Ms. Elizabeth M. Murphy
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

Dear Ms. Murphy,

I are writing on behalf the Consortium for IT Software Quality (CISQ), a Special Interest Group of the Object Management Group (OMG) to provide comments regarding the SEC's proposed rulemaking on Regulation SCI (Systems Compliance and Integrity).

The Object Management Group (OMG®) is an international, open membership, not-for-profit, computer industry standards consortium. Founded in 1989, OMG standards are driven by vendors, end-users, academic institutions, and government agencies. OMG Task Forces develop enterprise integration standards for a wide range of technologies and an even wider range of industries including financial services. OMG is dedicated to bringing together end-users and experts from government agencies, universities, and research institutions into communities of practice to develop computing standards such as the Unified Modeling Language (UML), Business Process Modeling Notation (BPMN), Common Object Request Broker Architecture (CORBA), and Model Driven Architecture (MDA). OMG standards are free and easily accessible from the OMG website (www.omg.org).

The Consortium for IT Software Quality (CISQ) is an IT industry leadership group comprised of IT executives and technical experts from the Global 2000 IT organizations, system integrators, outsourced service providers, and software technology vendors convened to develop standards for automating the measurement of software quality characteristics and size. CISQ measures are consistent with international standards such as the ISO 25000 series. However, CISQ extends and supplements these standards by providing specifications for automating the development of measures from the source code of IT applications. CISQ's agenda is driven by its members and sponsors, membership in CISQ is free, and members can access CISQ measurement specifications from its website (www.it-cisq.org). CISQ has become a voice for IT leaders on the importance of measuring and ensuring structural quality in mission critical IT software applications. Accordingly it is within the scope and objectives of both OMG and CISQ to provide the accompanying input to SEC concerning Regulation SCI.

Sincerely,

Dr. Bill Curtis

Director

Consortium for IT Software Quality (CISQ)

**Specific Comments Regarding proposed Rule 1000(b)(1)**

**60.** *Do commenters believe the proposed scope of required policies and procedures is appropriate?*

**Discussion:** We believe that scope of required policies and procedures is appropriate, but the specificity of issues within the scope needs elaboration to cover the concerns we believe are critical to ensuring the capacity, integrity, resiliency, availability, and security of SCI systems and the security of SCI security systems.

**61.** *Do commenters believe that it is appropriate to apply the requirements of proposed Rule 1000(b)(1) to SCI systems and, for purposes of security standards, to SCI security systems?*

**Discussion:** We believe it is appropriate to apply Rule 1000(b)(1) to SCI software. The cost of a serious operational problem can rise to 8 digits, and in extreme cases nine digits. Worse, these costs are often shared with market participants beyond the owners of the disrupted system. Consequently these requirements are reasonable and their cost can be balanced against the losses associated with the operational risks they address.

**64.** *Should the Commission require SCI entities to have a program to review and keep current systems development and testing methodology, as proposed to be required in proposed Rule 1000(b)(1)(i)(C)?*

**Recommendation:** SCI entities should initiate and maintain activities to measure the cost, quality, operational performance, and business risk of SCI software. Reviews of development and testing methods should incorporate these measures.

**Discussion:** We believe it is critical to have ongoing review of the effectiveness of system development and testing methods. Ample data exist demonstrating that more mature software development organizations continually review and adjust their development methods based on performance data and that this continual improvement is cost effective when compared to operational benefits *(J.D. Herbsleb et al., 1997, Communications of the ACM, 40(6), 30-40; B. Pitterman, 2000, Telcordia Technologies: The Journey to High Maturity. IEEE Software 17(4), 89-96)*. The most cost effective way to implement these reviews is to make them a component of a continual improvement program.

Reviews should be driven by data collected on the cost, schedule, testing, quality results, and operational performance of development and testing methods. Each new wave of development methods is touted with miraculous claims that can only be tempered with the collection and review of cost and quality performance results both during development and during operations. These reviews frequently identify circumstances under which various development and testing methods are either inappropriate or need to be supplemented by additional practices to be effective. These reviews can also identify practices that can be eliminated because they do not add benefit beyond those available from a leaner set of practices.

