Submitted to VLDB 2001 USA

Data Integration Needs an Industrial Revolution

Arnon Rosenthal, Scott Renner, Len Seligman, Frank Manola
The MITRE Corporation {arnie, sar, seligman, fmanola }@mitre.org

1. INTRODUCTION

The data resources in a large enterprise typically exist as many separate islands of data. Each is maintained by a distinct community for its purposes, and is largely unusable by others. It is common to see whole data archipelagos comprised of thousands of separate resources [Ston00]. We would, of course, prefer to see one single integrated data resource usable by all. This is the "grand vision" of data integration: discovery of and access to all data, with multiple sources properly combined, delivered in a form that each consumer can interpret. Decentralized organizations such as the US Air Force might accept for now a slightly less ambitious dream – the ability to establish a connection between islands, a way to obtain any desired information from any other source.

We know of several attempts at large-scale data integration, but few success stories – and none where the individual data islands enjoy great autonomy or must make large changes to conform. (Data warehouse projects do not count here, as they typically cover a small fraction of the enterprise's data.)

While failure is often blamed on a lack of commitment or resources, we suspect that doubling the budget would only lead to a failure twice as costly. Instead, these failures are really the result of an impractical technical vision, one that does not match any realistic acquisition process or human incentives. The failed vision assumes that diversity can be eliminated through standardization, and ignores the continual series of projects that create point-to-point connections or (relatively) local warehouses that meet high priority needs. Worse, it assumes that people will follow mandates (e.g., to provide good metadata and maintain it as the system evolves), even when compliance does not benefit them and cannot be checked. All are false assumptions for large enterprises with autonomous partners and constant change.

We offer an alternate vision, one in which data integration is transformed into an *industrial process* through changes inspired by the Industrial Revolution in manufacturing, a process which produces *data connections* between islands, or between an existing island and a new integrating view. These connections are the software and operating procedures that enable meaningful information exchange among participating systems. In our vision these connections are built faster and cheaper, with much better utilization of scarce labor, in such a way that building each connection also produces knowledge capital for building the next one.

We believe that our vision is feasible – *if supported by the necessary tool suites and practices*. Our goal in this paper is to present the vision and to focus on novel or understudied aspects of the proposed system and research agenda. Section 2 will consider the important aspects of an industrial revolution in data integration. Section 3 will describe the sort of industrial process that this revolution leads to. Section 4 proposes a research agenda directed toward this new vision. Section 5 discusses challenge problems that might provide realism and synergy among research projects.

2. DATA INTEGRATION AND THE INDUSTRIAL REVOLUTION

The term “industrial revolution” was coined to describe the important social and technological change in England that occurred around the turn of the 19th century. Since then the term has been more broadly applied, to include other innovations that have useful software analogies, such as product lines and automated assembly. Two aspects of this change are especially relevant to our vision:

- *Specialization of labor and interchangeable parts.* In 1797, muskets were manufactured with each gun a separate project, carried out by one craftsman skilled in all the necessary tasks. After Eli Whitney’s innovation, they were created by a series of tasks performed by several specialized, less-skilled workers. The outputs of the separate tasks were then assembled into the finished product. Any part could be used in any instance of the same product line.

- *Incentives and obligations:* Less remarked but just as important is the development of incentives, organizational practices (e.g., quality control), and commercial codes that allowed enterprises to take advantage of the new production techniques, and to collaborate with each other. These apply both to the internal organization of the enterprise, and to its external relations with others.

We use these aspects of the industrial revolution as an analogy to the changes we want to see in data integration. Like any analogy, some parts do not fit. Perhaps it is unnecessary to say that our vision does *not* include turning data integrators into exploited factory workers with mindless, repetitive jobs. And, unlike manufacturers, we are not interested in producing thousands of identical copies of the same connection. But we believe that the industrial techniques that do apply to data connections can provide the same explosive growth in productivity as the original revolution

Today, data connections are almost always produced by one (or a few) programmers each having *all* of the necessary skills and knowledge. But people who understand application programming, distributed systems programming, database management systems, *and* the subject domain are rare and thus expensive. Specialization of labor offers a huge advantage. We aim to empower experts in *one* of these to work as specialists, without needing all skills simultaneously to make progress.

