열차제어시스템을 위한 바이탈 소프트웨어 테스팅 자동화 도구의 제안

황종규, 조현정, 정의진
한국철도기술연구원

Automation Scheme of S/W Testing Tool for Railway Signaling System

Hwang Jong-gyu, Jo Hyun-jeong and Jeong Eui-jin
Korea Railroad Research Institute(KRRI)

Abstract - A S/W testing for vital railway signaling system have been important because of the increase of software usage for signaling. And also the safety of vital signaling system is required by int'l std. such as IEC 61508. While much efforts have been reported to improve electronic hardware's safety, not so much systematic approaches to assessment software's safety. In this paper, we propose a automation schemen of software testing tool for railway signaling system. From that, we show the functional architecture and internal components of the tool.

1. 서 론

열차제어시스템은 최근 기존의 기계적 장치로부터 컴퓨터시스템으로 전환되고 있으며, 소프트웨어에의 의존성이 급격하게 증가하고 있다. 최근의 컴퓨터 기술의 발달에 따라 지능화 및 자동화를 위해 열차제어시스템의 소프트웨어가 더욱 복잡해지게 되면서, 시스템에서 소프트웨어가 차지하는 비중이 더욱 증대되고 있다. 열차제어시스템 소프트웨어의 크기와 복잡도는 하드웨어의 발달 속도보다는 느리지만, 점차적으로 규모가 커지며, 복잡도도 증가할 것을 예상된다. 이에 따라 임베디드화 된 소프트웨어의 신뢰성과 안전성을 검증하는 것이 중요한 문제로 대두되기 시작했다.

소프트웨어의 안전성은 주로 소프트웨어의 개발초기 단계인 소프트웨어 설계 단계에서 안전성 활동을 수행함으로써 이루어진다. 이처럼 소프트웨어의 개발초기 단계에서의 안전성 활동에는 다양한 기법들이 적용되고 있으나, 개발이 완료된 이후에는 추가적인 안전성에 대한 테스팅은 자동화되어 있지 않으며 테스트 수행자에 의해 임의로 수행되고 있다. 이 단계에서 자동화된 테스팅 도구가 있다면, 테스팅 업무의 단축과 테스팅 결과의 신뢰성을 높일 수 있어 큰 도움이 될 것으로 예상된다[5][6].

본 논문에서는 열차제어시스템 소프트웨어를 자동으로 테스팅 할 수 있는 도구의 구조를 제안한다. 이를 위하여 철도소프트웨어 안전성관련 국제표준을 분석하였으며[1][2], 이를 토대로 테스팅 자동화 항목을 도출하였다. 그리고 도구에 구현된 각 테스팅 항목들에 대한 간략한 내용을 설명한다.

2. 열차제어시스템 소프트웨어 테스팅

열차제어시스템 임베디드 소프트웨어는 개발초기부터 테스팅 과정을 통해 버그를 확인하여 품질비용을 낮출 수 있다. 하지만 임베디드 소프트웨어는 하드웨어에 의존적이어서 자동화된 테스팅이 필요함에도 불구하고 매우 어렵다. 따라서 산업용 임베디드 시스템의 소프트웨어 테스팅을 위한 상용화된 자동화 툴이 일부 소개되고 있지만, 특정항목에 대한 테스팅 자동화 툴에 그치고 있다. 또한 철도시스템의 경우 IEC에서 소프트웨어 안전요구사항을 별도로 요구하고 있어 상용화된 자동화 도구들이 이러한 요구사항들을 만족하지 못하고 있다[4].

본 논문에서의 자동화된 테스팅 도구를 설계하기 위해 그림1처럼 철도시스템 소프트웨어의 안전성 규격인 IEC62278과 IEC62279 규격의 분석을 통해 열차제어시스템 소프트웨어 안전성 평가를 가이드라인을 작성하였으며[1]-[3], 이 가이드라인에서의 테스팅 항목 중 반드시 자동화를 하여야할 테스팅 항목 14가지를 도출하였다.

