모델 기반 개발방법에 기반한 임베디드 소프트웨어의 역공학

나동진, 이용순, 김희진, 유민수
한양대학교 전자컴퓨터통신공학과
e-mail : {djna, yslee, hjkim, msryu}@rtcc.hanyang.ac.kr

Reverse Engineering of Embedded Software based on Model-Driven Development

DongJin Na, Yongsoon Lee, Heejin Kim, Minsoo Ryu
College of Information and Communications, Hanyang University

요 약

모델 기반 개발방법은 개발자가 추상화된 모델만을 설계하는 것만으로도 소프트웨어를 개발할 수 있도록 하는 방법이다. 현재까지의 모델 기반 개발방법론은 모델에서 코드를 변환하는 것은 다루고 있지만, 반대로 코드에서 모델로의 변환은 고려하고 있지 않다. 본 논문에서는 모델이 아닌 기존에 작성된 C 언어 코드를 모델로 변환하는 역공학 기법을 제안한다. 이러한 역공학 기법을 사용하면, 새로운 모델을 작성할 때 기존의 코드로부터 모델을 얻어내 적용할 수 있다. 또한, 모델을 작성하고 작성된 모델을 통해 생성된 최종코드를 수정하였을 경우 역공학을 통해 모델과 수정한 코드를 일관성 있게 유지할 수 있다. 이를 지원하기 위해 C 언어를 UML 로 변환하는 방법 및 변환된 모델의 효율적인 구성을 위한 모델 재구성 방법을 제안한다.

1. 서론

최근 소프트웨어 생산성 문제에 대한 해결책으로서 모델 기반 개발방법(Model-Driven Software Development)이 주목받고 있다. 이는 소프트웨어 개발의 추상화 수준(abstraction level)을 높여 소요되는 비용과 시간을 크게 단축시킬 수 있는 방법론이다. 모델 기반 개발방법의 핵심은 개발자로 하여금 모델만을 설계하도록 하고, 모델 컴파일러를 통해 모델로부터 코드를 자동으로 생성하는 데 있다.

본 논문은 모델기반 방법론에 있어 추가적인 코드 재사용성과 유지 관리의 용이성을 제공하기 위한 역공학 기법을 다룬다. 여기서 역공학이란, 기존에 C 언어로 작성된 코드를 모델로 변환하는 것을 말한다. 최종 생성된 코드를 수작업으로 변경하는 경우 역공학을 통해 변경사항을 모델에 반영시킬 수 있으며, 기존에 사용되고 있던 C 언어코드를 역변환하여 모델을 얻어냄으로써 기존 코드의 재사용이 가능해지는 이점이 있다. 하지만, 대부분의 임베디드 시스템에서는 C 언어를 사용하는 반면 모델링을 위한 언어는 대부분 UML 을 사용하기 때문에 개념부터가 상이하다. 따라서, C 언어에서 UML 로 변환시키기 위한 방법이 필요하다. 본 논문에서는 역공학 과정을 여러 단계로 나누어 코드에서 모델로 변환하는 방법을 제공한다. 또한, 변환된 모델의 응집력 측정을 통해 이 모델의 효율성이 낮다면, 모델 재구성을 통해 이를 개선하는 방법을 제공한다.

본 논문은 다음과 같이 구성된다: Chapter 2 는 C-to-UML 변환 방법을 기술하고, Chapter 3 에서는 응집력을 이용한 클래스 재구성 방법을 설명한다. Chapter 4 에서는 클래스 재구성 방법을 적용한 역변환기 구현에 관한 내용을 다룬다. Chapter 5 에서는 논문의 요약과 앞으로 계획을 제시하고 논문을 결론짓는다.

1.1. 관련연구

Chidamber[7] 는 클래스의 메서드들의 이질성을 측정하기 위한 방법으로 LCOM(Lack of Cohesion in Methods)을 제안하였다. LCOM 은 공통으로 사용하는 인스턴스 변수가 없는 메서드쌍의 수로 정의 된다. LCOM 의 결과값이 낮으면 응집력이 높은 클래스를 의미한다. 응집력이 약한 클래스는 하나 또는 그 이상으로 나누어야 한다. 낮은 응집력은 복잡도를 증가시고 개발 중 에러를 증가 시킨다.

