Journal of KIIT. Vol. 24, No. 5, pp. 25-38, May 31, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.5.25

GitHub 협업 로그와 생성형 AI를 활용한 팀 기반 프로그래밍 프로젝트에서의 학습자별 기여도 분석

김유안*[1], 김난주*[2], 박찬정**

Analysis of Individual Learner Contributions in Team-based Programming Projects using GitHub Collaboration Logs and Generative AI

Yuan Kim*[1], Nanju Kim*[2], and Chan Jung Park**

요 약

본 연구는 팀 기반 프로그래밍 과제에서 교수자의 관찰 한계와 개별 코드 검토 부담으로 인해 학습자 개인의 실제 기여도를 정밀하게 파악하기 어렵다는 문제를 해결하고자, GitHub 협업 로그와 생성형 AI 기반 분석을 결합한 평가 지원 시스템을 제안한다. 커밋, 풀 리퀘스트, 이슈, 코드 리뷰 등 협업 로그를 개인 단위의 정량 지표로 재구성하고, 생성형 AI를 활용해 정성 분석을 병렬적으로 제공함으로써 평가 해석 정보를 확장하였다. 실제 대학 SW 수업 사례 적용 결과, 정량 지표는 안정적이었지만 AI 기반 정성 분석은 변동성 문제로 최종 평가 주체로 활용하기에는 보완이 되어야 함을 확인하였다. 본 연구는 데이터 기반 분석의 가치와 한계를 규명하며, 교수자의 평가 판단을 돕는 보조 도구로서의 가능성과 향후 협업 학습 평가를 위한 시사점을 제공한다.

Abstract

This paper proposes an assessment support system combining GitHub collaboration logs and Generative AI to address the challenges instructors face in accurately determining individual contributions in team-based programming assignments, due to observation limitations and the burden of code reviews. By reconstructing collaboration logs—including commits, pull requests, issues, and code reviews—into individual quantitative metrics and providing parallel qualitative analysis through Generative AI, the system expands the scope of evaluative interpretive information. Application in an actual university software course revealed that while quantitative metrics remained stable, AI-based qualitative analysis exhibited variability, confirming that further refinement is necessary for AI to serve as a final assessment authority. This research investigates the value and limitations of data-driven analysis, offering potential as a tool to assist instructors' evaluative judgment and providing implications for future collaborative learning assessment.

Keywords
github, generative AI, individual learner assessment, programming project, team-based learning

* 제주대학교 컴퓨터교육과 학사과정
- ORCID[1]: https://orcid.org/0009-0007-8509-7946
- ORCID[2]: https://orcid.org/0009-0004-5315-2427
** 제주대학교 컴퓨터교육과 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-3146-2342

· Received: Feb. 12, 2026, Revised: Mar. 06, 2026, Accepted: Mar. 09, 2026
· Corresponding Author: Chan Jung Park
Dept. of Computer Education, Jeju National University, Jejudaehak-ro 102 63243, South Korea
Tel.: +82-64-654-3296, Email: cjpark@jejunu.ac.kr

Ⅰ. 서 론

2022 개정 교육과정에서는 협력적 소통 역량을 핵심 역량으로 명시하고 있으며, 교수·학습 방법 측면에서도 협력적 문제해결력을 신장하기 위한 협력학습이 지속적인 관심을 받고 있다[1]-[3]. 특히 소프트웨어 교육에서는 실제 개발 맥락과 유사한 팀 기반 프로그래밍 과제가 학습자의 문제해결력과 협업 역량을 기르는 주요 수단으로 활용되고 있다. 그러나 이러한 교육적 기대와 달리, 팀 기반 프로그래밍 과제에 대한 평가는 여전히 교수자의 제한된 관찰과 최종 산출물 중심의 검토에 크게 의존하고 있으며, 이로 인해 협업 과정에서 학습자 개인의 실제 기여 양상을 정밀하게 파악하는 데 구조적인 한계가 지속적으로 제기되어 왔다[4]-[6].

이러한 평가 방식은 협업 과정에서의 개별 책무성을 약화시키고, 팀 성과에 편승하는 무임승차 문제나 핵심적인 역할을 수행한 학습자의 노력이 과소 평가되는 공정성 문제로 이어질 수 있다[4]-[6]. 결과적으로 팀 프로젝트가 학습자의 성장과 역량 발달을 촉진하기보다는 형식적인 과제로 전락할 위험이 있으며, 이는 협력학습의 교육적 효과를 저해하는 요인으로 지적되어 왔다[4]-[6].

이러한 문제를 보완하는 방안으로 동료 평가(Peer assessment)를 도입하려는 시도가 이루어져 왔다[5]. 일부 연구에서는 동료 평가가 협업 과정에 대한 학습자의 인식을 개선하고 책임감을 강화하는 데 긍정적인 영향을 미친다고 보고하였다[6]-[9]. 그러나 동료 평가 역시 평가의 공정성과 신뢰성, 학습자 간 관계로 인한 왜곡 가능성, 평가 부담 등의 한계를 지니며, 최종적으로는 학습자 스스로의 판단에 의존할 수밖에 없다는 점에서 보조적인 평가 도구의 필요성이 지속적으로 제기되고 있다[6]-[9].

한편, 최근 소프트웨어 교육 현장에서는 GitHub[10]와 같은 실무 기반 협업 플랫폼이 보편화되면서, 실제 산업 환경과 유사한 개발 경험을 제공할 수 있는 기반이 마련되었다. 이에 따라 GitHub에서 생성되는 커밋, 풀 리퀘스트, 이슈, 코드 리뷰와 같은 협업 로그를 교육 평가에 활용하려는 연구도 점차 증가하고 있다[11]-[13]. 이러한 접근은 협업 과정에서의 활동 이력을 객관적인 데이터로 확보할 수 있다는 점에서 의미가 있다. 그러나 기존 연구들은 최종 산출물 중심의 단편적 분석에 치우치거나, 일부 정량 지표에만 의존하여 개별 학습자의 실제 기여도와 수행 성실성을 충분히 반영하지 못하는 한계를 보인다[11]-[13]. 또한 교수자가 팀별·개인별 GitHub 활동 로그를 직접 확인하고 정리해야 하는 물리적 부담으로 인해, 특히 대규모 강좌 환경에서는 평가 효율성이 현저히 저하되는 문제가 존재한다. 더 나아가 기존 평가 지원 도구들은 GitHub의 정량적 활동 지표, 코드 품질 분석, 수행계획서 대비 기능 구현 수준을 통합적으로 분석하지 못하여, 협업 과정과 결과를 함께 고려하는 종합적 평가에는 한계를 가진다.

이에 본 연구는 팀 기반 프로그래밍 협업 과제에서 학습자의 성실도, 코드 품질, 역할 이행, 기능 구현 수준 및 프로젝트 완성도를 통합적으로 분석·지원하기 위한 GitHub 기반 인공지능 평가 지원 시스템을 제안한다. 제안하는 시스템은 GitHub 저장소에서 생성되는 커밋, 풀 리퀘스트, 이슈, 코드 리뷰 데이터를 GitHub API를 통해 자동 수집하고, 이를 개인 단위의 개발 활동으로 재구성한다. 특히 정적 코드 분석 결과와 대규모 언어 모델(LLM, Large Language Model) 기반의 코드 의미 분석을 결합하여 정량·정성적 평가지표를 병렬적으로 산출하며, ISO/IEC 25010 소프트웨어 품질 특성[14]과 수행계획서 기반 역할 - 기능 매핑을 반영함으로써 평가의 객관성과 해석 가능성을 확보하고자 한다. 또한 ChatGPT[15]를 활용하여 코드 가독성, 구조적 완성도, 예외 처리, 문서화 수준 등 정성적 요소를 체계적으로 분석함으로써, 기존 정량 지표 중심 평가의 한계를 보완한다.

