Context Mediation Approach to Improved Interoperability amongst Disparate Financial Information Services

Sajindra Kolitha Bandara Jayasena

Working Paper CISL# 2004-10

June 2004

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E53-320
Massachusetts Institute of Technology
Cambridge, MA 02142

Table of Contents

Context Mediation Approach to Improved Interoperability amongst Disparate Financial Information Services

by

Sajindra Kolitha Bandara Jayasena

Submitted to the SMA office on 30th June, 2004
In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science

Abstract

There is no entity such as a '*World Wide Bank*' managing the central database of all possible financial activities. Such a concept makes neither technical nor business sense. Each player in the financial industry, each bank, stock exchange, government agency, or insurance company operates its own financial information system or systems. The systems communicate via intranet, proprietary extranets or even the internet.

By its very nature, financial information, like the money that it represents, changes hands. Therefore the interoperation of financial information systems is the cornerstone of the financial services they support. Furthermore financial information is complex. Naturally these characteristics led to the development of standards for the management and interchange of financial information. Yet connectivity and standards alone are not the panacea: different groups of players use different standards or versions of a standard's implementation.

I believe that the solution lies in self-documented languages like XML, semantically rich services and meta-data as promised by the semantic Web, and a mediation architecture for the documentation, identification, and resolution of semantic conflicts arising from the interoperation of heterogeneous financial services.

As the first contribution, in this report I present a case study illustrating the nature of the problem and the viability of the solution we propose. The case is *Electronic Bill Presentment and Payment* Industry. I would describe and analyze the integration of services using four different formats, the IFX, OFX and SWIFT standards, and a proprietary format. To accomplish this integration we use the COntext INterchange (COIN) framework. The COIN architecture leverages a model of sources and receivers' contexts in reference to a rich domain model or ontology for the description and resolution of semantic heterogeneity. The focus has been on how COIN would facilitate such mediations in interoperability between IFX standard and Internal proprietary system, OFX and standards and a Internal proprietary system, SWIFT and a Internal proprietary system and direct mediation between OFX and SWIFT covering semantic heterogeneities spanning ontological, contextual and temporal conflicts.

Further, the ability to use such a complex system requires a certain level of skill and knowledge on underlying implementation mechanism resulting in narrowing down the user segment for COIN. I believe that in order to provide these services, a certain level of abstraction needs to be provided to the user with a less skill and computer science literacy that would facilitate him to model and create the required applications and services for mediating an actual scenario. The best approach is to provide with an interface which has a smaller learning curve and familiarity to an already existing Human Computer Interaction mechanism like windows, icons, pointers and graphical drawing capabilities. As the second contribution in this research I provide a detailed implementation of such a graphical modeling and designing tool that acts as the Metadata Management system in COIN, which would facilitate in graphically creating, modeling and defining the entire domain under analysis and the relevant mediation logic rather than modeling it using in a language like prolog.

Keywords: Semantic mediation, Financial Standards, COIN, Metadata Management, Semantic Web, Context Interchange, Ontology modeling, prolog, Human computer Interface, software design patterns

Dissertation Supervisors:

Professor Stuart Madnick, SMA Fellow,
John Norris Maguire Professor of Information Technology and Leaders for Manufacturing Professor of Management Science Sloan School of Management / Massachusetts Institute of Technology, Cambridge , USA

Associate professor Tan, Kian Lee, SMA Fellow,
School of Computing,
National University of Singapore,
Singapore.

1. Introduction

Effective and transparent interoperability is vital for the profitability and sustainability of the financial Industry. Adhering to a standard hasn't paid rich dividends due to the fact that multiple institutions and geographical segments utilize different standards resulting in sub-optimized catering to different standards as well as problems in interoperability. This is made worse since different entities interpret the same semantics in different ways. As a trivial example, the now obsolete MT100 message in SWIFT [11] was interpreted differently by Financial Institutions. The confusion was on the charging option field, where you can only indicate BEN (borne by creditor) or OUR (borne by debtor). If no option is identified, the User Handbook says that BEN should be understood. But BEN is processed differently by receivers: sometimes all charges are paid by the beneficiary; sometimes charges are shared, amounting to confusions and even losses. Therefore it is vital to facilitate a transparent and effective mediation/interconnection as well as interpretation solution.

As a more complex example, consider this scenario: A Financial Institution (FI) that is involved in *Electronic Bill Presentment and Payment* Industry operating in a European Union Country is faced with multitude of standards like IFX (Interactive Financial Exchange protocol)[10], OFX (Open Financial Exchange Protocol)[9] and the world wide inter-bank messaging protocol, SWIFT [11]. Making matters worse, the FI may have its own semantics for its internal systems that represent the same business domain in a different context.

The *Price* and *Invoice* concepts, in laymen terms would mean *"the amount of money, paid in return for a product of service"*, may be represented in different ways, e.g., excluding tax, with tax and fees, and even with inter-bank charges, resulting in a ontological heterogeneity due to definitional conflict in different contexts [1]. Interoperability of such definitional conflicts is vital in distinguishing intra-bank and inter-bank payment across borders. Further, different contextual heterogeneities exist on the *currency*, where in certain contexts like IFX and OFX; it is implicitly based on where the funds are directed to. As a result of different *Account types* and BANK/*BRANCH code,* financial institution would need to maintain complex mappings between different contexts. In addition, there can be data level heterogeneities like date formats and representations. Some of the conflicts are summarized in table 1.

One of the main objectives of this research is to analyze how COIN mediation technology [2, 3, and 8] could be applied to provide a declarative, transparent yet effective mediation solution to the various heterogeneities that exist in these financial standards.