Today, data connections are almost always produced as a unique monolithic artifact. When we divide the labor into tasks performed by specialists, we also aim to make the “parts” they produce into specifications which can be reused in other (different) connections. Our specialists will produce “interchangeable parts” that can be used anywhere in the data connection “product line”, even though we never produce more than one copy of any connection.

In the manufacturing revolution, incentives and obligations applied to factory line workers, foremen, factory owners, cartels and conglomerates, etc. In our data integration revolution, we can categorize the participants as:

- Program managers (PMs) of individual systems and their staff
- Program executive offices (PEOs), which manage collections of systems [1]
- Domain coalitions, of autonomous participants

---

[1] This vocabulary originates in the US Department of Defense. Other enterprises will use different terms.

- Cross-system developers (the connection builders)

Our revolution eventually will need funding from the first two categories of management, but it can also begin bottom up, if vendors provide tools that make it easier to “describe and generate” than to “code it all”. It seems worth pointing out that Whitney required ten years of funding before his process succeeded. Today’s military and commercial managers are less patient. It will be necessary to carry out the change in small steps, each of which has positive payoff. Fortunately, this makes good technical sense as well.

Our approach depends on making “capital goods” out of the metadata specifications produced by specialists. But capital goods give no long term benefit if their custodians have no incentives to maintain them. This truism has been observed with international aid projects, but applies even more strongly to metadata used in documenting software systems (e.g., conceptual data models, task models). We must have an approach in which the necessary metadata is either automatically generated from the executable software, or (better) is actually a part of that software. We will not succeed if metadata is mere “documentation” to the people who must create and maintain it.

Coalitions (of domain users, or of several related programs) will often be needed. They can develop shared vocabularies, and also conceptual schemas or message standards. The incentive here is a perceived need to cooperate – just as organizations meet on standards committees and consortia.

Finally, it is necessary to manage the obligations of data providers and consumers with an eye towards their individual incentives. What commitments are made when a data producer advertises a resource? Data producers must be able to make binding commitments to consumers, because many consumers will develop their own redundant copy of a resource if they cannot be certain of its future availability. But producers must also be able to advertise without making a permanent commitment; if not, many producers will conceal resources they would otherwise be willing to share. In either case, data integration opportunities are lost without flexibility in these sharing “contracts”. We expect that something like a “uniform commercial code” for data sharing contracts will supply the needed flexibility without requiring every pair of participants to negotiate every detail from scratch.

3. AN INDUSTRIAL PROCESS FOR DATA INTEGRATION

Our industrial process for data integration produces connections between data islands, or between one island and a new integrating view. In many ways it will be familiar to today’s developers – every functional task is one they perform now, so there is little need for retraining. The changes are driven by the principles of the industrial revolution: specialization of labor, interchangeable parts, incentives and obligations.

In describing the process we assume that producers have published descriptions of what they offer, and the consumer describes what they want. That is, they must document their knowledge in a machine-processable form. For simplicity, we assume they use the same metadata formalism, but (very likely) different vocabularies for domain constructs.[2] Then, to build the connection, developers must:

- Identify the differences that must be overcome by the connection

[2] Both assumptions may be relaxed. Explaining the process then becomes much more complicated.

- Find existing metadata “parts” specifying how to overcome some differences
- Create additional parts as needed for any differences that remain
- Assemble the parts into the desired connection

There are also broader management tasks, which span individual connection-building projects. A connection project might create a few vocabulary entries, but creation or import of entire vocabularies (or ontologies) requires more coordinated effort, perhaps organized by a domain coalition.

3.1 Identifying the differences between data islands

The hard part is to produce the right taxonomy of differences, the right specialization of labor. We need to partition the differences so that each difference can be tackled by a specialist and so that the solution parts can be composed into a complete connection. This task needs to be done once, by data integration researchers. The easy part is to compare the participants in one particular connection and enumerate the differences in each of these categories. This task is done once per connection, by whatever entity (human or automated) performs logical mediation when developing the data connection.

