From Semantic Integration to Semantics Management: Case Studies and a Way Forward

Arnon Rosenthal
The MITRE Corporation
Bedford, MA, USA
arnie@mitre.org

Len Seligman
The MITRE Corporation
McLean, VA, USA
seligman@mitre.org

Scott Renner
The MITRE Corporation
Bedford, MA, USA
sar@mitre.org

1. Introduction

For meaningful information exchange or integration, providers and consumers need compatible semantics between source and target systems. It is widely recognized that achieving this semantic integration is very costly. Nearly all the published research concerns how system integrators can discover and exploit semantic knowledge in order to better share data among the systems they already have. This research is very important, but to make the greatest impact, we must go beyond after-the-fact semantic integration among existing systems, to actively guiding semantic choices in new ontologies and systems – e.g., what concepts should be used as descriptive vocabularies for existing data, or as definitions for newly built systems. The goal is to ease data sharing for both new and old systems, to ensure that needed data is actually collected, and to maximize over time the business value of an enterprise's information systems.

The shift from semantic integration to *semantics management* requires the following:

- We need to proactively produce new areas of useful semantic agreement, and not simply document correspondences among existing systems. This will help the enterprise satisfy new requirements—e.g., to collect new data or establish new data sharing arrangements. This will also reduce costs by reducing unneeded semantic and representation diversity.
- We must consider more than the needs of technology-savvy system integrators. We also must assist people in many roles—e.g., enterprise owners, architects, end users, as well as developers—to have a greater shared understanding of what the data means. Without this, it is impossible to evolve information systems which truly meet the business needs of the enterprise.
- We must broaden our definition of "semantics" to describe what data instances are collected and desired (as in publish/subscribe advertisements), not just concept definitions and relationships. For example, we may agree on the meaning of "Aircraft" (say "fixed wing, not including helicopters or balloons"), yet if we do not manage the scope of data collected, our applications will not usefully share data. For example, you may provide data on aircraft operated by U.S. airlines, while I want data on all aircraft that regularly use U.S. airports. To properly interpret the data, I must understand what portion of my information need you can actually satisfy. We must also manage *desired* instance populations (i.e., the subscriber side), to guide future data collection and exploitation.

To be successful, an approach to semantics management must tolerate organizational realities that are often ignored: e.g., limited central authority (for example, of the Chief Information Officer), and the difficulty of enforcing management directives to supply rich, accurate metadata (and to update it with each system change).

The managers' task is not to build a "perfect" system in which each participant can seamlessly share information with every other participant, or to build an entirely new system. Their goal is to use their levers of limited influence (e.g. policy, money, reusable metadata, free tools, consulting support) to steer the enterprise from the imperfect status quo to a better (but still imperfect) future state.

To enable the new management approaches, technologists must advise on methodologies (and the architectures implicit in many of them), provide formal models to express managers' decisions, and tools to help make and analyze these decisions. As consultants, technologists also often identify unintended consequences or incentives of proposed mandates. In this way, we help create *implementable* guidance (specific advice or required products) to managers, developers, and administrators.

This paper makes the following contributions:

- We describe a reference model for characterizing data standards, particularly semantic agreements.
- We present data standardization case studies, lessons learned, and implications for effective semantics management.
- We introduce a semantics management approach based on "communities of interest"

that we believe respects organizational realities that are too often ignored.

- We describe research challenges that can advance tools and methodologies for the community of interest approach.

Large organizations will undoubtedly use a variety of formalisms, some traditional and some emerging (e.g., semantic web). For wide applicability, we thus make our main points formalism neutral, where possible.

2. Reference Model for Comparing Types of Data Standards

We now present a set of axes for describing a standardization approach. This helps us understand essential features of previous approaches, and also helps frame questions about what to standardize and how to proceed.

