WIPI 기반 모바일 애플리케이션 개발을 위한 모델변환에 관한 연구

고종원*, 송영재**
*경희대학교 컴퓨터공학과
**경희대학교 컴퓨터공학과
e-mail : jwko@khu.ac.kr

A Study on Model Transformation for Mobile Application Development on WIPI platform

Jong-Won Ko*, Young-Jae Song**
*Dept. of Computer Engineering, Kyung-Hee University
**Dept. of Computer Engineering, Kyung-Hee University

요 약

국내 표준무선인터넷플랫폼인 WIPI 에 대한 확장성 연구는 미들웨어 차원에서 여러 연구로 진행되었지만 이 논문에서는 애플리케이션 레벨에서 MDA 에서의 모델변환을 적용, 다른 플랫폼에서의 애플리케이션을 쉽게 WIPI 플랫폼으로 포팅할 수 있는 방법을 제안한다. 도메인 모델의 식별과 메타모델 정의, 그리고 변환규칙 등을 정의하여 XML 및 XSLT 를 이용한 모델변환을 수행한다. 또한 서로 다른 개발 플랫폼에서의 API 매핑과 WIPI 기반 도메인 제약조건 등을 인식하고 향후 제한적이지만 제안사항을 적용한 모델변환도구의 개발과 테스트 기능 지원으로 연구를 진행한다..

1. 서론

각 이동통신사별로 다양한 플랫폼과 개발환경에서 개발되던 형태의 모바일 애플리케이션을 WIPI 라는 표준 모바일 플랫폼을 개발함으로써 모바일 애플리케이션 개발 프로세스의 절감효과를 가져오긴 했지만 국내표준으로서의 WIPI 가 가지고 있는 확장성과 표준성에 대한 한계점 역시 앞으로 해결해야 할 과제로서 주목 받고 있다. 지난 2004 년 2 월 WIPI 2.0 이 발표된 이후로 현재까지 실제 개발업무에서의 WIPI 플랫폼 기반의 애플리케이션 개발에 있어서 WIPI 에 대한 위기론까지 대두되고 있지만 WIPI 서비스와 관련된 이동통신사나 단말기 제조사 및 컨텐츠 개발사로부터 다양한 컨텐츠 개발과 폭넓은 사용자층의 확대를 위해 개발비용을 절감할 수 있다는 공통된 요구를 만족할 수 있는 기술로 인식되고 있다. 하지만 앞서 이야기한 확장성에 대한 문제와 관련해서 현재의 WIPI 는 국내 핸드폰 OS 로 사용되는 퀄컴의 RexOS 에 국한되어 있기 때문에 해외의 Nokia 나 SonyErricson 에서의 SymbianOS 또는 WindowsCE 와 같은 핸드폰에서 사용되는 운영체제나 PDA 및 기타 임베디드 디바이스에 탑재되는 운영체제는 지원하지 못한다. 따라서 본 논문에서는 WIPI 플랫폼 확장을 위해 WIPI SDK 에 MDA 기반의 모델변환엔진을 적용함으로써 기존 CE.NET 으로 개발된 모바일 애플리케이션을 WIPI-JAVA 로 손쉽게 포팅할 수 있는 방법을 제안한다. 이 연구의 목적은 기존의 미들웨어 차원에서 진행되던 코어 API 나 HAL 에 대한 추상화 연구가 아닌 애플리케이션 레벨에서의 모델변환 및 소스 자동생성을 통해 다른 플랫폼에서 개발된 애플리케이션을 WIPI 플랫폼에서 실행가능케 함으로써 WIPI 의 확장성을 제공하는데 있다.

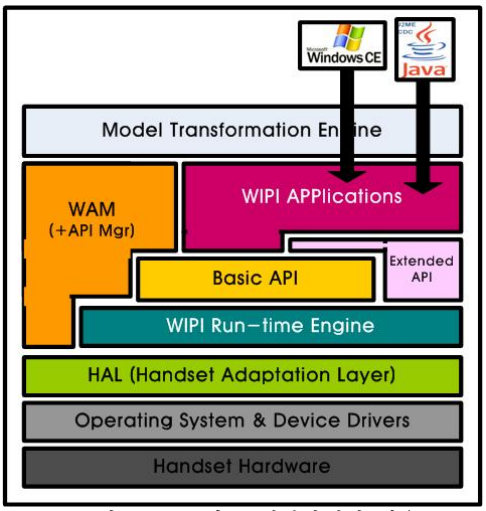
본 논문의 구성은 2 장에서는 WIPI 에 대한 소개 및 모델변환도구에 사용되는 모델변환 어프로치에 대한 여러 형태에 대해서 살펴보고 3 장에서는 제안하는 모델변환 워크플로우의 전체적인 개념과 모바일 애플리케이션의 도메인 모델을 정의하는 메타모델, 그리고 변환규칙에 대해서 소개한다. 4 장에서는 OMG 에서의 모델변환도구 평가항목을 기준으로 한 다른 모델변환도구와 비교평가를 제시하고 결론 및 향후 연구방향을 5 장에 기술하였다.

