제33회 한국정보처리학회 춘계학술발표대회 논문집 제17권 제1호 (2010. 4)

데이터 흐름을 반영하는 임베디드 시스템의 코드 자동 생성기 설계

이병용*, 류호동, 권진욱, 석미희, 이우진
*LG전자
경북대학교 전자전기컴퓨터과학과
e-mail:lby@lge.co.kr

A Design of Data Flow based Automatic Code Generator for Embedded System

Byeong-Yong Lee*,
Ho-Dong Ryu, JIn-Wook Kwon, Mi-Heui Seok, Woo Jin Lee
*LG Electronics
School of EECS Kyungpook National University

요 약

오늘날 임베디드 환경에서의 하드웨어의 발전에 더불어, 소프트웨어의 복잡도가 점점 증가하고, 유지보수에 대한 비용이 증가함에 따라 UML모델을 이용한 자동코드 생성에 대한 관심이 더욱 커지고 있다. UML을 이용한 코드 생성의 효과적으로 이루어지기 위해서는 설계된 모델의 무결성이 요구되고 이를 위해서는 모델의 논리적 검증이 선행되어야 한다. 아울러 설계자로 하여금 정의하는 모델이 명확하게 이해되고 구현될 있어야 한다. 하지만 코드 생성의 행위적 관점의 기본이 되는 상태머신 다이어그램에서 잘 드러나는 흐름과는 다르게 데이터의 사용은 다이어그램 내부에 숨겨져 있어 설계자로 하여금 모델에 대한 이해를 어렵게 하고 잠재적인 에러의 내포 가능성이 제기되어 왔다. 본 논문은 이러한 문제의 해결을 위해 코드 내포 상태머신 다이어그램의 데이터 시각화기법을 이용하고, 이러한 시각화 기법을 이용하여 데이터 사용관점에서의 모델의 이해를 도움과 동시에 이를 통하여 더욱 정확한 모델링을 수행하고 더불어 이를 통해 최종적으로는 더욱 효율적인 형태의 코드를 생성하는 코드 자동생성기의 설계를 제안 한다.

1. 서론

최근에 임베디드 환경에서 하드웨어의 비약적인 발전과 더불어 그에 대한 활용 범위가 넓어지면서 그에 따른 소프트웨어의 복잡도의 향상으로 인한 구현 및 유지보수에 대한 비용증가가 점점 큰 문제점으로도 대두되고 있다. UML을 이용한 코드 자동생성 기법은 점점 복잡해지는 소프트웨어의 개발 방법의 대안으로 점점 자리 잡고 있다. 이는 추상적인 관점의 설계와 코드 자동생성기를 통한 구현을 분리하여 기존의 설계된 모델의 재사용성을 높이고 새롭게 정의해야 할 부분을 감소시킴으로 최종적으로는 결과 코드의 무결성을 높임과 동시에 개발 비용을 감소시키는 결과를 가져온다. 이러한 장점은 유지보수 단계에서도 그대로 적용되어 문제 해결 시간을 단축하고, 유지보수 비용의 감소를 가져온다. 이러한 모델기반의 개발 방법은 오늘날의 임베디드 환경에서의 개발에서 오는 문제점의 해결을 위한 방안으로서 사용되어지고 있다.

모델 기반의 개발에서의 최종목적은 모델을 통한 소스코드의 생성이다. 본 논문에서는 기존의 상태 기반의 자동 코드생성기에서 한발 더 나아가 데이터 흐름 기반의 자동 코드생성기를 제안한다. 일반적인 모델링 과정에서는 상태머신 다이어그램을 통하여 쉽게 드러나는 상태변화와는 다르게 상태 속에 숨겨져 잘 드러나지 않는 데이터를 정의하여 이를 기반으로 하여 데이터 흐름에 대한 정의와 이해를 돕는다. 이를 통해 입력된 정보를 분석하여 더욱 효율성 있는 메모리 사용을 가능하게 하는 코드를 생성하는 자동 코드생성기를 설계하는 것이 본 논문의 주된 목적이다.

