테스트 시나리오 기반 바이브코딩 프로세스의 Java 웹 개발 품질 개선 연구

최미리*, 유동근**
*(주)지오인프라
**(주)지오인프라
e-mail:mlchoi@gioinfra.co.kr

A Study on Software Quality Improvement in Java Web Development through Test Scenario-Based Vibe Coding Process

Mi-Ri Choi*, Dong-Geun Yoo**
*Gioinfra Co., Ltd.
**Gioinfra Co., Ltd.

요 약

본 논문은 실무 Java 웹 개발 환경에서 바이브코딩(Vibe Coding)을 적용한 두 가지 프로젝트 사례를 비교 분석하여, 소프트웨어 품질에 영향을 미치는 핵심 요인을 규명하고 개선된 개발 프로세스를 제안한다. 사전 설계가 충분히 정의되지 않은 상태에서 구현과 변경을 반복하거나, AI가 생성한 테스트 시나리오를 개발자 검토 없이 그대로 적용할 경우 다수의 버그가 발생함을 실무 사례를 통해 확인하였다. 이를 바탕으로 개발자가 Given/When/Then 기반의 테스트 시나리오에 직접 개입하고, 기능을 점진적으로 슬라이싱하여 개발하며, AI 작업 완료 후 매뉴얼 테스트를 반드시 수행하는 테스트 시나리오 기반 바이브코딩 프로세스를 제안한다.

1. 서론

현대 소프트웨어 개발 환경에서 AI 코드 생성 도구의 활용이 급격히 확산되고 있다. 특히 개발자가 자연어로 요구사항을 입력하면 AI가 코드를 생성하는 이른바 '바이브코딩(Vibe Coding)' 방식은 개발 속도를 획기적으로 단축시키는 수단으로 주목받고 있다.

그러나 바이브코딩 환경에서 AI는 명시적으로 정의된 내용에 기반하여 결과물을 생성하기 때문에, 충분한 상세한 설계와 결정이 필요한 부분들까지 사전에 정의되지 않을 경우 예상치 못한 버그와 품질 저하가 발생할 수 있다는 문제점도 함께 대두되고 있다.

본 연구는 실무 Java 웹 개발 환경에서 바이브코딩을 적용한 두 가지 프로젝트 사례를 비교 분석하여, 테스트 시나리오에 대한 개발자 개입 여부가 소프트웨어 품질에 미치는 영향을 규명하고, 이를 바탕으로 실무에서 즉시 적용 가능한 바이브코딩 기반 개발 프로세스를 제안하는 것을 목적으로 한다.

2. 관련연구

2.1 바이브코딩의 개념

바이브코딩이란 개발자가 구현하고자 하는 기능을 자연어로 AI에게 전달하고, AI가 생성한 코드를 기반으로 개발을 진행하는 방식을 말한다. 대표적인 도구로는 Claude Code, GitHub Copilot, Cursor 등이 있으며, 본 연구에서는 Claude Code를 활용한 사례를 중심으로 분석한다.

바이브코딩은 반복적이고 정형화된 코드 작성 작업을 AI에게 위임함으로써 개발자가 설계와 검증에 집중할 수 있는 환경을 제공한다. 특히 소규모 팀이나 1인 개발 환경에서 생산성을 크게 향상시킬 수 있는 잠재력을 가진다.

2.2 바이브코딩의 한계 및 문제점

선행 연구 및 실무 사례에 따르면 바이브코딩 도입 초기에는 AI가 생성한 코드와 테스트시나리오를 개발자가 충분히 검토하지 않고 그대로 적용하는 경우가 많다. AI는 주어진 맥락 안에서 표면적으로 동작하는 코드를생성하지만, 실제 비즈니스 로직의 엣지 케이스나 예외 상황을 충분히 반영하지 못하는 경우가 발생하난. 특히 테스트 시나리오를 AI에게 전적으로 위임할 경우, AI가 테스트를 통과시키기 위해 하드코딩 방식으로 구현하는 사례도 발생하며, 이는 실질적인 품질 검증이 이루어지지 않은 채 테스트만 통과하는 결과로 이어진다. 결과적으로는 기능은 동작하지만 예상치 못한 버그가 다수 발생하는 품질 문제로 이어진다.