We begin with terminology from the ISO 11179 metadata registries standard [ISO99], which distinguishes a *data element concept* (i.e., meaning) from its value's representation (i.e., datatype, domain, units of measure); the combination of meaning and representation is called a *data element*.[1]

1. *What types of objects are being standardized?*

- What is the proper granule of semantics to standardize (e.g., data element concept, data element, a full schema)?
- Does the standard specify representation(s)? How many representations can one specify for a concept (0, 1, multiple). When there is only one, conformance is harder, but data is plug compatible – values need no mediation.
- What kinds of constraints and relationships among elements does the standard capture? Answers range from dictionaries of stand-alone element definitions, through XML or object schemas with constraint constructs, to ontologies with richer assertions and logical inference.
- If the standard defines schemas or ontologies, does it also specify instance sets? (E.g., what are country code values and their associated countries)?

2. *How will the standard structures (ontologies or schemas) be used?*

- Is the standard solely a descriptive vocabulary (e.g., an ontology) that expresses semantics, or is it also a schema to be instantiated? If the latter, then is the schema:.
- Implemented natively within a system?
- An additional interface that a system should support? If so, what operations should that interface support—e.g., the operations a typical DBMS supports on views)?
- An interchange format—i.e., the standard defines a structure for exported objects (files or messages), which will be physically materialized and then delivered to consumers?

3. *What are the characteristics of the community developing the standard?*

- Is there one primary stakeholder with most or all of the decision making authority (e.g., the Internal Revenue Service for U.S. tax submission formats) or is authority distributed?
- What are participants' obligations to support the standard? (e.g., use it for all new development, use it as one of multiple interfaces, map all legacy systems to the standard within a certain time period)
- Is there a pre-existing shared understanding of the domain? Standardization is easier where this exists (e.g., purchase orders).

3. Case Studies

We present three examples of semantics management, drawn from our experience with the U.S. Department of Defense (DoD). (This paper expresses the authors' views, not MITRE or government positions.)

3.1 DoD Data Administration Program

Like any giant organization, the DoD is plagued with data interoperability problems caused by differences in data semantics and representation. The DoD is now in the process of replacing its data administration policy and procedures. The previous effort (officially running from 1991 until 2003) was an attempt at enterprise data standardization. It attempted to eliminate these problems by eliminating the differences, DoD-wide. In the terms of our framework, the former data administration program:

- *Standardized individual data elements*, for publisher and subscriber systems
- *Did not support agreement on data element concepts* (independent of representation).
- *Allowed only one standard representation* for each concept.[2]

[1] The case studies below do not exploit (and sometimes predate) the ISO standard, which rather recently added APIs for tool vendors. Without supporting tools, the standard has little payback.

[2] The above choices allow a standard data element to be passed between systems, without extra

- *Captured no structural semantics.* The official standard addresses just data elements. However, organizations registered conceptual schemas (in IDEF1X, an extended entity-relationship formalism). These schemas were not maintained as the actual systems evolved. Occasionally they were used to educate new staff, and developers of new systems borrowed portions unofficially.
- *Provided no formalism for describing the instances a publisher offers, or a consumer wants.* Information needed for mediated publish/subscribe was deferred.
- *Defined standards intended to be implemented internally, by each system, for all data.*

The DoD data standardization effort aimed to produce a single, logical, fully-attributed data model of immense scale, probably exceeding 10^5 entities and 10^6 attributes. Today it seems impossible that such an effort could ever succeed, and in fact, by 2000, the need for a new approach was widely recognized. The program was costly, and the ~12,000 registered data elements were infrequently reused, and so did little to promote interoperability.

The process charged with deconflicting data elements in the standard was ineffective for several predictable reasons. Sometimes there are legitimate needs for several forms (e.g., to trade precision versus bandwidth). Other times, existing communities (e.g.., Army, Navy) wanted to continue using their own standards. In addition, the process introduced delays for developers. Most important, the process created a disincentive to agree on semantics, since agreement usually meant that somebody had to change implementation.[3]

However, within the overall program we have found several pockets of success. For example, the US Strategic Command successfully reengineered their war-planning systems to use a single data model. In almost every success, we found:

- A standard data model of reasonable size; the largest plausible success has ≈1,000 tables.
- A cohesive local enterprise, with relatively few, well-defined interfaces to the external world
- A single authority exercising effective control over the system requirements, the funding, the developers, and the users.

