# A White Paper on Software Assurance

Revised document – Nov. 28, 2005

## Introduction

While organizations are becoming increasingly dependent on software, targeted attacks against software are on the rise, causing harm to the infrastructure and disrupting business operations. There is a growing consensus in the industry and government that the software industry needs to actively address the root causes of exploitable vulnerabilities and implement methods to improve software resilience to attacks from the onset by identifying any known types of weaknesses in architecture, design, or implementation and addressing them, thereby enhancing **software trustworthiness**. Unfortunately, it is becoming increasingly difficult to establish or verify whether or not software is sufficiently trustworthy, due to a variety of factors:

1. *Complexity.* The size and complexity of software systems, as well as of the components from which they are built, is increasing.
2. *Interconnectivity*. Software implementations are becoming increasingly interconnected via ever-larger networks, distributing control of maintenance and providing more opportunities for attacks.
3. *Wireless*. Wireless technologies are being increasingly adopted, potentially introducing rogue elements into a network (making traditional designations of "internal" networks even less relevant).
4. *Net-centricity*. Increasing migration to "net-centric environments and service-oriented architectures," where there is a need to ensure that software components can be trusted to interact securely without supervision.
5. *Globalization*. Modern software development and supply chain support is becoming increasingly distributed worldwide and often focuses only on functionality for the lowest possible price without regard to its resistance to attack.
6. *OSS*. There is expanding interest in and use of open source software (OSS), which is often developed in significantly different ways than proprietary software. This can complicate ensuring the trustworthiness of contributing developers (i.e. pedigree of software) and challenges traditional funding mechanisms for evaluations.
7. *Hybridization*. Increasingly different software components are being interconnected that were developed using differing methodologies and under varying constraints. Increasingly, systems build on hybrid networks of COTS, GOTS, and custom software, both Open Source software (OSS) and proprietary software, all using a variety of technologies.
8. *Pace of Evolution*. Software evolves rapidly, as new technologies and supported feature sets continue to introduce ever greater levels of complexity to an already complex system of networked and autonomous software components.
9. *Lack of Knowledge*. Most software developers generally do not know how to develop software that resists attack, and it is rarely taught in formal settings.
10. *COTS/GOTS/Reuse*. There is an accelerating dependency upon commercial off-the-shelf (COTS) / government-off-the-shelf (GOTS) software components over time, driven both by economic factors, such as Time-To-Market (TTM) and development/integration costs, and by ever-changing technology requirements by acquirers.

**This justifiable trustworthiness in meeting established business and security objectives is called Software Assurance.**

As new complications to assessing trustworthiness are discovered we would continue to evolve, formalized and standardized new mechanisms/approaches to address these challenges to software assurance.

To achieve this we are going to focus on assessment transparency of product and its operating environment to:

- Establish a common framework of software properties that can be used to represent any/all classes of software so software suppliers and acquirers can represent their claims and arguments(respectively), along with the corresponding evidence, employing automated tools (to address scale)
- Verify that products have sufficiently satisfied these characteristics in advance of product acquisition, sufficient so that system engineers/integrators can use these products to build (compose) larger assured systems with them
- Enable industry to improve visibility into the current status of software assurance during development of its software
- Enable industry to develop automated tools that supports the common framework

# Strategic value of sustainable collaborative framework for Software Assurance

Achieving a breakthrough in software assurance requires collaboration of multiple stakeholders, including organizations that produce major software components, system and software integrators, acquisition organizations, certification agencies, insurance and regulators. Today's organizations rely on COTS/GOTS/Reuse/OSS products, developers from foreign and non-vetted domestic suppliers, integrated systems from mergers and acquisition activities, etc. Within this landscape, there is a growing need to either a) prolong the lifespan of existing legacy applications or b) change/improve existing applications to accommodate ever-changing market requirements and governmental regulations. Most software development efforts will depend on integration of Off-the-Shelf software, and not solely develop custom code, so there must be mechanisms for managing assurance requirements of this code.

Over time, as these software systems grow in scope and complexity, they introduce greater variability in design at the level of individual software components. As a result, component designs may conflict amongst one another or obscure functionality through complicated interfaces, subsequently hindering comprehension and reducing the ability to effectively "evolve" aging software components or respond to bugs. Ultimately, this may result in the fielding of a growing number of potentially unstable and vulnerable applications in our operational environments. The challenge for the software providers (i.e. vendors) is to:

- Ultimately prevent fielding unstable or vulnerable applications from ever happening, if possible
- Effectively manage these situations if and when they arise
- Reduce the risk to users and consumers that the software may pose in advance to deployment

The challenge for users and consumers is to understand the risks involved in deploying particular applications in their respective environments.

