소스코드 기반 소프트웨어 품질 측정 방법 및 도구 개발

지도교수 : 채흥석

팀 명 : 졸업참쉽조

201624567 전민기

201724492 성민우

201524593 천동혁

목차

1. 과제 배경 및 목표

1.1 과제 배경

갈수록 자동차, 군사, 우주항공, 의료와 같은 다양한 분야에서 소프트웨어 사용이 증가하고 있다. 이런 분야에서는 소프트웨어의 작은 결함이 엄청난 피해를 야기시킬 수 있다. 실제사례로는 2009년 도요타의 렉서스 자동차가 전자제어장치(ECU)에 내장된 SW의 결함 때문에 급 발진하여 탑승자 전원이 사망하였다. 이 사고로 인해 900만 대의 차량이 리콜 되며 5조 원 이상의 경제적 손실을 입은 것으로 알려져 있다. 또 받음각(AOA) 센서의 SW 오류로 보잉 737 MAX가 추락하여 총 346명이 사망하였다.

이처럼 소스 코드의 작은 결함이 엄청난 피해를 발생시킬 수 있고, 그렇기 때문에 소스 코드에 대한 철저한 분석이 필요하다. 대표적으로 정적 분석 기법이 있다. 정적 분석 기법이란 소스 코드의 실행 없이, 코드의 의미를 분석해 결함을 찾아내는 코드 분석 기법이다. 대표적인 정적 분석 도구인 Polyspace를 이용해 과제를 수행하고자 한다.

1.2 과제 목표

본 졸업 과제는 소스코드 기반 소프트웨어 품질 측정 방법과 품질 측정 도구를 개발하는 것을 목표로 한다. 사용자가 품질 측정을 원하고자 하는 C 소스 코드를 입력한다. 입력된 소스 코드에 대해 Polyspace를 이용하여 분석한다. 분석한 결과를 품질 요구사항과 매핑하여 품질 측정한다.

2. 배경지식

2.1 정적 분석

정적 분석이란 코드를 실행하지 않고 수행되는 품질, 안정성, 보안을 위해 소스코드를 분석하는 소프트웨어 검증이다. 이는 동적분석을 보완할 수 있는 몇 가지 이점을 가지고 있는데 다음과 같다.

1. 오류 감지

동시성, 오염된 데이터, 데이터 흐름, 보안, 정적 및 동적 메모리와 관련된 여러 버그 클래스를 식별할 수 있다. 일부 버그는 동적 테스트로 감지하는 것이 힘들기 때문에 정적 분석을 통한 식별이 도움이 된다.

2. 낮은 비용

정적 분석에는 동적 분석에 필요한 테스트 케이스 작성이나 프로그램 실행에 의한 오버헤드가 발생하지 않기 때문에 상대적으로 저렴한 비용에 행할 수 있다.

3. 코딩 표준 준수

코드가 MISRA C, SEI CERT와 같은 코딩 표준을 준수하는지 확인할 수 있다. 코딩 표준을 준수하면 각 지침에서 제시하는 잠재적인 결함을 방지하거나 가독성을 높이고 코드의 관리를 용이하게 할 수 있다.

4. 치명적인 런타임 오류가 없음을 증명

정적 코드 분석은 소프트웨어가 치명적인 런타임 오류로 실패하지 않음을 증명할 수 있다. 이러한 분석이 가능한 도구는 코드 증명을 위해 'formal method'를 이용한다. (이는 논리계산, 형식 언어, 오토마타 이론, 프로그램 의미론 등 컴퓨터 과학 이론을 사용한 코드에 대한 수학적 분석으로서 코드의 실패 지점을 실패할 것으로 입증된 것, 실패하지 않을 것으로 입증된 것, 절대 실행되지 않을 것, 입증되지 않은 것으로 식별할 수 있다.)

2.2 Coding Rules

Coding Rule(혹은 Coding conventions)는 특정 언어로 작성된 프로그램에 대해 프로그래밍 스타일, 사례, 방법을 권장하는 일련의 지침이다. 이는 파일 구성, 들여쓰기, 주석, 선언, 명령문, 공백, 명명 규칙, 프로그래밍 관행, 아키텍처 모범 사례 등을 다루어 소스코드의 가독성을 높이고 유지관리를 용이하게 하여 소프트웨어의 품질을 높일 수 있다. 이러한 규칙은 공식적으로 문서화되거나 회사, 팀, 개인 단위로 이루어질 수도 있다. 이러한 Rules중 고품질 코드를 생산하기 위해 디자인되어 널리 채택되는 경우 Coding Standards라고 부르며 우리는 이러한 Coding Standards 중 MISRA-C 와 SEI CERT를 이용할 것이다.

