

The Test and Evaluation Challenges of Following an Open System Strategy

by

Cyrus H. Azani
Senior Systems Engineer
Information Technology and Systems Group
TRW Corporation

ABSTRACT

The open system strategy is an effective business and technical approach for maintaining the superiority of the U.S. within growing constraints and an unprecedented rate of technological change. By following an open system strategy in the acquisition of systems, the government can better position itself to leverage private sector investments made in commercial products, practices, and technologies to field superior capability more quickly and affordably. This paper elaborates on distinctions between closed and open systems and will discuss strategies for implementing open systems. The paper also reviews and discusses the test and evaluation challenges associated with open systems and proposes a number of critical developmental issues as a checklist to supplement the information gathered by testers. The paper also emphasizes that the test of openness for the interfaces within a system must only be done when operational and developmental requirements either directly or indirectly require open system implementation and the use of open standards for selected interfaces within a system. The paper also underscores that the test of openness must only be initiated after careful review of testing issues and challenges involved, and after it has been proven that the benefits of openness testing are greater than the costs.

I. INTRODUCTION:

The 21st century is characterized by unprecedented change. Change is not simply a possibility, or even a high probability, but a virtual certainty. Although ideological differences will remain as a primary source for conflict with the third world, competition for scarce resources and markets will perhaps become the most dominant source of conflict throughout the world. Proliferation of weapons of mass destruction and use of information and biological warfare add new dimensions to evolving conflicts and pose a continuing challenge to the United States. The benefits of Open Systems (OS) such as more affordability, improved performance, and increased portability and interoperability, would enhance the U.S. capability, shorten the length of engagement, and ensure the mission success in light of evolving threats and technologies.

What is an Open System?

An open system is a system that can exchange energy, material and information with its environment on a continuing basis. Such exchange is enabled through the use of open (i.e., well defined, widely used and consensus based) standards, protocols, languages, and data formats in developing systems. The focus of attention in an open system is on key interfaces. An interface is designated as key interface when the technology turnover is rapid and design risk is high on either side of the interface, and/or the system elements on one or both sides of the interface exhibit a high failure rate or are very expensive. Use of an open standard is the preferred method for implementing a key interface.

The key interfaces have a profound impact on:

- ability to add new capabilities through planned and unplanned incremental improvements;
- capacity and flexibility to integrate entities and enable commonality, portability, and interoperability;
- capability to replace items with high replacement frequency and cost.

The open systems concept originated in the biological sciences and then migrated into physical and social sciences in the early parts of the 20th century. In the late 1960s and early 1970s, the concept began to be applied in commercial information technology. For many years, information systems buyers were limited to only a few major mainframe vendors, with one vendor clearly dominant in the marketplace. Competition was severely limited because a few –and sometimes one – vendor controlled access to the market. A number of different standards organizations initiated open system efforts, sometimes in competition with each other. Recently, some order was injected into the scene because of more standardization and some degree of convergence appears reasonable.

Even though the open system concept has been used by the commercial sector for sometime, it has only lately been embraced by the C4I and weapon system communities, the most important entities and one of the biggest expenditure categories in the Department of Defense (DoD). Bureaucratic structures and inflexible cultures are the main reasons for slow application of OS within government institutions. Also, the potential OS practitioners at DoD did not have access to a well-established body of knowledge on OS and consequently were not able to determine the appropriateness of OS to systems that are not defined as information systems. As a result, in 1994, the DoD leadership chartered the Open Systems Joint Task Force to establish the needed body of knowledge and promote the application of the OS in development and design of new systems.

II. OPEN VERSUS CLOSED SYSTEMS

There are considerable differences between open and closed systems. The closed system is characterized by closely held, privately owned standards, protocols, languages, and data formats that are either unavailable to outsiders or are available only at a very high license fee. Closed systems also include those that were designed by a single company for a single program, or small number of programs. In contrast, an *open* system is a system designed using a collection of interacting and integrated software, hardware and human components that are based on consensus-based, *de jure* or if not available *de facto* standards that are easily accessible to all interested parties. Table 1 summarizes the major differences between closed and open systems.

