Z 모델 기반 명세 기법

신지훈*, 윤용기**, 최진영*
*고려대학교 컴퓨터학과
** 한국철도기술연구원 열차제어통신연구실
e-mail : {jeehoon, choi}@formal.korea.ac.kr, ykyoon@krri.re.kr

Z model-based Specification

Jeehoon Shin*, Jin-Young Choi *
* Dept. of Computer Science, Korea University
** Train Control System Research Team of the Korea Railroad Research Institute(KRRI)

요 약

오늘날 많은 컴퓨터 시스템들이 우리의 삶에 사용이 되고 있다. 컴퓨터 시스템의 사용이 증가함에 따라 오류로 인한 문제들도 커지고 있다. 특히 열차 제어 시스템, 원자력 제어 시스템과 같은 안전필수 시스템에 오류가 발생하면 재산적, 인명적인 피해를 초래할 수 있다. 정형명세 언어 중 하나인 Z notation 은 안전필수 시스템 명세등에 많이 사용이 되고 있으며 Z 를 사용하여 명세하는 방법들이 제시되고 있다. 본 논문에서는 Correctness by Construction 에 기반하여 Z 를 사용하여 시스템을 명세하는 방법을 제안하며 안전필수 시스템인 DCM 을 적용하여 명세한다.

1. 서론

정보혁명이 시작된 이후 컴퓨터 시스템은 우리생활의 곳곳에서 사용되며 인간에게 많은 편리함을 제공하고 있다. 하지만 높은 기능성에 대한 요구사항 때문에 컴퓨터 시스템이 복잡해지고 규모가 커짐에 따라 시스템 내부의 치명적인 오류의 잠재성도 증가했다. 열차 제어 시스템, 원자력 제어 시스템 등 오류로 인한 기능실패로 재산적, 인명적인 피해를 초래할 수 있는 시스템을 안전필수시스템이라고 한다. 이러한 시스템들의 오류 발생 원인으로 자연어 명세서로 인한 모호함이 크다.

안전, 보안 필수 시스템과 같은 고 무결성 시스템 개발을 위해 영국의 소프트웨어 개발 업체인 Praxis 사는 오류의 근원인 결점들을 제거하고 예방하면서 시스템을 개발하는 방법인 Correctness by Construction[1] 개발 방법을 제시하였다.

본 논문에서는 정형명세언어 중 하나인 Z notation(이하 Z)[2]를 소개하고 Correctness by Construction 에 기반한 Z 명세 프로세스를 제안하며 이를 안전필수 시스템인 DCM 에 적용하여 명세한다.

본 논문의 구성은 다음과 같다. 2 장에서는 정형기법[3], Correctness by Construction 및 Z notation에 대해 소개하며 기존의 Z 명세 프로세스[4]에 대해 설명한다. 3 장에서는 Correctness by Construction을 기반하여 수정한 Z 명세 프로세스를 설명한다. 4 장에서 DCM 시스템을 위의 프로세스에 적용하여 명세한 후 본 논문을 마무리하며 향후 연구 방향을 제시한다.[1]

감사의 글 : 본 논문은 국토해양부가 출연하고 한국건설교통평가원에서 위탁시행한 철도종합안전기술개발사업의 결과입니다.

2. 배경 지식

2.1 정형기법

정형기법은 소프트웨어 시스템의 명세, 디자인, 검증 들을 위해 수학적 모델을 사용하는 기술 및 툴들의 집합을 의미한다. 수학적 엄밀함은 사용자에게 각 프로그램 개발 단계에서 모델들의 분석 및 검증을 가능케 하며 소프트웨어 명세에서 소프트웨어가 무엇을 하는지에 대한 정확하고 명료한 기술을 가능하게 한다.

2.2 Z notation 및 명세 프로세스

Z 는 정형명세언어 중 하나로써 집합론, 일차수리논리에 기반을 둔 언어이다. Z 는 시스템을 구조적으로 명세하기에 좋은 장점을 지니고 있다.

Z 명세에서는 아래의 3 가지 모델을 명세 한다.