Testing practices in particular should be reviewed to evaluate their effectiveness in identifying defects that put the capacity, integrity, resiliency, availability, and security of SCI systems at risk. Since the cost of the testing required to assure the full capacity, integrity, resiliency, availability, and security of SCI systems is prohibitive (*D. Jackson, A Direct Path to Dependable Software, Communications of the ACM,, 52(4), 78-88*), it is critical to continually review the effectiveness of testing practices to optimize the resources spent on quality assurance against the remaining risks in SCI systems. In our response to #66 we will expand quality assurance practices beyond testing and provide an explanation of why this is necessary. Root cause analyses of critical operational problems should be included in an evaluation program to ensure that practices that either inserted or failed to detect critical defects are corrected.

**65.** *Should the Commission specify the interval at which SCI entities would be required to conduct reviews and tests of SCI systems and SCI security systems, including backup systems, to identify vulnerabilities pertaining to internal and external threats, physical hazards, and natural or manmade disasters, as provided in proposed Rule 1000(b)(1)(i)(D)?*

**Recommendation:** In regard to 1000(b)(1)(i)(D), SCI systems should be reviewed and tested prior to each release of software to be placed into production.

**Discussion:** We believe that the Commission should specify an interval for conducting reviews and tests of SCI systems. We will focus our comments on 1000(b)(1)(i)(D) "regular reviews and testing of such systems". A complete set of quality assurance activities (to be elaborated in our comment on question 66) should be conducted with each release of software into production. The history of computing is replete with examples of serious operational problems caused when testing was curtailed to meet unrealistic delivery or 'go-live' operational dates.

**66.** *The Commission requests comment on whether the testing policies and procedures requirements in proposed Rule 1000(b)(1)(i)(B), (C), and (D) would be sufficiently comprehensive to foster development of the types of testing that Roundtable panelists and commenters recommended.*

**Recommendation:** Change the word 'testing' to the phrase 'quality assurance activities' since testing alone is not sufficient to ensure the capacity, integrity, resiliency, availability, and security of an SCI system.

**Recommendation:** Indicate that as regards 1000(b)(1)(i)(D),the phrase 'quality assurance activities' includes at a minimum (1) peer reviews, (2) testing at the unit and system level, (3) static analysis at the unit and system level, and (4) dynamic analysis at the system level (stress and load testing).

**Recommendation:** The structural attributes of a system should be measured at each release of software to provide one form of evidence that a SCI system can meet its targets for capacity, integrity, resiliency, availability, and security.

**Recommendation:** The structural quality of SCI Security Systems should be evaluated and measured, since many of the security vulnerabilities in software result from poor structural quality.

**Discussion:** Since testing is only one of many quality assurance activities, we do not believe that the testing policies and procedures as described in proposed in Rule 1000(b)(1)(i)(D) are sufficient to ensure the capacity, integrity, resiliency, availability, and security of SCI systems and the security of SCI security systems. Relying on SCI entities to choose among current industry standards provides too much leeway to ensure the objectives of this rule are achieved. One problem is the definition of 'testing'. While some consider 'testing' to encompass all quality assurance activities such as formal inspections, static analysis, etc., most only consider 'testing' to encompass the actual running of test cases. We will refer to this latter interpretation as 'testing', and use the term 'quality assurance activities' when referring to the broader ensemble of defect detection activities to be elaborated in a subsequent paragraph.

A panel on 'software for dependable systems' chartered by the National Research Council of the US National Academies and headed by Prof. Daniel Jackson of MIT (*D. Jackson, et al. (2007), Software for Dependable Systems, National Academies Press, Wash. DC; D. Jackson (2009), A direct path to dependable software, Communications of the ACM, 52(4), 78-88*) determined that, "as higher levels of assurance are demanded…testing cannot deliver the level of confidence required at a reasonable cost." When systems are required to achieve the highest levels of dependability, testing costs rise exponentially to cover the exploration of increasingly rare and complex circumstances. Other more cost-effective quality assurance techniques are available to supplement testing, especially for types of defects that are difficult to detect with traditional testing methods.