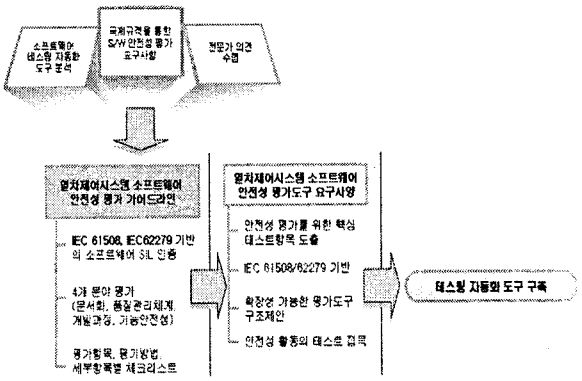

그림 1. 테스팅 자동화 도구의 개요

표 1. 열차제어시스템 S/W 테스팅 항목 및 상용 테스팅 도구 비교

테스팅 항목	기존도구
① Performance Testing	TestRealTime, Load Tester, Pure Load, Intel Vtune
② Boundary Value Analysis	CodeScroll
③ Equivalence Classes	
④ Design & Coding Standard	CodeScroll, LDRA
⑤ Control Flow Testing	CodeScroll, LDRA
⑥ Data Flow Testing	CodeScroll
⑦ Fagan Inspection	ReviewPro
⑧ Symbolic Execution	C++ Test
⑨ Checklist	CodeWizard
⑩ Metric 지원	CMT++
⑪ Decision Table	
⑫ Fault Tree Analysis	Relax
⑬ Probabilistic Analysis	
⑭ Impact Analysis	Changer Miner

표 1은 이러한 연구과정을 통해 도출한 열차제어시스템 소프트웨어의 테스팅을 위한 14가지 항목을 나타낸 것이며, 각 테스팅 항목별 상용화되어 있는 테스팅 도구를 나타낸 것이다. 표에서와 같이 IEC 규격에서 요구하는 열차제어시스템 소프트웨어 안전성 테스팅을 위해서는 다양한 항목들의 시험이 필요하지만 이러한 각 항목들을 모두 커버할 수 있는 상용화된 툴이 없으며, 또한 상용화된 몇 개의 툴을 동시에 사용하여야 테스팅이 가능한 실정이다.

본 연구에서는 국제규격에서 요구하는 열차제어시스템 소프트웨어의 안전성 평가를 위한 자동화된 도구의 구조 및 각 항목들에 대한 개략적인 내용들을 설명한다.

3. 제안 자동화 도구의 구조

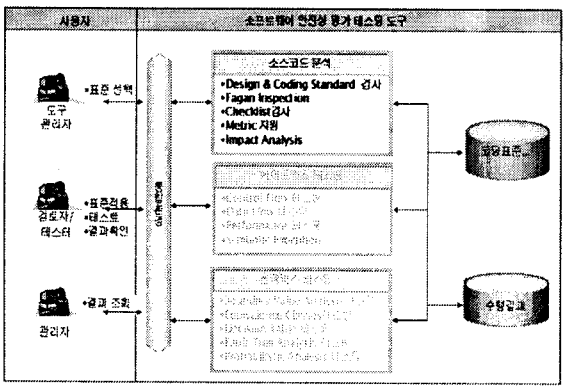

그림 2. 개발도구 전체구조

그림 1에서 나타낸 것처럼 철도시스템 소프트웨어 안전성 요구사항이 제시된 국제규격인 IEC 61508과 IEC 62279의 분석을 토대로 열차제어시스템 소프트웨어 안전성 평가를 위한 가이드라인을 작성하였다[3]. 이 평가 가이드라인을 통한 안전성 평가 중 자동화 가능한 부분은 표1과 같이 14가지 항목을 도출하여 개발 중에 있다. 그림 2는 이러한 열차제어시스템 안전성 평가 테스팅 자동화 도구의 전체 구조를 나타낸 것이다.

