안전 필수 시스템 설계를 위한 디자인 패턴†

이진호 *,O, 이혁 *, 윤용기 **, 최진영 *

고려대학교 컴퓨터학과 *, 한국철도기술연구원 **

{jhlee, hlee, choi}@formal.korea.ac.kr, ykyoon@krri.re.kr

Design Patterns for the Design of Safety-Critical Systems

Jeanho Lee *,O, Hyuk Lee *, Yongki Yoon **, Jinyoung Choi*

Dept. Computer Science, Korea University *, Korea Railway Research Institute **

요 약

열차제어 시스템은 안전필수 시스템으로 고신뢰성과 고품질이 요구되고 있다. 열차 제어 시스템 개발 과정에서 요구 사항 분석에 따라 결정되는 설계 사항의 경우 여러가지 요소들이 고려된다. 본 논문에서는 정형 요구 명세와 검증을 거쳐 제안된 설계 사항들을 하나의 패턴으로 정의하고, 시스템 개발 주기를 고려한 안전필수 시스템 설계단계에서 활용하는 프레임워크를 제안한다.

1. 서 론

열차 제어 시스템은 안전필수 시스템으로 고신뢰성과 고품질이 요구되며, 신뢰성과 품질을 높이기 위해 관련 업계의 국제 표준에서는 정형기법을 이용한 시스템 개발을 요구하고 있다[1]. 정형기법은 시스템을 명세하고 속성을 만족하는지 검사하는 기법으로 소프트웨어 개발 방법론에서 사용되고 있다[2]. 정형기법이 수학적 기반의 개념과 언어를 사용하기 때문에 시스템 요구분석이나 설계단계에서 정형기법을 적용할 때 정형 명세를 하기 어려운 점이 있다. 본 논문에서는 정형 요구 명세를 개발하기 위해 사용되는 설계 패턴을 제안하고 이를 적용하여 정형 요구 명세 개발한다. 본 논문에서는 정형 요구명세 설계 패턴을 사용하여 안전 필수 시스템인 열차제어시스템의 정형 요구 명세 개발 예제를 예시한다. 본 논문에서 정형기법 도구로 상태차트를 사용한다.

2. 관련 연구

2.1 정형 기법(formal methods)

정형기법은 수학적 기반의 개념과 언어를 도입해서 시스템을 명세하고 검증하는 기법과 도구를 가리키는 것으로, 신뢰성있는 시스템을 개발하기 위한 방법중의 하나이다[3]. 정형기법은 시스템 개발 생명주기 상에서 요구사항 분석과 상위수준의 설계단계에서 사용되고 있다[4]. 정형기법을 적용하여 작성한 요구 명세는 정형 명세 언어를 사용하여 자연어 요구 사항을 기술한다. 정형 요구 명세에 대해 정형 검증을 수행하여, 요구사항에 대한 검증속성 만족여부를 검사함으로써 일관성과 완전성을 검사할 수 있다[5].

2.2 상태차트(statechart)

상태 차트는 기존의 오토마타(automata) 상태기계에 동시성(concurrency), 계층구조(hierarchy), 상태 사이의 브로드캐스팅 (broadcasting) 통신 (communication) 기능을 추가한 형태의 정형 명세 언어이며, 대표적인 준정형(semi-formal) 명세 언어에 속한다[6]. 이벤트에 의해 시스템의 상태가 전이되며 기능이 수행되는 내장형 반응형 시스템의 행위를 기술하는데 적합하다.

