제19권 제2호
ISSN 2005-0011

Fall Conference 2012

제38회 한국정보처리학회

추계학술발표대회

논문집(하)

일 자 : 2012년 11월 22일(목)~23일(금)

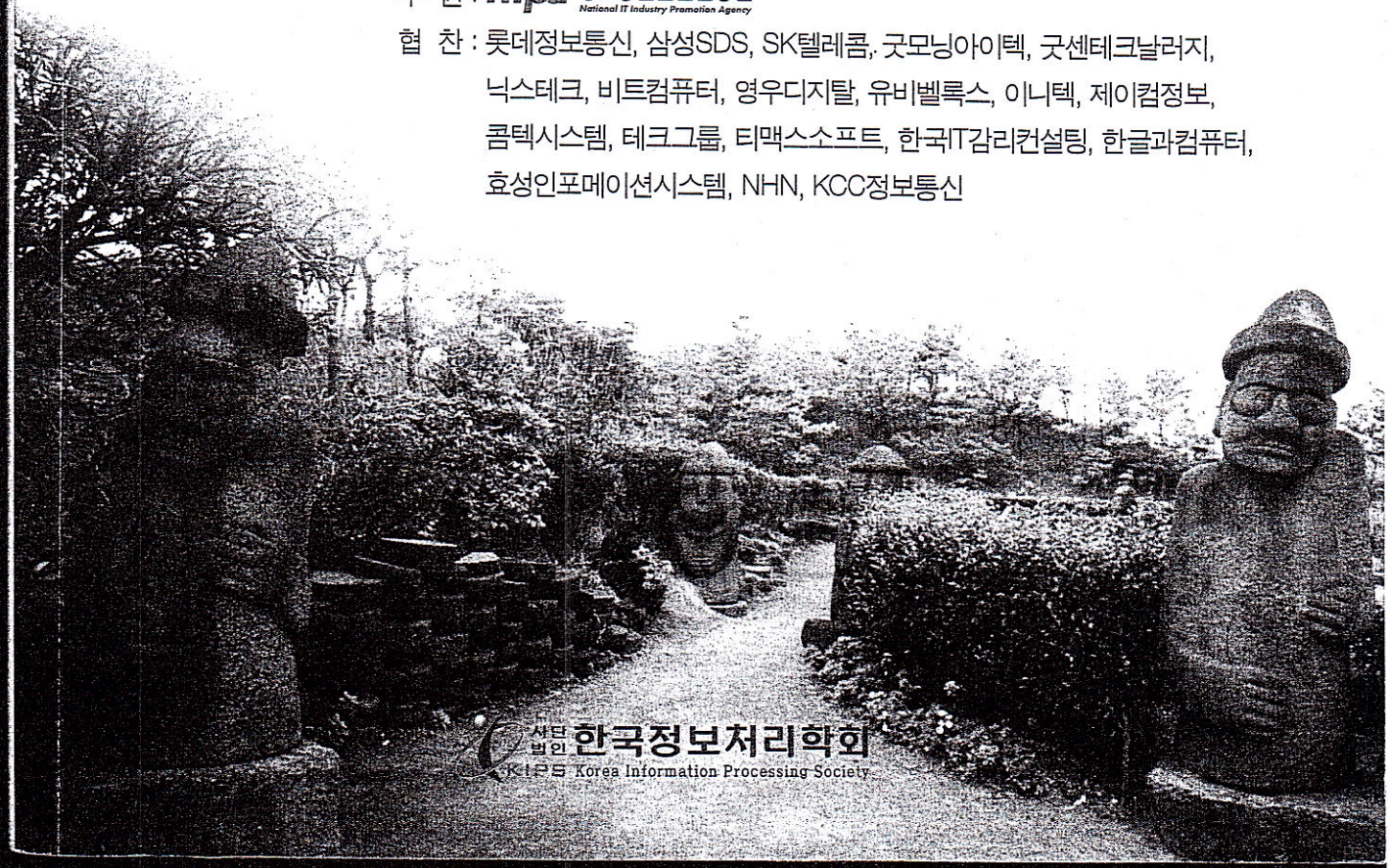
장 소 : **제주대학교** 아라캠퍼스

주 최 : 사단법인 **한국정보처리학회**

주 관 : 제주대학교 JEJU NATIONAL UNIVERSITY

후 원 : nipa 정보통신산업진흥원 National IT Industry Promotion Agency

협 찬 : 롯데정보통신, 삼성SDS, SK텔레콤,. 굿모닝아이텍, 굿센테크날러지, 닉스테크, 비트컴퓨터, 영우디지탈, 유비벨록스, 이니텍, 제이컴정보, 콤텍시스템, 테크그룹, 티맥스소프트, 한국IT감리컨설팅, 한글과컴퓨터, 효성인포메이션시스템, NHN, KCC정보통신



