UML 기반 고속열차 제어 소프트웨어 설계에 관한 연구
A Study On Software Design of High-Speed Train Control

심재철* 김찬용* 최권희** 김형인*** 정성윤***
Shim, Jae-Chul Kim, Chan-Yong Choi, Kwon-Hee Kim, Hyung-In Jung, Sung-Yun

ABSTRACT

The On-Board Computer System Used for rolling stocks is one of the core equipments in trains, which deals with collecting real-time data of a train to display and record the train's status, control the train, and assist the driver and maintenance function. To design control software for such complex control system, in this paper UML based modeling technique is applied.

A behaviorally expressive set of diagrammatic languages for modeling object-oriented systems is presented. It constitutes the constructive subset of UML, and also it enables model execution and full code synthesis.

1. 서 론

고속열차의 차량컴퓨터 제어장치는 실시간으로 열차의 데이터를 수집하여 상태를 현시 및 기록하며, 열차운전에 필요한 제어, 운전자 및 유지보수 지원 기능 등을 수행하는 열차의 핵심장치 중 하나이다. 이와 같이 다양한 기능을 수행하는 제어시스템의 소프트웨어를 설계하기 위해서 본 연구에서는 UML(Unified Modelling Language) 이라는 통합 모델링 언어를 기반으로 하여 객체 지향적인 분석과 설계방법론을 적용하였다.

고속열차 제어시스템은 분산형 구조로 구성되어 있고, 소프트웨어의 요구분석, 시스템 설계와 구현을 수행하는 과정에서 개발자간의 의사소통이 일치하지 않는 등의 문제 등으로 인하여 일관된 소프트웨어 설계에 애로를 겪게 되는데, 이러한 문제를 모델링 기법을 적용하면 개발자간의 의사소통이 원활해지며 시스템의 규모에 상관없이 설계가 가능한 장점을 갖게 된다. 본 연구에서는 모델링기법을 적용하여 고속열차 제어로직을 구현하고 임베디드 시스템에 적용하는 방안을 제시하고자 한다.

2. 본 문

2.1 새로운 개발 패러다임

산업분야에서 사용되고 있는 임베디드 시스템 설계는 시스템의 요구사양이나 그 기능이 컴퓨터 언어가 아닌 자연언어로 표현되어 있는 경우 정형화된 접근이 어렵고, 이러한 사양은 그 모호함 때문에 시스템을 분석하는데 어려움을 겪게 된다. 그러므로 설계자가 임베디드 시스템의 구조와 동작특성을 개념 단계에서부터 올바르게 정의하고 설계하기 위해서는 제대로 정의된 방법론으로 접근을 해야 한다.

* 인터콘시스템스(주), 기술연구소
E-mail : jcshim@interconsys.co.kr
TEL : (031)479-7460 FAX : (031)479-7462
** (주)로템, 기술연구소
*** 한국철도공사, 기술연구 TFT

임베디드 소프트웨어 설계는 일반적인 업무용 소프트웨어 개발과 비교하여 기능적으로 정확성을 유지하여야 할뿐만 아니라 여러 제약사항들을 만족시킬 수 있어야 한다. 즉, 특정 입력에 대해서 주어진 시간 내에 반드시 정의된 출력을 보장해주어야 하는 시스템이 임베디드 시스템이라고 할 수 있다. 설계의 초기단계에서부터 시스템의 사양을 보여줄 수 있는 그래픽 인터페이스 도구들을 사용하게 되면 설계자들이 관련된 정보들을 공유할 수 있고 모호하게 표현된 정보들을 보다 명확히 할 수 있게 될 것이다. 따라서 이 장에서는 임베디드 시스템의 설계에 UML(Unified Modeling Language)라는 소프트웨어 명세기술을 기반으로한 제어 소프트웨어의 개발에 대해서 기술하고자 한다.

2.2 플랫폼 기반의 설계

기존의 소프트웨어 개발 방식을 그림 1.에서와 같이 주어진 요구사양을 바탕으로 사양서를 작성하여 소프트웨어 설계를 하고 코딩 단계를 거쳐 테스트를 하고 코드 디버깅을 반복하게 되어 있다. 여기서 발생할 수 있는 문제는 시스템 개발 초기에 오류를 검출하고 시정하는데 어려움이 있고, 시스템이 구현된 이후에 오류를 검출하고 시정하는데 더 많은 노력과 비용이 들어간다는 점이다.

