TMO기반의 코드 자동 생성기 설계를 위한 실시간 시스템 모델링의 기법

석미희 ○ 신영술 류호동 이우진
경북대학교 전자전기컴퓨터학부
miyachan@knu.ac.kr, youngsulshin@gmail.com, hodong@gmail.com, woojin@knu.ac.kr

Modeling of Real-Time System for TMO-Based Automatic Code Generator

MiHeui Seok○ YoungSul Shin, HoDong Ryu, Lee Woo Jin
Kyungpook National University Electrical Engineering and Computer Science

요 약

실시간 시스템에서는 계산 결과의 논리적 정확성과 결과가 산출되는 시간의 정확성을 요구한다. 이러한 요구사항을 지원하기 위해 UCI Dream Lab에서 제안한 적시 서비스 능력을 보장하는 실시간 객체인 TMO 객체를 토대로 자동 코드 생성기의 설계를 제시한다. 기존의 모델 기반 개발방법론은 시스템을 추상화하고 그 모델을 상세화 과정을 거쳐 구현에 필요한 정보를 갖게 되고, 이를 바탕으로 자동으로 코드를 생성 할 수 있게 된다. 기존의 도구들은 표준 UML을 사용하고, 표준 UML은 시간 제약 조건을 기술하는 방법을 제공하지 않기 때문에 이에 따라 개발된 실시간 시스템 코드 자동 생성기는 완전한 코드를 생성하지 못한다. 본 논문에서는 실시간 시스템 개발에 모델 기반 개발방법론을 적용하기 위해 TMO객체를 이용하여 기존 UML 모델의 구조 모델과 행위 모델에 실시간 특성을 추가해 확장하고 확장된 모델을 토대로 자동 코드 생성기의 설계를 제안한다.

1. 서 론

실시간 시스템에서는 계산 결과의 논리적 정확성과 결과가 산출되는 시간의 정확성을 요구한다. 이 요구사항을 지원하기 위해 본 논문에서는 UCI Dream Lab.의 Kane Kim에 의해 제안된 적시 서비스 능력을 보장하는 실시간 객체 모델인 TMO[1][2]를 이용한다.

현재의 모델 기반 개발방법론은 시스템을 먼저 추상적인 수준에서 모델링하고, 그 모델을 점점 명확하게 상세화하는 방식으로 개발하는 소프트웨어 개발 방법론이다. 이렇게 상세화 과정을 거쳐온 모델은 구현에 필요한 모든 정보를 가지게 되고, 이를 바탕으로 자동으로 코드를 생성할 수 있게 된다. 이렇게 코드를 자동으로 생성하게 되면 설계와 구현간의 무결성을 높일 수 있어서, 개발 비용과 유지보수 비용을 줄일 수 있다는 장점을 가지게 된다[3].

하지만 이러한 장점에도 불구하고 현재의 모델 기반 개발방법론을 실시간 시스템을 개발하는데 적용하기에는 힘들다. 왜냐하면 모델 기반 방법론에서는 시스템을 모델링 하는데 객체 기반 시스템을 모델링에 많이 쓰이는 UML[4]을 많이 사용하고 있다. 그러나 표준 UML은 시간 제약 조건을 기술하는 방법을 제공하고 있지 않기 때문이다. 그러므로 이에 따라 개발된 코드 자동 생성기 역시 시간 제약 사항을 반영한 코드를 생성할 수 없다.

실시간 시스템 개발에 모델 기반 개발방법론을 적용하기 위해서는 모델에 시간 제약 사항을 기술하는 방법이 필요하다. 그리고 이 시간 제약 사항을 반영한 코드를 생성할 수 있어야 한다. TMO는 객체에 실시간성을 부여할 수 있는 C++ 라이브러리로, 이를 이용하면 모델에 표현된 시간 제약 조건을 코드로 표현하는 것이 가능해진다. 이를 모델 기반 개발에 이용하기 위해서는 모델이 TMO가 필요로 하는 시간 제약 조건을 나타낼 수 있도록 해야 하고, TMO 기반 코드를 생성할 수 있는 자동 코드 생성기가 필요하다.

본 논문에서는 실시간 시스템에 모델 기반 개발방법론을 적용할 수 있도록 UML 모델을 확장한다. 특히 실시간 시스템을 개발하는데 최적화된 실시간 객체 모델인 TMO 구조를 토대로 모델 기반 개발방법을 제시한다. TMO 객체를 코드 자동 생성기가 생성할 수 있도록 UML에 시간 제약 조건을 기술할 수 있도록 만든다.