Many of the differences involve large sets of objects (e.g., entire databases) in a producer or consumer’s interface. These include (but are not limited to):

- *Request protocols* (e.g., CORBA, ODBC, message passing)
- *Representation of composite objects* (e.g., XML text)
- *Quality of service issues* (e.g., speed and availability of response, cost)
- *Delivery mode* (e.g., push or pull, object or recent changes)

For each data element, there is a large set of aspects to be described and mediated. These include:

- Element representation (gif or jpeg, meters or feet)
- Schema (entity descriptions)
- Element data quality (currency, accuracy when captured, precision …)
- Obligations (cost, availability, future availability)
- *Element physical information* (for automated physical design, e.g., size, volatility)

The next two items guide data merging. They may be only partially automated, and may require instance-specific guidance. Still, they can be modular, with information reused, and done by end user personnel (with some development by power users).

- *Entity identification.* Knowledge and services to help identify whether data entities refer to the same real world entity (e.g., George Bush, George W Bush, the President, W. Clinton).

- *Element value resolution:* Knowledge and services to help resolve data value conflicts (for the author’s weight, should one use 190 pounds from 1999, 7 pounds as stated in birth records, 200 pounds from unspecified date, or –9999, all of which might appear in accessible databases.)

Finally, for collections, one must describe the *scope* and the *completeness* of coverage. The scope can be expressed as a query on some well known conceptual schema, using an approach known as “source as view” or “information manifold” [Levy96]. For example, one source might say "I have a list of *all* nations in the UN, and current head of government", plus "A list of *some* nations and their oil export levels, 1970-present."

3.2 Find any existing, reusable "parts"

The "reusable parts" in our process are metadata descriptions with sufficient detail to form executable specifications of how to resolve one particular difference between participants. The next step is to search for existing useful specifications – which will have been produced in the past as part of building other connections – in available sources of information, which we loosely refer to as the "meta-database". This may be centralized for an organization, or distributed. Search may involve database query or web search engines. Initially, the search process will involve humans; as information is captured in forms which have known schema or XML tag sets, considerable automation will become possible.

If we find all the necessary parts – if we already have enough metadata to resolve all the differences between participants – then we can skip the next step and use the discovered parts to build the connection. In this way, metadata drives connection creation.

3.3 Create the missing parts

For each remaining difference, we need to have the appropriate specialist produce the "part" solution that resolves the difference. It is very important to capture these solutions explicitly, as a reusable specification, and not permit them to be hidden in procedural code. In this way, connection creation drives metadata capture.

Creating these parts can be done at several different levels within the enterprise. Each individual program office will have a group responsible for this step, as will each funded connection. These groups can provide individual vocabulary entries, matches across vocabularies, transformers, and so forth. Where there is management over collections of systems, there may be budgets to fund standards efforts and tool acquisitions relevant to all of them. Other efforts require attention across enterprises, where there is no common management, e.g., in formal or informal standards consortia. Finally, user enterprises can fund only very limited tools, so these often need to come from vendors.

3.4 Assemble the parts into the desired connection

This part of the process is difficult. Where one can define a federated view, DBMSs can handle execution. But an N-way view over all sources is unmaintainable when N is large, so other approaches (notably "source as view") are needed. These are not yet well supported in products.

Physical optimization is both extremely important and extremely difficult. Today, each connection builder uses programmers to get acceptable performance. This makes it costly to create and to change connections, and does not encourage more global optimization across multiple requests. (The alternative, also common and also inflexible, is to have a team agree on information to be multicast, with each consumer responsible for selecting what they want.)

4. RESEARCH AGENDA FOR SUPPORTING THE INDUSTRIAL PROCESS

4.1 Scope of this section

We focus on research issues involved in creating an integration environment that supports the "Industrial Process" approach for integrating data in enterprises or cross-enterprise consortia. We emphasize the issues of building a first integration environment, rather than (recursively) considering the issues of how to integrate across integration environments. Furthermore, we do not want to repeat existing surveys of the research needed to accomplish data interoperability (e.g. [Ram99]). Instead, we focus on identifying how our "industrial approach" changes the research agenda.

Our effort to scope this research agenda has been against the background of the "semantic web" initiatives from the W3C. The semantic web vision has been valuable to us: It *starts off* with many of the assumptions (decentralization, extensibility, diversity) that we are trying to add to the traditional process. But we exclude many general web issues that the W3C is addressing, e.g., supporting typed relationships among arbitrary web objects.

