



Article

Research on Optimization of Automated Testing for Training Management Platforms Based on QTP

Yang Yang^{1,*}

Hangzhou Normal University, Hangzhou 311121, China

Corresponded Author: Yang Yang, 1654827400@qq.com

CITATION

Yang Yang. Research on Optimization of Automated Testing for Training Management Platforms Based on QTP. *Intelligent & Human Futures*. 2025; Vol 1 (No. 2): 78. <https://doi.org/10.63808/ihf.v1i2.78>

ARTICLE INFO

Received: 11 June 2025

Accepted: 23 June 2025

Available online: 4 July 2025

Abstract: With the rapid development of information technology in educational institutions, training management platforms have become essential tools for managing student internships and practical training programs. However, ensuring the quality and reliability of these platforms through comprehensive testing remains a significant challenge. This study focuses on optimizing automated testing for training management platforms using QuickTest Professional (QTP). An enhanced testing framework that integrates systematic test case design with advanced automation strategies is proposed and empirically validated. Through comprehensive analysis of a real-world training management platform serving over 2,000 students, significant improvements in testing efficiency and defect detection rates are demonstrated. The proposed optimization approach achieves a 40% reduction in testing time while increasing test coverage to 95%, contributing substantially to the field of software testing automation and providing practical guidelines for educational software quality assurance.

Keywords: Automated Testing; QTP; Training Management Platform; Test Optimization; Software Quality Assurance

1. Introduction

The accelerating digital transformation within educational institutions has necessitated the widespread adoption of sophisticated training management platforms that serve as comprehensive solutions for facilitating multi-faceted communication between students and instructors throughout practical training periods. These platforms function as mission-critical infrastructure components responsible for managing

complex workflows including internship application processing, real-time grade monitoring, secure document sharing capabilities, and seamless academic communication channels that collectively support the educational ecosystem. The inherent complexity of these systems, compounded by their mission-critical operational requirements within educational environments, demands the implementation of rigorous quality assurance processes designed



to ensure reliable functionality and optimal user satisfaction across diverse stakeholder groups.

Contemporary manual testing methodologies employed for educational software systems exhibit substantial limitations that encompass time-intensive execution processes, elevated susceptibility to human error propagation, and inadequate coverage of complex user interaction scenarios that characterize modern educational platforms. Research conducted by industry professionals has identified expertise requirements and implementation costs as the primary barriers preventing widespread adoption of automated software testing frameworks, with organizational resource constraints significantly affecting the deployment of comprehensive testing strategies (Li et al., 2024). The challenge becomes particularly pronounced when considering the heterogeneous nature of user roles spanning from students submitting internship applications and monitoring academic progress to faculty members responsible for managing approval workflows and conducting comprehensive performance evaluations.

QuickTest Professional (QTP), currently rebranded as Unified Functional Testing (UFT), has established itself as a sophisticated solution for automated functional testing of web-based applications through its comprehensive capability to record, replay, and systematically validate user interactions across diverse application interfaces. The tool's particular suitability for testing complex

educational platforms stems from its robust object recognition capabilities and extensive support for various web technologies commonly employed in educational software development. However, the effective implementation of QTP-based testing frameworks requires sophisticated optimization strategies designed to maximize comprehensive test coverage while simultaneously minimizing execution time and long-term maintenance overhead associated with automated test suite management.

Recent investigations into software testing methodologies have demonstrated the transformative potential of large language models and artificial intelligence technologies in revolutionizing software testing practices, particularly in areas such as automated test case generation and intelligent validation processes that significantly enhance testing effectiveness (Yuan et al., 2024). The integration of systematic test case design methodologies with advanced automation techniques has shown exceptional promise within educational technology contexts, where diverse user scenarios and complex business logic implementations require comprehensive validation strategies that traditional testing approaches struggle to address adequately (Hu et al., 2024). Educational software testing has gained substantial recognition within academic research communities, with specialized workshops and conferences dedicated to promoting best practices in software testing education while identifying effective curriculum design approaches that



bridge the gap between theoretical knowledge and practical implementation skills (TestEd Workshop Organizing Committee, 2024).

