논문 2026-2-3 http://dx.doi.org/10.29056/jsf.2026.06.03

인공지능 시대의 소프트웨어 감정 패러다임 전환

윤영선*†

A Paradigm Shift of Software Appraisal in the Age of Artificial Intelligence

Young-Sun Yun*†

요 약

본 논문에서는 대규모 언어모델(LLM)과 에이전트 기반 시스템의 확산이 소프트웨어 개발 방식과 소프트웨어 감정 체계에 미치는 영향을 분석하고 인공지능 시대에 적합한 새로운 감정 패러다임을 제안한다. 전통적 소프트웨어 개발이 인간 개발자의 명시적 설계와 구현, 결과물 중심의 평가에 기반해 왔으나, 최근의 개발 환경은 자연어 기반 요구 명세, AI 지원 코드 생성, 계획·실행·검증의 반복 구조를 특징으로 하는 AI 협업형 개발로 빠르게 전환되고 있다. 이에 따라 소프트웨어 감정 또한 완성된 소스코드나 기능 결과물만이 아니라 학습 데이터, 모델 구조와 가중치, 프롬프트 및 생성 이력, 인간-AI 협업 과정의 추적 가능성, 외부 API 및 운영 파이프라인을 포함하는 방향으로 확장될 필요가 있다. 이를 위하여 AI 활용 형태를 'of the AI, by the AI, for the AI'의 관점에서 체계화하고, AI 모델 자체, AI에 의해 생성된 산출물과 과정, AI가 작동하는 생태계를 포괄하는 감정 프레임워크의 필요성을 제시하였다.

Abstract

This paper examines the impact of the rapid spread of large language models (LLMs) and agent-based systems on software development methodologies and software appraisal systems, and proposes a new appraisal paradigm for the age of artificial intelligence. As software development shifts from explicit human design and implementation toward AI-collaborative development characterized by natural language-based requirement specification, AI-assisted code generation, and iterative cycles of planning, execution, and verification, software appraisal must also expand beyond completed source code and functional deliverables. It should encompass training data, model architectures and weights, prompts and generation histories, the traceability of human-AI collaboration, and external APIs and operational pipelines. From this perspective, this paper systematizes forms of AI utilization as "of the AI, by the AI, for the AI" and highlights the need for an appraisal framework covering AI models, AI-generated outputs and processes, and the ecosystem in which AI operates.

한글키워드 : 바이브 코딩, 인공지능-협업 SW 개발, 소프트웨어 감정 패러다임 전환

keywords : vibe coding, AI-collaborated SW development, SW appraisal paradigm shift

1. 서 론

인공지능 기술, 특히 대규모 언어모델(LLM)과 에이전트 기반 시스템의 발전은 소프트웨어 개발

* 한남대학교 정보통신공학과
† 교신저자: 윤영선(email: ysyun@hnu.kr)
접수일자: 2026.05.24. 심사완료: 2026.06.16.
게재확정: 2026.06.20.

환경에 큰 변화를 불러왔다. 전통적인 소프트웨어 개발인 폭포수(waterfall) 모델은 요구사항 분석, 설계, 구현, 시험, 배포의 단계적인 구조를 바탕으로 순차적 절차와 문서화를 강조하였으며, 애자일(agile)과 데브옵스(DevOps)는 반복적·점진적 개발, 신속한 피드백, 운영 자동화를 강조하였다[1].

이러한 개발 방법은 생성형 AI가 급속히 발달함에 따라 빠르게 변화하고 있다. 특히 LLM 기반 도구는 코드 초안 생성, 리팩터링, 테스트 케이스 작성, 문서화, 디버깅 지원, 설계 대안 제시 등 개발 생애주기 전반에 관여하며, 그 결과 AI는 더 이상 보조적 자동화 도구에 머무르지 않고 실질적인 협업 주체가 되었다.

전통적인 SW 개발 방법론에서 개발자는 요구사항을 명확히 기술한 후, 설계와 구현을 거쳐 테스트를 수행하고 결과물을 배포하는 역할을 중심으로 활동해 왔다. 이러한 방식은 절차의 추적 가능성과 통제 가능성이 높다는 장점이 있지만, 요구사항 변경에 대한 적응 속도가 느리고 반복 작업에 많은 시간과 비용이 소요된다는 한계를 보인다. 반면 최근의 SW 개발은 애자일과 데브옵스의 기반 위에 AI 에이전트(agent), 계획(plan), 대화(conversation) 중심의 상호작용이 결합된 형태로 전환되고 있다. 개발 현장에서는 코드 생성, 테스트 설계, 문서화뿐 아니라 배포와 모니터링 단계에 이르기까지 AI가 적극적으로 개입하고 있으며, 이는 생산성 향상을 넘어 SW 개발 패러다임 자체가 변화하고 있음을 시사한다. 이에 따라 개발자의 역할 역시 직접 구현 주체에서 문제를 구조화하고, 인공지능이 생성한 결과를 검토·수정·통합하는 조정자나 감독자로 변모하고 있다[2,3].

