[1,*]Xianping Wang

Construction and Exploration of Experimental Teaching Content in Software Engineering from a Systems Theory Perspective

Abstract: - Software engineering, as a branch of traditional computer science, distinguishes itself from other computer-related disciplines by its emphasis on practical application. Experimental teaching in software engineering, integral to professional education, serves as a crucial means to enhance students' practical skills. The content of experimental teaching, serving as a vessel for such instruction, plays a pivotal role in the holistic development of students. Addressing the current fragmented state of experimental teaching in applied universities, this study introduces a holistic and systematic view based on systems theory. By integrating the fundamental principles and methods of systems theory with the distinctive characteristics of software engineering, this study focuses on analyzing the holistic technology of software implementation and the interrelationships among various technical elements of software systems within the field of software engineering. From a holistic-part-holistic analytical perspective, this research constructs an analytical model for experimental teaching content in software engineering under the lens of systems theory. The study elaborates on the construction method of this model and its specific implementation details from the perspectives of talent cultivation programs, engineering elements, and concentrated practical exercises. Practical evidence indicates that the implementation of this experimental teaching content effectively enhances students' practical skills in software engineering, elevates students' recognition of their field of study, boosts their confidence in job placement, and provides robust support for seamless transition from graduation to employment. Furthermore, this study sheds light on the construction of experimental teaching content in computer-related fields in engineering disciplines at other applied universities.

Keywords: Systems Theory; Computer Science; Software Engineering; Experimental Teaching; Teaching Content.

I. INTRODUCTION

The rapid advancement of next-generation information technologies represented by Mobile Internet, Internet of Things, Cloud Computing, Big Data, Artificial Intelligence, and Blockchain has led to their widespread application, becoming a driving force behind social and economic development[1]. At the forefront of these technologies, the software industry has garnered significant attention as the soul of information technology. Xie Shaofeng, Director of the Ministry of Industry and Information Technology's Information Technology and Software Service Division in China, presented a work report titled "Accelerating the Development of Information Technology and Software Services, and Fully Advancing the Building of a Strong Manufacturing Nation". Director Xie emphasized, "As a strategic emerging industry with the most active global innovation, the strongest radiation, and the widest penetration, the software industry is rapidly penetrating and integrating into all aspects of the economy and society, becoming the core area and strategic high ground of international industrial competition." Subsequently, during the formulation of the "14th Five-Year Plan", the Ministry of Industry and Information Technology once again emphasized that software is the soul of the new generation of information technology, serving as the foundation for digital economic development and a key pillar supporting the construction of a manufacturing powerhouse, a strong network nation, and the digitalization of China. The development of the software and information technology service industry holds significant importance in accelerating the construction of a modern industrial system. It is evident that the software industry, as a component of the modern information industry, plays a pivotal role in the entire process of economic development[2].

In light of the current landscape shaped by Artificial Intelligence and the paradigm of "software defining everything", industries related to software are poised to drive an ever-increasing demand for software-related talents. Presently, the majority of higher education institutions across the nation offer computer-related programs aimed at nurturing more IT-related professionals to meet societal needs. However, it is regrettable that both enterprises and graduates find themselves at odds. Companies lament the difficulty in recruiting suitable talents, while graduates anguish over the threat of immediate unemployment upon graduation. Consequently, numerous IT education and training institutions have emerged, such as Beida Jade Bird, ChinaSoft International, Pactera, Neutech, and other prominent educational technology companies collaborating with domestic universities to provide students with specialized practical skills training. Of course, the country is also inundated with countless

