모델 변환 기법을 활용한 윈도우즈 모바일 어플리케이션 개발

(Development of Windows Mobile Applications using Model Transformation Techniques)

김 우 열 † 손 현 승 †
(WooYeol Kim) (HyunSeung Son)
김 재 승 †† 김 영 철 †††
(JaeSeung Kim) (R. YoungChul Kim)

요 약 기존의 스마트폰용 소프트웨어 개발은 플랫폼에 종속적으로 만들어 지기 때문에 플랫폼별로 개발해야한다. 애플은 코코아플랫폼, 구글은 안드로이드, 마이크로소프트는 윈도우 모바일 등 각 벤더마다 고유의 플랫폼으로 개발한다. 본 논문에서는 한 번의 개발을 통해 이종의 소프트웨어 개발할 수 있도록 모델변환기법을 적용한다. 이 방법은 독립 모델과 종속 모델을 분리하고 이 둘의 차이를 변환언어를 통해 자동 변환하는 기법이다. 모델 변환 기법 수행을 위해서는 모델, 메타모델, 모델변환언어가 요구된다. 본 논문에서는 스마트폰에 적용하기 위해서 모델은 UML, 메타모델은 UML 메타모델, 모델변환언어는 ATL을 사용하였다. 적용사례로 윈도우 모바일 플랫폼환경에서 모델변환을 이용하여 개발하는 방법을 보여준다. 본 논문의 플랫폼 독립모델을 사용하고 모델 변환 규칙을 재정의 하면 아이폰, 안드로이드 등의 이종의 플랫폼으로 변환이 가능하다.

키워드 : 모델변환, 스마트폰, ATL, 윈도우즈 모바일

Abstract The existing smart-phone software is dependent on the platform, which should be developed per each different platform. Each vendor will develop its own platform such as Apple's Cocoa platform, Google Android, Microsoft Windows Mobile, etc. In this paper, we apply model transformation technique for developing heterogenous software at a time in heterogenous smart phone area. This approach separates the independent model and dependent model. and automatically transforms the difference between them with model transformation language. To execute model transformation, it is required with meta model, model transformation language. In this paper, we are applied to smart-phones as follows: model will be UMLmodel, metamodel be UML metamodel, and choose ATL as Model transformation language. We show examples of the Windows Mobile platform environment to be developed using model transformation. As a result, if we use platform-independent model in this paper and redefine model transformation rules for the iPhone or Android, it will be automatically transformed into heterogenous platforms.

Key words : Model Transformation, Smart Phone, ATL, Windows Mobile

1. 서 론

최근(2009년 후반기) 국내에 아이폰이 출시되면서 국내에도 스마트폰에 대한 관심이 증가하고 시장이 활성화 되었다. 기존의 피처폰과(Feature Phone)는 다르게 스마트폰(Smart Phone)은 다양한 컨텐츠를 활용할 수 있는 점이 매력적이다. 메일을 주고받을 수 있으며 게임, 워드프로세스, 내비게이션 등 다양한 어플리케이션을 활용할 수 있다. 결국 스마트폰 시장에서 다양한 컨텐츠를 얼마나 빠르고 안전하게 개발해 줄 수 있는 환경이 매우 중요하다.

소프트웨어 개발을 빠르게 하기 위해서는 기존에 만들었던 자원들을 최대한 활용 가능한 재사용 체계가 갖추어져 있어야 한다. 그러나 스마트폰 소프트웨어는 시스템에 종속적인 특성을 지니고 플랫폼 중심으로 개발이 진행되기 때문에 이종 시스템에 재사용이 어렵다[1]. 게다가 각 휴대폰 제조 및 공급 회사마다 고유의 플랫폼을 이용한 개발만을 허용하고 있기 때문에 이를 해결할 수 있는 방법이 필요한 실정이다.

