January 17, 2017

Office of the Comptroller of the Currency

12 CFR Part 30

Docket ID OCC-2016-0016

RIN 1557-AE06

Federal Reserve System

Docket No. R-XXXX

RIN 7100-AD16

Federal Deposit Insurance Corporation

12 CFR Part 364

RIN 3064-AE45

Re: Joint advance notice of proposed rulemaking:

Enhanced Cyber Risk Management Standards

We are pleased to provide you with our comments and views about the interagency ANPR on Enhanced Cyber Risk Management Standards for the financial services industry.

CAST is a software company that serves technology-intensive enterprises, mostly in the US and Europe. We have 20 years of direct, hands-on experience helping complex IT departments improve the integrity of their software. Our products and services help customers assess and measure the integrity, robustness, efficiency, security and maintainability of their IT systems. Our customers include consumer banks, consumer finance, institutional banks, clearing & settlement agencies, exchanges, as well as enterprises in the telecoms, insurance, logistics, manufacturing, retail and the public sector verticals. On that basis, we have a deep level of expertise in software engineering as applied to modern business technology across industry, and specifically in the financial services space. CAST does not fall under the scope of this regulation.

Everyone reading this letter knows that technology is increasingly the backbone of business. Technology is a board-level topic at many companies, and increasingly the single most expensive asset. What most readers might not appreciate is that custom software built by enterprises has surpassed a threshold of complexity that would allow for any one person to understand an entire mission critical system. And, unlike any other part of our critical infrastructure, there is no single manager who is responsible and accountable for the construction integrity of their company's software backbone.

Having witnessed the increasing pace of notable outages and breaches in all industry sectors, including financial services, we see this regulation as a welcome sign of industry effort to address some fundamental problems in custom business software built for industry. As you have probably been told, complex systems are never immune to failure. This is indeed common knowledge in the industry, however most business technology organizations have a long way to go to ensure reasonable precautions are taken to prevent cyber events in software operation. The overall level of software risk management and process maturity in IT is not nearly comparable to that in other critical infrastructure, e.g., civil engineering.

Summary

Operational resilience could be significantly increased, and risks in the financial system reduced accordingly, if large and interconnected financial services firms, and those entities' service providers, were required to measure the structural risk of their software at the system level. This measurement would improve the structural robustness of multi-component software and the process for releasing new systems and enhancements into operation. The supervised entities and their service providers should be required to demonstrate, based on evidence, the lack of known software weaknesses for security and known structural flaws for stability.

Overall Comments

It is understandable that time and care is needed to define the systems, organizations and events that should be included in this rulemaking. The agencies must strive to approve rules that are effective and relevant, that have the greatest chance of adoption with the lowest cost, effort and risk to the organizations and systems they govern.

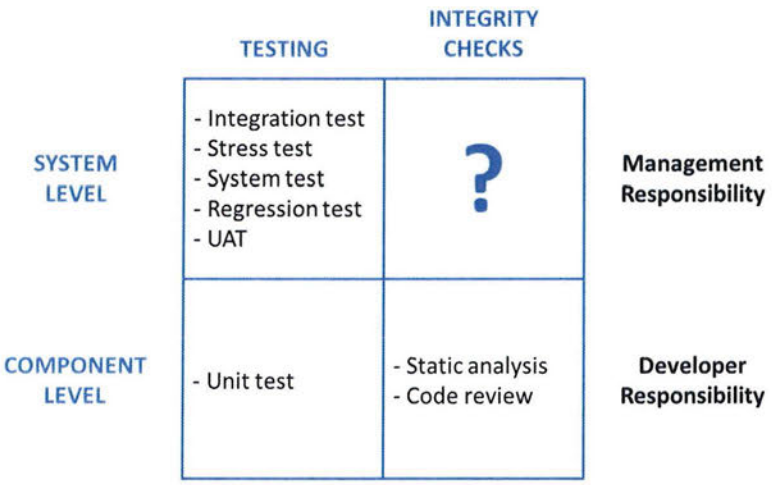
There are many aspects to operating software to ensure adequate "quality, integrity, and reliability," to quote the text of the ANPR. The actions to ensure these characteristics can be broadly categorized into two sets of activities:

1. ***Production*** – all the activities and procedures undertaken during the operation of the software (e.g., failover, disaster recovery, mirroring/duplication, proactive monitoring, multi-site rollover)
2. ***Development*** – the practices and procedures undertaken during the construction and testing of software (e.g., system-level testing, unit test, structural quality analysis, in-phase QA)

Much of the ANPR content deals with the first category of activities – those that occur in production – such as backup, failover and operation procedures. With regard to the ***Development*** category, the content of the rulemaking only specifies testing activities. Testing is an important component of QA in the software development lifecycle (SDLC), but it does not directly address software integrity concerns. While we understand the concern with scope and breadth of regulation, we believe it is equally important to address the root cause issues of system stability and security and to define the characteristics of good systems.

