CAST
ACHIEVE INSIGHT. DELIVER EXCELLENCE.

To: SEC Division of Trading and Markets

Securities and Exchange Commission
100 F Street, NE
Washington, D.C. 20549

Re: SEC Proposed Rule – Regulation SCI SEC File No. S7-01-13; Release No. 34-69077

Date: Monday, July 8, 2013

Dear SEC colleagues,

We are pleased to provide you with our comments and views about the SEC's proposed rulemaking on Regulation SCI (Systems Compliance and Integrity).

CAST is a software company that serves technology-intensive enterprises, mostly in the US and Europe. We have over 15 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 institutional banks, clearing & settlement agencies, exchanges, as well as enterprises in the telecoms, insurance, consumer finance, 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 markets. CAST does not fall under the scope of SEC regulation.

Everyone reading this letter knows that technology is increasingly becoming the backbone of business. Technology is becoming 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 in all industry sectors, including financial exchanges, 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 heard multiple times at the roundtables and other comment letters thus far, complex software is 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 adverse 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. This regulation, though incremental to prior voluntary measures such as ARP, is a laudable step in pulling the industry in the right direction. We are pleased to see the SEC take proactive steps in the area of system integrity.

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 Commission 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 it governs.

There are many aspects to operating software to ensure adequate "capacity, integrity, resiliency, availability, and security," to quote the text of Regulation SCI. The actions to ensure these characteristics can be broadly categorized into two sets of activities:

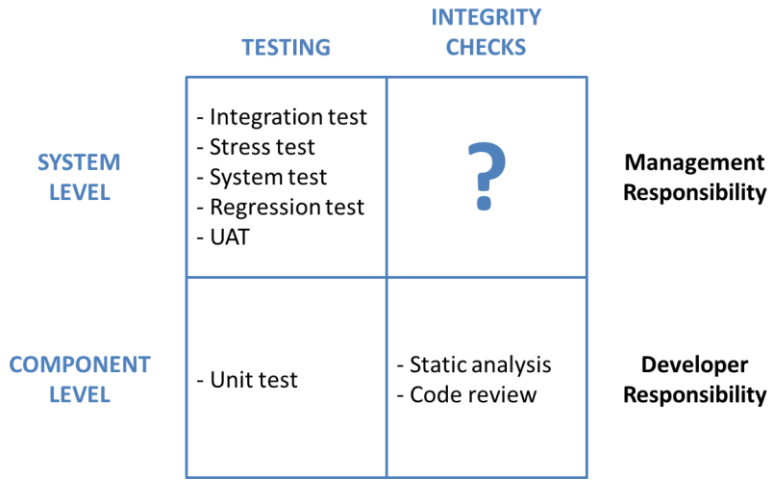
1. ***Production*** – all the activities and procedures undertaken during the operation of the software (e.g., failover, disaster recovery, mirroring/duplication, proactive monitoring, multi-site rollover)
2. ***Development*** – the practices and procedures undertaken during the construction and testing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

Much of the Rule 1000 content deals with the first category of activities – those that occur in production – including reporting of incidents. With regard to the ***Development*** category, the content of the rulemaking only specifies testing activities. Testing is an important component of QA in the SDLC, but it does not directly address software integrity concerns, such as the SCI events increasingly tolerated by the markets over the last 24 months. 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 to define the characteristics of good systems.

The fundamental issue the Commission is trying to address is to ensure that the SROs and their members do their utmost to build resilient software of high integrity. 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 an SCI event may be due to flawed logic programmed and executed by a market player (functional quality), experience has shown that most SCI events 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 of the code itself has a direct impact on software risk and most organizations do not take responsibility for the quality of the code beyond the developer level. Most companies assume that management just hires good developers, and good developers will produce good code. 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 blindspot in most IT organizations today, including those serving most Financial Services businesses in the markets.



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 the financial markets, the basics of software engineering are the same everywhere. The software issues faced by the markets 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 SCI events witnessed in the financial markets, much like software incidents in other industries.

Specific Comments and Areas of Concern

Our comments deal primarily with the proposed Rule 1000(b)(1), requiring that [each SCI entity's policies and procedures be reasonably designed to ensure that its SCI systems and, for purposes of security standards, SCI security systems, "have levels of capacity, integrity, resiliency, availability, and security, adequate to maintain the SCI entity's operational capability and promote the maintenance of fair and orderly markets."] – as stated on page 93 of the proposed rule.

The Commission requests comment generally on the proposed Rule 1000(b)(1), as well as specific points, as outlined in questions 60-95. Our comments deal with a subset of these questions, where they pertain to the content of the tests, assessments and methodologies that exist in industry today to identify issues that typically impact the reliability, efficiency, integrity, resiliency, availability and security of SCI 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.