본 논문에서는 먼저 2장에서 관련연구로 TMO객체의

※ 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다. (UD060048AD)

개념과 특징, 코드 자동 생성기의 구조와 특징을 설명하고 3장에서는 TMO 객체를 구조 모델과 행위 모델의 관점에서 UML의 확장을 제시하고 TMO객체로 설계된 자동 온도 제어장치의 예를 들어 설계를 보인 후 4장에서 TMO 모델 기반 자동 코드 생성기를 설계하고 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1. TMO 객체

TMO(Time-Triggered Message-Triggered Object)객체는 UCI의 Dream Lab에서 개발한 객체 모델로서 객체 모델을 실시간 시스템에 적합하도록 확장한 것이다. TMO객체는 자체적으로 실시간 특성을 보장하고 C++ 라이브러리로 구현된 실시간 객체이다. TMO객체의 기본 구조는 그림 1과 같고 내부구조는 크게 ODS(Object Data Store), SpM(Spontaneous Method), SvM(Service Method)의 세 부분으로 구성된다[5].

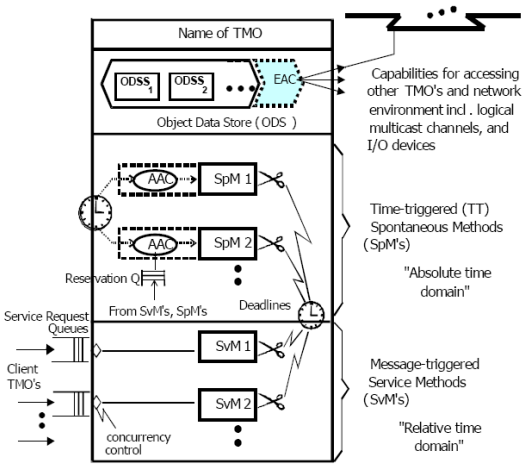

그림 1. TMO의 구조

각 부분을 살펴보면 ODS(Object Data Store)영역은 객체의 정보를 저장하는 영역을 의미하는 것으로 객체들의 데이터 멤버들로 이루어져 있고, EAC(Environment Access Capability)에서 원격 객체 호출과 입출력 장치 인터페이스를 지원하는 통신채널이 제공된다.

SpM메소드는 시간 구동 메소드로 주어진 시간 조건에 의해 자율적으로 구동 된다. SpM에는 시간 조건인 AAC(Autonomous Activation Condition)가 주어지고, 이에 따라 구동하게 된다. AAC는 시작과 종료시간, 주기, 데드라인으로 구성된다.

SvM메소드는 외부 또는 내부 클라이언트의 메시지 요청에 의해 구동하게 되는 메소드이다. SvM은 서비스 완료 시까지의 데드라인이 적용된다.

2.2. 모델 기반 코드 자동 생성

UML을 이용한 코드 자동 생성기는 컴포넌트 다이어그램을 기반으로 하는 구조적인 정보와 상태 머신 다이어그램을 기반으로 하는 행위적인 정보를 입력으로 받는다. 코드 자동 생성기가 코드를 생성할 때는 컴포넌트 다이어그램 토대로 컴포넌트 별 소스 파일을 생성하고, 각 컴포넌트의 행위를 표현하는 상태 머신 다이어그램을 토대로 그 컴포넌트의 내부 행위를 정의한다.

모델을 이용한 코드 자동 생성과정은 구현 대상을 UML과 같은 정형 언어로 정의하는 모델링과정을 거친 후 정의된 모델을 분석하여 패턴을 찾아내는 분석 과정, 미리 정의되어 있는 템플릿에 각 패턴을 적용하는 과정, 마지막으로 템플릿과 각 상태에서 정의된 코드들을 통합하는 코드 생성 과정의 3개 과정으로 나눌 수 있다. UML은 객체 지향 언어의 설계에 최적화 되어있는 모델링 언어로 실시간 소프트웨어에 있어 시간 제약 조건을 표현하지 못하는 한계점이 있다. 그렇기 때문에 표준 UML을 이용해서 TMO기반의 코드를 자동 생성하기에는 정보가 불충분하다. 이 문제를 해결하기 위해 본 논문에서는 클래스 다이어그램을 시간 제약 조건을 가질 수 있는 형태로 확장하고 상태 머신 다이어그램을 추가해 행위적인 측면을 나타낸다.

