The Transactions of the Korean Institute of Electrical Engineers Vol. 62P, No. 1, pp. 30~35, 2013
http://dx.doi.org/10.5370/KIEEP.2013.62.1.030

열차제어시스템 소프트웨어 정적 테스팅 적용시험 결과 분석

Analysis on Software Static Testing Results of Railway Signaling System

황 종 규† · 조 현 정* · 정 락 교**
(Jong-Gyu Hwang · Hyun-Jeong Jo · Rak-Gyo Jeong)

Abstract - Many function of railway signalling system which is in charge of most core function in a railway system are being operated by the software according to the development of computer technology. Accordingly, the source code testing to verify the safety of the railway signalling system software becomes to be more important, and related international standards highly recommend verifications on the source code also. For this reason, several related studies on vital source code verification were executed from several years ago in Korea. This paper performed tests through the application to railway signalling system being applied to the existing actual domestic railway sites through automated testing tools for coding rules of signalling system software and another signaling system software under development in Korea recently, and analyzed their results.

Key Words : Software static testing, Software coding rules, Software complexity

1. 서 론

철도시스템에서 가장 핵심기능을 담당하고 있는 열차제어시스템은 최근 컴퓨터 기술의 발달에 따라 기존의 전기적 장치로부터 컴퓨터 기반의 제어시스템으로 변경되어가고 있다. 이에 따라 열차제어시스템 주요 기능의 많은 부분이 소프트웨어화되어 가고 있으며, 소프트웨어에의 의존성이 계속해서 증가하고 있다. 열차제어시스템 소프트웨어가 더욱 복잡해지게 되면서, 열차제어시스템에서 소프트웨어가 차지하는 비중이 더욱 증대되고 있다.

열차제어시스템 소프트웨어 안전성 요구사항들이 최근 들어 IEC 61508과 IEC 62279에 의해 국제표준화 되었고[1][2], 또한 국내에서도 철도 안전법 및 하위 규칙 등이 제정되어 이러한 열차제어시스템 관련 국제표준에서 요구하는 각종 소프트웨어 테스팅 및 검증활동이 요구되기 시작하고 있다[3]. 특히, 관련 국제규격에서 SWSIL(Software Safety Integrity Level) 등급의 소프트웨어 설계 및 개발에 있어서 코딩규칙(Coding Rules)을 반드시 준수하도록 요구하고 있을 뿐, 구체적인 코딩규칙을 언급하고 있지는 않다. 따라서 열차제어시스템 개발 프로젝트별 별도의 코딩규칙을 설정하여 활용하고 있다. 이러한 문제점의 극복을 위해 관련 국제표준의 분석을 통해 열차제어시스템 소프트웨어가 준수하여야 할 코딩규칙에 대한 연구가 수년전부터 시작되었으며, 현재 코딩규칙(안)이 도출되어 국가표준화 작업이 진행되고 있다[4][5]. 또한 일반적으로 임베디드 소프트웨어 코딩규칙으로 많이 준용되고 있는 자동차 제어시스템 분야 표준 코딩규칙인 MISRA-C[6]와 연구를 통해 도출된 철도용 코딩규칙의 자동검사를 위한 테스팅 도구에 대한 연구도 진행되었다[4][7][8]. 이러한 최근의 국내에서의 열차제어시스템 소프트웨어 코딩규칙 및 테스팅 자동화 도구관련 연구의 진행 등으로 소프트웨어의 테스팅 필요성에 대한 인식이 많이 개선되고 있다.

이러한 열차제어시스템 소프트웨어 테스팅 관련 연구의 일환으로 실제 철도현장에 운용 중인 열차제어시스템과 최근에 국내에서 개발 되고 있는 열차제어시스템 소프트웨어에 개발 중인 소프트웨어 테스팅 도구의 적용성 연구를 수행하였다. 적용대상으로 한 기존 시스템은 국제 표준이 국내에 소개되기 전부터 철도현장서 적용되어 오고 있던 시스템이며, 다른 하나는 그 이후에 개발이 시작된 시스템이다. 이들 두 열차제어시스템 소프트웨어를 대상으로 한 적용성 시험 및 결과분석을 통해 개발하고 있는 검증도구의 유효성을 확인하고 또한 열차제어시스템의 안전성 검증 측면에서의 소프트웨어 정적 테스팅 기술이 철도산업 발전에 미치는 영향을 고찰하고자 한다.

