열차제어시스템 소프트웨어 Metric 분석 자동화도구 개발

Development of Metric Analysis Module for Railway Signaling Software

황종규† 조현정 정의진 김용규

Hwang, Jong-gyu† Jo, Hyun-jeong Jeong, Eui-jeong Kim, Yong-gyu

ABSTRACT

Recent advances in embedded system technology have brought more dependence on automating train control. While much efforts have been reported to improve electronic hardware's safety, not so much systematic approaches to evaluate software's safety, especially for the vital software running on board train controllers. In this paper, we have developed a software testing tool to evaluate train control system software safety, expecially "Metric Analysis" module. We have reviewed requirements in the international standards and surveyed available tools in the market. From this, we identified the S/W metric analysis module is required for software evaluation. So we have developed S/W metric analysis module for railway signaling systems.

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[1].

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[2][3].

이러한 열차제어시스템 소프트웨어의 테스팅 단계에서 기본적으로 소프트웨어의 Metric에 대한 분석이 국제규격에서 요구되고 있으며, 이를 위한 자동화된 도구가 필요하다. 본 논문에서는 열차제어시스템 소프트웨어 테스팅 자동화 도구의 한 부분으로써 프로그램과 관련된 여러 측정을 수행하고, 사용자에게 측정값을 제시하여 프로그램의 상태를 쉽게 파악할 수 있는 Metric 모듈의 설계 및 개발내용을 기술하였다. 소스코드로부터의 분석 메트릭은 소프트웨어의 복잡도, 응집도, 결합도, 크기, 모듈크기, 함수간 또는 인터페이스 형태를 분석 Metric으로 설정하였다. 이러한 소프트웨어의 Metric의 분석하기 위해 본 논문에서는 자동화 도구를 설계 및 개발하였으며 그 결과를 설명한다.

† 책임저자 : 정회원, 한국철도기술연구원, 열차제어통신연구실, 선임연구원
E-mail : jghwang@krri.re.kr
TEL : (031)460-5438 FAX : (031)469-5438

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.

본 논문에서의 자동화된 테스팅 도구를 설계하기 위해 그림1처럼 철도시스템 소프트웨어의 안전성 규격인 IEC 61508[4]과 IEC 62279[5] 규격의 분석을 통해 열차제어시스템 소프트웨어 안전성 평가를 가이드라인을 작성하였으며, 이 가이드라인에서의 테스팅 항목 중 반드시 자동화를 하여야할 테스팅 항목 14가지를 도출하였다.