사단법인 한국정보처리학회
KIPS Korea Information Processing Society

정보처리응용(IT 교육 등)

UML2.4.1 기반 메시지-순차적 다이어그램을 통한 테스트 케이스 추출 연구

우수정*, 김동호, 손현승, 김영철
*홍익대학교 일반대학원 소프트웨어공학연구실
e-mail : {woo, bob}@selab.hongik.ac.kr

Text Case Extraction with Message Sequence Diagram (MSD) based on UML2.4.1

SuJeong Woo*, D. H. Kim, S. H. Son, Robert Young Chul Kim
*Dept. of Computer Information & Comm., Hongik University

요 약

기존 연구에서는 순차적, 상태, 엑티브 다이어그램 기반의 테스트케이스 추출을 초점을 두고 있다. 하지만 현재 최신의 모델링 언어인 UML2.4.1(Unified Modeling Language) 기반으로 한 테스트케이스 추출 메커니즘은 없다. 그래서 본 논문은 UML2.4.1 기반에 기존의 원인-결과 다이어그램의 접목을 통해 테스트케이스 추출 메커니즘을 제안 한다. 이를 위해 UML2.4.1 의 메시지-순차적 다이어그램에 ECA Rule(Event Condition Action)기법을 적용하고, 제안한 접목 알고리즘을 통해 확장된 메시지-순차적 다이어그램을 원인-결과 다이어그램과 접목한 후, 결정 테이블화로 테스트케이스를 발생한다. 이러한 절차를 통해 모델링 기반에서 테스트케이스 추출 가이드가 제공된다. 본 논문에서는 복잡한 메시지-순차적 다이어그램을 통해 테스트케이스 발생 사례연구로서 자동차 와이퍼 시스템을 적용한다.

1. 서론

최근 임베디드 소프트웨어는 우리의 실생활에 있어 스마트 홈, 스마트 TV, 스마트폰, 셋톱 박스 등과 같은 여러 분야에 걸쳐 적용되고 있다. 그 중에서도 차량용 임베디드 소프트웨어 분야에 적용한 사례가 증가 하고있다. 그 적용한 사례는 현재 IBM 에서 GM 의 신형 전기차 '쉐보레 볼트'개발에 직접 참여하여 1000 만 라인의 소스코드가 들어가는 임베디드 소프트웨어를 개발 하고 있다[6].

이와 같이 실시간, 병렬적, 확률적인 자동차 임베디드 소프트웨어는 점점 복잡하고, 그에 따라서 정확하게 작동해야 한다[9]. 그러기 위해서는 결함에 대한 오류를 찾아 수정하는 테스트가 필수적이다. 현재 우리 연구실은 모델 기반 테스팅에 초점을 두고 있다[7,8]. 본 논문은 소프트웨어공학에서 많이 사용하고 있는 Unified Modeling Language 2.4.1(UML2.4.1)을 사용하여 테스트케이스 발생하고자 한다. UML2.4.1 의 여러 언어 중 본 논문은 메시지-순차적 다이어그램을 통하여 테스트케이스를 추출한다. UML2.4.1 의 메시지-순차적 다이어그램은 여러 객체 사이에 메시지 보내는 순서를 명확하게 했으며 프레임 엘리먼트를 적용시켜 다이어그램의 레이블을 위한 지정된 장소를 제공한다[2]. 하지만 메시지-순차적 다이어그램을 통해 테스트케이스 추출은 불가능 하기 때문에 메시지-순차적 다이어그램을 ECA Rule(Event Condition Action)의 기법을 적용하여 확장한다. ECA Rule 기법은 메시지-순차적 다이어그램의 각 객체에 Pre-Condition, Action, Post-Condition 로 세분화 하는 기법이다[10]. 확장된 메시지-순차적 다이어그램을 테스트케이스 발생이 가능한 원인-결과 다이어그램에 접목 시킨다[4]. 원인-결과 다이어그램 접목을 통해 발생시키는 것은 최소의 테스트케이스로 100%의 기능적인 요구사항 커버리지를 만족 시킬 수 있는 검증된 방법[3]이다.