We see huge expenditures on creating and maintaining hand-built connections, so we are willing to consider technologies that reduce these costs and time delays, even if they do not (immediately) scale to the universe. For example, we can imagine selecting an integration product and capturing (or translating) knowledge to fit that product. Open-ness is good, but need not be included in each research project.

We address organizations that have an enormous legacy, of data, software, and business practices. On the one hand, this is helpful. They already spend millions on hand-crafting connections, and recognize that the process reduces their responsiveness. They are accustomed to integration projects that address a limited set of systems. They do not demand that effort and skill go to zero, immediately (though they are rightly cautious about strategies that require retraining).

In other ways, enterprise systems are more demanding than most web applications. Behavior must be correct – not only do we return web pages, but we feed data to automated applications that make decisions which affect the real world. Many millions are spent creating and maintaining interfaces (views) to satisfy different job functions – we want to reduce the cost of doing so. Finally, support for Update is essential – enterprise computing is not Read Only.

4.2 General Considerations

Some issues recur in many of the items below, so we state them once now.

For every technique, research issues include how to improve (and measure, at least anecdotally) usability by the intended kinds of users, ability to give automated help, run-time efficiency, and (not further discussed here) ability to be shared with other integration environments.

Many tools have been developed to address a single aspect of data integration (e.g., [Li00], [Mill00]). Such research is necessary but not sufficient. There are related research problems that can often have a greater impact on the ability to build systems. We present them from lower to higher priority.

First, in an area where there are many heuristics (e.g., name matching, object matching), one wants to know their relative effectiveness, and ways of determining whether they will be effective on a given problem. Second, it is important to find ways to combine the techniques into a system, which may be far more effective than any particular technique. Third, the most important technique – reuse – may have received the least attention (perhaps because it requires experimentation and evaluation, rather than cleverness). There may be dozens of techniques for acquiring knowledge (e.g., semantic matches) the first time – the leverage point is to create a structure and incentives so that such knowledge can be reused. That is a primary goal of our "industrial" approach.

Finally, information needs to pass among tools, e.g., from knowledge capture to logical mediation, and hence to physical mediation. There are major issues in defining such interfaces, to find appropriate tradeoff points between simplicity and power. For example, if the results produced by prior application of other tools are not satisfactory, how can later stages make known their quality requirements, or request further alternatives.

4.3 Knowledge Capture

To have an industrial process, we need strong support for defining, capturing, and managing the knowledge, so it can be exploited for multiple purposes. Knowledge must be captured from existing stores, from heuristics, and (crucially) from humans. Much information describing single systems is already available on-line. Heuristics can generate further information, but their results often require human approval before being used to drive data integration.

To begin, one must define what knowledge must be captured. We need to define schema constructs that describe the classes of knowledge that the integration system expects to exploit. First, we need a good specification of the differences, the dimensions along which one needs agreement or resolution of heterogeneity. (We presented an initial list of differences in section 3.1, but we have no illusions that this list is complete or correct.)

Next, one wants a way for humans and tools to express a degree of belief when they provide difficult kinds of metadata (e.g., identifying semantic matches between concepts). Downstream tools then will determine which of these they are willing to use. This information is costly to acquire, and important to share across tools, but seems rarely discussed in standards groups. Good freeware might be able to establish *conventions*. (These conventions might be commonly known because a major vendor or data provider uses them and might become de facto standards.) In section 5, we propose that data integration researchers "eat our own dogfood", subjecting ourselves to experimentation in this area.

In section 2 we noted the need to manage the obligations of data providers and consumers, in order to align local incentives with interoperability requirements. There has been little research on the requirements for data delivery contracts. How are these different from ordinary commercial contracts? We need a range of constructs, beginning with "this resource may vanish whenever I choose", through removal with notification, to obligations for continuing availability. Boilerplate contracts – standard templates with blanks to be filled in – might be a good 80% solution. As a next step, how would one provide a "write your own contract" wizard? How much power do we need in the logic (or other formalism) behind it? Finally, what enforcement services would we want the environment to provide, and what (if anything) might one add to current DBMSs or configuration management environments?

