分类号：TP391　　学校代码：10363
密　级：公开　　学　　号：2210910107

安徽工程大学
Anhui Polytechnic University

硕士学位论文

题目　基于编程现场的编程新手编程技能度量研究

论 文 作 者　查方慧
指 导 教 师　王勇 教授 崔琳 教授
学科（专业）　软件工程
研 究 方 向　领域软件工程

论文提交日期：2024 年 5 月 23 日

分类号：TP391　　单位代码：10363
密　级：公开　　学　　号：2210910107

题　　目　基于编程现场的编程新手编程技能度量研究

英文并列

题　　目　Research on measuring programming skills of novice programmers based on coding scene

学 生 姓 名：查方慧
指 导 教 师：王勇（教授）/崔琳（教授）
专　　业：软件工程
研 究 方 向：领域软件工程

论文答辩日期：　2024 年 5 月 24 日

安徽工程大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律后果由本人承担。

学位论文作者签名：查方慧

日期：2024年5月23日

安徽工程大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权安徽工程大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密 □，在_____年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名：查方慧　　指导教师签名：王勇

日期：2024年5月23日　　日期：2024年5月23日

基于编程现场的编程新手编程技能度量研究

摘要

编程新手通常是指学过一门编程语言而尚未能独立开发软件系统的初级程序员。编程新手只能完成简单编程任务，需要不断学习才能进阶成为专业程序员。近年来，大语言模型具有超强“涌现”能力，编程新手所完成的简单编程任务可以被代码大语言模型自动完成。在当前的软件行业中，编程新手需要快速成长为专业程序员才能避免被淘汰的风险。度量编程新手的编程技能能为编程新手持续学习提供帮助。因此，如何度量编程新手的编程技能具有重要意义。

编程现场特指程序员编写程序的现场，包含了程序员在编程过程中所有产生的事件和行为。在编程现场中，编程新手主要是根据软件需求编写程序代码，再对软件代码进行测试。在本文中，将编程新手假设为学过编程语言的高年级本科学生，将编程新手在编程现场中的表现作为度量对象，具体研究如下：

（1）提出了一种基于编程测试的编程新手编程技能度量方法。在编程现场中，编程新手技能的主要表现为编程效率和代码的正确性。编程新手通过在线编程练习平台编写程序，并通过测试验证代码的正确性。该方法中，首先给出软件测试用例生成方法，随后，依据编程效率和程序正确性指标给出编程新手技能度量方法。最后，设计实验验证该方法的有效性。实验结果显示，该方法能度量编程新手的编程技能。

（2）提出了一种基于编程反馈的编程新手技能度量方法。在编程新手的编程场景中，大多编程新手需要多次迭代，也就是多次提交修改后代码，才能够完成编程任务。如何利用编程迭代的反馈信息提高编程新手编程技能的度量准确性具有重要意义。本文通过构建编程反馈技能追踪模型来度量编程新手在编程反馈练习过程中的编程技能。首先，分别通过反馈信息图来表示解决方案的代码特征和反馈跟踪图来测量相邻提交的代码之间的变化。然后，利用反馈信息图和反馈跟踪图分别对编程新手的编程知识和编码能力进行建模，最终实现对编程技能水平更细粒度的评估。

（3）在实际场景中，编程新手需要通过课程学习不断提升自身的编程技能。针对这一场景，本文提出一种编程新手编程技能与课程学习的相关性研究。在该探索性研究中，主要分析在 OBE 理念教学体系下的课程成绩能否度量新手的编程技能。首先，筛选和收集编程新手的课程成绩历史数据，然后对其进行数据预处理。最后，将编程新手的课程成绩与编程测试结果进行相关性分析，获得相关性显著的课程组。

关键词：编程现场，编程新手，技能度量，软件测试，编程反馈

RESEARCH ON MEASURING PROGRAMMING SKILLS OF NOVICE PROGRAMMERS BASED ON CODING SCENE

ABSTRACT

Novice programmers generally refer to junior programmers who have learned a programming language and are not capable of developing software systems independently yet. Novice programmers can only perform simple programming tasks and need to continue to learn to become professional programmers. In recent years, large language models have super strong "emergence" ability, and simple programming tasks completed by novice programmers can be automatically completed by code large language models. In the current software industry, novice programmers need to quickly grow into professional programmers to avoid the risk of being eliminated. Measuring the programming skills of novice programmers can help them to continue learning. Therefore, how to measure the programming skills of novice programmers is of great significance.

coding scene refers to the site where programmers write programs, including all the events and behaviors generated by programmers in the process of programming. In the programming field, novice programmers

mainly write program code according to software requirements, and then test the software code. In this dissertation, novice programmers are assumed to be senior undergraduate students who have learned programming languages. The performance of novice programmers in the coding scene is used as the measurement object, and the specific research is as follows:

(1) This dissertation proposed a method to measure programming skills of novice programmers based on programming test. In the coding scene, the main performance of novice programming skills is programming efficiency and correctness of code. Novice programmers write programs through the online programming exercise platform, and verify the correctness of the code through tests. In this method, firstly, the method of generating software test cases is given, and then the measurement method of novice programming skills is given according to the indicators of programming efficiency and program correctness. Finally, an experiment was designed to verify the effectiveness of the proposed method. Experimental results show that this method can measure the programming ability of novice programmers.

(2) This dissertation proposes a method to measure novice programming skills based on programming feedback. In the coding scene of novice programmers, most new programmers need multiple iterations, that is, multiple commits of modified code, before they can complete

their programming tasks. How to use the feedback information of programming iterations to improve the measurement accuracy of programming skills for novice programmers is of great significance. This dissertation constructs a programming feedback skill tracking model to measure the programming skills of novice programmers in the process of programming feedback exercises. Firstly, the feedback information graph is used to represent the code characteristics of the solution and the feedback trace graph is used to measure the changes between adjacent committed codes, respectively. Then, the feedback information graph and feedback trace graph were used to model the programming knowledge and coding ability of programming novices respectively, and finally a more fine-grained evaluation of programming skill level was realized.

(3) In actual scenarios, novice programmers need to continuously improve their programming ability through course learning. In view of this scenario, this dissertation proposes a correlation study between programming skills and curriculum learning of programming beginners. In this exploratory research, it mainly analyzes whether the course performance under the OBE concept teaching system can measure the programming skills of novices. Firstly, the historical data of course performance of programming novices were screened and collected, and the data preprocessing was carried out. Then, the course scores of novice programmers were correlated with the programming test results to obtain

the course groups with significant correlations.

KEY WORDS: coding scene, programming novice, skill measurement, software testing, programming feedback

目录

第 1 章 绪论

1.1 研究背景与意义

随着大语言模型的兴起，编程领域正在发生变革[1]。这些模型不仅提高了编程效率，还推动了编程范式的革新[2]。虽然这为自动化编码创造了条件，但也提高了对编程新手的编程要求。基础编程工作的自动化使编程新手需要快速适应变化并提高技能以保持竞争力。然而，没有评估就无法改进，只有开发和运用更有效的方法来度量编程新手的编程技能水平，才能帮助其清晰地认识到自身的编程相关问题，从而帮助编程新手进阶成为更专业的程序员[3]。

在现实编程场景中，编程新手主要是指那些学完一门编程语言到能独立开发软件系统这一阶段的程序员。编程现场就是程序员在实际的编程场景中，完成编码的过程[4]。而编程新手通常会在在线编程练习平台进行编码实践的练习，来提升自身的编程技能水平。在解决编程任务的过程中，编程新手会产生大量编程过程的行为数据和相应的源代码提交。

编程技能的常用度量方法可以分为直接度量法和间接度量法。其中，直接度量法可以分为基于编程产物的研究方法和基于编程行为的研究方法。基于编程产物的研究主要是通过软件测试的方式，根据编程任务的需求生成测试用例，然后对新手完成编程任务时提交的源程序进行评测，从而度量其编程技能；又或者通过运用静态分析技术，将新手提交的源代码表示成抽象语法树的形式，计算其源代码与正确答案代码之间的相似性，从而进行评估。而基于编程行为的研究主要是通过分析编程新手在编程过程中的日志数据等行为信息，从中提取某些行为特征，构建行为模型来度量编程技能并预测其表现[5]。间接法度量主要是通过收集编程新手自身与编程相关的特征数据来进行度量，这些特征数据包括：编程年限[6]、教育经历[7]、自我评估[8]、课程成绩[9]等。这些方法能够对编程新手的编程技能水平进行量化评估，并给予反馈和指导以促进其技能提升。

显然，直接度量法能够更好地对编程新手的技能水平进行度量。通过编程新手在在线编程练习平台中完成给定编程任务的方式，获得其提交的源代码进行评测度量。然而，编程练习不同于一般的练习过程[10]。在编程现场中，程序员往往会历经多次提交以达到正确编码的目的。如果仅仅依据编程新手在在线编程练习

平台中的评测结果来评估其技能水平，就会忽略每次提交的问题解决方案源代码中所蕴含的丰富反馈信息。通过编程新手在编程反馈过程中的反馈信息，构建基于编程反馈的知识追踪模型，更能动态地获取新手的技能掌握状态，从而提升他们的编程技能水平。

知识追踪（Knowledge Tracing，KT）是近年来研究的研究热点，它是根据学生的历史答题数据，分析学生知识状态的变化，并预测其未来时刻的作答表现，一般通过构建学生模型来实现[11]。2015 年，深度神经网络首次运用于知识追踪[12]，而后众多的深度知识追踪模型相继被提出，并被应用于诸多实际教学场景之中。随着编程教育的普及，人们对有效的编程学习方法和评估工具的需求也日益增加。编程知识追踪（Programming Knowledge Tracing，PKT）是知识追踪在学生编程学习这一特定教学情境下的应用，也是评估学生编程技能的重要工具[13]。编程要求程序员设计问题的解决方案，准确无误地将其转化为语法正确的执行指令，并评估这些指令的执行结果[14-15]。编程练习是指学生运用所学知识编写程序进行开放式答题，一般来说他们需要在同一道习题上进行多次尝试，不断修改并提交代码，才能通过该练习。

尽管编程在高校中得到了广泛重视，各专业也都开设了大学计算机基础等与程序设计相关的课程，然而这些课程的实际效果与预期效果仍有较大差距。比如，在解决编程问题时，多数学生在分解问题、制定解决方案以及运用编程语言实施方案等方面都面临困难[16]。

因此，面向编程新手的编程现场表现数据，通过编程测试对新手的编程技能进行评估，并构建出有效的技能追踪模型就显得尤为重要，对这些编程新手的编程技能进行有效的度量具有重要的研究意义。

本文旨在研究如何有效评估编程新手的编程技能水平，以便教育者更好地了解新手的学习状况。这不仅有助于高校检验教学质量，还能加强教育教学管理，进而培养出更专业的计算机人才。

1.2 研究现状

1.2.1 编程技能度量研究

在软件驱动的现代化社会中，编程技能是计算机科学和软件工程学科需要培

养的核心能力。通过文献的调研，本文将度量编程技能的主要方法分为直接法和间接法。本文将从这两个方面分析关于编程技能度量方法的研究进展和发展趋势。

（1）基于直接法的编程技能度量研究

直接法度量编程技能主要可以分为基于编程产物的度量和基于编程行为的度量。其中基于编程产物的度量主要分为动态测试分析和静态代码分析。基于编程行为的研究主要基于程序员编程过程中所产生的日志数据来分析。

1）基于编程产物的度量研究

使用编程测试来评估程序员的编程技能是目前最直接且常用的方法，通过在线评测系统，结合编程任务完成的时间和正确性可以客观的了解程序员的编程水平[17-19]。但是对于测试结果的评估有些研究采用人工评分的方式，耗时耗力，且会受到一定主观因素的影响，造成评估误差。例如，Hammer 等人[20]在研究如何评估学生的计算机编程技能时，采用基于计算机的实验室考试方法，结论证明该方法是一种有效，可靠、高效、以能力为导向的评估方法。Barkmin 等人[21]开发能力框架用于建模编程技能，在开发的框架中使用编程任务和项目对被试者的编程技能进行测试评估。Kittur[22]在研究测量电气和电子工程学生的编程自我效能感的实验中，使用基本编程任务、复杂编程任务对学生的编程技能进行度量。Adelakun[23]为了建立程序员技能的模型，在实验中要求被试人员完成编程测试以及一份问卷，以获取编程技能和相关信息。国内罗文劼等人[24]针对在线评测系统中缺少学习者编程技能的客观反馈和评估指标权重难以确定的问题，对在线评测系统的数据进行挖掘，提取编程技能评估指标，采用双参数平衡熵权法确定指标权重以及编程技能评估值。

早期的基于在线评测系统的评估，主要通过 TRY[25]、CourseMaker[26]、Online Judge[27]等平台评测源程序的功能完整性。随着研究深入，重点逐渐转向改进测试用例的构造生成方法及局部测试程序子功能[28]。

静态代码分析的一个核心步骤是将提交的源代码转换成抽象语法树的形式。通过这一过程，计算其源代码和正确答案代码之间的相似性，从而实现对程序的准确评分。这种分析方法无需实际执行程序，即可有效评估代码的质量和准确性。如在文献[29]中，Wang 等人提出了一种基于程序语义相似性的自动评分方法，针对给定的编程任务，通过计算学生程序和每个模板程序标准化后的语义相似性

来实现自动评分。文献[30]中，Hovemeyer 等人提出了以控制流为基础的抽象语法树（CFAST）来表示源程序，通过计算控制结构的数量来评估源程序与正确程序之间的距离。文献[31]中提出了一种结合词法、语法和语义分析的方法来理解源程序，通过模式匹配和抽象语法树（AST）来判断语义的相似性，并基于不同的权值计算得到最终的评分结果。此外，Srikant 和 Aggarwal 等人[32-33]通过对待测代码的 CFG 和 PDG 进行特征提取，训练单层的神经网络回归模型，用以对人工评分进行预测。

3）基于编程行为的度量研究

现有基于编程行为的编程技能度量研究，主要通过分析学生编程过程中的某些编程行为，提取合理的行为特征指标来构建预测学生表现的模型[34-35]。Estey[36]等人通过收集学生在编程过程中产生的四类数据：编译、运行、代码编辑以及提示使用。按照编译和提示使用的频次及其先后顺序，对学生的编程任务表现进行量化评分。然后根据提出的轨迹指标来衡量编程行为的变化，预测学生是否能顺利通过课程。Watson[37]等人通过分析日志文件中关于同一文件的连续编译行为记录，结合编译结果、错误信息以及编辑时间等特征，实现了对学生编程技能的预测。Spacco[38]等人从基于 Web 的编程练习系统 Cloud-Coder 中收集以两种编程语言教授的入门课程数据，探究学生在编译过程中产生的错误数量、类型、位置以及编辑位置等特征和编程结果的联系，以帮助编程新手提升编程技能。

（2）基于间接法的编程技能度量研究

除了直接度量编程技能的方法外，研究人员还探索了其他几种间接方式来度量程序员的编程技能，分别是编程年限、教育经历、自我评估以及第三方评估。这些方法都是采用间接的方式对程序员的编程技能进行度量。

1）编程年限

在度量程序员的编程技能时，他们从事编程的年限往往是比较重要的参考指标。这既涵盖了他们在公司中积累的编程经验，也包括了他们学习并熟练运用特定编程语言的时间长度。例如，Sillito[6]等人在实验过程中，通过调查参与者专业编程的年数来对其进行等级划分，通过变跟任务的实验评估其编程技能水平。类似地，Robillard[39]等人在研究开发人员如何调查源代码时，也要求被试人员具备一定的编程年限，并认为编程年限可以作为度量程序员编程技能的指标。

2）教育经历

程序员的教育经历也经常被用来表示他们的编程技能。教育经历包括教育水平（如本科生或研究生）或课程信息。例如，Ricca 等人[7]的研究中，本科生被视为编程技能较低的实验对象，而研究生则被视为编程技能较高的实验对象。Ceccato 等人[40]在评估源代码混淆的有效性时，认为博士生相较于硕士生具有更高的编程技能。Raymond[41]等人在研究代码可读性时，进一步将实验的参与者按照教育水平划分为四个等级，以区分不同的编程技能水平。

