CASE 도구를 이용한 ATC 차상 소프트웨어의 UML 모델링
Modeling of ATC On-Board Software in UML Using CASE Tool

양 찬 석*　임 재 식**　한 재 문***　김 치 조****　조 용 기****
Yang, Chan Seok　Lim, Jae Shik　Han, Jae Moon　Kim, Chi Jo　Cho, Yong Gi

---

ABSTRACT

ATC on-board equipment require realtime embedded software with high level of safety and reliability. To satisfy these requirements, many techniques are applied to the development of software during the lifecycle. In case of software modeling, object-oriented methodology is widening its niche replacing traditional structured methodology and modeling in UML using a CASE tool is a growing trend. In this paper, we modeled ATC on-board software in UML using Rhapsody, which is a modeling tool for realtime embedded software. We modeled the behavior of ATC on-board equipment based on state machine diagram and validated the model using the animation feature provided in the tool. According to our study, the CASE tool based on UML showed high level of applicability in modeling and verifying the software with complex behavioral characteristics.

---

1. 서론

ATC 차상 소프트웨어는 높은 수준의 신뢰성과 안전성을 갖추어야 한다. 이를 위하여 신뢰성 및 안전성 관련 규격들은, 소프트웨어 개발 생명주기 전반에 걸쳐 많은 기법들을 적용할 것을 요구하고 있다. 소프트웨어 모델링에 있어서는, 정형적인(formal) 또는 반-정형적인(semi-formal) 방법들 그리고 구조적인 방법론들을 이용하도록 요구하거나 권장하고 있다. 전통적으로, 특히 실시간 내장형 소프트웨어의 영역에서는 구조적인 방법론(structured methodology)이 강세를 보여 왔으나, 근래에 들어 객체지향 방법론(object-oriented methodology)의 전반적인 영향력 확대와 함께 변화의 조짐이 나타나고 있다.

본 논문에서는 객체지향 방법론의 대표적인 표기법인 UML을 기반으로 한 CASE 도구를 이용하여 ATC 차상 소프트웨어를 모델링 함으로써, 복잡한 행위특성을 가진 소프트웨어에서 UML기반 CASE 도구의 적용가능성을 타진해 보고자 한다.

2. 본문

UML(Unified Modeling Language)은 복잡한 시스템의 구성요소 및 요소들 간의 상호관계를 표현하기 위한 언어이다. UML은 객체지향 방법론의 표준을 수립하려는 OMG(Object Management Group)의 요청에 따라 처음 만들어졌으며, Rational Software의 Grady Booch, Jim Rumbaugh, Ivar Jacobson의 주도하에 OMG에 의해 표준으로 채택되었다. UML은 복잡한 시스템을 모델링 하는데 있어서 다른 방법들보다 완결된 구조를 갖고 있으며, 특히 실시간 내장형 시스템을 모델링 하는데 적합하다.

---

* LS산전(주) 중앙연구소, 책임연구원
** LS산전(주) 중앙연구소, 선임연구원
*** LS산전(주) 중앙연구소, 주임연구원
**** LS산전(주) 중앙연구소, 수석연구원

