



软件工程教学中创造力培养的实践与思考

姚敏, 罗铁坚*

中国科学院大学 计算机科学与技术学院, 北京 101408

摘要: 在软件工程教育中, 锻造学生的持续学习力、知识构建力和创造性思维对项目成功极为关键, 而现有教学方法在培育这些能力上尚显不足。本文提出了重构课程知识体系和创新的教学方法, 设计并实现了辅助教学系统, 帮助提升学生和从业人员的核心能力。首先回顾软件工程课程的现状、知识体系, 教学模式理论以及面向创造力的教学方法。然后提出了以软件工程核心问题和任务为基础的教学内容组织方式, 重构了一个简洁和聚焦问题的课程知识体系。在教学模式方面, 强化创造力的培养, 包括教学过程建模、教学安排, 并构建了基于创造力发展连续性的动态评估模型, 以衡量学生的创造力表现, 在实际教学中取得了积极反馈, 近三年超过70%的学生在满意度评价中给出了8分及以上的高分。最后设计并实现了一个辅助课堂教学的原型系统。该系统集成了新的课程知识体系和创新的教学方法, 全面支持软件工程的教學流程, 可实时高效地提升课堂教学质量, 并促进学生创造力的发展。本文的创新教学实践和思考给软件工程教育领域提供了新的视角和方法, 将有助于培养更具创造力的软件工程人才。

关键词: 软件工程课程; 创造力培养; 知识体系重构; 动态评估模型; 辅助教学系统

中图分类号: TP3; G42

文献标识码: A

文章编号: 1674-4969(2024)05-0544-20

引言

软件可以看成是基于计算机来增强人类认知技能的抽象系统或产品, 它具有可复用性, 能够通过不同的硬件平台和操作系统实现各种功能。软件具有很强的演进性, 随着技术的进步和需求的变化, 软件会不

断更新迭代, 以满足新的需求。如今, 软件无处不在, 从日常使用的手机、电脑, 到复杂的工业控制系统, 软件都扮演着重要的角色。软件不仅深刻影响着我们从事的科学研究、生产经营和教育学习等各种人类活动, 还深刻改变了社会的组织方式和人们的生活方式。

收稿日期: 2024-05-19; **修回日期:** 2024-05-25

作者简介: 姚敏 (1997—), 女, 硕士研究生, 研究方向为学习科学、软件工程。E-mail: yaomin20@mails.ucas.ac.cn

*罗铁坚 (1962—), 男, 博士, 教授、博士生导师, 研究方向为机器智能、学习科学、软件工程。E-mail: tjluo@ucas.ac.cn (通讯作者)

引用格式: 姚敏, 罗铁坚. 软件工程教学中创造力培养的实践与思考[J]. 工程研究——跨学科视野中的工程, 2024, 16(5): 544-563. DOI: 10.3724/j.issn.1674-4969.20240060. CSTR: 32282.14.JES.20240060

Yao M, Luo T J. Practice and Reflection on Cultivating Creativity in Software Engineering Education[J]. Journal of Engineering Studies, 2024, 16(5): 544-563. DOI: 10.3724/j.issn.1674-4969.20240060. CSTR: 32282.14.JES.20240060

然而, 软件开发是一个复杂的过程, 面临着需求的不确定性、技术复杂性、团队协作、项目管理等诸多挑战。开发出一个有价值的软件, 不仅需要掌握软件开发的技术, 还需要注重软件的质量, 保证软件的可靠性、可维护性、安全性等。软件工程教学也面临着诸多挑战, 如何将理论知识与实际开发相结合, 如何培养学生的创新能力和解决问题的能力, 都是值得深入探讨的问题。

笔者做过相关研究^[1-3], 收集并分析了1968—2014年国际计算机学会(ACM)和电气与电子工程师协会(IEEE)发布的10次计算机课程大纲调整版本, 从中发现学科概念从591个增加到5824个, 知识领域从9个扩展到19个, 知识单元从10个扩展到18个。研究表明, 学科领域有新概念的产生, 同时也有概念因过时而被淘汰。由于人的学习时间有限, 要求学生全面掌握所有内容是不现实的。先前的研究者已经探索了概念之间的联系^[4-5]以及学习路径^[6-8], 旨在提高学习效率并降低学习难度。这些研究为本文知识体系的重构和教学模式的设计奠定了理论基础。

图灵奖获得者大卫·帕特森(David Patterson)在他撰写的教科书《作为服务的工程软件: 使用云计算的敏捷方法》^[9]中提到, 尽管已有大量软件开发类书籍, 他仍要撰写另一本的原因时, 他指出, 随着软件在解决领域问题上的深入和广泛应用, 以及软件技术的快速迭代和升级, 以往软件工程的组织过于杂乱, 或是关注时髦技术, 或是侧重项目管理, 缺乏具体系统的代码实现。这导致软件从业人员在开发和维护软件时常遇到认知盲区, 对新问题无从下手, 从而导致开发成本增加、项目延期和软件质量低下等不良后果。

人们在采用计算机解决问题时, 要解决的三个基本问题包括: 哪些领域问题可以转化为基于计算机系统的解决方案? 如何设计和实现更简洁高效的机器系统来解决领域问题? 如何找到解决问题的算法并发现计算机执行解决方案时的错误? 作为软件工程师, 不仅要具备扎实的技术功底, 还要具备社会责任感, 开发出对社会有益的软件。

计算机软件学科不仅继承了数学理论、实验科学和工程方法的成就, 也演化出了具有独自特色的学科

推理规则、检验标准和解决问题的方法论。大部分一线软件实践者和专家学者已经认识到, 由于一个软件的运行环境差异, 可能使用多种程序语言, 实践中遇到的问题与书本知识有距离, 创造软件就是“检验已有知识和创造新知识”的过程。

卡尔·波普尔(Karl Popper)在《科学发现的逻辑》^[10]一书中提到两个观点。第一个观点是, 研究者必须能够识别和提出问题, 并评估和选择解决方案。第二个观点是, 面对现有的理论模型或系统, 必须通过实验对其进行验证。卡尔·波普尔认为, 无论是科学研究还是工程创新, 本质上都是检验已有知识或创造新知识。

罗恩·杰弗里斯(Ron Jeffries)在《软件开发的本质》^[11]一书中提出: 在软件开发过程中, 不断提升团队能力是提高软件项目成功率的根本途径。团队不仅需要对领域问题及知识有深刻的洞见, 还需具备增量迭代开发和软件交付运行的所有相关技能。产品或服务的定位、凝练需求和测试用例、设计架构和说明文档、代码实现和综合测试、软件产品上线等任务都由团队成员来完成, 团队成员间应避免沟通协调的时间成本和理解误差。由于项目开始之初, 团队成员并没有完成以上任务的所有技能, 因此, 他们会在开发过程中遇到新问题, 需要学习新知识, 逐步形成自主的知识体系。

选择核心知识和思维能力, 并确定如何教授和训练学生, 是教育的根本问题。软件开发是设计者基于对领域知识的深刻理解所进行的创造性过程。在课程内容和学习训练的设计中, 应帮助学生理解知识的起源、发展脉络和内在逻辑, 以及这些知识能解决哪些问题, 它们的优势和局限。此外, 应指导学生建立已知与未知知识之间的联系, 旨在培养学生提出有价值问题的能力, 并能高效检验已有知识和创造新知识的思维能力。

高质量的软件系统依赖于高质量的程序代码。深刻理解程序语言和应用框架是编写高质量代码的前提和必要条件。目前共有2500多种程序设计语言, 而且还在不断诞生新的程序设计语言, 只是有些消亡了, 有些存活了下来。纵观计算机发展史, 从早期的Algol60、Cobol、Fortran到后来的C、C++、Java、

Python, 编程语言家族在不断壮大。后期涌现的 Erlang、Ruby、Io、Prolog、Scala、Clojure 和 Haskell 等语言因其独特性也吸引了众多追随者。构建云服务软件是目前互联网应用中涉及程序设计语言和信息科学技术种类最广的复杂软件工程活动。对于高级软件工程师来说, 仅仅掌握甚至精通一门程序设计语言是不够的。那么如何迅速掌握一门新程序设计语言呢? 此外, 设计优秀的软件架构以实现应用功能也是一个挑战。

软件工程作为研究和应用如何以系统性的、规范化的、可量化的过程化方法去开发和维护软件, 以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来的实践性学科, 不仅需要学生掌握理论知识, 更需要通过实际项目锻炼解决问题的能力。

随着软件系统规模的不断扩大和复杂度的不断提高, 软件工程面临着诸多挑战, 如需求的不确定性、技术更新换代快、软件质量保证等。以往以技术或过程为导向的教学模式限制了学生创造性思维的发散, 本文提出的以问题为导向的教学模式, 通过选取现实中的问题、鼓励学生提出假设并验证、组织学生进行小组讨论等方式, 引导学生创造性思考, 促进学生创造力的发展。

为了激发学生的提问能力、发现有价值的问题、提高批判思维和欣赏能力, 本文设计了旨在引导学习者关注软件产品或服务的核心问题的知识体系, 并通过最少的概念阐明创新软件的推理逻辑。而且, 为了避免学生出现“知识掌握幻觉”和“注意力分散”状态, 该知识体系强调深入理解。文章还提出了面向三大基本问题, 融合了两种求解问题的技术路线, 并旨在完成九项核心任务的软件理论框架。

本文通过引导学习者关注软件的本质问题和业务应用场景, 深入探讨软件中的数据模型和具体代码, 采用逐步迭代的讲授方法和项目实践, 帮助学习者构建软件领域的知识体系。文章提出了软件三定律, 作为检验软件价值和成败的充要条件。这些定律旨在建立一个统一的理论框架, 用以认知和检验软件。学习者可以通过这一框架分辨智能软件、基础软件、应用软件等不同类型软件问题的共性和差异, 进而提出创新性解决方案。

创造力是软件工程领域一项至关重要的能力, 但关于培养创造力的理论和实践研究相对不足。大多数现有的创造力研究集中于教学策略和模式, 而忽视了学科知识体系对提升创造力的重要作用。本文首先研究软件工程教学中培养创造力的方法, 提出面向学科根本问题的知识体系, 并倡导以问题为导向的综合性教学模式。其次, 构建一个系统性的创造力评估体系, 适用于软件工程课堂教学, 并从多个层次和维度评估学生的创造力表现。最后, 为了进一步推动有利于创造力培养的知识体系和教学模式, 设计并实现一个辅助教学系统。

