Avoiding Proprietary Problems

A Software Clean-Room Method

Don O'Neill

Heads up! With 80 percent of government software procured as commercial off-the-shelf (COTS) and accorded limited or restricted rights, government acquisition managers need to be alert to intellectual property considerations. When modified and extended through government funding, COTS software becomes government off-the-shelf (GOTS) software entitled to government purpose rights. Unless the government acquisition manager insists on it, a contractor may engage in false claims practice by improperly marketing and selling GOTS software products as COTS. So instead of receiving the benefits of government purpose rights, the government may be charged a commercial product licensing fee and accorded only limited or restricted rights. Neglecting intellectual property rights can be costly!

O'Neill *was president of the Center for National Software Studies (CNSS) from 2005 to 2008. Following 27 years with IBM's Federal Systems Division (FSD), he completed a three-year residency at Carnegie Mellon University's Software Engineering Institute (SEI) under IBM's Technical Academic Career Program and has served as an SEI Visiting Scientist.*

Clean-Room Software-Engineering Summary

Proprietary information may taint a software product. This can occur when commercial off-the-shelf (COTS) software for which a commercial fee is paid for each use is modified or extended through government funding and becomes government off-the-shelf (GOTS) software entitled to government purpose rights following a one-time commercial fee. The difficulty arises when the contractor engages in false-claims practice by improperly marketing and selling GOTS software products as COTS, charging a repetitive commercial fee for each use.

"Clean-room" involves copying a design by reverse engineering, followed by redeveloping the code without infringing on the copyrights and trade secrets present in the original. In an effort to return the software to a permissible fee-bearing commercial off-the-shelf (COTS) status, a vendor may choose to develop a clean-room version free of proprietary information; hence, the need for a rigorously defined clean-room method to transform a proprietary-laden dirty system into a provably correct proprietary-free clean system, one convincingly devoid of reliance on proprietary information, copyrighted material and trade secrets—and not considered a derived work.

Clean-room software engineering entails the reengineering of the dirty system beginning with the production of a proprietary-free hand-over specification and its review by a lawyer to assure no proprietary information, copyrighted material or trade secrets are included or relied upon. The clean-room software-engineering team then prepares proprietary-free artifacts associated with functional specification, usage specification, increment planning, correctness verification, usage modeling, test planning, statistical testing and certification. The kernel of clean-room software-engineering assurance is trusted software engineering using structured programming with its rigorous and provably correct use of zero-and-one predicate prime programs along with proper programs composed of multiple prime programs limited to single entry and single exit.

Framing the Issue

Government acquisition managers sometimes overlook intellectual property considerations. GOTS products are often the result of COTS products extended, expanded and upgraded under government funding to operate in new and changing environments. The GOTS products may even be entered into a company's parts number database. As a result, these GOTS products may contain proprietary information, copyrighted material and trade secrets that may serve to limit or restrict the future use of the GOTS products. In effect, proprietary information, not unlike malware, may taint a software product.

Why is that a problem? COTS products represent more than 80 percent of government software procured. The originating commercial organization may attempt to restrict use of the proprietary-based COTS product and even an enhanced GOTS product in order to assert a competitive advantage in a downstream procurement with the objective and intended outcome of locking in a sole-source contract. Two software assurance challenges present themselves. The first challenge is how to detect proprietary information, copyrighted material or trade secrets. The second challenge is how to convincingly assure the clean provenance of GOTS products in such an environment.

Government systems and components may contain proprietary information, copyrighted material or trade secrets that serve to limit or impede use or reuse. A contractor may unintentionally drift into using such tactics only to find that it can exploit and leverage its proprietary information in later procurements and then attempt to do so. Beyond that, and perhaps less likely, a contractor may intentionally and stealthily seed proprietary information in systems and components only to reveal the proprietary presence later for the purpose of locking in its solution for future work. Government-funded systems and components also yield government-owned proprietary systems and components that contractors may attempt to package and resell as their proprietary products on the global market. Upon detecting a dirty system, a contractor may choose to shed the restriction by redeveloping the dirty GOTS software into a clean system. This can be done by employing a rigorously defined "clean-room" method to transform a proprietary-laden dirty system into a proprietary-free clean system, one devoid of reliance on proprietary information, copyrighted material, or trade secrets and not considered a derived work. Such a method employs both a "Chinese wall" protocol of separation and the clean-room software engineering technology and process.

