In: Federal Database Colloquium ’01, San Diego, August 2001.

A “Community of Interest” Approach to Data Interoperability

Scott A. Renner, Ph.D., Member, AFCEA
The MITRE Corporation
201 Burlington Ave., MS B265
Bedford, MA 01730, USA
(781) 271-8625
sar@mitre.org

Abstract

Existing data administration policy in the DoD has produced good results in some pockets, but has not delivered data interoperability for the enterprise as a whole. In this paper we present a concept of operations for a different approach to data interoperability, one based around the notion of “communities of interest”. This approach corrects the shortcomings of the current regime in three ways: First, it does not assume a single monolithic data standard. Instead, we work toward data interoperability within separate communities of interest. These groups serve a knowledge management purpose: they extract the shared knowledge necessary for data interoperability from the larger community, then make this knowledge explicit, and finally help to transfer that knowledge back into the whole community. Second, our approach captures knowledge at more than one level of abstraction. Our approach will produce higher-level definitions that can support interoperability in architecture descriptions and information retrieval “semantic tags” in addition to the current implementation-level data standards. Finally, our approach can become part of the system acquisition process, giving both better incentives for program offices to participate, and better metrics for measuring their results.

Key words - Community of interest, data interoperability, knowledge management

1. Data interoperability is a knowledge management problem

Data interoperability is a key element in the *Joint Vision 2020* goal of information superiority [2]. But we still do not have a workable plan for achieving DoD-wide data interoperability. This paper is intended to describe the principles on which such a plan might be implemented.

We define data interoperability as the ability to correctly interpret data that crosses system or organizational boundaries. The key points are illustrated below in Figure 1. We assume that the people on the left have information needed by the people on the right, and that data in one system is accessible to the other. Information can flow (and we have data interoperability) if and only if the receiving system and users properly understand the data they receive. Clearly, data interoperability is only one aspect of the overall interoperability problem. The left-hand side might not ever have the information needed by the right-hand side. Or it might be impossible to get data from one system to another (because of incompatible radios, physical connectors that don’t fit, etc.) While those would be interoperability problems, they would not be data interoperability problems.

Two things must happen before two systems or organizations can understand each other’s data. First, the people involved must identify a matching real-world thing of mutual interest. (All data describes some thing of interest. You can’t meaningfully exchange data unless it describes the *same* thing.) We call this step *establishing a semantic match.* Second, they must arrange to eliminate or otherwise deal with the differences in the name, structure, and representation in the data models they use to describe this real-world thing. For example, if you measure a certain distance in miles, but I expect distances measured in kilometers, then the appropriate conversion must be applied to the data before I can properly use and understand it. This second step is *coping with representation mismatch.* (More details, with examples, can be found in [6].)

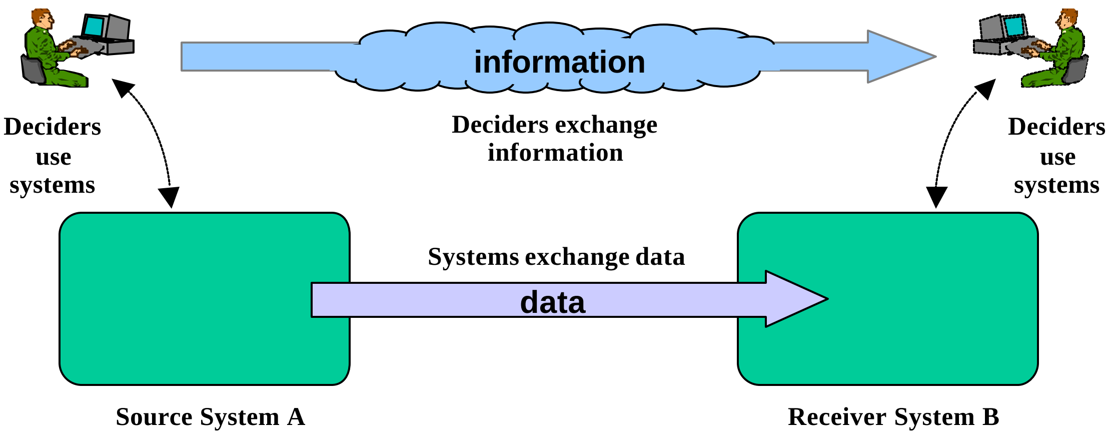

Figure 1: Data interoperability

In this paper we focus on the semantic match problem. It is by far the most challenging of the two. Representation mismatch problems can have automated solutions, either through hand-written translation software or data mediation tools. But there is never going to be an automated approach to the problem of semantic matching, because the solution takes place inside the human mind. That is, for every data interoperability success, there are at least two people with a shared understanding of the exchanged data. These people may be users who look at the data presented to them, or they may be programmers who decide what systems will do with the data they process. Either way, these are the people who know what the data should mean. If the data they receive does not conform to the definitions they know and expect, the result will be an error, not interoperability.