Bieman and Kang[8]은 클래스의 응집력을 측정하는 방법으로서 TCC(Tight Class Cohesion)과 LCC(Loose Class Cohesion)을 제안했다. 이 방법은 공통으로 사용하는 인스턴스 변수들을 이용하는 한 쌍의 메서드들을 이용한다. 인스턴스 변수들은 메서드에 의해 직접적 또는 간접적으로 사용될 수 있다. 여기서 직접적으로 사용된다는 의미는 메서드 Mi 의 내부에 인스턴스 변수가 있는 경우를 말하고, 반대로 간접적으로 사용된다는 의미는 메서드 Mj 의 내부에 인스턴스 변수가 있을 때 메서드 Mi 에 의해 간접적으로 호출된 경우를 말한다. TCC 는 직접적으로 연관있는 메서드들의 쌍으로 정의 될 수 있고 LCC 는 직

접적 또는 간접적으로 연관된 메서드 쌍의 수로서 정의 된다.

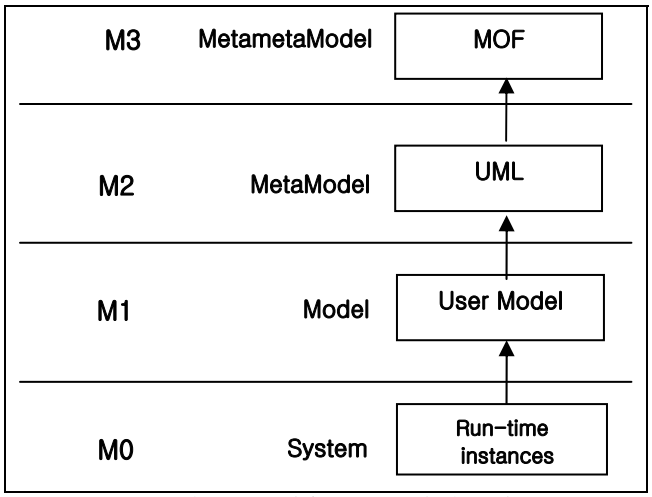
2. C-to-UML 변환 방법

2.1. 개요

C-to-UML 로 변환하기 위해서는 프로그래밍 언어인 C 언어를 모델링 언어인 UML 로 변환 시켜주어야 한다. 하지만, 모델링 언어인 UML 은 프로그래밍 언어인 C 언어와는 개념부터가 상이하기 때문에 C 언어와 UML 사이의 매핑을 통해 C-to-UML 변환을 지원하기 위한 매핑 규칙을 정의하였다.

2.2. 4 계층 모델 아키텍처에서 M2 수준 매핑

UML 은 메타모델링 방법을 사용하여 정의되었다. (그림 1)은 메타모델링 방법을 이용하여 모델을 정의하는 것을 보여준다.



(그림 1) 4 계층 모델 아키텍처

설계자들이 일반적인 모델링 도구를 이용해 특정 시스템을 설계한다고 했을 때의 메타 모델 단계가 사용자 모델을 도식하게 되는 M1 단계이다. M2 단계는 그러한 UML 기반의 설계를 가능하게 하는 어트리뷰트, 클래스, 인스턴스 등과 같은 모델 요소를 정의하는 메타 모델이며, UML 2.0 의 하부구조는 바로 위 4 계층 메타 모델 관점에서 M2 수준의 UML 메타 모델이 된다. M3 수준에 위치한 MOF(Meta Object Facility)는 M2 수준에 속한 메타 모델을 정의하는 메타메타 모델이 된다. 이에 따라 C-to-UML 변환을 위한 매핑을 지원하기 위해서는 UML 2.0 의 메타모델 계층인 M2 수준에서 이루어져야 한다. 그리고 매핑은 그 특성에 따라 모델의 구조적 특성을 나타내는 클래스 매핑과 모델간에 관계를 나타내는 관계 매핑으로 분류된다.

2.3. 클래스 매핑

클래스 매핑 규칙은 C 언어의 하나의 C file 은 UML 의 Class 에 대응한다. 왜냐하면 일반적으로 프로그래밍 언어에서 공통된 기능들의 집합을 하나의 파일단위로 처리하기 때문이다. 변수는 속성으로 대응시켰고, 함수는 메서드로 대응시켰다. <표 1>설명한 규칙 외에 C 언어와 UML 의 구조적 특성에 따른 매핑 규칙을 나타내고 있다.

