

# 生成式人工智能背景下软件框架课程教学评价重构

## ——基于2021~2025年课程作业的诊断与方案设计

刘光明\*, 王晓帆\*, 赵志强, 石伟伟, 贾 萌

西安理工大学计算机科学与工程学院, 陕西 西安

收稿日期: 2026年9月6日; 录用日期: 2026年10月7日; 发布日期: 2026年10月14日

### 摘 要

生成式人工智能能够参与代码生成、测试和文档编写, 使软件框架课程中以项目功能和运行结果为主的评价方式难以充分反映学生的真实学习情况。本文以《软件工具与环境(高级)》课程2021~2025年学生提交材料为基础, 分析课程任务演进及现有评价中存在的问题。结果表明, 课程任务逐步由基础框架练习扩展到综合项目, 但现有评价仍偏重最终成果, 对学生实际理解框架运行机制过程关注不足。本文以机制理解为核心, 重构课程任务和评价方式, 通过递进式实践任务和多源学习证据考察学生的机制解释、架构判断、代码审查和任务迁移能力, 为生成式人工智能环境下软件框架课程教学改革提供参考。

### 关键词

生成式人工智能, 软件框架课程, 机制理解, 代码审查

# Reconstruction of Teaching Assessment in Software Framework Courses in the Context of Generative Artificial Intelligence

## —Diagnosis and Reform Design Based on Course Assignments from 2021 to 2025

Guangming Liu\*, Xiaofan Wang\*, Zhiqiang Zhao, Weiwei Shi, Meng Jia

School of Computer Science and Engineering, Xi'an University of Technology, Xi'an Shaanxi

Received: September 6, 2026; accepted: October 7, 2026; published: October 14, 2026

\*通讯作者。

文章引用: 刘光明, 王晓帆, 赵志强, 石伟伟, 贾萌. 生成式人工智能背景下软件框架课程教学评价重构[J]. 教育进展, 2026, 16(10): 318-328. DOI: 10.12677/ae.2026.16102114

## Abstract

Generative artificial intelligence can participate in code generation, testing, and documentation, making it difficult for evaluation methods that mainly rely on project functionality and execution results to fully reflect students' actual learning in software framework courses. Using student submissions from the course Software Tools and Environments (Advanced) between 2021 and 2025, this study analyzes the evolution of course tasks and the limitations of existing assessment practices. The results show that course tasks have gradually expanded from basic framework exercises to comprehensive projects, while current assessment still places greater emphasis on final products and provides limited evidence of students' understanding of framework runtime mechanisms. To address this issue, the study takes mechanism understanding as the core learning objective and re-designs course tasks and assessment methods. Through progressive practical tasks and multiple sources of learning evidence, students' abilities in mechanism explanation, architectural judgment, code review, and task transfer are assessed. The proposed approach provides a reference for the reform of software framework courses in the context of generative artificial intelligence.

## Keywords

Generative Artificial Intelligence, Software Framework Course, Mechanism Understanding, Code Review

Copyright © 2026 by author(s) and Hans Publishers Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

## 1. 引言

新工科建设强调人才培养体系对技术环境和产业实践变化的持续响应，并要求课程内容、教学方式与评价机制形成协调关系[1][2]。对于软件开发类课程而言，教学目标不宜停留在语法、工具或框架 API 的调用层面，应覆盖系统顶层分析、软件结构设计和复杂工程问题处理。Spring、MyBatis、Spring MVC 等框架进入课堂后，传统教学路径通常由知识点讲授、框架局部功能练习和综合项目三个环节构成。学生在这一过程中逐步熟悉 Controller、Service、Mapper 以及相关配置文件的组织方式，该模式在传统开发环境下具有较强的可操作性，也有利于形成基本的工程实践经验。

生成式人工智能改变了这一模式中成果与能力之间的对应关系。当前大语言模型可以根据自然语言要求生成项目骨架、业务代码、配置文件、测试用例和技术文档，并参与错误定位与代码重构。相关研究既肯定其降低编程实践门槛和提供即时支持的作用，也持续关注代码归属、学习过程不可见以及能力评价失真等问题[3]-[8]。在缺少生成式 AI 介入的开发环境中，一个可运行的项目通常意味着学生完成了较多编码、配置和调试工作；在 AI 辅助条件下，相似的外在成果可借助现成模板并通过大模型生成较快获得，因此由最终作品推断学生个体理解水平的可靠性随之下降。

框架开发具有明显的模板化、配置化和分层化特征，Controller、Service、Mapper 及常见配置均可由模型快速生成。形式完整且能够运行的代码只能证明当前工程在特定条件下达到预期输出，并不能直接反映学生对控制反转、动态代理、接口映射、事务边界、请求分发和视图解析等机制的掌握程度。课程答辩中能够体现理解差异的内容，主要集中在对象创建与装配过程、接口调用到 SQL 执行的转换过程、方法增强的发生位置以及异常处理和事务控制的层次边界。若评价仍主要依据功能数量、目录结构和运