2. 열차제어시스템 소프트웨어 정적 테스팅 기준 분석

철도시스템 소프트웨어의 코딩규칙 준수를 국제 규격에서는 HR(Highly Recommend)로 엄격하게 적용하도록 규정하고 있지만, 아직까지 국내뿐 아니라 해외에서도 철도시스템용 코딩규칙이 표준화되어 있지 않은 실정이다. 따라서 해외에서는 개발 시스템별 또는 운영기관별 코딩규칙을 정하여 제작업체에게 이를 준수하도록 요구하고 있다. 철도 이외의 다른 산업 분야에서는 가장 일반적으로 알려진 자동차 제어장치용 MISRA-C 코딩규칙이 있으며, 이는 자동차 산업뿐 아니라 철도 등 다른 산업 분야에서 임베디드 소프트웨어 코딩규칙으로 많이 활용하고 있다. 이 외에 그림 1에서 나타낸 바와 같이 항공기 등의 경우 프로젝트별로 요구

* 정 회 원 : 한국철도기술연구원 선임연구원
** 정 회 원 : 한국철도기술연구원 책임연구원
† 교신저자, 정회원 : 한국철도기술연구원 책임연구원
E-mail : jghwang@krri.re.kr
접수일자 : 2013년 1월 9일
최종완료 : 2013년 2월 6일

사항 정립단계에서 특정한 코딩규칙을 개발 및 활용하고 있으며, 정보통신 분야의 모바일 단말의 경우도 별도의 자체적인 코딩규칙이 적용되고 있다.

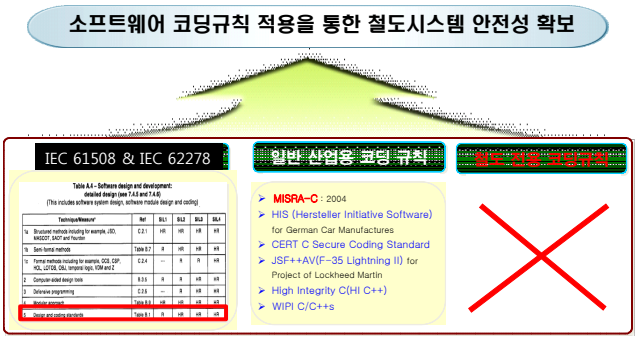

그림 1 철도시스템 소프트웨어 전용 코딩규칙의 필요

Fig. 1 Need of Exclusive Coding Rules for Railway System Software

철도는 대표적인 대중교통수단으로서 사고가 발생할 경우 사회경제적으로 파급효과가 큰 높은 안전성이 요구되는 제어시스템이다. 이에 따라 철도 소프트웨어는 오랫동안의 운영을 통해 안전성이 실증된 로직이 소프트웨어로 구현되는 등 철도 고유의 특성으로 인해 철도 전용의 소프트웨어 코딩규칙이 필요하다.

국내에서는 이러한 필요성에 따라 철도 안전관련 국제 규격인 IEC 61508, IEC 62279, 그리고 자동차 분야 코딩규칙인 MISRA-C의 분석을 통해 철도시스템 소프트웨어에 적합한 코딩규칙에 대한 연구가 진행되고 있다. 이러한 연구결과로 국제규격의 분석을 통해 도출된 코딩규칙 중 MISRA 규칙과 중복되는 부분을 제외하고 IEC 61508에서 12개와 IEC 62279에서 10개의 코딩규칙이 도출되었다[4][5]. 이에 따라 열차제어시스템의 내장형 소프트웨어를 개발할 때 준수하여야할 표준으로 자동차 업계에서 널리 사용되고 있는 MISRA-C 코딩규칙에 더불어 철도시스템 소프트웨어 개발 시 준수하여야 할 국제 표준인 IEC 61508, IEC 62279을 통해 도출한 코딩규칙을 준수하도록 지침을 만들어 국내 철도산업에 보급하고 있으며, 이를 국내 표준으로 준비 중에 있다[4][5].

또한 국제규격에서는 소프트웨어 메트릭에 대한 분석 및 확인하도록 하고 있다. 소프트웨어 메트릭은 복잡도, 응집도, 결합도 등 다양한 항목이 있지만, 이중 소프트웨어의 안전성 및 신뢰성에 가장 큰 영향을 미치는 메트릭은 소프트웨어 복잡도(Software Complexity)이다. 열차제어시스템 소프트웨어는 가능하면 함수별로 단순화시켜서 개발되어야 하며, 함수가 복잡해지면 잠재적인 오류가 내장될 수 있어 바이탈 소프트웨어에는 소스코드의 복잡도 크기를 일정 수준 이하로 제한되어야 한다. 복잡도는 소프트웨어 공학에서 여러 가지 모델이 있는데 이중에서 가장 일반적인 CC(Cyclomatic Complexity)를 본 연구에서는 기준으로 하였다[9]. 본 연구의 선행연구를 통해 국내 철도시스템 소프트웨어 복잡도의 적용 기준을 표 1과 같이 도출하여, 앞에서 언급한 열차제어시스템 소프트웨어 테스팅 기술지침(안)에 포함되어 있다[5][10].