3. 연구 과정

3.1 관련 도구 및 표준

3.1.1 **Polyspace**

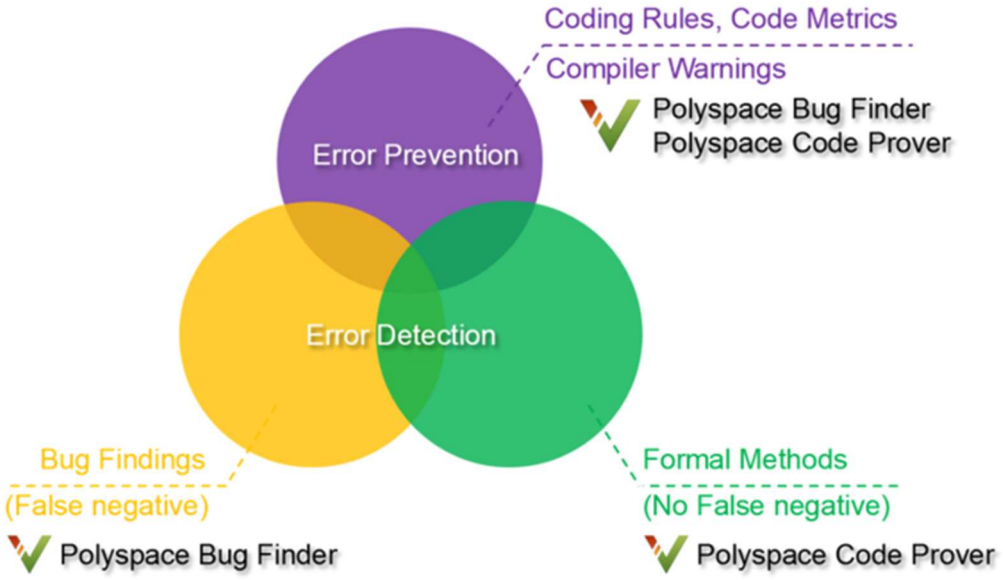

Figure 1 Polyspace 프로그램이 제공하는 기능

Polyspace는 Error prevention, Error Detection 기능을 제공한다. Error prevention은 코딩 규칙 검사를 말한다. Error Detection은 에러를 탐지하는 능력을 말한다.

Polyspcace가 분석하는 규칙은 크게 보라색 영역인 Error Prevention, 노란색 부분인 Bug Findings, 초록색 부분인 Formal Methods로 구분할 수 있다.

Error Prevention에서는 영역 에러를 미연에 방지하고자 하는 목적이 강하다. 이 분야에서 검사 가능한 항목들은 MISRA C와 같은 코딩 가이드라인에 대한 위반 사항을 찾아주거나 소프트웨어의 품질 지표로 삼을 수 있는 코드 메트릭이 있다. 위반 사항 혹은 목표 수치를 맞추지 못한 사항을 찾아 주고, 이를 수정함으로써 개발자가 코딩 규칙이나 코드 메트릭을 맞추기 위한 코딩 방식을 배우는데 매우 큰 도움을 줄 수 있다. 즉, 올바른 코딩 방식을 배움으로써 코딩 시 적용하여 에러를 미연에 방지할 수 있다. 그런데 사람들이 100개가 넘는 코딩 규칙 같은 것들을 외워서 위반 사항이 있는지 체크하기가 매우 어렵기 때문에 도구의 도움을 받아 위반 사항 혹은 목표 수치를 달성하지 못한 부분의 코드를 확인하는 것이다. 노란색 원 부분인 bug findings 부분은 코딩 규칙 위반 만으로는 찾아낼 수 없었던 버그들을 탐지할 수 있는 부분이다.

Bug Findings 부분은 소프트웨어 버그 발견을 의미한다. 위의 코딩 규칙 검사만으로 탐지할 수 없었던 버그들도 찾아낼 수 있다. Polyspace Bug Finder가 이 부분에 특화되어 있다.

Formal Methods 부분은 런타임 에러가 없음을 증명할 수 있는 것을 의미한다. 이 부분에서는 Polyspace Code Prover가 특화되어 있다.

다음 표로 Polyspace의 세부 도구인 Polyspace Bug Finder Polyspace Code Prover에 대해 자세히 비교해보았다.

	Polyspace Bug Finder	**Polyspace Code Prover**
목적	버그 및 오류 탐지	정적 코드 검증 및 증명
특징	프로그램의 버그와 오류를 찾고 수정하는 데 중점을 둔다. 코드를 정적으로 분석하여 잠재적인 문제를 식별하며, 배열 범위 초과, 널 포인터 참조, 정의되지 않은 변수 사용 등을 포함한 일반적인 프로그래밍 오류를 탐지한다.	프로그램의 정확성을 증명하고, 코드에 대한 formal methods를 수행한다. 코드의 모든 실행 경로를 검사하여 특정한 속성이 항상 참임을 증명하고, 버그와 오류 없음을 검증한다.