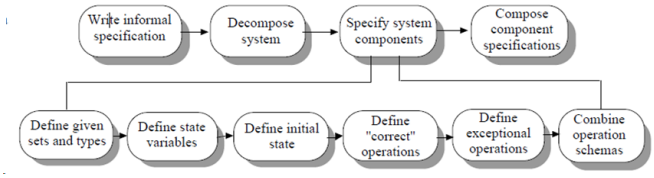
1) Static State Model
   시스템이 가지는 변수들과 이 변수들의 관계에 대해 기술하게 된다. 이 변수들의 관계는 시스템이 가질 수 있는 모든 상태에서 만족해야 하는 invariant를 의미한다.
2) Initial State Model
   시스템의 초기 상태를 기술한다.
3) Operation Model
   시스템의 상태를 바꾸거나 지속할 수 있도록 하는 오퍼레이션들에 대해 수행 전과 후의 상태의 관계에 대해 기술한다.

이렇게 명세 된 시스템들은 정리증명기법을 이용하여 속성을 검증할 수 있다.

정리증명기법이란 어떤 공리계와 관련 지어

시스템이 만족해야 할 속성이 그 계의 정리(참인 명제) 인지 아닌지를 결정하는 방법이다.

Z 를 이용하여 시스템을 명세하는 수많은 방법과 패턴들이 존재한다. 그 중에서도 많이 사용하는 명세 프로세스는 (그림 1)과 같다. [3]



(그림 1) Z 명세 프로세스

1) 시스템의 자연어 기능 명세서를 명세한다.
2) 시스템을 기능별로 서브 시스템들로 나눈다.
3) 나뉜 각 서브 시스템들을 명세한다.
   3-1) 상태 변수들 및 입• 출력 인자들을 위한 타입들을 정의한다.
   3-2) 상태 변수들을 정의한다.
   3-3) 서브 시스템의 초기 상태를 정의한다.
   3-4) 서브 시스템의 올바른 상태 전이를 표현하는 오퍼레이션을 정의한다.
   3-5) 3-4 의 오퍼레이션이 커버하지 않는 예외적인 상황에 대한 오퍼레이션들을 모두 정의한다.
   3-6) 3-4 와 3-5 를 조합하여 하나의 오퍼레이션을 정의한다.
4) 명세된 서브 시스템들을 조합한다.

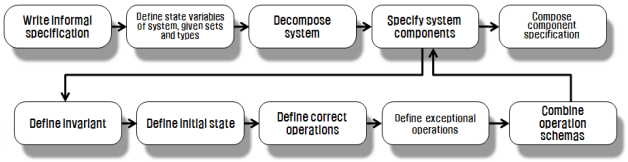
2.3 Correctness by Construction

Correctness by Construction 은 보안 및 안전 필수 시스템을 위해 높은 무결성을 보증하며, 근본적(radical)이며 효과적(effective)이고 또한 비용측면에서 효율이 높은(cost-effective) 소프트웨어 개발 방법이다. Correctness by Construction 은 이를 위해 정형 기법과 애자일 개발 방법을 결합하여 사용한다.

Correctness by Construction 에서는 개발 각 단계마다 검증을 하고 다음 담계로 넘어가기에 이를 쉽고 빨리 수행할 수 있도록 가능한 시스템을 간단하고 쉽게 이해가 되도록 명세, 코딩하도록 언급하고 있다.

3. Z 명세 프로세스 수정

기존의 방법에서는 자연어 명세(informal specification)에서 명세된 시스템을 서브 시스템들로 나누고 있다. 자연어 명세에서는 시스템의 상태를 정확히 정의해 놓지 않은 관계로 이 과정이 쉽지 않을 뿐더러 추후에 수정할 가능성이 크다. 그렇기에 (그림 2)의 프로세스에서는 전체 시스템의 상태를 정형화(formalizing) 후 관련된 상태 변수들을 기반으로 서브 시스템을 식별한다. 식별 한 후에 각 서브 시스템의 불변속성을 정의한다. 또한 상태 변수들을 정의하면서 해당 변수들의 타입들을 정의하도록 두개의 과정을 하나의 과정으로 축약하였다.



(그림 2) 수정된 Z 명세 프로세스

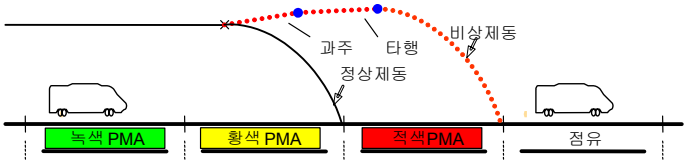
4. Case Study : DCM

4.1 DCM 소개

DCM(Distance Control Module)은 열차제어시스템에 내재된 서브 시스템으로써 열차간의 간격을 제어하고 DCM 범위 내에 존재하는 블록들을 관리하며 외부의 명령에 따라 각 블록을 폐쇄 및 개방하는 역할을 수행한다. DCM 의 기능 요구사항은 아래와 같이 크게 2 가지로 분류할 수 있다.