Crucially, what must be shared is knowledge, which is inside human minds, and not merely written definitions, which can be captured in words and paragraphs. For example, you and I might both be satisfied with the definition "an aircraft is a vehicle that moves through the air." But if in your world all aircraft happen to have a fixed wing and at least one engine, while my world includes free balloons and helicopters, then this difference in tacit knowledge is likely to cause some interoperability problems even though we are using the exact same explicit definition. Conversely, the developers on my software project might arrive at a common understanding of our data through conversation, without any written definitions at all. Shared knowledge, not shared documentation, is what counts. Common explicit definitions are of course useful – we are hardly arguing in favor of undocumented software – but they are neither necessary nor sufficient.

The semantic part of data interoperability is therefore a *knowledge management* problem: How do we arrange for the right people to have the right (shared) knowledge about what the data means? Automated tools can help solve this problem. They can never make it go away.

It is instructive to look at the places where shared semantic knowledge is important and at the roles of the people who need it. Here are the most important cases:

- Users need to understand the meaning of the data presented to them. Programmers need to understand the data their programs will manipulate. These people must have a shared understanding of the runtime instance data values. (This is the most common case, the one that everyone thinks of first.)
- Architects building products from the C4ISR Architecture Framework [1] need a common vocabulary for representing information exchanges. These architecture products include abstract descriptions of the instance data that will be exchanged between executing systems. That is, they need to describe *what* information will be exchanged, but not necessarily how it will be represented. Without a shared understanding of the information described in operational and system architectures, it will be impossible to relate architectures developed by separate teams, or to "roll them up" into an integrated view – or even to be confident that one product will convey the same meaning to any two readers.
- Information dissemination relies on commonly-understood metadata "tags" to distribute information products from producer to consumers. For example, one consumer might state in his profile that he wants information about "surface naval ships". He then receives documents that are so marked by their producers.

This depends on a shared classification taxonomy – it won't work unless producers and consumers have the same understanding of "surface", "naval", and "ship".

- The *semantic web* [4] is a plan to extend the existing Web by adding machine-processable semantic metadata to web pages. Software agents can use this metadata to "understand" the information and/or services provided by a web resource and then to act on our behalf: reserve us a rental car, change a doctor's appointment, etc. But unless the people involved (the end user, the agent programmer, the resource provider) have a shared understanding of this semantic metadata, we shouldn't expect these smart agents to actually do what the user wants done.

These people do not all need precisely the *same* knowledge. For example, software programmers (of a web agent or of a traditional application) need more detailed knowledge than software architects. But people in these different roles do not need completely *different* knowledge, either. Clearly the sort of high-level "what is an aircraft?" definition is useful in every role. The knowledge needed by architects (what information do these systems exchange?) should be an abstraction of the knowledge needed by the system developers (exactly what data are we passing to each other?). Figure 2 shows different roles and the level of detail and assurance they require. The determining factor is the location of the person who does the understanding: end users, who can examine instance data at runtime, need less than programmers, who cannot.

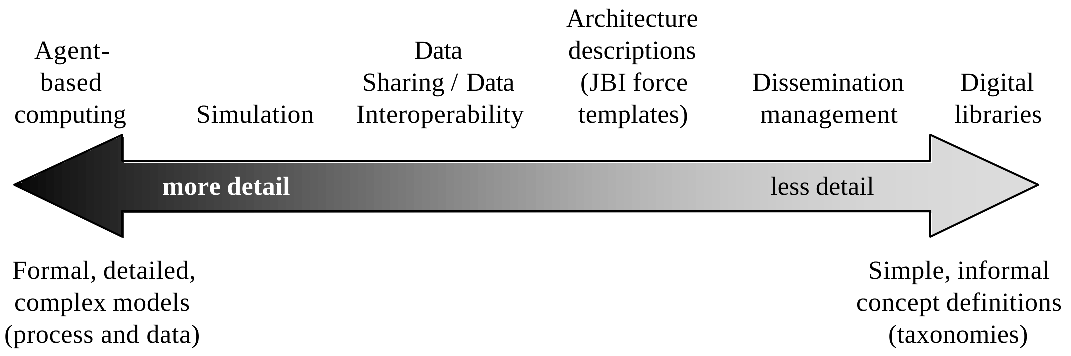
Agent-based computing
Simulation
Data Sharing / Data Interoperability
Architecture descriptions (JBI force templates)
Dissemination management
Digital libraries
more detail
less detail
Formal, detailed, complex models (process and data)
Simple, informal concept definitions (taxonomies)