2.3 디자인 패턴(design pattern)

디자인 패턴의 정의는 기존 환경에서 반복적으로 발생하는 문제들을 설명하고, 문제 해법의 핵심을 설명하는 것으로, 나중에 재사용을 할 수 있도록 하는 것이다[7]. 특히, 소프트웨어 공학에서는 공통의 설계구조에서 주요 요소들을 파악하여, 명칭을 정의하고 추상화함으로써 재사용가능한 객체지향 설계를 만드는데 사용되고 있다. 디자인 패턴은 크게 4가지 요소로 정의된다: 패턴이름, 문제, 해법, 결과. 디자인 패턴을 기술하는데 필요한 항목은 크게 13개 항목으로 구성된다: 패턴 이름과 분류, 의도, 별칭, 동기, 활용성, 구조, 참여자(participant), 협력방법(collaboration), 결과, 구현, 예제코드, 잘 알려진 사용예제, 관련 패턴.

† 본 논문은 국토해양부가 출연하고 한국건설교통평가원에서 시행한 철도종합안전기술개발사업의 결과입니다.

3.제안한 정형 명세 디자인 패턴

Gamma 가 소프트웨어 공학에 도입했던 디자인 패턴은 객체지향 소프트웨어의 개발에 적합하게 정의되어 있다. 반응형 시스템은 객체지향 소프트웨어와는 다른 특성을 가진다. 예를 들어, 시스템의 실행 도중에 클래스로부터 인스턴스 객체를 생성시키고, 객체 단위로 시스템의 기능을 수행하고 행위를 정의하는 형태의 설계는 반응형 시스템, 특히 안전필수 시스템의 경우에는 적합하지 않다. 본 논문에서는 내장형 반응형 시스템의 개발에 적합한 디자인 패턴을 제안한다. 기본적으로 4가지 구성요소를 갖추면서 객체지향적인 특성의 항목들을 제외한 나머지 항목들 중에서 6개 항목을 선택하였다. 디자인 패턴의 항목과 기술 내용은 표1과 같다: 패턴의 이름, 의도, 활용성, 구조, 결과, 사용 예제.

표 1 제안하는 디자인 패턴 항목과 기술 내용.

항목	기술 내용
1) 패턴의 이름(name)	특정 패턴의 명칭
2) 의도(intent)	해결하고자 하는 문제와 이슈, 또는 목적
3) 활용성(applicability)	적용 가능한 상황에 대한 기술
4) 구조(structure)	추상화된 패턴의 시각적 표현
5) 결과(consequence)	패턴을 사용한 결과가 갖는 장점과 단점
6) 사용 예제(example)	패턴을 가지고 실제 차트를 작성할 때, 작성단계를 보여주는 사례.

3.1 정형 명세 디자인 패턴의 작성 사례

정형 요구 명세 작성을 위해 디자인 패턴의 적용은 안전필수 시스템인 열차제어 시스템을 대상으로 하였다. 열차제어 시스템 중에서 핵심기능을 수행하는 모듈인 열차 간격 제어 모듈(DCM, distance control module)의 일부 기능에 대해 디자인 패턴을 적용하였다.

열차제어 시스템의 DCM은 크게 4개의 컴포넌트로 구성되어 있다: 시스템 전체를 관리하는 ATS(automatic train supervision), 열차 방호를 담당하는 CRD(control route distance), 열차 시뮬레이터인 ATP(automatic train protection), 신호 설비 시뮬레이터. CRD 컴포넌트는 2개의 하위 모듈로 구성된다: 열차 간격 제어를 위한 DCM, 열차의 진로 명령 처리를 위한 ICM(interlocking control module). DCM은 열차 간격 제어를 목적으로 5가지 기능을 수행한다: 블록의 개방과 폐쇄, 임시 속도 제한 명령 처리, 열차 위치 확인, 열차 이동방향 감시, 열차 간격 제어.

