# 한국형 고속전철의 열차 관리제어기의 설계 및 검증 방법

# Design and Verification Method for the Train Supervisory Controller of the Korean High-Speed Train

유 승 필, 이 두 용
(Seung Pil Yoo and Doo Yong Lee)

**Abstract** : This paper presents a design procedure for the supervisory controller(SC) of the high-speed train car. The proposed SC manages control logic and scenario of the overall train system, and monitors input/output signals among the sub-controllers in the system. The roles and functions of each sub-control system are defined, and the interfaces among the control systems are developed. Train control system is modeled with automata, and the model is implemented into a program using a computer-aided software engineering(CASE) tool, ObjectGEODE. The designed SC is verified and evaluated by using simulation. The SC is shown to successfully perform the designed functions without any errors.

**Keywords** : supervisory control, discrete event system, high-speed train, computer aided software engineering

## I. 서론

고속전철은 빠른 속도로 인하여 운전자의 수동 제어가 어려우므로, 열차의 자동 모니터링 및 진단과 제어를 위한 제어시스템이 필요하다. 제어시스템은 추진제어기, 제동제어기, 프리세트(preset) 속도제어기, ATC(Automatic Train Control)기 등의 많은 하위 제어기들로 구성되며, 이 제어기들을 관리하여 전체 시스템의 작동이 원활하게 되도록 하는 관리제어기가 필요하다.

관리제어(supervisory control) 이론은 Ramadge와 Wonham이 이산 사건 시스템(Discrete Event System)의 모델링 도구들 중 하나인 오토마타(Automata)를 이용하여 정형화하였다[1]. 이들은 관리제어기의 존재 여부와 구조 등의 제반 문제를 정성적, 논리적으로 표현하고 해결하는 방법을 제시하였으며, 플랜트(Plant)와 제어기를 각각 오토마타 모델로 표현하여 두 모델을 결합한 시스템에 대해 해석했다. 이후 관리제어에 대한 연구는 다방면으로 진행되어왔으나, 기존 연구의 대부분이 간단한 시스템들에 대해서 적용되었으며, 복잡한 시스템에 대한 적용연구는 거의 되어있지 않다.

본 논문에서는 복잡한 실제 시스템에 적용된 연구로서 한국형 고속전철의 관리제어기를 다룬다. 현재 국책연구과제로 개발하고 있는 한국형 고속전철을 위한 관리제어기의 설계 및 구현을 위해, 관리제어기의 기능을 정의하고, 고속전철 제어시스템을 오토마타 모델로 표현하고, 컴퓨터 원용 소프트웨어공학(Computer-Aided Soft-ware Engineering: CASE) 도구를 이용하여 내부로직을 프로그래밍하고 검증하였다. 이 논문에서는 관리제어기 설계의 초기단계를 논하며, 오류가 없는 상황에서의 관리제어를 고려하여 설계하였다. 고속전철 시스템은 많은 센서와 작동장치 및 제어기를 가지고 있으며, 안전성이 중요한 실시간 시스템이다. 프로그래밍 및 시뮬레이션은 대형시스템을 모델링하고 시뮬레이션 하는데 유용한 컴퓨터 원용 소프트웨어공학 도구인 Object GEODE를 사용했다. 기존 고속전철 시스템에 관한 정보로서 프랑스로부터 도입된 TGV자료[2]-[6]를 분석하였다.

접수일자 : 2000. 4. 8., 수정완료 : 2000. 8. 8.
유승필, 이두용 : 한국과학기술원 기계공학과

## II. 고속전철 제어시스템의 구조 및 기능

### 1. 프랑스 고속전철의 제어시스템

프랑스의 고속전철인 TGV의 열차 탑재용 제어시스템인 OBCS(On Board Computer System)는 백여개의 기능함수를 갖고 있다. OBCS의 기능은 크게 전기추진 및 제동, 공기제동, 보조전원장치, 그리고 운전 및 안전장치 등에 관한 기능으로 분류된다[4]. 운전 및 안전장치에 관한 기능은 작동이나 운전에 도움을 주는 기능, 열차를 준비하고 유지하는데 필요한 기능, 그리고 안전장치와 연관된 기능으로 분류된다. 각 기능은 세부기기들로 구성되며, 정보 검지 및 처리와 저장, 그리고 작동기(actuator) 실행 및 비정상상태에서의 정보처리 및 경고와 유지보수에 관한 내용을 포함한다.

OBCS는 각 주요 장치의 감시 및 제어가 주목적인 MPU(Main Processing Unit)[2]와 보조 장치의 감시 및 제어 역할과 MPU 고장시에 역할을 대신하는 APU (Auxiliary Processing Unit)[3] 등의 유닛(unit)으로 구성된다.

### 2. 한국형 고속전철의 제어시스템을 위한 기능 정의 방법

고속전철 제어시스템은 열차 제어, 고장 검지 및 진단, 운전 정보 현시, 데이터 수집 및 분석, 유지 보수 등의 작업을 해야한다. 열차 제어는 운전실 제어, 열차 운행제어, 승객 서비스 제어 등의 역할이 있다. 운전실 제어는 운전실 선택 제어, 경계(vigilance) 제어 등으로 분류되며, 열차 운행제어는 마스터 제어기의 신호처리제어, ATC (Automatic Train Control) 동작모드 제어, 샌딩(sanding) 제어, 세척기 모드제어 등으로, 승객서비스 제어는 승객 정보장치 제어, 출입문 제어, 조명 제어 등으로 분류된다. 이외에 판토그래프(pantograph) 제어, 차단기(circuit breaker) 제어 등이 있다. 이와 같이 각 제어

기능은 그 하부에 세부 제어기능들로 이루어진다.