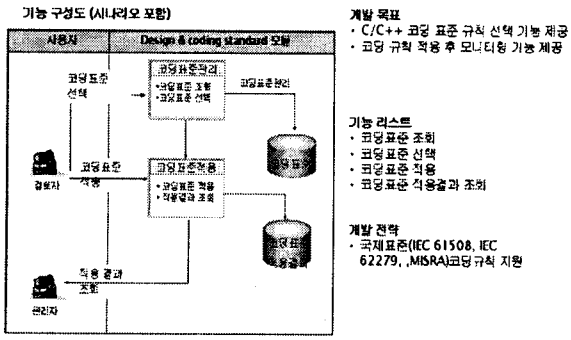

그림 3. Design & Coding Standard 모듈 기능구성도

그림에서와 같이 개발하고자 하는 테스팅 자동화도구는 소스코드 분석 모듈, 화이트박스 테스팅 모듈, 블랙박스 테스팅 모듈이라는 세 가지 하부 모듈로 구분할 수 있으며, 본 연구에서는 이들 각 하부 모듈들을 순차적으로 개발할 계획이며, 현재는 그림 2에서 하이라이트 된 소스코드 분석 모듈이 우선적으로 개발이 진행되고 있다. 소스코드 분석 모듈은 소스코드 자체를 통한 테스팅이 수행되어지고, 화이트박스 테스팅이나 블랙박스 테스팅 모듈은 소프트웨어가 임베디드 된 시스템을 통한 통합적인 테스팅이 수행되어지게 된다. 현재 개발이 진행중인 소스코드 분석 모듈에는 다음과 같은 5가지의 테스팅 항목이 있다. 이들 5가지 테스팅 항목 중 본 논문에서는 몇 가지 항목에 대한 설계 내용을 간략하게 설명하고자 한다.

- Design & Coding Standard 검사
- Fagan Inspection
- Checklist 검사
- Metric 지원
- Impact Analysis

① Design & Coding Standard 검사

'Design & Coding Standard' 검사는 소스코드를 분석해서 해당 소프트웨어가 준수하여야 할 코딩 규격이나 표준에 적합하게 코딩되었는지 여부를 검사하는 모듈로서, 코딩규격에 부적합한 부분을 검사하여 어떤 코딩 규격에 부적합한지를 제시함으로써 열차제어시스템 소프트웨어의 개발과정 및 테스팅 과정에 활용할 수 있도록 하였다. 열차제어시스템 소프트웨어의 코딩규칙은 현재 국내외 규격으로 정해진 것이 없지만, 그림 3에서와 같이 IEC 61508/62279 소프트웨어 안전관련 국제 규격 중 철도시스템 소프트웨어가 갖추어야 될 코딩 규칙을 도출하여 이를 검사할 코딩 규격으로 활용하였으며, 또한 자동차 임베디드 소프트웨어분야의 대표적인 코딩 기준인 MISRA 코딩규칙을 본 개발 도구에서 검사할 수 있도록 하였다. MISRA 코딩규칙은 자동차분야 소프트웨어를 위한 규칙이지만 다른 산업용 임베디드 소프트웨어의 코딩규칙으로 일반적으로 활용되고 있다.

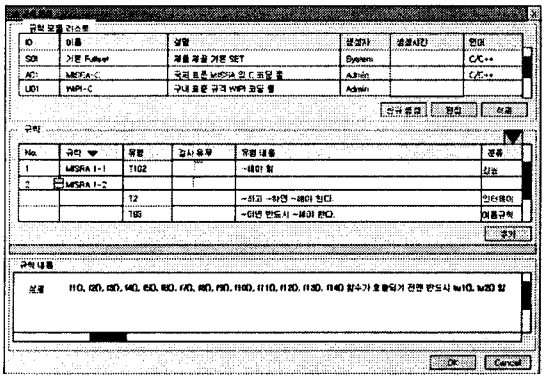

그림 4. 룰셋 추가 및 변경

이러한 기본적인 코딩규칙을 개발도구에 내장하였으며, 시험자가 임의로 검사하고자 하는 코딩규칙을 선택, 해제 또는 임의로 편집할 수 있도록 코딩규칙 룰셋 편집 모듈을 구현하여 본 개발 모듈의 사용자 편의성과 확장성을 갖도록 하였다. 그림 4는 이러한 코딩 룰셋 편집 모듈 중 룰셋 추가 및 변경 윈도우를 나타낸 것이다.

