

A Technique for UML-based System Development Using Generative AI

Eun-Sook Cho*

*Professor, Dept. of Software Engineering, Seoil University, Seoul, Korea

[Abstract]

Generative AI has been successfully applied across various domains such as healthcare, finance, and entertainment, offering transformative capabilities for complex tasks. In this paper, we propose a method for automating and optimizing UML-based software development using generative AI. The proposed method focuses on streamlining the development process by automating the generation of UML diagrams and improving model accuracy through AI-powered refinement. Key features include the ability to generate functional and structural UML models, such as use case, sequence, and class diagrams, directly from textual requirements with minimal manual intervention. The integration of generative AI reduces development time by 30% and error rates by 40% compared to traditional methods. Additionally, the approach supports iterative refinement, ensuring consistent and accurate models while minimizing redundant efforts. By leveraging AI for both analysis and design phases, the proposed method enhances efficiency and reliability in complex system development projects.

▶ **Key words:** Generative AI, Development Process, UML Diagrams, Efficiency, Reliability

[요약]

생성형 AI는 의료, 금융, 엔터테인먼트 등 다양한 분야에서 성공적으로 적용되며 복잡한 작업의 변화를 이끌어내고 있다. 본 논문에서는 생성형 AI를 활용하여 UML 기반 소프트웨어 개발을 자동화하고 최적화하는 방법을 제안한다. 제안된 방법은 요구사항으로부터 최소한의 수작업으로 UML 다이어그램(예: 유스케이스, 순차도, 클래스 다이어그램)을 자동 생성하고, AI 기반 정제를 통해 모델 정확성을 향상시키는 데 중점을 둔다. 생성형 AI의 통합을 통해 개발 시간은 30%, 오류율은 40% 감소하며, 반복적인 모델 정제 작업을 통해 일관되고 정확한 모델을 보장한다. 또한, 분석 및 설계 단계를 포함한 시스템 개발 프로세스 전반에 AI를 활용함으로써 복잡한 시스템 개발에서의 효율성과 신뢰성을 크게 향상시킨다.

▶ **주제어:** 생성형 AI, 개발 프로세스, UML 다이어그램, 효율성, 신뢰성

I. Introduction

최근 기술 환경에서 인공지능(AI)의 발전은 눈부신 속도로 진행되고 있으며, 특히 생성형 AI는 다양한 산업 분야에 혁신적인 변화를 가져오고 있다. 생성형 AI는 데이터를 기반으로 새로운 내용을 독립적으로 생성할 수 있는 기능을 가지며, 이는 의료, 금융, 엔터테인먼트 및 기술 분야에서 특히 강력한 효과를 발휘하고 있다[1]. 이러한 기술적 진보가 소프트웨어 개발, 특히 객체지향 시스템 개발에 적용될 경우, 기존의 접근 방식을 근본적으로 변화시킬 잠재력을 지니고 있음에도 불구하고, 이 분야에서의 적용 사례는 상대적으로 제한적이다.

객체지향 시스템 개발은 복잡성 관리, 코드 재사용성 향상, 그리고 유지보수의 용이성 측면에서 강력한 이점을 제공한다[2]. 하지만 전통적인 개발 접근법에서는 상당한 시간과 노력이 요구되며, 때로는 반복적인 작업으로 인해 개발자의 창의성이 저하되는 문제가 발생한다. 이러한 문제는 생성형 AI를 통해 해결할 수 있으며, AI가 개발 과정에서 반복적이고 기계적인 작업을 자동화하고, 정확하고 일관된 UML 다이어그램을 신속하게 생성함으로써 개발 시간을 단축하고 오류를 줄일 수 있다[3,4].

생성형 AI를 객체지향 시스템 개발에 통합하면, 복잡한 시스템의 설계와 구현 과정에서 요구되는 상세한 모델링 작업을 자동화할 수 있다. 이는 프로젝트의 초기 단계에서 요구 사항 분석과 설계의 정확성을 높이는 데 기여하며, 이후의 개발 단계에서는 더 빠르고 효율적인 코드 구현을 가능하게 한다. 또한, 생성형 AI는 다양한 설계 패턴과 최적의 해결 방안을 제안함으로써 개발자가 보다 창의적인 문제 해결에 집중할 수 있도록 지원한다[5,6].

이러한 이점을 바탕으로, 본 논문은 생성형 AI를 활용하여 UML 기반의 객체지향 시스템 개발 과정을 자동화하고 최적화하는 방법론을 제안하고, 실제 적용 사례를 통해 그 효과를 검증하고자 한다.