코드 생성 이전의 모델링 단계에서 데이터 시각화가 모델링 단계에서 데이터에 대한 명확한 이해에 그 목적이 있었다면, 코드 자동생성단계에서는 입력된 데이터에 대한 정보를 메모리 사용을 최소화하는 것이 그 목적이다. 물론 모델링 단계에서 위와 같은 작업을 통해 최적화를 시도할 수 있지만 이는 추상화 관점에 설계라는 모델링의 주된 목적중 하나를 지양한다는 문제가 있다. 즉 개발자로 하여금 최대한 직관적인 데이터 사용을 허락하고, 그로 인한 오버헤드는 코드 자동 생성기가 맡아 분석하여 좀 더 최적화된 코드를 생성한다.

본 논문에서는 먼저 2장에서 기존의 자동 코드 생성기의 구조와 특징들을 설명하고 3장에서는 데이터 흐름 정의 도구에 대한 설명과 시각화 도구에 대하여 설명한 후, 마지막으로 4장에서는 시각화 도구를 이용한 데이터 정보

※ 본 연구는 방위산업청과 국방과학연구소의 지원으로 수행되었습니다.

를 이용한 자동 코드 생성기의 설계를 보이고, 마지막으로 결론을 맺는다.

2. 코드자동 생성기 의 개요

일반적으로 모델로부터의 코드 생성과정은 상태 머신 다이어그램을 기반으로 하는 행위적인 부분과 컴포넌트 다이어그램을 기반으로 하는 구조적인 부분으로 나누어진다. 자동 코드 생성기는 각각의 구조 정보와 행위정보를 이용하여 각 소스파일간의 관계를 정의하고 파일 내부의 흐름을 정의하는 과정으로 코드를 생성한다. 아래의 그림 1은 모델로부터 코드를 자동으로 생성하는 과정을 도식화 한 것이다.

모델을 이용한 코드 자동 생성과정은 크게 구현 대상을 UML과 같은 추상적 언어로 정의하는 모델링 과정, 정의된 모델을 분석하여 각각의 패턴을 찾아내는 분석과정, 미리 정의 되어 있는 템플릿에 각 패턴을 적용하는 과정, 최종적으로 템플릿과 각 상태에서 정의된 코드들을 통합하는 코드 생성 과정의 4개 과정으로 나누어진다.

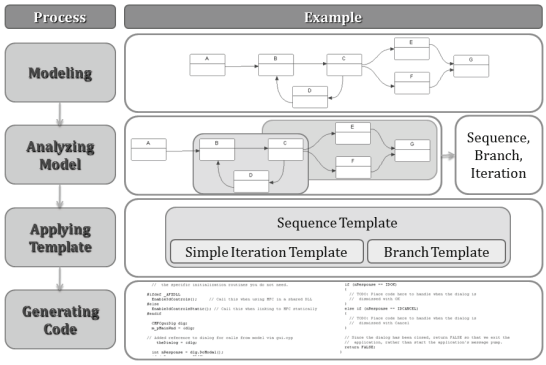

그림 1. 코드 자동 생성 과정

코드 생성에 대한 상세한 설명은 다음과 같다. 클래스 다이어그램과 컴포넌트 다이어그램과 같은 구조적 다이어그램에서는 자바의 클래스나 C에서의 파일과 같은 구조적인 형태의 틀에 대한 정보와 각각의 클래스나 파일 자체와 더불어 이들 간의 사용 및 피사용 관계도 추출한다. 예를 들어 클래스간의 상속관계나 사용관계, C언어에서의 include문이 소스 코드에 추가된다.

각 클래스의 메서드와 소스파일의 함수 내에서 실제 프로그램의 흐름에 대한 코드의 정의는 상태머신다이어그램의 정보로부터 추출한다. 스테이트머신 다이어그램은 스테이트머신을 최상위로 하여 영역, 상태, 상태의 전이로 나누어진다. 각각의 코드 생성기마다 각각의 요소를 정의하는 관점은 다르지만 일반적으로 스테이트머신은 메서드와 함수로 정의되고, 리젼은 메서드와 함수 내에서의 새로운 프로세서의 생성을 통한 분기상황을 정의하며, 상태와 상태 전이들의 집합은 분기문, 반복문과 같은 실제 코드상의 흐름으로 정의된다.

