제19권 제1호
ISSN 2005-0011

제37회 춘계학술발표대회 논문집(하)

일자 : 2012년 4월 26일(목)~28일(토)

장소 : 순천대학교 70주년 기념관

주최 : 사단법인 한국정보처리학회

후원 : 지식경제부, 2012 여수세계박람회 EXPO 2012 YEOSU KOREA

협찬 : 롯데정보통신, 삼성SDS, SK C&C, 정보통신산업진흥원, 한국게임과학고등학교, 이큐스앤자루, 커미트, 한국생산성본부, 한솔인티큐브, 나라인포테크, 마크애니, 동하테크, LG엔시스 (무순)

春

사단법인 한국정보처리학회
Korea Information Processing Society

347. 효과적인 SCL 엔지니어링 툴 설계를 위한 관련 S/W 툴 분석 KIPS_C2012A_0312
채창훈*, 정남준, 최효열, 안용호(한국전력공사) • 1215

348. LTL Synthesis를 이용한 다중 로봇 시뮬레이터 개발 KIPS_C2012A_0313
김성회*, 권령구, 권기현(경기대학교) • 1219

349. 앱개발 도구 : HTML5, App Inventor, M-BizMaker 어느 것을 선택할 것인가 KIPS_C2012A_0318
김시우*(숭의여자대학교), 전정훈(동덕여자대학교) • 1223

350. 실시간 GPS 좌표추적을 이용한 성범죄자 추적 및 알림 어플리케이션 KIPS_C2012A_0319
이동성, 김정윤, 황선명*(대전대학교) • 1226

351. 달빅 DEX 파일 브라우저의 설계 및 구현 KIPS_C2012A_0334
소경영*, 정택희(전북대학교), 박종필, 고광만(상지대학교) • 1228

352. 안드로이드 애플리케이션 GUI 테스팅 도구 적용 및 사례연구 KIPS_C2012A_0342
김태균*, 권기현(경기대학교) • 1231

353. 결함 위치 추적을 위한 테스트 케이스 자동 생성 기법 KIPS_C2012A_0346
박창용*, 김준희, 류성태, 윤현상, 이은석(성균관대학교) • 1235

354. 이종 임베디드 테스팅을 위한 MDA (Model Driven Architecture)기반의 테스트 프로세스 개선 및 확장에 관한 연구 KIPS_C2012A_0358
김동호*, 손현승, 김우열, 김영철(홍익대학교) • 1239

355. 테스트 프로세스 개선 모델(TPI next)을 통한 테스트 성숙도 모델 확장에 관한 연구 KIPS_C2012A_0361
김기두*(한국정보통신기술협회), 김영철(홍익대학교) • 1243

356. 사용자 니즈를 통한 사용자 선호도 요구사항 추출 및 우선순위화 KIPS_C2012A_0364
박보경*, 김영철(홍익대학교) • 1247

357. 원인-결과 다이어그램과 접목을 위한 메시지-순차적 다이어그램 확장 연구 KIPS_C2012A_0365
우수정*, 손현승, 김영철(홍익대학교) • 1251

358. 클라우드 컴퓨팅에서 BPEL분석 및 검증을 위한 Onion언어로의 변환 KIPS_C2012A_0376
최재홍*, 온진호, 이문근(전북대학교) • 1255

359. 비즈니스 프로세스 프레임워크상에서의 비즈니스 프로세스 모델, 서비스와 컴포넌트기반 개발의 매핑을 통한 소프트웨어 재사용 패러다임 KIPS_C2012A_0377
서채연*, 문소영, 김영철(홍익대학교) • 1259

360. 정보시스템감리와 회계감사의 적정성 비교 KIPS_C2012A_0431
권호열*(강원대학교) • 1262

361. 공공부문 정보화사업 PMO 도입의 과제 KIPS_C2012A_0432
권호열*(강원대학교) • 1264

정보처리응용(IT교육 등)

