AI 소프트웨어 개발을 위한 양방향 수렴 기반 중간 산출물 복원을 통한 요구사항 추적성

(Requirement Traceability based on Bidirectional Convergence of Intermediate Artifact Recovery for AI Software Development)

이진협*, 김장환, 서채연**, 전병국***, 김영철****

(Jinhyup Lee, Janghwan Kim, Chaeyun Seo, Byungkook Jeon, R. Young Chul Kim)

요 약

최근 대규모 언어 모델을 이용해 요구사항으로부터 코드를 직접 생성하는 개발 방식이 확산되고 있다. 이 방식은 개발 속도를 높이지만, 요구사항과 코드 사이의 중간 산출물을 생성하지 않는다. 아직도 명확한 사항 요구산출물이 정의되지 않았고, 특히 설계 단계에서는 유스케이스, 유스케이스 시나리오, 객체가 부재하여 요구사항 추적이 불가능하다. 현재 이슈인 중간 산출물 부재 문제를 해결하기 위해, 대규모 언어 모델을 적용한 설계 복원 및 추적 메커니즘을 제안한다. 제안 방안은 요구사항만으로 유스케이스를 생성하는 순방향 복원과 코드만으로 유스케이스를 복원하는 역방향 복원을 정보 차단 상태에서 독립 수행한 후, 두 결과를 대조하여 수렴시키는 양방향 복원이다. 실험 결과, 양방향 복원은 역방향 복원 대비 유스케이스 복원 커버리지를 65.5%에서 75.1%로 9.6%p 향상시켰으며, 추적 링크 재현율 또한 향상되었다.

■ 중심어 : 요구사항 추적성 ; 중간 산출물 복원 ; 양방향 수렴 ; 대형 언어 모델 ; 유스케이스

Abstract

With the recent shift toward development methods utilizing generative AI and AI agents, the software industry is experiencing accelerated development. However, this approach does not generate any intermediate artifacts—such as use cases, use case scenarios, and object artifacts—between requirements and source code, resulting in a critical problem where requirement traceability cannot be established. To address this issue, we propose a technique that utilizes Large Language Models (LLMs) to recover these missing intermediate artifacts and assign trace links during the recovery process. The proposed method performs forward recovery, which generates use cases solely from requirements, and backward recovery, which recovers use cases solely from source code. This bidirectional convergence approach increased use-case recovery coverage from 65.5% to 75.1%, a gain of 9.6 percentage points, and improved trace-link recall.

■ keywords : Requirement Traceability ; Intermediate Artifact Recovery ; Bidirectional Convergence ; Large Language Model ; Use Case

Ⅰ. 서 론

최근 소프트웨어 개발 분야에서는 생성형 AI의 발전과 대형 언어 모델(LLM) 기반의 AI 에이전트(AI Agent)를 이용한 개발 방법론이 널리 확산되며 개발 패러다임이 변화하고 있다. 개발자가 자연어 프롬프트로 요구사항을 기술하면 언어 모델이 즉각적으로 실행 가능한 코드를 생성하는 이른바 바이브 코딩(vibe coding) 방식은 소프트웨어 개발 속도를 비약적으로 높이고 비

* 정회원, 홍익대학교 소프트웨어융합학과
** 정회원, 홍익대학교 소프트웨어융합학과
*** 정회원, 국립 강원대학교 컴퓨터 공학과

접수일자 : 2026년 07월 30일
수정일자 : 2026년 08월 20일
게재확정일 : 2026년 08월 21일
교신저자 : 김영철 e-mail : bob@hongik.ac.kr

전문가의 접근성을 크게 향상시켰다[1].

그러나 이러한 프롬프트 기반의 개발 방식은 전통적인 소프트웨어 공학이 축적해 온 단계별 산출물 체계를 생략한다는 치명적인 한계를 지닌다. 기존 프로세스에서는 요구사항으로부터 유스케이스, 유스케이스 시나리오, 객체, 메소드에 이르는 중간 산출물을 단계적으로 생성하며, 이를 바탕으로 인접 단계 간 추적성을 확보해 왔다[2]. 반면, AI 에이전트를 활용한 방식에서는 요구사항과 최종 코드만 존재할 뿐 그 사이의 중간 생성물이 부재하여 추적 링크를 연결할 대상 자체가 사라지게 된다. 요구사항 추적성이 단절되면 요구사항 변경 시 영향을 받는 요소를 식별하기 어렵고, 개발된 코드가 요구사항을 충족하는지 검증할 수 없으며, 결함 발생 시 원인 단계를 추적하거나 인증 및 감사를 위한 근거 제시가 불가능해지는 심각한 문제가 발생한다.

본 논문은 이러한 프롬프트 기반 개발 환경에서의 중간 산출물 부재 문제를 인식하고, 언어모델을 활용하여 누락된 중간 산출물을 복원하는 동시에 추적 링크를 부여하는 방안을 제안한다. 구체적으로 요구사항과 코드만 존재하는 상황에서 유스케이스를 복원하기 위해, 요구사항만을 입력으로 유스케이스를 생성하는 순방향 복원과 코드만을 입력으로 유스케이스와 링크를 복원하는 역방향 복원을 서로 입력을 차단한 상태에서 독립적으로 수행한다. 이후 두 결과를 대조하고 통합하는 양방향 복원 방식을 적용하여 복원의 신뢰성을 높인다.