제어시스템의 기능 정의 방법은 첫째, 프랑스 고속전철 제어시스템의 기능 분석결과를 이용하여 각 기능에 필요한 기본적인 신호 및 절차를 확인한다. 이 기능 분석결과에는 각 기능의 목적과 하는 일, 그리고 필요한 입력과 나가는 출력, 세부적인 작동 등이 대략적으로 나와 있다. 둘째, 신호 흐름을 참조하여 각 유닛(unit)들의 연계를 확인한다. 이 신호 흐름은 유닛끼리 주고받는 신호를 확인할 수 있으며, 서로의 연관관계를 유추할 수 있다. 셋째, 열차의 운행 시나리오를 이용하여 운행순서를 확인하여 각 기능들이 시나리오에 따라가도록 설계한다. 넷째, 지금까지 합의된 한국형 고속전철 제어방안 초안을 참조하여 제어 시스템의 구성 및 각 하위 제어 시스템의 역할을 확인한다.

현재까지 설계된 관리제어기의 기능은 차량의 배터리 제어, 열차의 선/후두부 제어, 판토그래프 제어, 견인명령치 신호전송, 모니터링(monitoring) 제어, 주 공기압축기 제어, 샌딩(sanding)과 윤활 제어, 예비점검(preconditioning) 및 출발전 검사(pre-departure test) 등이다.

3. 관리제어기의 구조

관리제어기는 그림 1과 같이 관리제어기 전체를 총괄적으로 다루는 관리제어(supervisory control) 기능과 관리제어기에서 처리해야 할 여러 기능이 있고, 각 기능 안에는 세부기능이 있으며, 세부기능 안에 실제로 수행하는 역할을 정의한다. 예로, 예비점검(preconditioning) 기능은 열차의 시동을 걸기 전에 열차에 전원을 넣고 준비시키는 일을 한다. 이 기능은 3 개의 세부 기능을 갖고 있다. 세부 기능 중 continuous monitoring 기능의 경우, 열차상태점검 등의 역할을 포함한다.

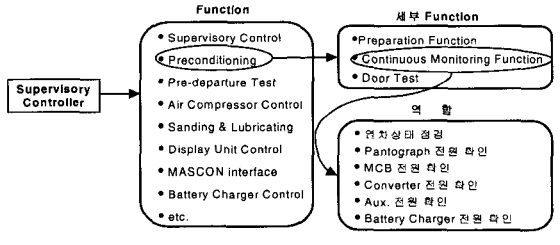



그림 1. 관리제어기의 계층.

Fig. 1. Hierarchy of supervisory controller.

현재까지 완성된 관리제어기는 관리제어기능을 포함하여 9개의 기능으로 구성되어있다. 관리제어기능은 하위 제어시스템들의 상태정보를 저장하며, 서로를 유기적으로 연결시켜주는 총괄관리의 역할을 한다. 예비점검기능은 예비점검 작업을 관장하며, 목적은 운전실로부터의 특정 명령이 없이 열차에 전원을 공급하는 등의 준비상태(standby mode)로 들어가는 것이다. 출발전 점검(pre-departure test) 기능은 출발전에 통상적으로 하는 시험들로 여러 가지 안전장치들을 시험하고 초기화한다. 이외에 공기압축기 제어(air compressor control) 기능, 배터리 제어 기능, 신호전송 기능, 샌딩과 윤활 기능, 모니터링 제어 기능, 그리고 앞의 8가지에 포함되지 않는 작업들을 처리하는 ETC 기능이 있다.

기능정의, 세부기능 및 역할에 기반하여 오토마타 모델을 설계하고, 그 모델을 이용하여 프로그래밍한 후, 시뮬레이션을 이용하여 검증한다.

## III. 관리제어기 설계 및 프로그래밍의 예

III장에서는 열차제어 시스템의 오토마타 모델링 및 관리제어기의 설계와 프로그램 과정을 설명한다. 열차가 출발하기 이전에는 시동 및 열차점검에 관련된 부분이 있고, 도착 후에는 세차 및 고장부분 점검이 있다. 열차의 동작 시나리오에 따르면 시동 시에 예비점검(preconditioning)을 하고, 출발 전 검사를 한 후, 출발장소로 이동한다. 승객들이 승차한 후 출발하고, 교량 및 터널과 철로조건에 따른 속도의 가감속을 하며 도착역에 도착한다. 승객이 하차하면 세차소로 이동해 세차를 하고, 고장부분을 점검한 후 차량기지로 이동한다. 예비점검 과정에는 시동을 걸면 판토그래프(pantograph)를 올려 전력이 들어오는지 확인한 후, 각 전력계통 기계가 제대로 작동하는지 확인하고, 들어온 전력으로 문(door)이 작동하는지 확인한다. 이런 형식으로 된 시나리오에 따라 열차가 제대로 작동하기 위한 관리제어기를 설계하며, 그 예로써 관리제어기의 기능 중 비교적 간단한 객차의 외부 문 개폐시스템에 관한 기능의 설계 및 프로그램 과정을 서술한다. 외부 문 개폐시스템은 그림 2와 같이 고속전철의 객차에 있는 외부문과 문의 직접적인 제어를 위한 객차제어유닛(vehicle control unit), 그리고 관리제어기로 구성된다.

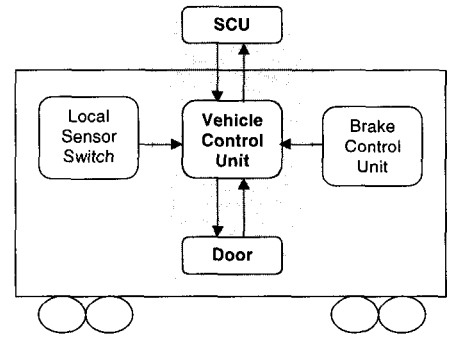



그림 2. 고속전철의 외부 문 개폐 시스템.

Fig. 2. Door system of high-speed train.