본 논문의 구성은 다음과 같다. 2 장은 테스트케이스 추출 과정에 3 장은 사례연구, 마지막 4 장은 결론 및 향후 연구에 대해서 기술한다.

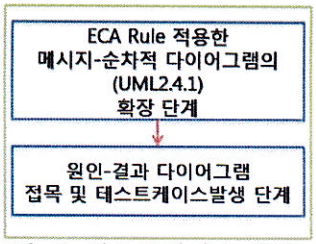
2. 테스트케이스 추출 과정

본 장에서는 테스트케이스 추출 과정에 대해 설명한다.

그림 1 은 테스트케이스 추출 과정이다. 첫 번째 단계는 메시지 순차적 다이어그램을 ECA Rule 을 적용하여 확장한다. 두 번째 단계는 확장된 메시지 순차적 다이어그램을 원인-결과 다이어그램을 접목하여

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업(NIPA-2012-(H0301-12-3004))과 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임.

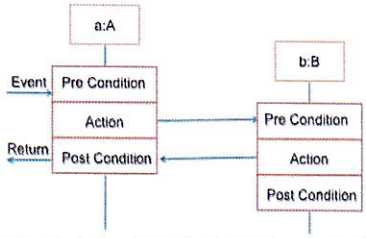
테스트케이스 발생시킨다.



(그림 1) 테스트케이스 추출 과정

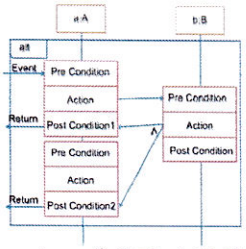
다음은 각 순서에 대한 설명이다.

1 단계 : ECA Rule 을 적용한 메시지-순차적 다이어그램 확장.

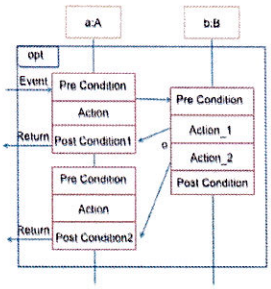


(그림 2) 1:1 메시지-순차적 다이어그램

첫 번째 절차는 메시지-순차적 다이어그램에 ECA Rule 을 적용하여 확장한다. 예를 들어 그림 2 는 객체 간 1:1 메시지-순차적 다이어그램을 나타냈다. 각 객체에는 ECA Rule 을 적용하여 Pre Condition, Action, Post Condition 으로 나누어져 있다. Event 가 A 객체에 Pre Condition 을 통해 들어오고 A 객체의 Action 은 B 객체의 Pre Condition 을 통해 들어간다. 이때 B 객체의 Action 은 A 객체의 Post Condition 으로 들어가 값을 반환하게 된다. 그림 3 은 Alternative 의 확장된 메시지-순차적 다이어그램이다. if/else 를 나타내기 위해 A 객체 안에 Post Condition1, 2 로 구분한다. 또한 B 객체에서 나가는 Action 메시지 부분은 "^"로 한다.



(그림 3) Alternative 메시지-순차적 다이어그램

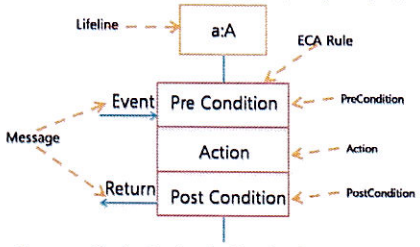


(그림 4) Option 메시지-순차적 다이어그램

그림 4 는 Option 의 확장된 메시지-순차적 다이어그램이다. if 만 나타내는 그림 3 과 비슷한 메시지-순차적 다이어그램이다. A 객체 안에 Post Condition1, 2 로 구분하고, B 객체에서 나가는 Action 메시지 부분은 "o"으로 한다.