요약하면, Polyspace Bug Finder는 코드의 일반적인 오류와 버그를 탐지하고 수정하는 데 초점을 두고, Polyspace Code Prover는 코드의 정확성을 증명하고 formal methods에 사용된다. 프로젝트에서는 코드의 증명보다는 버그를 탐지하고 품질을 측정하는데 초점을 두고 있으므로 Polyspace Bug Finder를 사용할 것이다.

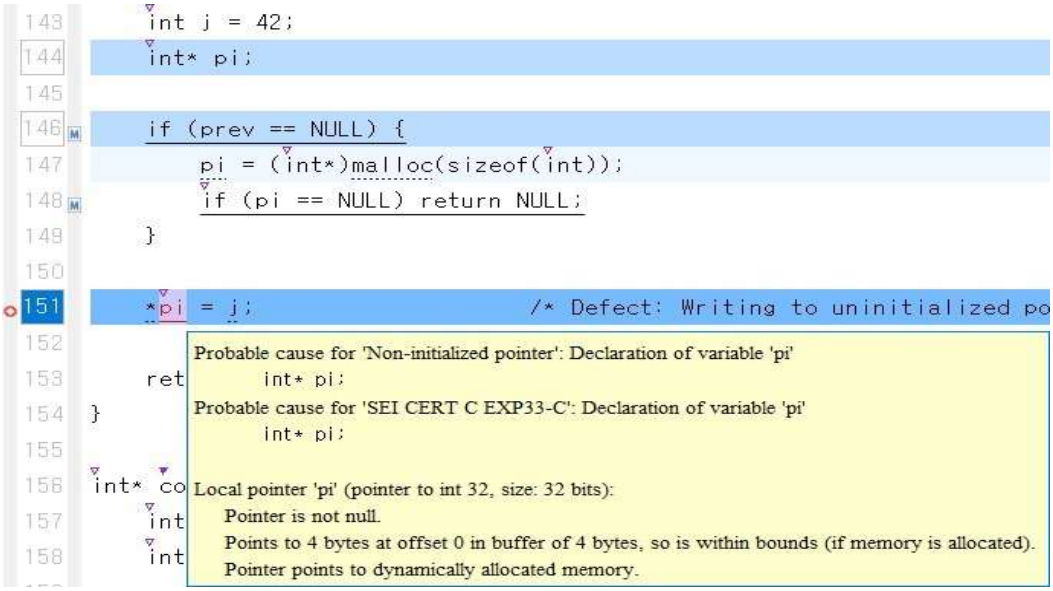

Figure 2 Polyspace Bug Finder를 이용한 Coding Rule 규칙 검사

3.1.2 MISRA-C

MISRA C: 2012는 MISRA에서 만든 C 언어용 개발 지침으로서 실수를 줄이거나 제거하기 위한 규칙을 정의한다. 본 과제에서는 2012년도 판인 Third Edition을 이용한다. Third Edition에서 개발 지침은 크게 Directives와 Rules로 분류되며 이는 코드만을 가지고 지침의 준수 여부를 확인할 수 있는지 여부로 나뉜다. Directives는 코드만으로 불가능하여 설계문서 등이 필요하고 Rules는 코드만으로 준수하였는지 확인할 수 있는 지침이다. 우리는 별도의 문서 없이 C 소스코드의 정적 분석 도구를 통한 분석 결과를 활용할 예정이므로 둘 중 Rules에 집중할 것이다.

또한 각 Rules는 어떠한 문제에 대한 지침인지에 따라 분류되어 목차에 따른 식별자를 가지며 이하와 같은 항목을 가진다.

항목	세부항목	상세
Category (지침을 준수해야하는 정도)	Mandatory	무조건 준수해야 함
	Required	준수해야 함. 하지 않는다면 'deviation'에 대한 공식적인 기술이 필요함.
	Advisory	준수할 것을 권장. 공식적인 'deviation'이 필요하지는 않으나 비준수에 대한 문서화를 위해 대체 조치 필요.
Decidability (정적 분석 도구가 코드가 지침을 준수하는지에 답할 수 있는지 여부)	Decidable	모든 경우에 예, 아니오 로 대답할 수 있음.
	Undecidable	모든 경우에 예, 아니오 로 대답할 수 없음. (ex 런타임 속성에 의존하는 경우)
Scope (준수 여부를 파악하기 위해 확인해야 하는 범위)	System	모든 소스코드를 분석하여야 정확하게 확인 가능.
	Single Translation Unit	Translation Unit(C 소스코드가 전처리 과정을 거친 결과) 단위로 개별적인 확인 가능.

3.1.3 SEI CERT

SEI CERT C Coding Standard는 C 언어의 보안 코딩을 위한 규칙을 제공한다. 이는 악용 가능한 취약점으로 이어지는 정의되지 않은 동작을 제거하는 등 안전하고 신뢰할 수 있는 시스템을 개발하는 것을 목표로 한다.

각각의 지침은 제목, 설명, 비 준수 코드 예제 및 준수 솔루션으로 이루어져 있다. 이 지침들은 규칙과 권고사항으로 나누어지는데 규칙은 세 가지 조건을 만족하여야 한다. 각각은 다음과 같다.

