HiLL PUBLISHING

Advances in Computer and Communication, 2024, 5(1), 20-25
https://www.hillpublisher.com/journals/acc/
ISSN Online: 2767-2875

Analysis of Computer Application Software Automation Development Technology

Tongwei Xie[1,*], Yanchang Song[2]

[1]Kyungil University, Gyeongsan, Gyeongsangbuk-do, South Korea.
[2]Zhengzhou No.4 Senior High School, Zhengzhou, Henan, China.

How to cite this paper: Tongwei Xie, Yanchang Song. (2024) Analysis of Computer Application Software Automation Development Technology. *Advances in Computer and Communication*, 5(1), 20-25.
DOI: 10.26855/acc.2024.02.003

Received: February 16, 2024
Accepted: March 13, 2024
Published: April 8, 2024

[*]**Corresponding author:** Tongwei Xie, Kyungil University, Gyeongsan, Gyeongsangbuk-do, South Korea.

Abstract

Software development is the process of constructing an entire software system or its components tailored to the users' needs. As a virtual platform controllable by humans, computer application software proficiently executes a variety of intricate tasks, thereby enhancing work efficiency and quality. With the ongoing advancement in automatic assembly technology, the development of application software is gradually transitioning from traditional modes to automated processes. This transformation not only significantly reduces labor costs in software generation efficiency but also caters to the individual needs of users. Particularly, with the rapid evolution of Artificial Intelligence (AI), its integration into the realm of software development can effectively reshape the development process, lower the professional threshold, and enable widespread participation in the software development process. In this research, software developers should proactively adapt to this new environment. On the one hand, attention should be directed towards the development and application of automation technology, channeling more efforts into areas less susceptible to automation. On the other hand, there is a need for a shift from a singular program or platform development model to a collaborative approach where multiple software systems seamlessly integrate. In consideration of these factors and drawing on practical insights, this paper delves into the application of automatic development technology in the domain of computer application software.

Keywords

Computer, Application software, Automation development technology

1. Research Purpose

This research aims to comprehensively investigate automated development techniques in computer application software, specifically focusing on processes, principles, automation methods, development platforms, artificial intelligence applications, and the use of the integrated test automation framework. The objective is to understand their impact on reducing development cycles, testing, and maintenance, and through empirical research, validate their practical value in enhancing software functionality for substantive recommendations and guidance.

2. Definition of Automation Development Technology

Automated development technology involves the use of advanced computer technology and tools to automate repetitive and mechanical tasks in the software development process, aiming to enhance the efficiency and quality of software development. Its essence lies in significantly increasing software development efficiency by minimizing manual intervention and reducing the proportion of manually written code. Furthermore, automated development technology positively influences software maintainability, testability, scalability, and reusability, facilitating the

organized management and maintenance of software. This technology also effectively diminishes the occurrence of code errors and vulnerabilities, thereby enhancing the overall quality and security of the software [1].

3. Literature Review

3.1 Current Methods of Computer Application Software Development

In the current field of computer application software development, adopting different development methods is crucial to the overall success of the project. This research will comprehensively analyze three aspects: the life cycle method, the original development technology, and automatic system development technology.

(1) Application Analysis of the Life Cycle Method: As a widely used development method of computer application software, the life cycle method emphasizes orderly and clear guidance throughout the entire process of software development. When software developers adopt the life cycle method, they need to outline the content of the software as if writing an article and analyzing it in detail. This helps keep thinking clear during the actual software development process, making the development work more orderly. However, this method is relatively time-consuming and expensive and is often used in large computer software development projects. With the increasing demand for the performance and technical level of computer application software, the research of life cycle development technology and the reasonable division of time become particularly important. This method plays a good role in problem detection and solution strategy formulation, ensuring the quality and normal operation of software development.