본 연구는 GitHub 협업 로그와 생성형 AI 기반 분석을 결합한 통합 평가 접근을 통해, 협업 환경에서도 개별 학습자 중심의 과정 기반 평가가 가능함을 제시한다. 나아가 교수자에게는 데이터 기반의 효율적인 평가 판단 근거를 제공하고, 학습자에게는 공정하고 투명한 피드백을 제공함으로써 팀 기반 프로그래밍 교육에서의 평가 패러다임을 확장하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 팀 기반 프로그래밍 협업 과제의 교육적 배경과 기존 평가 방식의 한계를 고찰한다. 3장에서는 GitHub 협업 로그와 생성형 AI를 활용한 개별 기여도 분석 개념을 제시하고, 제안하는 시스템의 설계 원리와 분석 지표를 설명한다. 4장에서는 제안하는 시스템의 구조와 구현 방법을 기술하고, 정량·정성 평가 절차를 제시한다. 5장에서는 실제 대학 소프트웨어 수업의 팀 프로젝트 사례에 적용한 분석 결과를 제시하고, 평가 지원 측면에서의 효과와 한계를 분석한다. 6장에서는 결론을 제시한다.

II. 관련 연구

2.1 기존 팀 기반 학습에서 학습자 평가 방식의 한계 고찰

형상 관리 도구는 소프트웨어 개발 프로젝트에서 코드 품질 유지와 팀원 간 협업을 지원하는 핵심 도구로, Git은 변경 이력 관리와 병렬 개발을 효과적으로 지원한다. 기존의 연구 중, 강민석은 Git 기반 조별 프로그래밍 과제에서 커밋 횟수와 커밋 레벨을 활용해 개별 기여도를 정량적으로 평가하는 방법을 제안하였다[11]. 이 연구는 커밋의 난이도를 단계화하여 기여도를 보완하였으나, 불필요한 커밋 증가와 코드 품질을 반영하지 못하는 한계를 지닌다. 또한, 박지은은 SVN 로그를 활용해 제출 일수, 수정 코드 라인 수 등을 자동 수집하여 참여도와 성실도를 평가하였으나, 정성 평가 과정에서 교수자의 수동 개입이 필요하다는 한계가 있었다[16]. 한편 캡스톤디자인 수업 평가 연구에서는 자기평가와 동료 평가를 활용한 개별 평가 방안이 제시되었으나[17][18], 주관성과 과대평가 문제로 보조적 수단에 그쳤다. 최근 김리한 등의 연구에서는 AI 기반 평가 서비스를 제안하였으나, 실제 코드 기여와 협업 활동을 충분히 반영하지 못하였다[19]. 이러한 선행연구들은 협업 과정의 정량적 분석에는 기여했으나, 코드 품질과 문제 해결 기여도를 통합적으로 평가하는 데에는 한계를 보인다.

기존 연구들은 SVN이나 Git 로그와 같은 정량 데이터를 활용하여 개별 기여도를 평가하는 데 기여하였으나, 코드 품질, 협업 태도, 문서화 수준과 같은 정성적 요소를 충분히 반영하지 못하는 한계를 보였다. 또한 동료 평가와 자기평가에 의존하는 방식은 교수자의 관리 부담이 크고, 학생 간 친분이나 갈등에 따른 주관적 개입 가능성이 존재하여 평가의 신뢰성을 저하시킬 우려가 있다. 이에 본 연구는 정량 지표 편중, 자동화 부족, 정성 평가의 한계, 교수자 부담이라는 기존 평가 체계의 문제를 종합적으로 개선하고자 한다.

2.2 생성형 AI 기반 교육 평가

최근 교육 평가는 결과 중심에서 학습 과정 전반을 지속적으로 점검하고 이를 교수·학습 개선에 활용하는 과정 중심 평가로 전환되고 있다[20][21]. 수행평가는 이러한 관점을 구현하는 핵심 수단이지만, 관찰과 판단에 따른 교사의 부담이 크다는 한계를 지닌다[21]. 특히 프로그래밍 교육에서는 커밋 수, 코드 라인 수, 테스트 통과 여부 등 결과 중심 지표가 주로 활용되어 왔으나[11], 설계 의도, 문제 해결 과정, 협업 맥락에서의 논리적 기여와 같은 정성적 요소를 충분히 반영하지 못했다[16].

대규모 언어 모델(LLM) 기반 생성형 AI의 확산은 평가 패러다임의 전환을 촉진하고 있다[22][23]. 생성형 AI는 코드의 구조적 완성도, 가독성, 예외처리 적절성, 설계 의도 등 고차원적 의미 분석을 수행함으로써 평가를 결과물의 정량적 측정에서 수행 과정에 대한 정성적 해석으로 확장한다[23]. 한 연구에서는 생성형 AI가 형성평가에서의 관찰 한계와 평가 부담을 완화하며, STEM 분야에서 학습자의 문제 해결 방식을 변화시키고 있어 전통적 평가 방식의 재설계 필요성이 제기되고 있다[23].

III. 시스템 설계

본 연구에서 제안하는 시스템은 팀 기반 프로그래밍 프로젝트를 공정하고 효율적으로 평가하기 위해 GitHub 활동 데이터를 활용한 기여도·성실성 평가 방법, 코드 품질·기능 구현·역할 수행을 통합하는 다차원 평가 산출 방식, 그리고 평가 결과를 교수자가 효율적으로 활용할 수 있는 자동화 시스템

이 되도록 설계하였다. 본 연구를 위한 시스템 구조도는 그림 1과 같다.

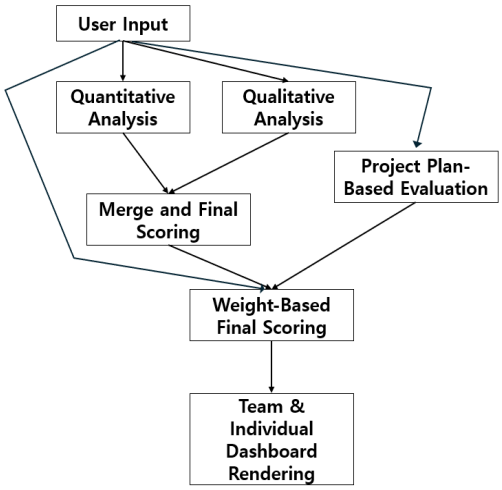


그림 1. 시스템 구조도
Fig. 1. System structure diagram

또한, 제안하는 시스템에 대한 설계 절차는 표 1과 같다. 제안하는 시스템은 팀 기반 프로그래밍 협업 과제의 개별 학습자 평가를 위해 정량 분석, 정성 분석, 수행계획서 기반 평가, 통합 점수 산출, 플랫폼 실행의 다섯 개 모듈로 구성된다. 먼저 정량 분석 모듈은 GitHub API를 활용하여 커밋, 풀 리퀘스트(PR, Pull Request), 이슈, 활동 일수, 수정 라인 수 등 협업 활동 데이터를 자동 수집한다. 이 과정에서 burst- spam, self-fix, copy-paste와 같은 이상 커밋을 탐지하여 유효 커밋 산출에 반영함으로써 정량 지표의 신뢰성을 확보한다.

정성 분석 모듈에서는 Pylint[24], Radon[25], Bandit[26]을 실행하여 원본 정적 분석 결과를 생성한 후, GPT 기반 분석을 통해 오탐(FP, False Positive)을 제거하고 미탐(FN, False Negative)을 보완함으로써 정적 분석 결과의 정확도를 향상시킨다. 이때, 오탐(FP)은 실제로는 문제가 없는 코드임에도 불구하고 오류, 취약점, 또는 품질 저하로 잘못 판단하여 결함으로 검출하는 경우이다. 미탐(FN)은 실제로 존재하는 오류, 취약점, 또는 품질 문제를 탐지하지 못하고 정상 코드로 판단하여 평가에서 누락하는 경우이다.