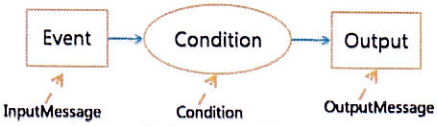
2 단계 : 확장된 메시지-순차적 다이어그램과 원인-결과 다이어그램과의 접목 및 테스트케이스 발생

두 번째 절차는 UML 을 통해서 테스트케이스를 발생시키는 것이 불가능 하기 때문에 테스트케이스를 발생 가능한 원인-결과 다이어그램을 접목 시키고자 한다. 이는 Gary E. Mogyorodi[4]가 제시한 원인-결과 다이어그램을 통하여 발생시키는 것은 최소의 테스트케이스로 100%의 기능적인 요구사항 커버리지를 만족 시킬 수 있는 검증된 방법이기 때문이다.



(그림 7) 메시지-순차적 다이어그램의 요소

그림 7 과 8 은 메시지-순차적 다이어그램과 원인-결과 다이어그램의 요소를 나타낸 것이다. 요소들을 통해서 나타낸 알고리즘은 다음과 같다.



(그림 8) 원인-결과 다이어그램 요소

표 1 은 메시지-순차적 다이어그램과 원인-결과 다이어그램의 접목 알고리즘이다. 메시지-순차적 다이어그램의 Event 메시지를 읽는다. 만약 메시지-순차적 다이어그램의 Return 메시지가 아무것도 없을 경우 종료된다. 아니면 다음과 같이 수행된다. 이때 메시지-순차적 다이어그램의 Return 메시지가 끝나지 않을 때까지 수행된다. 만약 메시지-순차적 다이어그램의 ECA Rule 의 Action 부분이 Alternative 이면 원인-결과 다이어그램의 InputMessage 는 "Evnet"라 하고, Condition 은 "alt"이라 표시한다. 그리고 메시지-순차적 다이어그램의 Lifeline 의 Action 을 수행한 뒤 원인-결과 다이어그램의 OutMessage 은 메시지-순차적 다이어그램의 Return 메시지로 표시하는데 alt 의 경우는 두가지의 메시지를 나타내게 된다.

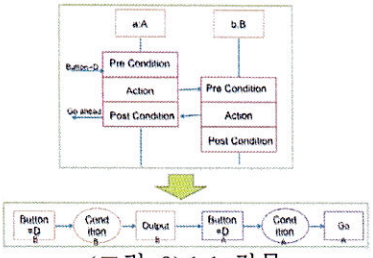
<표 1> 접목 알고리즘

```
//MSD : Message Sequence Diagram
//CE : Cause-Effect Diagram

Read MSD.Message.Event;
   if(MSD.Message.Return == Null){
         exit(0);
   }
   else{
      while(MSD.Message.Return != Null){

       if(MSD.Lifeline.ECARule.Message.Action == alt)
            {CE.InputMessage =”Event”;
             CE.Condition=”alt”;
             MSD.Lifeline.Action();
            CE.OutputMessage1<-MSD.Message.Return1;
            CE.OutputMessage2<-MSD.Message.Return2;
            }

elseif(MSD.Lifeline.ECA Rule.Message.Action == opt)
      {CE.InputMessage =”Event”;
       CE.Condition=”opt”;
       MSD.Lifeline.Action();
       CE.OutputMessage1<- MSD.Message.Return1;
      }
   }
}
```

만약 메시지-순차적 다이어그램의 ECA Rule 의 Action 부분이 opt 이면 원인-결과 다이어그램의 InputMessage 는 “Event”라 하고, Condition 은 “opt”이라 표시한다. 그리고 메시지-순차적 다이어그램의 Lifeline 의 Action 을 수행한 뒤 원인-결과 다이어그램의 OutMessage 은 메시지-순차적 다이어그램의 Return 메시지로 표시한다.

다음은 여러 가지 접목에 대한 테스트케이스 발생에 대한 것이다.

그림 9 는 메시지-순차적 다이어그램과 원인-결과 다이어그램의 1:1 접목을 나타낸 것이다. 사용자가 드라이브(D) 버튼을 눌렀을 때의 이벤트가 객체 B 에 메시지를 보낸다. B 의 액션이 ‘앞으로 간다’의 명령을 내려서 수행하게 된다. 이를 원인-결과 다이어그램으로 하면 D 버튼이 Input 이 되고 그에 맞는 Output 은 다른 객체의 Input 이 되어 최종적인 Output 은 ‘앞으로 간다’로 접목이 된다.



