WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 8, Number 3, Year 2026
DOI: https://doi.org/10.62051/ijcsit.v8n3.12
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Reform and Practice of Differentiated Teaching in Compiler Technology Courses Driven by Task Engines

JianCai Huang, Yan Yue

College of Control and Computer Engineering, North China Electric Prower University, Baoding, 071003, Hebei, China

ABSTRACT

To address issues such as the abstraction of knowledge points in compiler technology courses, significant differences in students' foundations, the disconnect between theory and practice, and insufficient integration of ideological and political education, a differentiated teaching model based on a task engine is proposed. By breaking down the core course knowledge points into stepwise tasks, a system for generating personalized learning paths is constructed, organically integrating real-life scenarios and ideological and political elements into task design, thereby achieving the dual objectives of "teaching according to students' abilities" and "value guidance." Teaching practice shows that this model can effectively enhance students' learning interest, practical ability, and ideological and political literacy, significantly improving teaching effectiveness.

KEYWORDS

Compiler technology; Task engine; Differentiated teaching; Integration of ideological and political education; Teaching reform

1. INTRODUCTION

Compiler technology is a core foundational course for computer science majors, tasked with the important mission of cultivating students' programming thinking, system optimization abilities, and understanding of underlying logic. However, the course involves abstract theories such as formal languages, automata, and syntax analysis, with highly interconnected knowledge points and a steep difficulty gradient. The traditional 'one-size-fits-all' teaching model is difficult to adapt to students' varying foundations: students with weaker foundations may develop anxiety due to being unable to keep up with the pace, while stronger students may find the content monotonous and lacking challenge. At the same time, the course has long faced the problem of 'emphasizing theory over practice and technology over value,' which not only disconnects it from practical engineering applications but also neglects the educational function of ideological and political elements.

In order to solve the pain points and bottlenecks in current teaching, this paper deeply integrates the task driven teaching method with the concept of differentiated teaching, and proposes an intelligent teaching framework based on the task engine. The core innovation of the framework lies in: first, the abstract and complex knowledge system of compilation technology is deconstructed into a series of concrete and stepped structured tasks to build a cognitive scaffold; Secondly, relying on students' full link learning data to build a dynamic learning portrait, to achieve accurate adaptation and personalized push of teaching paths. In addition, the framework innovatively embeds real life situations and ideological and political elements, opens up the education closed-loop of "knowledge teaching - ability cultivation - value guidance", and effectively promotes the transformation of curriculum from knowledge infusion to comprehensive quality improvement.

2. CURRENT STATUS AND PROBLEMS OF COMPILER TECHNOLOGY COURSE TEACHING

2.1. Abstract and Highly Interconnected Knowledge Points Are Hard to Grasp

The core knowledge points of compiler technology, such as finite automata in lexical analysis, LL (1) grammar in syntax analysis, and attribute grammar in semantic analysis, all belong to formal theory and lack intuitiveness. Moreover, these knowledge points are interconnected; for example, the output of lexical analysis is the input for syntax analysis. Weakness in any link can lead to obstacles in subsequent learning and exacerbate students' learning difficulties.

2.2. Significant Differences in Students' Basic Abilities and Lacks of Specificity in Teaching

The level of mastery of prerequisite knowledge (such as advanced programming, data structures, and discrete mathematics) among computer science students varies greatly: some students have strong programming skills and logical thinking, while others have a limited understanding of core concepts such as pointers and recursion. Traditional teaching uses unified teaching content, pace, and evaluation standards, which cannot meet the learning needs of students at different levels.

2.3. Disconnect Between Theory and Practice: Lack of Real-Life Relevance

Traditional teaching is mainly focused on theoretical lectures, with practical tasks mostly adapted from textbook examples, disconnected from actual engineering applications and real-life scenarios. Students find it difficult to understand 'what compiler technology is useful for,' leading to a lack of motivation to learn, and even a passive state of 'learning for the sake of exams.'

2.4. Insufficient Integration of Ideological and Political Elements and Lack of Value Guidance

Course instruction focuses on the transmission of technology, neglecting the cultivation of students' patriotism, craftsmanship spirit, and awareness of teamwork. Students lack recognition of the importance of independently controllable compiler technology, making it difficult to align personal development with the nation's information security needs.