② Checklist 검사

'Checklist 검사' 항목은 열차제어시스템 소프트웨어 안전성 평가를 위한 검사항목들을 도구화하여 테스팅 상태 및 결과를 일괄적으로 관리하기 위한 모듈로서, 다른 '소스코드 분석' 모듈과 분리시켜 별도의 모듈로 개발하고 있다. 이 항목은 테스팅 상태 등을 소스코드로 개발자나 시험자 등이 동시에 활용할 수 있도록 웹 버전으로 개발 중이며, 체크리스트에는 본 논문 2장에서 언급한 '열차제어시스템 안전성 평가 가이드라인' 을 본 개발도구의 체크리스트로 활용하고자 한다. '열차제어시스템 안전성 평가 가이드라인' 은 IEC 61508/62279를 기준으로 평가항목들이 도출된 것으로 이를 체크리스트 항목으로 활용함으로써 본 개발도구의 활용성을 높이도

록 하였다. 이 체크리스트 검사 항목도 앞에서 설명한 'Design & Coding Standard 검사' 항목과 마찬가지로 체크리스트를 입력, 수정, 삭제 등을 할 수 있는 GUI를 갖도록 개발하고 있다.

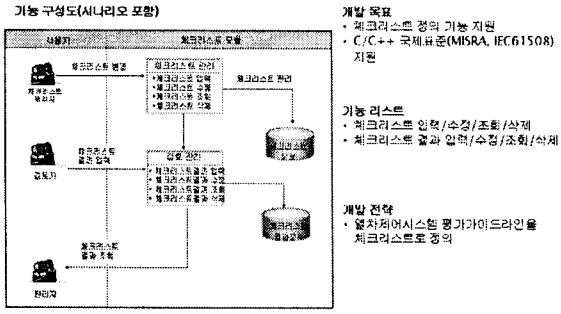

그림 5. Checklist 모듈 기능구성도

③ Impact Analysis

'Impact Analysis' 모듈은 소프트웨어의 개발단계에서 활용할 수 있는 항목으로 개발한 소프트웨어에서 임의의 한 부분을 수정할 경우 수정한 소스코드에 의해 영향을 미치는 부분이 전체 소스코드에서 어는 부분에 해당하는지를 찾아주는 모듈로서, 소프트웨어 개발과정에서는 반드시 필요한 부분이다. 또한 소프트웨어의 개발이 완료된 이후 운용도중 일부모듈의 추가 또는 변경이 발생할 경우에도 추가 또는 변경 코드에 의해 전체 코드에 영향을 미치는 부분을 추적하여 관리해 주는 기능을 갖는다.

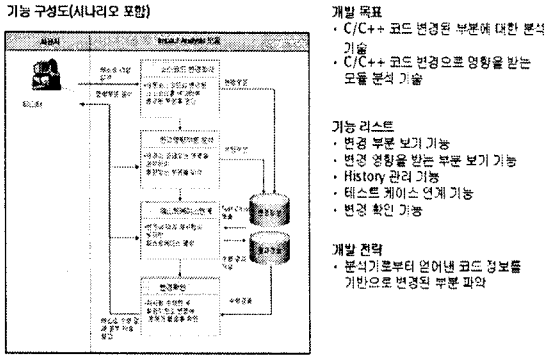

그림 6. Impact Analysis 모듈 기능구성도

또한 이 테스팅 항목은 임의의 코드를 변경할 경우 이 변경 코드에 의해 영향을 미치는 다른 부분을 찾는 기능을 수행함으로 인해, 이를 위해서는 소프트웨어의 전체 구조에 대한 분석이 이루어져야 한다. 따라서 이러한 소프트웨어 전체구조의 분석결과를 바탕으로 테스트케이스의 생성에 활용할 수 있다. 즉, 본 모듈은 소프트웨어의 개발과정에 유용하게 활용될 수 있지만, 테스팅 단계에서도 테스트케이스 생성 등에 효율적으로 활용할 수 있다.