The fundamental issue the agencies are trying to address is to ensure that financial institutions and their suppliers do their utmost to build resilient software of high integrity and security. The pursuit of system integrity has a great deal to do with what the software engineering field calls "nonfunctional" or "structural" aspects of systems. While some issues that can cause a cyber event may be due to flawed logic programmed and executed by a market participant (functional quality), experience has shown that most incidents are caused by technical problems having to do with the engineering and construction of the software itself (structural quality).

Most organizations spend a lot of time testing their software. Testing is the predominant approach to QA in most industries. While testing can cover the majority of day-to-day scenarios, testing has proven to not be enough to assess all forms of software risk. The quality at the code level has a direct impact on software risk. Most companies assume that management just hires good developers, and good developers will produce good code. So the responsibility is ambiguously laid at the developers' feet. Furthermore, even in the best of circumstances, assuming every company only recruits from the top of the bell curve, each developer is responsible only for the components they build. No developer is responsible for the construction quality – the structure – of the multi-component system. This is a huge blind spot in most IT organizations today, including those serving most financial services businesses.



It is worth noting that every industry believes in its uniqueness just as much as every company believes it is unique. While it's true that there are many context-specific tenets to developing custom software for financial services, the basics of software engineering are the same everywhere. The software issues faced by banks have been faced in other industries. The focus of our comments are to point the regulatory efforts in the direction of managing structural integrity, at the system level, to directly address the cause of cyber events witnessed by customers of financial services, much like software incidents in other industries.

Specific Comments and Areas of Concern

Our comments deal with a subset of the questions posed by the ANPR, where they pertain to the content of the tests, assessments and methodologies that exist in industry today to identify issues that

typically impact the quality, integrity, and reliability of core and sector-critical systems. Our comments do not extend to other sections of the rulemaking document, with the exception of some brief comments on the cost-benefit model for the regulation.

We support the proposal to organize the standards into five broad categories, namely Cyber risk governance; Cyber risk management; Internal dependency management; External dependency management; and Incident response, cyber resilience, and situational awareness.[1] In particular, we commend the agencies for recognizing that internal dependencies are of the same importance and deserve the same attention as external dependencies and responses to cyber incidents. As the agencies recognize with the reference to "the use of legacy systems acquired through a merger," enterprise applications at large entities are often an assembly of leftover software. Legacy software and newly developed application code are typically interacting with software packages from different vendors using different technologies, frameworks and standards. The software ultimately deployed by any given financial entity is a unique integration of components and modules that have been written at different times and in different places and never been reviewed in their entirety. If the entire system is not safe, robust, efficient and easy to maintain, operational resilience of these entities may be jeopardized and risks to the financial system may be multiplied.

Most software-intensive organizations have already done a lot of work on these areas to improve their ability to operate their software with high availability. This has yielded good results in most industries and, based on our experience, the level of sophistication at financial services institutions in all these areas is relatively high. That is to say, the banks and third parties that account for the majority of market share already meet or exceed the above requirements broadly speaking, depending on their interpretation.

We believe these points are not specific enough about very important aspects of software integrity and security that have to do with the weaknesses that can be introduced in software during construction. There is a well-established canon of structural software weaknesses that has been identified through various industry efforts – most notably led by the Consortium for Software Quality (CISQ), among others. Avoiding well known software flaws is becoming an important practice across industries. Some of the more advanced software engineering teams in IT even establish company-specific, or application-specific standards based on root cause analysis (RCA).

What constitutes high-integrity software?

We believe the rule-writing team should consider controlling structural quality of software. This has been proven to be an effective way to reduce the number of live system incidents that lead to stability, performance and security problems. In our experience, applying structural quality measurement standards at multiple financial services customers, we've seen these issues get directly to the core of software integrity, stability and performance issues.