4.4 Tool creation

There is already active research in many tool areas. Dozens of heuristics have been published for inferring semantic match and entity instance match, or for using quality and other metadata to select among values from alternative sources. For semantic match, researchers have recently moved to the next stage, combining the heuristics. Below, we will discuss additional promising areas.

Much of the work in the ontology research community concerns building a single ontology. However, large enterprises need to manage multiple interrelated ontologies. When one wishes to describe an implemented system, one will need to refer to multiple ontologies (e.g., for Shipping, Warehousing, Geography, Accounting, etc).

Work with ontologies (and mappings between them) needs to be made more convenient. We do not want to work one element at a time – we need constructs for defining chunks of knowledge. We expect chunks to overlap heavily –one can very rarely define disjoint problem domains.

Another challenge is to help user*s* deal with the thousands of knowledge capture tasks, by managing the workflow of tasks to domain experts. If an integration engineer has a Warehousing expert on call today, she wants to see questions related to Warehousing entities. If the goal is to produce a demo prototype, one might accept heuristic decisions for most issues. [Ros91] contains some initial requirements and speculations about implementation for such a tool. Further, a large integration project has many participants – what sort of collaboration services are most valuable for data integration?

The metadatabase is itself a major "tool". It might include a central repository describing operational applications, but also may need metadata captured by individual tools, and text information and ontologies from the web. Managing it raises all the usual challenges of heterogeneous autonomous databases. It seems particularly important to automate administration, to minimize participants' burden and avoid creating a disincentive to supply metadata. In addition, to the extent that metadata drives creation of the executable code, research is needed in managing invalidation, and permitting updates without forcing the system to quiesce.

4.5 Creation of Logical Data Mappings (via mediation)

This section examines how one does logical mediation, i.e., constructs a logical view from all the available knowledge. The task includes both *discovering* relevant knowledge (across many sites) and the "mediation" algorithms that exploit it.

Discovery algorithms have long been an active research area, employing both structured information and text search engines. As logical mediators are built, their interactions with the discovery phase will certainly require attention. But our discussion will focus on logical mediation.

The two major approaches to logical mediation are to create a federated view, or to treat each source as a view over a conceptual schema (the "source as view" or "information manifold" approach [Levy96]). Query processing for federated databases has a long history and requires relatively few radical changes (except due to limits on sources' processing capabilities and cost modeling.).

For "source as view", query processing research has emphasized algorithms to find maximal contained rewrites for various classes of queries. An important extension would be to find ways to describe the difference between the desired result and the actual. Which portions of the desired result had no relevant sources, and (assuming we track sources' completeness) how complete is each part of the returned result? The research problem is to take a formal expression for the difference and to produce something users can understand.

A more radical issue is that we cannot imagine defining a single conceptual schema for an enormous enterprise, such as the US Department of Defense (not to mention its interfaces to external organizations). A more realistic mediation problem would have multiple conceptual schemas, with partial overlap and correspondences incompletely specified among them. In such an environment, how does one do logical mediation to derive the consumer's data? How does one suggest to humans what additional information might be captured (e.g., a connection between the consumer's ontology and that of a likely source)?

There are other "source as view" research issues more related to data administration. First, the scalability advantages presented for "source as view" are clear when the goal is just to return all relevant data. But if the scope of the administration problem includes matching similar entities and resolving value conflicts, do the advantages remain? Are there classes of federated views that are easier to maintain as sources come and go? (Note that source as view does still help one to examine coverage and residues).

4.6 Views Created by Logical Mediation Should Be "First Class"

Perhaps the most fundamental aspect of cross-organizational systems is that producers and consumers see the world differently, in terms of different objects. Our process yields declarative mappings between systems (essentially, views). We anticipate that consumers will soon find it inadequate that the only supported operations are Get and Query. Research is needed to provide two additional kinds of services – updates and metadata.

First, consumers who receive data will often want to change it, either by direct modification, or by submitting an annotation (e.g., a comment about quality, or a change request) that the system keeps as related information. The theory of view update will apply to these views; this is a major advantage compared with the 3GL programs frequently written for complex information flows. However, to resolve ambiguities, one often must capture additional knowledge. How will that be managed, if no human was responsible for synthesizing the view?