기존 연구들에서는 산출물 간의 단계적 추적 단위를 정의하고 추적성 매트릭스를 구현한 바 있으나, 링크 생성을 위해 사용자가 직접 개입해야 하는 수동적인 한계가 존재했다[3]. 제안하는 기법을 적용하면 이러한 수동적인 개입이나 사용자의 직접적인 매트릭스 작성 없이도 자동화된 방식으로 누락된 중간 산출물과 추적 링크를 생성할 수 있다. 이를 통해 AI 기반 개발 환경에서 누락된 중간 산출물을 복원하고 요구사항 식별자와 코드 링크를 자동으로 연결함으로써 요구사항 추적성 확보를 지원할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 요구사항 추적성 관련 연구를 고찰한다. 제3장에서는 부재한 산출물과 추적 링크를 복원하기 위해 제안하는 양방향 수렴 기반 복원 방안을 상세히 설명한다. 이어서 제4장에서는 제안 방안의 적용 사례 및 실험 결과를 분석하며, 마지막으로 제5장에서는 결론 및 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

2.1. 기존 요구사항 추적성 모델 및 수동적 통제의 한계

요구사항 추적성은 산출물 내 저장된 데이터와 이들의 관계를 연결하여 다른 산출물로 찾아갈 수 있는 연결고리를 생성하는 능력을 의미한다. Gotel과 Finkelstein은 이를 요구사항의 생애를 순방향과 역방향 모두에서 기술하고 추적하는 능력으로 정의하였으며[4], 이후 Ramesh와 Jarke는 실무 사례 분석을 통해 조직의 성숙도에 따라 요구되는 추적 링크의 유형이 달라짐을 보였다[5]. 이를 바탕으로 Janghwan Kim과 R. Young Chul Kim은 객체지향 개발 환경에서의 추적 단위를 요구사항(REQ), 유스케이스(UC), 유스케이스 시나리오(US), 객체(OB), 메소드(MD)로 세분화하여 정의한 바 있다[2]. 또한 Jinhyub Lee는 이러한 단계 정의에 따라 추적성 매트릭스를 구현하였으나, 추적 링크의 생성이 사용자가 매트릭스의 셀을 직접 선택해야 하는 수동 방식에 머무른다는 한계를 지적했다[3]. 더불어 Wibowo와 Davis는 기존 온톨로지가 유연한 추적 단위 크기를 지원하지 못하는 문제를 짚으며, 프로젝트 특성에 따라 추적 단위를 유연하게 적용할 수 있는 RTOnto 모델을 제안하였다[6]. 하지만 이러한 선행 연구들은 모두 프로젝트 착수 시점에 사람이 추적 단위의 크기를 결정하고 수동으로 매트릭스를 통제할 수 있다는 점을

전제로 한다. 반면, 자연어로 요구사항을 기술하면 코드가 즉각 생성되는 바이브 코딩(vibe coding)[1] 환경에서는 산출물을 추적하는 도구의 부재가 아니라 중간 산출물 자체가 생성되지 않는다는 근본적인 문제가 존재한다. 언어모델이 산출물을 생성하는 환경에서는 사람이 사전에 추적 단위의 크기를 엄격하게 통제할 수 없으므로, 본 논문은 수동적인 매트릭스 작성이나 사람의 개입 없이 누락 된 중간 산출물을 자동으로 복원하고 그 과정에서 추적 단위를 유연하게 융합하는 새로운 자동화 접근법을 제시한다는 점에서 기존 연구와 뚜렷한 차별성을 가진다.

2.2 정형 모델 기반 자동화 및 단일 문맥 추론의 한계

추적 링크 생성 과정을 자동화하려는 시도도 지속적으로 이루어졌다. Yoshino와 Matsuura는 기존 도구들이 산출물 매핑을 사용자에게 맡겨 문서 해석의 오류가 발생한다는 점을 지적하며, 유스케이스 기술을 액티비티 다이어그램으로 형식화한 뒤 이를 변환 규칙에 따라 시퀀스 다이어그램으로 자동 생성하여 추적성을 확보하는 방안을 제안하였다[7]. 한편, Unterkalmsteiner는 하위 산출물이 아직 존재하지 않는 프로젝트 초기 시점의 추적성을 다루기 위해 요구사항에서 추출한 명사와 도메인 특화 분류체계를 활용하는 추천 시스템을 제안하였다[8]. 그러나 실무자를 대상으로 한 실험 결과, 단일 명사만을 고려하고 요구사항의 전체 문맥을 반영하지 못하는 추천 시스템은 오히려 사람의 수동 검색보다 정확성이 유의하게 떨어지는 한계를 보였다[8]. 본 논문이 제안하는 방식은 산출물 생성 시점에 추적 링크를 부여한다는 점에서 Yoshino와 Matsuura의 연구와 원리를 같이하지만[7], 정형화된 UML 모델의 존재를 필수적으로 전제하는 결정론적 변환 규칙과 달리, 어떠한 모델도 존재하지 않는 자연어 요구사항과 소스 코드만을 바탕으로 복원을 수행한다는 점에서 차별화된다. 또한, 단순 명사 기반 자동 추천의 실패 원인이 '문맥 미고려'에 있다는 Unterkalmsteiner의 통찰을 반영하여[8], 요구사항 기반 순방향 복원과 코드 기반 역방향 복원을 독립적으로 수행한 후 두 결과의 문맥을 병합 단계에서 교차 활용하는 양방향 복원 구조를 채택하였다. 이를 통해 기능 누락을 줄이고 복원 커버리지와 재현율을 향상시키는 것을 목표로 한다.