To avoid false claims charges, a contractor is advised to segregate core commercial software components produced by private funding from those produced through government funding and to do so at the lowest practical level.

There are many questions that a government acquisition organization needs to ask and answer to begin focusing on its intellectual property practices. To what extent are COTS products enhanced with government funds resulting in GOTS products? When COTS products are enhanced with government funds, resulting in GOTS products, is it the contractor's practice to enter the GOTS product into the parts number database or to append its own copyright? To what extent do such practices go undetected? To what extent are systems and components trusted with respect to contractor proprietary information, copyrighted information or trade secrets that could limit or impede use or reuse? To what extent are systems and components trusted with respect to government-funded proprietary information that could result in unauthorized release of GOTS

as contractor-proprietary products? To what extent do systems or components thought to be GOTS have restrictions on downstream use or reuse? To what extent should proprietary information concerns be included in the approach to trusted systems and networks and their supply chain risk-management (SCRM) assurance? To what extent should the National Institute of Standards and Technology Special Publication 800-161 SCRM Plan address the identification and risk mitigation of government systems and components that may contain proprietary information, copyrighted material or trade secrets that limit or impede use or reuse, lock in contractor solutions for future work or facilitate the packaging and reselling of GOTS as contractor-proprietary products?

False Claims Violations

An organization may cross the line and be guilty of false claims violations when it markets, sells, deploys or delivers a version of its product produced by mixed funding, company investment and government funding as commercial software with limited or restricted rights—thereby depriving the government agency of the government purpose rights it may have purchased and deserved. In an environment of ascending demand for a software product, a company may commit numerous false claims violations during rollout when it improperly markets, sells, deploys or delivers such a software product as a commercial product with limited or restricted rights rather than properly as a noncommercial software product with government purpose rights.

Faced with a mixed funding history in a software product, a company may elect to produce a commercial version by re-engineering the software product in question using the clean-room method and process. Once a clean-room project has been undertaken, the object of marketing and selling shifts to the clean-room version of the product under development with a future delivery date, not subject to charges of false claims violations. In effect, the company is insulated from charges of false claims violations during the window of "under development." Since clean-room reengineering is challenging and difficult, the clean-room project schedule may slip and become extended, exceeding the original estimate and plan, thereby setting up a dilemma for the company facing firm delivery commitment deadlines to customers performing in mission-critical operations. A company that has made a commitment to deliver a commercial product with limited and restricted rights and that is faced with an incomplete clean room may decide it has no alternative but to deliver the dirty system, the product of mixed funding, and may do so without reverting to government purpose rights—a false claims violation. Of course, it takes two to Tango and such an outcome must involve the company's intent and the government acquisition contract officer's and program manager's neglect of due diligence in accepting and relinquishing data rights asserted by the contractor.

In order for the clean-room window "under development" to insulate the company from numerous charges of false claims violations, the clean-room method and process must be bona fide and legitimate—that is, the clean-room method and process must ensure an environment and operation devoid of any use or knowledge of proprietary means or methods used in a predecessor implementation.

The Need for a Clean-Room Method and Process

Rigorously defined clean-room method and process are needed to transform a proprietary-laden dirty system into a provably correct proprietary-free clean system—one convincingly devoid of reliance on proprietary information, copyrighted material or trade secrets and not considered a derived work; one with methods of investigating legitimacy, confirming intent and wherewithal of people, verifying process execution and validating outcomes in determining that a legitimate clean room was in operation—nd one with an outcome based on trusted software engineering principles and practices in producing provably correct software components.