The challenge for regulators, auditors and insurers is to:

- Communicate regulations effectively to both software providers and software consumers
- Effectively enforce regulations and identify incidents where regulations and agreements are violated

The challenge for system integrators and acquirers is to evaluate and assess the trustworthiness of the components used to build the software system.

To address these challenges there is a need for:

- Agreement on collective vision of the necessary component-level software properties that constitute secure software, system-level properties that increase trustworthiness of a system running less trustworthy components, and properties for methods and tools that might be used to evaluate the trustworthiness of software and systems. Once firmly established, these properties are expected to serve as rudimentary "design" criteria to which
    - software providers and system integrators can engineer products that will meet the expectations for product-level assurance in advance for the acquisition and deployment of software
    - software intensive organizations and system integrators would be able to make arguments and verifiable claims to their customers that their software systems are sufficiently trustworthy
    - customers would have methods to verify those arguments and claims through collected and presented evidences
- Agreement on common terminology related to arguments, claims and evidences
- Agreement on common, structured and repeatable techniques to exchange data and information related to arguments, claims and evidences
- Agreement on interpretation of data and information related to arguments, claims and evidences (common meaning as opposed to common format)

In other words, there is a need to standardize on Software Assurance (SwA) where justifiable, sufficient trustworthiness can be measured in formal way. **This is achieved through a SwA common framework and delivered in the form of Software Assessment.**

The main benefits to be achieved as a part of the SwA efforts include:

- Secure and reliable software in its environment for the user's intended use
- Informed users and consumers demanding secure software assets and systems
- Development of tools to help build more secure and reliable software
- Better trained and educated software developers that utilize community-approved processes and tools to produce secure software

# Interoperability benefits from standardization

Over the past few years, a strong cross-section of SwA participants (software intensive organizations, users, consumers and regulators) has emerged with the objective of promoting software assurance within the community. Unfortunately – until now – these participants have been working mostly in isolation. Future standardization of SwA concepts and methodologies will build upon a combination of prior experiences, domain knowledge, and best practices, and will ultimately facilitate interoperability for the exchange of information among community participants. This will allow:

- Different participants to initiate collaboration and activities in areas of SwA through a common assurance meta-model framework
- Enabling of a new generation of supporting solutions that benefit all participants and

- Enhancement/improvement in automation of SwA activities by enabling interoperability between different supporting solutions (toolsets)

SwA specifications and corresponding solutions will enable projects with mission critical software portfolios to build secure and reliable products and produce evidence needed for users and consumers to have confidence in the products they leverage to support their mission objectives.

**Standardization will ensure that all participants are investing not just in individual activities but through coordinated strategy.**

# Software Assurance Framework

The common SwA framework will be built on prior experiences and best practices while utilizing existing related OMG specifications. The framework is expressed in terms of a SwA metamodel (SAM). The SAM is intended to facilitate the free exchange of information among SwA community participants – notably between software consumers and software providers. The community will collaborate on defining the structure of the SAM, which will allow for the establishment of a common repository structure that can be used to universally represent information regarding vendor claims, arguments, and evidence related to software assets and their operational environments. More specifically, the SAM provides the ability to document:

- Existing profile of software assets including associated data about the operational environments in which they operate and any relevant vendor-issued assurance claims
- Industry standardized Reference Models (including protocols and engineering requirements) explaining its intended use and requirements, enabling suppliers and acquirers to communicate and rigorously validate agreements, and
- The comparative difference between the existing profile and the industry standardized Reference Models.

The meta-model will also enable this information to be exchanged among different tools, which will enable vendors that specialize in certain languages, platforms or particular types of software systems to deliver solutions in conjunction with other vendors. The meta-model is not restricted to any particular implementation language, platform or specialized system and is designed to be flexible enough to correlate against all variations of software components at any level of functional abstraction.

