Research on Quality Assurance Methods for Software Products Based on Low Code Development*

Wenyuan Zhang[1,*,†], Jianxun Guo[2,†], Yangyang Zhang[3,*,†], Wenpeng Li[*,†]

[1] *(China Electronics Standardization Institute, Software Engineering and Evaluation Center, Beijing 100007)*
[2] *University of Twente, Drienerlolaan 5, 7522 NB, Enschede, The Netherlands*
[3] *University of Skövde, Högskolevägen 1, 541 28 Skövde, Sweden*

Abstract
With the acceleration of digital transformation, low-code (LC) has become an important trend in the field of software development. Products such as LC development tools and platforms provide users with graphical and visual programming interfaces, enabling non-professional developers to quickly build applications and significantly improve development efficiency, function suitability and etc.. However, the impact of LC on software engineering quality is a complex issue that involves multiple aspects such as development processes, testing strategies, selection and integration of automation tools, and performance testing standards. This paper aims to explore the impact of LC on software product quality and study a software quality characteristic framework for LC development to ensure reliability, performance, understandability and etc. when using LC.

Keywords
Low-Code, software engineering, quality characteristics

1. Introduction

LC is an emerging technological paradigm in the current field of software engineering. The practical approach of this technology is to use pre made code modules for programming, reducing manual coding to accelerate the software development process, thereby enabling nonprofessional developers to participate in the software development process. Since the 1980s, the process of LC transitioning from conceptual technology to thriving development has become one of the important drivers in today's wave of digital transformation.

1.1. Initial exploration (1980-2000s)

The origin of LC can be traced back to the 1980s, when early graphical programming tools and fourth generation programming languages (4GL) appeared on the market. Engineers attempted to simplify software development processes and code writing through these tools, making development work more intuitive and understandable. However, limited by the computing power and network environment at that time, the actual effectiveness of these tools was relatively limited and could not be widely popularized.

1.2. Technical accumulation (2000-2014)

In the 21st century, with the rapid development of Internet technology, cloud computing, mobile Internet and other emerging technologies provide a broader application scenario for LC. During this period, some enterprises and platforms specializing in LC technology development began to emerge, such as OutSystems, Mendix, etc. They greatly lowered the threshold for software development by providing drag and drop interface components and pre-set functional modules. However, the LC market at this stage is still in its early stages, and there are not many products that can be widely applied.

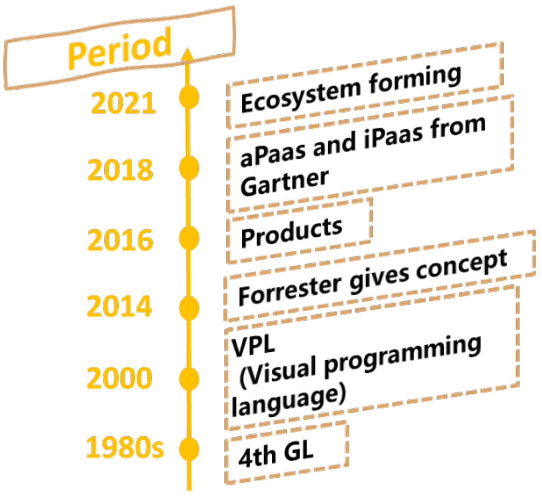

Figure 1: The developement path of LC

6th International Workshop on Experience with SQuaRE Series and its Future Direction, December 03, 2024, Chong Qing, China
* Corresponding author.
† These authors contributed equally.
Zhangwy@cesi.cn (W. Zhang); Liwp@cesi.cn(W. Li); Zhangyy@cesi.cn(Y. Zhang)
0009-0005-9868-5742(W. Zhang); 0009-0001-4065-2527(W. Li); 0009-0006-4940-8527(Y. Zhang)

1.3. Rapid Rise (2014 present)

Around 2014, as the demand for digital transformation in enterprises became increasingly urgent, LC development experienced explosive growth. In 2014, the internationally renowned consulting firm FORRESTER first proposed the term and concept of LC development, which is a software development method that allows enterprises to quickly build and deploy applications with minimal coding work. At this stage, major technology giants such as AWS, Google, Microsoft, and Oracle have successively laid out the LC market and launched a series of LC development tools and services. A typical event among them is in 2018, when Siemens acquired Mendix, a leading enterprise in the LC field, for 600 million euros, marking the official entry of LC technology into the mainstream.