본 논문의 구성은 2장에서는 관련 연구로 생성형 AI를 이용한 객체지향 소프트웨어 개발 방법, 그리고 기존 연구들이 가지는 한계점 등을 제시한다. 3장에서는 UML을 이용한 대화형 AI 시스템 모델링 방법을 제시한다. 4장에서는 제안한 모델링 기법을 Dialogflow 생성형 AI 모델을 기반으로 한 대화형 챗봇 시스템 개발에 적용한 사례 연구를 설명한다. 마지막으로 5장에서 결론과 향후 연구 과제를 기술한다.

II. Preliminaries

1. Related works

본 논문에서 제안하는 생성형 AI를 활용한 UML 기반의 객체지향 시스템 개발 방법론을 논의하기에 앞서, 관련 연구를 검토하여 이 분야에서의 연구 동향과 기존 문제점들을 이해하고자 한다. 이 절에서는 기존 연구를 바탕으로 이 논문의 방법론이 기존 방법론과 어떻게 차별화되는지, 그리고 어떤 새로운 기여를 제공하는지 설명한다.

1.1 Research on the Use of Existing AI

인공지능 기술이 소프트웨어 개발 과정에서 모델링, 코드 생성, 테스트 자동화 등 다양한 방면에서 활용되고 있다. 특히, 객체지향 분석 및 설계 단계에서의 UML 다이어그램 자동 생성 연구가 관심을 받고 있다. 일례로, ChatGPT를 사용하여 자연어 요구 사항에서 시퀀스 다이어그램을 생성하는 연구가 있다[7,8,9]. 이 연구의 목적은 시퀀스 다이어그램이 요구 사항을 보완하고 이해관계자와의 커뮤니케이션을 촉진하는 역할을 분석하는 연구이다. 연구는 ChatGPT가 생성한 다이어그램의 질과 실제 설정에서의 적용 가능성을 정량적으로 평가하고, 자연어 요구 사항에서 시퀀스 다이어그램을 생성할 때 발생할 수 있는 문제(예: 불완전성, 이해도 부족)를 정성적으로 탐색한다.

이 연구의 한계점으로는 AI가 생성한 다이어그램이 때때로 입력 요구 사항의 특정 특성이나 ChatGPT의 한계로 인해 문제를 일으킬 수 있다는 점이다. 또한, 이 연구는 주로 기능적 요구 사항에 초점을 맞추었으며, 품질 측면이나 다른 유형의 UML 다이어그램 생성은 다루지 않고 있다.

1.2 Introduction of Generative AI

최근 연구에서는 생성형 AI 기술이 어떻게 다양한 형태의 아티팩트를 생성할 수 있는지 탐구하고 있다. 이러한 기술은 자연어 처리를 넘어서 코드 생성, 테스트 케이스 작성 등 복잡한 개발 작업에도 활용될 수 있는 가능성을 보여주고 있다[10,11].

GitHub Copilot과 AWS의 CodeWhisperer는 소프트웨어 개발 프로세스에서 코드 제안 및 자동화된 코드 생성을 제공하는 대표적인 예시이다. GitHub Copilot은 개발자의 코드 작성을 보조하며, 실시간으로 코드 제안을 제공하여 개발자의 생산성을 향상시키고 반복적인 작업에서 벗어나도록 돕는다. CodeWhisperer 역시 유사한 기능을 제공하며, 라인 완성, 주석 추가 또는 전체 함수 및 코드 블록 생성을 자동화한다. 그러나, 이러한 도구들은 효율성

과 생산성을 크게 향상시키지만, 자동 생성된 코드는 때때로 정확하지 않을 수 있으며, 사용자의 의도와 완벽하게 일치하지 않는 경우가 있다. 이는 개발자가 추가적으로 검토하고 수정해야 하는 필요성을 낳는다. 뿐만 아니라, 이러한 AI 도구는 특정 프로젝트나 조직의 독특한 요구 사항을 완전히 이해하지 못할 수 있다. 따라서 복잡한 프로젝트 요구를 만족시키기 위해서는 여전히 인간 개발자의 개입이 필요하다.

2. Limitations of Existing Researches

생성형 AI를 소프트웨어 개발에 접목시키는 연구들은 여러 가지 혁신적인 잠재력을 보여주고 있지만, 동시에 몇 가지 문제점과 한계점을 갖고 있다.

