요구사항명세서

제 10회 공개 SW 개발자 대회

스펙트럼 기반 통합 테스트 플랫폼

SWQ&R

성균관대학교
소프트웨어공학연구실
김정호
jeonghodot@skku.edu
010-3111-3555

1. Preface

1.1. Objective

Preface에서는 본 문서의 독자가 누구인지를 밝히고 문서의 구조와 각 챕터의 역할을 소개한다. 또, Version History를 제시하여 문서의 새로운 버전이 만들어질 때마다 변경사항과 변경사유를 설명한다.

1.2. Readership

본 문서는 크게 User requirements와 System requirements 두 부분으로 구성되어 있다. User requirements는 사용자의 관점에서 요구사항을 간략히 명세한 것이고, System requirements는 프로그램이 제공하는 기능을 더욱 자세하게 설명한 것이다.

1.2.1. User requirements readership

User requirements의 주요 독자는 해당 프로그램의 고객들이기 때문에 프로그램이 어떻게 구현되는지 등에 관한 사안에 관심이 없다. 이를 고려하여 전문지식이 없는 독자들을 위해 자연어, 다이어그램 등과 같은 방법으로 프로그램이 어떤 기능을 제공하는지를 설명한다. 고객들 외에도 Client managers, system end-users, client engineers, contractor managers, system architects가 user requirements의 독자가 될 수 있다.

1.2.2. System requirements readership

System requirements의 주요 독자는 해당 소프트웨어 개발자들이다. 그렇기 때문에 System requirements에서는 해당 프로그램의 특정 기능이 어떻게 구현되어야 하는지에 대해 자세하게 설명되어 있다. Software Developer외에도 client engineers, system architects, 경우에 따라서는 System end-users까지 주요 독자가 될 수 있다.

1.3. Document Structure

이 문서는 총 10개의 부분으로 구성되어 있다. Preface, Introduction, Glossary, User Requirements Definition, System Architecture, System Requirements Specification, System Models, System Evolution, Appendices, Index 로 구성된다. 각 부분의 역할은 다음과 같다.

1.3.1 Preface

Preface에서는 본 문서의 예상 독자가 누구인지 밝히고, 문서의 구조, 각 챕터의 역할 등에 대해 소개한다. 또, Version History와 그 변경사유, 변경사항에 대해 설명하였다.

1.3.2. Introduction

Introduction에서는 해당 프로그램이 어떤 니즈(Needs)를 반영하였는지를 설명한다. 간략한 프로그램의 기능을 설명하고 프로그램이 다른 프로그램들과 어떻게 동작하는지를 설명한다.

1.3.3. Glossary

Glossary 에서는 본 문서에서 사용되는 기술 용어가 무엇인지 설명한다. 이 때, 독자의 경험이나 전문지식 등이 어떤지를 추정해서는 안되고, 가능한 한 모든 용어에 대해 서술해야 한다.

1.3.4. User requirements definition

User Requirements Definition 에서는 사용자에게 제공되는 서비스에 대해서 설명한다. 비기능적인 프로그램 요구사항도 서술한다. 자연어와 다이어그램과 같은 방법들로 설명되어 있다. 프로그램은 여기에 서술된 요구사항들을 반드시 충족할 것이다.

User requirements definition에서는 사용자에게 제공되는 서비스에 대해 설명한다. 이 챕터에서는 또한 Non-functional system requirements 역시 서술된다. 이 때, 설명은 자연어, 다이어그램 또는 독자들이 이해할 수 있는 다른 개념들이 사용될 수 있다. 제품이나 프로세스가 반드시 따라야 하는 표준이 있다면 반드시 서술되어야 한다.

1.3.5. System architecture

System Architecture에서는 목표 프로그램 architecture에 대한 high-level에서의 개요를 제시한다. 또, 프로그램 모듈 사이에서 기능들의 분포가 어떤지를 보여준다.

1.3.6. System requirements specification

System Requirements Specification에서는 functional requirements 또는 non-functional requirements에 대해 더욱 자세히 설명한다. 필요하다면, non-functional requirements에 더욱 자세히 설명한다. 다른 프로그램에 대한 인터페이스 역시 이 챕터에서 정의된다.

1.3.7. System models

System Models 에서는 프로그램 컴포넌트, 프로그램 그리고 프로그램 환경 사이의 관계를 보여준다. 가능한 그래픽 모델로는 object models, data-flow models, semantic data models 등이 있다.

1.3.8. System evolution

System Evolution에서는 프로그램이 세우고 있는 주요한 가정들에 대해 설명한다. 또, 프로그램에

일어날 수 있는 예상되는 변화들, 가령, 하드웨어의 발전, 사용자의 니즈 변화 등에 대해 설명한다. 이 챕터는 차후 발생할 프로그램의 설계 변경을 피할 수 있도록 도와주기 때문에 프로그램 설계자들에게 도움이 될 것이다.

1.3.9. Appendices

Appendices에서는 개발되는 프로그램과 관련된 대한 자세하고 구체적인 정보를 제공한다.

1.3.10. Index

Index 에서는 주요 용어, 다이어그램, 기능에 대한 인덱스 등이 포함된다.

1.4. Version of the Document

1.4.1. Version format

버전 번호는 major.minor[.maintenance]로 구성되며. 문서의 버전은 0.1부터 시작한다.

1.4.2. Version management policy

요구사항 명세서를 수정할 때 마다 버전을 업데이트한다. 다만 변경간의 간격이 1 시간 이내 일 때에는 버전 번호를 업데이트 하지 않고 하나의 업데이트로 간주한다. 이미 완성된 파트를 변경할 때에는 minor number를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 major number를 변경한다. 이미 작성한 부분에 대해서 오타 수정하거나, 문서의 구조를 변경할 경우 maintenance number를 추가하거나 변경한다.

1.4.3. Version update history

Version	Modified date	Explanation
0.1	2016-08-11	문서의 초안과 표지 작성
0.2	2016-08-12	Introduction 작성
0.3	2016-08-13	Introduction 수정
0.4	2016-08-14	Preface, Glossary, User Requirement Definition 작성
0.5	2016-08-15	System Architecture 작성
0.6	2016-08-16	User Requirement Definition, System Architecture 수정
0.7	2016-08-16	System Requirement Specification 작성
0.8	2016-08-17	System Model, System Evolution 작성
0.9	2016-08-17	Appendices
1.0	2016-08-18	전체 작성 완료
1.1	2016-08-18	System Model의 Sequence Model, Structural Model 수정

2. Introduction

2.1. Objectives

프로그램에 대한 개략적인 소개와 왜 프로그램이 필요한지, 다른 프로그램과 어떠한 관련성을 가지는지, 프로그램이 목적을 어떻게 반영하고 있는지에 대하여 기술한다.

2.2. Needs

2.2.1. 소프트웨어 개발 및 유지보수 비용의 증가

현대의 소프트웨어가 가지는 규모와 복잡성은 날로 심화되고 있으며, 소프트웨어를 개발하기 위한 소스 코드를 작성하고 이를 빌드하여 테스트하기 위한 과정에 많은 시간이 할애되고 있는 형편이다. 특히 구조가 복잡하거나 규모가 큰 소프트웨어의 경우 소스코드의 수와 이를 테스트하기 위한 테스트 케이스가 지속적으로 개발과 유지보수 됨에 따라 증가하기 때문에, 관련되는 모든 **테스트 케이스를 일일이 수행**하여 결함 여부를 판단하고, 발생한 **결함의 위치 및 상세한 원인을 파악하기 어렵다**. 즉 테스트에 많은 시간을 소비하게 하여 개발 비용 및 기간 증가의 원인이 되어 전체 소프트웨어 개발 비용을 증가시키는 원인이 된다. 그러므로 위 비용을 줄여주기 위해 **테스트 케이스를 자동으로 수행하고, 발견한 결함에 대해 의심스러운 위치를 추적하기 위한 도구가 필요**하다.

2.2.2. 저비용 결함위치 추적 기술의 활용

결함의 위치를 추적하기 위해 제안된 많은 기술들 중 SFL(Spectrum-based Fault Localization) 기술은 소프트웨어 실행정보와 테스트 결과만을 통해 결함으로 의심되는 위치를 통계적으로 계산할 수 있는 저비용의 결함위치 추적 기술로, 소프트웨어의 요소 중 하나인 소스코드의 라인 단위로 결함 의심도를 부여하는 기술이다. 해당 기술은 타 기술들에 비해 정확도가 높고 계산량이 작아 소프트웨어의 빠른 시장 진입을 도울 수 있다. 현재 SFL을 위해 상당히 많은 알고리즘들이 제안되어 있고, 각 알고리즘이 보이는 성능 또한 다르기 때문에 개발자가 이를 모두 구현하여 전체적인 알고리즘들이 공통적으로 추적한 결함 위치를 확인하여야 한다. 그러나 **이를 모두 구현하는 것은 관련 지식이 없는 개발자로써는 어렵고 시간이 많이 소모되는 일**일 수 있다. 또한 계산되는 결함 의심도는 수치적인 정보이기 때문에 개발자가 **직관적으로 알아보기 힘들 수 있다**. 그러므로 **관련된 알고리즘들이 미리 구현되어 개발자가 각 알고리즘들에 따르는 결함의 위치를 시각적으로 편하게 알아볼 수 있도록 하기 위한 도구**가 필요하다.

2.2.3. 빠르고 정확한 회귀테스트의 요구

소프트웨어 테스트 방법 중 하나인 회귀테스트는 결함을 수정한 이후에 기존에 작성되어있는 테스트 케이스들을 다시 수행시켜, 기존에 존재했던 결함이 다시 발생하지 않음을 보장하기 위한

방법이다. 2.2.1에서도 언급하였듯이 테스트 케이스는 지속적인 개발과 유지보수에 따라 수가 증가하기 때문에 회귀테스트에 사용되는 테스트 케이스의 수가 너무 많아 **테스트 시간이 오래 걸릴 수 있다**. 이를 위해 최소한의 시간으로 최대한 많은 결함을 찾기 위한 테스트 케이스 우선순위화 기술들이 제안되었으나 관련 도구들의 제공이 미흡하다. 그러므로 실제 개발자가 **회귀 테스트를 수행할 때 테스트 케이스를 관리해줄 수 있는 관련 도구**가 필요하다.

2.3. ISES

ISES이란 Integrated Software Environments for Spectrum-based fault localization의 약어로써, 성균관대학교 소프트웨어 공학 연구실에서 개발할 결함위치 추적 프로그램이다. 해당 프로그램은 사용자가 테스트하고자 하는 소프트웨어의 소스코드들과 관련된 테스트 스윗을 입력하면, 자동으로 테스트 케이스에 따라 소스코드를 수행시키고, 수행 결과를 통해 소스코드에 라인별 결함 의심도를 계산하는 프로그램이다. 해당 프로그램은 크게 5가지 기능을 제공한다.