1. 위반 시 시스템의 안전성, 신뢰성, 보안성에 악영향을 줄 수 있는 결함이 발생할 수 있을 것
2. 소스 코드의 annotation이나 assumption에 의존하지 않을 것
3. 적합성을 자동 분석 도구나 수동 검사 등으로 결정할 수 있을 것

권고사항의 경우 충족시키면 시스템의 안정성, 신뢰성, 보안성을 향상시킬 수 있지만 위의 세 가지 중에서 만족하지 못하는 것이 있을 때 해당된다.

3.1.4 ISO/IEC 25010:2011 Quality Model

ISO/IEC Quality Model은 ISO에서 제정한 제품 품질에 대한 모델로서 본 과제에서 측정하고자 하는 8가지 품질 속성을 정의한 것이다. ISO/IEC Quality Model의 품질 속성은 다음과 같다.

품질 속성	설명
Functional Suitability	제품이나 시스템이 특정 조건에서 사용될 때 명시적, 암시적 요구사항을 충족시키는 기능을 제공하는 정도.
Perfomance Efficiency	명시된 조건에서 사용된 자원의 양에 대한 성능.
Compatibility	동일한 하드웨어, 소프트웨어 환경을 공유하며 제품, 시스템 또는 구성요소가 다른 제품, 시스템 또는 구성요소와 정보를 교환하거나 필요한 기능을 수행할 수 있는 정도
Usability	제품이나 시스템이 특정 사용자가 목적을 달성하기 위해 효과, 효율성, 만족을 얻으며 사용할 수 있는 정도
Reliability	지정된 기간 동안 지정된 조건에서 지정된 기능을 수행하는 정도
Security	정보 및 데이터를 보호하여 사람이나 다른 제품, 시스템이 권한에 적합한 데이터 액세스 수준을 갖도록 하는 정도.
Maintainability	관리자가 제품 또는 시스템을 수정할 수 있는 효과 및

	효율성의 정도.
Portability	하나의 하드웨어, 소프트웨어 혹은 다른 사용 환경에서 다른 환경으로 이전될 수 있는 효과 및 효율성의 정도.