1 相关工作

1.1 软件工程课程

软件工程是计算机科学与技术专业的一门核心专业课程, 是计算机科学与技术专业研究生的学科基础课; 同时也是计算机应用专业本科生的必修课, 是建立在数据结构、数据库管理系统、面向对象的程序设计等课程基础上的一门重要的理论与实践相结合的课程。该课程主要讲解当今软件工程面临的新态势, 以及为解决这些问题的新技术的发展, 包括软件架构、敏捷软件过程、测试驱动开发、软件持续交付等。

通过该课程的学习, 期望学生掌握软件系统开发的基本方法, 主要包括结构化和面向对象方法, 并能够按照工程学的方法, 组织实施软件开发项目, 保证项目达到所需的软件功能、付出合理的开发成本、具有较好的软件性能、保证开放的扩充接口和支持简单的维护操作。该课程要求掌握软件工程中所涉及的开发过程、开发方法(需求分析、软件设计、实施和系统测试)及相关工具, 并了解项目管理内容, 培养团队开发意识, 增强实际系统设计的能力。

1.2 软件工程知识体

《软件工程知识体系指南》^[12]是由电气与电子工程师协会和国际计算机学会发布的关于软件工程知识体系的权威指南, 可为软件工程师、教育者和研究人员提供一个全面的知识框架, 为课程开发和专业认证

提供基础。

该指南指出软件工程共包含 15 个知识领域 (knowledge areas, KAs), 如表 1 所示, 包括软件的设计、构建、测试、维护、管理、模型与方法、质量、经济、基础和专业实践等。其中, 前五个知识领域涵盖核心开发生命周期阶段: 需求、设计、构建、测试和维护, 与软件能力成熟度模型 (capability maturity model for software, CMM) 的第二等级类似, 不包括特定于业务的知识; 后五个知识领域专注于流程改进和管理, 涵盖软件配置管理、工程流程和软件质量等领域, 类似于 CMM 的第三、四等级。

表 1 软件工程知识领域 (KAs)

Table 1 Software engineering knowledge areas (KAs)

序号	知识领域名称
1	software requirements 软件需求
2	software design 软件设计
3	software construction 软件架构
4	software testing 软件测试
5	software maintenance 软件维护
6	software configuration management 软件配置管理
7	software engineering management 软件工程管理
8	software engineering process 软件工程过程
9	software engineering models and methods 软件工程模型与方法
10	software quality 软件质量
11	software engineering professional practice 软件工程专业实践
12	software engineering economics 软件工程经济学
13	computing foundations 计算机基础
14	mathematical foundations 数学基础
15	engineering foundations 工程基础

1.3 教学模式理论

教学模式在教学过程中扮演着至关重要的角色, 对培养学生创造力起着重要作用, 不同的教学模式因其教学方法、知识内容等而各具特色和优势。1806 年, 约翰·弗里德里希·赫尔巴特 (Johann Friedrich Herbart) 提出包含明了、联想、系统、方法的教学

四阶段理论^[13]; 20 世纪初, 玛丽亚·蒙特梭利 (Maria Montessori) 提出蒙特梭利教育法 (Montessori Education)^[14], 注重自主学习和实践活动; 1949 年, 拉尔夫·泰勒 (Ralph W. Tyler) 提出包含“确定教育目标”“选择教育经验”“组织教育经验”“评价教育计划”的教学四阶段^[15]; 2001 年, 本杰明·布鲁姆 (Benjamin Bloom) 提出包含记忆、理解、应用、分析、评估、创造六阶段^[16]的学习层次。

传统的经验式课堂教学, 学生往往被动接受知识, 从而不自主地受制于现有知识, 束缚了学生的创造性思维和质疑能力。为了改善传统教学的单一模式, 多样化的教学模式被提出。BOPPPS 教学^[17]的核心是参与式学习, 学习形式多样化, 更容易激发学生学习兴趣。“5E”教学^[18]包含吸引、探究、解释、迁移和评价五个环节, 有利于激发学生学习兴趣, 培养学生批判性思维等。“翻转课堂”教学^[19]将课堂内外学习对调, 强调学生自主学习, 利于激发学生学习兴趣, 培养学生自主学习的能力。虽然多样化的教学模式各具特色, 但不同的教学模式也存在着不同的局限性, 缺少针对特定学科的模式特色。

1.4 面向创造力的教学方法

1.4.1 培养创造力的理论 1983 年, 霍华德·加德纳 (Howard Gardner) 提出了多元智能理论, 指出衡量一个人不应仅考虑学术智力, 应该多种智能形式综合衡量, 可以多方面地培养学生创造力。维拉·约翰-斯坦纳 (Vera John-Steiner)^[20]提出了一个创造性合作的模型, 该模型确定了四种合作模式: 分布式、互补式、家庭式和整合式。为了转变传统教育体系对创造力忽视的情况, 有研究者提出了“创造力实践教育”^[21]的框架, 结合了创造力、批判性思维、创造性实践等, 促进创造力发展。创造力的影响因素是多样的, 除了教育理论、教学策略、工具等的影响外, 个人和环境因素对创造力也会产生重要影响。有学者认为可以有效影响创造力的五个关键因素包括参与、灵感、自主、协作和评估^[22], 也有学者提出创造力是社会、文化和个人三个层面互动作用产生的多层次、多维度的综合性系统^[23]。此外, 个人

情绪^[24]、环境、教师专业性^[25]、团队合作^[26]等也会对学生创造力的提升产生重要影响。促进创造力的发展需要考虑多层面、多维度影响因素之间的关系和相互影响。

1.4.2 培养创造力的实践 为了培养学生创造力和实践能力,校企合作和创新创业教育的形式出现在学生的培养模式中。在对校企合作的有效性研究中,人才理论模型^[27]证实了组织创新鼓励三维度的中介作用。为了解除统一的课程教育对学生想象力与创造力的限制,英国政府推出了“创新型伙伴关系计划”^[28],鼓励学校与创新组织及专业人士长期合作,联合培养,采用跨学科的教学方法,多策略、多样化教学,培养学生创新思维和各项技能。澳大利亚同样通过教育工作者和组织之间的合作促进教育中创造力的提高,并取得成功^[29]。东北大学提出了“创新能力培养递进模型”^[30],并基于该模型构建了“惠普性、递进式、全过程”的创新教育模式,在实践教学取得了理想效果。欧洲儿童的建筑教育^[31]通过案例或问题为引,激发儿童想象力和独立思考能力,使学生在实践和试验过程中提高创造力。大学基础物理的课程研究表明可以通过交叉训练^[32]提高学生解决问题的能力,以促进创造力的发展。此外,有研究者通过研究不同教育背景的学童的知识结构,提出通过训练塑造大脑的语义记忆结构,可以促进创造性思维的发展^[33]。

1.5 典型辅助教学系统

随着互联网的出现与高速发展,为学习和教育带来了便利性系统,比如:百度、Google、Bing、维基百科、CSDN、Quora、MOOC、腾讯课堂、网易云课堂、雨课堂、ChatGPT、讯飞星火、AskBot 智能聊天机器人等。这些系统主要可以分为五类:(1)搜索引擎类:百度、Bing、Google等;(2)知识总结类:维基百科、CSDN、Quora等;(3)线上课程类:MOOC、腾讯课堂、网易云课堂等;(4)课堂教学辅助类:雨课堂、教务系统、国科大在线等;(5)互动对话类:ChatGPT、讯飞星火、AskBot 智能聊天机器人等。

(1) 搜索引擎类。这类系统作为学习软件工程的

辅助工具,其优势在于信息量大、学习灵活、知识更新快。然而,搜索引擎提供的信息往往杂乱无章,缺乏系统性,且实践指导不足。与之相比,评价类学习能帮助学生更深入地理解软件工程概念,但需要一定的实践基础和客观的评价标准。因此,要有效学习软件工程,建议将搜索引擎的便捷性和评价类学习的深度相结合,同时辅以系统的课程和实践项目。

(2) 知识总结类。这类平台上的内容大多由用户自由贡献,知识点零散且缺乏系统性。这些平台上的信息往往是碎片化的操作经验或概念,知识点多、杂、散,既没有经过权威机构的审核,也无法保证知识体系的完整性。用户在搜索答案时,能否快速找到相关信息,很大程度上取决于问题的表述是否与平台上已有的问题高度相似。为了找到准确的答案,用户可能需要不断调整搜索关键词或提问方式。此外,这些平台的内容覆盖面广,但缺乏针对性,无法为学习者提供一个完整的、系统的学习框架。因此,虽然这些平台提供了丰富的学习资源,但学习者仍需要结合其他学习方式,才能构建系统的知识体系。

(3) 线上课程类。这类学习系统为软件工程学习提供了极大的便利。它们内容丰富,形式多样,学习时间灵活,且成本较低。然而,这些平台也存在一些不足。首先,互动性较弱,学习者难以获得及时的反馈和帮助。其次,缺乏实践机会,理论与实践脱节。此外,视频教学的被动性也限制了学习者的主动思考和发散思维。虽然这些平台为软件工程学习者提供了大量的学习资源,但要取得更好的学习效果,学习者需要结合其他学习方式,如主动提问、参与讨论、进行实践项目等,才能真正掌握软件工程知识。

(4) 课堂教学辅助类。这类辅助型系统在提升教学效率、提供个性化学习、丰富教学形式等方面具有显著优势。然而,其在人际互动、技术依赖、创造性培养等方面存在局限。因此,在实际教学中,教师应合理利用这些系统,结合传统教学方式,才能更好地促进学生的全面发展。

(5) 互动对话类。ChatGPT 作为一种强大的语言

模型, 在软件工程教学中展现出了巨大的潜力。它可以实时解答学生提出的编程问题, 提供代码示例, 并解释代码背后的原理。通过与 ChatGPT 的互动, 学生可以更深入地理解编程概念, 提高解决问题的能力。然而, ChatGPT 也存在一些局限性, 如对复杂问题的理解不够深入, 以及生成内容的客观性有待提高。因此, 在教学中, 应将 ChatGPT 作为一种辅助工具, 结合教师的引导和学生的主动学习, 才能取得更好的教学效果。