최근의 SW 개발은 AI가 개발 전 과정에 실질적으로 개입하는 인간-AI 협업형(human-AI collaborative) 개발로 이해할 수 있다. 특히 간단한 프롬프트 기반 상호작용을 넘어, 목표 설정, 계획 수립, 실행, 검증을 연속적 순환 구조로 수행하는 에이전틱(Agentic) AI 개발 방식이 확산되는 추세다.

표 1. 전통적인 개발과 AI 협업 개발의 비교
Table 1. Comparison of traditional development and AI-assisted development

구분	전통적 개발	AI 협업 개발
개발	개발자 직접 구현	개발자-AI 협업
체계	단계적, 선형적, 문서 중심	반복적, 대화형, 실험 중심
산출물	순차적 생산	동시적 생산
개발자 역할	구현자, 작성자	문제 정의자, 검토자, 통합자
위험	변경 지연, 수작업 비용	환각, 품질 편차, 보안, 저작권 등

이와 함께 노코드/로우코드(no-/low-code) 문화, 조립형 개발 방식, 개발·운영 통합 패러다임의 확장은 AI 기반의 새로운 SW 생태계를 형성하고 있다.

그렇지만 SW 감정(SW appraisal)은 여전히 전통적인 개발 패러다임에 많은 부분을 의존하고 있다. 기존의 SW 감정은 주로 소스코드 유사성 판단, 기능·성능·신뢰성 등의 품질 평가, 테스트 결과 검증 등 완성된 산출물을 대상으로 수행되었다. 그러나 AI가 개발 도구이자 협업자이며, 더 나아가 소프트웨어의 핵심 구성요소가 되는 환경에서는 감정의 대상 역시 확장될 필요가 있다. 예를 들어 데이터 수집 및 전처리 과정, 모델 구조와 학습 절차, 생성된 모델 가중치, 에이전트·플래너·스킬(skill) 구조, 개발 및 운영 파이프라인은 중요한 기술 자산이 될 수 있다. 따라서 AI 시대의 SW 감정은 완성된 코드 결과물의 평가를 넘어, 개발 과정과 실행 구조, 학습 자원, 운영 환경을 함께 다루는 방향으로 확장되어야 한다.

본 논문에서는 거버넌스 관점에서 인용한 “of the AI, by the AI, for the AI”의 개념[4]을 SW

개발 패러다임 관점에서 검토하고, 이에 따른 SW 감정 핵심 요소를 도출하고자 한다. 2장에서는 AI 시대 SW 개발 패러다임의 변화를 기술하고, 3장에서 “of/by/for the AI” 관점에 기반한 소프트웨어 감정 패러다임의 개념적 프레임워크를 제안하며, 마지막으로 요약 및 결론을 맺는다.

2. SW 개발 패러다임의 변화

기술의 발전은 SW 개발 환경 전반에 지속적인 변화를 불러왔다. 이러한 변화는 단순한 사용자 인터페이스나 개발 도구 외에도 개발 방법론을 변화시켰다. SW 개발 패러다임의 변화는 기술적 진보와 함께 세부 구현에 대한 부담과 개발의 복잡도를 줄여 개발 주체의 확대를 가져왔다.

2.1 SW 개발 방법론의 변화

SW 개발 방법론의 변화는 개발 방식의 변화와 개발 주체의 확대라는 측면에서 이해할 수 있으며, SW 개발 패러다임은 시대적으로 다음의 네 단계로 구분될 수 있다[5].

첫째, 전통적 프로그래밍(1950년대~1990년대)은 전문 지식 중심의 개발 패러다임으로 설명할 수 있다. 해당 시기의 개발은 하드웨어 구조와 시스템 동작 원리에 대한 깊은 이해를 요구하였고, 기계 의존적 세부 사항을 고려해야 하는 고도의 전문성이 필수적이었다. 또한 절차 지향적 사고와 객체지향 방법론이 주요 개발 방식으로 정착되었으며, 분석, 설계, 구현, 시험의 단계가 엄격히 구분되는 장기적 개발 프로세스가 일반적이었다.