3）自我评估

研究人员经常让被试人员评估自己的编程技能。例如，Kleinschmager 等人[42]探究了大学成绩、自我评估和预测试对受试者编程经验的影响。该研究要求受试者进行两个编程实验，并将受试者在实验中的表现与自我评估、大学成绩和预测试进行比较。研究发现，自我评估的衡量标准似乎并不比大学成绩和预测试差，甚至可能比这两者更好。类似地，Feigenspan[43]等人从已发表的理解实验当中提取了用于评估编程技能的相关问题，并将其整合成调查问卷，其中也包括要求被试人员对自己的编程技能进行评估。

4）第三方评估

第三方评估的方法可以客观的反映程序员的编程技能。例如，Arisholm[44]等人根据被试者的专业知识将实验的参与者分为初级、中级、高级三个类别进行结对编程的对照实验，通过管理人员从第三方角度评估了被试人员的编程技能。Carver 等人[45]提出直接度量程序员的编程技能比较困难，因此采用同行评估的方式，度量程序员的编程技能。Hannay 等人[46]在研究程序员人格特质对结对编程影响时，比较了专业人员与独立程序员表现，每个被试者的编程技能都由公司经理进行评估。

1.2.2 深度知识追踪研究

虽然国内外学者在编程技能度量方面有诸多研究，但是目前为止依然没有形成统一的指标体系与评价方法。而在知识追踪领域中，对学习者认知状态的跟踪已经有了较为成熟的知识追踪模型。知识追踪模型旨在根据学生在不同时刻的习题作答数据，分析学生认知状态的变化，并预测他们在未来时刻的作答表现。它的形式化定义为：1) 求解学生在不同时刻的认知状态；2) 预测学生在(T+1) 时

刻的作答表现。分别应用于以下 5 个方面的场景：学生作答表现预测、认知状态评估、心理因素分析、习题序列分析和编程练习[47]。其中，针对编程练习的编程知识追踪是从学生的编程练习序列以及提交的代码当中，追踪学生的编程学习情况，被认可的是学生提交的源代码中隐含其对于各个知识点的掌握程度，因此研究者们尝试了不同的方式将学生代码特征融入至知识追踪模型，提升模型对学生在编程学习中后续答题表现的预测精度[48]。

随着深度学习的发展，基于神经网络的深度知识追踪模型得到广泛研究。Piech[12]提出的基于 LSTM（Long Short-Term Memory Network，LSTM）的深度知识追踪模型（Deep Knowledge Tracing，DKT）在预测准确率上显著高于传统的知识追踪模型。其中，LSTM 是循环神经网络（Recurrent Neural Network，RNN）的一种特殊形式。RNN 作为一种递归的动态模型，其当前信息的生成不仅依赖于历史信息，还结合了当前的输入。与传统的隐马尔可夫模型（HMM）相比，RNN 具有高维连续的潜在变量表示。因此，RNN 在处理时间序列问题时成效较好。其详细结构如图 1-1 所示。

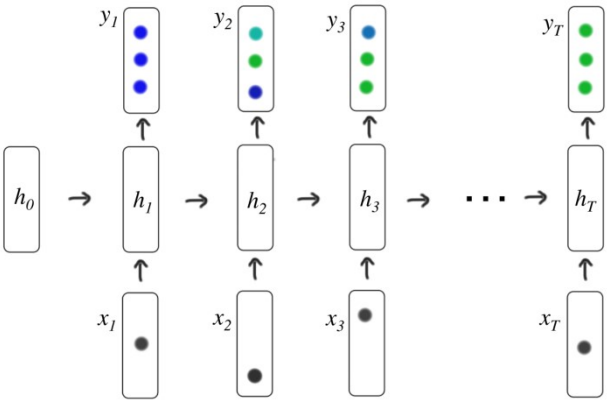

图 1-1 基于循环神经网络的知识追踪模型 (DKT)

Figure 1-1 Knowledge Tracking Model (DKT) based on recurrent neural network

在后续的研究中，许多研究者结合教育数据特征对模型进行了扩展。Zhang 等人[49]通过引入问题级别的更多特征信息改进 DKT 模型，提出了 Adaptive DKT 模型。通过增加自编码网络层将高维输入转换成低维特征向量，减少了模型训练所需的资源和时间。Lalwani 等人[50]通过对 funtoot 平台学生的互动分析，提出将时间间隔编码作为一个特征输入 DKT 模型，从而构建出 DKT-t 模型，实验发现

DKT-t 模型可提高 DKT 模型的预测性能，同时还可根据学生不同的作答序列及认知状态跟踪学生的遗忘曲线。学生数据的稀疏问题一直影响着知识追踪模型的预测精度和模型复杂度。Chen 等人[51]认为解决稀疏问题取决于从知识结构中充分探索概念之间的相互依存关系，特别是教学概念之间的先决条件。该研究提出了 PDKT-C 模型，通过引入知识点之间的先决关系，将其作为模型中的一个关键约束条件，以提高模型的精度。另外，Wang 等人[52]认为习题之间也存在相关关系（side relations）。比如两个习题关联的知识点之间具有相似关系或其他隐含关系，那么这两个习题必然存在相关关系。这些相关关系可以构造成为一个习题子图，该子图可为认知跟踪提供更加丰富的特征信息。因此，Wang 等人提出 DKTS 模型，将习题之间关系的特征信息整合至 DKT 模型中，提升了模型的预测性能。Nakagawa 等人[53]提出了不依赖技能标签的 end-to-end DKT 模型（E2E-DKT），通过学生问答日志中自动学习习题和知识点的向量嵌入，实验证明了该模型学习的向量嵌入对 DKT 的性能具有促进作用。

DKT 模型将 RNN 应用于知识追踪领域，开启了深度学习方法在该领域的应用。目前已有多种基于神经网络的深度知识追踪方法用于诊断学习者潜在的知识状态，例如，基于循环神经网络的知识追踪、基于记忆增强神经网络的知识追踪、基于因子分解机的知识追踪、基于自我注意力机制的知识追踪和基于图神经网络的知识追踪等。

基于神经网络的深度知识追踪的教育应用研究主要涉及以下五个方面。一是诊断学习者知识状态，如 Min[54]等人在基于游戏的智能学习环境中，对学习者在课堂学习中的学习行为、学习过程和学习结果数据实时评估，推断学习者的知识状态，以指导学习者进行个性化学习。二是预测学习者未来表现，如 Wang[55]以及 Swamy[56]等人基于学习者在开放式问题（例如编程练习）中的回答，利用 DKT 模型获取学习者知识概念掌握程度，准确地预测学习者下一次正确提交编程代码的概率。三是发现知识概念之间的联系，如 Zhang[57]利用 DKT 获取学习者对知识概念掌握的先后顺序，生成知识概念的拓扑序列图；Piech[12]利用 DKT 模型计算习题之间的影响因子来发现习题之间的相关性，并将习题进行聚类，进而发现习题中隐含的潜在概念。四是为学习者进行智能推荐，如艾方哲[58]基于动态键值记忆网络知识追踪，并考虑题目难度、关卡特征、做题时间等因素，提出

概念敏感的知识追踪模型，为学习者提供个性化的数学习题推荐，促进了学习者的长期成绩提升。如 Liu 等人[59]提出了元多智能体练习推荐(MMER)，并设计了多智能体练习推荐模块，将练习中涉及的 KCs 视为智能体，它们之间存在竞争和合作。五是模拟学习场景和验证相关理论，如 Lalwani 等人[50]基于 DKT 模型，通过纳入学习者答题时间间隔因素，显著提升模型预测的准确率；同时，通过对学习者历史答题数据的追踪，描绘学习者的遗忘曲线，从而验证了遗忘理论的正确性，即知识掌握程度会随着时间推移而逐渐衰减。Shen 等人[60]提出了基于测验的知识追踪（QKT）模型，以根据学生基于测验的学习互动来监控学生的知识状态。由于各种测验倾向于关注不同的知识概念，该研究分别用门控循环单元测量测验间知识替代和自我注意编码器测验间知识互补性，并采用感知注意机制测量测验间知识互补性。最后整合了测验间的长期知识替代和不同测验之间的互补性，以输出学生不断发展的知识状态。Sun[61]等人提出一种智能辅导模型，帮助编程平台针对不同层次的用户实现更好的辅导。

在编程领域里，学生作答开放习题的过程为研究者提供了丰富的数据资源。Kasurinen 等人[62]提出了一种三相评估（three-phase measuring）方法，旨在观察学生在编程过程中的错误、应用的编程结构，并结合贝叶斯学习模型来确定编程知识。Jiang 等人[63]在基于块的编程中，利用抽象语法树（Abstract Syntax Tree，AST）中的树编辑距离来度量学习者的中间解和最优解之间的相似性，并基于此提出了一种知识追踪模型，该模型能够有效地评估学生在特定编程练习中的认知状态。基于 DKT 模型的思想，文献[55]将嵌入式程序作为 RNN 的一个输入，以此来预测学生在未来编程练习中的表现。此外，Zhu 等人[64]提出一种简单的基于注意力的方法，从学生代码中学习反映每次编程练习所检查的多种编程技能的特征，即搜索学生提交的代码与编程练习检验的学生的编程技能之间的特定的潜在模式，然后将学习到的潜在模式（隐藏状态）与 DKT 结合，进行编程知识的追踪。Li 等人提出通过编程技能跟踪来度量编程练习过程中的技能熟练度。在编程技能的衡量中，知识和能力要更加突出，很少有研究者关注和度量学习者的编程技能。因此，Li[65]等人提出了一个衡量编程练习过程中技能熟练程度的模型，编程技能跟踪模型（PST）。通过代码信息图表示学习者的解决方案的代码特征，代码跟踪图度量相邻提交的代码内容之间的变化，从而度量学习者的编程技能熟

练度，并预测学习者在下一个问题练习过程中的表现。Shi[66]等人提出了基于代码的深度知识跟踪(code -DKT)模型，该模型使用注意机制自动提取和选择特定领域的代码特征来扩展 DKT。并通过收集来自一个 50 名学生班级完成的 5 个入门编程作业的情况，作为与基线模型比较的数据集。Liu[67]等人通过研究预测学生对开放式问题的确切回答，而对开放式知识追踪（Open Knowledge Tracing，OKT）进行了首次探索，并开发了 OKT 问题的初始解决方案，一种以学生知识为导向的代码生成方法，将语言模型的程序合成方法与学生知识追踪方法相结合，从而构建 OKT 模型。Xu[68]等人提出了一种学习行为导向的知识追踪（LBKT）模型，旨在探索学习行为对学习者知识状态的影响，通过分析学习过程中占主导地位的几种学习行为，包括速度、尝试和暗示，定量估计它们对知识获取的影响，然后将遗忘因素与学习者的知识获取相结合，更新学习者的知识状态并预测答题表现。

1.3 主要研究工作

1.3.1 研究内容

本文首先通过前期的文献调研，深入研究了编程技能的度量方法。编程技能度量首先分为直接度量和间接度量，然后分别对间接度量和直接度量的编程新手编程技能度量方法进行研究，为了能更直接地对编程新手的编程技能进行有效度量，本文基于编程现场的新手表现数据，结合知识追踪模型对编程新手的编程技能进行追踪，从而直观的体现编程新手的编程技能状态水平。主要的研究内容包括以下两点：

（1）基于编程测试的编程新手编程技能度量研究

本文提出一种基于编程测试的方式直接度量编程新手的编程技能。主要让编程新手通过在线完成编程任务的方式来进行度量，虽然目前利用编程任务对程序员的编程技能进行度量的研究大有人在，但是他们对于编程测试的结果一般采用人工评分的方式，这样会存在主观因素的影响，造成评估结果的误差。并且，只依靠分数也会造成度量的片面性。因此，本文针对这些不足之处，利用在线实践平台 educoder 对编程新手进行测试，结合系统给出的分数以及他们完成的时间等综合地对编程技能进行度量。

（2）基于编程反馈技能模型的编程新手编程技能度量研究

本文提出一种基于构建编程新手技能追踪模型的编程技能度量方法。通过构建编程反馈技能追踪模型从而获取编程新手的编程知识和编码能力状态，然后基于新手的编程知识和编码能力度量其编程技能水平。

1.3.2 研究思路

首先采用文献研究法，大量阅读编程技能度量和知识追踪领域的相关文献，总结出度量编程新手编程技能的总体路径，以及知识追踪模型的构建方法，并对编程技能度量研究和深度知识追踪研究的相关研究现状进行综述整理，初步确定本文的研究问题和研究内容。

然后设计实验，设计合理的编程任务，并给出有效的测试点作为得分点，在线上学习平台发布给本科生。他们需在规定时间内，完成每个编程任务，并综合他们的时间、评测次数、测试集通过情况进行评分，获得他们编程技能水平的分数值。

收集编程测试实验环节的本科生课程数据，选取他们部分潜在与编程技能度量相关和不相关的课程，并进行课程数据筛选和数据清洗等数据预处理操作，对预处理后的数据进行分析。

开展相关实证研究，分析编程测试实验结果的可靠性与有效性，然后将本科生编程测试的结果与他们的课程成绩进行相关性分析，得到相关实验结果。

构建编程反馈技能追踪模型，通过对编程现场数据的分析与整理，分别对编程新手的编程知识和编程技能进行追踪，从而获取编程新手的编程技能水平，动态度量编程新手的编程技能，得出实验结论。

总结与展望，最后归纳和总结本文的研究结论，对本文研究中的不足之处进行反思和展望。

1.3.3 研究方法

本文主要采用了文献研究法、实验法、统计分析法、深度学习技术进行研究。

（1）文献研究法

文献研究法是指通过收集和查阅文献资料，来进行分析从而获得科学认识的方法。本研究通过对国内外编程技能度量和深度知识追踪的相关文献的分析和研究，总结出度量编程新手编程技能的总体路径方法，以及深度知识追踪模型的构

建方法，系统地分析出编程技能度量的研究问题，为编程测试实验和编程反馈技能追踪模型的构建提供了必要的理论基础。

（2）实验法

实验法是研究者根据一定的研究目的选择一组研究对象，再依据具体的研究任务进行实验，获得定性或定量的实验结果。本文通过设计合理的编程任务，并以有效的测试点作为得分点，在线上学习平台发布给受试者，受试者需要在规定时间内，完成每个编程任务，并综合他们的时间、评测次数、测试集通过情况进行评分，获得他们编程技能水平。并将高年级和低年级的受试者的编程测试结果进行对比，验证初步的假设。

（3）统计分析法

统计分析法是通过调查获取研究对象的各种数据及资料，进行数理统计和分析以形成结论，其中包括对同等地位随机变量相关关系的相关性分析。本研究通过收集本科生的课程成绩数据以及获取他们的编程测试结果进行相关性分析，初步探索研究课程成绩能否度量编程新手的编程技能水平。

（4）深度学习方法

深度学习技术基于深度人工神经网络，旨在实现自动分类、预测和学习等功能。本研究通过结合最终解决方案代码的特征和编程反馈过程中的代码迭代信息构建编程反馈技能追踪模型，从而度量和追踪编程新手的编程技能水平。

1.4 本文组织结构

本文旨在探索和量化编程新手的技能水平，采用了多种方法和技术来实现这一目标。以下是本文的组织结构概述：

第一章：绪论

本章为整个研究设定了基调，介绍了研究的背景与意义、研究现状以及主要的研究工作。本章详细讨论了编程技能度量和深度知识追踪的研究现状，明确了本研究的内容、思路和方法。

第二章：编程新手技能度量概述

本章深入探讨了与编程技能度量相关的概念、理论和方法，不仅详细介绍了编程的基础知识，还探讨了不同的编程技能度量方法，包括直接法和间接法度量，

并通过探索性研究进一步探讨了编程技能间接法度量。

第三章：基于编程测试的编程新手编程技能度量研究

本章聚焦于通过编程测试对编程新手的技能进行直接法度量，详细描述了实验设计、目的、对象和方案，并对实验结果进行了深入分析。

第四章：基于编程反馈的编程新手编程技能度量研究

本章主要介绍了如何通过建立编程练习过程模型和编程反馈技能追踪模型来量化编程技能。本章详细阐述了问题定义、数据集选择、模型方法构建以及实验结果与分析。

第五章：总结与展望

本章总结全文，分析不足，并展望未来研究。

通过以上各章的有机结合，本文旨在为编程新手技能度量提供一个全面、系统的研究，为编程教育和个人技能提升提供有力的理论和实践支持。

第 2 章 编程技能度量概述

2.1 相关概念与理论

2.1.1 编程现场

编程现场是指在实际编程环境中，程序员基于现有的线上编程实践平台进行编码实践的场景。在这个场景中，编程新手利用线上编程实践平台提供的编程工具、编程语言和开发环境，进行实际编程操作，以提高自己的编程技能。这个场景涵盖了编程新手在编程过程中的学习、实践和探索，以及他们所需遵循的编程规范和代码规范。通过这个场景，编程新手可以更好地掌握编程知识和技能，不断提升自己的编程技能，为未来的职业发展打下坚实的基础。