[1] ANPR at page22.

As we work with our clients on their complex, mission critical systems, we occasionally are able to gather data about their system test results and operational incidents, in order to analyze the impact of their structural quality program together with our stakeholders. Some of these results are shown in Figure 1, below.

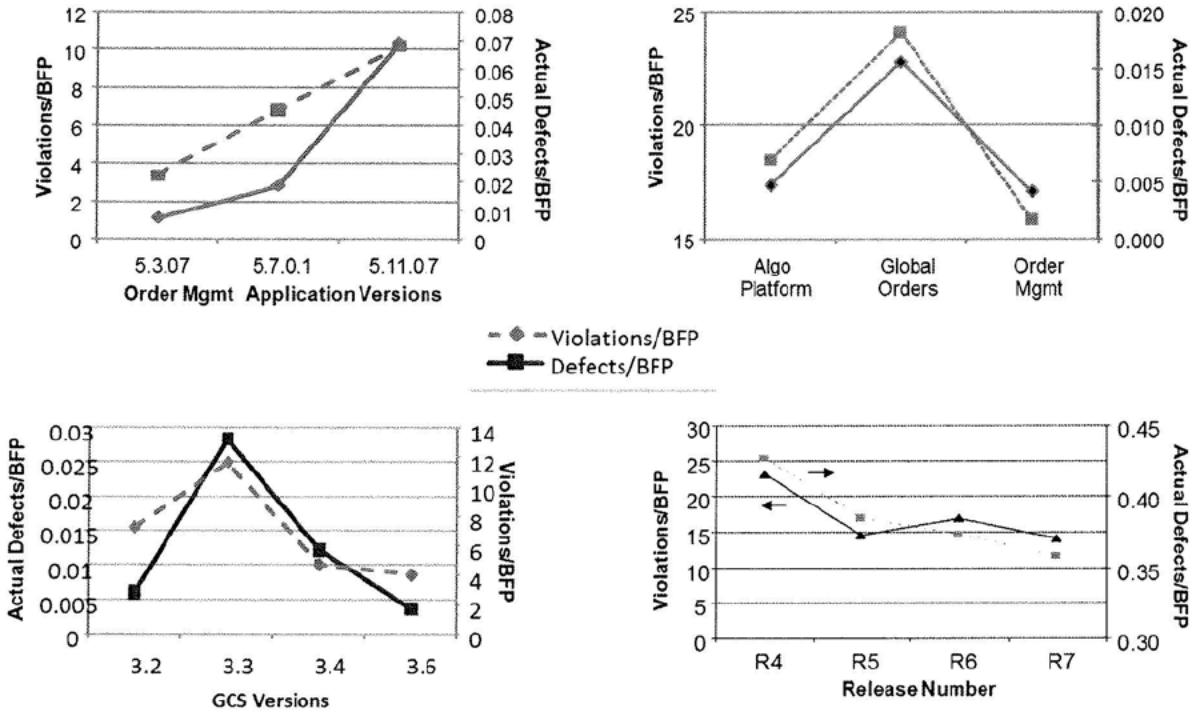


Figure 1. Correlation of Structural Quality and Issues in Operation[2]

While everyone knows in their gut that good software works better than poorly-engineered software, these data show an overwhelming correlation between software weaknesses and defects. Specifically at issue is the fact that by controlling for good engineering of software, management can proactively control for defects and cyber events.

Focus on System Robustness and Integrity Will Have Positive Impact

At CAST, we have been looking at several areas of nonfunctional quality that are relevant to most IT organizations. Security is among these, the others are Robustness, Performance Efficiency, and Maintainability. We have aligned our metrics to the prevailing standards from CISQ and ISO. Robustness is a measure of system integrity that has a direct tie to stability. Over time, CAST Research Labs (CRL) has collected a repository of data that we use for benchmarking and industry analysis. CRL publishes a detailed industry report every two years, based on this dataset, called the CRASH Report. This dataset currently contains about 2500 IT applications comprising about 3 billion lines of code.

[2] GCS, the version numbers, release numbers, are all fictionalized names and release numbers of actual trading systems in industry; BFP (Backfired Function Point) is a normalized measure of size based on lines of code

Examining the data from the 2012 CRASH Report[3], we have a data breakdown by major industry vertical. Not surprisingly, the financial services industry is well above average for its Security scores. Given the overriding focus, and pockets of regulatory scrutiny already placed on financial services regarding security. Regulatory focus seems to drive results. Yet, there is a long tail of poor security as well, as evidenced by the long whisker below the mean in the box-and-whisker in Figure 2, below.