표 1 철도시스템 소프트웨어 복잡도 기준

Table 1 Criteria of Software Complexity for Railway System Software

SWSIL	0	1	2	3	4
복잡도 기준	NA	복잡도 1~50 허용		복잡도 1~20 허용	

4. 소프트웨어 정적 테스팅 도구

소프트웨어 테스팅은 그림 2와 같은 소프트웨어의 개발 및 시험평가 절차에서, 크게 소스코드의 코딩규칙 및 메트릭을 검증하는 소스코드 정적 테스팅, 소프트웨어의 테스트 커버리지 등을 분석하는 화이트박스 테스팅, 그리고 소프트웨어의 기능안전성을 테스트하는 블랙박스 테스팅의 단계로 구분할 수 있다. 이 중 코딩규칙 검증 및 메트릭을 검증하는 정적 테스팅 부분이 소프트웨어 테스팅의 출발점이다[11].

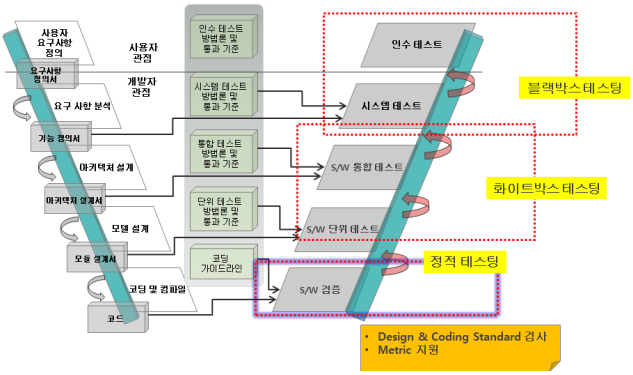

그림 2 소스코드 정적테스팅을 포함한 소프트웨어 테스팅 절차

Fig. 2 Software testing stages and static testing for source codes

열차제어시스템 소프트웨어 소스코드 코딩규칙 검사 기술은 열차제어시스템 소프트웨어가 관련 규격 등에서 요구하는 코딩 규칙을 준수하는지 여부를 분석하여 어느 정도 준수하는지를 확인하는 것이다. 이와 같은 검사 결과는 규칙 위배 사항뿐만 아니라 소스코드가 내재하고 있는 잠재적인 오류들을 검출해주어 소프트웨어의 오류로 인해 시스템이 오작동할 위험을 사전에 검증해줄 수 있다.

현재 일반 산업용 내장형 소프트웨어 소스코드 정적분석 도구들이 일부 상용화되어 있지만, 철도 소프트웨어 관련 국제표준에서 요구하고 있는 소프트웨어 안전 요구사항에 적합한 정적 테스팅을 수행하지 못하고 있다[12][13]. 이러한 필요성에 따라 국내에서는 철도시스템 안전관련 국제표준에 적합한 열차제어시스템 전용 검사 도구의 개발이 시작되었으며[4][7]-[10], 본 논문에서는 이 개발도구를 실제 열차제어시스템 소프트웨어를 대상으로 한 테스팅을 통해 그 적용성을 분석하였다.

