Semantic Interoperability in the Fixed Income Securities Industry: A Knowledge Representation Architecture for Dynamic Integration of Web-Based Information

Allen Moulton Stuart E. Madnick Michael D. Siegel
MIT Sloan School of Management
amoulton@mit.edu smadnick@mit.edu msiegel@mit.edu

Abstract

We examine a knowledge representation architecture to support context interchange mediation. For autonomous receivers and sources sharing a common subject domain, the mediator's reasoning engine can devise query plans integrating multiple sources and resolving semantic heterogeneity. Receiver applications obtain the data they need in the form they need it without imposing changes on sources. The KR architecture includes: 1) data models for each source and receiver, 2) subject domain ontologies, containing abstract subject matter conceptualizations that would be known to experienced practitioners in the industry, and 3) context models for each source and receiver that explain how each source or receiver data model implements the abstract concepts from a subject domain ontology. Examples drawn from the fixed income securities industry illustrate problems and solutions enabled by the proposed architecture.

1. Introduction

Efficiently integrating new sources of information from outside the enterprise is often critical to success in a world of global competition, interdependency, and rapid market change. Within an organization, data can be created, stored, and used by people and computers sharing a common implicit understanding of data semantics. We use the term *context* to refer to this implicit understanding of the relationship between data elements and structures and the real world that the data represents. The context interchange problem arises when organizations with different contexts must exchange information[8].

A context interchange (COIN) mediator is an automated reasoning engine to assist an organization in resolving semantic conflicts between its own receiver context and the contexts of data sources[6]. Because context definitions are declarative, they need only be prepared once for each source and receiver context[2]. Data sources may be relational databases, XML documents, HTML wrapped to appear as relations with limited query capability[5], and stateless computational procedures. Using declarative context knowledge, a COIN mediator identifies semantic conflicts and designs plans for combining sources with data conversions to meet receiver semantic requirements.

Our work is based on the semantic proposition that interchange of information can be mediated if sources and receivers share a common subject domain (or interlocking subject domains). Sources and receivers are seen as autonomous implementations of common subject domain abstractions (or interlocking abstractions). Source and receiver system designers make decisions about how to conceptualize abstract constructs and about how to represent that conceptualization in data and programs. The COIN mediator has the task of applying declarative information about the context of each source and receiver to device plans for integrating sources to meet receiver requirements.

Given a large number of component systems operating in a diversified and dynamic environment, COIN mediation facilitates: rapid incorporation of new information sources, dynamic substitution of information sources, extension and evolution of semantics, data representation in the user's context, access to the meaning of data represented, identification and selection of information source alternatives, and adaptation to changes in user and business operations.

Building on earlier work by Goh[6], we are exploring knowledge representation and reasoning methods to expand the functionality of COIN mediation to include: 1) identifying data representation conflicts and introducing conversions to transform data from source to receiver form, 2) applying subject domain and context knowledge to map between receiver schema and source schemata, 3) determining when and how to combine sources, feeding data from one source to another with appropriate data conversions, 4) deriving missing data by applying domain ontology, context knowledge, or by combining sources. In addition to databases and web-based data sources, we also include computational procedures as transformational sources. Where possible we employ a knowledge representation consistent with common system design practices (e.g., UML, E-R, and repositories).