MDD(Model Driven Development)는 플랫폼에 독립적인 모델을 플랫폼 종속 적인 모델로 변경하고 그 종속적인 모델을 통해 코드를 생성한다. MDD는 이러한 과정을 자동화 한다. 만약 응용 프로그램이 다른 환경에

· 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2010-(C1090-1031-0008))과 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임
· 이 논문은 2010 한국컴퓨터종합학술대회에서 '모델 변환을 이용한 윈도우즈 어플리케이션 개발'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 홍익대학교 전자전산공학과
john@hongik.ac.kr
sonhs1000@hotmail.com
†† 비 회 원 : 트라이콘 이사
ceo@tricon.co.kr
††† 정 회 원 : 홍익대학교 컴퓨터정보통신공학과 교수
bob@hongik.ac.kr
(Corresponding author임)

논문접수 : 2010년 8월 12일
심사완료 : 2010년 10월 4일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제11호(2010.11)

서 개발이 필요하다면 독립 모델을 그 환경에 대한 모델로 변환하여 코드를 생성하면 된다. 이때 응용 프로그램의 독립 모델을 수정할 필요는 없다. 이와 같은 방법으로 독립 모델을 재사용하여 코드의 생산성을 높일 수 있다[2-5]. 즉, 스마트폰 개발환경에 모델변환기법을 적용하면, 고수준의 모델을 재사용하여 이종 소프트웨어를 빠르게 개발가능하다.

본 논문에서는 기존의 윈도우 모바일 소프트웨어 개발에 모델 변환 기법을 적용하여 이종의 다른 플랫폼(안드로이드, 아이폰)으로 변환 할 수 있는 기반을 만든다. 이를 위해서 어플리케이션의 구조를 분석하고 독립적인 특성과 종속적인 특성들로 분류화 한다. 분류된 독립 모델들은 모델 변환 규칙을 통해서 종속 모델로 변환 가능하다. 본 논문에서는 윈도우 모바일 플랫폼에서 하나의 독립 모델에서 하나의 종속 모델로 변환하기 위한 변환 규칙들을 제안한다. 모델 변환 규칙은 ATL (ATLAS Transformation Language)으로 기술하고 변환 과정에 사용되는 타겟 독립 모델(TIM)은 플랫폼에 의존적이지 않은 모델을 의미하고 타겟 종속 모델(TSM)은 플랫폼에 의존적인 것을 말한다[2].

적용사례로 윈도우즈 모바일 소프트웨어를 UML을 클래스 다이어그램으로 독립모델을 만들고 모델변환을 수행한다. 모델 변환 언어는 ATL으로 기술되었기 때문에 이클립스 M2M[6]에서 모델 변환을 수행한다. 이클립스 M2M(Model to Model)은 모델 변환을 수행하기 위한 메타 모델 체계(Eclipse Modeling Framework), 모델 변환 언어(ATL), UML을 포함하는 통합 개발 환경이다. 윈도우 모바일 소프트웨어의 독립 모델이 종속 모델로 자동 변환 수행을 통해서 모바일 소프트웨어 개발에 적용할 수 있음을 확인할 수 있었다. 향후 연구로 제안한 방법을 아이폰, 안드로이드에 적용할 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 관련된 연구로서 기존의 MDD의 기본 개념과 모델변환 방법 및 ATL대해서 기술한다. 3장에서는 모델 변환을 수행하기 위해서 윈도우 모바일 어플리케이션 분석과 모델 변환을 위한 규칙에 대해서 언급한다. 4장에서는 적용사례로 윈도우 모바일 어플리케이션을 개발을 위해서 타겟 독립 모델을 만들고 이를 모델변환 수행과정에 대해서 다룬다. 마지막으로 5장에서는 결론 및 향후 연구를 언급한다.

2. 관련 연구

2.1 MDD(Model Driven Development)

개발의 중심이 Code가 아닌 아키텍쳐, 분석, 설계로 옮겨감에 따라 반복적이고 점증적인 부분의 자동화 필요성이 증가하였다. 이러한 개발 기간, 품질 등의 문제를 해결하기 위한 방법으로 MDD[7]가 등장하였다.

MDD는 설계 모델을 실행 가능한 시스템으로 변환하고 반복적으로 재정의되는 모델을 재사용하는 점증적인 소프트웨어 개발 프로세스이다. 이때 모델변환은 부분적/전체적으로 자동화된다. 기존의 UML을 통한 개발은 개발자가 수행 시간에 프로그램 코드를 변경했을 때, 모델로의 변경에 문제가 생길 수 있었다. 이런 문제를 MDD에서는 자동화 도구를 이용하여 해결한다[5].