1. 오토마타 모델

방대하고 복잡한 관리제어기의 프로그램을 작성할 때는 디버깅(debugging)에 많은 시간과 노력이 들어가며, 논리적 오류 및 시스템적인 관점에서의 오류를 찾기 어렵고, 고속전철에서는 특히 안전성이 중요하므로 오류가 없도록 검증해야한다. 시각적인 모델은 논리적 오류 및 시스템적인 관점에서의 오류를 찾기에 용이하기 때문에 제어로직 설계에 오토마타(automata)를 이용한다[7]. 전체 제어시스템과 관리제어기는 너무 많은 제어기와 센서, 구동기들을 가지고 있기 때문에 하나의 오토마타 모델로 표현하는 것은 대단히 어렵다. 그러므로 각 플랜트

및 관리제어기의 기능 일부를 각각 오토마타 모델로 표현한다.

일반적으로 관리제어를 위한 오토마타 모델은 제어기 모델과 플랜트 모델의 두 부분으로 나눌 수 있다. 외부 문 시스템의 경우, 관리제어기 모델과 객차제어유닛모델로 나누었다. 우선 다음과 같이 주요기능을 정의한다. 외부 문 개폐시스템에 관련한 관리제어기의 주요기능은 열기 및 닫기 명령을 객차제어유닛으로 인가하고, 출입문의 상태를 모니터링하며, 객차제어유닛의 기능은 명령에 따라 출입문의 열기 및 닫기 작업을 직접적으로 제어하고, 현재의 상태를 모니터링하여 관리제어기에 알려주는 것이다. 주요 기능을 정의한 후에는 그 기능을 실행하는데 필요한 작업을 결정하며, 그 작업에 필요한 입력과 출력(그림 3), 그리고 정보를 처리하는 과정 등에 대하여 각 작업별로 정리한다. 각 작업은 그림 1처럼 여러 계층으로 구성될 수도 있으며, 정보처리과정은 가장 하위 작업에서 다룬다.

그 다음으로 상태를 정의한다. 제어기 모델의 상태 정의는 제어로직에 따르고, 플랜트의 상태정의는 플랜트의 상황에 따라 한다. 객차제어유닛 모델의 상태는 그림 4처럼 외부로 나타나는 상황을 각각 표현한다. 상태를 정의한 후, 그림 5처럼 상태간의 관계를 정의한다. 이와 같이 정의한 객차제어유닛 모델의 상태 정의는 표 1과 같다. 입력과 출력에 관계된 신호 정의는 입출력 신호를

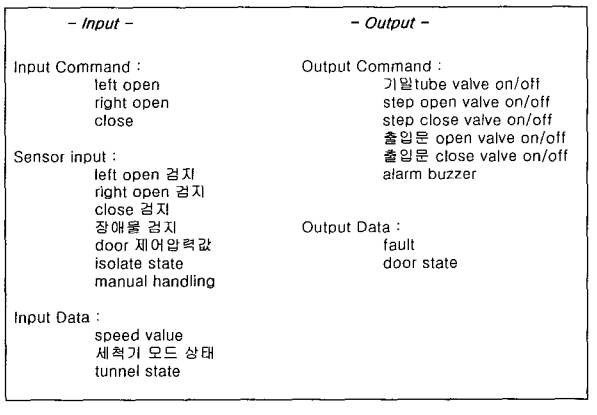



그림 3. 외부 문 플랜트의 입력과 출력신호.
Fig. 3. Input and output signals of the door plant.

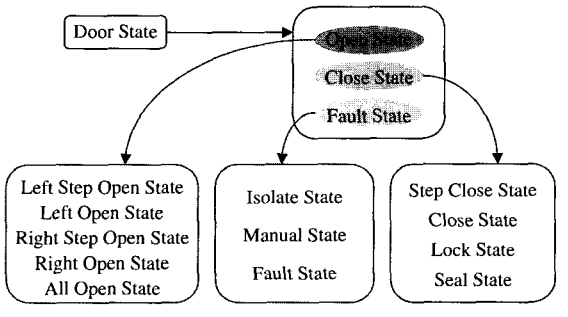



그림 4. 문 플랜트의 상태 분류.
Fig. 4. Classification of the door plant state.

위주로 필요한 입력과 나오는 결과에 대한 신호를 정의하며, 객차제어유닛 모델의 신호정의는 표 2와 같다.

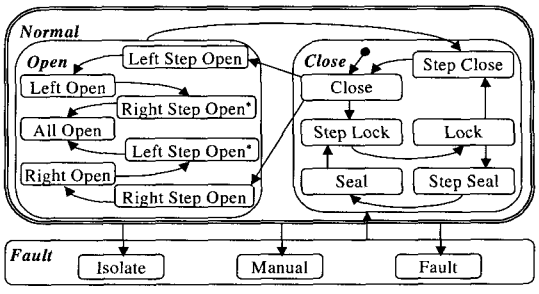



그림 5. 외부 문 플랜트의 상태 관계.
Fig. 5. Relation of the door plant states.

표 1. 외부 문 플랜트의 상태 정의.
Table 1. State definition of the door plant.

| State | Name | State | Name |
|---|---|---|---|
| $P_1$ | Open State | $P_{21}$ | Step Close State |
| $P_2$ | Close State | $P_{22}$ | Close State |
| $P_3$ | Fault State | $P_{23}$ | Step Lock State |
| $P_{11}$ | Left Step Open State | $P_{24}$ | Lock State |
| $P_{12}$ | Left Open State | $P_{25}$ | Step Seal State |
| $P_{13}$ | All Open State | $P_{26}$ | Seal State |
| $P_{14}$ | Right Open State | $P_{31}$ | Isolate State |
| $P_{15}$ | Right Step Open State | $P_{32}$ | Manual State |
| $P_*$ | Any State | $P_{33}$ | Fault State |

표 2. 외부 문 플랜트의 신호 정의.
Table 2. Signal definition of the door plant.