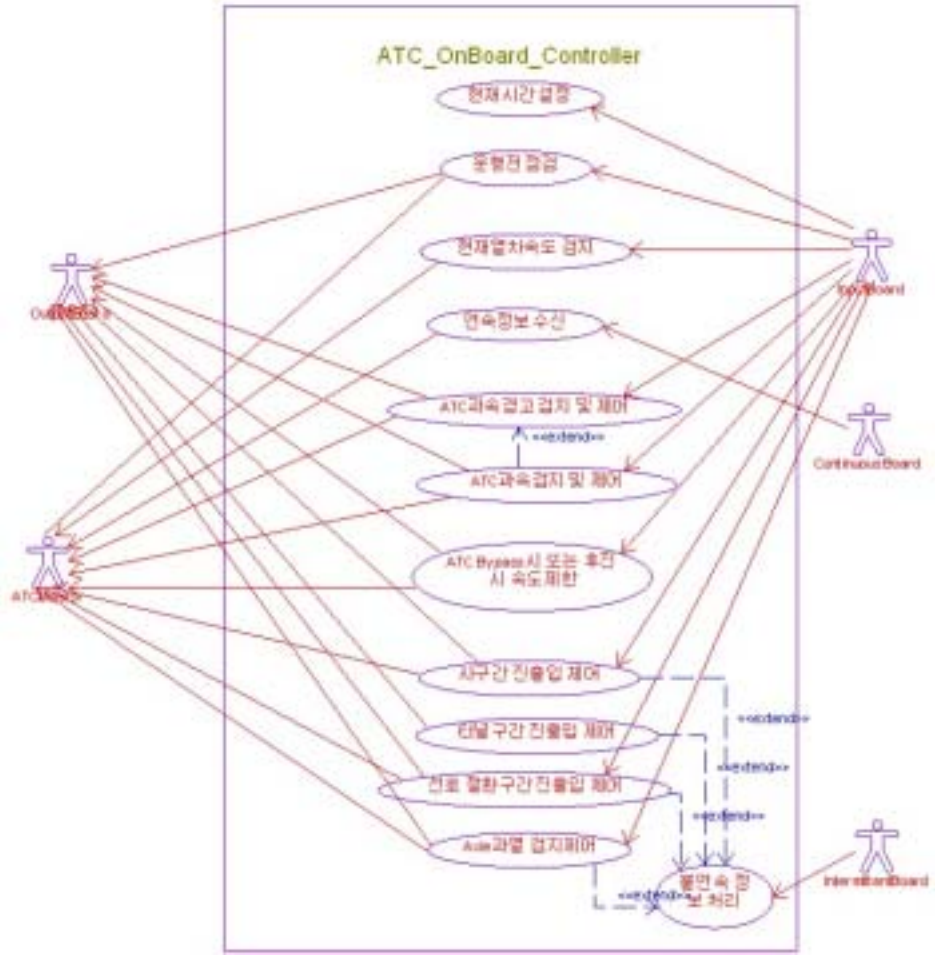
UML을 이용하여 시스템을 모델링 하는 경우, 시스템의 정적인(static) 특성을 모델링 하는 구조적(structural) 모델링과 동적인(dynamic) 특성을 모델링 하는 행위적(behavioral) 모델링을 통해 전체적인 특성을 파악하게 되며, 여러 가지 다이어그램(diagram)들을 이용하여 대상 시스템을 표현하게 된다. 구조적 다이어그램들에는 클래스(class) 다이어그램, 컴포넌트(component) 다이어그램, 배치(deployment) 다이어그램 등이 있고, 행위적 다이어그램들에는 활동(activity) 다이어그램, 시퀀스(sequence) 다이어그램, 상태 머신(state machine) 다이어그램 등이 있다.

본 논문에서는, I-Logix의 Rhapsody 도구를 이용하여 ATC 차상 소프트웨어를 모델링 하였으며, 유스 케이스(use case) 다이어그램, 클래스 다이어그램, 시퀀스 다이어그램, 상태 머신 다이어그램을 통해 대상 시스템을 표현하였다. 또한, Rhapsody 도구 내의 애니메이션(animation) 기능을 이용하여 모델을 시뮬레이션 해 봄으로써, ATC 차상 소프트웨어가 만족시켜야 하는 기능적인 요구사항이 충족되는지 검증하였다.

2.1 ATC 차상 소프트웨어 UML 모델링

2.1.1 유스 케이스 다이어그램

유스 케이스 다이어그램은 액터(actor)와 시스템, 그리고 유스 케이스(use case) 간의 관계를 표현하는 다이어그램이다. 여기서 유스 케이스는 시스템이 수행하는 일련의 행동들을 명세한 것을 의미하고, 액터는 시스템 경계의 외부에 존재하면서 특정 유스 케이스의 수행과 관련하여 시스템과 상호 작용하는 개체를 의미한다.



[그림 1] ATC 차상 소프트웨어 Use Case 다이어그램

[그림 1]은 ATC 차상 소프트웨어의 유스 케이스 다이어그램을 보여준다. 그림에서 타원형은 유스 케이스를 나타내고 사람 모양의 아이콘은 액터를 나타낸다. 추출된 액터들에는 '입력 보드(InputBoard)', '연속 보드(ContinuousBoard)', '불연속 보드(IntermittentBoard)', '출력 보드(OutputBoard)', '운전자 모니터(ATCMonitor)'가 있고, 유스 케이스들에는 '열차속도 검지', '연속정보 수신', 'ATC과속검지/제어', '불연속 정보처리' 등이 있다.