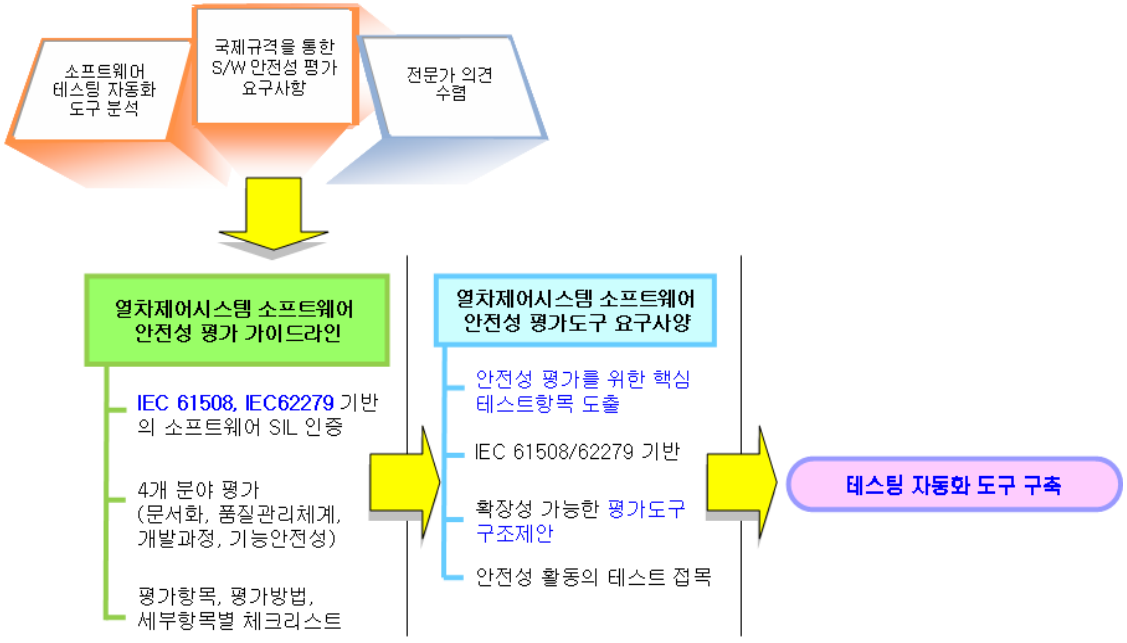


그림 1. 테스팅 자동화 도구의 개요

이처럼 테스팅 자동화 가능한 항목 중 본 논문에서는 'Metric 분석' 모듈에 대한 설계 및 개발 내용을 설명한다. '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 분석결과를 제시하도록 하였다. 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.

3. 소프트웨어 Metric 분석 모듈의 설계

3.1 소프트웨어 Metric 분석

소프트웨어 개발의 각 단계에서 필요한 소프트웨어의 특성을 개발과정 또는 테스트 이력으로부터 습

득하는 것이 아니라, 개발된 소프트웨어 자체의 구조적인 특징으로부터 이해할 때 사용하는 기술이다. 여기에는 소프트웨어의 구조적인 특성을 평가하여, 신뢰도 또는 복잡도와 같은 특성들이 요구되는 값을 만족하는지 확인하는 과정이 포함되며, 이런 결과물들은 소프트웨어의 효율적인 테스팅, 추가, 변경 작업에 기초자료로 사용된다.

최근 들어 다양한 종류의 메트릭을 소프트웨어 특성 파악에 적용하는 사례들이 증가하고 있다. 특히 자동화된 소프트웨어 분석 도구, 테스팅 도구, 버그 추적 도구 등에는 해당 작업을 도울 수 있는 지표들을 임의로 설정해서 메트릭 형태로 보여주는 기능을 탑재하고 있다. 이중, 일반적으로 사용되는 지표들은 다음과 같다.

- 복잡도 : 본 평가관점은 trade-off를 위한 생명주기 전반에 적용되며, 프로그램 제어 그래프 복잡도의 기본이 되며, 순환하는 수(Cyclomatic Number)에 의해 표현된다.
- 임의 모듈(이용할 수 있는)을 실행시키는 방법의 수: 많은 모듈에 대한 접근이 가능할수록 그만큼 디버깅도 가능해 진다.
- 소프트웨어 과학: 본 평가관점은 연산자들의 수를 계수하여 프로그램의 길이를 측정하는 것이다. 본 평가의 결과는 개발자원의 평가와 복잡도의 측정을 제공한다.
- 모듈당 진출입의 수: 진입과 탈출의 포인트 수의 최소화는 구조화 설계와 프로그래밍 기술들의 핵심이다.

이런 지표들에 대한 측정은 소스코드의 양이 늘어남에 따라 소프트웨어 평가자가 매뉴얼로 작성하기 불가능하다. 또한 열차제어시스템 같은 경우 관련된 국제규격에서 필수적으로 소프트웨어의 Metric들을 분석하도록 하고 있지만, 이 또한 수작업으로는 불가능하다. 따라서 이러한 소프트웨어 Metric들을 자동으로 분석할 수 있는 도구를 개발하였다.

3.2 Metric 분석 모듈의 설계

본 절에서는 열차제어시스템 소프트웨어 테스팅 자동화 도구의 한 부분으로써 프로그램과 관련된 여러 측정을 수행하고, 사용자에게 측정값을 제시하여 프로그램의 상태를 쉽게 파악할 수 있는 Metric 모듈의 설계에 대하여 기술한다. 소스코드로부터의 분석 메트릭은 그림 2와 같이 소프트웨어의 복잡도, 응집도, 결합도, 크기, 모듈크기, 함수간 또는 인터페이스 형태를 분석 Metric으로 설정하였다.

