THE JOURNAL OF KOREAN ASSOCIATION OF COMPUTER EDUCATION Volume 29, Issue 2, 2026

컴퓨터교육학회 논문지 2026년 제29권 제2호
https://doi.org/10.32431/kace.2026.29.2.010

생성형 AI 시대, 컴퓨팅 사고의 재조명과 교수학습 전략 탐색

The Era of Generative AI: Reexamining Computational Thinking and Exploring Instructional Strategies

최숙영†
Sookyoung Choi†

요약

생성형 AI 기술의 등장은 프로그래밍 교육의 패러다임을 근본적으로 변화시키고 있으며, 이는 교육의 초점을 단순한 코드 작성 능력에서 고차적 문제해결 능력, 즉 컴퓨팅 사고력 함양으로 이동시켜야 할 필요성을 제기한다. 이에 본 연구는 국내외 관련 선행연구들에 대한 심도 있는 분석을 통해 생성형 AI 시대 컴퓨팅 사고력이 왜 중요한지 살펴보고, 이를 증진하기 위한 교수학습 전략을 탐색한다. 또한 이를 기반으로 AI 협력 기반 컴퓨팅 사고력 증진 PBL 모델을 제안한다. 이 모델은 학습자가 AI와의 협력적 상호작용을 통해 생성형 AI를 단순한 정답 생성기가 아닌 사고력 확장 파트너로 활용함으로써, 기술의 주체적인 사용자이자 창의적인 문제 해결 전문가로 성장하도록 지원하는 데 그 목적이 있다.

주제어 생성형 AI, 컴퓨팅 사고, 프로그래밍 교육, 고차적 문제해결, 교수학습 전략

ABSTRACT

The advent of generative AI technology is fundamentally transforming the paradigm of programming education, necessitating a shift in educational focus from mere coding proficiency to cultivating higher-order problem-solving abilities—specifically, computational thinking skills. This study therefore examines why computational thinking is crucial in the generative AI era through an in-depth analysis of relevant prior research both domestically and internationally, and explores teaching and learning strategies to enhance it. Furthermore, based on this analysis, it proposes an AI-collaborative computational thinking enhancement PBL model. This model aims to support learners in growing into proactive users of technology and creative problem-solving experts by enabling them to utilize generative AI not merely as a correct-answer generator, but as a partner for expanding their thinking capabilities through collaborative interaction with AI.

Keywords Generative AI, computational thinking, programming education, higher-order problem solving, teaching and learning strategies

†종신회원 우석대학교 정보보안학과 교수

논문투고 2025년 10월 10일
심사완료 2026년 01월 13일
게재확정 2026년 01월 16일
발행일자 2026년 02월 28일

1. 서론

최근 몇 년간 인공지능(Artificial intelligence, AI) 기술은 전례 없는 속도로 발전하며 사회 전반에 걸쳐 혁신적인 변화를 일으키고 있다. 특히, 자연어 처리, 컴퓨터 비전(Computer vision), 머신 러닝(Machine learning) 분야의 급격한 발전은 산업, 교육, 문화 등 다양한 영역에서 새로운 가능성을 제시하고 있다. 이제 AI는 단순한 도구를 넘어, 인간의 지적 활동을 보조하고 확장하는 필수적인 파트너로 자리 잡아가고 있다. 이러한 변화의 중심에는 방대한 데이터를 학습하여 새로운 콘텐츠를 생성하는 생성형 AI(Generative AI)기술이 있다. 생성형 AI는 기존의 문제 해결 방식에 대한 근본적인 재고를 요구하고 있다. 인간은 더 이상 정해진 절차에 따라 문제를 해결하는 역할을 넘어, AI가 효과적으로 작동하도록 문제를 정의하고, 결과를 평가하며, 윤리적 판단을 내리는 역할이 보다 중요해지고 있다[1]. 과거 AI가 인간의 작업을 효율적으로 수행하기 위한 보조 도구로서 인식이 되었다면, 이제 AI는 인간과 협력하여 더 나은 결과를 창출하는 협업 파트너로서 주목받고 있다. AI는 방대한 데이터 처리와 패턴 인식에 뛰어나지만, 맥락적 이해, 창의적 발상, 윤리적 판단 등은 여전히 인간의 고유 영역으로 남아있다.

성공적인 인간-AI 협업을 위해서는 단순히 AI 도구를 사용하는 기술적 능력을 넘어서, AI의 강점과 한계를 이해하고, 효과적인 협업 전략을 설계할 수 있는 메타인지적 사고 능력이 필요하다[2]. 이는 곧 문제를 체계적으로 분석하고, 적절한 역할 분담을 통해 최적의 해결책을 도출하는 과정으로, 컴퓨팅 사고(Computational thinking:CT)의 핵심 요소들과 직접적으로 연결된다. 이에 따라, 최근 CT에 대한 재조명이 이루어지고 있다. CT는 단순히 코딩이나 프로그래밍에 국한되지 않으며, 복잡한 문제에 구조적이고 논리적인 방식으로 접근할 수 있도록 돕는 보편적인 문제해결 프레임워크라 할 수 있다. AI가 잠재력을 최대한 발휘하려면 인간의 능력을 효과적으로 보완해야 하며, 이때 CT가 인간과 기계 사이의 간극을 해소하는 필수적인 역할을 할 수 있다[3, 4]. 즉, CT는 인간이 AI 시스템과 효과적으로 협업하도록 지원하는데, 먼저, AI를 훈련시키고 개선하며 안내하는 데 도움을 주며, AI 출력 결과를 효과적으로 맥락화하여 의사결정 과정을 개선할 수 있도록 한다. 또한 데이터 내 편향을 강화할 수 있는 패턴을 식별하여 적절한 조치를 취하도록 함으로써 AI 편향을 완화하는 데도 도움을 줄 수 있다. 이러한 CT의 중요성에도 불구하고, CT는 그 추상적인 특성 때문에 학습하는 데 어려움이 있다.

그동안 CT를 함양하기 위한 교육은 주로 프로그래밍 교육에서 이루어져 왔다. 최근에는 생성형 AI의 급속한 발전으로 대규모 언어 모델(Large language model, LLM) 기반의 도구들이 프로그래밍 교육에 활용됨에 따라 큰 변화를 가져오고 있다[5]. 이러한 AI 도구들의 활용은 프로그래밍 학습의 효율성을 높이고, 학습자의 CT와 자기효능감을 증진시키는 등 긍정적인 영향을 미치고 있다는 연구들이 발표되고 있다. 그 사례로 Yilmaz 등의 연구[6]에서는 ChatGPT 사용 그룹의 CT 총점 및 하위 요인인 창의성, 알고리즘적 사고, 협업성, 비판적 사고, 문제 해결 등의 점수가 통제 그룹보다 유의미하게 높게 나타났으며, Chen 등의 연구[7]에서도 LLM 그룹의 CT 점수 증가가 대조군보다 유의미하게 높았다는 연구 결과들이 있다. 이러한 긍정적인 효과뿐만 아니라, 학생들이 생성형 AI 도구에 과도하게 의존하여 독립적인 사고 능력과 문제해결 능력 발달이 저해될 수 있다는 우려도 제기되고 있다[8]. 이와 같은 생성형 AI의 부정적인 영향을 줄이면서 학습자들의 CT 역량을 높이기 위해서는 보다 섬세하고 효과적인 교수학습 전략에 대한 연구가 필요하다.

본 연구에서는 생성형 AI를 활용한 프로그래밍 교육에서 CT를 함양하기 위한 다양한 연구들을 분석하고 이를 기반으로 하여 생성형 AI 시대에 CT를 효과적으로 향상시키기 위한 교수학습 전략을 탐색한다. 또한 이를 기반으로 하여 하나의 교수학습 모델을 제시한다. 이를 위해 본 연구는 다음과 같은 문제를 다루고자 한다.

첫째, 생성형 AI 시대에 CT 함양이 왜 중요한가?

둘째, 생성형 AI 기반의 프로그래밍 교육에서 CT 함양을 위해 어떤 전략들을 사용해야 하는가?

셋째, 생성형 AI 기반의 프로그래밍 교육에서 CT함양을 위한 교수학습 모델은 어떻게 구성되어야 하는가?