Second, without metadata, it is difficult for users to know how to interpret the data she has received. Users need metadata, e.g., timestamps, access control lists, lineage, units, accuracy, point of contact, etc., especially for derived values. (Security metadata is needed if the recipient is to know to whom else the data can be released). Rules to generate metadata could be added to the view definition, but they are so numerous that they must be generated automatically – especially for views generated automatically from previously-captured knowledge.

Lineage tracing becomes more urgent than in conventional systems, for two reasons. First, consumers need to know the original data sources and relevant processing history for important result values, to help them determine how much to trust the result. Second, there are important communities (academic research, intelligence analysis) where system providers have an incentive to make their data available (and to publish the needed high-quality metadata) if they believe they will get credit for providing the information. But existing definitions for tracing lineage don't work well enough to assign adequate credit.

4.7 Physical Optimization

Once one has logical mappings among systems, they must be translated into efficient implementations. There are several optimization research challenges:

First, mappings may be described using a wide variety of mechanisms including SQL views, XSLT, XQuery queries, and libraries of transformation functions. Given these different pieces (some of which may execute only in particular environments), how does one compile the mappings into an efficient execution strategy?

Second, the executable artifacts must be generated for a wide variety of components, including object-relational DBMSs, middleware products, mediators, gateways, XSL processors, replication tools, etc. How do we characterize the capabilities of these components in a way that is useful to optimizers? Issues include:

- What are the dimensions along which components must be characterized?
- How can the characterizations be generated and updated automatically?
- How do we characterize real products that span multiple aspects (pins)—e.g., Data Joiner, Oracle 9i, commercial Extract-Transform-Load tools, etc.

Currently, optimization in this environment is largely a manual process performed by highly-skilled, distributed systems programmers. Since full automation is not attainable in the near future, we need an incremental approach to optimization, in which the optimizer gradually fills in what had to be done by people before. Challenges include:

- How do you automatically generate pieces of the solution which are usable to the human expert who must complete the job?

- Where does control of the optimization process lie? Is the optimizer in charge, solving as much as it can and then specifying in human-understandable terms what remains to be done? Alternatively, must the process be controlled by a human expert who assigns well defined tasks to the optimizer?

- Can the optimizer take hints? What form do those take?

Another important issue is evolution of the executables: when a logical mapping or a component changes, one or more executables may need to change. How should this be done? How do we characterize the pattern of changes should we expect (e.g., how many are local to 1 attribute, how many are at the collection level, etc.)? How should we manage the necessary recompilations (e.g., priority, ways to do many at once, which to be eager and which lazy, etc.)? Do we need to change the whole executable, or should there be a module for each element that we could possibly change?

These are just a few of the optimization challenges motivated by the metaphor of an industrial revolution in data integration. In addition, these challenges must be met in a world with scanty or unreliable performance models, and major variations in available resources (bandwidth, fault tolerance).

5. SUMMARY AND CHALLENGE PROBLEMS

Many large organizations that need to share data among systems use software engineers and database administrators to build custom data bridges. The results are difficult to maintain and of little use the next time an integration requirement arises. Organizations that own individual systems are often reluctant to participate in interoperability initiatives because they perceive large costs and little benefit. Even when metadata is provided for these initiatives, it rapidly becomes obsolete. Standard schemas are hard to develop (one folklore estimate says 3 hours for each attribute) and even harder to connect to existing systems. Meanwhile, despite the many insights generated by years of data integration research, the research results are fragmented – we have many individual techniques but few results about how to combine them. Commercial data integration tools also seem each to deal with a narrow range of issues.

To break the logjam in data integration, we have proposed a new approach that exploits industrial metaphors – separation of skills, reusable parts, and incentives. In addition, we have described a research agenda designed to move us toward a more industrial process.

A good way to speed progress might be to have a challenge problem, for which data integration researchers are encouraged to submit freeware (code, ontologies, and data). The challenge problem should have value in its own right, and the challenge of combining resources will encourage our community to confront the real difficulties of data integration. While point techniques continue to be worthy of exploration, the emphasis now needs to be on integration environments, on how data integration techniques work together in context.

The challenge problem's goal should be appealing enough to attract researchers (probably without central coordination, although perhaps DARPA would be interested). We discuss two appealing external communities, and then discuss a more internal challenge. In all approaches, integration researchers would be encouraged to integrate their results – an "eat your own dogfood" test that will encourage focus on the critical problems of integration.