1) 소스코드 및 테스트 케이스 자동 수행 기능
2) 76여가지의 SFL 알고리즘 적용 기능
3) 테스트 케이스 우선순위화 알고리즘 적용 기능
4) 의심스러운 결함 위치의 시각화 지원
5) 수행 정보, 결과 및 결함 추적 결과 저장 기능

소스코드 및 테스트 케이스 자동 수행 기능은 입력 받은 소스코드들과 테스트 스윗을 분석하여 수행 가능 여부를 분석하고, 테스트 스윗 전체 또는 부분적으로 자동으로 수행하고, 수행 정보인 개별 테스트 케이스를 통해 트레이스 정보 및 해당 테스트 케이스의 결과를 수집한다.

76여가지의 SFL 알고리즘은 관련 SFL 연구들을 통해 제안되었던 수식들을 구현한 것으로, 유저가 한 개 이상의 알고리즘을 선택할 수 있도록 한다. 수집된 수행 정보와 테스트 케이스 결과의 집합과 유저가 선택한 알고리즘을 토대로 통계적으로 소스코드의 각 라인 별 결함 의심도를 계산하는 기능이다.

테스트 케이스 우선순위화 알고리즘은 관련 테스트 케이스 우선순위화 연구들을 통해 제안되었던 방법들을 구현한 것으로, 유저가 원할 시 가지고 있는 모든 테스트 케이스가 아닌 중요한 테스트 케이스부터 수행하는 기능이다.

의심스러운 결함 위치의 시각화 지원은 선택된 알고리즘들을 통해 계산되는 결함 의심도에 따라 순위를 매기고, 각 순위에 따라 가장 의심스러운 라인 순서대로 색상을 부여한다.

수행 정보, 결과 및 결함 추적 결과 저장 기능은 결함 추적 알고리즘을 통해 유저에게 제공된 정보들을 데이터 베이스에 저장하거나 파일로 출력하는 기능이다.

2.4. Expected Effect of the ISES

ISES은 소스코드들의 라인 별 결함 위치를 자동으로 추적해주는 프로그램으로, 소스코드를 작성하는 개발자와 소프트웨어를 시장에 배포하기 위한 기업의 입장에서의 긍정적인 효과를 기대한다.

2.4.1. Developers

ISES의 소스코드 및 테스트 케이스 자동 수행 기능은 개발자가 **테스트 케이스를 개별적으로 검색하고 수행시키는 시간을 감소**시킬 수 있으며, 76여가지의 SFL 알고리즘의 제공을 통한 결함 추적과 시각화 지원 기능을 통해 **결함을 추적하기 위해 소모되는 시간을 감소**시킬 수 있다. 추가적으로 이미 유효한 정확성이 확인된 다양한 알고리즘을 제공함으로써 **소프트웨어의 품질 향상**을 꾀할 수 있다.

2.4.2. Companies

소프트웨어 개발 시간 및 비용이 감축됨에 따라 **더 빠른 시장 진입이** 가능하며, 이를 통해 **시장 내 우위 선점과 높은 이익 창출**이 가능하다

3. Glossary

3.1. Objectives

3 장 Glossary 에서는 이 문서에 사용되는 기술적인 용어(technical term)에 대해 설명한다. 이 장은 배경지식이 없는 문서의 독자가 이 장을 참고하여 문서를 이해할 수 있도록 도움을 준다.

표 1 Glossary

용어	정의
ISES	본 팀이 개발하는 소스코드 내 결함 추적 소프트웨어로 Integrated Software Environments for Spectrum-based fault localization 약자
소프트웨어	컴퓨터 프로그램과 그와 관련된 문서들을 총칭하는 용어
소프트웨어 테스트	주요 이해관계자들에게 시험 대상 제품 또는 서비스의 품질에 관한 정보를 제공하는 조사 과정
회귀 테스트	오류를 제거하거나 수정한 프로그램이 오류 제거와 수정에 의해 새로이 유입되는 오류가 없는지를 확인하는 일종의 반복 시험.
소프트웨어 결함	소프트웨어 상의 오류를 일으킬 수 있는 징후를 의미
테스트 케이스 (TC)	소프트웨어 테스트를 위한 데이터로, 특정 요구사항을 준수하는지 검증하기 위해 입력값, 예상결과, 실행조건의 집합을 명세화한 문서
테스트 스윗 (Test Suite)	테스트 케이스의 집합
테스트케이스 우선순위화 (TCP)	테스트 케이스의 실행 순서를 스케쥴링하는 기술
디버깅	소프트웨어 결함을 추적하여 식별하고, 식별한 결함을 고치는 작업

소스코드 (SC)	컴퓨터 프로그램인 소프트웨어를 프로그래밍 언어로 기술한 글
스펙트럼	소스코드에 테스트 케이스를 입력한 결과로 나타나는 테스트 케이스가 지나간 소스코드 라인의 집합
트레이스 정보	소스코드에 테스트 케이스를 입력하여 확인 가능한 지나간 소스코드 라인 정보
SFL (Spectrum-based Fault Localization)	소프트웨어 실행정보와 테스트 결과를 통해 결함으로 의심되는 위치를 통계적으로 계산하는 결함위치 추적 기술
데이터베이스	규칙적으로 저장된 다수 데이터의 집합
유저	소프트웨어를 직접적으로 사용하는 사용자
운영체제	모든 하드웨어와 소프트웨어를 관리하는 컴퓨터 프로그램의 한 부분

4. User Requirement Definition

4.1. Objectives

User Requirements Definition 에서는 사용자에게 제공되는 서비스에 대해서 설명한다. 비기능적인 프로그램 요구사항도 서술한다. 자연어와 다이어그램과 같은 방법들로 설명되어 있다. 프로그램은 여기에 서술된 요구사항들을 반드시 충족할 것이다.

4.2. Functional Requirements

ISES에서는 다음과 같은 기능을 제공하여야 한다.

4.2.1. 소스코드 및 테스트 케이스 자동 수행 기능

ISES에서 가장 처음에 수행되는 기능으로 유저가 선택한 소스코드들과 테스트 스윗을 자동으로 수행시키는 기능이다. 사용자가 테스트하고자 하는 소프트웨어의 소스코드들과 테스트 스윗을 설정하면, 해당 소스코드들과 분석하여 설정된 테스트 스윗의 수행 가능 여부를 분석하고, 분석 결과에 따라 테스트 스윗을 소스코드에 수행시킨다. 각 테스트 케이스 별 수행 결과들은 결함 추적을 위해 메모리에 수집한다.

4.2.2. 테스트 케이스 우선순위화 알고리즘 제공

테스트 케이스 우선순위화 기술을 제공하는 기능이다. 유저가 테스트 케이스 우선순위화 수행을 선택하면 따라 모든 테스트 케이스가 아닌 중요한 테스트 케이스부터 테스트를 수행한다.

4.2.3. 76여가지의 SFL 알고리즘에 기반한 결함 추적 기능

기존에 유효한 효과를 보였던 76가지의 결함 추적 알고리즘들을 제공하는 기능이다. 유저가 선택한 알고리즘들은 수행 결과를 이용하여 소스코드의 모든 라인 별 의심도를 계산한다.

4.2.4. 의심스러운 결함 위치의 시각화 지원

SFL 알고리즘을 통해 계산되는 소스코드 라인 별 의심도를 유저가 결함을 추적하는데 쉽게 활용할 수 있도록 시각화를 지원하는 기능이다. 소스코드 라인 별 의심도에 따라 순위화를 수행하고 각 라인의 순위에 따라 라인을 강조하기 위한 색상을 부여한다.

4.2.5. 수행 정보, 결과 및 결함 추적 결과 저장 기능

소스코드와 테스트 케이스 수행정보, 결과 및 SFL 알고리즘을 통해 계산된 결함 추적 결과를 파일로 저장하는 기능이다.

4.3. Non-functional Requirements

Non-Functional Requirement로는 Product Requirement, Organization Requirement, External Requirement로 나눌 수 있다. 각 요소마다 세부적으로 더 분류될 수 있지만 여기서는 ISES에 필요로 되는 것만을 서술한다.

4.3.1. Product Requirements

A. Usability Requirements

해당 프로그램은 해당 프로그램을 잘 모르는 유저라도 쉽게 프로그램 이용에 도움을 줄 수 있도록 직관적인 단어를 사용한다. 또한 추적한 의심스러운 라인들을 개발자가 쉽게 발견할 수 있도록 의심도가 높은 라인일수록 눈에 띄는 색상을 부여한다.

B. Performance Requirements

해당 프로그램은 디버깅에 소요되는 시간을 줄이기 위한 프로그램이기 때문에 결함을 추적하기 위한 계산에 소모되는 시간을 최소화 해야 한다. 또한 제공하는 76 개의 알고리즘들에 대한 정확도를 보장하여야 한다.

4.3.2. Organizational Requirements

A. Environmental Requirements

해당 프로그램은 유저가 모든 운영체제에서도 사용 가능한 프로그램을 구현하여 사용자의 편리성을 높인다.

B. Operational Requirements

해당 프로그램은 유저가 결함을 추적하고자 하는 소스코드의 개발 언어와 관계없이 모든 기능을 수행할 수 있어야 한다.

5. System Architecture

5.1. Objectives

System Architecture 에서는 프로그램 아키텍처의 높은 수준(high-level)의 개요(overview)를 기술한다. 이 장에서는 프로그램 모듈간에 있어서의 기능의 배분을 명확히 하는 것을 목적으로 하며, 각 서브 프로그램의 기능을 기술한다. 개발하는 프로그램은 여러 세부 프로그램으로 구성되며 서로 세부 프로그램 간 상호 연관이 있다. 한 프로그램이 다른 프로그램에 영향을 주고 의존을 하면서 상호 작용을 한다.