行结果,个体理解差异容易被工程成品所掩盖。课程改革的关键因而不在于限制 AI 使用,而在于重新建立学习活动、能力目标与评价证据之间的稳定对应关系。

本文选取《软件工具与环境(高级)》课程 2021~2025 年的教学任务与学生提交材料进行分析。首先梳理课程任务在规模、结构和工程要求等方面的变化,并分析现有评价方式在反映学生机制理解方面存在的不足;在此基础上,结合生成式人工智能参与软件开发的新特点,重新设计课程目标、实验任务和评价方式。本文以机制理解为核心,将难以直接观察的知识掌握转化为机制解释、架构判断、代码审查和任务迁移等可观察表现,并通过递进式任务和多源学习证据将教学目标、学习过程与课程评价相衔接,为生成式人工智能环境下软件框架课程的教学改革提供参考。

## 2. 理论依据与研究设计

### 2.1. 相关研究与改革定位

软件框架及软件体系结构类课程知识覆盖面广,不同组件之间具有较强的关联性和协作性,因此,现有教学改革较多采用项目教学、案例教学等方式,强调抽象原理与工程实践的结合。相关研究通过案例分析、实践任务和教学内容重构等途径,将软件体系结构知识融入实际开发过程,以增强学生对系统设计与工程实现的整体认识[3]-[9]。这些教学方式在强化课程实践性、帮助学生建立知识与工程应用之间联系方面发挥了积极作用。但从课程评价方式来看,学生成绩仍较多依据项目实现、功能演示和文档成果进行判断,对学生掌握框架使用方法与理解框架运行机制这两类能力的区分还不够充分。

生成式人工智能进入教学场景后,程序设计和软件工程教育开始更加关注课程任务、学习过程和评价方式的调整。已有研究从教育应用风险、程序设计教学、软件工程教育以及软件设计模式教学等角度,对生成式人工智能背景下的课程改革与评价方式进行了讨论[10][11]。这些研究为软件框架课程提供了有益参考,但软件框架课程具有较强的运行机制特征。学生不仅要能够完成业务功能,还需要理解容器管理、代理增强、接口映射和请求调度等框架内部机制。如果课程评价仍主要依据项目是否能够运行和功能是否完整,就难以准确判断学生对这些机制的实际掌握程度。

本文所讨论的机制理解,是指学生能够结合代码、配置和运行结果,分析关键调用过程,说明主要组件之间的职责分工和执行关系,并判断代码或配置变化对系统运行产生的影响。由于这种理解难以通过最终项目成果直接体现,因此需要借助学生在具体任务中的表现进行评价。基于这一思路,本文将机制解释作为机制理解的直接表现,并进一步考察学生在架构判断、AI 生成代码审查和新情境迁移等方面的能力。成果导向教育强调围绕预期学习结果设计教学内容和评价方式,使课程目标、教学活动和评价保持一致[12];相关认知目标研究也指出,课程评价应关注理解、应用、分析、评价和创造等不同层次的学习表现[13]。根据以上分析,本课程在保留项目实践的基础上,将评价重点由项目完成结果进一步转向学生在解释、判断、审查和迁移等过程中的实际表现。

从认知负荷理论看,工作记忆能够同时处理的新信息数量有限,而软件框架学习具有较高的元素交互性:学生需要同时协调注解与配置、对象关系、跨层调用、开发工具以及业务规则。如果教学一开始就将完整框架、复杂业务和大量工程细节同时呈现,容易使与核心机制无关的搜索、切换和配置活动占用过多认知资源,不利于形成稳定的机制图式。本文将框架知识讲解组织为递进式任务链,先突出少量关键对象和因果关系,再逐步恢复真实框架中的复杂交互;同时采用同一项目贯穿、流程图、断点记录和调用栈等方式外化运行过程,减少频繁情境切换和无关信息加工,使学生将更多认知资源用于理解组件关系及其设计依据。

成果导向教育主要回答课程最终应形成什么能力以及如何保持目标、活动和评价的一致性,认知目标分类提供机制理解由解释、应用到分析、评价和迁移的层次参照,认知负荷理论说明复杂框架内容为

何需要分阶段呈现并控制非学习性负荷，建构主义则解释为何应通过真实问题、证据验证和反思重构来形成深层理解。

2.2. 资料来源与分析方法

研究资料包括 2021~2025 年《软件工具与环境(高级)》课程的实验任务、综合作业及学生提交材料，内容涵盖 Spring IoC、Spring AOP、MyBatis、Spring MVC 和 SSM 整合。由于不同年度在任务内容、提交要求和材料组织方式上存在差异，本文主要考察课程任务及学生作品的整体变化，不进行跨年度成绩比较。

本文采用描述性统计与内容分析相结合的方法进行研究分析。描述性统计主要用于呈现有效提交数量、项目规模以及代表性项目中源码和配置文件的变化，以反映课程任务规模。内容分析主要从任务类型、工程结构、组件协作和学习证据四个方面展开，重点考察课程任务由单一机制实验向综合项目发展的变化，以及学生提交材料对运行过程、设计依据和修改过程的反映程度。课程答辩和作业评阅记录用于补充分析典型现象。