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

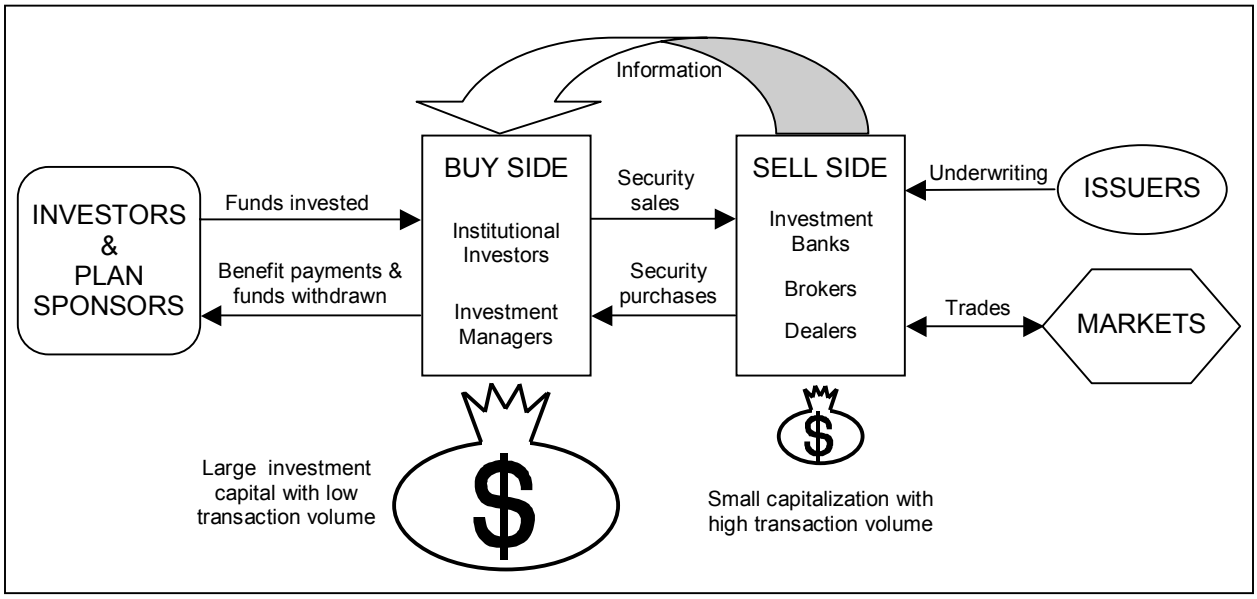

Figure 1. The fixed income securities industry

2. Fixed income securities industry

Information is the critical resource in the fixed income securities industry – information about securities and their issuers, information about markets, information about economic conditions and events, and information about methodologies and models (see Figure 1). Billions of dollars of debt instruments trade every week. Firms on the "buy side" (institutional investors and investment managers) manage capital on behalf of investors and benefit plan sponsors. Firms on the "sell side" (investment banks, brokers, and dealers, often known as "Wall Street") bring new securities to market and interact to create capital markets. While the popular connotation of sell-side firms is of great wealth, the capitalization of these firms is actually relatively low. The big money is on the buy-side. The sell-side makes money from a small commission or price spread on large volumes of transactions. The sell-side offers buy-side managers access to its skill, expertise, and knowledge in anticipation of purchase and sale orders.

Bonds and other fixed income securities are obligations to pay sums of money at points in time over the life of the security. Unlike equity securities, an investor has no stake in the financial entity that issued the security. To an investor, a fixed income security represents a stream of future cash flows. A purchase decision trades present money for future payments. There may be optional events that change the cash flow stream and there may be risk of default. In essence, however, all fixed income securities are interchangeable commodities from the point of view of an investor. The cash flows from one or more obligations may even be repackaged by selling off rights to payments or by combining rights to payments into new composite securities. This repackaging, or "financial engineering" can produce securities known as "derivatives." Faced with a vast array of combinations of cash flows, risks, and optional events, every industry participant needs timely information and effective methods for determining investment value from raw data.

2.1 Portfolio manager requirements

Fixed income portfolio managers may need to draw on external sources for data about security characteristics, for market valuation information, and for models and calculations[11]. All these sources may need to be combined with internal portfolio holdings data and accessed through a decision support application system (see Figure 2). One task involves obtaining current dealer offerings and presenting the portfolio manager, in a consistent manner, information about securities offered and dealer prices. Table 1 below shows a partial schema and semantics requirements for an offerings analysis application for a portfolio manager.

Receiver relation R (application requirements)		
attribute	**sample data**	**semantic notes**
secidn	191219AN4	CUSIP security identifying number
matdat	02/01/2012	maturity date, mm/dd/yyyy
cprate	8.500	interest rate, percent, decimal fraction
price	116.08	dollar price, percent, fraction in 32nds

Table 1. Receiver R data semantics

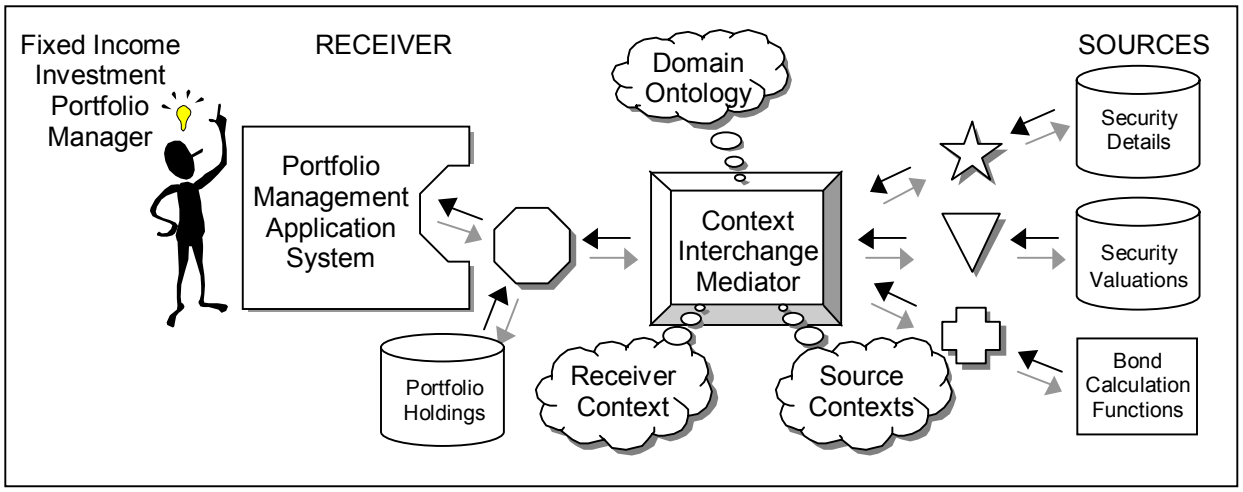

Figure 2. Fixed income securities investment mediation scenario

The first three attributes provide information about the security offered (standard CUSIP identifier, maturity date, and coupon interest rate). The last attribute is the price asked by the dealer, expressed as a percentage of face value with fractions in 32nds. A fragment of a query by the application might be:

```
SELECT secidn, matdat, cprate, price
  FROM R
  WHERE <criteria>
```

To explore context interchange problems, we consider two alternative sources for offerings. Dealer A provides a web page that has been wrapped for a relational query interface. Dealer B provides an XML document. Each dealer makes its own decisions about what information to present and the semantics of that information.