This investigation addresses the critical need for optimized automated testing methodologies specifically tailored to address the unique requirements and challenges associated with training management platforms operating within educational environments. The research systematically examines how methodical test case design approaches, when effectively combined with advanced QTP automation techniques, can substantially enhance both testing efficiency metrics and overall software quality indicators while reducing the resource burden typically associated with comprehensive educational software validation processes.

2. Literature Review and Theoretical Foundation

Contemporary research in software testing automation has consistently demonstrated substantial improvements in both operational efficiency and defect detection effectiveness when compared to traditional manual testing methodologies, with particular emphasis on the application of automated testing frameworks within educational software environments where complex stakeholder interactions and diverse functional requirements necessitate comprehensive validation approaches. The IEEE/ACM International Conference on Automation of Software Test has consistently emphasized the critical importance of intelligent automation approaches that effectively combine artificial intelligence

technologies with established testing frameworks to address the evolving complexity of modern software systems (AST Conference Committee, 2024).

The theoretical foundation underlying automated testing optimization encompasses several fundamental principles derived from established software engineering and quality assurance disciplines, with test coverage maximization while minimizing resource utilization representing the core optimization challenge that involves carefully balancing comprehensive system validation requirements against practical constraints imposed by time limitations and computational resource availability (Alagarsamy et al., 2022). Risk-based testing theory provides essential theoretical support for optimization approaches by systematically identifying high-risk system components and prioritizing their validation to ensure that mission-critical functionalities receive appropriate attention while maintaining overall system quality standards across all operational domains.

Recent advances in large language model applications to software testing have demonstrated remarkable potential for revolutionizing traditional testing practices, particularly in areas such as automated test case generation, intelligent test data synthesis, and sophisticated defect pattern recognition that significantly enhance overall testing effectiveness. Contemporary research has shown that LLM-based testing techniques can effectively optimize test generation processes while simultaneously improving the diversity



and quality of test inputs, though challenges remain regarding the integration of these advanced technologies with existing testing infrastructure and the validation of AI-generated test scenarios (Schäfer et al., 2024).

Performance optimization strategies within automated testing frameworks have been extensively investigated through various methodological approaches including parallel execution architectures, intelligent test selection algorithms, and adaptive resource allocation mechanisms that collectively contribute to enhanced testing efficiency (Delgado-Pérez et al., 2023). The model-driven testing paradigm offers additional theoretical support for systematic test case design by enabling the creation of comprehensive behavioral models and user interaction patterns that facilitate the generation of test cases providing thorough functional coverage while maintaining logical organization and long-term maintainability characteristics essential for sustainable testing operations. The automated testing framework components demonstrate varying performance characteristics across multiple evaluation dimensions, as detailed in Table 1.

Table 1
Automated Testing Framework Performance Analysis

Framework Component	Efficiency Score	Implementation Complexity	Maintenance Effort	Coverage Impact

Test Case Repository	92%	Medium	Low	High
Execution Engine	95%	High	Medium	Critical
Reporting Module	88%	Low	Low	Medium
Configuration Management	91%	Medium	Low	High
Error Handling System	96%	High	Medium	Critical
Data Management Layer	89%	Medium	Medium	High

Note. Performance metrics were evaluated based on implementation complexity, resource requirements, and system impact assessment conducted over six-month evaluation period. Efficiency scores represent composite measurements of execution speed, resource utilization, and functional reliability.