둘째, 비주얼 프로그래밍(1990년대~2000년대)은 직관적 인터페이스 중심의 개발 방식으로 이해할 수 있다. Drag & Drop 기반의 사용자 인터페이스와 GUI 중심의 통합개발환경(IDE)이 보급되면서 개발 생산성과 사용 편의성이 향상되었다. 이 방식 역시 기존의 코딩 논리와 프로그래밍 구조를 기반으로 하였기 때문에, 일정 수준의 개발 개념과 논리에 대한 이해가 필요했다.

셋째, 노코드/로우코드 개발(2000년대~2020년대)은 비즈니스 전문가의 개발 참여가 본격적으로 가능한 단계로 볼 수 있다. 시각적 모델링 도구, 사전 구축된 템플릿, 그리고 다양한 통합 프레임워크가 등장하면서 복잡한 코딩 과정을 최소화하고 개발 생산성을 크게 높이는 환경이 조성되었다. 특히 전문 개발 지식이 부족한 현업 실무자나 비즈니스 전문가도 현업에 적용 가능한 애플리케이션을 직접 설계하고 구현할 수 있게 되면서, 개발 주체의 범위가 전통적 개발자 집단을 넘어 확대되었다. 노코드/로우코드 패러다임은 기술 전문성 중심의 개발 구조를 사용자 중심 구조로 전환하는 중요한 전환점으로 볼 수 있다.

넷째, 생성형 AI 기반 개발(2023년~현재)은 의도 기반 프로그래밍(intent-driven programming)으로 설명할 수 있다. 사용자는 자연어로 자신의 요구사항과 의도를 제시하고 AI는 이를 코드나 시스템 구조, 기능 구현 결과물의 형식으로 생성한다. 이른바 ‘바이브 코딩(vibe coding)’[6]으로 불리는 개발 방식은 프로그래밍 문법 자체보다 의도 표현과 문제정의의 중요성을 높이고 있다. 즉, 개발 방식이 ‘어떻게 구현할 것인가(How)’에서 ‘무엇을 만들 것인가(What)’로 이동하고 있음을 의미한다. 생성형 AI 기반 개발 방식은 SW 개발을 기술적 실행 중심에서 의도 표현과 문제정의 관점으로 접근하는 패러다임 전환으로 볼 수 있다.

요약하면 SW 개발 방법론은 <표 2>에서 정리한 바와 같이 전통적 프로그래밍에서 비주얼 프로그래밍, 노코드/로우코드 개발, 그리고 생성형 AI 기반 개발에 이르기까지 기술 전문성은 낮추고 개발 편의성을 높이며 개발 참여의 문턱을 낮추는 방향으로 진화해 왔다[3]. AI 기반 개발

방식의 생산지수는 정량적 생산지표가 확인되지 않아 해당 항목은 '-'로 표기하였다. 생성형 AI 기술의 고도화는 SW 개발의 자동화 수준을 더 높일 것으로 예상되며, 이에 따라 개발자의 역할 역시 직접 구현자에서 문제정의자, 설계자, 그리고 AI 협업 관리자로 변화할 가능성이 크다.

표 2. SW 개발 방법론의 비교[3]
Table 2. Comparison of SW development methodologies[3]

구분	전통적 방법	노코드/ 로우코드	AI 기반
개발기간	주/월	일/주	시간/일
기술 전문성	높음	중간	낮음
기술 접근성	낮음	중간	높음
개발비용	높음	중간	낮음
생산지수[7]	기준(1x)	3x~10x	-
유지보수 복잡도	높음	중간	낮음

2.2 AI 기반 개발 환경 분석

현재 AI는 SW 개발의 도구적 보조 수준을 넘어 개발 대상, 개발 주체, 개발 생태계 전반을 재구성하는 방향으로 발전하고 있다. LLM, 에이전트 기반 시스템, 자동화된 운영 파이프라인이 결합되면서 SW는 단순한 코드 산출물이 아니라 데이터, 모델, 실행 환경, 오케스트레이션 규칙이 함께 작동하는 복합 시스템으로 변화하고 있다. 본 절에서는 이러한 변화를 "of the AI, by the AI, for the AI"의 AI 중심 관점에서 살펴보고자 한다.

