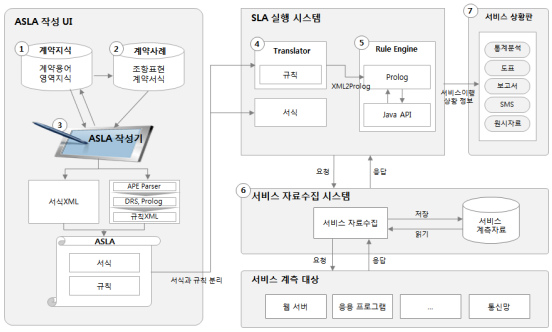


그림 9. ASLM 시스템 아키텍처
Fig. 9. ASLM System Architecture

ASLM의 시스템 아키텍처는 그림 9와 같다. 계약지식①에는 계약용어, 영역지식의 정보가 있고 계약사례②에는 조항표현과 계약서식이 있다. ASLA(Active SLA, 능동형 SLA) 작성기③으로는 SLA 서식과 SLA 규칙을 포함하는 ASLA를 작성한다. Prova 변환기④로는 ASLA를 Prova로 변환한다. Rule Engine⑤는 SLA 계약규칙을 포함하고 있는 Prova 언어를 실행한다. 서비스 자료수집 시스템⑥은 목표 서비스에 대한 이행상황을 기록한다. 이행상황 파악 및 분석, 이행상황에 따른 조치 등은 서비스 상황판⑦에서 이루어진다.

2. ASLM 시스템 기능

ASLM 시스템의 주요 기능은 표 2와 같이 시스템 관리, SLA 기준정보 관리, ASLA 변환 및 실행, 서비스 수준 측정 등 크게 4가지로 분류할 수 있다. 현재 제공되고 있는 서비스는 네트워크 트래픽, 웹 서버 가동시간, 백업, 네트워크 전송 속도 등이다. 서비스는 Gold, Silver, Bronze로 구분하여 서비스 수준을 관리한다.

표 2. ASLM 주요 기능
Table 2. Main Functions of ASLM

구분	기능 설명
1. 시스템 관리	1.1 사용자 등록(온라인)
	1.2 프로그램 등록(온라인)
	1.3 사용자 메뉴 등록(온라인)
2. SLA 기준정보관리	2.1 서비스 수준 등록(온라인)
	2.2 서비스 가격 등록(온라인)
	2.3 고객 등록(온라인)
	2.4 계약 등록(온라인)
	2.5 서비스 수준 조회(온라인)
3. ASLA 변환 및 실행	3.1 ASLA 작성(배치)
	3.2 ASAL를 Prova로 변환(배치)
	3.3 ASLA 실행(배치)
	3.4 SLA 데이터 수집(배치)
4. 서비스 수준 측정	4.1 서비스 수준 측정 그래프1(온라인)
	4.2 서비스 수준 측정 그래프2(온라인)
	4.3 서비스 수준 측정 그래프3(온라인)

3. ASLA 실행

자연어로 작성된 SLA 조항을 ASLA로 변환한다. 이것을 Rule Engine에서 실행되도록 Prova로 변환한다. 그림 10은 ASLA를 Prova 언어로 표현한 예이고, 그림 11은 Java 프로그램에서 Prova 언어로 작성된 프로그램을 호출하는 소스코드이다.

```
recovery(B, webserver) :- unavailable(webserver).
```

그림 10. ASLA Prova 표현
Fig. 10. Representation to Prova for ASLA

```
public void pingWebServer() {
  String input = ":-eval(consult('C:\\workspace\\asla
\\prova\\sla.prova')). :- recovery(B, SERVICE).";

try {
List resultSets = comm.consultSync(input,
Integer.toString(key++), new Object[]{});
～ 중략 ～
}
```

