http://dx.doi.org/10.7236/JIIBC.2013.13.1.129
JIIBC 2013-1-19

M2M 모델변환 기반의 UML 스테이트 다이어그램을 통한 테스트케이스 자동추출 메커니즘에 관한 연구

A Study on Automatic Test Case Extraction Mechanism from UML State Diagrams Based on M2M Transformation

김동호*, 김영철**

Dong-Ho Kim, R. Youngchul Kim

요 약 기존 연구실 연구에서는 메타모델 기반의 순차적 다이어그램을 통한 테스트케이스 생성의 자동화가 초점이었다[1][2]. 하지만 메타모델 기반의 스테이트 다이어그램을 통한 테스트케이스 자동 생성에 관한 메커니즘에 관한 연구가 부족하다. 그래서 본 논문에서는 Model Driven Architecture(MDA) 메커니즘인 메타모델과 모델변형기법을 채택하여, UML내 스테이트 다이어그램을 통해 테스트케이스를 자동 추출메커니즘을 제안한다. 이를 위해 테스트케이스 생성단계 프로세스를 정의하고 각 단계별 메타모델 정의와 모델간의 변형 규칙을 정의한다. 제안한 기법을 통해 임베디드 소프트웨어 테스트 설계 및 테스트케이스 추출비용과 시간을 줄이고 임베디드 소프트웨어의 품질을 높이고자 한다.

Abstract Previous research is focus on testcase generation automation using message sequence diagram based on metamodel. but that research is not enough for testcase genaeration automation using state diagram based metamodel. so in this paper is adopt Model Driven Architecture (MDA) mechanism for using metamodel and model transformation. and we suggest testcase automation mechanism using state diagram in UML. we will decrease cost of embedded software design and testcase genaration and increase quality of embedded software using metamodel mechanism

Key Word : MDA, UML, Metamodel, Model transformation, Test automation, Testcase genaration

Ⅰ. 서 론

최근 소프트웨어 개발에 있어 모델 기반 개발은 많은 이슈가 되고 있다. 모델의 재사용과 모델을 변환하여 얻을 수 있는 이점이 많기 때문이다. 현재 임베디드 산업은 매우 빠르게 성장하고 사용자들의 요구도 점점 다양해지고 더 많아지는 현실이다. 그래서 이런 산업 전반에 걸친 요구와 사용자들의 요구를 만족하기 위해선 모델기반개발이 필요하다[3],[4]. 그러나 모델기반개발에 관한 연구에 비해 모델기반 테스트에 관한 연구는 부족하다. 그리고 모델기반의 자동화 도구에 있어 중요한 메타 모델 및 모델변환 연구 또한 부족한 상황이다. 그래서 모델기반 개발뿐만 아니라 모델기반 테스트 및 이를 위한 자동화 도구를 위한 메타 모델 및 모델변환에 관한 연구가 필요하

*정회원, 홍익대학교 일반대학원 소프트웨어공학연구실
**종신회원, 홍익대학교 컴퓨터정보통신공학과
접수일자 2013년 1월 7일, 수정완료 2013년 2월 7일
게재확정일자 2013년 2월 8일

Received: 7 January 2013 / Revised: 7 February 2013 /
Accepted: 8 February 2013
**Corresponding Author: bob@selab.hongik.ac.kr
Dept. of Computer Information & Comm., Hongik University

다. 그래서 이 문제를 해결하고자 임베디드 환경 상에서의 메타모델과 모델변환을 사용하여 테스트케이스를 자동생성하고 이를 테스트에 적용하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 테스트 프로세스 기반의 테스트케이스 추출방법 및 메타모델링, 3장에서는 제안한 모델변환기법을 통한 테스트케이스 추출 및 모델변환기법, 4장에서는 제안한 모델 변환 기반의 테스트 케이스 추출방법을 통한 자동차의 전장부품의 테스트케이스 추출, 5장에서는 결론을 맺는다.

II. 관련연구

1. 임베디드 소프트웨어 테스트

임베디드 소프트웨어 테스트는 각 시스템에 맞게 개발되고 테스트 돼야 한다. 이로 인해 이종의 시스템의 개발과 소프트웨어의 재사용이 어렵다[3][4]. 그리고 아직까지 각 모델들에 대한 명세적인 모델에 관한 연구가 부족하며 이를 자동화 하는 부분 또한 필요하다[5][6].