- 정확성과 신뢰성의 문제: 생성형 AI는 자동화된 코드 생성, UML 다이어그램 작성, 요구 사항 분석 등을 포함하여 소프트웨어 개발의 다양한 단계에서 활용될 수 있다. 하지만 AI가 생성한 아웃풋의 정확성은 입력 데이터의 질과 직결되어 있으며, 때로는 부정확하거나 불안정한 결과를 초래할 수 있다. 이는 소프트웨어 개발에서의 정밀한 요구사항을 충족시키지 못하는 경우가 발생할 수 있다.
- 컨텍스트 이해 부족: AI 시스템, 특히 NLP 기반 도구는 때때로 주어진 프로젝트의 맥락이나 복잡성을 완전히 이해하지 못할 수 있다. 소프트웨어 개발은 특정 도메인에 대한 깊은 지식과 이해가 필요한 경우가 많은데, AI는 이러한 뉘앙스를 파악하는 데 한계를 보일 수 있다.
- 통합의 복잡성: 생성형 AI 도구를 기존의 개발 프로세스나 시스템과 통합하는 것은 기술적으로 복잡하고 도전적일 수 있다. 기존 시스템과의 호환성 문제, 개발 팀의 AI 기술 적용에 대한 저항, 기존 워크플로우와의 충돌 등이 발생할 수 있다.
- 성능과 확장성 문제: AI 기반 시스템의 성능은 대량의 데이터와 복잡한 연산을 요구할 수 있다. 이로 인해 소프트웨어 개발 프로젝트에서 요구하는 확장성과 유연성을 제공하는 데 한계가 있을 수 있다.

본 논문에서 제안하는 기법은 기존의 생성형 AI를 소프트웨어 개발에 접목하는 연구들이 직면한 몇 가지 핵심적인 한계점들을 해결하고자 한다. 이를 통해 AI 기반의 UML 다이어그램 생성과 코드 자동화 과정에서 더 높은 정확성과 효율성을 달성할 수 있는 방법을 모색하고자 한다.

III. Analysis & Design of Conversational AI System

이 장에서는 객체지향 모델링 언어인 UML과 생성형 AI 모델을 이용한 객체 시스템 개발을 위한 개발 프로세스와 프로세스 내 분석 및 설계 기법을 제시한다.

1. Basic Structure of Modeling Process

생성형 AI를 활용한 UML 기반 시스템 모델링 개발 프로세스는 [그림1]처럼 분석, 설계, 구현 단계로 구성된다.

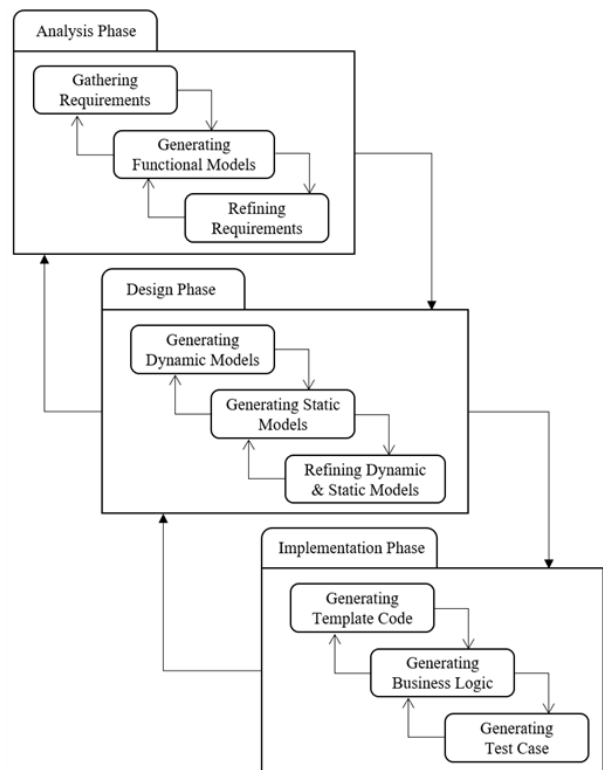


Fig. 1. Generative AI-based Development Process

개발 프로세스의 단계 또는 단계 내 활동은 반복적이면서 점진적으로 진행된다. 분석 단계에서는 첫 번째 작업으로 생성형 AI를 사용하여 이해관계자 인터뷰를 진행하고, 이를 통해 기능적 및 비기능적 요구 사항을 수집한다. 이때 생성형 AI는 구두 또는 서면 입력을 해석하여 초기 요구 사항 문서를 작성한다. 두 번째 작업에서는 생성형 AI를 활용하여 구조화 된 요구 사항을 입력으로 받아서 UML의 기능적 모델을 작성한다. 이를 통해 프로젝트 초기에 시스템의 범위와 상호 작용을 시각화할 수 있다. 세 번째 작업에서는 AI 도구를 사용하여 시스템 분석가와 이해관계자의 피드백을 바탕으로 요구사항과 기능모델을 정제한다.