Some of the limitations of testing result from test cases being developed primarily from a system's functional requirements and as a consequence they are designed to ensure that the software computes its outputs correctly. However, functional testing does not ensure that the engineering of the software, that is, the non-functional, structural attributes of the software are sound. As an analogy, the functional requirements for a bridge ensure that there are separate paths allowing cars, bicycles, pedestrians, and perhaps even trains to cross a river, while the non-functional, structural attributes ensure that the pillars and other weight-bearing elements of the bridge are sufficiently strong to support the weight of this traffic for decades, even if it grows by an order of magnitude or more. As Prof. Diomedis Spinellis, a leader in software engineering, stated in his book *Code Quality (2006, Addison-Wesley)* "…a failure to satisfy a non-functional requirement can be critical, even catastrophic…non-functional requirements are sometimes difficult to verify. We cannot write a test case to verify a system's reliability."

The NRC's panel on Software for Dependable Systems also found that, "the correctness of the code is rarely the weakest link." That is, the attributes of an SCI system most critically affecting its capacity, integrity, resiliency, availability, and security are predominantly structural (engineering) rather than functional (correctness). The majority of defects causing outages, data corruption, unauthorized access, performance degradation, and similar operational problems are structural rather than functional. If Rule 1000(b)(1)(i)(D) is to provide adequate

guidance for ensuring the capacity, integrity, resiliency, availability, and security of an SCI system, it must expand the coverage of this rule to include an ensemble of quality assurance activities that supplement testing which at a minimum include static analysis at the component and system levels.

Static analysis has become an increasingly critical technique for mission critical systems because of its power to identify structural weaknesses in software. Many software developers now use static analyzers for evaluating the structural attributes of the components they develop individually. However, the most insidious structural problems usually reside in the interactions among components in different layers of a software system and can only be detected by analysis at the system level. System level static analysis is critical because most modern business applications are constructed in multiple layers (e.g., user interface layer, business logic layer, data access layer, data storage layer, etc.). Each of these layers is usually written in a different computer language often on a different technology platform, magnifying the complexity of the system.

The complexity of modern mission-critical business systems has exceeded the capacity of any single individual or team to understand all of the structures and interactions within the system. Developers are usually expert in only one or two of the languages and technologies used in an application. Consequently they make assumptions about how the components they are building will interact with software in other layers of the system. While the majority of their assumptions are correct, those that are wrong can lead to unexpected interactions within the system that cause operational problems such as outages or security vulnerabilities. Since testing must be performed at both the component level (unit test) and system level (integration, system, and user acceptance test), static analysis must also be conducted at both the component and system levels to ensure structural soundness of a complete SCI system across its various layers, languages, and technologies.

Static analysis is also important to supplement dynamic stress and load testing. Most testing organizations complain that they do not have the resources required to adequately simulate the real operational environment. Consequently they are unable to place the stresses and loads on a system during testing that would fully verify its capacity, resiliency, and availability under actual operational conditions. However, there are known weaknesses in code that, while benign under low loads, can cause operational problems as the amount of data and frequency of processing rise. Many of these structures can be detected with static analysis long before they cause operational problems under growing operational loads.

The structural quality of a system can be measured. CISQ was formed to define standard measures for structural attributes such as reliability, performance efficiency, security, and maintainability that can be measured from the source code. Measures of structural attributes have been shown to predict problems in operations and maintenance. Consequently these measures should be collected on every release as one form of evidence that an SCI system is able to meet its targets for capacity, integrity, resiliency, availability, and security. TEH CISQ metrics, which are being prepared for submission to the OMG standards process, provide an excellent foundation for measuring the structural quality of SCI software systems. In particular,

the CISQ Security measure was developed from the top 25 weaknesses in the Common Weakness Enumeration (cwe.mitre.org), an repository funded by the Department of Homeland Security as a resource for the national software assurance community.