MDD의 이점은 다음과 같다. 첫째, 모델 기반의 개발에서 모델을 사용하여 표현된 내용은 모든 관련된 모델 뷰를 통해 추적성을 갖는다. 둘째, 모델의 점진적인 변화를 통해 비즈니스, 소프트웨어와 시스템 분석가들에게 의미있는 모델로 재정의 가능하다. 마지막으로 모델이 만들려는 시스템을 시뮬레이션 할 수 있다. 이 시뮬레이션은 확인과 검증을 자동화하여 비용 효과를 제공한다.

2.2 모델 변환 방법

모델 변환은 입력되는 소스 모델을 대상 모델로 변환하는 방법이다. 모델 변환의 종류에는 모델에서 모델, 모델에서 코드, 코드에서 모델 등이 있다. 또한 모델 변환에서 사용되는 모델은 UML 뿐만 아니라 컨트롤 플로우 다이어그램, 데이터 플로우 다이어그램 등 다양한 모델로도 변환이 가능하다. UML은 UML 메타모델이 지원이 되기 때문에 메타모델을 만들지 않아도 되지만 메타모델이 없는 모델일 경우 MOF(Meta Object Facility)[8]를 이용하여 메타모델로 설계가 요구된다.

모델 변환을 수행하기 위해서는 소스 모델, 소스 메타모델, 타겟 메타모델, 변형 정의가 필요하다[9]. 모델 변환 엔진은 이러한 요소들을 사용해서 타겟 모델로 변환한다. 모델 변환에서 중요한 요소로는 변환 룰을 정의하는 언어에 있다.

2.3 ATL(ATLAS Transformation Language)

ATL은 ATLAS INRIA & LINA 연구 단체가 개발한 모델 변환 언어이다[10]. 이것은 메타모델 및 모델로 지정된 모델 변환 언어이다. ATL은 개발자들에게 원본 모델에서 다수 대상 모델로 일으키는 방법을 지정하는 방법 제공한다. ATL 언어는 서술 적이고 긴급한 프로그램 언어의 일부이다. 변환 언어를 표현하는데 선호하는 방법은 서술적인 것이다. 그것은 단순히 원본과 대상 모델 성분 사이에 매핑을 표현하는 것을 가능하게 한다. 또한, ATL은 간단하게 서술적으로 표현될 수 없는 복잡한 명세를 위하여 OCL(Object Constraint Language)[11]을 사용한다.

ATL이 정의하는 규칙은 원본 모델의 성분이 어떻게 일치하고 대상 모델의 성분을 창조하고 초기화하기 위한 방법으로 구성된다. 기본적인 모델 변환 외에, ATL는 모델에 요구를 지정하는 것을 가능하게 하는 질의문을 정의한다. 또한 ATL은 ATL 라이브러리의 정의

를 통해 모듈화를 허용한다.

ATL 통합 개발 환경(IDE)은 다수 표준 ATL 변환의 디자인을 쉽게 하기 위하여 개발 도구를 제공한다. ATL 개발 환경은 또한 모델과 메타모델의 취급에 다수 추가 기능을 제안한다. 이 특징은 메타모델의 명세에 전념한 간단한 본문 표기법, 또한 일반적인 본문 통어론과 그들의 대응 모형 사이에 다수 기준 교량을 포함한다.

3. 모델 변환을 이용한 윈도우 모바일 어플리케이션 개발

3.1 어플리케이션 분석

스마트폰의 소프트웨어 구성은 하드웨어 처리나 로직을 처리하는 소스와 화면의 UI나 그림파일등의 정보를 가지고 있는 리소스로 구성된다. 대부분의 플랫폼들이 빠른 화면의 UI구성을 위해서 XML기반의 UI 편집 언어를 가지고 있다. 이러한 형식의 소프트웨어는 MVC 아키텍처로 만들어 낼 수 있다. 대부분의 스마트폰의 소프트웨어가 이를 적용한 것은 아니지만 Apple의 iPhone은 이를 철저하게 지킨다. MVC는 소프트웨어를 모델, 뷰, 컨트롤부분으로 분리하여 개발하는 방식으로 유지보수 및 이식성을 높일 수 있다.