在证据使用上，本文将历史课程材料与生成式人工智能相关研究分别处理。2021~2025 年的课程材料主要用于分析任务复杂度、工程结构及现有评价中机制理解证据不足等问题；生成式人工智能相关研究则用于分析代码生成工具参与开发后可能带来的过程信息缺失和成果识别困难等问题。基于上述分析，本文进一步提出课程任务与评价方式的改进方案，改革效果将在后续教学实践中进行验证。

3. 课程任务变化与教学问题诊断

3.1. 课程任务由框架配置向综合项目扩展

从 2021~2025 年的课程材料来看，《软件工具与环境(高级)》课程的实践任务经历了由局部框架功能学习向综合工程实践逐步扩展的过程。早期任务主要围绕 Spring、MyBatis 等框架的基本配置和单项功能实现展开，学生需要完成的工程环节相对集中。随着课程内容的推进，后续任务逐渐引入用户管理、信息管理以及综合 Web 应用等项目，控制层、业务层、持久层和前端页面等多个部分开始在同一任务中协同运行。课程实践的重点由掌握单个框架的基本使用，逐步转向在完整项目中组织不同组件并处理跨层调用关系。为进一步说明这一变化，表 1 列出了 2021~2025 年具有代表性的课程任务及学生提交情况。其中有效提交数是指经重复记录核验后纳入分析的学生提交材料数量，不包含项目内部文件数量。

Table 1. Statistics of representative course tasks from 2021 to 2025

表 1. 2021~2025 年代表性课程任务统计

| 年份   | 典型任务                | 有效提交数 | 平均大小/MB   | 最大大小/MB | 任务形态        |
|------|---------------------|-------|-----------|---------|-------------|
| 2021 | Spring 与 MyBatis 整合 | 78    | 2.69      | 34.12   | 框架配置与基础分层   |
| 2022 | 用户注册登录系列任务          | 65~68 | 3.05~3.62 | 18.66   | 业务功能实验      |
| 2023 | 用户管理综合应用            | 42    | 10.20     | 41.00   | 小型 Web 项目   |
| 2024 | 信息管理 Web 项目         | 29    | 21.43     | 43.76   | 多模块 Web 项目  |
| 2025 | 综合 Web 应用大作业        | 58    | 48.22     | 164.01  | 综合 Web 应用项目 |

表 1 显示课程任务总体呈现出由框架配置和单项业务功能练习向综合 Web 项目发展的趋势。2021 年的代表性任务主要涉及 Spring 与 MyBatis 的整合，2022 年开始增加较完整的业务功能，2023 年以后逐步形成包含多个功能模块的 Web 项目，2025 年的综合大作业已经涉及较为完整的项目结构。与这一变化相对应，代表性任务的平均压缩包大小由 2021 年的 2.69 MB 增加到 2025 年的 48.22 MB。2022 年包含同一知识系列的多次任务，因此有效提交数和平均大小采用区间表示，最大大小取该系列任务中的最大值。

相比文件规模的增加,更值得关注的是任务中工程关系的变化。早期任务主要考查对象配置、依赖管理和持久化调用等相对独立的内容,后期项目则需要同时处理请求接收、业务逻辑、数据访问、页面展示和异常处理等多个环节,不同组件之间的协作和跨层调用明显增加。这表明课程已经逐步形成基本功能机制实验与综合项目相结合的实践结构,也为理解框架在完整软件系统中的作用提供了更丰富的应用场景。

项目规模和工程结构只能反映课程任务形态的变化,不能直接作为学生知识掌握程度的判断依据。依赖库、静态资源、日志文件和项目打包方式都会影响压缩包大小,完整的目录结构也可能来自脚手架、模板复用或自动生成工具。生成式人工智能参与开发后,这种外在成果与学生实际学习过程之间的对应关系进一步减弱。因此后续分析不再侧重项目体量,而是进一步考察学生提交材料能够提供哪些与框架机制理解相关的学习证据。

3.2. 历史材料中的两类失配与 AI 条件下的新增风险

结合课程任务分析、项目评阅和课程答辩情况,现有教学主要存在三个方面的问题。首先,项目功能、运行结果和工程结构能够反映任务完成情况,但对理解框架运行机制的程度反映不足。其次,前置实验多围绕单一知识点或局部机制展开,与后续综合项目之间缺少进一步的关联和递进,学生在综合应用中难以充分建立局部机制与整体架构之间的联系。生成式人工智能能够参与代码编写、配置和文档生成,使最终作品与学生实际学习过程之间的对应关系进一步减弱,对传统评价方式提出了新的要求。前两类问题主要来源于历史课程材料的分析,第三类问题则反映生成式人工智能环境下出现的新变化。三者共同表明课程评价不能只关注最终作品,还需要增加能够反映学生理解、判断、分析和修改过程的学习证据,具体问题及其改革指向如表 2 所示。