2.3 대형 언어 모델(LLM)을 활용한 단방향 복원의 한계

최근에는 사람이나 규칙 기반 도구를 넘어 언어 모델을 직접 활용해 산출물을 복원하거나 정합성을 판정하려는 연구가 등장하고 있다. Xiao 등은 소스 코드로부터 유스케이스를 복원하는 성능을 평가하는 UCRBench를 제안하였으나, 언어 모델이 코드를 기반으로 시스템 기능을 재구성할 때 누락률이 높고 일관된 추상화 수준을 유지하지 못한다는 심각한 한계를 지적하였다[9]. Jin과 Chen 역시 언어 모델에 코드와 자연어 명세를 함께 제시하고 충족 여부를 판정하게 한 결과, 올바른 구현을 불충족이나 결함으로 오분류하는 사례가 다수 발생함을 확인하며 모델의 자체 판정을 온전히 신뢰할 수 없음을 입증하였다[10].

이러한 선행 연구의 결과는 코드만을 입력으로 사용하는 '역방향 복원' 방식 자체에 내재된 한계를 보여준다. 코드만으로는 어떤 기능이 본래 존재해야 했는지에 대한 기준이 없으므로 모델 스스로 기능 누락을 인지할 수 없으며, 추상화의 수준 역시 모델의 자의적 판단에 좌우된다.

표 1. 기존 연구와 제안하는 양방향 수렴 복원 방식비교

구분	기존 선행 연구	양방향 수렴 복원
대상환경 및 전제조건	중간 산출물 존재 전제, 수동 개입 및 추적 단위 사전 설정 필요 [2, 3, 6]	중간 산출물 부재 환경 대상, 사전 개입 없이 산출물 자동 복원
입력데이터 및 문맥활용	정형 모델(UML 등) 요구, 단일 명사 기반 처리로 전체 문맥 반영 한계 [7, 8]	정형 모델 없이 자연어 요구사항과 소스 코드 활용, 양측의 문맥 교차 반영
복원 방식 및 신뢰성	단방향(역방향) 복원에 의존하여 기능 누락 발생 및 추상화 수준 불일치 [9, 10]	순·역방향 독립 수행 후 양방향 수렴, 기능 누락 상호 보완 및 복원 범위 확대

본 논문은 이러한 단방향 및 역방향 복원의 한계를 완화하기 위해 요구사항만을 바탕으로 유스케이스를 생성하는 순방향 복원을 독립적으로 수행하고, 두 결과를 대조 및 병합하는 양방향 복원 기법을 제안한다. 이를 통해 단방향 복원에서 발생하는 기능 누락을 상호 보완하고 복원 범위를 확대한다.

Ⅲ. 양방향 수렴 기반 중간 산출물 복원 메커니즘

3.1 단방향 복원(순방향 및 역방향) 분석

중간 산출물이 부재한 상황에서 이를 복원하는 방향은 크게 요구사항으로부터 하위 산출물을 생성하는 순방향 복원과, 소스 코드로부터 상위 산출물을 복원하는 역방향 복원으로 나뉜다.

순방향 복원은 요구사항만을 입력으로 유스케이스를 생성하므로 요구사항 식별자를 그대로 계승하고, 요구사항에 기술된 기능이 빠짐없이 표현된다는 장점이 있다. 그러나 생성된 유스케이스가 실제 구현과 무관하여 코드까지의 추적 링크를 생성할 수 없다는 결정적인 한계가 존재한다.

반면 역방향 복원은 코드만을 입력으로 복원을 수행하므로 복원된 유스케이스가 실제 구현에 근거하며 코드 간 링크가 생성된다. 하지만 복원된 결과가 어느 요구사항에 대응하는지 알 수 없어 사람의 수동 개입이 필요하며, 예외 처리나 오류 상황처럼 코드에 명시적으로 드러나지 않는 암묵적 기능은 복원되지 않는 단점이 있다. 표 2는 이러한 두 단방향 복원 방식의 장단점을 명확하게 비교한 것이다.

표 2. 순방향 복원과 역방향 복원 비교 요약

구분	순방향 복원	역방향 복원
입력	요구사항	코드
요구사항대응	자동	불가, 수동개입필요
코드링크	생성불가	생성
구현근거	없음	있음
추적단위크기 통제	요구사항이고정	모델에의존
누락유형	없음	암묵적기능

3.2 양방향 수렴 복원 모델 설계

본 논문은 순방향과 역방향 복원의 단점을 상호 보완하기 위해 두 결과를 결합하는 양방향 수렴 복원 기법을 제안한다. 이 기법의 핵심 설계 원칙은 '정보 차단(Isolation)'이다. 그림 1에서 볼 수 있듯이, 두 복원을 순차적으로 수행할 경우 발생할 수 있는 순환 논리를 방지하기 위해 순방향 복원 에이전트는 코드에 접근할 수 없고, 역방향 복원 에이전트는 요구사항에 접근할 수 없도록 완전히 독립된 환경에서 작업을 수행한다. 이후 병합 단계에서는 두 결과에서 동일한 행위를 기술하는 유스케이스를 상호 대응시킨다. 대응된 쌍에 대해 요구사항 식별자는 순방향 결과에서, 클래스 링크는 역방향 결과에서 취합하여 하나의 통합 유스케이스를 생성한다. 이 과정을 통해 추적 링크는 별도로 탐색할 필요 없이 병합의 자연스러운 부산물로서 자동 구성된다.