표 1. Metric 모듈 기능 리스트

기능명	기능설명
복잡도 측정	등록된 소스를 기반으로 모듈별 또는 함수별 복잡도를 측정한다.
복잡도 메트릭 리포트	측정된 복잡도를 리포트로 보여준다.
응집도 측정	등록된 소스를 기반으로 모듈간 또는 함수간 응집도를 측정한다.
응집도 메트릭 리포트	측정된 응집도를 리포트로 보여준다.
결합도 측정	등록된 소스를 기반으로 모듈간 또는 함수간 결합도를 측정한다.
결합도 메트릭 리포트	측정된 결합도를 리포트로 보여준다.
소프트웨어 크기 측정	등록된 소프트웨어의 크기를 측정한다.
소프트웨어 크기 리포트	측정된 소프트웨어 크기를 리포트로 보여준다.
인터페이스 형태 분석	등록된 소스를 기반으로 모듈별 또는 함수별 인터페이스 형태 분석한다.
인터페이스 형태 리포트	분석된 인터페이스 형태를 리포트로 보여준다.
모듈 크기 측정	등록된 소스를 기반으로 모듈 크기를 측정한다.
모듈 크기 리포트	측정된 모듈 크기를 리포트로 보여준다.

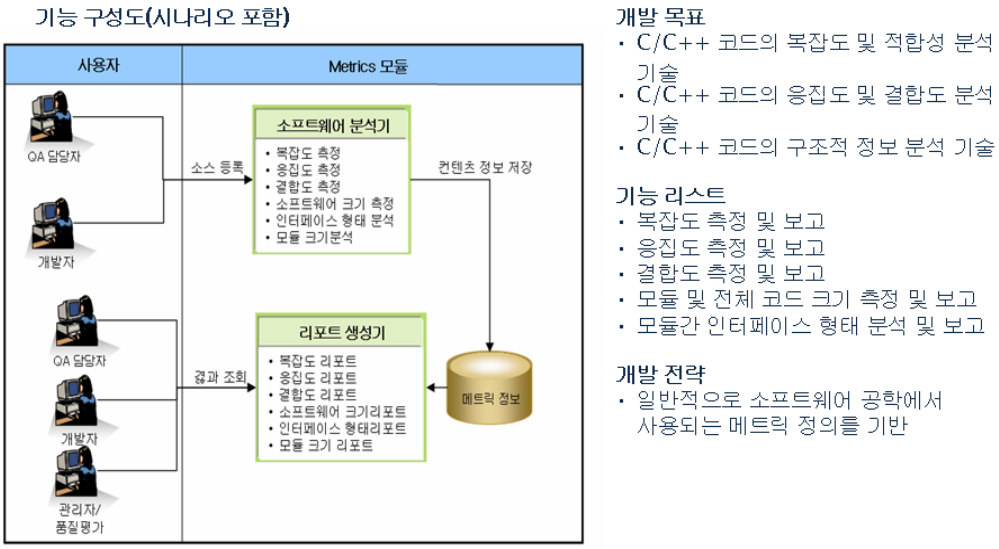


그림 2. Metric 모듈 개요

가. 복잡도 분석

함수에 대한 복잡도를 나타내는 대표적인 메트릭인 Cyclomatic Complexity 를 통해 분석하도록 하였으며, 다음과 같은 수식을 통해 분석되게 된다.

복잡도 = 전체 엣지 개수 - (전체 노드 개수 - 종료 노드 개수) + 1

즉, 함수의 조건문이 많아질수록 엣지가 많아져서 전체적인 복잡도가 높아지게 된다. 본 논문을 통해 개발한 자동화 도구에서는 분석된 함수 정보를 토대로 복잡도를 구한다. 그리고, 다음과 같이 복잡도의 값에 따라 진단메시지를 사용자에게 제공한다.

표 2. 복잡도 값 범위

복잡도 값 범위	설명
1 ~ 10	단순함 (Simple program)
11 ~ 20	약간 복잡함 (More complex)
21~50	복잡함 (Complex)
51 이상	매우 복잡함 (High risk)

나. 응집도 분석

소프트웨어 코드는 높은 응집도와 낮은 결합도를 가져야 한다는 것은 일반적으로 통용되는 개념이다. 응집도를 측정하는 방법은 여러 가지가 있지만, 본 도구의 개발에 있어서는, 일반적으로 널리 사용되는 LCOM4 (Lack of COhesion in Methods) 메트릭을 기반으로 분석하도록 한다. 한 모듈(클래스)내의 메소드간의 상호 연관성을 의미하며, 동일한 데이터를 사용하는 메소드들의 집합 (Connected components) 의 개수로 표현한다. 모듈 내의 함수들이 동일한 데이터에 대해 연산할 때 그것들을 하나의 그룹으로 묶게 되는데, 그 이유는 그것들 간에는 관계가 강하게 놓여져 있기 때문이다. 어떤 모듈의 그 그룹개수가 1이면 가장 높은 응집도를 갖는 경우이며, 그보다 커질수록 응집도가 낮아진다고 본다.