이를 바탕으로 ISO/IEC 25010 품질 특성에 따라 효율성, 유지보수성, 보안성 중심의 코드 품질 점수를 산출하며, 대규모 언어 모델을 활용해 기능 구현 점수(F)와 기능별 근거 문장을 생성한다. 또한 정성 분석 결과를 canonical ID 기준으로 재구조화하고, blame 기반 라인 단위 기여자 뷰어를 생성하여 코드 기여의 맥락을 명확히 제시한다.

표 1. 설계 절차
Table 1. Design procedure

Stage	Key contents	Techniques & tools
Requirements analysis	Identifying problems in existing team project evaluations and deriving system requirements	Literature review, case study
Data collection	Automatic collection and normalization of commits, PRs, Issues, and reviews from GitHub repositories	GitHub API, python scripts
Static analysis	Analysis of code quality, complexity, and security vulnerabilities based on pylint, radon, and bandit	Static analysis tools, python
FP/FN filtering	Removal of False Positives (FP) and supplementation of False Negatives (FN) using GPT	OpenAI GPT API, JSON structuring
Author mapping	Linking code lines to contributors using Git blame and Commit/PR data	Git blame, python mapping logic
Qualitative evaluation	Analysis of code quality, functional implementation, and role fulfillment	Requirement mapping, GPT-based analysis, JSON output
Integration & scoring	Integrating quantitative (S) and qualitative (Q/F/C) data to calculate final individual scores	Python, instructor-weighted calculations
Report & visualization	Generating team/individual dashboards and providing evaluation results	Flask Web dashboard

수행계획서 기반 평가 모듈은 수행계획서(PDF)를 입력으로 받아 기능 요구사항과 팀원 역할을 구조화된 JSON 형태로 추출하고, 이를 기준으로 역할 및 기능 완성도(C)를 평가한다. 통합 및 점수 산출 모듈에서는 GitHub ID와 canonical ID를 자동 매핑하고 alias 목록을 생성하여 정량(S) 및 정성(Q/F) 데

이터를 통합한다. 최종적으로 성실성(S), 코드 품질(Q), 기능 구현(F), 역할·기능 완성도(C)에 교수자 가중치를 적용하여 개인별 최종 점수를 산출한다.

마지막 실행 단계에서는 정량 분석부터 정성 분석, 데이터 병합까지의 전체 평가 파이프라인을 orchestrator가 순차적으로 실행하며, 사용자 입력 처리와 분석 결과를 기반으로 팀별·개인별 대시보드를 렌더링하여 평가 결과의 활용성을 높인다.

Ⅳ. 시스템 구현

4.1 입력 데이터 수집 및 전처리

GitHub 활동 데이터 수집을 위해 GitHub 저장소에서 커밋, PR, 이슈 데이터를 수집·정규화하여 개인별 활동을 재구성하고, 개발 참여도와 협업 패턴 분석을 위한 핵심 기초 자료로 활용하였다. 수행계획서 PDF를 분석하여 JSON으로 그림 2와 같이 변환하는 전처리 작업을 진행하였다. 추출된 정보는 기능 단위(FR1, FR2 …)와 팀원 역할 목록으로 정리되며, JSON 형태로 저장된다. 이를 통해 각 팀이 계획 단계에서 제시한 목표와 실제 구현 결과를 비교할 수 있는 기반이 마련되며, 후속 단계에서 기능 완성도와 역할 커버리지 평가의 핵심 입력 자료로 활용된다.

```
당신은 대학 소프트웨어 캡스톤디자인 수행계획서를 분석하여,
"기능 요구사항(functional_requirements)"과 "팀원 역할(team_roles)"을
정확히 추출하는 모델입니다.

[핵심 규칙]
1. 기능/역할 텍스트는 반드시 문서 원문 그대로 사용한다. (요약/변형 금지)
2. 표가 있으면 표의 셀 내용을 그대로 사용한다.
3. 표가 없으면 관련 문장을 문서에서 찾아 원문 그대로 리스트업한다.
4. 기능은 FR1, FR2, FR3... 순서로 자동 ID를 부여한다.
5. project_summary만 GPT가 한 문단으로 생성해도 된다.
6. JSON만 출력한다. 설명 금지. 생략 금지.

[JSON 구조]
{
  "project_summary": "...",
  "functional_requirements": [
    {"id": "FR1", "text": "..."}
  ],
  "team_roles": [
    {"name": "", "roles": ["..."]}
  ]
}

반드시 유효한 JSON으로 출력하라.
"""
```

그림 2. JSON 변환
Fig. 2. JSON transformation

제안하는 시스템은 평가 과정에서 교수자가 지정한 가중치와 분석 기준을 입력 데이터로 함께 수집한다. 여기에는 성실성, 코드 품질, 기능 구현도, 역할 완성도의 반영 비율과 비정상 활동 탐지를 위한 임계값 등이 포함된다. 이러한 설정값은 이후 정량·정성 분석과 최종 점수 산정의 기준으로 적용된다.

4.2 정량적 협업 활동 및 코드 품질 분석 데이터 생성

본 연구에서는 팀 기반 프로그래밍 협업 과정에서 나타나는 개별 기여 양상을 정량적으로 분석하기 위해 GitHub API를 활용하여 커밋, 풀 리퀘스트(PR), 이슈 등 개발 활동 데이터를 수집하고 이를 작성자 단위로 통합하였다. 수집된 데이터로부터 커밋 수, 활동 일수, 코드 변경량, 병합된 PR 수, 해결된 이슈 수 등의 정량 지표를 산출하였으며, 동일 학습자가 여러 계정을 사용하는 경우 이를 하나의 인물로 통합하여 분석의 일관성을 확보하였다. 산출된 정량 지표는 JSON 형태로 구조화되어 이후 성실성 분석과 종합적인 기여 양상 파악을 위한 기초 자료로 활용되었다.

정량 지표 산출 과정에서는 단기간 과도한 커밋 생성, 사소한 변경을 분리한 반복 커밋, 대량의 코드 복사·붙여넣기와 같은 이상 활동을 탐지하고, 해당 활동이 분석 결과에 과도한 영향을 미치지 않도록 점수 산정 과정에서 보정 절차를 적용하였다. 이를 통해 협업 활동량 중심의 정량 분석에서 발생할 수 있는 왜곡 가능성을 최소화하고, 분석 결과의 해석 가능성을 높이고자 하였다.

한편 코드 수준의 특성을 정량적으로 파악하기 위해 정적 분석 도구를 활용하여 코드 스타일 위반, 보안 취약점, 순환 복잡도 등의 분석 결과를 파일 및 라인 단위로 수집하였다. 수집된 정적 분석 결과는 git blame 정보를 활용하여 실제 작성자와 매핑함으로써 개인 단위의 코드 특성 분석이 가능하도록 구성되었다. 이러한 결과는 이후 코드 특성에 대한 해석 정보를 제공하기 위한 기초 자료로 활용되었다.

정적 분석 도구의 규칙 기반 경고에는 실제 코드 품질과 직접적인 관련이 없는 오탐(FP)이 포함될

수 있다는 점을 고려하여, 본 연구에서는 정적 분석 결과의 해석 가능성을 높이기 위한 GPT 기반 오탐 제거 절차를 추가로 설계하였다. 먼저 변수명 규칙 위반, 주석 부족, 사용되지 않는 변수, 낮은 수준의 순환 복잡도(CC1~CC3)와 같이 코드 품질에 미치는 영향이 상대적으로 미미한 경고를 규칙 기반으로 1차 제거하였다. 이후 남은 경고 항목에 대해서는 함수 전체 또는 인접 코드 문맥을 추출하여 GPT가 코드의 목적, 사용 맥락, 수정 난이도, 실행 빈도, 보안 민감도 등을 종합적으로 분석하도록 하였으며, 이를 바탕으로 실제 결함 여부를 판단하였다. 오탐 탐지 절차의 예시는 그림 3에 제시되어 있다.