2. 관련연구

2.1 WIPI(Wireless Internet Platform for Interoperability)

한국무선인터넷표준화포럼(KWISF)의 무선인터넷플랫폼표준이며, 한국정보통신기술협회(TTA)에 의해 TTA 단체표준 TTAS.KO-06.0036 으로 채택된 이동통신 단말기용 응용프로그램의 실행환경에 대한 표준규격이다. 각 이동통신사별로 다양한 플랫폼과 개발환경에서 개발되던 형태의 무선플랫폼을 각 통신사기술과 TTA 및 ETRI 에서 표준플랫폼 WIPI 를 개발하였다. 아래 (그림 1)은 기존의 WIPI 플랫폼에 논문에서의 모델변환엔진을 두었을 때 이루어지는 애플리케

이션 레벨에서의 변환 및 포팅을 보여주는 그림이다.



(그림 1) WIPI 와 모델변환엔진 적용

2.2 모델변환어프로치

최근 2 년 사이에 다양한 형태로 진행되고 있는 모델변환 어프로치들은 기존의 대표적인 모델변환 형태인 UML 모델로부터 코드생성으로부터 벗어나 다양한 형태의 모델에서 모델로의 변환연구가 활발히 진행되고 있다. 다양한 모델변환 어프로치를 살펴보기 전에 먼저, 모델변환에 필요한 요소들을 정의하면 아래와 같다.

1. 메타모델(Metamodel) 정의: 메타모델을 이용한 모델변환에서 필요한 소스모델 및 목표모델에 대한 메타모델 요소 정의
2. 메타언어(MOF) : 메타모델을 정의하기 위한 언어
3. 변환정의언어(Transformation Definition Language) : XSLT 나 ATL 과 같은 변환규칙을 정의하는 언어. 모델변환 어프로치는 크게 Model to Code 어프로치와 Model to Model 어프로치 두 가지로 구분된다.

1) Model to Code 어프로치

소스모델로부터 코드생성이 가능한 변환 메카니즘으로 기존의 변환도구들이 대표적으로 지원하던 UML 모델로부터 자바나 C++와 같은 소스생성 기능을 제공한다. 메타언어를 정의하기 위한 언어인 MOF 를 통한 새로운 메타모델 정의를 하지 않고 이미 정의된 모델요소 타입을 표현하기 위해 JAVA 클래스로 정의하며, 새로운 모델요소 타입이 요구될 때, 이 JAVA 클래스를 서브클래싱해서 UML 모델을 나타내는 자바클래스로 모델변환을 하는 Visitor-based Approach 와 Template 이라고 부르는 소스모델의 내부 폼에 사용자가 어떠한 필드나 조건 등을 입력함으로써 이에 대한 이벤트 API 함수가 소스코드를 생성하는 Logic 을 수행하는 Template-based Approach 가 있다.

2) Model to Model 어프로치

같은 메타모델이나 아니면 다른 메타모델로부터 정의되는 소스모델과 목표모델 사이에서의 모델변환 어프로치이다. 대부분의 현재 존재하는 MDA 도구는 단순히 Model to Code 변환, 즉 PIM 모델로부터 PSM 모델(구현코드)로의 변환만 지원하며 모델 간의 변환이 필요한 가장 큰 이유는 PIM 모델과 PSM 모델간의 추상화 차이가 클 때 Target PSM 모델을 곧바로 생성하는 것보다 중간에 Intermediate Model 을 생성하는 것이 보다 쉽게 PIM 모델과 PSM 모델간의 연결이 수월해진다.