'불연속 정보처리' 유스 케이스의 경우, '사구간 진출입 제어'를 포함한 몇 개의 유스 케이스들과 <<extend>>라는 라벨이 붙은 점선 화살표로 연결되어 있는데, 이는 '불연속 정보처리' 유스 케이스를 수행하는 중간에 각각의 경우에 따라 유스 케이스가 확장되는 것을 의미한다.

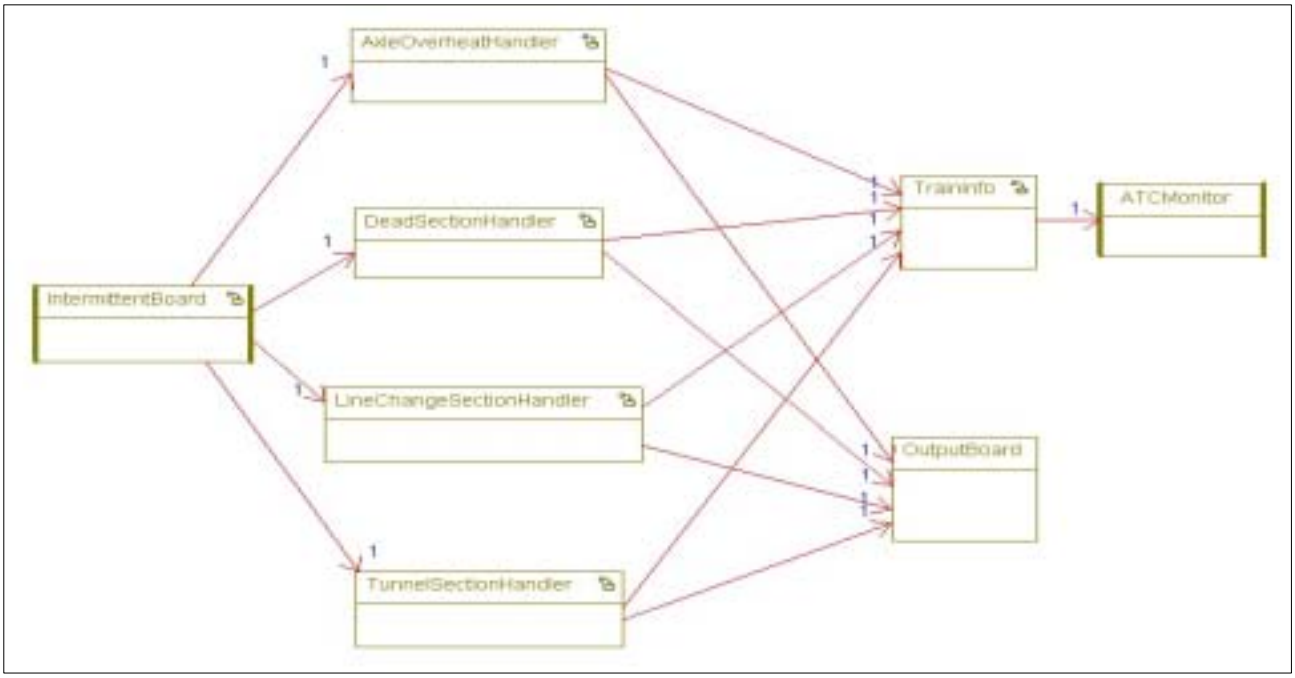
2.1.2 클래스 다이어그램

클래스 다이어그램은 클래스와 같은 정적인(static) 모델 구성요소들과 그들의 내용 및 상호관계를 표현하는 다이어그램이다. 여기서 클래스란, 동일한 기능(features), 제약사항(constraints) 및 의미(semantics)를 가지는 객체(object)들의 집합을 말한다.

[그림 2]는 ATC 차상 소프트웨어 클래스 다이어그램의 예를 보여준다. 사각형은 각각의 클래스를 표현하며, 사각형들 사이의 화살표는 클래스들 사이의 연관(association) 관계를 표현한다. 연관 관계가 맺어진 클래스들은, 연관 관계의 방향에 따라, 메시지 호출(message call)을 할 수 있다. 다시 말해, 화살표가 향하고 있는 클래스의 메쏘드(method)를 호출할 수 있다.