2. 메타 모델링

테스트 케이스를 자동화하기 위한 기법으로 메타 모델링 기법을 도입하였다. 메타 모델링을 사용하는 이유는 기존 도구에서 각 플랫폼 별로 필요로 하는 다이어그램으로 변환하기 위해서다. 메타 모델링을 위한 기법으로는 다음과 같다. MOF (Meta Object Facility)는 메타모델 정의언어를 기반으로 UML을 메타모델 작성도구로 이용해 특정 도메인의 메타모델을 정의하고, 이를 바탕으로 실제 운영될 어플리케이션 모델을 플랫폼 독립적으로 정의한 것이다[7].

EMF (The Eclipse Modeling Framework)는 MOF의 OMG 표준 스펙에 대한 구현을 목적으로 하였으며, MOF기반 모델링 및 코드생성을 지원하며 궁극적으로 MDA를 지향하고 있다. EMF 기반의 모델은 Ecore Model 이라 부르며 Ecore를 모델링 하는 방법은 XSD나 Annotated(@model) java Interface와 같은 언어로 작성해서 변환하는 방법과 직접 XMI(Xml Metadata Interchange)를 이용해 모델링 하는 방법이 있다. 이를 통해 Ecore로 모델링된 내용을 기반으로 EMF모델인 GEN Model을 생성한다. 생성된 GEN Model을 이용해 코드를 생성한다[8].

III. 메타모델 기반 테스트케이스 추출

메타모델 기반 모델변환은 MOF를 비롯해서 다양한 메타 모델링 기법들이 존재한다. 그렇지만 거의 대부분이 웹서비스나 큰 시스템을 기반으로 연구가 되고 있다. 그래서 임베디드 환경에 맞는 메타 모델이 필요하고 모델변형을 위한 규칙이 필요하다.

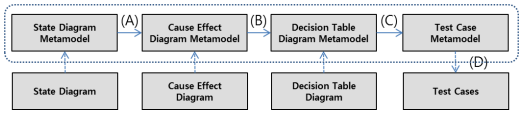

그림 1. 테스트케이스 추출을 위한 모델변환
Fig. 1. Model Transformation for Testcase Generation

기존 모델 기반 테스트 는. 유스케이스 다이어그램, 메시지 순차 다이어그램, 상태 다이어그램을 통한 기법 등이 있다. 테스트 케이스 생성 방법은 각 다이어그램들을 상태 테이블로 변환하고 테스트케이스를 생성한다. 원인-결과 다이어그램을 사용하는 이유는 최소한의 테스트 케이스로 100%의 기능적 요구사항을 만족하기 위해서다. [7][8][9].

1. 각 다이어그램 별 메타모델

OMG 표준의 스테이트 다이어그램은 상당히 방대한 정의와 구성을 이루고 있다. 그래서 필요한 부분만을 채택하여 스테이트 다이어그램을 축약했다. 스테이트 다이어그램 메타모델은 유한 스테이트 트랜지션 시스템의 행위적인 모델을 기술한 것의 집합을 콘셉트로 하고 State Machine, Time events, Protocol State Machines, Constraint로 구성된다[10]. 그림 3은 원인-결과 다이어그램과 메타모델과의 매핑이다. 원인결과 모델(CauseEffectModel)은 속성(Element)과 연결자(Connector)로 구성되어진다. 속성(Element)에는 원인(Cuase)과 결과(Effect)가 있으며 EString값은 조건부를 나타내기 위해 가지고 있다[8]. 그림 4는 결정테이블 메타모델을 정의한 그림이다. 결정테이블모델(Decision Table Model)은 결정테이블의 전체를 나타내고 있는 것이다[8]. 그림 5는 테스트케이스 메타모델을 정의한 것이다. 테스트케이스(TestCase)은 전체 모델을 나타낸다. 이 테스트케이스(TestCase)은 테스트케이스모델(TestCaseModel)을 포함하고 있다[8].