3.2 연구 내용

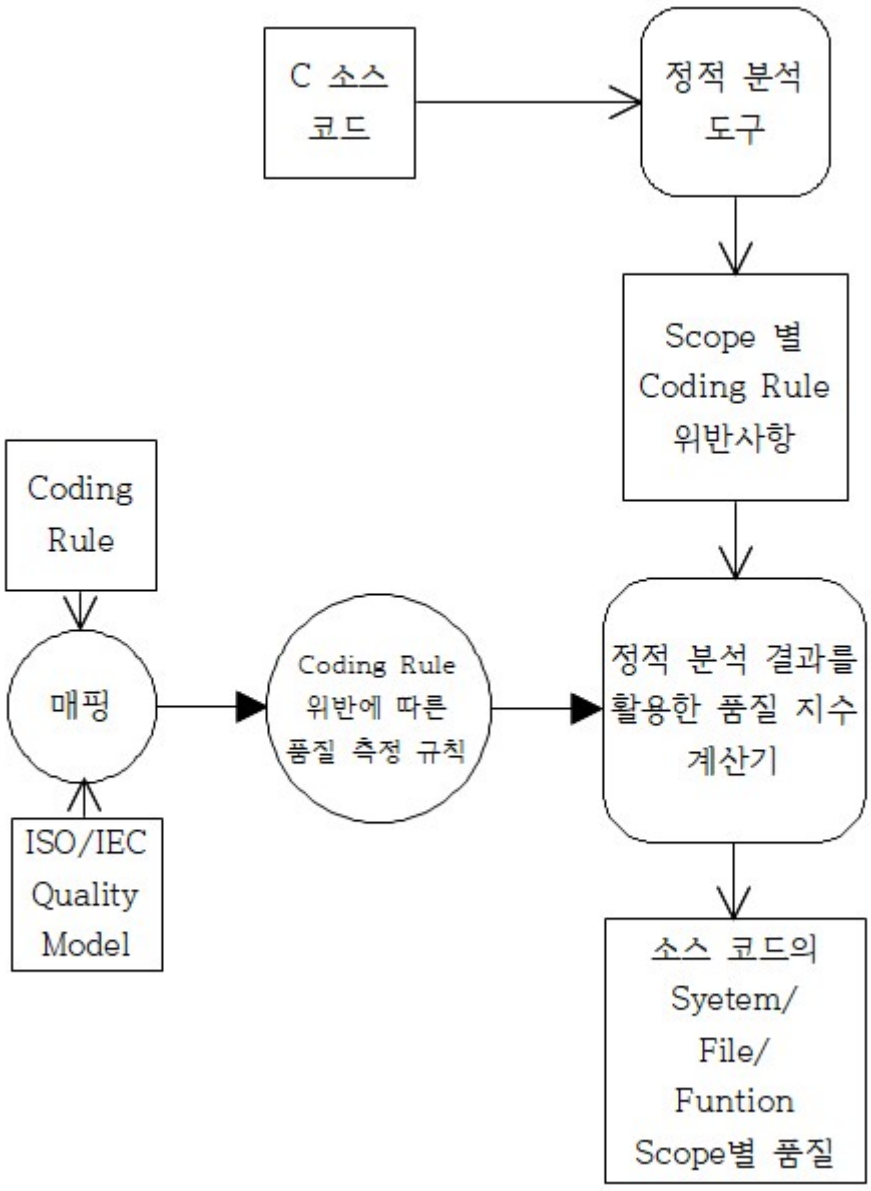

Figure 3 소스코드로부터 품질지수를 측정하는 과정

본 연구에서는 C 기반 소스코드로부터 ISO/IEC Quality Model에 기반한 품질지수를 측정하는 것을 목표로 한다. 위 그림은 대략적인 과정을 나타내고 있다.

3.2.1 C 기반 소스 코드 확보

먼저 정적 분석의 결과를 이용하기 위해서는 정적 분석을 행하기 위한 소스 코드가 필요하다. 따라서 몇 가지 기준에 따라 오픈소스 프로젝트를 찾아보았다. 각각의 기준은 다음과 같다.

1. C 언어 기반일 것

2. 충분히 큰 규모이며 현재에도 갱신이 이루어지고 있을 것

C 언어인 이유는 우리가 사용하고자 하는 Coding Rule이 MISRA-C와 SEI CERT로서 C 언어에 대한 규칙이기 때문이며, 충분한 양의 정적 분석 결과를 얻기 위해서는 규모 역시 필요하며 양질의, 유의미한 분석 결과를 얻기 위해서는 지속적인 갱신이 이루어져 현재에도 사용되고 있는 것이 좋다고 판단했기 때문이다.

이러한 기준의 따라 이하의 세 오픈소스 프로젝트를 찾을 수 있었다.

1. FFmpeg

https://github.com/FFmpeg/FFmpeg

FFmpeg는 오디오, 비디오, 자막 및 관련 메타데이터 등 멀티미디어 데이터를 처리하기 위한 라이브러리 및 도구 모음이다. 이 프로젝트는 1만번 이상 fork 되었으며 다음과 같이 꾸준한 commit을 확인할 수 있었다.

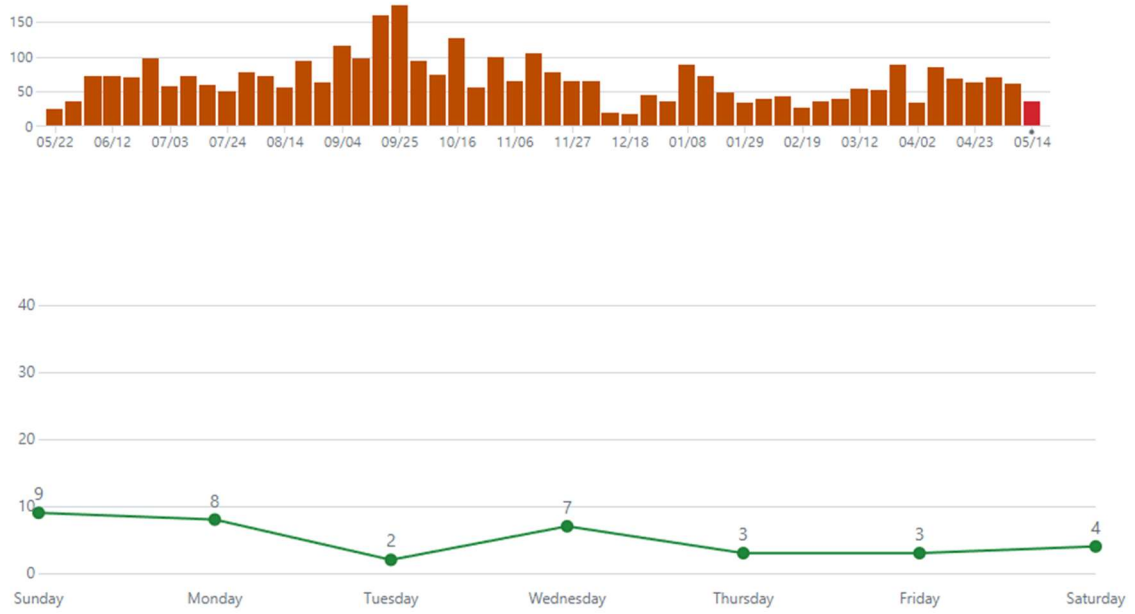

Figure 4 Github에서 제공하는 commit frequency

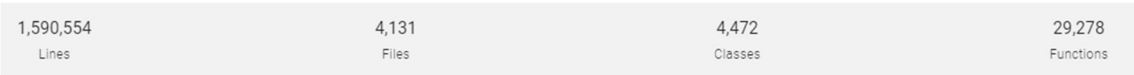

Figure 5 Understand를 통한 분석 화면의 일부(c 소스코드만 대상으로 함)

또한 150만 라인이 넘는 코드와 4000개를 넘는 파일, 클래스를 가졌으며 3만개에 가까운 함수를 가진 충분히 큰 규모의 프로젝트로서 우리가 세운 조건을 만족한다고 판단하였다.

