2026 한국 소프트웨어공학 학술대회 (KCSE 2026)

대규모 언어 모델을 활용한 단위 테스트의 적합성 자동 식별

Automated Identification of Unit Test Adequacy Using Large Language Models

김대원, 이영규, 유준범

2026. 02. 05

Dependable Software Laboratory

KU 건국대학교 KONKUK UNIV.

Contents

1. 서론

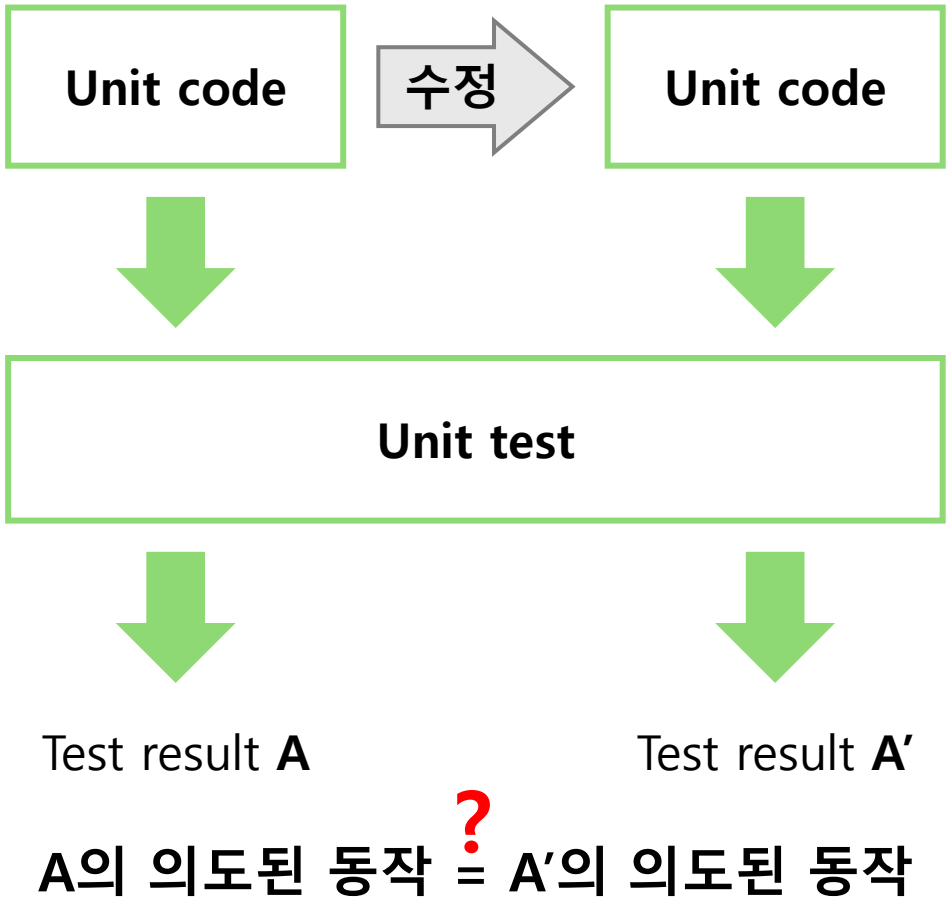
테스트 적합성

코드 수정과 검증의 불확실성

- 소프트웨어 개발 과정에서 코드 수정은 지속적으로 발생
- 코드가 수정된 후, 단위 테스트 결과만으로는 원래 의도된 동작과 동일한 동작을 한다는 것을 보장할 수 없음

테스트 적합성

- 단위 테스트가 검증하려던 코드의 의도된 동작이 코드 수정 후에도 유지되는 상태
- 코드 수정 후에도 테스트가 원래의 검증 목표를 달성할 수 있어야 함



연구의 목표

현재 적용되는 방법의 한계점

- 개발자 수작업에 의존하여 많은 시간과 노력 소요
- 자동화된 분석 도구 및 기법 부재

본 연구의 목표

- 코드 수정 후 테스트 적합성을 자동으로 식별하는 방법을 제안
- 대규모 언어 모델(LLM)을 활용한 자동 분석 -> 테스트 적합성 자동 식별

2. 자동 식별 방법

테스트 적합성 판단 기준

호출 함수가 동일한가?