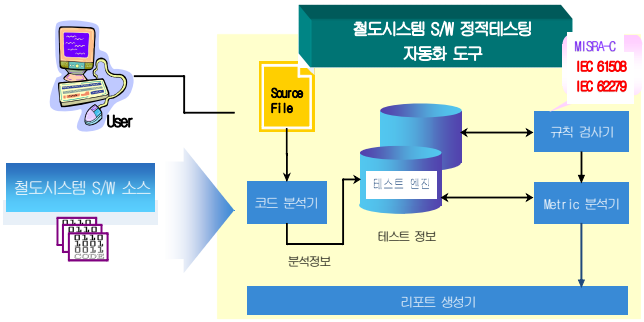

그림 3 소프트웨어 정적 테스팅 도구의 구성

그림 3 Configuration of the automated tool for software static testing

4. 대상시스템 소프트웨어 테스팅 적용 결과

그림 3과 같이 개발된 코딩규칙 검사 및 메트릭 분석이 가능한 정적 테스팅 자동화 도구의 적용성 확인을 위해 국내의 열차제어시스템 소프트웨어를 대상으로 적용성 시험을 수행하였다. 개발 도구의 적용성 확인을 위한 대상 소프트웨어는 먼저 철도시스템 소프트웨어 테스팅관련 국제 규격이 국내에 소개되기 이전에 개발되어 실제로 철도현장에 적용되고 있는 열차제어시스템(대상 시스템 A : 본 논문 이하에서는 'A 대상시스템'이라 부른다) 소프트웨어를 대상으로 하였다. 이 대상시스템 A 소프트웨어는 실제 철도현장에 적용되고 있지만, 최근에 소프트웨어 장애가 발생하여 열차운영에 장애를 초래한 적이 있어, 본 연구를 통해 개발된 테스팅 도구가 이러한 장애가 발생한 소스코드 부분을 검지할 수 있는지를 확인하고자 하였다. 이 장애가 발생된 소프트웨어는 시뮬레이터를 통한 테스트 검증, 또 오랜 기간 동안의 실제 철도현장에서 운용되는 동안 검출되지 않았던 장애로, 최근에 장애가 발생하여 해당 부분이 수정되었다. 즉, 기존의 관행적인 소프트웨어 개발 및 검증, 운용 단계에서 걸러지지 못하던 부분이 오랜 기간 운용하는 도중에 장애가 도출된 경우이다.

표 2 적용성 분석 대상 시스템 소프트웨어 개요

Table 2 Basic information software on applicable target systems

	A 대상 시스템	B 대상 시스템
프로그래밍 언어	C/C++	C/C++
전체 파일 개수	22 개 (입력파일 14개/포함파일 8개)	5 개 (입력파일 4개/포함파일 1개)
전체 라인 수	48,024 라인	3,626 라인
주석 라인 수(%)	10,778 라인(22.44%)	654 라인(18.04%)
함수 개수	517 개	42 개
주요 기능	진로제어 기능	진로제어 기능

또 다른 하나의 대상은 최근에 개발되고 있는 열차제어시스템 소프트웨어를 적용성 시험 대상으로 하였다(본 논문 이하에서는 'B 대상시스템'이라 부른다). 이 소프트웨어는 열차제어시스템 소프트웨어 테스팅 기술 연구가 국내에서 수행되기 시작한 이후에 개발을 시작한 것으로, 선행연구인 코딩규칙 연구 등 관련된 연구결과가 직접적으로 적용되지는 않았지만 국내에서 연구결과물인 테스팅 도구의 시연회 및 홍보, 코딩규칙의 철도 산업 실무자를 대상으로 한 교육, 코딩국내 표준으로 추진 등을 통해 국내 산업 현장에 어느 정도 파급되고 있는지를 파악할 수 있는 척도의 하나여서 적용대상으로 선정하였다. 즉, 이들 두 시스템 소프트웨어를 대상으로 한 시험 및 결과분석을 통해 개발하고 있는 검증도구의 유효성을 확인하고 또한 관련된 연구가 철도산업에서의 영향을 고찰하고자 한다.

대상 시스템의 기본 정보는 표 2와 같으며, 두 시스템 모두 VxWorks기반의 내장형 소프트웨어이다. 그림 4는 소프트웨어 정적시험을 수행한 후 분석 중인 화면을 나타낸 것이다. 이 그림에서와 같이 위배된 코딩규칙 리스트와 그 내용 및 파일 내의 위치가 표시되고, 해당 위배 코딩리스트를 선택하면 위배된 부분의 소스코드를 표시하도록 되어 있어 소스코드 디버깅에도 활용이 가능하다.

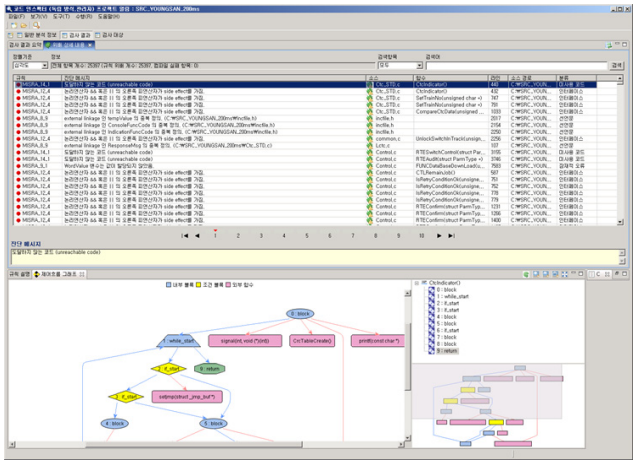

그림 4 정적 테스팅 결과 화면

Fig. 4 Screen of the static test result

표 3 A 대상시스템 소프트웨어 정적 테스팅 결과 요약

Table 3 Result of applying static testing to A target system software

검사수행 정보	적용 규칙	IEC 61508, IEC 62279, MISRA 2004, 복잡도(CC)
결과요약 정보	총 개선 필요 항목 수	2,310개(심각도 '매우 높음' : 223개, '높음' : 2087개)
	개선 항목 포함 파일 개수	20 / 22개
	개선 항목 식별 규칙 개수 / 전체 규칙 개수	86 / 142개 (심각도 '매우 높음' : 7개, '높음' : 15개, 해당 규칙 : 22개)
	*규칙 준수율	39.44%
	**규칙 위배 밀도	0.68