(가) Of the AI : AI 중심 SW

"Of the AI"는 AI가 SW 핵심 대상 또는 구성요소가 되는 경우를 의미한다. 기존 SW가 입력과 규칙 기반의 결정적(deterministic) 처리 방식으로 동작한다면, AI 중심 SW는 학습 데이터의 품질, 모델의 일반화 성능, 추론 시점의 맥락, 그리고 운영 중에 발생하는 누적 오차에 따라 성능과 결과가 달라질 수 있다. 따라서 "of the AI"는 소스코드 자체보다도 학습 데이터 세트, 모델 구조, 가중치 파일, 실험 이력, 평가 지표, 추론 로그 등이 관리 대상이 될 수 있다.

이러한 특성은 개발 및 관리 방식에도 영향을 줄 수 있다. 전통적인 형상 관리와 테스트 체계만으로는 AI 중심 SW를 충분히 설명하기 어렵다. 예를 들어 동일한 소스코드를 사용하더라도 학습 데이터의 변경이나 외부 사전 학습 모델의 교체만으로 시스템의 동작 방식이 달라질 수 있으므로 성능, 안전성, 설명 가능성, 편향성, 법적 적합성 등을 함께 검토할 필요가 있다. 따라서 "of the AI"는 AI가 단순한 기능 요소가 아니라 SW를 구성하는 본질적 요소가 되는 환경이며, 개발의 중심이 코드 구현에서 모델 중심의 시스템 개발로 이동하고 있음을 의미한다.

(나) By the AI : AI 주도 개발

"By the AI"는 AI가 개발 과정에 실질적으로 이용되는 경우를 의미한다. 이 환경에서 AI는 자동완성 기능을 제공하는 수준을 넘어, 요구사항을 바탕으로 코드 초안을 생성하고, 테스트 케이스를 작성하며, 버그 수정과 리팩터링을 제안하고, 문서를 작성하거나 배포 스크립트를 구성하는 등 개발 생애주기 전체에서 사용된다.

이 경우 개발자의 역할도 달라진다. 개발자는 모든 코드를 직접 구현하기보다, 문제를 정의하고 생성된 결과를 검토하며 품질과 정합성을 판단하는 데 집중한다. 특히 에이전트형 개발 환경에서는 목표 설정, 계획 수립, 작업 분해, 도구 호출, 결과 검증, 재시도와 같은 단계가 AI에 의해 반복적으로 수행될 수 있다. 이와 같은 환경에서는 개발 과정이 일회성 명령 및 수행 구조가 아니라, 계획과 실행, 검증이 반복되는 구조로 바뀌

게 된다.

그러나 “by the AI” 방식은 생산성 향상과 함께 새로운 위험을 수반한다. 대표적으로 환각에 의한 잘못된 코드 생성, 취약한 라이브러리의 무비판적 사용, 테스트 커버리지의 외형적 증가에도 불구하고 실질적 품질 저하, 생성물의 저작권 및 책임 귀속 문제 등이 발생할 수 있다. AI가 생성한 결과물이 다시 후속 AI의 입력으로 사용되는 경우, 오류가 누적·증폭되는 구조가 형성될 수 있다. 따라서 이 환경에서 핵심은 AI의 코드 생성 능력 자체가 아니라, 생성 과정과 결과물에 대한 인간의 검토 체계, 검증 기준, 추적 가능성 확보에 있다.

(다) For the AI : AI 작동 생태계

“For the AI”는 AI가 SW의 중심이거나 개발 주체에 머무르지 않고, 다른 AI·도구·모델·서비스를 활용하면서 작동하는 생태계를 의미한다. 즉, AI가 단독으로 작동하기보다 외부 API, 모델 허브, 벡터 데이터베이스, 플러그인, 에이전트 프레임워크, 보안 통제 시스템, 운영 자동화 도구와 연계하여 작동하며, 이 전체 구성이 곧 개발 및 실행 환경을 구성한다. 이러한 관점에서 SW는 개별 산출물이 아니라 상호의존적 구성요소들의 결합체로 이해되어야 한다.

이러한 생태계에서는 데브옵스 등의 이용과 확산으로 개발과 운영의 경계가 점차 모호해지고 있다. 특히 AI/ML 환경에서는 학습과 배포의 반복, 모델 재훈련, 성능 모니터링, 편향 및 안전성 검토, 보안 업데이트가 지속적으로 이루어지기 때문에, 운영 환경 자체가 하나의 개발 영역이 된다.

