딥러닝 기반 국방 SW 결함위치추정을 위한 변이 기반 데이터셋 구축의 체계적 연구

양희찬[O]
KAIST
heechan.yang@kaist.ac.kr

이아청
KAIST
ahcheong.lee@kaist.ac.kr

조규태
LIG Nex1
kyutae.cho2@lignex1.com

김문주
KAIST/브이플러스랩
moonzoo.kim@gmail.com

Systematic Study of Mutation-Based Dataset Construction for Deep Learning-Based Fault Localization on Military Defense SW

Heechan Yang[O]
KAIST

Ahcheong Lee
KAIST

Kyutae Cho
LIG Nex1

Moonzoo Kim
KAIST/브이플러스랩

요 약

딥러닝 기반 결함 위치 추정(Deep Learning-Based Fault Localization, DLFL)은 소프트웨어 디버깅 자동화 분야에서 혁신적인 성과를 거두고 있으나, 학습 데이터셋 구축에 막대한 계산 비용과 공개된 데이터셋 구축 도구의 부재로인해 실무 적용의 주요한 병목 현상으로 작용하고 있다. 본 논문은 DLFL의 실용성을 확보하기 위해 DLFL 데이터셋 구축 과정을 최적화하는 체계적인 방법론을 제시하고, 이를 자동화하는 도구를 구현하여 실제 L사의 국방 무기체계 소프트웨어에 적용하여 그 유효성을 입증한다.

먼저, 데이터셋 구축 비용에 결정적인 영향을 미치는 두 가지 핵심 파라미터인 (1) '변이체 생성 대상 라인 선택 비율'과 (2) '라인당 변이체 생성 개수'의 최적 임계값을 도출하기 위해 오픈소스 자바 벤치마크인 Defects4J를 대상으로 탐색적 실험을 수행하였다. 실험 결과, 기존 방식 대비 **계산 비용을 74.6% 단축**하면서도 결함 추정 성능을 유지할 수 있는 최적의 설정값을 확인하였다. 또한, 본 연구에서는 개발자의 디버깅 직관을 정량적으로 모델링한 (3) 스택 트레이스(Stack Trace, ST) 관련성 특징을 최초로 설계하여 제안하였으며, 이를 학습 특징으로 추가함으로써 결함 위치 추정 정확도를 기존 대비 **6.8%~11.0% 향상**시켰다.

최종적으로, 도출된 최적 파라미터와 신규 ST 특징 추출 로직을 탑재한 자동화 도구를 L사의 6개 국방 무기체계 소프트웨어(총 61 KLoC, 300개 인공 결함) 시스템에 탑재하여 기술의 실전 배치 가능성을 검증하였다. 그 결과, **Top-5 기준 85.0%의 높은 탐지 정확도**를 달성함과 동시에 데이터셋 구축 비용을 **약 79%(9,081→1,907 CPU-hours) 절감**하였다. 이를 통해 본 연구가 개발한 자동화 도구가 고신뢰 국방 소프트웨어 분야에서 실무자가 별도의 전문 지식 없이도 고성능 DLFL 기술을 즉각 운용할 수 있는 실질적인 기술적 토대를 마련하였음을 입증하였다.

1. 서론

소프트웨어 결함 위치 추정(Fault Localization, FL)은 프로그램 실패의 원인이 되는 특정 코드 요소(파일, 함수, 라인 등)를 식별하여 개발자의 디버깅 노력을 최소화하는 것을 목표로 한다. 전통적인 기법으로는 테스트 커버리지 패턴을 분석하는 스펙트럼 기반 FL(SBFL)[1]과 변이 프로그램을 활용하여 결함의 전파 과정을 분석하는 변이 기반 FL(MBFL)[2,3]이 존재한다. 최근에는 이런 전통적인 기법으로부터 추출된 데이터를 신경망에 학습시킨 딥러닝 기반 결함 위치 추정(DLFL) 기술이 기존 기법들보다 높은 정확도를 보이며 차세대 FL 기술로 주목받고 있다[4, 5, 6].