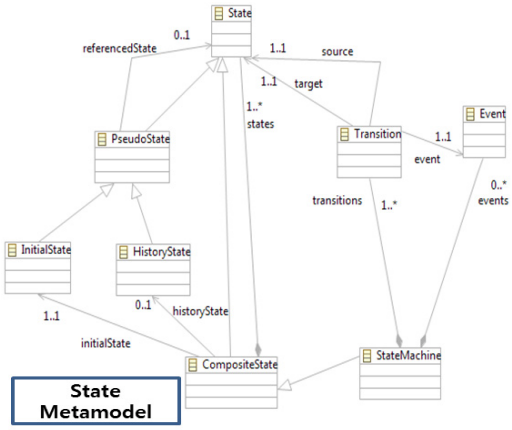

그림 2. 스테이트 다이어그램에 대한 메타모델
Fig. 2. Metamode for IState Diagram

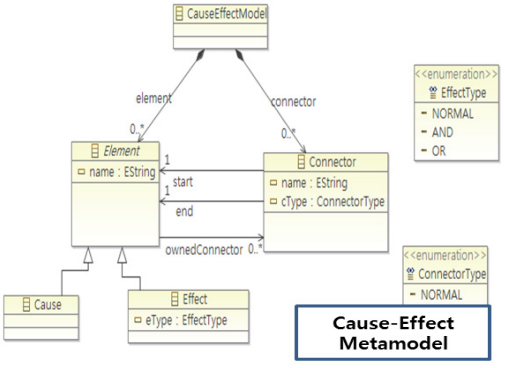

그림 3. 원인-결과 다이어그램에 대한 메타모델
Fig. 3. Metamodel for Cause-Effect Diagram

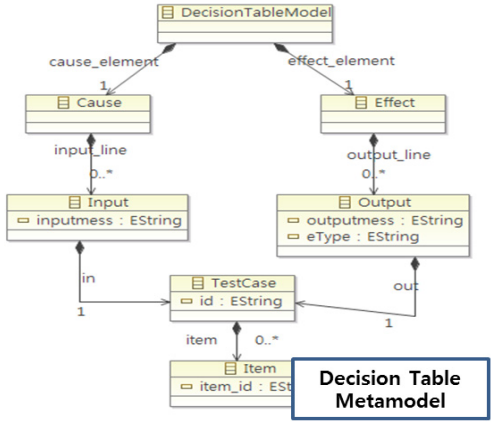

그림 4. 결정테이블에 대한 메타모델
Fig. 4. Metamodel for Decision Table

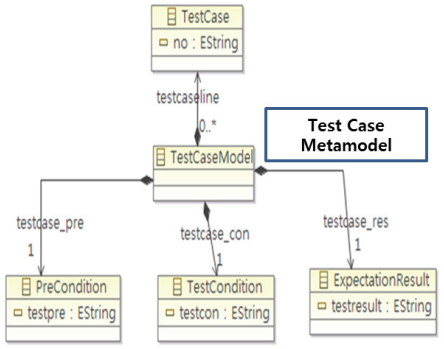

그림 5. 테스트케이스에 대한 메타모델
Fig. 5. Metamodel for Testcase

2. 모델 변형 규칙 정의

각 모델에 대한 변형 규칙을 모델변환언어(ATL : Atlas Transformation Language)로 정의한다. ATL은 모델 기반 공학(MDE : Model Driven Engineering)의 분야에서 ATL 소스 모델의 집합에서 대상 모델 세트를 생성할 수 있는 방법을 제공한다. 이클립스 플랫폼의 상단에서 개발하며, ATL 통합 환경(IDE)은 ATL 변환의 개발을 용이하게 하는 것을 목적으로 표준개발 도구(구문 강조, 디버깅 등)을 제공한다(Eclipse ATL)[11]. 다음 표 1은 스테이트 다이어그램에서 원인-결과 다이어그램을 생성하기 위한 ATL rule의 일부이다. 스테이트 다이어그램을 입력으로 받아 원인-결과 다이어그램을 생성하고 각각의 원인 및 결과 부분을 정의한다.