2.2 Data representation semantics

Table 2 below shows a section of the schema for Dealer A offerings.

Source relation A (dealer A offerings web page)		
attribute	**sample data**	**semantic notes**
cusip	191219AN4	CUSIP security identifying number
maturity	40940	maturity date, Lotus/Excel 1900 date
coupon	0.08500	interest rate, factor, decimal fraction
price	116.25	dollar price, percent, decimal fraction

Table 2. Dealer A data semantics

The first step in developing a mediation plan is to note that each row in A matches the requirement for a row in R – each represents a dealer offering for a security. Next, by examining the semantic notes, it is seen that, although three attribute names are different, each attribute in R can be obtained from one attribute in A.

The final step is to resolve data representation differences. R.secidn and A.cusip are the same. R.matdat requires a date in "mm/dd/yyyy" format; A.maturity is provided as a Lotus date sequence number. R.cprate is in percent; A.coupon is in factor form commonly used in spreadsheets. The price attributes, though named the same, are subtly different in semantics. R.price requires a percent with fraction in 32nds; A.price is expressed as a percent with decimal fraction. Failure to convert from decimal to 32nds could result in a substantial error in the price. Having identified the semantic conflicts, the mediator inserts appropriate data conversions: multiplying by 100 to convert factor to percent, the Excel "dollarfr" function to convert a decimal fraction into 32nds, and a wrapped date conversion function, source F (Table 3):

Source F (wrapped date conversion function)		
attribute	**sample data**	**semantic notes**
out	02/01/2012	reformatted date output
outformat	"mm/dd/yyyy"	format for output date
in	40940	date input
informat	"Lotus"	format for input date

Table 3. Date conversion data semantics

The query rewritten in terms of the source schema with necessary data representation conversions would be:

```
SELECT A.cusip as secidn, F.out as matdat,
   A.coupon*100 as cprate, dollarfr(A.price,32) as price
  FROM A, F
  WHERE <criteria>
    AND F.outformat = "mm/dd/yyyy"
    AND F.in = A.maturity AND F.informat = "Lotus"
```

2.2 Derived data and multiple source integration

The use of XML simplifies the access to data in many respects, but still leaves a wide range of semantic issues to be resolved. Adoption of standards can reduce the degree of semantic heterogeneity. Nevertheless, in the

securities industry and many others, innovation will proceed faster than standards. Consider Dealer B offerings provided as an XML document such as:

```
<OFFERSHEET>
  <OFFER>
    <BOND> 191219AN4 </BOND>
    <PRICE> 103.28 </PRICE>
  </OFFER>      ...
</OFFERSHEET >
```

Dealer B offerings have a tabular structure that can be represented as a relation as shown in Table 4. Comparing the semantics of B to requirements R, we note that two of the attributes of the security are missing. Furthermore, the price is expressed as a "nominal spread" in "basis points" instead of a "dollar price" in percent. To meet the receiver's requirements, general industry knowledge and additional data sources must be brought to bear, along with conversion of units and scaling.

Source relation B (dealer B offers XML document)		
attribute	**sample data**	**semantic notes**
BOND	191219AN4	CUSIP security identifying number
PRICE	103.28	nominal spread, basis points

Table 4. Dealer B data semantics

To resolve the semantic conflicts, the mediator must know all of the following:

- source C (Table 5) can provide security details,
- nominal spread means the difference between yield on a security and a benchmark yield,
- the 10-year T-note yield is an appropriate benchmark,
- which can be obtained from source D (Table 6),
- bond calculation source E (Table 7) can convert yield to price given the security's interest rate and other details and the date of settlement,
- rules for converting data codes,
- basis points are 1/100th of a percent, and
- methods for converting data representations as discussed in section 2.2 above.

Source relation C (security characteristics web site)		
attribute	**sample data**	**semantic notes**
coupon	8.500	interest rate, percent, decimal fraction
maturity	02-01-2012	maturity date, MM-DD-YYYY
cusip	191219AN4	CUSIP security identifying number
datedDate	02-11-1992	issue date, MM-DD-YYYY
firstCoup	08-01-1992	first payment date, MM-DD-YYYY
market	US Corporate	market/type of security, text
payFreq	Semi-Annual	interest payment interval

Table 5. Source C data semantics

After analyzing the semantic differences between the receiver and source B, the mediator identifies additional the data sources C (Table 5) and D (Table 6).

Source relation D (Treasury yield curve web site)		
attribute	**sample data**	**semantic notes**
10yr	5.091	yield on current 10 year T-note

Table 6. Source D data semantics

The mediator would also make use of the necessary yield-price calculation procedure, wrapped as if it were a data source E as described in Table 7.