2. 이론적 배경

2.1 CT와 문제해결

CT는 복잡한 문제를 해결하기 위해 컴퓨터 과학의 핵심 개념과 방법론을 활용하는 사고 과정이다. 이것은 Papert[9]에 의해 처음 소개되었고, Wing에 의해 대중에게 알려지게 되었다[10, 11]. Wing은 CT가 컴퓨터 활용 능력이나 프로그래밍을 위한 기계적인 기술이 아니라 문제 해결자를 위한 기본적인 기술이라고 강조했다. CT는 디지털 시대의 기본 기술로, 개인이 구조적이고 논리적인 방식으로 복잡한 문제에 접근할 수 있도록 함으로써 다양한 산업과 시나리오에 적용할 수 있는 보편적인 문제해결 프레임워크라 할 수 있다. 이와 관련하여 국내외적으로 문제해결 과정에서 CT 요소를 고려한 수업 개발 및 평가 루브릭 등 다양한 연구들이 수행되었다[12]. CT라는 용어는 정보 및 컴퓨터 교과에서 처음 사용되기 시작하여 과학교육과 같은 다른 분야의 문제해결 과정에서도 이를 적용하기 위한 연구들이 수행되어 왔다.

CT의 일관된 정의에 대한 논쟁은 여전히 진행 중이지만[13-15], CT의 요소는 크게 분해, 패턴인식, 추상화, 알고리즘의 4가지 또는 추상화, 알고리즘적 사고, 분해, 디버깅, 일반화 및 반복의 5가지로 분류하기도 한다[16, 17]. 4가지 요소로 분류하는 것이 문제해결의 기본적 인지과정에 초점을

맞추고 있다면 5가지 요소로 분류하는 것은 실제 문제 해결 과정에서의 실용적 측면을 더 강조한 면이 있다.

2.2 생성형 시대 CT의 재조명

인간과 AI의 협업이 혁신과 성공의 핵심인 산업에서 CT는 점점 더 중요해지고 있다. 교육에 CT를 통합하는 것은 학습자들이 AI 시대에 성공적으로 적응하기 위해 필수적이라는 점이 점점 설득력을 얻고 있다[18, 19]. AI가 코드를 생성해주더라도, 문제를 올바르게 정의하고 분해하는 능력, 해결책의 논리적 타당성을 검증하는 능력, 그리고 다양한 접근법을 비교 평가하는 능력은 여전히 인간이 담당해야 할 핵심 역량이기 때문이다. 특히 추상화 능력은 복잡한 요구사항을 AI가 이해할 수 있는 형태로 변환하는 데 필수적이며, 알고리즘적 사고는 AI가 제안한 해결책의 효율성과 정확성을 판단하는 데 중요한 역할을 한다. 디버깅 능력 또한 새로운 차원의 중요성을 갖게 되었다. AI가 생성한 코드에는 때로 논리적 오류나 비효율적인 부분이 있을 수 있으며, 이를 찾아내고 수정하는 것은 인간 개발자의 몫이기 때문이다. 이와 같이 CT는 AI 모델과 시스템을 비판적으로 분석하고 효과적으로 활용, 개선하는 데 핵심적인 기술로 간주되고 있다[20, 21]. 일부 연구자들은 AI와 CT를 사용하여 문제를 해결하는 과정이 유사성을 가지며 동일한 요소를 포함한다고 주장한다. 즉, 추상화, 분해, 논리적 추론과 같은 CT 요소가 AI 분야의 문제해결에도 필수적이라고 보는 것이다. 한편, CT는 AI의 근간을 이루는 사고방식이지만 AI 시대의 CT는 기존의 전통적 CT와는 달라야 한다는 주장도 있다[3].

CT와 AI의 밀접한 관계를 바탕으로, CT 교육에 대한 장벽을 해결하기 위해 ChatGPT와 같은 LLM 기반 생성형 AI의 적용을 모색하는 연구가 진행되었다. 연구 결과에 따르면 ChatGPT는 맥락에 맞는 예를 제공함으로써 추상적인 CT 요소에 대한 학생들의 이해를 도울 수 있다고 한다. 또한 ChatGPT는 의미 있는 스캐폴딩과 적응형 피드백을 제공하여 다양한 학습 속도를 수용하고 좌절감을 줄여 동기를 강화할 수 있는 것으로 밝혀졌다[6, 22]. 뿐만 아니라, CT 교육에서 생성형 AI를 이용하여 특정 프로그래밍 언어에서 자연어 프로그래밍으로 전환하여 교육하기 위한 연구도 진행되었다[1].

2.3 생성형 AI와 프로그래밍 교육, 프롬프트 공학

전통적인 프로그래밍 교육은 문법(Syntax) 학습에서 시작하여 알고리즘 구현, 데이터 구조 이해, 그리고 복잡한 시스템 설계로 이어지는 단계적 접근을 취했다. 학습자들은 각 프로그래밍 언어의 세밀한 문법 규칙을 암기하고, 컴파일 오류나 런타임 오류를 직접 해결하면서 프로그래밍 능력을 기르는 것이 일반적이었다. 이러한 방식은 체계적이고 기초가 탄탄하다는 장점이 있었지만, 초보자들에게는 높은 진입장벽을 형성하고 창의적 문제해결보다는 기술적 세부사항에 매몰되기 쉬운 한계가 있었다. 생성형 AI의 등장은 이러한 교육 방식에 변화를 가져오고 있다. 이제 학습자들은 자연어로 원하는 기능을 설명하면 AI가 해당하는 코드를 생성해주기 때문에, 문법 암기보다는 문제 정의와 해결책 설계에 더 집중할 수 있게 되었다[8]. 이는 마치 계산기가 등장하면서 수학 교육이 암산 능력보다는 수학적 사고와 문제해결 능력에 더 집중하게 된 것과 유사한 변화라고 할 수 있다.

학습자들은 이제 '어떻게 코딩할 것인가'보다는 '무엇을 만들 것인가'와 '왜 이런 방식으로 해결해야 하는가'에 더 많은 시간을 투자할 수 있게 되었다. 또한 학습자들은 이제 처음부터 실제적인 프로젝트를 수행하면서 AI와 협력하는 방법을 배우게 되는 것이 필요하게 되었다[23]. 이 과정에서 프롬프트 작성 기술, AI 결과물 평가 능력, 그리고 인간-AI 협업 능력이 요구되고 있으며, 코드 리뷰(Code review)와 리팩토링(Refactoring)의 중요성이 커지고 있다. 즉, AI가 생성한 초기 코드를 이해하고, 개선하고, 유지보수 가능한 형태로 발전시키는 능력이 보다 중요한 역량으로 인식되고 있다[24].

이와 같이 생성형 AI의 등장은 프로그래밍 교육 패러다임에 근본적인 변화를 가져왔으며, 동시에 프롬프트 공학이라는 새로운 기술 영역이 탄생되었다. 프롬프트 공학은 단순히 AI에게 명령을 내리는 것을 넘어서, AI의 능력을 최대한 활용하여 원하는 결과를 얻기 위한 체계적인 접근법이라 할 수 있다. 효과적인 프롬프트는 명확한 맥락 설정, 구체적인 요구사항 명시, 예시 제공, 그리고 단계별 지시사항을 포함해야 한다. 프롬프트 공학은 기존의 프로그래밍과 많은 공통점을 가지고 있다. 둘 다 논리적 사고, 문제 분해, 명확한 의사소통이 필요하며, 반복적인 테스트와 개선 과정을 거쳐야 한다. 이 세 영역은 서로 연관될 수 있으며, 현대 프로그래밍 교육에 변화를 요구하고 있다[1].

결론적으로, 생성형 AI, 프로그래밍 교육, 그리고 프롬프트 공학은 서로를 강화하고 발전시키는 상호보완적 관계를 형성하고 있다. 이러한 변화는 단순한 기술적 진보를 넘어서, 문제해결에 접근하는 방식 자체를 근본적으로 변화시키고 있으며, 미래의 디지털 사회를 준비하는 교육과 업무 환경의 새로운 전략을 요구하고 있다.

2.4 생성형 AI를 활용한 프로그래밍 교육

여러 연구들을 통해 생성형 AI를 프로그래밍 교육에 활용한 사례들이 발표되고 있다. 대부분의 연구에서 검색 대비 신속한 맞춤형 응답으로 인지적 부담과 탐색 시간을 줄여 주고 학습의 효과를 향상시켰다는 결과들이 발표되었다[4, 6, 24, 25, 26]. 이러한 긍정적인 면뿐만 아니라 부정확한 코드, 무비판적 의존이나 표절 등의 부정적인 문제도 제기되었다[25, 27].