- 테스트가 호출하는 함수/메서드 변경 여부
- 함수 오버로딩, 오버라이딩, 조건문으로 인한 변경 검토
- 호출 대상의 실질적 변경 파악

예상되는 입출력이 동일한가?

- 동일 입력에 대한 출력 유지 여부
- 구현 변경(루프→재귀) vs. 동작 변경 구분
- 결과의 논리적 동등성 확인

명세가 변경되었는가?

- 함수 기능의 실질적 변경 판단
- 리팩토링 vs. 기능 변경 구분
- 의도된 검증 목표 충족 여부 확인

코드 변경 분류

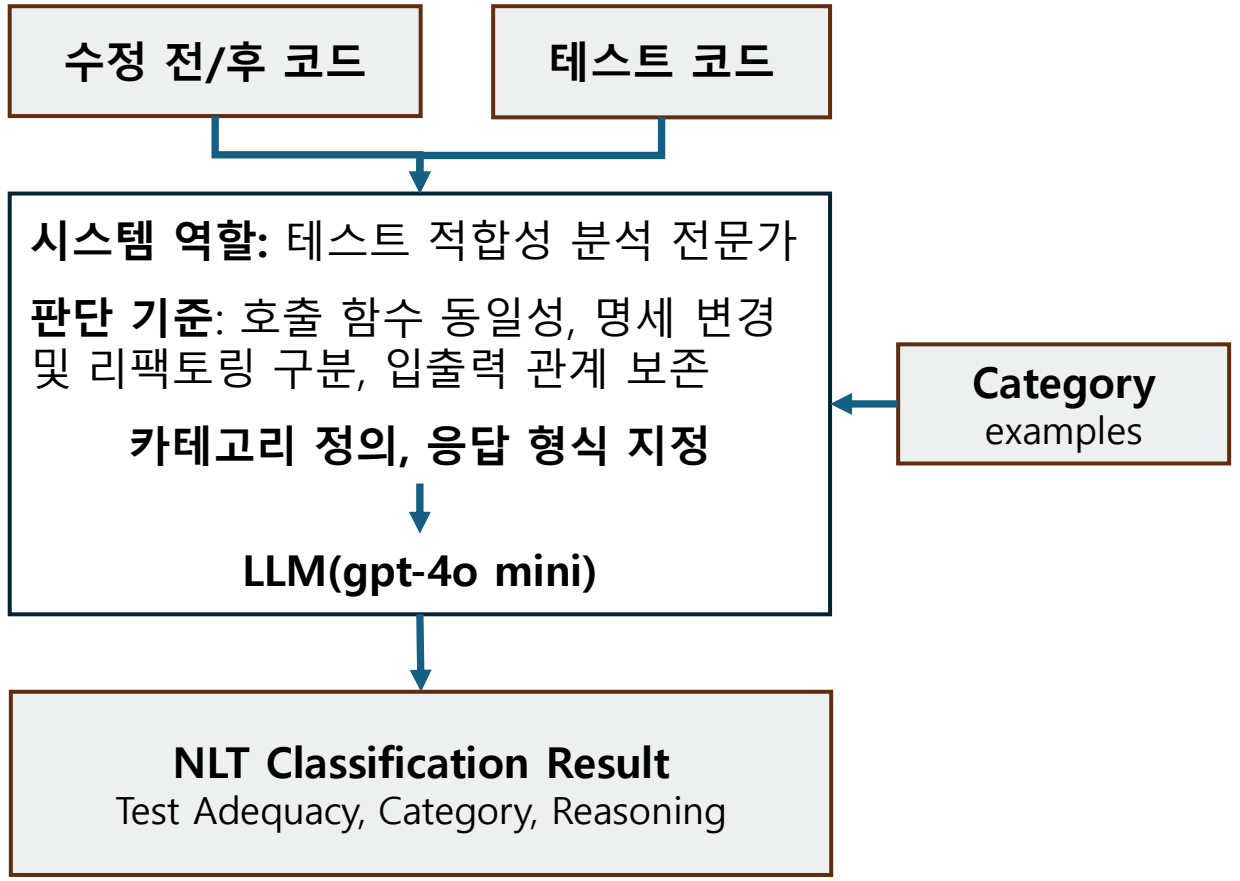
No-Longer-Testable

- 테스트를 여전히 통과하지만 더 이상 업데이트된 단위코드를 검증하지 못하는 단위 테스트
- 객체지향언어인 C++ 대상

Type	Category	Description
NLT	1) Overloaded Call	1) 테스트가 다른 오버로드된 메서드를 호출
	2) Overridden Call	2) 기존 호출에서 오버라이드된 메서드로 변경
	3) Binding Change	3) 가상 바인딩으로 인해 호출 대상 변경
	4) Hierarchy Shift	4) 상속 계층 구조의 변경
	5) Condition Change	5) 조건 로직으로 인해 호출 대상 변경
	6) Output Change	6) 동일 입력에 대해 다른 출력
	7) Type Mismatch	7) 입력 데이터 타입 변경
	8) Call Replacement	8) 다른 동작을 가진 메서드 호출 대체
Non-NLT	1) Variable Renaming	1) 동일 동작의 변수명 변경
	2) Method Renaming	2) 동일 동작의 메서드명 변경
	3) Helper Replacement	3) 동등한 헬퍼 메서드로 대체
	4) Instance Switch	4) 다른 인스턴스에 의한 동일 동작 실행
	5) Delegated Call	5) 다른 객체로 호출 위임
	6) Recursion Replacement	6) 반복문을 재귀 호출로 대체
	7) Order Change	7) 호출 순서 변경
	8) Mediator Adjustment	8) 중간 객체 변경으로 동작 영향 없음

LLM을 활용한 식별 과정

- 수정 전/후 코드와 테스트 코드를 입력
- 사전 정의된 카테고리를 활용하여 프롬프트를 구성
- 지정된 형식에 따라 답변하도록 정의



3. Case Study

실험 설계

실험 대상

- GoogleTest 샘플 10개 중 8개 선정
  (나머지 2개는 프레임워크 코드이므로 제외)
- 각 샘플에 16개 변경 유형 모두 적용 → 128개 테스트 케이스

실험 설정

- LLM: GPT-4o mini
- 평가 지표: 정확도 (테스트 적합성 식별 결과 + 변경 유형 분류 정확도)

프롬프트 구성

```