3. DIFFERENTIATED INSTRUCTION DESIGN BASED ON A TASK ENGINE

3.1. Knowledge Point Decomposition and Task-based Reconstruction

With the main line of "lexical analysis-syntax analysis- semantic analysis-intermediate code generation-object code generation" of compilation technology, abstract knowledge points are decomposed into three tiered tasks "basic tasks promotion tasks innovation tasks" to ensure the representativeness and relevance of tasks. In order to realize the trinity of "knowledge transfer, ability cultivation and value guidance", the following three-dimensional corresponding table is constructed to clarify the knowledge points, ladder tasks and ideological and political integration points of each module.The following table takes the lexical analysis knowledge module as an example to introduce knowledge point decomposition and task-based reconstruction:

Table 1. Knowledge points, ladder tasks and ideological and political integration points (Take lexical analysis as an example)

Knowle-dge point module	Core knowledge points	Ladder mission system	Elements of ideological and political integration	Life scene associ-ation
lexical analysis	-Regular expressin -Finite automata (NFA/DFA) -Lexical scanning principle	Basic task: use regular expressions to define identifiers and keywords Promotion task: implement finite automaton simulated lexical scanning (identify constants/variables/operators) Innovative task: develop text editor syntax highlighting function (suitable for C++language)	Craftsman spirit:(preci-se definition of regular expressions &cultivation of preciseness) Feelings of family and country: (innovative tasks are integrated into the "highlighting development of domestic editor grammar") Responsibili-ty: (identify malicious code keywords and associate network security)	VS Code syntax highlig-hting, browser search keyword matchin-g [1]

3.2. Differentiated Path Generation

Based on the task engine, students' ability model is dynamically constructed through multi-dimensional data:

Pre knowledge diagnosis: acquire basic ability data through advanced language programming, data structure, discrete mathematics and other pre course tests;

Real time learning feedback: analyze process data such as students' task completion quality, correct answer rate, and code debugging efficiency;

Learning preference collection: collect students' preferences for theoretical learning, practical operation, team cooperation, etc. through questionnaires.

Based on the above data, students are divided into three categories to generate differentiated learning paths:

• Basic students: give priority to completing basic tasks, strengthen the understanding of knowledge points, and provide auxiliary resources of "theory micro class+example disassembly";

• Promotional students: enter the promotion task after the basic task reaches the standard, focusing on practical application, and supporting "engineering case+code template";

• Innovative students: skip some basic tasks, focus on innovative tasks, support "cutting-edge literature+open source projects", and encourage independent exploration [2].

Differentiation path generation is shown in the following figure:

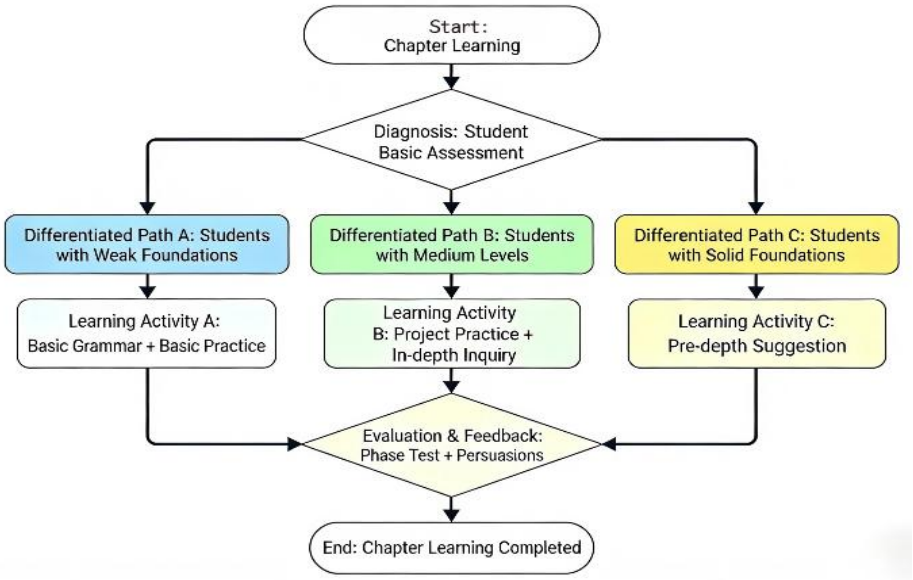

Figure 1. Differentiation path generation

Dynamic branch trigger condition (trigger based on AI assistant): take "task completion degree" as the core indicator (such as basic task completion degree ≥ 80%, promotion task completion degree ≥ 90%) to trigger path upgrade; If the degree of completion is not up to the standard, students will enter the "supplementary task" or "consolidation exercise" to avoid giving up because of difficulty.

Differentiation path characteristics:

Basic students: "basic tasks→consolidation exercises→ simple promotion tasks", focusing on the consolidation of knowledge points, and supporting "case disassembly+micro class explanation" resources;

Promotional students: "basic task rapid clearance→core promotion task→optional innovation task", focusing on practical ability, supporting "code template+engineering case";

Innovative students: "skip basic tasks→improve tasks→ deeply innovate tasks", focus on the integration of innovation and family and country feelings, and support "open source projects+cutting-edge literature".

3.3. Integration of Real Life Scenes

The task design is closely combined with life and engineering practice, so that abstract technology can be "landed":

Lexical analysis module: design the task of "identifying code keywords and coloring" in combination with the syntax highlighting function of text editors (such as VS Code);

Grammar analysis module: design the task of "analyzing the subject predicate object structure of simple Chinese sentences" by relating to "sentence component analysis" in natural language processing;

Intermediate code generation module: interface with cross platform application development and design the task of "converting simple Java code to cross platform intermediate code";

Object code generation module: contact the mobile APP installation process and design the task of "simulating the simplified process of compiling Android applications into APK" [3].

3.4. Organic Integration of Ideological and Political Elements

The organic integration of ideological and political elements follows the principle of "no trace of ideological and political elements, no noise", and integrates ideological and political elements into task objectives, cases and evaluations:

Cultivate a sense of family and country: set the task of "domestic compilers adapt to Godson chips" in the object code generation module, introduce the development history of China's independent compilers (such as Mulan compiler) and Godson chips, and emphasize the importance of independent and controllable underlying technology for national information security;

Artisan spirit building: In the task of code debugging, students are required to strictly check grammar errors, optimize code efficiency, and cultivate the craftsman spirit of preciseness, meticulousness and excellence;

Team cooperation awareness: set up a group task of "compiler module collaborative development", requiring students to be responsible for lexical analysis, grammatical analysis and other modules, and cultivate communication and cooperation ability and collectivism spirit through cross module docking;

Education of responsibility: introduce the case of "compiler vulnerabilities lead to system collapse", guide students to understand the social responsibility of technicians, and establish the value of "technology for good".

4. TEACHING IMPLEMENTATION PATH

(1) Before class: diagnosis and positioning, accurate empowerment

Students complete the pre knowledge test through the task engine, and the engine generates the initial portrait and preview tasks: basic students preview the "regular expression foundation", advanced students preview the "finite automata principle", and innovative students preview the "open source compiler architecture". Teachers view the learning situation data through the engine and design the classroom teaching content pertinently.

(2) In class: layered implementation, dynamic adjustment

1) Basic task stage (30 minutes): all students complete the basic tasks corresponding to the core knowledge points, and the engine counts the completion in real time;

2) Path differentiation stage (60 minutes): basic students consolidate exercises under the guidance of teachers, improvement students carry out practical tasks (such as writing simple lexical analyzers), and innovative students expand exploration (such as researching compiler optimization algorithms);

3) Ideological and political integration stage (15 minutes): deepen students' understanding of the value of technology through case sharing, group discussion and other forms. For example, after the task of "domestic compiler", organize a discussion on the theme of "independent innovation of underlying technology".