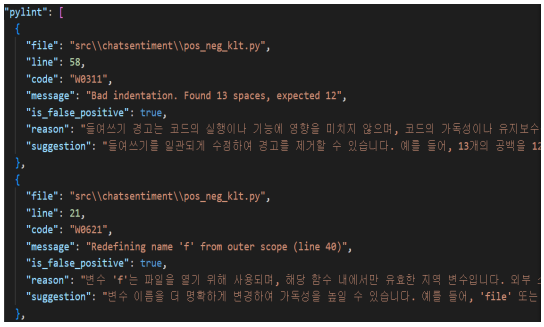


그림 3. 오탐(FP) 탐지
Fig. 3. Example for FP detection

GPT 기반 분석 과정에서는 결함 유형 분류와 함께 근거가 되는 코드 위치와 추정 문제 지점을 함께 제시하도록 하였으며, 이러한 결과는 기존 정적 분석 도구의 산출물과 통합되어 코드 특성 분석 결과에 반영되었다. 이를 통해 단순 규칙 위반 수준을 넘어, 코드의 의미적 맥락을 고려한 품질 관련 정보까지 함께 제공할 수 있었다. 미탐(FN) 탐지 사례의 예시는 그림 4에 제시한다.

4.3 자동 작성자 매핑

GitHub 저장소에서 수집되는 정량·정성 분석 결과에는 동일 학습자가 서로 다른 작성자 ID(author)로 기록되는 문제가 빈번히 발생하며, 이는 개별 기여도 분석의 정확성을 저하시킨다. 본 연구는 이러한 문제를 해결하기 위해 교수자가 제공한 팀원 명단을 기준으로 한 canonical ID 기반 자동 작성자 매핑 절차를 설계하였다.

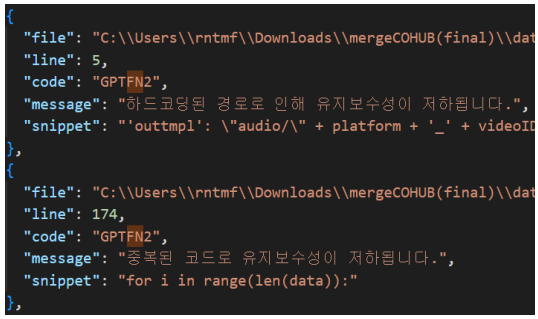


그림 4. 미탐(FN) 탐지
Fig. 4. Example for FN detection

먼저 정량 분석, 정성 평가, 정적 분석 결과에 등장하는 모든 작성자명을 수집하여 후보 목록을 구성하고, 문자열 유사도 분석, 특수문자 제거, 대소문자 정규화, 한국어 이름의 로마자 변환 규칙 등을 적용해 canonical ID와의 자동 매칭을 수행한다. 철자 변형이나 숫자 추가 등 단순 변형 사례는 규칙 기반으로 처리되며, 자동 판별이 어려운 경우에는 GPT를 활용해 한국 학생 GitHub ID의 일반적 패턴을 바탕으로 가장 적절한 canonical ID를 추론한다. 최종적으로 모든 별칭은 canonical ID별로 통합되어 저장되며, 작성자 매핑의 사례는 그림 5와 같다.

그림 5. 작성자 매핑 사례
Fig. 5. Mapping example

4.4 성실성 점수 생성

성실성 점수는 GitHub 활동 로그 기반의 정량 지표를 바탕으로 산정한다. 커밋 수, 활동 일수, 코드 변경량, PR·이슈 처리 건수 등의 변수를 포함한다. 각 변수에는 log 변환과 10 - 90 퍼센타일 기반

Robust Scaling을 적용하여 극단값의 영향을 완화하였다. 이후 모든 항목을 0~1 구간으로 정규화한 뒤 평균을 산출하고, 이를 0~100점으로 환산하여 최종 성실성 점수로 정의한다.

4.5 코드 품질 점수 생성

본 연구는 규칙 기반 정적 분석의 한계를 보완하기 위해 GPT 기반 정성 평가 모듈을 도입하여 학생별 코드 품질을 평가한다. 이 모듈은 ISO/IEC 25010 품질 모델의 실행 효율성, 유지보수성, 보안성을 중심으로 코드의 구조적 특성을 분석한다. 오탐(FP) 제거와 미탐(FN) 보완이 완료된 정적 분석 결과를 안전하게 로딩한 뒤, 작성자별 데이터를 ISO/IEC 기준에 맞춘 전용 프롬프트로 변환하여 GPT 모델(gpt-4o-mini)[27]에 입력한다. GPT는 반복 구조, 모듈성, 입력 검증 등 하위 요소를 종합적으로 해석하여 각 품질 특성에 대해 0~10점의 점수와 근거 설명을 생성한다. 비정형 응답에 대비한 자동 보정 절차를 통해 결과의 안정성을 확보하며, 최종 산출물은 JSON 형태로 저장되어 이후 기여도 분석과 최종 점수 산정에 활용된다.

예를 들어, 실행 효율성은 반복 구조·자료 구조 선택·로직 경량성을 중심으로 판단하며, 유지보수성은 모듈성·복잡도·함수 분리 여부 등을, 보안성은 입력 검증·민감정보 처리·취약한 함수 사용 여부 등을 근거로 한다. 또한 reason 생성 시 “정적 분석 근거 → 코드 구조 문제 → ISO/IEC 하위 기준 저하”의 3단 구성으로 기술하도록 강제해 설명의 일관성을 확보한다.

4.6 기능 구현 점수 생성

기능 구현 점수는 팀 리포트 및 개인 리포트 생성 결과를 기반으로 산출한다. 개인 리포트 생성 단계는 프로젝트 저장소의 전체 소스 파일을 대상으로 팀원별 실제 기여 내용을 의미 기반으로 분석하여 기능적 역할을 재구성하는 과정이다. 시스템은 저장소 내 파일을 작성자 단위로 분류한 뒤, 각 파일의 코드를 GPT 모델에 입력하여 해당 코드가 프로젝트 전체 구조에서 수행하는 기능적 책임과 설계 의도를 분석한다. 이때 프롬프트는 구현 세부 사항보다는 기능, 문제 해결 방식, 역할 중심의 의미 해석에 초점을 둔다. 파일별로 생성된 기능 요약, 강점, 개선점은 개인 리포트의 원천 데이터가 되지만, 중복 서술과 과도한 항목으로 인한 가독성 저하 문제가 발생한다. 이를 해결하기 위해 Gemini 모델을 활용한 요약·압축 단계를 추가하여 유사 항목을 통합하고 핵심 역할 중심으로 3~5개 항목으로 정제함으로써, 분석의 정확성과 서술의 가독성을 동시에 확보하였다.

팀 리포트 생성 단계에서는 개인 리포트의 원본 데이터를 종합하여 프로젝트 개요와 핵심 기능 구조를 도출하고, 기능적 범주와 모듈 간 관계를 상위 관점에서 재구성한다. 또한 개인별 기능 구현 점수는 COCOMO II[28] 기반 기술적 난이도와 MoSCoW[29] 기준 기능 중요도를 각각 평가한 뒤 평균값으로 산출하여, 단순 구현량이 아닌 역할의 질적 수준과 프로젝트 기여도를 반영하도록 설계하였다.

4.7 역할 완성도 점수 생성

팀 완성도 평가는 수행계획서에 정의된 기능 요구사항이 실제 프로젝트에서 어느 정도 구현되었는지를 의미적으로 검증하는 단계이다. 시스템은 수행계획서에서 추출한 기능 요구사항과 개인 리포트 생성 과정에서 확보된 기능 요약 데이터를 통합하여 팀 전체의 구현 기록을 구성한다. 이후 GPT는 각 기능 요구사항을 기준으로 구현 요약과의 의미적 일치 여부를 분석하여, 기능이 구현된 경우와 미구현된 경우를 구분하고, 구현된 기능의 작성자도 함께 추적한다. 팀 완성도 점수는 전체 요구사항 대비 구현된 기능의 비율로 산출되어 프로젝트의 기능 달성 수준을 정량적으로 제시한다.