설계 단계에서의 첫 번째 작업은 시스템의 동적 모델링 작업으로, 분석 단계에서 생성된 기능 모델을 입력으로 받아서 각 유스케이스 별로 순차도(Sequence Diagram)을 작성한다. 작성된 순차도를 입력으로 받아서 생성형 AI를 이용해서 두 번째 작업인 시스템의 정적 모델링 작업을 하게 된다. 이와 같은 작업을 생성형 AI의 도움을 받음으로써 문제를 해결하기 위해 필요한 디자인 패턴을 적용할 수 있다. 이는 시스템의 견고성 및 유지 관리성을 향상시킬 수 있다.

마지막으로 구현 단계에서는 생성형 AI를 이용해서 코드 생성을 하게 되고, 생성된 코드가 비즈니스 로직 부분은 채워지지 않은 템플릿 코드이므로 비즈니스 로직에 대한 부분은 개발자가 프롬프트를 다시 입력하여 생성형 AI의 도움을 받아 로직을 완성한다. 이렇게 코드가 완성된 후, 단위시험을 위한 테스트 코드를 생성형 AI를 통해 생성하도록 한다.

2. Analysis & Design of System in 3 Views

2.1. Generative AI-based Functional Modeling

이 절에서는 UML을 이용하여 시스템 기능 모델링 할 경우, Use Case 다이어그램과 Use Case 명세서를 작성한다. 본 논문에서 Use Case 다이어그램이나 Use Case 명세서를 작성하는 데 있어서 생성형 AI의 도움을 받아서 분석가가 생성형 AI에게 개발하고자 하는 시스템에 대한 내용을 프롬프트로 질의하여 Use Case나 Actor 추출을 한다.

예를 들어 게시판 시스템을 구축하고자 할 때 [그림2], [그림3], [그림4]와 같은 형태로 게시판 시스템에 필요한 Use Case와 Actor 추출 뿐만 아니라 Use Case와 Actor 들 간의 연관 관계 대한 목록, Use Case들 간의 관계 유형을 생성형 AI가 생성해 준다. 이렇게 생성된 결과들을 토대로 분석가가 CASE 도구를 이용해서 Use Case 다이어그램을 작성하면 된다.

Use Case 다이어그램이 작성되면, 각각의 Use Case 별로 Use Case 명세서를 작성해야 한다. 과거에는 이 Use Case 명세서를 작성하기 위해 시스템 분석가들이 많이 시간과 노력이 들었다. 그러나 이 부분도 이제는 생성형 AI에게 프롬프트를 잘 기술해서 던지면, 생성형 AI가 기존 투입 시간 대비 엄청난 속도로 Use Case 명세서를 작성해 준다. 작성된 Use Case에 명세서를 보고 분석가가 오류를 수정하거나 흐름의 부족한 부분들을 추가하면 된다.