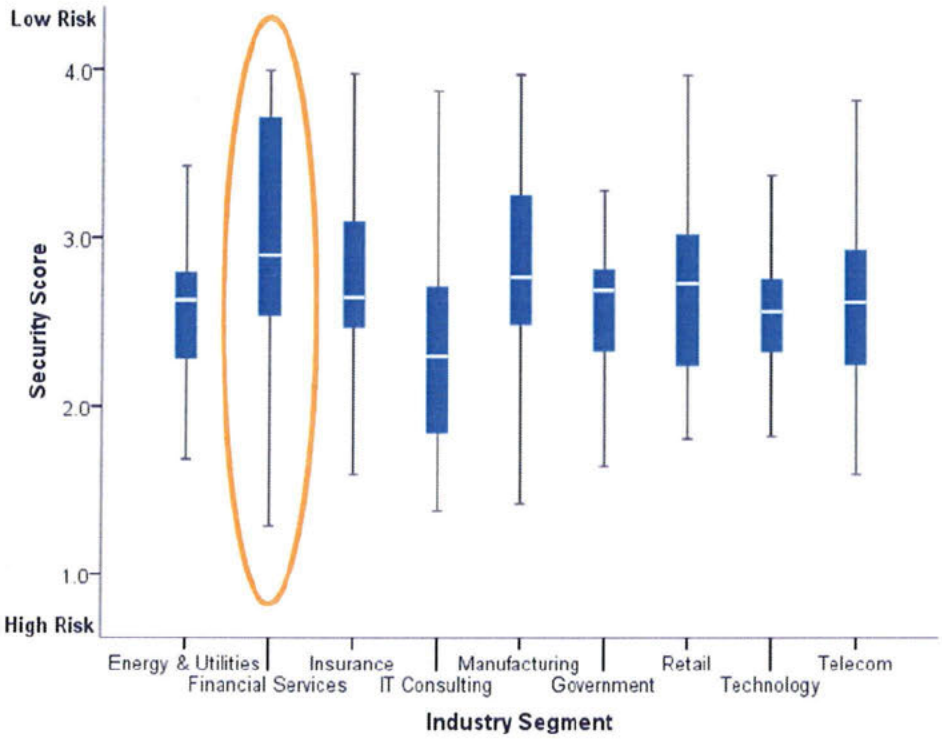


Figure 2. Security Score Distribution by Industry Sector

While the robustness of systems in financial services is pretty much average with the rest of the industries analyzed, collected in a tighter distribution – never terrible and never excellent. Shown in Figure 3, the below statistics are based on the same dataset as figures 2 and 4.

[3] CAST Research on Application Software Health – CRASH Report 2012, http://www.castsoftware.com/research-labs/crash-reports; n=365 million lines of code comprising 745 applications