그러나 기존 MBFL 기반 DLFL 연구들은 주로 모델의 구조적 고도화에만 집중할 뿐, 모델의 성능과 비용을 좌우하는 **데이터셋 구축 방법론**에 대해서는 표준화된 절차없이 연구자의 임의적인 방식에 의존하고 있다. 따라서, 실무 환경에서는 MBFL 기반 DLFL을 적용하는데 다음과 같은 어려움이 따른다. 1. 공개된 도구나 방법론이 없어 적용과정에 많은 불확실성이 생긴다. 2. 데이터셋 구축 시 막대한 계산 자원과 시간 비용이 예상된다. 일례로, 61 KLoC 규모의 소프트웨어에 대한 DLFL 데이터셋을 구축에 임의적인 방법으로 시행할 시 약 9,081 CPU-hours이 소요될 것으로 예상되며, 이는 자원이 한정된 실무 환경에서 MBFL 기반 DLFL 기술을 도입하는 데 큰 진입 장벽이 된다.

본 연구는 이러한 한계를 극복하고 **최종적으로 L사의 실제 국방 무기체계 소프트웨어에 DLFL 기술을 실용적으로 적용하는 것을 목표**로 한다. 이를 위해 본 논문은 자동화 도구 개발, 탐색적 실험, 그리고 실무 사례 연구로 이어지는 연구 방법론을 채택한다.

본 연구의 주요 기여는 다음과 같다:

1. **데이터셋 구축 시간 비용의 74.6% 단축:** 체계적인 실험을 통해 변이체 생성 대상 라인 선택 비율(70%)과 라인당 변이체 생성 개수(3개)의 최적값을 도출, 기존 방식 대비 데이터셋 구축 시간 비용을 74.6% 절감하였다.
2. **개발자의 디버깅 직관을 정량적으로 모델링한 '스택 트레이스(Stack Trace, ST) 관련성 특징 설계를 통한 정확도 향상:** 본 연구에서 최초로 고안한 ST 관련성 특징을 수치화 하여 기존 특징(SBFL, MBFL)과 결합하여 활용하여, DLFL 모델의 결함 탐지 성능을 기존 대비 6.8%~11.0% 향상시켰다. 기존 연구들이 실행 경로의 통계적 유의성에만 의존했던 것과 달리, 본 특징은 프로그램 비정상 종료 시 발생하는 런타임 문맥 정보를 딥러닝 모델이 직접 학습할 수 있도록 설계된 독창적인 기술적 기여이다.
3. **국방 무기체계 SW 적용을 통한 실무적 실효성 검증:** 탐색적 연구에서 얻은 최적 가이드라인과 신규 특징을 바탕으로 직접 개발한 데이터셋 자동 구축 도구를 L사의 실제 개발 환경에 통합하였으며, 6개 국방 무기체계 소프트웨어에 적용하여 Top-5 기준 85.0%의 높은 정확도를 달성하고, 시간 비용을 79% 단축하여 제안 방법론의 실무적 유용성을 입증하였다.

이 논문은 산학연주관 핵심 SW(응용연구) 연구개발 과제(계약번호 UC210018AD)와 정부의 재원으로 한국연구재단의 지원(NRF-2021R1A5A1021944, NRF-RS-2024-00357348)의 지원을 받아 수행된 연구임.

2. 배경

2.1 데이터셋 구성 요소

MBFL 기반 DLFL 모델의 학습 데이터셋은 소스 코드의 각 라인을 하나의 레코드(Record)로 구성한다. 기존의 일반적인 데이터셋은 각 라인에 대하여 총 8개의 특징값을 포함하며, 이는 6개의 서로 다른 SBFL 기법과 2개의 MBFL 기법으로 도출된 의심도 점수로 이루어진다.

2.2 특징 추출 방법

데이터셋을 구성하는 특징들은 프로그램의 동적 실행 정보를 바탕으로 추출된다. SBFL 특징은 결함 프로그램에 테스트 스위트를 실행하여 얻은 라인 커버리지(Line Coverage) 정보를 활용하는 반면, MBFL 특징은 소스 코드에 인위적인 수정을 가하는 변이 분석(Mutation Analysis) 과정을 거쳐 추출된다. 구체적인 산출 방법은 다음과 같다.