2.1.2 编程新手

一般的程序员是指从事软件开发、编程工作的人员。他们具备一定的计算机科学和编程基础，能够使用各种编程语言编写程序，解决各种计算机软件和硬件方面的问题。程序员需要具备良好的逻辑思维能力、分析问题和解决问题的能力，以及较强的学习和适应能力。程序员的工作内容包括软件开发、系统分析与设计、系统维护等。

程序员等级划分可以有很多种，常见的有初级程序员、中级程序员、高级程序员和架构师。

（1）初级程序员：通常具备一定的编程基础，能够独立完成一些简单的编程任务，对计算机科学和软件开发有一定的了解，但可能对复杂的技术问题处理能力较弱。

（2）中级程序员：具备一定的编程经验和技能，能够独立完成较复杂的编程任务，对软件开发流程和方法有一定的了解，能够解决一些常见的技术问题。

（3）高级程序员：具备丰富的编程经验和深入的技术理解，能够独立完成复杂的编程任务，对软件开发流程和方法有深入的了解，能够解决一些复杂的技术问题。

（4）架构师：是软件开发团队中的技术领导者，负责软件系统的架构设计，能够制定和实现软件系统的整体设计方案，对软件开发过程中的技术问题有深入

的理解和解决能力。

因此，编程新手是指在掌握一门或多门编程语言后，尚未达到能够独立开发软件系统的水平，但在编程领域有一定的基础知识和实践经验的程序员。他们通常具备一定的编程思维和逻辑能力，能够理解和运用基本的编程概念和方法，但尚未具备独立开发软件系统的能力。在实际编程过程中，编程新手可能需要寻求导师或资深程序员的指导和帮助，以解决在编程过程中遇到的问题，提高自身的编程水平。

2.1.3 编程技能

技能是处理相关知识项以执行任务所需的能力，技能词典提供执行这些任务所需的技能。技能词典由四层组成，分为三个技能层和相关知识项，如图 2-1 所示。技能词典引用和分类来自世界上主要知识体系/过程和技能标准的项目。

编程技能是程序员利用编程知识和算法能力，准确且高效地执行任务的能力。它体现了程序员对编程语言、算法和数据结构的熟悉程度，以及对软件开发过程和方法的理解与运用能力。此外，编程技能还涵盖了软件工程、软件设计、系统架构等多方面的知识和能力。程序员通过不断提升编程技能，不仅能更出色地解决问题，还能提高软件质量和稳定性，从而更好地适应日新月异的技术环境。

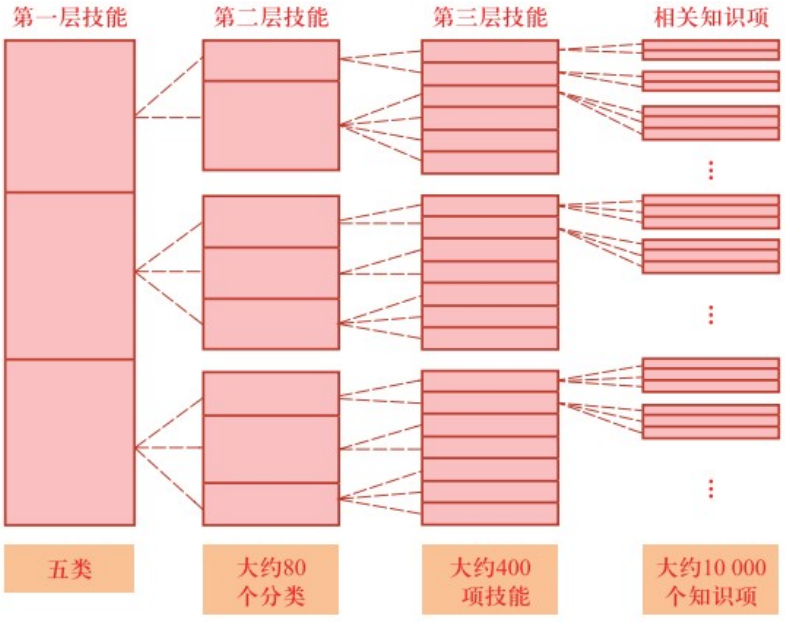

图 2-1 技能词典结构图

Figure 2-1 Skill dictionary structure

2.1.4 测试驱动开发

测试驱动开发（Test-Driven Development，TDD）是一种软件开发方法，它强调在开发过程中进行单元测试，以确保代码的质量和可维护性。测试驱动开发的基本思想是，在编写代码之前，先编写测试用例，然后根据测试用例编写代码，以确保代码满足预期的需求[69]。在测试驱动开发中，测试用例通常是独立、单元化的，它们只测试某个特定的功能或行为。在编写测试用例时，程序员应该尽可能地覆盖所有可能的情况，以确保代码的健壮性和可靠性。在实际开发中，测试驱动开发方法通常与其他软件开发方法，如敏捷开发、持续集成和持续交付等相结合，以实现高效的软件开发流程。通过测试驱动开发，程序员可以更好地理解代码的需求和功能，提高代码质量和可维护性，并能够更快地发现和修复代码中的问题。

因此，在基于程序员测试驱动开发的思想下，本文根据相应的编程任务生成合理的测试集作为得分点，评估编程新手的编程技能水平。

2.1.5 认知诊断与知识追踪

认知诊断是一种评估学习者在特定知识或技能领域的掌握情况的方法，旨在识别学习者在特定认知能力上的强项和弱点。这种诊断方法依赖于认知诊断模型（Cognitive Diagnosis Models，CDM），这些模型通过分析学生的作答模式来推断他们对各个认知属性的掌握情况。认知诊断的目标是提供更加细致和具体的反馈，帮助教师或学习者了解哪些具体的知识或技能需要改进，从而实现个性化的教学和学习。

知识追踪是一种动态模拟学习者知识状态变化的技术，用于预测学习者在学习过程中对特定知识点的掌握情况如何随时间变化。知识追踪通常依赖于贝叶斯网络或类似的概率模型，通过分析学习者的学习活动和作答记录来更新对其知识掌握水平的估计。知识追踪的目的是跟踪和预测学生的学习进展，从而支持个性化学习路径的设计和实时反馈的提供。

认知诊断和知识追踪虽然都旨在评估和改进学习者的学习成果，但它们侧重点不同。认知诊断更侧重于评估学习者在某一时间点的知识或技能掌握情况，侧重于诊断评估和提供针对性反馈；而知识追踪则侧重于随时间跟踪学习者的知识状态，更注重预测和个性化学习路径规划。

2.2 编程技能度量方法

编程技能度量是一种评估程序员编程技能的方法，它通过多种方式来衡量程序员的编程水平、编程风格和编程效率。这些方式包括编写代码的速度、准确性、参加项目的数量、编写代码的总行数、以及在规定时间内完成程序设计题目的表现等。目前，编程技能的度量方法主要可以划分为直接度量法和间接度量法。

2.2.1 直接法度量

直接法度量编程技能主要是通过线上或者线下的编程任务直接地对受试者进行考察，通过考察结果评估受试者的编程技能水平。

在线上测验的编译环境下编写程序，分为基于编程产物的评估和基于编程行为的评估。其中，基于编程产物的评估就是对受试者完成编程任务的解决方案进行分析，其分析方法大致可以分为两类：动态分析法，根据编程任务设计合理的测试集，最后根据运行的程序在该测试集上通过的测试用例的数量进行评估；静态分析法，根据提交的源代码，将其转换成抽象语法树的形式，然后计算提交的源代码与正确答案之间的相似性，从而对其进行评估。

2.2.2 间接法度量

间接法度量编程技能主要是通过收集编程新手自身与编程相关的特征数据来进行评估和度量，这些特征数据包括：编程年限、教育经历、自我评估、课程成绩等。获取编程年限、教育经历以及自我评估结果的途径主要是通过调查问卷的方式，而课程成绩则是在相关课程的成绩数据中收集，具体实施如下：

（1）调查问卷：通过发放包含编程技能相关问题的调查问卷的形式来度量和预测程序员的编程技能水平。这种方法通常会设计一系列与编程技能相关的问题，并通过分析回答这些问题的方式，来评估程序员的编程技能。

（2）课程成绩：通过采用课程的最终成绩来度量和预测程序员的编程技能水平。这种方法通常需要考虑课程成绩的合理性，通过分析学生在相关课程中的表现，来评估他们的编程技能水平。

2.3 编程技能间接法度量的探索性研究

2.3.1 问题提出

随着互联网企业对提升软件质量的的需求越来越大，它们对程序员的编程技能要求也越来越高。而复杂度和可变性是软件研发中的根本困难，概念结构在说明、设计和测试上的复杂度，在短期内没法通过更好的编程语言和更好的工具来消除。虽然软件行业在这 40 年里蓬勃发展，但从业者依然在消除复杂度上缺乏有效的方法，这也导致了软件行业在工业化的进程中依然极度依赖于程序员的编程技能。

然而，对于在学校学习，并即将迈入社会的编程新手来说，他们提升编程技能的主要的方式就是通过课程的学习，从而得到锻炼。因此，这些课程是否能够度量编程新手的编程技能就显得十分重要。

在 2013 年左右，我国引入了用成果导向教育理念引导工程教育改革，使得 OBE 理念（Outcomes-based Education，OBE）在中国具有现实意义。在现今的高等教育教学中，人们更加关注教育投入的回报与实际产出的现实需要，成果导向教育在我国成为了教育改革的主流理念。在 OBE 理念中更加注重学生对应能力的培养，以学生为中心，进行个性化评定，按照课程目标中的不同能力的要求对学生进行最终成果的评定，让学生真正的理解并解决问题。

因此，在 OBE 理念下，课程成绩就是学生在学习课程相关的知识和获得对应的能力后，对学生进行考核得到的总成绩，代表的是学生某一科学习情况的综合。跟计算机专业相关的课程即考核的是计算机相关的知识与能力，这里就可能涉及到学生的编程技能。

那么，学生的哪些课程与其编程技能相关，并能够度量编程技能的呢？关于度量编程技能的探索，很多的学者都已经发现了诸多影响因素，但对于哪些是主要影响因素，以及各因素具体的影响值并不清楚。本节主要研究哪些课程可以度量编程新手的编程技能，分析不同的课程成绩与编程技能的相关性，找出显著相关的课程。

2.3.2 数据收集

（1）课程目标达成评价体系

根据面向产出理念要求，注重对学生能力的培养，以及对课程目标达成情况进行合理的评价。因此，在本专业的学生培养过程中需要制定一个完备的课程目标评价体系。本专业课程目标达成情况评价以考核成绩评价法为主，并参考课程结束后的学生调查问卷和座谈会，形成最终课程目标达成情况评价。具体的评价过程如下图所示。

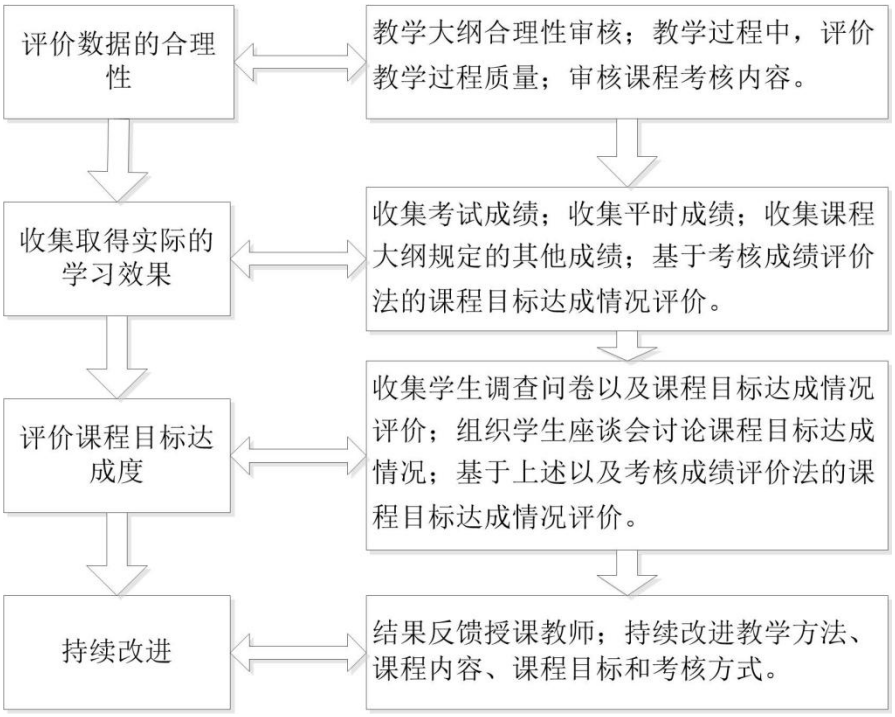

图 2-2 课程目标达成情况评价流程

Figure 2-2 Evaluation flow of the achievement of curriculum objectives

课程总体目标达成度的计算方法如下：

因为：

C：课程的总体目标达成度计算值；

W_i：课程目标 i 的达成度权重系数；

OB_i：课程目标 i 的达成度计算值，OB_i = 目标 i 的得分B_i/目标 i 的权重W_i。

B_i：课程目标 i 的得分，B_i由目标 i 的各考核分项的达成值T_j加权求和得到，设各考核分项的权重为w_j，则：

$$B_i = \sum T_j * w_j \tag{2-1}$$

T_j：考核分项达成值，T_j = 考核分项平均成绩b_j/分项满分M_j。

w_j：考核分项的权重，w_j由教学大纲规定的课程考核方式所占权重分解得到，具体见各课程大纲。

b_j：考核分项平均成绩，b_j成绩种类，一种是作业、测验、考试等有具体得分的考核方式，另外一种是实验、设计报告、实习报告、验收等根据观察点评价得到的成绩（如果是等级制，需要转换成百分制进行计算）。

所以根据以上关系，得到 C 的简化计算公式如下：

$$C = \sum OB_i * W_i = \sum \frac{B_i}{W_i} * W_i = \sum B_i \quad (2\text{-}2)$$

（2）数据的合理性保证

课程目标的达成评价的数据来源包括：教学大纲、试卷、试卷评分标准、试卷分析表、教学过程质量评价表等。

首先，在每门课程结束以后，任课老师要对课程目标的评价所采用的数据内容、来源、收集方法，以及这些数据与学生能力的体现达成的相关性进行分析和说明。然后就对应这四个合理性的保证：

a.保证课程大纲的合理性。其中，需要确保课程目标的合理性，课程目标-教学内容的相关性、考核方式的合理性等。

b.保证教学内容与课程大纲相符合。在教学过程中需要保证教师是否按照预期的进度、教学内容进行授课来完成课程的预定目标。

c.保证考核内容实现评价课程目标的合理性。教师要对考核内容如何对接课程目标进行分析说明，保证合理性。

d.保证基于调查问卷进行课程目标评价的合理性。主要是对调查问卷效度分析，解释说明调查问卷的内容，对异常数据进行处理。

2.3.3 数据预处理

课程由于学生的课程种类繁多，学生课程成绩的数据集较为庞大，里面可能出现数据的缺失、数据噪声、数据不一致、数据冗余以及数据重复等情况。因此，需要对课程成绩的数据进行一定程度的预处理，使整个数据的分析结果更加真实可靠。

（1）数据筛选

首先在学校系统中提取出整个专业的所有学生的所有课程成绩的数据集，这

里包括整个专业的四个年级的学生成绩，然后从中分别筛选出本文需要的大三、大四学生的所有课程成绩。

在筛选出大三学生的课程成绩中，再进行筛选，然后根据《计算机课程体系规范报告》选取出学生集体参与人数较多且符合一定代表性的课程，以便于统计与比较，其中包括 Java 程序设计、软件工程、操作系统、大数据存储与处理、计算方法、计算机组成与结构、数据库原理及应用 I、数字图像处理 II、数据结构、高级语言程序设计（C 语言）、科学计算语言（python 语言）、大学英语（1）、马克思主义基本原理概论一共 13 门课程。