Figure 2: Level of detail and assurance needed for different purposes

How can we explicitly capture this shared knowledge in a way that maintains these relations without requiring people to learn details they do not need? In [7] we suggest organizing semantic metadata into the following layers:

- *Ontologies* are organized collections of definitions, including relations between the defined terms. In its simplest form an ontology is a dictionary of terms. Data models may specify the meaning of their entities and attributes by reference to the definitions in the ontology. (An ontology is not itself really a model; instead, it defines the universe of things that can be included in a data model.)
- *Abstract schemas* define information without specifying a particular data representation. They allow application developers to establish a semantic match with a database (or second application). Abstract models are sometimes called *conceptual data models*
- In a *concrete schema*, all of the attributes are fully-defined as implementable data types. A concrete schema defines everything the application developer needs to know about the data when interfacing with a database (or second application): a complete description of the names, structures, and representations of the data. A fully-attributed, normalized *logical data model* is a concrete model. Standard message formats (e.g. the Air Tasking Order) are also concrete data models.

Organizing semantic metadata in this fashion preserves the connections between related knowledge, because metadata in one layer points to metadata in a higher layer to specify its meaning. It helps with the level-of-detail problem, because people need not learn facts in the lower levels if they do not need them. We still have a big problem with the *scope* of knowledge required. It is impossible for *anyone* to know all of the definitions that are required for *everyone* to exchange data. That is, the shared knowledge required for the union of all data exchanges is too large to fit inside any one person's head. We must partition semantic knowledge into smaller, more tractable packages. The mechanism for doing this is the subject of the next section.

2. Communities of interest are a knowledge management solution

Data interoperability poses a knowledge management problem: how do you get the needed semantic knowledge into the people that must understand the exchanged data? The community of interest concept is the knowledge management solution we propose. We define a *community of interest* (COI) as the collection of people that are concerned with the exchange of information in some subject area. The community is made up of the users/operators that actually participate in the information exchange; the system builders that develop computer systems for these users; and the functional proponents that define requirements and acquire systems on behalf of the users. The subject area is the *COI domain* – whatever the people in the COI need to communicate about. And in order for the systems and organizations to communicate – to have data interoperability – the people in the COI must all know and understand the consensus definitions for the data they will exchange.

Because we assert *a priori* that the people in a COI need to communicate with each other, we know that it must be possible for them to agree on consensus definitions for the things of mutual interest. But it takes hard work to establish what these things are, and what the agreed definitions will be. This work is the function of the *COI data panel*. The people on the data panel are drawn from the COI. Their job is to represent every part of the COI, determine where the data interests of these parts intersect, and then produce the shared definitions for the union of these intersections. In their job the data panel is performing a knowledge management task. First, the data panel extracts the shared knowledge necessary for data interoperability from the larger community, it then makes this knowledge explicit, and finally it works to transfer that knowledge back into the whole community. The data panel's job is done when everyone in the COI knows the definitions they must have in common in order to exchange information.

2.1 What do COI data panels do?

The main task of the data panel is to produce and maintain the *common data representation* (CDR) for the COI domain. The CDR is the explicit representation of the COI's shared semantic knowledge. It will be organized into ontology, abstract schema, and concrete schema as described above. We expect the data panel to produce reference data sets; that is, the values of coded fields and their meanings. They may also produce business rules (constraints on the data values) and process models (which describe how data is used).

The process of building the CDR is not vastly different from the collaborative data modeling sessions used in the present data administration regime. There are three important differences: The panel will produce more than standard data elements and models and other concrete schema artifacts; it will also build ontologies and abstract schemas. The panel will focus on exchanged data, not all data, and will consider cost/benefit when choosing the exchanged data that will be modeled first. Finally, existing systems are not required to change their internal data representation to conform to the CDR. Instead the individual program managers will make that decision on a cost/benefit basis. [5] contains a description of a process that could be used to build CDRs, plus more discussion of the three differences above and the advantages they bring. Also, the Health Level Seven (HL7) consortium has developed a detailed modeling methodology that could be adapted to building CDR for COIs [3].