2.2.1 스펙트럼 기반 결함위치추정(SBFL) 특징

SBFL 특징은 각 테스트 케이스의 실행 경로와 통과/실패 여부를 결합하여 특정 코드 라인이 결함을 포함하고 있을 가능성(의심도)을 계산한다. 따라서, 본 연구에서는 선행 연구들에서 성능이 검증된 6종의 SBFL 기법(DStar, GP13, Naish1, Naish2, Ochiai, Tarantula)을 활용하여 기본 학습 특징을 구축한다.

2.2.2 변이 기반 결함위치추정(MBFL) 특징

MBFL 특징은 코드 라인에 미세한 변경을 가한 변이체(Mutant)를 생성하고, 이에 대한 테스트 동작 변화를 분석하여 도출한다. 본 연구에서는 변이체 생성 도구인 MUSIC++[7]를 활용하여 대상 라인에 대해 M개의 변이체를 생성한다. MBFL은 변이 적용 전후의 실행 결과 변화를 통해 결함 위치를 추정하며, 본 연구에서는 대표적인 MBFL 기법인 MUSE[2]와 Metallaxis[3] 공식을 활용한다. MBFL은 SBFL보다 높은 정확도를 제공하지만, 모든 라인에 대해 수많은 변이체를 생성하고 테스트하므로 막대한 계산 비용이 발생한다는 한계가 있다.

3. 제안 방법론: 데이터셋 최적화 및 스택 트레이스(ST) 관련성 특징

본 연구는 MBFL 기반 DLFL의 실효성을 확보하기 위해 **(1) 데이터셋 구축의 시간 비용 단축**과 **(2) 결함 추정 정확도 향상**이라는 두 가지 핵심 목표를 설정한다. 이를 달성하기 위해 변이 분석 과정을 최적화하는 파라미터 탐색 전략을 수립하고 (3.1장), 개발자의 디버깅 문맥을 정량적으로 모델링한 새로운 특징 추출 방법을 제안한다 (3.2장).

특히, 본 연구에서는 제안된 방법론을 실제 소프트웨어 개발 현장에 즉각적으로 적용할 수 있도록 약 6,000라인의 파이썬 스크립트로 구성된 '범용적 MBFL 기반 DLFL 데이터셋 구축 자동화 도구'를 직접 설계하고 개발하였다. 이 도구는 복잡한 변이 분석 및 특징 추출 과정을 자동화하여 구축 비용을 절감하는 동시에, 본 연구의 핵심 기술적 기여인 ST 관련성 특징을 생성하는 중추적인 역할을 수행한다.

3.1 데이터셋 구축의 시간 비용 단축

DLFL 데이터셋 구축 과정에서 가장 많은 연산 자원이 소요되는 단계는 MBFL 특징 추출을 위한 변이 분석 단계이다. 본 연구는 변이 분석 비용에 결정적인 영향을 미치는 두 가지 핵심 파라미터를 정의하고, 개발된 자동화 도구를 통해 모델의 성능 저하를 최소화하면서도 비용을 극대화하여 절감할 수 있는 최적값을 탐색한다. 특히 국방 소프트웨어와 같은 대규모 시스템에 기술을 적용하기 전, 통제된 환경(Defects4J)에서의 탐색적 연구를 통해 보편적인 가이드라인을 도출하는 것이 본 전략의 핵심이다.

표 1: RQ1 '변이체 생성 대상 라인 비율' 실험

Line Selection Ratio	#Mutants Per Line	Top-1	Top-3	Top-5	MFR	p-value
100%	10	52.7	98.3	123.4	29.9	1.0000
90%		51.7	96.7	122.3	30.7	0.5536
80%		51.0	96.4	120.8	30.0	0.5967
70%		**50.7**	**92.5**	**117.4**	**31.4**	**0.0750**
60%		49.3	91.3	118.2	33.1	0.0495
50%		47.4	87.7	114.0	34.4	0.0027
40%		45.9	87.2	111.8	32.9	0.0004
30%		42.6	83.8	108.9	36.7	0.0000
20%		42.3	84.3	111.1	38.6	0.0000
10%		43.2	86.6	112.8	40.8	0.0000

첫 번째 파라미터는 **'변이체 생성 대상 라인 선택 비율'**이다. 모든 코드 라인에 대해 변이 분석을 수행하는 것은 막대한 비용을 발생시키므로, 연산이 가벼운 SBFL(Ochiai) 의심도 점수를 기준으로 라인을 정렬한다. 이후 상위 $N\%$의 라인만을 변이체 생성 대상으로 선정함으로써 분석 범위를 전략적으로 제한하고 테스트 실행 횟수를 감축한다.

