철도내장형제어기의 안전확보를 위한 소프트웨어 코딩규약 확보방안

신경호, 정의진
한국철도기술연구원

A Study on The Software Coding Standard for Safety of Railway Embedded System

Kyung-Ho Shin, Eui-Jin Joung
Korea Railroad Research Institute

Abstract - Safety is important factor in railway system. Now most of the electric and electronic system which is applied in railway system is the embedded system which software is used. The constitution rate of software which is involved in railway system is gradually increasing. Most of the software which is used in railway system is implemented by the software developer. Thus the implemented code has different features according to the developer and this may cause the bad effect on the software's maintenance. International standard IEC 62279 requires an adoption the coding standard to developing the railway software. And it is plan to recommend complying with the coding standard in safety criteria for railway software which is noticed as the regulation of the Korean railway safety law. In this paper, we review the requirement of coding standards which is present in the software criteria for railway software and international standard. Also it investigates the coding standard which is in other safety related industry and presents the effective way to apply the software coding standard to domestic railway industry.

1. 서 론

철도시스템은 규모가 큰 수송 시스템이기 때문에 대형 사고 발생 시 대규모의 인명피해 및 재산 손실이 발생할 수 있으며, 하나의 하부 시스템 장애가 철도 전체 시스템에 영향을 줄 수 있다. 철도에서 사고를 방지하기 위하여 컴퓨터 및 디지털 내장형 제어기의 활용이 다양한 분야에서 이루어지고 있다. 디지털 제어기가 철도에 적용되는 분야는 신호제어분야, 건널목, 열차의 차상신호 및 제동, 시설물의 감시 및 진단분야 등에 사용되고 있으며, 사용분야가 점점 확대되고 있어 단순기능에서부터 기능 의존도가 높은 복잡한 분야까지 다양한 분야에서 고루 적용되고 있다. 기능 의존도가 높은 분야에 적용되는 내장형 제어기의 고장이 발생할 경우 사고의 직접적인 원인이 될 수 있으며 하드웨어와 소프트웨어 관점에서의 안전성 확보 활동이 필요하며, 현재까지는 주로 하드웨어 관점에서의 안전성 확보 활동들이 수행되어 왔다. 하지만 철도시스템의 안전성을 향상시키기 위해서는 철도시스템에 탑재되는 소프트웨어의 신뢰성, 안전성을 증대를 위한 소프트웨어에 대한 안전성 활동 및 안전성 검증이 반드시 필요하다. 효과적인 철도 소프트웨어의 신뢰성 및 안전성 향상을 위해서는 철도 소프트웨어의 안전을 확보하기 위한 기준과 평가 및 검증체계의 구축이 필요하다. 현재 국토해양부 추진사업인 철도종합안전기술개발사업 중 한국철도기술연구원이 주관으로 수행하는 "철도소프트웨어 안전기준 및 체계구축" 과제에서 이에 관한 안전기준 및 안전체계를 개발하고 있다[1].

일반적으로 소프트웨어의 구현은 개발자 개인에 의해 작성된다. 안전에 영향을 줄 수 있는 소프트웨어는 적합한 절차에 따라 작성되어야 하며 이를 위해서는 공통된 구현 기준과 절차가 필요하다. 철도소프트웨어와 관련한 국제규격인 IEC62279에서는 철도시스템에 탑재되는 소프트웨어의 개발 시 코딩규칙의 준수를 요구하고 있으며 국내의 철도안전법의 하부 규칙으로 고시될 철도소프트웨어 안전기준에서도 코딩규칙의 준수를 권고하고 있다. 본 논문에서는 국제규격 및 철도소프트웨어 안전기준에서 제시하는 코딩규칙의 요건을 확인하고, 다른 안전필수 산업계에서 적용중인 코딩규칙을 조사하여 철도시스템의 안전성 확보를 위한 소프트웨어 코딩규칙의 확보 방안을 제시한다.