Overall the content of the proposed ruling is hitting many of the right points. To take directly from the document, p.87, the specific actions proposed by Regulation SCI include:

- (A) the establishment of reasonable current and future capacity planning estimates;
- (B) periodic capacity stress tests of such systems to determine their ability to process transactions in an accurate, timely, and efficient manner;
- (C) a program to review and keep current systems development and testing methodology for such systems;
- (D) regular reviews and testing of such systems, including backup systems, to identify vulnerabilities pertaining to internal and external threats, physical hazards, and natural or manmade disasters;
- (E) business continuity and disaster recovery plans that include maintaining backup and recovery capabilities sufficiently resilient and geographically diverse to ensure next business day resumption of trading and two-hour resumption of clearance and settlement services following a wide-scale disruption; and
- (F) standards that result in such systems being designed, developed, tested, maintained, operated, and surveilled in a manner that facilitates the successful collection, processing, and dissemination of market data.

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 the exchanges in all these areas is relatively high. That is to say, the exchanges that account for the majority of market volume 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 quality and integrity 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). This is a topic we will comment on in more detail.

The language of the proposed rule comes close to specifying these practices, but we believe it should be more explicit. Section (D), regarding regular reviews to "identify vulnerabilities pertaining to internal and external threats, physical hazards, and natural or manmade disasters" comes close to specifying this level of structural quality review. Though the term "vulnerabilities" typically pertains to weaknesses exploitable by potential hackers. Security is only part of the issue the Commission is trying to address, hence the language should specify all types of software weaknesses that comprise structural quality. Specific language could say something like: "Identify system vulnerabilities and weaknesses that pertain to internal and external threats, structural integrity, physical hazards, system reliability, and natural or manmade disasters."

Section (F) also speaks to having "standards that result in such systems being designed, developed, tested, maintained, operated, and surveilled in a manner that facilitates the successful" operation of systems. The proposed regulation then itemizes the standards in Table A, which seem to be the

standards being proposed to meet Section (F). All the standards in Table A, however, only address ***Production*** issues. None of these standards address policies, procedures or any guidance for ***Development***. To be effective, Table A should also list standards that are relevant for ***Development***, such as the CISQ standards that specify requirements in order to control the stability, efficiency, security and maintainability of IT software.

What constitutes high-integrity software?

We believe the Commission 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.

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.