Table 2. Major mismatches in course teaching and reform directions  
表 2. 课程教学中的主要失配及改革指向

| 失配类型           | 典型表现                                                   | 形成原因                               | 改革指向                   |
|----------------|--------------------------------------------------------|------------------------------------|------------------------|
| 作品结果与机制理解证据不匹配 | 项目能够运行,但提交材料和部分答辩表现不足以支持对依赖注入、Mapper 代理或请求分发等机制理解的充分判断 | 评价集中于功能演示和代码完整性,缺少调用链说明与因果解释任务     | 增加机制追踪、流程表达和口头答辩       |
| 项目复杂度与实验组织不匹配  | 综合项目跨越多个层次,前置实验仍主要按单个知识点展开                             | 知识点实验与综合项目之间缺少明确的递进关系,学生难以建立整体运行图景 | 建立递进任务链,并增加层次职责与依赖关系判断 |
| AI 条件下的过程证据不足  | AI 能够生成结构完整的代码,若仍只保留最终提交,教师难以判断学生对生成内容的核验、取舍和修改过程      | 最终成品无法完整呈现生成内容的来源、验证方式和人工决策        | 将 AI 输出纳入审查任务并保留关键过程证据 |

第一类问题是项目结果与机制理解之间的对应关系不够稳定。部分学生能够完成查询、登录或增删改查等功能,但在答辩中难以完整说明对象进入容器、接口调用转化为 SQL 执行、请求匹配控制器以及事务作用范围等运行过程。这说明现有评价能够较好反映功能完成情况,却难以进一步判断学生对框架内部机制的理解程度。前置实验多围绕注解、配置和 API 调用分别展开,而综合项目则要求多个框架组件协同工作,学生即使完成了各项实验,也未必能够建立从请求进入、业务处理、数据访问到结果返回的整体运行认识,这说明综合项目与前置实验之间也存在衔接不足。

生成式人工智能的应用进一步增加了评价的难度。模型可以快速生成较为完整的分层代码和配置内容,但生成结果仍需要结合业务约束、事务边界、安全要求和具体运行环境进行判断与修改。如果课程评价仍主要依据最终代码和项目结果,教师就难以了解学生在生成内容基础上进行了哪些验证、调整和工程决策,也难以判断最终成果中哪些部分真正体现了学生的理解和能力。因此生成式人工智能带来的

并不是单纯的代码生成问题，而是进一步放大了学习过程不可见、能力证据不足的问题。

综合来看，现有课程评价的主要不足在于学习证据链不够完整。历史课程材料反映出机制理解和跨层整合缺少充分证据，生成式人工智能则使最终成果与学生实际学习过程之间的对应关系进一步减弱。课程改革因此不宜简单增加任务数量，而应重新建立学习目标、实践任务和评价证据之间的联系，在保留项目实践的基础上，加强对机制解释、架构判断、代码审查和任务迁移等能力的考察。

4. 机制理解导向的改革目标与总体框架

4.1. 机制理解与四类可观察能力

课程改革应以机制理解为核心学习目标，并进一步转化为四类可以通过具体任务表现加以观察的能力。首先是机制解释能力，要求学生结合代码、配置和运行现象，分析 IoC、AOP、MyBatis 和 Spring MVC 等框架的关键运行过程，说明主要组件的职责、执行关系及其设计依据，并能够判断关键配置变化可能带来的系统行为变化。在此基础上，进一步形成架构判断能力，即依据关注点分离、依赖倒置和分层架构等原则，分析控制层、业务层和持久层的职责划分，识别参数校验、业务处理、事务控制和数据访问等职责放置中的不合理之处。

在生成式人工智能参与开发的环境下，课程还需要考察学生对生成内容的分析和修正能力。AI 代码审查能力主要体现在学生能够说明生成内容的来源和用途，识别其中可能存在的逻辑、结构、依赖、安全和可维护性问题，并通过运行验证、单元测试、文档核对或源码分析形成修改依据。与前两类能力相比，这一能力更强调学生面对不确定生成结果时的判断与验证过程，而不是对生成代码的直接接受。

任务迁移能力则用于进一步检验学生能否将已有理解应用到新的情境中。当业务对象、数据表或交互方式发生变化时，学生需要根据已有机制知识重新完成设计与实现，而不是机械复制原有项目结构。四类能力形成了由机制理解到工程判断，再到生成内容审查和新情境迁移的递进关系。代码实现仍然是课程的重要组成部分，但其评价意义需要与学生在解释、判断、审查和迁移过程中的具体表现结合起来，才能更全面地反映实际学习情况。

4.2. 诊断、目标、任务与证据的对应关系

改革方案围绕前述问题展开，并按照学习目标、教学任务和评价证据之间的一致关系进行设计。针对项目结果难以充分反映机制理解的问题，增加机制解释任务和过程性证据；针对前置实验与综合项目衔接不足的问题，通过递进式实验和迁移任务加强不同知识点及工程环节之间的联系；针对生成式人工智能参与开发后学习过程不易识别的问题，引入生成代码审查、人工修改和验证记录，保留学生分析、判断和修正的过程。与此同时，我们将机制解释、架构判断、AI 代码审查和任务迁移四类能力分别落实到具体教学活动和评价任务中，并形成相应的学习证据，具体设计见表 3。