1) 열차 간격 제어

열차 간격 제어는 모든 블록에 Red, Yellow, Green 과 같이 PMA(허용이동권한)을 열차 이동이 이루어짐에 따라 블록 조건 및 상태를 고려하여 (그림 3)과 같이 지속적으로 설정한다.



정상제동 : 상용제동을 작동하여 황색PMA 구역 내에 정지(실선곡선)
비상제동 : 최대 출력의 추진상태로 오동작이 발생했을 시, 최소 보장 비상제동율로 열차가 적색PMA구역 내에 정지(점선곡선)

(그림 3) PMA 를 이용한 안전한 열차이격 확보

2) 블록 개방/폐쇄

블록 폐쇄는 궤도유지보수 공사, 열차분실 및 제한된 진로 보호가 필요한 경우 외부 시스템의 명령에 따라 수행한다. 블록 개방 역시 외부 시스템의 명령에 따라 해당 블록이 개방 가능한 경우 블록 개방을 수행한다.

4.2 DCM 명세 과정

1) 상태 식별

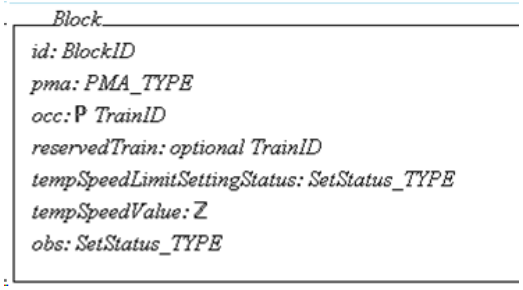
DCM 기능은 각 블록별로 처리가 되기에 블록 정보와 함께 전체 블록을 관리 할 수 있는 상태 변수가 필요하기에 아래와 같이 상태 변수와 타입을 정의한다.

[Block]
BlockID == Z

DCM
managingBlocks: BlockID ⇸ Block

(그림 4) 블록에 대한 상태변수 정의

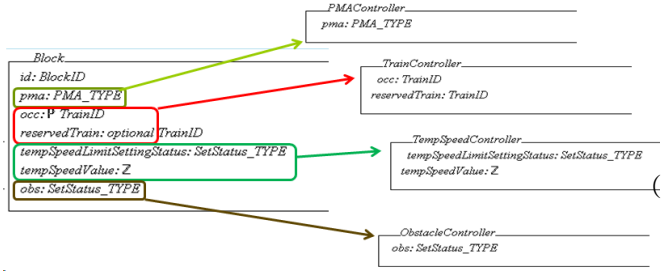
또한 블록 정보를 상세화하기 위해 (그림 4)에 정의되어 있는 Block 타입을 상세화하여 정의한다. (그림 5)



(그림 5) 블록 정보 상세화

2) 서브 시스템 식별

(그림 5)에서 상세화된 블록 정보를 기반으로 관련된 상태 변수들끼리 묶은 후 서브시스템을 식별하여 정의한다.(그림 6)

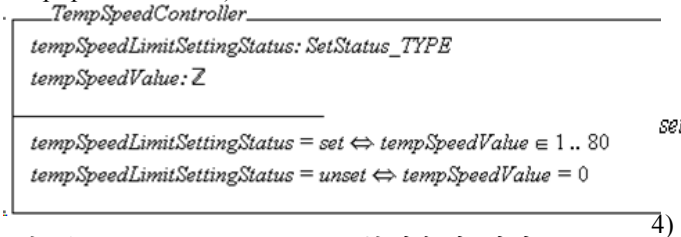


(그림 6) 서브 시스템 식별

3) 서브 시스템 명세

3-1) 불변속성(invariant) 정의

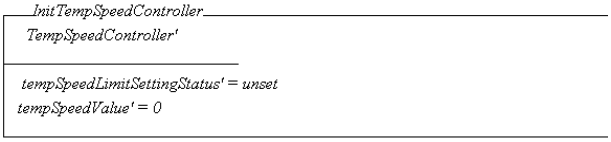
식별된 서브 시스템의 불변속성을 정의한다. (그림 7)의 임시속도제한명령을 처리하는 TempSpeedController 의 경우 임시속도제한이 설정된 경우 설정속도가 1 부터 80 사이여야 하며 (tempSpeedLimitSettingStatus = set ⇔ tempSpeedValue ∈ 1..80) 설정되지 않은 경우 설정속도가 0 이어야 함 (tempSpeedLimitSettingStatus = unset ⇔ tempSpeedValue = 0) 을 의미한다.