Closed System Characteristics	Open System Characteristics
Use of closely held, private interfaces, languages, data formats and protocols (government or vendor unique standards)	Use of publicly available and widely used interfaces, languages, data formats and protocols
critical importance is given to unique design and implementation	critical importance is given to interfaces management and widely used conventions
less emphasis on modularity	heavy emphasis on modularity
vendor and technology dependency	vendor and technology independence
minimization of the number of implementations	minimization of the number of types of interfaces
difficult and more costly integration	easier and more cost effective integration
difficulty with portability, connectivity interoperability and scalability	high degree of portability, connectivity, interoperability, and scalability
use of sole-source vendor	use of multiple vendors
expansion and upgrading usually requires considerable time, money and effort	easier, quicker and less expensive expansion and upgrading
higher total ownership cost	lower total ownership cost
slower and more costly technology transfer	technology transfer is faster and less costly
components, interfaces, standards, and implementations are selected sequentially	components, interfaces, standards, and implementations are selected interactively
systems with shorter life expectancy	systems with longer life expectancy
use of individual company preferences to set and maintain specifications	use of group consensus process to maintain interface specifications
less adaptable to change in threats and technologies	more adaptable to evolving threats and technologies
focusing mostly on development cost and meeting present mission	focusing on total costs of ownership, sustainment, and growth
user as the producer of systems	user as the consumer of components
rigid and slow system of influence and control	real time and cybernetic system of influence and control
adversarial relationship with prime contractors/suppliers/vendors	Symbiotic relationship with prime contractors/suppliers/vendors
mostly confined to traditional suppliers	non-traditional suppliers can compete
simple conformance testing	very challenging conformance testing

Table 1: Open versus Closed Systems

III. THE NEED FOR AN OPEN SYSTEM STRATEGY

A. What is an Open Systems Strategy?

An OS strategy is an integrated business approach and design method that relies on sound systems engineering processes and continuing market research to evaluate alternative concepts and if appropriate, develop systems architectures based on modularity principles and well defined and widely used consensus-based interface standards, protocols, languages, and data formats. An OS strategy is an effective enabler for achieving rapid acquisition with demonstrated technology, evolutionary and conventional development, interoperability, life-cycle supportability, and incremental system upgrade without redesign of an entire system or large portions thereof. OS also enables continued access to innovative technologies and products from multiple sources, and prevents the buyers from being locked into proprietary technology.

An OS strategy is usually implemented by an Integrated Product Team (IPT) which besides conducting market research also:

- ◆ Applies a disciplined systems engineering process that examines open versus closed system tradeoffs;
- ◆ Defines key interfaces and establishes optimum level(s) of openness for the system to be acquired/modernized and devises a strategy to achieve it;
- ◆ Develops modular open architectures that conforms to standards adopted by recognized standards organizations, or when not effective, to de facto standards, and;
- ◆ Ensures continued access to technological innovation supported by many customers and a broad industrial base.

The OS strategy is an effective strategy for adapting to the current pattern of change brought about by rapid technological advances and the pervasive globalization of economies, markets, and conflicts. The new pattern of change has business and engineering dimensions reinforcing each other and creating a paradigm shift greater than the sum of its parts. By following an OS strategy, an organization will be in a better position to reduce its total ownership costs in the following areas:

- (1) Research and Development: At least some of the required subsystems or components are likely to be readily available, or can be developed without direct government investments;
- (2) Production: There are multiple sources of supply to select from which may also mitigate the problem associated with a diminishing defense dependent manufacturing base;
- (3) Operation and Support: The required level of openness shifts the burden of continual improvement and repair to the supplier rather than the user of products and technologies.

From an engineering perspective, by following an OS strategy an organization establishes a flexible, modular, and open architecture, which will last longer and be subject to less obsolescence. An important task undertaken in developing a standards-

based architecture is the selection of conventions (standards, protocols, languages, and data formats). Performance, cost, security, privacy, long term availability and supportability, upgrade potential, and openness are examples of criteria used for selecting these conventions. Preference is always given to the use of open conventions first, then de facto, and finally proprietary and government conventions. Chart 1 depicts the preferred type of conventions (standards, protocols, language, and data formats) to use.

