CISQ Comments on Dramatically Reducing Software Vulnerabilities

1) **Section 2.3**—In the section on Additive Software Analysis Techniques does a good job of surveying much of the current static analysis technology capabilities. With most modern business applications being stacks of technologies often written in different languages and hosted on different platforms, CISQ believes it is important to stress the need for static analysis technologies to conduct system-level architectural analysis, since many critical weaknesses involve interactions among components in different layers frequently written in different languages. This challenge becomes even more important with the expansion of 'systems of systems' and the Internet of Things, where weaknesses can only be realized by analyzing interactions among components across systems. System-level technologies will be needed to detect incorrect assumptions about logic and capabilities among interacting systems.
2) **Line 1204**—You argue that most metrics have not been rigorously validated. For a controlled validation of the ability of Halstead's E and McCabe's Cyclomatic Complexity measures to predict time to find a bug, see:
   Curtis, B., Sheppard, S.B., and Milliman, P. (1979). Third time charm: Stronger prediction of programmer performance by software complexity metrics. *Proceedings of the 4th International Conference on Software Engineering*. Washington, DC: IEEE Computer Society, 356-360.
3) Section 3.4.5?—The report could use a Section 3.4.5 on Weakness-Based Measures. Although the reports hints at this possibility in earlier sections, measures based on counting, and in some cases weighting weaknesses have been developed, are in use, and standards have been built around them. An example of a community-developed repository of weaknesses is the Common Weakness Repository maintained at MITRE Corporation. The top 25 common weaknesses (22 of which can be detected by static analysis) in this repository have been used by the Consortium for IT Software Quality as the basis for defining a measure calculated by detecting these CWE weaknesses in the source code.
4) The report should reference to or perhaps a paragraph/sub-section on the ISO/IEC 25000 series which provides the international infrastructure of standards for measuring software product quality. In particular ISO 25010 provides a conceptual model of software product quality, while 25023 lists actual measures. These measures are primarily external/behavioral. CISQ has provided standards for internal/structural measures based on 25010 definitions that supplement 25023.
5) Although the report does not review processes and methods, it might include a subsection in Section 3 on using measures in a disciplined process, as is done in CMMI Level 4 or the Team Software Process. The sub-section would emphasize that the power of measures in enhanced by the discipline with which they are derived, interpreted, and used in a rigorous software engineering method.