*규칙 준수율 = ((전체 규칙 개수 - 위배 규칙 개수) / 전체 규칙 개수) X 100

**규칙 위배 밀도 = 규칙 위배 개수 / 유효 라인

4.1 A 대상 시스템 적용 결과

A 대상시스템 소프트웨어의 정적 테스팅 결과를 표 3에 요약하여 나타내었다. 적용규칙은 관련 국제규격으로부터 도출한 코딩규칙과 MISRA-C 코딩규칙, 그리고 메트릭으로서 복잡도를 그 대상으로 하였다. 대상시스템 복잡도 분석은 SIL 2를 기준으로 하였다. 분석결과 복잡도 50을 초과하는 함수가 총 16개이고, 이중 복잡도가 가장 큰 것이 221까지 나타나고 있었다. 이는 대상 소프트웨어가 안전성면에 대한 고려 없이 기능구현 위주로 개발됨으로 인해 함수 각각이 매우 복잡하게 구현되어 복잡도가 큼을 알 수 있었다. 이러한 매우 높은 복잡도를 가지는 함수는 목적으로 하는 기능의 구현은 가능하지만, 함수가 복잡함으로 인해 개발자가 알지 못하는 잠재적인 에러를 내포하고 있을 수 있다. 따라서 이처럼 복잡도가 기준치보다 높은 함수들은 우선은 장애가 발생하지 않을지라도 향후에라도 장애가 발생할 수 있으므로, 안전성 측면에서 해당 함수들을 여러 함수로 분리하거나 동일한 기능을 갖는 복잡도가 낮은 소프트웨어 로직으로 변경이 필요하다.

표 4 Return 값 부재 코딩규칙 위배 형태 및 개선 형태
Table 4 Violated form due to the absence of return value of function and improved

위배 형태	개선 형태
int g_var = 0; int func(int var) { int ret = 1; ret += var; g_var = var; } int calc(void) { int local=0; local = func(0); if(local > 0) execute; else error; }	int g_var = 0; int func(int var) { int ret = 1; ret += var; g_var = var; return ret; }

코딩규칙 검사 결과에서는 총 2,310개의 개선 항목을 검출했으며, 개선이 필요한 코딩규칙은 86개, 코딩규칙 준수율은 39.4%, 코딩규칙 위배밀도는 0.68로 분석되었다. 분석결과 위배되는 코딩규칙 및 위배개수가 많았으며, 이에 따라 코딩 준수율이 40% 이하로 낮았다. 이는 대상 소프트웨어가 국내에서 안전성 측면의 정적 테스팅관련 연구가 시작되기 전부터 개발되어 사용되어온 것 때문으로 보인다.

특히 이 시스템의 운용 중에 최근에 장애가 발생되어 수정이 된 부분이 본 연구를 통해 개발한 정적 테스팅 도구를 통해 검출되었다. 이는 '함수 선언과 정의의 상이' 코딩규칙으로, 일반적으로 함수선언과 해당함수의 정의가 틀려도 기능적인 측면에서 문제가 안될 수도 있지만, 경우에 따라서는 실행 도중 메모리의 오류로 오동작이 발생할 수 있다. 이러한 부분이 실제 시스템의 운용 도중에 장애가 발생하여 열차의 운용에 지장을 초래하는 등의 문제점이 발생되었다. 만약 이 부분이 정적 테스팅 과정을 통해 사전에 검지되었으면 충분히 장애 발생을 예방할 수 있는 부분이었음이 본 적용성 테스팅을 통해 확인되었다. 실제 위배된 부분과 개선 형태를 표 4에 나타내었다. 표에서 보듯이 반환 타입이 void가 아닌 함수는 명시적인 반환 값이 존재해야 하며, 명시적인 반환 값이 존재하지 않을 경우 의도하지 않은 값이 반환되는 경우가 발생함으로 이를 방지하기 위함이다. 예를 보면 calc 함수 호출 시 func 함수 호출 후 반환 결과에 의해 의도했던 execute 문장이 아닌 error 구문이 수행되게 된다.