표 1. 스테이트 메타모델에서 원인-결과 메타모델을 생성하기 위한 ATL Rule의 일부
Table 1. A part of ATL rule from State Metamodel to Cause Effect Metamodel

```
--@path SMDM=/SMDtoCauseEffect/SMD.ecore
--@path CEM=/SMDtoCauseEffect/CED.ecore
module SMDMtoCEM;
//스테이트 다이어그램에서 원인-결과 다이어그램의 생성
create OUT : CEM from IN : SMDM;
helper def : getCEMRootModel() : OclAny =
        CEM!CauseEffectModel.allInstances().first()
;
//스테이트 메타모델 SMDModel에서 원인-결과 다이어그램 메타모델 CauseEffectModel로 변환
rule ModelCreation {
        from
                i : SMDM!SMDModel
        to
                mm : CEM!CauseEffectModel ()
}
//원인-결과 다이어그램의 원인부분 정의
rule CreateCause(name : String) {
```

```
        to
                newCuase : CEM!Cause(
                        name <- name
                )
        do {
                newCuase;
        }
}
//원인-결과 다이어그램의 결과부분 정의
rule CreateEffect(name : String, type : String) {
        to
                newEffect : CEM!Effect(
                        name <- name,
                        eType <- type
                )
        do {
                newEffect;
        }
}
```

Ⅳ. 사례연구

사례로는 T업체의 자동차의 Junction Box를 스테이트 다이어그램으로 모델링을 한 후 테스트케이스를 추출했다[12]. 비교를 위해 기존 테스트케이스는 업체에서 추출한 테스트케이스를 사용하였다[12]. 사례연구로써 Junction Box의 28가지 기능중 한가지인 Cooling System을 기반으로 모델링 한 후 테스트케이스를 추출했다. 그림 6은 자동차 쿨링 시스템의 일반적인 스테이트 다이어그램을 표현한 것이다. OFF상태에서 외부 신호에 의해 시스템이 ON되고 Main Relay에 의해 LOW팬, HIGH팬, OFF상태로 전이된다. 이때 조건 값은 팬작동 여부를 판단하는 센서와 온도센서이다.

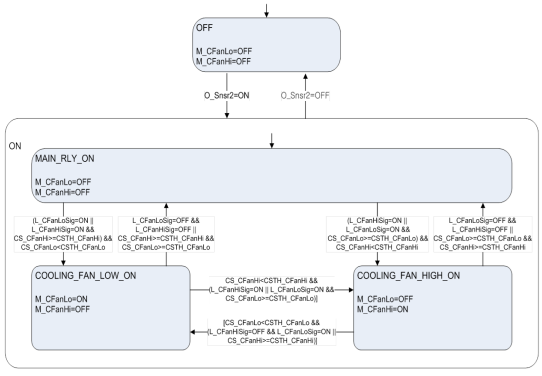

그림 6. 쿨링 시스템 스테이트 다이어그램
Fig. 6. Cooling System State Diagram

시작 조건	테스트 내용	예상결과
O_Snsr2=OFF L_CfanLoSig=OFF L_CfanHiSig=OFF CS_CFanLo < CSTH_CFanLo CS_CFanHi < CSTH_CFanHi	변화없음	O_CFanLo=OFF O_CFanHi=OFF
	O_Snsr2=ON	O_CFanLo=OFF O_CFanHi=OFF
	L_CfanLoSig=ON	O_CFanLo=OFF O_CFanHi=OFF
	L_CfanHiSig=ON	O_CFanLo=OFF O_CFanHi=OFF
	CS_CFanLo >= CSTH_CFanLo	O_CFanLo=OFF O_CFanHi=OFF
	CS_CFanHi >= CSTH_CFanHi	O_CFanLo=OFF O_CFanHi=OFF
O_Snsr2=OFF L_CfanLoSig=OFF L_CfanHiSig=OFF CS_CFanLo < CSTH_CFanLo CS_CFanHi >= CSTH_CFanHi	변화없음	O_CFanLo=OFF O_CFanHi=OFF
	O_Snsr2=ON	O_CFanLo=OFF O_CFanHi=OFF
	L_CfanLoSig=ON	O_CFanLo=OFF O_CFanHi=OFF
	L_CfanHiSig=ON	O_CFanLo=OFF O_CFanHi=OFF
	CS_CFanLo >= CSTH_CFanLo	O_CFanLo=OFF O_CFanHi=OFF
	CS_CFanHi < CSTH_CFanHi	O_CFanLo=OFF O_CFanHi=OFF
O_Snsr2=OFF L_CfanLoSig=OFF L_CfanHiSig=OFF CS_CFanLo >= CSTH_CFanLo CS_CFanHi < CSTH_CFanHi	변화없음	O_CFanLo=OFF O_CFanHi=OFF
	O_Snsr2=ON	O_CFanLo=OFF O_CFanHi=OFF
	L_CfanLoSig=ON	O_CFanLo=OFF O_CFanHi=OFF
	L_CfanHiSig=ON	O_CFanLo=OFF O_CFanHi=OFF
	CS_CFanLo < CSTH_CFanLo	O_CFanLo=OFF O_CFanHi=OFF
	CS_CFanHi >= CSTH_CFanHi	O_CFanLo=OFF O_CFanHi=OFF