Table 1.1: Some conflicts in different standards

Property	Internal Schema	OFX	IFX	SWIFT 103/103+
Price	1000 FFR (French Franc)	1000 USD + 1000 * 5%	1000 USD + 1000 * 5%	1000 USD + 1000* 5% + 10 USD (inter-bank charge if outside EU)
Currency	FFR (before 2003 – FFR and After 2003 EURO)	Currency of country of incorporation of payee bank i.e. USD	Currency of country of incorporation of payee bank i.e. USD	Specified in message – can be the payee or payer's currency
Account types	CHK,SVG, MNYMRT	CHECKING, SAVINGS	DDA,SDA	N/A
Bank and branch code	Internal ID	Dependent on the country i.e. clearing #,sort #	Dependent on the country i.e. clearing #,sort #	BIC / BEI (branch ID + bank Id)
Invoice	Net	Net + fees + tax	Net + fees + tax	Included in *Amount* – N/A
Due date	*23022002*	*20020223*	*2002-03-23*	*030223*

In this research I analyze how COIN can be utilized in modeling such complex mediations. I have focused on the Electronic Bill Presentment and Payment Industry as the representative domain. Analyses have been done on a context focusing how financial institutions would cope with such issues. The analysis is broken down mainly to 4 parts. First I will look at how COIN would mediate between Inter-bank payment standard like IFX [10] and an internal system of a financial systems. Here I would be looking at various semantic heterogeneities that might arise ranging from simple data representation heterogeneities into contextual ontological as well as temporal heterogeneities. Next analysis would be looking at how OFX [9] and an Internal System would interoperate and how COIN can be used to mediate those real-life scenarios. Moving away from these two intra-bank standards, I have analysed the interoperability between an internal system of a financial system and SWIFT [11] which have different conflicts compared to the previous two standards. The next analysis focuses on how COIN would mediate some heterogeneity that would arise between intra-bank standards like OFX and SWIFT. A Final analysis has been done on the temporal issues that would be faced by institutions and how COIN framework has been extended to facilitate this. Further I would be addressing the features that need to be extended in order to support more complex mediations.

Having being able to model such complexities and heterogeneities, it requires a substantial effort from the modeler in coming up with the domain ontology, Contexts, various data sources, modifiers and Conversion functions. Representing the domain ontology, conversions and metadata of various data sources spanning from relational databases to text and XML based sources like *Cameleon* [12] in Prolog language is a very much error prone and cumbersome venture. One other problem is this would hinder the usage of COIN as a commercial application to model and mediate real life scenarios since the domain creator need to have the knowledge on the prolog abduction framework, how the constraints logic programming engine works and all the intricate details of the core mediation framework. In practice this would not be a viable solution and there is a need to abstract and encapsulate these complex structures and provide an easy –to-use interface for the mediation framework.

Thus second objective of this research was to build a graphical Metadata management and Domain ontology creation application that would alleviate the pain in programming in prolog. The user is facilitated with a Graphical modeling environment to define the Domain ontology consisting of the model of the domain under scrutiny, different contexts, automatically extract metadata from disparate data source (without manually entering the data source fields, data types and constraints) , and define contextual modifier values including both static and dynamic modifiers ,contextual Attribute management and Conversion functions, abstracting to a greater extend than the previous textual and sub-graphical interfaces [28, 29] and provided a new user experience in modeling in a true Graphical environment. The user has been provided with the functionality to generate the underlying application file structure including the prolog representation as well as other ontology modeling languages like Resource description framework (RDF) [31], RuleML [32], RFML [33] etc giving opportunity to differentiate, appreciate and understand different modeling mechanisms. The framework facilitates to use the Metadata modeling application in a *stand-alone* application context as well as in a web based application context, with the same look and feel and interface provided in both contexts.

Further I have come up with an extendable framework that is modular, encapsulating and abstracting each application layer using software engineering best practices and design patterns. This would facilitate future expansion and seamless module addition to the architecture proposed in the report.

The organization of the following sections is as follows. First we would be looking at the plethora of financial messaging standards that infest the financial world followed by related work in mediation technologies and specific work related to interoperability in the financial industry. Then we would be looking at the intricate details of the COIN mediation framework. Then I would address the importance of mediation of disparate standards from a business perspective. Next, the focus would be on the analysis done on how COIN can be applied in one of the critical industries in the financial world – The *Electronic Bill Presentment and Payment* (EBBP) industry – which comprises the first major deliverable from this research work. This would be followed by a section addressing the implementation details, architecture, design and functionality of the COIN Graphical Metadata Manager. In the final section, we summarize and briefly discuss future research.

2. Background Related work

2.1 Financial Standards

The standards addressed herein are involved in business banking, Electronic Bill presentment and Payment, Securities and Derivatives, Investments, Economic and Financial indicators, straight through processing and other *over the counter* derivatives. As a whole, financial industry is cluttered with numerous protocols and standards that are utilized in different segments in the financial industry. Prominent ones are Financial Information Exchange protocol (FIX), S.W.I.F.T., Interactive Financial Exchange (IFX) and Open Financial Exchange (OFX). SWIFT is the leader in inter bank transactions, but also have gained a significant market holding on Securities and derivatives, payments as well as investments and treasury after introducing a set of messages for securities and derivatives industry. OFX is the leader in Intra-bank transaction systems followed by its successor, IFX. IFX is opting to replace OFX, through its rich and extended messaging standards. Both of these standards are widely used in business banking, Electronic Bill presentment and Payment, ATM/POS Industry. FIX is the leader in securities and derivatives market, used by major stock markets around the world. Most of these protocols use XML as the medium of messaging. Non-XML based standards like FIX and S.W.I.F.T have come up with XML versions, namely *FIXML* and '*SWIFTStandards XML'*. In addition to these major players, some of the other protocols are *RIXML* – Research Information exchange and IRML – Investment research markup , focusing on fixed income securities and Derivatives market, MDDL - Market Data Definition and *REUTERS* in economic and industrial indicators, *STPML* – Straight through processing markup language - a superset protocol to replace *FIX,SWIFT ISITC* and *DTC ID*, *FinXML* – Financial XML which focuses on Capital market instruments and straight through processing (STP) and finally *FpML* - Financial products markup language focusing on interest rate swaps, forward rate agreements, Foreign Exchange and other *over the counter* derivatives.