3.2 Meteorology Data Standard

This is a DoD-wide standard for meteorology (weather) and oceanography data. It contains on the order of 1,000 attributes, and was developed over a period of five years, at a total cost easily exceeding $1M. It was developed as a part of the data administration program described above, and in our framework it has almost the same description; the difference is that this is a conceptual model intended to specify interfaces and data exchange formats and was not intended to be implemented natively by participating systems. The standard is successfully used by several systems that exchange weather data. This is an exception to the single-authority pattern described above, because these systems are *not* built or operated under any single authority. We explain this in part by observing the pre-existing shared understanding of the domain. Concepts like "dry adiabatic lapse rate" are the same everywhere for everyone, and graduate students have spent years acquiring this shared body of knowledge.

3.3 "Cursor-On-Target" Schema

Cursor on Target is an XML schema for documents that describe the time and geographic location of an event of interest. The underlying model is very simple: 3 entities, 13 attributes. It is used to share targeting information between automated systems. In terms of our framework, this particular effort:

- Standardized an entire schema
- Did not define free-floating data elements independently of the schema structure
- Specified a single representation for each data element
- Captured the relationship constraints that XML schemas could express
- Made no mention of instance populations
- Defined an interchange format

This effort has been judged a huge success, going from concept to operations in under 18

validation or conversion. However, it gave little help in shipping data between systems that already used different representations (e.g., feet versus meters, GIF versus JPG, 32 versus 64 bits). An unfortunate side effect is that each participant contributed their own definition to the standard independently, differing only in representation details. The registry captured no similarity between notions like Fuel_Load in liters, or gallons.

[3] It is interesting to compare DoD's flat concept space with the ontology of Cyc [Lenat95]. The careful structure and separation of concept and representation are helpful. Still, Cyc's goal is more for semantic integration than active management of semantics. If one allows developers to choose freely from Cyc's 10^6 concepts, their systems will have too many inconsistent semantic choices.

months.[4] Many systems in many organizations have implemented the standard – another exception to our single-authority rule. We observe that it is feasible to get fairly widespread agreement (e.g., among approximately 40 systems) on a very small set of definitions, especially when the end-user demand is strong. Nevertheless, many other systems will not implement this schema, preferring standards produced by other communities.

3.4 Lessons

The first lesson seems obvious: a very large enterprise cannot hope to construct a single data model (or even a single set of universally-understood concept definitions) for all the data it requires. The "eliminate diversity" approach promised simplicity, abstracted into a hub and spokes model. But such simplicity could not be delivered, and the hub and spokes model gave little help in dealing with messy realities. Approaches that require perfect coordination and altruism are of no practical interest.

The second lesson explains the first. Semantic agreement comes at a cost, and that cost is driven both by the number of people who require a shared understanding, and the number of concepts they must all understand. The cost element appears to be the *person-concept*. We are aware of many examples of small numbers of participants agreeing on large, complex standards (e.g., meteorology) and of larger numbers of participants agreeing on modest standards (e.g., cursor on target), but we have seen few successes where large numbers of autonomous participants agreed to a large, complex standard.

Third, practical semantics management must recognize the limited ability of managers to enforce conformance to standards. Clearly, this is true in *megasystems*—i.e., "enterprises" where no person has authority over all parts of a system (e.g., Health Care, and decentralized institutions like the DoD or Intelligence Community). But enforcing data standards is problematic even at a smaller scale where there is centralized authority.

The problem is that a good manager's preferences are not absolute–he must recognize that some systems in his community may have good reasons not to conform. For example, in MITRE, a 5000-person company, the CIO can select standards for the company's operational systems. Signing a mandate (e.g., to select a data standard or an architecture) constitutes (only) an influence on future actions by our own enterprise. He cannot decree that the legacy data and systems will be instantly upgraded. He cannot affect the standards of our government partners or of the important packages that we buy (e.g., Microsoft Outlook, PeopleSoft). There may also be applications with special requirements (e.g., extreme precision, host country rules, close integration with a customer or COTS or legacy system) for which he must allow violations of the organizational standard, even for new development.

[4] Development often takes several years in similar military systems, in part because of rigorous test and certification procedures.