여러 연구들을 분석한 결과 생성형 AI가 프로그래밍 학습에 도움을 주는 것을 다음과 같이 정리할 수 있다. 첫째, 학습자에게 개인화된 학습 지원으로 학습자 수준에 맞는 코

드 해설과 설명, 피드백을 받을 수 있다. 또한 언제든지 질문하고 도움을 받을 수 있는 개인 튜터와 같은 역할을 한다. 둘째 신속한 디버깅 및 문제해결이다. 코드 작성중 실시간으로 오류를 발견하고 수정 제안을 받을 수 있다. 이에 따라 학습자는 좌절감을 덜고 핵심적인 개념 이해에 더 집중할 수 있다. 셋째, 코드 생성 및 아이디어 탐색이다. 복잡한 알고리즘을 구현하거나 새로운 프로젝트 아이디어를 구체화할 때, AI는 유용한 부분 코드를 생성해주거나 다양한 접근 방식을 제시하여 창의적인 문제 해결 능력을 촉진한다. 넷째, 학습동기 부여 및 효율성 증대이다. 반복적이고 지루한 작업을 AI가 대신 처리해주면서 학생들은 프로그래밍의 더 흥미롭고 창의적인 측면에 몰입할 수 있다. 또한 학습자는 빠른 결과물의 도출로 학습 동기를 유지할 수 있다.

한편, 이러한 장점과 더불어 다음과 같은 단점이 존재한다. 첫째, 기초 개념 이해의 저하이다. 학생들이 AI가 생성한 코드를 깊이 있는 이해 없이 그대로 사용하는 '복사-붙여넣기'식 학습에 익숙해질 경우, 프로그래밍의 근본적인 원리나 논리적 사고 능력을 제대로 갖추지 못할 위험이 있다. 둘째, AI에 대한 과도한 의존이다. 문제해결 과정에서 스스로 고민하기보다 즉시 AI에게 답을 구하는 습관은 장기적으로 학생들의 독립적인 문제해결 능력을 저해할 수 있다. 또한 AI 없이는 아무 것도 못하는 상황이 발생할 수 있다. 셋째, 코드 품질 및 이해도 문제이다. AI가 생성한 코드의 작동 원리를 완전히 이해하지 못할 수 있다. 뿐만 아니라, AI가 제안하는 코드가 항상 최적의 해결책은 아닐 수 있기 때문에 코드의 품질이 저하될 수도 있다. 넷째, 창의적 사고의 제한이다. AI의 패턴에 따른 획일적 해결 방식을 학습함으로써 오히려 학습자의 창의적 사고를 저해할 수 있다. 또한 조금만 상황이 바뀌거나 복잡한 실제 프로젝트에 적용해야 할 때는 전혀 힘을 발휘하지 못하는 비활성 지식의 학습으로 인해 학습의 전이가 안되는 상황이 될 수 있다. 결국 생성형 AI는 프로그래밍 학습 과정에 유용한 보조 도구로 사용될 수 있지만, 위의 단점들을 보완하는 정교한 수업 설계, 윤리 프레임, 적절한 평가 체계와 결합 될 때 교육적 효과를 이끌어 낼 수 있다.

3. 생성형 AI와 CT 교육

본 장에서는 생성형 AI와 CT에 관한 관련 연구 분석으로 CT 함양 목적을 위해 생성형 AI를 활용한 연구들과 생성형 AI를 활용한 프로그래밍 교육에서 CT 효과를 분석한 연구들을 구분하여 살펴본다. 관련 연구의 자료 수집을 위해 'RISS'와 'Web of Science'를 데이터베이스로 선정하였다. 최근 3년 동안 게재된 학술 문헌을 대상으로 '컴퓨팅사고','프로그래밍 교육','생성형 AI'를 키워드로 사용하여 검색을 하였다. 처음에 검색 된 논문들(40편) 중 실제 연구주제와 관련된 것인지를 알아보기 위해 키워드가 제목이나 초록에 포함되었는지를 살펴보았고 최종 9편의 논문이 최종 분석 대상에 포함되었다.

3.1 CT 함양을 위한 생성형 AI 활용 교육

Hsu[1]는 생성형 AI 시대의 도래로 교육 현장에 생성형 AI가 활용됨에 따라 프로그래밍 학습에서도 변화가 필요함을 주장하고 있다. 그의 연구에서는 CT 교육에 생성형 AI를 폭넓게 활용하는 방안으로 CT 요소와 프롬프트 구성요소 간의 개념적 대응관계를 체계적으로 설명하고 CT 함양을 위한 구성주의적 프롬프트 프레임워크를 제시하고 있다. 이 프레임워크는 5가지 핵심 프롬프트로 구성되는데, 의미있는 프롬프트, 반복적 프롬프트, 사회적 프롬프트, 메타인지적 프롬프트, 학습자 주도 프롬프트가 이에 해당한다. 이 프레임워크를 통해 학습자는 학습 과제를 완료하거나 문제를 해결하기 위해 학습자 주도적으로 CT 핵심 요소를 상황에 맞게 자연스럽게 적용하고 발전시킬 수 있다.

Ouaazki 외[28]는 대학생을 대상으로 하여 프로그래밍 문제해결 영역에서 CT 기반 프롬프트를 활용한 프롬프트 튜닝(Prompt-tuning)을 통해 생성형 AI의 효율성을 탐구하였다. 프롬프트 튜닝은 기본 매개변수의 수정 없이 특정 작업에 대한 모델의 행동을 지시하도록 입력 프롬프트를 재조정하는 기술로, OpenAI의 GPT-3 모델에 CT 프롬프트를 도입하여 생성형 AI 대화형 에이전트가 분해, 패턴 인식, 추상화, 알고리즘 설계 같은 CT 전략을 사용하도록 안내했다. 이를 통해 학습자들이 문제를 단계별로 이해하고 해결할 수 있도록 조교 역할을 하는 생성형 AI 대화형 에이전트의 유용성을 검증하고자 했다.

Liao 외[22]는 CT와 스캐폴딩의 이론적 틀에 따라 ChatGPT를 활용한 지능형 프로그래밍 스캐폴딩 시스템을 개발하여 적용하였다. 이 연구에서는 프로그래밍 과정에서 스캐폴딩을 적용하여 학생들의 CT를 향상시키기 위해 스캐폴딩의 핵심요소와 ChatGPT를 통합하고 CT 발달 과정에 따라 ChatGPT와 대화형 모듈을 설계하였다. 그들의 시스템은 SA(Scaffolding Analysis), CA(Coding Assistance), FI(Free Interaction) 모듈과 같은 3가지 스캐폴딩 모듈을 포함하고 있다. SA 모듈은 복잡한 작업을 작은 작업으로 분해하고 유사한 작업에 대한 해결 방법으로 추상화하는 것을 지원한다. CA 모듈은 학생들이 알고리즘을 설계하고 코드를 작성한 후 지속적인 디버깅을 통해 코드 오류를 파악하고 수정하는 과정을 지원한다. FI 모듈에서는 ChatGPT와 직접 상호작용하여 과제를 완료할 수 있도록 한다. 본 연구 결과에 의하면 ChatGPT를 통합한 스캐폴딩 프로그래밍 교육을 통해 학생들의 CT 능력이 전반적으로 향상되는 것으로 나타났다.

Ouaazki 외[29] 연구에서는 대학원 학생들을 대상으로 한 CT 교과에서 ChatGPT를 사용하는 것이 학생들의 학업 성과와 컴퓨팅 사고에 어떤 영향을 미치는지 분석하였다. 이 연구는 학생들이 LLM을 단순한 작업 대행 도구가 아닌 학습 개선을 위한 도구로 활용하는 방식에 대해 새로운 관점을 제시하고자 했다. 과정의 핵심 학습 목표는 크게 세가지

로 1) 문제를 CT 방식으로 정의할 수 있는 능력, 2) 알고리즘적 해결책을 평가할 수 있는 능력, 3) 알고리즘적 형태로 해결책을 도출할 수 있는 능력이다. 연구 결과에 의하면 학생들이 자기주도적 학습 활동에 ChatGPT를 더 많이 사용할수록 최종 시험 성적도 더 우수한 것으로 나타났다.