[1] School of Mathematics and Artificial Intelligence, Chongqing University of Arts and Sciences, Yongchuan, Chongqing, China
*Corresponding author: Xianping Wang

small-scale IT training institutions. These training companies primarily opt for post-graduate concentrated training programs spanning from 3 to 6 months, establishing internship bases in schools, or forging deep partnerships between schools and enterprises to enhance students' practical skills. During our visits to Beijing, Shanghai, Guangzhou, Chengdu, Chongqing, and other locations, we conducted follow-up investigations with IT-related enterprises and graduates. Our findings revealed that enterprises perceive recent graduates to lack practical skills, thus delaying their onboarding, while students commonly express concerns regarding outdated technology taught in schools, inadequate project management skills, and the realization that they learn more within a few months of company training than during their years in school. What could be causing these issues? Why do students believe that a few months of company training surpasses four years of systematic learning in school? Could it be an issue with our teaching methods? Are the students' poor practical skills indicative of flaws in our experimental (practical training) teaching? These questions are imperative for all engineering education professionals to contemplate.

The cultivation of students' hands-on skills and engineering capabilities indicates a need for reflection on the entire talent development process. Talent development primarily comprises theoretical teaching and experimental (practical) teaching. Among these, experimental teaching is an indispensable and vital component of the professional talent development system in higher education. It serves as a crucial avenue for students to effectively consolidate theoretical teachings, cultivate practical skills, and nurture innovative thinking. It provides the most direct and efficient means for students to apply theoretical knowledge to real-world problems, thereby enhancing their engineering capabilities and fostering engineering innovative thinking. Experimental teaching primarily encompasses curriculum-based laboratory experiments and comprehensive experiential training. Curriculum-based laboratory experiments utilize experimental methods to validate textbook theory effectively, serving as a teaching method to reinforce theoretical content. On the other hand, comprehensive experiential training, a significant part of experimental teaching, involves the refinement and integrated application of acquired knowledge. It serves as a means of enhancing students' overall abilities and stands as a pivotal approach within the entire practical teaching system. Currently, both Chinese and international scholars have extensively discussed experimental teaching, focusing on aspects such as the practical teaching system, teaching models, curriculum practice, quality monitoring of practical teaching, and university-enterprise collaboration. Authors such as Luodai Zhong, Lu Min, Zhang Huiyi, Wang Haiyan, Zhang Jian, and Li Mingdi mainly offer macroscopic insights into the experimental teaching system. They develop a comparably scientific, rational, and comprehensive practical teaching system, acting as the top-level design for professional practice teaching, thereby playing a pivotal role in professional development and overall talent cultivation[3-6]. On a microscopic level, authors like Guo Wei, Wu Zhinan, Tang Miao, and Huang Guangneng elaborate on how to reform experimental teaching or comprehensive experiential training. They provide detailed explanations from specific implementation methods and teaching management perspectives, offering significant insights for the concrete implementation of experimental teaching[7-10]. Authors including Zheng Dapeng, Zhang Xiaoyan, and Zhang Shenyong delve into software engineering comprehensive experiential training reform from the perspectives of curriculum integration and corporate environment11]. Xiong Wenyuan, Yang Li, among others, comprehensively discuss experimental teaching from the perspective of university-enterprise collaborative education, emphasizing the introduction of corporate resources such as course materials, project resources, and teaching staff resources. They focus more on reforming and implementing teaching models to seamlessly align with corporate practices[12,13]. Lastly, authors Li Panjie and Li Shengli address the issue of weak professional practice capabilities among university students, proposing significant improvements in experimental courses to reform the content of professional courses[14].

In conclusion, the author believes that there is a crucial issue that cannot be overlooked - the lack of research on how to design and construct a more scientific and rational experimental teaching content, especially the scarcity of studies on establishing a cohesive experimental teaching content system. Systematic experimental teaching content systems have been inadequately explored in the literature, indicating a significant gap in the entire talent development process. Therefore, focusing on experimental teaching content as the carrier of practical instruction, its quality will directly impact the overall quality of experimental teaching, consequently influencing the engineering capabilities of students in STEM fields. Researching and exploring a systematic experimental teaching content system is deemed essential. Taking software engineering as an example, this paper briefly outlines the primary issues currently present in experimental teaching content. Subsequently, it introduces the basic principles and methods of systems theory, elaborating in detail on the approach and implementation of constructing experimental teaching content in software engineering.

