Unpacking The Semantics of Source and Usage To Perform Semantic Reconciliation In Large-Scale Information Systems

Ken Smith, Leo Obrst
kps@mitre.org, obrst@mitre.org
The MITRE Corporation

Abstract

Semantic interoperability is a growing challenge in the United States Department of Defense (DoD). In this paper, we describe the basis of an infrastructure for the reconciliation of relevant, but semantically heterogeneous attribute values. Three types of information are described which can be used to infer the context of attributes, making explicit hidden semantic conflicts and making it possible to adjust values appropriately. Through an extended example, we show how an automated integration agent can derive the transformations necessary to perform four tasks in a simple semantic reconciliation.

1 Background

Over the past year, the MITRE corporation has undertaken several globally-scaled information integration projects on behalf of the United States Department of Defense (DoD). These efforts have been driven by a growing need to interoperate across the military services, and to free users from the conceptual gaps and biases inherent in a single information source. One result has been the emergence of the vision of an *Integrated Information Space* (IIS), in which users are provided with content-based access, via appropriate ontological concepts, to a set of integrated concepts based on information ultimately stored in numerous globally distributed sources. In an IIS, user communities can operate independently of information sources and be insulated from the details of their operation. By providing access to concepts which consult *many* sources, an IIS provides users a more complete treatment of their subjects of interest, and protects users from single-source bias.

An important obstacle to achieving an IIS is semantic heterogeneity. Consider the case of altitude data. In an orbital context, the concept of "altitude" is typically interpreted with respect to the earth's center. Therefore "orbital" altitude sensors produce altitude data in which "0" means the center of the earth. In an aviation context on the other hand, altitude is interpreted as vertical distance from the earth's surface. Since the space shuttle both orbits the earth and lands on a runway like an aircraft, altitude data on the space shuttle may be produced under both interpretations. If *semantic reconciliation* is not performed, this data cannot correctly interoperate. In fact, unreconciled data can "lie to you" in that a person or system's assumptions about meaning may result in an incorrect interpretation of values. For example, applying an orbital interpretation to an aviation altitude would make it appear that the space shuttle was somehow flying deep beneath the surface of the earth! Such "lies" may also be less dramatic, causing them to remain undetected until the integrated system is well into its use, if ever.

Note that this problem cannot be resolved by purely syntactic or structural approaches. Syntactic heterogeneity pertains to representational differences between data. For example altitude data could be given in meters or kilometers, or with differing significant digits, and in dates the month can be spelled out ("June 4th") or represented numerically ("6/4"). Structural heterogeneity pertains to differences in data structure. For example, web sites can structure similar content in many ways. However, these do not fully characterize the above problem: even if the format and units of an altitude measurement are reconciled, the interpretational difference remains. The *semantics*[1] of the data is an issue as well.

Some of the syntactic and structural aspects of integration are becoming much easier due to advances in middleware (e.g. [1,2]), emerging standards (e.g. XML namespaces), and sheer accumulation of experience. However no similar infrastructure is in place to help address this challenge of semantic reconciliation.

In this paper, we present some initial and practical insights into a semantic reconciliation infrastructure suitable for an IIS, and into an approach to semi-automate reconciliation. These insights are the basis of an ongoing prototyping effort, and are derived from the experiences of the authors and of numerous DoD "integration engineers" we interviewed. The infrastructure we describe includes: an architecture (Section 2), a reconciliation process (Section 3), and a set of explicitly represented semantic information about data sources, data usage, and canonical attributes (Section 4). In Section 5 we use an example to illustrate the automation of basic semantic reconciliation tasks using this framework.

---

[1] The issues we describe as "semantics" actually cover both semantics and pragmatics, with "pragmatics" more appropriate terminology for usage, user intent, and (generally) metadata aspects, and "semantics" more appropriate for the ontological coverage and the general "semantic" reconciliation process.