2 重构软件工程课程知识体系

2.1 理论依据

本文对数学、科学和工程学科的基本逻辑和研究范式 (表 2) 进行分析, 并通过对经典学科思维方

式、理论方法和工具系统成果等的继承和发扬, 总结了计算/信息/智能学科的研究范式 (表 3)。分析发现计算/信息/智能学科除了继承了数学理论、实验科学和工程方法的学科成就外, 也演化出具有独自特色的学科推理规则、检验标准和解决问题的方法论。无论是数学、科学和工程等经典学科, 还是计算/信息/智能学科的发展, 都需要学习者和研究者具备批判性思维、创造性思维、发散性思维、提问题能力、创造力等。

软件开发本质上是一个发现问题、解决问题、验证方案并不断迭代更新的过程。它要求开发人员面向领域问题设计和实现有价值的应用场景, 并根据反馈不断迭代优化系统。学生在学习过程中需要将不同的概念相互联系, 打破传统思维模式, 提出能够解决复杂问题的创新性解决方案, 并不断检验已有知识和创

表 2 经典学科研究范式
Table 2 Research paradigms of classical disciplines

方法论	数学	科学	工程
观察现象	观察研究对象, 以抽象方式表示, 并给予其明确定义	观察事物发生的现象, 发现其背后可能存在某些规律或模式	观察用户对设施、装置或系统等事物应具备功能的期望
提出问题 形成概念	给出研究对象之间必然有某个关系的成立的定理或命题	给出事物间存在某些因果关系的假设, 并用定性或定量的方式抽象表达成理论模型	给出构想的事物应有功能的具体需求规格说明
求解过程	用“数学推理”方法证明拟求证的定理的真假	若理论模型计算结果与实验数据相符, 则暂时认定理论有效	使用相应的材料和工艺, 给出系统设计方案并以此生产出产品
验证方式	推理是否建立在已被证实的定理或定义的基础上, 并与其他定理或命题不矛盾	理论模型是否与现有已有实验结果的理论判断不矛盾、与相关的实验结果和观察到的事实相吻合	根据需求规格说明和测试工具, 对产品进行测试验收, 符合设计要求
实际应用	把定理应用到某些应用场景中, 观察是否有效	把模型应用到某些应用场景中, 观察其是否有效	在具体环境中使用产品, 观察是否符合预期

表 3 计算/信息/智能学科研究范式
Table 3 Research paradigms of computing/information/intelligence

方法论	计算/信息/智能
观察现象	观察并分析人、自然或社会等事物的预测、决策或调控中的能力不足现象
提出问题形成概念	找出影响决策、调控或预测等的系统行为信息, 提出基于信息表达的优化模型
求解过程	把信息优化模型转化为基于计算机的界面、数据存取方式和程序代码等, 在具体应用场景中把计算机硬件或软件系统关联到拟优化事物中运行, 根据系统运行效果不断迭代完善
验证方式	设计检验事物优化效果的测试用例和数据集, 使用自动或手工方式在开发出的系统中运行, 采集系统反馈数据, 进行定性或定量对比分析, 判断系统是否满足预期的优化效果
实际应用	面向更多用户和应用场景, 把设计出的计算机软件或硬件系统安装到拟优化的设施、装置或系统中, 观察它是否能解决设施、装置或系统能力不足的问题

造新知识。

2.2 聚焦问题的知识体系

软件是人类创造的一种智力产品或服务, 它对我国的科研、生产和教学产生了深远的影响。多年来软件工程领域知识内容组织庞杂零散、叙事方式偏离核心, 导致从业人员对软件内涵和创造过程存在认识上的偏差和误区。这使得学习难度陡增, 并导致了软件开发维护成本高、质量低等后果。

以面向问题和核心任务的知识组织方式对软件工程课程知识进行梳理, 总结了面向三大基本问题, 融合两种求解问题的技术路线, 完成九项核心任务的软件工程知识体系(图1)。知识体系全面覆盖了软件设计中需要考虑的所有内容, 包括设计软件前需要思

考的实际问题, 软件设计前的调研分析, 软件设计中的模型、架构、代码、接口等的思考, 以及软件设计完成后的检测与部署维护。

以软件三大基本问题、两种技术路线和九项核心任务为主的软件领域知识理论框架, 通过引导学习者关注业务本质问题和应用场景, 深入数据模型和具体代码, 渐进迭代讲授和项目实践, 帮助读者自主构建软件领域的知识体系。本文提出了软件三定律, 并用它来建立一个认知和检验软件的统一理论框架, 以便读者能分辨智能软件、基础软件、应用软件等各自问题的共性和差异, 从而提出创新性解决方案。

2.3 传统知识体系与新建知识体系

为了解决软件工程领域知识庞杂问题, 相关领域

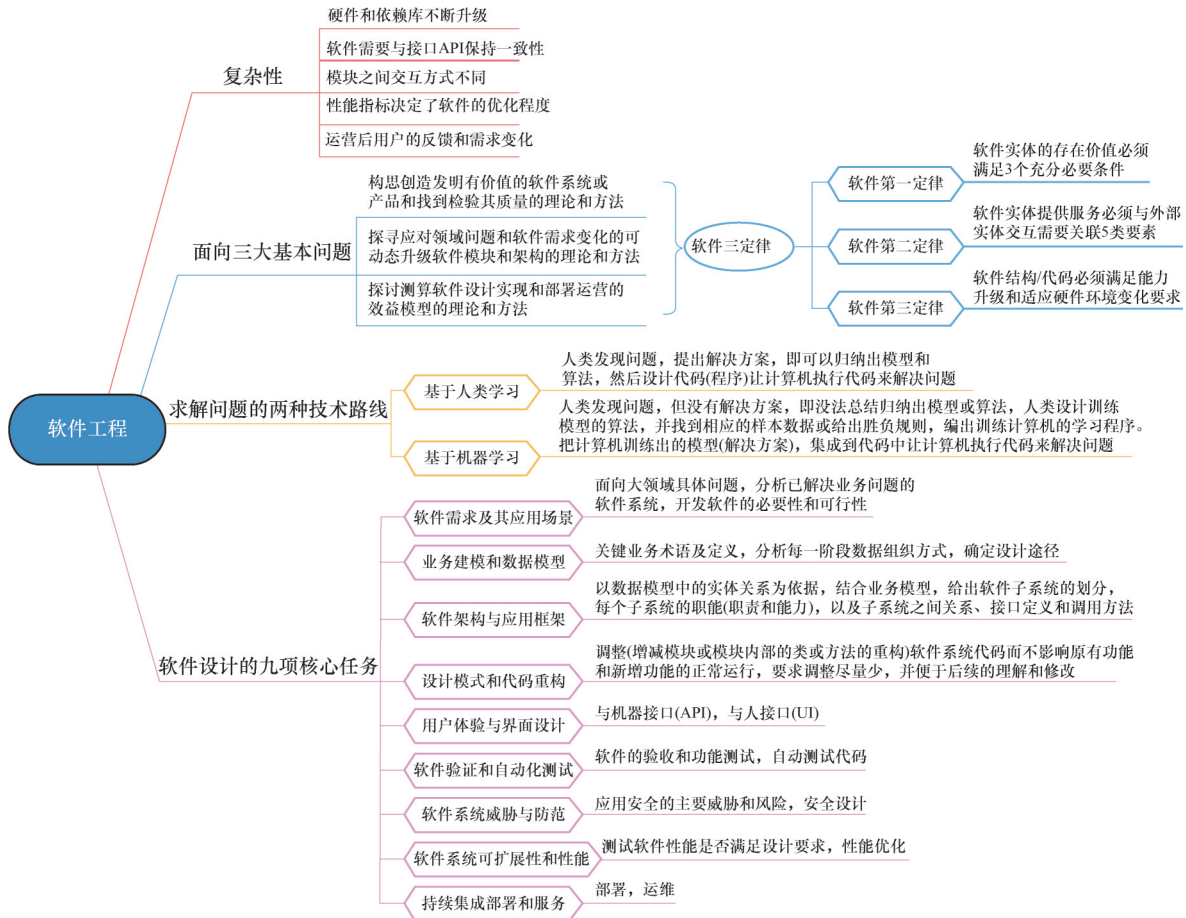


图1 聚焦问题的软件工程知识体系

Figure 1 A problem focused software engineering knowledge system

专家或学者进行了很多相关研究, 并整理出很多软件工程指导书籍。《机器学习工程》^[34]以面向机器学习训练过程方式组织知识概念, 《作为服务的工程软件》^[9]以面向工具的方式组织软件工程基于 SaaS 的知识概念, 《软件工程的技术与社会史》^[35]以面向时间的方式组织软件工程发展历史。表4展示了传统组织方式的知识体系与本文所提及的聚集问题课程知识体系之间的对比。传统的以过程、工具及时间为导向的知识组织形式, 不利于围绕少量核心概念系统性地构建学科知识体系, 同时也阻碍了知识的相互关联与拓展。

笔者总结的课程讲义共11章: 第1章介绍用软件

解决问题的范式和自主构建软件知识体系的原因, 给出本书的学习目标、预期效果和各章节内容; 第2章论证软件工程的三个基本问题、两种技术路线、九大核心任务; 第3章讲解领域问题及其应用场景; 第4章介绍业务建模与数据模型; 第5章讲解软件架构和应用框架; 第6章介绍设计模式与代码重构; 第7章讲解用户体验和界面设计; 第8章介绍软件验证与自动化测试; 第9章讲解软件系统威胁和防范; 第10章介绍软件系统可扩展性与性能; 第11章讲解软件持续集成部署和服务。该讲义的9个附录主要介绍软件开发的实践技能, 内容包括: 知识能力、代码管理、实践项目、持续集成、实践项目、应用框架分析、数

表4 聚集问题课程知识体系与传统组织方式的知识体系对比

Table 4 Comparison between the knowledge system of the clustering problem course and traditional organizational methods