위의 일반적인 정의와 더불어 실제 코드 생성을 위해서는 각각의 모델 패턴에서 코드를 생성하기 위한 템플릿이 필요하다. 실제로 코드 생성기로부터 생성된 코드가 효율성을 가지기 위해서는 다양한 패턴에 대한 템플릿의 정의가 필요하다. 복잡한 형태의 스테이트머신의 각각의 스테이트를 연결하는 가장 간단한 방법은 모두 펼쳐서 switch문으로 만들고 각각의 스테이트를 case로 만드는 것이다. 각각의 스테이트의 코드는 case문 안으로 들어가게 된다. 그다음 다음 스테이트의 case값을 결정하는 if-else문을 추가 하고 마지막으로 switch문을 while 같은 반복문 안에 삽입하는 방식으로 이루어진다.

앞에서 설명한 기본적인 방식에서 템플릿은 switch문의 비효율성을 감소시키기 위한 방법으로 사용된다. 예를 들면 다른 분기 없이 일련으로 연결된 2개 이상의 스테이트들이 있을 때 switch문 기반에서는 n개의 스테이트로 구현되지만 실제로는 일련으로 연결된 패턴을 인식하고 이를 하나의 case문에 넣는 다거나, 반복되는 패턴을 파악하여 이를 하나의 while 혹은 for문으로 만드는 방식도 가능하다.

앞에서 열거한 일반적인 방법 외에도 코드 자동 생성기로부터 생성되는 코드의 효율성을 위한 많은 연구가 진행되고 있다. 지금까지도 개발자를 통해 직접 구현된 코드보다는 성능 면에서 부족하지만, 유지 보수상에서의 이점과 재사용 등과 같은 부수적인 이점으로 인하여 코드 자동 생성기에 대한 평가가 점점 나아지고 있다.

3. 시각화 도구를 이용한 데이터 흐름 정의

모델 기반의 코드 생성에 있어 효율적인 코드 생성은 효율적인 모델의 정의로부터 이루어진다. 이는 모델이 구조적, 논리적으로 무결해야 함을 의미한다. 본 논문에서는 모델 단계의 데이터의 사용으로 인한 논리적 오류 감소를 위해 데이터 시각화 기법을 이용한다. 이는 일반적으로 모델의 행위적인 정보를 나타내는 상태머신 다이어그램에서 프로세서의 흐름과는 달리 각각의 상태의 내부에서 정의되고 사용되는 데이터를 시각화하여 사용자로 하여금 모델링 과정을 돕고 데이터의 사용 과정을 드러냄으로써 잠재적인 논리적 에러를 감소시키는 기능을 가진다. 본 논문에서는 이러한 데이터 시각과 도구의 기본적인 기능과 더불어 모델링 과정에서 컴포넌트 다이어그램과 상태머신다이어그램의 데이터를 정의하는 인터페이스 기능과, 개발자에 의해 정의된 데이터 흐름을 분석하는 분석 기능을 이용하여 코드 생성기의 입력값으로 사용하기 위하여 사용한다.

3.1. 데이터 흐름 관점의 접근

데이터 시각화의 기본적인 개념은 일반적인 모델링 과정에서 숨겨지는 데이터를 밖으로 드러내는 것이다. 이를 위해서 다이어그램 편집기를 기반으로 하는 독립된 데이터 시각화 도구가 제안 되었다. 이 도구는 크게 데이터의 입력 기능과 분석 기능, 그리고 시각화 기능으로 나누어진다. 각각의 다이어그램의 각각의 구성 요소에서 별도로 이루어지던 데이터 입력 과정을 일원화 하여 모든 데이터

입력을 하나의 창에서 이루어지도록 하였다. 또한 실시간 분석을 통해 현제 입력되는 데이터와 연관되는 데이트를 분석하여 최종적으로는 각각의 데이터가 어떻게 사용되는지를 보여준다. 그림 2는 데이터 시각화 기능의 한 예로서 도구에서 선택된 변수가 사용되는 상태를 실시간으로 보여는 화면이다. 이러한 기능은 사용자로 하여금 데이터의 시각화를 통해 모델상의 데이터에 대한 이해를 도움과 동시에 입력된 데이터 정보를 코드 자동 생성과정에서 유용하게 사용 할 수 있도록 한다.