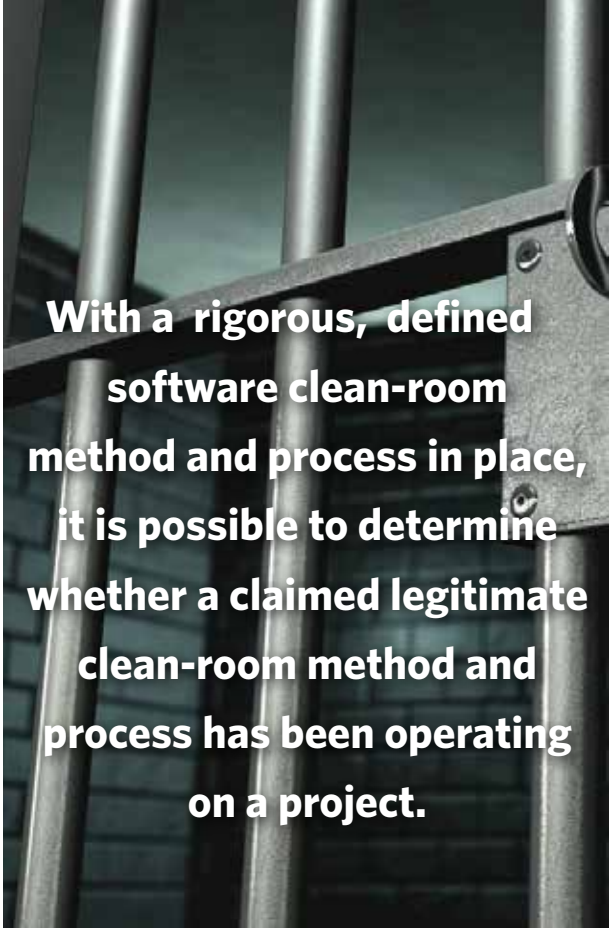
Goal

The clean-room method and process are intended to assure an environment and operation devoid of any use or knowledge of proprietary means or methods used in a predecessor implementation. A Chinese wall is used—and management, specification, development and certification personnel are involved. Clean-room method and process assurance encompass an explicit statement of intent and adherence to specified practices. These include intellectual property practices, protocol of separation, clean hand-over specification process, clean-room software engineering process and software clean-room investigation process. These practices form the basis for the

assurance, assessment, examination and investigation of an organization's clean-room method and process spanning confirmation through people, process execution-based verification and outcome-based validation.

The essential focus of a software clean-room investigation revolves around the following questions:

- Was there clean project access to the target code comprising the direct expression of the copyright material, and was there substantial similarity exhibited by the clean system?
- Does the clean system stand up to the abstraction-filtration-comparison (AFC) test, a three-step process for determining substantial similarity of the nonliteral elements of a computer program?



Specification

Defined software clean-room methods and processes are needed to transform a software system or component based on proprietary information, copyrighted material or trade secrets to a functionally equivalent clean software system or component devoid of any traces or reliance on the proprietary information, copyrighted material or trade secrets. The proprietary system is termed the dirty project and the transformed system is termed the clean project. The challenge is to insulate the clean system from the dirty system so it will not be considered a derived work in form or function. What are the criteria for a legitimate clean room? A rigorous, defined software clean-room method employs both a Chinese wall and the clean-room software-engineering technology and process.

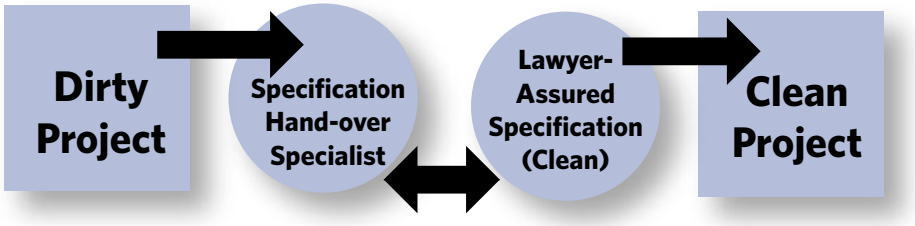
Defined Clean-Room Method

The method features a multidimensional Chinese wall spanning defined protocols of separation for physical location, people, electronic infrastructure and software development tools. The Chinese wall is composed of a clean environment demonstrably uncontaminated by any proprietary information or knowledge of such through a discipline of multidimensional separation.