Ponzini 외[30]는 생성형 AI가 CT 중에서도 특히 하위 요소인 문제 분해 능력을 향상시키는 데 어떤 역할을 할 수 있는지를 탐구하였다. 이 연구에서는 프로그래밍 교육 초기 단계에서 학생들이 복잡한 문제를 작은 단위의 하위 문제로 나누고, 각 단계별로 해결책을 구상할 수 있도록 지원하는 시각적 도구를 개발하였다. 사용자는 먼저 자연어로 전체 문제를 서술하고, 도구의 안내에 따라 이를 단계별 작업 단위로 분할한다. 이후 각 세부 작업에 대해 ChatGPT에게 질문하거나 코드 생성을 요청함으로써, 구조화된 문제해결 흐름 속에서 AI의 지원을 받을 수 있도록 하였다. 이러한 설계는 단순히 정답을 얻는 것이 아니라, 문제 분석, 하위 작업 식별, 알고리즘 구성, 코드 생성 및 검토로 이어지는 전 과정을 학습자 스스로 설계하고 조율할 수 있도록 도왔다. 이 연구들을 정리한 내용은 Table 1.에 보여진다.

Table 1. Generative AI-Based Education for Cultivating Computational Thinking

Researcher	Research Target	Main Approach	Main Approach
Hsu (2025)	Programmer in the age of generative AI	• CT-Prompt Framework (5 core prompts) : meaningful, iterative, social, metacognitive, learner-led	• Enables learners to apply CT contextually through prompt -based learning
Ouaazki et al (2024)	University student programming problem solving	• Prompt tuning • CT-based prompts • Graasp Bot (AI assistant)	• Useful as an assistant for debugging and supporting basic understanding • ChatGPT has a limited effect on problem-solving
Liao et al (2024)	2nd-year university students majoring in educational technology	• Scaffolding -based ChatGPT programming system	• Generative AI improves creativity, algorithmic thinking, and critical thinking • AI dependency issues
Ouaazki et al (2023)	Graduate business master's program	• CT-based curriculum design and ChatGPT learning assistance	• Promotes self-directed learning activities • Utilizes AI as a tool for learning improvement
Ponzini et al (2024)	Novice programmers	• Visual tool for problem decomposition • Linked with ChatGPT each step	• Helps learners to design and coordinate the problem-solving process themselves

3.2 생성형 AI 활용 프로그래밍 교육에서의 CT 효과

이 절에서는 생성형 AI를 활용한 프로그래밍 교육에서의 컴퓨팅 사고 효과를 분석한 연구들에 대해 살펴본다. 프로그래밍 교육에 마인드 매핑(Mind mapping)과 생성형 AI 챗봇을 통합한 학습 방법을 적용한 연구가 이루어졌다[31]. 이 연구에서는 마인드맵이 학생들의 아이디어를 체계적으로 정리하고 발전시켜 챗봇과 상호작용할 때 보다 명확하고 구조화된 정보를 제공함으로써 AI 피드백의 정확성을 높일 수 있음을 가정한다. 한편, 학습자의 발달 특성을 고려하여 3단계의 점진적 마인드맵(갭메우기, 프롬프팅, 자기스스로 구성) 스캐폴드(Scaffold)를 설계했다. 또한 이 연구에서는 교육 활동을 문제 명확히 하기, 문제 분석하기, 문제해결방안 정리, 프로그램 작성하기, 프로그램 요약 및 반성하기 등의 5단계로 구성하였다. 그 결과, 마인드맵 세션을 포함함으로써 학생들이 생성형 AI 챗봇에서 직접 답을 얻으려는 수동적 경향을 방지하여 비판적 사고와 능동적 문제해결 능력 개발에 크게 기여한 것으로 분석했다.

Chen 외[7]는 프로그래밍 디버깅 교육에서 LLM을 활용한 연구를 수행하였다. 디버깅은 프로그래밍 학습의 핵심 기술이지만, 초심자들은 자신감 부족으로 어려움을 겪는다는 문제의식에서 출발했다. 이 연구에서는 GPT-4를 기반으로 개발된 SpeechGPT라는 AI 대화형 플랫폼을 구축했다. 또한, 적응형 학습 경로를 사용하여 학생의 진도와 성과에 따라 개인화된 과제를 제공함으로써 학습에 대한 흥미와 몰입을 지속적으로 유지할 수 있도록 설계되었다. 학생들의 디버깅 기술 교육은 실행, 디버깅, 성찰, 확장의 4 가지 주요 세션으로 구성되었다. 특히 학생들이 과제를 완료한 후 자신의 사고 과정과 해결책을 체계적으로 검토할 수 있도록 성찰적 프롬프트와 자가 평가 도구를 제공하여 메타인지적 학습을 강화했다.

Yilmaz 외[6]는 ChatGPT와 같은 생성형 AI 도구가 프로그래밍 교육에 미치는 영향을 연구했다. 연구 결과, 실험 그룹 학생들의 CT, 프로그래밍 자기효능감, 수업 동기가 통제 그룹보다 모두 높게 나타났다. CT가 향상된 이유는 ChatGPT를 활용한 학생들이 직접 코드를 작성하는 시간 대신 창의적 사고, 독창적인 질문, 알고리즘적 사고, 문제해결, 비판적 사고에 더 많은 시간을 할애했기 때문으로 분석된다.

Hijón-Neira 외[32]는 유아교육 예비교사를 대상으로 AI가 생성한 교육 맥락을 활용하여 교육용 로봇을 가르치는 교수법이 학습자의 CT, 기술 수용 태도, 로봇 활용 자기효능감에 어떠한 영향을 미치는지 분석하였다. 연구 결과에 의하면 CT 역량 향상은 통계적으로 유의미하지 않았으나, 실험군은 대조군보다 전반적으로 더 높은 성취를 보였으며, 특히 문제 해결과 알고리즘적 사고 영역에서 긍정적 효과가 나타났다. 기술수용성 측면에서는, 실험군이 실제 사용항목에서 대조군보다 유의미하게 높은 점수를 기록하였다. 즉, AI 생성 맥락이 학습자의 로봇 활용 의도를 구체적으로 강화했음을 보여준 것이다. 이 연구들을 정리한 내용은 Table 2.에 보여진다.

Table 2. Computational Thinking in Programming Education Using Large Language Models

Researcher (Year)	Research Target	Main Approach and System	Main Results and Implications
Ye et al. (2024)	Programming education for middle school students	• Integrated learning with mind maps and a generative AI chatbot • 3-stage mind map scaffold	• Improved CT, and self-efficacy • Suppresses passive AI dependency and promotes critical and active problem-solving skills.
Chen et al. (2025)	6th-grade elementary school students, programming debugging education	• Built SpeechGPT, a GPT-4 based interactive platform • Adaptive learning • Reflective prompts	• Improved both debugging performance and CT • Excellent results in post- and delayed tests
Yilmaz et al. (2023)	Undergraduate students in computer science	• Programming learning using ChatGPT • Focused on creative thinking and question design instead of code writing	• Designed and performed educational robot activities using contextual tasks generated by AI
Hijón-Neira et al. (2024)	Early childhood education students	• Designed and performed educational robot activities using contextual tasks generated by AI	• Positive effects on CT (problem-solving, algorithmic thinking) • AI context enhances the intention to use robots

4. 생성형 AI 시대, CT 함양을 위한 교수학습 전략 탐색

생성형 AI의 등장은 프로그래밍 교육의 패러다임을 근본적으로 바꾸고 있다. 이제 교육의 초점은 프로그래밍 문법의 기계적 암기에서 벗어나, 복잡한 문제를 정의하고, 해결책을 논리적으로 설계하며, AI와의 효율적인 협업을 통해 창의적인 결과물을 만들어내는 고차적 사고 능력, 즉 CT를 함양하는 것으로 전환되어야 한다. 생성형 AI는 전통적인 교사 주도 교육과는 다른 학습 경험을 제공하며, 기존 CT 교육을 보완하고 개선할 수 있는 가능성을 시사한다. 그러나 생성형 AI의 활용에 따른 부정적인 면 또한 존재하기 때문에 효과적인 교육적 통합을 위한 정교한 수업 설계, 윤리 프레임, 평가 체계가 필수적이라 할 수 있다. 이러한 시대적 요구에 부응하기 위해, 본 장에서는 선행연구 분석을 토대로 생성형 AI를 활용하여 학습자의 CT를 효과적으로 강화할 수 있는 교수학습 전략을 탐색하고자 한다.