5.2. Overall System Architecture

ISES은 유저가 소스코드 및 테스트 케이스를 선택하고 결함 추적 결과를 확인 및 저장하기 위한 1) 인터페이스와 2) 파일 입출력을 위한 세부프로그램, 선택된 여러 개의 3) 테스트 케이스 우선순위화를 위한 세부 프로그램, 외부 컴파일러 API를 이용하는 4) 소스코드와 테스트 스윗을 자동 수행하기 세부 프로그램, 테스트 케이스 수행 정보와 테스트 케이스 관리 기술에 따른 5) 결함 위치를 추적하기 위한 세부프로그램으로 구성된다

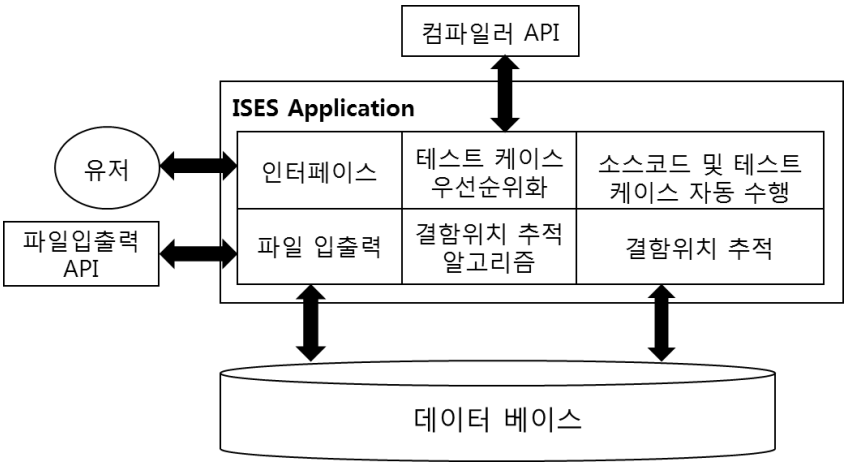

그림 1 ISES 전체 아키텍처

5.3. 파일 입출력을 위한 세부 프로그램

파일 입출력을 위한 세부 프로그램은 파일 입출력 모듈, 소스코드 분석 모듈, 테스트 케이스 분석 모듈, 소스코드 및 테스트 케이스 저장 모듈로 구성된다. **파일 입력 모듈**은 유저가 선택한 소스코드와 테스트 케이스를 분석하기 위해 입력 받는 기능을 수행하고, **파일 출력 모듈**은 소스코드와 테스트 케이스와 그들의 분석 결과를 파일로 출력하고 데이터 베이스에 저장하는 기능을 수행한다. **소스코드 분석 모듈**은 등록한 소스코드를 분석하여 실제 프로그램이 가동되기

위해 최초로 수행되는 메인 함수를 찾고 필요한 입력 값들의 형식과 개수를 분석한다. **테스트 케이스 분석 모듈**은 선택된 테스트 케이스들이 일관적인 형식과 수를 가지고 있는지 분석하고 각 테스트 케이스들이 문제 없이 프로그램을 수행시킬 수 있는지 분석한다. **소스코드 및 테스트 케이스 저장 모듈**은 분석 결과 테스트 케이스가 소스코드를 명확히 수행될 수 있다고 한다면 향후 재테스트를 수행할 때 해당 결함 추적 정보를 이용하기 위해 해당 소스코드 및 테스트 케이스를 데이터 베이스에 저장하고, 관련 정보를 파일로 출력하는 기능을 수행한다.

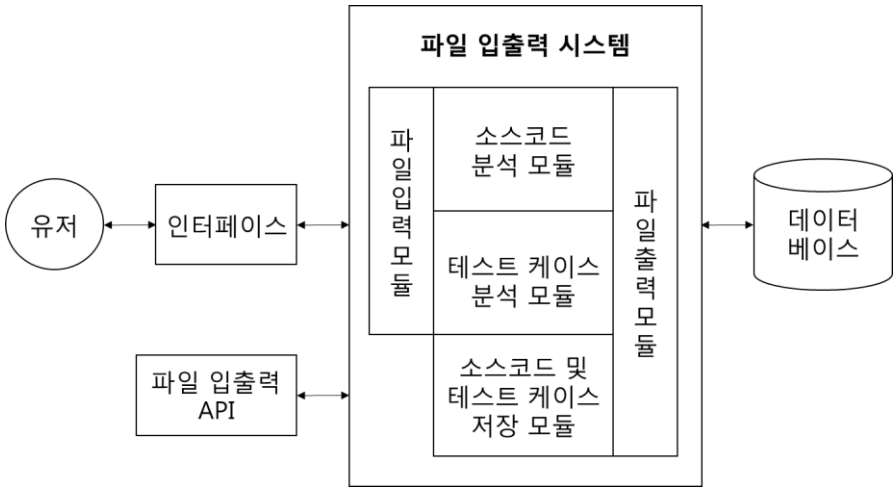

그림 2 파일 입출력을 위한 세부 프로그램 아키텍처

5.4. 테스트 케이스 우선순위화 프로그램

테스트 케이스 우선순위화를 위한 세부 프로그램은 과거 결함 추적 정보 분석 모듈, 테스트 케이스 우선순위화 수행 모듈로 구성된다. 유저 인터페이스를 통해 유저가 우선순위화 기능을 사용하기를 원한다면, **과거 결함 추적 정보 분석 모듈**은 이전에 미리 테스트 되었던 테스트 케이스들의 수행 정보와 결함 추적 결과를 분석하여 우선순위화 알고리즘에 필요한 값들을 산출하기 위한 모듈이다. **테스트 케이스 우선순위화 수행 모듈**은 과거 정보를 통해 산출된 값들에 기반하여 현재 선택된 테스트 케이스들의 수행 우선 순위를 결정하는 기능을 수행한다.

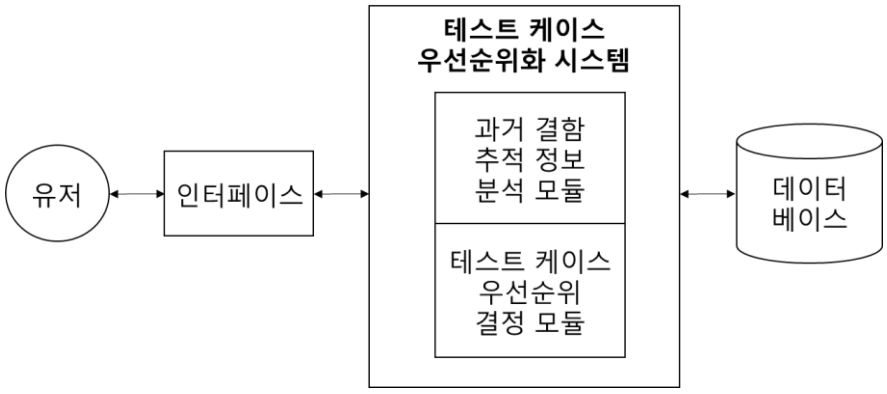

그림 3 테스트 케이스 우선순위화 세부 프로그램 아키텍처

5.5. 소스코드 및 테스트 케이스 자동 수행 세부 프로그램

해당 세부 프로그램은 소스코드 자동 빌드 모듈, 테스트 케이스 자동 수행 모듈, 수행 결과 저장 모듈고 구성된다. **소스코드 자동 빌드 모듈**은 외부 컴파일러 API를 통해 소스코드를 빌드하고 수행 가능한 프로그램을 만드는 기능을 수행한다. **테스트 케이스 자동 수행 모듈**은 테스트 케이스의 입력 값을 해당 프로그램에 입력하여 해당 결과를 얻기 위해 프로그램을 수행 시키는 모듈이다. **수행 결과 저장 모듈**은 테스트 케이스들이 프로그램에서 수행된 후 테스트 케이스에 작성된 기대 값과 실제 값을 비교하여 나온 수행 결과들을 데이터 베이스에 저장하는 모듈이다.

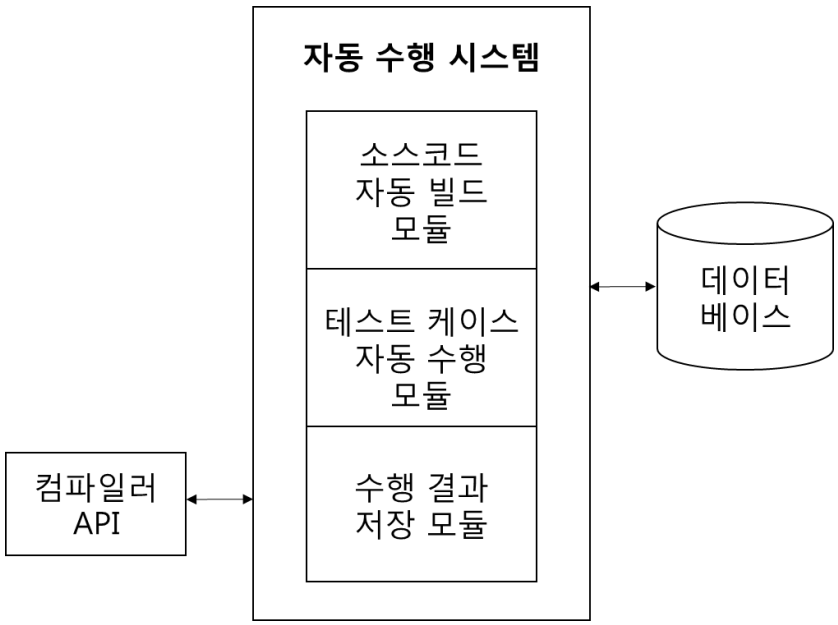

그림 4 소스코드 및 테스트 케이스 자동 수행 세부 프로그램 아키텍처

5.6. 결함 추적 프로그램

해당 세부프로그램은 수행 정보 분석 모듈, 추적 알고리즘 수행 모듈, 추적 결과 분석 모듈, 파일 출력 모듈로 구성된다. **수행 정보 분석 모듈**은 테스트 수행 정보와 결과가 저장된 데이터 베이스에서 정보를 가져와 해당 정보들이 올바르게 저장이 되었는지 분석하고 스펙트럼화 하는 작업을 수행한다. **추적 알고리즘 수행 모듈**은 유저가 선택한 추적 알고리즘들을 사용하여 수행 정보 및 결과를 통해 소스코드의 라인 별 의심도를 계산하는 모듈이다. **추적 결과 분석 모듈**은 계산된 의심도에 따라 순위화를 수행하고 순위에 따라 시각화를 제공하기 위한 라인 별 색상을 결정하는 기능을 수행한다. **파일 출력 모듈**은 유저의 선택에 의해 수행 정보, 결과 및 추적 결과에 대한 정보를 파일로 출력하기 위한 모듈이다.