Table 3. Alignment among course objectives, teaching activities, and assessment evidence  
表 3. 课程目标、教学活动与评价证据的对应关系

| 能力目标    | 核心教学活动                       | 主要评价任务                | 学习证据                    |
|---------|------------------------------|-----------------------|-------------------------|
| 机制解释    | 简化机制实现、断点调试、<br>调用链追踪、关键对象观察 | 解释完整运行过程<br>并预测配置修改结果 | 流程图、运行记录、答辩             |
| 架构判断    | 分层对比、反例分析、<br>职责重构、设计模式映射    | 判断代码职责分配<br>并完成重构     | 架构图、重构前后代码、<br>说明文档     |
| AI 代码审查 | 提示词设计、生成结果核验、<br>缺陷分类、人工修改   | 审查 AI 生成项目<br>并说明修改依据 | 提示词、AI 输出、<br>问题清单、修改对比 |
| 任务迁移    | 变式任务、约束变化、<br>模块扩展、独立实现      | 在新业务或新约束下<br>完成设计与交付  | 迁移任务代码、<br>测试结果、现场说明    |

上述设计形成了课程评价的基本证据链，学生能力不能仅依据最终作品进行判断，还需要通过相应任务观察其分析、解释、判断和修改过程，并结合过程记录和评价量规形成综合判断。在这一过程中，AI 生成内容既可以作为开发辅助，也可以转化为代码审查和重构任务的学习材料。课程评价不以是否使用 AI 作为评分依据，而重点考察学生对生成结果的验证、问题识别、修改依据以及在新任务中的应用表现。

## 5. 教学内容重构与实施路径

### 5.1. 围绕运行过程重组课程内容

原有课程主要按照 Spring、MyBatis 和 Spring MVC 等框架分别组织教学内容，这种安排便于知识点讲解，但不同章节之间容易停留在注解、配置和 API 调用层面，学生对各框架在实际运行中的联系认识不足。改革方案保留原有知识体系，在此基础上增加贯穿各框架的运行过程分析，使框架概念、代码结构与系统执行过程相互对应。

在具体教学中，Spring IoC 重点分析对象创建、Bean 注册、依赖解析和生命周期，使学生理解容器管理对象与直接创建对象之间的差异；AOP 围绕方法调用经过代理对象后的执行过程展开，并结合日志、事务和权限等内容说明横切逻辑与业务逻辑的分离；MyBatis 从 Mapper 接口调用出发，分析参数传递、SQL 执行和结果封装过程；Spring MVC 则以一次完整的 HTTP 请求为主线，说明 DispatcherServlet、HandlerMapping、HandlerAdapter、参数解析和响应处理之间的协作关系。设计模式和企业应用架构相关知识穿插在上述分析过程中，用于解释容器、代理、前端控制器和数据映射等设计背后的基本思想。

通过这种组织方式，课程教学由注解和配置的使用进一步转向对框架运行过程的理解。项目评价也不再只关注目录结构和功能实现，而是进一步考察各层职责、依赖关系和运行边界，使学生能够将项目结构与框架机制联系起来。

### 5.2. 递进式实验任务设计

围绕机制理解、工程判断、代码审查和迁移应用，我们将课程实践组织为五个相互衔接的阶段。简化机制实现用于呈现框架中的关键对象及其关系，真实框架追踪用于将简化实现与实际运行过程建立对应。AI 生成代码审查和人工重构侧重对生成代码进行分析、验证和修改，使学生能够依据框架机制和工程要求发现问题并说明修改理由。变式迁移通过改变业务对象或约束条件，考察学生在新任务中运用已有知识解决问题的能力。五个阶段共同形成由框架原理解逐步延伸到综合工程实践的任务链。

Table 4. Experimental task design for mechanism understanding

表 4. 面向机制理解的实验任务设计

| 实验              | 核心任务                                             | 提交材料                              | 评价重点                  |
|-----------------|--------------------------------------------------|-----------------------------------|-----------------------|
| 简化 IoC 容器       | 手写对象注册、查找和属性注入，<br>比较直接创建对象与容器管理对象的差异            | 简化容器代码、<br>对象关系图、对比说明             | 控制反转、依赖管理<br>和对象生命周期  |
| 代理机制<br>与 AOP   | 使用静态代理和 JDK 动态代理实现日志增强，<br>再迁移到 Spring AOP       | 代理实现、AOP 配置、<br>调用过程图             | 代理对象、方法拦截<br>和横切关注点分离 |
| MyBatis<br>映射流程 | 追踪 Mapper 接口调用至 SQL 执行和结果封装，<br>分析参数与返回值变化       | 调用链图、断点记录、<br>Mapper 示例           | 接口代理、参数绑定<br>和结果映射    |
| MVC<br>请求分发     | 跟踪 HTTP 请求从浏览器进入控制器<br>并返回响应的完整过程                | 请求流程图、<br>组件职责表、调试记录              | 前端控制器、处理器<br>映射和参数解析  |
| AI 辅助<br>综合项目   | 使用 AI 生成项目初稿，对分层、事务、异常、<br>权限和依赖进行审查，完成人工重构与功能扩展 | 项目代码、提示词、AI 输出、<br>问题清单、修改对比、答辩材料 | 架构判断、证据化<br>审查和迁移应用   |