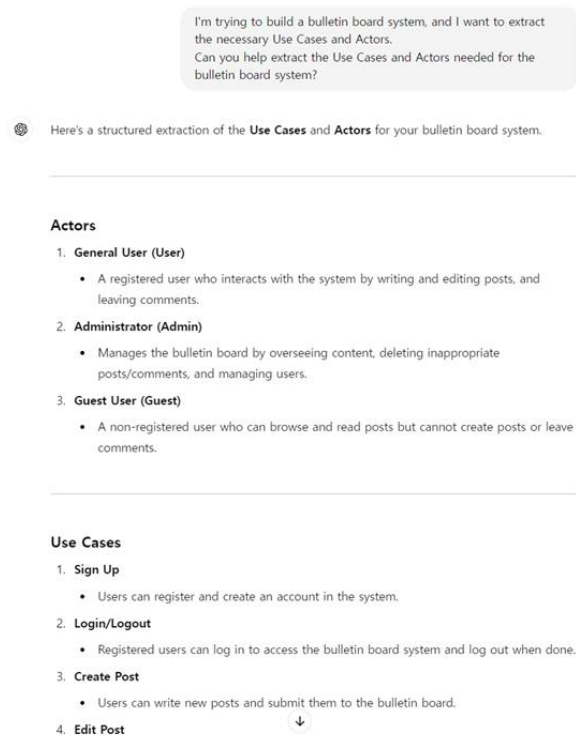


Fig. 2. Identify Use Cases & Actors by chatGPT

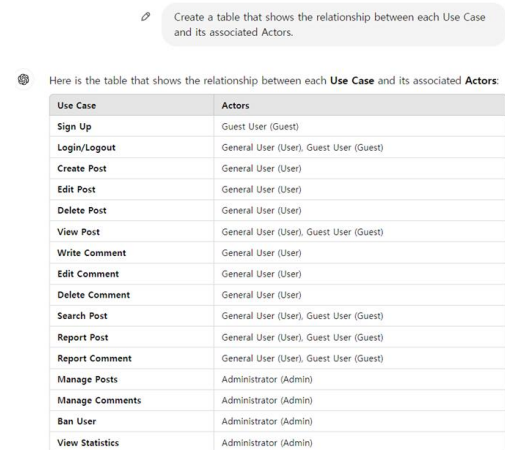


Fig. 3. Generating Relationship of Use Cases & Actors

Use Case Specification: Create Post

1. Use Case Name:
Create Post

2. Related Actors:

- General User (User)

3. Related Use Cases:

- Login/Logout (Include)

4. Overview of the Use Case:
The **Create Post** Use Case allows a registered user to write and publish a post on the bulletin board. The user provides a title and content for the post. If the post passes validation, it is saved and displayed on the bulletin board. Users can optionally preview the post or save it as a draft for future publishing.

5. Normal Flow:

Actor Actions	System Responses
1. The user logs into the system.	2. The system verifies the login and provides access.
3. The user clicks the "Create Post" button.	4. The system displays the post creation form.
5. The user enters the post's title and content.	6. The system validates the inputs (E-1).
7. The user clicks the "Submit" button.	8. The system saves the post and displays a success message.

Fig. 4. Generating Use Case Specification

2.2. Dynamic Modeling Using Generative AI

앞 절에서 작성된 Use Case 명세서를 가지고 생성형 AI를 이용해서 이루어진다. 본 논문에서는 동적 모델링 도구로 UML의 순차도를 작성한다. 분석가는 생성형 AI에게 작성된 Use Case 명세서를 입력하고, 이를 토대로 각각의 Use Case 별로 순차도를 어떻게 명세할 것인가에 대한 프롬프트를 작성해서 전송하면, 생성형 AI가 와 같은 결과를 생성해 준다. 이 결과를 토대로 분석가가 검증한 후, 순차도를 작성하면 된다.

I want to create a UML Sequence Diagram based on the Create Post Use Case Specification described above. Please extract the Actors and Objects that will appear in the sequence diagram. Additionally, classify the objects as Boundary Object, Controller Object, or Entity Object.

Based on the **Create Post Use Case Specification**, here are the extracted **Actors** and **Objects** that will appear in the **UML Sequence Diagram**, classified as **Boundary Object**, **Controller Object**, and **Entity Object**:

1. Actor

- **User (General User):** The main actor who creates the post by interacting with the system.

2. Boundary Objects

Boundary objects are responsible for interacting with the user, such as forms and UI components.

- **PostForm:** Displays the post creation form to the user and collects input (title and content).
- **PostPreview:** Displays the preview of the post for the user to review.
- **ErrorMessage:** Shows error messages to the user if the input is invalid or if an exception occurs.
- **SuccessMessage:** Displays a confirmation message to the user upon successful creation of the post.

Fig. 5. Generating Sequence Diagram

2.3. Static Modeling Using Generative AI

여기서는 앞에서 작성된 순차도를 토대로 생성형 AI를 이용해서 UML의 클래스 다이어그램을 작성한다. 분석가는 생성형 AI에게 작성된 순차도의 객체들을 입력하고, 이를 토대로 각각의 Use Case 별로 관련 클래스들을 작성해 달라는 프롬프트를 전송하면, 생성형 AI가 [그림 6]과 같은 결과를 생성해 준다. 전체 Use Case들에 대해서 이와 같은 작업을 반복해서 진행된 결과들을 토대로 분석가가 검증한 후, 전체 시스템에 대한 클래스 다이어그램을 완성하면 된다.

I want to create a Class Diagram based on the Sequence Diagram for Create Post. Please extract the classes from the sequence diagram and list the ones that should be reflected in the class diagram.

Based on the **Sequence Diagram for Create Post**, here are the classes that should be reflected in the **Class Diagram**. These classes correspond to the actors and objects in the sequence diagram, grouped by their responsibilities.

1. User Class (Entity Class)

Represents a registered user interacting with the system.

Attributes:

- userID: String
- username: String
- password: String
- email: String
- role: String

Methods:

- login()
- logout()
- createPost(post: Post)

Fig. 6. Generating Class, Attributes, Methods

2.4. Generating Template Code by Generative AI

앞 절에서 생성형 AI를 통해 추출된 클래스, 클래스의 속성, 클래스의 메소드, 그리고 클래스들 간의 관계 등을 통해 작성된 클래스 다이어그램을 토대로 생성형 AI를 이용해서 소스 코드 구현도 가능하다.