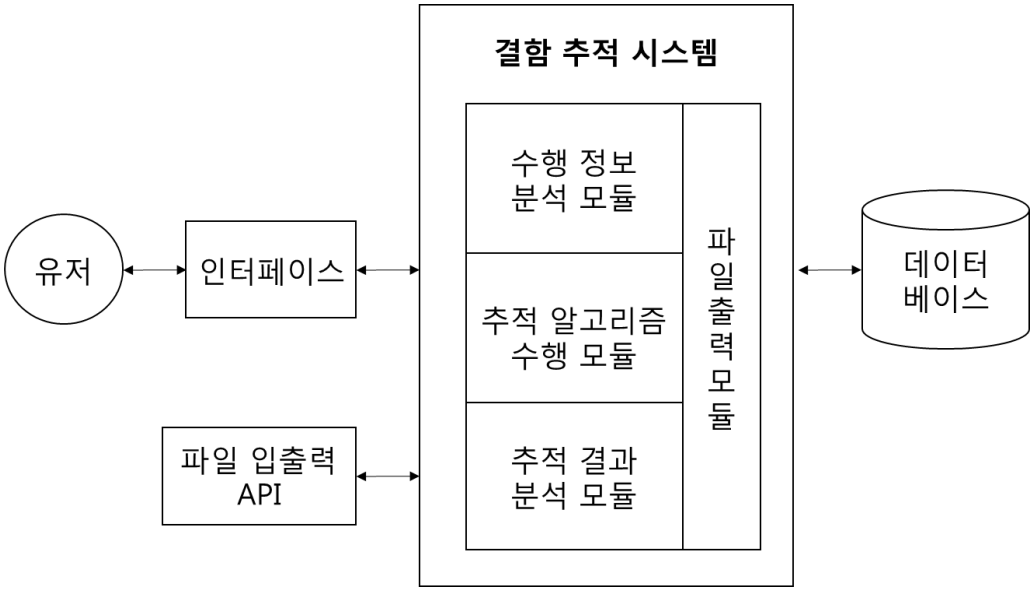

그림 5 결함 추적 세부 프로그램 아키텍처

6 System Requirement Specification

6.1. Objectives

System Requirements specification에서는 ISES의 기능적이거나 비기능적인 요구사항에 대해 더 자세히 설명한다.

6.2. Functional Requirements

6.2.1. 소스코드 및 테스트 케이스 자동 수행 기능

A. 소스코드 입력 기능

기능	소스코드 입력 기능
설명	유저가 소스코드 파일의 위치를 설정하면, 해당 소스코드들을 화면에 출력한다.
입력	소스코드 파일의 위치
출력	소스코드 내용
처리	소스코드 파일 위치로부터 파일을 입력 받고, 해당 파일에 작성된 내용들을 라인 별로 출력한다.
조건	파일 위치가 정확하여야 한다.

B. 테스트 케이스 목록 제공 기능

기능	테스트케이스 목록 제공 기능
설명	유저가 테스트 스윗 파일의 위치를 설정하면, 유저가 수행시키고자 하는 테스트 케이스들을 선택할 수 있도록 해당 파일에 등록된 테스트 케이스들을 출력한다.
입력	테스트 스윗 파일 위치
출력	테스트 케이스들 목록
처리	테스트 스윗 파일 위치로부터 파일을 입력 받고, 해당 파일에 작성된 내용들을 출력한다.
조건	테스트 스윗 파일의 위치 및 형식이 정확하여야 한다.

C. 테스트 케이스 선택 기능

기능	필요한 테스트케이스 선택 기능
설명	테스트 케이스들의 목록으로부터 유저가 수행시키고자 하는 테스트 케이스 목록에 체크박스를 추가하여 선택할 수 있도록 한다.
입력	테스트 케이스들 목록
출력	테스트 케이스 선택 목록
처리	선택된 테스트 케이스들을 수행하기 위한 집합으로 결정한다.
조건	테스트 케이스 목록이 명확히 불러와져야 한다.

D. 소스코드 및 선택된 테스트 케이스 수행 기능

기능	소스코드 및 테스트케이스들의 수행 기능
설명	유저가 선택한 소스코드와 테스트 케이스들 수행시킨다.
입력	소스코드, 테스트 케이스 선택 목록
출력	소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과
처리	소스코드를 선택된 테스트 케이스 각각에 대해 수행시키고 이에 대한 수행 결과를 수집한다.
조건	유저가 실행을 클릭하였을 때 수행된다.

E. 소스코드 및 우선순위화 된 테스트 케이스 수행 기능

기능	소스코드 및 우선순위화된 테스트케이스들의 수행 기능
설명	유저가 선택한 소스코드와 우선순위화 된 테스트 케이스들을 수행시킨다.
입력	소스코드, 우선순위화 된 테스트 케이스
출력	소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과
처리	소스코드를 우선순위화된 테스트 케이스 각각에 대해 수행시키고 이에 대한 수행 결과를 수집한다.
조건	유저가 우선순위화 알고리즘 사용에 체크하고, 실행을 클릭하였을 때 수행된다.

F. 소스코드 및 우선순위화 된 테스트 케이스 조건부 수행 기능

기능	소스코드 및 우선순위화된 테스트케이스들의 조건부 수행 기능
설명	유저가 선택한 소스코드와 우선순위화 된 테스트 케이스들을 최대 테스트 시간 내에서만 수행시킨다.
입력	소스코드, 우선순위화 된 테스트 케이스, 최대 테스트 시간
출력	소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과
처리	소스코드를 우선순위화된 테스트 케이스 각각에 대해 최대 테스트 시간까지만 수행시키고 이에 대한 수행 결과를 수집한다.
조건	유저가 우선순위화 알고리즘 사용에 체크하고, 최대 테스트 시간을 입력한 후 실행을 클릭하였을 때 수행된다.

6.2.2. 테스트 케이스 우선순위화 알고리즘 제공

기능	테스트 케이스 우선순위화 알고리즘 제공
설명	유저가 선택한 테스트 케이스들에 대해 우선순위화 알고리즘을 적용할 지 여부를 결정하고 이에 따라 테스트 케이스를 우선순위화 한다.
입력	우선순위화 알고리즘 적용 여부
출력	우선순위화 된 테스트 케이스
처리	과거 결함 추적 정보를 분석한 정보와 구현된 우선순위화 알고리즘을 사용하여 선택된 테스트 케이스들을 우선순위에 따라 정렬한다.
조건	유저가 우선순위화 알고리즘 사용에 체크한다.

6.2.3. 76여가지의 SFL 알고리즘에 기반한 결함 추적 기능

A. 76여가지의 SFL 알고리즘 선택 기능

기능	76여가지의 SFL 알고리즘 선택 기능
설명	유저가 사용하고자 하는 SFL 알고리즘의 목록과 간략한 장단점 제공하고 결함 추적에 사용하기 위한 알고리즘을 체크 박스를 통해 선택 받는다.
입력	SFL 알고리즘 정보 목록
출력	선택된 SFL 알고리즘 목록
처리	현재 구현된 SFL 알고리즘 목록 및 정보를 불러와 목록화 하여 체크 박스와 함께 제공한다.
조건	ISES이 실행되어 있어야 한다.

B. 선택된 SFL 알고리즘기반 결함 추적 기능

기능	선택된 SFL 알고리즘기반 결함 추적 기능
설명	유저가 사용하고자 하는 SFL 알고리즘의 목록과 간략한 장단점 제공하고 결함 추적에 사용하기 위한 알고리즘을 체크 박스를 통해 선택 받는다.
입력	선택된 SFL 알고리즘 목록, 소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과
출력	소스코드 라인 별 의심도
처리	소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과를 스펙트럼화 하여 선택된 SFL 알고리즘들을 수행하여 소스코드 라인 별 의심도를 계산한다.
조건	소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과가 수집되어 있어야 한다.

6.2.4. 의심스러운 결함 위치의 시각화 지원 기능

기능	의심스러운 결함 위치의 시각화 지원
설명	계산된 소스코드 라인 별 의심도에 대해 순위화 및 색상 부여를 수행하여 시각화를 지원한다.
입력	소스코드 라인 별 의심도
출력	소스코드 라인 별 의심스러운 순위 및 색상
처리	소스코드 라인 별 의심도에 따라 순위를 부여하고 상위 10%는 빨간색, 20%는 노란색으로 표시되도록 한다.
조건	소스코드 라인 별 의심도가 계산되어 있어야 한다.

6.2.5. 수행 정보, 결과 및 결함 추적 결과 저장 기능

기능	수행 정보, 결과 및 결함 추적 결과 저장 기능
설명	유저의 선택에 따라 결함 추적 결과를 파일로 저장한다.
입력	결함 추적 결과 저장 여부, 소스코드 라인 별 테스트 케이스들의 지나간 라인 및 테스트 결과, 소스코드 라인 별 의심도/순위/색상
출력	결함 추적 결과 파일
처리	시각화된 모든 정보들을 파일로 출력한다.
조건	결함 추적이 완료 되어야 한다.

6.3. Non-functional Requirements

6.3.1. Produce Requirements

A. Usability Requirements

해당 소프트웨어를 잘 모르는 사람이라도 쉽게 의심스러운 라인을 식별할 수 있도록 표시한다. 이를 위해 직관적으로 의심스러운 라인을 표현할 수 있도록 한다. 계산된 의심도에 따라 라인을 순위화 하고 높은 라인에 색칠을 수행한다. 시각 효과를 극대화 하기 위해 상위 1~3위까지의 라인에는 빨간색을 3~10위까지의 라인에는 주황색으로 표시하고, 그 외에 평균 이상의 의심스러운 라인에 대해서는 노란색으로 표시한다.

B. Performance Requirements

디버깅에 소요되는 시간을 줄이기 위한 도구이기 때문에 결함을 추적하는데 소요되는 시간을 최소화 해야 한다. "ISES" 프로그램은 모든 알고리즘에 대한 정확도를 보장하여야 한다. 또한 더 정확하게 의심스러운 라인을 추적할 수 있어야 한다.

6.3.2. Organizational Requirements

A. Environmental Requirements

해당 프로그램은 유저가 모든 운영체제에서도 사용 가능한 프로그램을 구현하여 사용자의 편리성을 높인다.

B. Operational Requirements

해당 프로그램은 유저가 결함을 추적하고자 하는 소스코드의 개발 언어와 관계없이 모든 기능을 수행할 수 있어야 한다.

6.4. 시나리오

6.4.1. 프로그램 수행 시나리오

해당 프로그램을 수행하기 위해서 유저는 분석하고자 하는 소스코드와 테스트 스위트를 작성한다. 작성 후 해당 프로그램을 실행시킨다.

6.4.2. 소스코드 및 테스트 스위트 설정 시나리오

A. Normal flow

유저가 결함을 추적하고자 하는 소스코드 파일들이 위치한 폴더와 테스트 스위트들이 작성된 파일의 위치를 설정하고 등록한다.

B Possible errors

테스트 스위트들과 소스코드 파일의 매칭이 제대로 되지 않아 테스트 스위트 중 잘못된 테스트 케이스들에 대해 수정을 요청하고, 결함 추적을 수행하지 않는다.

C. System state on completion

올바르게 수행될 것으로 판단되면 수행하고자 하는 테스트 케이스들과 테스트케이스 최적화 기법 및 추적 기법에 대해 유저에게 선택을 요청한다.

6.4.3. 테스트스위트 수행 시나리오

A. Normal flow

유저가 소스코드에 수행하고자 하는 테스트 케이스들의 체크박스를 선택하고, 테스트 케이스 최적화 기법들과 추적 기법들을 선택하고 실행을 클릭한다.

B. Possible errors

유저가 기법들을 잘못 선택했을 수 있는 상황을 고려해 해당 기법들을 재확인 받고 취소할 수 있다.