다양한 플랫폼에서 수행되는 시스템 설계의 요구가 있고 이러한 개발 프로세스의 변화요구는 개발 패러다임을 변경시키고 개발도구의 필요성을 증가시키게 되었다. 그동안 객체지향 프로그래밍, 컴포넌트 기반의 설계, 자바와 같은 플랫폼에 종속되지 않는 기술 등이 시도되었으나, 이러한 시도들은 소프트웨어의 양이 많아짐에 따라 기하급수적으로 비용이 발생하는 버그 문제에 대처하기에 어려움이 있었고 개발의 각 단계에서 작성하는 문서와 소프트웨어 개발을 동기화시킬 수 있고 설계 당시부터 테스팅에 대한 고려가 가능한 접근 방법이 되지는 못하였다. 따라서 사양에서부터 설계, 구현, 시험에 이르기까지 일관성있고 생산성을 높일 수 있도록 새로운 개발 프로세스의 해결책으로 MDD(Model-Driven Development) 접근방식을 채택하였다.

2.3 UML 소개

UML은 시스템 또는 소프트웨어 모델링 표준언어로서 1997년 OMG(Object Management Group)에 의해 표준화가 이루어진 객체-지향 모델링 언어이다. 점점 복잡해지는 실시간 시스템을 반영하여 확장 및 변경이 가능하고 다양한 분야에 적용이 가능한 언어로 정착하여 현재 UML 2.0 버전으로 자리잡게 되었다.

UML은 그림 2.와 같이 여러 가지의 다이어그램들로 표현되면 이러한 그래픽적인 인터페이스를 제공해줌으로써 시스템 이해를 위한 폭넓은 시야확보가 가능하고 개발자간의 원활한 의사소통이 이루어지도록 만들어주며 요구사항 변경 등에 능동적으로 대처하고 소프트웨어의 재사용성을 증가시켜 준다.

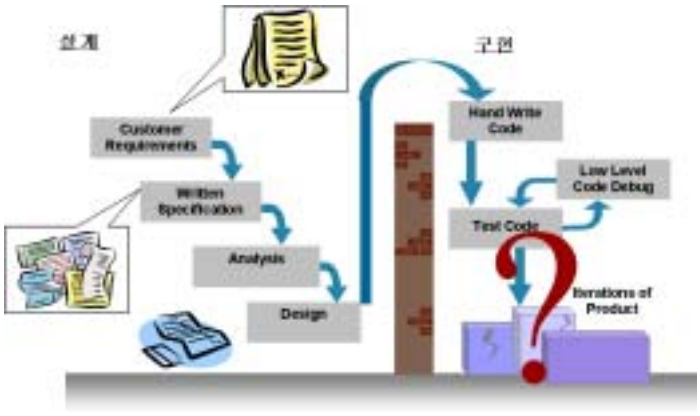

그림 1. 기존의 S/W 개발 방식

2.4 UML을 이용한 개발 프로세스

UML은 단순한 소프트웨어 개발 방법론이 아니고 방법론을 따르는 설계시 사용되는 컴퓨터 언어이기 때문에 요구사항을 분석하고 시스템을 분석하여 설계, 구현 그리고 테스트하는 개발 프로세스 전반을 포함하고 있다.

요구분석은 설계할 시스템의 요구사항을 충분히 반영하여 이를 문서화하는 단계이고 UML에서는 유즈 케이스 다이어그램(Use Case Diagram), 간단한 클래스 다이어그램(Class Diagram) 그리고 액티비티 다이어그램(Activity Diagram)등을 사용하여 표현한다.

시스템을 분석하는 단계에서는 시스템의 입출력 관계와 시스템이 수행하는 기술적인 문제들을 풀기 위한 분석을 수행하고 이를 위해서 적당한 클래스들을 선정하고, 클래스들 사이의 정적인 관계를 모델링하여 관계를 표현한다. 오브젝트들 사이의 행위(behavior)와 협력(collaboration)등을 상태도(State Diagram), 시퀀스도(Sequence Diagram) 등을 사용하여 표현하고 기본적인 사용자 인터페이스의 프로토-타입도 필요에 따라서 작성한다.

설계 단계에서는 위의 분석 결과에 따라서 시스템을 어떻게 구현할 것인지를 중심으로 어떻게 시스템이 동작하고 어떠한 제약 사항들이 있는지를 기술적으로 정하는 단계이다. 이렇듯 설계와 기술적인 부분을 분리시키는 이유는 분석 단계에서 도출된 결과들이 설계단계에서도 변경시키지 않고 유지하면서 하부구조를 좀 더 쉽게 만들어나가기 위함이다. 이 단계에서는 필요한 외부 클래스 라이브러리나 오브젝트 내에서 사용할 컴포넌트들을 명시하고 시스템에서 예상되는 예외상황에 대처하기 위한 에러처리를 고려하여 설계한다.