4.1 문제 해결 설계자로서의 학습자

생성형 AI 시대의 CT 함양을 위한 프로그래밍 교육은 학습자를 단순히 코드 작성자로 양성하는 것을 넘어, 문제해결의 전 과정을 주도하는 설계자로 성장시키는 것을 목표로 삼아야 한다. AI가 코드 생성을 보조하는 파트너가 되면서, 인간 학습자는 다음과 같은 핵심 역할에 집중하는 것이 필요하다. 먼저, 문제 정의 및 분해를 수행하는 단계이다. 학습자는 해결해야 할 문제를 명확하게 정의하고, 그 문제의 핵심 기능을 정리하고, 각 기능들을 작업 단위로 분해하는 것이 필요하다. 이는 CT의 핵심 요소인 분해와 추상화 단계와 연결된다. 이 과정에 생성형 AI를, 단순한 정답 제공자가 아닌 학습 협업 도구로 사용한다. 예를 들어, 생성형 AI가 전체 해결책을 제시하는 대신 문제해결의 실마리가 되는 질문이나 힌트를 제공하도록 한다. 즉, 학습자의 주체적인 사고 과정을 침해하지 않고 사고를 촉진하는 방향으로 현명하게 활용하는 것이 필요하다.

다음으로 분해된 하위 문제들을 해결하기 위한 전체적인 프로그램의 구조와 데이터 흐름, 즉 알고리즘을 설계하는 능력이다. 이때 생성형 AI에게 각 부분의 코드 생성을 요청하더라도, 이를 통합하여 효율적으로 작동하는 시스템을 구상하는 것은 인간의 몫이다. 이 과정에서 추상화 능력이 핵심적으로 요구되며, 시스템의 전체적인 구조와 모듈 간의 상호작용을 논리적으로 설계할 수 있어야 한다. 또한, 사용자는 AI가 생성한 코드가 논리적으로 타당한지, 효율적인지, 잠재적인 오류나 편향은 없는지를 비판적으로 검토하고 개선하는 것이 필요하다. 이는 AI의 결과물을 맹목적으로 수용하는 것을 넘어, 그 한계를 이해하고 주도적으로 해결책의 완성도를 높이는 메타인지적 능력을 요구한다. 특히 AI가 생성한 코드의 논리적 결함이나 효율성 문제를 찾아내고, 더 나은 대안을 제시할 수 있는 비판적 사고력이 필수적이다.

4.2 교수자의 역할 변화

학습자의 역할 변화는 교수자의 역할 변화를 필연적으로 요구한다. Hsu[1]가 제안한 것처럼 교사는 전통적인 지식 전달자에서 벗어나, 학습자가 스스로 사고하고 AI와 효과적으로 협력하며 성장할 수 있도록 지원하는 스캐폴더로서의 역할이 중요해진다. 스캐폴더로서 교수자의 핵심 역할은 다음과 같다.

첫째, 학습자에게 의미있는 질문을 유도하는 것이다. 학습자가 생성형 AI에게 피상적인 질문을 던져 정답만 얻으려 하는 것을 방지하고, 문제의 본질을 깊이 탐색하는 의미 있는 프롬프트를 생성하도록 안내해야 한다. 이를 위해 교수자는 소크라테스식 질문법 등을 활용하여 학습자의 사고를 자극하고, 문제에 대한 다각적 접근을 유도해야 한다. 둘째, 학습자가 AI의 답변을 비판적으로 분석하고, 다양한 대안을 탐색하며, 더 나은 해결책을 찾기 위해 프롬프트를 반복적으로 개선하는 과정을 촉진해야 한다. 셋째, 학습자가 자신의 문제해결 과정을 성찰하고, AI를 활용한 전략의 효율성을 스스로 평가하도록 돕는 성찰적 질문을 제시하도록 한다. 특히 학습 과정에서 어떤 사고 전략이 효과적이었는지, 어떤 부분에서 어려움을 겪었는지를 스스로 분석하도록 유도하는 것이 중요하다. 넷째, 개별 학습뿐만 아니라 동료 학

습자들과의 협력을 통해 집단지성을 활용할 수 있도록 학습 환경을 설계해야 한다. 생성형 AI에게 프롬프트를 제시할 때, 학습자들이 서로의 프롬프트를 분석하고 개선하는 협력적 학습을 통해 더 깊은 이해에 도달할 수 있게 한다. 다섯째, AI 활용에 있어서 학술적 정직성과 윤리적 기준을 명확히 제시하고, 학습자들이 AI를 적절히 활용하면서도 자신의 학습 목표를 달성할 수 있도록 지도해야 한다. 이는 표절 방지뿐만 아니라 AI에 대한 과도한 의존을 막고 독립적 사고 능력을 기르는 데 필수적이다.

4.3 CT 함양을 위한 시스템 차원의 지원

4.3.1 CT 기반 프롬프트 튜닝 및 구조화된 플랫폼 활용

학습이 단순히 AI와의 자유로운 대화에 그치지 않도록, CT의 핵심 단계를 의도적으로 경험하게 하는 구조화된 접근이 필요하다. Ouaazki 외[28]의 연구처럼, CT 기반 프롬프트 튜닝을 통해 학습자가 문제를 해결할 때 분해, 추상화, 알고리즘 설계, 디버깅 등 CT의 각 요소를 명시적으로 고려하여 프롬프트를 작성하도록 안내한다. 이는 학습자가 문제 해결 과정을 체계적으로 내면화하는 데 도움을 준다. 예를 들어, '이 문제를 어떻게 더 작은 문제들로 분해할 수 있을까요?','공통적인 패턴이나 규칙을 찾을 수 있나요? 등의 질문을 하도록 하는 것이다. 또한 Liao 외[22]가 개발한 지능형 스캐폴딩 시스템(IPSSC)처럼, CT의 각 단계에 모듈화한 학습 플랫폼을 활용할 수 있다. 이러한 플랫폼은 학습자가 각 단계에서 필요한 사고 활동에 집중하고, AI의 지원을 적절히 받으며 과제를 수행하도록 구조적으로 지원할 수 있다.

4.3.2 사고 시각화 도구를 활용한 아이디어 구체화

프롬프트를 작성하기 전, 자신의 생각을 정리하고 구조화하는 단계는 중요하다. 이 과정은 AI에 대한 맹목적인 의존을 막고 능동적인 문제해결 태도를 기르는 데 핵심적인 역할을 할 수 있다. Ye 외[31]의 연구에서 증명되었듯이, 마인드맵을 활용하여 문제해결을 위한 아이디어를 시각적으로 구조화하고 발전시킬 수 있다. 잘 구조화된 마인드맵은 더 정교하고 효과적인 프롬프트 작성을 돕고, AI로부터 고품질의 피드백을 유도하는 선순환 구조를 만든다. 또한, Ponzini 외[30]가 개발한 도구처럼, 복잡한 문제를 시각적으로 분해하고 각 하위 문제별로 AI와 상호작용하도록 지원하는 도구는 초보 학습자가 문제분해 능력을 체계적으로 학습하는 데 큰 도움이 될 수 있다.