가. Direct-manipulation Approach
: 내부 모델표현과 이를 제어하는 API 를 포함하고, Object-oriented framework 처럼 보통 구현이 되며 모델변환을 지원하기 위한 추상클래스의 구현을 통해 최소한의 인프라스트럭처를 지원한다. 이 접근방법은 사용자가 직접 java 와 같은 프로그래밍 언어를 통해서 변환규칙을 구현한다.

나. Relational Approach
: 소스모델과 목표모델 요소간의 Relation 을 Constraints 을 이용해서 명세한다. 이러한 명세는 mapping rule 을 포함하며, 바로 실행 및 생성을 지원하는 게 아니라 mapping rule 에 대한 수학적인 접근이 주된 아이디어이다.

다. Graph-transformation based Approach
: 소스모델과 목표모델의 요소를 정의하는 그래프를 생성하며 UML 과 유사한 일종의 그래프 모델을 PSM 모델로 변환하는 사례가 프로세스 그래프와 플로우 그래프를 이용해서 그래프 변환을 수행한다.

라. Structure-driven Approach
: 소스모델과 목표모델의 메타모델 정의에 있어서 모델요소에 대한 Attributes 들과 Reference set, 즉 변환에 대응하는 API 함수 등에 대한 계층구조로 목표모델을 생성하여 모델변환을 수행한다.

2.3 OMG 의 모델변환도구 평가항목

MDA 를 지원하는 여러 UML 변환도구에 대한 평가는 OMG MOF 2.0 QVT 에서 제공하는 변환도구들에 대한 평가항목은 보면 아래와 같다.

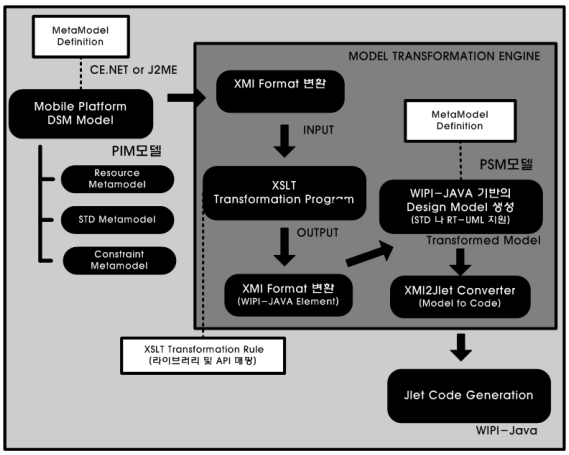
1. Self containing : 어떻게 변환이 이루어지는가에 대한 이해 정도가 충분하게 도큐먼트가 지원되는가?
2. Scalability : 변환규칙에 대한 확장성
3. Simplicity : 변환규칙이 이해하기 쉽고 명확한가?
4. Bi-directional mapping
   : 양방향 모델변환 매핑을 지원하나?
5. Ease of Adoption
   : 제공하는 변환 메커니즘이 적용하기 쉬운가?
6. Provide an abstract syntax for the transformation language : 변환규칙 정의언어

가 어느 정도의 추상화 레벨을 제공하나?
7. Support composition and reuse
: 변환모델에서의 재사용 및 합성 지원여부
8. Provide complete example
: 변환도구에서 사용예제 지원여부
9. Support complex transformation scenarios
: 복잡한 변환규칙 시나리오에 대한 지원여부

3. 본론

3.1 WIPI 기반 애플리케이션 모델변환 워크플로우

모바일 애플리케이션을 모델변환을 통해서 WIPI-Java 로의 포팅하기 위해서는 MDA 에서의 모델기반 개발방법을 바탕으로 모바일 애플리케이션의 도메인 모델로부터 WIPI-Java 애플리케이션으로의 모델변환을 필요로 하는데 그 과정을 정리하면 아래 (그림 2)과 같다.