Source E (Excel analytic toolkit function PRICE)		
attribute	**sample data**	**semantic notes**
price	117.875	flat price, percent, decimal fraction
settlement	37196	settlement date, Excel 1900 date
maturity	40940	maturity date, Excel 1900 date
rate	0.0850	interest rate, factor, decimal fraction
yld	0.061238	yield, factor, decimal fraction
redemption	100	redemption value, percent
frequency	2	coupon frequency per year (1,2,4)
basis	0	day count basis, code (0,1,2,3,4)

Table 7. Source E data semantics

The mediator must also insert data conversions for dates and percentages as described above. Data codes for payment frequency from source C must be mapped to E's context and the day count basis inferred from market conventions. Combining these sources, data conversions, mappings, and inferences, the resultant mediated query would look like:

```
SELECT B.cusip as secidn, v.out as matdat,
   C.coupon as cprate, dollarfr(E.price,32) as price
 FROM B, C, D, E, F v, F w, F x
   Cfreq, Cmarket, Efreq, Ebasis, Mmarket
 WHERE <criteria>
   AND v.outfmt = "mm/dd/yyyy"
   AND v.in = C.maturity AND v.infmt = "mm-dd-yyyy"
   AND C.cusip = B.BOND
   AND E.settlement = x.out
   AND x.outfmt = "Lotus"
   AND x.in = "11/01/2001" AND x.infmt = "mm/dd/yyyy"
   AND w.outfmt = "Lotus"
   AND w.in = C.maturity AND w.infmt = "mm-dd-yyyy"
   AND E.rate = C.coupon/100
   AND E.yld = ( B.PRICE/100 + D.10yr ) / 100
   AND E.redemption = 100
   AND E.frequency = Efreq.xcode
   AND Cfreq.freq = Efreq.freq
   AND C.payFreq = Cfreq.xcode
   AND E.basis = Ebasis.xcode
   AND Mmarket.daycount = Ebasis.daycount
   AND Mmarket.mcode = Cmarket.market
   AND C.market = Cmarket.xcode
```

Without mediation, the portfolio manager would see a price of 116.25 from Dealer A and 103.28 from Dealer B. With mediation, Dealer B's quote is converted to a dollar price of 117 28/32 and the comparison is reversed.

3. Knowledge representation architecture

As illustrated in Figure 3, our architecture divides the knowledge used for mediation into three layers. At the top of Figure 3, a domain ontology captures abstract subject domain concepts used by experienced practitioners and system designers in the industry. At the bottom, data models for each source and receiver include the sort of information that programmers would use to access data. In the middle are context definitions for each source and receiver that explain how each source or receiver data model implements the abstract concepts from a subject domain ontology.

The framework of a subject domain ontology is a structural conceptual model with classes of abstract objects, attributes of objects, and relationships. Semantic types capture alternative data representations as in [6]. Enumerated conceptual categories model object property distinctions that may be implemented differently by each source and receiver. Rules capture functional relationships among conceptual model attributes that would be known from general domain knowledge. Default and contingent rules allow for deriving attributes based on partial information following the reasoning that industry participants would use.

Data models for the receiver and for relational sources import schema and catalog information. For XML, an XML schema or DTD can be used or the schema inferred from documents themselves. For HTML sources, the data model is provided by the web wrapper. For computational procedural sources, arguments and return values are treated as relational attributes in a data model that is augmented with functional dependency and input-output combination constraints.

Context models for each source and receiver explain how each data model implements the general concepts in the domain ontology. Classes from the domain ontology conceptual model can be used directly or augmented with context-specific extensions. Context-specific functional or