在筛选出大四学生的课程成绩中，进行同样的筛选，以数据结构和高级语言程序设计（C 语言）作为参考线，再筛选出软件工程、科学计算语言（python 语言）、数据库原理及应用、操作系统、计算机组成与结构一共 7 门课程。

（2）数据清洗

1）缺失值处理

在现实世界中，获取信息和数据的过程中常常会导致数据丢失和空缺。为了有效应对这些缺失值，本文主要依据变量的分布特性和其重要性来制定处理策略。在本实验中，部分学生（主要是转专业的学生）的某些课程成绩存在缺失，经过评估，缺失的变量对整个实验的重要性较高，所以采用统计量填充的方式，初步使用平均值填充。然而，其中也有一部分留级学生的分数缺失，由于此类数据缺失率较高，参考性不足，所以将此类的变量都删除。

2）数据不一致处理

在某一课程的分数表里，存在同一个学生的分数前后不一致存在矛盾的情况。其中显示学生完成这门课程的学期不同，这是由于这个学生在第一次学习这门课程后，未能通过该课程，于是需要重修课程。针对这种数据，本文考虑到学生是因为重修导致课程中存在多个分数值，而重修课程相较于正式的课程会缺少严谨，所以取学生的第一次课程成绩作为其该课程的最终成绩，使得结果更加真实。

3）数据冗余处理

数据冗余是数据量或者属性数目超出数据分析需要的情况。在导出的学生课程成绩的数据集中，存在其他属性如学年度、课程序号、课程代码、授课教师等，这些都属于多余的属性，超出了数据分析的需要，因此需要删除。

（3）数据描述

对学生课程成绩预处理完以后，可以对这些数据进行可视化的一个操作，来呈现出他们的分布情况。

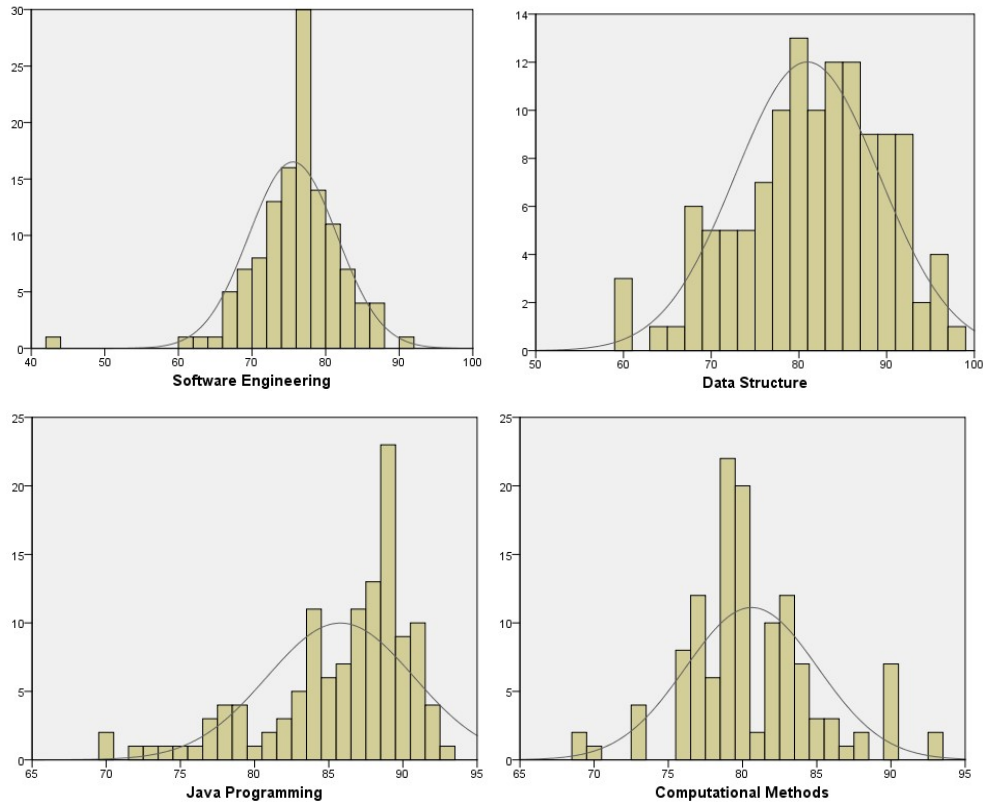

图 2-3 课程成绩分布图

Figure 2-3 Distribution of course scores

从上面大三学生的课程成绩分布图中可以看到这些课程的分数基本都符合正态分布，分布比较合理。同样在对大四学生的课程成绩进行可视化操作后显示的分布结果也是较为合理。

2.3.4 实验验证

构建和验证编程测试题是一个复杂的工作，需要设计合理的编程题发布任务，并与课程成绩进行分析验证。将测试题的综合分数作为学生的一个编程技能水平基线，分析课程成绩与编程技能的相关性。

在课程数据收集之前，本研究已进行了相关编程测试题目的实验设计与数据收集工作，通过在线实践平台发布编程任务，主要是由计算机科学与技术专业大三的本科生参与完成。通过在线实践平台，限制学生答题时间以及答题行为规范等对学生的编程过程进行严格的管理，以此收集更为可靠的答题数据。再对答题

数据进行预处理，获得编程测试任务的整体数据。

然后本研究将学生完成编程测试任务的结果与课程成绩数据进行相关性分析，获得初步的研究结果。

2.3.5 实验结果与分析

在实验结果中可以得出，在将所有学生的编程技能测试分数与他们的课程成绩进行相关性分析后得出，软件工程的课程成绩与编程测试分数最相关，并且该分数与大学英语的课程成绩有较高相关性，与马克思主义基本原理概论很不相关。

本文对 124 名学生的测试数据进行了整理，然后将所有学生的编程技能测试分数与他们的课程成绩进行一个相关性的分析。由于并不确定这些数据是否呈线性关系，本研究采用的是斯皮尔曼秩相关。获得初步的结果，这些课程里测试分数和软件工程课程成绩相关性最强，然后与科学计算语言，数据库原理及应用 I，数据结构，操作系统，Java 程序设计，计算机组成与结构也很相关，其中与马克思主义基本原理概论很不相关。在 13 个课程里，有 7 个课程是最显著相关的。具体情况如表 2-1 测试分数与课程成绩的斯皮尔曼秩相关。

表 2-1 测试分数与课程成绩的斯皮尔曼秩相关

Table 2-1 Test scores correlate with Spearman rank of course scores

序号	课程名称	ρ	N
1	软件工程	.454**	124
2	科学计算语言（python）	.334**	124
3	数据库原理及应用 I	.297**	124
4	数据结构	.273**	124
5	操作系统	.265**	124
6	计算机组成与结构	.254**	124
7	Java 程序设计	.232**	124
8	大数据存储与处理	.222	124
9	数字图像处理 II	.197	124
10	高级语言程序设计（C）	.183	124
11	大学英语 (1)	.179	124
12	马克思主义基本原理概论	.117	124
13	计算方法	.078	124

2.4 本章小结

本章围绕编程技能度量展开，涵盖了相关概念与理论、方法以及探索性研究。首先，对编程现场、编程新手、编程技能、测试驱动开发和认知诊断与知识追踪等概念进行了阐释。其次，介绍了直接法和间接法两种度量编程技能的方法。

在探索性研究部分，提出问题并进行数据收集、预处理和实验验证，最终对结果进行分析。研究表明，软件工程课程与编程技能的相关性最强，还有另外 6 门课程构成的课程群也有较强相关性。此外，英语能力对编程新手具有一定重要性，因为编程中常涉及英语术语。比较大三和大四学生的测试结果，发现编程技能更好的学生做题更快、更准确。另外，相较于其他的课程，实行 OBE 理念教学的课程成绩更能体现出编程新手的编程技能水平。

因此，提升编程新手的编程技能至关重要。除了实践练习，还需建立完善的教学体系，利用课程成绩评估学生的编程技能，从而有效提升其编程水平。

第 3 章 基于编程测试的编程新手编程技能度量研究

3.1 基于编程测试的直接法度量

基于编程测试的直接法度量主要是根据给出的编程任务，生成合理的测试集，以受试者提交的源程序通过的测试用例的数量为依据，结合受试者完成编程任务的时间，评测次数以及代码的规范情况评估受试者的编程技能水平。首先，本文提出了三点假设：

（a）受试者的编程技能水平越高，他们解决任务的正确率就越高。因为编程技能水平越高，编程经验也更加丰富，有经验的学生比没有经验的学生参加的编程项目更多，因此也能正确地完成更多的编程任务。

（b）受试者的编程技能水平越高，完成编程任务的时间更短，速度更快。（在不考虑受试者非正规答题的情况下）

（c）受试者的编程技能水平越高，完成编程任务时评测通过的次数也更少。（在不考虑受试者非正规答题的情况下）

其中，针对测试用例的设计，本研究结合了题目中涉及到的知识点情况，依据不同的知识点设计对应的测试用例作为测试集，以学生通过测试集的数量作为评估的标准之一。

基于编程测试的直接法度量编程新手的编程技能水平，主要是将受试者的源程序的正确性评分结合完成编程任务的通关时间以及代码质量评分作为编程技能水平的度量指标，具体的编程技能度量准则如下：

（1）程序正确性评分

受试者在第 i 项编程任务中通过的测试用例数量 l 所获得的实际分数 W_i 即为该项编程任务的程序正确性评分，同时对于未按时通关而开通补交的受试者，其程序正确性得分将由测试得分和补交扣分组成，本研究按照补交通关扣除总测试分数的 25%作为补交受试者的实际测试分数。

因此，关于受试者的第 i 项编程任务的正确性评分 Q_i 计算规则如下：

$$Q_i = \sum_{j=1}^{l} x_j \times w_j \tag{3-1}$$

其中，Q_i是受试者在第i项编程任务中通过的测试用例所得分数，l是受试者在第i项编程任务中通过的测试用例数量，并且$l \le m$（m是第i项编程任务的所有测试用例数量），x_j是受试者在第i项编程任务中通过的第j项测试用例，w_j是对应第x_j项测试用例的分数占比。

针对受试者是否按时通关提交源程序，其第i项编程任务的程序正确性评分W_i计算规则如下：

$$W_i = \begin{cases} Q_{i_a}, Q_{i_a} = Q_{i_b} \\ max\{Q_{i_a}, Q_{i_b} * (1 - 25\%)\}, Q_{i_a} < Q_{i_b} \end{cases} \quad (3\text{-}2)$$

其中，Q_{i_a}是补交前第i题的关卡得分，Q_{i_b}是补交后第i题的关卡得分。若Q_{i_a}与Q_{i_b}相等，则表示受试者未参与补交流程，或通过的测试用例个数并未改变，因此按照补交前得分进行计算；若$Q_{i_a} < Q_{i_b}$，则表示受试者参与补交流程后，对程序进行了一定的程度的修改，从而使得通过的测试用例个数增加，程序相比于补交前的完成情况更好。在这种情况下，将对补交后得分扣除一定分数，然后再与Q_{i_a}进行比较，取其最大值。

（2）效率分

本研究通过受试者的通关时间和评测次数综合得出其第i项编程任务的效率分得分E_i。具体计算规则如下：

$$e_i = \log^{W_i / T_i} \quad (3\text{-}3)$$

$$E_i = \frac{e_i}{(e_i)_{\max}} \times s \quad (3\text{-}4)$$

其中，e_i表示第i项编程任务学生的学习效率，T_i表示该项编程任务的总耗时，$(e_i)_{\max}$表示课堂学生的最高学习效率，s表示对应分值。另外，当学生未通关时，即学生未通过该项编程任务所有的测试用例，学生效率分为0。

（3）代码质量评分

代码质量评分ΔC_i是指平台基于受试者提交的源代码，通过对其中的缺陷、

漏洞以及代码规范性进行全面评估，进而为其源代码划分相应的质量等级。并且，只有当受试者通过全部测试用例，才能对其源代码进行质量评估；否则，说明其代码存在严重缺陷，代码质量检测意义不大。具体的等级划分规则如下表。

表 3-1 代码质量等级划分表

Table 3-1 Code quality levels

等级	缺陷评估规则	漏洞评估规则	代码规范评估规则
A	0 个错误	0 个漏洞	0 - 5%
B	至少 1 个次要错误	至少 1 个次要漏洞	6% - 10%
C	至少 1 个主要错误	至少 1 个主要漏洞	11% - 20%
D	至少 1 个严重错误	至少 1 个严重漏洞	21% - 50%
E	至少 1 个阻断错误	至少 1 个阻断漏洞	超过 50%

综上，基于编程测试的直接度量法度量编程新手编程技能水平 S 的计算公式如下：

$$S=\sum_{i=1}^{n}\left(W_i+E_i+\Delta C_i\right) \tag{3-5}$$

3.2 测试用例的生成

测试用例是软件测试过程中的核心元素，是为了验证软件的某一特定功能或特性是否按照预期工作而设计的一系列条件或变量。每个测试用例包括输入数据、执行条件和预期结果，这些都是为了模拟特定的使用场景或者检查软件行为的正确性。测试用例的主要目的是确保软件程序的质量，通过预先定义的测试用例来发现程序中的错误、缺陷或不符合要求的行为[70]。

本章节主要是基于测试用例的生成来实现对编程新手提交的源程序的动态评估。结合软件测试的相关方法，包括黑盒测试等针对编程任务对源程序的需求功能自动化生成合理的测试用例，从而对编程新手完成编程任务的情况进行评估，度量编程新手的编程技能。

3.2.1 软件测试方法

软件测试是确保软件质量和性能符合预期要求的关键过程。它涉及到多种测试方法，每种方法都有其特定的目标和适用场景。在众多测试方法中，黑盒测试和动态测试是两种常用的测试方法，黑盒测试侧重于从用户的角度测试程序的功能，而动态测试则在程序运行时通过实际执行来发现问题[71-72]。

（1）黑盒测试

黑盒测试，又称功能测试或外部测试，主要关注于软件的功能性是否符合用户需求，而不考虑程序内部的逻辑结构。其目的就是验证软件或程序的功能是否按照需求来执行[73]。

黑盒测试一般可分为功能测试和非功能测试两大类[74]。功能测试方法主要包括等价类划分、边界值分析、因果图法、错误推测等，其主要是验证系统的各项功能是否正常运行，包括输入输出是否正确，系统是否符合预期等。非功能测试方法主要包括性能、兼容性、安全性、配置等测试。

（2）动态测试

动态测试方法主要依据源程序在运行过程中展现出的特征，执行包括跟踪、时间分析和测试覆盖在内的多种测试任务。该方法的核心在于实际运行被测程序，通过输入有效的测试用例，深入分析程序的输入与输出对应关系，从而达成检测目标。

动态测试方法的基本步骤如下：

1）选取定义域内的有效值，或选取定义域外的无效值；

2）对已选取值决定预期的结果；

3）用选取值执行程序；

4）执行结果与预期的结果比较，不符合则说明程序有错。

动态测试需要在实际的或接近实际的环境中进行，覆盖软件的功能、性能、安全性等多个方面，通过实际运行软件来进行测试。

3.2.2 基于程序功能的黑盒测试方法

本章的研究主要是结合黑盒测试和动态测试的方法，生成测试用例，即基于程序功能的黑盒测试方法，对编程新手提交的源程序进行评估，结合测试结果、

完成编程任务的时间、通关评测次数以及代码质量评估的结果综合地对其编程技能水平进行度量。黑盒测试中的功能测试方法主要关注程序的功能实现情况，通过输入数据的合法性和异常情况，程序的执行流程和执行结果等，来判断验证程序的正确性和稳定性。

具体的功能测试方法包括：等价类划分法、边界值分析法、因果图法等。

（1）等价类划分法

等价类划分就是将所有可能的输入数据，即程序的输入域划分成若干个等价类，每个等价类内的数据具有相同的数据特性，然后从每个等价类中选取一个或多个数据作为测试用例进行测试，验证程序的正确性和稳定性。

等价类的划分包括有效等价类和无效等价类，其中有效等价类是指针对程序的功能需求，合理且有意义的输入数据构成的集合；无效等价类是指针对程序的功能需求，不合理的、无意义的输入数据构成的集合。其具体步骤流程图如下图所示。