章节	<i>Machine Learning Engineering</i> ^[34]	<i>Engineering Software as a Service: An Agile Approach Using Cloud Computing</i> ^[9]	<i>The Technical and Social History of Software Engineering</i> ^[35]	课程讲义
1	introduction	introduction to SaaS and agile development	prelude: computing from ancient times to the modern era	导论
2	before the project starts	the architecture of SaaS applications	1930 to 1939: the foundations of digital computing	软件理论框架
3	data collection and preparation	saas framework: introduction to ruby	1940 to 1949: computing during world war II and the postwar era	领域问题及其应用场景
4	feature engineering	saas framework: introduction to rails	1950 to 1959: starting the ascent of digital computers and software	业务建模与数据建模
5	supervised model training (part 1)	saas framework: advanced rails	1960 to 1969: the rise of business computers and business software	软件架构和应用框架
6	supervised model training (part 2)	saas client framework: JavaScript introduction	1970 to 1979: computers and software begin creating wealth	设计模式与代码重构
7	model evaluation	requirements: BDD and user stories	1980 to 1989: the rise of personal computers and personal software	用户体验和界面设计
8	model deployment	testing: test-driven development	1990 to 1999: expansion of the world wide web and the rise of dot-coms	软件验证与自动化测试
9	model serving, monitoring, and maintenance	maintenance: legacy, refactoring, and agile	2000 to 2009: the rise of social networks and economic crises	软件系统威胁和防范
10	conclusion	project management: scrum, pairs, and VCS	2010 to 2019: clouds, crowds, blogs, big data, and predictive analytics	软件系统可扩展性与性能
11	—	design patterns for SaaS classes	modern software problems	持续集成部署和服务
12	—	performance, releases, reliability, and security	a brief history of cybercrime and cyberwarfare	—
13	—	afterword	—	—

据模型案例、程序正确性证明、总复习题。该讲义以较少的核心概念系统性地构建了软件工程知识体系, 有利于知识的关联与拓展。

3 强化创造力培养的教学模式

3.1 教学过程建模

软件工程领域的创造力主要表现为两点: “0→1”, 即提出创新性的有价值的问题或工作任务; “1→n”, 即在现有工作基础上, 提出创新性方法或功能。本文提出了一种综合性教学模型, 该模型以问题为导向, 结合课堂讲授、课堂讨论、课堂练习、课后实践、文献导读和课程项目。通过这些教学方式的动态迭代, 引导学生在多个方面不断实现“0→1”和“1→n”的创新, 培养学生自主检验已有知识和创造新知识的能力(图2)。

课堂讲授以问题引导, 激发学生思考和提问, 培养发散性思维, 并提高提问能力。课堂讨论围绕知识点所涉及的开放性问题或项目案例, 提出创新性想法或解决办法体现“1→n”的创造力; 提出新问题或工作则体现“0→1”的创造力。通过课堂练习对学生核心知识的相关练习, 加深理解与掌握, 并检验已有知识; 或基于所学内容提出开放性问题, 鼓励发散思维和想象力, 进行多角度分析, 拓展对所学知识的理解。课后实践通常涉及对已有工作特定功能的实现, 通过复现已有工作检验知识; 提出新的解决方法体现“1→n”的创造力, 提出创新性问题则体现“0→1”的创造力。文献导读通过对代表性及前沿文章或书籍的学习、分析及评价, 培养批判性思维。在学习分析

时提出新问题或工作体现“0→1”的创造力; 提出新方法或功能则体现“1→n”的创造力。课程项目是对学生综合能力的考验, 既要背景意义及研究现状进行批判性分析, 又要自主解决遇到的复杂问题, 运用创造性思维实现项目创新。在项目开发中提出创新性问题或改进方法体现“0→1”和“1→n”的创造力表现。

3.2 教学安排

3.2.1 课堂教学安排 本文以中国科学院大学2013年至2024年研究生的60学时高级软件工程课程为例, 探讨如何实施新的软件工程教学模式。课程为期15周, 每周两次, 每次包含两学时。每学时开始时, 教师会点评上次的课堂练习或课后作业, 然后以问题引导, 依次进行基础概念讲解、主要问题分析及案例分析, 每节课的讲解内容如表5所示。学生在学习过程中应对每个核心任务进行纵向拓展, 掌握所需的概念、理论、算法、编程工具和语言; 同时也要进行横向拓展。以“用户体验和界面设计”任务为例, 学生知识点学习流程如图3所示。在前端编程学习中, 学生需纵向深入学习其概念、核心技术、编程语言、编程工具和语法; 在核心知识点学习时, 也需横向拓展, 如在学习编程语言时, 对多种语言进行全面学习。在涉及其他学科知识时, 同样需要进行横向拓展, 鼓励跨学科、跨领域的知识关联。

3.2.2 随堂练习 软件工程课程为期15周, 每周进行两次授课, 每次授课均包含一次课堂练习。通常在每周的第二次授课中进行课堂练习, 以巩固学生当周所学的知识和检验已有知识。表6展示了某学期的课

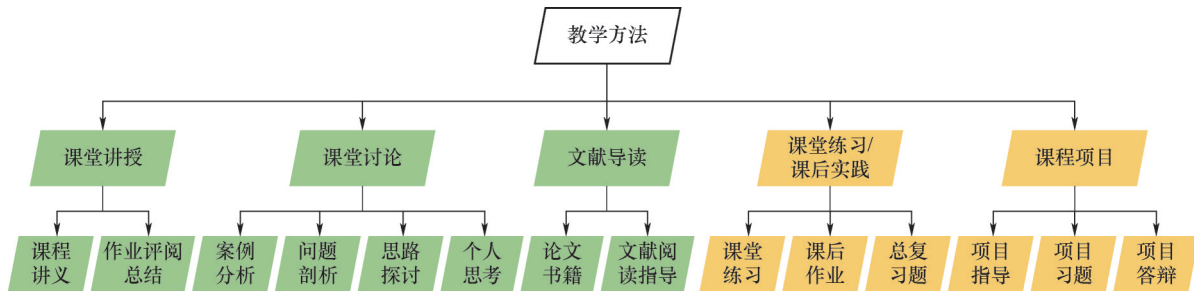


图2 软件工程教学模型

Figure 2 Software engineering teaching model

表5 课时安排
Table 5 Course schedule

章节	小节	周次	学时/h
导论	课程概述；预期学习效果；知识能力及学习策略；课程目的和考核要求	第1周	4
研究主题	软件理论的奠基者工作；软件工程的三个基本问题；软件开发的两种技术路线；软件开发的九项核心任务；练习题和阅读文献	第2周	4
领域问题及其应用场景	本章学习目的；案例1操作系统和应用框架（Linux and Ruby on Rail）；案例2交互程序设计语言环境（Jupyter notebook）；案例3玩魔方软件（Web版与实体版）；案例4池塘大战软件（智能体博弈）；案例5软件版本管理与持续集成平台（GitHub等）；练习题和阅读文献；代表性小组项目开题（4h）	第3、4周	8
业务建模和数据模型	本章学习目的；领域知识与建模语言；业务数据的外在形式与计算机的内部表示；数据表示与应用功能；练习题和阅读文献	第5、6周	8
软件架构与应用框架	本章学习目的；架构与功能；架构与数据；抽象共性与知识重用；练习题和阅读文献	第7、8周	8
设计模式和代码重构	本章学习目的；共性与特殊功能的分离；应对功能变化的策略；案例分析；练习题和阅读文献	第9周	4
用户体验与界面设计	本章学习目的；用户体验设计原则；信息表示的目的与方法；选择合适的可视化形式；练习题和阅读文献	第10周	4
软件验证和自动化测试	本章学习目的；程序正确性证明；软件验证方法；构造测试用例；练习题和阅读文献	第11周	4
软件系统威胁与防范	本章学习目的；软件安全的原因；防御措施的建构；练习题和阅读文献	第12周	4
软件系统可扩展性和性能	本章学习目的；观察性能指标；提升应用性能；练习题和阅读文献	第13周	4
持续集成部署与服务	本章学习目的；持续集成与部署；软件上线与服务模式；练习题和阅读文献	第14周	4
项目实践报告	项目演示、汇报、点评	第15周	4

堂练习安排。

3.2.3 课后实践 在15周的课程教学中，每隔两周安排一次课后实践。课后实践作业包括对已有工作的某一特定功能的复现及改进，目的是为了检验学生的已有知识并鼓励他们创造新知识。表7展示了某一学期的课后实践安排。

3.2.4 课程项目 课程项目采用“7+1”模式，其中，“7”指课程提供的项目课题，包括详细的任务要

求和实验指导（表8），学生们也可以进行更为挑战和创新的功能任务。“1”指自选题目，选择自选课题的同学需要参考提供的7个案例，撰写文档明确阐述选择该课题的目的、意义及相关任务和评分标准，教师和助教将根据同学们提供的文档决定其可行性和公平性。课程项目将在课程第七周进行开题答辩，在最后两周组织结题答辩。