C language	UML 2.0	Use
C files	Class	C 파일은 클래스와 대응
Global Variables	Member Attributes	전역변수는 클래스의 멤버 변수로 대응
Local Variables	Local Variables	지역변수는 클래스의 메서드에 지역변수로 대응
Static Variables	Static Variables	정적변수는 클래스의 정적 변수로 대응
Pointer Variables	Pointer Variables	포인터 변수은 개념이 같음
array	array	배열은 개념이 같음
Structure	Class	구조체는 클래스로 대응
Function	Method	함수는 메서드로 대응

<표 1> C to UML 클래스 매핑 테이블

2.4. 관계 매핑

관계 매핑 규칙은 C 언어에서 각 파일간의 관계를 UML 에 표현하기 위한 매핑 규칙을 정의한다. Table 2 는 C-to-UML 의 관계 특성에 따른 매핑 규칙을 나타내고 있다.

C language	UML 2.0	Use
Include (1)	Inheritance	한 파일의 헤더만 포함한 경우 상속관계
Include (2)	Association	양쪽 파일의 헤더를 포함한 경우 연관관계
Struct pointer	Inheritance	구조체인경우 구조체 포인터는 상속관계

<표 2> C to UML 관계 매핑 테이블

3. 응집력을 이용한 클래스 재구성

메트릭이란 소프트웨어가 가지고 있는 속성의 크기나 정도의 계량적인 척도를 말한다. 본 논문에서는 클래스의 효율적인 구성을 위해 메트릭을 이용하여 소프트웨어가 가지고 있는 속성을 측정하고, 메트릭 결과에 따른 소프트웨어 재구성을 수행한다. 클래스의 측정 속성으로는 응집도를 이용한다. 클래스의 응집도란 클래스가 내부적으로 서로 연관된 요소로 이루어졌는지를 나타낸다. 따라서, 클래스 응집도 측정 결과에 따라 클래스 재설계 과정을 수행하여 클래스를 재구성한다.

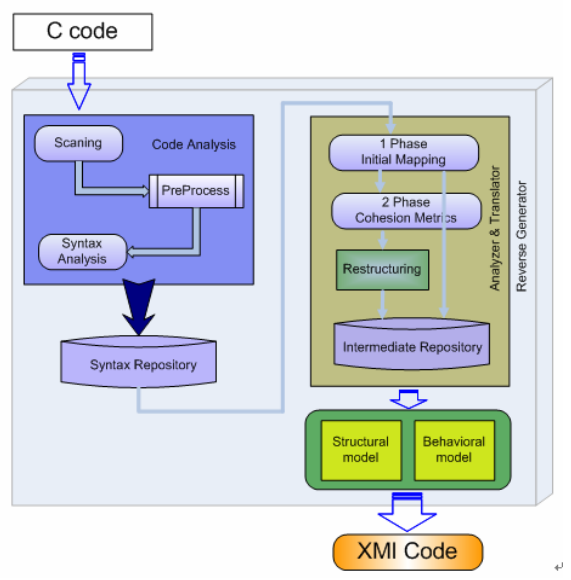
3.1. 클래스 재구성 방법

분리 클래스란 클래스 내부 구성 요소간의 연관성이 없는 경우를 말한다. 분리 클래스가 있다면 분리 클래스를 선택하여 내부 구성 요소간의 연관성이 없는 요소를 제거해야 한다. 클래스 내에 있는 오퍼레이션들이 공통으로 서로 작용하는 부분이 없는 경우를 보통 구성 요소간의 연관성이 없다고 볼 수 있다.