개인 완성도 평가는 수행계획서에 명시된 팀원별 역할이 실제 구현 과정에서 충실히 수행되었는지를 평가하는 절차이다. GPT는 개인별 기능 요약을 바탕으로 역할의 책임과 기능적 목적이 의미적으로 구현되었는지를 분석하고, 수행 여부를 판별한다. 역할 매칭은 단순 키워드가 아닌 기능 흐름과 책임 분담을 기준으로 이루어지며, 개인 완성도 점수는 전체 역할 대비 수행된 역할의 비율을 10점 만점으

로 환산하여 산출함으로써 계획 대비 책임 수행 수준을 정량화한다. 예를 들어, 총 4개의 역할 중 3개가 구현과 일치한다고 판단되면 점수는 round((3 / 4) × 10) = 8점으로 계산된다.

4.8 최종 점수 평가

최종 점수 평가는 지금까지 산출된 성실성, 코드 품질, 기능 구현, 역할 완성도의 네 가지 평가지표를 통합하여 팀원별 종합 성과를 정량화하는 단계이다. 각 지표는 0~100점 스케일로 변환되며, 교수자가 설정한 가중치를 적용해 최종 점수를 계산한다. 성실성 점수는 GitHub 활동 로그를 기반으로 커밋 수, 활동 일수, 코드 변경량, PR·이슈 처리량 등 정량 변수를 활용하고, 로그 변환과 퍼센타일 기반 정규화를 통해 극단값의 영향을 보정한다. 코드 품질 점수는 ISO/IEC 25010 기준에 따라 GPT 기반으로 평가된 실행 효율성, 유지보수성, 보안성 점수를 평균하여 산출한다. 기능 구현 점수는 COCOMO II 기반 기술적 난이도와 MoSCoW 기준 기능 중요도를 종합하여 팀원의 기능적 기여 수준을 반영한다. 역할 완성도 점수는 수행계획서에 명시된 역할 대비 실제 구현된 기능의 일치 비율을 기준으로 계산된다. 최종 점수는 교수자가 네 영역의 가중치를 자유롭게 조정할 수 있으며(예: 25 - 25 - 25 - 25), 총합은 100을 기준으로 한다. 최종 점수는 다음 식 1의 가중합 형태로 계산된다. 식 (1)에서 S, Q, F, C는 차례대로 성실성 점수, 코드 품질 점수, 기능 구현 점수, 역할 완성도 점수이며, wS, wQ, wF, wC는 각 점수의 비중을 의미한다. 모든 영역이 동일한 0~100 스케일이므로, 가중치를 조정하면 프로젝트 특성에 따라 평가의 집중도를 달리할 수 있다.

$$final\_score = S*\frac{wS}{100} + Q*\frac{wQ}{100} + F*\frac{wF}{100} + C*\frac{wC}{100} \quad (1)$$

4.9 교수자 평가를 위한 팀 대시보드

팀 대시보드는 프로젝트 전체 수준의 개발 성과를 한눈에 파악할 수 있도록 구성된 화면으로, 정량 분석·정성 분석·완성도 평가 결과를 팀 단위로 통합하여 보여준다. 각 영역은 교수자가 실제 수업이나 평가 과정에서 필요한 정보 흐름에 따라 구조화되어 있으며, 프로젝트의 전반적 품질과 팀원별 기여도를 빠르게 확인할 수 있도록 설계되었다.

V. 시스템 평가

5.1 GPT 기반 필터링의 효용성 평가

4.2절에서 제시한 규칙 기반 1차 필터링과 GPT 기반 2차 필터링(FP 제거 및 FN 보완) 절차가 실제 코드 품질 평가에 미치는 영향을 정량적으로 파악하기 위해, GitHub 기반 프로젝트를 대상으로 5회 반복 실험을 수행하여 단계별 필터링 비율을 분석하였다.

분석 결과, 정적 분석 도구(Pylint, bandit, radon)를 통해 최초 식별된 전체 경고 중 단순 스타일 위반 등을 제외하는 규칙 기반 1차 필터링 단계에서 평균 37.7%의 비율로 경고가 사전 제거되었다. 도구별 1차 필터링 평균 제거 비율은 pylint 16.6%, bandit 68.6%, radon 69.5%로 나타나, 정적 규칙만으로도 상당수의 노이즈가 걸러짐을 확인하였다.

이어 1차 필터링을 통과한 유효 경고를 대상으로 GPT 기반 2차 오탐(FP) 필터링을 반복 수행한 결과, 검사 대상의 평균 18.5%가 오탐으로 추가 판별되어 최종 제거되었다. 도구별 2차 오탐 평균 판별 비율은 pylint 13.8%, bandit 9.1%, radon 53.8%로 분포하였다. 동시에 진행된 미탐(FN) 보완 단계에서도 규칙 기반 도구가 놓친 의미 기반 결함들이 추가로 탐지되었다. 대상 파일 전체에 대해 반복 분석을 수행한 결과, 1차 정적 필터링을 통과한 기존 유효 경고 규모의 평균 약 25.6%에 달하는 새로운 결함들이 식별되었다. 이는 4.2절에서 설계한 2차 GPT 분석이 단순 문법 검사를 넘어, 코드의 문맥과 실행 환경을 반영한 실질적인 결함 판별에 기여하고 있음을 시사한다. 다만, 본 평가는 실제 프로젝트 환경에서 후처리 파이프라인이 노이즈를 얼마나 억제하고 유의미한 결함을 추가로 찾아내는지 그 실효성을 평균 비율로 보여주는 의의를 지니나, 전문가

가 직접 구축한 정답 데이터를 기반으로 한 판별 정확도 검증이 수행되지 않았다는 한계가 존재한다. 따라서 향후 정교하게 구축된 평가 데이터셋을 활용한 성능 검증이 추가로 요구된다.

5.2 반복 실험 기반 변동성 평가

본 절에서는 제안한 시스템의 신뢰도를 검토하기 위해 반복 실험 기반 변동성 평가를 수행하였으며, 동일 학생에 대한 반복 평가 점수의 일관성을 급내 상관계수(ICC, Intraclass Correlation Coefficient)를 통해 분석하였다. ICC는 반복 측정 신뢰도를 평가하는 표준 통계 지표로, 심리학·의학·교육 평가 분야뿐 아니라 생성형 AI 평가의 신뢰도 분석에도 활용되고 있다[30]. ICC 해석은 선행연구에서 제시한 기준에 따라 0.50 미만은 낮음(Poor), 0.50 이상 0.75 미만은 보통(Moderate), 0.75 이상 0.90 미만은 우수(Good), 0.90 이상은 매우 우수(Excellent)로 구분하였다[31].

생성형 LLM의 확률적 응답 특성을 고려하여, K 대학교 GitHub 기반 캡스톤디자인 프로젝트 4개를 대상으로 프로젝트별 5회 반복 실험을 실시하였다. 모든 반복 실험은 동일한 프롬프트와 생성 설정을 유지한 상태에서 수행하였으며, 점수 일관성 평가는 성실성(S), 코드 품질(Q), 기능 구현(F), 역할 완성도(C), 최종 점수(Final)에 대해 ICC를 산출하였다. 분석 결과, 기능 구현 점수(F)는 ICC 0.78로 '우수(Good)' 수준의 신뢰도를 보였으며, 이는 정적 분석 결과와 구조화된 기능 요약을 기반으로 평가가 이루어져 LLM의 확률적 변동 영향을 상대적으로 적게 받기 때문으로 해석된다.

반면 코드 품질(Q, ICC=0.41)과 역할 완성도(C, ICC=0.33)는 '낮음(Poor)' 수준으로 나타나, 코드 문맥 및 역할 해석이 요구되는 정성 평가 영역에서 변동성이 발생함을 확인하였다. 본 연구에서는 LLM의 무작위성을 최소화하기 위해 모든 추론 과정의 생성 파라미터를 0.0으로 고정하여 실험을 진행하였다. 그럼에도 불구하고 정성 지표에서 변동성이 나타난 원인은 프롬프트 설계 요소와 입력 코드의 복잡도 구조에서 기인한 것으로 분석된다.