4.4 CT 함양을 위한 고차적 사고 전략

4.4.1 메타인지 강화

AI의 도움으로 과제를 쉽게 해결하는 경험은 자칫 학습자의 독립적인 사고 능력 발달을 저해할 수 있다. 이를 방지하기 위해 학습 과정을 되돌아보는 메타인지 활동을 교육 설계에 포함하는 것이 필요하다. 이를 위해 Chen 외[7]의 연구처럼, 과제 완료 후 성찰적 질문을 통해 학습자가 자신의 사고 과정을 깊이 이해하고 개선하도록 유도한다. 또한 이러한 성찰 프롬프트를 보다 세분화하여, 즉시 성찰(Immediate reflection), 과정 성찰(Process reflection), 메타 성찰(Meta reflection)과 같은 다층적 성찰 활동을 고려할 수 있다. 즉시 성찰의 경우, AI가 제시한 결과물 중 학습자에게 의미 있다고 판단되는 요소가 무엇인지, 그리고 해당 결과물에 논리적 오류나 한계는 없는지를 점검하도록 유도하는 성찰 활동을 의미한다. 이를 통해 학습자는 AI의 응답을 무비판적으로 수용하지 않고, 그 타당성과 적절성을 즉각적으로 검토하게 된다. 과정 성찰은 문제해결 전반의 과정을 되돌아보는 성찰로서, 문제 해결 과정에서 가장 어려웠던 단계가 무엇이었는지, 사용한 프롬프트 전략 중 효과적이었던 요소는 무엇이었는지, 그리고 향후 유사한 문제 상황에서 어떤 접근 전략을 적용할 것인지를 분석하도록 한다. 메타 성찰은 학습자의 사고 방식과 AI 활용 태도에 대한 상위 수준의 성찰로, AI의 도움 없이도 해당 문제를 해결할 수 있는지 여부와 그 이유를 검토하고, 학습 과정에서 AI에 대한 의존도가 과도하지 않았는지를 비판적으로 점검하도록 한다.

또한, 동료들과 자신의 AI 활용 전략이나 성찰 내용을 공유하고 피드백을 주고받는 활동을 추가함으로써 다른 사람의 문제 해결 과정을 보며 자신의 사고를 확장하고, 더 다양한 관점을 배울 수 있다. 서로의 해결책을 비판적으로 검토하고 개선점을 도출하는 과정에서 메타인지 능력을 강화하는 효과가 있을 수 있기 때문이다. 한편, 학습자의 성찰과 자기 평가가 유의미한 성장으로 이어지려면 교사의 적절한 피드백이 필수적이다. 따라서, 포트폴리오나 성찰 기록에 대해 교사가 어떤 코멘트나 추가 질문을 제공할지에 대한 가이드라인을 마련하도록 한다. 성찰 질문에 대해 단순한 긍·부정 응답에 그친 학생에게는, 자신의 판단에 이르게 된 이유를 구체적인 경험을 근거로 재서술하도록 유도하는 추가 질문을 제공할 수 있다.

4.4.2 스캐폴딩 페이딩(Fading) 전략

학습 효과를 극대화하기 위해서는 적절한 수준의 인지적 노력이 동반되어야 한다. Ouaazki 외[28]의 연구에서 나타난 '바람직한 어려움' 현상을 교육적으로 활용하는 것이다. 즉, 학습자가 점차 독립적으로 문제를 해결할 수 있도록 지원 수준을 단계적으로 줄여나감으로써 AI의 즉각적인 도움에 대한 의존도를 줄이도록 하는 것이다. 이를 위해 먼저, 단계별 도전 과제를 설계하는데, 초급 단계는 AI의 적극적 도움을 활용하여 기본 개념을 학습하는 단계이며, 중급단계는 AI에게 힌트나 부분적 해결책만 요청하도록 제한하도록 하는 것이다. 고급 단계는 AI 없이 문제해결 후, AI 해결책과 비교 검토하도록 하는 것이다. 예를 들어, AI의 도움을 받아 특정 유형의 문제를 해결했다면, 다음에는 유사한 문제를 AI의 도움 없이 스스로 해결해 보도록 과제를 제시하는 것

이다. 또한 교수자는 AI에게 직접적인 정답을 요청하는 대신, 문제 해결에 필요한 핵심 개념이나 힌트를 제공하도록 프롬프트를 설계하여 학습자가 스스로 답을 찾아가는 과정을 경험하게 할 수 있다. 특정 시간 내에 AI 상호작용 횟수를 제한하거나 AI에게 직접적인 코드 요청 금지, 또는 개념적 설명만 요청하도록 하는 것이다. 또한 복잡한 문제를 AI 도움 없이 먼저 분해해 보도록 하는 것도 필요하다.

4.4.3 비판적 사고와 창의성 함양

생성형 AI 기반의 프로그래밍 수업에서는 더 이상 정답을 찾아내는 활동이 아니라, 대안들을 비교하고 검증하고 재구성하는 과정이 중요하기 때문에 비판적 사고와 창의성 함양을 위한 학습 전략이 필요하다. 즉, 생성형 AI가 생성한 코드가 정확한지, 효율적인지, 숨겨진 오류는 없는지 등을 검증하고, 더 나은 방향으로 개선하기 위해서 비판적 사고는 필수적이다. 동시에 창의성은 생성형 AI가 제시한 여러 가지 제안과 변형 가능성을 기반으로 하여 새로운 조합과 프레이밍을 시도하게 하는 설계적 상상력의 엔진으로 작동할 수 있다. 즉, 비판적 사고와 창의성은 AI를 단순한 답변 생성기가 아닌, 새로운 가능성을 탐색하게 하는 파트너로 삼아 문제 해결의 지평을 넓히는 역할을 하게 된다.

먼저 비판적 사고 함양을 위한 학습 전략으로는 AI에게 특정 문제에 대한 코드를 생성하게 한 후, 학생들에게 버그 찾기, 더 효율적인 코드로 개선하기, 가독성을 높이는 방향으로 리팩토링 하기 등의 과제를 부여할 수 있다. 또한 AI가 제시한 알고리즘이나 코드 구조에 대해 '왜'라고 질문을 하는 것도 필요하다. 창의성 함양을 위한 학습 전략으로는 AI의 제안을 출발점으로 삼아, 새로운 아이디어를 결합하고 독창적인 결과물을 만드는 경험을 제공하는 것이다. AI가 생성한 프로그램에 새로운 기능을 추가해서 코딩을 해보도록 하거나 동일한 문제에 다양한 제약 조건을 추가하거나 변경하여 새로운 해결책을 탐색하도록 할 수도 있다. 뿐만 아니라, 단순히 주어진 문제를 해결하는 것을 넘어, 학습자가 스스로 문제를 정의하고 해결책을 찾아 나가도록 유도함으로써 문제해결 중심이 아닌 문제 발견 중심 학습으로의 전환도 필요하다.

4.4.4 평가 방식의 혁신

전통적인 결과 중심 평가에서 벗어나, 학습자의 사고 과정과 AI 활용 역량을 종합적으로 평가할 수 있는 새로운 평가 방식이 필요하다. 먼저, 문제해결 과정에서 주어진 문제를 해결하는 것을 넘어, 스스로 문제를 발견하고 정의하는 능력, 사용한 프롬프트의 질과 변화, 성찰 일지의 깊이와 성장 정도, 답변에 대한 비판적 분석 능력 등을 평가하는 것이

Table 3. AI-Collaborative Computing Thinking Enhancement PBL Model

Step	Learner's Activity	AI's Role	Instructor's Role
Problem Discovery and Definition	• Brainstorming problems • Organizing ideas using visualization tools like mind maps • Defining goals	• Presenting various perspectives and ideas • Helping to explore potential causes or related data	• Guiding the discovery of meaningful problems with open-ended questions • Presenting initial ethical guidelines
Problem Decomposition and Abstraction	• Listing core functions • Logically decomposing each function into work units • Designing data flow and structure with flowcharts/block diagrams	• Proposing alternative decomposition strategies • Reviewing and suggesting modifications to the design plan	• Guiding CT-based prompt tuning • Inducing comparison and selection among several alternatives
Prototyping through Collaboration with AI	• Cycle of writing prompts for code generation per work unit, reviewing results, modifying codes and re-requesting prompts • Integrating modules and resolving errors • Understanding and modifying the generated code	• Code generation and debugging • Diagnosing errors and suggesting corrections • Diagnosing code quality and suggesting improvements • Verifying stability by generating various test cases	• Guiding learners to understand and modify the code generated by AI • Observing the collaboration process between learners and AI • Scaffolding and fading in the problem-solving process
Critical Evaluation and Creative Ideation	• Reviewing the efficiency and validity of LLM code • Code refactoring • Exploring creative alternatives (considering if other approaches exist)	• Diagnosing code quality and suggesting improvements • Verifying stability by generating various test cases • Idea promotion	• Guiding learners to understand and modify the code generated by AI • Observing the collaboration process between learners and AI • Scaffolding and fading in the problem-solving process
Synthesis and Sharing	• Presenting and sharing the problem-solving process, including problem definition, prompt strategy, AI collaboration, and reflection • Sharing roles, contributions, and collaboration experiences within the team	• Assisting with drafting presentation materials • Composing summary slide drafts • Generating anticipated questions	• Process-oriented evaluation: comprehensively grading prompt records, reflections, peer evaluations, test logs, and final products • Providing individual feedback
Reflection and Generalization	• Writing a reflection journal • Extracting core principles from the problem-solving process • Applying them to other problems	• Visualizing the learner's learning process • Recommending advanced learning materials and similar problem-solving cases related to the principles extracted by the learner	• Inducing deep thought by presenting multi-layered reflective questions • Encouraging the transfer of learning

필요하다.