결국 “for the AI” 환경에서는 기능 품질뿐 아니라 연결성, 상호운용성, 공급망 신뢰성, 접근권한 통제, 로그와 메타데이터의 추적성, 모델·도구·플러그인의 출처와 무결성 등이 중요한 검토 대상이 된다. 따라서 AI 기반 개발 환경을 이해하기 위해서는 코드와 개발자만이 아니라, AI가 작동하고 소비하는 전체 생태계를 함께 고려해야 한다.

3. AI 시대의 SW 감정 패러다임

AI 기술이 SW 개발의 대상, 주체, 환경을 동시에 변화시키고 있다는 점을 고려할 때, 소프트웨어 감정 역시 기존의 결과물 중심 평가 패러다임에서 벗어나 보다 확장된 관점으로 재구성될 필요가 있다. 전통적인 SW 감정은 주로 소스코드 유사성 판단, 기능 구현 여부, 품질 속성, 테스트 결과, 계약상 산출물 충족 여부 등을 중심으로 수행되었다. 그러나 AI 시대에는 SW가 데이터와 모델, 파이프라인, 에이전트, 외부 서비스 연계 등을 포함하는 복합 시스템으로 변화하였으며, 개발 과정 또한 인간과 AI의 협업, 나아가 AI 주도의 순환적 생성 구조로 전환되고 있다. 따라서 SW 감정은 더 이상 완성된 코드 산출물만을 대상으로 할 수 없고, AI 자체, AI가 개입한 생산 과정, AI가 작동하는 생태계 전반을 포괄하는 방향으로 확장되어야 한다.

본 장에서는 이러한 문제의식에 따라 “of the AI, by the AI, for the AI”의 관점에서 AI 시대 SW 감정 패러다임을 정리하고자 한다.

3.1 Of the AI : 모델 및 시스템의 감정

“Of the AI” 관점에서의 SW 감정은 AI를 구성하는 모델 및 시스템 자체를 감정 대상으로 삼는다. 이는 전통적인 SW의 기능 구현 여부를 평가하는 수준을 넘어, AI 모델이 어떠한 데이터와 구조를 바탕으로 학습되었는지, 성능과 안전성이 어떠한 조건에서 확보되는지, 그리고 결과의 재현성과 신뢰성이 어느 정도 보장되는지를 종합적으로 평가하는 접근이라 할 수 있다.

이 경우 감정 대상은 소스코드에 한정되지 않는다. 학습 데이터의 수집 경위와 적법성[8], 전처리 과정의 적정성, 데이터 편향 여부, 라벨링의 신뢰도 등 데이터 계층 자체가 중요한 검토 대상이 된다. 다음으로 모델 구조와 학습 절차, 사용된 사전 학습 모델의 출처, 하이퍼파라미터 설정, 평가 방식, 모델 가중치와 버전 이력, 추론 로그 및 메타데이터 등이 모두 감정의 범주에 포함될 수 있다[9]. 이는 ML 시스템에서 데이터 세트, 모델, 배포 패키지, 로그, 파이프라인 설정이 모두 중요한 보호 및 관리 대상이 된다는 MLSecOps 관점과도 부합한다[10].

나아가 AI 모델 및 시스템의 감정은 단순 정확도 평가를 넘어 설명 가능성, 강건성, 편향성, 보안성, 법적 적합성 등을 함께 다루어야 한다. 동일한 기능을 수행하는 두 AI 시스템이라 하더라도, 한쪽은 특정 데이터 편향을 내포하거나 적대적 입력에 취약할 수 있으며, 다른 한쪽은 학습 데이터의 출처가 불명확하여 저작권 또는 개인정보 침해가 발생할 수 있다. 따라서 “of the AI”에 대한 감정은 성능 수치 중심 평가가 아니라, 데이터-모델-운영의 연계 구조 속에서 신뢰 가능성을 다층적으로 검토하는 방식을 고려해야 한다.

결국 이 영역의 SW 감정은 기존의 프로그램 유사도·완성도 판단을 넘어, AI 모델의 생성 경로와 작동 특성을 포함한 구조적 감정으로 확장될 필요가 있다. 이는 AI가 단순한 프로그램 부품이 아니라 독립적 판단과 생성 기능을 수행하는 시스템 구성요소가 되었기 때문이다.

3.2 By the AI : 과정 및 결과물의 감정

“By the AI” 관점에서의 SW 감정은 AI가 개발 과정에 실질적으로 이용된 경우를 대상으로 한다. 여기에서는 결과물의 품질만이 아니라 (결과 감정), 그러한 결과물이 어떠한 입력과 절차를 거쳐 생성되었는지 (과정 감정)도 함께 검토하여야 한다. 이는 인간 개발자가 직접 작성한 산출물을 전제로 하던 기존 감정과 구별되는 부분이다.