def get_system_prompt() -> str:
    return """You are an expert software testing analyst specializing in test validity assessment.
Your expertise includes:
- Understanding test intent and specification alignment
- Analyzing code changes and their semantic impact
- Evaluating runtime execution behavior changes
- Distinguishing between implementation changes and behavioral changes

Your task is to classify unit tests as either "NLT (No-Longer-Testable)" or
"Non-NLT" based on whether they still validate their intended behavior after code modifications.

OUTPUT FORMAT (strict):
Classification: [exactly "NLT" or "Non-NLT"]
Category: [exact category name from provided list]
Reasoning: [clear explanation with specific evidence]"""
```

테스트 적합성 식별을 위한 시스템 프롬프트

```
NON_NLT_CATEGORIES = {
    "Variable Renaming": {
        "description": "Variable names changed but the behavior and logic remain identical.",
        "example": "int count renamed to int total, same computation. ...
    }, ...
```

프롬프트에 사용되는 카테고리 형식

```
{  "classification": "Non-NLT",
   "category": "Recursion Replacement",
   "reasoning": "The original intent of the test was to validate the functionality of
   the Factorial function for negative, zero, and positive inputs. ...
}
```

LLM 응답 결과

System Prompt

- LLM의 역할 지정: 테스트 분석 전문가
- 테스트 의도와 코드 변경의 의미 분석, 동작 변화 평가 등의 역할을 명시
- 제시한 테스트 적합성 판단 기준에 따라 프롬프트를 구성