두 번째 파라미터는 **'라인당 변이체 생성 개수'**이다. 선정된 각 대상 라인에 대해 생성되는 변이체의 수를 조절하여 연산량을 제어한다. 기존 연구들이 임의의 개수를 설정했던 것과 달리, 본 연구는 변이체 개수를 단계적으로 축소하며 DLFL 모델이 유의미한 패턴을 학습하는 데 필요한 최소한의 변이체 수를 도출하고자 한다.

3.2 결함위치추정 정확도 향상 전략

데이터셋의 효율성뿐만 아니라 추정의 정밀도를 높이기 위해, 기존 DLFL 연구에서 사용되던 실행 경로 기반 특징(SBFL, MBFL) 외에 **본 연구에서 최초로 설계하여 제안하는 'ST(Stack Trace) 관련성 특징'**을 추가한다. 해당 로직은 본 연구에서 개발한 자동화 도구의 핵심 모듈로 구현되어 데이터셋 생성 시 자동으로 산출된다.

3.2.1 스택 트레이스(ST) 관련성 특징

본 연구는 딥러닝 모델의 학습 성능을 고도화하기 위해, **실제 개발자의 디버깅 직관을 수치화한 ST 관련성 특징을 세계 최초로 제안**한다.

실무 개발자들은 프로그램 비정상 종료 시 발생하는 스택 트레이스 정보를 가장 먼저 활용하며, 특히 최상단 프레임이 가리키는 코드 위치가 결함과 밀접할 것이라고 판단한다. 본 연구는 이러한 인간의 디버깅 휴리스틱을 딥러닝 모델이 학습 가능한 형태의 정량적 특징으로 설계하였다. 각 코드 라인의 ST 관련성 특징값은 실패 테스트 실행 시 발생하는 스택 트레이스 정보를 바탕으로 다음과 같이 계산된다.

$$STRelevance(s) = max_{f \in stackTrace(P,s)}\left(\frac{1}{position(f)+1} \times e^{-distance(s,f)^2}\right)$$

해당 공식의 주요 구성 요소는 다음과 같다.

- *stackTrace(P, s)*: 프로그램 P에서 발생한 스택 트레이스 중, 라인 s와 동일한 함수 내에 존재하는 프레임들의 집합
- *position(f)*: 해당 프레임의 위치 (최상단 프레임은 0이며, 아래로 갈수록 1씩 증가)
- *distance(s, f)*: 소스 코드 라인 s의 번호와 프레임 f가 가리키는 실제 라인 번호 간의 거리

해당 공식은 비정상 종료 지점(최상단 프레임)과의 거리(position)가 가까울수록, 그리고 프레임이 지칭하는 코드 위치와 물리적으로 인접할수록(distance) 높은 가중치를 부여하도록 설계되었다. 본 연구에서 최초로 제안한 **ST 관련성** 특징은 런타임 오류 문맥을 직접적으로 반영한다.

표 2: RQ2 '라인당 변이체 생성 개수' 실험 결과

Line Selection Ratio	#Mutants Per Line	Top-1	Top-3	Top-5	MFR	p-value
100%	10	52.7	98.3	123.4	29.9	1.0000
70%	10	50.7	92.5	117.4	31.4	0.0750
	9	49.8	93.6	119.4	31.3	0.1977
	8	49.1	93.6	118.8	31.3	0.1533
	7	48.8	94.0	117.6	30.5	0.1286
	6	50.9	93.2	119.4	31.2	0.1836
	5	49.7	92.1	116.4	31.2	0.0753
	4	50.6	94.5	118.6	31.3	0.1024
	3	48.6	93.1	118.3	31.0	0.0792
	2	48.8	92.3	116.0	32.0	0.0296
	1	44.6	88.9	113.2	32.8	0.0015

기존 연구들이 Stack Trace의 단순 텍스트 정보만을 활용했다면, 본 연구에서는 각 라인과 비정상 종료(crash) 간의 관련성을 함수 및 라인 번호를 기반으로 수치화하여 학습 특징으로 활용한다. 결과적으로 실행 경로와 문맥 기반 특징을 결합함으로써, 모델이 결함의 논리적 연관성을 더욱 정교하게 학습할 수 있도록 설계한 것이 본 연구의 주요 기여이다.