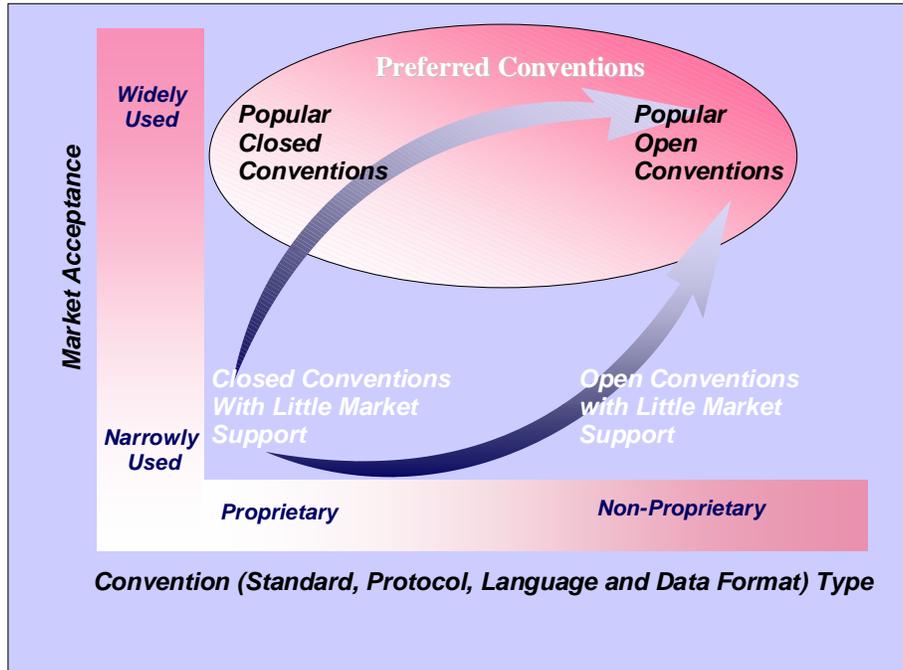


Figure 1 Preferred Conventions

B. Types of Open System Strategies

Generally speaking, one can follow two strategies for designing and implementing open systems, top-down and bottom-up. Traditional practice in the development of systems has been to develop systems from the top down, where high level requirements were analyzed, partitioned, and allocated to hardware and software elements. The need to satisfy demanding performance requirements in harsh environments usually led to unique and often proprietary designs. Taking advantage of design from similar applications or commercial products was rarely practiced in the development of systems. A top-down OS strategy (Figure 2) also applies top-down system development approach but designs systems flexibly to take advantage of commercial products and technologies.

By following a top-down OS strategy, the organization establishes an overall implementation/deployment plan for OS implementation, sets priorities for applications, constitutes an enterprise-wide policy for development, and establishes a list of preferred key interfaces that must remain open to enable exchange of information and products. Appointment of a corporate champion to promote the concept, development of an

enterprise level OS architecture and directing a detailed development and deployment plan for implementation of OS are among other tasks associated with a top-down OS strategy. The underlying assumption of a top-down OS strategy is that the corporate top executives are in a better position to understand the business model, system of system requirements, and the overall cost constraints.

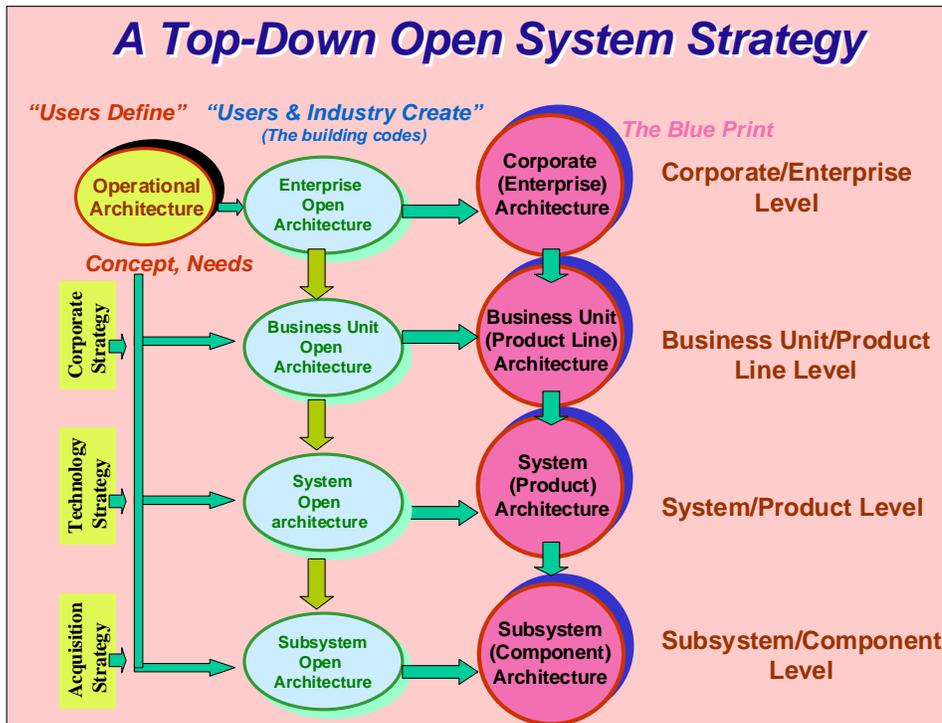


Figure 2: Top-Down Open System Strategy

The implementation of open systems through a top-down strategy may prove to be more direct and efficient since policies, procedures and the selection of a particular subsystem/component for transition originates from the corporate/enterprise level. This approach integrates complex development efforts with uniformity and economy of scale, but constrains rapid development and local innovation at the end-user level.