우선 과정 감정에서 핵심적 요소는 개발의 추적 가능성(traceability)이다. 어떤 요구사항 또는 프롬프트가 주어졌고, 어떠한 AI 도구와 모델이 사용되었으며, 생성된 결과가 어떠한 검토·수정·승인 절차를 거쳤는지에 대한 기록이 확보되어야 한다. 특히 에이전트 기반 개발에서는 계획 수립, 도구 호출, 외부 지식 참조, 결과 재생성 등이 다단계로 수행되므로, 단일 결과물만으로는 책임 귀속과 적정성을 판단하는 데 한계가 존재한다. 따라서 프롬프트 이력, 생성 로그, 버전 변경 기록, 개발자 검토 내역, 테스트 결과의 연계 기록 등이 감정의 핵심 증거로 사용될 수 있다.

다음으로 결과물 감정의 측면에서는 AI 생성 산출물의 기능적 완성도뿐 아니라 생성 특유의 위험 요인을 함께 검토해야 한다. 예를 들어 AI가 작성한 코드가 외형적으로는 작동하더라도, 보안 취약점이 내재되어 있거나, 특정 오픈소스 코드와의 실질적 유사성이 존재하거나, 테스트 코드가 형식적으로만 작성되어 실제 품질을 보장하지 못하는 경우가 발생할 수 있다. DORA 2025 보고서가 지적하듯이 AI는 조직의 역량을 자동으로 개선하기보다는 기존의 품질관리 수준과 협업 구조를 증폭시키는 경향이 있으므로, 결과물의 품질은 AI 사용 여부 자체보다 관리 체계의 성숙도에 의해 좌우된다[11].

따라서 “by the AI”에 대한 SW 감정은 단순히 “누가 작성했는가?”의 문제가 아니라, “어떠한 인간-AI 협업 구조 속에서 생성되었으며 그 과정이 검증할 수 있게 관리되었는가?”를 판단하는 문제에 가깝다. 즉, 과정의 투명성, 검토 절차의 존재, 결과물의 독창성 및 안전성, 테스트 및 승인 체계의 충실성이 주 감정의 기준이 된다.

3.3 For the AI : 환경 및 생태계 감정

“For the AI” 관점에서의 SW 감정은 AI 시스템이 작동하는 환경과 생태계를 감정 대상으로 접근한다. 이는 특정 프로그램이나 모델 자체의 평가를 넘어, 그러한 시스템이 배포·운영되는 인프라, 연계되는 외부 서비스, 공급망, 통제 체계, 보안 및 운영 파이프라인 전체를 함께 검토한다는 점에서 기존 감정과 구별된다.

AI 시스템은 일반적으로 단독으로 작동하지 않는다. 외부 모델 API, 벡터 저장소, 데이터 파이프라인, 모델 허브, 에이전트 프레임워크, 플러그인, 접근통제 시스템, 모니터링 및 로그 수집 체계와 결합하여 동작한다. 따라서 특정 장애나 분쟁이 발생하였을 때, 문제의 원인이 코드 자체에 있는지, 외부 모델 의존성에 있는지, 데이터 공급망에 있는지, 혹은 권한관리·배포 정책의 미비에 있는지를 함께 검토해야 한다. 이러한 이유로 AI 시대의 SW 감정은 구현 결과물뿐 아니라 외부 시스템의 상호의존성을 고려하는 생태계적 접근이 필요하다.

특히 데브옵스처럼 개발과 운영이 통합된 환경에서는 보안·배포·운영의 자동화가 강화되는 동시에, 모델 공급망과 데이터 공급망의 신뢰성이 핵심 이슈로 부상한다. 이 경우 데이터, 모델, 배포 패키지, 로그, 설정값, 접근 권한 정보 등은 모두 감정 과정에서 중요한 증거가 될 수 있다.

결국 “for the AI”의 SW 감정은 SW가 설치 및 운영되고 있는 환경의 건전성과 신뢰성을 평가하는 문제로 이해할 수 있다. 여기에는 외부 연계 자원의 출처 및 무결성, 권한관리와 접근통제의 적정성, 배포·재학습·모니터링 파이프라인의 추적 가능성, 로그 및 메타데이터의 보존성과 감사 가능성, 장애 및 오류 발생 시 책임 경계의 식별 가능성 등이 포함된다. 즉, AI 시대의 SW 감정은 더 이상 결과물 단위의 정적 평가에 머물 수 없으며, 실행 환경과 운영 생태계의 동적 구조를 함께 판단하는 방향으로 확장될 필요가 있다.