그림 7. 기존 테스트 케이스
Fig. 7. Existing Testcase

그림 7은 기존 업체에서 사용하는 테스트케이스이다. 첫 번째 테스트케이스에 관해 설명하면 다음과 같다. 테스트 조건은 팬 모듈 작동여부센서(O_Snsr2)가 OFF이고 LOW팬 작동여부센서(L_CfanLosig)이 OFF이고 HIHGH팬 작동여부센서(L_CfanHiSig)이 OFF이고 LOW팬의 온도센서(CS_CFanLo)가 LOW팬 작동기준온도(CSTH_CFanLo)보다 작고 HIGH팬의 온도센서(CS_CFanHi)가 HIGH팬 작동기준온도(CSTH_CFanHi)보다 작을 경우 예상되는 팬의 동작은 OFF임을 나타낸다.

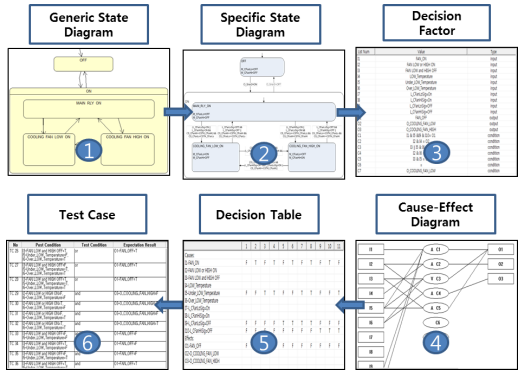

그림 8. 테스트 케이스 생성 단계
Fig. 8. Testcase Generation step

그림 8은 제안한 모델변환 기법을 통해 테스트케이스를 생성하는 단계를 보여준다. 일반적인 스테이트 다이어그램에서 명세적인 스테이트 다이어그램으로 변환하고 이를 통해 Decision Factor를 추출한 후 원인-결과 다이어그램으로 변환한다. Decision Table로 변환한 후 Test Case를 생성한다. 그림 9와 같이 제안한 기법을 통한 테스트 케이스를 살펴보면 다음과 같다. Post Condition에서 각각의 테스트 조건이 나열되고 Test Condition에서 각각의 조건들이 어떻게 결합되는지를 설정할 수 있다. TC1은 Post Condition은 팬의 동작여부센서, 팬의 온도센서, 팬의 기준온도를 나타내며 각 요소들은 and 로 조합된다. 업체의 첫 번째 테스트케이스와 유사하나. Test Condition의 조합에 따라 더욱 풍부하고 다양한 Coverage를 갖는 테스트케이스를 추출할 수 있다. 기존 업체의 방법으로 테스트케이스를 추출했을 경우 64개이고 제안한 방법을 사용했을 경우 72개로 더욱 풍부한 테스트케이스를 추출할 수 있었다.

No	Post Condition	Test Condition	Expectation Result
TC 1	I1-FAN_ON=F, I5-Under_LOW_Temperature=F, I9-L_CFanLoSig=OFF=F, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 2	I1-FAN_ON=T, I5-Under_LOW_Temperature=F, I9-L_CFanLoSig=OFF=F, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 3	I1-FAN_ON=F, I5-Under_LOW_Temperature=T, I9-L_CFanLoSig=OFF=F, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 4	I1-FAN_ON=T, I5-Under_LOW_Temperature=T, I9-L_CFanLoSig=OFF=F, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 5	I1-FAN_ON=F, I5-Under_LOW_Temperature=F, I9-L_CFanLoSig=OFF=T, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 6	I1-FAN_ON=T, I5-Under_LOW_Temperature=F, I9-L_CFanLoSig=OFF=T, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 7	I1-FAN_ON=F, I5-Under_LOW_Temperature=T, I9-L_CFanLoSig=OFF=T, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 8	I1-FAN_ON=T, I5-Under_LOW_Temperature=T, I9-L_CFanLoSig=OFF=T, I10-L_CFanHiSig=OFF=F	and	O1-FAN_OFF=F
TC 9	I1-FAN_ON=F, I5-Under_LOW_Temperature=F	and	O1-FAN_OFF=F