C. System state on completion

해당 기법들이 적용된 소스코드와 테스트 스위트 수행이 시작되고, 분석 결과가 출력된다.

6.4.4. 분석 결과 확인 시나리오

A. Normal flow

유저는 선택한 기법들을 토대로 나온 분석 결과를 확인한다. 소스코드의 각 라인별로 의심스러운 순위와 점수가 출력되고 유저는 이를 확인한다.

6.4.5. 분석 결과 저장 시나리오

A. Normal flow

유저가 분석 결과를 파일로 저장하기 위해 분석 결과 저장을 누르고 파일명을 입력 받는다.

7. System Models

7.1. Objectives

이 섹션에서는 프로그램 컴포넌트와 프로그램 환경 간의 관계를 다이어그램으로 표현한다. 이를 통해 프로그램 전체적으로 눈으로 보이도록 하는 것을 목표로 한다.

7.2. Context Models

7.2.1. Context Model

ISES 프로그램은 다음과 같은 프로그램 관계를 갖고 있다.

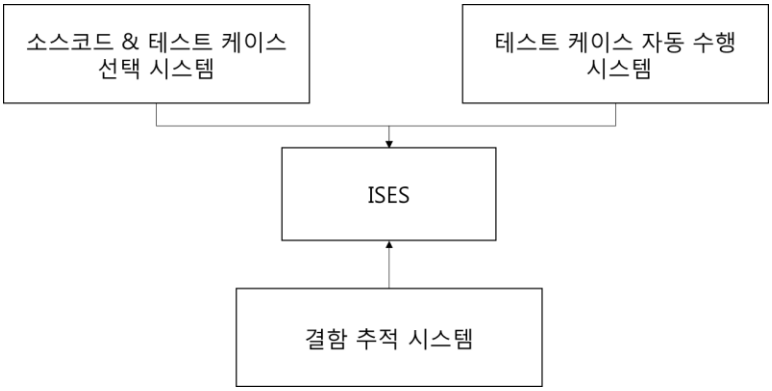

그림 6 ISES의 Context Model

7.2.2. Process Diagram

ISES 프로그램의 전반적인 프로세스를 볼 수 있도록 유저가 직접 사용하는 ISES 프로그램을 기준으로 프로세스 다이어그램을 구성하였다. ISES 프로그램은 결함 추적을 위한 소스코드 및 테스트 케이스를 선택하고, 테스트 케이스 우선순위화 기술 적용 여부, 결함 추적 분석 기법 선택 여부에 따라 결함 추적 결과를 제공하며 결과 내용을 저장 할 수 있는 기능으로 구성된다.

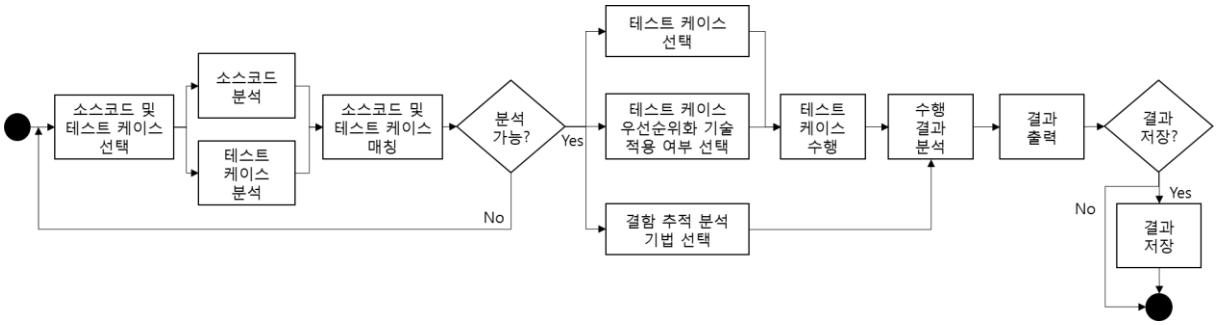

그림 7 ISES의 Process Diagram

7.3. Interaction models

7.3.1. Use case modeling

A. 프로그램 전체 Actor와 use case

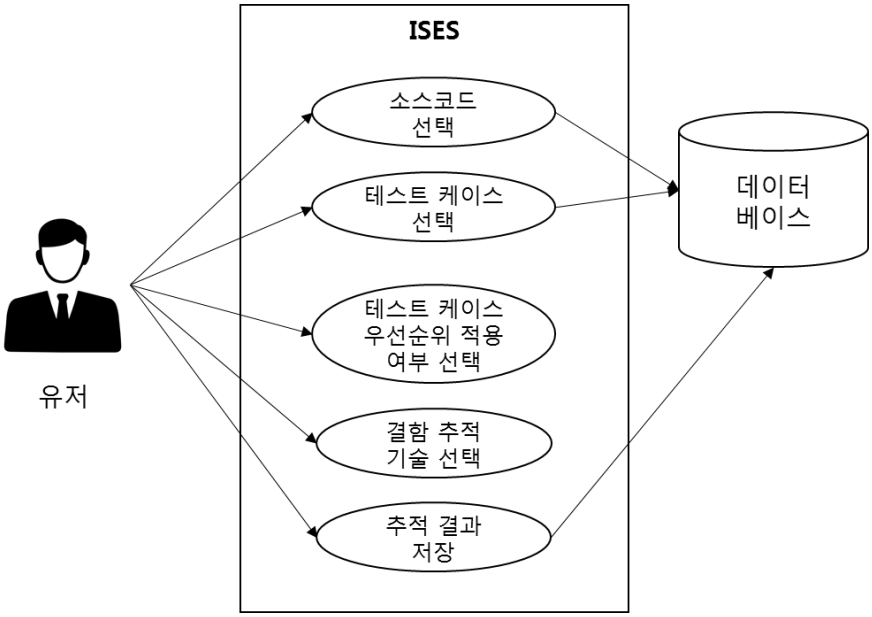

그림 8 프로그램 전체 Actor와 Use case

B. 소스코드 선택

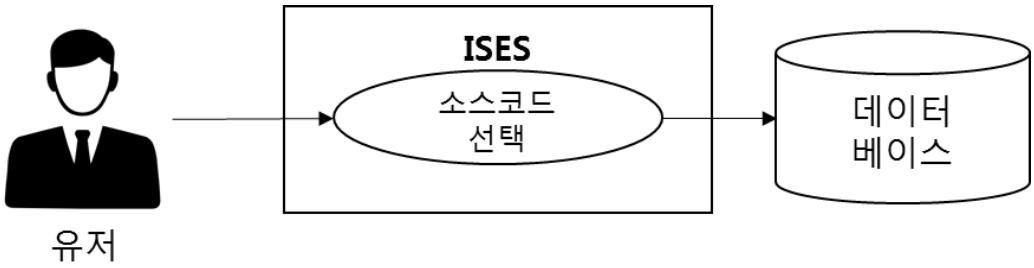

그림 9 소스코드 선택 use case

Actor	유저
Description	유저가 결함 추적을 위한 소스코드를 결정하여 ISES에 제공한다.
Data	소스코드 파일
Stimuli	ISES을 실행하고 소스코드 파일 위치를 설정한다.
Response	소스코드 파일이 결함 추적을 위해 등록되고 데이터 베이스에 저장된다.
Comments	소스코드 파일의 위치가 정확하여야 설정할 수 있다.

C. 테스트 케이스 선택

그림 10 테스트 케이스 선택 use case

Actor	유저
Description	유저가 결함 추적을 위해 선택한 소스코드를 테스트 하기 위한 테스트 케이스들을 선택한다.
Data	선택된 테스트 케이스
Stimuli	ISES을 실행하고 테스트 케이스들의 위치를 설정한다.
Response	테스트 케이스들이 소스코드를 수행하기 위해 등록되고 데이터 베이스에 저장된다.
Comments	테스트 케이스들이 소스코드를 수행시킬 수 있는 값들을 가져야 한다.

D. 테스트 케이스 우선순위 적용 여부 선택

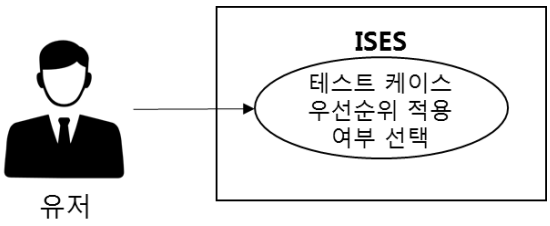

그림 11 테스트 케이스 우선순위 적용 여부 선택 use case

Actor	유저
Description	유저가 테스트 케이스 우선순위화 기술의 적용여부를 선택한다.
Data	테스트 케이스 우선순위 적용 여부
Stimuli	1) 우선순위 적용을 선택하면 테스트 케이스 우선순위화 기술을 통해 테스트 케이스들을 우선순위화 한다. 2) 우선순위 적용을 선택하지 않으면 기존에 선택된 테스트 케이스 모두를 저열 없이 사용한다.
Response	1) 우선순위화된 테스트 케이스들이 결함 추적을 위해 사용된다. 2) 우선순위화 되지 않은 테스트 케이스들이 결함 추적을 위해 사용된다.
Comments	테스트 케이스들이 미리 선택되어야 수행 가능하다.

E. 결함 추적 기술 선택

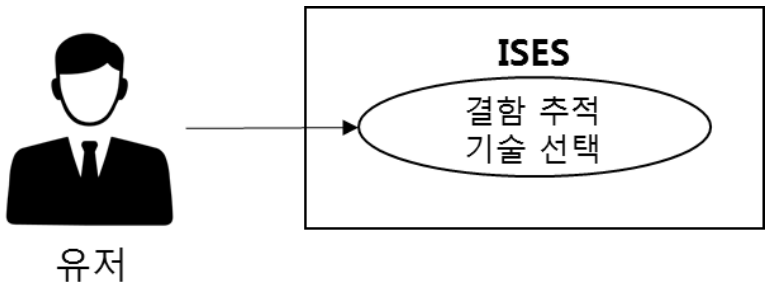

그림 12 결함 추적 기술 선택 use case

Actor	유저
Description	76가지의 결함 추적 기술들을 유저에게 보여주고 적용하고자 하는 알고리즘을 선택하면 해당 알고리즘에 기반하여 결함 추적이 수행된다.
Data	적용할 결함 추적 기술 목록
Stimuli	결함 추적을 수행 명령
Response	선택된 결함 추적 기술에 기반하여 결함 추적을 수행한다.
Comments	결함 추적 기술이 최소 1개 이상 선택되어야 한다.

