Second-Best Practices for Interoperability

Martin Libicki
National Defense University, Washington, D.C.

If the GII is to realize its full potential, it will have to support distributed applications among heterogeneous user systems exchanging not only bits, but also mutually understood meanings (nouns and verbs). This requires standards. Yet the best method of generating such standards, through explicit consensus, may simply not work well, or on time. Second-best approaches such as middleware, lexical primitives, or metalanguage need to be considered.

The vast world of information technology standards may be characterized by its two largest realms: public communications and private computation. Public communications entails hauling bits (with the requisite level of service, block definition, reliability and security) among entities that may be anonymous to each other. This is the province of the telephone system, the Internet, etc. Private computation is epitomized by the fully functional corporate network maintained by a specified office to support applications using specific and well-understood information definitions.

Both realms are characterized by specific standards. Interoperability in public communications is supported by ITU and Internet standards. Portability of applications across heterogeneous architectures is supported by an ad hoc mixture of language standards, data-item standards, operating system standards, and emerging application portability interfaces. Communications tends to get the standards it needs—which it must if public communications is to exist. Computer uses tend to be covered by standards less often, but privately managed computer systems can use hand-crafting and tight management to get over the bumps.

Over the next ten years, the formation of a global information infrastructure—the great challenge in information technology—will require a merger of the two realms. That is, public systems will need to find ways of exchanging and interpreting not only bits, but meanings. They will have to find ways of referring to common concepts using, if not identical vocabulary, at least a vocabulary that permits translation. Moreover, they will have to exchange information without the labor-intensive pre-negotiation usually entailed in the construction of private infrastructures.

One can already glimpse applications that run over heterogeneous equipments (that is, nodes) operated by heterogeneous owners to transfer and understand information. Network management is an early application, albeit one well-standardized by its origin in telecommunications. More typical in the future may be environmental monitoring. An adequate environmental picture may require the fusing of data from ground sensors, water sensors, airborne laser-fed

The formation of [GII] will require a merger of public communications and private corporations.

sensors, weather stations, and even satellites. These sensors may be owned by state, local, or federal governments, by private corporations, or perhaps by transnational governments. They may be tied into processors and databases that are also under heterogeneous ownership. Public health or transportation monitoring may have similar architectures. Tomorrow's battlefield may require a similar concatenation of information, not only from the traditional four Services, but from allies, interested third parties, international organizations, and the media. Other applications may include cooperative manufacturing and design, military planning systems, simulation, and software knowledge agents. Clearly, there is a great potential to create powerful distributed systems if the problems of semantic-level interoperability can be solved.

Problems with Best Practice

How is all this interoperability going to come about? The best answer, of course, is standards—agreed-upon conventions for exchanging data and metadata that are unambiguous, comprehensible as well as comprehensive, technically compatible with their media, publicly available, and actually implemented everywhere they are needed (and, which would be nice, that only provide one or two ways of doing the same thing).

But how likely is it that the existing standards apparatus will actually generate these standards at all, much less in a sufficiently timely fashion? There are many grounds for skepticism:

- —The field is enormous—so many domains, so many algorithms, so many instruments to coordinate.
- —The standards process has difficulty coping with even relatively simple problems (which are made more complex by the need to accommodate every plausible vantage point).
- —The question of semantic-level interoperability is a complex problem all of its own.
- —The heterogeneity level of a single institution's equipment, systems, and software is already high; when different institutions (especially those from different countries) must work together the heterogeneity level rises astronomically.

The metaproblem has yet to be well characterized. How do various nodes of a complex distributed system characterize themselves (e.g., send-or-receive, data-push or demand-pull, poll-or-pulse) to other nodes of the network? How do they adjust their organization to external events? Most such systems are designed under the assumption that there are many clients and few servers. Some distributed application networks may have the reverse architecture: few clients and many servers (e.g., sensors).

Most standards processes are driven by vendors who in turn are driven by a wide spectrum of users. Users (telecommunications companies largely aside) rarely get together to make standards so that they may interoperate. Unfortunately, there does not seem to be a vendor community waiting to arise to serve the needs of cross-institution collaboration (and it is not clear whether the drivers for standards will be found among vendors of instruments, networks, software, third-party service providers, or defense firms). Many users are not yet aware that they may someday need to interoperate with their peers (or perhaps they assume that some third party will take care of the problem, much as the phone companies do for bit hauling).

In a military context, institutions such as NATO take interoperability seriously, but, owing to national differences, progress is often slow. Tomorrow's allies, however, may not be in a position to give us much warning before asking for help (e.g., Saudi Arabia circa 2 August 1990).

Even if the various domains can work out ways of interoperating, the economies that come from their each solving the problem the same (or at least in a compatible) way are unlikely to be realized. The various standards communities that are working on issues close to semantic-level interoperability are not talking to each other. Query-and-response standards are being generated by the database community (e.g., structured query language), the library community (e.g., Z39.50), the artificial intelligence community (e.g., KQML—knowledge query manipulation language), the agent community (e.g., General Magic's Telescript), the network community (e.g., Simple Network Management Protocol), and the intelligence/aerospace community (e.g., Structured Full-Query Language, CD-Rx). These "interoperability communities" themselves do not interoperate very well.

The history of standards suggests that, as often as not, a standard rises to dominance through a killer application. The Internet made TCP/IP what it is; Mosaic did the same for HTML. SQL became popular not because of its technical superiority but because IBM used it as a vehicle to implement the useful concept of relational databases. C rode on the back of Unix (which was its own killer application). Could

Could the most likely killer application for distributed . . . collaboration come from international security requirements?

the most likely killer application for distributed heterogeneous collaboration come from international security requirements?