2.2 Different mediation strategies

Over the last two decades, there have been several studies focusing on interoperability of disparate information sources.

These approaches have been grouped in the literature as static vs. dynamic [14], global vs. local schema [15], and tightly vs. loosely coupled [16, 17] approaches. These groupings can roughly be thought of referring to the same distinction characterized in [16] by:

· Who is responsible for identifying what conflicts exist and how they can be circumvented; and

· When the conflicts are resolved.

We will briefly look at these approaches under the categories of tightly and loosely coupled approaches.

In tightly coupled approaches, the objective is to insulate the users from data heterogeneity by providing a unified view of the data sources, and letting them formulate their queries using that global view. A system administrator takes the task of creating a global schema before the system can be used. In *bottom up* approaches the global schema is constructed out of heterogeneous local schemas by going through the tedious process of schema integration [18]. In *top-down* approaches global schema is constructed primarily by considering the requirements of a domain, before corresponding sources are sought. In tightly coupled approaches, data heterogeneities between sources are resolved by mapping conflicting data items to a common view. Early prototypes which have been constructed using the tight-coupling approach include Multibase [19], ADDS [20], and Mermaid [21]. More recently, the same strategy has been employed for systems adopting object-oriented data models (e.g. Pegasus [22] based on the IRIS data model), frame-based knowledge representation languages (e.g. SIMS [17] using LOOM), as well as logic-based languages (e.g. Carnot [23] using CycL, an extension of first-order predicate calculus).

Loosely coupled approaches object to the feasibility of creating unified views on the grounds that building and maintaining a huge global schema would be too costly and too complex. Instead they aim to provide users with tools and extended query languages to resolve conflicts themselves. Hence, instead of resolving all conflicts *a priori*, conflict detection and resolution are undertaken by receivers themselves, who need only interact with a limited subset of the sources at any one time. To facilitate this task, research on this front has focused on the invention of powerful data manipulation languages (DMLs) which allow queries on multiple sources to be intermingled with operations for data transformations. MRDSM [15] is probably the best-known example of a loosely-coupled system, in which queries are formulated using the multidatabase language MDSL. Kuhn et al [24] have implemented similar functionalities in VIP-MDBS, for which queries and data transformations are written in Prolog. They showed that the adoption of a declarative specification does in fact increase the expressiveness of the language in terms of allowable data transformations. Litwin et al [25] has defined another query language called O*SQL which is largely an object-oriented extension to MDSL.

2.3 Current integration and mediation strategies in Financial Standards

Due to intricacies and inefficiencies that exist in using and integrating multiple standards that incurs additional overhead , Financial institutions as well as government organizations have put effort in merging different standards or coming up with new super set standards to replace the existing diverse standards.

One example is the effort by FIX, SWIFT, OPEN APPLICATIONS GROUP and THE TREASURY WORKSTATION INTEGRATION STANDARDS TEAM (TWIST) to outline a framework of cooperation and coordination in the area of the content and use of a core payment kernel XML transaction.

Also the Organization for the Advancement of Structured Information Standards (OASIS) is carrying out research on one XML based super set protocol that would cover all business areas. But all these effort are focused on futuristic direction rather than the problem at hand. Even though they may lower cost and effort in the long term, the effort migrating the diverse world-wide standard to a common standard would be an enormous task. Current business integration efforts like the Microsoft^{TM} BizTalk Server support diverse messaging standards integration through its rich messaging and mapping framework, but lack the sophistication in mediating complex ontological and contextual heterogeneities in a declarative manner. Thus takes a considerable time and effort in mapping diverse context. There is a serious vacuum in such services and products that actually perform the mediation in an abstracted and a convenient manner.

2.4 COIN Metadata Management systems

There haven been some numerous Metadata Management systems being developed previously for COIN. The work done in [28] provides a JAVA based semantic modeling approach but lacks the detail and user friendliness in modeling them. Also the usage of third party graphical libraries might have slowed down the system. Further it lacks the graphical representation capability to model the context relationships as well as unable to extract the metadata associated with the underlying sources. A more recent work [29] has taken a different approach where the graphical modeling has been replaced by a textual interface for creating the semantic types, contexts, sources, elevation functions and modifiers. It provides a primitive graphical representation of the semantic model after the model has been created using the textual interface. But it does not provide a true graphical representation while modeling the domain ontology. Further a major area for errors is during the creation of sources where the user has to manually enter the source information including column names, types, uniqueness etc. But one major contribution from this work is the effort in modeling domain ontology in different representation by the likes of RDF, RuleML, RFML and prolog- the native language in COIN's abduction framework. Further the work provided mechanism to transform an ontology represented in one technology to another using XML Style Sheets (XSL/XSLT) [29]

3. The COntext INterchange (COIN) approach

The COntext INterchange (COIN) framework is neither a tightly coupled or loosely coupled system but it's rather a hybrid system. It is based on a mediator-based approach for achieving semantic interoperability among heterogeneous information sources. The approach has been detailed in [8]. The overall COIN approach includes not only the mediation infrastructure and services, but also wrapping technology and middleware services for accessing source information and facilitating the integration of the mediated results into end-users' applications. The set of context mediation services comprises a context mediator, a query optimizer, and a query executioner. In this paper we would give an overall view of the mediation framework in COIN.

The context mediator is in charge of the identification and resolution of potential semantic conflicts induced by a query. This automatic detection and reconciliation of conflicts present in different information sources is made possible by general knowledge of the underlying application domain, as well as the informational content and implicit assumptions associated with the receivers and sources.