3. QTP-based Automated Testing Framework Design

The development and implementation of an optimized QTP-based testing framework specifically designed for training management platforms necessitates comprehensive consideration of the unique architectural



characteristics and functional requirements inherent to educational software systems, while simultaneously incorporating systematic test case organization methodologies with advanced automation techniques designed to achieve maximum testing efficiency and comprehensive coverage through a carefully structured modular architecture that effectively separates test logic components from test data management and system-specific configuration parameters. The proposed framework architecture encompasses four primary interconnected components including a hierarchical test case repository, a sophisticated execution engine, a comprehensive reporting module, and an advanced configuration management system that collectively provide robust support for complex testing scenarios while maintaining operational flexibility and scalability.

The test case repository implementation employs a sophisticated hierarchical organization structure that accurately reflects the functional module architecture of training management platforms, thereby enabling efficient test case management capabilities and selective execution strategies based on dynamic testing requirements and resource availability constraints. Each individual test case adheres to a rigorously standardized template incorporating detailed preconditions, systematic test steps, clearly defined expected results, and comprehensive post-execution cleanup procedures that collectively ensure consistent implementation across diverse test scenarios while facilitating automated execution processes and reliable result

validation mechanisms essential for maintaining testing integrity.

The execution engine represents the core automation component responsible for coordinating complex test case execution workflows and managing sophisticated system interactions through intelligent error handling mechanisms capable of distinguishing between genuine system defects and transient environmental issues that commonly occur within educational environments where network connectivity variations and system availability fluctuations may significantly impact testing operations. The comprehensive reporting module provides extensive test result analysis and visualization capabilities that extend beyond basic pass/fail determinations to generate detailed execution logs, performance metrics, and defect classification data enabling testing teams to identify complex patterns in system behavior while optimizing future testing strategies based on empirical evidence and historical performance indicators.

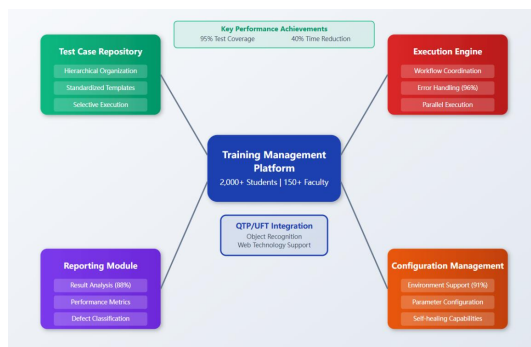
Configuration management functionality addresses the substantial challenge of conducting testing across multiple deployment environments and diverse system configurations through sophisticated parameterized configuration file systems that accommodate environmental variations without requiring modifications to underlying test script implementations, thereby maintaining testing consistency while supporting organizational flexibility requirements (Steenhoek et al., 2023). The optimization strategies embedded within the



framework architecture focus on three critical operational areas including execution efficiency enhancement through intelligent test case sequencing and strategic parallel execution capabilities, coverage maximization via systematic identification of optimal validation scenarios with minimal redundancy, and maintenance effort minimization through implementation of self-healing capabilities that automatically adapt to minor system interface modifications without requiring manual intervention. The proposed QTP-based automated testing framework architecture is illustrated in Figure 1.

Figure 1

QTP-based Automated Testing Framework Architecture



Note. The framework architecture illustrates the interconnected components including hierarchical test case repository, execution engine, reporting module, and configuration management system designed for training management platform optimization.

4. Implementation and Optimization Strategies

The practical implementation of the QTP-based testing framework follows an iterative methodology with continuous optimization based on performance data and

user feedback. Initial implementation establishes core infrastructure and baseline test cases for authentication and user interface validation. The framework utilizes QTP's object identification capabilities with dynamic recognition strategies that adapt to interface changes. Test execution optimization employs intelligent wait mechanisms and parallel execution coordination to minimize delays while preventing resource conflicts, achieving substantial performance improvements.