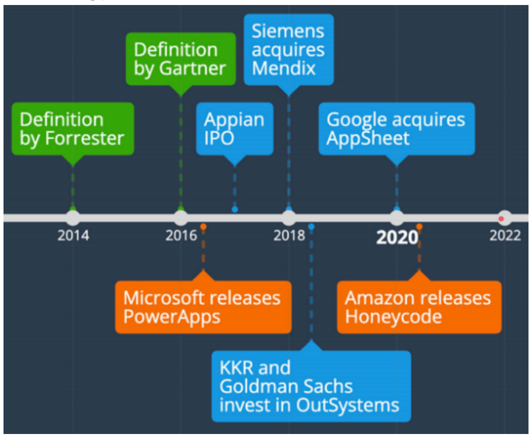

Figure 2: Main events after 2014 [5]

In recent years, with the integration and application of advanced technologies such as artificial intelligence and big data in the field of software engineering, the functions of LC development tools, platforms, and other products have become increasingly powerful, covering a wider range of industries and fields. They are capable of developing everything from simple form based software to complex enterprise level systems.

1.4. Current trend

At present, LC is no longer limited to rapid prototyping and small-scale application development of general software, but is increasingly being applied in enterprise level projects. Meanwhile, with the continuous advancement of technology, LC development related products are also developing towards more intelligent, personalized, and open directions, providing enterprises with more flexible and efficient development solutions. In addition, LC development promotes collaboration between cross functional teams within enterprises or organizations, improving the efficiency of the entire software development lifecycle.

This article mainly focuses on the research of software product quality based on LC. LC here mainly refers to a software development technique and tool pattern, which is mainly aimed at developers. By providing a visual programming interface and configurable approach, it reduces the workload of traditional handwritten code and enables rapid development and testing of software. Therefore, the quality characteristic models/frameworks studied in this article revolve around the impact and innovation of LC on software engineering quality characteristic models.

2. Analysis of the Impact of LC

2.1. Key features of LC

Compared with traditional software products, software products developed based on LC have the main advantages of improving development efficiency, reducing technical barriers, promoting collaboration between business and IT, and utilizing modular code modules to achieve convenient development and management throughout the entire lifecycle. LC development has the following advantages in the software development process:

2.1.1. Agility

LC development embodies multiple advantages when applied to agile software development processes, which align with the core principles of agile development, including improving adaptability, enhancing collaboration, improving efficiency, strengthening flexibility, and iteration. For businesses or organizations, they need the ability to quickly respond to market changes and develop innovations based on consumer demand. LC development compresses the consumption of design, prototyping, iteration, and other steps in the agile development process, introduces automated testing methods, strengthens team collaboration and transparency, and provides better agility for enterprises, thereby achieving the need for rapid delivery of enterprise requirements.

2.1.2. Scalability

In the current market, enterprises need to be able to quickly adapt to market changes and easily expand the functionality and performance of applications to meet evolving business needs. The related products of LC development continuously strengthen the basic capabilities of LC development through their own rapid iteration, expansion of cross platform and cross domain data integration, expansion of integration markets and APIs, configuration of cloud native architecture, etc., providing scalable underlying support for the adoption of LC development tools and platforms.

2.1.3. Team collaboration

The convenience and development efficiency of LC allow users from various business teams to participate in software development together, greatly enhancing team collaboration efficiency and driving overall project development progress. Products mainly based on LC platforms are mainly developed on cloud services, providing users with consistent development standards and standardized component libraries, establishing clear roles and division of labor within the team, improving software version control efficiency, and enabling cross departmental and cross project work. This helps development teams maintain and update code more easily, achieving a significant improvement in team collaboration efficiency.

2.1.4. Quality Assurance [6]

As LC applications move beyond the prototype stage and become part of the IT environment, they begin to pose challenges in terms of design culture, enterprise processes, security, and performance. The LC platform supports automated testing by providing preset components and logic, as well as a large number of testing and debugging tools, helping developers discover and solve problems in a timely manner, significantly reducing errors, defects, and maintenance work caused by human coding, and improving software quality and stability.

2.1.5. Modular assembly

The biggest feature of software development based on LC is the use of fully tested code modules for intuitive assembly and integration through graphical or visual means, which can significantly improve development efficiency, reliability, and code readability, creating conditions for subsequent maintenance and integration of complex systems.

2.2. Improvement of Software Quality Characteristics by LC

LC development significantly improves the efficiency and quality consistency of software development by providing users with standardized and modular code or model components, as well as graphical and visual development interfaces. It simplifies the software development and testing process, enhances security and reliability, improves software maintainability and user experience, strengthens project management and risk management, and thus enhances the quality management level of a software project at multiple levels.