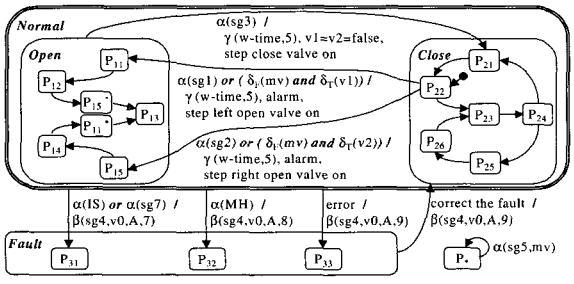
| Signal | Name | Sensor | Name |
|---|---|---|---|
| sg1 | left open command | LO | left door open 검지 |
| sg2 | right open command | RO | right door open 검지 |
| sg3 | close command | CL | door close 검지 |
| sg4 | door state | LCK | door lock 검지 |
| sg5 | train moving state | SL | door sealing 검지 |
| sg6 | washing mode state | OBS | obstacle detect |
| sg7 | isolate command | IS | manual isolation command |
| sg8 | tunnel state | MH | manual door handling |
| sg9 | high speed state | | |

door state = { 1: open, 2: close, 3: lock, 4: seal, 7: isolate, 8: manual, 9: fault }

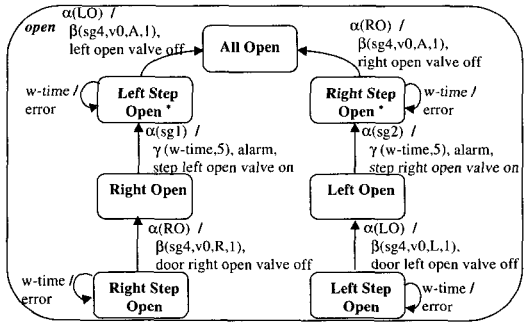
표 3. 사건 정의.
Table 3. Definition of events.

| 이벤트 | 이름 | 설명 | 예 |
|---|---|---|---|
| $\alpha(\cdot,\cdot)$ | 신호 입력(signal input) | 어떤 신호를 받아들이며, 신호는 값을 갖고 있을 수 있다. | $\alpha$(open), $\alpha$(speed,150), $\alpha$(stat,3,4,true)<br>open, speed, stat : 신호이름 |
| $\beta(\cdot,\cdot)$ | 신호 출력(signal output) | 신호 또는 명령을 내보낸다. 입력과 같이 값을 가질 수 있다. | $\beta$(close), $\beta$(Inservice,false), $\beta$(sand,2,true,false,true)<br>close, Inservice, sand :신호이름 |
| $\pi(\cdot,\cdot)$ | 타이머(timer) | 정해진 시간 후 특정 이벤트를 일으킨다. | $\pi$(r,5), $\pi$(time,20)<br>r, time : 발생시킬 이벤트<br>5, 20 : 정한 시간, 단위:초(sec.) |
| $\delta_T(\cdot)$<br>$\delta_F(\cdot)$ | True/False 조건 | 변수가 각각 true 또는 false일 경우에만 이벤트가 일어난다. | $\delta_T$(var), $\delta_F$(fault)<br>var, fault : 변수 이름 |

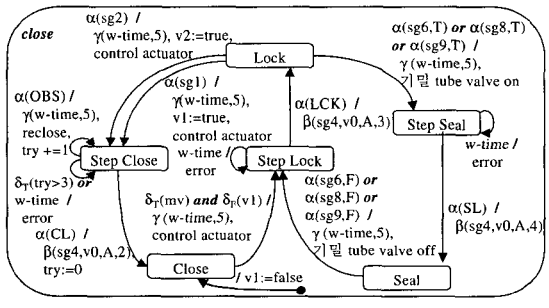
오토마타 모델에서 시스템의 사건(event)은 외부로부터 받는 입력 사건과 외부로 내보내는 출력 사건으로 나눈다. 그리고 모델 표현을 쉽게 하기 위하여 두 가지 사건을 더 정의한다. 그것은 현재 시스템이 가지고 있는 정보의 값을 확인하기 위한 참/거짓을 판별하는 사건과, 시스템 내에서의 반응속도 등 시간을 확인하기 위한 타이머(timer) 사건이다. 이 네 가지 사건으로 오토마타 모델의 모든 사건을 표현하며, 표 3과 같이 정의한다. 정의한 4종류의 사건은 열차제어시스템에서 사용하는 모든 종류의 신호에 대하여 고려하여 오토마타 모델에서 일어날 수 있는 사건들로 분류한 것이다.




(1) 오토마타 모델




(2) open 상태의 확대 그림




(3) close 상태의 확대 그림.

그림 6. 외부 문 플랜트의 오토마타 모델.
Fig. 6. Automata model of door plant.

이와 같이 정의된 상태와 입출력신호 및 이벤트를 이용하여 오토마타 모델을 완성한다. 이때 작업 결정 시에 만들어 놓은 정보처리과정을 참고하여 모델링하며, 상태간의 관계 도면에서 상태의 변화를 일으키는 사건을 정의한다. 사건은 변화를 일으키는 사건과 변화하면서 다른 사건을 일어나게 하는 두 가지가 있으며, 입력사건과 출력사건은 /으로 구별한다. 예를 들어 'a/b'는 a사건이 일어나면 상태가 변하면서, b라는 사건을 일으키는 것이다. 이렇게 만든 외부 문과 VCU의 오토마타 모델은 그림 6처럼 된다. $P_{11}^*$과 $P_{15}^*$ 상태는 $P_{11}$과 $P_{15}$와 이름은 같지만 다른 상태를 나타낸다.