Second-Best Practices

So the problem remains. The GII is standing on the verge of some audacious distributed applications, but there is little evidence that detailed domain-optimized standards exist to help the various nodes communicate meaningful information to each other. It is time to consider a variety of second-best practices: middleware, a common set of lexical primitives, and, if all else fails, a metalanguage.

MIDDLEWARE

Middleware, it is widely proclaimed, is the answer. The problem now is figuring out whether distributed applications over heterogeneous nodes is the right question.

The epitome of the middleware promise is the network-distributed object. In object theory, data come wrapped together with the applications that may legally manipulate them. As an example, a complex document that one calls up with the network would come bundled with (or with pointers to) software that sets itself up on your machine and, once it gets going, can set up, link, and display the document regardless of the machine's architecture. The Object Management Group is doing some excellent work in developing the various interface handles and housekeeping standards required to make such processes smooth. A similar approach may emerge with a combination of a standard Web language (Netscape HTML version N+1) and something like Sun's HotJava which enables the distribution of applets over the Web. (OLE is another answer for those to whom the problem is one of coordinating applications that produce documents and whose notion of heterogeneity is different applications which run over Microsoft Windows—more typically, over Microsoft Office over Microsoft Windows.)

Will middleware suffice? Maybe not. CORBA's target model is not the world of nodes run by heterogeneous users, but a world of corporate networks which run heterogeneous equipment. CORBA's interface definition language assumes some common linguistic formats and some recognizable architectural principles in its machines (not unreasonable ones, but not necessarily universal—particularly since OMG is Unix-centric at heart). It makes implicit assumptions about what machines can do and how they do it. The corpus of CORBA standards is also very large and complex and not yet entirely accepted. In addition, despite the many safeguards that can be built into objects, passing code amongst strangers is always a dicey business. Middleware speaks to the issue of passing verbs around (given a recognizable set of computational primitives), but is less adept at passing nouns—with all their meanings—around. It is also not an obvious vehicle for communicating taskings and expectations.

LEXICAL PRIMITIVES

If the fundamental problem of distributed networks is how to pass around queries and responses, as well as taskings and expectations, then perhaps a better place to start is with a language specifically engineered for that purpose.

Structured Query Language is a good base on which to consider the nouns and verbs that distributed computing may require. The language can easily be extended to cover metaqueries, text, and geographical-based information. From there, one could mix in concepts from other fields to encompass tasks such as document exchange, process control, security, user agents, and perhaps even negotiations. Nouns could be generated using the same process by which EDI forms are standardized; indeed, EDI has expanded beyond the business community into the engineering and health-care fields. If the design is kept sufficiently English-like, even amateurs could understand what certain types of conversations mean. In a sense, what standardization could create is the neo-Wittgenstein dream of a logical language, but one which is sufficiently domain-limited to obviate the more paradoxical features (an issue that has preoccupied analytical philosophers for most of this century).

This approach echoes the ways humans have handled the problem of coordination; it deals explicitly with nouns; and by using parsable terms rather than raw code, it avoids some obvious security risks. Nevertheless, in most other respects, it is even less of a solution to the problem of distributed computing than is CORBA. Its terms incorporate the meaning of some complex objects explicitly, but every time a fundamentally new object comes along, a new set of vocabulary will be required.

Middleware at least has an institutional sponsor in OMG, and some momentum. So does SQL, but the database community has yet to start thinking seriously about how to manage applications that cross institutional boundaries (for instance, the 1992 version of SQL had a wonderful array of table manipulation

tools, but no way to inquire as to what was in the tables themselves). ANSI's EDI standard X12 works, but only because large buyers have forced smaller ones to use it in order to stay in business. It has grown large and complex and is unlikely to be the basis for consumer electronic commerce.

METALANGUAGE

Imagine yourself suddenly in contact with someone who speaks no English. If a dictionary is not at hand, one plausible approach is to start pointing to objects and naming them. Verbs are a little trickier but enough pantomime will do. Other word forms (e.g., adverbs) are tougher yet but still not impossible. Abstract nouns (e.g., justice) probably require the most work. Sooner or later (and remember, time to a processor is parceled out in nanoseconds), a common linguistic base can be established and communication can begin.

One might describe this approach as more of a last resort than a second-best practice. Yes, it can be made to work, but it requires two prerequisites: a common base of material that can be referenced and a relatively small but essential metalanguage with which to convey the modalities of linguistic translation.

The first task for the metalanguage is simple bitstream connectivity followed by some introductory handshaking. Yet this is but a start. To show why, suppose one is pointing at a large brown table. What term is being conveyed: pointing? large? brown? table? the command to look? the assertion that one can command the other to respond to pointing? Enough pointing to objects that are similar only insofar as they are brown would probably narrow the concept down, but one also needs a mechanism that is the conceptual equivalent of pointing in order to refer to common words (something like a uniform resource locator, but more flexible).

The primary advantage of a metalanguage is that it presumes the least commonality—but still must include some. The primary disadvantages, however, are the need to access commonly addressable material and the ambiguity that results from never really knowing whether the communicators' mutual understanding is all that precise.

Conclusions

The best, it is said, is the enemy of good enough. A perfect world that anticipates and promotes the global information infrastructure and its marvelous array of distributed intelligence, mirroring humanity's own, requires standards to match. If the primary route to good standards is unavailable (or at least, slow to mature), second-best approaches will be required. Neither middleware, common primitives, or metalanguages is necessarily the answer, but it is a fair bet that whatever best second-best approach emerges will have elements from all three. **SV**