2 Architecture

The 4-level architecture shown in Figure 1 illustrates one approach to building an IIS, and is being followed in an ongoing DoD integration project. The highest levels (3 and 4), represent the user's view of the IIS. Level 4 consists of an ontology O of very general concepts (e.g. "event", "unit") to support content-based search. Typically, O is determined by an integration engineer working with the important user communities. Multiple user communities may interpret these concepts differently, as will be shown, but at this highest level we assume the list of terms in O are shared[2]. For a given user community (e.g. analyst, logistician), each concept $\mathbf{O_i} \in O$ in level 4 is in a 1-1 correspondence with an elaborated *reconciled concept* $\mathbf{RC_i}$ in level 3. A reconciled concept is a named class definition with a set of attributes, a set of instances (*reconciled objects*), and a hyperlinked structure (which we omit for the purposes of this paper). Intuitively, a reconciled concept contains an integrated view of "all the information space has to say" about its ontological concept. Note each user community may have its own specialized reconciled concept definition, similar to the construction of specialized database views. Therefore, multiple reconciled concepts may be associated with each ontological concept; the visible view depends on the user's community.

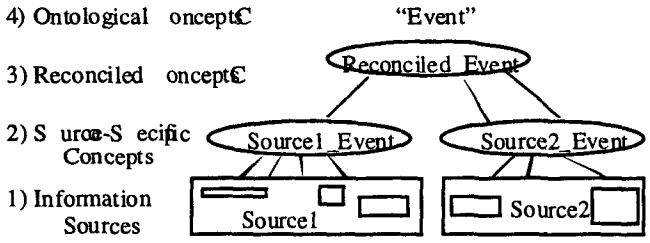

Figure 1. Architecture

The lowest levels (1 and 2) address the source-specificity of information. For each ontological concept a given information source contains pertinent information on, a corresponding source-specific concept (SSC) is produced at level 2. The production of an SSC from its source can involve denormalization (in the case of relational tables) to gather all related information into one object, and some simple data transformations where community-wide standards exist. As shown in Figure 1, a one-to-many relationship therefore exists between an RC and the *constituent SSCs* associated with it. Intuitively, for a given reconciled concept **RC**, a constituent SSC derived from source $\mathbf{S_i}$ contains "all $\mathbf{S_i}$ has to say about **RC**". The attribute labels and data values of an SSC instance need not match the attributes of its RC. In fact, attribute labels among a set of constituent SSCs are likely to vary because differing ontologies may be in use at the various data sources consulted.

Separating levels 2 and 3 in this manner has several values. First, the process and issues of semantic reconciliation are somewhat isolated from other types of reconciliation. Second, this modular approach promotes reuse, since the same set of SSCs can be repeatedly reconciled to the RCs of *multiple* user communities.

3 Semantic Reconciliation Process

Semantic reconciliation results in a set of RC instances with correctly interpreted values assigned to their attributes. In the following, we define four tasks comprising this process. Although we formalize these tasks somewhat for descriptive purposes, note that the following tasks are presently done without automated tools or a guiding formal framework.

1) *Ontology Development.* The IIS-ontology, the RCs and their attributes, and the SSCs and their attributes provide the basic framework for reconciliation. First, the set of terms in the IIS-ontology is developed in concert with relevant user communities. These are then further elaborated into a set of RCs, one set for each user community. (We assume a great deal of overlap.) Separately, a set of SSCs is developed for each data source

2) *Relevance Mapping.* For each attribute $\mathbf{A_i}$ of an RC, a set of *relevant* SSC attributes $\{\mathbf{A_{jk}}, \mathbf{A_{lm}}, ...\}$ (where $\mathbf{A_{jk}}$ is the kth attribute of the jth SSC) is identified which may affect its value. As shown in Figure 2, identifying relevant attributes can be straightforward: SSC attributes "pretax-salary" and "weekly-pay-amt" are relveant to the RC attribute "pay".

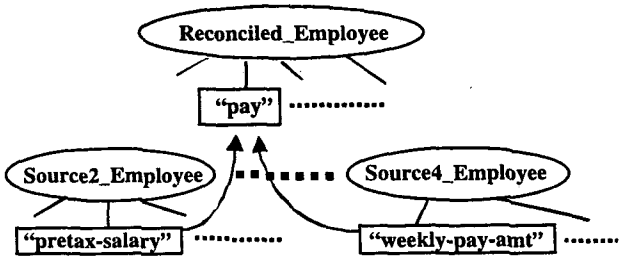

Figure 2. Relevant Attributes.