그림 6의 모델을 수식적으로 표현하면 7개 요소(7 tuple)로 형성되는 집합으로 표현되며 간단하게 예를 들면 다음과 같다.

$$G=\{E, X, \Gamma, f, x_0, Y, g\}$$

$E$ : 이벤트 집합 (event set)
$X$ : 상태 집합 (state space)
$\Gamma$ : 그 상태에서 실행가능한 이벤트 집합 (set of enabled events)
$f$ : 상태 전이함수 (state transition function), $f: X\times E\rightarrow X$
$x_0$ : 초기상태 (initial state)
$Y$ : 출력이벤트 집합 (output set)
$g$ : 출력 함수 (output function), g: $X\times E\rightarrow Y$

$$E=\{\ \alpha(sg1),\alpha(sg2),\alpha(sg3),\alpha(sg6,T),\alpha(sg6,F),\ \alpha(LO),\delta_T(v_1),\delta_F(v_1),w-time,error,\cdots\ \}$$

$$E_c=\{\alpha(sg1),\alpha(sg2),\alpha(sg3)\}$$

$$X=\{P_{11},P_{11}^*,P_{12},P_{13},P_{14},P_{15},P_{15}^*,P_{21},\cdots,P_{33}\}$$

$$Y=\{error,\gamma(w-time,5),\beta(sg4,v_0,A,1),\beta(sg4,v_0,R,1),\cdots\}$$

$$x_0=P_{22}$$

$$\Gamma(P_{11})=\{\ w-time,\alpha(sg3),\alpha(sg7),\alpha(LO),\alpha(IS),\alpha(MH),\ \alpha(sg5,mv),error\}$$

$$f(P_{11},w-time)=P_{11},\quad g(P_{11},w-time)=error$$

$$f(P_{11},\alpha(sg7))=P_{31},\quad g(P_{11},\alpha(sg7))=\beta(sg4,v_0,A,7)$$

열차제어 시스템에서는 위와 같이 매우 많은 이벤트 및 상태전이함수와 출력함수가 있기 때문에 수식표현을 모두 보여주기 곤란하며, 시스템의 상태 및 천이관계를 시각적으로 알 수 있는 오토마타의 특성을 이용한다.

외부 문 개폐를 위한 오토마타 모델의 관리제어로직은 그림 7과 같다. 상태와 변수는 표 4에 정의했다. 관리제어기 모델의 상태는 제어 로직에 따라 정의하며, 기능은 앞에 앞에 언급하였다. 외부문 개폐시스템은 하나의 관리제어기 모델이 여러 차량의 외부문 플랜트 모델을 제어하는 것이다. 관리제어기의 기능은 전체 열차의

표 4. 외부 문 오토마타 모델의 기호 정의.
Table 4. Definition of symbols.

| 종류 | 기호 | 정의 |
|---|---|---|
| state | $S_1$ | Ready |
| | $S_2$ | Left doors opening |
| | $S_3$ | Right doors opening |
| | $S_4$ | All doors closing |
| | $S_5$ | Fault state of left door opening |
| | $S_6$ | Fault state of right doors opening |
| | $S_7$ | Fault state of doors closing |
| | $S_*$ | any state |
| variable | v1 | moving (boolean) |
| | v2 | total car number (integer) |
| | v3 | state of right door (integer array) |
| | v4 | state of left door (integer array) |
| | c | dummy (integer) |
| | t | dummy (integer) |

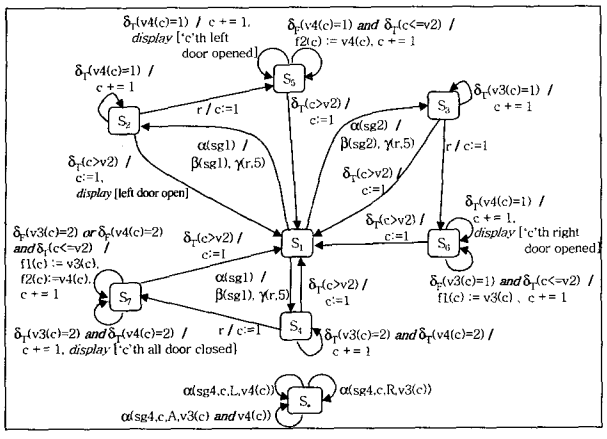


그림 7. 외부 문 개폐를 위한 제어기 모델.
Fig. 7. Automata model of external door function.

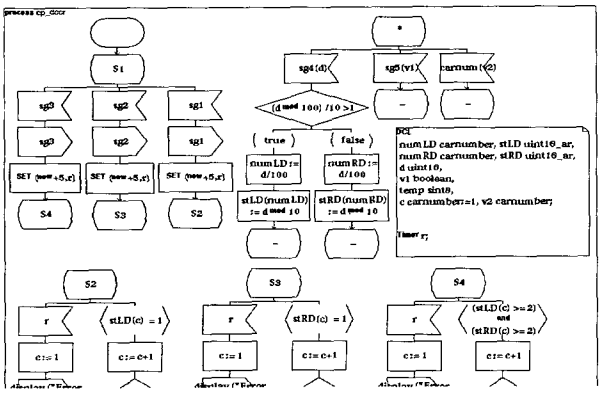


그림 8. 관리제어기 외부문 기능의 SDL 프로그램 일부.
Fig. 8. Part of SDL program of function, external door.

문들을 열고 닫는 명령을 내보내고, 각 차량의 문의 상태들을 확인하는 등의 출입문에 관련된 것으로, 기본동작을 위한 4개의 상태와 고장상태를 위한 3개의 상태로 구성되어 있다. 초기상태는 상태 1이다. 변수 1이 F값을 갖는 상태에서 신호 4가 들어오면 상태 2로, 신호 2가 들어오면 상태 3으로 간다. 't:=1'은 변수 t에 값 1을 대입하는 것이다. 5초 후에 이벤트 r을 일으키는 타이머를 사용하였다. 시간 내에 작동하지 않는 것을 오류상황으로 인식하는 것이다. 고장이 일어나면 고장에 관한 변수 행렬인 f1이나 f2에, 문제가 일어난 차량의 오류상태를 기록한다. 변수 c는 차량넘버를 뜻한다.