그림에서 다루고 있는 내용은, 불연속 보드를 통해 수신된 여러 이벤트들(사구간 진출입, 터널구간 진출입, 선로절환구간 진출입, Axle과열 검지)을 각각의 핸들러를 이용하여 처리하는 것이다. 추출된 클래스들에는 불연속 정보를 받아들이는 IntermittentBoard 클래스, 각각의 이벤트에 대한 Handler 클래스들, 열차 및 주행 상태 값을 관리하는 TrainInfo 클래스, 출력을 담당하는 OutputBoard 클래스, 운전자 모니터로의 출력을 담당하는 ATCMonitor 클래스가 있다.

화살표의 끝부분에 표시된 숫자는, 해당 연관 관계를 통해 얼마나 많은 개체들과 관련을 맺는가를 표시한다. 또한, 클래스를 표시하는 사각형의 우측 상단에 있는 아이콘은, 해당 클래스가 <2.1.3>절에 설명되는 상태 머신 다이어그램을 가지고 있다는 표시이다.



[그림 2] ATC 차상 소프트웨어 Class 다이어그램의 예

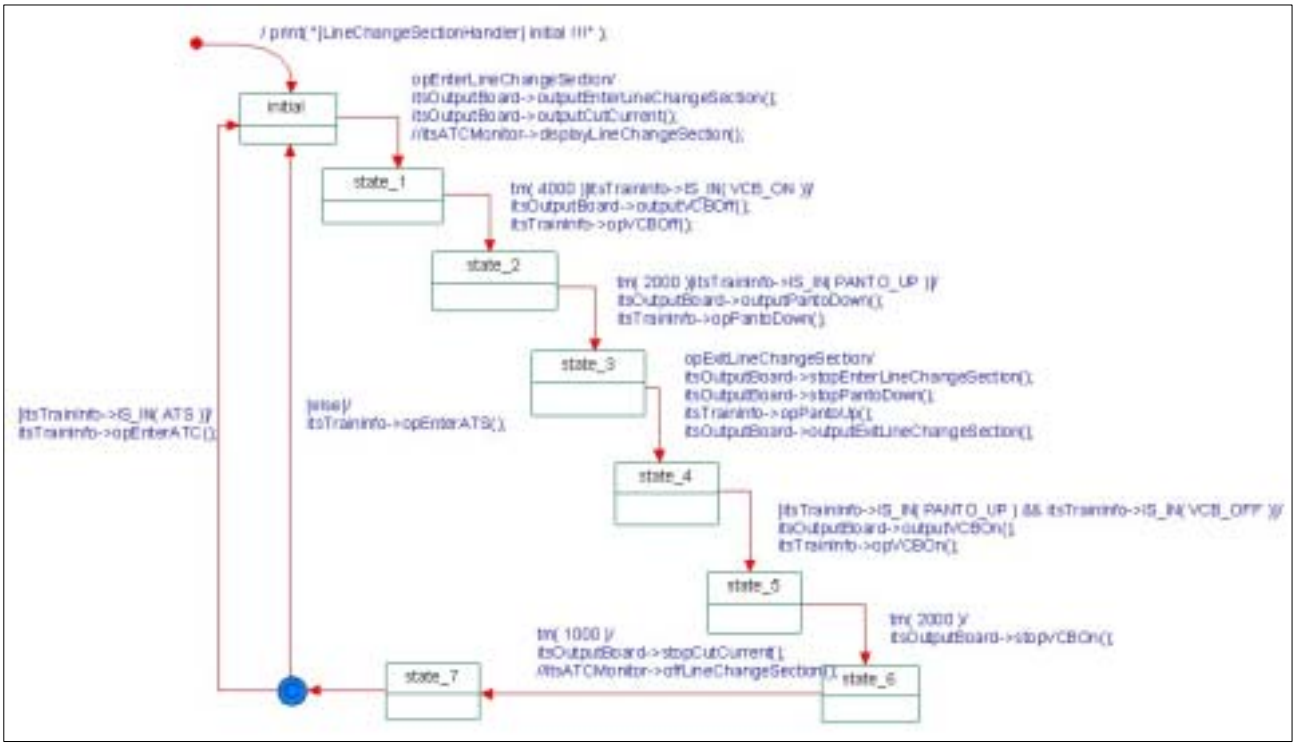
2.1.3 상태 머신 다이어그램

상태 머신 다이어그램은, 유한 상태-전이(finite state-transition) 시스템을 통하여 이산 행위(discrete behavior)를 묘사하는 다이어그램이다. 여기서 상태 머신이란, 객체 또는 상호작용(interaction)이 자신의 활동 중에 거치게 되는 상태들의 시퀀스를 기술하는 것으로서, 사건(event)과 그에 대한 반응 및 행동(action)들이 표시된다. 상태 머신 다이어그램은 시스템의 반응적(reactive) 특성을 표현하는 중요한 역할을 담당하며, Rhapsody 도구에서 소스코드 생성의 기반이 된다.