3.2. 모델링 과정에서의 데이터 흐름 정의 과정

다음의 그림 2는 컴포넌트 다이어그램과 상태머신 다이어그램, 마지막으로 상태머신 다이어그램상의 상태에 직접 입력된 변수들이 분석되는 과정을 보이고 있다. 데이터 시각화 도구는 다이어그램 편집기의 애드온 형태로 실행되며, 데이터 입력 부분과 분석부분, 시각화 부분으로 나누어진다. 데이터 입력 부분은 최상위의 전역변수부터 최하위의 지역변수까지의 변수를 입력 받는 부분이다. 이러한 입력은 컴포넌트 다이어그램에서 미리 정의된 전역변수 컴포넌트를 선택하거나, 각 상태를 선택하여 이루어진다.

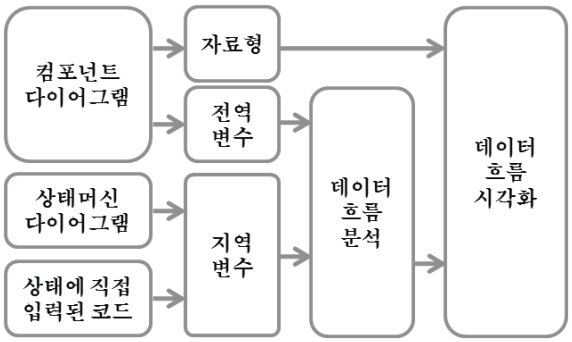

그림 2. 데이터 흐름 분석 과정

기본적으로 데이터 시각화 도구는 상태머신 다이어그램 편집기를 기반으로 한다. 시각화 도구는 편집기로부터 모델링 정보를 받아 각각의 정보를 분석하고 전역변수로 정의된 변수들의 정보를 함께 저장한다. 이들 정보는 변수의 정의, 할당 및 참조하는 부분을 추출하는 분석기로 전달하고, 분석기에서는 각각의 모델에서 어떤 변수가 어디에서 사용 되는지를 분석한다. 최종 분석된 정보는 다시 상태머신 편집기로 전달되어 특정 상태를 하이라이트 시키거나 상태내의 특정 변수를 하이라이트 하게 된다.

3.3 분석된 데이터 흐름의 사용

앞에서 제시한 데이터 시각화 도구의 특징은 크게 2가지로 나누어진다. 우선 데이터의 시각화로 인한 모델에 대한 이해에 용이하다는 점이다. 일반적인 모델링 과정에서 여러 데이터들은 상태 내부에서 정의되고 대입되는 과정을 거친다. 전체적은 소프트웨어의 흐름만큼 데이터의 사용 과정 역시 중요하지만 이전의 모델링 과정에서는 한눈에 드러나는 상태의 변화와는 달리 직접 혹은 다이얼로그를 통해 입력하는 방법만이 가능했다. 데이터 시각화는 이러한 감추어진 데이터를 밖으로 꺼내 데이터 사용에 대한 좀 더 직관적인 이해를 돕는다. 아울러 코드 상에서 하드코딩된 코드 혹은, 클래스의 애트리뷰트 형태 관점에서만 여러 데이터를 보던 틀에서 벗어나 데이터를 최상위의 전역 변수부터 최하위의 지역변수까지 체계적인 구조로 정의하여 각각의 데이터 들이 서로 연관하고, 연관되는 관계를 보여주어 각 데이터에 대한 좀 더 효율적인 관리를 가능하게 한다. 아울러 모델링 과정에서 입력된 정보는 코드 4장에서 설명할 코드 생성 최적화를 위한 정보로 사용된다.

4. 데이터 흐름을 이용한 코드 자동 생성기 설계

본 논문에는 모델링 과정에서는 정의 가능하지만 실제 하드웨어 상에서는 그대로 구현될 수 없는 동시성 문제와 유한한 메모리의 크기로 대표되는 하드웨어적 제약의 2가지 관점에서 코드 자동 생성기의 코드 최적화 방법을 제안한다.