学生们从上述“7+1”中选取项目，完成课题要

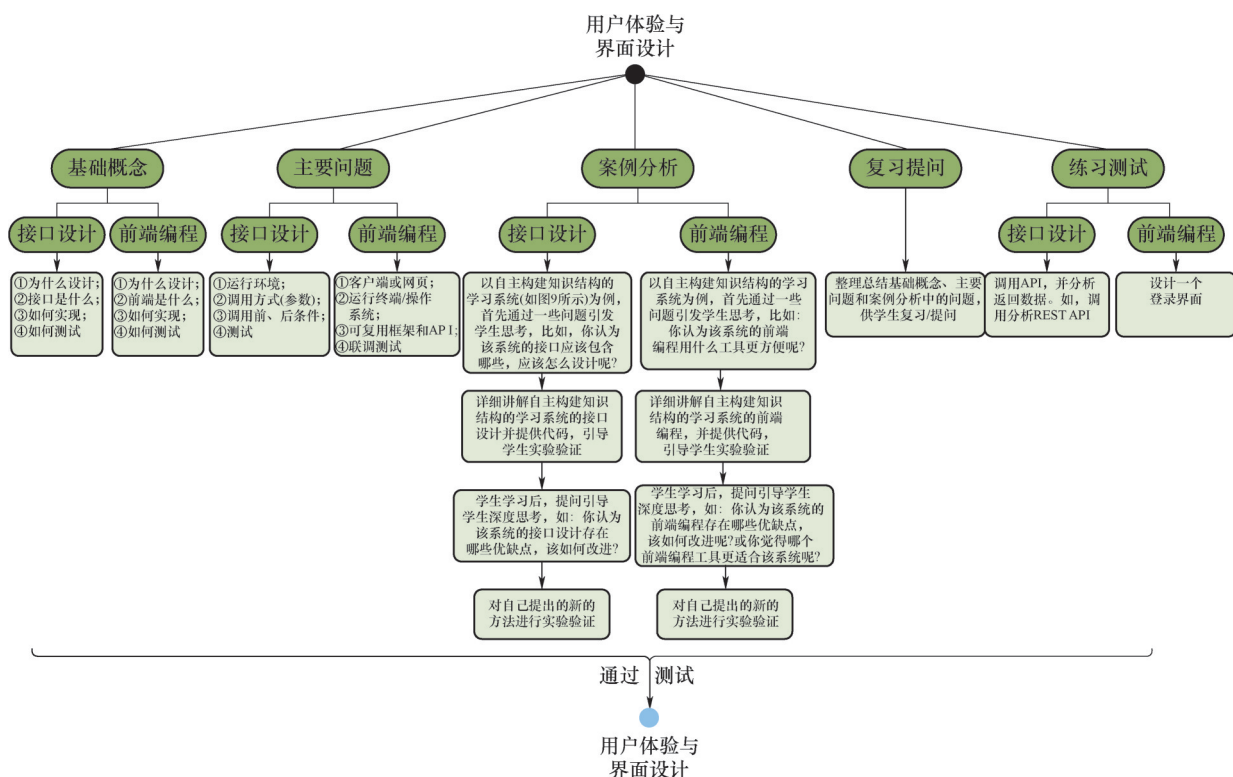


图3 学生知识点学习流程

Figure 3 Student knowledge point learning process

求的相关任务, 最终评分将依据以下考核要求: (1) 提交要求: 为了检验工作成果, 要求所有学生必须提交高质量的项目报告、Github上的代码链接、PPT、录屏和海报等。(2) 报告内容: 实验动机和问题陈述, 工作基础和对标目标, 采用的工具和方法, 实验中遇到的问题和解决办法, 实验结果的展示和详细解释, 实验总结, 参考文献和相关网址等。(3) PPT及录屏要求: 实验PPT限制在10页以内, 关键实验步骤录屏限制在10 min以内; 课程项目PPT限制在20页以内, 关键实验步骤录屏同样限制在10 min以内, PPT讲解视频限制在20 min以内。

3.3 创造力评估建模

本文结合软件工程综合性教学模式建立了一个动态评估模型, 创造力得分可以通过式(1)得到:

$$A_n = \alpha A_0 + \sum_{j=1}^4 \beta_j B_j + \gamma \frac{1}{n-1} \sum_{i=1}^{n-1} A_i \quad (1)$$

式中, A_n 为学生第 n 次创造力评估得分; A_0 为学

生创造力初次评估得分; B_j 为各阶段得分, 包括课堂练习、课后实践、测试成绩和课程项目; A_i 为学生第 i 次创造力评估得分; $\alpha, \beta, \gamma \in [0, 1]$ 分别代表各项评估得分的占比情况, 可以体现各个评估参量在创造力评估中的重要性。

在软件工程课程开始前, 采用门槛较低的Blockly Games^[36]测试方法检测学生基础编程能力和创造性思维能力, 将该次得分作为学生创造力初次评估得分 A_0 。课堂练习提交后, 由教师和助教对学生提交内容进行评阅打分, 得分记为 B_1 ; 课后实践提交后, 同样由教师和助教对学生提交内容进行评阅打分, 得分记为 B_2 ; 测试成绩记为 B_3 ; 课程项目开题报告和结题报告由教师和助教进行打分, 得分记为 B_4 。教师或助教打分从创造性思维、实践能力、解决复杂问题能力、批判性思维等方面对学生进行多方面、多层次评估。每次进行课堂练习、课后作业、测试或课程项目评分时, 都将对学生的创造力进行一次评估。

表6 课堂练习安排

Table 6 Classroom exercises

周次	主要内容
第二周	观察讨论“用深度学习构建的识别手写数字的模型”：(1) 为什么1980—2000年计算机人工智能博士论文关于手写数字识别率都不达80%?(2) 人总结规律用代码写出程序的方法与用数据集训练程序的共同点和不同点是什么?(3) 你认为要利用这个模型实现一个有价值的软件还需要做哪些工作
第三周	针对提供的几个文档材料：(1) 你认为这些文档合理吗?(2) 为什么要写文档? 如何写好软件项目文档? 我们上次给出的纲目与他们这些的区别或各自的特色是什么
第四周	(1) 用Ruby on Rails 开发一个简单版本的Blog 需要多少Gems?(2) 如果用BDD的方法, 还需要什么样的Gems
第五周	观看以往课程项目答辩视频, 分析总结项目优缺点
第六周	(1) 根据老师课堂上介绍的“基于积木块编程平台”软件系统的要求和个人理解, 写出一个项目的任务书, 包括: 面向用户、潜在价值、环境和条件、应用场景、功能需求和验收标准等。(2) 就以上练习, 有哪些困惑
第七周	评价四个小组的课程项目开题报告优缺点(从6个维度)
第八周	根据“SEMAT”(software engineering method and theory)的理论观点, 在管理一个软件项目时, 通常要进行三个层次(客户、解决方案和任务)共七方面(机会价值、利益相关者、用户需求、软件系统、常规工作、团队、思维方式)内容。回顾软件三大定律, 这个方法论的优点和不足在哪
第九周	根据课堂老师的演示和讲解, 如果你要设计和实现类似Jupiter Notebook 软件平台, 你认为需要什么技术支撑, 你设计第一版本时, 你计划选择三个主要功能来实现
第十周	(1) 在Ruby 语言中, load、require、include、extend 四种方法的区别是什么?(2) Ruby on Rails 和 Python Django 中业务数据或逻辑的模块(如rails的controller ,Django 中的view 与页面交换信息的方法是什么
第十一周	(1) 在Ruby 语言中, 哪些语言特性让你“惊奇”? 如果要你设计一种新语言, 你认为什么特性是应该增加或改进的?(2) 在屏幕上的代码增加内容, 可以输出创建实例对象的个数是多少(不应出现return)
第十二周	请在Ruby on Rails 中找出一个用yield关键词的例子并解释它
第十三周	什么是A/B测试? 为什么做A/B测试? 怎么做A/B测试
第十四周	已知一个Web应用的所有源代码, 你如何进行安全加固
第十五周	请列举出软件性能优化的评价指标以及基本的优化策略和方法

表7 课后实践安排

Table 7 Extracurricular homework

周次	主要内容
第三周	体验网站 https://blockly.games/ , 介绍自己的用时、完成情况、心得等
第五周	Keras 训练手写体数字识别及改进, 包括分析已有MNIST 代码, 模型架构、运行过程的细节及为何能达到这个准确率; 针对该任务提出自己的改进方案, 并动手实践改进的结果
第七周	根据老师对开题答辩的反馈修改开题报告, 包括: 明确用户需求, 写好需求故事; 制订测试指标; 明晰技术路线; 设计反馈函数、技术自动化路线和增强路线等
第九周	(1) 访问英译德网站: https://www.gradio.app/guides/using-blocks-like-functions , 任务是修改代码实现英译汉。(2) 访问网站 https://www.gradio.app/playground , 点击左侧Streaming Chatbot的选项, 任务是解释slow_echo(message, history)这个函数中的history的含义和作用
第十一周	(1) 猜词的应用的自动测试环境;(2) web应用框架的对比分析;(3) 客户信息CRUD操作的自动测试环境
第十三周	如图红色框中所示(GitHub 的“By Solution”), 自行设计prompt 向ChatGPT 提问, 这三个术语的相关信息, 具体包括(Prompt 和结果、实验过程、体验等写入文档): (1) 这三个组件是什么? 为什么要设计这三个组件? 该如何使用这三个组件?(2) 根据提问结果对这个三个组件的相关情况进行归纳总结, 总结差异和共性。(3) 如果想学习三个组件的使用方法, 该如何学习? ChatGPT 是否有更好的学习路径
第十五周	使用Ruby on Rails 构建个人blog 项目

表8 课程项目
Table 8 Course project

项目名称	实验目的和要求	现有实验条件	具体任务	实验项目提示	练习题 量/个
聊天机器人	Ruby 基础语法, 构建、部署聊天机器人	相关书籍	基础任务: ①配置并部署; ②项目持续集成; ③基础功能。 进阶任务: 进阶功能	ChatBot 环境配置; Ruby Lita 功能配置; 简易聊天室部署	7
魔方机器人	前后端网页开发和管理, 了解机器智能领域相关知识和应用	论文代码; 网站开发平台; 环境参考; 重现实验提示	基础任务: 类似工程或复现网站。 拓展任务: ①实现魔方教学; ②训练 5×5 魔方; ③其他功能。 创新任务: 机械智能复原魔方	开发平台安装; 环境配置; 工程创建; 前后端设计	6
基于积木块编程	为了提高用户体验, 以及为初学编程的用户提供更有针对性及更具人性化的教学软件	网页平台; 相关代码; 环境参考; 重现实验提示	基础任务: 类似工程或复现网站。 拓展任务: 新增记录每关用时和总用时功能并展示用时排行榜。 创新任务: 新增指导交互和历史代码块记录功能	网页平台介绍	6
智能体对抗小游戏	更深刻地理解与认识基于强化学习的对抗策略	相关代码; 环境参考; 重现实验提示	基础任务: 智能体对抗小游戏。 创新任务: 与强化学习相结合	环境配置; 代码简介; 训练、测试模型	8
基于视觉的手势识别与交互	学会手部检测、跟踪和手势识别的关键技术	重现实验提示; 参考文献	基础任务: 识别 5~10 种手势, 准确率 90% 及以上。 拓展任务: 人机交互应用。 创新任务: 总结研究现状, 提出问题, 并提出创新方法	搭建环境; 训练和测试	6
图片修复系统	掌握基于 MMT-Web 的前后端开发技术和基于 Pytorch 的面向视觉的深度网络模型知识, 构建、开发部署 AI 系统	重现实验提示; 参考文献	基础任务: 图片软件系统的复现。 拓展任务: 增加系统新的功能。 创新任务: 模型或系统结构创新	环境配置; Web 系统搭建; 模型训练	7
软件系统集成	(教学系统) 基于应用开发框架从零开始, 或在开源系统之上做二次开发和集成	相关代码; 环境参考	①统一用户管理; ②展示内容管理和基本功能; ③对比实现的系统与课程网站和国科大在线各自特点; ④云上测试并部署	—	9

创造力动态评估将从概念理解、实验复现、提出新的解题办法、发现并提出新问题、合理质疑、项目/问题分析、基础课程项目复现、课程项目创新等几个方面对学生在课堂练习、课后时间、课程项目等的表现进行评估打分, 以实现对学生创造力的评估。图4展示了采用新的知识体系和教学模式前后学生创造力评估对比, 图4(a)为学生初始创造力评估, 图4