이상적인 클래스는 내부 구성 요소 사이에 응집도가 높고, 다른 클래스와는 결합도가 낮아야 한다. 이렇게 되면 클래스의 독립성이 향상된다. 클래스 내부의 구성 요소 간에 상호 작용이 없는 경우는 응집도를 떨어뜨리는 경우이므로, 응집도 저해 요인을 찾아 제거하여 응집도를 향상시키고 결과적으로 클래스의 독립도를 향상시키고자 한다. 그러므로 먼저 응집도가 낮은 분리 클래스가 있는지를 메트릭을 이용하여 판단한다. 만일, 분리 클래스가 있다면 분리 클래스를 선택하여 응집력을 낮게 하는 요소를 제거한다. 클래스가 분리 클래스인지의 판단은 인스턴스 변수 사용도를 통해 가능하다. 인스턴스 변수 사용도를 이용한 응집력 측정방법으로는 Bieman and Kang[8]이 제안한 TCC (Tight Class Cohesion)를 사용하여 측정한다. 이 TCC 방법은 암에서 언급한 바와 같이 메서드가 속성을 공통으로 사용한다는 성질에 의해, 클래스의 메서드들이 인스턴스 변수를 직접적 또는 간접적으로 사용하는 것을 이용한다. TCC 는 직접적으로 연관있는 메서드들의 쌍으로 정의되는데, TCC 의 결과 값이 0.5 이하인 경우 분리 클래스로 판단한다. 분리 클래스로 판단되면, 클래스 내에 메서드가 공통으로 사용하는 인스턴스 변수가 없는 경우로, 즉 연관관계가 낮은 메서드가 있는 경우로 연관관계가 낮은 메서드를 분리한다. 관련이 없다고 판단되어 분리된 메서드들은 다시 다른 클래스와의 연관관계를 조사한다. 만약, 제거된 메서드가 다른 클래스와 연관관계가 있다면 분리된 메서드를 연관관계가 있는 클래스에 삽입하고, 다른 어떤 클래스와 연관관계가 없다면, 새로운 클래스를 생성한다. 이런 재구성 과정을 통해 결과적으로, 클래스의 응집력을 향상시킬 수 있다.

4. 구현

역변환기는 크게 Code Analysis, Analyzer & Translator, Code Generator 의 3 단계로 구성되어 있다. 이와 함께, Code Analysis 단계와 Analyzer & Translator 단계에는 처리 시 사용되는 정보 및 데이터들을 저장할 2 개의 저장소(repository)를 구성하였다. 역변환기의 시스템 아키텍처는 (그림 2)에 나타나 있다.



(그림 2) 시스템 아키텍처

4.1. Code Analysis

파서 단계에서는 세부적으로 스캐닝, 전처리, 구문분석 단계로 세분화 하여 처리한다.

스캐닝: 첫 번째 단계인 스캐닝에서는 code 를 입력으로 받아 최소 의미 단위인 토큰으로 분할하는 역할을 한다. 여기에서 토큰은 이후의 단계에서 구문분석과 같은 변환 관련 정보들을 처리하기 용이하도록 하기 위한 단위를 의미한다.

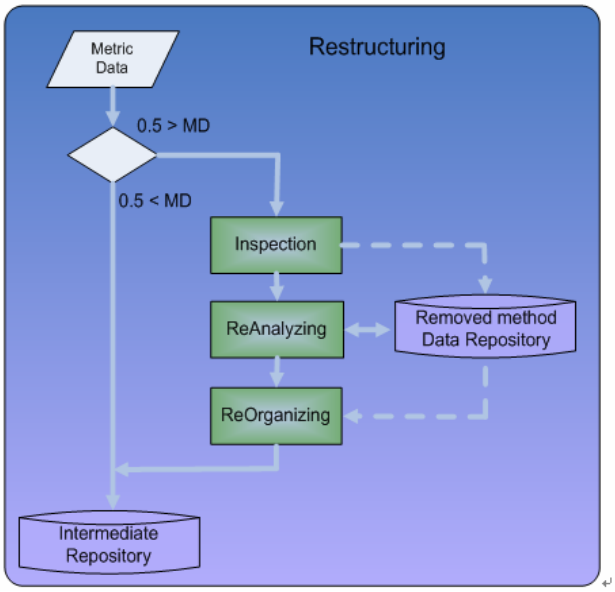
전처리: 전처리기에서는 파일간의 연관관계를 분석하기 위한 단계로 헤더파일을 분석하여 각각의 파일간의 관계를 파악한다.

구문분석: 구문분석기에서는 스캐닝단계에서 얻어진 토큰을 입력 받아서 프로그램의 구조를 결정하는 구문 분석을 수행한다. 구문분석기는 구문 트리를 생성하는데, 여기서 생성되는 구문 트리는 동적 할당을 위해 포인터기반 구조로 구축되었다.