2.2 How are COI data panels formed?

The process always begins when someone identifies the need for a new data panel. This can happen bottom-up, when separate organizations realize that together they form a COI and need a data panel to construct their consensus CDR. It can also happen top-down, by direction, without waiting for organizations A, B, and C to realize (or admit) that they need to cooperate on a COI data panel. No matter how the process begins, the next step is always to prepare a draft charter for the COI data panel. This work will be led by the proposing organization. The following issues must be considered in preparing the draft charter:

- The proper scope of the new COI (the people) and of the COI domain (the subject matter). This is more an art than a science. The first principle is that data interoperability problems arise when data crosses system and organization boundaries. Therefore, useful COIs will also cross these boundaries. A COI completely contained within a single organization (or worse, within a single system) would not be worth the bother. Second, COIs become more valuable as they include more people and cover a larger domain. However, the costs of creating/maintaining a COI increases along with its size. Eventually the costs increase faster than the value. We do not have a hard objective rule for computing the optimum size of a COI – consensus among the participants (is this group getting too big to be effective?) is probably the best we can do, at least for now. As a rule of thumb, consider that everyone in the COI ought to understand all of the definitions in the CDR. When that seems too hard, consider partitioning the COI into smaller specialized domains.

- The organizations that should belong to the COI and be an active part of the data panel. (That is, try to distinguish those who want and need to build the CDR from those who will simply use it.) This will be an iterative process; adding an organization to the COI may change the domain, which may in turn change the organizations that should belong.
- The cost and benefit of chartering a data panel for the proposed COI. Both of these are difficult to quantify exactly. Costs include both the work of building the CDR, and of transferring the knowledge in the CDR to the whole COI. Some benefits are avoided costs elsewhere; e.g. the system interfaces become cheaper to build and maintain. Other benefits appear as increased capability.
- Does the proposed COI cover the intersection of two or more major existing COIs? Then it must be a substantial intersection, something worth the overhead of a data panel. Be sure there really is a distinct community of people who must work in both domains A and B; be sure the necessary work can't be done within the separate A and B panels.
- Is the proposed COI a subset of an existing COI? This only makes sense if the smaller subgroup needs a specialized vocabulary and if it's too expensive to expect everyone in the larger group to know this vocabulary.

COI data panels are established when their draft charter is approved by the chartering authority. We suggest that the service Chief Information Officer (CIO) is the proper authority for those communities that do not cross a service boundary, and that the DoD CIO is the right authority for communities that include more than one service.

The data panel needs to include representatives from every part of the COI. When some systems or organizations are not represented, it is likely that some of their knowledge will not be captured by the data panel, or that some of the knowledge codified by the panel will not be transferred back into those groups. In either case, data interoperability with the missing groups will suffer.

Most of the people on the panel will be subject matter experts (SMEs) in the COI domain. These, after all, are the people who possess the knowledge the data panel is trying to capture. It is likely that most of the SMEs will be functional proponents. However, because some of the knowledge we are trying to capture is implementation-level detail, the data panel will need to include SMEs from the system builders. Finally, each data panel will typically need some people whose expertise is in capturing data definitions, rather than in the COI domain to be defined.

Most of the work in the data panel will be done by the SMEs drawn from the functional and builder communities, and so most of the costs will be carried by those organizations. Program offices will know the COIs they belong to, and will need to include costs of participation in their budget. A small part of the work should be centrally funded, and not directly paid for by the participating organizations. We suggest that responsibility for developing and maintaining the procedures and infrastructure used by the data panels probably belongs to the DoD CIO.

3. Incentives and motivation: why this solution will work

Suppose we charter a COI data panel and allow it to build the common data representation that captures semantic knowledge for the COI domain. This by itself doesn't do any good. We are no closer to data interoperability until this knowledge is acquired by the rest of the community and used by system developers to implement data exchange. How can we make that happen? Experience teaches that it is not enough to simply give orders to the developers. Instead, we need to make COIs part of the work process they are already following. We also need to show how the COI process makes it easier for developers to satisfy the interoperability requirements they already have.