그림 11. Java 프로그램에서 Prova 코드 실행
Fig. 11. Executing Prova code in Java program

V. 실험 및 평가

1. ASLA 실험 : 웹 서버 모니터링 및 장애처리

ASLA를 실행하는 엔진을 담당자가 가동한다. ASLA 엔진이 가동되면서 서비스 수준 이행 상황이 실시간으로 감시된다. 그림 12는 웹 서버 모니터링 및 장애복구 SLA에 대한 자연어와 ASLA 표현이다. 계약조항에 명시된 내용에 따라 웹 서버 상태를 확인하고, 웹 서버가 정지된 경우 데이터베이스에 기록을 남기고, 업무 담당자는 계약조항에 명시된 고장수리 시간 내(MTTR, Mean Time to Repair)에 자신의 역할에 따라 조치를 취한다. 실험용 ASLA의 계약조항은 그림 13과 같이 웹 서버 상태를 시간대별로 확인하고 장애 발생시 BPM(Business Process Management)엔진을 실행하여 업무 담당자에게 장애를 통지하는 프로세스 생성 과정에 대한 내용을 담고 있다. 웹 서버가 가동을 멈춘 경우 '장애처리 프로세스'를 생성한다. 이 프로세스는 프로세스 관리자의 업무 목록에 나타나게 되며 프로세스 관리자는 웹 서버 복구 작업을 수행하고, 그 결과를 품질관리 부서장에게 보고한다. 결과 보고 내용은 품질관리 부서장의 업무 목록에서 조회된다. 품질관리 부서장은 보고 내용을 확인하고 작업 프로세스를 종료한다.

웹 서버 모니터링 일정을 '중요', '표준', '유지보수' 등으로 구분한다. 상세 구분 내용은 표 3과 같다. 웹 서버 모니터링에 대한 업무 단계 및 담당자별로 역할과 책임을 정의한다. 프로세스 관리자는 서비스 가동에 문제가 발생했을 경우 10분 이내로 서비스를 복구해야 한다. 품질 관리자는 문제 발생 원인을 파악하고 서비스 복구와 관련한 작업을 수행한다.

표 3. 웹 서버 모니터링 일정
Table 3. Web server monitoring schedule

구분	시간	가용성(%)			모니터링 주기
		상	중	하	
중요	09:00 - 18:00	100	99	98	10초
표준	09:00 - 18:00	99	97	95	1분
유지 보수	09:00 - 18:00	80	50	20	10분

그림 12의 SLA를 실행하면 ASLM 시스템은 표 3의 일정에 따라 서버 상태를 확인한다. 만약 서버가 동작하지 않으면 그림 13의 (a)와 같이 장애처리 프로세스를 생성한다. 이후 생성된 프로세스는 처리 절차에 따라 진행된다. (b)는 프로세스 관리자의 업무목록 화면이다. 목록에 6건의 웹 서버 장애 데이터가 보인다.

이 가운데에서 ASLA장애처리1과 ASLA장애처리6 등 2건의 장애를 조치하고 품질부서 부서장에게 업무를 이관하였다. (c)는 품질부서 부서장이 처리해야 할 프로세스를 조회하는 화면으로 프로세스 관리자가 이관한 업무를 조회한다. (d)는 완료된 프로세스 현황을 조회한다.

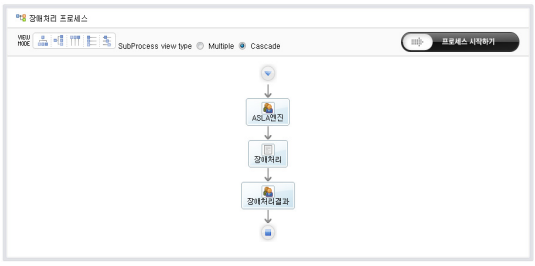
웹 서버 가동상태는 오전 9시부터 오후 6까지 10초 간격으로 점검한다. 만약 서비스가 유지보수 상태가 아니면서 가동되지 않으면 자동적으로 웹 서버 장애 프로세스가 생성되어 프로세스 관리자에게 통지된다. 프로세스 관리자는 정지된 서버를 재실행할 수 있다. 만약 프로세스 관리자가 지정된 시간 내에 서비스 복구에 실패한다면, 보고서를 작성하여 품질관리 부서장에게 보고한다. 웹 서버 가동상태에 관한 보고서를 매월 작성한다.

```
service(http://www.ulsan.ac.kr).
schedule(prime, Service):-
        sysTime(datetime(Y,M,D,H,Min,S)),
        lessequ(datetime(Y,M,D,H,Min,S),
        datetime(Y,M,D,18,0,0)),
        moreequ(datetime(Y,M,D,H,Min,S),
        datetime(Y,M,D,8,0,0)),
        interval(timespan(0,0,0,10),
        datetime(Y,M,D,H,Min,S)), service(Service),
        sysTime(T),
        aslm.utils.WebserverMonitoring.insertPingInfo(T),
        eca(schedule(T,S),not(available(S)),
        not(maintenance(S)),escalate(S),_,restart(S)).
available(S) :- println(["ping ",S]),
    aslm.utils.WebserverMonitoring.pingWebserver(S),

aslm.utils.WebserverMonitoring.updatePingInfo(S,"service"),
    println([S," alive"]).
maintenance(S) :-

aslm.utils.WebserverMonitoring.updatePingInfo(S,"service"),
    println([S," not alive"]),
    aslm.utils.WebserverMonitoring.updatePingInfo(
        "not alive","status"),
    sysTime(T), holdsAt(maintenance(S),T).
~ 중략 ~
```