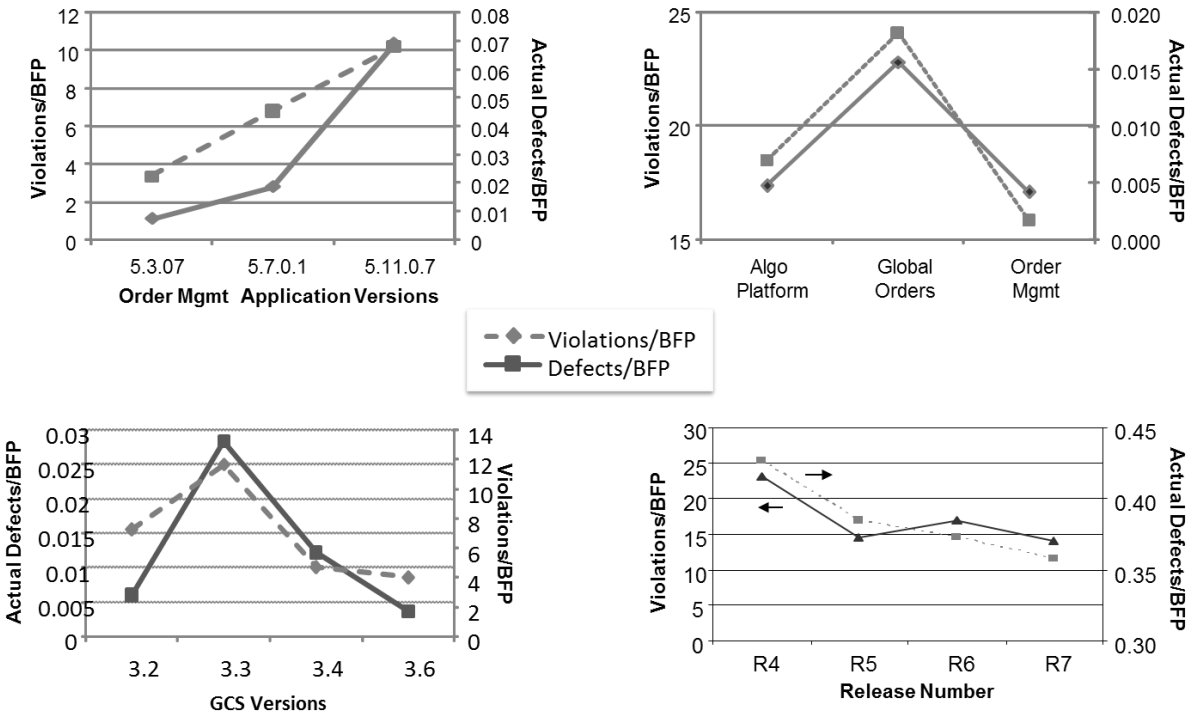


Figure 1. Correlation of Structural Quality and Issues in Operation[1]

[1] 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

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 SCI 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, Changeability and Transferability. 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 1500 IT applications comprising about 700 million lines of code.

Examining the data from the 2012 CRASH Report[2], 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. See Figure 2, below.

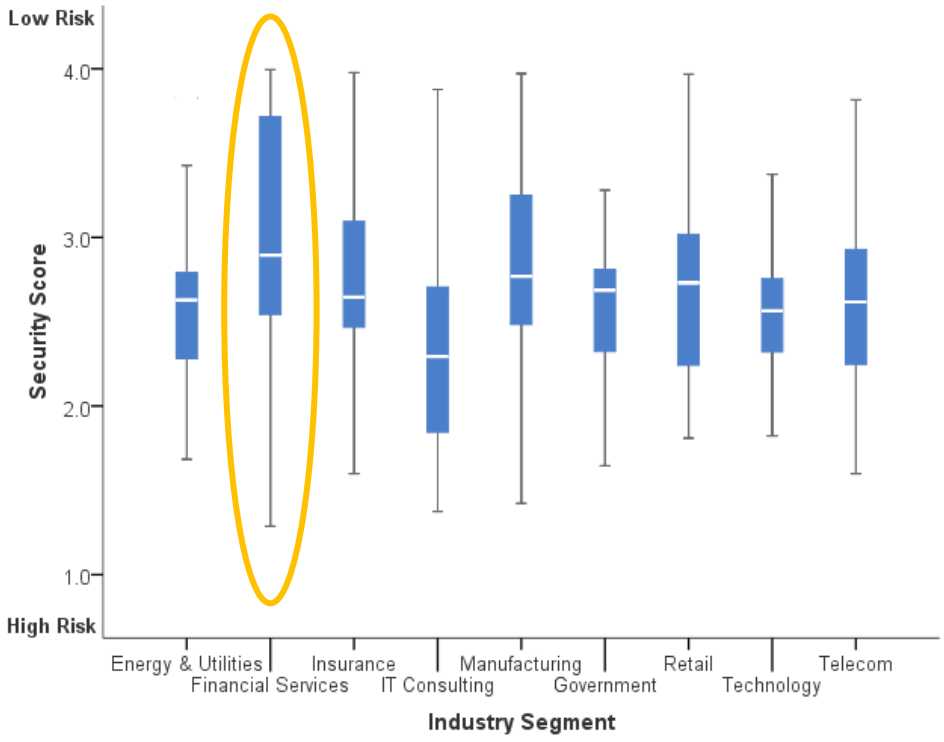


Figure 2. Security Score Distribution by Industry Sector

[2] 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

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.