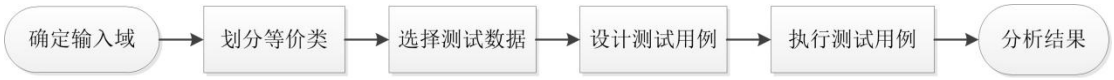

图 3-1 等价类划分步骤流程图

Figure 3-1 Flowchart for dividing equivalence classes

（2）边界值分析法

边界值分析法主要是通过测试程序在输入域的边界值（最大值、最小值、唯一值等）的行为，来验证程序的正确性和稳定性。一般来说，边界值分析和等价类划分是密不可分的，二者同时运用。其具体步骤流程图如下图所示。

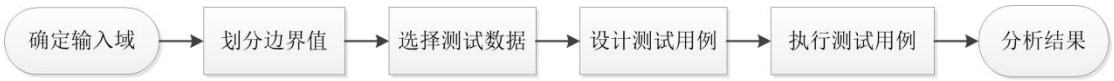

图 3-2 边界值分析步骤流程图

Figure 3-2 Flowchart of boundary value analysis

（3）因果图法

因果图法是利用图解分析输入组合，设计测试用例的方法，适用于检查程序输入条件组合，利于全面覆盖测试各种输入场景。其基本思想是通过构建因果图，来分析程序中的各变量之间的因果关系，从而找出潜在的缺陷和错误。

在因果图中，Ci 表示原因，ei 表示结果，有 4 种符号分别表示了规格说明中的 4 种因果关系。Ci 与 ei 可以取值 0 或 1，0 表示状态不出现，1 表示状态出

现。其基本符号与约束条件如图 3-3 所示。

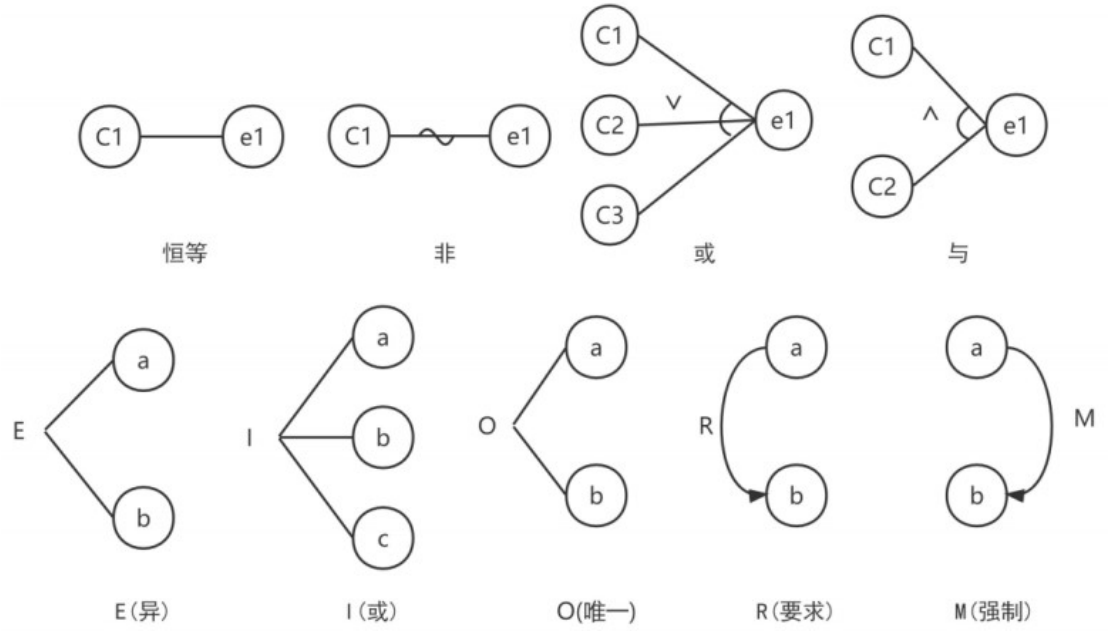

图 3-3 因果图的基本符号与约束条件

Figure 3-3 Basic symbols and constraints of causal diagrams

输入状态之间还可能存在某些依赖关系，或输出结果之间相互制约，这被称为约束。其所描述的这种制约关系一般可被分为 5 类:互斥、包含、唯一、要求和屏蔽。

3.2.3 测试用例的生成算法

本章研究中测试用例的设计生成部分主要是将黑盒测试与动态测试的思想结合，通过等价类划分和边界值分析生成对应编程任务的测试集，然后通过在线编程平台运行源程序，获得每个学生的每个编程任务的测试用例通过的数量，从而计算出其程序的正确性评分。

例如，在编程任务 4 中对应的是分数序列求和：有一分数序列：2/1，3/2，5/3，8/5，13/8，21/13...求出这个数列的前 n 项之和的整数部分。n 的值表示序列项，运行时通过键盘输入给定，并且 n<=30，输出值只保留运行结果的整数（不要四舍五入）。

关于该编程任务主要考虑等价类划分和边界值分析方法，根据任务需求确定输入域 n<=30，然后可以将该输入域划分为有效等价类与无效等价类的输入值范围，具体如下图所示。

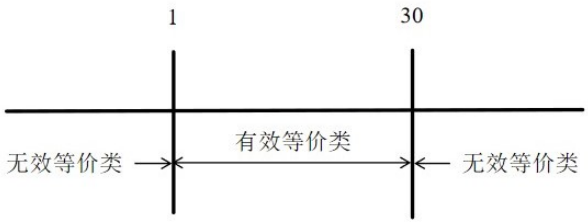

图 3-4 输入值范围

Figure 3-4 Input value range

然后根据输入值的范围划分，可以生成对应的测试用例，如下图所示。

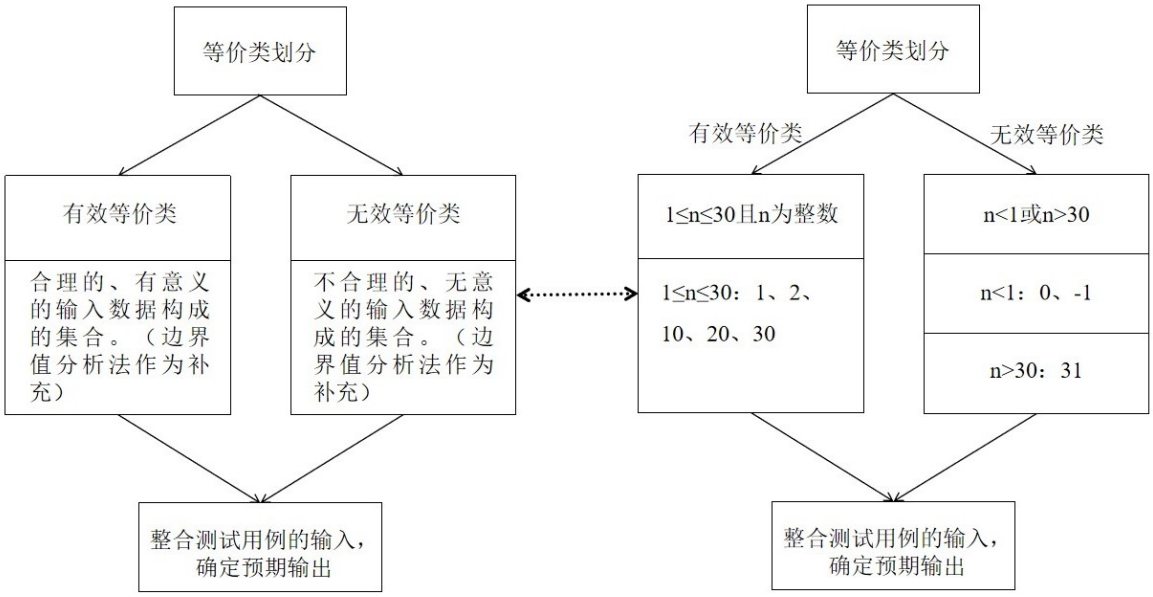

图 3-5 分数序列求和测试用例图

Figure 3-5 fractional sequence summation test case diagram

3.3 实验设计

3.3.1 实验目的

基于编程测试的直接法度量编程新手的编程技能水平，以计算机与信息学院的本科生为实验对象，通过在线实践平台设计具体的实验方案，规定具体的时间范围和实验环境，获得实验结果，评估基于编程测试的编程新手编程技能度量方式的有效性和合理性。

3.3.2 实验对象

本实验考虑编程新手特征的特殊性，他们通常是指在掌握一门或多门编程语言后，尚未达到能够独立开发软件系统的水平，但在编程领域有一定的基础知识和实践经验的程序员。因此，本研究选取了 124 名计算机与科学技术专业的本科生完成在线实践平台的编程任务。他们通过学习不同的计算机专业课程，已经具备了一定的编程思维和逻辑能力，能够理解和运用基本的编程概念和方法，能够作为本次编程测试的实验对象。

3.3.3 实验环境

头歌在线实践平台（Educoder，https://www.educoder.net/）是一个为学生和开发者提供编程实践环境的平台。它提供了丰富的编程项目、算法和数据集，用户可以在平台上进行编程练习和项目开发。它提供了多种编程语言和工具，包括 Python、Java、C++、JavaScript 等。用户可以根据自己的需求选择不同的编程语言进行实践。头歌实践平台还提供了丰富的数据集和算法，用户可以在平台上使用这些数据集和算法进行项目开发和算法训练。总的来说，该平台是一个为学生和开发者提供编程实践环境的平台，它提供了丰富的编程项目、算法和数据集，用户可以在平台上进行编程练习和项目开发。其充分支持 OBE（成果导向教育）理念和机制，支持作业、实验、考试、课堂等多粒度、多维度的教学成效分析，具体功能如下图。

图 3-6 头歌课堂教学工具图

Figure 3-6 Educoder classroom teaching tool diagramm

在该平台，教师可以创建教学课堂，发布试卷、问卷、课堂实训作业、学习资料以及课堂实验等，协助老师完成课堂教学

内容以及管理课堂，并帮助学生学习。其次，头歌平台针对教学课堂发布的实训作业，制定了一套评分机制，根据学生完成实训作业的通关时间、提交评测的次数、以及完成的效率评估学生答题的分数值。另外，头歌平台还开通了代码质量评估功能，可以根据学生提交的源代码，评估提交的代码质量，主要考量代码缺陷、代码漏洞以及代码的规范性等。

因此，借助头歌在线实践平台可以综合地对学生完成编程测试任务的情况进行评估，获得更为真实有效的测试分数，确保实验结果的可靠性。

3.3.4 实验方案

关于本研究后续的实验方案设计主要分为两部分，分别是编程任务的设计和实验过程的设计。

（1）编程任务的设计

针对本科生编程新手的学习特点，本研究主要借鉴了头歌平台发布的若干实践项目类题目。这些题目涵盖了编程基础、数据结构以及常用算法（包括枚举、贪心、递归、动态规划、深度搜索、广度搜索等）的考核。在其中选取算法类编程题目，结合一些基础的编程语言概念题作为编程测试的主要题目。本研究设计了 5 项程序设计任务，分别是 2 项 Java 语言编程任务和 3 项 C 语言编程任务。

通过头歌平台的线上编程实践平台，学生完成了一系列已发布的编程任务，随后进行了编程任务的完成情况统计。在图 3-7 中展示了第一项编程任务的题目原文及代码示例，鉴于这是序列中的首个任务，其难度相对较低，目的是要求学生使用 Java 语言编写程序，以统计每位学生的总分。

其余的几个编程任务分别是输出学生还是老师，分数序列求和，插入排序和打印日历，其中打印日历具有一定的复杂度，能更好的区分学生的编程技能。这里还对学生的编码规范有一定的考察，测试学生的代码注释以及函数命名等是否符合规范。本文不仅统计头歌平台中学生的具体成绩，还具体查看未通关的学生的实验详情，并进行打分，针对这类学生给一个具体的分值。

为了匹配本科生的平均编程水平，本研究选择了一些基础的程序设计算法和问题。但还是设置了一道有一定难度的题目（后来对学生进行题目难度反馈的小

调查中也能获得这样的信息），打印出年某月的日历。因为一些编程经验丰富的学生在实际的编程项目中也更加清楚自己解决问题的思路和目的，所以期望只有编程经验丰富的学生能够较好地完成这个任务。

```
/*
* 任务：统计每人的平均分。
* 输出样式：x 号学生的总分：y
*
* */

public class PassWord {
    public static void main(String[] args) {
        // 创建二维数组存储所有人的成绩
        int[][]arr = new
          int[][]{{90,88,87},{89,90,77},{66,78,60},{77,90,90},{89,78,67},{78,87,88}};

        // 请在 Begin-End 间编写代码
        /********** Begin **********/
        // 第一步：对每个人的各科成绩求和
        int x,y;
       for( x=0;x<arr.length;x++){
            int sum=0;
            for(y=0;y<arr[x].length;y++){

                sum+=arr[x][y];
          }
        // 第二步：输出每个人的总分
            System.out.println(x+1+"号学生的总分："+sum);
        }

        /********** End **********/
    }
}
```

图 3-7 任务 1 题目及代码图

Figure 3-7 Task 1 problem and code diagram

（2）实验过程的设计

该实验于 2022 年 4 月在这 124 名学生参加的课程的实验课上完成，在头歌平台上发布任务的同时，邀请了一名教师和三名助教，共同确保实验过程中的秩序。起初，学生们并未意识到实验的真正目的，他们将其视为实验课的一部分，在之后的做题过程中向他们简要解释了题目的背景和实验过程中的相关规定。

在实验过程中，学生们必须独立完成编程任务，不得查阅课本和资料，也不

得与他人讨论或共享答案。此外，在每项编程任务开启一段时间后关闭提交入口，以确保实验数据的可靠性。在实验结束后，还对部分学生进行了回访，了解他们对这些题目的看法，以及实验过程中是否按照计划进行，题目的难度如何等。这些举措旨在提高实验过程的规范性，增强实验结果的可信度。

另外，在对比实验中，本文还针对 5 名大四学生进行了类似的测试，他们独立完成了其中的 3 个编程任务。由于这些学生已知实验目的，实验过程更加顺利，完成情况也更为真实。

3.4 实验结果与分析

3.4.1 实验结果

在实验初步结果中可以得出，随着编程任务的难度以及答题时间的不同，学生的答题分数也不同，相对来说，简单且靠前的任务正确人数更多。最后一个较难的编程任务，正确完成的人数就较少。

在表 3-2 中概述了学生完成这些编程任务的情况。其中，总评分均值和程序正确性评分均值分别表示的是每个编程任务的总评分和正确性评分完成的均值情况，时间均值表示的是包含以分钟为单位的平均完成时间。因为不是所有的学生都完成了这些任务，所以需要统计他们的完成人数和正确完成的人数。从表中可以看到，前三个任务耗费时间较短，正确完成的人数也比较多。任务 4 耗费的时间最长，参与的人数也最多，但是完成的人数最少。这是由于在设置任务时，前面几个任务在一段时间后就依次关闭提交入口，只有任务 4 设置在最后，并且任务 4 也最为复杂。

表 3-2 每个任务的完成概述

Table 3-2 Overview of each task

变量	总评分均值	程序正确性评分均值	时间均值	完成数量	正确数量
任务 1	9.75	9.40	10.94	81	71
任务 2	6.89	6.37	8.57	56	22
任务 3	7.82	9.55	16.07	92	53
任务 4	15.97	15.22	32.90	111	13

考虑到实验时关闭提交入口的时间过早，以及对学生的编程技能的全面考察。针对学生未通过所有用例测试的情况，在后续工作中对他们的编程任务实际完成情况进行查看审阅，由计算机领域专家对他们提交的源代码进行评估打分，给予代码规范性审查的分数。

3.4.2 比较实验

为了验证实验结果的真实性和可靠性，本文还将 5 名大四学生的实验数据与大三学生的实验数据进行比较和分析。主要是对他们的编程任务中任务 3 和任务 4 的得分进行比较，很明显，大四学生的做题情况要比大三的学生好，得分也更高。如表 2-3 平均分数对比所示，大四学生的平均分明显要比大三学生的平均分高得多。

表 3-3 平均分数对比

Table 3-3 Comparison of average scores

变量	低年级	高年级
任务 3	7.82	19.06
任务 4	15.97	20.00

另外，本文还将他们的编程测试任务的结果与计算机专业具有代表性的算法课程—数据结构的课程成绩进行相关性分析和比较，得出前 20 名学生的编程测试结果与数据结构的课程成绩相关性较高，说明本次实验设计比较成功，得到的实验结果的可靠性较高。