(2) Application Analysis of the Original Development Technology: The original development technology focuses on customer needs, facilitating software testing and quality control. Designers iteratively adjust and update the software based on customer feedback, ensuring satisfaction. After development, vulnerabilities are detected and repaired, forming an interactive cycle with customers that enhances software efficiency and promotes automation development technology improvement [2].

(3) Application Analysis of Automatic System Development Technology: Building on the original development technology, automatic system development is innovative and refined. Designers, after understanding customer needs, analyze requirements and execute scientific automation programming. This maximizes the automation function of computer application software, enhancing development quality and efficiency. Monitoring software running status is integral, enabling timely fault location and resolution, and ensuring optimal software performance. In summary, the choice of computer application software development methods depends on project size, requirements, and complexity. The life cycle method suits large-scale projects, original development emphasizes customer interaction, and automatic system development enhances efficiency and maintenance convenience. Method selection involves trade-offs, considering the specific project context.

3.2 Application of Artificial Intelligence in Computer Application Software Development

With the rapid development of information technology and the ongoing digital transformation of society, the demand for computer application software is increasing, and the requirements for software development are progressively rising. The traditional approach to software development is confronted with numerous challenges, including a prolonged development cycle, high costs, and difficulties in addressing complex requirement changes. In this context, AI has emerged as a new technology that gradually improves the software development process, enhances development efficiency, and serves as a crucial means for optimizing software performance.

AI holds significant potential in computer application software development by leveraging machine learning and deep learning to handle large-scale data, identify patterns, and enhance intelligent decision-making. It optimizes user interface design and improves overall user experience by automatically analyzing user behavior during development. In software testing, AI-based tools enhance efficiency, ensuring stability under various conditions. Additionally, AI contributes to project management, requirements analysis, and code generation, utilizing natural language processing for better user requirement understanding. Despite its promising applications, challenges such as data privacy, algorithmic interpretability, and human-machine cooperation need addressing, emphasizing the importance of in-depth research for advancing software development both theoretically and practically.

4. Processes and Principles of Computer Application Software Development

Following an exhaustive literature review on computer application software development technology, this study

investigates specific processes and key principles to uncover pivotal nodes and technical challenges. Emphasizing the elucidation of development principles, the research aims to provide theoretical guidance for software engineering practice, fostering improved collaboration with developers and optimizing iterative processes. Strict adherence to fundamental software engineering principles ensures the final product aligns with user needs, demonstrating commendable maintainability and scalability.

4.1 Software Development Process

The development of computer application software adheres to clearly defined steps to ensure orderly project progression:

(1) Planning: Demonstrating project feasibility by understanding the realistic environment and user needs, and initiating preliminary development task planning.

(2) Analysis: Crystallizing a clear vision of the overall software development process, expressing user needs accurately through professional development language.

(3) Design: Divided into outline and detailed design stages, emphasizing overall structure and fine design aspects such as program logic, algorithms, and data structures.

(4) Coding: The core link in software development, transforming the design scheme into a computer-recognizable program.

(5) Testing: Identifying and rectifying potential errors in the software development process, reducing trial-and-error costs, enhancing functionality, and optimizing the final user experience [3].

4.2 Software Development Principles

In various development methods (e.g., waterfall, spiral, iterative), adherence to core principles is essential:

(1) Normative: Adhering to objective laws and technical standards, while balancing user needs and habits, ensures the practical utilization of the application software.

(2) Simplicity: Maintaining a concise structure and function, while meeting user needs, enhances efficiency, reduces errors, and facilitates future maintenance, expansion, and upgrading.

(3) Easy Maintenance: Designing interfaces and incorporating preset maintenance methods during application software development enables seamless integration and minimizes maintenance difficulties.

Efficient progress through each stage of the development process, considering user needs, ease of use, and future maintenance, ensures exemplary practical results in the final application software.