4. Progressing via "Communities of Interest"

Having learned from previous experiences, the DoD recently published a new data strategy [DoD03]. Semantics are to be managed "within *communities of interest* (COIs) rather than standardizing data elements across the Department." The strategy defines a COI as a set of stakeholders "who must exchange information in pursuit of their shared goals, interests, missions, or business processes and who therefore must have shared vocabulary for the information they exchange." Community stakeholders include users that participate in information exchanges, developers that build systems for these users, enterprise architects that define requirements based on mission needs, and managers that acquire systems on behalf of the users.

The above is just a first step toward defining the COI approach. As yet, it says nothing about how communities are formed, who should be included, what authority do they have, how do they collaborate with other communities, and (perhaps most important to specify carefully) what do they produce. DoD is working to address these questions; in this section, we give our own (emerging) proposals.

We focus on defining community concepts and tasks that should be useful in many different methodologies. We anticipate that communities will be run in many different ways, depending on size, culture, time available, and so forth. Following the UML approach, we define conceptual categories, and products to be produced, rather than the detailed process that produces the product. Communities will be loosely coupled – each can choose its internal processes, and groups of them can choose how they collaborate to form new communities. The focus on products also should enable research ideas and tools to be used in many settings.

Three basic tasks that communities must accomplish to enable data sharing are *create definitions, adopt definitions (or more generally,* *state preferences),* and *form agreements.* One innovation here is to allow a different community to be formed for each type of task. This enables each community to be concerned with different sets of information, e.g., "DoD spatial data" versus "data required to plan an air transport mission". We now examine the three types (identified by their products) in more detail.

1. *Definition-development communities.* These produce concept definitions, element definitions, ontologies, and schemas. They seem best organized by subject matter area, spanning organizational boundaries (including external organizations). They have free speech (e.g., to publish their proposals), but no command authority, because the span of useful data sharing greatly exceeds the span of organizational control. They merely publish definitional resources (e.g., ontologies, schemas), and may express preferences among alternatives. The Cursor-on-Target effort (section 3.3) is an example. Diverse participants contributed to development of the interchange standard, but they had no authority to impose the use of that standard on anyone else.
2. *Custodial / Authority communities.* Each aspect of a data object's semantics (conceptual meaning, instance set) has a custodian. Custodians have two kinds of responsibility:
    - *To describe the systems currently in place.* For this they need to choose a descriptive vocabulary. For example, an organization might tell its system managers "Describe spatial data using the OGIS ontology, or (second choice) the ARC-INFO GIS schema. Describe measurement data using NIST's measurement ontology or ISO". Even with alternatives, descriptions may be imperfect.
    - *To guide (semantic and other) choices on future data collections or software development.*

    Other authorities may be able to express preferences that the custodian considers (e.g., partners, higher and lower management levels).
3. *Agreement communities.* These are groups of participants who agree to provide or consume information using certain definitions. Agreements must specify semantic concepts (so the recipient will be sure that they can understand the data they receive). Often they specify instance sets, e.g., FuelNeeded for all units at a given Airbase. (One does not leave it to chance that someone is tasked to obtain the information their downstream successors need). Agreements can also include representations (to reduce mediation effort). Agreement areas beyond data semantics (services, workflows, secure handling) will not be discussed here.

Organizations and consortia often function as more than one type of community. For example, a standards group may both develop definitions and secure agreement from participants to use those definitions. Some participants may also have authority to express a preference that their subordinates use the standard. However, we find it useful to identify these distinct community roles to guide what products they should produce and how they should operate.

Across communities, reuse is essential, to avoid unnecessary definitional differences. Many concepts (and organizations) belong to multiple communities. Definitions should be published in small granules—i.e., data element concepts, independent of representation. COIs can also publish standard structures (e.g., an XML exchange schema), but there must be a way for other communities to reference and reuse individual concepts independent of these structures.