3. 실무 적용 사례 분석

3.1 사례 개요

본 연구에서는 동일한 개발자가 Claude Code를 활용하여 개발한 두 가지 Java 웹 프로젝트를 비교 분석한다.

[표 1] 프로젝트별 바이브코딩 적용 방식 비교

구분	CMS	이슈 관리 시스템
프로젝트 규모	대	중
사용 도구	Claude Code	Claude Code
테스트 시나리오 작성	AI 생성 그대로 적용	AI 초안+개발자 직접 보완
Given/When/Then 정의	미적용	개발자 직접 정의
발생 버그 수	다수	1건

3.2 CMS 프로젝트 사례

CMS 프로젝트는 Claude Code를 활용하여 C4 다이어그램, ERD, 시퀀스 다이어그램을 설계하고, 기능을 슬라이싱하여 개발을 진행하였다. 그러나 유닛 테스트 및 E2E 테스트 시나리오를 AI가 생성한 그대로 적용하였으며, 개발자의 별도 검토 및 보완 없이 코드 생성을 진행하였다.

그 결과 AI가 테스트를 통과시키기 위해 하드코딩 방식으로 구현하는 사례가 발생하였고, 실질적인 품질 검증이 이루어지지 않은 채 다수의 버그가 누적되었다. 이는 테스트 시나리오에 검증 대상과 검증 방법이 명확히 정의되지 않았기 때문으로 분석된다.

3.3 이슈 관리 시스템 프로젝트 사례

이슈 관리 시스템은 CMS 프로젝트의 경험을 바탕으로, 테스트 시나리오 작성 방식을 근본적으로 개선하였다. AI가 생성한 유닛 테스트 및 E2E 테스트 시나리오 초안을 개발자가 직접 검토한 뒤, 각 시나리오에 대해 Given(사전 조건), When(실행 조건), Then(기대 결과)을 자연어로 직접 정의하였다. 이를 통해 검증 대상과 검증 방법을 명확히 규정하였으며, 실제 구현은 이 시나리오를 기반으로 Claude Code가 수행하도록 하였다.

그 결과 AI가 테스트만 통과시키기 위한 하드코딩 방식으로 구현하는 문제가 방지되었고, 현재까지 발생한 버그는 1건에 불과하다. 이는 개발자가 Given/When/Then 수준까지 직접 정의한 테스트 시나리오 실질적인 품질 검증의 기준으로 작용하였기 때문으로 분석된다.

4. 테스트 시나리오 기반 바이브코딩 프로세스 제안

4.1 제안 프로세스 개요

앞선 사례 분석을 바탕으로, 본 연구에서는 다음과 같은 테스트 시나리오 기반 바이브코딩 프로세스를 제안한다.

1단계: 설계(AI협업+개발자 적극 개입) AI와의 반복적인 대화를 통해 C4 다이어그램, ERD, 시퀀스 다이어그램을 작성한다. 이 단계에서 개발자는 AI의 제안을 수동적으로 수용하는 것이 아니라, 적극적이고 비판적인 태도로 AI와 내용을 주고 받으며 시스템의 전체 구조와 데이터 흐름, 그리고 세부 설계 결정 사항까지 명확히 정의해야 한다. 충분히 상세한 설계 없이 다음 단계로 넘어갈 경우, 이후 구현 단계에서 잦은 변경이 발생하여 품질 저하로 이어질 수 있다.