F. 추적 결과 저장

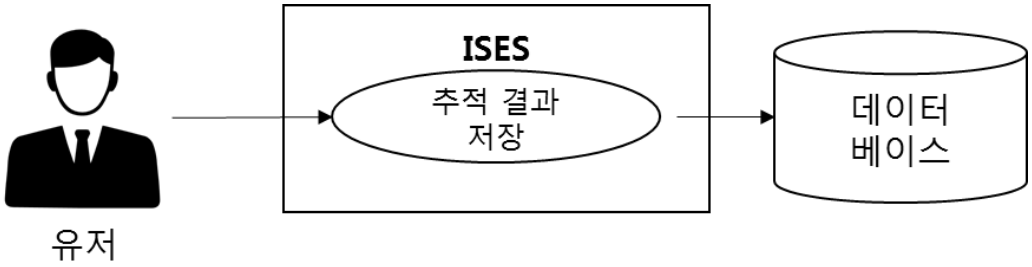

그림 13 추적 결과 저장 use case

Actor	유저
Description	유저에게 시각적으로 제공된 분석 결과를 파일로 저장한다.
Data	결함 추적 결과
Stimuli	결함 추적 결과 저장 명령
Response	분석 결과를 파일로 저장한다.
Comments	파일 형태는 선택할 수 있도록 한다.