There are several ways that the COI process can be inserted into the existing system acquisition process. The following suggestions call for the definitions in a COI's common data representation to be used in products that are already being created. If a program needs a definition that does not exist in any CDR, then that program has not been adequately participating in the corresponding COI data panels.

- *MNS and ORD:* At a minimum, we should require each Mission Needs Statement and Operational Requirements Document to specify the list of COIs in which that system will participate. (A system which does not belong to any COI is by definition a stovepipe. Who wants to fund any more of those?) We might also measure the proportion of nouns and verbs in the functional requirements that are defined in the ontologies from the corresponding COIs.
- *C4I Support Plan:* These are tools for managing implementation issues related to C4I infrastructure for a specific system or program. They may describe information exchange requirements; when they do, these should be expressed using abstract schemas defined by the relevant COIs.
- *Operational and system architectures:* All information exchange requirements described in C4ISR Architecture Framework products should be entirely captured in an abstract schema defined by a COI. The logical data model (OV-7) should map to an abstract or concrete schema defined by a COI.
- *Data exchange agreements, message formats, and ICDs:* All of these should map to a concrete schema defined by a COI.
- *Database definitions:* Definitions of the shared portion of a physical database model should be drawn from concrete model defined by the relevant COIs. That is, when you are building a physical database model, you should define the meaning of your data elements using definitions from the COI's definitions whenever these are available.
- *Semantic tags:* The terms used in profiles for information dissemination should be drawn from an ontology defined by a COI.

The COI approach can make the developer's job of building interoperable systems easier. To a certain extent the COI data panels only reorganize work that programs are already performing. Programs already invest much effort in building interfaces with other systems. Part of that work is creating a shared understanding of the data to be exchanged. That work is no harder when done in a data panel instead of in pairwise interface meetings. Data panels offer the possibility of reuse; that is, some of the shared semantic knowledge needed to interface with system A may be helpful when interfacing with system B. Finally, parts of the CDR produced by the data panels can be used to implement data exchanges with other systems; for example, as inputs to a data mediator.

4. Conclusion

The hardest part of data interoperability is the semantic problem: arranging a shared understanding of what the data means among all the people involved in the data exchange. This same problem occurs when systems share data through direct database exchange, when systems pass XML documents to each other, when separate architecture teams try to integrate their work, and when producers and consumers try to use semantic "tags" for information dissemination. We can approach the semantic problem in all of these cases by dividing people into communities of interest and making explicit the knowledge each community needs to communicate. That will take hard work, but it is work that would have to be done anyway, possibly over and over. By inserting our "community of interest" approach into the normal system acquisition process, we can arrange for that work to be done once. The result will be a data interoperability solution that can work, because it is technically feasible, and will work, because it will actually be executed by the programs.

References

[1] C4ISR Architecture Framework, version 2.0, December 1997.
http://www.c3i.osd.mil/org/cio/i3/AWG_Digital_Library/pdfdocs/fw.pdf

[2] Department of Defense, Joint Chiefs of Staff, *Joint Vision 2020*, June 2000. http://www.dtic.mil/jv2020/jvpub2.htm

[3] G. Beeler et. al., *Message Development Framework,* Health Level Seven, Inc., 1999.
http://www.hl7.org/Library/mdf99/mdf99.pdf

[4] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web", *Scientific American,* May 2001.
http://www.sciam.com/2001/0501issue/0501berners-lee.html

[5] S. Renner and J. Scarano, "Organization, Process, and Technical Foundation for Data Interoperability", *DoD Database Colloquium*, San Diego, September 1997.

[6] S. Renner, A. Rosenthal, and J. Scarano, “Data Interoperability: Standardization or Mediation”, *1st IEEE Metadata Conference*, Silver Spring, MD, 1996. http://www.nml.org/resources/misc/metadata/proceedings/renner/data-interop.html

[7] A. Rosenthal, E. Sciore, S. Renner, “Toward Integrated Metadata for the Department of Defense”, *2nd IEEE Metadata Conference*, Silver Spring, MD, 1997. http://www.llnl.gov/liv_comp/metadata

Author Biography

Dr. Scott A. Renner is a Principal Scientist in MITRE’s Center for Air Force C2 Systems at Bedford, MA. He has a Ph.D. in computer science from the University of Illinois at Urbana-Champaign. He participated in the USAF Scientific Advisory Board studies *Building the Joint Battlespace Infosphere* and *Database Migration for Command and Control*, and is a contributor to the *DoD-wide Data Interoperability* Rapid Improvement Team.