I am planning to develop a web-based bulletin board system. The development languages will be Java and JSP. Based on the extracted classes above, please generate the code using Java and JSP according to the purpose of each class.

Below is the **Java** and **JSP** code for the bulletin board system based on the extracted classes. The code is organized according to the purpose of each class.

1. User Class (Java - Entity Class)

Manages the user's information and login functionality.

```
java
public class User {
    private String userID;
    private String username;
    private String password;
    private String email;
    private String role;

    public User(String userID, String username, String password, String email, String role) {
        this.userID = userID;
        this.username = username;
        this.password = password;
        this.email = email;
        this.role = role;
    }

    // Getters and Setters
    public String getUserID() { return userID; }
    public void setUserID(String userID) { this.userID = userID; }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}
```

Fig. 7. Generating Source Code

사용자는 생성형 AI에게 이전 단계에서 생성된 클래스 다이어그램의 클래스들을 입력하고, 이를 토대로 시스템에 대한 구현언어와 구현 플랫폼을 명시한 뒤, 해당 시스템에 대한 구현 코드를 생성해 달라는 프롬프트를 전송하면, 생성형 AI가 [그림 7]과 같은 결과를 생성해 준다.

생성형 AI는 각각의 클래스별로 소스 코드들을 생성해 준다. 그런데 일부 클래스의 메소드들 중에 비즈니스 로직에 해당하는 메소드들을 메소드 이름만 생성해 주고, 내부 로직은 개발자가 채워 넣도록 스텝(Stub) 코드로 제공한다. 그러나 이 부분도 데이터베이스 연동과 같은 수준의 로직의 경우는 생성형 AI에게 해당 메소드의 내부 로직을 생성해 달라는 프롬프트를 주면, [그림 8]과 같이 소스 코드를 생성해 준다.

```
private void save() {
    private String createdAt;
    private String status;

    // Constructor and other methods omitted for brevity

    public void save() {
        // JDBC 관련 변수
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            // 1. JDBC 드라이버 로드
            Class.forName("com.mysql.cj.jdbc.Driver"); // MySQL을 사용하는 경우

            // 2. 데이터베이스 연결
            String dbURL = "jdbc:mysql://localhost:3306/your_database_name"; // 데이터베이스 URL
            String dbUser = "your_db_username"; // 데이터베이스 사용자 이름
            String dbPassword = "your_db_password"; // 데이터베이스 비밀번호
            conn = DriverManager.getConnection(dbURL, dbUser, dbPassword);

            // 3. SQL 쿼리 준비
            String sql = "INSERT INTO posts (postID, title, content, authorID, createdAt, status) VALUES (?, ?, ?, ?, ?, ?)";
            pstmt = conn.prepareStatement(sql);

            // 4. PreparedStatement에 파라미터 설정
            pstmt.setString(1, postID); // 게시물 ID
            pstmt.setString(2, title); // 게시물 제목
            pstmt.setString(3, content); // 게시물 내용
            pstmt.setString(4, author.getUserID()); // 작성자의 사용자 ID
            pstmt.setString(5, createdAt); // 작성 날짜
            pstmt.setString(6, status); // 게시물 상태 (예: 'published')

            // 5. 쿼리 실행
            int rowsAffected = pstmt.executeUpdate(); // INSERT 쿼리 실행
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (pstmt != null) pstmt.close();
            if (conn != null) conn.close();
        }
    }
}
```

Fig. 8. Generating Business Logic

IV. Evaluation & Experiment

이 장에서는 본 논문에서 제안한 UML과 생성형 AI 모델을 결합한 시스템 개발 방법의 효과를 입증하고 평가하기 위해 실험 사례와 평가 결과를 제시한다. t-검정과 ANOVA 분석을 사용하여, 개발 시간, 오류 빈도, 사용자 만족도를 비교 분석하였다.

본 논문에서는 전통적인 시스템 개발 방법과 생성형 AI를 적용한 방법을 사용하여 동일한 요구 사항을 가진 두 개의 프로젝트를 진행했다. 각 방법의 효과를 평가하기 위해 개발 시간, 오류 발생 빈도, 사용자 만족도 및 비용 효율성을 중심으로 데이터를 수집하고 분석했다(표 1 참조).