그림 7의 관리제어기 모델과 그림 6의 플랜트 모델을 연결하면, 그림 2의 출입문 개폐 시스템의 모델을 완성할 수 있다.

2. 프로그래밍 및 시뮬레이션

관리제어기를 프로그램으로 구현하기 위해 Object GEODE라는 CASE 도구를 이용했다. 이 도구에서 하나의 시스템은 블록과 프로세스로 구성된다[8][9]. 여기서 프로세스는 오토마타에 기반하는 언어인 SDL (Specification and Description Language)[10]로 표현하며, 오토마타 모델을 이용하여 프로그래밍한다. SDL은 표 5과 같은 기호 등을 이용하여 표현하는 형식언어(Formal Language)이다. 그림 7의 오토마타 모델은 그림 8의 SDL 프로그램으로 변환될 수 있다. 그림 8의 ⬭는 프로그램이 시작될 때나 프로세스가 새로 생성되었을 때, 프로세스의 시작위치를 지정한다. 따라서 그림 8의 프로그램은 처음에 ⬭의 선을 따라 S1상태로 가게된다. S1 상태에서는 신호입력을 기다리고 있는데, sg1 신호가 입력되면 입력된 신호를 각 VCU로 보내고, 5초 후에 동작하는 타이머(r)를 작동시킨 뒤 S2상태로 간다. sg2신호나 sg3신호가 들어올 경우, sg1의 입력경우처럼 각 선을 따라서 작동을 한 후에 각각 S3, S4상태로 간다. S2 상태에서는 타이머로 생기는 사건 r이 일어나거나, 행렬변수 stLD의 값이 1이 되는 사건이 발생할 때까지 기다리며, 사건이 발생하면 선을 따라 작업을 수행한다. *상태는 S*상태로서 어떤 상태에서든 주어진 사건인 sg4, sg5, 또는 carnum신호가 입력되면, 작업을 수행하고 이전 상태로 돌아간다. sg5나 carnum신호가 입력됐을 때는 신호가 가져온 값을 변수 v1, v2에 각각 저장한다. carnum은 전체 차량 대수를 알리는데 사용하는 신호다. sg4 신호가 입력됐을 때는 값을 변수 d에 저장한다. d값은 4자리 10진수로서, 앞의 2자리는 차량번호, 세 번째는 문의 좌우위치, 네 번째는 문의 상태에 관한 값을 가지고 있다. (d mod 100)/10은 세 번째 수의 값인 문의 좌우위치를 확인하는 것으로서 1보다 크면 true 선을 따라 좌측 문에 관련한 각 변수에 값을 저장하고, 크지 않으면 false 선을 따라 각 우측 문에 관련한 각 변수에 값을 저장하고 나서, 이전에 있던 원래 상태로 돌아간다. 이와 같은 방법으로 상태간의 전이관계를 SDL로 프로그램하며, 오토마타의 상태/전이 관계를 이용하여 SDL로 쉽게 전환될 수 있다.

설계 검증은 ObjectGEODE에서 지원하는 시뮬레이션 기능을 이용했다[11]. 열차의 시동에서부터 출발역과 도착역을 거쳐 차량기지로 들어갈 때까지의 시나리오에 대한 시뮬레이션 동안 각 제어기능을 담당하는 프로세스들의 상태를 확인하였으며, 교착상태(dead lock) 및 순환상태(livelock)에 빠지지 않았다. 그리고 시뮬레이션의 결

표 5. SDL의 기호.
Table 5. Symbols of SDL.

| 기호 | 이름 | 기호 | 이름 |
|---|---|---|---|
| ⬭ | 상태(state) | ⊟ | 이전상태 (previous state) |
| ◇ | 비교/판단 | ⬭ | process의 시작위치 |
| set | 타이머 시작 (timer setting) | reset | 타이머 해제 (timer reset) |
| (*) | 모든 상태 (any state) | ▭ | Task |
| ⬎ | 신호 입력 (input signal) | ⬏ | 신호 출력 (output signal) |
| ⟨ ⟩ | 조건 (condition) | | |

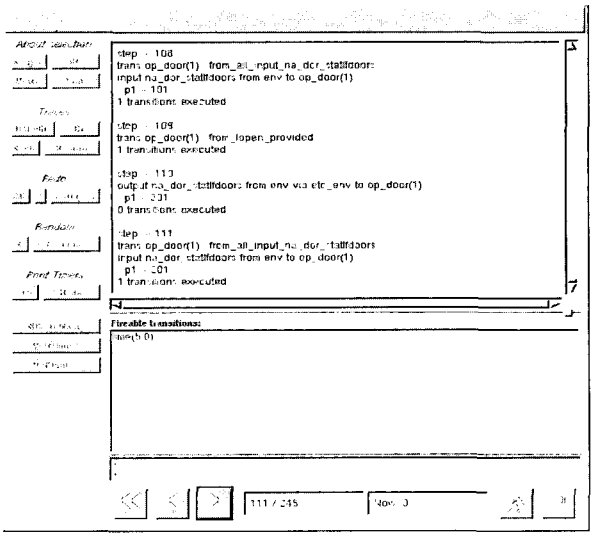


그림 9. ObjectGEODE의 시뮬레이션 작업화면.

Fig. 9. Simulation view of ObjectGEODE.

과로 나오는 MSC(Message Sequence Chart)와 천이상태가 저장된 시뮬레이션 시나리오를 이용하여 검증했다. 그림 9는 시뮬레이션 중의 화면이다. 사용한 CASE 도구의 보기 기능을 이용하여 각 프로세스의 상태와 변수 또는 신호의 값 등을 시뮬레이션 중에 실시간으로 볼 수 있다. 이 기능들을 이용한 시뮬레이션을 통하여 외부 문 개폐 시스템의 관리제어기가 정상적으로 작동함을 확인했다. 외부 문 시스템이 고장났을 때는 운전자 및 승무원에게 알리며, 오류에 관한 변수에 문이 고장난 객차번호를 저장한다.