II. Current Primary Issues in Experimental Teaching Content

A. *Fragmented Course Experimental Projects Lacking Coherence*

Course experiments primarily serve to validate theoretical teaching content and deepen students' understanding of such theories. However, at present, the majority of university instructors in various disciplines develop experiments based on theoretical content, either through textbook-recommended experiments or by designing experiments themselves for validation. However, the contents among teachers of different courses are essentially isolated, leading to a lack of cohesion. Even when there are discussions, they often remain superficial; the experimental designs across courses lack interconnection, resulting in a lack of holistic perspective. Taking software engineering as an example, in the course Principles and Applications of Databases, the most commonly used case study nationwide is the student course enrollment system. This example is often referenced when discussing relational data theory and the SQL database language. However, when delving into database design, other examples like material management systems and human resource management systems are utilized. Additionally, when discussing concurrency control, examples such as airplane ticket booking systems are employed. In the "Operating Systems" course, examples like the dining philosophers problem are often used when discussing concurrency issues and resource allocation strategies. Moreover, in Web Programming, common system projects like online shopping or online bookstores are frequently referenced. It is evident that there is limited correlation among different courses. The database content used varies, and even within the same course, such as Principles and Applications of Databases, there is a lack of internal coherence. This lack of integration can lead students to feel that some courses are learned in vain, as they may not feel capable of practical application afterward. This perception can result in students graduating with the belief that they haven't acquired substantial knowledge at school, criticizing the education for being fragmented and seemingly useless. Some may even resort to spending significant time and resources in additional training at specialized institutions to secure employment. Where does this issue lie? It seems to stem from the scattered nature of course experimental projects, with a lack of information sharing and overall cohesion among these projects. By considering the experimental projects in a more holistic manner, focusing on one or two comprehensive software system examples, and utilizing databases consistently throughout these projects, students can gain a sense of unity in their learning experiences. Addressing this approach could lead to a significant improvement in the situation.

B. *One-sided Contents of Comprehensive Practical Training Lacking Systematical Depth*

Comprehensive experimental (practical) training serves as a platform for students to apply their acquired knowledge holistically, primarily to cultivate their engineering and innovative capabilities. Hence, it should embody a sense of systematic organization. Presently, across all universities, comprehensive practical training adopts a project-driven and case-driven approach, striving to replicate authentic enterprise environments and foster an integrated training methodology[15,16]. The majority of commonly used cases include systems like train ticket reservation systems, hotel management systems, book sales systems, and personnel archive systems - all standard information management systems employed to train students in foundational software development skills. While emphasizing the software development process, the current content of comprehensive practical training appears somewhat limited and lacks a cohesive systematization. In these information management systems, students typically engage in rudimentary tasks like adding, deleting, modifying, and querying individual data tables, thus confining the scope of their training capabilities. Therefore, it prompts questioning whether the educational curriculum, initially designed to align with the objectives of talent cultivation, adequately covers a comprehensive array of topics such as multithreading, database design and optimization, transaction processing, mainstream frameworks, as well as data analytics and statistical methodologies. This invites a discussion on the sufficiency of the curriculum in addressing these critical technical areas essential for student development.

Certainly, many universities have established collaborative programs with enterprises[13], where foundational courses are taught by university professors, while specialized technical courses and intensive practical components are led by industry engineers. These engineers often dissect their company's commercial projects and transform them into hands-on experiments for students, significantly enhancing their practical skills. However, these industry engineers are primarily familiar with their own company's projects, and their application may not necessarily align deeply with the educational objectives, positioning, and overall curriculum of the collaborating universities. Therefore, it is worth investigating whether their project content covers as many key points from the talent development plan as possible.