표 5 의도하지 않는 연산 수행불가 코딩규칙 위배 및 개선 형태
Table 5 Violated form due to the inability of performing intended operations and improved

위배 형태	개선 형태
while ((var == 0)&&(x != i++)) { execute; if(i > 1000) break; }	while((var==0)&&(x!=i) { execute; i++; if (i >10000) break; }

그림 5는 검출한 개선 항목 중 각 심각도 수준의 비율과, 심각도가 높은 항목 중 많은 비중을 차지하고 있는 8개의 위배 항목을 나타낸 도표이다. 그림에서 나타낸 바와 같이

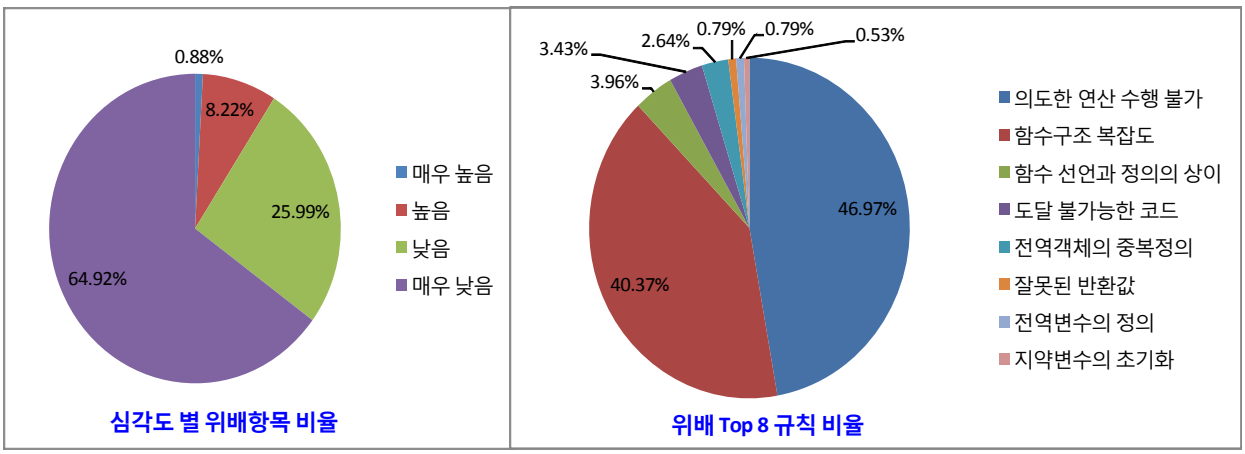

그림 5 A 대상시스템 코딩규칙 검사 결과 요약
Fig. 5 Test overview of coding rules inspection to A target system

주요한 위배 코딩규칙은 의도한 연산 수행불가, 함수구조 복잡도, 도달 불가능한 코드 등이었다. 이중 '의도한 연산 수행불가' 코딩규칙의 예를 표 5에 나타내었다. 논리연산자 '&&' 혹은 '||'의 오른쪽 피 연산자가 의도한 연산이 수행되지 않아 오동작을 발생시킬 수 있다. 즉, '&&'과 '||'는 왼쪽 피 연산자에 따라 오른쪽 피연산자를 수행하지 않을 수도 있다. 이로 인하여 실제 기능 요구사항을 만족하기 위한 기능이 수행되지 않는 경우가 발생할 수 있으므로 개선이 필요한 부분으로 본 연구를 통해 개발된 도구에 의해 검사됨을 확인할 수 있었다.

4.2 B 대상 시스템 적용 결과

시험적용에 대한 결과분석 기준은 A 대상시스템과 같은 SWSIL 2 레벨로 선정하였으며, 소프트웨어 시험적용 결과를 요약하면 다음 표 5와 같다.

표 5 B 대상시스템 소프트웨어 정적 테스팅 결과 요약
표 5 B Result of applying static testing to B target system software

검사수행 정보	적용 규칙	IEC 61508, IEC 62279, MISRA 2004, 복잡도(CC)
결과요약 정보	총 개선 필요 항목 수	98개(심각도 '매우 높음' : 3개, '높음' : 95개)
	개선 항목 포함 파일 개수	2개(동적 생성 : 3개)
	개선 항목 식별 규칙 개수 / 전체 규칙 개수	33 / 146개
	*규칙 준수율	77.40%
	**규칙 위배 밀도	0.19

소스코드 복잡도 분석은 A 대상시스템에 적용한 동일한 기준인 CC를 기준으로 하였으며, 시험 대상에 해당하는 함수 중 복잡도가 가장 큰 것이 41로 기준을 초과하는 항목은 존재하지 않았다. 코딩규칙 위배여부 검사 결과 98개의 개선 항목을 검출했으며, 개선이 필요한 코딩규칙은 33개, 코딩규칙 준수율은 77.4%, 코딩규칙 위배밀도는 0.19로 분석되었다. 이러한 시험결과는 결과는 B 소프트웨어가 A 소프트웨어보다 전체적으로 코딩규칙을 잘 준수하고 있고, 또한 소스코드의 복잡도도 SIL 등급 기준을 충족함을 확인할 수 있다.