362. 항만건설공사 전자설계 · 준공도서 서비스 시스템 개발 KIPS_C2012A_0008
정성윤*, 김남곤(한국건설기술연구원) • 1269

363. 도로현황조서시스템 구축방안 연구 KIPS_C2012A_0011
김영진*, 김병곤, 임재규(한국건설기술연구원) • 1271

364. 정량적 기고서 분석을 통한 MPEG 표준화 과정 연구 KIPS_C2012A_0016
이광훈*, 김현규, 장의선(한양대학교) • 1275

365. 매쉬업을 위한 Open API 유사성 탐색 방법 KIPS_C2012A_0019
이용주*(경북대학교) • 1279

원인-결과 다이어그램과 접목을 위한 메시지-순차적 다이어그램 확장 연구

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

A Study on Extending Message-Sequence Diagram for Mapping Cause-Effect Diagram

SuJeong Woo*, Hyun Seung Son, R. Young Chul Kim
Dept. of CIC, Hongik University

요 약

본 논문은 Gary E. Mogyorodi[1]가 제시한 기법을 기반으로 Use-Case Approach 접목을 통해 테스트케이스 추출을 제안하고자 한다. 최근 이슈가 되고 있는 임베디드 시스템은 기존의 결정적 소프트웨어와 달리 비결정적, 실시간 또는 병렬적 시스템이다. 그래서 이러한 복잡한 시스템을 모델링 하기 위해서, 메시지-순차적 다이어그램을 확장을 통해 해결하고자 한다. 또한 Gary E. Mogyorodi[1]가 제시한 기법과 확장된 메시지-순차적 다이어그램을 접목을 통해 Test Case 를 생성하기 및 추출하고자 한다. 이 테스트케이스로 선 시험함으로써 실제 개발과 구현단계에서 오류를 참조하여 시간과 비용을 줄이고자 한다.

1. 서론

최근 임베디드 소프트웨어 개발에 가장 중요한 이슈는 사용자 요구사항을 충족하는지를 검증하는 것이다. 임베디드 소프트웨어는 다른 소프트웨어와 달리 시간이나, 효율, 하드웨어 등의 분야에서 제약이 있다. 이러한 문제점을 위해 디자인 단계를 확장하여 테스트케이스를 추출하여 효율적으로 검증하고자 한다.

디자인 단계에서의 UML 은 "Unified Modeling Language"의 약자이며, "통합된 모델링 언어"라고도 한다. 즉 시스템을 모델링하는 과정에서 사용하는 언어이다. UML 의 다양한 모델 언어 중 Message Sequence Diagram(MSD)은 여러 객체의 상호작용을 표현하는 것이다.

또한 Cause-Effect Diagram(CED)은 프로그램의 외부 명세에 의해 원인에 해당되는 입력 조건과 그 원인으로부터 발생되는 출력 결과를 논리적으로 연결시킨 Diagram 으로 표현하는 방법[1]이다.

기존 모델 기반 테스트케이스 발생 방법으로는 State Machine 을 통하여 테스트케이스를 발생[5,6] 하였고, UseCase Diagram 을 통해서 테스트케이스를 발생시키는 방법[7] 등 있다. 그러나 본 논문은 Gary E. Mogyorodi[1]가 제시한 CED 를 통하여 테스트케이스를 발생시키는 것은 최소의 테스트케이스로 100%의 기능적인 요구사항 커버리지를 만족 시킬 수 있는 검증된 방법[1]이기 때문에 CED 를 통해서 테스트케이스를 발생시키고자 한다. 이를 통하여 확장된 MSD 와 CED 를 접목(Mapping)통해 자동적으로 테스트케이스를 생성하고자 한다.