2. OpenSSL

https://github.com/openssl/openssl

OpenSSL은 SSL 프로토콜로 알려진 TLS 프로토콜을 위한 강력하고, 상용 등급의 모든 기능을 갖춘 오픈소스 툴킷이다. 프로토콜 구현은 독립적으로 실행할 수 있는 강력한 범용 암호화 라이브러리를 기반으로 한다. 이 프로젝트는 9000회 이상 fork 되었으며 다음과 같은 꾸준한 commit을 확인할 수 있었다.

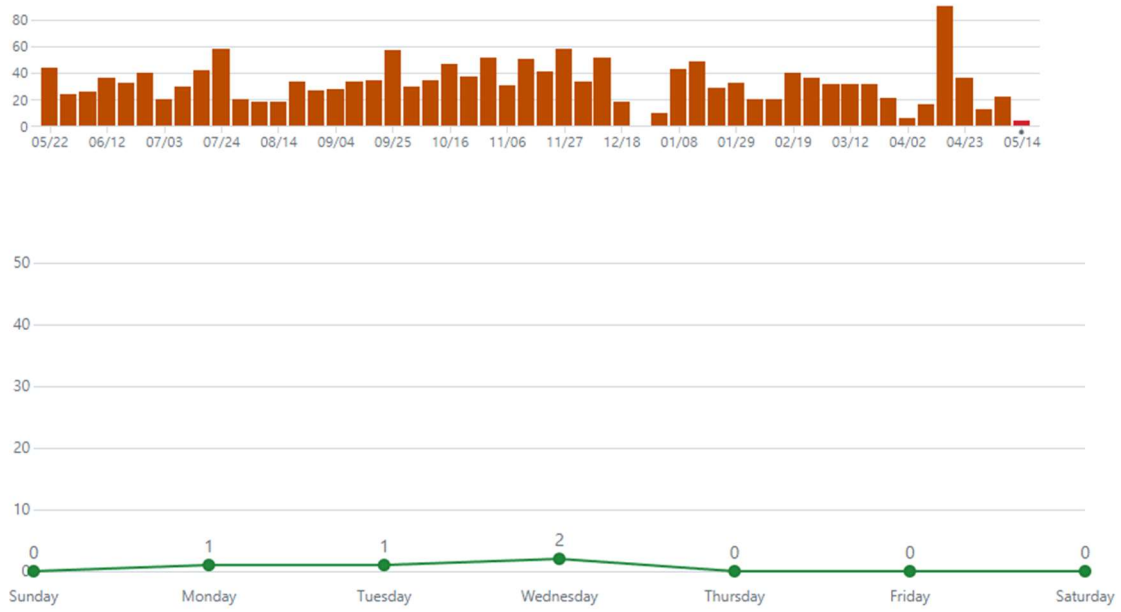

Figure 6 Github에서 제공하는 commit frequency

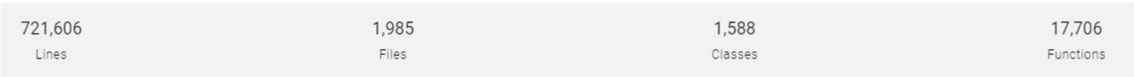

Figure 7 Understand를 통한 분석 화면의 일부(c 소스코드만 대상으로 함)

또한 70만 라인이 넘는 코드와 2000에 가까운 파일, 1500개를 넘는 클래스와 17700개가량의 함수를 가진 충분히 큰 규모의 프로젝트로서 우리가 세운 조건을 만족한다고 판단하였다.

3. SQLite

SQLite는 독립형 트랜잭션 SQL 데이터베이스 엔진을 구현하는 in-process 라이브러리로서 별도의 서버 프로세스를 요구하지 않는 임베디드 SQL 데이터베이스 엔진이다. 이 프로젝트는 600회 이상 fork 되었으며 다음과 같이 꾸준한 commit을 확인할 수 있었다.