(그림 9) 1:1 접목

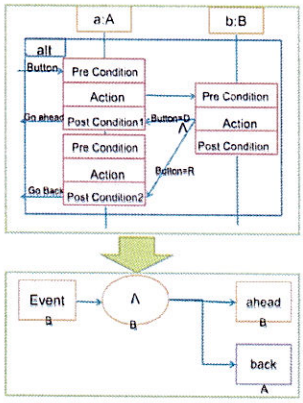
표 2 은 1:1 접목의 테스트케이스다. 총 2 가지의 테스트케이스가 발생된다.

<표 2> 1:1 접목의 테스트케이스

NO	Post condition	Test Condition	Expectation Result
TC1	I1-Drive 버튼=F	N/A	O1-앞으로 간다=F
TC2	I1-Drive 버튼=T	N/A	O1-앞으로 간다=T

그림 10 은 Alternative 접목에 대해 나타낸 것이다. 사용자는 드라이브(D) 버튼 또는 리턴(R) 버튼을 눌렀을 때 ‘앞으로’ 가거나 ‘뒤로’ 가게 되는 것이다. 이 때 원인-결과 다이어그램의 Condition 부분은 ‘^’을 표시하여 alt 를 나타낸다.

표 3 는 Alternative 접목에 대한 테스트케이스이다. 총 8 가지의 테스트케이스가 발생된다.

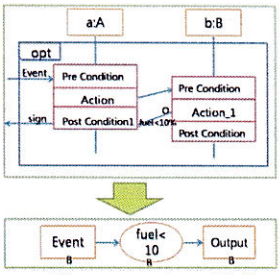


(그림 10) Alternative 접목

<표 3> Alternative 접목의 테스트케이스

NO	Post condition	Test Condition	Expectation Result
TC1	I1-Drive 버튼=F, I2-Reverse버튼=F	alt	O1-앞으로 간다=F
TC2	I1-Drive 버튼=T, I2-Reverse버튼=F	alt	O1-앞으로 간다=T
TC3	I1-Drive 버튼=F, I2-Reverse버튼=T	alt	O1-앞으로 간다=T
TC4	I1-Drive 버튼=T, I2-Reverse버튼=T	alt	O1-앞으로 간다=T
TC5	I1-Drive 버튼=F, I2-Reverse버튼=F	alt	O2-뒤로간다=F
TC6	I1-Drive 버튼=T, I2-Reverse버튼=F	alt	O2-뒤로간다=T
TC7	I1-Drive 버튼=F, I2-Reverse버튼=T	alt	O2-뒤로간다=T
TC8	I1-Drive 버튼=T, I2-Reverse버튼=T	alt	O2-뒤로간다=T

그림 12 는 Option 접목에 대해 나타낸 것이다. 연료가 10 미만이 있을 경우 표시하여 사용자가 볼 수 있도록 한다. 이때 원인-결과 다이어그램은 두 가지 조건에 맞게 Output 을 나타낸다. opt 접목이기 때문에 ‘o’표시를 해둔다.



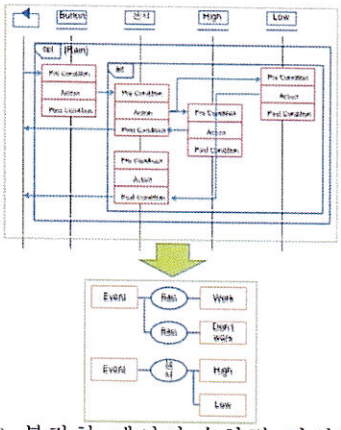
(그림 12) Option 접목

<표 4> Option 접목의 테스트케이스

NO	Post condition	Test Condition	Expectation Result
TC1	I1-Fule<10 =F	opt	O1-Sign=F
TC2	I1-Fule<10 =T	opt	O1-Sign=T

3. 사례연구

복잡한 메시지-순차적 다이어그램의 테스트케이스 발생 사례연구로서 그림 14 와 같이 자동차 와이퍼에 대한 테스트케이스 추출에 대해 언급한다. 비가 올 경우 사용자가 와이퍼 버튼을 눌렀을 때 센서가 작동된다. 센서는 비가 많이 오는지 적게 오는지 판별한 후, 와이퍼를 빨리 움직이거나, 천천히 움직이는 경우를 나타낸 것이다. 이때의 테스트케이스는 표 5 와 같이 발생 한다. 총 12 가지의 테스트케이스가 발생되며 Option 에 관하여 4 가지 Alternative 에 관하여 8 가지이다.