## IV. 열차 시스템 관리제어기

Ⅲ장에서 설명한 방법을 적용하여 외부 문 개폐 이외의 모든 관리제어기 기능들의 오토마타 모델을 구성하고 전체 모델을 프로그램으로 구현하였다. 관리제어기 전체에 대한 프로그램은 시스템, 블록, 프로세스를 각각 관리제어기의 기능(function), 세부기능(sub-function), 역할(role)로 대응시켜 프로그램 했다. 열차 시스템 관리제어기를 하나의 시스템으로 하고, 관리제어블록을 중심으로 다른 블록들이 정보 또는 명령을 주고받는다. 각 블록 아래에는 세부기능에 해당하는 프로세스들이 있고, 프로세스 안의 내부로직을 프로그램 했다.

SDL 프로그램의 가장 상위계층인 시스템은 관리제어기 자체로서 그림 10과 같으며, 관리제어기의 기능부분인 블록(block) 9개로 구성되어있다. 각 블록은 서로에게 필요한 신호를 주고 받거나, 또는 필요한 신호를 외부와 직접 주고 받는다. 신호를 주고 받는 선의 이름은 각 블록이나 외부(environment) 등 서로 연결된 것들의 이름을 이용했고, [ ] 안의 문자는 각각 주고 받는 신호의 이름들이다. 신호의 자리에 괄호로 묶여 있는 것은 신호 리스트(signal list)이다. 신호 리스트는 여러 신호의 집합이다. 그림 11은 그림 10에 있는 preconditioning 블록을 확대한 것으로, 블록이 프로세스로 구성되어 있음을 보여준다. 프로세스도 블록과 같이 정보를 교류하는 신호

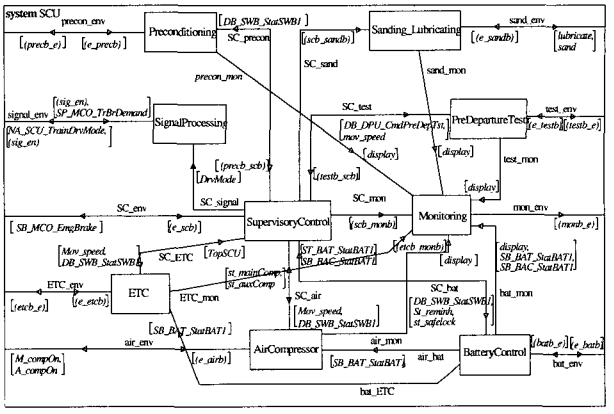



그림 10. ObjectGEODE로 프로그램한 관리제어기의 시스템 구조.

Fig. 10. System-level-view of the supervisory controller in ObjectGEODE.

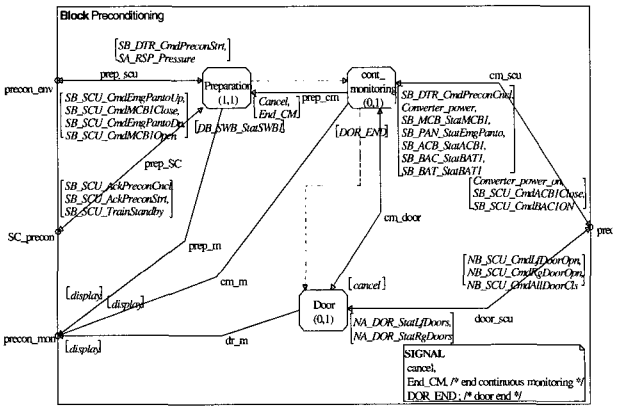



그림 11. 관리제어기의 예비점검 블록.

Fig. 11. Preconditioning block of supervisory controller.

연결선을 갖는다. Preparation 프로세스의 신호 연결선 prep_scu의 경우 precon_scu와 연결되는데, 이 precon_scu는 그림 10에 있는 신호 연결선의 이름이다. 각 프로세스는 동일한 것이 여러 개가 동시에 존재할 수 있으며, 프로세스의 이름 위치에 '프로세스 이름(최저갯수, 최대갯수)'로 표현한다. 점선인 화살표는 프로세스의 생성을 뜻한다. 열차 시스템 관리제어기는 현재까지 9개의 블록과 16개의 프로세스가 완성되었다.

설계한 관리제어기의 검증을 위하여, 출발역에서 도착역까지의 가상 시나리오를 수립하고, 이에 따른 시뮬레이션을 수행하였다. 시뮬레이션은 CASE 도구인 Object GEODE에서 제공하는 시뮬레이션 기능을 이용했으며, 시뮬레이션 동안 실시간으로 각 프로세스의 상태 및 변수와 정보들의 값을 확인하고 교착상태를 체크했다. 시뮬레이션 중에 교착상태 등의 오류상황에 빠지지 않았으며, 도착역까지 무사히 도착함을 확인할 수 있었다. 그림 12는 시뮬레이션의 결과물 중 하나인 예비점검 과정에 대한 MSC로 열차 예비점검의 시작부터 각 프로세스들이 주고받은 신호 입출력, 프로세스 생성 및 종료 등을

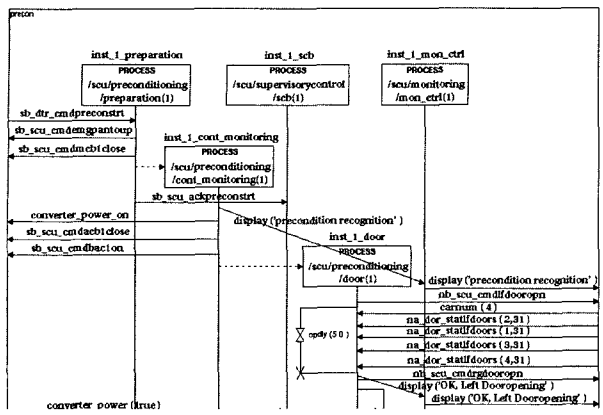



그림 12. 시뮬레이션 중 예비점검과정에 대한 MSC의 일부.