(b)为接受新教学模式培养后学生期末能力评估, 图4(c)为之前教学方法下的学生期末能力评估。

3.4 课程教学评估

通过对比 2013 年、2015 年、2022 年、2023 年的选课学生成绩[图 5(a)], 可以看出面向问题和工程应用的知识组织形式以及综合性教学模式有效提高了学

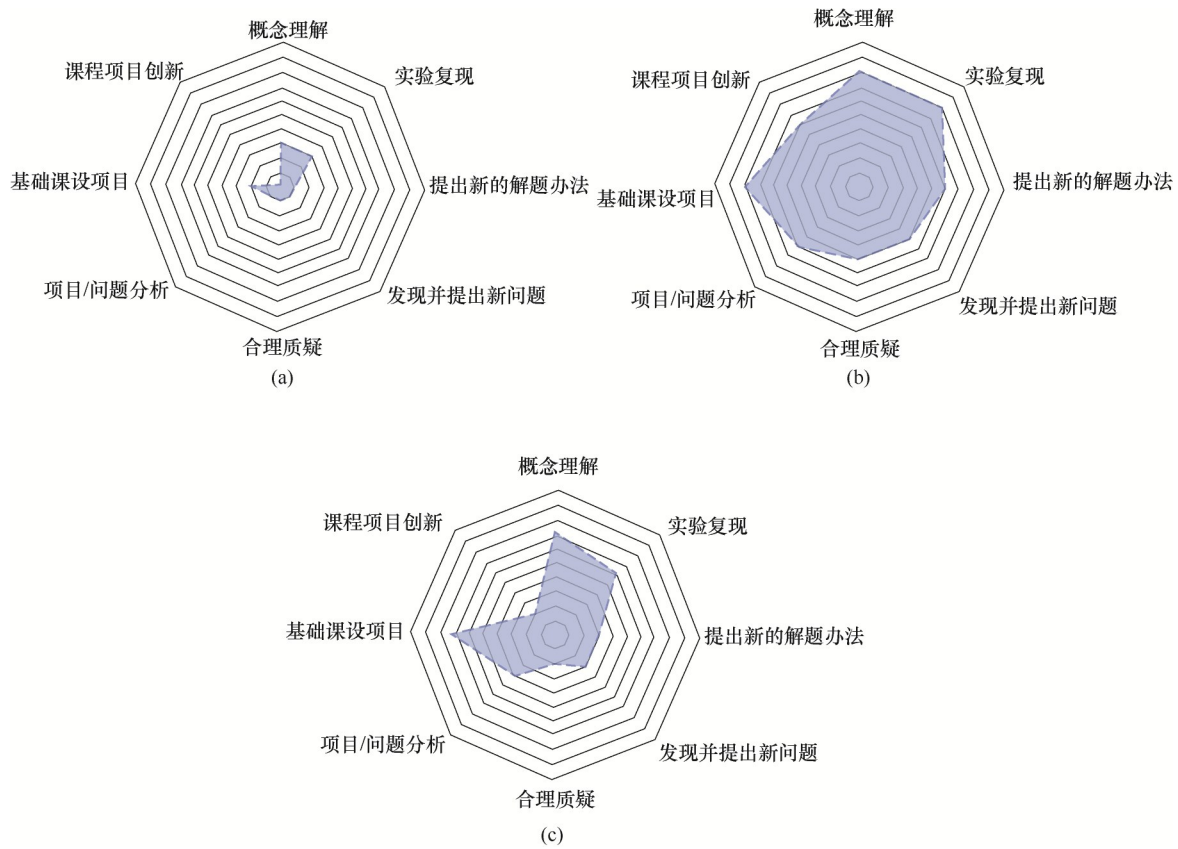


图4 学生创造力评估对比: 新教学模式下学生初始能力评估 (a); 新教学模式下学生期末能力评估 (b); 新教学模式前学生期末能力评估 (c)

Figure 4 Student creativity assessment: Initial ability assessment of students under the new teaching mode (a); Assessment of students' final abilities under the new teaching mode (b); Assessment of students' final abilities before the new teaching mode (c)

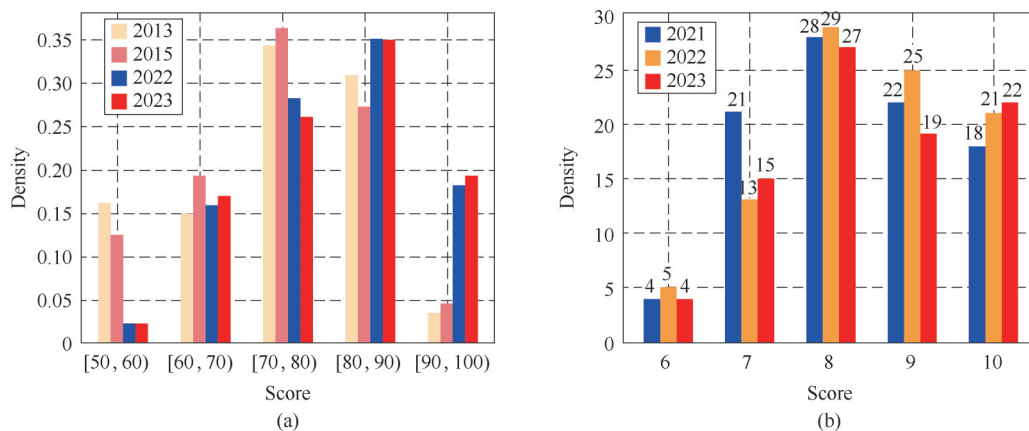


图5 学生成绩与课程评价: 2013年、2015年、2022年、2023年学生成绩 (a); 2021—2023年课程评分 (b)

Figure 5 Student grades and course evaluation: Student grades for 2013, 2015, 2022 and 2023 (a); Course evaluation from 2021 to 2023 (b)

生成成绩。在未采用所提知识体系和教学模式时, 学生平均成绩在 74 分; 应用所提知识体系与教学模式后,

学生平均成绩提升到 80 分, 且学生整体成绩显著提高。

每学年课程项目结题答辩时, 每个学生会为其他学生小组进行项目评分, 在评分表最后, 学生对学科知识内容和课程教学进行评价, 并给出自己的建议或意见。评分总分为10分, 要求学生根据自己的学习体验和收获做出真实评估。绝大多数学生对学科知识内容和课程教学的评价在8分及以上, 反映出所提学科知识体系和教学模式的有效性和实用性。近三年评分为8分及以上学生达到70%以上, 满意度评分8分及以上的比例最高达到了80%左右[图5(b)]。

4 辅助教学原型系统

4.1 设计原则

为了更好地实施本文提出的软件工程知识体系及教学模式, 设计了一款课堂教学辅助系统, 以满足教师、学生、助教的需求。图6展示了原型系统促进学生学习和能力提升的过程。系统知识体系为面向问题和核心任务的软件工程知识架构, 学习者可以通过系统知识体系系统性地学习软件工程知识概念。学习者在系统中的学习过程模拟本文提出的以问题为导向的综合性教学模式, 包括概念学习、论坛讨论、练习测试、项目实践、文献阅读等; 学习者在系统上的学习

过程数据通过创造力评估模块对学习者的创造力表现进行评估。系统从公开知识库(互联网或开放教育资源等)和内部知识库“汲取”所需知识, 同时通过不断积累/提炼系统中的数据丰富内部知识库。

4.2 系统实现

该计算机系统共三类用户, 分别是教师、助教和学生。教师登录系统后, 可以执行以下主要功能: (1) 课程知识管理、(2) 测试题库管理、(3) 测试题管理、(4) 论坛管理、(5) 学生学习情况管理。学生登录系统后, 可以执行以下主要功能: (1) 课程知识学习、(2) 参与测试、(3) 作业提交、(4) 论坛讨论、(5) 学习情况总结。助教登录系统后, 可以执行以下主要功能: (1) 测试题管理、(2) 学生作业管理、(3) 学生学习情况管理。各用户系统功能框架如图7所示。

系统数据库整合了软件工程知识体系、教学模式和创造力评估模型, 展示了知识元素与知识体系之间的关系(图8)。系统设计基于高级软件工程的教学框架, 融合了问题解决技术和教学活动, 支持软件工程的教學流程。