3) *SSC Value Transformation.* The value of each relevant SSC attribute is adjusted by a *semantic transformation function* $\mathbf{ST(A_{jk})}$, that compensates for the difference between the interpretation used in its data

[2] In level 4 we assume general terms can be effectively shared across our user communities, which appears satisfactory in practice. Well known problems involving multiple ontologies [3, 4] are encountered and addressed in levels 1-3.

source, and in user community of the RC it is relevant to, producing a "correct" value with respect to the RC. In the "altitude" example, a function transforming center-of-the-earth to surface-of-the-earth coordinates is used.

4) *RC Value Determination.* RC value A_i equals $f(ST_1(A_{jk}), ST_2(A_{lm}), ...)$. In the simplest case, f() simply performs a selection function over the relevant attributes, picking one (transformed) value from the list. If the list is exactly one value, selection trivially reduces to assignment. In practice, selection often involves a judgment call based on perceived quality of data sources, their accuracy, the freshness of their data, etc. In general, f() can be an arbitrary function. Functions computing *aggregates* present a special challenge, because they can raise the complicating issue of the semantics of set membership (as illustrated in Section 5).

4 Explicitly Represented Information

As data migrates from its environment-of-origin (in which it is presumably well understood) and into an integrated system, explicity represented information giving insight into the semantics of the data is needed to ensure the data continues to be interpreted properly in its new environment. This is one of the major challenges in data warehousing, and motivates the ongoing development of metadata repositories.

Such information also provides the framework for semantic reconciliation. For example, representing the *meaning* of each attribute, along with its value, enables us to detect a mismatch in the meanings of two related attributes, and to begin to develop a semantic transformation function between them. However, this particular approach can be costly in three regards:

1) Some representations of meaning are quite complex, requiring significant effort to encode and are often inaccessible to human integrators (e.g. various logics).
2) It is challenging to annotate a representation of "meaning" onto each attribute of all information sources which may need to interoperate; some relational DBMSs have 1000's of attributes.
3) Extensive information about data semantics must be maintained and evolved (in close coordination with source administrators), adding to what may already be a large administrative cost [5].

These considerations led us to seek *efficient* semantic information, such as simple information descriptive of large classes of attributes.

We began by studying the information valued by expert human users who routinely perform semantic reconciliation, and discovered such users heavily utilize *source-of-origin* information to determine how to interpret values from a source: "Source 1 is updated weekly, and during a crisis it's updated every half hour. Source 2 is only updated once a month, but it's more accurate because it's been through more levels of review." "That data source is mostly training data and is very dirty, however I make sure it's up-to-date in my topic areas, and use it for that purpose only." "Locations in that source are extremely accurate because they were derived by an accurate measurement system". Similarly, they value *usage* and user community information, as illustrated in our running "altitude" example, in which the attribute's intended usage (e.g. for orbital or aviation purposes) provides important cues to how it should be interpreted.

Sophisticated users clearly "unpack" the semantics inherent in a simple source label or usage descriptor to determine a specific *context* (e.g. "training", "orbital"...) in which to properly interpret data for the purposes of reconciliation. We employ a special notion of *context* here, defined to be a situation in which an common ontological term assumes a specialized meaning, hence, a particular semantic frame of reference. Context in this sense is a primary reason for semantic heterogeneity, and has also been identified as a powerful tool for semantic reconciliation [4, 6, 7, 8]. This notion of context as a particular semantic frame of reference is also particularly efficient, because a single context can be applied to a large set of data at once (e.g. the source-of-origin context applies to all attributes in a source). Information about meaning can be inferred from this context, so the meaning itself may not always have to be explicitly represented as long as it can be inferred or attached from the context. In the following, we describe three types of information which efficiently support the use of context to perform semantic reconciliation.