(그림 14) 복잡한 메시지-순차적 다이어그램과 원인-결과 다이어그램 접목

<표 5> 복잡한 접목의 테스트케이스

NO	Post condition	Test Condition	Expectation Result
TC1	I1-Rain=T	opt	O1-Work=T
TC2	I1-Rain=T	opt	O1-Work=F
TC3	I2-Rain=F	opt	O2-Don't Work=F
TC4	I2-Rain=F	opt	O2-Don't Work=T
TC5	I3-센서(강)=T	alt	O3-High=T
TC6	I3-센서(강)=F	alt	O3-High=F
TC7	I4-센서(약)=T	alt	O4-Low=T
TC8	I4-센서(약)=F	alt	O4-Low=F
TC9	I3-센서(강)=T	alt	O3-Low=T
TC10	I3-센서(강)=F	alt	O3-Low=F
TC11	I4-센서(약)=T	alt	O4-High=F
TC12	I4-센서(약)=F	alt	O4-High=T

4. 결론 및 향후 연구

UML2.4.1 버전의 메시지-순차적 다이어그램은 여러 객체 사이에 메시지 보내는 순서를 명확하게 했으며 프레임 엘리먼트를 적용시켜 다이어그램의 레이블을 위한 지정된 장소를 제공한다[2]. 총 11 가지의 컨바인드 프래그먼트를 통해 간결하고 여러 가지 객체 사이에 메시지 보내는 순서가 명확하게 가능했다[2].

본 논문의 메시지-순차적 다이어그램 2.4.1 을 통해서 테스트케이스 발생시키기 위해 ECA Rule 기법을 적용하여 확장하였다. ECA Rule 기법의 Pre-Condition, Action, Post-Condition 을 메시지-순차적 다이어그램 2.4.1 에 적용시켜 확장시킨다. 확장된 메시지-순차적 다이어그램은 테스트케이스가 발생 가능한 원인-결과 다이어그램과 접목시켰다. 접목한 원인-결과 다이어그램을 통해서 테스트케이스 추출하였다.

향후 연구는 확장된 메시지-순차적 다이어그램과 원인-결과 다이어그램을 메타모델정의를 한다. 메타모델을 통해 알고리즘을 구현할 것이다.

참고문헌

[1] "OMG Unified Modeling Language Specification", Version1.4, 2001

[2] "OMG Unified Modeling Language (OMG UML), Superstructure", Version 2.4.1, 2010

[3] Gary E. Mogyorodi, "Requirements Based Testing Cause-Effect Graphing", 2010

[4] 우수정, 손현승, 김영철, "원인-결과 다이어그램 접목을 위한 메시지-순차적 다이어그램 확장 연구", 한국정보처리학회 춘계학술발표대회, 2012

[5] 우수정, 손현승, 김우열, 김제승, 김영철, "Pre-Testing 을 위한 M&S 기반 테스트케이스 추출연구", 소프트웨어공학학회, 2012

[6] 권문주, "자동차 소프트웨어 정부정책방향", 정보처리학회지 Vol.19, No.03, 2012

[7] 김동호, 손현승, 김우열, 김영철, "Model Driven Architecture 기반의 임베디드 테스트 프로세스에 관한 연구" 한국정보과학회 가을학술 발표 논문집 Vol.38, No.2, 2011

[8] 김규원, 김우열, 손현승, 김영철, "금융 부가가치 망 환경의 실시간 트랜잭션 검증을 위한 테스트 케이스 연구", 소프트웨어공학회 Vol.13, No.1, 2011

[9] 문소영, "임베디드 소프트웨어 시스템을 위한 모델링과 시뮬레이션에 관한 연구 : Stochastic 기반 상태 다이어그램", 홍익대학교, 2007

[10] 김규원," 금융 부가가치 망 환경의 실시간 트랜잭션 검증을 위한 테스트 케이스 연구", 홍익대학교, 2011