3.5 本章小结

本章通过编程测试方法探讨了直接评估编程新手技能水平的可行性。研究采纳了测试驱动开发的策略，针对选定编程新手的具体情况，设计了一系列旨在衡量编程能力的任务和相应测试用例。这些任务随后在头歌平台上公布，以收集参与者完成任务的数据。通过分析测试成果，包括程序的正确性、开发效率和代码质量等关键指标，本研究对所提方法的有效性进行了验证。所获得的测试结果与研究开始时提出的三个假设一致，证明了方法的可靠性。

第 4 章 基于编程反馈的编程新手编程技能度量研究

4.1 编程反馈过程建模

基于编程测试的编程新手编程技能度量研究仅仅利用了编程新手在编程测试任务中的最终结果，却忽略了编程新手在每次提交问题解决方案的源代码中所包含的丰富的反馈信息。而这些反馈信息更能体现编程新手的编程技能水平，充分且合理地利用这些过程反馈信息能够更加全面的度量编程新手的编程技能。

以往的很多直接法度量编程技能研究主要是基于编程产物的研究。通过在线编程平台，采用动态分析的方法对学习者提交的源程序进行评测，从而进行自动化评估；又或者采用静态分析的方法将最终提交的源代码用抽象语法树的表达方式表示，然后计算学习者提交的源代码和正确的程序代码之间的相似性。这些方法虽然能以学习者提交的最终解决方案的源程序作为编程技能的评估内容，但却忽略了编程反馈练习过程（PFEP，Programming Feedback Exercise Process）与一般练习回答过程的不同。一般练习过程仅以学习者的最终回答结果作为评价内容如图 4-1 所示。而编程反馈练习过程中包含学习者通过多次迭代来解决问题的信息反馈过程，如图 4-2 所示。在每个练习中，学习者可以提交他们的代码反馈，得到评分后可以迭代修改他们的解决方案，因此学习者的编程技能熟练程度不仅反映在他们是否正确回答了题目，还直接反映在他们提交的代码反馈中（显示编程知识），以及解决方案的迭代过程中（显示解决问题的编码能力）。

编程反馈练习过程，就是在编程现场中学习者利用在线编程实践平台提供的工具、语言和环境来完成实际编程任务，他们可以通过提交代码、获取反馈并迭代修改解决方案来不断改进自己的编程技能水平。

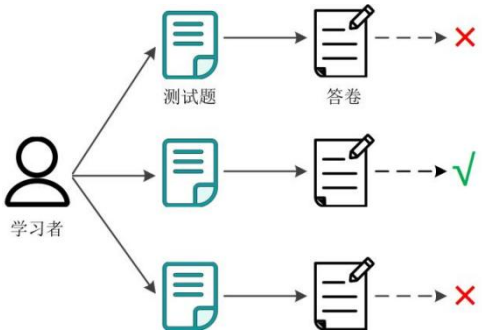

图 4-1 一般练习过程图

Figure 4-1 General exercise process diagram

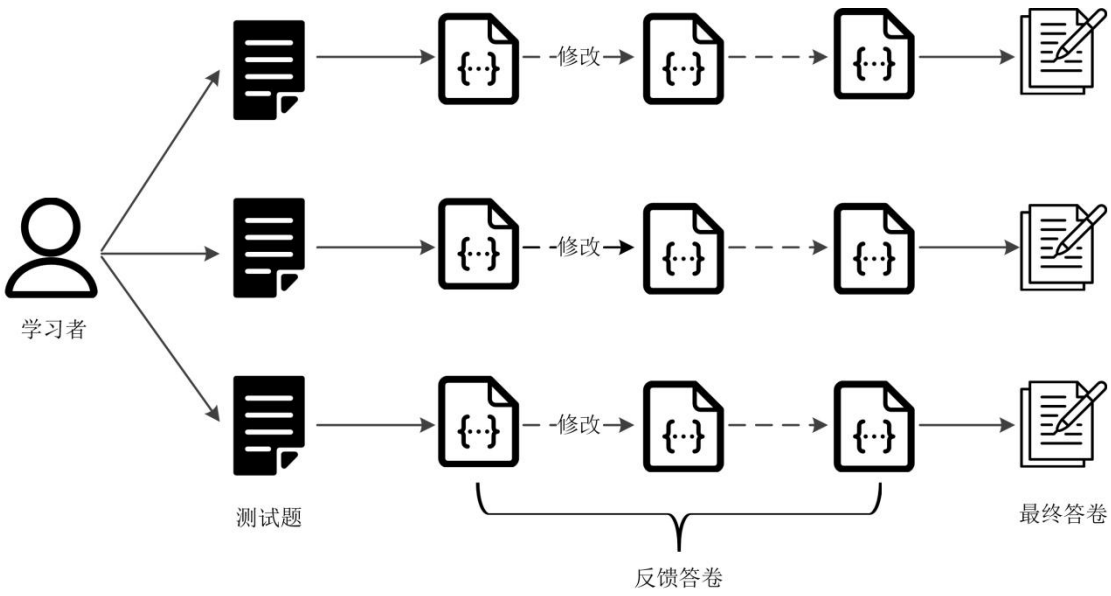

图 4-2 编程反馈练习过程图

Figure 4-2 Programming feedback exercise process diagram

考虑到编程反馈练习过程中所包含的丰富代码反馈信息与一般练习过程不同，因此准确度量编程技能水平主要面临三个挑战：

（1）代码表示

首先，需要很好地表示学习者提交的源代码，代码表示是一种用于编写和理解计算机程序的语言和符号系统，它通过特定的语法和语义规则来描述程序的逻辑和行为。现有的代码表示方法更多的只是关注最终解决方案的源代码，且存在一定的局限性。

（2）代码反馈的迭代过程建模

学习者代码反馈的迭代过程的建模方式，主要是考虑到初始源代码的代码表示构建，以及代码反馈过程中相关的反馈信息与代码在迭代过程中的修改。

（3）编程技能划分

本研究中需要考虑到的是编程技能不仅与学习者的编程知识掌握程度相关，还取决于他们的解决问题的能力，在编程过程中，解决问题通常需要程序员分析问题、理解问题、设计解决方案、编写代码并调试等步骤。

4.2 编程反馈技能追踪模型构建

4.2.1 问题定义

本文将在线评测网站中，学习者对某编程题的答题顺序记录为$LSn=\{s_1,s_2,s_3,...,s_n\}$，其中$s_i=(e_i,c_i,f_i)$表示学习者在第 i 步中提交的解决方案。一般来说，当提交的解决方案的代码c_i满足题目要求的代码e_i时，即可以得到接受的结果（AC），此时$f_i=1$，否则$f_i=0$。

因此，度量编程技能水平问题定义如下：根据学习者编程练习过程的日志LS_n，从而对每个学习者的从编程练习过程第 1 步到第 N 步的编程技能水平度量。

4.2.2 反馈信息定义

本研究将学习者解决问题过程中提交的代码数据作为反馈信息。本研究使用直接度量法获得程序正确性评分W_i，时间效率分E_i和代码质量评分ΔC_i。最后得到反馈表征信息的三元组表示：$S_i=(W_i,E_i,\Delta C_i)$。

本研究还通过反馈信息图（FIG）来表示学习者在解决方案s_i中提交的代码c_i。反馈信息图是表示代码结构信息的有向无环图（DAG），图中的结点表示代码元素，如函数、循环、条件和变量，图中的边表示代码元素之间的控制流和数据流，通过对源代码进行解析并提取相关信息构建反馈信息图。然后通过反馈跟踪图（FTG）来捕捉学习者解决方案迭代的变化。反馈跟踪图也是一个有向无环图，它表示代码的执行跟踪，图中的节点表示代码的执行状态，例如函数调用、循环迭代和条件分支，图中的边表示执行状态之间的转换，该图是通过执行代码和记录执行轨迹来构造的。通过反馈信息图和反馈跟踪图相结合的方式来度量编程新手的编程练习过程中的技能水平。

4.2.3 数据集

本研究主要采用的是两个真实世界的数据集，分别是来自 IBM 提供的 Project_CodeNet 数据集中的 AIZU_Cpp 和来自在线编程竞赛网站 Atcoder.org 的 Atcoder_C 数据集，主要由 C 语言练习组成。

（1）Project_CodeNet 数据集

CodeNet 数据集[75]由大量具有大量元数据的代码样本集合组成。它还包含文

档化的工具，用于将代码样本转换为中间表示，访问数据集并进行定制选择，旨在为编程教育提供丰富的教学资源和实验数据。该数据集包含了大量的编程题目，涵盖了多种编程语言和领域，如 Python、Java、C++、C#等。同时，数据集也涵盖了不同难度级别的题目，适合不同层次的编程教育需求。

在在线评测网站中，以课程和竞赛的形式发布编程问题。数据集由对这些问题的提交组成，这些提交由自动审查过程进行正确性判断。问题描述、提交结果和相关元数据可通过各种 REST APIs 获得。

1）规模和统计

CodeNet 共包含 13916868 个提交，分为 4053 个问题。在提交的材料中，53.6%（7460588）被接受（可编译并通过规定的测试），29.5%的材料被标记为错误答案，其余的材料因未能满足运行时间或内存要求而被拒绝。据了解，这是迄今为止同类数据中最大的数据集。提交的材料有 55 种不同的语言；其中 95%是用 C++、Python、Java、C、Ruby 和 C#编写的。C++是最常见的语言，提交了 8008527 份（占总数的 57%），其中 4353049 份被接受。有了丰富的代码样本，用户可以提取针对其训练需求的的大型基准数据集。具体情况见图 4-3。

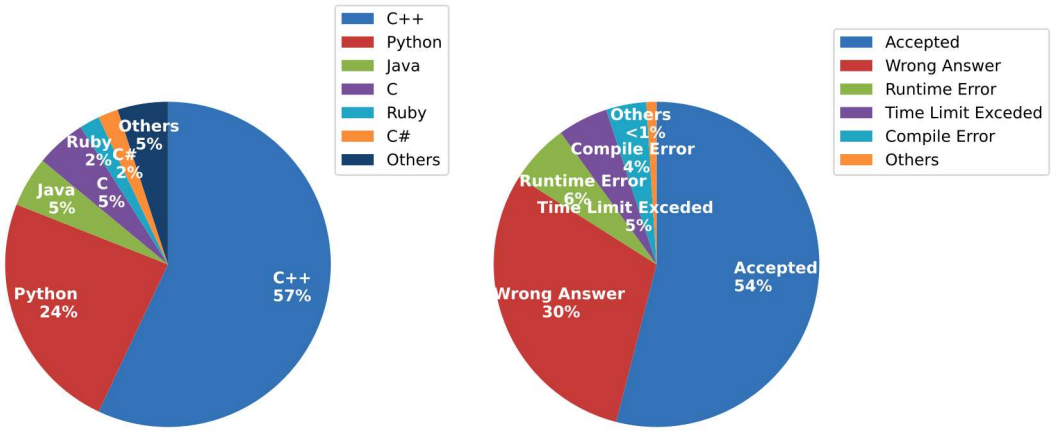

图 4-3 每种语言和每种状态提交的百分比

Figure 4-3 Percentage of commits per language and per state

2）代码示例

每个代码样本都是一个单独的文件，包括输入测试用例和打印计算结果。文件名使用表示编程语言的标准扩展名，例如 Python 的.py。大多数代码示例只包含一个函数，而提交给更复杂的问题则可能包含几个函数。

3）元数据

元数据支持在大量问题、语言和源文件集合中进行数据查询和选择。元数据按两级层次结构组织。第一个是数据集级别，它描述了所有问题；第二个是问题级别，它详细说明了单个问题的所有提交内容。另外，元数据和数据在数据集结构中是分开的。下面将介绍元数据所包含的信息情况：

表 4-1 问题级别的元数据

Table 4-1 Metadata of problem levels

name of colum	data type	unit	description
submission_id	string	none	提交的解决方案的唯一匿名 ID
problem_id	string	none	问题的匿名 ID
user_id	string	none	提交者的匿名 ID
date	int	seconds	提交日期（Unix 格式）
language	string	none	提交语言映射（如 C++14->C++）
original_language	string	none	原始语言规范
filename_ext	string	none	使用的编程语言的文件扩展名
status	string	none	接受状态/错误类型
cpu_time	int	millisecond	执行时间
memory	int	KB	使用内存
code_size	int	bytes	提交源代码大小
accuracy	string	none	提交测试数（AIZU 平台）

表 4-2 数据集级别的元数据

Table 4-2 Metadata at the data set level

name of colum	data type	unit	description
id	string	none	问题的唯一匿名 ID
name	string	none	问题的简称
dataset	string	none	原始数据集，AIZU 或 AtCoder
time_limit	int	millisecond	允许提交的最长时间
memory_limit	int	KB	允许提交的最大内存
rating	int	none	问题的难易等级
tags	string	none	用“|”分隔的标签列表；未使用
complexity	string	none	问题的难易程度；未使用

整体数据集的数据统计如下表：

表 4-3 数据统计表

Table 4-3 Statistical tables

统计情况	AIZU_Cpp	Atcoder_C
参与的学习者数量	5268	6282
编程练习题目数量	2206	1670
解决方案的数量	271189	425238
学习者解决方案的平均值	51.48	67.69
学习者的编程练习过程平均	30.89	36.30
编程练习过程长度的平均值	1.66	2.00
学习者分数的平均值	0.77	0.62
最终解决方案的通过率	74.3%	92.17%

4.2.4 模型方法构建

关于编程反馈技能追踪模型的构建，本研究首先将所有解决方案和解决方案修改转换为嵌入。然后，编程反馈技能追踪模型利用解决方案信息逐步更新编程技能。为了提取和分析每个解决方案的关键信息，本文在编程练习过程中利用不同的信息来更新编程知识和编码能力。其中，预测模块可以利用编程知识和编码能力进行预测。具体的模型框架图如下图所示，并在以下小节中介绍上述子模块。

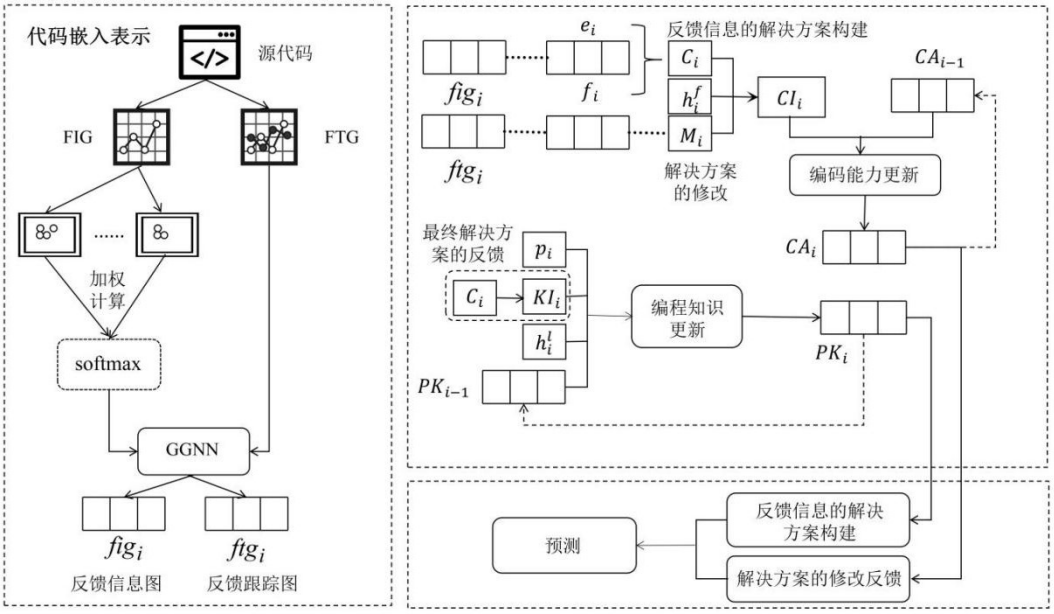

图 4-4 编程反馈技能追踪模型框架图

Figure 4-4 Programming feedback skill tracking model framework diagram

（1）代码嵌入表示

在编程反馈技能追踪模型中，本文分别通过三个嵌入矩阵：$E \in R^{I \times d_e}$，$F \in R^{J \times d_e}$，$P \in R^{K \times d_e}$来表示练习集、反馈集和位置的相关信息。因此，每一个练习、相关反馈和解决方案中的位置可以表示为矢量$e_i \in E$，$f_i \in F$，$p_i \in P$。在文献[76]中，将数据流和函数调用信息集成到抽象语法树中，以表征源代码中的句法和语义信息，从而构造成一个程序图 FDA，其中包括数据流图（DFG）、函数调用图（FCG）以及抽象语法树（AST）的信息。在文献[77]中，为了检测代码的语义克隆，作者构建了一种程序的图表示，称为流增强抽象语法树（FA-AST），通过在 FA-AST 上应用两种不同类型的图神经网络来测量代码对的相似性。受以上文献的启发，本研究采用了反馈信息图和反馈跟踪图来分别对学习者的编程知识和编码能力进行追踪。其中，通过反馈信息图（FIG）来表示学习者在解决方案s_i中提交的代码c_i；通过反馈跟踪图（FTG）来捕捉学习者解决方案迭代的变化。