2. 국내외 관련기준 분석

2.1 IEC62279의 코딩규약관련 조항

IEC62279는 철도 제어 및 방호시스템의 소프트웨어 요구사양에 대한 국제규격으로 철도시스템에 적용되는 소프트웨어의 안전확보를 위해 필요한 전 생명주기상의 요건을 다루고 있다[3]. 소프트웨어의 안전확보를 위한 요건은 5등급의 SWSIL(Software Safety Integrity Level)로 구분되어 등급화 되어 있다. 여기서 SWSIL0은 비안전소프트웨어로, SWSIL1,2는 안전관련소프트웨어로, SWSIL3,4는 안전필수소프트웨어로 구분된다.

IEC62279에서는 소프트웨어의 설계와 구현단계에서 SWSIL 등급에 따라 설계 및 코딩규약의 적용수준을 달리 정의하고 있으며 SWSIL0,1,2 등급은 강력한 권고(Highly Recommend)를, SWSIL3,4 등급은 필수적용(Mandatory)을 요구하고 있다.

또한 IEC62279규격의 10.4.12 조항에서는 모든 소프트웨어의 개발에 코딩규약을 정의하여 적용할 것을 강제하고 있으며, 10.4.13 조항에서는 코딩규약이 가져야할 특징을 나타낸다. 여기서 제시하는 코딩규약의 특징으로 코딩규약은 우수한 프로그래밍 예제를 포함하고, 비안전 프로그래밍언어특징을 사용치 못하게 하여야 하며, 소스코드 문서화 절차를 포함하여야 한다. 또한 최소요건으로서 각각의 모든 소프트웨어 모듈은 작성자, 변경이력, 간단한 설명 등의 정보를 소스코드 내에 반드시 포함해야 하며, 이러한 정보 또한 표준화된 양식의 사용을 권고한다. 표 1은 IEC62279에서 제시하는 코딩규약의 일부 예를 나타낸다. 또한 표 2는 IEC62279에서 권고하는 프로그래밍 언어의 일부를 표시하는 것으로 국내철도산업계에서 주로 사용하는 프로그래밍 언어인 C언어의 경우 비 제약적인 사용은 SWSIL0를 제외한 나머지 등급에서의 사용이 불가능하며, 코딩규약에 따른 제한적인 C언어의 사용만이 모든 SWSIL등급에서 적용가능함을 확인할 수 있다.

2.2 철도소프트웨어 안전기준(안)관련 조항

철도소프트웨어 안전기준의 체계는 그림 1과 같이 1개 규칙, 5개 그룹으로 구분되어 23개의 하부지침으로 구성되어있다. "철도소프트웨어 안전기준에 관한 규칙"의 하부지침 중 "소프트웨어 구현에 관한 지침"은 소프트웨어 구현에 관련된 세부기준을 규정하고 있으며, 세부적으로 구현활동, 코드신뢰성확보활동, 코드견고성확보활동, 코드추적성확보활동, 코드유지보수성확보활동, 코드기능응집성확보활동, 코드유연성확보활동, 코드이식성확보활동, 소프트웨어통합활동으로 구성되어 있다.

표 1. IEC62279에서 제시하는 설계 및 코딩규약

기법/평가척도	SWSIL 0	SWSIL 1	SWSIL 2	SWSIL 3	SWSIL 4
1.코딩규약 정의	HR	HR	HR	HR	HR
2.코딩스타일지침	HR	HR	HR	HR	HR
3.동적객체사용불가	-	R	R	HR	HR
4.동적변수사용불가	-	R	R	HR	HR
5.포인터 사용 제한	-	R	R	R	R
6.재귀 사용 제한	-	R	R	HR	HR
7.무조건분기 사용불가	-	HR	HR	HR	HR
요구사항 1. 상기 3, 4, 5번 기법은 검증된 컴파일러 또는 변환기에 포함될 수 있다. 2. SWSIL에 따라 상기 기법들을 적합하게 조합하여 사용하여야 한다.					