본 논문의 검증방법은 기존의 V 모델을 기반으로 Modeling & Simulation(M&S), UML Modeling, Cause-Effect Diagram(CED), Decision Table(DT), Test Case(TC), 마지막으로 가상환경에서 TC 를 테스트하는 단계로서 계속 순환하는 모델[8]이다. 각 단계 중 MSD 가 CED 와 Mapping 되는 과정을 본 논문의 초점으로 설명하고자 한다.

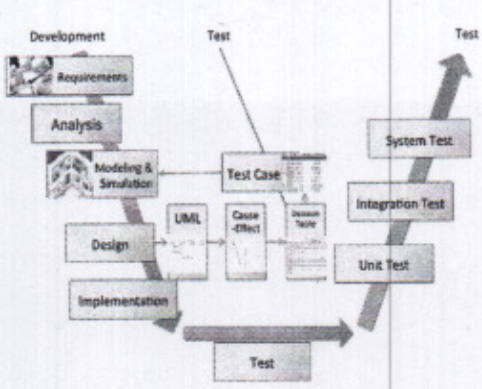
본 논문의 구성은 다음과 같다. 2 장에서는 관련연구에 대해 설명하고 3 장은 확장된 MSD 와 CED 을 설명하고 4 장은 Mapping 방법에 대해 소개한다. 마지막으로 5 장에서는 결론 및 향후 연구에 대해서 기술한다.

2. 관련연구

그림 1 은 Pre-Testing Process[8]이다. 첫 번째 M&S 단계는 사용자의 요구사항을 분석하여 조건에 맞게 가상환경에 가상개체를 구축한다. 두번째 UML Modeling 단계 중 MSD 으로 순차적인 상호작용을 표현한다. 이때 MSD 를 확장하여 CED 에 Mapping 한다. 세번째 단계인 CED 단계는 기존의 Input, Output 만 나타내던것을 Input, Condition, Output 으로 확장하여 나타

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

낸다. 네번째 단계는 CED 를 테이블화 하여 DT 를 나타내고 이를 통하여 다섯번째 단계에서 TC 를 발생하도록 한다. 발생된 TC 를 가상환경에 가상개체를 테스트하는 Pre-Testing 의 Process 가 완성된다.



(그림 1) Pre-Testing Process

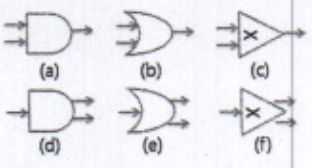
3 장에서는 확장된 MSD 와 CED 의 Mapping 을 언급하고자 한다.

3. 순차적다이어그램의 확장

3.1 Concurrent Message Diagram(CMD)

일반적인 소프트웨어를 모델링 하기 위해서는 UML 을 사용한다. 기존의 UML 은 복잡하고 주위환경에 신속하게 대처해야 하는 실시간 임베디드 시스템에서는 사용하기에는 적합하지 않다[2]. 임베디드 시스템은 병렬적 시스템이고 실시간 또는 비결정적 시스템이기 때문에 UML 을 확장해야 한다. 여러 가지 UML 을 표현하는 Diagram 들 중에 MSD(Message Sequence Diagram)을 확장이 필요하다.

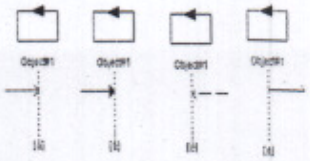
기존의 Sequence Diagram 은 객체와 액터로만 이루어져 있다. 하지만 본 논문은 임베디드를 위한 도구이기 때문에 이바 야콥슨(Ivar Hjalmar Jacobson)의 Stereotype 을 채택하였다. Stereotype 의 객체는 인터페이스, 컨트롤, 서비스 3 가지로 나눈다. 또한 기존의 Sequence Diagram 은 결정적 시스템이기 때문에 한가지 메시지가 액터와 객체를 순차적으로 상호교류를 할 수 있다. 첫 번째는 비결정적 시스템, 두번째는 실시간으로 수행되는 Real Time System, 세번째는 한가지 메시지만 수행되는 것이 아니라 병렬적인 복잡한 시스템을 모델링이 가능하도록 확장하였다. 이를 위해 그림 2 와 같이 논리 게이트(AND, OR, NOT)를 차용하여 확장했다.