Summary of Pragmatic Advice

In terms of our standardization framework (Section 2), and based on the experiences described in Section 3, we believe successful methodologies will adopt the following:

- Standardize in small granules—i.e., data element concepts, but assert conformance in biggest feasible chunks [CDFP98].
- Specify representation separate from meaning. Allow alternative representations, and provide mechanisms for expressing preferences (Section 5.2).
- Describe explicitly the set of instances managed by systems and desired by users, to ensure that needed data is actually collected and exported.
- Be robust with both simple and rich modeling formalisms. Capturing richer constraints and relationships (e.g., in RDF or OWL) will be useful. However, organizations must be allowed to govern technology change based on staff expertise and tool support.
- Focus on interfaces and exchanged information, not systems' internal representations.

5. Related Work and Research Agenda

Most research assumes that higher authority is either all powerful (centralized) or absent (peer to peer). However, there has been little research on enterprises, where managers have partial influence.

Schema matching research is active and promising, and is covered in other articles in this issue. Commercially, many tools can capture semantic relationships graphically and generate code. Unfortunately, they lock the knowledge into that vendor's suite by using proprietary schemas and representation ontologies.

Reuse and incentives have been emphasized in only a few papers. In [RS94, RSRM01], we emphasized reuse, because it bypassed the many difficult research problems of matching. We also advocated learning the lessons of the *industrial revolution,* i.e., separation of different tasks, incentives, parts usable in different finished products within a product line.

More recent research has explored particular reuse techniques in more depth. Both schema matching and semantic web researchers have noted that transitive laws enable new matches to be inferred from older ones. [HEDI03] advocated incentives ("instant gratification") for individual users, and [MDKV03] explored the effectiveness of P2P approaches to gathering information, for matching a consumer application to many sources for the same domain.

There are many areas where researchers could help put the Community of Interest approach on a firmer footing, with better abstractions and automated support.

5.1 Methodologies and Tools

The conceptual modeling literature is extensive, but rarely stresses reuse, separate models of semantics and representation, and other enablers of data sharing. Schema integration techniques also fit poorly for designing standard models – one rarely has clear consensus on what the future "standard" should model, or integrates entire schemas that are all available in advance. A principled reformulation is needed, to identify characteristics of a "good" model, perhaps with different answers for interchange schemas versus descriptive ontologies.

The technical problem is not "provide a process that can be carried out by omniscient saints". The process must be robust against imperfection, such as participants who expertly evade directives that hurt their budgets. For that reason, one needs *model-driven* tools that repay metadata providers by supporting auto-generation of needed software interfaces.

Enterprise scale semantics management requires new kinds of tools. Imagine a metadata registry with O(10^3) schemas, O(10^5) data element concepts, and thousands of assertions about mappings, preferences, and data quality. What tools are necessary to make it all useful—e.g., how does one find definitions, schemas, etc. relevant to one's information need? Today's point-to-point schema matchers are a first step; P2P approaches are intriguing but make it difficult to assign contractor responsibility. Research breakthroughs are needed.

5.2 How to Transmit Authority -- "Preferences" as a Primitive

Any large scale management process will often assert preferences rather than absolute constraints and split decisions among definitional, custodial/authority, and agreement communities (and roles within them, e.g., data collectors, software builders). What formalisms, theories and tools are needed for their cooperation?

Can researchers devise a single construct for all "Preferences"?—e.g., a community's preferences among overlapping standards, a manager's preferences to be communicated to suborganizations, or the set of definitions used by a system. Opinions will need to be aggregated, and differences negotiated. Precision is less important than minimizing administrative burdens. Preferences could then drive tools, e.g., to select defaults or order the responses from a discovery tool.

5.3 Metrics

Metrics are important for demonstrating and measuring progress. Even imperfect metrics can guide tools that assist humans (e.g., to suggest definition choices that minimizes a metric of community diversity). Metrics collection must be largely automated.

Useful system metrics might include:

- amount of data sharing enabled
- amount of glue code needed to handle representation diversity (lines of code, info loss, by conversion, time for conversion)
- quality of data shared -- closeness of definition match, plus precision, recency, etc. (Quality is a widely underestimated difficulty.)

5.4 Helping Subscribers Understand the Responses

Semantics management also includes steps to guarantee that someone collects the data that applications need. In difficult cases, there may be a shortfall, which managers, application developers, and in some cases the user receiving the response need to understand.