본 논문에서는 DCM의 5가지 기능 중에서 '열차 위치 확인' 기능에 대해 디자인 패턴을 적용하였다. '열차 위치 확인' 기능은 DCM이 열차(ATP)로부터 열차의 이동 정보를 수신하여, 열차의 위치를 확인하는 목적이 있다. 제안한 디자인 패턴을 적용한 결과는 다음과 같다:

1) 패턴의 이름

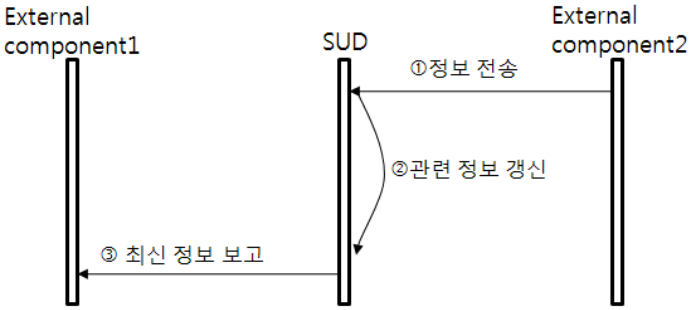
수신 정보의 업데이트 패턴

2) 의도

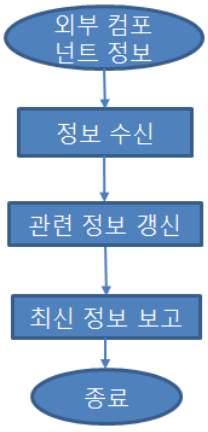
어떤 외부 구성요소로부터 정보를 수신하여, 수신정보와 관련 정보를 갱신하고, 갱신된 정보를 다른 외부 구성요소에게 보고하는 경우에 갱신 작업과 보고 절차에 대한 제어가 필요하다.

3) 활용성

어떤 외부 구성요소로부터 주기적으로 정보를 수신하고, SUD(system under development)는 수신한 정보와 관련된 정보를 업데이트시킨다. 갱신된 최신 정보를 상위 외부 구성요소에게 보고한다. 정보 수신, 관련 정보 갱신과 최신 정보 보고의 작업이 인터럽트없이 수행되어야 한다.



4) 구조



- 외부 컴포넌트는 주기적으로 정보를 전송한다.
- 수신자인 SUD는 단지 관련 정보만을 갱신시키며, 수신여부(acknowledgement)를 송신자인 외부

컴포넌트에게 다시 알려주지 않는다.

- SUD는 갱신된 최신 정보를 다른 외부 컴포넌트에게 보고한다. 보고한 후에 전송한 정보에 대한 수신자인 외부 컴포넌트로부터 수신여부를 기다리지 않고 해당 작업을 종료한다.

5) 결과

- 외부 컴포넌트로부터 정보를 수신하는 즉시, 내부적으로 관리하는 관련 정보를 갱신할 수 있다.
- 관련 정보 갱신과 보고할 최신 정보 사이에 연관성을 고려해야 한다. (갱신되지만 보고되지 않는 부분도 존재한다.)
- 최신 정보 보고 작업이 중요한 의미를 가지기 때문에 작업의 정확성이 보장되어야 한다.