표 2. IEC62279에서 제시하는 프로그래밍 언어(일부)

기법/평가척도	SWSIL 0	SWSIL 1	SWSIL 2	SWSIL 3	SWSIL 4
1. ADA	R	HR	HR	R	R
4. Fortran 77	R	R	R	R	R
5. C or C++(unrestricted)	R	-	-	NR	NR
6. Subset of C or C++ with coding standards	R	R	R	R	R

표 3. 소프트웨어 구현에 관한 지침의 주요 코딩규약

활동구분	주요 세부 코딩규약
코드신뢰성	-동적메모리할당최소화 -안전등급에 따른 메모리페이징 및 스와핑의 제한 -모든 변수및 포인터의 초기화 -모호한 인터페이스의 최소화 -명확한 데이터형의 정의 -모든 컴파일러경고 확인 및 해결 -동적바인딩 및 연산자 오버로딩 최소화 -멀티타스크(다중작업)의 최소화 -인터럽트 사용 최소화(determinism 확보)
코드견고성	-지역적 예외처리 -일관성있는 예외처리 -입/출력 데이터값의 적합성 확인
코드추적성	-들여쓰기, 서술적 식별자, 설명문 사용 -서부루틴의 크기, 모호한 프로그램, 연계요소의 분산 최소화 -컴파일러종류에 따라 달라질 수 있는 내장함수 또는 내장 라이브러리의 사용 최소화
코드유지보수성	-전역변수 및 복잡한 인터페이스의 최소화
코드기능응집성	-단일목적을 위한 함수 및 서브루틴의 사용
코드유연성	-상수와 변수의 위치 구분 및 사용

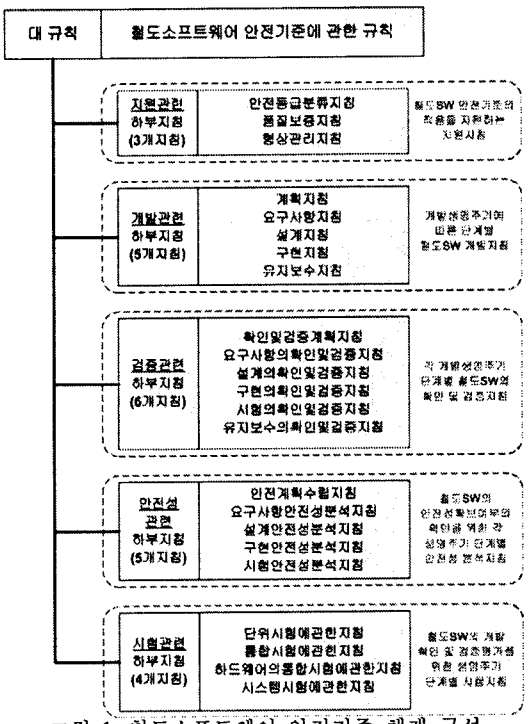

그림 1. 철도소프트웨어 안전기준 체계 구성

각각의 활동에서는 해당 활동의 수행내용에 코딩규약의 준수를 명시적으로 표현하고 있지는 않지만 각 세부 활동내용에 해당단계에 대한 코딩규약이 세부적으로 정의되어 있다. 대표적인 상세지침의 내용은 다음의 표 3과 같다.

2.3 MISRA-C

자동차산업 또한 철도산업과 유사하게 안전특성이 강조된 산업으로 최근 들어 소프트웨어가 포함된 내장형 시스템의 사용이 증가되고 있는 추세이다. MISRA-C는 자동차산업소프트웨어 신뢰성 협회(MISRA: Motor Industry Software Reliability Association)에 의해 개발된 C 언어를 위한 소프트웨어 개발 기준으로 ANSI C로 프로그램된 내장형 시스템의 코드 이식성과 신뢰성을 확보하는 것을 목적으로 한다[4].

MISRA-C 규약은 1998년에 처음으로 만들어져 "Guidelines for the use of the C language in vehicle based software", MISRA-C:1998란 이름으로 배포되었으며 가장 최신의 MISRA-C 규약은 일부지침을 개정한 2번째 MISRA-C 규약으로 2004년에 "Guidelines for the use of the C language in critical systems", MISRA-C:2004으로 배포되었다. MISRA-C:2004는 기존 규약대비 15개의 규칙이 삭제되어 모두 21개의 카테고리로 분류되며 총 141개의 규칙을 정의하고 있다. 141개의 규칙중 121개는 "요구(required)"규칙이며, 20개는 "권고(advisory)"규칙이다. 아래 표는 21개의 카테고리분류체계를 나타낸다.