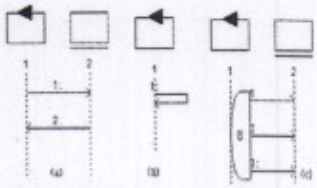
(그림 2) Incoming, Outgoing Messages

그림 2 의 (a), (b), (c)는 두개의 Input 메시지가 들어가서 한개의 Output 메시지가 나오게 되고, (d), (e), (f)는 한 개의 Input 메시지가 들어가 한 개의 Output 메시지가 나오게 된다.

그림 3 은 확장된 Concurrent Message Diagram 의 메시지는 4 가지 타입으로 구성 되어 진다. [a]는 일반적인 Flat Message 를 나타내고, [b]는 동기 Message 고 [c]는 리턴 Message, [d]는 비동기 Message 을 나타낸다.



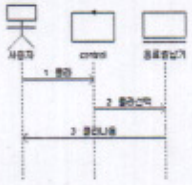
(그림 3) 비동기화, 동기화 Messages



(그림 4) Communication

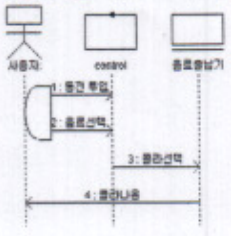
그림 4 은 Communication 의 그림이다. (a)은 객체 간 통신인 Peer-to-Peer 를 나타내고, (b)는 자기 자신과의 통신은 Message to Self 를 나타내며, (c)은 비동기 메시지를 보내기만 하는 Broadcasting 을 나타내고 있다[3].

아래부터는 확장된 객체와 이를 전달하는 메시지들을 적용한 예를 보여 준다.

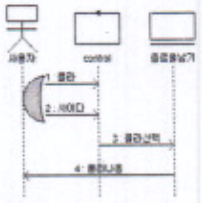


(그림 5) 1:1 확장된 Sequence Diagram

그림 5 은 사용자가 자판기에서 콜라를 선택했을 때 컨트롤이 인식하고 음료수 출납기에 명령어를 보내 콜라가 나오는 1:1 CMD 을 보여 준다.



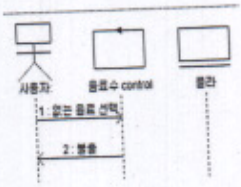
(그림 6) AND gate on Concurrent Message Diagram



(그림 7) OR gate on Concurrent Message Diagram

그림 6 은 사용자가 동전을 투입하고 음료를 선택했을 때 두 가지 경우가 만족하기 때문에 음료수가 나오는 AND 게이트 개념의 CMD 을 보여 준다.

그림 7 는 사용자가 자판기에서 콜라 또는 사이다를 선택했을 경우 한가지만 만족하면 음료수 출납기에서 음료수가 나오게 되는 OR 게이트 개념의 CMD 을 보여 준다.



(그림 8) NOT gate on Concurrent Message Diagram

그림 8 은 자판기에서 없는 음료를 선택하게 되면 아무것도 나오지 않는 NOT CMD 을 나타낸다.

3.2 Cause-Effect Diagram(CED)

Cause-Effect Diagram 은 명세를 원인(Cause)와 결과(Effect)로 분류해서 상세하게 분석한다. 외부 명세의 의미적 내용을 분석하여 그 외부 명세를 입력(Cause), Condition, 부울 형태로 변형된(Effect) 그래프 사이의 논리 관계로서 재정리 한 것이다. 또한 최소의 테스트케이스로 100%의 기능적인 요구사항 커버리지를 만족시킬 수 있는 검증된 방법[1]이다. 그림 9 은 기존의 CED 을 나타낸 것이다.