Source-descriptors. We associate a *source-descriptor* with each source, and with the SSC attributes derived from it. A source-descriptor is a record with nine fields:

1) *Mission.* Label describing data source mission (e.g. logistics, medical records, training, etc.)
2) *Organization.* The organization maintaining the data (e.g. which military service).
3) *Label.* The official name of the data source.
4) *Periodicity.* Frequency of refresh.
5) *Age-avg.* Average time since last update for values in the data source.
6) *Age-stdev.* Standard dev. of time since last update.
7) *Fill.* Percentage of non-null values (a measure of data quality).
8) *Accuracy.* Minimal measurement standards for data in this source.
9) *Coverage.* A list of ontological concepts for which SSCs have been constructed.

Some field domains are shared by fields in the usage-descriptor (below), to permit them to be used together. We make no claims of completeness (e.g. a name and phone number of a human point of contact would also be quite useful); our goal here is to illustrate a means to substantially improve the current reconciliation process at reasonable cost. Note also that many data sources are themselves integrations of data sources (three levels of integration being not uncommon in the DoD), with partitions varying by the *ultimate* source-of-origin. This

problem can be addressed by assigning source-descriptors to important partitions of a source.

Usage descriptors. While source-descriptors provide context information about information sources, usage-descriptors provide context information about the task (or major concern) of a particular user community using the data. A usage-descriptor can be statically associated with a set of RCs and their attributes, or dynamically associated with an individual query over the set of RCs (as in [9]), differing in when the usage information is held to be valid for the purposes of semantic reconciliation. A usage-descriptor is represented as a record with 6 fields:

1) *Mission.* Label describing mission of those using this information (e.g. medical administration, ballistic missile defense, logistics planning). Domain corresponds to the domain of "mission" in the source-descriptor.
2) *Organization.* Organization of users. Corresponds to source-descriptor "organization".
3) *Recency.* Maximum permissible age of information for this usage. Domain corresponds to source-descriptor "periodicity".
4) *Accuracy.* Minimal permissible accuracy. Corresponds to source-descriptor "accuracy".
5) *Interests.* Ontological concepts relevant to the mission and users. Corresponds to "coverage".
6) *Selector.* Function of source-descriptor attributes; used in the "selection" task.

Canonical attributes. A *canonical attribute* (CA) represents a class of similar attributes (comparable to a data type) and contains information about how the meaning of this class of attributes varies with its context. We assume a set of predefined CAs suitable to the DoD, such as "quantity-on-hand", "unit size", "altitude" and "location". During ontology development, a CA can be associated with an RC attribute. A canonical attribute includes the following:

1) *Attribute name.* The canonical name used in RC definitions.
2) *Context list.* A list of semantically important context identifiers (e.g. Army, Joint, Planning, Space, Aviation...) for this attribute, for which variation of interpretation is known to exist.
3) *Context inference mechanism.* A means of assigning a unique context to an RC attribute (from the context list of its CA), or to one of its relevant SSC attributes. Context inference is based on the information in source and usage descriptors.
4) *Description.* Textual description of how this attribute is interpreted in each context in the list.
5) *A library of semantic transformation functions (STFs).* Given context X of an RC attribute associated with this CA, and context Y of a relevant SSC attribute, where X and Y are in the attribute's context list, the $Y->X()$ STF performs the required transformation.

Canonical attributes address an important DoD-specific problem. Recent efforts have attempted to achieve DoD-wide interoperability among classes of similar attributes by the syntactic standardization of all attribute names (thus identically named attributes are expected to interoperate). Unfortunately, a confusing proliferation of new names has resulted to distinguish specific semantic differences. For example, 35 separate standardized attributes have been defined pertaining to "altitude" alone, including: "accident-aircraft-flight-data pressure altitude identifier" and "joint-air-operations-training-event altitude dimension". Note that important context information (e.g. "joint", "training") is buried in attribute labels, limiting its usefulness (and frustrating would-be authors of SQL queries!). A more appropriate approach semantic differences would be to associate existing altitude-relevant attributes with a cannonical attribute for "altitude", and interoperating by reasoning about each attribute's context.