其中反馈信息图提供了关于代码结构复杂性的信息，而反馈跟踪图提供了关于代码执行行为的信息。但是并不是每一次反馈都是高质量的反馈信息，学习者在进行答题练习时会有很多不可控的人为因素，例如随意提交或者误触提交按钮等等。所以本研究对学习者完成的每道答题练习中，提交的多幅反馈信息图赋予了不同的权重来表示它们的重要程度。本研究使用反馈表征信息的调和平均值计算每副反馈信息图的权重，最后利用 softmax 对最终结果进行映射。例如学习者对某编程题的答题过程为：$\mathrm{LSn} = \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, \ldots, \mathrm{s}_\mathrm{n}\}$，其中第 i 个反馈信息图$\mathrm{S_i}$的反馈表征信息为$\mathrm{S_i}$ =（$\mathrm{W_i}$，$\mathrm{E_i}$，$\Delta\mathrm{C_i}$），则该图的权值$\mathrm{gW_i}$为：

$$gW_i = \frac{1}{\frac{1}{W_i} + \frac{1}{E_i} + \frac{1}{\Delta C_i}} \tag{4-1}$$

因此，第 j 题的所有反馈信息图的权值为：

$$\mathrm{GW_j} = \mathrm{softmax}(\mathrm{gG_1}, \mathrm{gG_2}, \mathrm{gG_3} \ \cdots \ \mathrm{gG_n}) \tag{4-2}$$

其中，n 表示第 j 题中反馈信息图的数量。

然后使用门控图神经网络（GGNN）来学习带权值的反馈信息图中的解决方

案嵌入 fig_i 和解决方案迭代的嵌入 ftg_i，在解决方案 s_i 用于通过预训练任务来预测源代码是否能够获得接受的结果。接着，利用解决方案 s_i 的三元组来表示完整的解决方案信息 c_i，并利用之前完整的解决方案信息 $c_{i\text{-}1}$ 和修改信息 ftg_i 在解决方案 s_i 中表示该解决方案的修改信息 M_i。

（2）解决方案构建

在进行编程时，编程学习者首先根据自己的编程知识构建练习的解决方案。因此，本文利用完整的解决方案信息来更新他们的编程知识。首先融合完整的解决方案信息 c_i 获取知识信息 KI_i，然后利用 $PK_{i\text{-}1}$ 和 KI_i 来更新编程知识 $PK_{i\text{-}1}$，如下所示：

$$
\begin{aligned}
KI_i &= tanh(W_1 \cdot C_i + b_1) \\
\Gamma_i^{lk} &= \sigma(W_2 \cdot (KI_i \oplus PK_{i-1} \oplus p_i) + b_2) \\
\Gamma_i^{fk} &= \sigma(W_3 \cdot (KI_i \oplus PK_{i-1}) + b_3) \\
LG &= tanh(W_4 \cdot (KI_i \oplus PK_{i-1}) + b_4) \\
PK_i &= \left(1 - h_i^l\right) \cdot PK_{i-1} + h_i^l \cdot \left(\Gamma_i^{fk} \cdot PK_{i-1} + \Gamma_i^{lk} \cdot LG\right)
\end{aligned}
\qquad (4\text{-}3)
$$

其中，如果 s_i 是编程练习过程中的最终解决方案，则将 h_i^f 设置为全一向量，否则将 h_i^f 设置为全零向量。

（3）解决方案的迭代

通常，在编程练习过程中初始的解决方案代码并不能通过练习题目的要求。因此，在学习者提交源代码后，他们会收到有关该源代码的反馈信息，并决定是否更新他们的解决方案。因此，本文专注于编程学习者对源代码的修改，即解决方案的迭代。

根据上述描述，本研究通过两方面的信息来更新编码能力：初始解决方案和解决方案的迭代修改，然后利用 CI_i 更新编码能力 $CA_{i\text{-}1}$，如下所示：

$$
\begin{aligned}
CI_i &= tanh\left(W_5 \cdot \left(h_i^i \cdot C_i \oplus \left(1 - h_i^i\right) \cdot M_i\right) + b_5\right) \\
\Gamma_i^{lc} &= \sigma(W_6 \cdot (CI_i \oplus CA_{i-1}) + b_6) \\
\Gamma_i^{fc} &= \sigma(W_7 \cdot (CI_i \oplus CA_{i-1}) + b_8)
\end{aligned}
$$

$$CA_i = \Gamma_i^{fc} \cdot CA_{i-1} + \Gamma_i^{lc} \cdot LG \quad (4\text{-}4)$$

其中，如果s_i是编程练习过程中的初始解决方案，则将h_i^i设置为全一向量，否则将h_i^i设置为全零向量。

（4）性能预测

在建模过程之后，本研究通过对一个完整的编程练习过程来进行建模，利用编程技能来预测下一个编程练习过程的性能。首先通过编程练习要求e_{i+1}和编程知识PK_i得到初始解决方案$solution_{i+1}$，然后对代码编写过程进行建模以形成答案$answer_{i+1}$，即解决方案的源代码$solution_{i+1}$和编码能力CA_i，最后，对下一个编程练习过程的性能和下一个解决方案的得分进行预测如下：

$$\begin{aligned} solution_{i+1} &= tanh(W_9 \cdot (PK_i \oplus e_{i+1}) + b_9) \\ answer_{i+1} &= tanh(W_{10} \cdot (CA_i \oplus solution_{i+1}) + b_{10}) \\ pred_{i+1} &= tanh(W_{11} \cdot answer_{i+1} + b_{11}) \\ pred_{i+1}^s &= tanh(W_{12} \cdot answer_{i+1} + b_{12}) \end{aligned} \quad (4\text{-}5)$$

（5）目标函数

为了学习模型中的参数，本文选择预测回答 *pred* 和实际回答 a 的二进制之间的交叉熵对数损失，以及预测得分 $pred^S$ 和实际得分 s 的 MSE 损失作为目标函数的连续准则。如下所示：

$$L = \alpha \cdot \sum_{i=1}^{\mathrm{M}} (a_i \log pred_i + (1 - a_i) \log (1 - pred_i)) + \beta \cdot \sum_{j=1}^{N} \left| pred_j^s - s_j \right| \quad (4\text{-}6)$$

其中，α和β是超参数，N 和 M 分别是编程练习过程和解决方案的数量。

4.2.5 实验结果和分析

本章节用编程技能水平度量任务来验证模型的有效性。首先通过任务 1 训练编程反馈技能追踪模型，然后在其他任务中微调模型。具体的任务描述如下：

（1）任务 1：预测下一个编程练习过程是否会得到 AC 结果。

（2）任务 2：预测下一个编程练习过程中所有解决方案的平均得分。

（3）任务 3：预测下一次编程练习过程中初始解决方案的得分。

（4）任务 4：预测下一次编程练习过程中最终解决方案的得分。

任务 1 用于评估效率以测量模型的编程技能。此外，在假设中，编程练习过程的初始解和提交时间可以用来评估模型如何适应编码能力，因此本研究选择任务 2 和任务 3 来评估模型如何跟踪编码能力。

此外，考虑到最终解决方案与学习者的真实解决方案之间的相似性，因此提出任务 4 以评估模型在适应学习者实际解决方案方面的表现。对于第一个任务，本文将学习者在编程练习过程中至少得到一个 AC 结果表示为 1，否则表示为 0。然后根据准确度（ACC）和曲线下面积（AUC）来评估模型性能。对于最后三个任务，本研究选择均方根误差（RMSE）来量化预测分数和实际分数之间的距离。

表 4-4 Atcoder_C 编程技能水平测量的比较方法的结果

Table 4-4 Results of comparison methods for measuring Atcoder_C programming skill level

数据集	Atcoder_C				
方法	任务 1		任务 2	任务 3	任务 4
	AUC	ACC	RMSE	RMSE	RMSE
codeBERT	0.6679	0.8001	0.3017	0.3592	0.3005
PDKT	0.6066	0.7556	0.3494	0.3970	0.3569
DKT	0.6955	0.8100	0.2962	0.3576	0.2904
EERNNA	0.7106	0.8040	0.2998	0.3597	0.2954
EERNNM	0.7138	0.8085	0.2952	0.3556	0.2908
PFST	**0.7146**	**0.8103**	**0.2931**	**0.3541**	**0.2903**

表 4-5 AIZU_Cpp 编程技能水平测量的比较方法的结果

Table 4-5 Results of comparison methods for measuring AIZU_Cpp programming skill level

数据集	AIZU_Cpp				
方法	任务 1		任务 2	任务 3	任务 4
	AUC	ACC	RMSE	RMSE	RMSE
codeBERT	0.5054	**0.9600**	0.2309	0.3165	0.1765
PDKT	0.5078	0.9184	0.3054	0.3773	0.2662
DKT	0.5119	0.9588	0.2714	0.3689	0.1918
EERNNA	0.5022	0.9598	0.2535	0.3519	0.1765
EERNNM	0.5258	0.9592	0.2501	0.3475	0.1746
PFST	**0.5279**	0.9581	**0.2245**	**0.3156**	**0.1740**

实验结果如表 4-4 和表 4-5 所示。首先，与 codeBERT 相比，本文的 PFST

模型在 Atcoder_C 和 AIZU_Cpp 中的最后三个任务中表现出色，但在 AIZU_Cpp 中，PFST 相对于 codeBERT 的优势很小。FIG 表示学习者源代码的效率已经得到证明，但可能由于预训练过程阻碍了 FIG 很好地表示学习者的源代码，因此与 codeBERT 相比，PFST 在 AIZU_Cpp 中的表现相似。

其次，PFST 在 AIZU_Cpp 中的最后三个任务上比 DKT 和 EERNN 都获得了更好的结果。与之前的观察相反，知识跟踪模型在 Atcoder_C 中可以获得更接近 PFST 的结果。细粒度的解决方案信息确实可以帮助模型更好地表示编程学习者（尤其是新手）的编码能力，因此 PFST 可以通过更容易的练习（具有更高的 AC 率和更高的平均分数）优于 AIZU_Cpp 上的知识跟踪模型。将编程技能划分为编程知识和编码能力是有效的。

第三，与不考虑编程练习过程的 PDKT 相比，PFST 模型在所有数据集中的所有任务上都表现出色，这说明了编程练习过程建模过程的效率。另外值得注意的是，EENNM 在两项任务中表现优于 EENNA，推测可能是由于相似的练习文本影响了 EENNA 的注意力机制，导致其无法捕捉两项练习之间的关系。

4.3 本章小结

本章主要研究基于编程反馈技能追踪模型来度量学习者在编程练习过程中的编程技能水平。首先通过反馈信息图来表示学习者解决方案的代码特征，再利用反馈跟踪图来测量学习者相邻提交的代码之间的变化。通过反馈信息图和反馈跟踪图分别对学习者的编程知识和编码能力进行建模，从而实现对编程技能水平的更细粒度的评估。

第 5 章 总结与展望

随着互联网企业对提升软件质量的的需求越来越大，它们对程序员的编程技能要求也越来越高。而复杂度和可变性是软件研发中的根本困难，概念结构在说明、设计和测试上的复杂度，在短期内没法通过更好的编程语言和更好的工具来消除。虽然软件行业在这 40 年里蓬勃发展，但从业者依然在消除复杂度上缺乏有效的方法，这也导致了软件行业在工业化的进程中依然极度依赖于程序员的编程技能。因此，度量并提升编程新手的编程技能水平对软件行业来说十分重要。本章对全文进行了总结，也分析了本文的不足之处，并对未来的研究作出展望。

5.1 研究总结

本研究通过相关文献调研，对编程新手的编程技能度量方法进行了深入的了解和剖析，然后基于编程现场的编程数据，实践了基于编程测试的编程技能度量研究，并提出了课程分数度量编程新手的编程技能水平。然后结合知识追踪领域模型知识，构建了编程反馈技能追踪模型，动态评估编程新手的编程技能水平。基于编程现场的编程数据，本研究得出的结论有：

（1）合理的编程测试任务可以对编程新手的编程技能水平进行度量。

基于编程测试的编程新手编程技能度量研究主要是基于测试驱动的软件开发的思想下，在线上编程判定系统中通过有效的编程任务，结合编程新手完成编程任务的时间，评测次数以及代码的规范情况评估编程新手的编程技能水平。但在设计相关编程任务时，需要考虑编程任务的区分度，以更细粒度的评估指标才能更为准确地度量编程技能。

（2）基于成果导向教育理念的教学课程成绩在一定程度上能够度量编程新手的编程技能水平。

在基于编程测试的编程技能度量结果的基础上，收集计算机专业本科生的相关课程成绩，并进行预处理，将获得的课程成绩数据与他们的编程测试结果进行相关性分析，得到了一个初步结果，在 13 个课程里，有 7 个课程是最显著相关的。这些课程里，测试分数相关性和软件工程最强，然后与科学计算语言，数据库原理及应用 I，数据结构，操作系统，Java 程序设计，计算机组成与结构也很

相关，其中与马克思主义基本原理概论很不相关。其中，大学英语也与编程技能比较相关，说明英语能力对于编程新手来说也很重要。

（3）基于编程反馈技能追踪模型构建的方法可以有效追踪编程新手的编程技能水平。

基于编程反馈技能追踪模型的编程新手编程技能度量研究主要是针对编程新手编程练习过程中的数据信息，将编程技能划分为编程知识和编码能力，然后分别对其编程练习过程构建的解决方案信息以及解决方案的迭代进行建模，从而实现对编程技能水平的更细粒度的评估。

5.2 研究不足与展望

本文基于编程现场的编程新手编程技能度量研究在对编程新手的编程技能度量和追踪中提出了有效的度量方式，以及编程反馈技能追踪模型，实现了编程新手的编程技能的有效度量，为后续的编程新手编程技能水平的评估和提升提供了可供参考的度量方法。但是，本文的研究仍然存在许多不足，具体如下：

（1）编程测试任务的实验设计不够完善。对于编程练习过程来说，在线编码是十分重要的环节，由于本人的能力和时间有限，且编程新手的编程技能水平普遍不高，提升空间很大。因此，在对编程新手进行评估的编程测试任务的设计上缺乏一定的可区分性，无法在更细粒度的区间里对编程新手的编程技能水平进行划分和度量。

（2）编程反馈技能追踪模型的准确度还有待提升。在源代码的解决方案的信息迭代上还可以改进，以更好地提取学习者的代码特征，并在将来可以对学习者的多维编程知识进行度量。

在未来的研究工作中，可以对编程测试的实验环节进行改进，更换更能细粒度区分的编程任务作为实验的基本度量任务，并将编程新手的测试过程中的编程行为信息与编程测试分数相结合，从而对编程新手的技能水平实现更细粒度的评估。

参考文献

[1] Wu T, He S, Liu J, et al. A brief overview of ChatGPT: The history, status quo and potential future development[J]. IEEE/CAA Journal of Automatica Sinica, 2023, 10(5): 1122-1136.

[2] Chunmao J, Ying D. Preliminary exploration of computer practical course reform in context of ChatGPT[J]. Experimental Technology & Management, 2023, 40(12).

[3] 吴兰岸, 闫寒冰, 黄发良, 等. 大型语言模型在高等教育中的应用分析与现实挑战[J]. Modern Educational Technology, 2023, 33(8).

[4] 陶传奇, 包盼盼, 黄志球, 等. 编程现场上下文深度感知的代码行推荐[J]. 软件学报, 2020, 32(11): 3351-3371.

[5] 杨宇霞.编程能力评估模型研究进展[J].现代计算机(专业版),2019,(02):33-36+40.