3. TMO 기반 모델링

TMO객체를 표현하기 위해 기존 UML에서 추가되어야 할 부분을 정의한다. TMO에서는 SvM메소드와 SpM메소드에 시간 제약 조건을 부여해야 한다는 것에서 기존의 객체와 다른 점을 지닌다. SpM메소드는 시작시간, 주기, 데드라인과 서비스 요청 마감시간 등의 정보가 필요하다. 이와 달리 SvM메소드는 서비스 완료시까지의 데드라인 정보만이 필요하다.

예를 들어 SpM메소드가 10시에 시작해서 10시 50에 끝나고, 매 10분마다 서비스를 요청하고 요청한 서비스는 5분 안에 서비스 제공을 완료해야 하는 시간 제약사항을 표현해야 한다면, 시작시간은 10시이고, 서비스 요청 마감시간은 10시 50분, 주기는 10분이다. 그리고 데드라인은 5분으로 설정하게 된다. SvM의 데드라인도 SpM의 데드라인과 마찬가지로 서비스 제공을 완료해야 하는 시간이다.

이러한 추가 정보는 기존의 UML에서 표현할 수 없기 때문에 기존의 UML을 확장하여 추가적인 정보를 표현할 수 있는 표기법이 요구된다.

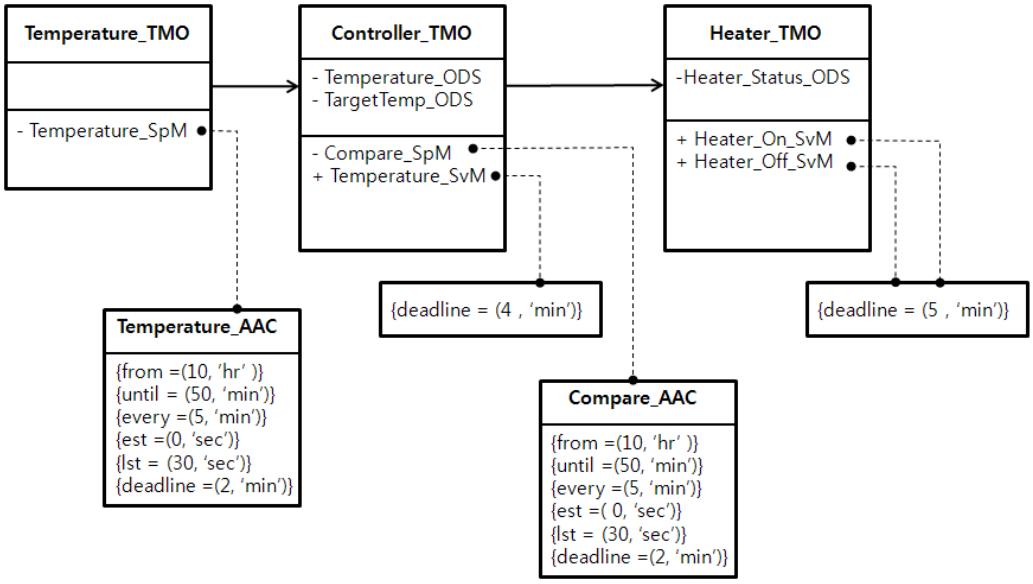

그림 2. TMO객체 기반 자동 온도 제어장치의 클래스 다이어그램

먼저 SpM의 AAC와 SvM의 데드라인을 클래스 다이어그램에 추가노드로 나타내고, 상태 머신 다이어그램에서 SpM메소드와 SvM메소드를 시간 제약 조건을 추가해 병렬적으로 나타낸다.

3.1. TMO 기반 시스템의 구조 모델

클래스 다이어그램에서 SpM의 AAC와 SvM의 데드라인 노드가 추가되었다. 시간 제약 조건을 명시하기 위해 확장된 클래스 다이어그램을 그림 2의 자동 온도 제어장치를 예로 들어 설명한다.