原型系统由Python开发完成, 框架由Flask搭建, 通过MySQL搭建系统数据库, 由PyCharm完成

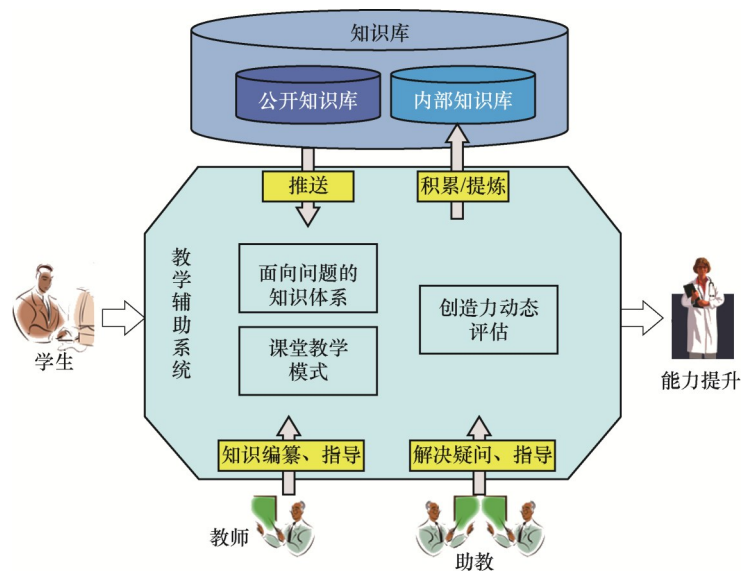


图6 原型系统促进学习框架图

Figure 6 Prototype system promotes learning framework diagram

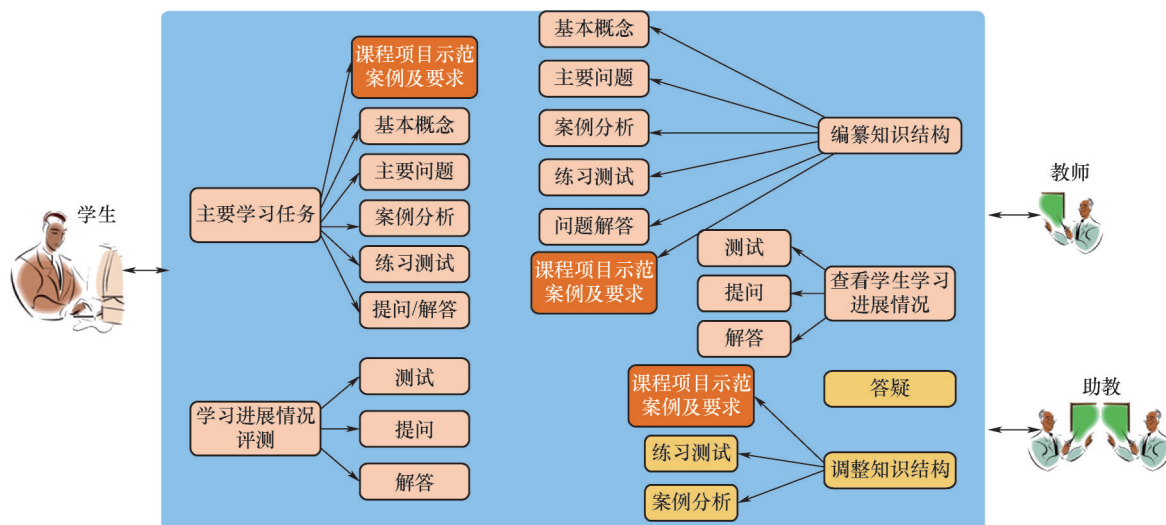


图7 系统功能框架图

Figure 7 System function framework diagram

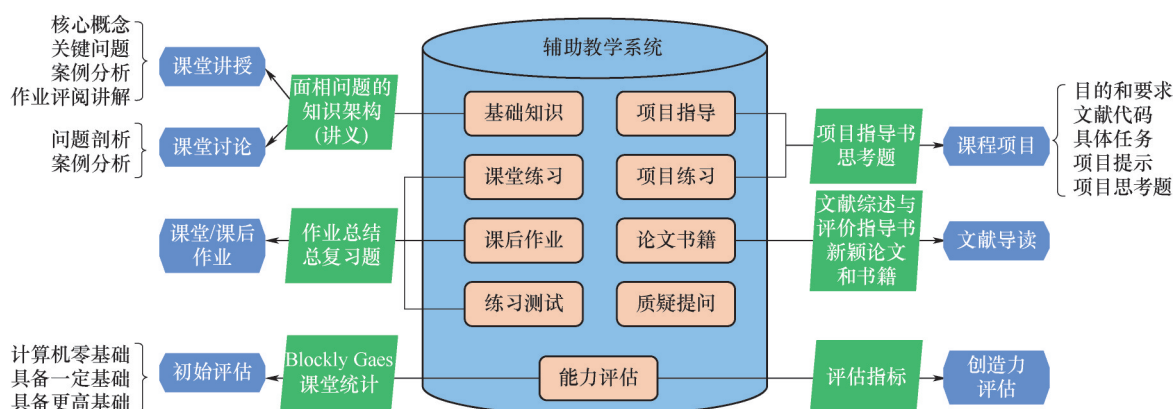


图8 系统数据库知识与知识体系关联图

Figure 8 Diagram of the relationship between system database knowledge and knowledge system

完整的系统开发。系统源码已托管在 GitHub 上, 可通过以下链接访问: <https://github.com/ym-Rain/Teaching-assistance-system>。

5 总结与展望

软件工程作为一门应用计算范式理论来构建、维护有效、实用且高质量的软件学科, 对于现代社会的发展起着至关重要的作用。然而, 软件开发过程中的不确定性和复杂性要求我们持续提出创新性的理论框架和方法, 以有效应对这些挑战。

本文针对软件工程学科知识体系的繁杂性, 探索

了一种基于核心问题和任务的优化方法, 旨在以最精简的知识概念构建一个高效的知识体系。我们从多角度出发, 提出了一种综合教学模式, 该模式通过讲授、讨论、练习、实践、文献评审和课程项目等六方面的迭代活动, 激发学生的创造力。此外, 还建立了一个动态评估模型, 用于衡量学生在不同教学活动中的创造力表现。

在实际教学中设计并实现了一个辅助软件工程课堂教学的原型系统。该系统遵循高级软件工程的教学框架, 整合了问题解决技术和多样化的教学活动, 全面支持软件工程的教学流程, 可实时高效地提升课堂教学质量, 并促进学生创造力的发展。

未来工作将继续深入探索软件工程领域创造力培养的策略和评估指标, 不断完善创造力培养模式和评估模型, 以适应不断变化的教育需求和技术进步。此外, 还将致力于推动这一培养模式的普适性应用, 使其能够广泛应用于不同的教育环境和课程设置中。通过不断的研究和实践, 可以为软件工程教育领域提供更加有效、创新的教学方法, 培养出更多具有高度创

造力和解决问题能力的软件工程人才。这不仅将为学生个人职业发展带来深远影响, 也将为整个软件工程行业乃至社会的进步做出重要贡献。

随着技术的发展和教育理念的更新, 期待能够开创软件工程教育的新篇章, 为未来的技术革新和社会发展培育出更多杰出的创新者。

参考文献

- [1] 张立波, 张飞, 罗铁坚. 计算机科学知识体系演化与评估方法[J]. 中国科学院大学学报, 2016, 33(6): 844-850.
Zhang L B, Zhang F, Luo T J. Knowledge structure evolution and evaluation method in computer science[J]. Journal of University of Chinese Academy of Sciences, 2016, 33(6): 844-850.
- [2] 陈霞, 罗晨希, 张立波, 等. 一种分析学科演化的模型及方法[J]. 工程研究——跨学科视野中的工程, 2018, 10(2): 168-179.
Chen X, Luo C X, Zhang L B, et al. Methodology of modeling and analyzing disciplinary evolution[J]. Journal of Engineering Studies, 2018, 10(2): 168-179.
- [3] Luo T J, Zhang L B, Yang L, et al. TACE: A toolkit for analyzing concept evolution in computing curricula[J/OL]. 2016, https://ksresearchorg.ipage.com/seke/seke16paper/seke16paper_17.pdf
- [4] 张立波, 孙一涵, 罗铁坚. 一种基于大规模知识库的语义相似性计算方法[J]. 计算机研究与发展, 2017, 54(11): 2576-2585.
Zhang L B, Sun Y H, Luo T J. Calculate semantic similarity based on large scale knowledge repository[J]. Journal of Computer Research and Development, 2017, 54(11): 2576-2585.
- [5] 杨林, 张立波, 罗铁坚, 等. 一种基于链接和语义关联的知识图示化方法[J]. 计算机研究与发展, 2017, 54(8): 1655-1664.
Yang L, Zhang L B, Luo T J, et al. Knowledge schematization method based on link and semantic relationship[J]. Journal of Computer Research and Development, 2017, 54(8): 1655-1664.
- [6] Liu D, Zhang L B, Luo T J, et al. Learning path generation method based on migration between concepts[M]//Knowledge Science, Engineering and Management. Cham: Springer International Publishing, 2017: 495-506.
- [7] 陈霞, 罗晨希, 张立波, 等. 一种学习路径规划的模型及方法[J]. 科研信息化技术与应用, 2017, 8(6): 3-12.
Chen X, Luo C X, Zhang L B, et al. Model and approach for learning path programming[J]. e-Science Technology & Application, 2017, 8(6): 3-12.
- [8] 罗中凯, 张立波. 学习路径规划方法[J]. 中国科学院大学学报(中英文), 2024, 41(1): 11-27.
Luo Z K, Zhang L B. Learning path planning methods[J]. Journal of University of Chinese Academy of Sciences, 2024, 41(1): 11-27.
- [9] David A P, Armando F. Engineering Software as a Service: An Agile Approach Using Cloud Computing[M]. Virginia (US): Strawberry Canyon LLC, 2013.
- [10] Popper K R. The Logic of Scientific Discovery[M]. London; New York: Routledge, 2002.
- [11] Jeffries R. The Nature of Software Development: Keep It Simple, Make It Valuable, Build It Piece by Piece[M]. Frisco, Texas: Pragmatic Programmers, 2015
- [12] Bourque P, Fairley R E. Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0[M]. 3rd edition. Washington, DC, USA: IEEE Computer Society Press, 2014.
- [13] 李毅. 赫尔巴特《普通教育学》[J]. 山东师大学报(哲学社会科学版), 1985, 30(3): 86-89.
Li Y. Herbert's *General Pedagogy*[J]. Journal of Shandong Normal University, 1985, 30(3): 86-89.
- [14] 李颖. 《蒙台梭利早期教育法》的出版与启示: 兼论蒙台梭利的儿童本位理念[J]. 出版广角, 2019(6): 94-96.
Li Y. The publication and enlightenment of *Montessori's Early Education Law*: Also on montessori's children-oriented concept[J]. View on Publishing, 2019(6): 94-96.
- [15] 拉尔夫·泰勒. 课程与教学的基本原理[M]. 北京: 中国轻工业出版社, 2016.