5 Enhanced Semantic Reconciliation

In this section, we describe a (fabricated, but realistic) simple example of semantic reconciliation, illustrating the use of the semantic information described in Section 4 to improve, and to some extent automate, the reconciliation tasks described in Section 3. We assume a human integrator, assisted by an automated "integration agent", and show how each exploits semantic information.

As illustrated in Figure 3, an integration is being performed for air and space tracking communities. Despite tracking different objects (aircraft and satellites), these two communities periodically must interoperate and share data. We focus on the specific problem of reconciling the notion of "altitude". Figure 3 shows an RC attribute "altitude" used by both communities (with a specialized meaning in each), and its relevant SSC attributes (with their own specialized meanings), each from a relevant information source. The user groups are characterized by their usage-descriptors, and the information sources by their source-descriptors. In addition to the descriptors shown, we also assume the existence of canonical attribute "altitude" with two contexts: "space" and "atmospheric" having the previously described semantics, and two STFs: "space->atmospheric()" and "atmospheric->space()".

1) *Ontology Development.* In ontology development, the canonical attribute "altitude" is used in RC definition. Canonical attributes make ontology definition both easier, since a library of canonical concepts can be drawn on, and more precise, since semantic information is explicitly represented.

2) *Relevance Mapping.* Relevant mapping is simplified because both a human integrator and an automated integration agent can infer that Source_1 is inadequate for either user community, and eliminate any SSC attributes derived from it from consideration. The human will notice Source_1's use in training, and immediately recognize that training systems are not good

sources of "live" data. The agent can detect that Source_1 is an aperiodic source which appears to have been loaded with data about 6 months ago, and the probability of finding data meeting the requirements of either user community is extremely low. Thus, only SSC_2 attribute "altitude" and SSC_3 attribute "height" are relevance-mapped to canonical RC attribute "altitude".

Note that the ability use descriptor information to match sources with usages generalizes powerfully beyond the task of relevance mapping: a form of data source brokering [3] can also be performed in this manner. In a global information space with many unfamiliar (but characterized) data sources, a user (who characterizes his or her data usage appropriately) could be automatically matched with a ranked list of data sources based on a correlation of fields in descriptors (e.g. ontological interests correlated with ontological coverage).

3) *SSC Value Transformation.* To utilize the semantic transformation functions supplied with the CA "altitude", we must first infer context. For the human, it is straightforward to assign the "space" context to usage 1 and source 2 (both associated with NASA and dealing with satellites), and similarly the "atmospheric" context to usage 2 and source 3. An agent could make these inferences by automated means. In the simplest case, contexts could be preassigned to usages and information sources by a human expert, and later looked up by the agent. In a more sophisticated approach, production rules, Bayesian techniques [10], or data mining could be used to automatically infer context.