또한 실제 문제 상황에서 AI와 협력하여 창의적 해결책 도출, 다양한 도메인의 문제에 CT 원리 적용 능력, 팀 프로젝트에서의 AI 활용 협업 능력 등을 평가하는 것도 요구된다. 마지막으로 포트폴리오 기반 평가로 학습 전 과정에서 축적된 산출물과 성찰 기록을 종합적으로 평가하여, 학습자의 성장과 발전을 전체적으로 조망할 수 있도록 한다.

4.5 AI 협력기반의 CT 증진 PBL 모델

이 절에서는 위에서 기술한 교수학습 전략을 반영하여 AI 협력 기반의 CT 함양을 위한 PBL 학습 모델을 제시한다. 이 학습 모델의 타당성 검증을 위해 AI 교육 분야의 전문가 3인으로부터 자문을 받았다. 자문에 참여한 전문가들은 컴퓨터교육을 전공하고 AI 교육 연구와 실무 경험이 있는 대학 교수 2명, 현직 교사 1명으로 구성되었다. 전문가 자문은 CT 요소 반영의 적절성, AI 활용의 교육적 타당성, 교육 현장 적용 가능성, 기존 연구와의 차별성 등을 중심으로 검토가 이루어졌다. 전문가 자문위원들은 본 연구에서 제안한 학습 모델이 생성형 AI 시대의 프로그래밍 교육 방향성을 잘 반영하고 있으며, CT를 중심으로 학습자, AI, 교수자의 역할을 구분하여 구조화하려는 시도에 대해 긍정적인 평가를 했다. 다만 일부 단계에서 역할 구분과 사고 과정의 위계가 불명확하여, 현장 교수자들이 이미 수행하고 있는 일반적인 PBL 수업과 개념적으로 충분히 구별되지 않을 수 있다는 점이 지적되었다. Table 3. 은 전문가들의 의견을 반영하여 수정한 내용이다.

Table 3. 에서 볼 수 있는 바와 같이 본 연구에서 제안한 AI 협력 기반 CT 증진 PBL 모델은 6개의 단계로 구성되어 있으며, 각 단계에서 학습자, AI, 교수자가 서로 다른 역할을 수행하면서 유기적으로 협력하는 구조를 갖추고 있다. 이 여섯 단계는 선행연구 분석 결과를 토대로 구성되었다. 1단계는 문제 발견 및 정의 단계로, 생성형 AI 환경에서 학습자가 문제를 수동적으로 수용하기보다 스스로 문제를 설정하도록 요구한 연구들[1, 31]과 같이 학습자는 능동적인 문제 발견자로서의 역할을 수행한다. AI는 단순히 답을 제공하는 것이 아니라, 다양한 관점과 아이디어를 제시하여 학습자의 사고 확장을 돕는다. 교수자는 탐색 촉진자로서 개방형 질문을 통해 의미 있는 문제 발굴을 유도하고, 초기 윤리 가이드라인을 제시하여 바른 방향성을 제공한다. 2단계는 다수의 연구[22, 29, 30]에서 그 중요성을 강조한 분해와 추상화 단계로, 학습자는 핵심 기능 모듈화와 각 기능들의 작업 단위로의 논리적 분해, 순서도나 블록 다이어그램을 통한 데이터 흐름 등을 설계한다. AI는 설계안에 대해 논리적 오류나 비효율적인 부분에 대한 검토를 제공함으로 학습자의 사고 과정을 보완하고 개선한다. 3단계는 AI와 협업을 통한 프로토타입 제작 단계로 이론적 사고를 실제 구현으로 연결하는 단계이다. 학습자는 작업 단위별 코드 생성을 위한 프롬프트를 작성하고, 작업 단위별 생성된 코드를 수정 및 통합하고 오류를 해결한다. AI는 코드 생성과 디버깅, 오류 진단 및 수정 제안을 통해 기술적 파트너로서의 역할을 수행한다. 4단계는 비판적 평가 및 창의적 발상 단계로 LLM 코드의 효율성과 타당성을 검토하고, 코드 리팩토링과 창의적 대안을 탐색한다. AI는 코드 품질 진단과 개선을 제안하고 아이디어를 촉진한다. 이러한 3, 4단계는 생성형 AI를 단순한 코드 생성 도구가 아닌, 사고 촉진 파트너로 활용할 것을 제안한 연구들[6, 7]의 교수학습 흐름을 반영하여 구성되었다. 5단계는 종합 및 공유 단계로 전체 해결 과정을 발표하고 공유한다. AI는 학습자의 발표 준비를 돕는다. 이때 교수자는 종합 평가자로 다양한 관점들을 통해 평가하고 개인별 피드백을 제공한다. 6단계는 성찰 및 일반화 단계로 성찰 일지를 작성하고 문제 해결 과정에서의 핵심 원리를 추출하여 일반화하는 능력을 기르도록 한다. 이 단계에서는 [7, 28]의 연구에서 제안하고 있는 성찰적 프롬프트와 메타인지적 학습 지원 전략을 반영하여, 문제 해결 전 과정을 회고하고 학습 전이를 유도하도록 구성하였다

5. 결론

최근 ChatGPT와 같은 대규모 언어 모델의 등장은 학습자가 단순한 코드 작성자에서 문제 해결의 설계자로 전환될 수 있는 가능성을 제시한다. 생성형 AI는 학습자에게 맞춤형 지원을 제공하고 문제 해결의 효율성을 높이는 긍정적 가능성을 제시하지만, 동시에 학습자의 독립적 사고 능력 저하와 AI에 대한 과도한 의존이라는 잠재적 위험 또한 내포하고 있다. 따라서 AI를 교육 현장에 효과적으로 통합하기 위해서는 기술의 긍정적 측면은 극대화하고 부정적 영향은 최소화할 수 있는 정교한 교수학습 전략이 필수적으로 요구된다.

본 연구는 선행연구 분석을 통해 생성형 AI 시대에 CT를 효과적으로 강화하기 위한 교수학습 전략을 탐색하였다. 이를 위해 먼저, 학습자를 코드 작성자가 아닌, 문제 해결의 전 과정을 주도하는 문제 해결 설계자로 인식하고, AI를 협업 파트너로 활용하여 창의적이고 효율적인 해결책을 도출하는 능동적 주체로서의 역할을 강조하였다. 다음으로, 학습자의 역할 변화에 따라 교수자의 역할 역시 지식 전달자에서 학습 촉진 스캐폴더로의 전환에 대해 기술하였다. 마지막으로, 이러한 역할 변화를 바탕으로 교수학습 전략을 제안하였다. 이러한 교수학습 전략으로 CT 기반 프롬프트 튜닝이나 구조화된 학습 플랫폼을 활용하여 학습자가 CT의 핵심 요소를 의도적으로 경험하게 하고, 마인드맵과 같은 사고 시각화 도구를 통해 아이디어를 구체화하여 AI에 대한 의존도를 낮추고 능동적 문제해결 태도를 기르도록 하는 것에 대해 논의하였다. 또한, 성찰 프롬프트와 자기 평가를 통해 메타인지 활동을 강화하고, 학습자의 수준에 따라 AI의 지원을 점진적으로 줄여나가는 스캐폴딩 페이딩전략을 적용하여 독립적 문제 해결 능력을 신장시킬 수 있는 부분에 대해서도 논의하였다. 또한 이러한 전략을 기반으로 AI 협력기

반 CT 증진 PBL 모델을 제안하였다. 이 PBL 모델은 생성형 AI 시대에 필요한 인간 고유의 역량인 창의성, 비판적 사고, 메타인지, 협업 능력 등의 고차원 사고 능력을 함양할 수 있도록 하였다. 또한 AI를 단순한 답안 생성 도구가 아닌, 학습자의 사고를 촉진하고 CT를 내면화하는 파트너로 활용함으로써, 학습자가 미래 사회에서 AI와 협업하며 문제를 해결하는 능력을 기를 수 있도록 하였다.