4. 탐색적 실험 설정

본 장에서는 제안하는 데이터셋 구축 최적화 방법론과 새롭게 설계된 ST 관련성 특징의 유효성을 검증하기 위한 탐색적 실험 설정을 기술한다. 본 실험의 목적은 가용 자원이 한정되고 복잡도가 높은 국방 소프트웨어에 DLFL 기술을 적용하기에 앞서, 통제된 벤치마크 환경에서 비용과 성능 사이의 최적 임계값을 도출하고 신규 특징의 기여도를 확인하는 데 있다. 본 실험은, MBFL 논문들에서 가장 많이 연구가 된 Defects4J v1.2.0의 5개 프로젝트에서 추출된 257개의 실제 결함 버전을 활용한다.

4.1 연구 질문 (Research Questions)

본 실험은 다음 세 가지 연구 질문에 답함으로써 효율적이고 정밀한 DLFL 모델 구축 가이드를 수립한다.

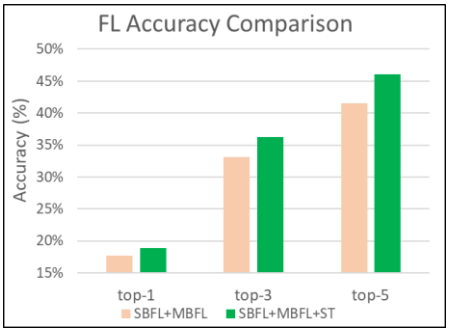
- **RQ1 (변이 라인 선택 비율):** SBFL(Ochiai) 의심도를 기반으로 변이 분석 대상 라인을 상위 ***N%*** (10%~100%)로 제한할 때, ***N***값의 변화가 DLFL 모델의 정확도와 데이터셋 구축 비용에 미치는 영향은 무엇인가?
- **RQ2 (라인당 변이체 개수):** 각 라인당 생성하는 변이체의 수 ***M*** (1~10개)을 조절할 때, ***M***값의 변화가 모델의 정확도와 구축 비용에 미치는 영향은 무엇인가?
- **RQ3 (스택 트레이스 관련성):** 본 연구에서 최초로 설계하여 제안한 스택 트레이스(ST) 관련성 특징을 학습 데이터셋에 추가하는 것이 DLFL 모델의 결함 위치 추정 정확도 향상에 어느 정도 기여하는가?

4.2 실험 모델 및 수행 방법

실험 모델은 최신 DLFL 연구인 CodeHealer[6]의 구조를 계승한 Multi-Layered Perceptron(MLP)을 사용한다. 해당 모델은 입력층의 노드 수를 특징(Feature)의 개수와 동일하게 설정하고, 하나의 은닉층(Hidden Layer)을 포함하도록 설계되었다.

실험 수행을 위해, 앞서 언급한 데이터셋 자동 추출 도구를 활용하여 각 결함 프로그램으로부터 SBFL, MBFL 및 ST 관련성 특징을 자동으로 산출하고 통합 데이터셋을 생성하였다. Java 프로그램에 변이 생성 도구로는 PITest를 활용한다. 결과의 신뢰성을 확보하고 특정 데이터셋에 대한 과적합(Overfitting)을 방지하기 위해 10-fold cross-validation을 적용하였으며, 변이 분석 및 학습 과정의 비결정적 요소를 고려하여 전체 과정을 총 10회 반복 수행한 후 평균값을 최종 결과로 사용하였다.

그림 1: RQ3 '스택 트레이스 관련성 특징' 실험 결과



4.3 평가 지표 및 통계적 검증

본 연구는 데이터셋 구축의 **시간 효율성**과 모델의 **결함위치추정 정확도**를 평가한다.