Categories Prompt

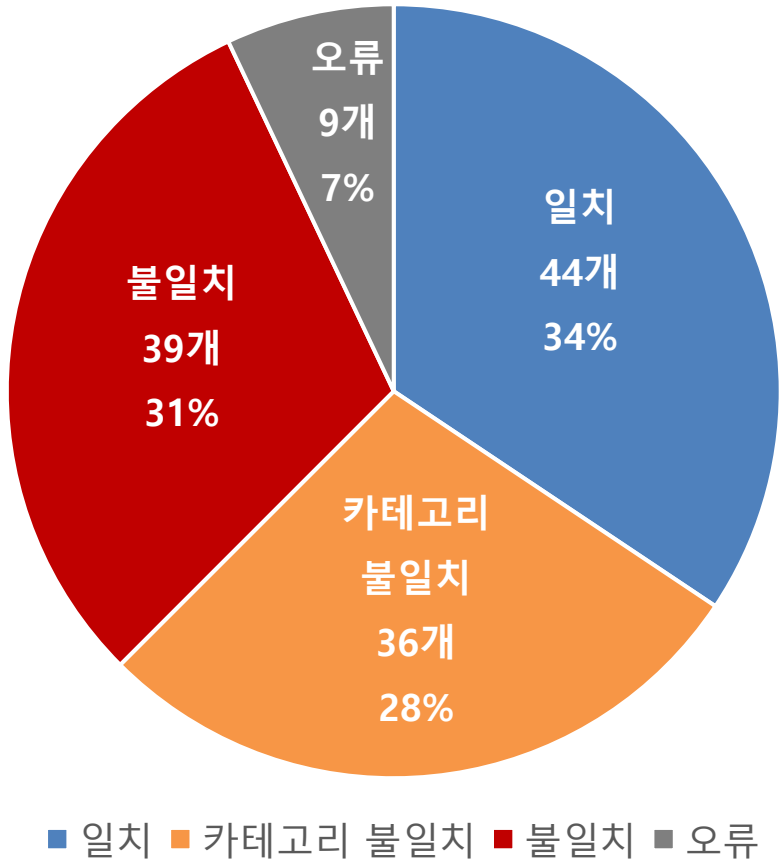
- 사전 정의된 16개의 카테고리를 프롬프트로 구조화

Response

- 테스트 적합성 유무 및 판단한 카테고리와 판단 근거를 응답하도록 지시

실험 결과

테스트 적합성 예측 결과



- **일치: 44개** (34%) - 적합성 및 카테고리 모두 정확
- **카테고리 불일치: 36개** (28%) - 적합성은 맞으나 세부 카테고리 오류
- **적합성 불일치: 39개** (31%) - NLT를 Non-NLT로 오판
- **카테고리 오류: 9개** (7%) - 주어진 16개 카테고리로 분류 불가

카테고리별 정확도

LLM 예측 결과

Type	Category	Accuracy
NLT	Overloaded Call	50%
	Overridden Call	50%
	Binding Change	25%
	Hierarchy Shift	37.5%
	Condition Change	0%
	Output Change	50%
	Type Mismatch	37.5%
	Call Replacement	50%
Non-NLT	Variable Renaming	12.5%
	Method Renaming	12.5%
	Helper Replacement	25%
	Instance Switch	37.5%
	Delegated Call	50%
	Recursion Replacement	50%
	Order Change	25%
	Mediator Adjustment	37.5%

실험 결과 분석

- 변경 유형이 직관적인 경우
  예) 반복문 → 재귀함수
- 변경 유형이 복잡한 경우
  예) 조건 변경으로 인한 분기 변경 발생
  → 코드 실행 정보 X ➔ 분기 변경 확인 X

4. 결론 및 향후 연구

결론 및 향후 연구

Conclusion

- 테스트 적합성을 LLM을 활용해서 자동 식별하기 위한 과정과 프롬프트 구성 제안
- Case study 결과, 명확한 변경 유형에 대해서는 비교적 잘 식별함
- 제안 방법의 정확도는 낮지만 LLM을 활용한 테스트 적합성 자동 식별의 가능성을 보임

Future Work

- 추가 정보 제공: 테스트를 실행시켜 실행 로그를 제공하는 등 효과적인 정보를 선정
- 프롬프트 최적화: 카테고리 판단 기준 구체화 및 명확한 예시 제공
- 복잡한 케이스 분석: 산업 프로젝트의 복잡한 테스트 케이스에 대한 실용성 평가

감사합니다

Q&A: kimdw010130@konkuk.ac.kr