The declarative knowledge is represented in the form of a domain model, different sources, a set of elevation axioms, and a set of context theories, respectively. The result of the mediation is a mediated query. To retrieve the data from the disparate information sources, the mediated query is transformed into a query execution plan, which is optimized, taking into account the topology of the network of sources and their capabilities. The plan is then executed to retrieve the data from the various sources; results are composed as a message, and sent to the receiver.

At the center of the mediation is the domain model/ontology, which defines the different elements needed to implement the strategy in a given application: The domain model/ontology is a collection of rich types (*semantic types, attributes, etc.*) and relationships (*is-a relationship*) defining the domain of discourse for the integration strategy. This declarative knowledge about the domain is represented independent of various information sources and represents the generic concepts associated with the domain under consideration which depends on the modeler's expertise and knowledge regarding the specific area.

Semantic types resemble the different entities in the underlying domain. For example *Account, Person* can be entities in a financial domain. The attributes represents the generic features those semantic types can have. i.e. *bankBalance, creationDate* attributes of *Account* semantic type. Further, attributes can be used to infer relationships between different entities. For example the *holder* attribute of an *Account* could refer a *person* semantic type. In some instance the attribute can constitute a basic type; either a string or a numeric value represented by the super semantic type, *basic*.

Then the different Contexts in the model represent the diverse representations that are being resolved. These contexts would encapsulate and segregate different semantical, contextual and ontological representations that the underlying data sources contain.

The mediation framework would be useless without the data sources. All these sources are represented using the *Source* concept where the type of the sources could be any data source ranging from a relational table, an XML stream, to a web page. Different wrapper implementations including the web aggregator *Cameleon* [12] provides different interfacing mechanisms to diverse sources.

The sources and the domain model needs to be linked in order to facilitate mediation. This is achieved through the definition of *Elevation axioms*. Its usage is two fold. First, each source is elevated to a Context definition, defined earlier. Second, each attribute of the source is elevated to a particular semantic object (instances of semantic types) that is represented in the Domain Model. This facilitates in bridging the link between the context-independent, '*generic'* domain model and the context dependent sources.

As the context definitions define the different interpretations of the semantic objects in the different sources or from a receiver's point of view we use a special type of attributes, *modifiers*, to define the context of a data type. For example *currency* modifier may define the context of objects of semantic type *moneyAmount,* when they are instantiated in a context (i.e., currency = USD).

Finally, there is a conversion library, which provides conversion functions for each *modifier* to define the resolution of potential conflicts. The conversion functions are defined in First order logic. The relevant conversion functions are gathered and composed during mediation to resolve the conflicts. No global or exhaustive pair-wise definition of the conflict resolution procedures is needed. Both the query to be mediated and the first order logic program are combined into a definite logic program (a set of Horn clauses) [26] where the translation of the query is a goal. The mediation is performed by an abductive procedure, which infers from the query and the First Order Logic programs a reformulation of the initial query in the terms of the component sources. The abductive procedure makes use of the integrity constraints in a constraint propagation phase to accomplish semantic query optimization. For instance, logically inconsistent rewritten queries are rejected, rewritten queries containing redundant information are simplified, and rewritten queries are augmented with auxiliary information. The procedure itself is inspired by the abductive logic programming framework [27] and can be qualified as an abduction procedure. One of the main advantages of the abductive logic-programming framework is the simplicity in which it can be used to formally combine and to implement features of query processing, semantic query optimization, and constraint programming. COIN framework elegantly addresses *data-level* heterogeneities among data sources expressed in terms of context axioms. Further it resolves certain *ontological* conflicts that exist between different sources and map to a common semantic type in the domain ontology. An example would be, in one source/context the notion of *payment* would consist of the net amount plus tax and relevant fees while in another source it would be just the net amount.

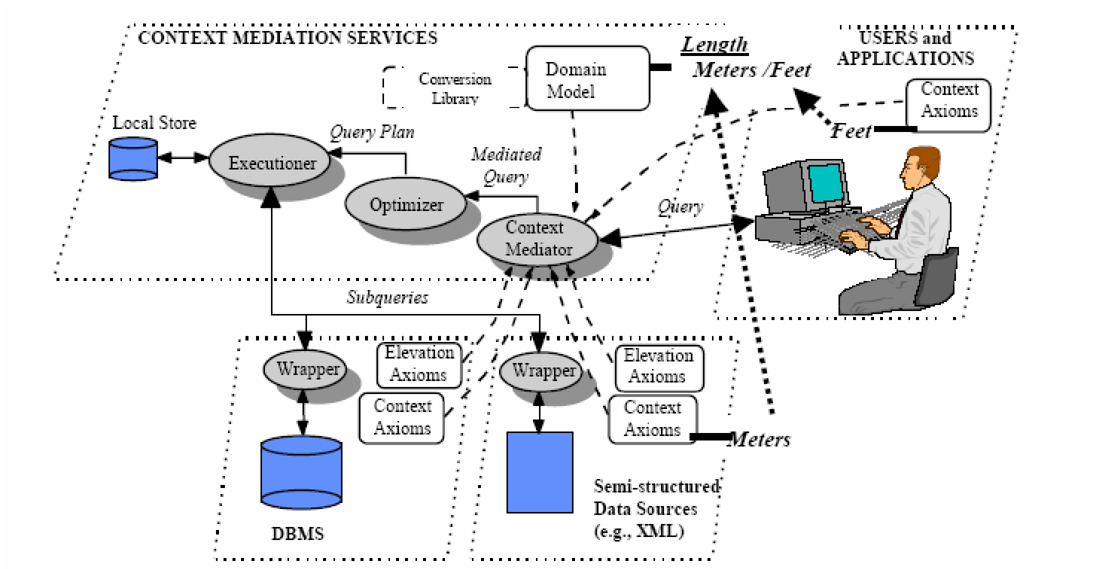


Figure 3.1: Architecture of COIN Mediation System

4. Why mediation is an important factor in Financial Standards?

As mentioned earlier Financial Industry is infested with various messaging standards that focus on subset of financial domains. For example messaging standards like FIX holds a major share of the Securities and derivatives industry where as IFX, OFX have the lion's share on EBPP and intra-bank transactions. In today's global economy interoperability and sharing of information is of prime importance. The need for efficient interoperability is one aspect of addressing the problem. In the other hand replacing all the diverse standards with a new global standard would be impractical considering the work requiring re-engineering major systems spanning multiple continents.