III. The Fundamental Idea and Approach of Systems Theory

The core idea of systems theory revolves around the concept of systemic wholeness[17]. Bertalanffy emphasized that every system is an organic whole, far from being a mere mechanical aggregation or simple summation of its parts. The holistic functionality of a system possesses qualities that individual elements lack in isolation. Drawing upon Aristotle's notion that "the whole is greater than the sum of its parts", he illustrated the systemic wholeness, rejecting the mechanistic view that good performance of elements ensures overall excellence. Furthermore, he asserted that elements within a system do not exist independently; each element holds a specific position within the system, playing a distinct role. Interconnected, these elements form an indivisible whole. These elements are integral components of the whole; if one were to sever an element from the system, it would lose its functional significance within the whole[17,18].

The fundamental idea of systems theory lies in treating the subject of study and management as a unified whole system, emphasizing systemic thinking. The primary objective of systems theory is to focus on systems as entities, starting from the whole to investigate the interrelationships between the system as a whole and its individual components. This essential examination delves into the structure, functions, behaviors, and dynamics of the system, aiming to grasp the entirety of the system and achieve optimal objectives[18]. Based on the characteristics of systems theory thinking, it elucidates the fundamental concepts of systems theory and the specific methodological principles for applying this theory to problem-solving.

A. The Principle of Wholeness

Wholeness stands as one of the most fundamental and crucial characteristics within systems theory, serving as the essence and core of the discipline. All systems are organic wholes composed of relatively independent elements that collectively manifest distinctive functionalities. When these sub-elements come together in a specific manner or adhere to certain rules, they embody the functions and attributes of the entire entity. This, of course, transcends a mere physical summation of individual element functions[4,6].

B. The Principle of the Whole Being Greater than the Sum of Its Parts

The concept that the whole is greater than the sum of its parts signifies that a system as a whole possesses qualities that its individual parts do not. The functionality of the whole is not simply a mechanical aggregation of the functions of its parts. The founder of general systems theory, Ludwig Von Bertalanffy, once highlighted: to comprehend a whole, or a system, one must not only understand its individual components but also grasp the relationships between them. The premise for the whole being greater than the sum of its parts is that the parts are interrelated and function in a synergistic manner[20].

Software engineering is the study and application of systematic, standardized, and quantifiable procedural approaches to developing and maintaining software. It is evident that software engineering itself is a systems engineering discipline, emphasizing practical engineering aspects; thus, its experimental teaching content must possess a systemic nature. As mentioned earlier, the majority of universities nowadays focus on the holistic development of talent, encompassing both theoretical and experimental teaching systems. While there are syllabi detailing the course content, these serve as mere outlines without addressing the specifics of the design. The current practice involves designing experimental content corresponding to theoretical syllabi, thereby training on the knowledge associated with each theory. Hence, the conventional belief is that all syllabi adequately cover the necessary content, fulfilling the requirements of the entire talent development program. However, I perceive that the experimental content merely represents a simplistic aggregation of elements. The interrelatedness among the various experimental contents lacks wholeness and systemic integration. Furthermore, each experimental component is largely independent or loosely correlated, failing to manifest the principle that the whole is greater than the sum of its parts.

IV. Analysis Model of Software Engineering Experimental Teaching Content from the Perspective of Systems Theory

Viewed through the lens of systems theory, software engineering itself constitutes a complex system. Its integrated experimental system represents a vast and intricate macro system, characterized by intricate structures and content. In light of this complexity, it becomes imperative to meticulously analyze the educational goals and standards of software engineering talent development programs, as well as the current knowledge requirements for positions in the software industry, in order to thoughtfully arrange experimental teaching content. Simultaneously, the Chinese Ministry of Education, in its guidance on Guiding Opinions on Promoting the Transformation of Some

Local Ordinary Undergraduate Universities into Applied Universities, emphasizes the need for innovation in cultivating applied technical skills-based talents. This involves establishing a talent development process guided by enhancing practical abilities, bolstering practical training elements, where the proportion of practical training and internship hours exceeds 30% of the total professional teaching hours. It is evident that implementing and advancing comprehensive experimental teaching content reforms in higher education institutions, particularly in transitioning applied undergraduate universities, serves as a means to enhance the cultivation of innovative applied technical talents and is indeed a necessary choice to elevate the quality of university education. Therefore, taking software engineering as an example, we have introduced the principles and methods of systems theory to construct an analysis model of software engineering experimental teaching content from a systems theory perspective. This model, as depicted in Figure 1, aims to enrich experimental content, scientifically and rationally design and construct experimental content, and truly materialize experimental teaching, effectively enhancing students' engineering capabilities and engineering literacy.