The "constraint" model explored in [FKMP03] can describe what instances should be in a query response—namely all "certain" answers. This criterion is appealing to theoretically inclined CS PhDs, but how do we tell others (e.g., a doctor asking about dangerous drug interactions, or a military planner who needs to know about friendly forces near a target) what they can rely on about a response's completeness? What if their query goes outside the classes (e.g., conjunctive) that are well behaved?

A second problem is communicating the answer the theory gives. Even CS PhDs obtain little insight from reading the difference between two complicated SQL queries. What should be presented to application developers, or end users? Perhaps they should be shown examples, or an easy-to-understand bound, with option for more, or a union of simple parameterized cases? Queries to a warehouse may not receive answers as complete as sources might provide. How can one help an application developer decide whether it is important to drill down to the original sources? (We conjectured about this phenomenon in [RSRM01], and it was shown to occur in [FKMP03].)

5.5 Optimization

As noted above, model-driven tools can reward participants for providing the accurately maintained metadata that will enable progress. The goal is to generate needed code and interfaces from declarative models. This will require advances in query optimization for complex, multisystem environments. How can one model the capabilities of rich (multilingual, multiprotocol) systems in ways a query processor can understand? We anticipate mixes of SQL views, XSLT, XQuery, and libraries of transformation functions, to execute in DBMSs, application servers, gateways, Extract-Transform-Load tools, and replicators. Can existing techniques for characterizing and discovering servers' capabilities and efficiency (e.g., [BFMV00]) be applied at this scale?

Also, optimization in this environment is currently performed at design time by skilled distributed systems programmers. How can we introduce automation gradually?

- How do you automatically generate partial solutions that will be convenient for the human expert who must complete the job?
- Where does control of the optimization process lie?
- How should compilation units be managed, from clusters of knowledge (metadata and mappings)?

References

[BFMV00] L. Bouganim, F. Fabret, C. Mohan, P. Valduriez, "A Dynamic Query Processing Architecture for Data Integration Systems," *Data Engeineering*, 23(2), June 2000

[CDFP98] S. Castano, V. deAntonellis, M. Fugini, B. Pernici, "Conceptual Schema Analysis: Techniques and Applications", *ACM Transactions on Database Systems*, 23(3), Sept. 1998

[DoD03] Department of Defense Net-Centric Data Strategy, May 9, 2003, http://diides.ncr.disa.mil/mdreg/user/index.cfm?id=48

[FKMP03] R. Fagin, P. Kolaitis, R. Miller, L. Popa "Data Exchange: Semantics and Query Answering" ICDT, 2003.

[HEDI03] A. Halevy, O. Etzioni, A. Doan, Z. Ives, J. Madhavan, L. McDowell and I. Tatarinov, "Crossing the Structure Chasm," *Proc. First Conf. on Innovative Data Systems Research,* 2003

[ISO99] ISO 11179-1, "Information technology — Specification and standardization of data elements, Part 1: Framework for the specification and standardization of data elements," First edition, International Standards Organization, Dec. 1999

[Lenat95] D. Lenat, "Cyc: A Large-Scale Investment in Knowledge Infrastructure," *Communications of the ACM*, 38(11), Nov. 1995

[MDKV] R. McCann, A. Doan, A. Kramnik, V. Varadarajan.. "Building Data Integration Systems via Mass Collaboration", WebDB03 at SIGMOD03

[RSRM01] A. Rosenthal, L. Seligman, S. Renner, F. Manola, "Data Integration Needs an Industrial Revolution," *International Workshop on Foundations Of Models For Information Integration (FMII)*, Viterbo Italy, 2001

[RS01] A. Rosenthal, L. Seligman, "Scalability Issues in Data Integration", *AFCEA Federal Database Conference*, 2001

[RS94] A. Rosenthal, L. Seligman, "Data Integration in the Large: The Challenge of Reuse", *Conf. on Very Large Data Bases*, Sept. 1994

Acknowledgments

The authors thank the very helpful comments of the special issue editors, Ken Laskey, James Caverlee, Joe DeRosa, Joe Wood, and Mike Kuras.