4.1. 동시성 영역의 직렬화

동시성 문제의 대표적인 예는 모델링 상에서 하나의 상태 안에 2개의상의 영역이 동시에 수행된다고 정의하였으나 실제 코드에서 이를 그대로 적용 할 수 없을 때이다. 아래의 그림 3은 동시성이 적용된 상태 머신다이어그램의 예를 보이고 있다. 아래에서처럼 기압측정과 온도 측정이 동시에 이루어 져야 한다고 정의 되었으나 실제 하드웨어 상에서는 동시에 2개 이상의 프로세스가 실행 될 수 없는 상황일 때, 이를 구현하는 방법으로 본 코드 생성 도구에서는 2개 이상의 영역 중에서 주 영역을 임의로 선택하여 부 영역에 있는 코드를 주 영역에 임의로 삽입하는 방법을 제시한다. 즉 실제로는 CPU가 번갈아 가면서 수행해야 할 일을 코드 상에서 대신 구현하는 것이다. 이를 위해서는 각각의 영역에 있는 코드들의 분석이 필요하다. 이 과정에서 가장 중요한 부분은 반복문의 처리 부분이다. 본 논문에서는 가장 간단한 방법 중 하나로 각 반복 구분을 분해하여 삽입하는 방법을 사용한다.

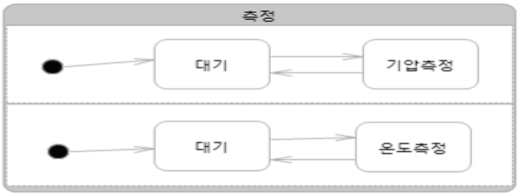

그림 3. 2개의 영역이 동시에 수행되는 상태

4.2. 메모리 최적화 기법

생성되는 코드의 메모리 사용을 최소화 하기위해서 본 도구에서는 모델링 과정에서 데이터 시각화 도구를 통해 입력된 데이터 정보를 이용한다. 이를 위해 코드 자동 생성기는 데이터를 크게 수직적인 구조 관점의 변수의 유효범위와, 수평적인 시간 관점에서의 변수의 수명의 2가지

관점으로 분석한다.

수직적인 관점은, 모델링 과정에서 사용되는 변수들의 수직적인 구조관계를 분석한다. 특정 클래스, 혹은 특정 소스파일에서만 사용되는 변수들과 이와는 반대로 코드 전체의 여러 부분에서 사용되는 변수를 찾아 분류하는 것이 첫 번째 관점의 주된 과정이다. 이 과정에서 코드 생성기는 전역으로 선언되었지만 특정 함수에서만 쓰이는 변수를 지역 변수로 만들거나, 같은 의미의 지역변수들을 통합하여 하나의 전역 변수로 만든 코드를 생성할 수 있다. 변수를 보는 또 다른 관점은 상태머신 다이어그램의 각 상태에서의 사용되는 변수들을 분석하는 방법이다. 앞에 단계가 변수들의 구조적인 관점에서의 최적화였다면, 이 단계에서는 특정 상태에서 동시에 접근 가능한 변수들의 재사용 가능성에 대하여 분석한다. 서로 영향을 주지 않는 각각의 블록에서 사용되는 지역변수를 블록 밖으로 꺼내어, 하나의 변수로 만들어 재사용하는 방법이다. 이를 위해서는 데이터 자체의 분석과 더불어 상태머신 다이어그램의 분석이 필요하다. 예를 들면 일반적인 모델링 과정에서 변수에 선언만 있을 뿐 변수가 어느 시점까지만 사용되는지에 대한 정보가 없기 때문에 특정 변수가 어느 상태에서 사용되지를 분석하여 그 변수를 다른 데이터를 위해 재사용 가능한 시점을 파악하는 것이다.

4.3. 코드 자동생성기의 구조

앞에서 제시한 2가지 방법 중 첫 번째 방법의 구현을 위해서는 2개 이상의 영역을 합성하는 도구와, 이에 필요한 템플릿이 정의 되어야 하고, 두 번째 방법의 구현을 위해서는 3장에서 제시한 데이터 시각화 도구로부터 입력된 데이터 흐름 정보가 필요하다. 다음의 5는 메모리 최적화를 위한 도구의 세부 구조를 보인다. 본 논문의 최종결과물인 자동 코드 자동 생성기는 2장에서 설명한 일반적인 코드 생성기의 형태에 앞서 제시한 2개의 도구가 추가된 형태로 구성된다.