그림 3은 본 논문에서 적용한 응집도는 분석하는 방법인 LCOM4를 나다낸 것이다. 그림의 좌측 그래

프를 보면 A함수가 B함수를 호출하고 B함수가 x변수를 사용하고 있는데, 이것이 하나의 Connected component 가 된다. C함수와 D함수가 y 변수를 사용하고 있는데 이 또한 하나의 Connected component가 된다. 결국 좌측 그림은 두 개의 Connected component가 존재함으로써 각각 독립적으로 나뉘어질 수 있는 형태가 되며 이 모듈의 응집도는 2로 나타낸다. 반대로, 우측 그림은 응집도가 1인 모습을 나타내고 있다. 좌측 그림에서 C가 x변수를 사용함에 따라 전체가 하나의 그룹으로 묶여졌기 때문이다.

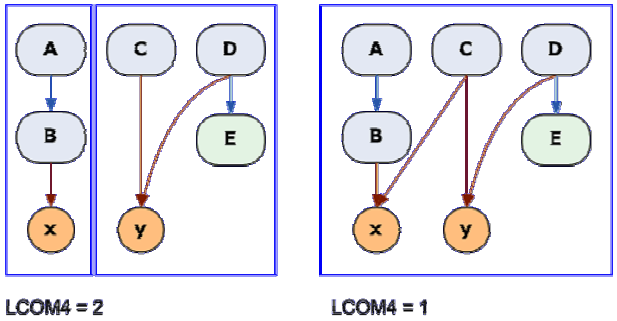


그림 3. LCOM4의 개념

이처럼 LCOM4는 본래 OOP 에 적합한 메트릭이지만 C 언어 프로그램에도 적용할 수 있도록 다음과 같이 재 정의하도록 한다. 하나의 C소스파일내에서 정의된 변수들 (include한 헤더파일내 구조체 등 변수 포함) 을 사용하는 함수들을 함께 하나의 Connected component로 규정한다. 즉, 정적분석도구에서의 응집도는 소스파일단위로 측정/보고되는 형태가 된다.

다. 결합도 분석

앞 절에서 응집도를 정의할 때 결합도는 낮을수록 보다 좋은 소프트웨어 모델이라고 소개하였다. 왜냐하면, 모듈들은 다른 모듈에 의해 접근되는 잘 정의된 인터페이스를 가져야 한다. 한 모듈의 여러 개 객체를 통해 동작되는 대신 하나의 인터페이스를 통해 접근되는 것이 더 나은 구조를 갖게 되고 결합도가 낮아지게 된다. 낮은 결합도는 복잡성을 줄이고 유지보수성 및 프로그램 가독성을 높인다. 코드가 복잡해져서 결합도가 높아지면 코드 변경사항에 대한 side-effect 범위가 커지면서 결함발생 소지가 더 높아질 수 있다.

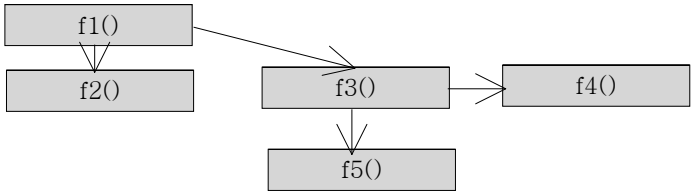


그림 4. Fan-In/Fan-Out의 개념

이 결합도 주요 메트릭으로써 CBO 또는 RFC등이 있는데 주로 OOP 프로그램에 적합한 성격을 지니고 있다. 따라서, Safety-critical 소프트웨어의 개발에 많이 사용되는 C 언어를 위해서는, 결합도를 구조적 방법론에서 말하는 함수 각각의 Fan-In, Fan-Out 주요 메트릭으로 측정/보고하도록 한다.

Fan-In : 자신을 호출하는 모듈의 수 (caller)
Fan-Out : 자신이 호출하는 모듈의 수 (callee)

즉, 다음 그림에서 보면 f1()의 Fan-Out은 f2()와 f3()이 된다. f5() 의 Fan-In 은 f3() 이 된다.