- **시간 효율성**: 데이터셋 구축부터 모델 학습 완료까지 소요되는 전체 시간(CPU-hours)을 측정하여 비용 절감 효과를 수치화한다.
- **Top-N**: 모델이 예측한 의심도 상위 ***N***개 라인 이내에 실제 결함 코드가 포함된 결함의 총 개수를 의미한다.
- **MFR (Mean First Rank)**: 개발자가 결함을 발견하기 위해 조사해야 하는 첫 번째 실제 결함 코드의 평균 순위이다. 수치가 작을수록 디버깅 효율이 높음을 의미한다.

또한, 서로 다른 설정 간의 성능 차이가 통계적으로 유의미한지 확인하기 위해 Mann-Whitney U Test를 수행한다. 본 연구에서는 유의 수준 0.05를 기준으로 각 설정 간의 성능 변화를 엄격히 검증하여, 성능 저하 없이 최대로 비용을 절감할 수 있는 최적의 파라미터 조합을 도출한다.

5. 탐색적 실험 결과

본 장에서는 4장에서 설정한 세 가지 연구 질문에 따라, 앞서 개발한 자동화 도구를 통해 수행된 탐색적 실험의 결과를 분석한다. 모든 실험 결과는 베이스라인(라인 선택 비율 100%, 라인당 변이체 개수 10개) 대비 성능 변화와 통계적 유의성을 기준으로 평가하였다.

5.1 RQ1: 변이체 생성 대상 라인 선택 비율 (Target Line Selection Ratio)

실험 결과, SBFL 의심도 상위 **70%의 라인만을 변이 분석 대상으로 선정**한 설정은 베이스라인과 비교하여 통계적으로 유의미한 성능 저하를 보이지 않았다(p-value = 0.0750). 반면, 표 1의 60% 이하 설정부터는 정확도가 급격히 하락하는 양상이 관찰되었다. 이를 통해 정확도 손실 없이 데이터셋 구축 시간 비용을 약 29.8% 절감할 수 있는 최적의 임계값이 상위 70%임을 확인하였다.

5.2 RQ2: 라인당 변이체 생성 개수 (Mutant Count Per Line)

표 2은 RQ1에서 도출된 70% 라인 선택 설정을 고정한 후, 변이체 개수(M)를 조절하며 측정한 결과이다. 데이터 분석 결과, **라인당 3개의 변이체**만으로도 베이스라인과 통계적으로 대등한 수준의 성능을 유지할 수 있음을 입증하였다(p-value = 0.0792). 최종적으로 표 3의 '라인 선택 비율 70%'와 '라인당 변이체 3개' 조합을 통해, 베이스라인 대비 전체 구축 시간을 **74.6% (198.2시간→50.4시간)** 단축하는 최적의 효율성을 확보하였으며, 이 가이드라인은 이후 국방 SW 적용의 기술적 근거가 된다.

5.3 RQ3: 스택 트레이스 관련성 특징의 효과

본 연구에서 설계하여 최초로 제안한 ST 관련성 특징의 성능 향상 기여도는 그림 1의 그래프를 통해 명확히 확인할

표 3: 국방 무기체계 소프트웨어
(FT: Failing Tests, PT: Passing Tests, AB: Artificial Bug)

Subjects	Language	#Artifical Buggy Program	Size (LoC)	Line Cov.	Avg. #FTs (on AB)	Avg. #PTs
System_A	C	50	18,854	57.6%	2.5	74.4
System_B	C	50	4,870	55.8%	1.4	11.6
System_C	C	50	6,217	66.9%	3.6	7.1
System_D	CPP	50	10,788	60.9%	3.6	3.2
System_E	CPP	50	8,333	68.4%	3.2	31.9
System_F	CPP	50	12,021	45.8%	5.4	20.9

수 있다. 기존 특징 조합(SBFL+MBFL)에 본 연구의 ST 특징을 추가하여 학습시킨 결과(SBFL+MBFL+ST), 모든 지표에서 성능이 향상되었다. 구체적으로 그림 1에 나타난 바와 같이 Top-1/Top-3/Top-5 정확도는 각각 6.8%, 9.3%, 11.0% 개선되었으며, MFR 지표 또한 14.4% (36.2→30.9) 향상되었다. 이는 실행 경로 중심의 기존 특징들과 본 연구에서 새롭게 제안한 에러 문맥 기반 ST 특징이 모델의 결함 식별력을 극대화했음을 보여준다.