④ Metrics 모듈

'Metrics 모듈 '은 소프트웨어의 복잡도, 응집도, 결합도 같은 메트릭을 분석하는 모듈로서, IEC 61508에서 소프트웨어 메트릭을 측정하도록 요구하고 있다. 분석되는 소프트웨어의 메트릭에 대한 결과들을 통해 본 개발도구에서는 대상 소프트웨어가 적합한지 유무는 판단하지 않으며, 단지 평가 메트릭에 대한 분석결과만을 제시할 예정이다. 관련 국제규격에서도 소프트웨어 메트릭에 대한 평가를 하도록 되어있지만 이들 메트릭 각 항목별 어느 정도를 만족하여야 하는지에 대해서는 정량적으로 제시하고 있지 않고 있으며, 현재 제시된 사례도 없다. 이러한 메트릭들에 대한 판단기준은 본 논문에서의 테스팅 자동화 도구 개발과는 별개로 연구되어야 할 것으로 판단된다.

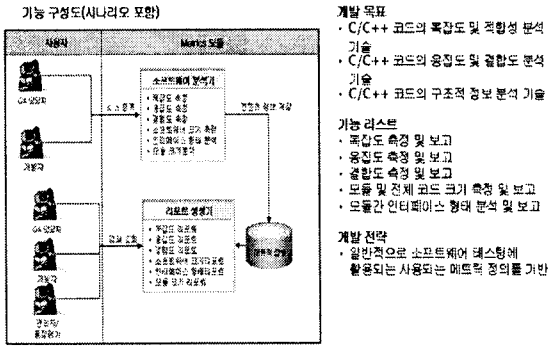

그림 7. Metrics 모듈 기능구성도

3. 결 론

철도소프트웨어 관련된 국제규격 등을 통해 열차제어시스템의 소프트웨어 안전성 테스팅의 필요성이 증대되고 있다. 열차제어시스템과 같은 임베디드 소프트웨어의 테스팅은 수작업에 의해 수행하기는 거의 불가능하며, 자동화된 도구의 지원이 필수적이다. 하지만 현재 상용화되어 있는 테스팅 도구들은 철도 소프트웨어 테스팅에 적합하지 않아 직접적인 활용이 어렵다. 본 논문에서는 현재 개발 중인 열차제어시스템 소프트웨어 테스팅을 위한 자동화 도구의 구조를 제시하였다. 특히, 전체 개발도구 중 우선적으로 설계 및 개발이 진행 중인 '소스코드 분석' 모듈의 하부 테스팅 항목들에 대한 기능구조에 대해 간략하게 설명하였다. 본 논문에서 제시한 테스팅 자동화 도구의 개발이 완료될 경우 열차제어시스템 소프트웨어의 안전성 평가에 많은 도움이 예상된다.

[참 고 문 헌]

[1] IEC 61508, "Railway Applications :The specification and demonstration of RAMS", 2002.

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

[3] 한재중, 황종규, 조현정, 김형신, "열차제어시스템 소프트웨어 안전성 평가기법", 한국철도학회 춘계학술대회, 2007.5.

[4] 황종규, 조현정, 외, "열차제어시스템 바이탈 소프트웨어 안전성 평가를 위한 테스트 도구의 검토", 대한전기학회 추계학술대회, 2008. 10.

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

[6] 한상섭, "임베디드 시스템 소프트웨어의 테스트 자동화", STEN Journal, Col. 2, 2005.