[그림 3]은 ATC 차상 소프트웨어 상태 머신 다이어그램의 예를 보여준다. 둥근 모서리를 가진 사각형은 각각의 상태를 나타내고, 화살표는 상태 전이를 나타낸다. 화살표에 붙어있는 라벨은 "**사건[조건]/행동**"의 형태를 가지며, 이는 "**조건**이 만족되는 상황에서 **사건**이 발생하면, **행동**을 취한다"는 의미이다. 사건, 조건 및 행동은 항상 필수적인 것은 아니다. 예를 들어, 조건 없이 어떤 사건의 발생에 의해서만 상태 전이가 일어날 수도 있고, 반대로 사건의 발생이 없이 어떤 조건만 만족되면 상태 전이가 일어날 수도 있다. 또한, 상태가 전이되면서 어떠한 행동도 취하지 않을 수도 있다.

그림은 선로절환구간 진출입을 다루는 LineChangeSectionHandler 클래스의 상태 머신 다이어그램을 표현하고 있으며, 다이어그램에 표현된 상태 전이는, CSS(Control System Specification) 문서를 기반으로 한다. 최초 initial상태에서 opEnterLineChangeSection이라는 사건이 발생하면, OutputBoard쪽으로 선로절환구간 진입신호(itsOutputBoard->outputEnterLineChangeSection()) 및 견인력 차단신호(itsOutputBoard->outputCutCurrent())를 출력하면서 state_1으로 전이한다. state_1에서는 정해진 시간동안 대기한 후(tm(4000)), VCB 투입상태 조건에서 ([itsTrainInfo->IS_IN(VCB_ON)]) VCB 차단제어 신호(itsOutputBoard->outputVCBOff())를 출력한다.

그림에서  아이콘은 해당 객체가 생성될 때 최초로 갖게 되는 초기 상태를 가리키는 지시자이며,  아이콘은 조건 분기를 나타내는 것으로서, 현재 열차의 제어모드가 ATS 모드라면([itsTrainInfo->IS_IN(ATS)]) ATC 모드로 전환하고(itsTrainInfo->opEnterATC()), 그렇지 않다면([else]) ATS 모드로 전환하게 된다(itsTrainInfo->opEnterATS()).