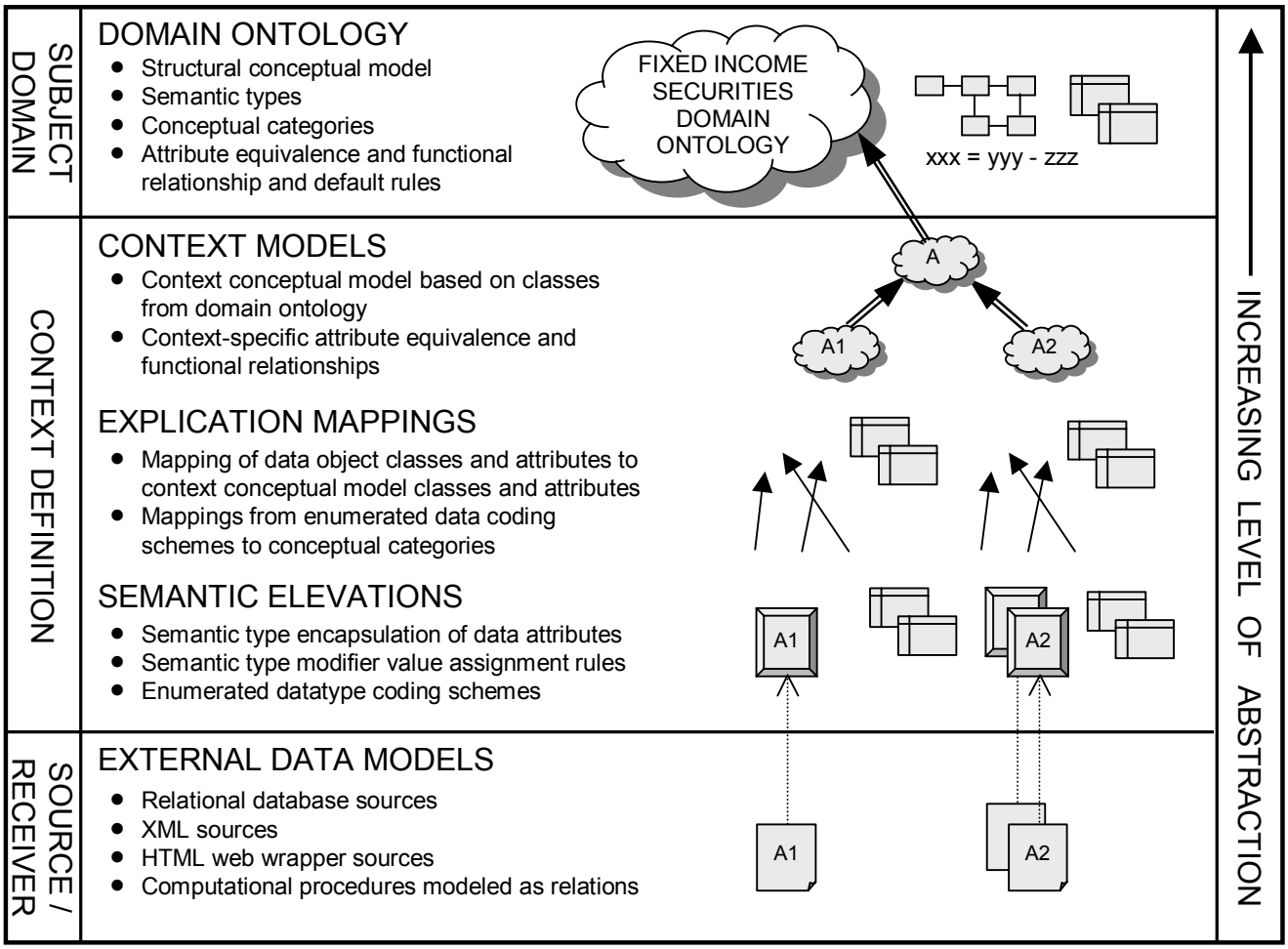

Figure 3. Knowledge Representation Architecture

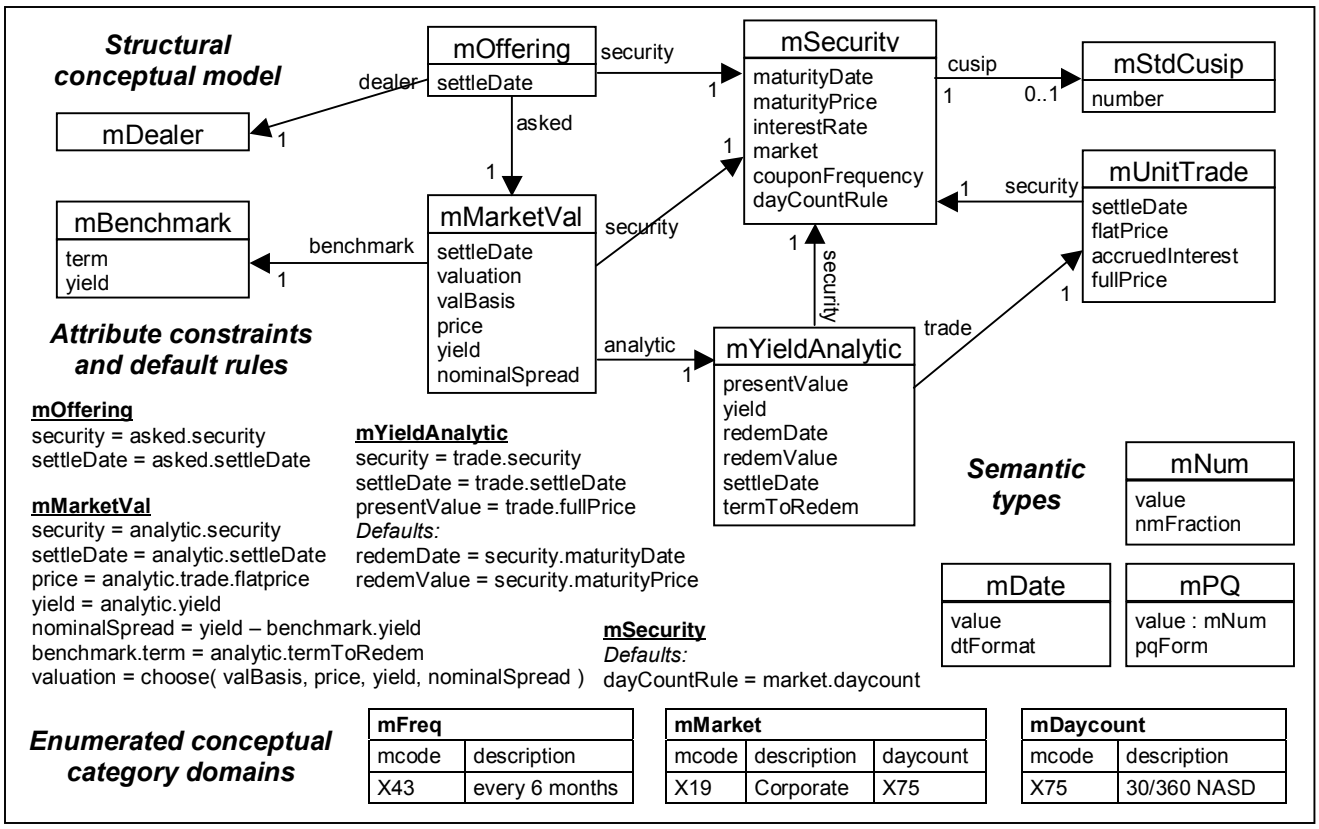

Figure 4. Fixed Income Subject Domain Ontology Fragment

equivalence relationships tie elements of the conceptual model to elements of the data model. For coding schemes, enumerated attribute domains are mapped to conceptual categories from the domain ontology. Semantic types can be used to logically encapsulate data attributes and associate context-specific modifier values to identify the data representation used.

3.1 An example fixed income domain ontology

A domain ontology is not a global schema. Rather, it is an abstract representation of the subject matter that each source and receiver data model implements in its own way. Neither sources nor receivers need to accept the domain ontology as the "right way" of representing information about the subject matter at hand, avoiding some of the practical user acceptance problems noted in [10]. By allowing each context model to extend the domain ontology and to explain how context-specific concepts map to general domain ontology concepts, mediation is facilitated without imposing the rigidity seen in view-based systems.

Figure 4 illustrates some elements of a subject domain ontology for fixed income securities. At the top is a *structural conceptual model* with classes for concepts used in the example discussed in section 2 above. The diagram uses a UML notation. For example, the *mOffering* entity class represents an offering of a *security* by a *dealer* for purchase at the *asked* price for settlement on *settleDate*. The security and dealer are also represented by entity classes (*mSecurity* and *mDealer*), associated with the offering by many-to-one relationships. Properties of the security are represented by attributes of the *mSecurity* class. The U.S. standard CUSIP identifying number is captured as an attribute of the *mStdCusip* class associated with the security.

The remaining classes represent conceptual objects that may be used in security valuation. In the example, the dealer specifies an asked price at which the security is offered for sale. This "price" may be given as a flat price (percent of principal face value), a yield (internal rate of return to redemption), or a nominal spread (difference in yield between the offered security and a benchmark security). Given sufficient information, all these forms of market valuation are mutually interchangeable, with values derivable by methods well known in the industry. The *mMarketVal* class represents this concept of a security valuation that may be specified in any of the above three ways. The *mYieldAnalytic* class represents the internal rate of return analytic method. The *mUnitTrade* class represents a trade in a unit of face value of a security and captures the industry convention for adding accrued