현재 MISRA-C 규약의 준수를 위해 다양한 소프트웨어 검사도구들이 개발되어 사용되고 있으나 MISRA에서는 아직까지 MISRA-C 준수여부에 대한 인증프로세스를 가지고 있지 못하고 있으며 검사도구 개발자에게도 이러한 지침을 제공하고 있지 못하고 있다. 따라서 현재의 MISRA-C 소프트웨어 검사도구는 단지 검사기능만 가능하며, MISRA-C 준수를 위한 코드의 수정은 개발자가 직접 수행해야 한다.

표 4. MISRA-C 규약의 분류 체계

분류체계 : 21개 카테고리		
Environment	Language extentions	Character sets
Identifiers	Constants	Declarations and definitions
Initializations	Arithmetic type conversions	Pointer type conversions
Expressions	Control statement expressions	Control flow
Switch statements	Functions	Pointers & arrays
Structures & unions	Preprocessing directives	Standard libraries
Runtime failures	Documentation	Types

3. 국내 철도SW의 구현활동 분석과 대응방안

3.1 국내철도산업계의 SW구현 활동 현황 분석

국내철도산업계의 SW구현 활동의 현황 분석은 철도신호관련 시스템의 제작사를 대상으로 설문 및 인터뷰를 통해 수행하였다[2]. 철도신호시스템은 차상 및 지상 철도설비를 모두 포함하고 있으며 타 분야 대비 안전성 요건과 내장형 제어기의 활용이 많은 분야이다.

국내 철도신호시스템 제작사의 대부분은 중소산업체로 국산화된 철도시스템은 기능구현 위주의 개발과 성능시험, 설치시험, 시운전시험 등의 시험을 통해 개발 및 운영이 이루어지고 있다. 특히 철도산업계 공통의 개발 프로세스가 확립되어 있지 못하기 때문에 각 업체별 또는 프로젝트별로 독자적인 프로세스를 적용하고 있는 실정으로 안전성 확보 활동에 따른 개발이 부족한 상황이다. 또한 국내 철도산업계에서 사용하고 있는 프로그래밍 언어에 대하여 살펴보면 PC기반 소프트웨어제어기의 경우에는 Microsoft사의 Visual Studio를 활용하여 C/C++언어로 개발을 하고 있었으며, 임베디드 소프트웨어의 개발을 위해서는 실시간운영체계(RTOS)가 적용되는 복잡한 제어기의 경우 VxWorks OS를 적용하여 C 언어로 제어소프트웨어를 개발하며, 그렇지 않을 경우에는 상용 OS없이 C 언어를 사용하여 소프트웨어 제어기를 개발하고 있었다. 따라서 국내 철도소프트웨어의 대부분은 C 언어로 개발된다고 할 수 있으며, C 언어의 무조건적인 사용은 IEC62279에서 SWSIL0를 제외하고는 그 사용을 권고하지 않는 상황으로 C 언어 기반의 철도산업 공통 코딩규약의 확보가 필요한 이유이다.

물론 현재까지도 소프트웨어 코딩규약은 각각의 철도산업체의 조직 특성에 따라 적용되어 왔으나 그 수준은 버전관리, 주석처리 등과 같은 간단한 스타일링 정의 수준 정도로서 안전성의 확보를 위한 수준의 규약이 필요하며 철도산업 전체의 안전확보를 위해서는 업계 표준으로 공통적으로 지켜질 수 있는 코딩규약이 되어야 한다.