Establishing and implementing the OS strategy from the top requires highly technical expertise at the corporate headquarter, especially if the business entity is comprised of many different subsystems operating in various environments. Often, the technical expertise lies outside the corporate/enterprise level and thus the selection of the OS champion becomes an insurmountable task for system/product level engineers and managers to accept. Lower level engineers/managers may think that there is politics involved with the open system mandates from the top. They may also believe that the selected open standard, protocol, language, or data format may not be appropriate since the subsystems/components have to adopt to architectures that are proposed by someone else who may not be familiar with needs and constraints at the lower level.

To overcome the above-mentioned drawbacks of a top-down strategy, an organization may follow a bottom-up OS strategy that relies on inputs and wisdom at the lower level to develop and deploy open systems throughout the organization. Figure 3 depicts the upward flow of information in a bottom-up OS strategy.

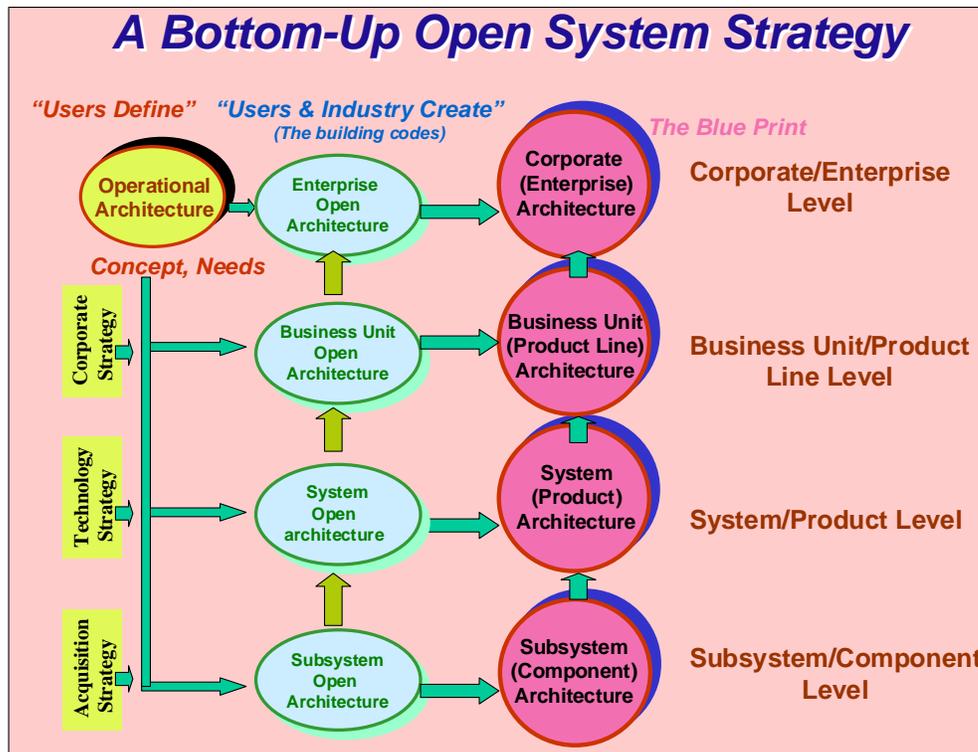


Figure 3: Bottom-up Open System Strategy

A bottom-up strategy for implementing OS is usually initiated from the people/programs at the lower end of the organizational hierarchy. The driving force for adoption of an open system strategy is common sense and an immediate-felt need by experienced system engineers. They want to develop a viable and life-long system and be able to upgrade the system, as new technologies become available. They use COTS software or hardware to reduce the overall development cost and want to take advantage of competition to get the best value for their organization. So in the absence of mandates from the top, they initiate OS feasibility studies and begin to use widely supported and well-established standards for selected interfaces within the system. As the benefits of using open systems are realized, the lessons will then be shared with other programs/systems both laterally and vertically which will result in application of open systems in other places in the organization. With this strategy, the OS benefits such as reduction in the overall cost, higher conformity, better system of systems interoperability and commonality/reuse may not be realized in a timely fashion.

Bottom-up OS strategies may be proven to be more robust than top-down strategies, especially if the architects/engineers possess sound systems engineering skills

and adequate understanding of the OS concept. A bottom-up approach encourages rapid and innovative development of open architectures at the subsystem or user level. The engineers/architects who work at these levels have a better understanding of the technical requirements and what works or doesn't work within their subsystem. While fast and effective for users, this approach leads to duplication of effort and lack of uniformity among similar subsystems, compromising cost-efficiency. Moreover, the subsystem/component level implementers may lack a broader understanding of the mission, overall cost constraints, and the required system of system interoperability.