첫째, 프롬프트 설계 측면에서 점수 산정 구간의 모호성과 정보 필터링이 영향을 미쳤다. 최종 점수 산정 프롬프트가 구체적인 채점 기준 예시(Few-shot) 없이 COCOMO II 기준에 따라 넓은 점수 구간(예: 8~10점)을 제시하는 Zero-shot 형태로 구성되어, 미세한 점수 결정 시 일관성이 저하되었다. 또한, 초기 파일 분석 프롬프트에서 '기술적 세부 사항(API, 라이브러리 등) 배제'를 지시한 것이 결과적으로 코드 품질(Q) 판단에 필요한 핵심 근거를 누락시켜 평가의 변동성을 유발했다.

둘째, 입력 코드의 복잡도와 다단(Multi-step) 요약 파이프라인에 따른 정보 손실 문제이다. 제안 시스템은 [파일별 요약(GPT) → 작성자별 압축(Gemini) → 최종 평가(GPT)]의 체인 구조로 동작한다. 따라서 입력 코드가 길고 순환 복잡도가 높은 파일일수록 1, 2차 요약 및 압축 과정에서 맥락 누락이 발생하기 쉬우며, 이 오차가 최종 평가 단계로 전파(Error Propagation)되어 역할 완성도(C) 지표의 하락에 기여한 것으로 판단된다.

그런데도 최종 점수의 ICC는 0.71로 '보통(Moderate)' 수준을 나타내어, 안정적인 정량 지표(S)와 기능 구현 점수(F)가 정성 지표의 변동성을 완화하는 효과를 보였다. 이는 개별 정성 항목의 불안정성이 존재하더라도 종합 점수 차원에서는 일정 수준의 일관성이 확보됨을 의미한다.

순위 일관성 평가는 반복 실행 간 학생들의 상대적 순위 안정성을 검증하기 위해 Spearman 순위 상관계수와 Kendall's W를 사용하였다. Spearman ρ는 평가 회차 간 순위 일치도를 측정하는 지표로, GPT 기반 평가와 전문가 평가 간 순위 일치도를 분석한 선행연구에서도 활용되었다[30]. Kendall's W는 팀 전체 순위 합의 정도를 분석하기 위해 산출하였으며, Rovai의 해석 기준[32]과 Landis & Koch의 기준[33]을 적용하였다. 분석 결과에서 평균 Spearman ρ는 0.80, Kendall's W는 0.65로 나타나 각각 '높음' 및 '상당한 일치' 수준의 순위 안정성을 보였다. 즉, 점수의 미세한 변동에도 불구하고 학생 간 상대적 서열 판단은 통계적으로 유의미한 순위 상관성이 확인되었다.

5.3 평가 정확성 및 모델 적합성

이번 절에서는 제안된 분석법이 실제 협업 결과

와 얼마나 일치하는지를 검토하기 위해 분석 결과의 정확성을 확인하였다. 이를 위해 각 프로젝트팀이 제출한 최종 결과보고서와 Git blame을 통해 확인한 전체 소스코드 기여자 정보를 기준 데이터로 설정하고, 구현한 시스템이 생성한 분석 결과와 교차 검증을 수행하였다. 대규모 소스코드 내 실제 구현 위치를 정밀하게 확인하기 위해 상위 성능 모델(GPT-5.1)을 보조 도구로 활용하여 코드 근거를 추출하였으며, 이를 기능 명세와 대조하고 코드 검토를 통해 구현 여부를 최종 확정하였다. 또한 Gemini 모델을 비교군으로 활용하여 다중 모델 기반 하이브리드 분석 전략의 특성을 함께 검토하였다.

다만 평가 대상 시스템(GPT-4o mini)과 동일 계열의 상위 모델(GPT-5.1)을 기준 데이터 확정 과정에 보조적으로 활용한 점은 방법론적으로 순환 논증(Circular reasoning)의 가능성을 내포할 수 있다. 본 연구는 이를 초기 타당성 검증 단계에서의 실용적 접근으로 한정하며, 향후 연구에서는 교수자의 직접 코드 리뷰, 독립적 전문가 평가, 학습자 인터뷰 등 LLM에 의존하지 않는 검증 절차를 병행하여 방법론적 독립성을 강화할 필요가 있다.

기능 구현 및 기여자 식별 정확도를 분석한 결과(표 2 참조), 기능 구현 평가(F)는 구현 여부 정확도 84.0%, 기여자 식별 정확도 80.0%로 나타났다. 이는 기능 구현 점수의 반복 평가 신뢰도가 ICC 0.78로 나타난 결과와 일관되며, 코드 존재 여부와 Git 로그가 명확한 영역에서는 분석 결과의 일관성과 정확성이 비교적 안정적으로 유지됨을 보여준다.

반면 전체 정확도는 76.0%로, 복잡한 라이브러리 내부 로직의 미탐지나 코드가 없는 기능을 구현된 것으로 판단하는 환각 오류에서 한계가 확인되었다.

기능 구현 점수의 등급 정확도를 검증한 결과, 구현한 시스템은 11명 중 9명을 학점 등급 허용 오차(±10점) 이내로 분류하여 81.8%의 정확도를 보였으며, 평균 오차는 0.69점으로 나타났다(표 3 참조). 이는 기능 규모 추정과 중요도 반영 과정에서 GPT-5.1과 유사한 기준으로 COCOMO II 및 MoSCoW 평가 논리가 작동하고 있음을 시사한다. 특히 최상위 성과자를 정확히 식별하였으며, 반복 평가 평균을 통해 하위권에서도 미세한 기여도 차이를 구분하는 등 상대적 순위 변별력 측면에서 안정적인 경향이 관찰되었다.

개인 역할 이행 여부(C)에 대한 분석 결과, 수행 계획서에 명시된 26개 역할 항목을 100% 정확히 추출하여 계획 대조의 안정성을 확인하였다(표 4 참조). 역할 이행 판단 정확도는 73.1%로 나타났으며, 이는 직책 명칭과 실제 기술 구현 간 괴리로 인해 일부 과대·과소 판단이 발생한 결과로 해석된다.

그런데도 실제 코드 기여가 전혀 없는 무임승차 사례는 100% 정확히 식별되어, 협업 책임성 측면에서는 의미 있는 분석 정보를 제공함을 확인하였다.

단일 모델의 한계를 보완하기 위해 비교군 모델인 Gemini-2.5-flash를 동일한 분석 파이프라인에 적용하여 하이브리드 분석 전략의 특성을 검토하였다. 분석 결과, GPT-4o mini는 평균 오차 0.69점으로 기여도 차이에 따른 점수 변별력이 우수한 반면, Gemini-2.5-flash는 하위권 평가에 상대적으로 관대한 경향을 보여 평균 오차 0.88점을 기록하였다. 특히 완전 미구현 사례에서 GPT는 정확히 미이행으로 판단한 반면, Gemini는 수행계획서 맥락에 의존한 환각 오류를 보였다. 반대로 Gemini는 다중 기여자 탐지와 서술형 피드백 생성 측면에서 상대적 강점을 보였으며, 이를 바탕으로 본 연구는 점수 산출은 GPT가, 서술형 피드백 제공은 Gemini가 담당하는 하이브리드 분석 전략을 적용하였다. 분석 결과로부터 제안하는 시스템은 교수자가 협업 과정과 기여 양상을 다각도로 이해하고 판단하는 데 참고할 수 있는 분석 정보를 제공하는 지원 도구로서 실질적인 의미를 지닌다.