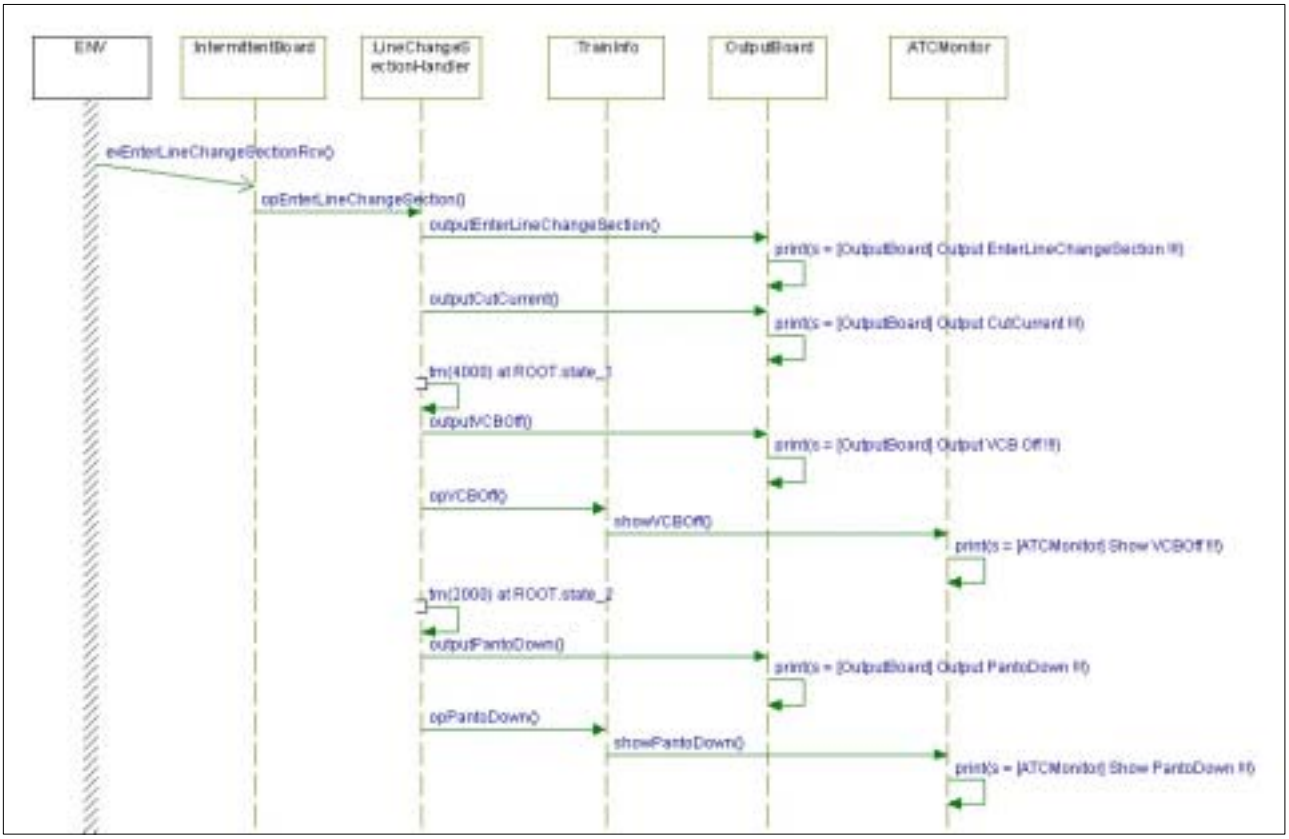
[그림 3] ATC 차상 소프트웨어 State Machine 다이어그램의 예

2.1.4 시퀀스 다이어그램

시퀀스 다이어그램은 객체들 사이에 교환되는 메시지들의 시퀀스를 시간 순서에 따라 표현하는 다이어그램이다. 시퀀스 다이어그램은, 초기 모델링 시에 시스템을 구성하는 요소들 사이의 상호작용을 명세할 때 유용하며, 상태 머신 다이어그램과 함께 시스템의 기능적 요구사항 만족 여부를 검증할 때도 유용한 다이어그램이다. 상태 머신 다이어그램을 이용하면 특정 시점에서 상태의 전이를 확인해 볼 수 있는 반면, 시퀀스 다이어그램을 이용하면 적절한 순서에 따라 기능들이 수행되는지를 확인해 볼 수 있다. Rhapsody 도구에서는 분석/설계자가 시퀀스 다이어그램을 직접 작성할 수도 있지만, 모델을 시뮬레이션할 때 자동으로 다이어그램이 생성되므로, 초기에 모델링하면서 구상했던 시퀀스 다이어그램과 시뮬레이션 결과로 자동 생성된 다이어그램을 비교하여 오류를 찾아낼 수 있다.

[그림 4]는 ATC 차상 소프트웨어 Sequence 다이어그램의 예를 보여준다. 그림에서 시간의 흐름은 위에서 아래로 향하며, 사각형은 객체를 의미하고 화살표는 객체 사이의 메시지 호출을 의미한다. 기울어진 화살표(evEnterChangeSectionRcv())는 비동기적 이벤트를 나타내며, 내부적인 메시지 큐를 이용하여 전달되므로 메시지 호출 지연이 발생할 수 있다. 반면 수평 화살표(opEnterLineChangeSection())는 동기적으로 작동하는 트리거 연산(triggered operation)을 나타내며, 동작 특성이 결정적(deterministic)이다. 빗금으로 표시된 시간선을 가진 'ENV' 객체는 시스템 경계 외부를 나타낸다.