그림 6은 검출한 개선 항목 중 각 심각도 수준의 비율과, 심각도가 높은 항목 중 많은 비중을 차지하고 있는 8개의 위배 항목을 나타낸 도표이다. 위배한 코딩 규칙 중 함수선언과 정의 상이, 도달 불가능한 코드 규칙 등은 A 대상시스템과 같다. 즉, 이러한 부분이 소프트웨어 개발자가 실수하기 쉬운 코딩규칙임이 간접적으로 분석되어질 수 있다.

이러한 위배한 코딩 규칙 중 MISRA 14.1인 '도달 불가능한 코드'의 위배 내용과 해당부분을 개선할 경우를 표 6에 나타내었다. 즉, 해당코딩규칙이 위배된 것으로 분석된 코드의 경우 무한 루프가 동작하므로 return 문을 수행할 수 없다. 또한, 해당 코드에서는 무한루프 동작을 빠져 나올 경우 무조건 'OK'를 반환하므로 의도하지 않은 동작이 이루어 질 수 있으므로, 시스템의 장애요인이 될 수 있는 부분이다.

표 6 도달 불가능한 코드 코딩규칙의 위배 및 개선 형태
Table 6 Violated form due to the unreachable code and improved form

위배 형태	개선 형태
// FOREVER : for(;;) FOREVER { switch(…) { case …: case …: } } *return(OK);* }	// FOREVER : for(;;) FOREVER { switch(…) { case …: case …: } } }

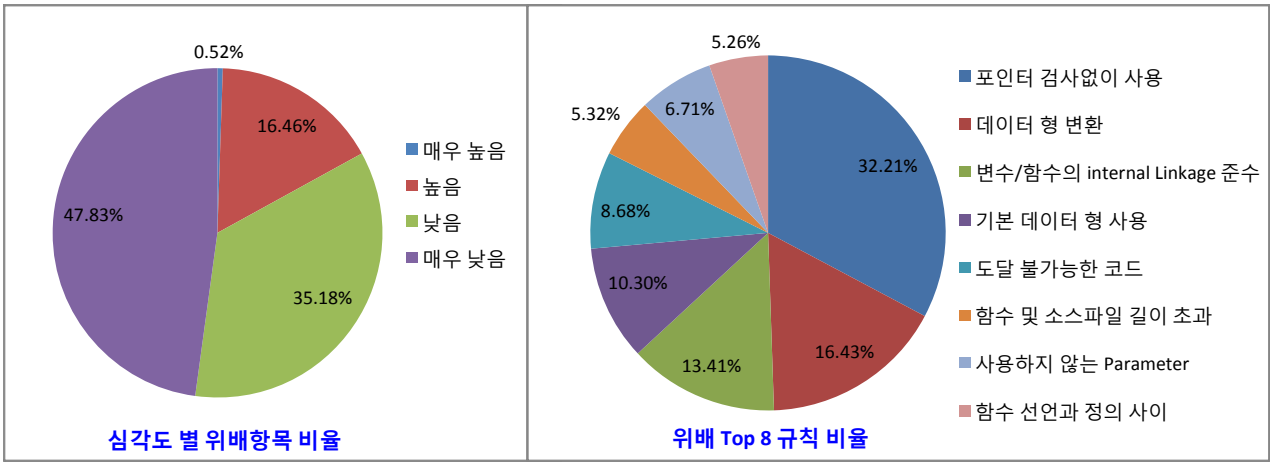

그림 6 B 대상시스템 코딩규칙 검사 결과 요약
Fig. 6 Test overview of coding rules inspection to B target system