다만 본 연구는 단일 대학의 GitHub 기반 캡스톤 디자인 프로젝트 4개 팀(총 11명)을 대상으로 수행된 초기 타당성 검증 단계의 결과로 정적 분석 도구를 활용한 Python 중심 환경에 한정되어 있다. 따라서 대규모 강의, 타 교과목 유형, 또는 다른 프로그래밍 언어 기반 프로젝트로의 일반화에는 구조적 차이가 존재할 수 있다. 본 연구의 결과는 특정 교육 맥락에서의 적용 가능성을 제시하는 수준으로 해석되어야 하며, 외적 타당성 확보를 위해서는 다양한 수업 유형과 언어 환경에서의 추가 사례 축적이 필요하다.

표 2. 기능 구현 및 기여자 식별 정확도 검증 결과
Table 2. Verification results of functional implementation and contributor identification accuracy

Project	Total FR count	Implementation accuracy	Contributor identification accuracy	Final accuracy	Error analysis
A	8	87.5% (7/8)	87.5% (7/8)	87.5% (7/8)	1 Overestimation: Falsely identified unimplemented 'Recommendation Feature' as implemented.
B	5	80.0% (4/5)	100% (4/4)	80% (4/5)	Falsely identified implemented 'Obstacle Notification (FR1)' as unimplemented.
C	12	83.3% (10/12)	62.5% (5/8)	66.7% (8/12)	Misidentified contributors for FR6 and FR11. 1 Overestimation: Falsely identified unimplemented FR9 as implemented. Falsely identified implemented FR12 as unimplemented.
Total	25	84.0% (21/25)	80.0% (16/20)	76.0% (19/25)	※ Contributor Identification Accuracy is calculated using the items detected (20 items) as the denominator.

표 3. 기능 구현 점수(F)의 정확도 검증 결과 (10점 허용 오차 기준)
Table 3. Accuracy verification results for function implementation score (F) (Based on a 10-point tolerance)

Project	Member	Our system (GPT-4o mini)	GPT-5.1 (Ground truth)	Error	Result	Remarks (Rankings & special notes)
A	a	8	7.5	+0.5	Correct	Identified as tied for 1st (Actual: 2nd)
	b	8	8	0	Correct	Identified as tied for 1st (Actual: 1st)
	c	7	4.5	+2.5	Incorrect	Overestimated (Simple debugging)
B	d	9	10	-1	Correct	Accurately identified 1st in team
	e	7.2	7	+0.2	Correct	
	f	8	6	+2.0	Incorrect	Overestimated (UI assistance)
C	g	8.8	9	-0.2	Correct	Accurately identified 1st in team
	h	8.8	9	-0.2	Correct	Accurately identified tied for 1st
	i	8	8	0	Correct	
	j	7.4	7	+0.4	Correct	Successful discrimination of lower ranks
	k	6.4	7	-0.6	Correct	
Total	11	Mean error: 0.69			Score accuracy: 81.8%	

표 4. 개인 역할 이행(C) 식별 정확도 검증 결과
Table 4. Accuracy verification results for individual role fulfillment (C) identification

Project	Member	No. of roles	Our system (GPT-4o mini) Judgment (O/X)	GPT-5.1 ground truth (O/X)	Match	Remarks (Cause of error)
A	a	4	O / O / O / X	O / O / O / X	4/4	
	b	3	O / O / O	X / O / X	1/3	Overestimation: Falsely judged unimplemented AWS/DB setup as fulfilled.
	c	2	O / X	O / X	2/2	
B	d	2	O / O	O / O	2/2	
	e	3	O / O / X	X / X / X	1/3	Overestimation: Failure to implement core logic was masked by leadership role/minor contributions.
	f	2	O / X	O / X	2/2	
C	g	2	O / O	O / O	2/2	
	h	2	X / X	X / X	2/2	
	i	2	X / O	O / O	1/2	Underestimation: Audio feature extraction was implemented but falsely judged as unfulfilled.
	j	2	X / X	O / X	1/2	Underestimation: Chat feature extraction was implemented but falsely judged as unfulfilled.
	k	2	O / X	O / O	1/2	Underestimation: Contributed to unit testing but falsely judged as partial non-fulfillment.
Total	11	26				Role Fulfillment Identification Accuracy: 73.1% (19/26)

Ⅵ. 결 론

본 연구는 팀 기반 프로그래밍 교육에서 지속적으로 제기되어 온 개별 기여도 평가의 한계를 극복하기 위해, GitHub 협업 로그와 생성형 AI 기반 코드 의미 분석을 통합한 평가 지원 시스템을 설계·구현하고 그 실효성을 실증적으로 검증하였다. 기존의 결과물 중심 평가나 교수자의 수작업 검토 방식이 협업 과정의 실제 기여 양상을 충분히 반영하지 못한다는 문제의식에서 출발하여, 본 연구는 협업 로그를 개인 단위의 정량 지표로 재구성하고, 대규모 언어 모델(LLM)을 활용해 코드 품질과 역할 이행 수준을 정성적으로 분석하는 이중 구조의 통합 프레임워크를 제안하였다.

본 연구는 협업 평가를 산출물 확인 중심 접근에서 데이터 기반 과정 분석 중심 접근으로 확장함으로써, 팀 프로젝트 평가의 개념적 틀을 재정립하였다. 특히 수행계획서 기반 역할-기능 매핑과 소프트웨어 품질 특성을 반영한 분석 구조를 도입함으로써, 협업 학습 평가에 공학적 품질 기준과 교육적 책무성 개념을 통합한 점은 중요한 학문적 기여라 할 수 있다. 또한 GitHub API를 활용한 자동 수집 및 분석 체계를 통해 정량 지표의 안정성과 재현 가능성을 확인하였으며, 무임승차 사례를 명확히 식별함으로써 데이터 기반 기여도 분석의 실효성을 입증하였다. GPT와 Gemini 모델의 비교 분석을 통해 점수 산출과 서술적 피드백 생성의 상이한 특성을 확인하고, 이를 결합한 하이브리드 분석 전략을 제안하였다. 이는 생성형 AI를 단일 평가 주체로 사용하는 접근을 지양하고, 상호 보완적 분석 도구로 구조화해야 함을 보여주는 실증적 근거를 제공한다.

한편, 정성 평가 영역에서 관찰된 상대적으로 낮은 신뢰도는 생성형 AI의 확률적 생성 특성에 기인하며, 이는 AI를 최종 판정자가 아닌 교수자의 전문적 판단을 지원하는 의사결정 보조 도구로 활용해야 한다는 본 연구의 설계 원칙을 뒷받침한다. 이러한 결과는 AI 기반 평가 자동화에 대한 과도한 기대를 경계하면서도, 데이터 기반 해석 정보를 병렬적으로 제공함으로써 교수자의 평가 부담을 실질적으로 경감할 수 있음을 시사한다. 향후 연구에서는 정성 평가 신뢰도 향상을 위한 다단계 분석 파이프라인 고도화와 협업 상호작용 구조를 반영한 정밀 기여도 모델 개발을 통해 본 접근의 일반화 가능성과 확장성을 검증할 필요가 있다.

References

[1] Ministry of Education, 2022 revised national curriculum, Ministry of Education, Republic of Korea, Dec. 2022.

[2] S. Faja, "Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses", Information Systems Education Journal, Vol. 12, No. 6, pp. 36-45, Nov. 2014.

[3] K. Y. Lim, "Self-efficacy in Group Investigation Collaborative Learning", Theory and Practice of Education Vol. 16, No. 2, pp. 19-36, Aug. 2011. UCI : G704-001913.2011.16.2.005.

[4] J. A. Mello, "Improving Individual Member Accountability in Small Work Group Settings", Journal of Management Education, Vol. 17, No. 2, pp. 253-259, May 1993. http://doi.org/10.1177/105256299301700210.

[5] C. M. Brooks and J. L. Ammons, "Free Riding in Group Projects and the Effects of Timing, Frequency, and Specificity of Criteria in Peer Assessments", Journal of Education for Business, Vol. 78, No. 5, pp. 268-272, Mar. 2003. https://doi.org/10.1080/08832320309598613.