By following common sense and sound systems engineering principles, one soon recognize the need to follow a **balanced OS strategy**. Such a strategy is built upon the advantages of both a top-down and a bottom-up strategy. Inputs from the lower levels as well as from the suppliers and customers are gathered and analyzed to create a shared OS vision and a well-thought deployment plan for the organization. A balanced OS strategy will take advantage of prior lessons learned and will establish organization-wide policies and processes to implement open systems. Figure 4 shows the upward and downward flow of information in a balanced OS strategy.

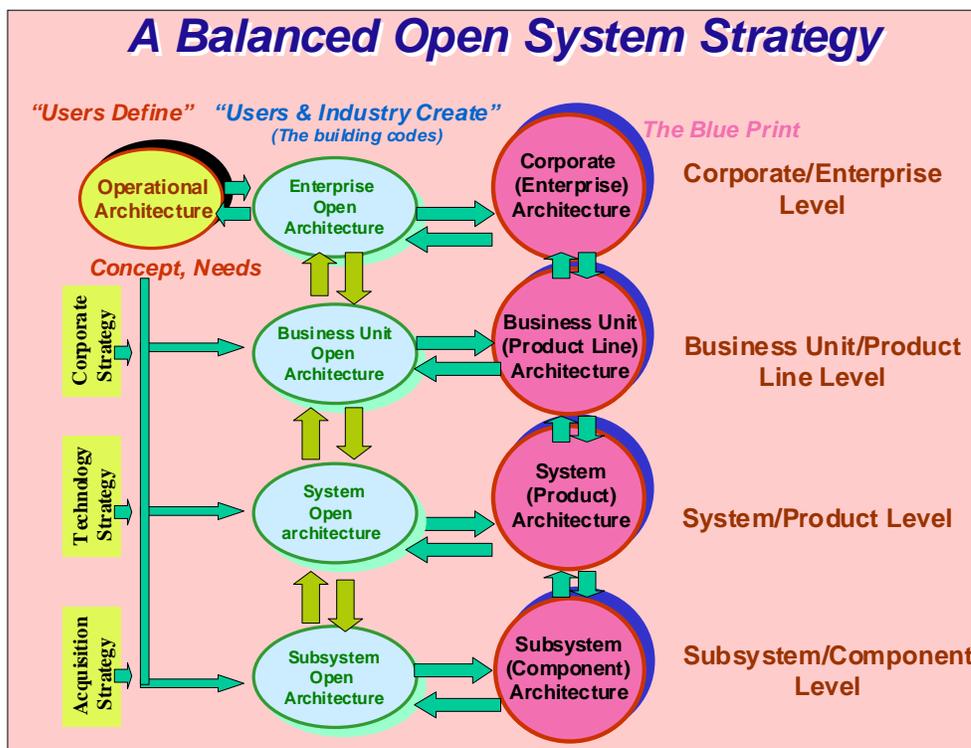


Figure 4: Balanced Open System Strategy

IV. TESTING OF OPENNESS

Test and evaluation of an open system is usually conducted for two purposes – discovery during system development, and confirmation of system performance after development. Through **discovery testing** one will ensure that a program office, during the system

development and design phase has indeed considered the OS strategy, and open system is in fact a viable strategy to pursue. If the viability of an OS strategy is proven during the development phase, one then needs to use **confirmation testing** to test the conformance to a particular open standard, and validate the openness of key interfaces to ensure present/future performance. Both of these tests are important for achieving the full benefits of open systems. Figure 5 shows the different types of tests that comprise the test of openness.

The OS strategy is not an appropriate strategy to use in all kinds of systems. Blindly forcing open standards, protocols, languages, and data formats and mandating openness testing can seriously impact the performance and increase the cost. As a rule, the benefits of discovery and confirmation tests must be always assessed against the costs of doing so, and only when the benefits are greater than the costs should one proceed with testing. Testing of openness is a very challenging and expensive task. There are no specialized test labs that can test the openness of a system and test suits for validating or verifying the conformance with interface standards, protocols, and data formats are at best scarce and very expensive. Only by exercising sound systems engineering practices and by making a strong business case can one reach an appropriate decision as to whether any type of discovery or confirmation testing is needed.