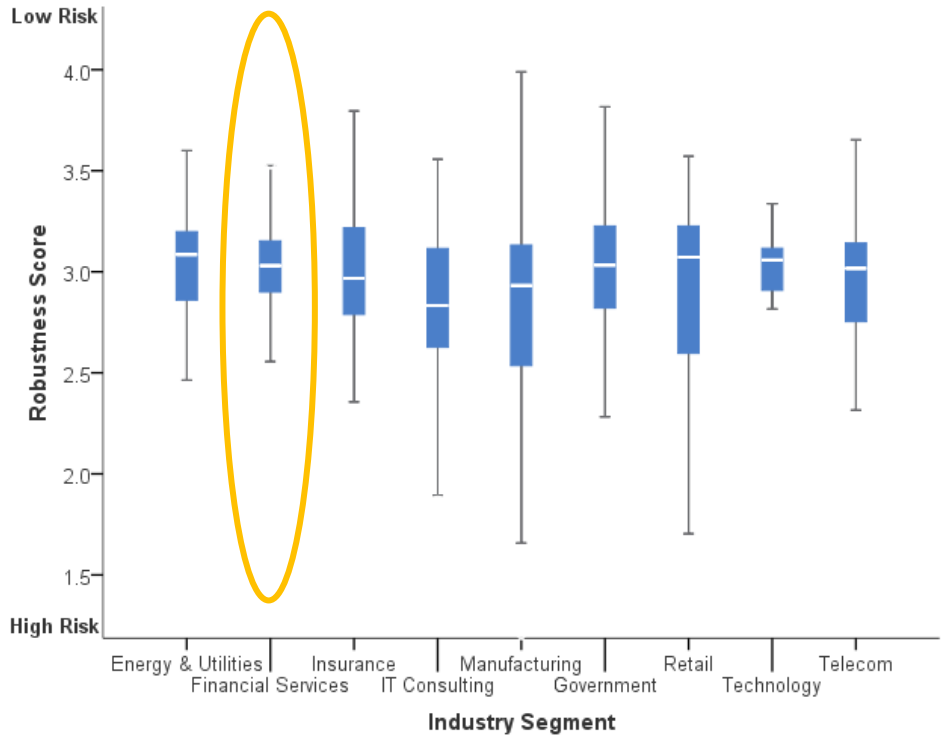


Figure 3. Robustness Score Distribution by Industry Sector

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, SCI 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[3] is to create standards

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

for measuring Software Quality Characteristics that are automated, objective, economical to use, and technically feasible. The objective of the work leading to these specifications is 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.

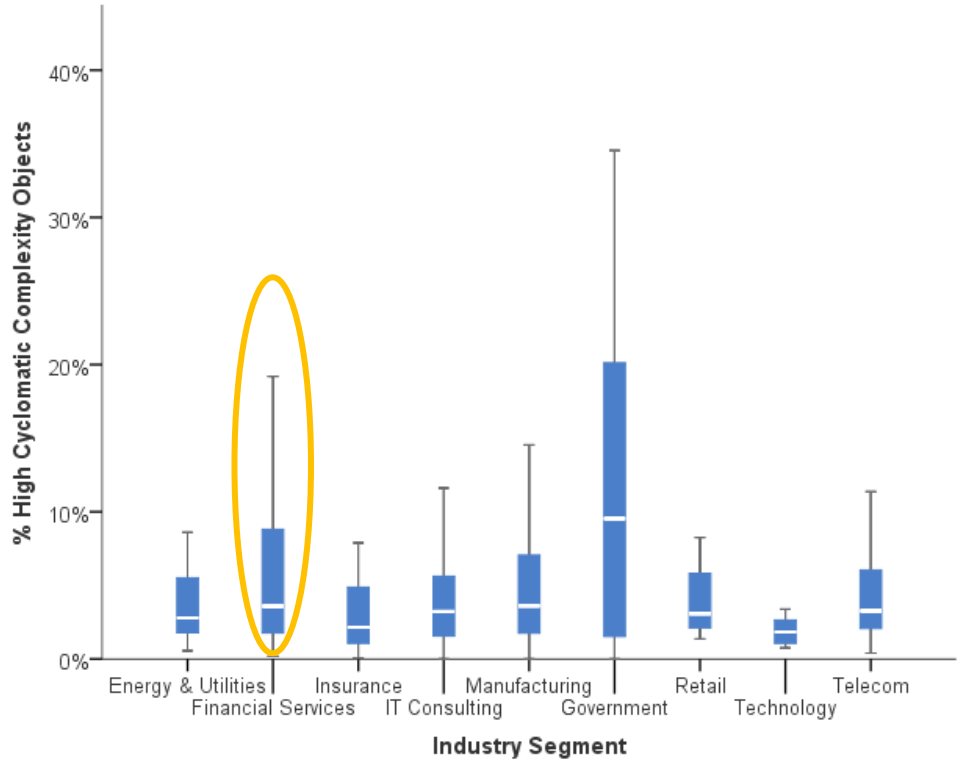


Figure 4. Complexity Distribution by Industry Sector

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[4] series is replacing ISO/IEC 9126 and is the international standard for defining the elements of internal software quality.

System Level

Structural quality can be measured at the individual component level, but it's important for complex software to be considered as a whole system. The roundtables had a lot of discussion of the difference between the quality of a component vs. the quality of multiple components working together. This is as much an issue within one SRO organization, or within one system, as it is clearly an issue when considering the entire trading network across multiple organizations. That is to say that every matching

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

engine and every order 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.

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 SCI 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[5]. 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 SCI 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.

Root Cause Analysis and Beneficial Information Sharing

Root cause analysis (RCA) is the practice of understanding the reason incidents occur, categorizing those reasons and analyzing them with the goal of preventing known patterns from occurring in the future. Much of software engineering best practice and known weaknesses stems from years of RCA conducted across software development.