3.2 소프트웨어 코딩규약 확보방안

철도안전법의 하부 규칙으로 고시될 철도소프트웨어 안전기준(안)은 2008년부터 단계적으로 고시가 진행될 예정이다. "철도소프트웨어 안전기준에 관한 규칙"의 하부지침 중 "소프트웨어 구현에 관한 지침"에서 소프트웨어 구현에 관련한 세부기준을 규정하고 있으며 각각의 세부 활동에서 코딩활동에 대한 안전요건을 정의하고 있다. 하지만 철도소프트웨어 안전기준에서는 특정 프로그래밍 언어를 규정하고 있지 않다. 반면 국내 철도산업계에서는 C 언어를 주로 적용하고 있으며 C 언어에 대한 코딩규약의 확보는 철도소프트웨어의 안전확보를 위한 기본활동이라 할 수 있다.

C 언어의 코딩규약 중 MISRA-C는 자동차 산업의 대표적인 안전 및 임무필수 소프트웨어 코딩규약으로 총 141개의 세부적인 규약을 가지고 있다. 반면 MISRA-C와 유사한 철도산업용 코딩규약은 존재하지 않으며 관련 국제규격인 IEC62279에서 제시하는 코딩규약도 총 7가지로 최소요건만을 정의하고 있다. MISRA-C는 IEC62279의 7가지 최소요건을 모두 포함하고 있으며 다양한 상용 MISRA-C 소프트웨어 검사도구의 활용이 가능한 상태로 철도 내장형제어기의 소프트웨어 개발시 MISRA-C의 적용은 국제규격에 따른 소프트웨어 구현활동을 지원하며, 산업계에서 더욱 신속하고 효과적으로 코딩규약을 적용할 수 있도록 한다.

다만 MISRA-C와 같은 코딩규약은 코드작성 시 개발자에 의해 지켜져야 할 규약으로 검사도구의 사용만으로는 적용에 한계가 있다. 따라서 코딩규약의 효율적인 적용과 이를 통한 소프트웨어의 안전확보를 위해서는 교육 및 기타 장려수단의 활용이 필요할 것이다. 물론 철도소프트웨어의 특성을 고려하여 철도소프트웨어 안전기준에 적합한 코딩규약의 마련과 적용을 고려할 수도 있으나 이러한 적용방식은 현실적으로 기업에 시간적, 비용적, 기술적 부담을 제공할 수 있으므로 부담을 최소화하는 방향으로 먼저 적용을 유도하여야 할 것이다.

3. 결 론

철도시스템은 안전확보가 필수적인 시스템으로 현재 철도시스템에 탑재되는 소프트웨어의 비율이 점진적으로 증가하고 있으며 기존의 전기, 기계적인 제어기들은 소프트웨어가 탑재된 내장형 제어기로 발전되고 있는 추세이다. 철도시스템의 안전확보를 위하여 하드웨어 측면에 대한 강조뿐만 아니라 소프트웨어 측면에서의 안전성 활동도 함께 필요하며, 철도안전법의 하부규칙으로 철도소프트웨어 안전전기준이 고시될 예정으로 철도소프트웨어의 안전확보를 위한 규칙과 여러 지침이 제시될 예정이다. 철도소프트웨어 안전확보를 위해 생명주기별로 여러가지 안전활동이 요구되며 소스코드는 소프트웨어의 가장 기본요소라 볼 수 있다. 따라서 공통된 소프트웨어의 코딩규칙에 따른 코드 작성은 코드의 신뢰성, 견고성, 추적성, 유지보수성, 기능응집성, 유연성을 증대시키고, 확인 및 검증활동을 용이하게 하여 시스템의 안전성을 향상시킬 수 있으므로 철도시스템의 안전확보 증대를 위해서도 반드시 그 적용이 필요하다 할 수 있다.

[참 고 문 헌]

[1] 정의진, "철도 안전필수 소프트웨어를 위한 안전기준 도출", 대한전기학회 하계학술대회 논문집, 2007

[2] 신경호, "철도소프트웨어 안전기준의 현장 적용방안 도출", 대한전기학회 전기기기 및 에너지변환시스템부문회 추계학술대회 논문집, 2007

[3] IEC 62279-Railway applications:Communications, signalling and processing systems-Software for railway control and protections systems, 2002

[4] "MISRA-C:2004 Guidelines for the use of the C language in critical system", MISRA, 2004