5. 결 론

철도시스템에서 열차의 안전운행을 담보하는 열차제어시스템은 소프트웨어의 안전성 검증 관련된 연구의 하나로 국내에서 철도소프트웨어 전용의 정적 테스팅 도구가 개발되었다. 본 논문에서는 개발된 소프트웨어 정적 테스팅 자동화 도구의 적용성 확인을 위해 기존에 사용되고 있는 열차제어시스템과 최근에 개발하고 있는 열차제어시스템 소프트웨어에 적용해 보았다. 그 중 하나는 국내 철도현장에 운용하면서 최근에 소프트웨어 장애가 발생되어 수정이 된 적이 있는 것으로, 장애 발생된 버전을 대상으로 하여 개발된 테스팅 도구가 그 에러를 검지할 수 있는지 여부의 확인하였다. 테스팅 결과 최근에 에러가 확인된 부분이 본 적용성 시험을 통해 정상적으로 검지됨을 확인할 수 있었다. 만약 이 시스템이 운용 전에 정적 테스팅 과정을 통해 사전에 검지되었으면 충분히 장애 발생을 예방할 수 있는 부분이었음이 본 적용성 분석을 통해 확인되었다. 또한 최근에 개발이 시작된 소프트웨어를 대상으로 한 적용성 시험에서는 오래전에 개발되어 운용되고 있는 소프트웨어보다 많이 향상되고 있음이 확인되었다. 이 두 가지 대상을 통한 적용성 시험결과, 국내에서도 소프트웨어 안전성 관련 연구가 시작되면서 개발자들이 코딩규칙이나 복잡도와 같은 소프트웨어 안전성에 영향을 미치는 부분에 대한 많은 노력이 이루어지고 있음이 간접적으로 확인할 수 있었다. 또한 코딩규칙 준수여부 테스팅 결과 공통적으로 많이 위배되는 규칙들이 있음이 확인되었다. 두 가지 소프트웨어만을 대상으로 하여 일관성을 갖기는 어렵지만, 적용 대상 소프트웨어를 확대해서 분석할 경우 공통적으로 많이 위배되는 코딩규칙 리스트의 도출이 가능할 것으로 보이며, 본 연구에서는 그 가능성을 확인할 수 있었다. 이러한 일반화될 수 있는 위배코딩규칙이 분석될 경우 이를 기준으로 국내 산업에 교육 등을 통해 전파할 경우 철도 소프트웨어의 안전성 향상에 많은 기여가 예상된다. 결론적으로 본 적용성 연구에서는 잠재되어 있던 에러 검지를 통해 개발된 도구의 유효성을 확인하였다.

참 고 문 헌

[1] IEC 61508-3, “Functional safety of electrical/electronic /programmable electronic safety-related systems - Part 3 Software requirements”, 1998.

[2] IEC 62279, "Railway Applications - Software for railway control and protection systems", 2002.

[3] 철도시설 안전기준에 관한 규칙 (국토해양부령 제 356호, 2011.6.7)

[4] 황종규, 조현정, "열차제어시스템 소프트웨어 안전성 확인을 위한 코딩규칙 테스팅 자동화 도구의 개발", 한국철도학회논문집 제12권 제1호, 2009년.

[5] 한국철도기술연구원, “열차제어시스템 안전성능 평가 및 사고방지기술 개발”, 국토해양부 국가연구개발사업 최종보고서, 2011년 6월.

[6] MISRA-C Coding Standard, MISRA(Motor Industry Software Reliability Association), 2004.

[7] 황종규, 조현정, 김형신, “열차제어시스템 소프트웨어 안전성 평가도구의 설계”, 한국철도학회 논문집, 제11권 제2호, pp.139-144, 2008. 4.

[8] 황종규, 조현정, "국제표준기반 열차제어시스템 소프트웨어의 정적 테스팅지원 도구의 개발", 대한전기학회논문집 제58P권 제2호, 2009년 6월.

[9] T.J. McCabe., “A Complexity Measure”, IEEE Trans. on Software Engineering, Vol. SE-2, No. 4, Dec. 1976.

[10] 황종규, 조현정, 김용규, "열차제어시스템 소프트웨어 Metric 분석 자동화 도구 개발", 한국철도학회 논문집 제12권 제4호, 2009년 8월.

[11] Atolic AB, “White paper, “Improving software quality with static source code analysis”, July, 2011.

[12] M. Fewstar, D. Graham, “Software Testing Automation: Effective use of test execution tools”, ACM Press, Addison Wesley, 1999.

[13] J.D. Lawrence, "Software qualification in safety applications", Reliability Engineering & System Safety, Vol. 70, No. 2., pp. 167-184, 2000.

저 자 소 개

황 종 규 (黃 宗 奎)

1994년 건국대학교 전기공학과 졸업. 1996년 동 대학원 석사졸업. 2005년 한양대학교 전자통신전파공학과 박사졸업. 1995년 ~ 현재 한국철도기술연구원 책임연구원. 2011~2012 Virginia Commonwealth Univ. 방문연구원

Tel : 031-460-5438
Fax : 031-460-5449
E-mail : jghwang@krri.re.kr

조 현 정 (趙 賢 庭)

2003년 한국항공대학교 항공전자공학과 졸업. 2005년 광주과학기술원(GIST) 정보통신공학과 졸업. 2005년~현재 한국철도기술연구원 선임연구원.

Tel : 031-460-5458
Fax : 031-460-5449
E-mail : hjjo@krri.re.kr

정 락 교 (鄭 樂 敎)

1991년 2월 인하대학교 전기공학과 졸업. 1999년 8월 동 대학원 전기공학과 졸업(석사). 2005년 2월 동 대학원 전기공학과 졸업(박사). 1990년 12월~1994년 12월 한진중공업 사원. 1995년 1월~현재 한국철도기술연구원 수요응답형교통연구단 단장(책임연구원)

Tel : 031-460-5725
Fax : 031-460-5036
E-mail : rgjeong@krri.re.kr