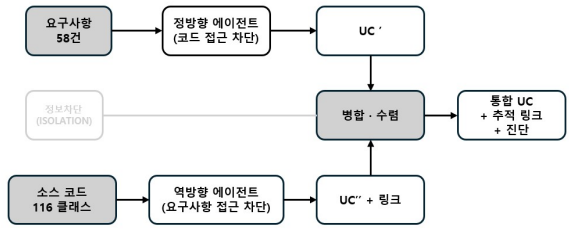

그림 1. 양방향 수렴 기반 중간 산출물 복원 구조

양방향 수렴 단계에서 두 유스케이스의 대응 여부는 단순한 명칭 일치가 아니라 기능적 의미를 기준으로 판단한다. LLM은 각 유스케이스의 행위(Action), 대상(Object/Target), 수행 목적(Goal), 그리고 행위자 및 실행 문맥을 비교하여 두 결과를 Match, Semantic Match, 또는 Unmatched로 분류하도록 구성하였다. 명칭과 기능이 모두 일치하는 경우는 Match로 판정하고, 명칭이 다르더라도 핵심 행위와 대상 및 수행 목적이 의미적으로 동일한 경우에는 Semantic Match로 판정한다. 반대로 동일한 명칭을 사용하더라도 행위자 또는 기능 문맥이 서로 다른 경우에는 별개의 유스케이스로 유지한다. Match 또는 Semantic Match로 판정된 경우 순방향 결과의 요구사항 식별자와 역방향 결과의 클래스 링크를 결합하여 하나의 통합 유스케이스를 생성하며, 대응되지 않은 항목은 강제 병합하지 않고 독립적으로 유지한다.

요구사항과 소스 코드를 동시에 하나의 LLM에 제공하여 유스케이스와 추적 링크를 직접 생성하는 방식도 고려할 수 있다. 그러나 이 방식은 두 정보가 동일한 문맥에서 동시에 사용되므로 각 결과가 어느 입력 근거로부터 도출되었는지를 분리하기 어렵다. 반면 제안 방법은 요구사항과 코드의 정보를 차단한 상태에서 순방향 및 역방향 복원을 독립적으로 수행한 후 두 결과를 비교·병합한다. 따라서 제안 방법의 핵심은 두 입력을 단순히 함께 제공하는 것이 아니라, 서로 독립적으로 생성된 두 복원 결과를 상호 보완적으로 수렴시키는 데 있다.

IV. 양방향 수렴 기반 중간 산출물 복원 적용 사례

4.1 실험 데이터셋 및 환경 구축

표 3. eTOUR 데이터셋구성

항목	내용
시스템	eTour, 관광안내시스템
원천산출물	유스케이스기술서58건
대상산출물	Java 클래스116개
정답추적링크	308개
유스케이스당 평균링크수	5.40개(중앙값5, 최대12)
출처	TEFSE challenge, 6th International Workshop on Traceability in Emerging Forms of Software Engineering, 2011

제안하는 양방향 수렴 복원 방안을 검증하기 위해 요구사항 추적성 분야의 벤치마크인 CoEST의 eTOUR 데이터셋을 활용하였다. 표 3은 해당 데이터셋의 세부 구성을 보여준다. 원천 산출물인 유스케이스 기술서 58건과 대상 산출물인 Java 클래스 116개, 정답 추적 링크 308개로 구성되어 있어 본 연구가 다루는 복원 구간과 정확히 일치한다.

데이터 전처리 과정에서 정답 링크가 없는 유스케이스 1건(UC37)은 평가에서 제외하였다. 또한, 표 4와 같이 시스템상 동일 기능이나 대행사와 업소 운영자 관점으로 명칭이 분리된 4건의 중복 유스케이스 쌍에 대해서는 정답 링크 집합이 서로 다름을 고려하여 1:N 대응을 예외적으로 허용하였다.

표 4. eTOUR 데이터셋의 명칭 중복 유스케이스

명칭	유스케이스	정답링크수
MODIFICA BANNER	UC17 / UC39	12 / 10
ELIMINAB ANNER	UC19 / UC40	9 / 7
INSERISCI BANNER	UC20 / UC38	12 / 10
MODIFICA COMMENTO	UC26 / UC54	2 / 8

4.2 실험 설계 및 파이프라인

표 5. 실험에 사용한 언어모델

개발사	모델	추론 수준
Anthropic	Claude Opus 5	높음
OpenAI	GPT-5.6 Sol	높음
Google	Gemini 3.6 Flash	기본

실험은 순방향, 역방향, 양방향 복원의 세 가지 조건에서 진행되었으며, 표 5와 같이 추론 능력이 뛰어난 3개사의 대형 언어 모델(LLM)을 선정한다.

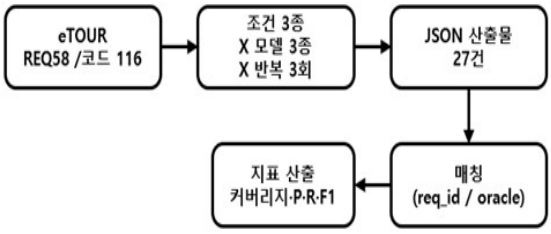

그림 2. eTOUR 데이터셋을 활용한 실험 파이프라인

그림 2는 연구의 전체 실험 흐름을 가시화한 파이프라인 도식이다. 그림에 나타난 바와 같이, eTOUR 데이터셋의 요구사항 58건과 116개의 소스 코드를 입력으로 하여, 3종의 모델과 3가지 조건을 결합한 총 27회의 세션을 실행한다. 모델의 과거 기억이나 웹 검색 등 외부 변수를 차단한 독립 환경에서 생성된 JSON 산출물들은 이후 정답 데이터(Oracle)와 매칭되어 최종적으로 커버리지, 정밀도, 재현율, F1 지표를 산출하는 과정으로 이어진다.