(그림 7) TempSpeedController 불변속성 정의

3-2) 초기상태 정의

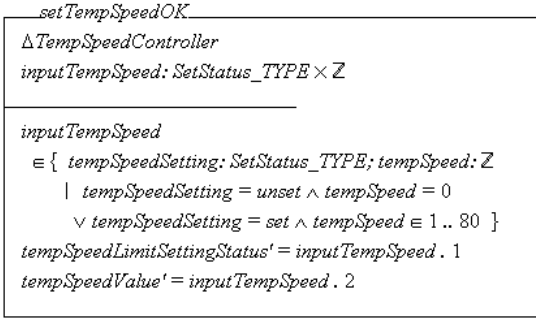
불변속성을 정의한 후 시스템의 초기상태를 정의한다. TempSpeedController 의 초기 상태는 (그림 8)과 같다.



(그림 8) TempSpeedController 초기상태 정의

3-3) 올바른 오퍼레이션 정의

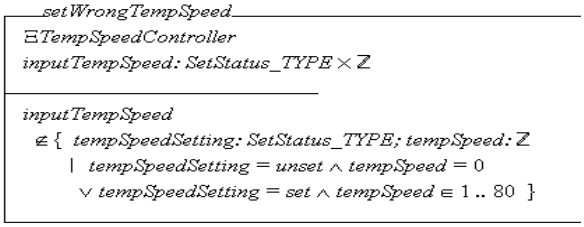
TempSpeedController 의 상태를 전이시키는 오퍼레이션을 정의하는 과정으로써 (그림 9)의 명세는 임시속도제한명령에 대한 입력이 들어왔을 시 입력이 올바르면 이를 수행하는 오퍼레이션을 나타낸다.



(그림 9) 올바른 입력에 대한 임시속도제한 처리

3-4) 예외 오퍼레이션 정의

항상 (그림 9)와 같이 입력이 올바를 수만은 없기 때문에 올바르지 못한 입력이 들어왔을 경우 역시 명세를 해주어야 한다.(그림 10)



(그림 10) 올바르지 못한 입력에 대한 임시속도제한 처리

3-5) 오퍼레이션 조합

3-3, 3-4 에서 정의된 올바른, 올바르지 못한 입력에 대한 오퍼레이션을 하나의 임시속도제한 처리 오퍼레이션으로 조합한다.(그림 11)

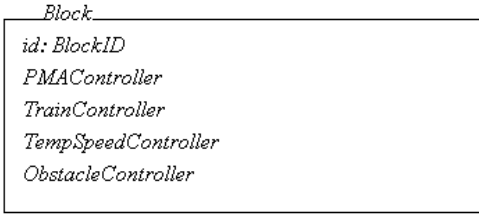
$$setTempSpeed \triangleq setTempSpeedOK \vee setWrongTempSpeed$$

(그림 11) 오퍼레이션 조합

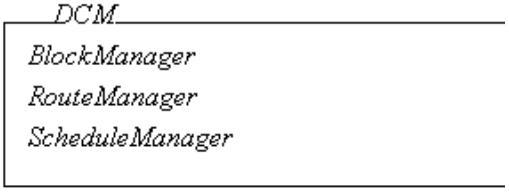
4) 시스템 조합

4-1) 서브 시스템들의 상태 조합

지금까지 명세한 서브 시스템들의 상태를 조합하여 상위 시스템의 상태를 정의한다. (그림 12)는 Block 상태 (그림 13)은 DCM 의 상태를 나타낸다.



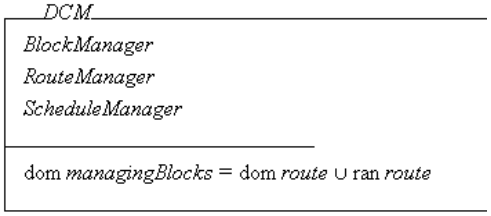
(그림 12) 조합을 통한 Block 상태 정의



(그림 13) 조합을 통한 DCM 상태 정의

4-2) 조합된 시스템 상태간의 불변속성 정의

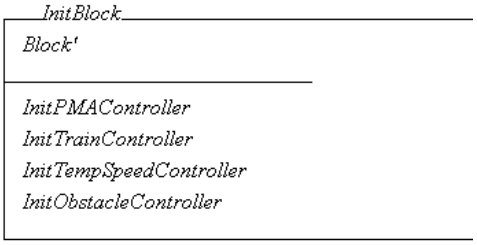
4-1)에서 조합된 시스템 상태간에 항상 만족되어야 할 관계가 존재할 시 이를 정의한다. (그림 14)는 BlockManager 내의 관리되는 블록들(managingBlocks)과 RouteManager 내의 선로(route)를 구성하는 블록들이 동일해야 함을 나타내고 있다.