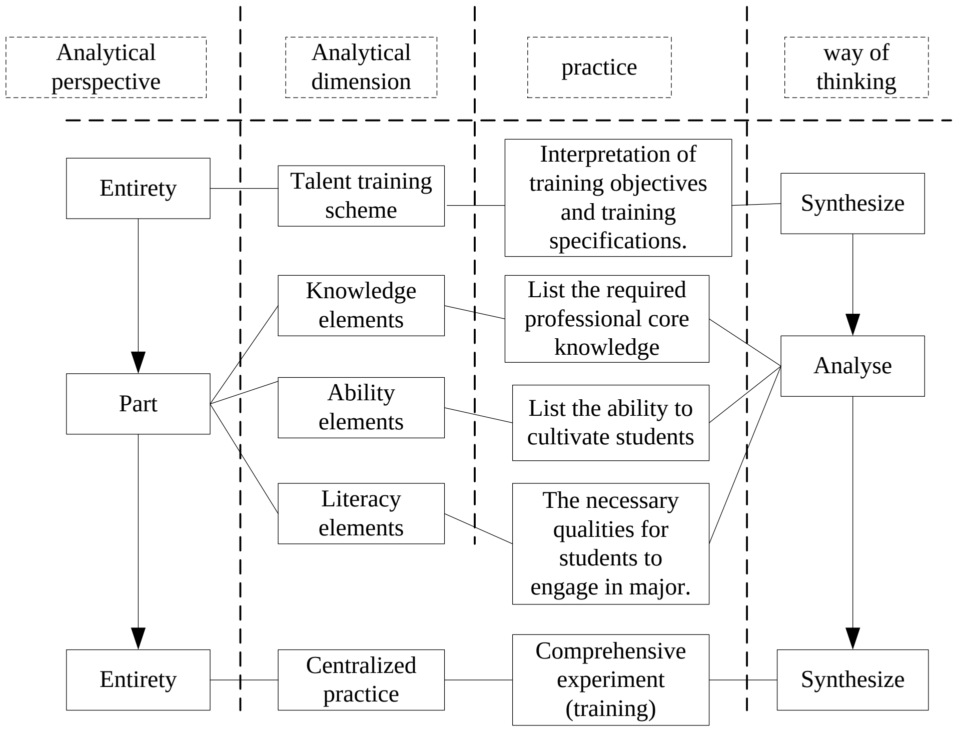
Analytical perspective
Analytical dimension
practice
way of thinking
Entirety
Talent training scheme
Interpretation of training objectives and training specifications.
Synthesize
Knowledge elements
List the required professional core knowledge
Analyse
Part
Ability elements
List the ability to cultivate students
Literacy elements
The necessary qualities for students to engage in major.
Entirety
Centralized practice
Comprehensive experiment (training)
Synthesize

Figure 1: Analysis Model of Experimental Teaching Content from the Perspective of Systems Theory

From Figure 1, it is apparent that the model delineates specific operations from four major aspects: the analytical perspective, analytical dimensions, practice, and way of thinking. Each of these aspects further elucidates specific operations from the perspectives of the wholeness, parts, and the entirety.

A. The First Perspective of Wholeness

The talent development program serves as the overarching design encompassing talent development objectives, fundamental specifications, as well as the overall process and methodology of talent cultivation. It stands as the fundamental instructional document through which schools ensure teaching quality; it is the prerequisite for the development of teaching conditions such as faculty resources and on-campus training bases; it forms the fundamental basis for organizing the teaching process, arranging teaching tasks, and determining teaching arrangements. Consequently, the formulation of a professional talent development program, refined through multifaceted research, discussions, formulation, and evaluations by program developers, constitutes a systematic undertaking that should be viewed holistically.