任务阶段与具体实验项目属于两个不同的组织层次。表 4 中的前四个机制实验主要承担简化机制实现和真实框架追踪,综合项目则进一步融入 AI 代码审查、人工重构和变式迁移,并将前期形成的机制理解应用到完整工程情境中。机制解释主要在前两个阶段进行考察,架构判断和 AI 代码审查更多体现在代码审查与重构过程中,任务迁移则通过最后的变式任务进行检验。这样既保持了实验项目本身的完整性,也使不同能力能够在任务推进过程中逐步形成并得到评价。

前四个实验将框架中的关键机制转化为可观察、可追踪的运行过程,综合实验则将这些机制重新置于完整工程环境中加以应用。以 AOP 实验为例,实验从静态代理中增强逻辑与业务代码的耦合现象入手,引导学生分析 JDK 动态代理中的代理对象生成和方法拦截过程,并进一步对应到 Spring AOP 中切点、通知和代理调用等概念。这样的实验设计将具体代码行为与框架抽象联系起来,使学生在理解实际工程问题的过程中认识框架机制及其设计作用,减少对注解和配置的孤立记忆。

### 5.3. 同一项目贯穿的实施过程

为加强单项实验与综合项目之间的联系,课程选取规模适中的信息管理项目贯穿主要教学环节。项目内容随着机制学习逐步扩展,通过增加功能模块、调整工程结构和引入新的业务约束,使各阶段实验能够回到同一项目中进行验证。这样既保留了单项机制训练的针对性,也使学生能够持续观察不同框架组件在完整工程中的协作关系。

教学过程围绕运行现象、机制实现和框架验证展开。教师提供可运行案例或包含典型问题的代码,引导学生分析对象装配、方法增强、接口映射和请求分发等运行关系;学生通过简化实现观察容器、代理和映射等关键过程,并结合断点、日志、调用栈及必要的源码片段,在真实框架中核对相应机制。机制分析进入综合项目后,AI 可用于生成局部模块或项目初稿,学生需要依据已有知识对生成结果进行检查和修改。课程后期通过调整业务对象、数据库约束或权限要求,引导学生完成项目重构和功能扩展,使机制理解进一步落实到工程判断和实际修改中。

AI 参与开发时需要保留能够反映学生判断和修改过程的必要记录,包括关键提示词、模型输出以及人工修改内容。直接采用的代码应说明验证方式,删除或重写的部分应说明存在的问题和修改依据。涉及课程数据或个人信息的任务统一使用脱敏材料,模型生成的框架版本、依赖配置和安全建议则通过官方文档、运行测试或教师提供的资料进行核验。相关记录主要用于判断学生是否对生成结果进行了分析和验证,避免将过程记录本身变成额外的形式性任务。

## 6. 课程评价方案与后续验证设计

### 6.1. 基于多源证据的综合评价

课程评价在保留项目功能和运行结果的基础上,进一步关注学生在机制解释、架构判断、代码审查和任务迁移等方面的实际表现。项目能够正常运行仍是基本要求,但主要反映任务的完成情况,不能单独作为判断学生机制理解和工程能力的依据。课程将不同任务中形成的过程记录、分析结果和项目表现纳入综合评价,并与前述四类能力相对应,具体评价内容见表 5。

表 5 将四个评价维度进一步操作化为 1~5 分五级描述符,各维度先依据学生实际表现确定等级,再按照相应权重折算。不同等级不以代码数量或文档篇幅区分,而以解释的完整性、判断依据的充分性、验证证据的可靠性以及迁移中的独立性作为核心判据。机制解释主要依据流程图、调试记录、机制说明和项目答辩;架构判断主要依据架构图、代码审查表及重构前后对比;AI 代码审查主要依据提示词、AI 输出、问题清单、测试与修改记录;任务迁移主要依据迁移任务代码、系统演示、测试结果和现场说明。

为提高评价的可靠性,实施前可选择若干典型作品建立 5 个等级的评分样例,并组织评分者进行共

同试评。正式评价中可抽取部分作品由两名评分者独立评分，并采用一致率或组内相关系数等指标检查评分一致性；当评分差异超过预设范围时，再通过复核样例和协商评分处理。课程答辩则结合学生提交的代码和修改记录，要求学生说明关键运行过程，并根据需要完成局部代码分析或修改，使量规中的高低等级能够对应到具体、可复核的行为证据。