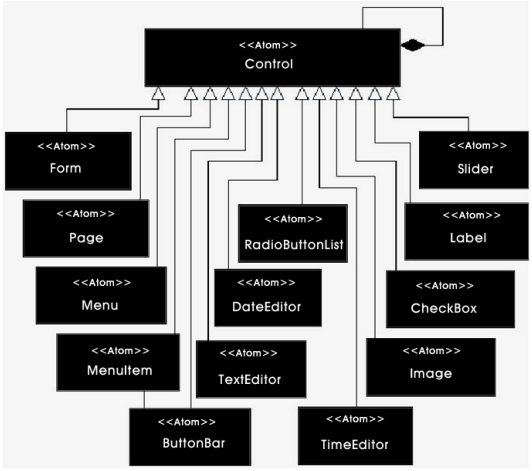


(그림 2) 모델변환 워크플로우

(그림 2) 에서처럼 전체적인 모델변환 워크플로우는 모바일 애플리케이션에 해당하는 도메인 모델을 크게 Resource 모델, State Transition Diagram 인 STD 모델, Constraint 모델 이상 세 가지 모델로 구분하고 이에 대한 메타모델을 통해 각 모델들을 구성할 모델요소들을 정의한다. 즉, MDA 에서 플랫폼 독립적인 PIM 모델로 나타낼 수 있는 모바일 애플리케이션에 대한 도메인모델을 생성한다. 생성된 도메인 모델을 XMI 포맷으로 변환해서 XSLT 로 정의된 변환규칙에 의해서 WIPI-Java 기반의 XMI 포맷으로 변환된다. 변환된 XMI 포맷에 의해서 WIPI-Java 에서의 메타모델로 정의된 요소에 의해 PSM 모델 설계를 하기 위한 Jlet 애플리케이션 설계모델을 생성하고, 변환된 XMI 코드에서 WIPI-Java 코드변환을 통해서 Jlet 코드를 자동생성한다

3.2 모바일 애플리케이션 도메인 메타 모델 정의

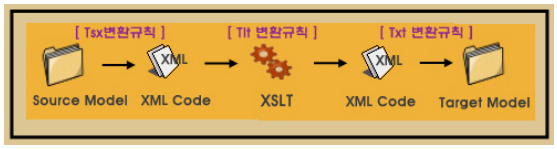
(그림 2)에서의 PIM 모델로 정의할 수 있는 모바일 플랫폼 도메인 모델은 모델변환 시에 메타모델로 정의가 되며, 메타모델로 정의된 형태의 도메인모델을 WIPI 기반의 Jlet 모델로 변환코자 변환규칙 등을 적용한다. 이러한 도메인 모델은 모바일 애플리케이션의 구성요소가 되는 리소스 모델과 시간속성과 상태속성을 가지고 메시지 호출 등의 행동명세를 나타내는 STD 모델, 그리고 제약조건 등을 기술하여 변환규칙에 적용하는 제약조건(Constraint)모델의 세 부분으로 정의하고 아래 (그림 3)은 그 중 리소스 메타모델을 나타내는 그림이다.



(그림 3) 모바일 애플리케이션 리소스 메타 모델

3.3 모델변환규칙 정의

메타모델 정의를 통해 모델변환에서의 입력모델을 생성할 수 있고 생성된 입력모델, 즉 모바일 애플리케이션 도메인 모델을 다시 XML 코드로 변환하는 모델 to 코드 변환과 XSLT 를 이용한 목표모델로의 XML 코드 변환이 이루어진다. 아래 (그림 4)에서는 이러한 모델변환규칙의 구조를 설명하고 있다.



(그림 4) 모델변환규칙의 구조

모델변환에서 요구하는 모델변환규칙은 아래와 같이 크게 세 단계를 가진다. 즉, 모델변환을 수행하는 변환엔진 t 는 아래의 변환명세 T 를 가지며, 변환명세 T 는 <Tsx, Tlt, Txt>의 세 요소로 정의된다.

- Tsx : 입력모델에서 XML 코드 생성 변환규칙
- Tlt : XML 코드와 XML 코드 간의 XSLT 를 통한 변환규칙
- Txt : XML 코드에서 목표모델 생성 변환규칙

이 중 지면상 입력모델에서 XML 코드로 생성되는 변환규칙을 정의하는 Tsx 변환을 보면,

입력모델 Ms = {Mresource,Mstd,Mcont} 로 구성된다. 그 중에서 리소스메타모델의 구성요소는

Mresource={Menu,Image,Buttonbar,Form,Label,CheckBox, RadioButtonList,MenuItem…}의 컨트롤로 정의되며, 각 메타요소에 대한 XML 정의를 통해 모델에서 요소가 생성될 때마다 소스코드가 생성되는 형태처럼 XML 코드가 생성된다. 생성된 XML 코드는 위의 (그림 4)에서처럼 목표모델을 생성하기 위한 XML 코드로 다시 XSLT 를 이용한 변환이 수행된다.