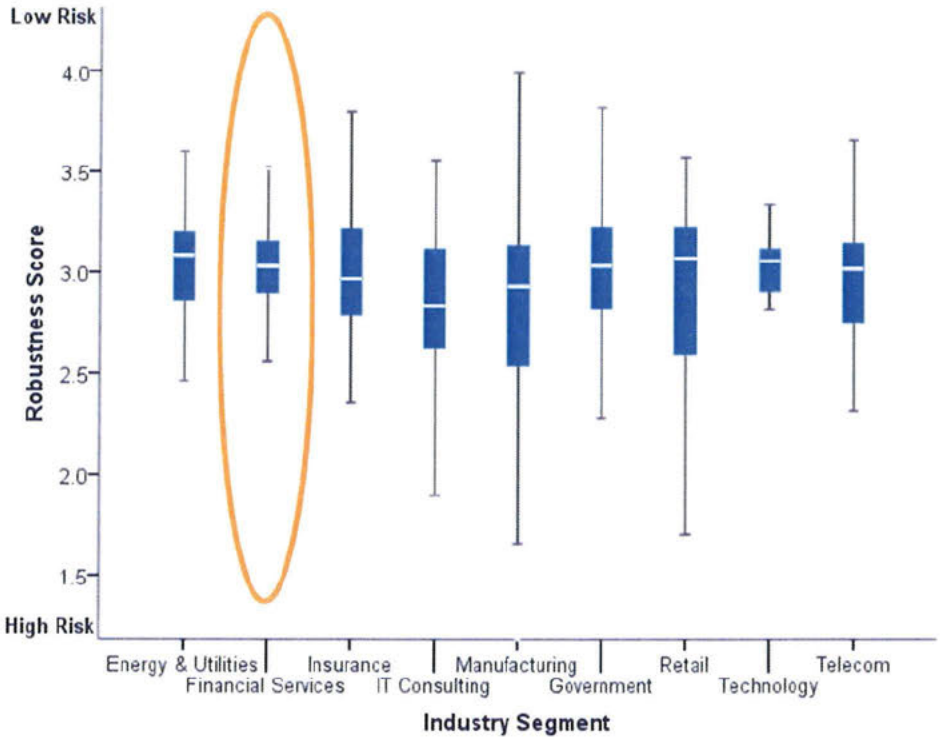


Figure 3. Robustness Score Distribution by Industry Sector

The Robustness scores are a reflection of the extent to which the systems are built with defensive coding in mind, and with failsafe mechanisms built into the software to ensure it withstands extreme, edge situations. Clearly the pursuit of fault tolerance at financial services institutions has not kept up with their pursuit of application security. All this as the complexity of financial systems software continues to increase. We know our industry feels more complex than the others and in this case, this industry is different than the others. Consider Figure 4, below, where another dataset from the same CRASH report shows this clearly.

While the Public Sector has the highest proportion of high-complexity software, financial services is by far above the rest of the private sector. As complexity increases, keeping the same pace of business change and the same QA techniques without direct focus on structural robustness of software, cyber events are bound to happen.

Measuring the nonfunctional or structural quality aspects of software has a long history in software engineering. The purpose of measurement specifications such as ISO and CISQ[4] is to create standards for measuring Software Quality Characteristics that are automated, objective, economical to use, and technically feasible. The objective of the work leading to these specifications has been to provide international standard definitions against which IT organizations, IT service providers, and software vendors can implement automated measurement of the structural quality of software.