The SAM based models will likely contain large amounts of information, which will be difficult to manually manage for the essentially infinite variance among potential software component extensions. To overcome such a roadblock, models will support the capability to aggregate (summarize) information to different levels of abstraction. This requires the SAM to be scalable. In addition, the SAM will represent both **primary** and **aggregate** information. Primary information is assumed to be automatically extracted from the system itself through source/binary code analysis, reference models and/or manually entered by analysts or experts. Aggregate information is extrapolated from primary information. Aside from scalability, the models should be both composable and actionable. This way the information can be accessed at progressively deeper levels that reflect the varying mission or business-level objectives that need to be achieved. Mechanisms for dealing with proprietary data, including mechanisms to release aggregate data while protecting primary data in some cases, will need to be identified. During development of this framework, a few specific and justifiable aggregate metrics will be identified and standardized, along with specific mechanisms to measure them. These specific aggregate standard metrics, which need to be useful for decision-making, will demonstrate the utility of the overall framework.

# Software Assurance Metamodel (SAM)

Metamodel will include (as a minimum) the following information:

- Documented software assets "as they are" with their operational environments
  - Software assets knowledge – Static Analysis techniques could be used to automatically extract required information from software itself
    - ♣ Knowledge Discovery
      - Software artifacts related to Structure, Architecture, Behavior, Data …
    - ♣ Analysis performed on extracted knowledge
      - Findings related to: Reliability, Robustness, Security, Presence of Known Types of Weaknesses, ... (e.g., detailed information on the breadth and depth of input validation, strength of defense-in-depth / defensive programming measures, security test coverage, or fuzz testing depth)
    - ♣ Metrics and trends extracted from the system
      - Design, Security, Complexity, Sustainability, Risk metrics, Freedom from Known Types of Weaknesses,...
  - External information about software assets
    - ♣ System requirements: security, architecture, functional …
    - ♣ Environment (description) under this system runs, including the accepted boundaries of the system (e.g., resources and protocols required by the system, so that users can automatically enforce these boundaries knowing that the software will never exceed them in normal operation.)
    - ♣ Claims about system (against security target requirements)
    - ♣ Process (description) under which this system was developed
- Documented industry standardized reference models relevant to this system "as they should be" and operational environments that they will run under
  - Security reference model
  - Architectural reference model
  - Quality reference model
  - Common Weakness Enumeration
  - Functional requirements …
  - Environment where system will be deployed (description)
- Software Asset Assessment Results - compares the existing profile with the industry standardized Reference Models to produce results in the following terms
  - Trustworthiness based on the following information
    - ♣ Found weaknesses and faults using a variety of robust tools and processes
    - ♣ Risks (including probability and consequences) that these faults present
    - ♣ Are there solutions that could be deployed to mitigate given risks
    - ♣ Design robustness, validation of design and resiliency to corruption
    - ♣ Process integrity and configuration control
    - ♣ Understanding of environmental dependencies and relationships

# Why is the Object Management Group the best place to develop this standard?

The breadth of technical backgrounds needed for such a comprehensive SwA framework can be found in Object Management Group (OMG).

The OMG is an open membership, not-for-profit consortium dedicated to producing and maintaining specifications for interoperable enterprise applications. The OMG membership roster includes many of the most successful and innovative companies in the computer industry, as well as those at the forefront of using technology to gain a competitive edge in their business. All have made the commitment to actively participate in shaping the future of enterprise, Internet, real-time, and embedded systems.

OMG has developed some of the industry's best-known and most influential specifications,
The Model Driven Architecture (MDA) group within OMG builds on these successes, providing a comprehensive interoperability framework for defining the interconnected systems of tomorrow. The Architecture-Driven Modernization (ADM) Task Force was established to develop specifications related to modernization of existing software systems. This activity is often referred to as "*MDA-in-reverse*", as it addresses the need to apply modeling techniques to the software products that are already in production for the need of understanding, evaluation, assessment, certification, or modernization. ADM techniques reach new frontiers in the area known as software understanding.

Software Assurance Framework will build upon the foundation of existing OMG specifications.

---

White Paper Development, Contributors and Review Team

Djenana Campara, Klocwork
David Wheeler, Institute for Defense Analysis
Rama Moorthy, Booz Allen Hamilton
Mitchell Komaroff, OASD(NII)
Patrick Holley, Booz Allen Hamilton
Nikolai Mansurov, Klocwork
William Ulrich, TSG
Joe Jarzombek, DHS
Samuel T. Redwine, James Madison University
Michael Kass, NIST
John Barkley, NIST
Liz Fong, NIST
Paul Black, NIST
Larry Wagoner, NSA
Paul Croll, CSC
Ken Hong Fong, OUSD
Charles Johnson, DAC
Robert Martin, MITRE Corporation
Austin Maher, BAE Systems
Sean Barnum, Cigital