One appealing external problem might be biomedical informatics (notably for gene and protein researchers); another might be databases used for administering educational institutions. In both cases, there are large amounts of data, much of it publicly available, and created with decentralized responsibility. Both need small scale integration ("convert this information so it can be used with my interface") and large scale aggregations (for statistical or cross-species comparisons). Data volumes are high, so performance issues will arise.

The educational arena could serve both universities (e.g., preparing information for accreditors) and at other levels, e.g., for curriculum comparison and perhaps outcome comparisons. Also, while domain experts will be needed to explore shades of meaning, database researchers are familiar with the main concepts. The biomedical area is well funded, involves interesting new datatypes, has rich services as well as data, and (to us) has a greater excitement factor. In addition, there is a cleaner separation between data integration expert and domain expert.

Self-reference provides an alternative challenge – *how can we create a data integration system by integrating resources contributed by many independent researchers* who might have different external targets for integration. Here *we* must describe and integrate the semantics and representations. Database researchers have the knowledge to cope with immature tools (though we also need to experiment with more typical users). Finally, we will not be able to blame domain experts for preferring short-term hacks to the facilities provided – *we* will be the domain experts.

REFERENCES

[Bati86] C. Batini, M. Lenzerini, S. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *Computing Surveys*, 18(4), 1986

[Cui] Y. Cui, J. Widom, "Lineage Tracing for General Data Warehouse Transformations", Stanford University Technical Report, 2001.

[Doan00] A. Doan , P. Domingos , A. Levy, “Learning Source Descriptions for data integration,” *Proceedings of the International Workshop on The Web and Databases (WebDB),* 2000

[Levy96] A. Levy, A. Rajaraman, J. Ordille, “Querying Heterogeneous Information Sources Using Source Descriptions,” *VLDB*, 1996

[Li00] W. Li, C. Clifton, S. Liu, “Database Integration Using Neural Networks: Implementation and Experiences,” *Knowledge and Information Systems,* 2(1), 2000

[Mil01] R. Miller et. al., “The Clio Project: Managing Heterogeneity,” *ACM SIGMOD Record,* March 2001

[Mill00] R. J. Miller, L. M. Haas, M. Hernández, “Schema Mapping as Query Discovery,” *VLDB 2000*

[Mitr00] P. Mitra, G. Wiederhold, M. Kersten, “A Graph-Oriented Model for Articulation of Ontology Interdependencies,” *Proceedings EDBT 2000*

[Ouks99] A. Ouksel and A. Sheth, “Semantic Interoperability in Global Information Systems: A brief introduction to the research area and the special section,” *SIGMOD Record*, 28(1), March 1999

[Rahm01] E. Rahm, P. Bernstein, “On Matching Schemas Automatically,” Microsoft Research Technical Report MSR-TR-2001-17, Feb. 2001

[Ram99] S. Ram, V. Ramesh, “Schema Integration” Past, Current and Future”, in A. Elmagarmid, M. Rusinkiewicz, A. Sheth, (ed.) *Management of Heterogeneous and Autonomous Database Systems*, Morgan Kaufmann, 1999

[Ros91] A. Rosenthal, M. Siegel, "Toward Flexible, Extensible Tools for Metadata Integration", *Workshop on Information Technology Systems*, Cambridge MA, Dec. 1991

[Ros00] A. Rosenthal, F. Manola, S. Renner, “Getting Data to Applications: Why We Fail, and How We Can Do Better”, *AFCEA Federal Database Conference*, Sept. 2000 (see http://www.mitre.org/resources/centers/it/staffpages/arnie/).

[Shet92] A. Sheth, Kashyap, “So Far (Schematically) yet So Near (Semantically),” *DS-5*, 1992

[Ston00] M. Stonebraker, “Integrating Islands of Information”, *EAI Journal*, Sept 1999.http://www.eaijournal.com/DataIntegration/IntegrateIsland.asp

[Vass97] V. Vassalos and Y. Papakonstantinou, “Describing and Using Query Capabilities of Heterogeneous Sources,” *VLDB*, 1997

Acknowledgement: Chris Clifton provided useful perspective on several important issues.