본 연구는 문헌 분석과 전문가 자문을 통해 AI 협력 기반 CT 증진 PBL 모델을 제안하였으나, 실제 교육 현장에서의 적용 효과를 실증적으로 검증하지는 못했다. 따라서 향후 연구에서는 본 모델을 실제 프로그래밍 수업에 적용하여 CT 하위 요소별 변화, 학습자 인식, AI 의존도 변화 등을 정량·정성적으로 분석하는 후속 연구가 필요하다. 또한 델파이 조사와 같은 보다 체계적인 외부 검증 절차를 통해 모델의 일반화 가능성을 검토할 필요가 있다.

참고문헌

[1] Hsu, H. (2025). From programming to prompting: Developing computational thinking through large language model-based generative artificial intelligence. *TechTrends*, 69, 485–506. https://doi.org/10.1007/s11528–025-01052-6

[2] Sidra, S., & Mason, C. (2025). Generative AI in human–AI collaboration: Validation of the Collaborative AI Literacy and Collaborative AI Metacognition scales for effective use. *International Journal of Human–Computer Interaction*, 1–25. https://doi.org/10.1080/10447318.2025.2543997

[3] Li, R. (2025). Cultivation of computational thinking in the context of AI general education. *International Conference on Artificial Intelligence and Educational Systems (ICAIES 2025)*, April 25–27, 2025, Beijing, China. ACM. https://doi.org/10.1145/3744367.3744424

[4] Tian, S. (2024). Exploring the interactive relationship between computational thinking and the development of artificial intelligence. *Journal of Education and Educational Research, 9*(2), 177-180.

[5] Prather, J., et al. (2024). Beyond the hype: A comprehensive review of current trends in generative AI research, teaching practices, and tools. *arXiv*. https: //doi.org/10.48550/arXiv.2412.14732

[6] Yilmaz, R., & Yilmaz, F. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence, 4*, 100147. https:// doi.org/10.1016/j.caeai.2023.100147

[7] Chen, S., Shan, X., Liu, Z., & Chen, C. (2025). Employing large language models to enhance K–12 students' programming debugging skills, computational thinking, and self-efficacy. *Educational Technology & Society, 28*(2), 259–278. https://doi.org/10.30191/ETS.202504_28(2).TP01

[8] Lepp, M., Lahtinen, E., & Malmi, L. (2025). Does generative AI help in learning programming? Students' perceptions, reported use and relation to performance. *Computers in Human Behavior Reports, 18*, 100642. https://doi.org/10.1016/j.chbr.2025.100642

[9] Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books.

[10] Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

[11] Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K–12 classrooms. *TechTrends, 60*(6), 565–568. https://doi.org/10.1007/s11528-016-0087-7

[12] Choi, S. (2016). A study on teaching–learning for enhancing computational thinking skill in terms of problem solving. *Journal of The Korean Association of Computer Education, 19*(1), 53–62.

[13] Butler, D., & Leahy, M. (2021). Developing preservice teachers' understanding of computational thinking: A constructionist approach. *British Journal of Educational Technology, 52*(3), 1060–1077. https://doi.org/10.1111/bjet.13090

[14] Lyon, J., & Magana, A. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education, 28*(5), 1174–1189. https://doi.org/10.1002/cae.22295

[15] Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, 103798. https://doi.org/10.1016/j.compedu.2019.103798

[16] Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K–6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society, 19*(3), 47–57.

[17] Ballard, E., & Haroldson, R. (2022). Analysis of computational thinking in children's literature for K–6 students: Literature as a non-programming unplugged resource. *Journal of Educational Computing Research, 59*(8), 1487–1516. https://doi.org/10.1177/07356331211004048

[18] Department of Education and Youth. (2022). *Digital strategy for schools to 2027*. Retrieved January 22, 2025, from https://www.gov.ie/en/publication/69fb88-digital-strategy-for-schools/

[19] Fagerlund, J., Leino, K., Kiuru, N., & Niilo-Rämä, M. (2022). Finnish teachers' and students' programming motivation and their role in teaching and learning computational thinking. *Frontiers in Education, 7*, 948783. https://doi.org/10.3389/feduc.2022.948783

[20] Asunda, P., Faezipour, M., Tolemy, J., & Do Engel, M. (2023). Embracing computational thinking as an impetus for artificial intelligence in integrated STEM disciplines through engineering and technology education. *Journal of Technology Education, 34*(2), 43–63. https://doi.org/10.21061/jte.v34i2.a.3

[21] Markauskaite, L., Marrone, R., Poquet, O., Knight,

S., Martinez-Maldonado, R., Howard, S., Siemens, G. (2022). Rethinking the entwinement between artificial intelligence and human learning: What capabilities do learners need for a world with AI? *Computers and Education: Artificial Intelligence, 3*, 100056. https://doi.org/10.1016/j.caeai.2022.100056

[22] Liao, J., Zhong, L., Zhe, L., Xu, H., Liu, M., & Xie, T. (2024). Scaffolding computational thinking with ChatGPT. *IEEE Transactions on Learning Technologies, 17*, 1668–1682. https://doi.org/10.1109/TLT.2024.3392896

[23] Garg, A., Soodhani, K., & Rajendran, R. (2025). Enhancing data analysis and programming skills through structured prompt training: The impact of generative AI in engineering education. *Computers and Education: Artificial Intelligence, 8*, 100380. https://doi.org/10.1016/j.caeai.2025.100380

[24] Menoll, A., Strik, B., & Rodrigues, L. (2024). Teaching refactoring to improve code quality with ChatGPT: An experience report in undergraduate lessons. *Proceedings of the XXIII Brazilian Symposium on Software Quality.* https://doi.org/10.1145/3701625.3701681

[25] Surameery, N. & Shakor, M. (2023). Use ChatGPT to solve programming bugs. International *Journal of Information Technology & Computer Engineering, 3*(1), 17–22. https://doi.org/10.55529/ijitc.31.17.22

[26] Tian, H., Lu, W., Li, T., Tang, X., Cheung, S., Klein, J., & Bissyandé, T. (2023). Is ChatGPT the ultimate programming assistant—How far is it? *arXiv*. https://arXiv:2304.11938

[27] Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *International Journal of Educational Technology in Higher Education, 21*, 14. https://doi.org/10.1186/s41239-024-00446-5

[28] Ouaazki, A., Bergram, K., Farah, J., Gillet, D., & Holzer, A. (2024). Generative AI-enabled conversational interaction to support self-directed learning experiences in transversal computational thinking. *Proceedings of the 2024 ACM Conference on Conversational User Interfaces(CUI'24)*. https://doi.org/10.1145/3640794.3665542

[29] Ouaazki, A., Bergram, K., & Holzer, A. (2023). Leveraging ChatGPT to enhance computational thinking learning experiences. *IEEE International Conference on Engineering, Technology and Education (TALE)* https://doi.org/10.1109/TALE56641.2023.10398358

[30] Ponzini, D., Adorni, G., Delzanno, G., & Guerrini, G. (2024). Toward the use of generative AI to develop computational thinking by supporting problem decomposition. *4th National Conference on Artificial Intelligence* (CEUR Workshop Proceedings).

[31] Ye, X., Zhang, W., Zhou, Y., Li, X., & Zhou, Q. (2025). Improving students' programming performance: An integrated mind mapping and generative AI chatbot learning approach. *Humanities and Social Sciences Communications, 12*, Article 558. https://doi.org/10.1057/s41599-025-04846-4

[32] Hijón-Neira, R., Pizarro, C., Borrás-Gené, O., & Cavero, S. (2024). AI-generated context for teaching robotics to improve computational thinking in early childhood education. *Education Sciences, 14*(12), 1401. https://doi.org/10.3390/educsci14121401

최숙영

· 1998년 전북대학교 전산학과 (이학사)
· 1991년 전북대학교 전산학과 (이학석사)
· 1996년 충남대학교 전산과학과 (이학박사)
· 2008년 Nova Southeastern University 교육공학 및 원격교육 (교육학박사)
· 1996년~현재 우석대학교 정보보안학과 교수

관심분야 : 인공지능 교육, 컴퓨팅사고 교육, 이러닝 시스템, 인공지능 윤리

sychoi@woosuk.ac.kr