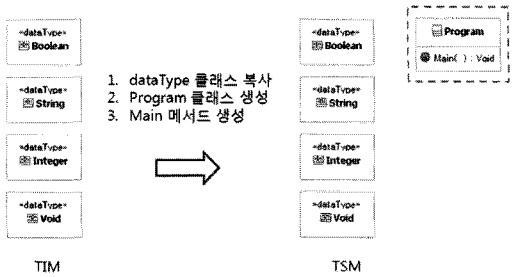
윈도우 모바일 플랫폼도 MVC를 지원하지 않지만 UI와 프로그램 코드를 분리되어있다. 윈도우 모바일의 어플리케이션의 구조를 살펴보면, UI를 생성하는 Form. Desiner. cs와 이를 제어하는 Form.cs로 구분되어 있다. Visual Studio가 제공하는 UI 편집 도구를 사용하여 Button, Panel, Edit Text 등을 만들면 Form.Desiner.cs에 자동으로 UI 코드가 생성된다. 각 UI 컴포넌트가 발생될 이벤트를 처리해주는 핸들을 등록하고 Form.cs에서 구체적인 코드를 작성하는 형식으로 프로그래밍 한다. 이러한 어플리케이션의 구조를 통해서 타겟 독립 모델을 View와 Controller의 형태로 구분하여 설계를 해야 한다는 것을 알아냈다. 그래서 본 논문에서는 적용사례의 타겟 독립 모델을 설계에서 View와 Controller를 분리하였다.

3.2 ATL을 이용한 모델변환 규칙 작성

본 논문에서는 독립 모델을 종속 모델로 변환하기 위해서 4가지 모델 변환 규칙을 사용한다. 모델 변환을 수행하기 위해서는 대상 모델의 마크가 필요하다. 이때 사용한 것은 UML의 스테레오 타입이다. 사용한 스테레오 타입은 클래스에서는 <<view>>, <<controller>> 메서드에는 <<view_ondDraw>>, <<button_onClick>>를 사용하였다. 이러한 이유로 모델 변환을 규칙은 기본 구조 생성, view 생성, view_onDraw 생성, controller 생성, button_onClick 생성 총 5가지 형태로 구성된다.

3.2.1 규칙 1 : 기본 구조 생성

기본 구조의 생성은 윈도우 모바일 어플리케이션을 실행하기에 필요한 기본 요소를 말한다. 여기에는 프로그램이 요구하는 구조와 UML이 요구하는 데이터 타입이 있다. 모델 변환 규칙은 첫 번째로 UML이 필요한 데이터 타입 생성을 위해 dataType 클래스를 복사한다. 그 다음 윈도우 모바일 수행을 위해 필요한 Program 클래스를 생성한 후 이 클래스에 Main메서드를 생성한다. 이러한 과정은 그림 1에 자세히 설명하였다.




```
rule ModelCreation
{
    from
        o : TIM!"uml::Package"
    to
        m : WindowsMobile!"uml::Package"{
            name <- 'WindowsMobileTSM',
            packagedElement <- Sequence{o.packagedElement, ProgramClass}
        }, ProgramClass : WindowsMobile!"uml::Class" {
            name <- 'Program', ownedOperation <- mOperation
        }, mOperation : WindowsMobile!"uml::Operation" {
            name <- 'Main', upper <- 1, visibility <- #public,
            isStatic <- true, ownedParameter <- mPara
        }, mPara : WindowsMobile!"uml::Parameter" {
            direction <- 'return', type <- thisModule.getRefDataType('Void')
        }
}
```