구현 단계에서는 실제 사용하게 될 소스코드들 생성하게 되는데, 앞선 설계단계가 원하는 대로 이루어졌다면 이 구현단계는 각 다이어그램들에게서 특정 언어(예를 들면, C, C++, Java 등) 구문으로 번역을 해주고, 컴파일 및 디버깅하는 작업을 수행하게 된다. 이 과정에서 필요한 연산기능(operation)들을 추가하고 코딩하는 것도 가능하다. 이렇게 구현된 어플리케이션은 마지막으로 소프트웨어 시험을 거쳐 코드에서 발생하는 에러들을 찾아 수정하게 되고, 테스트의 결과로 발생하는 에러들은 문서로 생성시켜 이것이 다음 버전에서 고쳐질 수 있도록 한다.

2.5 실시간 어플리케이션 적용

UML을 사용하여 생성된 어플리케이션을 RTOS(Real_Time Operating System)에서 사용하기 위하여 그림 3.과 같은 프레임워크를 사용하여 기존의 C와 같은 언어로 작성된 코드와 UML에서 생성된 소스 코드 등을 병합하는 것이 손쉬워 진다. 플랫폼에 종속적이지 않은 PIM(Platform Independent Models)을 사용함으로써 플랫폼의 API(Application Program Interface)를 바탕으로 다양한 RTOS에 활용이 가능하다.

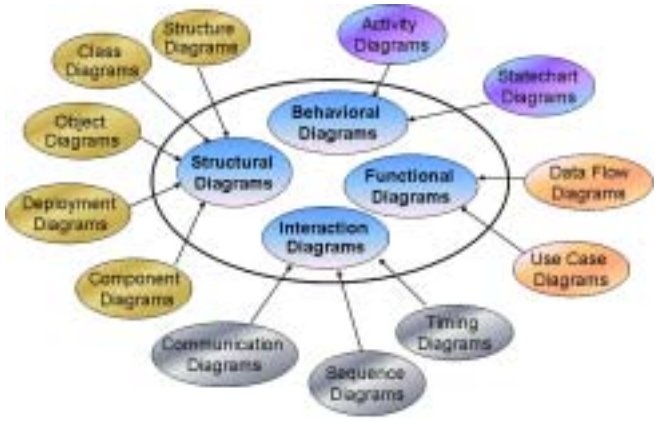

그림 2. UML에서 사용되는 다이어그램들

이와 같이 플랫폼 API를 추상화시킴으로서 다양한 플랫폼에 이식이 용이하고 개발 단계에서도 플랫폼에 상관없이 구현하고 테스트하는 것이 가능해짐으로써 개발기간을 단축시켜주고 모델과 프로그래머가 직접 작성한 코드간의 연계가 되기 때문에 소프트웨어의 개발 생산성이 향상되는 장점이 있다.

개발 초기 단계에서부터 개발 팀원들이 모두 자원을 공유할 수 있는 다이어그램들을 사용하였기 때문에 일관성 있는 개발이 가능하고 타겟에 어플리케이션을 이식하기 전까지는 일반 PC의 소프트웨어 환경에서 컴파일하고 실행시켜서 제어로직의 오류 또는 코드의 오류가 있는지 테스트가 가능하였다.

2.5 소프트웨어의 시험

UML을 사용하여 소프트웨어를 개발하는데 있어서의 또 한가지 장점은 소프트웨어 요구사양으로부터 시스템을 테스트하기 위한 시스템 시나리오를 제공해 줄 수 있다는 점이다. UML 개발환경에서 모델은 실행이 가능하며 개발팀원들 간의 협업 디버깅이 가능하고 개발 전체 과정에서 반복적으로 수행이 가능하여 비용 및 시간을 절감할 수 있다.

시험을 수행하는 동안 발견되는 오류 또는 코드의 수정은 기존의 디버깅 방식에 비교해서 코드 자체에 집중하지 않아도 되기 때문에 효율적이고, 모델에 수정을 가한 경우에는 이 행위가 반영되어 새로운 코드가 생성되고 반대로 코드에 수정을 행한 경우에도 이 내용이 반영되어 모델이 갱신되기 때문에 오류를 수정하는 과정에서 일관된 작업이 가능하였다. 개발되는 시스템을 전체 코드가 완성되기 이전에도 즉시 실행시켜 보면서 제대로 구현되는지 볼 수 있어서 에러를 개발단계에서부터 미리 줄이고 이로 인해 빠른 시간내에 개발이 가능하였다.