This has not been helped out with ever changing standards themselves. This requires in changing existing systems resulting time, effort and money. The following table summarizes different messaging standards that are used currently in financial domain.

Area	Standards
EBPP	- Interactive Financial Exchange protocol (IFX) - Open Financial Exchange protocol (OFX) - S.W.I.F.T.
Banking	- Interactive Financial Exchange protocol (IFX) - Open Financial Exchange protocol (OFX) - S.W.I.F.T.
Investments and Investment accounts	- Interactive Financial Exchange protocol (IFX) - Open Financial Exchange protocol (OFX)
Securities and Stock markets	- Financial Information Exchange protocol (FIX) and FIXML - S.W.I.F.T. - Financial products markup language (FpML) - Financial XML (FinML) - ISO 15022 XML – a super set protocol covering FIXML, FpML etc
Investment Research	- Research Information exchange ML (RIML) - Investment research markup language (IRML) - Research Information Exchange Markup Language (RIXML)
Market data and reporting	- Reuters - Market Data Definition language (MDDL) - Market Data Markup Language (MDML) - Extensible Business Reporting language (XBRL)
Mortgage Industry	- XML Mortgage Partners Framework - Mortgage Bankers Association of America MISMO Standard
Straight through processing	- Straight through processing markup - Financial XML (FinML)

Table 4.1 different financial standards

Most recently, banks, risk management firms, and insurance companies have been focusing on automating business processes and building systems that reduce the time from negotiating a trade to settling it to running risk analytics on trade positions. This is referred to as *Straight Through Processing* (STP); according to the Tower Group, the financial services industry will spend over \$12.2 billion on STP technology through 2005. The current trend in most of the standards bodies is to represent their standards using XML or some form of a markup language moving away from proprietary formats like FIX and SWIFT with corresponding XML versions, FIXML and SWIFTML. Current constraints on efficiencies in exchanging financial information have also been realized by SWIFT, which claims that its biggest priority is to move to XML. In fact, it is planning to incorporate FIX and FpML into a new standard referred to as ISO 15022XML. The resultant protocol is to leverage FIX Protocol's expertise in the pre-trade/trade execution domain, and Swift's post-trade domain expertise to bring together different parts of the trade life cycle and work through issues hindering straight-through-processing (STP).

Over the past 24 months, there has been a huge explosion of XML initiatives and it's arguable that this is what has prompted the two organizations to act quickly, before the situation spirals out of control. "With so many initiatives currently under way, there is a real danger that the standard could lead to the sort of divergence or even fragmentation that the industry has long battled to curb," says Alan Line, European co-chair of the FIX Steering Committee. "We believe that ISO 15022 XML will provide the glue between the pre-trade and post-trade domains. Pursuing a single best-of-breed standard is essential for the industry."

But this would clearly not address the semantic, ontological and temporal heterogeneities but would only look at unifying into a common data representation format.

Having such standards inevitability would result in issues in achieving global harmony and synergy. It’s not only in financial industry even in electronic commerce various standards have hindered the interoperability among business to business entities resulting in incurring heavy costs for interoperability [30]. These various standards would span from generic service provision frameworks, trading model, Payment models to security models and even mobile commerce frameworks. Table 4.2 categorizes various such standards and frameworks.

Sponsor	**Frameworks, models and Architectures**
Generic Models	
Microsoft	Biztalk Server - data representation and formatting
UN/CEFACT and OASIS	ebXML - messaging, semantics, processes and increasingly modeling
CommerceNet	eCo Framework- business processes/sub-processes
Internet Engineering Task Force	Electronic Commerce Modeling Language (ECML)
OMG Electronic Commerce Domain Task Force	OMG Electronic Commerce Domain Specifications
Trading Models	
IETF TRADE Working Group	IOTP - Internet Open Trading Protocol
Open Applications Group	Open Applications Group XML Framework
CommerceNet	OBI - Open Buying on the Internet
RosettaNet	RosettaNet
Payment Models	
VISA and MasterCard	SET - Secure Electronic Transaction
CEN/ISSS TC 224 and ISO/TC68/SC6	Trading and Payment Model in joint report on card-related secure commercial and financial transactions
Security Models	
IETF PKIX Working Group	PKIX

CEN/ISSS TC 224 and ISO/TC68/SC6	Security Model in joint report on card-related secure commercial and financial transactions
Mobile Commerce Models	
Ericsson, Motorola, Nokia, Panasonic, Siemens and Sony	MeT - Mobile electronic Transactions

Figure 4.2: Different standards used in electronic commerce

The proliferation of architectures is often accompanied by exhortations to converge efforts on a single architecture, or to pool resources to develop a common "baseline" or "reference". However, in so far as the vast majority of these architectures do not share the same foundations, especially at the level of business process and semantic definitions, it is difficult to see how convergence can be realized, even on technical grounds alone.

Generally interoperability has been plagued with formal standardization processors that are slow, unresponsive to needs and doesn't represent the all the interests of the stakeholders.

If we consider where standards and interoperability would matte in the value chain of providing services to clients, it's all about providing a common interoperable set of specifications. First when a standard has been introduced, it needs to be implemented into products by the vendors. But frequent changes in standards results in various software versions and re-engineering work from the vendor. Then the products need to be bought and used by users, corporate and individual clients for delivering and subscribing to a particular service. Changes in one's standards as well as migrating from one standard to another results in business downtimes incurring dissatisfaction and losses.