In the first perspective of wholeness, the analytical dimension is the professional talent development program, which serves as the foundational framework when constructing the software engineering experimental teaching content. The specific practice entails thoroughly studying the professional talent development program of software engineering, with a focus on exploring its professional orientation, training objectives, training specifications, course correlation matrices, and the entire curriculum system. It involves clarifying the relationships between courses within the curriculum system, understanding the training objectives, training specifications, and the required abilities that students should possess. The way of thinking involves comprehensive consideration of the type of individuals to be nurtured.

Taking software engineering as an example, the aim is to cultivate individuals who can meet the needs of social and economic development. These individuals should possess a solid foundation in the fundamental theories of software engineering and the mainstream technologies used in enterprise-level software development. They should be familiar with software development technology standards, exhibit strong social adaptation skills, practical software engineering capabilities, innovative thinking, entrepreneurial awareness, and be competent in engaging in software analysis, design, development, testing, and management within the software and information services industry, the Internet, and other IT-related sectors. These individuals should be high-quality specialized professionals. Following this program, all members of the research and teaching department are mobilized to collaborate on designing each course seamlessly. The core courses primarily revolve around software development and include frontend development, Java programming, database technologies, operating systems, software engineering, software testing, and project management. We have meticulously crafted a university student management system to thread through all courses, aiming to cover knowledge essential for software development as comprehensively as possible.

B. The Perspective of Parts

From a perspective of systems theory, a system is an integrated whole with specific functions formed by the interconnected arrangement of various elements. The efficacy of system functions relies on the interplay of its individual components. Therefore, when delving deep into interpreting the talent development program and the course outline, we must start by investigating each element of the talent development program[19]. The second aspect involves a perspective of parts, where the analytical dimension entails breaking down the professional talent development program's composition into knowledge elements, skill elements, and quality elements. We need to analyze these three parts, adhering to a partial mode of thinking. The specific approach will involve analyzing knowledge, capability, and qualities separately.

1) Knowledge elements: The section concerning knowledge elements forms the essence of the talent development program. It constitutes the curriculum framework of the program, encompassing theoretical, practical, and intensive practical courses. Based on this curriculum framework, teaching outlines for each course are devised. Specifically, the approach involves scrutinizing the knowledge content of each course within the curriculum framework. This includes creating diagrams that show the interrelationships between courses, detailing specific knowledge points, and clarifying the boundaries of courses and knowledge points. By mapping out these connections, we can understand how knowledge progresses through each stage of learning and its relationships before and after, thereby avoiding redundant repetition of knowledge across different courses and making the most of our valuable time.

Based on the previously designed university student management system, the various components of the system are divided into the major core courses. For instance, the front-end web design and dynamic validation of the system can be integrated into the front-end development course. The design of data tables can be incorporated into the database course. The overall project design can be included in the software engineering course. Project management and testing can be part of the software testing and project management courses, and so forth. Each instructor can use this case study for teaching, with students completing their practical exercises accordingly. There is a correlation between the courses, where the static web pages developed in front-end development serve as materials for dynamic web pages. Knowledge of databases also contributes to the system's database content, while testing in later stages reflects real project scenarios. This interconnected structure forms a complete project where each element is linked. By engaging in these core courses and practical exercises, students can successfully undertake a full-fledged project.

2) Capability elements: Building upon the foundational knowledge acquired, it is imperative to cultivate students' engineering capabilities by identifying the essential skills required and integrating them into the content of each course. This includes both exploratory experiments that complement theoretical understanding and practical training aimed at equipping students with the capacity to address initial solutions to complex engineering issues using the course materials.

Aligned with the integrated design of the university student management system, students are systematically nurtured in various coding languages, problem analysis and solving skills, as well as teamwork abilities.

3) Capability elements: In the realm of real-world demands, the call for multifaceted individuals is resounding, making a student's cultivation of qualities essential. Therefore, upon scrutinizing talent development schemes, it becomes vital to delineate the engineering qualities required. This includes traits such as craftsmanship, collaborative teamwork, dedication to duty, among others, all of which necessitate an integrated design approach.