By combining the ISO/IEC 25010:2023 standard, a mapping relationship can be established between LC and software quality characteristic frameworks, which mainly have important impacts on features such as functionality, performance efficiency, usability, reliability, maintainability, and safety.

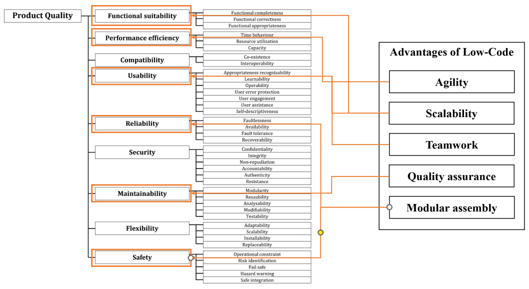

Figure 3: The correlation between LC characteristics and software quality characteristics

3. Research on Quality Assurance Methods for Software Products Based on LC Development

Based on the above research on LC characteristics, combined with the software quality characteristic model described in the current international standard ISO/IEC 25010, research is conducted around the quality characteristics and requirements of LC. Based on 9 quality characteristics, we will focus on exploring the beneficial impact of LC on software quality characteristics for development tools, platforms, and software products formed from LC, and form a quality characteristic framework for LC. [7]

3.1. Functional Suitability

The functional suitability of software involves the ability to meet explicit and implicit requirements, including functional integrity, functional correctness, and functional appropriateness. LC brings about improvements in development efficiency, reusability, maintainability, and other functions during software development, which further supplements and updates the requirements of traditional software quality characteristics.

LC provides users with a series of components, templates, visual design tools, and graphical programming interfaces built through a LC platform. These elements have undergone rigorous testing and validation to meet the functional requirements of users for rapid programming. Combining the new features brought by LC such as reusability and support for automation, LC compliance has improved and perfected the various sub feature requirements of functionality.

To meet the quality requirements of LC, the following new requirements should be proposed for functionality: through user interviews, survey questionnaires, user story mapping, and other methods, a deep understanding of user pain points and needs should be achieved. Enhance methods for user research and requirement gathering to ensure that software functionality truly meets users' actual needs. Verify the integrity of functionality through comprehensive testing, including unit testing, integration testing, and system testing. Implement code review and refactoring, adopting design patterns and best programming practices. Adopting microservice architecture or plugin system, supporting dynamic expansion of functions.

3.2. Performance Efficiency

The performance efficiency of software involves the ability of software products to provide appropriate performance under specific conditions, including time behavior, resource utilization, and capacity. The resource utilization improvement, integration process simplification, automated testing deployment, quality control and other functional and performance improvements brought by LC have put forward new requirements for quality feature frameworks.

LC provides users with automated deployment capabilities, pre configured APIs, intelligent integration tools, quality control, and version control tools through LC tools and platforms, which greatly assist developers and maintainers in improving quality and stability in their work, and enhance performance efficiency requirements.

To meet the quality requirements of LC, the following new requirements should be proposed for performance efficiency: clarify users' requirements and functional characteristics for LC tools and platforms, conduct performance and functional optimization self inspection checklists, and form product selection guidelines. Propose that LC products should maintain stable performance requirements in high concurrency situations to avoid performance bottlenecks. LC platforms should provide powerful integration capabilities that seamlessly integrate with existing enterprise systems and third-party services, including support for APIs, data synchronization, and business process integration. Automated testing should be supported to ensure the quality of applications during development, and continuous integration/continuous deployment (CI/CD) processes should be supported to achieve fast and reliable application deployment.

3.3. Usability

The usability of software involves the level of effort and satisfaction of users when using the product, including ease of understanding, ease of learning, usability, user error protection, user interface aesthetics, and accessibility. By using the LC in development process, there are some advantages like reducing the learning curves of users, enhancing collaboration capabilities, plug-in architecture, and scalability. Those advantages will bring some new requirements to address the quality characteristics of traditional software.

To meet the quality requirements of LC, the following new requirements should be proposed for usability: development platforms such as LC platforms should provide a simple and intuitive interface, allowing users to quickly build data tables, design user interfaces, and configure workflows through click operations. Users' learning costs should be reduced by providing rich pre-built templates and component libraries. Support multi-user collaborative development, provide version control and real-time collaboration functions to improve team efficiency. Adopting a microkernel and plugin architecture design, allowing users to customize and extend system functions according to their needs to meet specific business requirements. We should provide users with flexible customized data models based on business needs, optimize data migration and processing processes, in order to improve the efficiency of data management.