**Table 5.** Comprehensive course assessment rubric oriented toward mechanism understanding  
**表 5.** 机制理解导向的课程综合评价量规

| 评价维度    | 权重  | 5 分                                                       | 4 分                                    | 3 分                                      | 2 分                                      | 1 分                                  |
|---------|-----|-----------------------------------------------------------|----------------------------------------|------------------------------------------|------------------------------------------|--------------------------------------|
| 机制解释    | 30% | 能准确、完整追溯关键调用链，说明组件交互与因果关系，并结合设计原则阐明设计意图，能正确预测关键配置或代码变化的影响 | 能较完整追溯调用链并解释主要交互和设计意图，少量细节遗漏但不影响总体机制判断 | 能描述主要执行步骤和核心组件职责，但对组件间交互、因果联系或设计意图存在明显遗漏 | 只能列举零散步骤或概念，调用顺序、对象职责存在多处混淆，需要提示才能完成解释   | 不能形成基本调用链，对关键对象职责和运行机制存在根本性错误        |
| 架构判断    | 25% | 能准确识别职责边界、依赖关系和事务边界问题，依据设计原则说明原因，并提出可验证的重构方案              | 能识别主要架构问题并给出合理重构，设计依据基本充分，仅有少量边界问题遗漏   | 能发现明显的职责混杂或不合理依赖，但对问题成因、跨层影响或重构依据说明不完整   | 主要依据目录或表面结构作判断，只能发现少量问题，且存在多处误判或重构不当     | 不能区分基本层次职责，无法识别关键依赖或事务边界问题           |
| AI 代码审查 | 25% | 能系统识别逻辑、结构、依赖、安全和可维护性问题，使用文档、测试或运行证据验证，并为每项关键修改提供充分依据     | 能识别多数关键问题并进行有效验证和修改，问题覆盖或证据完整性存在少量不足   | 能识别较明显的问题并完成部分验证，但问题类型覆盖有限，对修改依据的说明不够充分  | 主要凭直觉或 AI 再次生成进行修改，验证证据较少，问题识别和修改理由均较薄弱  | 基本直接接受 AI 生成结果，不能识别关键缺陷，或无法说明验证与修改依据 |
| 任务迁移    | 20% | 能在新业务或新约束下独立完成分析、设计、实现和测试，并说明已有机制知识如何迁移及为何需要调整            | 能基本独立完成迁移任务并正确调整主要设计，少量环节需要提示或解释不够深入   | 能在一定指导下完成核心功能，但较多沿用原方案，对新约束下的适配理由说明不足    | 高度依赖原模板、示例或 AI 输出，约束变化后出现明显适配错误，难以说明迁移依据 | 不能完成关键迁移任务，也不能说明已有知识与新情境之间的联系        |

## 6.2. 后续改革成效的验证设计

现有历史材料主要用于分析课程中存在的问题，改革效果还需要在后续教学实践中进一步验证。后续可在同一课程中设置前测和后测，并在条件允许时采用平行班进行比较。测试内容与前述能力目标保持一致，包括机制解释、架构判断、AI 代码审查和任务迁移，不再仅以课程总成绩作为评价依据，而是结合机制解释、问题识别、修改理由、迁移任务完成情况和答辩表现等指标，考察学生不同能力维度上的变化。

在获得较为完整的教学数据后，可根据样本情况选择相应的统计方法分析改革前后的差异，并结合典型作品、学生访谈和 AI 使用记录进行综合判断。评价量规在实施前统一确定，部分作品采用多人评分并进行一致性检查，以减少主观评分带来的影响。通过统一任务、统一量规和多种学习证据，可以较为客观地评价改革方案的实际效果。

## 7. 讨论

随着课程任务由单项实验逐步扩展到综合项目，项目功能、工程结构和运行结果虽然能够反映任务完成情况，却难以充分说明学生对框架运行机制的实际掌握程度；生成式人工智能参与代码开发后，最

终成果与学生学习过程之间的对应关系进一步减弱。针对这一变化,课程评价在保留项目成果作为基本工程要求的同时,引入机制说明、代码审查、修改依据和迁移任务等过程性证据,以提高能力判断的完整性。

在生成式人工智能逐渐成为常用开发工具的情况下,其教学价值也不应只体现在提高编码效率。模型生成的代码在形式上往往较为完整,但仍可能存在业务逻辑、事务边界、安全设置、依赖配置和可维护性等问题。将这些生成结果纳入课堂,可以为代码分析、验证和重构提供具体材料,促使学生结合框架机制、设计原则、技术文档和运行结果作出判断。相应的教学实施需要教师准备可追踪的运行案例、典型缺陷代码和评分样例,并通过统一量规、分组答辩和局部代码修改等方式兼顾评价真实性与教学效率。对于人数较多的班级,可对必要的过程记录进行规范化,但应避免使记录本身成为额外负担。

## 8. 研究局限性与展望

为进一步检验本文所构建评价方案的适用性与稳定性,后续研究需要在更为严格和多样化的教学情境中积累证据。本研究所依据的教学资料主要来自同一所高校的一门软件框架课程,且 2021~2025 年不同年度在任务内容、提交要求和项目组织方式等方面存在一定差异,项目规模也可能受到依赖资源、模板设置和打包方式等因素影响。因此,现有历史材料更适合作为课程评价问题识别、评价指标提炼和改革方案设计的经验基础,而改革方案的实际效果仍需通过规范的教学实验进一步验证。后续可在统一任务、统一评价量规和相近教学条件下开展前测-后测研究,并在条件允许时设置平行班,同时逐步扩展至不同课程、不同编程语言和不同高校。通过学习结果量化分析与访谈、学习过程记录、学生作品分析等质性证据的相互印证,可进一步考察评价量规的信度、效度及其在不同教学情境中的迁移边界。