The clean environment begins with physical separation in a separate, undisclosed location. Separation extends to the personnel participating in both the dirty project and the clean project, including their training and organizational policy governing the rules of separation and no contact. Separation extends to the electronic infrastructure including e-mail and telephone access and online document access and to the software development tools employed, including text editors, programming language, language compilers, test suites, configuration management tools and parts number database management tools.

The clean-room software-engineering process extends the discipline of multidimensional separation through discrete teams for management, specification, development and certification of the clean system. Clean-room software-engineering entails the reengineering of the dirty system beginning with the production of a proprietary-free hand-over specification and its review by a lawyer to assure no proprietary information, copyrighted material or trade secrets are included or relied upon (see Figure 1). Clean-room software engineering teams prepare proprietary-free artifacts associated with functional specification, usage specification, increment planning, correctness verification, usage modeling, test planning, statistical testing and certification. The kernel of clean-room software-engineering assurance is trusted software engineering using structured programming with its rigorous and provably correct use of zero-and-one predicate prime programs along with proper programs composed of multiple prime programs limited to single entry and single exit.

Figure 1. Specification Hand-Over Procedure Involving Hand-Over Specialist and Assurance Lawyer



Outcome-Based Validation

Improper use of proprietary software involving proprietary information, copyrighted material and trade secrets increasingly goes undetected. Uncovering such use and detecting specific instances of substantial similarity is a technical challenge that usually requires full and ready access to the dirty system source code for best results as well as the wherewithal and means to express the proprietary information, copyrighted material and trade secrets in a precise, rigorous and trusted abstract manner suitable for computer searching and comparison.

Proprietary software is licensed under exclusive legal right of the copyright holder with the intent that the licensee is given the right to use the software only under certain conditions and restricted from other uses. In the legal community, the AFC test is a three-step process for determining substantial similarity of the nonliteral elements of a computer program. Abstraction distinguishes which aspects of the program constitute its expression, and which are the ideas. Filtration removes from consideration aspects of the program that are not legally protectable by copyright—such as elements associated with efficiency, external factors and the public domain. Comparison considers whether just those aspects of the program that constitute its expression and not those aspects not legally protected by copyright are present in the clean system.

In addition, proprietary information, copyrighted material, and trade secret detection can potentially be determined using NIST's approximate matching text strings to detect

Table 1. Software Clean-Room Investigation Process: Confirmation

Confirmation	Software Clean-Room Investigation Process
Intellectual Property Practices	1. Had the original commercial off-the shelf (COTS) product been entered into the parts number database earlier? Was it assigned a unique number? What was that number? 2. Had the dirty system been entered into the parts number database earlier? Was it assigned a unique number? What was that number? 3. Was the clean system entered into the parts number database? Was it assigned a unique number? What was that number? 4. Was a copyright notice appended to the original COTS product? What was the copyright notice? 5. Was a copyright notice appended to the dirty system product? What was the copyright notice? 6. Was a copyright notice appended to any open source software used? What was the copyright notice?
Statement of Intent	1. Did the project have a clean-room process definition? 2. Was there an explicit management commitment to follow the defined process? 3. In actual practice, was the defined process followed? 4. Did the clean-room process definition include protocols of separation, clean-room software-engineering process, clean hand-over specification process? 5. Was the result a clean system?
Protocol of Separation	1. Was a protocol of separation defined? 2. Was there an explicit management commitment to follow the defined protocol of separation? 3. In actual practice, was the defined protocol of separation followed? 4. Did the defined protocol of separation include physical location, people, electronic infrastructure and development tools?
Clean Hand-Over Specification Process	1. Was there a defined clean hand-over specification process? 2. Was there an explicit management commitment to follow the defined clean hand-over specification process? 3. In actual practice, was the defined clean hand-over specification process followed? 4. Did the clean hand-over specification process include having a lawyer review the clean hand-over specification to assure that no proprietary information, copyrighted material or trade secrets were included? 5. Did any clean system personnel ever have any access to any dirty system code?
Clean-Room Software-Engineering Process	1. Was the clean-room software-engineering process defined? 2. Was there an explicit management commitment to follow the defined clean-room software-engineering process? 3. In actual practice, was the defined clean-room software-engineering process followed? 4. Did the defined clean-room software-engineering process include functional specification, usage specification, increment planning, correctness verification, usage modeling, test planning, statistical testing and certification?