2단계: 기능 슬라이싱(점진적 개발의 기반) 설계를 바탕으로 구현할 기능을 독립적인 단위로 분리하여 리스트화한다. 기능 슬라이싱을 수행하지 않을 경우, 프로젝트 규모가 일정 수준 이상 커졌을 때 전체 구현 범위에 압도되어 중요한 기능의 검증을 스킵하거나 누락하는 상황이 발생할 수 있다. 따라서 기능을 작은 단위로 분리하고 각 단위를 점진적으로 구현 및 검증하는 방식을 반드시 준수해야 한다.

3단계: 테스트 시나리오 작성(개발자 개입 필수) AI가 각 기능에 대해 유닛 테스트 및 E2E 테스트 시나리오 초안을 작성하면, 개발자가 반드시 직접 검토하여 각 시나리오에 대해 Given(사전 조건), When(실행 조건), Then(기대 결과)을 자연어로 정의한다. 이를 통해 검증 대상과 검증 방법을 명확히 규정하며, 실제 업무 흐름의 엣지 케이스와 예외 상황을 추가로 보완한다. 본 단계는 AI가 테스트만 통과시키기 위한 하드코딩 구현을 방지하는 핵심 단계로, 개발자의 직접적인 개입이 반드시 필요하다.

4단계: 코드 생성 및 실행 개발자가 정의한 테스트 시나리오를 기반으로 AI가 코드를 생성하고 테스트를 실행한다. 이 단계는 AI가 전담하되, 앞선 단계에서 명확히 정의된 시나리오를 기준으로 수행한다.

5단계: 매뉴얼 테스트 및 시나리오 보강 AI가 모든 작업을 완료한 이후에도 개발자의 매뉴얼 테스트가 반드시 수행되어야 한다. 자동화 테스트만으로는 발견하기 어려운 실제 사용 흐름상의 문제나 UI/UX 측면의 오류를 매뉴얼 테스트를 통해 확인하며, 발견된 문제를 바탕으로 테스트 시나리오를 보강한 후 4단계로 돌아가 반복한다.

4.2 기존 방식과의 비교

기존 바이브코딩 방식과 제안 프로세스의 차이를 비료하면 표2

와 같다.

[표 2] 기존 바이브코딩 방식과 제안 프로세스 비교

설계	기존 바이브코딩 방식	제안 프로세스
설계	AI 의존	AI+개발자 적극 협업
테스트 시나리오	AI 생성 그대로 적용	AI 초안+개발자 Given/When/ Then 정의
코드 생성	AI 전담	AI 전담
매뉴얼 테스트	미흡 또는 생략	반드시 수행
버그 발생	다수	최소화

3. 결론

본 연구는 실무 Java 웹 개발 환경에서 바이브코딩을 적용한 두 프로젝트 사례를 비교 분석하여, 바이브코딩의 품질을 결정하는 pt 가지 핵심 요소를 도출하였다. 첫째, 테스트 시나리오 작성 단계에서 개발자가 Given,/When/Then 수준까지 직접 개입하여 검증 대상과 검증 방법을 명확히 정의해야 한다. 둘째, 기능을 작은 단위로 슬라이싱하여 점진적으로 구현하고 검증하는 방식을 준수해야 한다. 셋째, AI의 자동화 테스트 완료 후에도 개발자의 매뉴얼 테스트가 반드시 동반되어야 한다.

이 세가지 요소를 갖춘 테스트 시나리오 기반 바이브코딩 프로세스는 AI의 코드 생성 속도라는 장점을 유지하면서도, 개발자의 도메인 지식을 설계 및 검증 단계에 집중적으로 투입함으로써 실무 수준의 소프트웨어 품질을 확보할 수 있는 실용적인 방법론이다. 향후 연구에서는 다양한 프로젝트 규모와 도메인에 본 프로세스를 적용하여 효과를 정량적으로 측정하는 연구가 필요하다.

참고문헌

[1] Anthropic, “Claude Code: Agentic Coding”, Anthropic Documentation, 2025.

[2] Kent Beck, “Test Driven Development: By Example”, Addison-Wesley, 2002.