기존의 CED 는 Input, Output 로만 이루어져 있다.

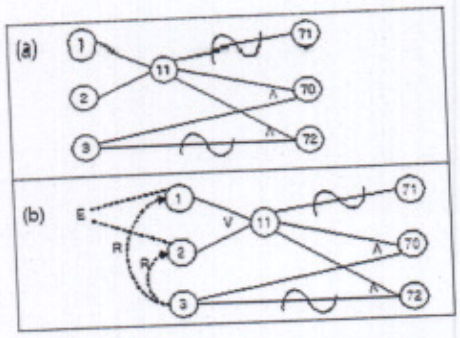
하지만 본 논문에서는 임베디드 시스템을 위한 도구 이다. 임베디드 시스템은 Control 이 가장 중요한 부분이다. 이를 표현하기 위해서는 Input 과 Output 사이에 Condition 을 추가함으로써 CMD 와 Mapping 할 수 있다.

기호	설명	기호	설명
IF X then Y	X=1이면 Y=1 X=0이면 Y=0	Require	원인 Y가 존재하기 위해서는 반드시 원인 X가 존재 즉, X=1이면 Y=1 또는 Y=0 (X=0 일때 Y=0, Y=1 불가능)
IF X then Y	X=1이면 Y=0 X=0이면 Y=1	Exclusive	원인 X와 원인 Y는 동시에는 존재하지 않음 즉, X=1이면 Y=0 X=1이면 Y=1 (X=Y=0 가능, X=Y=1은 불가능)
IF X OR Y Then Z	X=1, Y=1 또는 X=0, Y=1 이면 Z=1 또는 X=1, Y=1 X=0, Y=0 이면 Z=0	Inclusive	원인 X와 원인 Y는 적어도 그 한쪽에 반드시 존재 즉, X=1일때 Y=1, Y=0 Y=1일때 X=1, X=0 (X=Y=0은 불가능)
IF X AND Y Then Z	X=1, Y=1이면 Z=1 X=1, Y=0 X=0, Y=1 X=0, Y=0 이면 Z=0	one and only one	원인 X와 원인 Y는 적어도 그 한쪽에 존재하며 다른 한쪽은 존재하지 않음 즉, X=1이며 Y=0 또는 X=0이며 Y=1 (X=Y=0은 불가능)
		Implies	결과 X=가 존재하면 결과 Y는 존재하지 않음. 즉, X=1이면 Y=0

(그림 9) 기본적인 Cause-Effect Diagram

그림 9 은 기본적인 CED 의 기호와 설명을 언급하였다. 또한 그림 10 은 복잡한 CED 상에서 (a)은 원인-결과만 나타냈으며 (b)은 제한조건을 포함한 원인-결과를 나타낸다.

기존의 UML 은 바로 테스트케이스를 발생할 수 없다. 그래서 UML Modeling 을 확장하여 Cause-Effect

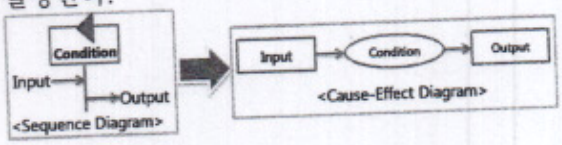


(그림 10) The complex Cause-Effect Diagram

Diagram 과 접목하고자 한다.

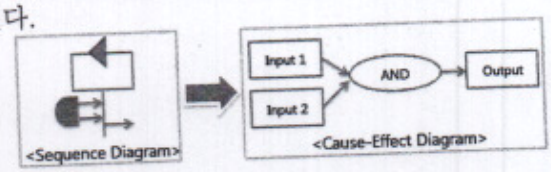
4. 확장된 Concurrent Message Diagram 을 통한 Cause-Effect Diagram 접목

아래는 확장된 MSD 와 CED 를 Mapping 시키는 그림을 설명한다.