In this value chain of transforming standards into services, many parties or intermediaries could be involved, including service providers, integrators/aggregators, consultants etc resulting lots of levels interoperating together. Change in one level might affect the operation of other levels in a serious manner.

Even a standard like an IFX, OFX being 'open' standards is publicly available. But this is different from the implementation aspects or the architecture been adhered. Even though the standard is open there can be myriad of technical specifications, with different versions and even sometimes with different licensing and implementation rights. Lack of interoperability between implementations of the same specification as well as lack of continuation between standards of different versions is a common problem. This would be worsened with ones individual implementations being changed to cater diverse and changing requirements might lead to being deviating from standards adhered. Further compliance itself is a malleable term, depending upon the compliance criteria, which may not ensure objectivity, neutrality or completeness

Considering all these it's not just feasible to fix to a particular standard or replacing all the different standards with one 'global' standard. It requires intelligent, declarative and efficient mediation framework that would provide the required mediation between different standards as well as maintaining harmony among different versions of the same standard.

5. Analysis - Electronic Bill Presentment and Payment domain

5.1 Introduction to the Domain

In order to demonstrate the usage of the COIN framework's ability to mediate the diverse standards, a subset of the standards, namely '*Electronic Bill Presentment any Payment – (EBPP)* 'domain is selected. The EBPP sub domain represents the messaging standards adhered in the electronic bill payment industry which is widely been utilized presently by many financial institutions .EBPP domain is a rich subset of the financial services messaging frameworks that have considerable amount of heterogeneities. The main standards are OFX, IFX for intra-bank payment schemes and SWIFT for inter-bank payment and funds transfer.

The overall functionality can be visualized from Figure 1. The key intermediaries in an EBPP scheme are as follows:

- Biller Payment Provider (BPP) is an agent (usually a financial institution) of the Biller that originates and accepts payments on behalf of the Biller.
- Biller Service Provider (BSP) is an agent of the Biller that provides an electronic bill presentment and payment service for the Biller
- Customer Payment Provider (CPP) is an agent (usually a financial institution) of the Customer that originates payments on behalf of the Customer.
- Customer Service Provider (CSP) is an agent of the Customer that provides an interface directly to customers, businesses, or others for bill presentment. A CSP enrolls customers, enables presentment, and provides customer care, among other functions.
- Financial Institution (FI) is an organization that provides branded financial services to customers. Financial Institutions develop and market financial services to individual and small business customers. They may serve as the processor for their own services or may choose to outsource processing.

Both IFX and OFX provide XML based messaging framework for individuals as well as businesses in bill payment and presentment electronically. But the most acclaimed inter-bank fund transfer framework, SWIFT uses a non XML base messaging protocol and recently went through a major restructuring in phasing out one of the most utilized messaged for inter-bank customer fund transfer, the M100 and introduced modified versions of MT103 and MT103+.

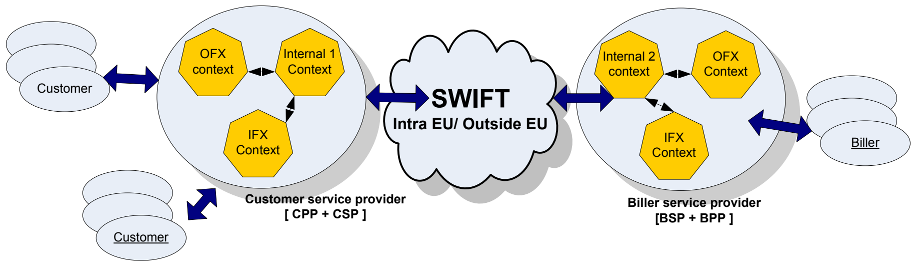


Fig. 5.1. Interfaces in EBPP

In order to depict the usage of COIN in EBPP mediation, the scenarios were divided into three aspects.

- Mediation between an internal context and OFX context.
- Mediation between an internal context and IFX context.

- Mediation between an internal context and SWIFT context.

The internal context represents the context and the schema associated with a particular Financial Institution's (FI) representation of the business domain. This could be an in-house developed system or third-party (off the shelf) system utilized for internal operations of the Financial Institution. The IFX, OFX and SWIFT contexts represent the semantics and definitions adopted by IFX, OFX and SWIFT messaging frameworks respectively. SWIFT distinguishes intra European Union (EU) fund transfer and outside EU fund transfers for accounting for inter-bank charges.

5.2 A Practical Scenario

This research would not be useful unless it addresses a real-life scenario. Let's look at a possible scenario involving paying a bill electronically spanning two countries, thus different financial systems.

Let's assume a Customer in NUS, Singapore uses his savings account in HSBC bank at the *Atrium@orchad* branch (location in Singapore) to pay a bill of Singapore Dollars 1870 equivalent to a German based *PCWorld* whose corporate account is with the Deutsche Bank in Frankfurt, Germany on April 20th 2004. He is using his online banking service of HSBC. HSBC uses OFX for the EBPP and Deutsche bank used IFX for the same purpose. Both Banks use S.W.I.F.T as the inter-bank fund transfer mechanism. The scenario is depicted in figure 5.2.

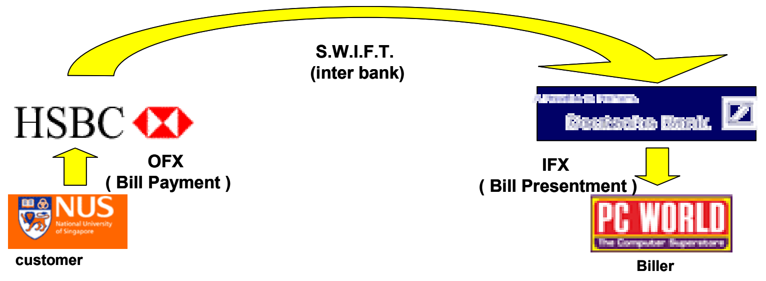


Figure 5.2: A simplified scenario

The first step is the OFX representation of the transaction from HSBC's payment gateway system as shown in fig 5.3. Please note that the fields / components in the transactions marked in grey are the aspects that have either semantic, ontological or simple data level heterogeneities with its adjacent messaging standard (i.e between HSBC internal system and S.W.I.F.T.)