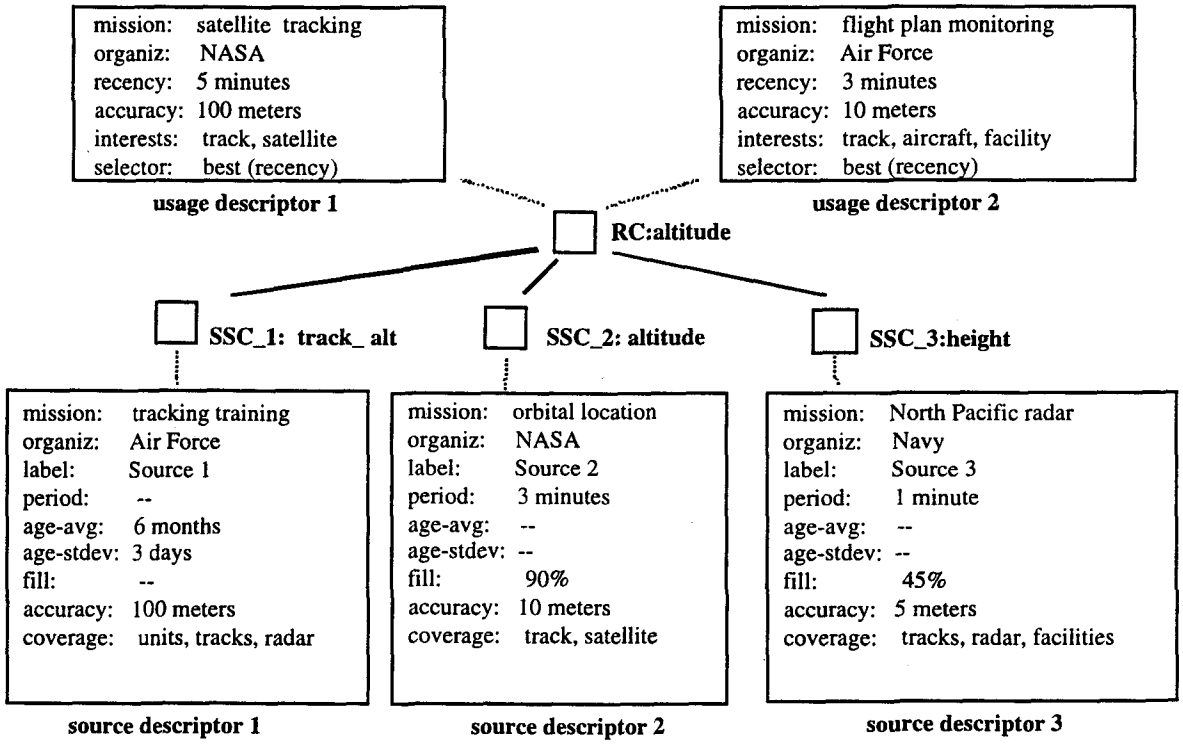

Figure 3. Descriptors for Reconciliation of "Altitude" Attribute

Given a context for RC:altitude, and for SSC_2:altitude and SSC_3:height, an STF can be chosen from the canonical attribute to do the transformation. In our example, two functions are needed: atmospheric->space(), space->atmospheric(). For example, when "RC: altitude" is contextualized as "space" for satellite tracking, the correct value of SSC_3: height is computed by:

atmospheric->space(SSC_3: height)

which changes surface-of-earth coordinates to center-of-earth coordinates. When contexts are identical, as is frequently the case, no STF is used.

4) *RC Value Determination.* Assume Source2 and Source3 both track the same high flying aircraft, and we must determine its altitude for the satellite tracking community. In the (simple) case of selection, value determination consists of generating a function **f()** which returns the value of one of the relevant (and transformed) SSCs based on some selection criteria:

RC:altitude = f(SSC_2:altitude, atmospheric->space(SSC_3:height)).

Human integrators typically use the qualities of each SSCs data source as a selection criterion, and select the value from the most appropriate source. Clearly, source-descriptors greatly enhance the human's ability to make such a judgement by providing more knowledge about data source qualities, especially in the case of unfamiliar data source (increasingly the case in a global environment). The human integrators we interviewed were excited at the prospect of access to such information; measures such as fill, mission, and organization tell them a "deeper story".

Integration agents can generate a selection function **f()** by using the usage-descriptor's *selector* function as the selection criterion. In our example, both user groups specify **best(recency),** which selects the value from SSC_3 in each case.

Therefore, based on the information provided, an integration agent would (automatically) derive the correct "altitude" in the RCs for satellite tracking to be:

RC: altitude = atmospheric->space (SSC_3:height).

When values are determined by some combination of the relevant values (e.g. aggregation), it can be more challenging to derive **f()**. Consider a request for the "total_units" in the joint armed forces. In the Army, a special case of "unit" exists: a unit's equipment can be positioned in advance of its arrival. In the Army only, the equipment receives a separate unit number of its own, distinct from that of the arriving unit. Assume the "joint" context for a canonical attribute "unit", which semantically requires the presence of people. If each military service has an information source with (locally contextualized) attribute "total_units", we cannot simply total these attributes, because the Army's "total_units" is semantically incompatible with the joint "total_units". Note that it is *impossible* to write an STF correctly transforming Army "total units" according to joint semantics given this information alone, since we cannot tell how many Army units in the total contain no people! However, we can write an (SQL-based) STF by accessing information on the *individual* Army units:

```
SELECT   count(unit.id) as army_total_units
FROM     army_units
WHERE    head_count > 0.
```

In general, "missing knowledge" (such as information on individual units) may not be available anywhere in the data source, and may require the collection of additional information than that described in Section 4 for semantic reconciliation of aggregates to take place.