These attributes should be structured hierarchically, with systematic implementations planned at each stage. Throughout the comprehensive implementation of the university student management system, a gradual *cultivation of team collaboration and leadership in project management capabilities has been fostered.*

C. The Second Perspective of Entirety

The second perspective of entirety emphasizes concentrated practical engagement, encompassing experiments, internships, design tasks, and integrated practical training. Its core focus lies in nurturing students' ability to apply their acquired professional knowledge towards solving real-world issues. A specific approach involves comprehensive experiments (training), particularly in specialized integrated practical training, which necessitates meticulously designed activities to integrate various perspectives. Particularly, professional integrated practical training demands meticulous planning, starting from preliminary research, needs assessment, system design, software coding and testing, to software project management. This holistic approach aims to train students in utilizing their acquired knowledge to tackle intricate engineering problems, promoting a comprehensive mindset that underscores the supremacy of the whole over mere aggregation of parts.

During the software engineering comprehensive practice phase, we engage students in practical training using another project, the Student Achievement Management System. Students are tasked with completing this project, encompassing requirements analysis, software design, coding, testing, and project management. Moreover, students are encouraged to innovate based on their own insights, integrating additional features to address tangible issues, with the ultimate goal of potentially applying the project in real-world customer scenarios.

V. ASSESSMENT OF PRACTICAL EFFECTS

Taking the 2016 cohort of Software Engineering students as an example, the university Student Management System and Student Achievement Management System serve as illustrative cases that span the entire four-year educational process. One system aids in lectures and student experiments, while the other is integral to students' final comprehensive practice. Elements of these systems are conducted both online and offline concurrently. When students encounter challenges during their learning journey, they can supplement their knowledge through the online platform. The outcomes of the education process indicate the following:

a) Conducting surveys among graduating students revealed a positive response overall. Students exhibited a strong sense of professional identity, expressing high levels of satisfaction upon completing both systems by the time of graduation.

b) Observing the employment status of graduates, there has been a notable increase in the number of students transitioning to medium to large-scale software enterprises, with corresponding competitive salaries.

c) Subsequent tracking surveys conducted with employing organizations indicated that our graduates exhibit a rapid grasp of tasks, swiftly transitioning into software development roles upon employment, thereby promptly contributing to the productivity and success of their respective organizations.

VI. CONCLUSION

Within the context of the modern era, software development technology stands as a crucial component of the computer realm. Its significance extends beyond the refinement of software applications, playing an indispensable role in people's daily lives[21]. The cultivation of software development talent has become an indispensable aspect of the new generation of the computer age. The content of software engineering laboratory teaching serves as the carrier of practical education in software engineering, playing a pivotal role in fostering the practical abilities of software engineering students. To address the shortcomings of traditional laboratory teaching content, the fundamental ideas and methods of system theory have been introduced. From the holistic perspective of talent cultivation programs to the three elements of knowledge, capability, and qualities, culminating in concentrated practical experiences, a systematic framework for software engineering laboratory teaching content has been meticulously constructed. The aim is to integrate the research subject into a comprehensive system, systematically build a functionally robust laboratory teaching content system, and facilitate experimental teaching through the carefully designed pair of software systems that traverse the entire framework. This effort not only supports the effective implementation of software engineering laboratory teaching content but also continuously enhances the engineering and innovative capabilities of professional students, significantly elevating the quality of software engineering talent development. Furthermore, as the country vigorously promotes the Computer "101 Plan", this research also serves as a valuable contribution to its advancement.

ACKNOWLEDGMENT

This work was supported by Talent introduction project of Chongqing University of Arts and Sciences ,Chongqing Higher Education Teaching Reform Research Project. (Research on Enterprise Operation Optimization Strategy Based on Big Data and Artificial Intelligence(R2023RG17),Exploration and practice of "three-chain integration" mode in characteristic demonstration software colleges from the perspective of symbiotic development).