그림 1 기본 구조 생성

3.2.2 규칙 2 : view 생성

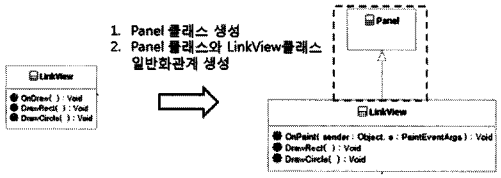
윈도우 모바일에서 화면에 그림을 그리기 위해서는 Panel 클래스가 요구된다. 그렇기 때문에 스테레오 타입 <<View>>에 매치된다면, 그림 2와 같이 먼저 Panel 클래스를 생성하고 그 다음 Panel 클래스와 LinkView 클래스 간에 일반화관계를 생성한다. Panel클래스를 생성하는 이유는 Panel이 윈도우 모바일에서는 View 역할을 해주기 때문이다.

3.2.3 규칙 3 : view_onDraw 생성

Panel 클래스만 추가되었다고 화면에 그림을 그릴수는 없다 그렇기 때문에 Panel클래스에 정의되어 있는 OnPaint 메서드를 오버라이딩해야 한다. 또한 파라메터로 사용되는 Object 클래스와 PaintEventArgs 클래스를 추가해야한다. 이러한 종속적인 특징을 실행하기 위한 순서는 Object 클래스 생성, PaintEventArgs 클래스 생성, OnDraw 메서드를 OnPaint 메서드로 변경, OnPaint 메서드에 파라메터 추가 순으로 수행한다. 이러한 과정을 그림 3에 자세히 설명하였다.

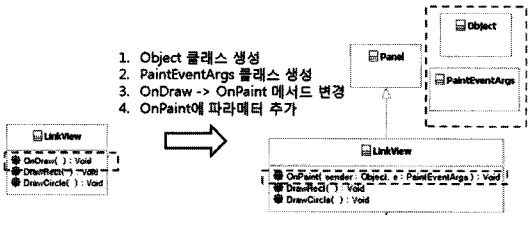
3.2.4 규칙 4 : controller 생성

<<view>>는 화면을 보여주는 것이라면 <<controller>>



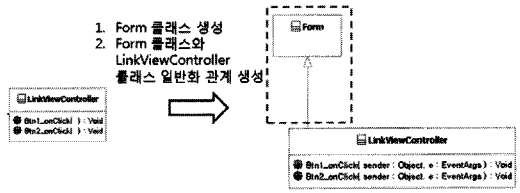

```
rule ClassView {
    from
        o : TIM!"uml::Class"( o.KeyClassValueCount('view') > 0 )
    to
        mm : WindowsMobile!"uml::Class"(
            name <- o.name, generalization <- generalPanel,
            ownedAttribute <- o.ownedAttribute,
            ownedOperation <- o.ownedOperation
        ),
        generalPanel : WindowsMobile!"uml::Generalization" (
            general <- thisModule.getRef('Panel')
        )
}
```

그림 2 view 생성



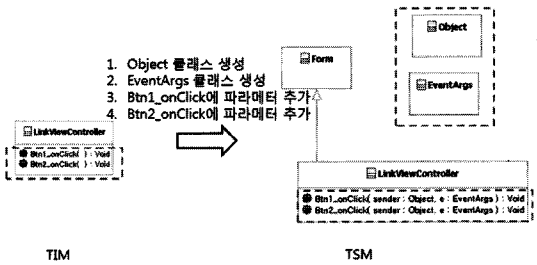

```
rule MethodOndraw {
    from
        o : TIM!"uml::Operation"( o.KeyOperValueCount('view_onDraw') > 0 )
    to
        mm : WindowsMobile!"uml::Operation" (
            name <- 'OnPaint', visibility <- o.visibility, upper <- o.upper,
            ownedParameter <- Sequence{return, par1, par2}
        ), return : WindowsMobile!"uml::Parameter" (
            direction <- 'return', type <- thisModule.getRefDataType('Void')
        ), par1 : WindowsMobile!"uml::Parameter" (
            name <- 'sender', type <- thisModule.getRef('Object')
        ), par2 : WindowsMobile!"uml::Parameter" (
            name <- 'e', type <- thisModule.getRef('PaintEventArgs') )
}
```