라. 모듈 크기, 전체 코드 크기 측정 및 보고

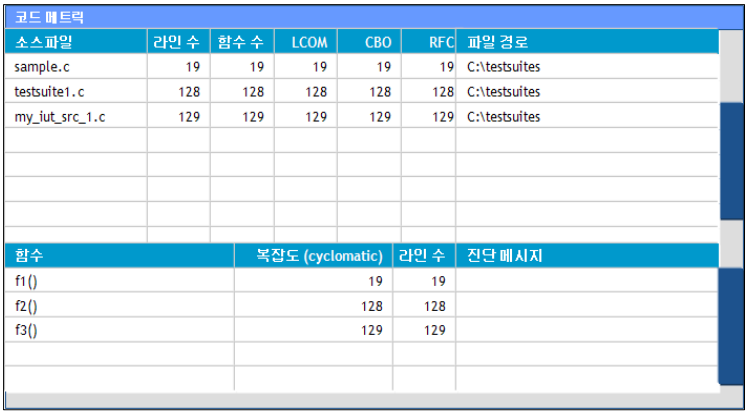
프로그램의 사이즈를 소스코드의 파일별, 함수별로 각각 분석 및 제공하도록 한다.

마. 모듈 간 인터페이스 형태 분석 및 보고

함수의 제어흐름그래프(Control Flow Graph)에서 외부 함수 호출관계를 나타냄으로써 그 함수의 의존성을 그래프 상에서 확인할 수 있다. 또한, 전체 함수간의 호출 관계를 분석하여 그래프로 제공하는데, 이를 통해 전체 함수들 간의 연관 관계 파악을 지원한다.

마. 화면 설계

이러한 Metric 분석 모듈은 소스코드에 대한 분석의 화면구성에서 Metric 분석 결과를 다음과 같은 뷰(View)를 통하여 사용자에게 전체적으로 제공하도록 하고, 모듈간 인터페이스 형태 분석 및 보고 또한 그림과 같은 뷰 형태로 제공한다.



코드 메트릭

소스파일	라인 수	함수 수	LCOM	CBO	RFC	파일 경로
sample.c	19	19	19	19	19	C:\testsuites
testsuite1.c	128	128	128	128	128	C:\testsuites
my_iut_src_1.c	129	129	129	129	129	C:\testsuites

함수	복잡도 (cyclomatic)	라인 수	진단 메시지
f1()	19	19	
f2()	128	128	
f3()	129	129	

그림 5. Metric 측정 화면

3. Metric 분석 모듈 개발

지금까지 설명한 열차제어시스템 소프트웨어를 위한 Metric 분석 모듈의 시스템 및 기능구조, 화면설계 및 코딩규격 설계내용을 바탕으로 테스팅 자동화 도구를 구현하였다. 개발한 Metric 분석 모듈은 소스코드에 대한 분석의 화면구성에서 Metric 분석 결과를 다음과 같은 뷰(View)를 통하여 사용자에게 전체적으로 제공하도록 하고, 모듈간 인터페이스 형태 분석 및 보고도 뷰 형태로 제공하도록 하였다. 그림 6은 개발한 툴을 이용한 Metric 분석 결과 중 소스코드별 복잡도 분석 결과를 나타내는 것이다. 그림 7은 이러한 복잡도 분석결과를 그림의 왼쪽 하단과 같이 그래픽 형태로 표시되도록 하는 윈도우이다. 이처럼 다양한 형태의 출력을 생성하도록 함으로써 사용자가 본 도구를 보다 효율적으로 활용할 수 있도록 하였다. 그림 6~8과 같이 복잡도에 대해 분석된 결과를 다양한 형태로 출력을 할 수 있도록 도구를 개발하였으며, 또한 응집도에 대한 분석결과도 이와 동일하게 다양한 출력을 갖도록 하였다.

4. 결론

최근 들어 컴퓨터 기술의 발달에 따라 열차제어시스템들이 컴퓨터 소프트웨어에의 의존성이 급격하게 증가하고 있으며, 이러한 기술발전에 따라 열차제어시스템 소프트웨어에 높은 신뢰성과 안전성이 요구되고 있다. 본 논문을 통해 제시한 열차제어시스템을 위한 소프트웨어 Metric 분석 모듈은 소프트웨어의 안전성 확보를 위해 필수적으로 준수해야 하는 국제규격에서 요구하는 항목들을 자동으로 분석할 수

있는 모듈을 개발한 내용을 설명하였다. 본 논문을 통해 제시한 열차제어시스템 소프트웨어 Metric 분석 자동화 도구는 기본적으로 소프트웨어 평가단계에 평가자 입장에서는 대상 소프트웨어에 요구되는 Metric의 수준을 활용되어질 수 있으며, 또한 소프트웨어 개발과정에서 활용되어질 수도 있다. 즉, 소프트웨어 개발단계에서 디버깅 과정에 본 자동화 도구를 이용할 경우 적합한 소스코드를 개발할 수 있고, 이를 통해 보다 높은 안전성을 갖는 소프트웨어가 확보될 수 있을 것으로 기대된다.