그림 12. 웹 서버 모니터링 및 장애처리 SLA에 대한 Prova 표현
Fig. 12. Representation to Prova for SLA of web server monitoring and failover process



(a) 장애복구 프로세스 (Failover Process)

(b) 프로세스 관리자 업무 목록
(To-Do List of Process Manager)

(c) 품질관리 부서장 업무 목록
(To-Do List of Quality Manager)

(d) 장애복구 완료 업무 목록 (Completed Task List for Failover)

그림 13. 웹 서버 장애복구 BPM 화면
Fig. 13. BPM Forms for Web Server Failover

웹 서버 상태 점검 시 SLA 계약조항에 따라 로그를 데이터베이스에 기록한다. 그림 14는 웹 서버 실시간 모니터링 로그정보를 보여주고 있다. 로그 데이터베이스에는 웹 서버 점검 이벤트 발생 시점과 웹 서버의 주소, 웹 서버 상태, 담당자에게 통보한 메시지 등이 정보로 관리된다.

2. ASLA의 Prova 표현 실험에 대한 평가

ASLA의 Prova 표현 실험에서는 기존 연구[2-4]의 범위를 확장하여 실제 업무에 적용하기 위해서 비즈니스 프로세스 로직을 추가하였다. ASLA를 기반으로 하는 ASLM 시스템의 각종 기능인 1) 계약실행 및 모니터링, 2) 업무 프로세스 진행 모니터링, 3) 업무처리 등의 실험을 통해 ASLA가 서비스 수준 협약 자동화에 중요한 기술임을 확인할 수 있었다.

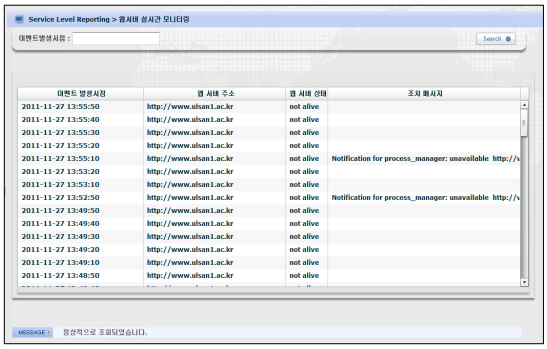


그림 14. 웹 서버 실시간 모니터링 화면
Fig. 14. Form for Web Server Real-time Monitoring

Ⅵ. 결론 및 향후연구

상업적인 SLA 관리 툴들은 애플리케이션 내에 SLA 조항을 하드코딩하고 있어 유지보수와 SLA 프로비저닝 업무의 자동화를 어렵게 한다. 반면 본 논문에서 제안하고 있는 ASLA를 기반으로 하는 SLA 관리 시스템(ASLM)에서는 애플리케이션 로직과 계약 규칙을 분리하여 이러한 기존 상업적인 툴들의 단점을 보완하였다. ASLA기반 서비스 수준 협약을 수행하는 ASLM 시스템은 지능적으로 SLA를 관리할 수 있는 차세대 SLM 웹 응용 프로그램으로 활용이 가능하다. 기업간 계약문서 교환 시스템은 기업의 업무 프로세스나 업무문서 등이 이질적이기 때문에 교환이나 공유가 쉽지 않다. 이에 ASLA를 계약문서 교환의 표준으로 채택한다면 기업내외의 계약관리, 계약문서관리 등의 업무를 효과적으로 수행할 수 있을 것이다. 또한 기업들이 IT 업무를 외부에 위탁할 때 실시간으로 서비스 품질 수준 및 계약 이행사항을 감시하는데 ASLA와 ASLM 시스템을 활용할 수 있을 것으로 기대한다. 미국에서는 이미 범국가적으로

SLA에 기반한 클라우드 컴퓨팅 환경 도입을 추진하고 있다. 우리나라에서도 고유의 독자적 SLA 기술개발이 필수적이다.

