Submitted to the AIAG ActionLine Magazine.

Making interoperability happen

John Horst
National Institute of Standards and Technology (NIST)

Interoperability has become a buzzword. Everyone in manufacturing has heard it. But just what is it? Interoperability is the ability of two system components to communicate correctly and completely with each other—with minimal cost to either component user or component vendor—where the components can be from any vendor worldwide.

Each part of this definition is important:

"…two system components…" Even if one component can broadcast to multiple recipients, such as on a network, interoperability can always be narrowed down to two components.

"…system components…" These components can be of any size, as long as they are distinct—or separable—components.

"…communicate correctly…" There must not be any encoding or decoding errors.

"…communicate completely…" The interface language must allow each component to express anything and everything that both vendors and users agree needs to be said over the interface.

"…the components can be from any vendor worldwide…" The system is not really interoperable if the Europeans have interoperability among their products but not with the North American products. A critical mass of vendors worldwide supporting the standard is essential, particularly in this age of global corporations.

"…with minimal cost to either user or component vendor…" The system is not interoperable if it takes a team of five software developers two weeks to get the two components communicating correctly and completely. The system also is not interoperable if the component vendor had to spend lots of money maintaining the various interface language versions as well as having to spend money for training and support for the multiple versions.

A precise definition of interoperability makes it less likely that someone can falsely make a claim of interoperability. For example, say an OEM (original equipment manufacturers) requires that all suppliers write and read in a single computer-aided design (CAD) vendor's native CAD format. The OEM may claim that this solves the interoperability problem; however, interoperability has not been achieved, because additional burden has been placed on suppliers who have to support the sometimes different CAD formats required by all the other OEMs they work with. By definition, interoperability has not been achieved because one of the users—the supplier—is not expending "minimal effort." Of course, the OEM will end up paying for it because its suppliers need to add that cost to the cost of their products and services.

Interoperability problems

To determine if a company has an interoperability problem, take a look at a few scenarios.

An automobile manufacturer needs to move model operations to another location. The new location has a different software application for the same function, so all software programs must be rewritten, potentially resulting in millions of dollars lost because of a product launch delay, additional labor costs and loss of product quality.

An automobile supplier wants to purchase a new inspection probe from another vendor and integrate it into his current system. He cannot because his current coordinate measuring machine (CMM) will not interface with the other vendor's probe. His current CMM vendor offers to modify the system interface, but for the quoted price of integration, the supplier can almost buy a new CMM.

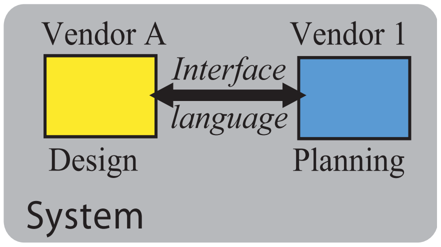
An aerospace corporation has trained CMM operators on a particular inspection software package. They now want to use a new CMM, but they cannot buy the new CMM unless they commit to using the new CMM vendor's inspection software. Choosing the new CMM will require additional software training and re-programming costs.

An inspection software vendor spends a large percentage of his or her development costs maintaining version compatibility with proprietary interface languages.

These scenarios are, of course, biased towards dimensional metrology. Nonetheless, the interoperability problem is essentially the same in the automotive, aerospace, electronics or medical sectors, or in the machining, design, or quality control functions.

The question to ask is, are there any costs—hidden or obvious—in the operation that are related to the kinds of problems illustrated in the scenarios?

Who suffers from the interoperability problem?



Shown are two example components with two separable functions, design and planning, each from a different vendor that need to communicate. For there to be any concern over interoperability, there must exist several vendor instances of at least one side of the interface. Sometimes the component might be a human producing computer-readable output, such as a file of words in the interface language.

The cost of a lack of interoperability—the interoperability problem—is currently known most through anecdotes and it seems to be high and persistent. Even so, there have been several formal studies[1] [2] [3] [4] on the cost of the interoperability problem. They have concluded that the cost of the interoperability problem is

- $1B for the automotive sector for engineering and business data
- $5B for the discrete manufacturing supply chain
- $22B to $59B for inadequate software testing infrastructure

It should come as no surprise that everyone suffers from the interoperability problem: OEMs, suppliers, vendors and customers. There are some interesting twists to the distribution of the suffering, however. The problem occurs broadly among users, but large users seem to suffer the most. This is because large users—through acquisitions, preferences of employees, preponderance of legacy systems and simply the size of their corporations—typically have multiple products from multiple vendors that perform the same function. Furthermore, these products are typically not interoperable.

Smaller users can sometimes require a single vendor's products for a certain manufacturing function throughout the corporation—the "single-vendor solution"—, which may provide a temporary solution. System users cannot easily and cheaply build systems from components. Users cannot achieve best-in-class because the lack of interoperability makes it more difficult to exploit technology changes. The manual reinterpretation and reentry of data is costly and error-prone. Excessive operator/programmer training also is costly. New technologies either cannot be introduced, are delayed, or cost too much.

The interoperability problem also occurs broadly among vendors, but small vendors suffer the most. Large vendors can sometimes dominate the market so that their proprietary interface languages become *de facto* standards. If large vendors define the interface language, small vendors are always playing catch-up, increasing their time-to-market. Furthermore, they may find it hard to differentiate their products by the introduction of new technology because the larger vendors have a lock on the interface language definition.

Finally, the dominance of single vendors reduces competition, which invariably increases costs to the users. Component vendors have to support many different proprietary interface languages, sometimes costing them up to 60 % of their software development budget.