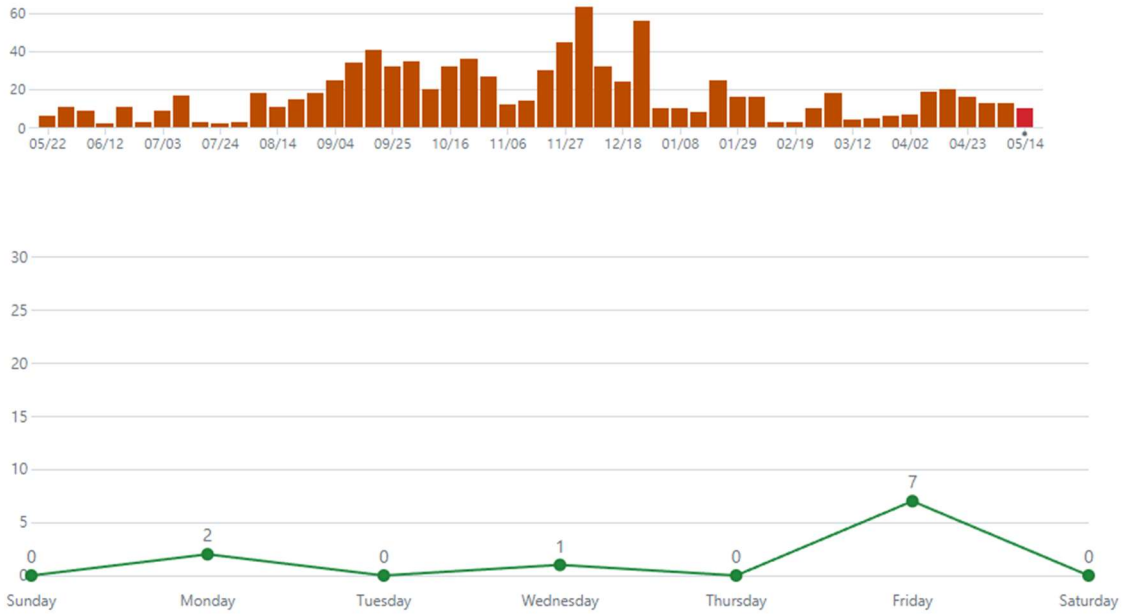

Figure 8 Github에서 제공하는 commit frequency

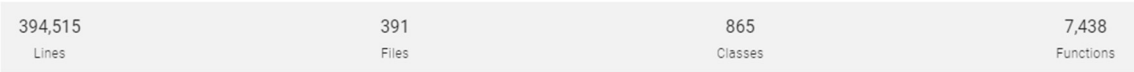

Figure 9 Understand를 통한 분석 화면의 일부(c 소스코드만 대상으로 함)

또한 39만 라인이 넘는 코드와 391개의 파일, 865개의 클래스와 7400개가량의 함수를 가진 충분히 큰 규모의 프로젝트로서 우리가 세운 조건을 만족한다고 판단하였다.

따라서 본 과제에서는 이상의 세 프로젝트를 분석에 이용하고자 한다.

3.2.2 Coding Rule 분석 Quality Model 품질 요구사항 매핑

3.1.2와 3.1.3의 Coding Rule은 각각 어떤 문제에 대한 규칙인지는 분류되어 있지만 직접적으로 소프트웨어의 품질 요구 사항과 연결되어있지는 않다. 각각의 Rule 들이 어떤 결함을 줄이는 데에 도움이 되는지 파악하고 그 결함의 감소가 어떤 품질 요구 사항을 개선시키는지 연결시켜 매핑을 진행한다. 가령 Runtime Error에 대한 Coding Rule은 Reliability에 영향을 줄 것이며 언어 확장에 대한 Coding Rule은 Portability에 영향을 줄 것이다. 물론 각각이

단 하나의 품질 요구사항에만 영향을 주는 것이 아니기 때문에 복수의 품질 요구사항에 매핑 될 수 있을 것이다.

3.2.3 품질 지수 측정

3.1.1의 Polyspace를 이용하여 수집한 c 기반 오픈소스 프로젝트에 대한 정적분석을 행한다. 이후 정적분석의 결과를 이용하여 3.2.2에서 매핑한 품질 요구사항과의 관계를 이용하여 해당 프로젝트에 대한 품질 지수를 측정한다.

먼저 Polyspace를 이용하여 정적분석을 행하면 다음과 같은 결과를 얻을 수 있다.

Table 6.1.
/mathworks/devel/bat/filer/batfs2510-0/BR2023ad.2254940.BR2023ad.2249993.pass/build/matlab/polyspace/examples/cxx/Bug_Finder_Example/sources/concurrency.c

ID	Defect	Impact	Function	Detail	Severity	Status	Comment
8415	Data race	High	File Scope	Certain operations on variable 'bad_glob1' can interfere with each other and cause unpredictable values.	Unset	Unreviewed	
8414	Data race	High	File Scope	Certain operations on variable 'bad_glob2' can interfere with each other and cause unpredictable values.	Unset	Unreviewed	
8604	Deadlock	High	File Scope	Multiple tasks are waiting for resources held by each other.	Unset	Unreviewed	
8600	Double lock	High	File Scope	Task is waiting for already acquired resource.	Unset	Unreviewed	
8603	Double unlock	High	File Scope	Task is trying to release already released resource.	Unset	Unreviewed	
8601	Missing unlock	High	File Scope	The locked resource is not released within the task.	Unset	Unreviewed	
8602	Missing lock	Medium	File Scope	Task is trying to unlock a	Unset	Unreviewed	