interest to the price paid for the principal (*flatprice*) to obtain the total investment cost (*fullprice*) used in yield calculations. The *mBenchmark* class illustrates the notion of a benchmark security or index for the nominal spread form of valuation.

Attribute constraints provide additional information about the relationship among the attributes of a given class and across classes. For example, *mOffering* requires that the *asked mMarketVal* object refer to the same security and use the same settlement date. This is specified by universal constraints attached to the *mOffering* class:

security = asked.security
settleDate = asked.settleDate

In writing these rules, a period indicates traversal from one entity to another via its role name in the originating class. Constraints may specify either the equivalence of two attributes (as above), or an arithmetic formula that relates the values of attributes, e.g.:

nominalSpread = yield – benchmark.yield

Default rules are similar to constraints, but are not universal in applicability. Instead, they may be used by the mediator to derive a value for an attribute when no other constraint or source specifies the value.

Semantic types are parameterized polymorphic classes used to capture alternative data value representations as described in [6]. Figure 4 shows three example semantic types. The *mDate* semantic type encapsulates a data value representing a data and associates modifier *dtFormat* to specify which format (e.g. "Lotus" or "mm-dd-yyyy") is used by a source or receiver. The *mNum* semantic type has modifier *nmFraction* to identify the fractional notation used (e.g., "decimal" or "32nds"). For percentage quantities, *mPQ* encapsulates a value that is itself of semantic type *mNum* and adds a modifier *pqForm* to specify which for of percentage is used (e.g., fraction, percent, basis points).

The final part of a subject domain ontology are *enumerated conceptual category domains*[12] used to represent conceptual distinctions often implemented as enumerated datatype coding schemes. The mediator uses conceptual categories to determine whether an arbitrary code used in one context can be converted to another code in a different context. Three conceptual category domains are illustrated in Figure 4. In each, space permits inclusion of only one conceptual category out of a longer list. The *mcode* column contains a meta-code used to stand for the instance of the concept in the ontology and context definitions. These codes never appear in data. The *description* column is a human-readable description of the concept. Two conceptual categories shown represent properties of a security: *mFreq* payment frequencies; *mDayCount* interest accrual conventions. The *mMarket* conceptual category represents a common classification of securities by trading market that may be used to derive a default value for a security *dayCountRule* (using the default rule associated with the *mSecurity* class). A more complete discussion of our use of enumerated conceptual category domains can be found in [12].