Formal software inspections (*M.E. Fagan 1986, Advances in Software Inspections, IEEE Transactions on Software Engineering, 12(7), 744-751*) have been proven to be one of the most effective defect detection techniques (*T. Gilb, et al., 1993, Software Inspection. Addison-Wesley*). These inspections involve members of the development team thoroughly reviewing design documents or the software to identify defects, many of which may be structural. However, newer development methods such those collected under the banner of 'agile methods' no longer use formal inspection processes in the interest of faster, more frequent deliveries. Under these circumstances the importance of static analysis, especially at the system level, is even greater. Peer reviews, even if informal, should be encouraged, but should also be supplemented by automated static analysis to provide a more thorough analysis of structural quality.

**67.** *Should the Commission require SCI entities to have, and make available to their members or participants, certain infrastructure or mechanisms that would aid industry-wide testing or direct testing with an SCI entity, such as test facilities or test symbols?*

**Recommendation:** To the extent possible, SCI entities should explore providing facilities in 'the cloud' for SCI members that simulate the loads and stresses of the operational SCI environment.

**Discussion:** As mentioned in the answer to question 66, most quality assurance organizations complain that for cost reasons they do not have sufficient resources to simulate their operational environment. To the extent possible SCI entities should make available to members or participants infrastructure or mechanisms that would aid industry-wide or direct testing to better simulate their operational environment, since such simulations are difficult to establish within member facilities. Many IT organizations are now developing test environments in 'the cloud' (mega-scale computational facilities accessed through the internet on which time and storage can be rented for temporary use) in order to more cost-effectively simulate operational environments.

**78.** Is the requirement in proposed Rule 1000(b)(1)(i)(F) for "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" appropriate? Are there other factors that the Commission should consider in determining whether standards to process data are adequate? Or, should some of the proposed standards be eliminated or modified?

**Recommendation:** Policies compliant with 1000(b)(1)(i)(F) should include a broader range of standards than those listed in Table A.

**Discussion:** Organizations rarely expend effort to comply with standards unless executive management has indicated compliance as an organizational policy for which there is visible

enforcement. However, these standards must be credible to the workforce based on their currency with generally accepted industry practice. Standards which are excessive costly, or bureaucratic can be used as guides for selecting practices. However, some form of compliance with industry best practices must be established as policy. The primary cause for failure of improvement programs has been lack of executive support. Policies are where executives state and enforce their objectives. We will list some relevant standards to be included in Table A in our response to question .

**79.** Do commenters believe there are specific internal controls or other mechanisms that would reinforce the effectiveness of an SCI entity's reasonably designed policies and procedures under proposed Rule 1000(b)(1)?

**Recommendation:** An individual who does not report to the development manager should have the final authority to approve the release of software into production based on it having completed its quality assurance process.

**Recommendation:** Review and testing practices should be periodically audited for compliance with policies and procedures by an individual(s) whose reporting relationship to senior management is independent of the development and testing organizations.

**Discussion**: Disciplined practices should be integrated into the release process for each SCI system that require an individual who does not report to the development manager to ensure that all quality assurance and release processes have been completed prior releasing software into production. This person should have the authority to stop a release from being placed into production. This independence is important to ensure that release decisions are not biased by the schedule pressures on development managers.

Enforcement of the policies governing development and testing activities should be conducted by a 'process audit' role that evaluates compliance with policies, provides guidance to development and testing teams on how to comply, and reports on compliance to senior management. These audits should be conducted at least annually for each SCI system, and more often for SCI systems with operational problems, a record of non-compliance, or those being developed, tested, or operated by an inexperienced staff. Process auditors who perform a mentoring role to software teams have proven a cost effective mechanism for on-the-job training.