As Regulation SCI stipulates additional reporting about SCI events, it would be beneficial for SCI entities that are reporting on events to share root cause information with regard to whichever production incidents are deemed appropriate for reporting to SEC. The reporting framework should endeavor to provide safe haven for entities to report this data freely and to disseminate such data, perhaps anonymized, for the benefit of other market members. Some of the common issues can be codified in a common framework of known technical problems that are to be measured and avoided.

Summary of Our Specific Recommendations

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

1. SCI 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-

[5] OMG Paper on Resilient Systems, http://www.omg.org/CISQ_compliant_IT_Systemsv.4-3.pdf

component the software and as a decision mechanism for releasing new systems and enhancements into operation

2. Include standards in Table A that pertain to structural software quality, such as ISO 25010 and the CISQ Software Quality Specification[6]
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.

These are the quick summary of recommendations talked about in the text above. Each could be elicited in far more detail than this format permits. Please feel free to contact us for any clarifications or detailed recommendations on these points.

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.

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*[7] 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*[8], 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.

[6] http://www.it-cisq.org/standards
[7] P. Crosby, 1980, Quality is Free, Mentor, ISBN 0451625854
[8] C. Jones, O. Bonsignour, 2012, The Economics of Software Quality, Addison-Wessley, ISBN 0132582201; quote taken from page xxii in the Preface

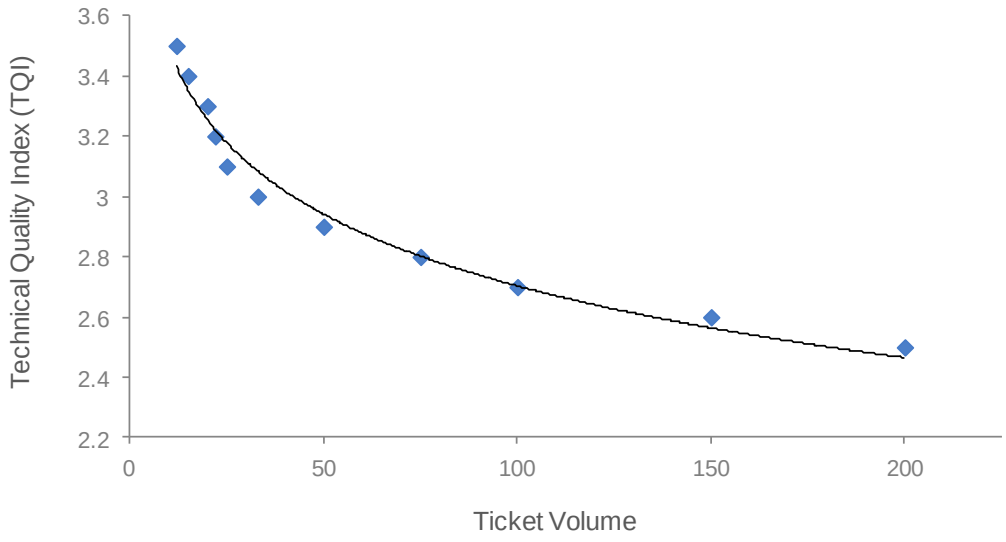


Figure 5. Correspondence of Structural Quality to Ticket Volumes

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 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 Regulation SCI.

Conclusion

To summarize, we applaud the direction of this SEC rulemaking. The SEC is 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 suite.

The procedures and content mandated by Regulation SCI, as summarized in section 1000(b)(1), will not introduce significant new practices at the SROs and member companies that will directly impact system integrity. As it stands, the regulation threatens to add process and paperwork without moving the needle to directly address software risk. We believe the SEC would be more proactive in its approach if the content of this regulation addressed structural quality weaknesses and vulnerabilities head on by asking for evidence of their absence and the measurement thereof. This would represent the state of the art in managing system integrity and would position the SEC in the lead on this topic among its constituents.

As the industry reports standardized structural quality data to the SEC, the SEC can build a repository of scores to feed back to SCI entities and provide a valuable service to the markets in setting a level playing

field for the technical component of go/no-go decisions to roll out new software. We fully support this effort and stand ready to support any organization that decides to take a deeper analysis of system integrity in the financial markets.

We hope our comments will be helpful to the Commission in its pursuit of improving system integrity in the markets. We appreciate the opportunity to offer CAST's views to the Commission on this important issue. Please do not hesitate to contact us with any further inquiry or clarification on the points above.

Respectfully submitted,

Lev Lesokhin
Executive Vice President, Strategy and Markets

CAST, Inc.
373 Park Avenue South, Floor 5
New York, NY 10016
(212) 871-8330