6. 국방 무기체계 소프트웨어 적용

본 장에서는 Java 기반 데이터셋인 Defects4J를 통한 탐색적 실험으로 검증된 가이드라인과 ST 관련성 특징을 실제 C/C++ 기반 국방 무기체계 소프트웨어에 적용한 결과를 기술한다. Java와 C/C++ 간의 구문적 차이가 존재하나, 프로그램의 실행 경로를 변형하여 결함을 식별하는 MBFL의 핵심 기저 원리는 동일하다. 이러한 원리적 공통성으로 인해 Java 환경에서 도출된 최적 가이드라인은 C/C++ 분석에서도 그 유효성이 유지된다. 본 연구에서 개발한 자동화 도구는 L사의 실제 DLFL 운용 시스템에 성공적으로 통합되어 실무적 타당성과 실전 배치 가능성을 입증하였다.

6.1 대상 국방 무기체계 소프트웨어 및 실험 설정

본 사례 연구는 국내 유수의 방산 기업인 L사의 항공, 해양, 유도무기 등 실제 무기체계 시스템에 탑재되어 운용 중인 6종의 C/C++ 미들웨어 소프트웨어를 대상으로 수행되었다. 대상 시스템의 전체 코드 규모는 약 61 KLoC에 달하며, 이는 높은 신뢰성과 복잡한 로직을 요구하는 국방 소프트웨어의 특성을 고스란히 반영하고 있다.

국방 분야의 보안 규정상 과거 결함 데이터 및 실결함 로그에 대한 직접적인 접근이 제한됨에 따라, 본 연구에서는 소스 코드의 각 라인에 변이를 삽입하여 소프트웨어당 50개씩, 총 300개의 인공결함 프로그램을 구축하여 평가를 진행하였다. 표 3는 실험에 사용된 각 시스템의 상세 지표를 보여주며, 보안 유지를 위해 시스템 명칭은 익명화하여 기술한다.

6.2 사례 연구 결과 분석

탐색적 실험에서 도출된 최적 파라미터(라인 선택 비율 70%, 라인당 변이체 3개)와 본 연구의 독창적 기여인 ST 관련성 특징을 국방 SW 데이터셋에 적용한 결과, **자원 소모를 최소화하면서도 압도적인 결함 추정 성능**을 달성하였다.

첫째, 결함 위치 추정 정확도 측면에서 **Top-1 62.7%, Top-3 82.6%, 그리고 Top-5 기준 85.0%**라는 매우 높은 정확도를 기록하였다. 특히, 개발자가 버그를 찾기 위해 조사해야 하는 코드의 평균 순위를 의미하는 MFR 지표는 **18**로 나타났다. 이는 수만 라인의 거대한 코드 베이스 내에서 실무자가 상위 18줄 내외만 조사하더라도 실제 결함 위치를 특정할 수 있음을 입증하는 것이며, 실제 무기체계 개발 현장의 디버깅 노력을 획기적으로 절감할 수 있는 수치이다.

둘째, 데이터셋 구축의 비용 효율성 측면에서 괄목할 만한 성과를 거두었다. 기존의 보수적인 방식(전체 라인 대상, 다수의 변이체 생성)을 고수할 경우 막대한 계산 자원이 요구되지만, 본 연구의 최적화된 가이드라인을 적용한 결과 전체 데이터셋 구축 시간을 합계 약 **1,739 CPU-hours** 수준으로 억제하였다. 이는 이론적 베이스라인 대비 **약 79%의 시간 비용을 단축**한 결과이다. 이러한 극적인 비용 절감은 자원이 제한된 실무 환경에서 DLFL 모델을 주기적으로 갱신하고 실전 배치할 수 있는 강력한 실무적 타당성을 부여한다.