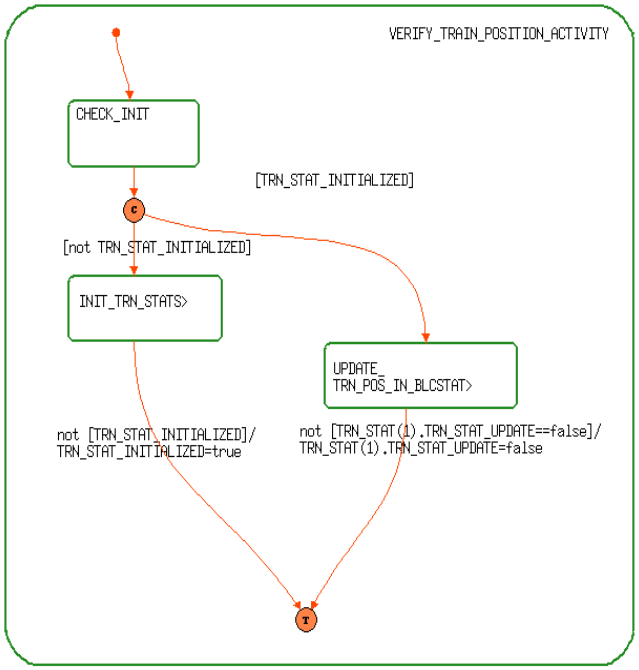
6) 사용 예제

- 열차 위치 확인을 실행하는 정형 요구 명세

- 자연어 요구사항

DCM은 ATP와의 양방향 통신을 통해 시스템 전반에 걸친 열차의 위치를 확인한다.
열차의 이동은 블록 단위로 이루어지며, DCM은 ATP로부터 전송된 정보 중 열차 위치 정보를 식별하여, 현재 열차의 위치를 확인하고 결정한다.
ATP는 운행중인 열차의 위치 정보를 DCM으로 지속적으로 전송한다.
열차가 블록으로 진입하면 ATP가 현재 위치한 블록을 DCM에게 보고한다.
DCM은 해당 블록을 점유된 블록으로 인식하고 ATS에게 전송한다.

- 상태차트



- 정형 요구 명세

ATP로부터 열차 정보가 수신되면, 열차가 위치한 블록을 점유 상태로 설정하고 ATS에게 보고한 후에 종료한다.
만약 열차 정보가 초기화되어 있지 않으면, 열차 정보 업데이트가 가능하도록, 열차와의 연결 상태를 'true'로 설정하고, 열차 정보 초기화 상태도 'true'로 설정하고 종료한다.
열차 정보가 초기화되어 있으면, 열차가 위치한 블록에 점유상태를 설정하고, ATS에게 보고한 후에 종료한다.

4.정형 명세 디자인 패턴을 이용한 정형 요구 명세 개발 프레임워크

Gamma의 디자인 패턴 도입 이후, 기존의 디자인 패턴 기법들은 설계 단계에 중점적으로 초점이 맞춰져 있었기 때문에, 시스템 중에서 각각의 컴포넌트마다 할당된 기능성을 달성시키는 데 목적이 있었다[8]. 시스템 개발 주기를 고려했을 때, 각 개발 단계마다 추상화 수준에 따라 결정되어야 할 시스템의 설계 사항들이 존재하며, 이런 결정사항들도 조건과 환경을 기준으로 하나의 그룹으로 분류할 수 있다. 이런 분류는 시스템의 구조에 대한 결정을 내릴 때, 재사용되거나 참조된다.
본 논문에서는 요구사항 분석단계와 초기 상위수준의 설계단계에 해당하는 산출물로 소프트웨어 시스템 구조(software system architecture)[9]에 대한 구조 디자인 패턴을 제안한다. 소프트웨어 공학의

소프트웨어 시스템 구조는 객체지향 시스템의 클래스들 사이의 관계를 기술하는데 사용되지만, 본 논문에서 언급하는 소프트웨어 시스템 구조는 객체지향 시스템이 아닌 전통적인 구조적 시스템 구조(structured system architecture)에 더욱 적합하다. 시스템 구조에 대한 구조 디자인 패턴의 구성은 3장에서 제안했던 디자인 패턴과 동일하며, 패턴이 다루는 대상만 다르다.

시스템 구조에 대한 디자인 패턴의 적용 사례는 마찬가지로 안전필수 시스템인 열차제어 시스템의 일부 컴포넌트에 대해 적용하였고, 결과는 다음과 같다:

1) 패턴 이름

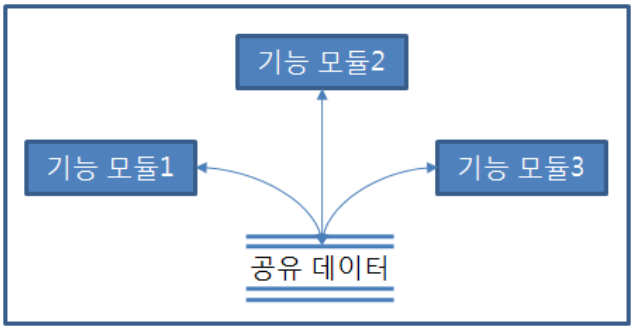
공통 기능 요소 분할 패턴

2) 의도

시스템의 하위 모듈을 구성하는 컴포넌트들을 시스템이 수행하는 기능을 달성하기 위해, 시스템의 목표 기능성을 겹치지 않게 분할하고, 각각의 분할된 기능성을 하나의 컴포넌트에 분배하고, 컴포넌트 단위로 개발하고자 한다. 기능성의 분할은 동시에 처리될 수 있는지 여부와 공유(공통) 데이터에 대한 일관적인 처리 여부에 따라, 즉 동시성(concurrency)과 데이터 일관성(data consistency)을 기준으로 이루어진다.

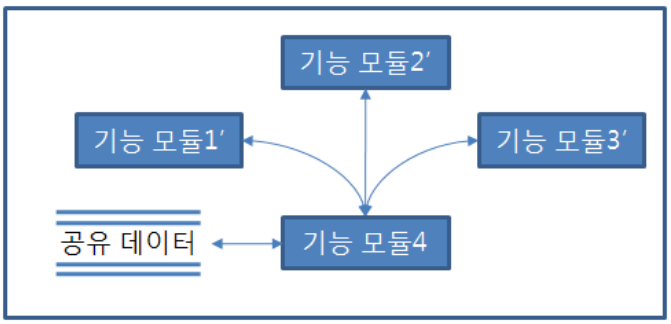
3) 활용성

비록 시스템의 기능성 분할이 상위 모듈 수준에서는 동시성이 보장되도록 이루어졌다 하더라도, 모듈의 하부 모듈 수준에서 여러 모듈의 하위 모듈들의 기능성이 서로 동일한 공유 데이터를 처리해야 하는 경우, 공유 데이터에 대한 일관성을 깨뜨릴 수 있는 위험성(race condition)이 발생하는 경우에 구조적인 조정이 필요하다.



4) 구조

기본적으로 공유 데이터 처리 절차 부분을 단위적(atomic)으로 중간에 인터럽트없이 처리하게 하고 작업 수행시간을 줄이기 위해, 각 기능의 하위 기능들 중에서 공통적으로 가지고 있는 공유데이터 처리 부분만을 별도의 독립 모듈로 분리시키고, 요청과 응답의 방식을 통해 공유데이터를 접근한다.



5) 결과

- 공유 데이터에 대한 race-condition을 전담해서 여러 가지 기법으로 처리할 수 있는 모듈을 별도로 만들어 해결할 수 있다.
- 시스템 전체의 성능적인 면에서 병목현상의 원인이 될 수 있다.

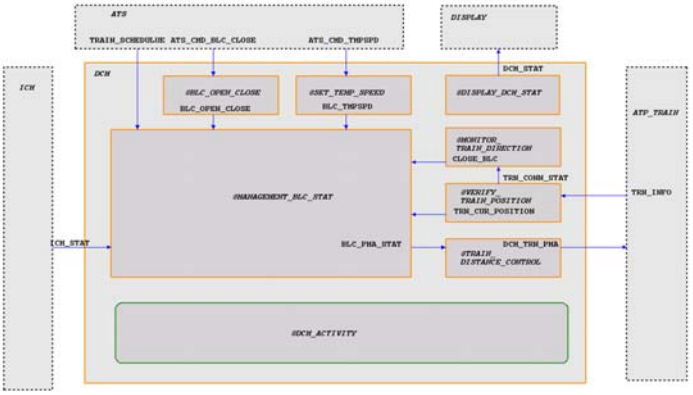
6) 예제

- 자연어 요구 사항

  DCM은 ATS와 통신하여 블록 개방/폐쇄와 같은 블록관리와 관련된 명령을 처리하며, ATP와 통신하여 허용이동 권한 정보 전송을 통해 열차간격 제어 기능을 수행한다.