5. Application of Integrated Test Automation Framework in Computer Application Software Testing

This study investigates the impact of the integrated test automation framework on computer application software testing, examining its potential to enhance test efficiency, reduce costs, and optimize component collaboration. A comparative analysis is conducted between manual testing and automated testing, specifically evaluating ITAF and TAF (Test Automation Framework). The goal is to gain insights into the automation process in application software testing, offering robust support for future automation development.

5.1 Comparison between ITAF and Manual Testing

In comparing ITAF and manual testing, this study employed a computer application software case study to assess their practical performance. Findings indicate that ITAF significantly reduces automated testing costs, demonstrating advantages in project development cycles and efficiency. Firstly, despite not achieving full coverage, ITAF minimizes labor costs, as evidenced by an 8-hour automated test execution compared to 480 hours for professional technicians during the development phase. This reduces human testing costs and accelerates overall project development. Secondly, the time investment comparison highlights ITAF's efficiency in test cycle reduction, crucial in fast-paced software development environments for swift issue identification and rectification, enhancing product release speed and quality.

5.2 Comparison between ITAF and TAF

In the comparison between ITAF and TAF, this study further explores the similarities and differences in the

functionality and performance of the two automation test frameworks. This comparison not only encompasses the scope of technical support but also examines the efficiency and convenience of actual test projects.

Firstly, concerning functionality, ITAF outperforms TAF. ITAF supports both API testing and the framework's own unit testing, providing more comprehensive support for testing efforts. This allows the test team to cover all test levels comprehensively, thereby enhancing the overall quality of testing. In contrast, TAF has relatively limited support in these areas, potentially restricting the test team's ability to achieve full system coverage.

Secondly, in actual test projects, ITAF demonstrates higher efficiency and greater convenience than TAF. Specifically, for UI smoke testing, ITAF takes only 1 hour, while TAF takes 6 hours; in end-to-end regression testing, ITAF takes only 2 hours, while TAF cannot be completed due to a lack of support for automated testing. In API smoke testing and regression testing, ITAF takes only 0.1 and 5 hours, respectively, while TAF takes 1 and 80 hours, respectively. These comparison results clearly indicate that ITAF has a significant advantage over TAF in terms of test efficiency.

Finally, the overall performance and efficiency of ITAF make it a more comprehensive and efficient test tool. By supporting a broader range of tests, ITAF provides test teams with a more flexible and efficient automated testing framework. This flexibility is crucial for addressing complex test scenarios and rapidly changing requirements [4].

Through an in-depth comparison of ITAF versus manual testing and ITAF versus TAF, this study offers a more comprehensive understanding of the advantages and disadvantages of these two approaches and frameworks. This serves as a powerful guide for testing teams when choosing the appropriate testing methodology and framework for their projects.

6. The Application of AI in the Construction of Computer Application Software Automation Development Platform

6.1 Application of AI in Business Component Generator

The business component generator, a core element of the automated development platform, rapidly creates commonly used business components, enhancing development efficiency and reducing complexity. The integration of AI in the business component generator focuses on the following aspects:

(1) Database Management

AI optimizes database management through intelligent algorithms and data analysis, identifying and analyzing developers' and users' data needs. This results in smarter and more efficient database design solutions, reducing data redundancy, and enhancing query efficiency, thus expediting business component generation.

(2) Processing System

In a business component generation, the processing system is a crucial implementation link. AI automates business logic processing by learning and optimizing algorithms. Drawing insights from developers' past code and execution data, AI generates more efficient and maintainable processing regimes, easing developer workload and enhancing generator performance.

(3) Expansion Mechanism

The business component generator's flexibility lies in its expansion mechanism, adaptable to varying business needs. AI predicts potential business expansion directions by analyzing market trends, user feedback, and other data, providing corresponding expansion mechanisms. This adaptability enables the generator to better navigate the dynamic business environment.

(4) Graphical Input Interface

AI enhances the user experience in the graphical input interface through technologies like natural language processing and image recognition. This allows generators to better understand user needs and create intelligent user interfaces. Intelligent design of the graphical input interface reduces user learning costs and enhances satisfaction.