REFERENCES

[1] Yan Shi. Trends in Computer Science Development and Their Impact on Computer Education. Computer Education, 2021, (01): 5-7.

[2] Cao Yinping. Digital Economy Driving Industrial Software to Rejuvenate “New Life”. Automation Panorama, 2023, 40(07): 3.

[3] Luo Daizhong, Wang Yuehao, Wang Weihua, et al. Exploration and practice on construction of experimental teaching center for computer engineering . Experimental Technology and Management, 2013, 30(9): 118-121.

[4] Li Mingdi, Lu Xiaoyang, Meng Lingjun, et al. Innovative Experimental Teaching System Based on System Theory . Laboratory Research and Exploration, 2012, 31(3): 119-132.

[5] Lu Min. Discussion on the Practical Teaching System of Computer Science and Technology Major. Education and Vocation, 2010, 32: 148-149.

[6] Zhang Jian, Zhang Yousheng, Wei Liangfen, et al. Research on the Reform of Practical Teaching System for Computer Major in Applied Undergraduate Colleges, 2011, 6(9): 179-180.

[7] Guo Wei. Reform of Comprehensive Training on Software Engineering Based on CDIO . Journal of Experiment Science and Technology, 2015, 12(5): 115-117.

[8] Wu Zhinan. Reform and Innovative Practice Methods of Software Engineering Experimental Teaching. Journal of Mudanjiang University, 2011, 20(7): 145-146.

[9] Tang Miao. Research on Project-Driven Experimental Teaching Mode in Software Engineering Major. Experimental Technology and Management, 2012, 29(4): 267-271.

[10] Huang Guangneng. The Quality Control and Innovation Management for Software Engineering Experiment and Practice Teaching . Computer Engineering and Science, 2011, 33(A1): 172-174.

[11] Zheng Dapeng, Zhang Xiaoyan, Zhang Shenyong. Comprehensive Software Engineering Integrated Training based on Course Integration and Simulation of Enterprise Environment. Computer Education, 2015(10): 74-77.

[12] Xiong Wenyuan, Ruan Jian, Wang Xiaoyan. Deepening the University-enterprise Cooperative Education Path Based on “Internet + Experimental Teaching” Mode . Laboratory Research and Exploration, 2017(8): 270-273.

[13] Yang Li, Liu Xiaoming. Insights into JAVA Experimental Teaching from School-Enterprise Joint Certification Training. Research and Exploration in Laboratory, 2011(7): 351-353.

[14] Li Panjie, Li Shengli. Discussion on Methods for Reforming Experimental Teaching Content of Professional Course. Gansu Science and Technology, 2021, 37(13): 65-66+131.

[15] Wang Bianqin, Liu Shuyu, Xu Haizhou, et al. Design and practice on comprehensive experiment courses for computer information major . Experimental Technology and Management, 2015, 32(4): 213-215.

[16] Tang Miao. Research on “Project-Driven” Experimental Teaching Mode in Software Engineering Major. Experimental Technology and Management, 2012, 29(4): 267-271.

[17] Ludwig Von Bertalanffy, Wang Xingcheng. History and Current Situation of General Systems Theory. Foreign Social Sciences, 1978 (2): 69-77.

[18] Sang Zhenqun, Chen Guiqiang, Zhao Li. Optimization Study of Heavy Protection Mode based on Systems Theory and PDCA Cycle. China Storage & Transport, 2024, (01): 113-114.

[19] Xiao Weiping. Analysis on the Optimization of China's Higher Education Arrangement Structure from the View of System Theory . Modern Educational Science, 2011 (2): 89-92.

[20] Wang Junsheng. Construction of Geographic Textbook Analysis Model and Element Analysis from the Perspective of Systems Theory. Teacher Education Forum, 2015, 28(10): 49-52.

[21] Zhang Yu*. The Application and Development Trend of Computer Software Development Technology in the New Era . International Progress in Computer Science, 2022, 2(3).