  DCM은 다음과 같이 5가지 기능을 수행한다: 열차간격 제어, 열차 위치 확인, 열차 이동/방향 감시, 임시 속도 제한명령 처리, 블록 개방/폐쇄

- 모듈 차트



- 정형 구조 명세

  DCM이 SUD(system under development)가 되고, 주변 외부 환경으로 ATS, ATP, ICM 이 있다. DCM 내부에서 5가지 기능을 수행하는 BLC_OPEN_CLOSE, SET_TEMP_SPEED, MONITOR_TRAIN_DIRECTION, VERIFY_TRAIN_POSITION,

TRAIN_DISTANCE_CONTROL 과 블록 정보 업데이트 기능을 전담하는 MANAGEMENT_BLC_STAT 기능을 수행한다.

5.분석

기존의 디자인 패턴 기법들은 디자인 단계에서 객체지향 시스템의 관점에서 문제를 해결하고자 하는 접근방식이었고, 본 논문에서 제안하는 디자인 패턴은 정형 명세를 사용하고 시스템 개발 생명 주기에 근거하여 구조 정의 단계와 디자인 단계에서 적용할 수 있으며, 전통적인 구조적 시스템의 관점에서 문제를 해결하고자 하는 접근 방식이다. 표2는 기존의 디자인 패턴 기법들과의 비교를 나타낸 것이다.

일부 상태차트 도구에서 generic chart라는 불리우는 재사용가능한 일종의 상태차트 라이브러리 기능을 제공하고 있다[10]. 그러나, 메타모델의 역할을 하는 디자인 패턴과는 달리, 상태차트 자체의 모델만을 패턴화한다는 면에서 적용대상과 용도가 다르다.

표 2 디자인 패턴 기법 비교

디자인 패턴	정형 명세 기반 디자인 패턴
Gamma 정의	Gamma 정의
객체 지향 시스템	구조화된 시스템
디자인 단계	개발 생명 주기의 상위 구조 설계단계와 디자인 단계
정형기법 사용(X)	정형기법 사용(O)

6.결론

본 논문에서는 정형 요구 명세와 검증에 사용된 모델의 설계 사항들을 디자인 패턴으로 정의하고, 안전필수 시스템인 열차제어 시스템을 대상으로 적용하였다. 제안된 디자인 패턴은 시스템 개발 생명주기를 바탕으로 개발 단계에서 요구되는 디자인 결정사항을 고려하며 구조화된 시스템을 적용 대상으로 하고 있다.

본 논문에서 제안한 디자인 패턴 항목에는 코드와 검증과 관련된 항목이 없는데, 향후 과제로 실제 디자인 패턴을 적용할 때 사용되는 코드와 검증속성과 조건까지 항목으로 포함하는 연구가 추가적으로 필요하다.

참고문헌

[1] IEC Std. 61508, “Functional safety of electrical/electronic/programmable electronic safety-related systems”, 1998.

[2] R.S.Pressman, Software Engineering, 4ed, 1997. McGraw-Hill.

[3] E.M.Clarke, J.Wing, Formal Methods:State of art and future directions, ACM Computing Surveys, 1996.

[4] J.Bowen, Safety-critical systems, Formal methods and standards, Software Engineering Journal, vol.8, pp.189-209, 1993.

[5] D.Zowghi, V.Gervasi, The Three Cs of Requirements: Consistency, Completeness, and Correctness, REFSQ 2002.

[6] D.Harel, Statecharts: A Visual Formalism for Complex Systems, Science of computer Programming, Vol.8, issue3, pp.231-274, 1987.

[7] E.Gamma, R.Helm,R.Johnson, J.Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[8] I.Bayley, H.Zhu, Formal specification of the variants and behavioral features of design patterns, the journal of systems and software Vol.83, pp.209-221, Elsevier, 2010.

[9] I.Sommerville, Software Engineering, 8ed, Addison-Wesley, 2007.

[10] Statemate Magnum, I-Logix Inc., Andover, MA, 2007.