[6] Sillito J, Murphy G C, Volder K D. Asking and Answering Questions during a Programming Change Task[J]. IEEE Transactions on Software Engineering, 2008, 34(4):434-451.

[7] Ricca F, Di Penta M, Torchiano M, et al. The role of experience and ability in comprehension tasks supported by UML stereotypes[C]. 29th International Conference on Software Engineering (ICSE'07). IEEE, 2007: 375-384.

[8] Bunse C. Using patterns for the refinement and translation of UML models: A controlled experiment[J]. Empirical Software Engineering, 2006, 11(2): 227-267.

[9] Silva L, Mendes A J, Gomes A, et al. Investigating programming students problem comprehension ability and its association with learning performance[J]. IEEE Transactions on Education, 2022, 66(2): 156-162.

[10] Li R, Yin Y, Dai L, et al. PST: Measuring Skill Proficiency in Programming Exercise Process via Programming Skill Tracing[C]//Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022: 2601-2606.

[11] Corbett A T, Anderson J R. Knowledge tracing: Modeling the acquisition of procedural knowledge[J]. User modeling and user-adapted interaction, 1994, 4: 253-278.

[12] Piech C, Bassen J, Huang J, et al. Deep knowledge tracing[J]. Advances in neural information processing systems, 2015, 28.

[13] 朱梦霞. 基于深度学习的编程知识追踪[D].华东师范大学,2022.

[14] 刘敏娜, 张倩苇.国外计算思维教育研究进展[J].开放教育研究, 2018,24(1):41-53.

[15] EsteyBrookshear, J. G. Computer science: An overview. Boston: Pearson Education. Inc, 2003.

[16] 邹艳. 基于深度编程知识追踪的计算思维能力培养研究[D].湖北大学,2021.

[17] Tang T, Smith R, Rixner S, et al. Data-driven test case generation for automated programming assessment[C]//Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education. 2016: 260-265.

[18] Robinson P E, Carroll J. An online system for monitoring and assessing the programming process[J]. Bulletin of the IEEE Technical Committee on Learning Technology, 2016, 18(2/3): 2.
[19] Almajali S. Computer-based tool for assessing advanced computer programming skills[C]//2012 International Conference on E-Learning and E-Technologies in Education (ICEEE). IEEE, 2012: 114-118.
[20] Hammer S, Hobelsberger M, Braun G. Using laboratory examination to assess computer programming competences: Questionnaire-based evaluation of the impact on students[C]. 2018 IEEE Global Engineering Education Conference (EDUCON). IEEE, 2018: 355-363.
[21] Barkmin M, Brinda T. Analysis of Programming Assessments—Building an Open Repository for Measuring Competencies[C]. Proceedings of the 20th Koli Calling International Conference on Computing Education Research. 2020: 1-10.
[22] Kittur J. Measuring the programming self-efficacy of Electrical and Electronics Engineering students[J]. IEEE Transactions on Education, 2020, 63(3): 216-223.
[23] Adelakun O. A Metrics-Based Model for Programming Skill[C]. Proceedings of the 2020 ACM Conference on International Computing Education Research. 2020: 338-339.
[24] 罗文劼,李明杰,肖梓良.基于熵权-离差的 GA-BP 神经网络编程能力评估方法[J].科学技术与工程,2021,21(10):4117-4123.
[25] Reek K A. The TRY system-or-how to avoid testing student programs[C]//Proceedings of the twentieth SIGCSE technical symposium on Computer science education. 1989: 112-116.
[26] Higgins C, Hegazy T, Symeonidis P, et al. The coursemarker cba system: Improvements over ceilidh[J]. Education and Information Technologies, 2003, 8: 287-304.
[27] Cheang B, Kurnia A, Lim A, et al. On automated grading of programming assignments in an academic institution[J]. Computers & Education, 2003, 41(2): 121-131.
[28] Vujošević-Janičić M, Nikolić M, Tošić D, et al. Software verification and graph similarity for automated evaluation of students' assignments[J]. Information and Software Technology, 2013, 55(6): 1004-1016.
[29] Wang T, Su X, Wang Y, et al. Semantic similarity-based grading of student programs[J]. Information and Software Technology, 2007, 49(2): 99-107.
[30] Hovemeyer D, Hellas A, Petersen A, et al. Control-flow-only abstract syntax trees for analyzing students' programming progress[C]//Proceedings of the 2016 ACM Conference on International Computing Education Research. 2016: 63-72.
[31] 张宏伟. 基于语义理解的编程题自动评分系统的研究与实现[D]. 大连海事大学, 2010.
[32] Srikant S, Aggarwal V. A system to grade computer programming skills using machine learning[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 1887-1896.
[33] Singh G, Srikant S, Aggarwal V. Question independent grading using machine learning: The case of computer program grading[C]//Proceedings of the 22nd

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016: 263-272.

[34] Ahmadzadeh M, Elliman D, Higgins C. An analysis of patterns of debugging among novice computer science students[C]//Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education. 2005: 84-88.

[35] Altadmri A, Brown N C C. 37 million compilations: Investigating novice programming mistakes in large-scale student data[C]//Proceedings of the 46th ACM technical symposium on computer science education. 2015: 522-527.

[36] Estey A, Keuning H, Coady Y. Automatically classifying students in need of support by detecting changes in programming behaviour[C]//Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. 2017: 189-194.

[37] Watson C, Li F W B, Godwin J L. Predicting performance in an introductory programming course by logging and analyzing student programming behavior[C]//2013 IEEE 13th international conference on advanced learning technologies. IEEE, 2013: 319-323.

[38] Spacco J, Denny P, Richards B, et al. Analyzing student work patterns using programming exercise data[C]//Proceedings of the 46th ACM Technical Symposium on Computer Science Education. 2015: 18-23.

[39] Robillard M P, Coelho W, Murphy G C. How effective developers investigate source code: An exploratory study[J]. IEEE Transactions on software engineering, 2004, 30(12): 889-903.

[40] Ceccato M, Di Penta M, Nagra J, et al. The effectiveness of source code obfuscation: An experimental assessment[C]. 2009 IEEE 17th International Conference on Program Comprehension. IEEE, 2009: 178-187.

[41] Buse R P L, Weimer W R. Learning a metric for code readability[J]. IEEE Transactions on Software Engineering, 2010, 36(4): 546-558.

[42] Kleinschmager S, Hanenberg S. How to rate programming skills in programming experiments? A preliminary, exploratory, study based on university marks, pretests, and self-estimation[C]. Proceedings of the 3rd ACM SIGPLAN Workshop on Evaluation and Usability of Programming Languages and Tools. 2011: 15-24.

[43] Siegmund J, Kästner C, Liebig J, et al. Measuring and modeling programming experience[J]. Empirical Software Engineering, 2014, 19(5): 1299-1334.

[44] Arisholm E, Gallis H, Dyba T, et al. Evaluating pair programming with respect to system complexity and programmer expertise[J]. IEEE Transactions on Software Engineering, 2007, 33(2): 65-86.

[45] Carver J, Hochstein L, Oslin J. Identifying Programmer Ability Using Peer Evaluation: An Exploratory Study[C]. OOPSLA, 2009.

[46] Hannay J E, Arisholm E, Engvik H, et al. Effects of personality on pair programming[J]. IEEE Transactions on Software Engineering, 2010, 36(1): 61-80.

[47] 胡学钢, 刘菲, 卜晨阳. 教育大数据中认知跟踪模型研究进展[J]. 计算机研

究与发展, 2020.
[48] Lei P I S, Mendes A J. A systematic literature review on knowledge tracing in learning programming[C]//2021 IEEE Frontiers in Education Conference (FIE). IEEE, 2021: 1-7.
[49] Zhang L, Xiong X, Zhao S, et al. Incorporating rich features into deep knowledge tracing[C]//Proceedings of the fourth (2017) ACM conference on learning@ scale. 2017: 169-172.
[50] Lalwani A, Agrawal S. What Does Time Tell? Tracing the Forgetting Curve Using Deep Knowledge Tracing[C]//International Conference on Artificial Intelligence in Education. Springer, Cham, 2019: 158-162.
[51] Chen P, Lu Y, Zheng V W, et al. Prerequisite-driven deep knowledge tracing[C]//2018 IEEE International Conference on Data Mining (ICDM). IEEE, 2018: 39-48.
[52] Wang, Zhiwei, et al. Deep knowledge tracing with side information.[C] International conference on artificial intelligence in education. Springer, Cham, 2019: 303-308.
[53] Nakagawa H, Iwasawa Y, Matsuo Y. End-to-end deep knowledge tracing by learning binary question-embedding[C]//2018 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, 2018: 334-342.
[54] Min W, Frankosky M H, Mott B W, et al. DeePFSTealth: leveraging deep learning models for stealth assessment in game-based learning environments[C]//International conference on artificial intelligence in education. Springer, Cham, 2015: 277-286.
[55] Wang L, Sy A, Liu L, et al. Deep knowledge tracing on programming exercises[C]//Proceedings of the fourth (2017) ACM conference on learning@ scale. 2017: 201-204.
[56] Swamy V, Guo A, Lau S, et al. Deep knowledge tracing for free-form student code progression[C]//International Conference on Artificial Intelligence in Education. Springer, Cham, 2018: 348-352.
[57] Zhang J, King I. Topological order discovery via deep knowledge tracing[C]//International Conference on Neural Information Processing. Springer, Cham, 2016: 112-119.
[58] 艾方哲. 基于知识追踪的智能导学算法设计[D]. 北京交通大学, 2019.
[59] Liu F, Hu X, Liu S, et al. Meta multi-agent exercise recommendation: A game application perspective[C]//Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023: 1441-1452.
[60] Shen S, Chen E, Xu B, et al. Quiz-based Knowledge Tracing[J]. arXiv preprint arXiv:2304.02413, 2023.
[61] Sun Y, Wang L, Xie Q, et al. Online programming education modeling and knowledge tracing[C]//Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part I 13. Springer International Publishing, 2020: 259-270.
[62] Kasurinen J, Nikula U. Estimating programming knowledge with Bayesian knowledge tracing[J]. ACM SIGCSE Bulletin, 2009, 41(3): 313-317.

[63] Jiang B, Ye Y, Zhang H. Knowledge tracing within single programming exercise using process data[C]//Proceedings of the 26th international conference on computers in education. 2018: 89-94.
[64] Zhu M, Han S, Yuan P, et al. Enhancing Programming Knowledge Tracing by Interacting Programming Skills and Student Code[C]//LAK22: 12th International Learning Analytics and Knowledge Conference. 2022: 438-443.
[65] Li R, Yin Y, Dai L, et al. PST: Measuring Skill Proficiency in Programming Exercise Process via Programming Skill Tracing[C]//Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022: 2601-2606.
[66] Shi Y, Chi M, Barnes T, et al. Code-DKT: A code-based knowledge tracing model for programming tasks[J]. arXiv preprint arXiv:2206.03545, 2022.
[67] Liu N, Wang Z, Baraniuk R, et al. Open-ended knowledge tracing for computer science education[C]//Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022.
[68] Xu B, Huang Z, Liu J, et al. Learning behavior-oriented knowledge tracing[C]//Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining. 2023: 2789-2800.
[69] Suetake K, Tanaka T. A Support System for Programming Exercises Using Test-first Approach[C]//2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI). IEEE, 2022: 178-181.
[70] 聂鹏, 耿技, 秦志光. 软件测试用例自动生成算法综述[J]. 计算机应用研究, 2012, 29(2): 401-405.
[71] 单锦辉, 姜瑛, 孙萍. 软件测试研究进展[J]. 北京大学学报: 自然科学版, 2005, 41(001): 134-145.
[72] 朱少民，软件测试方法和技术[M]. 清华大学出版社有限公司, 2005.
[73] 万年红, 李翔. 软件黑盒测试的方法与实践[J]. 计算机工程, 2000, 26(12): 91-93.
[74] 李宁, 李战怀. 基于黑盒测试的软件测试策略研究与实践[J]. 计算机应用研究, 2009 (3): 923-926.
[75] Puri R, Kung D S, Janssen G, et al. Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks[J]. arXiv preprint arXiv:2105.12655, 2021.
[76] Lu M, Wang Y, Tan D, et al. Student program classification using gated graph attention neural network[J]. IEEE Access, 2021, 9: 87857-87868.
[77] Wang W, Li G, Ma B, et al. Detecting code clones with graph neural network and flow-augmented abstract syntax tree[C]//2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2020: 261-271.
[78] 章淼, 代码的艺术: 用工程思维驱动软件开发[M]. 电子工业出版社, 2022.

攻读学位期间参与的科研项目

[1] “面向编程新手的编程能力度量方法研究”（项目编号：YJS20210453），安徽高校研究生科学研究项目，参与。

[2] “基于编程现场多源数据融合的软件故障定位与跟踪研究”（项目编号：1908085MF183），安徽自然科学基金面上项目，参与。

攻读学位期间发表的学术论文

[1] Fanghui Zha, Yong Wang, et al. Can University Marks Measure Programming Skills for Novice Programmers? An Exploratory Study[J]. Journal of Internet Technology, 2023, 24(6): 1189-1197.（第一作者，SCI 收录）

[2] Fanghui Zha, Yong Wang. Correlation Analysis of Subject Competition and Programming Ability for Novice Programmers[C]. 2021 7th International Symposium on System and Software Reliability (ISSSR). IEEE, 2021: 25-31.（第一作者，EI 收录）

致谢

行文至此落笔，始于金秋终于盛夏。总觉得来日方长，却不知岁月清浅，时间如流水。回首在安徽工程大学读书的这七年光阴，从不谙世事到历经风霜雨雪，这一路我与许多人相遇，也与许多人告别，或许词不达意，仍想向我所拥有的一切致以深深的谢意。

桃李不言，下自成蹊。感谢我的硕士导师王勇教授，从读研伊始，王勇老师就给予我很多指导，从选题时的反复斟酌和悉心指导，到写作过程中的适时督促和耐心修改，都离不开导师的教诲。初次见面时，便能感受到王勇老师待人温和，处事周到。后来，紧跟老师步伐学习三年，不禁钦佩老师的待人处事之道。王勇老师在学术研究中也保持严谨的科研态度，总是叮嘱我们要精读论文，规划好自己的科研进度。师恩难忘，一路走来老师为我们筹谋了太多，非常感谢王勇老师对学生的关心和帮助。同时也要感谢 PRiSE 课题组的卢桂馥老师、唐肝翌老师、李钧老师、王坤老师等，以及王忠群老师、刘三民老师、陶皖老师等计算机与信息学院所有老师们，老师们对我的关心与指导我也将铭记于心，终生不忘。同时还要感谢宿州学院的崔琳老师，一直给予我支持。

春晖寸草，山高海深。感谢我的家人们，有家人的陪伴，这一辈子才会过得不那么艰辛。感谢我的母亲，母亲这一生过得也非常不容易，等我们长大了母亲也老了，希望未来的时光能好好报答母亲的养育之恩。感谢我的爷爷和奶奶，从小生活在爷爷奶奶膝下，虽缺少太多父母的关爱，但爷爷奶奶所给予的爱也足以让我的童年十分快乐。

山水一程，三生有幸。感谢 PRiSE 课题组的全体同学，很幸运在读研生涯中能与之共处一段时光。感谢我的师姐王雪，师姐勤奋严谨的科研态度永远值得我学习，与师姐共同研究课题的日子也十分难忘。感谢我的同门陈林俊，这三年同甘苦共患难，送别师兄师姐，迎接师弟，同门之谊非常珍惜。感谢我的师兄周芃、乐文革、王守杭，感谢我的师弟方应涛、李傲、周华东、张一凡以及课题组其他的伙伴们，感谢大家陪我度过研究生的三年光阴。

感谢我的研究生室友汲广艳、汪玉洁、陈燕菲，我们一起读研一起吃饭的时光很是珍惜。读研的时光终是短暂，有朋自远方来，又各赴远方。愿我们都能以梦为马，在以后的日子里熠熠生辉。感谢一路陪伴我成长的朋友们，我们都站在彼此的青春里，虽然分散在不同的地方，但却是最关心彼此的朋友，漫漫光阴也走了快十年，相信我们会一直相伴前行。

最后，感谢与我相守五年多的马先生，我们相识于安工程，贯穿了我的大学和读研时光，感谢一路陪伴并且不断引领着我。希望以后平凡的日子里我们都能快快乐乐。

回头望去，是我最炙热的青春。祝愿所有相遇，天高海阔，万事胜意！