AI 시대의 SW 감정은 그 대상이 코드 결과물에 한정되지 않고 데이터, 모델, 생성 과정, 운영 생태계로 확장된다는 점에서 기존 감정과 구별된다. 이러한 점을 보다 명확히 하기 위하여, 본 논문에서는 “of the AI, by the AI, for the AI”의 각 관점에 따라 감정 대상, 핵심 증거, 주요 위험, 평가 기준을 <표 3>과 같이 정리하였다.

<표 3>에서 볼 수 있듯이, AI 시대의 SW 감

표 3. AI 시대 SW 감정 패러다임의 변화에 따른 관점별 감정 요소 정리
Table 3. Summary of appraisal viewpoints in response to the changing paradigm of SW appraisal in the AI era

관점	감정 대상	핵심 증거	주요 위험	감정 기준
Of the AI	데이터, 모델, 가중치, 평가결과, 추론 로그 등	데이터 출처, 학습 기록, 모델 구조, 실험 로그, 성능 보고서 등	편향, 적법성 문제, 재현성 부족, 보안 취약성 등	적법성, 성능, 재현성, 설명가능성, 강건성 등
By the AI	요구사항, 설계 문서, 생성 코드, 테스트 문서, 협업 과정 등	프롬프트 이력, 생성 로그, 변경 기록, 리뷰 및 승인 기록 등	환각, 유사성 문제, 책임 불명확, 품질 저하 등	추적 가능성 (재현성), 기능 적합성 (완성도), 독창성, 안정성, 검토 충실성 등
For the AI	API, 도구, 파이프라인, 공급망, 운영환경 등	호출 기록, 배포 로그, 권한 정책, 감사 로그, 운영 문서 등	공급망 위험, 외부 의존성, 권한 오남용, 무결성 훼손 등	신뢰성, 무결성, 접근통제, 감사 가능성, 운영 안정성 등

정은 단일 산출물에 대한 정적 판단이 아니라 데이터-모델-과정-환경을 포괄하는 다층적 검토를 요구한다. 특히 동일한 소프트웨어라 하더라도 어느 관점에서 감정하느냐에 따라 확보할 증거와 중점적으로 검토할 위험 요인이 달라질 수 있다. 따라서 실제 감정 실무에서는 개별 사안의 성격에 따라 세 관점을 단독 또는 결합하여 적용할 필요가 있다.

4. 결 론

본 논문에서는 AI 기술의 발전이 SW 개발 환경과 SW 감정 체계에 미치는 영향을 고찰하였다. SW 개발 패러다임은 전통적 프로그래밍에서 비주얼 프로그래밍, 노코드/로우코드 개발, 생성형 AI 기반 개발로 이어지며, 개발의 추상화 수준을 높이고 개발 참여의 문턱을 낮추는 방향으로 변화하고 있다. 최근에는 개발자와 AI가 함께 참여하는 협업형 개발이 확산되면서, 개발자의 역할도 직접 구현자에서 문제 정의자, 검토자, 통합자, AI 협업 관리자로 변화하고 있다.

이와 같은 변화는 SW 감정의 대상과 방식에도 직접적인 영향을 미친다. 기존의 감정이 완성된 코드와 기능 결과물 중심으로 이루어졌다면, AI 시대에는 학습 데이터, 모델 구조와 가중치, 프롬프트 및 생성 이력, 인간-AI 협업 과정의 기록, 외부 API, 운영 파이프라인과 같은 요소도 함께 검토할 필요가 있다. 즉, 감정의 범위가 결과물 중심 평가에서 데이터, 모델, 과정, 환경을 포함하는 다층적 평가로 확대되고 있다고 볼 수 있다.

본 논문에서는 이러한 변화를 “of the AI, by the AI, for the AI”의 관점에서 정리하였다. “Of the AI” 관점에서는 AI 모델과 시스템 자체가 감정의 대상이 되며, 데이터 적법성, 편향성, 재현성, 설명 가능성, 강건성 등이 중요한 기준이 된다. “By the AI” 관점에서는 AI가 직접적으로 사용된 개발 산출물과 그 절차가 대상이 되며, 프롬프트 이력, 생성 로그, 인간 검토 기록, 테스트 및 승인 절차 등이 감정 요소가 된다. “For the AI” 관점에서는 AI가 작동하는 운영 생태계와 외부 연계 환경이 대상이 되며, 공급망 신뢰성, 접근통제, 로그 보존성, 파이프라인 추적 가능성 등이 주요 검토 요소가 된다.