In the end, no matter whether users or vendors suffer, higher costs are passed on to the consumers in the form of higher prices for products and services. Solving the interoperability problem can save time, money and resources, and these resources can be used to solve other problems.

Response options

Everyone has experienced the interoperability problem, even if it is as simple as being unable to paste a graphic into a document. Most of the so-called solutions to the interoperability problem are not really solutions at all, but either get the system working, at substantial cost, or push the problem onto someone else in the supply chain. Responses to the interoperability problem fall into three categories: point-to-point, single vendor and common language.

The point-to-point response—just getting local components to work together—is not really a solution because it cannot be done with minimal cost to either user or component vendor, and it is costly, error prone and stagnant. It is also shortsighted since it impedes future integration of more advanced components.

The single-vendor response can provide a tentative relief from incompatible systems, but it has problems. It can encourage lack of competition, which can raise costs—cost savings depend on the stability of the single vendor—and one is restricted in choosing best-in-class solutions. The single-vendor response is costly, particularly because it generally pushes the problem on to someone else. On the other hand, the common language response actually solves the interoperability problem.

The common language response—defining an open, non-proprietary language for the interface, that is complete and correct and has worldwide support—reduces measurement errors caused by translation mistakes. It requires less time and effort of both user and vendor and provides greater freedom for successful acquisitions and best-in-class preferences. There seems to be little concrete data that the cost of the time and effort required to enable a common interface language actually is much less than the cost of the point-to-point or single vendor responses. In spite of this, consider the following analogy.

Does having a single common interface solution for plug-in cards, printers, network connections and wireless links enable productive, inexpensive use of the computer? Were things better prior to the emergence of these various standards? Of course, in several cases there are multiple competing standards, for example, wireless systems and plug-in printed circuit board standards, but most will agree that that situation is much better than having a slew of proprietary standards for each vendor's PC board or wireless networking system.

The common interface language response is the most satisfying solution to the interoperability problem.

Investing in solving the problem: the standards development process

Reasons why the interoperability problem should be solved and why the common interface language solution is the best solution have been discussed, but what does it cost to realize such a solution? How can the industry expect to get competing users and vendors worldwide to agree to a single common language? Timeliness and concurrency are essential ingredients.

The interoperability solution requires worldwide support and concurrent development of the following elements:

- **Interfaces**: Identify appropriate interfaces, identify existing interface standards, and identify gaps and overlaps
- **Interface languages**: Develop timely, unambiguous, sufficiently functional, and consensus-based languages
- **Implementations**: Create implementations that are timely, compliant, fully functional, interoperable, and performed by a critical mass of vendors worldwide
- **Tests**: Ensure that products pass conformance and interoperability tests before purchase

The first step is to identify the key interfaces. All the key stakeholders worldwide need to participate in this identification, or at least be in agreement with the conclusion. Then the industry needs to identify what languages currently exist worldwide for those interfaces, considering both proprietary and non-proprietary, closed and open languages.

If new interface languages must be developed, develop them. Vendors, typically, are the best ones to do this development with user participation. Make sure that the interface languages are correct and complete as they are being developed, that they have accurate syntax and well-defined words and grammar, and that they define all the functionality required by both users and vendors. There also must be a process in place to make changes, additions and improvement to the standard as technologies change. Get agreement from users worldwide to support a single interface language, by specifying the language in their purchase requirements. Only this step will discipline the vendors to commit resources to the development of the standard, which only they can do. These resource commitments can be somewhat costly.

At the same time, vendors worldwide need to be participating in the implementation of the interface language in prototype versions of their products. This participation must be mandated by a critical mass of users worldwide, otherwise vendors worldwide will typically not participate. There must be a process whereby the knowledge gained about the standard by doing implementations is fed back to the standards writing group. It also is expeditious if the language writing committee is a small group, either a single vendor or small group of vendors. However, this small group must be responsive to critical input from the entire community worldwide.

Tests must be developed and run against all implementations concurrently with all other development activities. At least two types of tests are required, conformance and interoperability tests. Conformance tests are tests of one implementation, on either side of the interface, against test utility software using carefully defined test cases. The test cases must cover as much of the anticipated functionality as is possible and reasonable.

Interoperability tests are between two implementations, one on each side of the interface. Public demonstration events work well because such events act as a motivator for vendor implementers to work diligently on their implementation.

Organizing, defining and arranging these tests and generally enabling and ensuring successful timely execution of the standards development process outlined above has turned out to be an excellent role for an organization like the National Institute of Standards and Technology (NIST). NIST has performed this role for many different interface language standards, including the I++ DME (Dimensional Metrology Equipment) and DML (Dimensional Markup Language) standards, which have been popular and successful in dimensional metrology systems.

NIST has observed that neither defining interface languages by large committee works because it takes too long to develop a working interface language; nor does supporting an open, proprietary interface language because all the other vendors are frustrated and hamstrung by the single-vendor control of the language definition.

Interoperability is a goal worth seeking and, if the common interface language approach and the correct process are employed, there is hope in attaining the goal.

But do the benefits outweigh the costs? If considering the analogy of the personal computer, surely the benefits will outweigh the costs.

Finally, is a partial solution satisfactory? Yes, as long as everyone recognizes that there is still more work to do to gain more benefit.

Interoperability is worth the effort, but it is an effort that requires some level of commitment from everyone.

---

[1] Economic Impact of Inadequate Infrastructure for Supply Chain Integration , June 2004

[2] Economic Impact Assessment of the International Standard for the Exchange of Product Model Data (STEP) in Transportation Equipment Industries , Dec 2002

[3] Economic Impacts of Inadequate Infrastructure for Software Testing , May 2002

[4] Interoperability Cost Analysis of the U.S. Automotive Supply Chain, March 1999