4.2. Analyzer & Translator

두 번째 Analyzer & Translator 단계에서는 Code Analysis 단계에서 파싱된 데이터를 세 단계로 나누어 선택적으로 처리한다. Initial Mapping 단계에서는 매핑 규칙을 기반으로 일대일 매핑을 통한 변환을 수행하고, Cohesion Metric 단계에서는 Initial Mapping 의 변환된 결과를 가지고 각 클래스 내부의 응집력을 Bieman and Kang[8]이 제안한 TCC(Tight Class Cohesion)을 이용하여 각각 측정하여 그 측정 결과를 알려준다. Restructuring 단계에서는 Cohesion Metric 에서 분석한 결과를 기반으로 클래스를 재구성 할 것인지를 결정한다. 클래스가 응집력이 높으면 클래스의 재구성 없이 Intermediate Repository 에 저장을 하고 응집력이 낮다면, Inspection 을 통해 응집력을 낮게 하는 요소를 해당 클래스에서 제거하여 Removed method Data

Repository 에 임시 저장한다. 모든 클래스를 처리한 후, ReAnalyzing 모듈을 통해 제거된 메서드가 다른 클래스와의 연관관계가 있는지를 분석하여 다른 클래스와 연관 관계가 있다면, ReOrganizing 모듈에서 해당 메서드를 그 클래스에 삽입한다. 만약 제거된 메서드가 어떤 클래스 내부의 메서드와도 연관관계가 없다면 새로운 클래스를 생성한다. 아래 (그림 3)는 Restructuring 의 전체 과정을 보여주고 있다.



(그림 3) 재구성 과정

다음은 재구성 알고리즘을 나타내고 있다.

```
if TCC > 0.5 then it is OK
if TCC < 0.5 then
find_nonDC( Ci ) => { Mi | Mi is a method
that hasn't DC in Ci }
for all classes do
  if find_DC ( Cj, M ) then
    Mi is inserted to Cj
  else
    if Cn+1 is not exit then create Cn+1
    insert into the Cn+1
```

(그림 4) 재구성 알고리즘

4.3. Code Generator

마지막 Code Generator 단계에서는 Intermediate Repository 에 저장된 데이터를 Structural model 과 Behavioral model 로 각각 나누어 XMI Code 로 생성한다.

5. 결론

본 논문에서 우리는 임베디드 시스템 환경에서 모델 기반 방법론을 지원하기 위한 역변환 기법을 제안하였다. 또한, 단순히 코드에서 모델로 변환만이 아닌 모델의 완성도를 높이기 위해 클래스 재구성 방법도 제공하였다.

우리는 현재 코드에서 모델로 변환하는 방법 및 구현만을 하였지만, 특정 모델 기반 개발 방법론에 적용하지는 않았다. 앞으로 모델 기반 방법론 중 하나인 MDA 방법론에 적용하는 방법을 제안할 것이다.

참고문헌

[1] OMG. UML homepage: http://www.uml.org/
[2] Object Management Group Inc., "MDA Guide v1.0.1," http://www.omg.org, June 2003.
[3] Grady Booch, James Rumbaugh, and Ivar Jacobson. "The Unified Modeling Language User Guide," Addison Wesley, 1999
[4] Jichan Maeng, Jonghyuk Kim, Minsoo Ryu “Model-Driven Development of RTOS based Embedded Software”, the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
[5] Telelogic AB, "TAU," http://www.telelogic.com/.
[6] Stephen J. Mellor, Kendall Scott, Axel Uhl and Dirk Weise “MDA Distilled: Principles of Model-Driven Architecture,” Addison Wesley, 2004.
[7] Shyam R. Chidamber and Chris F. Kemerer “Towards a metrics suite for object oriented design”, ACM SIGPLAN Notices, Conference proceedings on Object-oriented programming systems
[8] James M. Bieman and Byung-Kyoo Kang “Cohesion and Reuse in an Object-Oriented System”, Proceedings of the ACM Symposium Software Reusability
[9] Martin Hitz, Behzad Montazeri. “Measuring Coupling and Cohesion in Object Oriented Systems.” Proceedings of the International Symposium on Applied Corporate Computing