- Tyler R W. Basic Principles of Curriculum and Instruction[M]. Chicago: University of Chicago Press, 2013.
- [16] 卢科青, 王文, 陈占锋, 等. 基于布卢姆教育目标分类学的课程设计一致性评价[J]. 教育教学论坛, 2020(52): 234-235.
Lu K Q, Wang W, Chen Z F, et al. Consistency evaluation of curriculum design based on bloom's taxonomy of educational objectives[J]. Education Teaching Forum, 2020(52): 234-235.
- [17] 郑刚, 韩伟, 李贞源. 综合教学目标导向的课程教学体系创新: 基于BOPPPS方法的探索[J]. 高教学刊, 2024, 10(15): 63-66, 71.
Zheng G, Han W, Li Z Y. Innovation of curriculum teaching system oriented by comprehensive teaching objectives: Based on the exploration of BOPPPS method[J]. Journal of Higher Education, 2024, 10(15): 63-66, 71.
- [18] 郑登泰. 基于"5E"教学模式视域下信息科技教学实践探究[J]. 学苑教育, 2024(1): 40-42.
Zheng D T. Exploration of information technology teaching practice based on "5E" teaching mode[J]. Academic Education, 2024(1): 40-42.
- [19] 刘畅. JavaScript语言程序设计课程翻转课堂教学模式探索[J]. 绥化学院学报, 2024, 44(5): 137-139.
Liu C. Exploration on flip classroom teaching mode of JavaScript language programming course[J]. Journal of Suihua University, 2024, 44(5): 137-139.
- [20] John-Steiner V. Creative Collaboration[M]. Oxford: Oxford University Press, 2006.
- [21] Schultz E, Garcia L S, Fernandes L A, et al. Cultivating creative coexistence(s): Towards a critical education for creativity praxis to construct fairer human coexistences[C]//Proceedings of the 16th Participatory Design Conference 2020 - Participation(s) Otherwise-Volume 1. Manizales Colombia. ACM, 2020: 33-43.
- [22] Carroll E A, Latulipe C, Fung R, et al. Creativity factor evaluation: towards a standardized survey metric for creativity support[C]//Proceedings of the seventh ACM conference on Creativity and cognition. Berkeley California USA. ACM, 2009: 127-136.
- [23] Csikszentmihalyi M. Society, culture, and person: A systems view of creativity[M]// The Systems Model of Creativity. Dordrecht: Springer, 2014: 47-61.
- [24] de Rooij A, Jones S. Mood and creativity: An appraisal tendency perspective[C]//Proceedings of the 9th ACM Conference on Creativity & Cognition. Sydney Australia. ACM, 2013: 362-365.
- [25] Michael J. Creativity in education: The role of classroom environment and teacher expertise[J]. Educational Psychology Review, 2021(33): 123-145.
- [26] Barrett M S, Creech A, Zhukov K. Creative collaboration and collaborative creativity: A systematic literature review[J]. Frontiers in Psychology, 2021, 12: 713445.
- [27] 覃庆华. 校企合作教育对创新型人才创造力培养的质性研究[J]. 百色学院学报, 2019, 32(6): 125-134.
Qin Q H. A qualitative study on the effect of school-enterprise cooperation education on the cultivation of innovative talents' creativity[J]. Journal of Baise University, 2019, 32(6): 125-134.
- [28] 杨梅, 赵巍. 国家课程束缚下的英国创造力教育: 英国"创新型伙伴关系计划"述评[J]. 教师教育学报, 2018, 5(5): 90-98.
Yang M, Zhao W. Creativity cultivation in Britain under the constraints of national curriculum: A review of "creative partnership program"[J]. Journal of Teacher Education, 2018, 5(5): 90-98.
- [29] Lassig C J. Promoting creativity in education: from policy to practice: An Australian perspective[C]//Proceedings of the Seventh ACM Conference on Creativity and Cognition. Berkeley California USA. ACM, 2009: 229-238.
- [30] 张立志, 黄晓颖. 基于创造力结构理论的工科高校创新教育模式探究[J]. 教育与职业, 2014(36): 170-172.
Zhang L Z, Huang X Y. Research on innovative education mode in engineering colleges based on creativity structure theory[J]. Education and Vocation, 2014(36): 170-172.
- [31] 孙雪丹, 刘楠. 创新力与创造力培养: 欧洲儿童建筑教育初探[J]. 美术研究, 2021(2): 130-136.
Sun X D, Liu N. Innovation and creativity cultivation: A preliminary study of European children's architectural education[J]. Art Research, 2021(2): 130-136.
- [32] Samani J, Pan S C. Interleaved practice enhances memory and problem-solving ability in undergraduate physics[J]. NPJ Science of Learning, 2021, 6: 32.
- [33] Denervaud S, Christensen A P, Kenett Y N, et al. Education shapes the structure of semantic memory and impacts creative thinking[J]. NPJ Science of Learning, 2021, 6: 35.
- [34] Andriy B. Machine Learning Engineering[M]. Quebec, Canada: True Positive Inc., 2020.
- [35] Jones C. The Technical and Social History of Software Engineering[M]. Boston, USA: Addison Wesley, 2013.
- [36] Blockly Games[EB/OL]. [2024-05-20]. <https://blockly.games/>.

Practice and Reflection on Cultivating Creativity in Software Engineering Education

Yao Min, Luo Tiejian *

School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China

Abstract: Software can be seen as an abstract system or product based on computers that enhances human cognitive skills. It is reusable and capable of implementing various functions across different hardware platforms and operating systems. Software is highly evolutionary; as technology advances and demands change, software continuously updates and iterates to meet new requirements. Today, software is ubiquitous, playing a crucial role in everything from everyday devices like smartphones and computers to complex industrial control systems. Software not only profoundly impacts scientific research, production operations, and educational activities but also significantly alters societal organization and people's lifestyles.

However, software development is a complex process that faces many challenges, including uncertainty in requirements, technical complexity, team collaboration, and project management. Developing valuable software requires not only mastering software development techniques but also focusing on software quality to ensure reliability, maintainability, and security. Software engineering education also faces numerous challenges, such as integrating theoretical knowledge with practical development and fostering students' creativity and problem-solving abilities, all of which are worthy of in-depth exploration.

Turing Award winner David Patterson, in his textbook *Engineering software as a service: An agile approach using cloud computing*, he mentioned the reasons why needed another book on software development despite the abundance of existing literature. He pointed out that as software becomes more deeply and widely applied to solve domain-specific problems, and as software technology rapidly iterates and upgrades, previous knowledge organization in software engineering has become too fragmented, often focusing on trendy technologies or project management, while lacking detailed and systematic code implementations. This has led software practitioners to encounter cognitive blind spots during software development and maintenance, leaving them at a loss when faced with new problems, thereby increasing development costs, causing project delays, and leading to poor software quality.

The authors have conducted related research, collecting and analyzing 10 versions of computer science curriculum guidelines published by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) from 1968 to 2014. We found that the number of discipline concepts increased from 591 to 5824, the knowledge areas expanded from 9 to 19, and the knowledge units grew from 10 to 18. The study concludes that new concepts emerge in the discipline while outdated ones are eliminated. Given the limited time for learning, it is unrealistic to expect students to master all content comprehensively. These studies lay the theoretical foundation for the restructuring of the knowledge system and the design of teaching models presented in this paper.

When using computers to solve problems, three core questions must first be answered: Which domain problems can be transformed into computer-based solutions? How can we design and implement simple and efficient systems to address these problems? And how can we find appropriate algorithms, detect, and improve errors during execution? Therefore, software engineers need not only a solid technical foundation but also broad cross-disciplinary knowledge and a sense of social responsibility to develop software that truly benefits society.

The computer software discipline has inherited the essence of mathematical theory, experimental science, and engineering methods, while also developing unique reasoning rules, testing standards, and problem-solving methodologies. Many software practitioners and scholars have recognized that, due to differences in software operating environments and the variety of programming languages that may be used, the problems encountered in practice often differ from those described in textbooks. As a result, the process of creating software is, in essence, a process of "examining existing knowledge and creating new knowledge".

Software engineering is a discipline that is highly practical in nature, and the process of software development is one that faces numerous complex challenges. In software engineering education, cultivating students' abilities in continuous learning, knowledge construction, and creative thinking is crucial to the success of projects. However, current teaching methods remain insufficient in fostering these abilities. Additionally, the vast and complex body of knowledge in the field significantly impacts students' learning efficiency. This paper begins by investigating and analyzing existing textbooks on software development and software engineering. The research finds that while there is an abundance of books on software development, the rapid advancement of software technology has rendered the previous knowledge organization—often oriented toward time, process, or tools—too fragmented, with differing focuses, and lacking systematic case code analysis.

This paper deeply analyzes the logic and research paradigms of mathematics, science, and engineering disciplines, and on this foundation, it inherits and develops the classic thinking methods, theoretical approaches, and tool systems of these disciplines, summarizing the research paradigms of computing disciplines. The results show that computing disciplines have inherited the achievements of mathematical theory, experimental science, and engineering methods, while also developing unique reasoning rules, testing standards, and problem-solving methodologies. Whether in traditional disciplines such as mathematics, science, and engineering, or in emerging fields such as computing, information, and intelligence, development requires learners and researchers to possess critical thinking, creative thinking, divergent thinking, questioning ability, and creativity.

This paper reconstructs a theoretical framework of software domain knowledge based on the core issues and tasks of software engineering, focusing on three fundamental software problems, two technical approaches, and nine core tasks. This framework comprehensively covers all aspects that need to be considered in the software design process, including analyzing real-world problems before design, designing models, architectures, code, and interfaces during the software design process, as well as testing, deployment, and maintenance after the design is completed. By guiding learners to focus on essential business issues and application scenarios, delving into data models and specific code, and using a gradual iterative teaching method and project practice, this framework helps learners independently construct a knowledge system in the software domain. Additionally, the paper introduces three laws of software to establish a unified theoretical framework for understanding and testing software, enabling learners to distinguish the commonalities and differences in intelligent software, foundational software, and application software, and to propose innovative solutions. Finally, this paper reviews existing teaching mode theories and creativity-oriented teaching methods, finding that there is a relative lack of theoretical and practical research on creativity cultivation in the field of software engineering.

This paper proposes a comprehensive teaching model that is problem-oriented, combining classroom lectures, discussions, exercises, practice, literature reading, and course projects to strengthen the cultivation of students' creativity. Through dynamic iterations of various teaching methods, students are encouraged to innovate across multiple areas, transitioning from disruption innovation to incremental innovation, while developing their ability to independently assess existing knowledge and generate new ideas. At the same time, this paper also constructs a dynamic evaluation model based on the continuity of creativity development to measure students' creative performance, which has received positive feedback in actual teaching. This dynamic, iterative approach has been implemented in the advanced software engineering course at the University of Chinese Academy of Sciences from 2013 to 2024, receiving high student satisfaction in recent years. This paper also presents the specific implementation process of the proposed teaching model, including the arrangement of various aspects and the use of the dynamic evaluation model.

Finally, through analyzing the advantages and disadvantages of typical auxiliary teaching systems, this paper designs and implements a prototype system to support classroom teaching. This system integrates a problem- and core-task-oriented software engineering knowledge system, fully supporting a problem-oriented comprehensive teaching model in software engineering, capable of improving classroom teaching quality in real-time and efficiently, while also promoting the development of students' creativity. The prototype system acquires the necessary knowledge from public and internal knowledge bases and continuously accumulates and refines user data to enrich the internal knowledge base. The innovative teaching practices and reflections presented in this paper provide new perspectives and methods for the field of software engineering education, contributing to the cultivation of more creative software engineering talent.

Keywords: software engineering course; creativity cultivation; knowledge system restructuring; dynamic assessment model; assistant teaching system