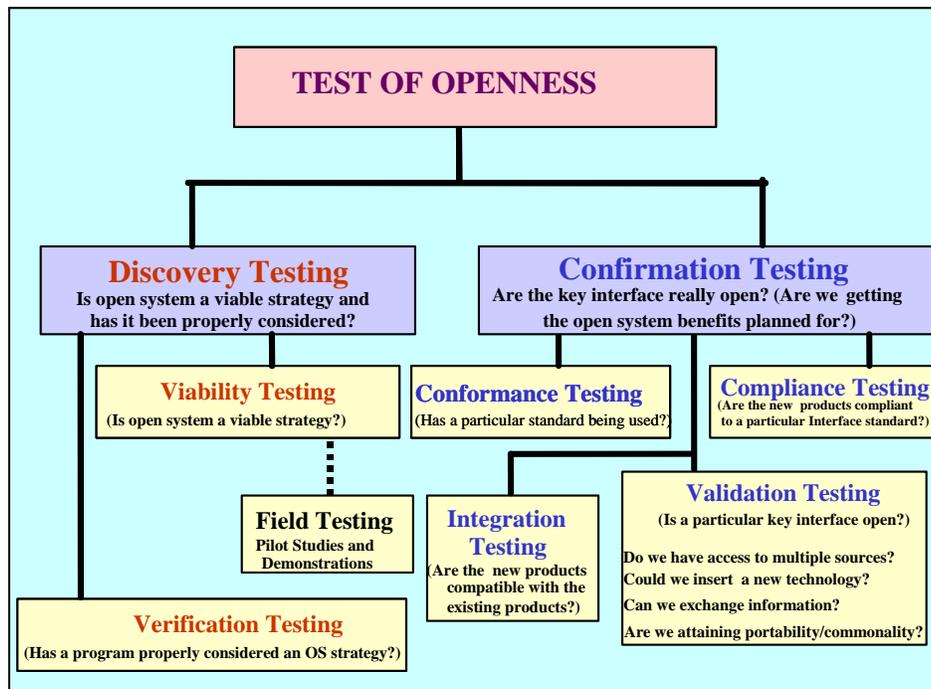


Figure 5: Different Types of Openness Testing

A. Operational/Performance Conditions that Call for Openness

Before rushing to implement open systems and conduct discovery or confirmation tests, we must ensure that the operational or developmental requirements directly or at least indirectly call for openness of a system, subsystem, component, or a particular key interface. The following conditions and operational/performance requirements may call for application of an OS acquisition strategy:

- a. Whenever there is a need to specify operational requirements in an incremental manner over time, matched with time-phased threat assessments and available technology, an open systems design will be appropriate for providing the needed flexibility.
- b. When the nature of the threat is unknown, its magnitude constantly changes, and the technology is not proven, the portability, scalability, and adaptability associated with an open systems design will be of great help.
- c. When the main emphasis is on long-term sustainment and affordability, or affordability is the basis for fostering greater program stability. These requirements may call for optimizing cost-effective commonality of hardware, software, and support systems, simplifying sustainment, and reducing the total cost of ownership. In some cases, supportability is a performance requirement that relates to a system's operational effectiveness, operational suitability, and life cycle cost reduction. An OS design enables access to commercial products from multiple suppliers and as a result will diminish the need for large depot and personnel needed to keep track of an obsolete inventory of proprietary spare parts.
- d. Capability to quickly reconfigure forces and systems. High dependence upon rapid and collaborative responses with distinct ad hoc forces will be more economically and effectively realized by plug and play capability of open architectures. Moreover, the ability to constitute and readily integrate functionally compatible entities and systems is greatly facilitated by architectures and standards that are truly compatible.
- e. Need for seamless, high speed, digital information exchange among diverse entities/elements. These requirements demand joint and combined operations over multiple and diverse hardware and software components and communication networks, and as such are more effectively fulfilled by application of open architectures. The cost of integration, interoperability, and modernization will more likely increase if the key hardware and software interfaces in a digitized battlefield environment are not defined by open standards.
- f. Ability to receive and disseminate commands and controls data in real time. There will most likely be adverse impact on performance, future upgrades and total ownership costs if real time command and control systems are designed based on proprietary protocols and standards.
- g. Need for creation of overarching capabilities for a mission area to take advantage of system of systems or family of systems benefits. When similar open interface standards are applied across a family of systems or a product line, commonality and reuse of components become possible and interoperability is facilitated.

- h. Need for reprogramming of software modules and communication networks. Open systems will enable software reuse and increase the flexibility for reconfiguring the communication networks.
- i. Application of an integrated approach for adding and facilitating the incorporation of future capabilities or advanced technologies with minimum impact on existing systems. Requirements that call for integrated and modular communications and navigation capability are more effectively fulfilled by standardization of interfaces.
- j. Requirements that are defined in terms that enable and encourage the use of commercial and non-developmental item equipment, or call for minimizing the risks associated with being captive to specific products or sources.
- k. Requirements that call for future growth capabilities and performance characteristics that will be highly dependent on continuous use of emerging technologies in computer, communication, surveillance, and navigation technologies.
- l. Requirements that call for interoperable solutions and development of architectures that must comply with predefined standards. Such requirements may demand interoperability across platforms and among subsystems, and may necessitate interoperability and commonality of components/systems that are similar in function to other programs.
- m. Requirements that call for application of modular, reusable, portable, extensible, and non-proprietary software.