[4] Consortium for IT Software Quality, www.it-cisq.org

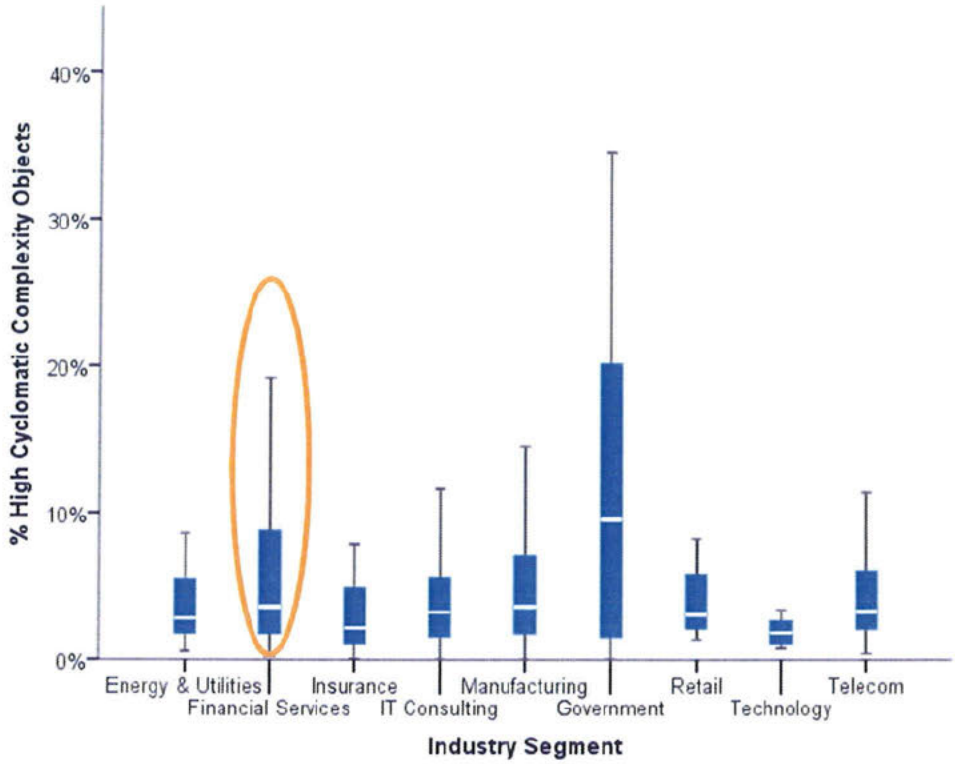


Figure 4. Complexity Distribution by Industry Sector

In order to maintain consistency with the ISO/IEC 25000 series of standards (System and software Quality Requirements and Evaluation, SQuaRE), Software Quality Characteristics are defined for the purpose of this specification as attributes that can be measured from the static properties of software, and can be related to the dynamic properties of a computer system as affected by its software. The ISO/IEC 25000[5] series is replacing ISO/IEC 9126 and is the international standard for defining the elements of internal software quality. The CISQ published their standards for Reliability, Performance Efficiency, Security and Maintainability in 2015.

System Level

It is important to note that Structural quality can be measured at the individual component level, but it's critical for complex software to be considered as a whole system. The ANPR has some discussion of the systemic risk a sector-critical system might pose if it's linked to other systems in the market. This is as much an issue within within one system, as it is clearly an issue when considering the entire network across multiple organizations. That is to say that every banking system within one organization is comprised of multiple unit-level components, often built by many individuals and sometimes by outsourced vendors. These components need to come together congruously and the only time we get to see them work together is during testing.

[5] International Standards Organization, http://www.iso.org/iso/catalogue_detail.htm?csnumber=35683

This is the technical crux of the issue – that the only approach currently applied to assure the integrity of multi-component software is testing. Longer testing windows will surely provide more assurance. Having test symbols and mandatory coordinated test windows for market players to test their interoperation are absolutely essential. But, within each entity, testing cannot be the only approach to identifying software risk at the system level. Code-level structural analysis of software has to take place at the integration level as much as it should be done by each developer.

Just as multiple market players can interact in unpredictable ways during extreme events, components of a multi-component application can also behave in unpredictable ways. It is well known in software engineering that the most important and least visible issues are those that arise from the interactions of even the most beautifully-coded components. Industry research suggests that over 90% of adverse events in production are caused by multi-component defects[6]. Thus the analysis of structural quality, even within one organization, must take the entire system view into consideration as much as possible, in order to assess its integrity and target the risk of cyber events taking place. Testing alone cannot possibly go through all the scenarios necessary to simulate real world events. We heard that already from market participants at the roundtables. Structural analysis is a necessary supplement.

Summary of Our Specific Recommendations

Based on the above analysis, our recommendations are fairly straightforward additions to the existing language in the regulation.

1. Financial services entities should measure the structural risk of their software at the system level. This measurement should serve as a way to continually improve the structural robustness of multi-component software and as a decision mechanism for releasing new systems and enhancements into operation
2. Include standards in that pertain to structural software quality, such as ISO 25010 and the CISQ Software Quality Specification[7]
3. Establish a system of evidence-based claims of the lack of known software vulnerabilities for security and known structural weaknesses for stability as safe harbor criteria
4. Enable the sharing of root cause information as well as structural quality benchmark data for the benefit of market participants.

Cost of Compliance

We realize that introducing new hurdles to being compliant has implications on the cost-benefit analysis that goes along with the proposed regulation. We would like to submit some important evidence in support of structural quality control from a financial cost-benefit standpoint.

---

[6] OMG Paper on Resilient Systems, http://www.omg.org/CISQ_compliant_IT_Systemsv.4-3.pdf
[7] http://www.it-cisq.org/standards

As much as it is well known in software engineering that zero-defect software is unattainable, it is also well known that preventing defects from entering in software construction is the most cost effective approach to QA. It is 10x cheaper to find a defect in development than it is during system test. It is 100x cheaper to fix in development than in production – and that's not accounting for the impact to business. In addition, software of higher quality is cheaper to maintain and easier to enhance. The concepts explained by Philip Crosby in his famous book *Quality is Free*[8] apply equally to software engineering. It is indeed less expensive to put quality into the product, than to deal with it in testing and in production.

It is well known that CMM Level 1 organizations spend 40% of their application development time doing rework – most of that happens during the QA phase, the industry euphemism for testing. "In general, testing schedules for low-quality, large software projects are two to three times longer and more than twice as costly as testing for high-quality projects," say the authors of *The Economics of Software Quality*[9], a recent definitive text on cost of quality. This has an impact on project launch schedules and overall effort to get new functionality out the door. Anything that helps address quality during construction has a positive cost impact on overall development.