[6] S. K. Kim, "Teaching Method Using Job Assignment as a Solution on the Adverse Effects of Peer Evaluation in Team-Based Learning", Journal of the Korea Academia-Industrial Cooperation Society, Vol. 12, No. 6, pp. 2543-2547, Jun. 2011. https://doi.org/10.5762/KAIS.2011.12.6.2543.

[7] M. Kim, "Perceptions of Peer Assessment and the Effectiveness of Peer Assessment Based on the Specificity of Assessment criteria in Team

Contribution", Educational Research, Vol. 46, No. 3, pp. 79-101, Nov. 2024. https://doi.org/10.35510/JER.2024.46.3.79.

[8] H. J. Ju and H. Park, "Analysis of Medical Students' Experiences with Peer Assessment Using Text Mining", The Journal of Humanities and Social Sciences, Vol. 25, No. 4, pp. 521-542, 2024. https://doi.org/10.15818/ihss.2024.25.4.521.

[9] D. Lee, "The Role of Peer Evaluation in Online Collaborative Learning Environments: Promoting Individual Accountability by Experiencing the Roles of Evaluator and Evaluatee", Journal of Lifelong Learning Society, Vol. 9, No. 2, pp. 181-211, May 2013. UCI : G704-SER000015054. 2013.9.2.003.

[10] https://github.com. [asseessed: Sep. 02, 2025]

[11] M. Kang, "A Study on Quantitative Evaluation Methods for Collaborative Programming Tasks Based on Integration with Configuration Management Tools", Master's Thesis, Yonsei University, Aug. 2017.

[12] J. Mira and A. Kirsh, "Analyzing Individual Contribution in Team-Based Software Engineering Projects", 2024 ASEE Annual Conference & Exposition, Portland, Oregon, USA, Jun. 2024.

[13] C. Park and J. Hyun, "A ChatGPT-Based Individual Student Assessment Method Through Analysis of Collaborative Programming Outcomes Using GitHub", Proc. of the Korea Contents Association Conference, Jeju, Korea, pp. 197-198, May 2025.

[14] H. J. Jung, "A Software Quality Evaluation Model and Test Case Proposal Based on ISO/IEC 25010", Journal of the Korea Institute of Information Technology, Vol. 9, No. 10, pp. 197-205, Oct. 2011. UCI : G704-001947.2011.9.10.007.

[15] https://chat.openai.com. [asseessed: Sep. 02, 2025]

[16] J. Park, Design and Implementation of Software for Performance Assessment in Programming Education Using Subversion, Master's Thesis, Ewha Womans University, Feb. 2018.

[17] J. Kim, "Fair Assessment Method Reflecting Individual Competence in Capstone Design Courses", Journal of Engineering Education Research", Vol. 22, No. 2, pp. 36-45, Apr. 2019. https://doi.org/10.18108/jeer.2019.22.2.36.

[18] S. Cho, "Characteristics and Utilization of Peer-evaluation and Self-evaluation of Team Activities in University Project Based Classes", Journal of Engineering Education Research, Vol. 25, No. 1, pp. 55-64, Feb. 2022. https://doi.org/10.18108/jeer.2022.25.1.55.

[19] R. Kim, J. Lee, J. Lee, S. Kim, and E. Nam, "Proposal for a Conflict Resolution and Evaluation Service for Team Project Learning - Focusing on University Students and Professors -", Korea Design Forum, Vol. 29, No. 3, pp. 239-248, Nov. 2024. https://doi.org/10.21326/ksdt.2024.29.3.020.

[20] T. J. Seong, K. Si, and Y. J. Choi, "A Paradigm Shift and the Future Direction of Educational Assessment in the Era of Generative AI", Journal of Educational Evaluation, Vol. 37, No. 1, pp. 1-28, Mar. 2024. https://doi.org/10.31158/JEEV.2024.37.1.1.

[21] J. Zhao, E. Chapman, and P. G. P. Sabet, "Generative AI and educational assessments: A systematic review", Education Research and Perspectives Vol. 51, pp. 124-155, Dec. 2024. https://doi.org/10.70953/ERPv51.2412006.

[22] U. M. Borghoff, M. Minas, and J. Schopp, "Generative AI in Student Software Development Projects: A User Study on Experiences and Self-Assessment", Proc. of the 6th European Conference on Software Engineering Education, Seeon, Germany, pp. 161-170, Jun. 2025.

[23] M. Verdicchio, "Adapting Program Assessment for the Age of Generative AI", 2025 IEEE Engineering Education World Conference (EDUNINE), Montevideo, Uruguay, pp. 1-6, Mar. 2025.

[24] https://pypi.org/project/pylint. [asseessed: Sep. 02, 2025]

[25] L. D. Martini, D. d'Abate, A. Margara, and G. Cugola, "Radon: A Programming Model and Platform for Computing Continuum Systems", 2025 IEEE 45th International Conference on Distributed Computing Systems Workshops (ICDCSW), Glasgow, United Kingdom, pp. 588-593, Jul. 2025.

[26] E. Y. Kim, Exploring the Python ecosystem together, https://wikidocs.net/book/14021. [asseessed: Sep. 02, 2025]

[27] https://platform.openai.com/docs/models/gpt-4o-mini. [asseessed: Sep. 02, 2025]

[28] J. Baik, "COCOMO II, Model Definition manual, version 2.1.", Center for Software Engineering at the University of Southern California, Vol. 18, pp. 45-49, 2000.

[29] S. Vijayakumar, "Assessing the Effectiveness of MoSCoW Prioritization in Software Development: A Holistic Analysis across Methodologies", EAI Endorsed Transactions on Internet of Things, Vol. 10, Oct. 2024. https://doi.org/10.4108/eetiot.6515.

[30] H. Lee, J. Jung, and S. Kim, "GPT-based Creativity Assessment: Focusing on Comparison with Human Experts", Journal of Creativity Education Research, Vol. 25, No. 3, pp. 1-24, Sep. 2025. https://doi.org/10.36358/JCE.2025.25.3.1.

[31] T. K. Koo and M. Y. Li, "A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research", Journal of Chiropractic Medicine, Vol. 15, No. 2, pp. 155-163, Jun. 2016. https://doi.org/10.1016/j.jcm.2016.02.012.

[32] A. P. Rovai, J. D. Baker, and M. K. Ponton, "Social science research design and statistics: A practitioner's guide to research methods and IBM SPSS Analysis", Watertree Press, Sep. 2013.

[33] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data", Biometrics, Vol. 33, No. 1, pp. 159-174, Mar. 1977. https://doi.org/10.2307/2529310.

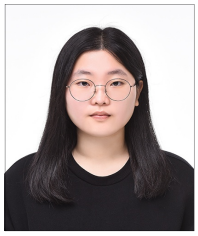
저자소개

김 유 안 (Yuan Kim)

2023년 3월 ~ 현재 : 제주대학교 컴퓨터교육과 학사과정

관심분야 : 에듀테크, 소프트웨어 교육

김 난 주 (Nanju Kim)



2023년 3월 ~ 현재 : 제주대학교 컴퓨터교육과 학사과정

관심분야 : 에듀테크, 인공지능

박 찬 정 (Chan Jung Park)

1988년 2월 : 서강대학교 전자계산학과(공학사)

1990년 2월 : KAIST 전산학과 (공학석사)

1998년 2월 :서강대학교 대학원 전자계산학과(공학박사)

1990년 3월 ~ 1994년 2월 : 한국통신 소프트웨어연구소 전임연구원

1998년 2월 ~ 1999년 9월 : 한국통신 멀티미디어연구소 전임연구원

2011년 1월 ~ 2011년 12월 : UC Berkeley 방문학자

2013년 4월 ~ 2015년 2월 : 제주대학교 교육과학연구소 소장

1999년 9월 ~ 현재 : 제주대학교 컴퓨터교육과 교수

관심분야 : 기술기반 교육, 에듀테크, 데이터마이닝, 생성형 점진적 프롬프팅 기반 교수학습 설계