The classes in the subject domain ontology look much like entities that might be used in a database design, but are intended to remain entirely abstract and to be used to reason about how each context implements of those abstractions. Context definitions provide the information that the mediator needs to compare contexts and design a plan for meeting receiver requirements.

3.2 Defining context for receiver R

Defining context using our architecture is analogous to the process that a programmer would follow to design a program to extract data from sources. The first step is to model each source or receiver relation using conceptual objects from the domain ontology. In the example in section 2, rows of R are modeled by an *rOffering* class that derives from *mOffering*.

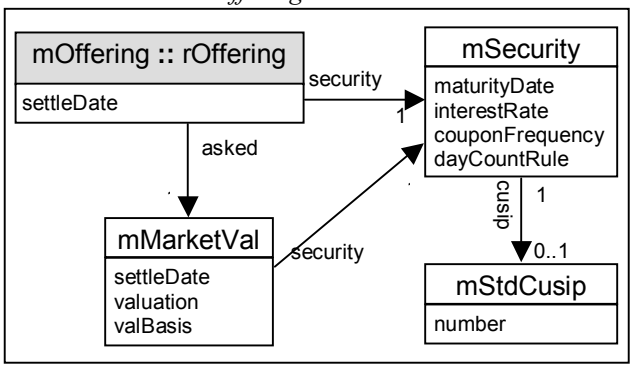

Figure 5. Context model for receiver R

Note that all the unshaded attributes and related classes shown in Figure 5 are inherited from the ontology, along with all the constraints and default rules that attach to those classes. The next step is defining explication mappings that model the role of each data attribute in R by associating it with a conceptual attribute drawn in the context model as in Table 8.

attribute in R	path from rOffering concept
secidn	security.cusip.number
matdat	security.maturityDate
cprate	security.interestRate
price	asked.valuation

Table 8. Receiver R explication mappings

Table 1 describes the first attribute, *secidn*, to be a CUSIP security identifier, which is located in the context model by traversing from the *rOffering* object to its *security* and thence to its *cusip number*. The next two attributes, *matdat* and *cprate*, are located within the

security object. The final attribute, *price*, is associated with the *valuation* attribute located within the *asked mMarketVal* object. An additional constraint must tbe added to the context definition for R to specify that the security valuation in R is in the form of a flat price:

asked.valBasis = 'flatprice'

The final step in defining the context of receiver R is to introduce semantic elevations [6]. Each data attribute is assigned a semantic type and the modifiers for that type are specified (see Table 9).

attribute in R	semantic type	modifiers
secidn	mText	
matdat	mDate	dtFormat = 'mm/dd/yyyy'
cprate	mPQ	pqForm = 'percent' nmFraction = 'decimal'
price	mPQ	pqForm = 'percent' nmFraction = '32nds'

Table 9. Receiver R semantic elevations

3.3 Defining context for source A

Defining context for source A follows the same steps as described for receiver R. Because each row in source A is an offering, the context conceptual model looks quite similar to Figure 5, except that the shaded class defined will be called *aOffering*. All the other elements of the context model are inherited from the ontology and remain unchanged. From the data semantics shown in Table 2, the explication mappings for the attributes in source A would be defined as in Table 10 and semantic elevations as in Table 11

attribute in A	path from aOffering concept
cusip	security.cusip.number
maturity	security.maturityDate
coupon	security.interestRate
price	asked.valuation
asked.valBasis = 'flatprice'	

Table 10. Source A explication mappings

attribute in A	semantic type	modifiers
cusip	mText	
maturity	mDate	dtFormat = 'Lotus'
coupon	mPQ	pqForm = 'factor' nmFraction = 'decimal'
price	mPQ	pqForm = 'percent' nmFraction = 'decimal'

Table 11. Source A semantic elevations

3.4 Sketch of mediation to obtain R from A

Once the context definitions are prepared, it is readily apparent that source A is conceptually compatible with the requirements of receiver R. Rows in both R and A are modeled by context model concepts inheriting from the same *mOffering* concept in the ontology. The differences in attribute names are resolvable using the explication mappings. The valuation in each case is a flat price. The other differences between the contexts are captured by the modifiers on the semantic types. The mediator then follows a logic similar to [6] to introduce conversions between source A's data representations and receiver R's data requirements. For data format conversion, the mediator would draw on the procedural component of source F, which would also have been explained with its own context definition using similar methods.

3.4 Defining context for other sources

Source B also provides offerings and would be modeled with a similar *bOffering* concept derived from *mOffering* in the ontology. Based on Table 4, explication mapping would look like Table 12.

attribute in B	path from aOffering concept
BOND	security.cusip.number
PRICE	asked.valuation
asked.valBasis = 'nominal spread'	

Table 12. Source B explication mappings

Again referring to Table 4, source B semantic elevations would be as in Table 13.

attribute in B	semantic type	modifiers
BOND	mText	
PRICE	mPQ	pqForm = 'basis points' nmFraction = 'decimal'

Table 13. Source B semantic elevations

Since B does not provide all the attributes needed by R and since the price is given as a nominal spread, the mediator would need additional sources and context definitions for those sources.