그림 3 view_onDraw 생성




```
rule ClassViewController {
    from
        o : TIM!"uml::Class"( o.KeyClassValueCount('controller') > 0 )
    to
        mm : WindowsMobile!"uml::Class"(
            name <- o.name, generalization <- generalForm,
            ownedAttribute <- o.ownedAttribute,
            ownedOperation <- o.ownedOperation
        ), generalForm : WindowsMobile!"uml::Generalization" (
            general <- thisModule.getRef('Form') )
}
```

그림 4 controller 생성




```
rule MethodBtnClick {
    from
        o : TIM!"uml::Operation"( o.KeyOperValueCount('button_onClick') > 0 )
    to
        mm : WindowsMobile!"uml::Operation" (
            name <- o.name, visibility <- o.visibility, upper <- o.upper,
            ownedParameter <- Sequence{return, par1, par2}
        ), return : WindowsMobile!"uml::Parameter" (
            direction <- 'return', type <- thisModule.getRefDataType('Void')
        ), par1 : WindowsMobile!"uml::Parameter" (
            name <- 'sender', type <- thisModule.getRef('Object')
        ), par2 : WindowsMobile!"uml::Parameter" (
            name <- 'e', type <- thisModule.getRef('EventArgs') )
}
```

그림 5 button_onClick 생성

은 어떠한 이벤트가 발생하였을 때 이를 처리해주는 클래스 이다. 윈도우 모바일에서는 이러한 처리를 해주기 위해서 Form 클래스가 필요하다. 그렇기 때문에 LinkViewController클래스는 Form클래스를 상속 받아야 한다. 생성되는 순서는 Form 클래스 생성, From 클래스와 LinkViewController클래스의 일반화 관계 생성 순이다. 그림 4에 자세히 설명하였다.

3.2.5 규칙 5: button_onClick 생성

Button이 눌렸을 때 어떤 행위를 수행하기 위해서는 Panel 클래스의 OnPaint의 메서드와 마찬가지로 이를 처리해줘야 하는 이벤트 핸들러가 필요하다. OnPaint의 함수는 미리 정해져 있는 형태이지만 버튼의 이벤트 핸들러의 메서드 이름은 사용자가 지정할 수 있다. 사용자 임의로 정의한 후 이 메서드를 Form클래스에 등록해주면 된다. 그렇기 때문에 독립 모델에서 정의한 메소드의 이름이 변경되지 않는다. 생성순서는 Object 클래스 생성, EventArgs 클래스 생성, 메서드에 파라메터 추가 순이다. 그림 5에 자세히 설명하였다.

4. 적용사례

모델 변환을 수행하기 위한 적용사례는 버튼을 누르면 웹브라우져를 열고 등록되어 있는 사이트로 이동하는 간단한 주소록 관리 프로그램이다. 이러한 프로그램을 만들기 위해서는 그림 6과 같이 윈도우 화면에 버튼이 필요하다.

4.1 타겟 독립 모델

타겟 독립 모델을 정의 하면 그림 6과 같이 할 수 있다. LinkView는 화면에 그림을 표현하는 클래스이다.

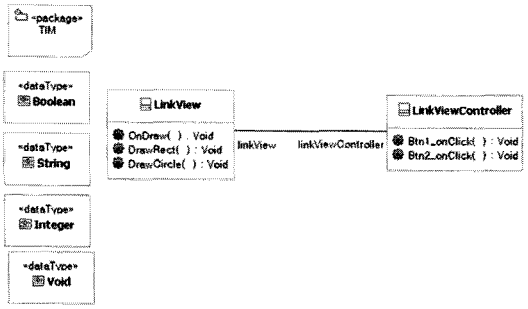


그림 6 타겟 독립 모델의 클래스 다이어그램

LinkViewController는 화면에서 어떠한 이벤트가 발생되었을 때 처리해주는 핸들러를 관리해주는 클래스 이다. 본 논문의 사례에서는 버튼 2개가 사용되었으므로 Btn1_onClick()과 Btn2_onClick()메서드가 컨트롤 클래스에 추가하였다.