CAST has done some research on this in our client base as well, specific to another area of cost of quality that has to do with the maintenance of mission critical systems. Consider the data in figure 5, below.

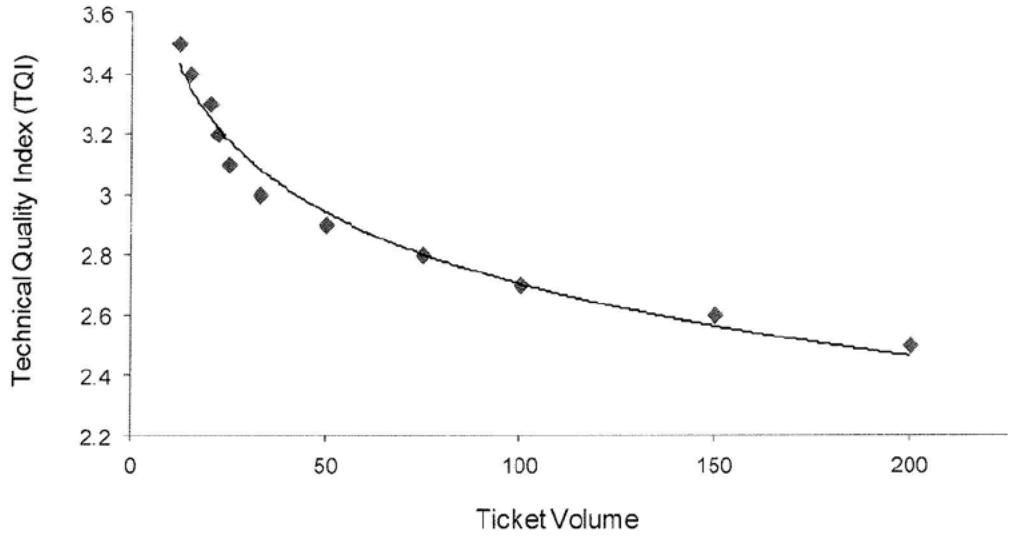


Figure 5. Correspondence of Structural Quality to Ticket Volumes

This is a snapshot of one of the major cost drivers in routine corrective maintenance – having non-critical bug fix tickets to handle during routine operation. CAST experts studied this correlation across 20 large, mission critical systems across several industries. What was found is that improving overall structural quality by 10% reduces ticket volume by over 30%. This is another example of cost that can be

[8] P. Crosby, 1980, Quality is Free, Mentor, ISBN 0451625854

[9] C. Jones, O. Bonsignour, 2012, The Economics of Software Quality, Addison-Wessley, ISBN 0132582201; quote taken from page xxii in the Preface

reduced by having higher quality software. This is common sense knowledge in software engineering, but few organizations in financial services have taken steps to measure structural quality in order to target root drivers of maintenance cost. This will be an inadvertent benefit of controlling integrity at the structural level that may even compensate for the cost of other aspects of the ANPR.

Conclusion

To summarize, we applaud the direction of this rulemaking. The agencies proposing this rule are moving to take an important step in helping set a minimal level of software integrity in the technology that runs our financial markets. Some industries, such as aerospace and medical devices, already carry this form of regulation. Other industries have yet to follow suit.

While the procedures and content outlined by the ANPR are a step in the right direction, additional steps along the lines of our specific recommendations are necessary in order to address software risk directly. We believe the ANPR would be significantly more likely to achieve its stated goals of increasing operational resilience and reducing risks in the financial system if it asked for measurement of structural quality weaknesses and vulnerabilities. Requiring large and interconnected financial institutions and their service providers to produce evidence of the absence of such weaknesses and vulnerabilities would represent the "state of the art" in managing system integrity and place the agencies in the lead on this topic in the regulatory community.

We hope our comments will be helpful to the interagency team in its pursuit of improving system integrity in the markets. We appreciate the opportunity to offer CAST's views on this important issue. We would be happy to meet in person to go over these points in more detail. In the meantime, please do not hesitate to contact us with any further inquiry or clarification.

Respectfully submitted,

Lev Lesokhin/mf

Lev Lesokhin
Executive Vice President, Strategy and Analytics

CAST, Inc.
321 West 44th Street, Floor 5
New York, NY 10036
(212) 871-8330