6 Summary And Future Work

Semantic heterogeneity is becoming an increasingly important problem as user communities seek to interoperate in global information spaces. Recent advances in information integration infrastructure have not had a significant impact on the task of semantic reconciliation; an infrastructure for semantic reconciliation must be built as well. In this paper, we have illustrated how specific types of information about data semantics can help automate the reconciliation process, significantly aiding human integrators as well. We have defined three useful types of information: source-descriptors, usage-descriptors and canonical attributes. Source and usage descriptors are succinct descriptions of a source-of-origin and of an intended data usage, respectively. Canonical attributes are an augmented typing mechanism for attributes being reconciled. Together, they can be used to appropriately contextualize attributes, revealing hidden semantic conflicts and making it possible to adjust values accordingly. Each of these types of information efficiently characterizes large classes of attributes. Through an extended example, we have shown how an automated integration agent derived the transformations necessary to perform the four tasks of a simple reconciliation.

We are continuing to incorporate these ideas in an ongoing integration project. In our current application work, we have chosen to compromise against representation in favor of efficiency. Further work, however, is needed in several areas including: more extensive semantic representations for aggregates, parts and wholes, and similar specialized relations; context representation and inference; and more extensive automation of reconciliation. Pertinent to the latter, we are seeking to further develop the basic semantic mappings described in this paper into a more general *semantic algebra* [11, 12].

Acknowledgments. We gratefully acknowledge comments on previous versions of this paper from the editors and several anonymous reviewers. We would also like to thank Lou Mason, Terry Boschert, David Markham, Bonnie Blades, Chuck Yokely, Rex Haddix, and others for discussions concerning semantic interoperability problems they have encountered; any inaccuracies are our own.

References.

[1] http://cjis.telos.com/catalog/eworks/virtualdb.html

[2] http://www.cyberexp.com/products/ceframe-eng.html

[3] Mena, E.; Kashyap, V.; Illarramendi, A.; Sheth, A. 1998. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In *Formal Ontology in Information Systems*, N. Guarino, ed., pp. 269-283. Proc FOIS'98, June 6-8, 1998, Trento, Italy. Amsterdam: IOS Press.

[4] Kashyap, V.; Sheth, A. 1997. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. In M. Papazoglou, and G. Schlageter, (Eds.), Boston: Kluwer Acad. Press, 1997.

[5] Rosenthal, A. and L. Seligman, "Data Integration in the Large: The Challenge of Reuse," Proc. VLDB Conf., Santiago, Chile, September 1994.

[6] Kashyap, V.; Sheth, A. 1996. Schematic and Semantic Similarities between Database Objects: A Context-based Approach. *VLDB Journal* 5 (4), 1996.

[7] Ouksel, Aris M; Naiman, Channah. 1994. Coordinating Context Building in Heterogeneous Information. Journal of Intelligent Info. Systems 3,1,151-183, 1994.

[8] McCarthy, J., "Notes on Formalizing Context", In *Proc.* IJCAI, pages 81-98, 1993.

[9] Ouksel, Aris M. 1999. Ontologies are not the Panacea for Data Integration. To appear in J Parallel and Dist. Sys., 7, 1-29, 1999. http://www.uic.edu/cba/arisol/.

[10] Pearl, J. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241-288.

[11] Meseguer, José. 1998. Formal Interoperability. In Proceedings of Fifth Int'l Symp on AI and Mathematics, Fort Lauderdale, January 4-6, 1998,

[12] Wiederhold, Gio. An Algebra for Ontology Composition. *Proc. 1994 Monterey Workshop on Formal Methods*, Sept 1994, U.S. Naval Postgraduate School, Monterey CA, 56-61.