4.2 타겟 종속 모델

타겟 독립 모델을 제안한 규칙 5개를 수행하면 그림 9와 같은 클래스 다이어그램을 생성할 수 있다. 그림 7의 숫자는 적용된 규칙 번호이다.

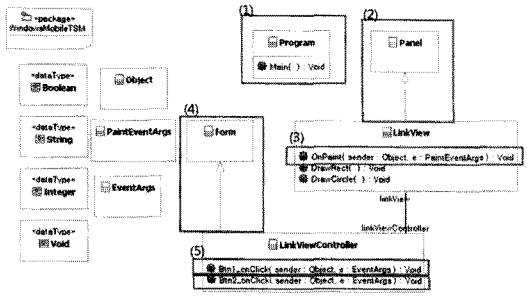


그림 7 타겟 종속 모델의 클래스 다이어그램

5. 결 론

본 논문에서는 이종의 스마트폰 개발을 위해서 윈도우 모바일 플랫폼 개발에 모델 변환 기법을 적용하였다. 모델 변환 기법을 적용하기 위해서 윈도우 모바일 플랫폼의 어플리케이션의 구조를 분석하였다. 분석한 결과를 통해서 독립적인 특성과 종속적인 특성들로 구분하고 이를 기반으로 모델 변환 규칙을 만들었다. 만들어진 규칙은 5개의 규칙이며 크게 구분하면 기본구조, 클래스 단위, 메서드 단위 3가지로 분류화 할 수 있다. 또한 모델 변환 규칙을 ATL을 사용하여 기술하였고 이것을 이클립스 환경에서 모델 변환을 수행하였다. 그 결과 모델 변환이 변환 규칙대로 클래스 다이어그램을 생성한 것을 확인할 수 있었다.

윈도우 모바일 환경의 적용사례를 통해서 생성된 타겟 독립 모델이 이종의 다른 플랫폼으로 변환될 수 있는 가능성을 보았다. 본 논문의 플랫폼 독립모델을 사용하고 모델 변환 규칙을 재정의 하면 아이폰, 안드로이드 등의 이종의 플랫폼으로 변환이 가능하다.

향후연구로 아이폰, 안드로이드 플랫폼을 분석하고 본 논문에 적용한 방법을 각 플랫폼에 맞게 확장할 것이다.

참 고 문 헌

[1] Axel Jantsch, (2004), Modeling Embedded System and SOCs, Mogan Kaufmann.

[2] W. Y. Kim, H. S. Son, Y. B. Park, B. H. Park, C. R. Carlson, R. Y. C. Kim, "The Automatic MDA Transformations for Heterogeneous Embedded Systems," *2008 SERP*, vol.2, pp.409-414, 2008.

[3] W. Y. Kim, R. Y. C. Kim, "A Study on Modeling Heterogeneous Embedded S/W Components based on Model Driven Architecture with Extended xUML," *KIPS*, vol.14-D, no.1, 2007.

[4] 손현승, 김우열, 서채연, 김동호, 김동우, 김재수, 김영철, "이종 임베디드 소프트웨어를 위한 코드 생성 메커니즘 및 지원도구", *2007 KCSE*, vol.9, no.1, pp. 170-177, 2007.

[5] W. Y. Kim, H. S. Son, R. Y. C. Kim, C. R. Carlson, "MDD based CASE Tool for Modeling Heterogeneous Multi-Jointed Robots," *CSIE 2009*, vol.7, pp.775-779, 2009.

[6] Eclipse M2M, http://www.eclipse.org/m2m/

[7] Selic, B, "The Pragmatics of Model-Driven Development," *IEEE Software special issue on Model-Driven Architecture*, 2003.

[8] OMG, Meta Object Facility Specification, In OMG Unified Modeling Language Specification, Version 2.0, January 2006.

[9] K. Czarnecki, S. Helsen, "Feature-Based Survey of Model Transformation Approaches, IBM Systems Journal," vol.45, no.3, pp.621-64, 2006.

[10] Wikipedia, "ATLAS Transformation Language," http://en.wikipedia.org/wiki/ATLAS_Transformation_Language

[11] OMG, Object Constraint Language Specification, In OMG Unified Modeling Language Specification, Version 2.0, May 2006.