- 개발 시간

생성형 AI를 통합한 프로젝트는 전통적인 방법을 사용한 프로젝트에 비해 개발 시간이 평균 30%정도 단축되었다. 통계적 분석을 통해 이 차이가 통계적으로 유의미함을 확인했다($p < 0.05$).

- 오류 발생 빈도

생성형 AI를 적용한 프로젝트에서는 오류 발생 빈도가 전통적 방법에 비해 40% 감소했다. 이 결과는 AI의 자동화된 검증 및 개선 메커니즘이 오류를 줄이는 데 효과적임을 보여준다.

- 사용자 만족도

개발자와 최종 사용자를 대상으로 한 설문조사를 통해 생성형 AI를 적용한 방법의 사용자 만족도를 평가했다. 설문 결과, 응답자의 85%가 AI 방법의 사용성과 결과물의

품질에 만족한다고 응답했으며, 이는 전통적 방법의 65%에 비해 높은 수치이므로 유의미하다고 볼 수 있다.

- 비용 효율성

프로젝트의 총 비용을 분석한 결과, 생성형 AI를 적용한 방법을 사용함으로써 인건비 및 기타 운영 비용이 전체적으로 25% 절감되었다. 비용 절감은 주로 개발 시간의 단축과 오류 수정에 소요되는 비용 감소에서 비롯되었다.

Table 1. Quantitative Comparison

Metric	Traditional Method	AI-Enhanced Method
Development Time Reduction	0%	30%
Error Reduction	0%	40%
User Satisfaction	65%	85%
Cost Reduction	0%	25%

1. Evaluation

본 논문에서 제시한 생성형 AI를 이용한 개발 방법은 기존의 관련 연구들과 비교했을 때 개발의 다양한 단계들을 간소화하고, 효율성을 향상 시키며, 오류를 줄이고, 유지보수성을 개선시킬 수 있다.

기존 AI 기술을 소프트웨어 개발에 활용하는 [7,8,9]의 연구는 AI가 생성한 UML 다이어그램의 질과 기능적 요구 사항에 초점을 맞췄으나, 품질 속성이나 다른 유형의 UML 다이어그램 생성은 반영하지 않았으며, 반복적 정제 작업이 미흡했다.

GitHub Copilot과 같은 생성형 AI 도구를 다양한 소프트웨어 개발 작업에 적용하는 [10,11]의 연구는 코드를 실

시간으로 제안하거나 생성하는 데 중점을 두었지만, 요구 사항 분석과 UML 다이어그램 생성까지의 접근은 부족했다. 또한 자동 생성된 결과의 품질이 일관되지 않거나 맥락을 완전히 파악하지 못하는 부분도 있다.

기존 연구 방법들과 제안한 방법과의 차이를 [표2]에서 볼 수 있다. [표 2]에서 보는 바와 같이 요구사항 수집 측면에서 기존 방법은 수동 인터뷰 및 분석으로 오류와 불일치가 발생하기 쉬운 반면에, 제안한 방법은 AI 지원 인터뷰와 자동화된 분석으로 요구 사항을 더 정확하고 빠르게 파악할 수 있다. 모델 생성에 있어서도 기존 방법은 분석이나 설계가가 수동으로 UML 다이어그램을 작성하기 때문에 시간이 많이 소요되고 전문 지식이 필요하다. 그러나, 제안한 방법은 요구 사항에서 자동으로 UML 다이어그램을 생성하여 빠르고 일관성 있게 처리한다. 모델 개선에 있어서도 기존 방법은 동료 검토 및 피드백을 바탕으로 반복적으로 수동 조정이 이루어지는 반면에, 제안한 방법은 AI가 주도하는 디자인 개선 제안과 자동 일관성 검사를 통해 모델을 개선한다.

V. Conclusions

본 논문에서는 UML과 생성형 AI를 활용한 시스템 개발 방법을 제시하였다. 기존 연구들이 자연어 처리(NLP)와 기계 학습 기술에 초점을 맞춘 반면, 본 연구에서는 시스템의 구조화된 설계와 유지보수를 위한 객체지향 모델링의 필요성을 강조하였다. UML과 생성형 AI를 결합한 방

Table 2. Qualitative Comparison

Comparison Item	Existing AI Application Study[7,8,9]	Generative AI Introduction Study[10,11]	Proposed Method
Scope of Application	Generating sequence diagrams from natural language	Code generation and test case creation	Comprehensive: requirements analysis, UML generation, and code implementation
Types of UML Diagrams	Sequence diagrams only (limited)	No support for UML diagrams	Full support: use case, sequence, and class diagrams
Model Refinement	None	Limited iteration and review	AI-driven iterative refinement and automated consistency checks
Context Understanding	Focus on functional requirements only	Limited understanding of code context	Comprehensive understanding of requirements and overall system context
Output Quality	Limited quality & accuracy	Inconsistent quality Contextual errors	High accuracy & Consistency
Development Time Reduction	Not reported	Not reported	30% reduction
Error Reduction	Not reported	Not reported	40% reduction