2.6 적용 사례

본 논문에서는 고속열차의 차량제어 컴퓨터장치에서 운용되는 실시간 제어로직을 UML 언어를 사용하여 구현하였다. 제어로직에는 열차의 추진장치, 제동장치, 출입문 제어 등과 같은 다양한 하부장치와 입출력을 통하여 제어행위를 수행하며, 열차의 고장을 진단하고, 기관사가 화면을 통하여 고장시 지원을 받을 수 있는 기능 등이 포함되어 있다. 고속열차는 추진을 담당하는 동력차와 승객이 탑승하는 객차로 구분되어 있어서 동력차에는 동력차 처리장치가 있고, 객차에는 객차 처리장치가 각각 탑재되며 이에따라 제어 소프트웨어도 기능이 구분되어 별도로 탑재된다.

동력차에서 주로 수행하는 소프트웨어 기능으로는 견인/제동 처리, 추진장치 및 보조전원 제어와 감시, 운행중 사건기록 등이 있고, 객차에서는 열차 승무원 정보현시, 객차 열감지 등의 기능을 담당한다.

소프트웨어 요구사양은 상태전이도로 작성되어 각 상태에서 다음 상태로 전이되는 전이조건들을 기술하고 이에 필요한 입출력 변수, 네트워크 변수, 그리고 정해진 시간지연 변수 등을 정의하여 사양서로 작성하였다.

소프트웨어 요구 사양서를 분석하고 시스템이 수행하게 되는 제어로직을 분석하여 해당 제어로직의 오브젝트 다이어그램을 작성하고, 상태전이도를 UML언어에서 사용하는 다이어그램들 중에서 상태 차트(State Chart)를 이용하여 구현하였다. 그림 4.에서와 같이 작성된 상태 차트에는 상태 진입시 및 진입 진출 시 처리해야할 코드를 포함시켰고, 전이 조건들은 오퍼레이션(operation)으로 지정하여 직접 코드를 입력시켰다. 본 논문에서는 RTOS 환경으로 VxWorks를 적용하였고, UML 개발툴로는 Rhasody를 선정하여 고속열차 제어로직을 모델링하고 이를 테스트하여 타겟에 다운로드까지 수행을 하였다.

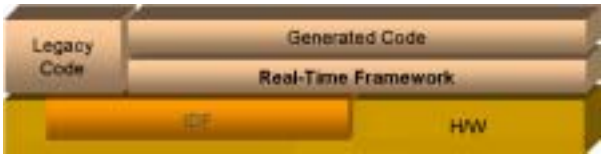

그림 3. Real-Time Framework

이렇게 작성된 모델들은 컴파일 과정을 거쳐서 소스코드가 생성되는데, 본 개발에서는 C++ 코드를 생성시키도록 하였다. 코드는 모델로부터 자동 생성되며 생성된 코드는 숙련된 프로그래머가 일관되게 작업한 것과 같이 출력되어 가독성이 좋아서 코드를 보고 디버깅하거나 루틴을 추가하는데 어려움이 없었다. 개발자는 개발 초기시부터 어떻게 코드를 구현해야 할지를 고민하기보다는 어떻게 모델간의 관계를 만들고, 주고받는 데이터에 대해서만 그래픽 환경에서 고려하면 되기 때문에 보다 문제의 본질에 집중할 수 있게 되었다.

모델링 작업은 예상했던 개발기간보다 소수의 인원으로 짧은 시간내에 마칠 수 있었고, 그림과 같이 직관적인 인터페이스를 통해서 작업을 하기 때문에 개발자간의 서로 다른 개발 스타일로 인해 발생하는 문제점을 근본적으로 막을 수 있어서 팀원들 간에 의사소통이 원활할 수 있었다. 개발된 어플리케이션을 타겟에 다운로드하기 전에 시험을 위하여 별도의 플랫폼을 사용하지 않고 개발자의 컴퓨터에서 그림 5.와 같이 단순한 시험용 절차를 작성하여 어플리케이션과는 별도로 수행이 되도록 구현하였다.

작성된 어플리케이션은 개발자의 컴퓨터에서 수행되기 때문에 별도의 입출력을 가질 수 없다. 따라서 이를 위해 그림 5.의 시험용 State Chart에서 어플리케이션으로 가상의 입력을 주고 이벤트를 발생시켜 실제 환경에서 수행되는 것과 동일하게 시험이 가능하였고, 이 과정을 단위 로직별로 수행하여 발생할 수 있는 경우의 수를 모두 고려한 시험이 가능하였다.