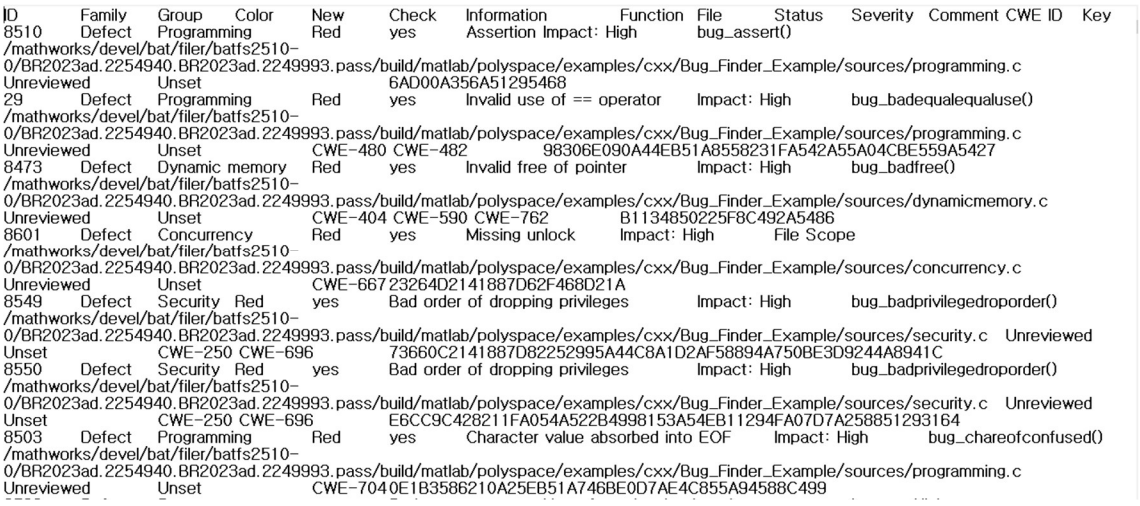

Figure 10 Polyspace를 통한 분석 결과의 일부

여러 가지 항목이 있지만 ID에서는 Polyspace 내부의 Coding Rule 번호, File에서는 해당 source 파일, CWE ID에서는 위반한 CWE라는 다른 Coding Rule의 ID를 표시하고 있다. 또한

두 번째 그림에서는 어떤 파일의 어떤 함수에서 규칙을 위반하였는지 보여준다. 이를 통해 우리는 분석한 프로젝트의 특정 파일에서 특정 Coding Rule을 위반하고 있다는 사실을 파악할 수 있는 것이다. 만약 직접적으로 원하는 Coding Rule의 위반사항을 확인하지 못한다면 Polyspace 내부 Rule ID와 우리가 찾고자 하는 Coding Rule ID를 매핑하는 과정을 거친다.

그러면 우리는 각각의 함수가 전체 Coding Rule에서 몇 개의 Coding Rule을 위반하였는지 알 수 있을 것이고 이를 통하여 함수의 품질 지수를 측정할 수 있을 것이다. 그리고 이러한 함수가 모인 파일에 대한 점수 역시 파일에 대한 Coding Rule에 의한 점수와 File을 이루는 함수들의 점수를 합쳐서 계산할 수 있을 것이며 전체 시스템 역시 System 범위에 대한 Coding Rule에 의한 점수와 프로젝트를 구성하는 File들의 점수를 합침으로서 계산할 수 있을 것이다.

한 Coding Rule을 같은 범위에서 여러 번 위반하는 것에 대한 가중치나 Rule 각각의 가중치, 혹은 하위 범주의 점수를 상위 범주의 점수를 계산할 때 얼마나 반영할 지와 같은 수치적인 결정 역시 필요한 단계이다.

4. 연구 일정 및 역할 분담

4.1 연구 일정

5월		6월				7월				8월				9월			
3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주
소스코드 확보																	
		MISRA, CERT 조사															
		Coding Rule 분석															
						Quality Model 품질 요구사항 매핑											
								중간 보고서 작성									
										정적 분석 결과를 통한 품질지수 측정법 개발							
														품질지수 측정법의 세부 수치 조정			
																최종 보고서 작성	

4.2 역할 분담

이름	역할
전민기	Polyspace 라이센스 확보 Coding Rule과 Quality Model 품질 요구사항 매핑 정적 분석 결과를 통한 품질지수 측정법 개발
천동혁	C 기반 소스코드 확보 Coding Rule과 Quality Model 품질 요구사항 매핑 정적 분석 결과를 통한 품질지수 측정법 개발
성민우	MISRA, CERT에 대한 기본 내용 파악 Coding Rule과 Quality Model 품질 요구사항 매핑 정적 분석 결과를 통한 품질지수 측정법 개발