B. Test and Evaluation Challenges

As mentioned earlier, openness testing is a very challenging and expensive task that must be done if the requirements specifically call for the use of OS and the benefits of doing so are greater than the costs. Several challenges emerge when dealing with test and evaluation issues corresponding to open systems. The following questions may be asked in regard to test and evaluation of open systems:

1. Could the openness of a system be tested? The answer in most cases is negative because no one is likely to know how much “openness” is needed in a system to make it an open system. Moreover, increased openness may not be necessarily better. For example, a 100% open system is not practical and may not be better than a system, which uses open standards for 50% of its interfaces. Moreover, every system is unique and there is no effective index that one can use as a metric to measure the degree or the extent of openness of a system.
2. Should we verify the openness during the development and design or after the deployment? If a strong case is made for design of an open system we definitely need to confirm that the selected interfaces are in fact open before we deploy the system. These tests must be done again during planned upgrades to ensure that new COTS are complying with the selected standards and will not negatively affect overall system performance.
3. How do we verify that a key interface is open at a particular level within a system? In most cases we should be able to verify the openness of an interface or the application of specific open standards for key interfaces within a system. This type of testing is currently being done for connectivity and interoperability testing purposes.

Unfortunately, the test labs that specialize in performing such tests and certifying the use of certain standards are very scarce. Enormous costs of establishing such facilities and lack of human expertise to operate them prohibit the widespread use of these labs. Moreover, the current labs have not developed the test suite needed to test all kinds of interface standards, especially the emerging ones.

4. Do we need to test the previously certified products? As mentioned earlier, a product's conformance to a standard is more complicated by ambiguities inherent in any language, by products that either barely meet the standard or performance in the upper end of the tolerance range, or products that have extensions to make themselves more attractive in a market. In most cases, the previously certified products such as COTS, government-off-the-shelf (GOTS), front end processors, modems, and radios do not need to be retested in and of themselves, as they most probably have already been certified. But, if these products must effectively interface with existing products developed by different vendors, or the performance of the system is likely to be degraded because of extensions to standards, we may have no choice but to at least perform compatibility testing.
5. Is it enough to rely only on test certifications from contractors/suppliers, or do we still need to validate/verify their compliance? In most cases, it is a normal way of doing business for a client to refuse contractor assurances and go to some independent consultants/test facilities to validate the claims made by them regarding the openness of an interface. The need for independent test/conformance certification could become a requirement in contracting documents such as a Request For Proposal. Independent verification may become more necessary if most of the benefits from using an open interface standard will be realized in the long run.
6. Is it better to evaluate a system to verify the achievement of OS benefits rather than test compliance with specific interface standards? This is a preferred – and difficult - evaluation method. It requires a controlled environment for testing to ensure that the benefits gained are being achieved only from using open systems. Creating a controlled environment is very challenging.
7. What about evaluating a program to verify its compliance with the application of a written process or procedure for implementing an OS strategy? For example, whether or not a program has done a comprehensive study to assess the feasibility of using open systems or has done extensive market research to identify and evaluate open standards, etc. This is perhaps the least costly approach but its focus is on the process rather than the outcome. It assumes that we already have a “best business practice process” for implementing open systems as a benchmark to compare the subject process with it and reach effective conclusions. In the absence of a benchmark process we may at best rely on a list of developmental effectiveness and suitability issues (not parameters, objectives or thresholds) that we could use as a supplement to other needed developmental test and evaluation efforts. The collective responses to the relevant questions asked regarding these issues might then help us to make a better educated guess about whether or not a system will be open when it is fielded.

C. Critical Developmental Issues

Due to lack of tools and inability to measure systems openness, a number of “critical developmental issues” are recommended as a checklist for testers to ensure compliance with open systems policies. Critical developmental issues are defined as developmental effectiveness and suitability issues (not parameters, objectives or thresholds) that must be examined in developmental test and evaluation to evaluate or assess the degree to which a program has considered an OS strategy and the key interfaces within a system are open.

Following are examples of critical developmental issues that can be used as a checklist to supplement the other types of information gathered by the testers:

- a) Has the Program Manager (PM) assessed the feasibility of using widely supported commercial interface standards in developing systems?
- b) Has the technical and operational concepts that directly or indirectly call for use of an open system strategy been properly evaluated?
- c) To what extent have OS requirements been fed into the acquisition process?
- d) Is open systems strategy an integral component of the overall program acquisition strategy for enabling rapid acquisition with demonstrated technology, evolutionary and conventional development, interoperability, life-cycle supportability, and incremental system upgrade without redesign of entire system or large portions thereof?
- e) What approach has the PM used to enable continued access to cutting edge technologies and products, and to prevent being locked into proprietary technology?
- f) Has the PM documented his approach for using open systems and included a summary of his approach as a component of his overall acquisition strategy?
- g) Does the program have a documented process or procedure for implementing an OS strategy? Does the process or procedure explain how the program plans to use the OS strategy as an enabler to achieve predetermined OS related objectives in its acquisition strategy? For example, whether or not a program uses an OS strategy to mitigate the risk associated with obsolescence, dependency on a single source of supply, and proprietary technology. Also, the extent to which a system is capable to quickly and affordably interconnect with and be assembled into existing platforms and systems as needed.
- h) Are there any requirements to test and certify systems/products?
- i) Is the PM using a modular standards-based architecture in designing systems or families of systems?
- j) Does the program have a system architecture description that is traceable to an open systems architecture requirement in the Operational Requirements Document (ORD)?
- k) Is the system architecture traceable to a functional reference model for a specific platform/domain?
- l) At what level(s) is the architecture for the system defined by open interfaces? How were these levels of the architecture chosen?
- m) What level of modularity has been employed for the system? Does functional modularity align with physical modularity to permit easier technology insertion?

- n) Have key subsystems or components and their interfaces been identified? Are the identified key interfaces adequately defined? What percent of key interfaces are defined by open standards?
- o) What type of standards (consensus, de facto, or proprietary), were used to define key interfaces?
- p) What criteria has the PM used to select the interface standards?
- q) To what extent are selected interface standards widely supported and publicly available?
- r) Has the PM provided sufficient justification for using proprietary interface standards for the selected key interfaces?
- s) Are there a large number of suppliers that provide products compliant with the selected interfaces? Does market research support the selection of interfaces?
- t) Does the program use standards that are common to its specific systems domain/platform?
- u) Has the program specified any options or extensions to the interface standards? Do these options prevent using similar components available from other programs or from the commercial sector?
- v) Does the program have a conformance management plan to document the use of standards?

There are also a number of critical supplemental questions related to the above-mentioned issues that the test and development communities should answer before conducting discovery and confirmation tests. They are:

1. Has the test and evaluation community been a member of the Integrated product and Process Development team responsible for considering and implementing the OS strategy?
2. Has the test and evaluation community developed and analyzed OS testing methodologies in light of the above mentioned challenges and critical developmental issues?
3. Is there an appropriate test and evaluation method available for testing the conformance to selected open standards?
4. Is the test and evaluation community assessing the OS related technical or operational performance under the realistic conditions of interrelated or interacting systems?
5. Have areas of concern in testing and certification been identified and the findings been properly reported to appropriate developmental and operational testing organizations (e.g., the Operational Test Readiness Review)? Does the report delineate which interfaces worked and which had limitations, were not tested, or did not work?

V. CONCLUSION:

An open system strategy leverages commercial products and practices to make systems more interoperable and adaptable to evolutionary changes in operational

requirements and emerging technology, and extends the life span of a system. A *closed* system strategy is one that uses privately held, proprietary standards and specifications that are not accessible to any but a single vendor, or a small group of favored vendors. By using an OS strategy, program managers can use widely available open systems-based commercial and non-developmental items/products rather than deal with products of proprietary technologies, standards, and specifications. Consequently, interoperability will be enhanced and system upgrades can be accomplished faster and cheaper because a standards-based architecture facilitates information exchange and incremental technology insertion rather than large-scale system redesign.

An effective OS design must apply open standards, protocols, languages, and data formats in developing systems and ensure adequate conformance to such conventions. Open standards, protocols, languages and data formats are those adopted by recognized standardization organizations and widely supported by the commercial market place. Validation of openness and ensuring interoperability could be costly and complex. There are challenges and issues that we need to be aware of. As a rule, the benefits of gathering data and conducting the test must always be greater than its costs. Finally, an OS strategy is not appropriate to all systems and by blindly forcing open standards, protocols, languages, and data formats and mandating discovery and confirmation testing we may negatively impact performance and increase costs.

References

Azani, Cyrus H. “*Joint Space Operations via Secured Integrated Network of Modular Open Architectures.*” Proceedings of the Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium and Exhibition, 23-27 July 2001, San Diego, California.

Azani, Cyrus H. “*Discovery and Confirmation Testing of Open Architectures.*” Proceedings of the International Test and Evaluation Association Workshop, August 6-9, 2001, Boston Massachusetts

Azani, Cyrus H. “*The Open Systems Strategy: A Viable Business and Engineering Approach for Building And Sustaining Advanced Complex Systems.*” Proceedings of the Defense Manufacturing Conference, November 26-28, 2000, Tampa Florida

National Research Council, “*Aging Avionics in Military Aircraft.*” National Academy Press, Washington, D.C. May 11, 2001