Diving Into Code Quality: Measuring Code Quality

By Elizabeth So, formerly of DisputeSoft

The first installment in this series, Diving into Code Quality: Factors Affecting Code Quality, discussed the factors which contribute to software code quality. Part 1 considered how poor code quality can contribute to failures of software development and implementation projects. In this installment, we discuss tools and techniques utilized by experts to measure code quality.

Measuring Code Quality

An expert usually chooses an appropriate methodology and corresponding technical tools to measure code quality based on the size and complexity of the software under examination. For large, complicated programs, experts usually choose to perform a code quality analysis using automated tools, which can help identify defective code, security vulnerabilities, and code maintainability issues, and which may provide insight into the code's architecture and dependencies. Automated tools also allow experts to examine software for adherence to coding best practices or to coding conventions followed by a development team. After reviewing results generated by an automated tool, an expert may manually review sections of code which appear worthy of further investigation. For small, less complicated programs, or for disputes which hinge entirely on no more than a few small modules of code, experts may forego the use of automated tools and conduct their analysis entirely based on manual review. Most software experts use both automated tools and manual review to assess code quality.

Some of the metrics experts use to measure code quality include complexity, technical debt, code duplication, and coupling. Experts may use automated tools to take these measurements, or they may calculate them manually by, for example, writing and running scripts.

– Code Complexity: Code complexity is a measure of how easy code is to understand and maintain. Smaller blocks of code containing a small amount of functionality are easier to understand and maintain than larger blocks of code containing a large amount of functionality. Complicated code which uses many different looping or nesting structures increases code complexity and can contribute to poor code quality.

– Code Duplication: Code duplication refers to similar or identical sections of reused or refactored source code. Code duplication can reduce maintainability and increase time needed to modify functionality if the same section of code must be modified in multiple parts of a program. Thus, code duplication may indicate poor code quality. However, a developer may decide to use duplicate code in cases where it improves code readability, with a tradeoff of reduced maintainability. Removing duplicate code and functionality into reusable methods or classes that can be called upon when needed (e.g., by inheritance from a superclass, via utility methods) can eliminate or at least reduce instances of code duplication.

– Coupling: Coupling is a measure of interdependence between sections of code. Tightly coupled code connects functionality between two otherwise unrelated sections of code. As a result, modification of tightly coupled code may affect functionality in unintended ways and impact how well the program functions as a whole, which can contribute to poor code quality. On the other hand, loosely coupled code groups together code with related functionality, which allows for easier modification of a program's architecture with minimal impact to other sections of code. Thus, loosely coupled code generally indicates high code quality.

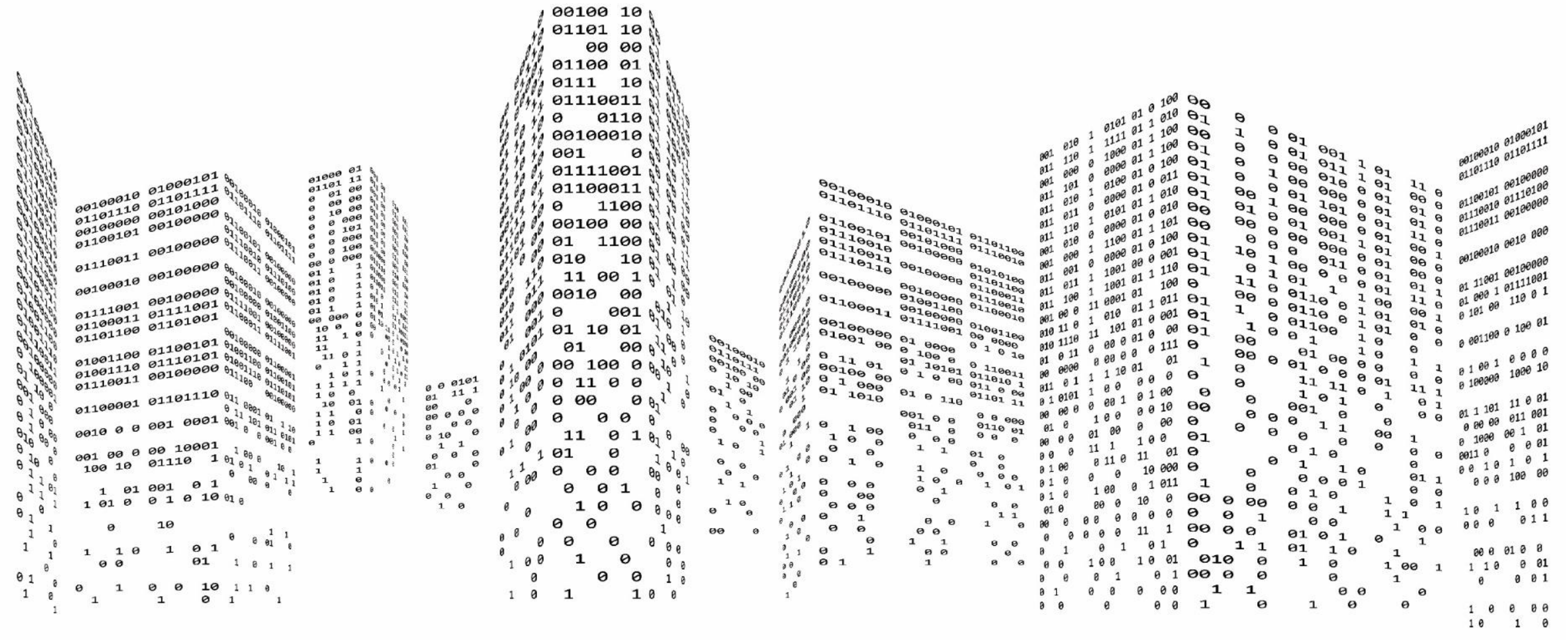
– Technical Debt: Technical debt refers to the cost of additional work needed to correct work performed incorrectly the first time. For example, upon initial detection of a coding or design issue, a developer may apply a quick, temporary solution to keep the project on schedule. However, as the project proceeds, the work needed to replace the temporary solution with a permanent one eventually exceeds the work that would have been necessary to implement a permanent solution in the first place. Thus, technical debt can contribute both to project delay and to poor code quality, as many temporary solutions never receive necessary revisions. Factors including skill of the development team, stability of existing software development environments and processes, and strength of project management can affect the code quality of a software project.

Experts measure code quality using the methods discussed above to arrive at supportable and peer-reviewable opinions regarding the quality of source code. To learn more about the services we provide as software failure experts, please don't hesitate to contact us at inquiries@disputesoft.com.

Read the first installment: Diving into Code Quality: Factors Affecting Code Quality

Read about DisputeSoft's software project failure services.

Read about DisputeSoft's source code examination services.



If you are an attorney in need of a source code expert, we invite you to consider DisputeSoft.

Contact Information

Jeff Parmet, Managing Partner

301.251.6182

jparmet@disputesoft.com

12505 Park Potomac Ave. | Suite 475 | Potomac, MD | 20854