评价方案的持续实施还需要兼顾评价真实性与教学可操作性。本文提出以过程证据和机制答辩补充最终项目评价,有助于从单纯关注“程序能否运行”转向考察学生对框架机制、设计依据和迁移过程的真实理解,但相应地也提高了评价组织的复杂程度。教师需要准备可追踪案例和缺陷代码,建立评分样例,组织答辩并复核过程材料;如果要求学生提交过多提示词、截图和调试记录,也可能使过程性评价本身演变为新的形式化任务。实际教学中可围绕四类核心能力筛选最具判别力的证据,减少低价值材料的重复收集,并通过标准化记录模板、分阶段抽样检查、分组答辩以及部分作品双人评分等方式降低评价成本。后续还可系统比较不同证据组合下的评分一致性、教师耗时和学生负担,为大班教学形成更具可执行性的评价流程。

生成式人工智能环境的快速变化也使软件框架课程评价具有明显的动态性。不同生成式 AI 模型、框架版本和开发工具的能力持续更新,同一任务在不同时间、不同模型条件下的生成质量可能发生变化,进而改变任务本身的难度及不同年度评价结果的可比性。为提高评价过程的可追溯性,后续教学可记录学生使用的生成式 AI 模型、关键提示词、框架版本和主要依赖,并设置若干相对稳定的基准审查任务,以形成跨学期、跨工具条件下可比较的评价参照。相比直接限制学生使用某一类工具,更值得关注的是建立能够跨越具体工具变化、持续考察机制理解与工程判断能力的评价任务。

技术工具还可进一步承担过程证据采集和辅助评价功能。例如,可利用单元测试与覆盖率分析、静态代码分析、依赖与安全扫描、调用链和日志追踪、版本差异比较等方法自动提取学生项目中的客观证据,并探索使用大语言模型对机制说明、代码审查报告和调试记录进行初步标注、一致性检查或异常提示。涉及框架机制理解、设计依据解释以及迁移能力判断的评价仍应保留教师复核,并持续比较自动评价与人工评分之间的一致性及可能存在的偏差。通过跨情境验证、评价流程优化和技术辅助机制的逐步完善,可以进一步形成兼顾真实性、可靠性、可操作性与可扩展性的生成式人工智能环境下软件框架课程评价体系。

## 基金项目

2024 年西安理工大学教育教学改革研究项目(项目编号: xjy2470)和 2025 年西安理工大学教育教学改革研究项目(项目编号: xjy2550)。

## 参考文献

- [1] 钟登华. 新工科建设的内涵与行动[J]. 高等工程教育研究, 2017(3): 1-6.
- [2] 林健. 面向未来的中国新工科建设[J]. 清华大学教育研究, 2017, 38(2): 26-35.
- [3] 吴砥, 李环, 陈旭. 人工智能通用大模型教育应用影响探析[J]. 开放教育研究, 2023, 29(2): 19-25.
- [4] 吴砥, 吴河江. 通用大模型教育应用的潜在风险及其规避——基于技术伦理的视角[J]. 华东师范大学学报(教育科学版), 2024, 42(8): 64-75.
- [5] 刘凌, 吴永芬, 陈卫卫, 等. 大语言模型对高校程序设计类课程的挑战与机遇[J]. 计算机教育, 2025(2): 118-122.
- [6] 张红卓, 周小宝, 许玉焕, 等. 生成式人工智能赋能计算机程序设计类课程教学创新[J]. 计算机教育, 2024(7): 44-48.
- [7] 徐晓飞, 贺祥. 生成式人工智能赋能软件工程教育与学生能力提升[J]. 计算机教育, 2025(8): 1-8.
- [8] 肖成龙, 王珊珊. 生成式人工智能在软件设计模式课程教学中的应用[J]. 计算机教育, 2024(11): 161-166.
- [9] 高洪皓, 陈章进. 人工智能赋能程序设计课程教学改革[J]. 计算机教育, 2024(7): 41-43+48.
- [10] 覃征, 邢剑宽. 软件体系结构课程教学: 抽象与实践的协调与统一[J]. 中国大学教学, 2009(7): 14-15.
- [11] 李曦, 文艳军, 刘万伟, 等. 软件体系结构课程教学改革规划与实施[J]. 计算机教育, 2015(6): 19-21+37.
- [12] 李志义, 朱泓, 刘志军, 等. 用成果导向教育理念引导高等工程教育教学改革[J]. 高等工程教育研究, 2014(2): 29-34.
- [13] 王小明. 布卢姆认知目标分类学(修订版)的教学观[J]. 全球教育展望, 2016, 45(6): 29-39.