그림 9. 모델 변환을 통한 테스트 케이스
Fig. 9. Testcase Using Model Transformation

V. 결 론

본 논문에서는 MDA 메커니즘인 메타모델과 모델변형기법을 채택하여, UML내 스테이트 다이어그램을 통해 테스트케이스를 자동 추출하는 메커니즘을 제안하였다. 메타 모델을 기반으로 하는 모델변환 및 변환 규칙과 스테이트 다이어그램, 원인-결과 다이어그램, 결정 테이블을 메타모델화한 후 각각의 모델간의 변환 룰을 정의하였다. 이를 기반으로 메타 모델 기반의 테스트케이스를 자동 생성이 가능하다. 그리고 기존 T업체의 테스트케이스와 자동 생성된 테스트케이스들이 유사하고 더 많은 경우수를 발생하는 것이 가능하다. 결과적으로 테스트 설계 및 테스트케이스 추출비용과 시간을 줄이고 임베디드 소프트웨어의 품질을 높일 수 있다. 향후 이종 임베디드 소프트웨어에 적용하기 위해선 각 메타 모델 및 변환 규칙의 보강이 필요하다.

참 고 문 헌

[1] “OMG Unified Modeling Language Specification” Version 1.4, September2001.

[2] “OMG Unified Modeling Language (OMG UML), Superstructure”, Version 2.4.1, 2010.

[3] Hyun-Jeong Jo, Jong-Gyu Hwang “Analysis of S/W Test Coverage Automated Tool & Standard in Railway System” Journal of the Korea Academia-Industrial cooperation Society / Vol.11, no.11, pp.4460-4467, 2010,

[4] Bart Broekman, Edwin Notenboom 2002, “Testing Embedded Software”

[5] Dong-Kuk Ryu, Young-Chul Kim “Design and Implementation of M&S Based Test Environment for Interoperability Verification of Heterogeneous Composite Embedded System” Journal of Korean Institute of Information Technology, vol. 7, issue 2, pp. 33-40, Apr 2009.

[6] “Eclipse Modeling Framework Project(EMF)

[7] Stephan WeiBleder, Dehal Sokenou, (June 2008). "Cause-Effect Graphs for Test Models Based on UML and OCL" Treffen der GI-Fachgruppen TAV und RE.

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

[9] Woo, Su Jeong, (December 2012) "Metamodel oriented Automatic Test Case Generation Based on

Transforming UML 2.4.1 Message-Sequence Diagram via Cause-Effect Diagram"

[10] Wooyeol Kim, Hyunseung Son, Junbeom Yoo, Young B. Park, R. Youngchul Kim "A Study on Target Model Generation for Smartphone Applications using Model Transformation Technique".

[11] "Simplified State Machine Execution Verification" http://web.univ-pau.fr/~ecariou/contracts/simplified-state-machine.html

[12] Eclipse ATL, http://eclipse.org/atl/

[13] Dong Ho Kim, Hyun Seung Son, Woo Yeol Kim, Robert Young Chul Kim "Test Case Extraction for Intelligent Power Switch Heterogeneous Vehicles " ITCS(Information Technology and Computer Science)2012

※ 본 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업(2012-0001845)과 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임.

저자 소개

김 동 호(정회원)

- 2005년 : 홍익대학교 소프트웨어공학 공학석사
- 2009년 : 홍익대학교 소프트웨어공학 박사 수료

<주관심분야 : 테스트 성숙도 모델(TMM), 임베디드 소프트웨어 개발 방법론, 테스트 프로세스,. 모델 기반 테스팅, 메타모델>

김 영 철(종신회원)

- 2000년 : Illinois Institute of Technology (공학박사)
- 2000년 ~ 2001년 : LG 산전 중앙연구소 Embedded system 부장
- 2001년 ~ 현재 : 홍익대학교 컴퓨터정보통신 교수

<주관심분야 : 테스트 성숙도 모델 (TMM), 임베디드 소프트웨어 개발 방법론, 모델 기반 테스팅, 메타모델, 비즈니스 프로세스 모델, 사용자 행위 분석 방법론>