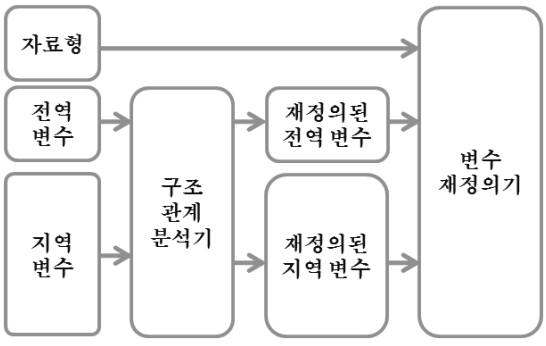

그림 4. 데이터 흐름 정의 과정

위 그림에서 보이듯이 본 논문에서 제시하는 메모리 최적화 도구는 크게 2개의 단계를 거친다. 우선 앞에 시각화 도구를 통해 입력된 전역 변수와, 지역변수들은 구조 관계 분석기를 거쳐 각각의 변수가 사용되는 상태들을 분석한다. 각 상태의 분포를 파악하여 특정 상태머신 다이어그램에서만 사용되는 전역 변수나 광범위하게 사용되는 지역 변수들을 각각 지역 변수와 전역 변수로 바꾸고 변수 재정의기에서는 서로 다른 상태에서만 사용되는 변수를 찾아 동일한 변수 명으로 재 정의하고 재 정의된 변수를 같이 사용하는 형태로 코드를 수정한다. 이 과정에서 시각화 도구에서 입력된 자료형 정보가 참조되어 재 정의되는 변수들이 서로 다른 데이터 형을 가지고 있을 때 이를 자동으로 형을 변환하는 과정에 사용한다.

마지막으로 동시성 영역의 직렬화 도구는 상태 머신 다이어그램으로부터 받은 상태 목록에서 동시성으로 정의된 목록을 추출하고 이를 직렬화하여 돌려준다. 이 도구는 상태머신다이어그램부터 동시성으로 정의된 상태를 추출하고, 상태의 영역들을 분석하여 패턴을 찾는다, 찾아낸 패턴은 미리 정의되어 있는 여러 형태의 템플릿 중 적합한 템플릿을 찾는데 사용되고, 최종적으로 선택된 템플릿을 기초로 상태내의 모든 영역이 하나의 주 영역 안으로 병합된다. 아래의 그림 5는 앞에서 설명한 일련의 과정을 도식화 한 것이다.

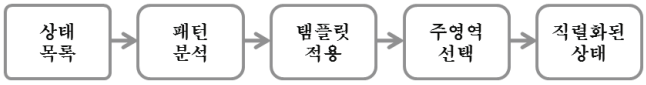

그림 5. 동시성 영역 직렬화 과정

5. 결론

본 논문에서는 시각화로 드러난 데이터를 코드로 생성하는 코드 자동생성기를 제시하였다. 기존의 흐름기반의 상태머신다이어그램에서 상태 내부에 숨어있던 데이터를 밖으로 꺼내, 모델 설계자로 하여금 모델 상에서 데이터 사용과정의 이해를 돕고, 아울러 모델의 무결성을 돕는 데이터 시각화 기법을 그대로 코드 자동생성과정에 추가하여, 데이터 사용 정보를 코드 생성에 사용하여 좀 더 효율적인 코드 생성을 가능하게 하였다. 2가지 관점에서의 최적화는 하드웨어적 제약이 심한 임베디드 환경에서 메모리 사용의 효율성을 가져오는 또 하나의 방법이라 생각한다.

참고문헌

[1] Object Management Group, OMG Unified Modeling

[2] 박영준, 맹한민, 이태종, 이우진, “Eclipse GMF 기반의 컴포넌트 다이어그램 편집기 개발”, 한국 정보 공학회 영남지부 학술 발표 논문지, 제 14권, 210-205쪽, 2006.

[3] Anoosh Hosseini, Dimitrios Mavroidis, Pavlos Konas, “Code generation and analysis for the functional verification of micro processors”, Proceedings of the 33rd annual Design Automation Conference, Annual ACM IEEE Design Automation Conference, pp. 305-310, 1996

[4] Orehek M, Robl C, Farber G, "Model-based design of an ECU with data- and event-driven parts using auto code generation", PROC IEEE INT CONF ROB AUTOM. Vol. 2, pp. 1346-1351. 2001