<2.1.3>절에서 예를 들었던 선로절환구간 진출입이 일어날 때, 관련된 객체들 사이의 상호작용이 표현되어 있다. 불연속 보드(IntermittentBoard)를 통해, 외부로부터 선로절환구간 진입 신호가 수신되면, 불연속 보드는 선로절환구간 진출입을 관장하는 LineChangeSectionHandler 객체에게 정해진 동작을 수행하도록 명령한다(opEnterLineChangeSection()). LineChangeSectionHandler 객체는 자신의 상태 머신 다이어그램에 정의된 바와 같이, TrainInfo 객체 및 OutputBoard 객체에게 필요한 동작을 수행시킨다.

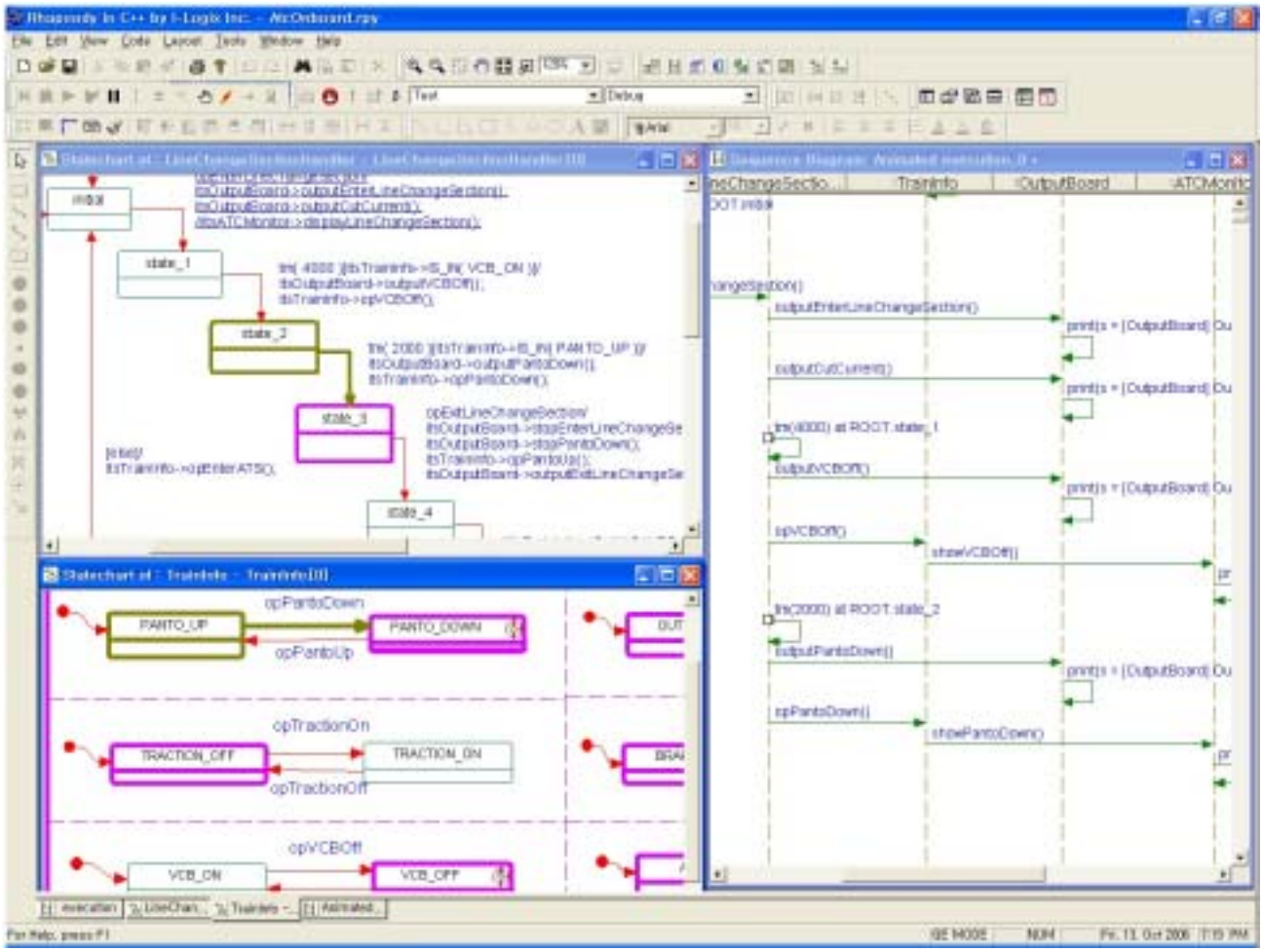


[그림 4] ATC 차상 소프트웨어 Sequence 다이어그램의 예

2.2 CASE 도구를 이용한 모델 검증

복잡한 동작 특성을 가진 시스템의 경우, 시스템을 구성하는 여러 요소들이 상호작용하면서 실시간으로 상태가 변경되므로, 기능을 테스트하고 검증하는 작업이 쉽지 않다. Rhapsody 도구는, 분석/설계자가 모델링한 모델을 시뮬레이션해 볼 수 있는 애니메이션 기능을 제공한다. 이를 통해, 관련된 구성 요소들의 상태 변화를 동시에 관찰할 수 있으며, 시간 순서에 따른 동작 시퀀스와 조합하여 모델의 정합성을 검증해 볼 수 있다.

[그림 5]에서는 <2.1.3>절 및 <2.1.4>절에서 예를 들었던, 선로절환구간 진출입과 관련된 모델 검증 작업을 보여준다. 애니메이션을 시작하면, LineChangeSectionHandler 객체와 TrainInfo 객체의 상태 머신 다이어그램이 활성화되면서, 현재 상태가 분홍색으로 밝게 표시된다. 또한, 선로절환구간 진출입과 관련된 객체들을 포함한 시퀀스 다이어그램의 경우, 시간 흐름에 따라 실제 수행되는 메시지 호출이 자동으로 표시된다. 따라서, 검증하고자 하는 기능을 촉발시키는 사건을 인위적으로 주입시킨 후, 변화되는 다이어그램들을 시각적으로 모니터링 함으로써, CSS(Control System Specification) 문서에 기술된 각각의 항목에 대하여 모델의 동작을 검증할 수 있었다.