3.4. Reliability

The reliability of software involves the ability of a product to maintain its performance level under specified conditions, including maturity, availability, fault tolerance, and recoverability. In response to the improvement of data security, system stability, code and vulnerability review, compliance, and other aspects provided by LC, corresponding new requirements should be put forward to address the quality characteristics of traditional software.

Data encryption and protection measures should be provided to ensure the security of data during transmission and storage in various usage scenarios. A high availability architecture should be provided to ensure the stable operation of applications, reduce system failures and interruptions. Regular code audits and vulnerability scans should be conducted to detect and fix potential security vulnerabilities. Should comply with data protection regulations, support compliance best practices, and help meet industry standard requirements.

3.5. Maintainability

The maintainability of software involves the ability of software products to be modified, including analyzability, modifiability, testability, and reusability. LC provides the ability to improve maintenance efficiency, simplify error fixes, enhance readability and consistency in the software lifecycle process, and puts forward new requirements for traditional processes.

In terms of maintainability, LC should be based on the characteristics of the technology itself, and quality requirements should be further improved: code generated by LC tools and platforms should have clear comments and follow consistent naming conventions, while providing detailed development and maintenance documentation. It should support a highly modular architecture, allowing each functional module to be independently developed, tested, and deployed. Built in automated testing tools and seamless integration with CI/CD processes ensure that every change is tested. Developers should be allowed to extend the functionality of their applications by writing custom code or using APIs provided by the platform. Strict data security measures should be implemented to ensure compliance with industry regulations and standards.

3.6. Safety

The safety of software is a quality characteristic added to the ISO/IEC 25010: 2023 standard, which involves the safety performance of products under specific conditions, including operational constraints, risk identification, fault safety, hazard warning, and safety integration. LC can help build safer and more reliable software applications by reducing human errors, improving system stability, simplifying fault diagnosis, and enhancing maintainability.

In order to meet the quality requirements of LC products, some new safety requirements have been proposed: it should be ensured that LC platforms provide powerful error handling mechanisms that can capture, record, and respond appropriately to abnormal situations to prevent system crashes and data corruption. A highly fault-tolerant system architecture should be designed to ensure that the system can continue to operate in the event of component failure, such as through redundancy and mechanisms. Those mechanisms are also defined in ISO standards for Functional Safety and affects quality measures. Regular security audits should be conducted on LC platforms and built applications to identify potential risk points and take corresponding mitigation measures. Comprehensive testing tools should be provided, including unit testing, integration testing, and stress testing, to ensure the stability and performance of the application. It should be ensured that the decision-making logic and data processing flow of LC platforms are transparent to developers and security experts, and can be reviewed and verified.

4. Conclusion

LC development has brought new vitality to the field of software engineering, providing new solutions to the cumbersome development process, but at the same time, it has had many impacts on quality management. On the one hand, it significantly improves development efficiency, but on the other hand, it brings challenges to code quality and maintenance. By exploiting the software quality model in ISO/IEC 25010:2023, a set of quality characteristics suitable for LC development, testing and delivery is proposed, which can effectively ensure the quality and performance of LC applications. This requires close collaboration between development teams, testers, and maintainers, as well as a deep understanding of LC platforms and tools. In the current situation where LC has become a trend in software engineering development, it is necessary to continuously clarify and refine specific requirements in conjunction with quality characteristic models, in order to make greater contributions to the improvement of software product quality.

References

[1] LC Development: A Game Changer for Software Engineering. IEEE Software, 2023.

[2] The Impact of Low-Code Platforms on Software Quality and Maintenance. ACM Transactions on Software Engineering and Methodology, 2023.

[3] Best Practices for Testing Low-Code Applications. Software Quality Journal, 2023.

[4] Richardson, C., Rymer, J.: New Development Platforms Emerge for Customer-Facing Applications. Forrester Research, Cambridge (2014)

[5] Davide R., Dimitris K.:Low-code development and model-driven engineering: Two sides of the same coin? Software and Systems Modeling (2022) 21:437–446, https://doi.org/10.1007/s10270-021-00970-2

[6] Noebauer, M., Dhungana, D., Groher, I. (2023). Quality Assurance in Low-Code Applications. In: Yilmaz, M., Clarke, P., Riel, A., Messnarz, R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2023. Communications in Computer and Information Science, vol 1890. Springer, Cham. https://doi.org/10.1007/978-3-031-42307-9_3

[7] ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model