fragments. More advanced, Carnegie Mellon University's function extraction for abstracting intended function and Oak Ridge National Laboratory's Hypernion using behavior specification units (BSUs) for detecting intended function offer promise in this space. The Defense Advanced Research Projects Agency's Mining and Understanding Software Enclaves (MUSE) program incorporates a continuously operating specification-mining engine to conduct deep program analysis on the corpus of software drawn from the hundreds of billions of lines of open-source code to identify and understand deep commonalities.

Operations and Risks

With a rigorous, defined software clean-room method and process in place, it is possible to determine whether a claimed legitimate clean-room method and process has been operating on a project. Numerous clean-room method and process risks must be assessed. Does the organization explicit commitment match the intent and means employed? Do the means employed match the defined software clean-room method and process? Does the protocol of separation ensure verifiable separation under all circumstances of use? Do the actual organization intellectual property practices reveal the organization's intellectual property ownership intentions? Is the clean hand-over specification process with lawyer-assured clean specification confirmed through people and verified through process execution evidence? Is the clean-room software-engineering process verified through process execution evidence? Are clean-room method and process execution outcomes validated through clean product results achieved devoid of proprietary information? Was there clean project access to the target code comprising the direct expression of the copyright material? Was there substantial similarity to the target code exhibited by the clean system?

Conclusion

With the rigorous, defined software clean-room method and process specified, the question of whether a legitimate clean room was in place and operating can be addressed by confirming the equivalency of the intent and means employed, verifying the extent to which the defined protocols of separation were practiced, validating the clean-room software-engineering process execution and outcome with respect to convincingly achieving the intended result of a proprietary-free clean system, and reporting the results in terms of findings, rationale, recommendations and consequences.

Confirmation that a software clean-room investigation process was undertaken begins with obtaining answers to the pertinent questions (see Table 1). Other more probing questions focus on verification through process execution and validation through outcomes achieved.

The author can be contacted at **oneilldon@aol.com.**

SECTION 3685, TITLE 39, U.S.C. SHOWING OWNERSHIP, MANAGEMENT, AND CIRCULATION

Defense AT&L is published bimonthly at the Defense Acquisition University, Fort Belvoir, Va. 22060-5565. The university publishes six issues annually. The director of the DAU Press is Randy Weekes; the managing editor of *Defense AT&L* is Benjamin Tyree; and the publisher is the Defense Acquisition University Press. All are colocated at the following address: Defense Acquisition University, Attn: DAU Press, 9820 Belvoir Rd., Ste. 3, Fort Belvoir, VA 22060-5565.

Average Number of Copies Each Issue During The Preceding 12 Months

A. Total number of copies printed (net press run): 3621
B. Paid and/or requested circulation: 3513
   1. Sales through dealers and carriers, street vendors, and counter sales: 0
   2. Mail subscriptions paid and/or requested: 3513
C. Total paid and/or requested circulation: 3513
D. Free distribution by mail, carrier, or other means; samples, complimentary, and other free copies: 75
E. Total distribution: 3588
F. Copies not distributed: 33
G. Total distribution: 3621

Actual Number Copies of Single Issue Published Nearest To Filing Date

A. Total number of copies printed (net press run): 3607
B. Paid and/or requested circulation:
   1. Sales through dealers and carriers, street vendors, and counter sales: 0
   2. Mail subscriptions paid and/or requested: 3491
C. Total paid and/or requested circulation: 3491
D. Free distribution by mail, carrier, or other means; samples, complimentary, and other free copies: 70
E. Total distribution: 3561
F. Copies not distributed: 46
G. Total distribution: 3607