(그림 14) 조합된 시스템 상태간의 불변속성 정의

4-3) 조합된 시스템의 초기상태 정의

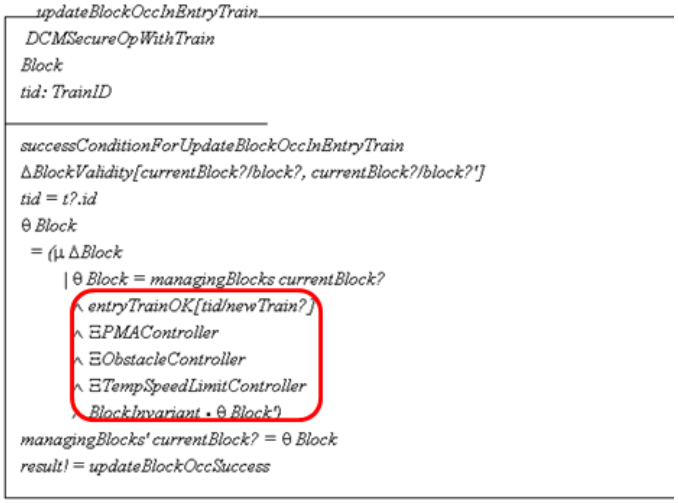
3-2)에서 정의된 각 서브시스템들의 초기상태와 4-1)에서 정의된 시스템 조합을 이용하여 시스템의 초기상태를 정의한다. (그림 15)에서는 4개의 서브시스템의 초기 상태를 이용하여 조합된 Block 의 초기 상태를 정의한다.



(그림 15) 조합된 시스템의 초기상태 정의

4-4) 조합된 시스템의 오퍼레이션 정의

조합된 DCM 시스템의 오퍼레이션을 정의하기 위해 각 서브시스템들의 정의된 오퍼레이션들을 조합한다. (그림 16)에서는 4 개의 서브시스템에서 3 개의 서브시스템의 상태가 변하지 않고 TrainController 에서만 enterTrainOk 를 수행하는 updateBlockOccInEntryTrain 을 정의하고 있다.



(그림 16) 조합된 시스템의 오퍼레이션 정의

5. 결론 및 향후연구

많은 연구에서 안전필수 및 보안필수 시스템 개발에 있어 Correctness by Construction 을 적용하여 낮은 오류율이란 결과들을 산출하고 있다[5].

따라서 본 논문에서는 Correctness by Construction 에 기반한 Z 명세 프로세스를 제안하고 안전필수 시스템인 DCM 을 대상으로 이를 적용하였다.

향후연구로 제안한 Z 명세 프로세스를 적용한 시스템을 개발해보고 생산성과 오류율을 측정해 봄으로써 낮은 오류율을 유지하면서 생산성이 더욱더 높아졌음을 보여야 할 것이다.

본 논문은 안전필수 시스템 개발에 있어 Correctness by Construction 을 적용하여 개발 시 또는 복잡한 시스템을 Z 를 이용하여 명세하고 검증하고자 할 경우에 대해 시스템 명세에 대한 가이드라인 역할을 할 것으로 생각한다.

참고문헌

[1] Roderick Chapman, "Correctness by construction: a manifesto for high integrity software", Proceedings of the 10th Australian workshop on Safety critical systems and software, 2006

[2] Mike Spivey, “The Z Notaion : a reference manual, 2 edition”, Prentice Hall, Englewood Cliffs ,1998

[3] Edmund M. Clarke, Jeannette M. Wing, “Formal methods: state of the art and future directions”, ACM Computing Surveys (CSUR), 1996

[4] Lex Bijlsma, “Model-based specification”, Inf. Process. Lett., vol 77, num 2-4, pp 77-84, 2001

[5] Anthony Hall, Roderick Chapman, “Correctness by Construction: Developing a Commercial Secure System”, IEEE Computer Society Press, 2002