본 논문에서는 AI 시대의 SW 감정이 기존의 정적 결과물 평가만으로는 충분하지 않음을 살펴봤다. 앞으로의 SW 감정 체계는 AI 자체, AI가 이용된 개발 과정, AI가 작동하는 생태계를 함께 고려하는 방향으로 확장될 필요가 있다. 이를 통해 AI 기반 SW의 신뢰성, 안전성, 공정성, 책임성을 더욱 실질적으로 평가할 수 있을 것이다.

다만 본 논문은 개념적 프레임워크의 제시에 초점을 두고 있으므로, 실제 감정 사례를 통한 실증적 검증까지 수행하지는 못하였다. 또한 AI 기반 개발의 생산성 향상 효과나 위험 요인에 대해서는 아직 축적된 연구와 공인된 지표가 충분하지 않은 부분도 있다. 향후에는 실제 감정 사례 분석, 평가 항목의 구체화, 법적 책임 귀속 기준, 인간-AI 공동 산출물의 감정 절차 등에 대한 후속 연구가 필요할 것이다.

이 논문은 2025학년도 한남대학교
학술연구비 지원에 의하여 연구되었음.

참 고 문 헌

[1] Calvina Suhas Maharao, Dr. Archana Tukaram Bhise, “Comparison of agile vs waterfall vs DevOps methodologies in software project success”, IJRSM, vol. 10, no. 12, pp. 24-39, Dec. 2023. ISSN: 2349-5197

[2] Korea Copyright Commission, Copyright Issue Trend, vol. 38, 2024. 9, https://www.copyright.or.kr/information-materials/trend/tmis/view.do?brdctsno=53329, ISSN: 2983-1946
[3] J. Kim, A Paradigm shift in Software Development in the age of Generative AI, Weekly Technology Trends no. 2180, 2025, https://www.itfind.or.kr/admin/getStreamDocsRegi.htm?identifier=JUGI_8726, IITP, ISSN 3058-3225
[4] A. W. Torrance and B. Tomlinson, “Governance of the AI, by the AI, and for the AI,” arXiv:2305.03719 [cs.CY], May 2023. DOI: 10.48550/arXiv.2305.03719.
[5] J. Y. Oh, “Present Status and Future Outlook of Vibe Coding,” ICT Industry Trends SPOT 2025-07, Information and Communications Technology Planning & Evaluation (IITP), Daejeon, South Korea, 2025. ISSN: 2982-8279
[6] A. Karpathy, “There’s a new kind of coding I call ‘vibe coding’,” X (formerly Twitter), post, Feb. 2025, https://x.com/karpathy/status/1886192184808149383?s=20
[7] J. Varajão, A. Trigo, M. Almeida, “Low-code Development Productivity”, acmqueue, vol. 21(5), Sept. 2023, pp. 1~21, DOI: 10.1145/3631183
[8] C. N. Lee, “Copyright Issues in Open Source Software and Generative AI Environments”, The Journal of Law&IP, vol. 14, no 2, 2024, pp. 59-93, DOI: 10.23190/lawnip.2024.14.2.002
[9] U.S. Copyright Office, Copyright and Artificial Intelligence, Part 3: Generative AI Training, May 9, 2025, https://www.copyright.gov/ai/
[10] A. Gan, Z. Ghodsi, “Sentry: Authenticating Machine Learning Artifacts on the Fly”, Proceedings of ACM SIGSAC CCS ‘25, pp. 3550-3563, DOI: 10.1145/3719027.3765070
[11] Google Cloud, “2025 DORA State of AI Assisted Software Development,” Google Cloud, 2025. https://cloud.google.com/resources/content/2025-dora-ai-assisted-software-development-report?hl=en&region=US. Accessed: May 17, 2026.

저 자 소 개

윤영선(Young-Sun Yun)

2001.2 KAIST 전산학과 박사
2006.4-2007.2 한국전자통신연구원 초빙 연구원
2012.8-2013.7 University of Washington 방문학자
2001.3-현재 한남대학교 교수
<주관심분야> 음성인식, 음성변환, 화자인식, 인공지능, 저작권침해, 유사도, 완성도 감정, 오픈소스 등