순방향 복원은 요구사항만을 입력으로 사용하므로 코드와의 추적 링크를 생성할 수 없다. 따라서 코드 링크를 기준으로 산출되는 Precision, Recall 및 F1의 정량 비교에서는 제외하였으며, 추적 성능 비교는 역방향 복원과 양방향 복원을 중심으로 수행하였다.

본 연구에서 유스케이스 복원 커버리지는 정답 유스케이스 집합 중 복원 결과에서 대응되는 유스케이스가 식별된 비율로 정의하였다. 데이터 전처리 과정에서 제외한 UC37을 제외한 총 57개의 유스케이스를 평가 대상으로 사용하였다. 따라서 복원 커버리지는 수식 (1)처럼 계산하였으며, 여기서 Nmatched는 복원 결과와 기능적으로 대응된 정답 유스케이스 수이고 Noracle=57이다. 각 조건별 결과는 3회 반복 실험의 평균과 표준편차로 제시하였다.

$$Coverage = Nmatched / Noracle \times 100 \quad (1)$$

4.3 유스케이스 복원 커버리지 분석

표 6. 유스케이스 복원 커버리지(%, 평균±표준편차)

언어모델	역방향행위	역방향추적	양방향	행위대비 차이
Claude	69.5 ± 2.6	60.3 ± 1.7	82.2 ± 5.0	+12.7
GPT	68.4 ± 5.3	54.6 ± 3.6	70.1 ± 4.0	+1.7
Gemini	58.6 ± 3.4	16.7 ± 1.0	73.0 ± 9.8	+14.4
전체(9회)	65.5 ± 6.2	43.9 ± 20.7	75.1 ± 8.0	+9.6

양방향 복원 기법은 역방향 복원 대비 유스케이스 복원 커버리지를 유의미하게 향상시켰다. 표 6에 명시된 바와 같이 3개 모델 전체 평균을 기준으로 역방향 복원의 커버리지는 65.5%였으나, 양방향 복원을 거치며 75.1%로 9.6%p 상승하였다 (t-검정 p=0.0051).

그림 3은 3종의 언어 모델별로 역방향 복원과 양방향 복원의 유스케이스 커버리지(위측) 및 추적 링크 F1 점수(아래측)를 오차막대와 함께 비교한 막대그래프이다. 위쪽 그래프를 보면 Claude, GPT, Gemini 모든 모델에서 짙은 색으로 표시된 양방향 복원의 막대가 역방향 복원보다 확연히 높게 형성되어 있어, 양방향 수렴을 통한 복원율 향상 효과가 모든 모델에서 일관되게 나타남을 확인할 수 있다.

반면 아래쪽의 F1 점수 그래프에서는 두 조건 간의 막대 높이 차이가 거의 없거나 오차 범위 내에 겹쳐 있어, 복원 커버리지 상승이 전체적인 F1 점수의 상승으로 직결되지는 않음을 시각적으로 보여준다.

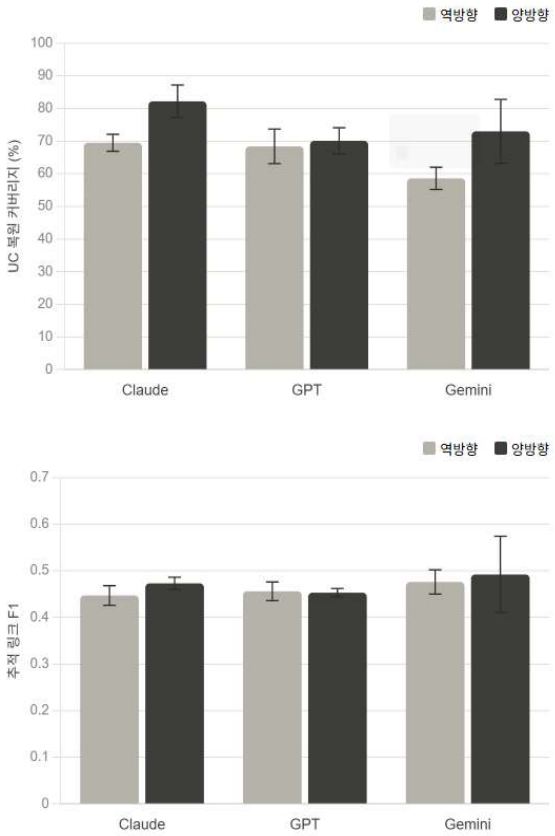

그림 3. 복원 커버리지 및 추적 링크 F1 점수 비교

4.4 추적 링크 정확도 및 과잉 링크 현상

추적 링크 정확도에서 F1 점수가 크게 오르지 않은 원인은 정밀도와 재현율 간의 상충 관계에 있다. 표 7에서 양방향 복원은 역방향 대비 재현율(Recall)을 평균 0.548에서 0.620으로 높였으나, 정밀도(Precision)는 0.398에서 0.385로 소폭 하락하였다.