`Debit Acct # 004-4356-4356 of acct type =`***'SAVINGS'*** `of` ***routing_transit_no =12343565SIN*** `With` ***1870 SGD*** `and Credit Acct # 4345-6543-9542 of acct-type='`***CHECKING***`' of` ***routing_transit_no =3234-5434_GD_DM*** `on` ***20040420***`.`

Figure 5.3 Step 1: OFX representations in HSBC's payment gateway

Then this OFX specific representation needs to be translated to the ontology and representational framework of HSBC's Internal Accounting system. This is required since unless their internet payment system would be a closed system without interacting with its banking backbone. Figure 5.3 denotes that representation.

```
Debit Acct # 004-4356-4356 of acct type ='SVG' of bankID =1345HSBC and
Branch_Id=1200012 With 1700 net + 170 tax SGD and Credit Acct # 4345-6543-
9542 of acct-type='CHK' of bank_id=bcl0000 and branch = 100021 on 2004/20/04
```

Figure 5.4 Step 2: Internal representation in HSBC

The reader should observe the different meanings for account type, account location description, date format payment representation (with and without tax) between step 1 and step 2 as shown in fig 5.3 and fig 5.4. These are the different semantic heterogeneities that exist between the internal and OFX representations.

After going through the internal system of HSBC, then to make the inter-bank fund transfer to Deutsche bank, HSBC uses S.W.I.F.T System. The representation in S.W.I, F.T would be as in figure 5.5.

```
Debit   Acct   #   4345-6543-9542   of   acct-type='SWIFT.SVG'   of
bank_BIC#=SIN_HSBC_4532 and Credit Acct # 4345-6543-9542 of acct-
type='CHK' of bank_BIC# =GD_DB_0043 on 040420 for a value of 1010 EUR
```

Figure 5.5: Step 3: S.W.I.F.T representation

Observer the differences in date format, source and destination bank Identifiers (that uses the ISO Bank Identification Code – BIC), change in currency from Sin$ to Euros and account type representation compared to the representation in HSBC's internal system. The SWIFT gateway system has to perform these transformations before actually sending the message to SWIFT system.

After receiving the SWIFT based representation by Deutsche bank, it needs to be stored in Deutsche Bank's Internal Accounting system. This is represented in figure 4.6: step 4. Observe the difference in account code representation (between fig 5.5 and 5.6), bank and bank branch Identification differences compared to the SWIFT's representation and meaning.

```
Debit Acct # 4345-6543-9542 of acct-type='SVG' of bank_id =HSBC_0012
and  branch_id=2345  and  Credit  Acct  #  4345-6543-9542  of  acct-
type='CHK' of bank_id =DB_0043 and branch_id=DB_03234 on 040420 for a
value of 1010 EUR
```

Figure 5.6: ste4: Deutsche bank's internal system representation

Now let's assume that the PC World Accountant check the daily transaction using his corporate account with Deutsche bank and it has a IFX based internet corporate account management system for its corporate clients. Figure 5.7 shows the representation as well as heterogeneities between how IFX represent the transnational details and Deutsche bank's internal system.

```
Debit Acct # 4345-6543-9542 of acct-type='SDA' of
bank_BIC#=SIN_HSBC_4532 and Credit Acct # 4345-6543-9542 of acct-
type='DDA' of bank_BIC# =GD_DB_0043 on 2004-04-20 for a value of 1010
EUR.
```

Figure 5.7: Step 5: IFX representation

Observe the differences in the account type code, bank Identification mechanism and date format in figure 5.7 which identifies the conflicts between IFX and internal system. By even observing this simplified example the reader should be amazed of the heterogeneities that exist in different financial systems and accounting systems and the effort required to convert between these.

This would be further aggravated if these standards keep on changing requiring in modifying the software systems frequently. The COIN mediation framework's objective is to mediate such heterogeneities in a user transparent, declarative and automated manner.

5.3 Domain Ontology

Figure 5.8 represents the context independent, domain ontology for the EBPP domain denoting the concepts used by IFX, OFX, SWIFT and financial institution's own internal schema. The semantic types denote the entities and their relationships in the EBPP domain. Further the entities that constitute conflicts in the contexts are modeled through modifiers. As an example, the *paymentAmount* can include/exclude various taxes in different contextual representations and in SWIFT it would incur an additional inter-bank service charge. These are represented by *paymentScheme*, *includesInterBankCharge* modifiers respectively. Further all monetary amounts would have conflict in different currency usage. This is modeled using the *currency* modifier for the super-semantic type *moneyAmount.* The *paymentAmount* inherits from *moneyAmount*. Therefore it would inherit the modifiers and their context dependent values, defined for its super type, *moneyAmout*. The *phone number, payment amount, invoice amount, dates, account code* and *money amount* has one or more modifiers to represent their semantic heterogeneities. COIN framework's modifier inheritance have been put into use in the *is-a* relationship of *money amount* and *payment amount*. Payment amount not only posses two modifiers on its own but would inherit the *currency* modifier of its super semantic type, *moneyAmount.* *Bankname* and *personname* have been extended from *Identifier* semantic type while *bankLoc* and *branchLoc* has been extended form *location* semantic type. The main semantic types are payment, invoice, *paymentAmount* and *moneyAmoun.* The *invoice* semantic type represents the invoices associated with a payment. In some contexts it includes fees as well as taxes while in other contexts not. This is represented using *invoiceScheme* modifier.

The mediations that we have focused in this section resemble some actual real life scenarios that are faced while attempting heterogeneous systems integration. Following sections addresses each one of them separately.