법이 시스템 개발 과정에서 개발시간 단축, 정확성, 개발 효율성, 사용자 만족도 측면에서 유의미한 결과를 보였음을 확인할 수 있었다. 이 결과는 생성형 AI가 시스템 개발 프로세스에서 효율성을 증가시키고, 개발 비용을 절감하며, 최종 제품의 품질을 향상시킬 수 있음을 입증한다. 또한, AI를 활용함으로써 개발자들이 더 창의적인 작업에 집중할 수 있도록 하여 전체적인 프로젝트 성과를 향상시킬 수 있다. 향후 연구 과제로는 다양한 도메인에서 UML을 적용한 생성형 AI 시스템의 성능을 검증하는 것이 필요하다. 현재의 UML 모델링은 개발자가 수작업으로 작성해야 하므로, 대화형 AI 시스템의 규모가 커질수록 복잡성이 증가한다. 이를 해결하기 위해, 요구사항 명세서에서 자동으로 UML 다이어그램을 생성하는 도구나 알고리즘을 개발하는 연구가 필요하다.

ACKNOWLEDGEMENT

The present research has been conducted by the Research Grant of Seoil University.

REFERENCES

- [1] Jaakko Sauvola, et. al, "Future of software development with generative AI, Automated Software Engineering", Vol. 31, No. 26, March, 2024, DOI:10.1007/s10515-024-00426-z
- [2] Ranjith Kumar Gatla, et. al, "Advancements in Generative AI: Exploring Fundamentals and Evolution", 2024 International Conference on Electronics, Computing, Communication and Control Technology(ICECCC), May 2024 DOI:10.1109/ICECC C61767.2024.10594003
- [3] Ivar Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley Longman Publishing Co., Inc., August, 2004, ISBN:978-0-201-40347-3
- [4] Amannejad, Yasaman, et al. "From requirements to software design: An automated solution for packaging software classes." Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on. IEEE, 2014, DOI:10.1109/IRI.2014.7051869
- [5] Mariana Coutinho, et al, "The Role of Generative AI in Software Development Productivity: A Pilot Case Study", Alware 2024: Proceedings of the 1st ACM International Conference on AI-Powered Software, 10 July, 2024, DOI:10.48550/arXiv.2406.00560
- [6] Javier Cámara, et al, "On the Assessment of Generative AI in Modeling Tasks: An Experience Report", Software and Systems Modeling, Volume 22, May, 2023, DOI:10.1007/s10270-023-01105-5
- [7] Ahmad Azzazi, "A Framework using NLP to automatically convert UserStories into Use Cases in Software Projects", IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.5, May 2017.
- [8] C. R. Narawita and K. Vidanage, "UML generator- use case and class diagram generation from text requirements," Int J on Adv. in ICT for Emerging Countries, vol. 10, no. 1, pp. 1-10, May 2017, DOI: 10.4038/icter.v10i1.7182
- [9] S. Ahmed, A. Ahmed, and N. U. Eisty, "Automatic Transformation of Natural to Unified Modeling Language: A Systematic Review", in 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA), May 2022, DOI: 10.1109/SERA54885.2022.9806783
- [10] H.Rukshan Piyumadu Dias, et. al., "Automated use case diagram generator using NLP", published in arXiv.org 12 June 2023, DOI:10.48550/arXiv.2306.06962
- [11] O. S. Dawood and Abd-El-Kader Sahraoui, "From Requirements Engineering to UML using Natural Language Processing - Survey Study", EJERS, European Journal of Engineering and Technology Research, Vol.2, No.1, January 2017. DOI:10.24018/ejers.2017.2.1.236
- [12] Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, Eray Tüzün, "Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT", published in arXiv.org 22 October 2023, DOI: 10.48550/arXiv.2304.10778

Authors



Eun-Sook Cho received the B.S. degree in Computer Science from DongEui University, Korea in 1993. She received the M.S and Ph.D degree in Computer Science from SoongSil University, Korea, in 1996 and

2000, respectively. Dr. Cho joined the faculty of the Department of Software Engineering at Seoil University, Seoul, Korea, in 2005. She is currently a Professor in the Department of Software Engineering, Seoil University. She is interested in framework modeling and development, Big Data, AI, Service-Oriented Computing, and IoT Applications.