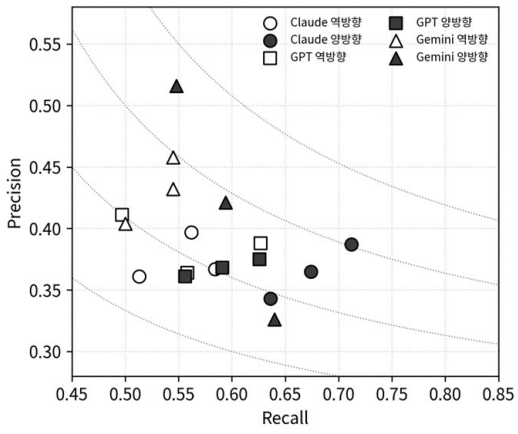

그림 4. 조건별 추적 링크의 정밀도-재현율 산점도

표 7. 추적 링크 정확도(평균±표준편차)

언어모델	조건	정밀도	재현율	F1
Claude	역방향	0.375 ± 0.019	0.553 ± 0.036	0.447 ± 0.021
Claude	양방향	0.365 ± 0.021	0.674 ± 0.038	0.473 ± 0.013
GPT	역방향	0.387 ± 0.024	0.561 ± 0.065	0.456 ± 0.020
GPT	양방향	0.368 ± 0.007	0.591 ± 0.034	0.453 ± 0.009
Gemini	역방향	0.431 ± 0.027	0.530 ± 0.026	0.476 ± 0.026
Gemini	양방향	0.421 ± 0.095	0.594 ± 0.054	0.492 ± 0.082
전체	역방향	0.398 ± 0.033	0.548 ± 0.042	0.460 ± 0.023
전체	양방향	0.385 ± 0.056	0.620 ± 0.055	0.473 ± 0.045

그림 4는 각 모델의 조건별 추적 링크 평가 결과를 정밀도와 재현율을 두 축으로 하여 나타낸 산점도이다. 점선의 등고선은 동일한 F1 점수 구간을 의미한다. 그래프를 살펴보면, 역방향 복원을 의미하는 흰색 도형들보다 양방향 복원을 나타내는 검은색 도형들이 전체적으로 우측(재현율 상승 방향)으로 이동해 있음을 명확히 확인할 수 있다. 그러나 동시에 검은색 도형들이 아래쪽(정밀도 하락 방향)으로도 이동하면서 결국 기존의 F1 등고선 흐름을 크게 벗어나지 못하는 양상을 보인다. 이는 양방향 복원이 기능 누락을 줄이는 데는 성공했으나, 정밀도 하락이 동반되어 종합 효율은 유사한 수준에 머무름을 방증한다.

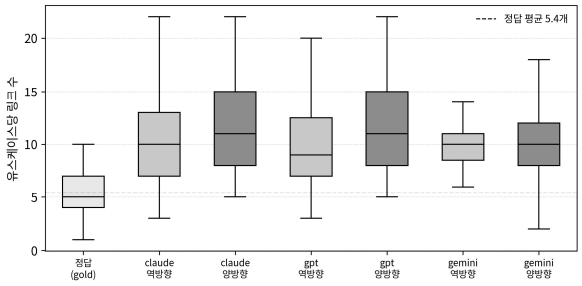

그림 5. 언어모델 및 조건별 유스케이스당 추적 링크 분포

정밀도 하락의 핵심 원인은 LLM이 관련성이 있어 보이는 클래스를 과도하게 연결하려는 '과잉 링크(Over-linking)' 현상이다. 그림 5는 이를 보여준다.

그림 5는 단일 유스케이스당 생성된 추적 링크 수의 분포를 나타낸 박스플롯이다. 하단의 점선으로 표시된 정답 데이터의 평균 링크 수는 5.4개 부근에 좁게 밀집해 있다. 그러나 Claude, GPT, Gemini가 생성한 복원 결과의 박스들은 조건과 무관하게 모두 정답 점선보다 2배가량 높은 위치(평균 10~12개)에 형성되어 있으며, 최대 값의 꼬리 역시 매우 길게 뻗어 있다. 특히 양방향 복원 조건의 박스들이 역방향 복원보다 높은 중앙값을 가지는 현상은 병합 과정에서 양측의 잉여 링크들이 취합되어 과잉 링크가 심화되었음을 가시적으로 증명한다.

4.5 추적 단위 크기(Granularity) 최적화

LLM 자동 복원 시스템은 생성되는 산출물의 '단위 크기'를 사람이 사전 통제할 수 없다는 문제점을 지닌다. 그림 6은 3종의 언어 모델이 역방향 복원 시 정답(원본 요구사항) 단위 대비 얼마나 포괄적인 크기로 유스케이스를 묶어서 생성했는지를 산출한 '추적 단위 크기 비율' 막대 그래프이다.

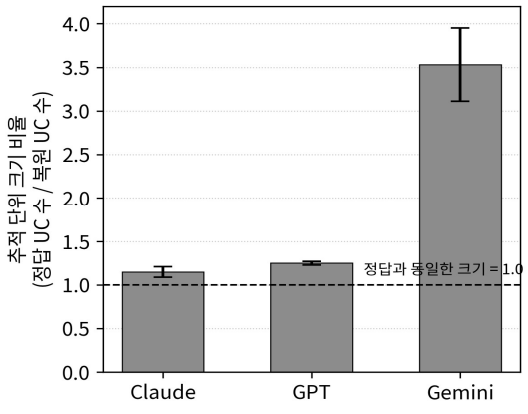

그림 6. 언어모델별 추적 단위 크기 비율

점선(1.0)은 정답과 완벽히 동일한 추상화 수준을 의미한다. 도표를 보면 Claude와 GPT 모델의 막대는 점선 부근(1.15~1.25)에 위치하여 비교적 정상적인 크기를 유지하고 있으나, 우측의 Gemini 모델 막대는 3.5를 넘어서며 비정상적으로 높게 솟아 있다. 이는 Gemini 모델이 시스템의 세부 기능들을 여러 개씩 통폐합하여 단 11~12개의 거대한 덩어리로만 복원하는 등 추상화 수준을 일관되게 유지하지 못했음을 시각적으로 증명한다. 그러나 양방향 복원을 적용할 경우, 순방향 결과에 담긴 요구사항 문맥이 분할 기준으로 개입하면서 이러한 포괄적 덩어리들이 정상적인 단위(37~50건)로 분할되는 강력한 최적화 효과가 나타났다.