[그림 5] CASE 도구를 이용한 모델 검증

3. 결론

본 논문에서는 UML기반의 CASE 도구를 이용하여 ATC 차상 소프트웨어를 모델링하고 검증해 보았다. 그 결과, UML의 여러 다이어그램들을 이용하여 복잡한 행위특성을 표현할 수 있었으며, CASE 도구의 기능을 활용한 모델 시뮬레이션을 통해 기능적 요구사항의 만족 여부를 검증해 볼 수 있었다. 적절하게 도구를 활용할 경우, UML기반의 객체지향 방법론은 ATC 차상 소프트웨어를 모델링 하는 효과적인 방법이 될 수 있음을 알 수 있었다.

신뢰성 및 안전성 관련 규격에서 구조적 방법론을 선호하는 이유 중 하나는, 구조적 방법론이 오랜 시간동안 사용되고 검증되어 온 것에 비해, 상대적으로 객체지향 방법론에 대한 검증이 충분히 이루어지지 않은 점일 것이다. 그러나, 객체지향에 기반을 둔 여러 방법론 및 도구들이 안정화되어 감에 따라, 실시간 내장형 소프트웨어 영역에서도 점차 그 영향력이 확대될 것으로 예상되며, 향후에는, UML을 확장한 SysML을 기반으로 ATC시스템 전반에 대한 모델링 작업을 수행할 예정이다.

참 고 문 헌

1. "UML 2.0 Superstructure Specification", OMG, 2005
2. Bruce P. Douglass, "Doing Hard Time", Addison Wesley, 1999
3. "Control System Specification - Rev. C", 한국철도차량, 2000
4. "Rhapsody in C++ V6.0 - Essential Tool Training", I-Logix, 2005
5. "IEC 62279 - Software for railway control and protection systems", IEC, 2002