클래스 다이어그램의 Controller_TMO클래스를 보면 Compare_SpM의 시간 제약 조건을 표현한 Compare_AAC 노드와 Temperature_SvM 메소드의 시간 제약 조건을 표현한 deadline 노드가 추가로 연결되었음을 볼 수 있다. 세부적으로는 AAC(Autonomous Activation Condition)영역에는 from, until, every, est, lst, deadline 정보가 포함된 것을 볼 수 있다. 위의 시간정보는 from은 SpM 메소드의 시작 시간을 나타내고, until은 동작 기간, every는 주기, est(early start time)과 lst(lastest start time)로 명시된 시간 사이에는 무조건 실행되어야 한다는 것을 의미한다. deadline은 이 시간 안에는 끝나야 함을 의미한다. deadline에 명시된 시간을 초과할 경우 이는 데드라인 미스가 발생한다. SvM에 있는 deadline도 SpM의 deadline과 같은 의미로 수행 가능한 시간을 의미한다.

예를 들어 Contoller_TMO 클래스의 Compare_SpM 메소드는 '10시부터 시작해서 10시 50분까지 5분마다 다시 동작하고 시작한 시점부터 30초가 지나기 전까지는 수행되어야 하며 시작한 시점부터 2분 안에 동작을 마쳐야 한다.'는 것을 의미한다. 같은 클래스의 Temperature_SvM은 '호출된 시점부터 4분 안에 동작을 마쳐야 한다'는 것을 의미한다.

3.2. TMO 기반 시스템의 행위 모델

객체가 가진 SpM메소드와 SvM메소드를 표현하기 위해 확장된 상태 머신 다이어그램을 그림 3에서 보인다. 상태 머신 다이어그램에서는 각각의 SpM메소드와 SvM메소드를 병렬적으로 나타낸다.

상태 머신 다이어그램에서 SpM메소드와 SvM메소드는 각 메소드의 영역에 나타내고 각 영역은 표현의 추상화 수준이 다르게 나타난다. SvM은 외부에 노출되는 메소드이므로 SvM메소드 영역에서는 클래스의 상태 변화가 나타난다. SvM과 달리 SpM은 외부에서 호출되는 메소드가 아니기 때문에 SpM메소드 영역에서는 메소드 내부의 흐름을 나타낸다.

Controller_TMO클래스를 상태 머신 다이어그램으로

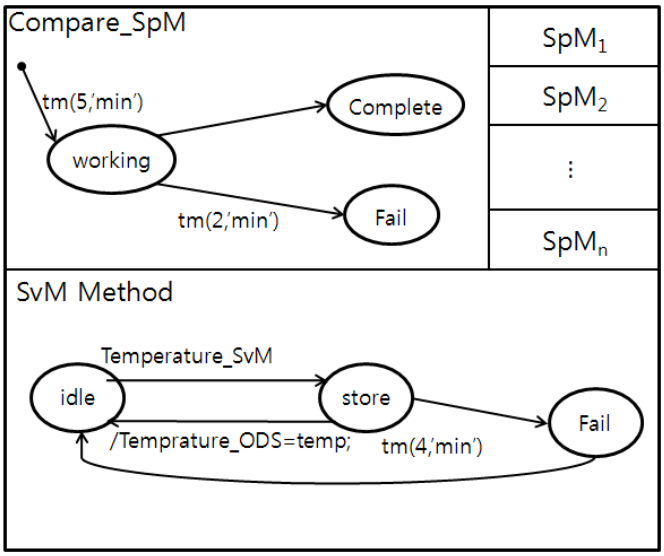

그림 3. Controller_TMO 클래스의 상태 머신 다이어그램

표현한 예를 그림 3에서 보인다. 각각의 영역에 Compare_SpM메소드와 Temperature_SvM의 행위를 나타내었다.

Compare_SpM메소드는 5분마다 호출되어 수행하고 데드라인인 2분의 시간 안에 서비스 수행을 완료하면 complete상태가 된다. 만약 데드라인을 넘기게 되면 fail상태가 된다. 만약 SpM메소드가 여러 개라면 영역을 추가해 병렬적으로 표현한다. Temperature_SvM 메소드는 호출되면 store상태가 되고 데드라인인 4분의 시간 안에 서비스 수행을 완료하면 idle로 돌아가서 대기하게 되고, 데드라인을 넘기게 되면 fail상태가 된다.

4. TMO객체 기반 자동 코드 생성기의 설계

TMO객체 기반 코드를 자동으로 생성하기 위해서는 앞 장에서 설명한 일반적인 코드 자동 생성방식에 더불어 2가지 기능이 추가 되어야 한다.