7.3.2. Sequence Model

A. 소스코드 선택

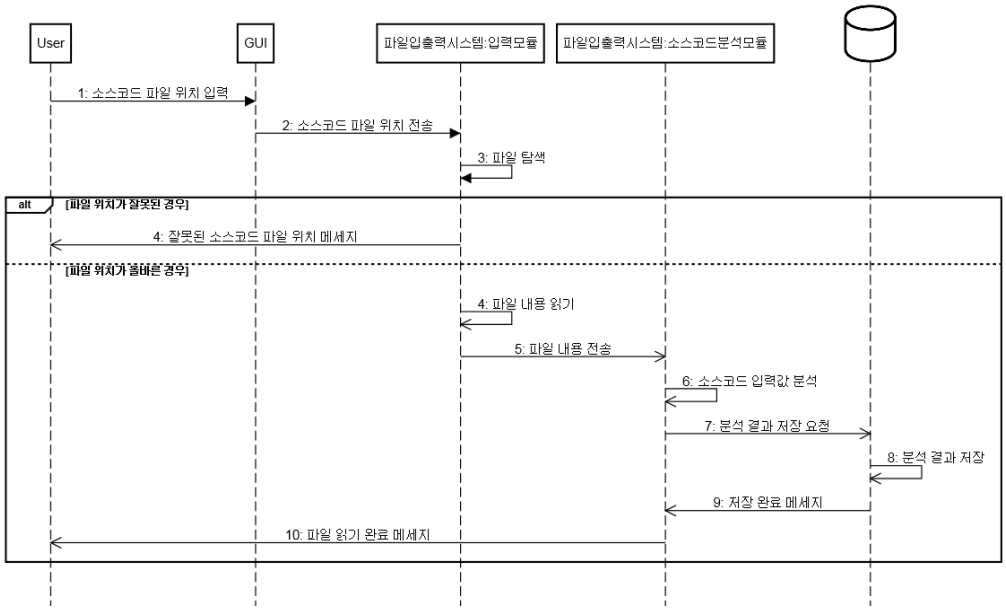

그림 14 소스코드 선택 sequence model

B. 테스트 케이스 선택

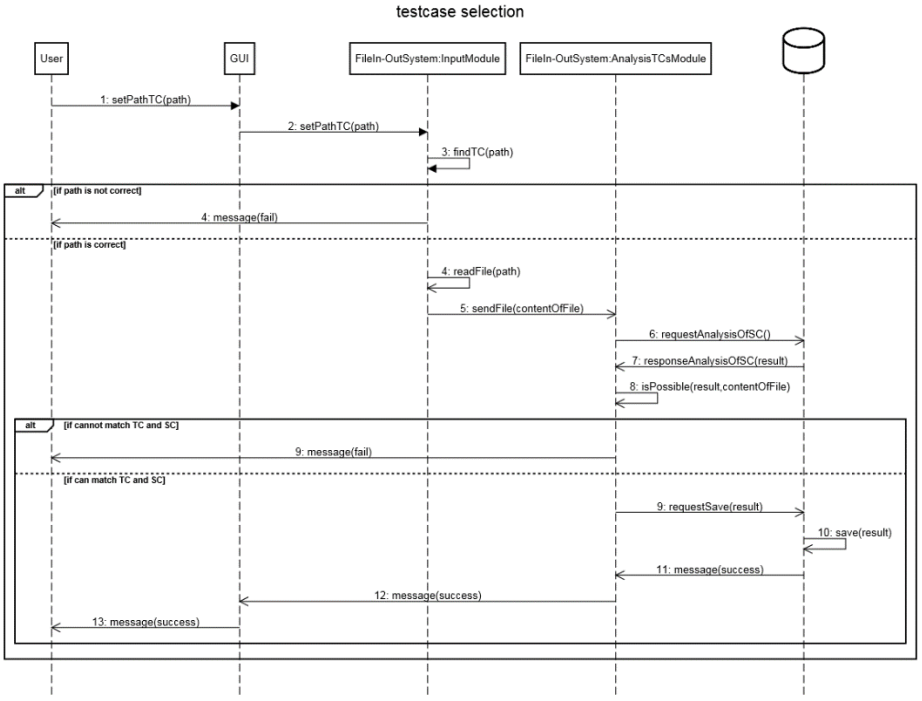

그림 15 테스트 케이스 선택 sequence model

C. 테스트 케이스 우선순위 적용 여부 선택

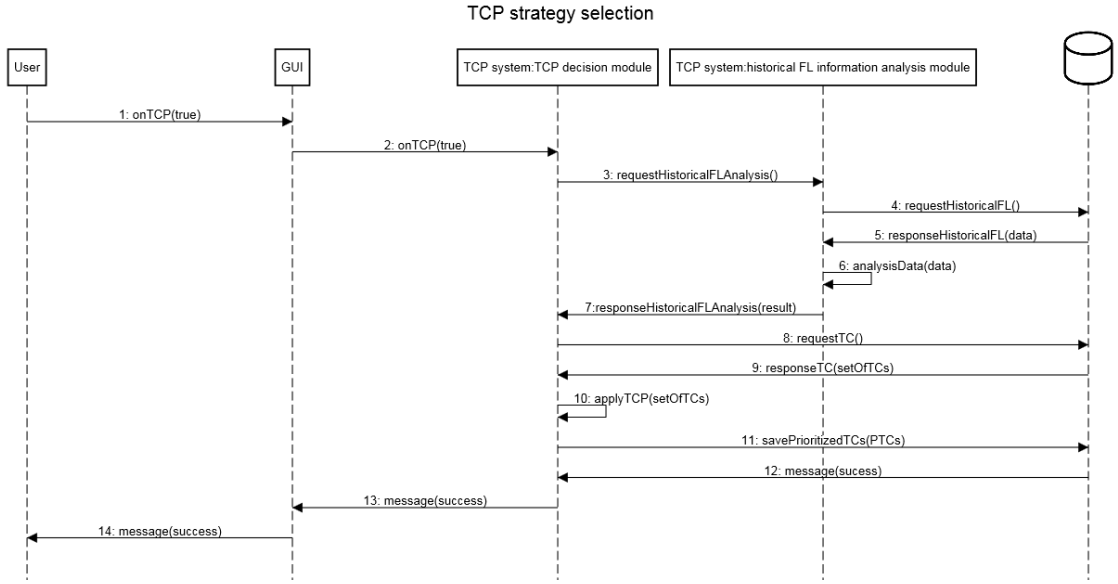

그림 16 테스트 케이스 우선순위 적용 여부 선택 sequence model

D. 결함 추적 기술 선택

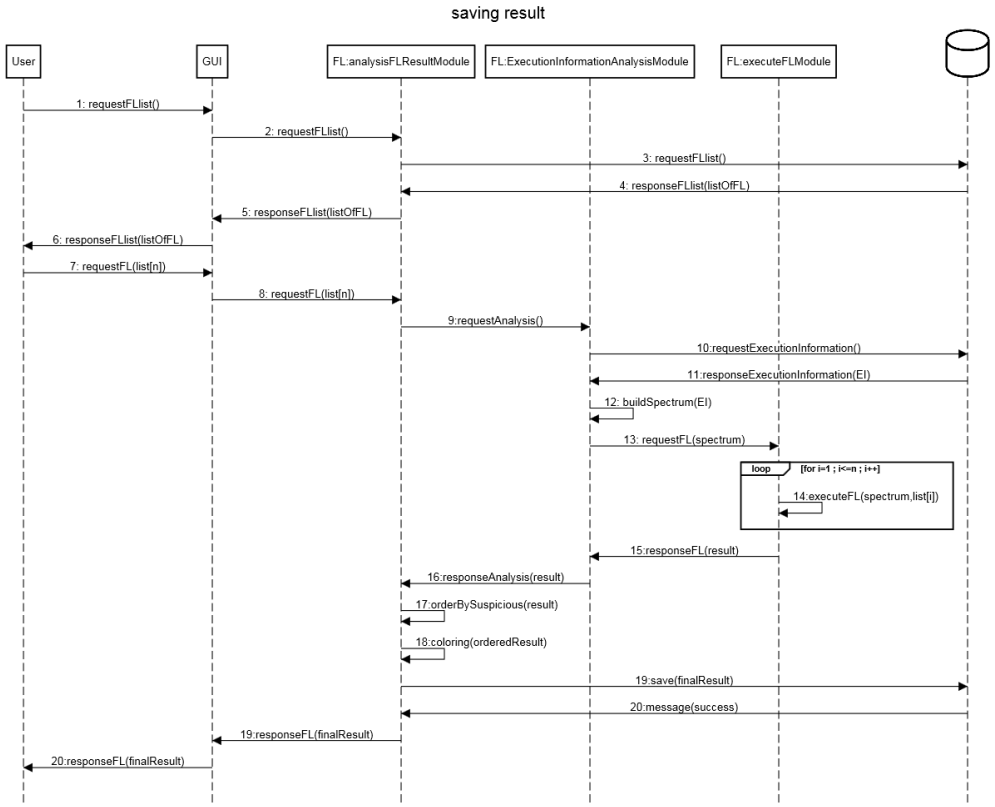

그림 17 결함 추적 기술 선택 sequence model

E. 추적 결과 저장

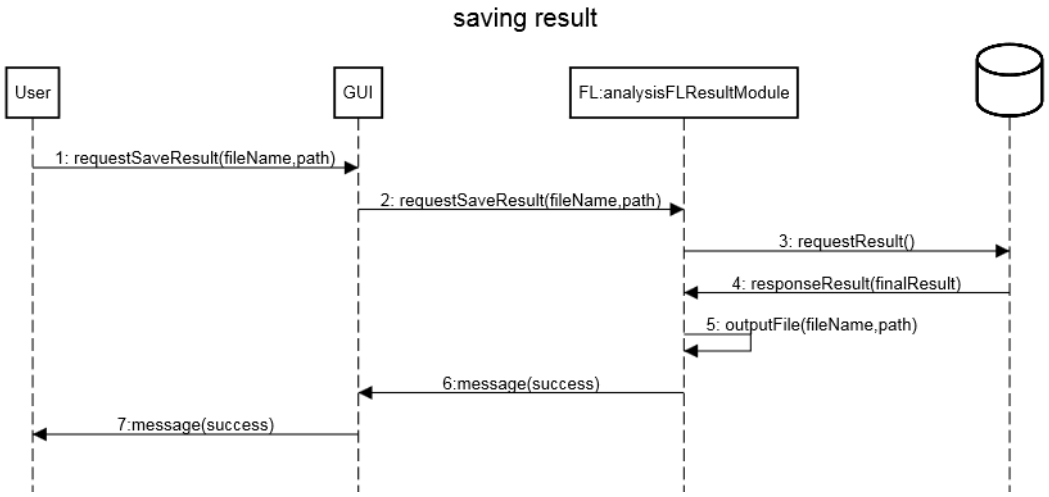

그림 18 추적 결과 저장 sequence model

7.3.3. Structural models

A. Overall System class diagram

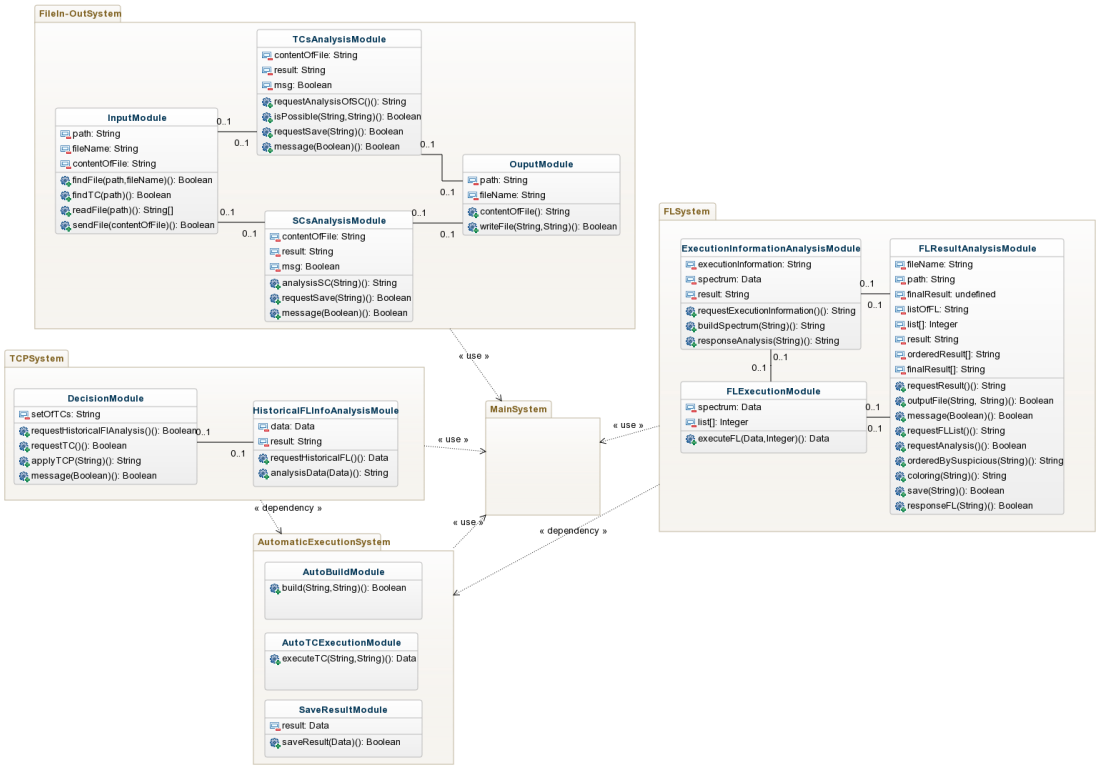

그림 19 Overall System Class diagram

7.3.4. Behavioral models

A. Data-driven modeling

유저 입장에서 2개 Action으로의 소스코드 및 테스트 케이스 입력, 결함 추적 알고리즘 요청이 있다.

- 소스코드 및 테스트 케이스 입력

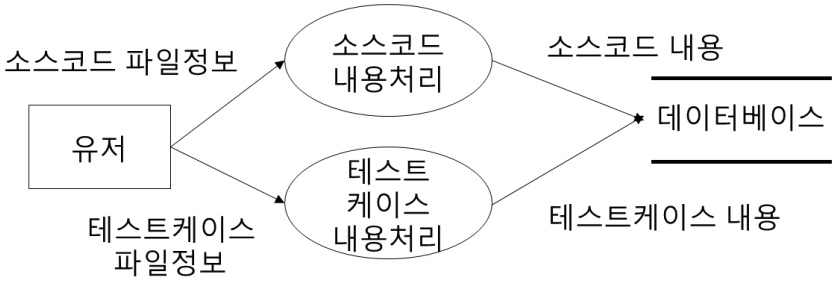

그림 20 소스코드 및 테스트 케이스 선택 data-driven model

- 결함위치 추적 알고리즘 요청

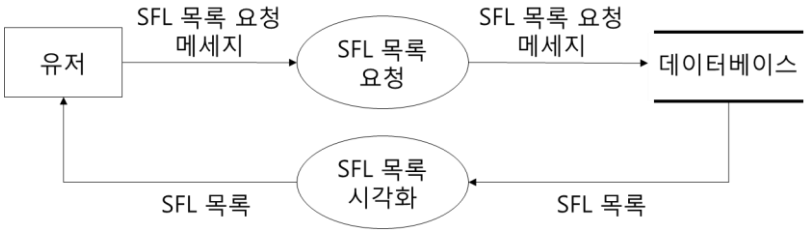

그림 21 결함 위치 추적 알고리즘 요청 data-driven model

8. System evolution

8.1. Objectives

이 섹션에서는 ISES 이 기반하는 기본적인 가정들에 대해 설명한다. 그리고 프로그램에 일어날 수 있는 가능한 모든 변경에 대해서도 설명한다. 하드웨어의 발전, 사용자 요구의 변경과 같은 것들에 대해 설명한다. 이는 후에 프로그램 변경 시에 유의해야 할 사항들을 이 장을 통해 참고할 수 있도록 하기 위함이다.

8.2 Assumption

- 일반적인 개발자는 SFL 알고리즘에 대한 구현 이해도가 낮다.

- 기업에서의 회귀테스트는 제한적인 시간 내에 끝내야 한다.

- 소프트웨어가 지속적으로 개발됨에 따라 많은 테스트 케이스들이 존재한다.

- 테스트 케이스들 중 하나 이상의 실패한 테스트 케이스가 존재한다,

8.3. Evolution of Hardware

현재 프로그램은 GUI 를 갖는 개발 소프트웨어로 개발되었으나 온라인으로 사용될 수 있도록 발전해 나가야 할 것이다

8.4. Evolution of User Requirement

user requirement 가 바뀌면 그에 맞게 software 적인 면에서 기능을 추가해 user 를 만족시켜야 할 것이다. UI 의 측면에서 유저는 조금더 평리하게 확인하기를 원할 수 있다. 또한 예를 들어 이 "ISES" 에 새로운 알고리즘을 유저가 요청함에 따라 이를 추가할 수 있어야 한다. 관련 알고리즘들은 현재도 지속적으로 제안되고 있으며 테스트 케이스 활용 연구들 또한 활발이 연구중이기 때문에 이를 추가하는 것은 바람직한 일이다. 해당 기술과 UI 는 개발자가 디버깅을 함에 있어 개별 소프트웨어가 아닌 전체 디버깅을 위한 프로그램을 위해 함께 사용될 수 있다. 엑셀 파일 외의 다른 인풋을 원할 수 있는 상황도 고려하여야 한다 또한 현재는 C 나 java 에 초점을 맞춘 결함 추적이 가능하나 소프트웨어에 다른 언어로 개발된 소프트웨어오 추적할 수 있도록 개발되어야할 것이다. 또한 현재 버그리포트 관련 연구들이 진행되고 있으며 테스트 케이스 외에 버그리포트의 디버깅에 유효한 정보로 활용되고 이씩 때문에 이에 대한 디버깅을 돕기 위한 기능들도 추가될 수 있다. 이에 따라 발정될 수 있어야 한다.

8.5. Evolution of Environment

현재 프로그램은 GUI 를 갖는 개발 소프트웨어로 개발되었으나 온라인으로 사용될 수 있도록 발전해 나가야 할 것이다

9. Appendices

9.1. Objectives

해당 세션에서는 이전 장에서 설명하지 못했던 ISES 에 대한 더 자세하고 구체적인 정보들을 제공한다. 사용하고 있는 기술들, 이 프로그램을 개발하기 위해 사용할 개발 기법 및 언어에 대해 소개한다.

9.2. 적용 개발 프로세스 V Model

V 모델은 폭포수 모델에 프로그램 검증과 테스트 작업을 강조한 모델이다. 상세화에 초점을 둔 과정과 검증에 초점을 둔 과정으로 나뉘어져 있다. 최상위층은 요구의 추출과 운영을 다루고 있고, 중간층은 문제의 이해를 소프트웨어 구조로 매핑하는데 집중하고 하위층은 모듈의 조립과 코딩에 초점을 두고 있다. 모든 단계에 검증과 확인과정이 있기에 오류를 줄일 수 있다. 그러나 생명주기의 반복을 허용하지 않아 변경을 다루기는 쉽지가 않다. 때문에 요구의 명세가 아주 확실하여 개발하는 동안에 변경이 없어야 바람직하다.

9.3. 개발 언어: C#

C#은 마이크로 소프트웨어에서 개발된 언어로, 모든 것을 객체로 취급하는 커뮺넌트 프로그래밍 언어이다. 기본적인 개발 언어인 C 를 확장한 C++에 기본을 두고, 닷넷(.net)플랫폼을 위해 개발되었다. 해당 언어는 C++에 기반을 두면서도, 비주억 베이직이나 자바와도 비슷하기 때문에, 이들 모두의 장점을 지닌다. 즉 비주얼 언어가 가진 사용자 친화성, C++의 객체 지향성, 자바의 분산환경 처리에 적합한 다중성등을 모두 지니는 컴포넌트 기반의 소프트웨어 개발 패러다임을 반영한다. 이 언어를 통해 하나 이상의 OS 에서도 사용할 수 있는 응용 프로그램을 만들 수 있기 때문에, 소프트웨어나 서비스를 값싸고 빠르게 시장에 내놓을 수 있다.