(3) After class: expansion and extension, closed-loop evaluation

1) Differentiated homework: basic students complete the task of adapting textbook exercises, advanced students complete small application development (such as text grammar checking tools), and innovative students complete research reports (such as "suggestions for the development of domestic compilers");

2) Feedback optimization: students submit homework and learning feedback through the engine, teachers generate evaluation reports after correction, and the engine adjusts the follow-up learning path according to the feedback;

3) Achievement exhibition: "Compilation Technology Application Achievement Exhibition" is held regularly to encourage students to share their works (such as grammar highlighting tools and simple compilers) and enhance their sense of learning achievement.

5. TEACHING EFFECT VERIFICATION

Two parallel classes of Compilation Technology in the autumn semester of 2025 are selected as the research objects. The experimental class (40 people) uses task engine driven differentiated teaching, and the control class (61 people) uses the traditional teaching mode. The experimental period is one semester (40 class hours of theoretical lessons+16 class hours of practical lessons). The effect is evaluated by the following indicators:

(1) Academic performance: compare the theoretical test scores (40%) and practical task scores (60%) of the two classes;

(2) Learning interest: use questionnaires (10 questions) to assess students' interest and initiative in the course;

(3) Ideological and political literacy: evaluate the value leading effect through the ideological and political awareness questionnaire (8 questions) and the ideological and political perception score in the task report;

(4) Ability improvement: compare the programming practice ability and problem solving ability of the two classes of students (by scoring the course design works).

Result analysis:

(1) Academic performance: The average score of the experimental class is 82.3 points, and that of the control class is 70.4 points. The passing rate (96%) and excellence rate (32%) of the experimental class are significantly higher than that of the control class (82%), indicating that differentiated teaching can effectively improve students' knowledge mastery;

(2) Learning interest: the average scores of the students in the experimental class on "curriculum practicality" and "learning initiative" were 4.2 and 4.1, respectively, while those in the control class were 3.1 and 2.9, indicating that the integration of life scenes has improved the students' learning motivation;

(3) Ideological and political literacy: the average score of the students in the experimental class on the cognition of "the importance of technological autonomy" and "craftsmanship spirit" is 4.3 points, and the average score of the control class is 3.2 points, and the depth and breadth of ideological and political perception in the task report of the experimental class are better than the control class;

(4) Ability improvement: The quality of completion of curriculum design works in the experimental class (such as functional integrity and code optimization) is significantly higher than that in the control class. 80% of the students in the experimental class can independently complete the development of small compiler modules, while only 45% of the students in the control class can.

6. CONCLUSION

The task engine driven differentiated teaching mode constructed in this paper effectively solves the teaching pain points of the compiling technology course through the task-based disassembly of knowledge points, the generation of personalized paths, the integration of life scenes and the integration of ideological and political elements. Practice has proved that this model can not only improve students' theoretical level and practical ability, but also strengthen the value guidance, and cultivate computer professionals with family feelings, craftsmanship spirit and collaboration awareness.

In the future, we can further optimize the intelligence level of the task engine, and introduce machine learning algorithms to improve the accuracy of student portraits and path matching; Expand the integration scenarios of ideological and political elements, and design more contemporary teaching tasks in combination with cutting-edge technologies such as artificial intelligence and big data; Strengthen inter school cooperation and resource sharing, build a trans college task library of compiling technology courses and a library of ideological and political cases, and promote the in-depth promotion of teaching reform.

ACKNOWLEDGMENT

This work was financially supported by higher education teaching reform research and practice project in Hebei Province (No.2025GJJG691, Project Title: Research and Practice on Addressing Learning Difficulties in Computer Science Courses with Large AI Models).

REFERENCES

[1] Wang Ting, Xu Chang, Yang Haiyan, "Construction of Practical Teaching Resources for Compiler Course," J. Computer Education, no. 11, 2025, pp. 1–7.

[2] Zhang Li, Zhang Yu, Xu Chang, "Compilation Course Textbook Construction," J.Computer Education no. 11,2024, pp.22–27.

[3] Feng Xiaobing, Hao Dan, Gao Yaoqing, "Preface to Special Topics on Compilation Technology and Compiler Design," J. Journal of Software, vol 35, no 6, 2024, pp. 2583-2584.