Fig. 12. Part of message sequence chart for preconditioning.

순서대로 보여준다. 이 MSC의 정보를 이용하여, 시뮬레이션 중에 프로세스들이 주고 받은 입출력 신호와 그 값을 확인했다.

## V. 결론

본 논문에서는 복잡한 대형 시스템인 한국형 고속전철의 열차 제어시스템을 총괄 관리하는 관리제어기의 설계 및 검증 방법을 논하였다. 기존 고속전철의 제어시스템을 파악하기 위하여, 프랑스 고속전철의 제어시스템인 OBCS를 분석하였고, 그 자료를 바탕으로 한국형 고속전철의 제어시스템 기능을 정의하였다. 그리고 제어시스템을 관리하는 관리제어기의 기능 및 각 기능의 세부내역을 정의하고, 기능에 따른 입출력신호와 신호연계(signal interface)를 구성했다. 현재까지 9개의 큰 기능이 완성되었다.

관리제어기의 각 기능과 기능에 관련한 하부 시스템을 오토마타를 사용하여 모델화 하였다. 모델의 상태 및 상태관계는 플랜트의 외부상태 및 제어로직에 따라 정의하였다. 전체 시스템에 사용하는 신호를 분석하여 대형 시스템의 오토마타 모델에 필요한 4가지 사건을 정의하였고, 각 기능에 대해 필요한 신호를 정의하였다. 기능정의 및 하부시스템에 관한 상태정의와 신호정의를 이용하여 각각의 오토마타 모델을 만들고, 두 모델을 결합하여 전체 제어기의 오토마타 모델을 완성하였다. 열차시스템은 순차적인 제어(sequence logic control)를 하므로, 플랜트의 오토마타 모델로부터 관리제어 오토마타 모델을 만들지 않고 제어체계에 따른 관리제어 모델과 플랜트에 따른 오토마타 모델을 각각 만들어 연결하는 방법을 사용하였다. 오토마타의 특성을 이용함으로써 관리제어기의 기능 설계시 수정과 확장이 용이해졌고, 오토마타 모델을 이용하여, 서로 다른 프로세스간의 상태 변화를 동기화하는 연결상의 오류와 같은 시스템 수준의 오류 및 상태천이관계를 잘못 설정하여 일어나는 교착상태, 몇 개의 상태만을 오가며 다른 상태로 천이되지 못하는 순환상태(livelock) 등의 오류상황을 제거했다.

관리제어기 설계에 오토마타 모델을 이용하고, 오토마타를 기본으로 만들어진 SDL을 이용하여 프로그램을 만드는 보다 쉬운 방법을 제시했다. 설계된 관리제어기는 컴퓨터 원용 소프트웨어공학 도구(CASE tool)인 Object GEODE를 이용하여 프로그램 했다. 관리제어기는 주요기능을 수행하는 9개의 블록(block)들로 구성됐고, 각 블록은 프로세스(process)들로 구성되었으며, 프로세스의 내부는 오토마타 모델을 이용하여 SDL로 프로그램했다. 이렇게 구현된 관리제어기 프로그램은 Object GEODE에서 지원하는 시뮬레이션 기능과 현재까지 수립된 운행 시나리오를 적용하여 성공적으로 검증되었다.

## 참고문헌

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optimiz.*, vol. 25, no. 1, pp. 206-230, Jan., 1987.

[2] GEC Alsthom, *TGV-K CAB Software Requirement Specification Document*, 1996.

[3] GEC Alsthom, *TGV-K Descriptive and Operational Manual: APU01*, 1996.

[4] GEC Alsthom, *TGV-K Software Specification: Connecting the Equipment*, 1996.

[5] GEC Alsthom, *TGV-K Control System Specification* (1)~(3).

[6] GEC Alsthom, *TGV-K Data Processing Specification* (1)~(8).

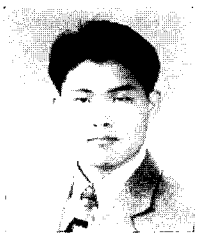
[7] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*, Aksen associates, 1993.

[8] Verilog, *ObjectGEODE Training Manual*, 1998.

[9] Verilog, *ObjectGEODE Tutorial*, 1997.

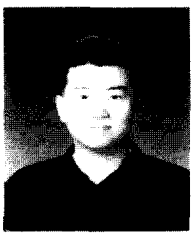
[10] F. Belina, D. Hogrefe, and A. Sarma, *SDL with Applications from Protocol Specification*, Englewood Cliffs: Prentice Hall, 1991.

[11] Verilog, *ObjectGEODE SDL Simulator-Reference Manual*, 1997.



**유 승 필**

1974년 12월 17일생. 1996년 동국대학교 기계공학과, 공학사. 1999년 한국과학기술원 기계공학과, 공학석사. 1999년~현재 동대학원 박사과정. 관심분야는 관리제어, 능동제어, 이산이벤트 시스템 제어.



**이 두 용**

1985년 서울대학교 제어계측공학과, 공학사. 1987년 Rensselaer Polytechnic Institute, M.S. 1993년 동 대학교, Ph.D. 1993년~1994년 동 대학교, Postdoctoral Research Associate. 1994년~현재 한국과학기술원 기계공학과 교수. 관심분야는 로보틱스, 생산시스템 최적화 및 제어, 가상현실, 이산이벤트시스템 제어.