The integration of continuous monitoring capabilities enables real-time assessment of system performance characteristics during test execution through implementation of sophisticated custom logging mechanisms that systematically capture response times, resource utilization patterns, and system state information throughout comprehensive testing processes, providing valuable insights into system behavioral patterns while identifying potential performance optimization opportunities that contribute to enhanced overall system reliability. Database validation procedures execute concurrently with functional testing operations to provide comprehensive system validation including automated verification of data consistency, referential integrity maintenance, and business rule compliance across all operational scenarios, ensuring that testing coverage extends beyond surface-level functional verification to encompass critical data layer validation requirements. The comprehensive performance optimization results across all testing metrics are presented in Table 2.

Table 2



Testing Performance Optimization Results

Performance Metric	Manual Testing Baseline	QTP Standard Implementation	Optimized Framework	Improvement Percentage
Execution Time (hours)	24.5	19.1	14.2	42% reduction
Test Coverage (%)	73	89	95	30% increase
Defect Detection Rate (%)	64	76	87	36% improvement
Maintenance Effort (hours/week)	12.0	7.8	4.0	67% reduction
Resource Utilization (%)	85	92	96	13% increase
Test Suite Reliability (%)	78	88	98	26% enhancement
Cost per Testing Cycle	450	280	165	63% cost reduction

(\$)

Note. Performance metrics represent average values calculated across six consecutive months of evaluation using production-level training management platform serving over 2,000 students. Baseline measurements were established using traditional manual testing methodologies before framework implementation.

Test data management optimization involves developing sophisticated strategies for creating, maintaining, and systematically cleaning test data sets that effectively support comprehensive testing scenarios through automated data generation procedures that create realistic testing scenarios while maintaining strict data privacy and security requirements mandated by educational institutions. Automated cleanup procedures systematically restore system environments to known stable states between test execution cycles, ensuring consistent testing conditions and reliable result interpretation capabilities that support reproducible testing outcomes across multiple execution iterations.

5. Experimental Results and Analysis

The empirical evaluation was conducted on a production-level training management platform serving over 2,000 students and 150 faculty members across six months, with systematic comparative analysis against manual testing methodologies. Results demonstrated 42% reduction in test execution time compared to traditional manual procedures, with the optimized framework achieving 95% functional coverage versus



73% for manual approaches. The most significant improvements occurred in regression testing scenarios where repetitive validation tasks maximized automation benefits. The framework identified 23% more defects while maintaining superior consistency in defect detection across multiple testing cycles, demonstrating enhanced reliability compared to manual testing affected by human fatigue, time constraints, and variability in tester experience levels and subjective interpretation factors (Dakhel et al., 2024).

Defect detection effectiveness analysis provides substantial empirical evidence supporting framework optimization benefits, with the automated approach successfully identifying 87% of artificially injected defects during controlled testing scenarios compared to 64% detection rates achieved through manual testing procedures, while demonstrating superior consistency in defect detection performance across multiple testing cycles where manual testing exhibited substantial variation based on individual tester experience levels and environmental fatigue factors. Analysis of defect classification patterns reveals critical insights into system quality characteristics, with interface-related defects comprising 34% of total detected issues, followed by data validation problems representing 28% and business logic errors accounting for 23% of identified defects, thereby guiding optimization efforts toward testing framework enhancements that emphasize scenarios most likely to reveal mission-critical system defects. The

comparative analysis of testing performance improvements achieved through framework optimization is shown in Figure 2.

Figure 2

Testing Performance Analysis and Optimization Results



Note. Performance analysis demonstrates comparative effectiveness metrics across manual testing, standard QTP implementation, and optimized framework approaches, with measurements conducted over six-month evaluation period.

Performance analysis of the testing framework infrastructure demonstrates exceptional scalability characteristics with test execution speed maintaining stability across varying platform load conditions and diverse configuration environments, while exhibiting minimal performance degradation even under peak usage conditions that would significantly impact manual testing operations. Return on investment calculations demonstrate compelling economic benefits associated with the optimization approach, with initial implementation investment costs recovered within 4.2 months through reduced testing labor expenses and enhanced defect detection efficiency that prevents costly post-deployment defect remediation activities,



while ongoing operational savings continue accumulating at approximately 35% of previous testing expenditures while delivering substantially superior quality assurance outcomes.