4.6 암묵적 기능 누락 완화 효과

표 8은 역방향 및 양방향 복원 과정에서 9회 실험 중 복원에 실패한 유스 케이스들의 횟수와 유형을 분류한 표이다.

표 8. 역방향 및 양방향 복원 유스케이스별 복원실패횟수

유스 케이스	명칭	유형	역방향	양방향
UC32	ERROREMODIFICAPASSWORD	오류처리	9	8
UC44	RICHIESTACONVENZIONE	업소관점	9	8
UC43	MODIFICADATIPUNTODIRISTORO	업소관점	9	6
UC45	VISUALIZZASTATISTICHEPERSONALI	업소관점	9	6
UC56	LOCALIZATION	시스템기능	9	6
UC29	ERRORETAGESISTENTE	오류처리	9	5
UC55	FEEDBACKGIA'RILASCIATO	오류처리	9	4
UC11	VISUALIZZASTORICOCONVENZIONI	이력조회	9	3
UC36	LOGINERRATO	오류처리	9	1
UC18	CONTROLLANUMEROBANNER	제약검사	9	0
UC35	LOGIN	인증	9	0
UC41 / UC42	MODIFICA / ELIMINAMENU	메뉴관리	6	6

표 8의 결과에 따르면 오류 처리 및 제약검사 기능은 역방향 복원에서 9회 모두 식별에 실패했다. 이는 해당 기능들이 소스 코드 내에서 명시적이지 않고 예외 처리 분기로만 존재하여 모델이 이를 독립 기능으로 파악하지 못했기 때문이다. 반면, 양방향 복원에서는 요구사항 기반의 순방향 산출물이 해당 기능의 존재에 대한 문맥적 정보를 제공함으로써 전반적으로 복원 실패 횟수가 감소하였다. 특히 UC36과 UC18의 경우 실패 횟수가 각각 1회와 0회까지 감소하였다. 이는 양방향 수렴 기법이 단일 역방향 복원의 근본적인 누락 한계를 상호 보완을 통해 완화할 수 있음을 입증하는 결과이다.

V. 결 론

본 논문은 대형 언어 모델(LLM)을 활용하여 코드를 직접 생성하는 개발 환경에서 중간 산출물이 부재하여 요구사항 추적성이 단절되는 문제를 해결하기 위해, 양방향 수렴 기반의 산출물 복원 및 추적성 확보 방안을 제안하였다. 제안 방안은 정보가 차단된 독립적인 환경에서 요구사항 기반의 순방향 복원과 소스 코드 기반의 역방향 복원을 각각 수행한 뒤, 두 결과를 대조하고 수렴시키는 구조를 갖는다.

CoEST의 eTOUR 데이터셋을 대상으로 3종의 언어 모델과 3가지 조건에 대해 총 27회 실험을 진행한 결과, 제안 기법의 주요 성과와 한계는 다음과 같다.

첫째, 양방향 복원은 역방향 복원 대비 유스케이스 복원 커버리지를 65.5%에서 75.1%로 9.6%p 향상시켰다(p=0.0051). 특히 소스 코드 내에서 명시적으로 드러나지 않아 단일 역방향 복원에서 반복적으로 식별에 실패했던 유스케이스들의 복원 실패 횟수가 양방향 복원 과정에서 전반적으로 감소하여 암묵적 기능 누락 문제를 완화하였다.둘째, 모델이 지나치게 포괄적인 유스케이스를 생성하여 실질적인 추적 커버리지가 16.7%에 그쳤던 경우에도, 양방향 복원을 거치며 커버리지가 14.4%p 개선되는 등 추적 단위의 최적화 효과가 확인되었다. 셋째, 가장 핵심적으로 양방향 복원은 요구사항 식별자를 산출물에 자동으로 부여함으로써, 기존 역방향 복원에서 필수적이었던 사람의 수동 대응 작업 없이 요구사항 식별자와 코드 링크를 하나의 통합 산출물에 자동으로 결합할 수 있음을 확인하였다.

다만, 과잉 링크 현상으로 인해 추적 링크의 재현율은 0.548에서 0.620으로 향상되었으나 정밀도가 0.398에서 0.385로 하락하면서, 결과적으로 F1 점수에는 유의미한 차이가 없었다(p=0.418). 이러한 한계를 보완하고 연구를 고도화하기 위한 향후 연구 방향은 다음과 같다.