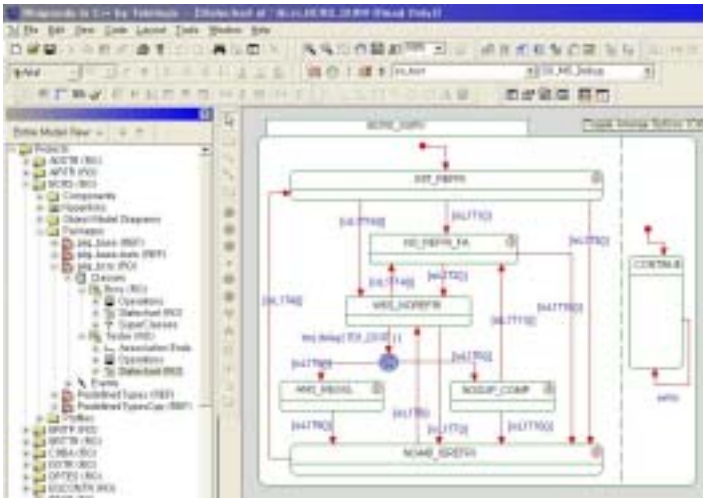

그림 4. 상태 챠트(State Chart)

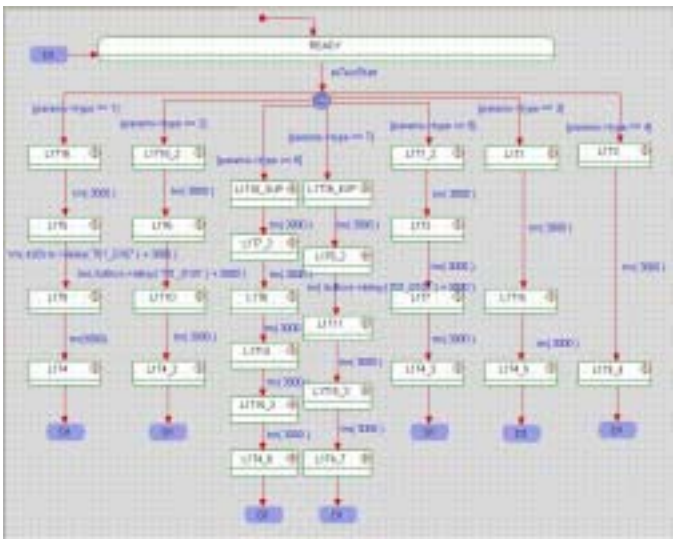

그림 5. 제어로직 시험용 절차

3. 결론

본 논문에서는 UML을 기반으로 하는 실시간 고속열차 제어 소프트웨어의 개발에 대해서 살펴보았다. 명확하지 않고 모호하게 작성된 요구사양으로부터 원하는 기능을 수행하는 시스템을 개발하는데는 많은 어려움이 따르겠지만 특히, 소프트웨어 명세를 명확히 하지 못해서 이 문제를 개발자들이 서로 다르게 이해하여 프로젝트를 수행하게 될 경우 원하는 결과를 얻기도 힘들뿐더러 개발이 진행되고 난 이후에 이를 수정하는데는 더 많은 노력이 필요하게 될 것이다.

적용 사례에서 보았듯이 UML 언어를 사용하여 개발 초기 단계부터 모델을 사용하여 통합된 요구분석 및 요구사항 추적이 가능하고, 그래픽적인 인터페이스를 제공해주기 때문에 효과적인 소프트웨어 설계 협업이 가능하였다. 품질 높은 코드의 자동생성으로 개발 기간을 단축할 수 있었고 완벽한 모델과 코드의 재사용을 통한 유연한 개발환경에서 개발이 가능하였다. 임베디드 시스템에 적용하기 위해서 간결하고 실행하기에 가벼운 코드가 생성되어야 하는 이러한 요구 조건들을 만족시켜 주었다.

UML기반의 개발 환경은 개발의 생산성 저감요소들을 최소화시켜주어 개발에 따른 위험을 감소시켜주고 소프트웨어의 재사용이 가능하기 때문에 추후 업그레이드 및 유지보수 측면에서 용이하다고 볼 수 있겠다. 아울러 개발기간을 단축시켜줌으로써 빠르게 변화하는 시장환경에 대응할 수 있고, 개발비용을 절감할 수 있는 장점이 있다.

참고문헌

1. 건설교통부(2006), “경부고속열차 차량컴퓨터제어장치개발“, 1차년도 연차 보고서
2. Rick Boldt(2004), "Model Driven Architecture", I-Logix
3. Grady Booch(1998), "Unified Modeling Language User Guide", Addison Wesley
4. Jerome L. Krasner(2004), "Reducing OEM Development Costs and Enabling Embedded Design Efficiencies Using the Unified Modeling Language(UML 2.0)", Embedded Market Forecasters