본 논문에서 제안하는 ASLA 핵심 기술은 미래 컴퓨팅 환경에서 국가적인 소프트웨어 경쟁력으로 작용할 것이다. 또한 이들은 IT 서비스 관리 자동화 구성요소로서 SOA 및 클라우드 컴퓨팅 발전에 기여하는 도구가 될 것이다.

향후 연구로는 초기계약과 서비스 수준에 따른 과금 체계 자동화, 소프트웨어와 하드웨어 중요 측정지표들을 자동으로 측정하는 소프트웨어 에이전트 개발 등이 있다.

References

[1] SLA INFORMATION ZONE, http://www.sla-zone.co.uk.

[2] S.-R. Kim, J.-G. Yang, J.-H. J. Bae, and G.-S. Jang, “A Comparison of SLA Implementations: ASLM and SLA@SOI”, Proc. of 2011 KIPS Spring Conference, Korea Information Processing Society, Vol. 18, No. 1, May 2011, pp. 329-332.

[3] G.-I. Lee, S.-R. Kim, J.-H. J. Bae, and G.-S. Jang “UI Implementation of an Active Document Based ASLM System”, Proc. of 2011 Korea Computer Congress, Vol. 38, No. 1(B), June 2011, pp. 242-244.

[4] A. Paschke., "RBSLA: A declarative Rule-based Service Level Agreement Language based on RuleML", International Conference on Intelligent Agents, Web Technology and Internet Commerce, Austria, 2005.

[5] A. Paschke and M. Bichler, “SLA Representation, Management and Enforcement”, Proc. of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005.

[6] A. Paschke, J. Dietrich, and K. Kuhla, “A Logic Based SLA Management Framework”, Proc. of Semantic Web and Policy Workshop (SWPW), 4th Semantic Web Conference (ISWC 2005), Galway, Ireland, 2005.

[7] W. Theilmann, R. Yahyapour, and J. Butler, “Multilevel. SLA Management for Service-Oriented Infrastructures,” Proc. of the 1st European Conference on Towards a Service-Based Internet, Dec. 2008.

[8] C.-K. Nam, G.-S. Jang and J.-H. J. Bae, "An XML-based Active Document for Intelligent Web Applications," Expert System with Applications, Vol. 25, No. 2, pp. 165-176, August 2003.

[9] Prova Rule Language, http://www.prova.ws.

[10] RuleML, http://ruleml.org.

[11] S.-R. Kim, G.-I. Lee, J.-H. J. Bae, and G.-S. Jang, “Representation of Active SLA's”, Proc. of 2011 Korea Computer Congress, Vol. 38, No. 1(B), June 2011, pp. 245-248.

[12] S.-R. Kim and J.-H. J. Bae, “Automated Representation of Active SLA's”, Proc. of 2011 KIPS Fall Conference, Korea Information Processing Society, Vol. 18, No. 2, November 2011, pp. 279-282.

[13] ACE, http://attempto.ifi.uzh.ch/site.

[14] M. Covington, D. Nute, N. Schmitz, and D. Goodman, “From English to Prolog via Discourse Representation Theory," ACMC Research Report 01-0024, Advanced Computational Methods Center, University of Georgia, 1988.

※ 이 논문은 2010년 울산대학교 연구비에 의하여 연구되었음

저자 소개

김 상 락(정회원)

- 1992년 : 울산대학교 중어중문학과 학사
- 2010년 : 울산대학교 메카/IT 석사
- 2012년 : 울산대학교 정보통신공학 박사
- 2000년 ~ 2009년 : 아이티스타 연구소장
- 2010년 ~ 현재 : 울산대학교 경영정보학과 외래강사, 비케이앤씨 대표

<주관심분야 : SLA, 빅데이터>

배 재 학(정회원)

- 1981년 : 중앙대학교 전자계산학과 학사
- 1983년 : KAIST 전산학과 석사
- 2003년 : POSTECH 컴퓨터공학과 박사
- 1985년 ~ 현재 : 울산대학교 전기공학부 교수

<주관심분야 : 자동문서요약, {자동, 논리}프로그래밍, 지식경영 및 기술, 전략경영정보시스템>

강 만 모(정회원)

- 1998년 : 울산대학교 전자계산학과 학사
- 2000년 : 울산대학교 전자계산학과 석사
- 2011년 : 울산대학교 정보통신공학 박사
- 2006년 ~ 2009년 : 울산대학교 객원교수
- 2009년 ~ 현재 : 대광산업 기술연구소 책임연구원, 울산대학교 전기공학부 외래강사

<주관심분야 : 멀티에이전트, 소프트웨어공학, 빅데이터 분석>