첫째, 정밀도 하락의 원인인 과잉 링크 억제 방안을 연구한다. 생성된 추적 링크에 신뢰도를 부여하여 임계값으로 선별하거나, 호출 그래프 분석 기법과 결합하여 구조적 근거가 부족한 링크를 필터링하는 방안을 고려할 계획이다. 둘째, 복원 대상을 유스케이스 시나리오 및 객체 단계로 확장하여, 선행 연구[2]에서 정의한 5단계 전체 산출물 구간에 대한 통합 추적성을 확보한다. 셋째, 복원된 유스케이스 시나리오를 실행 가능한 인수 테스트로 변환하고 코드 커버리지를 계측함으로써, 생성된 추적 링크를 실제 실행 근거를 바탕으로 검증하는 메커니즘을 연구한다. 넷째, eTOUR 외의 다양한 데이터셋과 산업 도메인으로 검증 범위를 확대하고 반복 횟수를 늘려 제안 기법의 일반화 가능성을 확립할 예정이다.

다섯째, 요구사항과 소스 코드를 동시에 하나의 LLM에 제공하는 direct joint-input 방식과 제안하는 정보 차단 기반 양방향 수렴 방식을 동일한 조건에서 비교하여, 독립적 복원 및 수렴 과정의 효과를 추가로 분석할 예정이다.

REFERENCES

[1] A. Sarkar, I. Drosos, “Vibe Coding: Programming through Conversation with Artificial Intelligence”, *in Proceeding of the 36th Annual Conference of the Psychology of Programming Interest Group* (PPIG), UK, 2025.

[2] Janghwan Kim, R. Young Chul Kim, “Current Issues on Requirement Traceability Mechanism for Software Organization of 4th Industrial Revolution”, *International Journal of Advanced Smart Convergence,* Vol. 9, No. 4, pp. 167-172, 2020.

[3] Jinhyub Lee, “A Study on Requirement Traceability Matrices based on Automatic Software Process”, *M.S. thesis, Dept. Electron. Comput. Eng., Hongik Univ., Korea*, 2018.

[4] O. C. Z. Gotel, A. C. W. Finkelstein, “An Analysis of the Requirements Traceability Problem”, *in Proceeding of the 1st International Conference on Requirements Engineering (ICRE)*, pp. 94-101, Colorado Springs, USA, 1994.

[5] B. Ramesh, M. Jarke, “Toward Reference Models for Requirements Traceability”, *IEEE Trans. Softw. Eng.,* Vol. 27, No. 1, pp. 58-93, Jan. 2001.

[6] A. Wibowo, J. G. Davis, “Requirements Traceability Ontology to Support Requirements Management”, i*n Proceeding of the Australasian Computer Science Week Multiconference (ACSW)*, Australia, 2020.

[7] K. Yoshino, S. Matsuura,“Requirements Traceability Management Support Tool for UML Models”, *in Proceeding of the International Conference on Software and Computer Applications (ICSCA)*, pp. 163-166, Malaysia, 2020.

[8] M. Unterkalmsteiner, “Early Requirements Traceability with Domain-Specific Taxonomies: A Pilot Experiment”, *in Proceeding of the IEEE 28th International Requirements Engineering Conference (RE)*, pp. 322-327, Switzerland, 2020.

[9] S. Xiao, Y. Zhang, W. Sun, X. Chen, Y. Liu, Z. Jin, “UCRBench: Benchmarking LLMs on Use Case Recovery”, *arXiv preprint arXiv:2512.13360*, Dec. 2025.

[10] H. Jin, H. Chen, “Uncovering Systematic Failures of LLMs in Verifying Code Against Natural Language Specifications”, *in Proceeding of the 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1-6, Korea, 2025.

[11] eTOUR Dataset(2011), http://sarec.nd.edu/coest/datasets.html (07, 30, 2026).

[12] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, “Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods”, *IEEE Transaction Software. Engineering*, Vol. 32, No. 1, pp. 4-19, Jan. 2006.

[13] D. Cuddeback, A. Dekhtyar, J. H. Hayes, “Automated Requirements Traceability: The Study of Human Analysts”, *in Proceeding of the 18th IEEE International Requirements Engineering Conference (RE)*, pp. 231-240, Sydney, Australia, 2010.

저 자 소 개

이진협(정회원)

2016년 홍익대학교 컴퓨터정보통신공학과 학사 졸업.

2018년 홍익대학교 전자전산공학과 석사 졸업.

<주관심분야 : SW품질, 테스트자동화, 추적성>

김장환(정회원)

2019년 아이다호 주립 대학교 Computer Science 학사 졸업.

2022년 홍익대학교 소프트웨어융합학과 석사 졸업.

2024년 홍익대학교 소프트웨어융합학과 박사 수료

<주관심분야 : 소프트웨어공학, 소프트웨어 가시화, AI 소프트웨어테스팅, 강화학습소프트웨어 검증>

서채연(정회원)

2014년 홍익대학교 소프트웨어공학 박사 졸업.

2021년 우즈베키스탄 한국국제대학 조교수

2024년 홍익 메타버스융합소프트웨어 사업단 총괄팀장/초빙교수

<주관심분야:소프트웨어공학Component Repository, AR/VR, 생성형 AI, 웹, 언리얼엔진5, 디자인씽킹>

전병국(정회원)

2000년 광운대 컴퓨터과학 박사졸업

2026년 강원대 컴퓨터공학과 교수

<주관심분야 : AI, Mobile Agent 분산컴퓨팅>

김영철(정회원)

2000년 일리노이공대(IIT) 소프트웨어공학 박사졸업

2026년 홍익대학교 소프트웨어융합학과 교수 재직

<주관심분야 : 인공지능 검증, 강화학습 소프트웨어 검증, 소프트웨어공학, 메타버스(AR/VR), 소프트웨어테스팅, 소프트웨어 가시화, 소프트웨어 품질>