Source C would be modeled with a concept derived from the *mSecurity* ontology class. The one-to-one relationship between an *mStdCusip* and *mSecurity* allows the mediator to associate the *mSecurity* implicitly referred to in B, allowing the mediator to draw on the data available from source C .

Attributes *C.payFreq* in Table 5 and *E.frequency* in Table 7 represent the same concept using different context-specific symbols. In defining a context, tables of symbols or codes are gathered from documentation or

usage and then associated with conceptual categories from the ontology, e.g.

Cfreq	
xcode	freq
Semi-Annual	X43

Efreq	
xcode	freq
2	X43

Here the code 'Semi-Annual' in C is associated with the meta-code *X43* from the ontology conceptual category domain *Mfreq*. The code '2' in E would be similarly modeled. The mediator can then convert a C code to an E code when the sources need to be joined.

Deriving the *basis* attribute in E requires an additional step. In context E and *Ebasis* code of '0' refers to the '30/360 NASD' interest accrual convention, which is represented by the meta-code *X75* in the ontology conceptual category domain *Mdaycount*. Source C provides a code for the market of the security, but no day count basis.

Cmarket	
xcode	market
US Corporate	X19

Ebasis	
xcode	daycount
0	X75

In order to combine sources C and E, the mediator must draw on knowledge of the industry convention that U.S. corporate bonds use a day count of 30/360 NASD. We capture this knowledge by linking the conceptual category domain *Mmarket* to *Mdaycount*. The mediator uses this general knowledge to infer the day count basis.

We have elaborated the knowledge representation and mediation for enumerated datatypes is at greater length in [12].

4. Conclusion

Context interchange mediation brings automated methods to the important task of assuring that data exchanged across organizations can meet the semantic requirements of the receiver – and do so without obligating source organizations to change their way of doing business or to know about or accommodate the needs of the receiver.

We are continuing to examine techniques for specifying contexts and subject domain ontologies using well established methodologies of business systems analysis and database design [*viz.* 3, 4]. The evolving ISO TC68/SC4/WG10 [13], securities industry standards may partially solve the interoperability problem, and should also provide detailed substantive information models for building ontologies and context models for remaiing information interchange requirements [10,11].

Our work complements research such as Bernstein's on generic model management [1] on one side and Mong-Li Lee's work on using conceptual modeling methods to rigorously design data published on the web[9].

Ongoing research includes the specification of logic rules and reasoning algorithm to traverse the analytic process from receiver data model requirements, through receiver context models and subject domain ontologies, thence to potential source context models and data models, devising plans for meeting the receiver's needs from available source data combined with generated conversion relations. Where defaults come into play, we are drawing on Grosof's work on "courteous logic"[7].

Future work will include: contextual conflicts among interrelated data items within a source and across multiple sources, extended integration of semi-structured, unstructured and image data sources, domain ontology and context model development and evolution methodologies and tools, interlocking subject domain integration, and automatic source selection during query execution.

References

[1] P. A. Bernstein. "Generic Model Management: A Database Infrastructure for Schema Manipulation," CoopIS 2001: 1-6.

[2] S. Bressan, C. H. Goh, N. Levina, S. E. Madnick, A. Shah and M. D. Siegel. "Context Knowledge Representation and Reasoning in the Context Interchange System," Applied Intelligence (13:2), Sept. 2000, pp. 165-179.

[3] Stephen Cranefield and Martin K. Purvis. "UML as an Ontology Modelling Language," Proc. Workshop on Intelligent Information Integration, IJCAI 1999.

[4] Ramez Elmasri, Shamkant B. Navathe. Fundamentals of Database Systems, 3rd Edition. Addison-Wesley, 2000.

[5] A. Firat, S. E. Madnick, and M. D. Siegel. "The Caméléon Web Wrapper Engine," Proc. VLDB 2000 Workshop on Technologies for E-Services, Sept., 2000.

[6] C. H. Goh S. Bressan., S. E. Madnick and M. D. Siegel "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information," ACM Trans. on Office Information Systems, July 1999, pp 270-293.

[7] B. N. Grosof, Y. Labrou, and H. Y. Chan. "A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML,". Proc. 1st ACM Conf. Electronic Commerce (EC-99), 1999.

[8] S. E. Madnick. "Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Heterogeneity," Proc. IEEE Meta-Data Conf., April 1999.

[9] Mong-Li Lee, Sin Yeung Lee, Tok Wang Ling, Gillian Dobbie, Leonid A. Kalinichenko. "Designing Semistructured Databases: A Conceptual Approach," DEXA 2001, pp. 12-21.

[10] A. Moulton, S. Bressan, S. E. Madnick and M. D. Siegel. "An Active Conceptual Model for Fixed Income Securities Analysis for Multiple Financial Institutions," Proc. ER 1998.

[11] A. Moulton, S. E. Madnick and M. D. Siegel. "Context Mediation on Wall Street," Proc. CoopIS 1998, pp. 271-279.

[12] A. Moulton, S. E. Madnick and M. D. Siegel. "Semantic Interoperability in the Securities Industry: Context Interchange Mediation of Semantic Differences in Enumerated Data Types," Proc. DEXA WEBH 2002.

[13] SWIFT. "ISO Working Group 10," http://www.swift.com/index.cfm?item_id=6610

v20a