첫째로, TMO엔진에 SpM메소드와 SvM메소드를 등록해야한다. SpM메소드와 SvM메소드는 TMO 엔진에 등록되어 수행된다. SpM의 6개의 시간 제약 조건은 API를 통해 TMO 엔진에 등록할 때에만 사용하고, 등록 후에는 코드 생성기상에서 추가로 생성해야할 코드는 없다. 이는 SvM메소드 구현에서도 마찬가지이다. 실제 SpM 모델링 과정에서는 그림 3에서 보이듯, 6개의 시간 정보를 트랜지션에 나타내는 방식을 통해 정의되지만, 모든 수행은 엔진상에서 이루어진다. 코드 생성기에서는 Working 컴포지트 스테이트의 바디에 해당하는 모델 정보를 이용하여 코드를 생성하고, 이를 SpM메소드 바디로 등록한다. SvM메소드 바디는 받은 메시지를 처리하는 구조로서 일반적인 코드 생성방법으로 생성이 가능하다.

둘째로, TMO객체에서 메시지를 보낼 대상 TMO 객체의 SvM메소드를 위한 Gate를 생성해야 한다. TMO상에서 SvM메소드의 호출은 해당 메소드의 이 Gate를 통해 이루어진다. 본 논문에서 제안하는 자동 코드 생성기는 클래스 다이어그램에서 각 TMO객체간의 연결 정보를 이용하여 각 TMO가 의존관계에 있는 TMO의 SvM메소드를 위한 Gate를 자동으로 생성한다. Gate의 생성을 위해서는 대상 TMO 객체의 이름과 대상 SvM메소드의 이름이 필요하다. 이를 위해 코드 자동생성기에는 서로 연결된 각 TMO들의 목록과 각 TMO의 SvM메소드 목록을 가져와 등록해서 사용한다.

5. 결론 및 향후 연구

본 논문에서는 시간 제약 사항을 반영한 코드를 생성하기 위해, 모델이 TMO가 필요로 하는 시간 제약 조건을 나타낼 수 있도록 클래스 다이어그램과 상태 머신 다이어그램을 확장했다. 클래스 다이어그램에는 각 클래스의 시간 제약 조건을 노드를 추가해 확장하고 시간 제약 조건들을 열거해 필요한 정보를 추가하였다. 상태 머신 다이어그램에서는 각 메소드의 영역을 병렬적으로 나타내어 독립적으로 실행되는 것을 표현하였다. 또한 시간 제약 조건을 트랜지션에 표현하여 TMO의 특징을 코드 자동 생성기에 반영할 수 있도록 모델링하였다. 이를 바탕으로 TMO객체 기반의 자동 코드 생성기를 설계하였다. 자동 코드 생성기에서는 모델링한 정보를 바탕으로 SpM메소드와 SvM메소드를 등록하는 과정과 의존관계에 있는 TMO 객체의 SvM메소드를 위한 Gate를 생성하도록 설계했다.

향후 연구로 확장한 상태 머신 다이어그램과 확장한 클래스 다이어그램을 이용하여 TMO객체 기반의 자동 코드 생성기를 구현하는 것을 논의 중에 있다.

참 고 문 헌

[1] K.H. Kim, R. Paul, “The Distributed Time-Triggered Simulation Scheme Facilitated by TMO Programming,” In Proc. ISORC 2001 (Object-Oriented Real-Time Distributed Computing, 2001, on Fourth IEEE International Symposium (), pp. 41-50, 2001

[2] Kane Kim, “An Implementation Model for Time-Triggered Message-Triggered Object Support Mechanisms in CORBA-Compliant COTS Platforms”, In Proc. IEEE 1st Int’l Symp. On Object-oriented Real-time dependable Computing(ISORC), pp. 12-21, 1998

[3] 이병용, 류호동, 권진욱, 석미희, 이우진, “데이터 흐름을 반영하는 임베디드 시스템의 코드 자동 생성기 설계”, 한국정보처리학회 춘계학술발표대회 논문집, 제 17 권, 제 1 호, 2010

[4] Object Management Group, OMG Unified Modeling

[5] DREAM Laboratory, University of California, “TMOSL_Manual_v4_2_2”, http://dream.eng.uci.edu/TMOdownload/