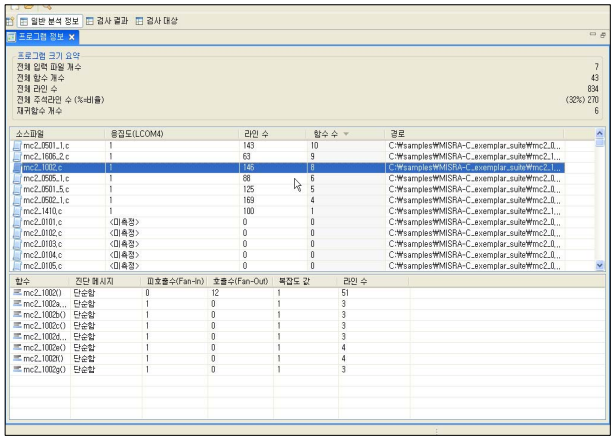


그림 6. Metric 분석 결과 윈도우-1

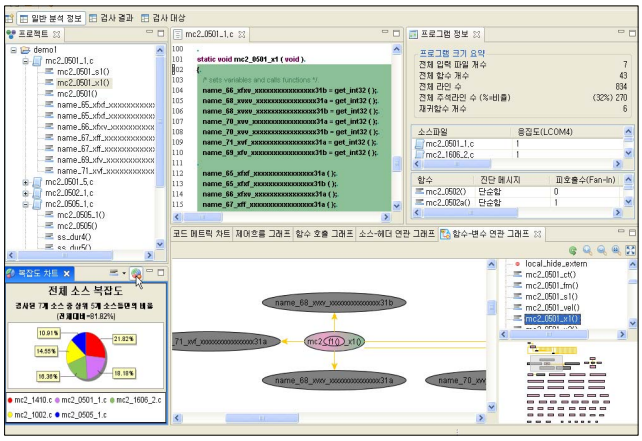


그림 7. 복잡도 분석 윈도우

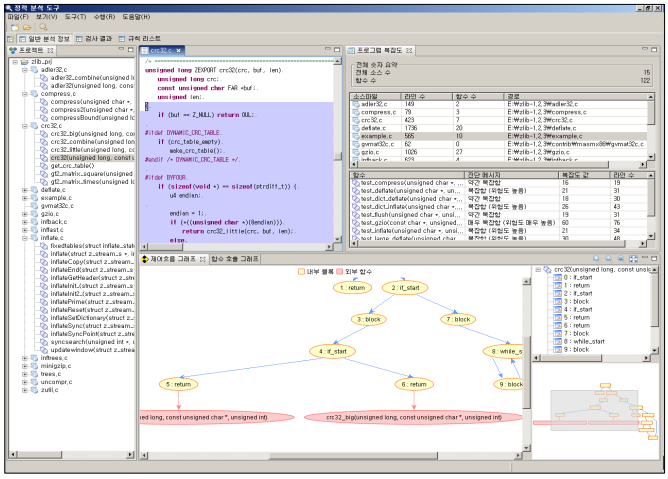


그림 8. Metric 분석 결과 윈도우-2

참고문헌

1. Zage, W.M and Zage, D.M., "Evaluating design metrics on large-scale software", IEEE Trans. om Software Eng., pp.75-81, Vol. 10, Issue 4, Jul 1993.
2. M. Fewstar, D. Graham, “Software Testing Automation: Effective use of test execution tools”, ACM Press, Addison Wesley, 1999.
3. 황종규, 조현정, 김형신, “열차제어시스템 소프트웨어 안전성 평가도구의 설계”, 한국철도학회 논문집, 제11권 제2호, pp.139-144, 2008. 4.
4. IEC 61508, "Railway Applications :The specification and demonstration of RAMS", 2002.
5. IEC 62279, "Railway Applications : Software for railway control and protection systems", 2002.