6.3 L사의 사내 DLFL 시스템 구축

본 사례 연구를 통해 확인된 성과는 이론적으로 제안된 방법론이 실제 방산 기업의 DLFL 시스템에 성공적으로 탑재되어 실질적인 가치를 창출했다는 점이다. 본 연구에서 약 6,000줄 규모로 개발된 도구는 자동으로 학습 데이터를 생성하고 정밀한 추론 결과를 도출하여, 국방 SW 개발 과정의 자동화를 앞당겼다.

결과적으로, L사 시스템에 탑재된 본 MBFL 기반 DLFL 데이터셋 자동 구축 도구는 기술적 진입 장벽을 낮추어 실무자가 별도의 딥러닝 및 자동 디버깅 기술에 대한 전문 지식 없이도 결함 위치 추정 기술을 현업에 적용할 수 있도록 하였다. 이는 국방 소프트웨어의 품질 보증 및 유지보수 단계에서 발생하는 디버깅 병목 현상을 해결할 수 있는 실질적인 기술적 토대를 마련한 것으로 평가된다.

7. 결론

본 논문은 국방 소프트웨어 분야의 딥러닝 기반 결함 위치 추정(DLFL) 도입을 위해, 변이 기반 DLFL 데이터셋 구축 과정을 최적화하는 체계적인 연구 방법론을 제시하였다. 실무적 기여 측면에서는 도출된 가이드라인과 ST 특징 추출 로직을 탑재한 자동화 도구를 **L사의 DLFL 시스템에 성공적으로 통합**하였다. 해당 도구는 데이터셋 구축부터 학습 및 추론까지의 전 과정을 자동화하여 사내 직원이 즉시 운용 가능한 실무 인프라를 제공한다. 탐색적 실험에서 도출한 최적의 파라미터와 ST 관련성 특징을 6개 국방 무기체계 C/C++ 소프트웨어에 적용한 결과, **Top-5 기준 85.0%**의 높은 정확도와 약 **79%**의 **구축 비용 단축**을 달성하여 그 유효성을 입증하였다.향후 연구로는 구축된 고품질 변이 데이터셋을 대규모 언어 모델(LLM)[8,9,10]의 의미론적 특징과 결합하여, 실행 분석과 언어적 이해가 통합된 차세대 결함 위치 추정 프레임워크를 개발할 계획이다.

참고 문헌

[1] S. Yoo. (2012). Evolving human competitive spectra-based fault localisation techniques. SSBSE 2012.
[2] S. Moon, Y. Kim, M. Kim and S. Yoo, "Ask the Mutants: Mutating Faulty Programs for Fault Localization," ICST 2014.
[3] M. Papadakis and Y.L. Traon. Metallaxis-FL: mutation-based fault localization. STVR 2015.
[4] X. Meng, X. Wang, H. Zhang, H. Sun, and X. Liu. Improving fault localization and program repair with deep semantic features and transferred knowledge. ICSE 2022.
[5] X. Wang, H. Yu, X. Meng, H. Cao, H. Zhang, H. Sun, X. Liu, and C. Hu. MTL-TRANSFER: Leveraging Multi-task Learning and Transferred Knowledge for Improving Fault Localization and Program Repair. TOSEM 2024.
[6] L. Zhang, S. Guo, Y. Guo, H. Li, Y. Chai, R. Chen, X. Li, and H. Jiang. Context-based Transfer Learning for Structuring Fault Localization and Program Repair Automation. TOSEM 2025.
[7] D. L. Phan, Y. Kim and M. Kim, "MUSIC: Mutation Analysis Tool with High Configurability and Extensibility," ICSTW 2018.
[8] S. Kang, G. An, and S. Yoo. A Quantitative and Qualitative Evaluation of LLM-Based Explainable Fault Localization. FSE 2024.
[9] C. Xu, Z. Liu, X. Ren, G. Zhang, M. Liang, and D. Lo. FlexFL: Flexible and Effective Fault Localization With Open-Source Large Language Models. TOSEM 2025.
[10] A.Z. H. Yang, C.L. Goues, R. Martins, and V. Hellendoorn. Large Language Models for Test-Free Fault Localization. ICSE 2024