9.4. 테스트 케이스 우선순위화 기술

해당 프로그램에 구현된 테스트 케이스 우선순위화 기술은 해당 기술들 중 과거 시험 정보를 이용한 테스트 케이스 우선순위화 기법들 중 하나를 적용하였다. 해당 기술은 그림과 같이 테스트 케이스 세트의 실행에 대하여 특정 크기로 설정된 윈도우를 기준으로 테스트 항목의 과거 결함 발견 여부와 실행 여부를 가지고 우선순위화 하는 기술이다.

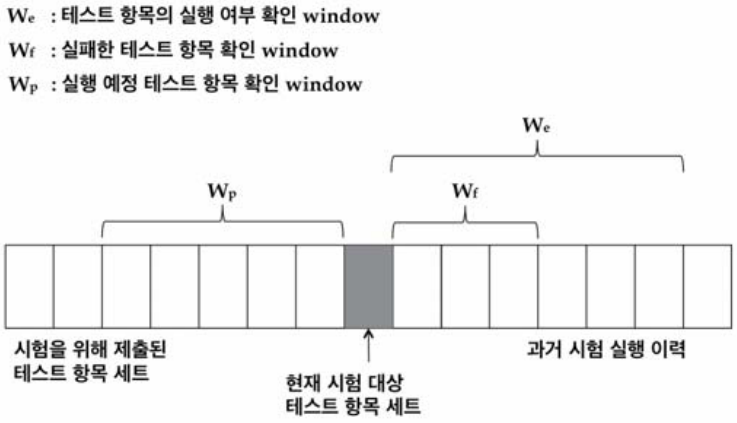

그림 22 테스트 케이스 우선순위화 기술

9.5. SFL 기술

본 ISES 에 구현될 SFL 기술들은 아래 표기된 총 76 개의 식을 통하여 설계된다.

$$S_{JACCARD} = \frac{a}{a+b+c} \quad (1)$$

$$S_{DICE} = \frac{2a}{2a+b+c} \quad (2)$$

$$S_{CZEKANOWSKI} = \frac{2a}{2a+b+c} \quad (3)$$

$$S_{3W-JACCARD} = \frac{3a}{3a+b+c} \quad (4)$$

$$S_{NEI\&LI} = \frac{2a}{(a+b)+(a+c)} \quad (5)$$

$$S_{SOKAL\&SNEATH-I} = \frac{a}{a+2b+2c} \quad (6)$$

$$S_{SOKAL\&MICHENER} = \frac{a+d}{a+b+c+d} \quad (7)$$

$$S_{SOKAL\&SNEATH-II} = \frac{2(a+d)}{2a+b+c+2d} \quad (8)$$

$$S_{ROGER\&TANIMOTO} = \frac{a+d}{a+2(b+c)+d} \quad (9)$$

$$S_{FAITH} = \frac{a+0.5d}{a+b+c+d} \quad (10)$$

$$S_{GOWER\&LEGENDRE} = \frac{a+d}{a+0.5(b+c)+d} \quad (11)$$

$$S_{INTERSECTION} = a \quad (12)$$

$$S_{INNERPRODUCT} = a+d \quad (13)$$

$$S_{RUSSELL\&RAO} = \frac{a}{a+b+c+d} \quad (14)$$

$$D_{HAMMING} = b+c \quad (15)$$

$$D_{EUCLID} = \sqrt{b+c} \quad (16)$$

$$D_{SQUARED-EUCLID} = \sqrt{(b+c)^2} \quad (17)$$

$$D_{CANBERRA} = (b+c)^{\frac{2}{2}} \quad (18)$$

$$D_{MANHATTAN} = b+c \quad (19)$$

$$D_{MEAN-MANHATTAN} = \frac{b+c}{a+b+c+d} \quad (20)$$

$$D_{CITYBLOCK} = b+c \quad (21)$$

$$D_{MINKOWSKI} = (b+c)^{\frac{1}{1}} \quad (22)$$

$$D_{PATTERNDIFFERENCE} = \frac{4bc}{(a+b+c+d)^2} \quad (26)$$

$$D_{LANCE\&WILLIAMS} = \frac{b+c}{(2a+b+c)} \quad (27)$$

$$D_{BRAY\&CURTIS} = \frac{b+c}{(2a+b+c)} \quad (28)$$

$$D_{HELLINGER} = 2\sqrt{\left(1-\frac{a}{\sqrt{(a+b)(a+c)}}\right)} \quad (29)$$

$$D_{CHORD} = \sqrt{2\left(1-\frac{a}{\sqrt{(a+b)(a+c)}}\right)} \quad (30)$$

$$S_{COSINE} = \frac{a}{\sqrt{(a+b)(a+c)^2}} \quad (31)$$

$$S_{GILBERT\&WELLS} = \log a - \log n - \log(\frac{a+b}{n}) - \log(\frac{a+c}{n}) \quad (32)$$

$$S_{OCHIAI-I} = \frac{a}{\sqrt{(a+b)(a+c)}} \quad (33)$$

$$S_{FORBESI} = \frac{na}{(a+b)(a+c)} \quad (34)$$

$$S_{FOSSUM} = \frac{n(a-0.5)^2}{(a+b)(a+c)} \quad (35)$$

$$S_{SORGENFREI} = \frac{a^2}{(a+b)(a+c)} \quad (36)$$

$$S_{MOUNTFORD} = \frac{a}{0.5(ab+ac)+bc} \quad (37)$$

$$S_{OTSUKA} = \frac{a}{((a+b)(a+c))^{0.5}} \quad (38)$$

$$S_{MCCONNAUGHEY} = \frac{a^2-bc}{(a+b)(a+c)} \quad (39)$$

$$S_{TARWID} = \frac{na-(a+b)(a+c)}{na+(a+b)(a+c)} \quad (40)$$

$$S_{KULCZYNSKI-II} = \frac{\frac{a}{2}(2a+b+c)}{(a+b)(a+c)} \quad (41)$$

$$S_{DRIVER\&KROEBER} = \frac{a}{2}(\frac{1}{a+b}+\frac{1}{a+c}) \quad (42)$$

$$S_{JOHNSON} = \frac{a}{a+b}+\frac{a}{a+c} \quad (43)$$

$$S_{DENNIS} = \frac{ad-bc}{\sqrt{n(a+b)(a+c)}} \quad (44)$$

$$S_{SIMPSON} = \frac{a}{\min(a+b,a+c)} \quad (45)$$

$$S_{BRAUN\&BANQUET} = \frac{a}{\max(a+b,a+c)} \quad (46)$$

요구사항 명세서

$$S_{FAGER\&McGOWAN} = \frac{a}{\sqrt{(a+b)(a+c)}} - \frac{\max(a+b, a+c)}{2} \quad (47)$$

$$S_{FORBES-II} = \frac{na-(a+b)(a+c)}{n\min(a+b, a+c)-(a+b)(a+c)} \quad (48)$$

$$S_{SOKAL\&SNEATH-IV} = \frac{\frac{a}{(a+b)}+\frac{a}{(a+c)}+\frac{d}{(b+d)}+\frac{d}{(b+d)}}{4} \quad (49)$$

$$S_{GOWER} = \frac{a+d}{\sqrt{(a+b)(a+c)(b+d)(c+d)}} \quad (50)$$

$$S_{PEARSON-I} = \chi^2 \text{ where } \chi^2 = \frac{n(ad-bc)^2}{(a+b)(a+c)(c+d)(b+d)} \quad (51)$$

$$S_{PEARSON-II} = \left(\frac{\chi^2}{n+\chi^2}\right)^{1/2} \quad (52)$$

$$S_{PEARSON-III} = \left(\frac{\rho}{n+\rho}\right)^{1/2} \text{ where } \rho = \frac{ad-bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}} \quad (53)$$

$$S_{PEARSON\&HERON-I} = \frac{ad-bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}} \quad (54)$$

$$S_{PEARSON\&HERON-II} = Cos\left(\frac{\pi\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}\right) \quad (55)$$

$$S_{SOKAL\&SNEATH-III} = \frac{a+d}{b+c} \quad (56)$$

$$S_{SOKAL\&SNEATH-V} = \frac{ad}{(a+b)(a+c)(b+d)(c+d)^{0.5}} \quad (57)$$

$$S_{COLE} = \frac{\sqrt{2}(ad-bc)}{\sqrt{(ad-bc)^2-(a+b)(a+c)(b+d)(c+d)}} \quad (58)$$

$$S_{STILES} = \log_{10}\frac{n(|ad-bc|-\frac{n}{2})^2}{(a+b)(a+c)(b+d)(c+d)} \quad (59)$$

$$S_{OCHIAI-II} = \frac{ad}{\sqrt{(a+b)(a+c)(b+d)(c+d)}} \quad (60)$$

$$S_{YULEQ} = \frac{ad-bc}{ad+bc} \quad (61)$$

$$D_{YULEQ} = \frac{2bc}{ad+bc} \quad (62)$$

$$S_{YULEw} = \frac{\sqrt{ad}-\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}} \quad (63)$$

$$S_{KULCZYNSKI-I} = \frac{a}{b+c} \quad (64)$$

$$S_{TANIMOTO} = \frac{a}{(a+b)+(a+c)-a} \quad (65)$$

$$S_{DISPERSON} = \frac{ad-bc}{(a+b+c+d)^2} \quad (66)$$

$$S_{HAMANN} = \frac{(a+d)-(b+c)}{a+b+c+d} \quad (67)$$

$$S_{MICHAEL} = \frac{4(ad-bc)}{(a+d)^2+(b+c)^2} \quad (68)$$

$$S_{GOODMAN\&KRUSKAL} = \frac{\sigma-\sigma'}{2n-\sigma'} \text{ where}$$
$$\sigma = \max(a,b)+\max(c,d)+\max(a,c)+\max(b,d),$$
$$\sigma' = \max(a+c,b+d)+\max(a+b,c+d) \quad (69)$$

$$S_{ANDERBERG} = \frac{\sigma - \sigma'}{2n} \quad (70)$$

$$S_{BARONI-URBANI\&BUSER-I} = \frac{\sqrt{ad} + a}{\sqrt{ad} + a + b + c} \quad (71)$$

$$S_{BARONI-URBANI\&BUSER-II} = \frac{\sqrt{ad} + a - (b + c)}{\sqrt{ad} + a + b + c} \quad (72)$$

$$S_{PEIRCE} = \frac{ab + bc}{ab + 2bc + cd} \quad (73)$$

$$S_{EYRAUD} = \frac{n^2(na - (a+b)(a+c))}{(a+b)(a+c)(b+d)(c+d)} \quad (74)$$

$$S_{TARANTULA} = \frac{\frac{a}{(a+b)}}{\frac{c}{(c+d)}} = \frac{a(c+d)}{c(a+b)} \quad (75)$$

$$S_{AMPLE} = \left| \frac{\frac{a}{(a+b)}}{\frac{c}{(c+d)}} \right| = \left| \frac{a(c+d)}{c(a+b)} \right| \quad (76)$$

10. Index

10.1. 그림 Index