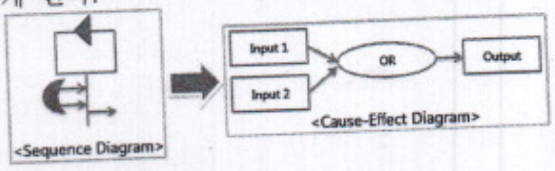
(그림 11) 1:1 Mapping

그림 11 는 1:1 관계이다. 한 개의 메시지가 들어가서 컨디션에 만족하면 한 개의 아웃메시지가 나오게 된다.



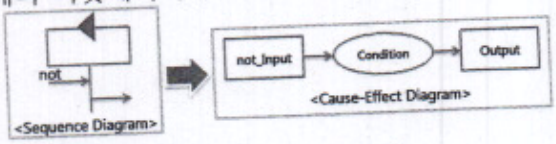
(그림 12) AND on Mapping

그림 12 은 AND 게이트를 응용한 Mapping 구조이다. 두 개의 메시지가 들어가는데 AND 게이트와 같이 두 개의 메시지가 True 일 때만 한 개의 아웃메시지가 나오게 된다.



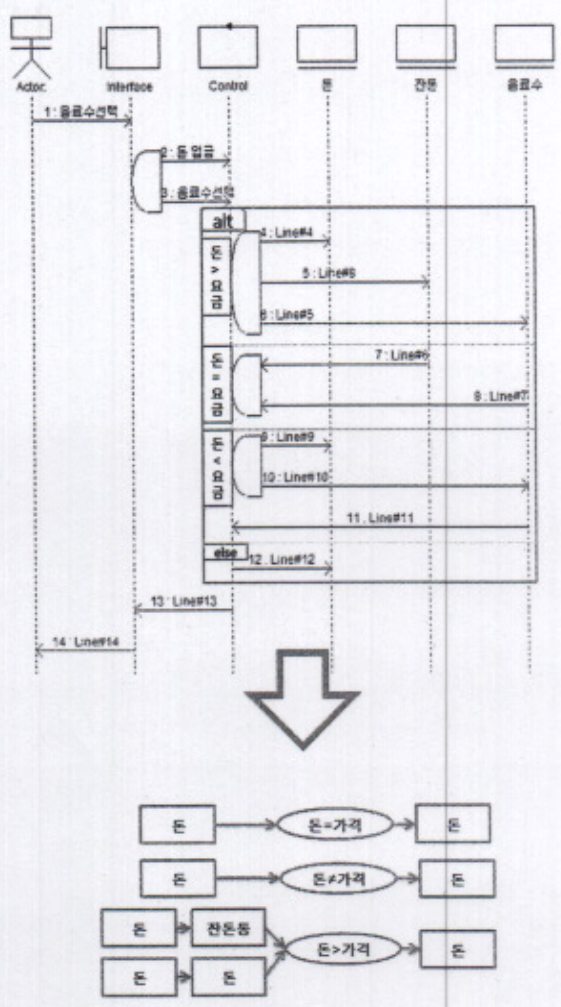
(그림 13) OR on Mapping

그림 13 은 OR 게이트를 응용한 Mapping 구조이다. 두 개의 메시지가 들어가는데 OR 게이트와 같이 두 개의 메시지 중 한 가지의 메시지라도 True 하게 되면 한 개의 아웃메시지가 나오게 된다.



(그림 14) NOT on Mapping

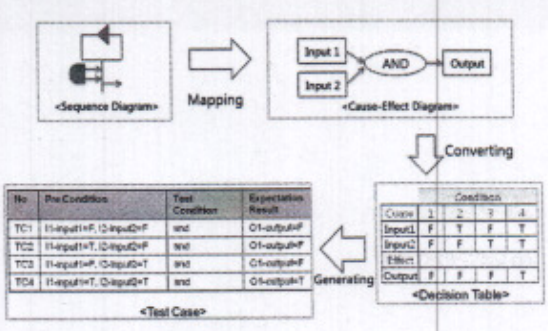
그림 14 은 Not 게이트를 응용한 Mapping 구조이다. Not 인 메시지가 들어가면 조건에 맞게 아웃메시지가 나오게 된다.