3.4 변환모델에 대한 비교 알고리즘

위의 절차대로 진행된 모델변환을 통해서 생성된 목표모델과 입력모델 간의 비교알고리즘을 통해서 매핑된 메타모델 요소간의 비교와 또한 차이점에 대한 내용을 아래와 같이 정리할 수 있다.

```
Input: M1, M2
Output: MS, DS {N, D, C}

Begin
1. For each vertex v^1 in M1
      If there is a vertex v^2 in M2 mapped to v^1
         Mark v^1 and v^2 mapped
         Add pair (v^1, v^2) to MS
         For each attribute f of v^1 and v^2,
            If the value of v^1 (val^1) does not equals to
            the value of v^2 (val^2)
               Add (v^1, v^2, f, val^1, val^2) to C
      Else
         Add v^1 to N
2. For each edge e^1 in M1
      If there is a edge e^2 in M2 mapped to e^1
         Mark e^1 and e^2 mapped
         Add pair (e^1, e^2) to MS
      Else
         Add e^1 to N
3. For those elements in M2 without mapped mark
      Add them to D
End
```

위의 알고리즘을 살펴보면, 입력모델인 M1 과 변환된 목표모델인 M2 를 그래프모델의 형태로 모델 간의 요소를 vertax, 그 요소들 간의 관계를 edge 로 정의해서 M1 과 M2 사이에 서로 매핑되는 vertax 가 있으면 매핑셋인 MS 에 (v1,v2)로 추가하고 해당 vertax 의 속성은 f 로 정의해서 서로 매핑되지 못한 요소들은 Difference Set 으로 구성한다. Edge 에 대해서도 입력모델인 M1 과 M2 에 대해서 서로 비교해서 MS 와 DS 를 구성요소로 추가시킴으로 두 모델에 대한 비교를 수행할 수 있다.

4. 결론 및 향후 연구방향

모바일 애플리케이션과 같은 임베디드 시스템을 도메인으로 하고 있는 분야는 특정 도메인에 제한적이라는 점 때문에 재사용이 어려운 분야이면서도 그렇기 때문에 재사용이 요구되기도 한다. 이 논문은 모델변환을 이용해서 WIPI 플랫폼의 확장성을 기존의 연구인 HAL(Handset Adaptation Layer)과 같은 미들웨어의 추상화를 통한 연구에 비교해서 애플리케이션 레벨에서의 모델변환을 통해서 다른 플랫폼의 애플리케이션을 WIPI 기반 플랫폼에 쉽게 포팅할 수 있는 방법에 대해서 제안한다. 하지만 애플리케이션 모델 변환을 통한 포팅을 위해서는 서로 대응하는 API 의 완벽한 매핑이 사실상 어려움으로써 별도로 추가적인 코드보완작업이 뒷받침되어야 하며, 사실상 이런 점이 WIPI 플랫폼이 표준모바일플랫폼으로써 애플리케이션 개발자에게 충분히 자리잡지 못하는 부분이기도 하다. 또한 이 논문에서는 특정 도메인에 맞는 도메인모델에서만 모델변환이 적용된다는 점 역시 제한적이다. 이를 개선하기 위해 향후 제한적인 도메인모델에 대한 확장 메카니즘과 생성된 모델을 비교하고 비교된 내용을 바탕으로 요구분석 및 설계 단계에서 테스트를 적용할 수 있는 테스트 기반 모델변환 연구를 진행하고 이에 대한 지원도구를 개발하기 위한 연구가 계속되어야 한다. .

참고문헌

[1] Krzytof Czarnecki and Simon Helsen, “Classification of Model Transformation Approaches”,OOPSLA’03

[2] Wensheng Wang, “Wein Evaluation of UML Model Transformation Tools”,2005

[3] Jon Oldevik, Arnor Solberg, Brian Elvesater, Arne J.Berre,” Framework for Model Transformation and Code Generation”, EDOC’02

[4] MOF 2.0 Query/View/Transformation RFP, OMG,2002

[5] Yuehua Lin,Jing, Jing Zhang and Jeff Gray, “A Testing Framework for Model Transformation”, 2005

[6] Pedro J.Clemente, Juan Hernandez, Fernando Sanchez, “An MDA Approach to Develop Systems Based on Components and Aspects”,SAC’07

[7] Gabor Karsal, Janos Sztipanovits, Akos Ledeczi,Ted Bapty, “Model-Integrated Development of Embedded Software”, Proceeding of the IEEE Vol.91,No1,2003

[8] 고종원,한정수,송영재, “모델 재사용을 위한 모델변환접근방법”,정보처리학회지, 2006