6.2 Intelligent Design of AI in Business Component Design Workflow

The business component design processor, overseeing business processes in software development, aims for intelligent design to enhance process accuracy and efficiency. Key AI applications in this processor include:

(1) Application Library

AI facilitates intelligent application library management by analyzing developer usage habits and project requirements. The system can automatically recommend or generate applications aligning with current business needs,

expediting process design and minimizing repetitive tasks.

(2) Management Process

AI optimizes the management process through process optimization and real-time monitoring. Learning from historical data, AI provides intelligent optimization suggestions, making the process more realistic. Real-time monitoring identifies potential problems, offering timely solutions to enhance process accuracy and reliability.

(3) Graphical Business Process Design Interface

AI enhances the graphical business process design interface by providing intelligent design tools. Automatically identifying and correcting design errors and offering design recommendations, AI assists developers in designing business processes more efficiently, reducing complexity, and improving design quality.

6.3 Application of AI in Computer Application Software Development

As a frontier in computer science, AI profoundly impacts application software development by enhancing technical feasibility and reducing maintenance difficulty. This study explores innovative applications of neural networks (NNs), expert systems, artificial immunity, and agent technology in computer application software development.

(1) Neural Networks

Neural networks, large-scale parallel distributed processors, exhibit potential in automation development. Resembling the human brain in information processing, NNs enhance work efficiency and information security in application software development. Applications like circular networks and intrusion detection improve risk identification speed and accuracy [5].

(2) Expert Systems

Expert systems, integrating expert-level knowledge for scientific decision-making, focus on misuse analysis in application software automation. Real-time monitoring enhances intrusion detection, ensuring stable software operation. Examples, such as NIDIS technology, employ new statistical methods for abnormal operation checks and feature-rich intrusion scenario coding and analysis models.

(3) Artificial Immunity

Artificial immunity technology, central to virus recognition and antivirus measures, strengthens identification and antivirus functions for secure software development. Through processes like negative selection and gene bank utilization, artificial immunity detects intrusion behavior, enhancing the ability to identify computer viruses and ensuring software protection performance.

(4) Agent Technology

Agent technology, addressing distributed application problems, senses software environments, enabling automatic task execution and environment perception. Widely applied in software security defense, it optimizes process design and enhances software environment awareness. Agent technology, integrated through plug-ins, further improves security protection functions and monitors various software operations in the automatic development of application software [6].

7. Conclusion

Automated development technology has significantly enhanced software development efficiency and quality by automating code generation, test cases, and documentation, resulting in reduced costs and time. While presenting challenges such as potential knowledge diminishment and limited flexibility, the evolving landscape of artificial intelligence and machine learning technologies, alongside the emergence of low-code/no-code development, foretells a future where automated development continues to dominate and drive the software industry forward, requiring a careful balance of strengths and weaknesses for sustainable prosperity.

References

[1] Sarker, I. H. (2022). Ai-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems. SN Computer Science, 3(2), 158.

[2] Ribeiro, J., Lima, R., Eckhardt, T., & Paiva, S. (2021). Robotic process automation and artificial intelligence in industry 4.0–a literature review. Procedia Computer Science, 181, 51-58.

[3] Anthony, A. R. V., Prasad, G. D., Randunuge, S. U., Alahakoon, S. R. A. M. P. A., Wijendra, D. R., & Krishara, J. (2020). Software

Development Automation: An Approach to Automate the Processes of SDLC. International Journal of Computer Applications, 875-887.

[4] Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100-308.

[5] Javaid, M., Haleem, A., Singh, R. P., Khan, S., & Suman, R. (2021). Blockchain technology applications for Industry 4.0: A literature-based review. Blockchain: Research and Applications, 2(4), 100-127.

[6] Kunduru, A. R. (2023). Cloud BPM Application (Appian) Robotic Process Automation Capabilities. Asian Journal of Research in Computer Science, 16(3), 267-280.