(그림 15) 복잡한 Mapping

그림 15 은 복잡한 Mapping 구조이다. 사용자가 지불한 금액이 음료의 가격과 같거나, 많거나, 적었을 때의 경우를 나타냈다. 음료의 가격과 같으면 음료수만 나오고, 지불금액이 많을 경우 잔돈과 음료수가 같이 나오게 된다. 적었을 때의 경우는 음료수도 잔돈도 나오지 않는 경우이다.

5. 테스트케이스 발생 메카니즘



(그림 16) 테스트케이스 발생

그림 16 은 테스트케이스 발생 과정을 나타낸 것이다. 확장된 CMD with AND Gate 을 컨트롤을 중심으로 Input 메시지 2 개와 Condition, 1 가지의 Output 을 CED 에 접목을 시킨다. 이를 통해서 Input, Output 을 Condition 에 맞게 True, False 로 나타낸 결정테이블을 작성한다. 이를 통하여 자동적으로 테스트케이스가 발생되는 과정이다.

6. 결론

본 논문은 복잡하고 다양한 임베디드 시스템을 모델링과 테스트케이스 추출에 초점을 두고 있다. 즉, 비결정적, 실시간, 병렬적, 확률적인 복잡한 임베디드 시스템을 모델링 하기 위해서 메시지-순처적 다이어그램을 확장하였다. 기존의 Sequence Diagram 의 객체를 인터페이스, 컨트롤, 서비스 3 가지로 세분화하였고, 논리게이트를 이용하여 병렬메시지 등등 전달 방법을 제시하였다. 이를 통해서 복잡한 시스템을 모델링 할 수 있게 했다. 또한 Gary E. Mogyorodi[1]를 통해 최소한의 테스트케이스로 100%기능적인 요구사항 커버리지를 만족 시킬 수 있는 검증된 방법인 CED 를 확장된 MSD 에 접목시켜 자동적으로 테스트케이스가 발생될 수 있게 했다. 이 테스트케이스를 시험함으로서 실제 개발과 구현단계에서 오류를 참조하여 시간과 비용을 줄일 수 있는 효과를 볼 수 있으리라 본다.

참고문헌

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

[2] 김예진, 손현승, 김우열, 서진원, 김동호, 서윤숙, 류동국, 김영철, " 확장된 xUML 을 사용한 장애물 회피용 소형 무인차 모델링 연구" , 한국소프트웨어공학기술 합동 워크샵, 2007

[3] 김우열, " Model Driven Architecture 기반의 임베디드 소프트웨어 모델링에 관한 연구" ,홍익대학교, 2005

[4] Kyu Won Kim, Woo Yeol Kim, Hyun Seung Son, and Rober Young Chul Kim ,"A Validation Process for Real Time Transactions" ,Software Engineering Business Continuity and Education Vol.257

[5] Houde Fekih, Leila Jemni Ben Ayed, Stephan Merz ," Transformation of B Specifications into UML Class Diagram and state Machines" , ACM Vol.2

[6] WooYeol Kim, HyunSeung Son, Robert YoungChul Kim ,"A Study on Test Case Generation Based on State Diagram in Modeling and Simulation Environment", CCIS 199 ,2011

[7] R.YoungChul Kim, Bok-Gyu Joo, Kyng-Chul Kim, ByungKook Joen, "Scenario Based Testing & Test Plan Metrics Based on A Use Case Approach for Real Time UPS", LNCS, 2003

[8] 우수정, 손현승, 김우열, 김제승, 김영철, " Pre-Testing 를 위한 M&S 기반 테스트케이스 추출 연구 , 소프트웨어공학학회 ,2012