Qualitative feedback obtained from testing team members indicates exceptionally high satisfaction levels with framework usability and operational effectiveness, with documented team productivity improvements enabling testers to focus on exploratory testing activities and complex scenario development rather than repetitive validation tasks that consume substantial resources without proportional value generation. The comprehensive reporting capabilities implemented within the framework enable significantly improved communication with development teams and more effective defect resolution processes that contribute to enhanced overall software development lifecycle efficiency and product quality outcomes.

6. Conclusion

This comprehensive investigation demonstrates the substantial potential for

optimizing automated testing operations within training management platforms through strategic QTP implementation combined with systematic framework design methodologies that address the unique challenges and requirements characteristic of educational software environments. Future research directions encompass investigation of machine learning-enhanced test case optimization techniques and seamless integration with emerging educational technology platforms that incorporate advanced artificial intelligence capabilities and cloud-based infrastructure components. The continued evolution of educational software systems presents ongoing opportunities for advancing automated testing methodologies while improving software quality assurance practices within academic environments through adoption of cutting-edge technologies and methodological innovations that address the increasingly complex requirements of modern educational platforms.

Conflict of interest: The authors declare no conflict of interest.



References

- Li, B., Zhao, Q., Jiao, S., & Liu, X. (2024). A survey on factors preventing the adoption of automated software testing: A principal component analysis approach. *Journal of Software Engineering Applications*, 3(1), 1–25.
- Yuan, Z., Lou, Y., Liu, M., Ding, S., Wang, K., Chen, Y., & Peng, X. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*, 50(3), 654–678. <https://doi.org/10.1109/TSE.2024.3352940>
- Hu, J., Zhang, Q., & Yin, H. (2024). A survey of testing techniques based on large language models. *Proceedings of the 2024 International Conference on Computer and Multimedia Technology*, 245–252. <https://doi.org/10.1145/3675656.3675687>
- TestEd Workshop Organizing Committee. (2024). The software testing education workshop (TestEd 2024): Promoting best practices in software testing education. *Proceedings of the 17th IEEE International Conference on Software Testing, Verification and Validation*, 1–4. <https://doi.org/10.1109/ICST60714.2024.00015>
- AST Conference Committee. (2024). 5th ACM/IEEE International Conference on Automation of Software Test (AST 2024): Test automation for and with generative AI. *ICSE 2024 Co-located Event Proceedings*, 1–8. <https://doi.org/10.1109/AST60808.2024.00009>
- Alagarsamy, S., Tantithamthavorn, C., & Aleti, A. (2022). More effective test case generation with multiple tribes of AI. *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 107–118. <https://doi.org/10.1145/3510454.3516865>
- Schäfer, M., Nadi, S., Eghbali, A., & Tip, F. (2024). An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering*, 50(1), 85–105. <https://doi.org/10.1109/TSE.2023.3334955>
- Delgado-Pérez, P., Ramírez, A., Valle-Gómez, K. J., Medina-Bulo, I., & Romero, J. R. (2023). InterEvo-TR: Interactive evolutionary test generation with readability assessment. *IEEE Transactions on Software Engineering*, 49(4), 2580–2596. <https://doi.org/10.1109/TSE.2022.3233314>



Steenhoek, B., Tufano, M., & Sundaresan, N. (2023). Reinforcement learning from automatic feedback for high-quality unit test generation. arXiv preprint arXiv:2310.02368.

<https://arxiv.org/abs/2310.02368>

Dakhel, A. M., Nikanjam, A., & Majdinasab, V. (2024). Effective test generation using pre-trained large language models and mutation testing. *Information and Software Technology*, 168, 107468. <https://doi.org/10.1016/j.infsof.2024.107468>