In order to carry out the analysis we need to model the Sources that would represent the data and schema in different contexts. Even though they have semantic, temporal and ontological heterogeneities the logical schema of all those data sources would be quite similar. Figure 5.9 denotes such a relational schema that represents a generic schema on how each context would constitute.

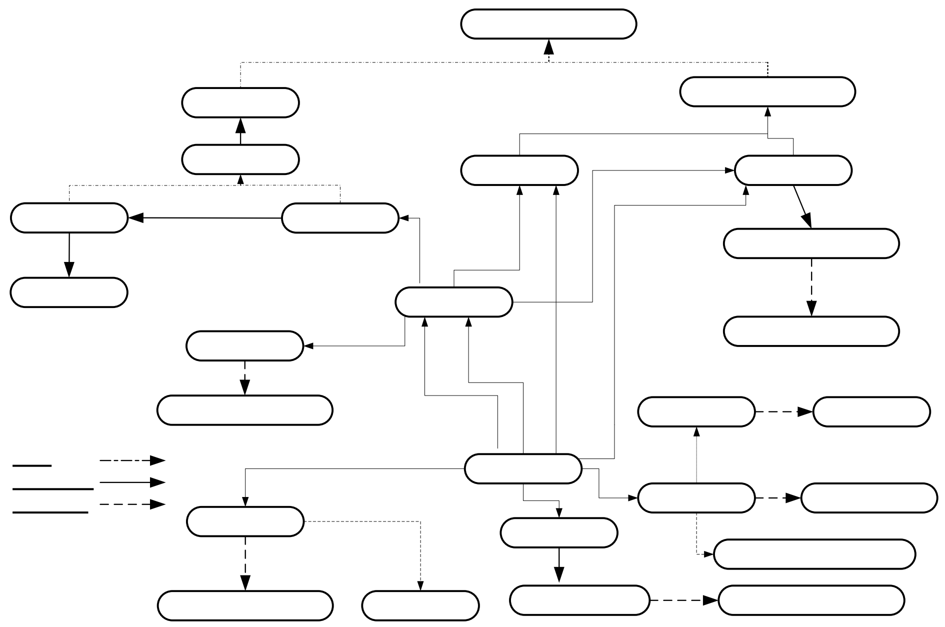


Fig. 5.8. Domain Ontology for EBPP

5.4 Internal Schema vs. OFX

First we will look at the mediations attempted between OFX and an internal schema of a financial institution. Table 2 summarizes the heterogeneities identified in the two schemas. As denoted in COIN's mediation strategy, the modifiers and relevant conversion functions are the main ingredients in facilitating the mediation for a particular heterogeneity exiting between two different contexts. As shown in table 2, there are different types of heterogeneities between the two contexts. The significant conflicts are payment amount, currency type and Account code reference identifiers. They are discussed in the next sections.

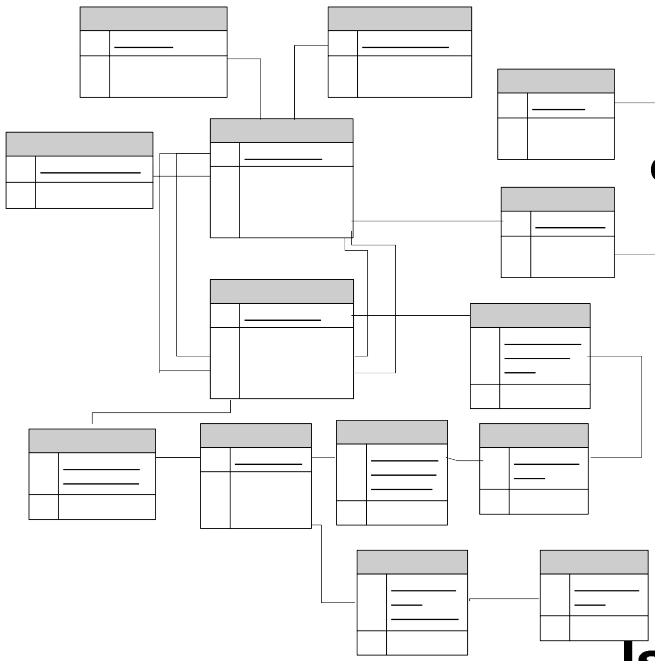


Figure 5.9: Generic relational schema for EBPP Domain

Table 5.1: conflicts in Internal and OFX contexts

Conflict	Internal Ontology	OFX Ontology	Mapped modifier (refer ontology)
Payment amount	Net amount without tax	Net + tax amount	*PaymentScheme*
Account Location Identifier – BANK reference	Bank identifier represented in the internal scheme	Bank Identifier depends on the Bank's country of Incorporation.	*BankLocType*
Account Location Identifier – BANK BRANCH reference	Branch identifier of the account	Branch Identifier dependent on the bank's country of incorporation.	*BranchLocType*
Payment due date format	European format	US format	*DateFormat*
Payment due date Style	dd/mm/yyyy 03/03/2003	Yyyymmdd 20030303	*DateStyle*
Account type code	CHECKING,SAVINGS etc	CHK,SVG etc	*AccountCodeScheme*
Currency type (Exchange rate)	"EUR"	Currency of country of incorporation of payee bank	*currency*
Phone number format	415.445.4345	1-415-445-4345	*PhoneNumberScheme*

Payment amount - The mediation strategy for payment amount is as follows. The mediator needs to apply two conversion functions in order to obtain the mediated payment amount, namely the currency conversion inherited from the *moneyAmount* super semantic type, and the tax adjustment for the payment. For simplicity let's assume that in both schemas the currency is denoted in three letter ISO 4217 format (i.e. USD, GBR, and EUR etc).

Assume that the query '*select AMOUNT FROM PAYMENT'* is called in *OFX* context;

First, payment amount is adjusted for the tax inclusion. For simplicity let's assume that the applicable tax is 'GST'. Then;

$$\text{Payment}_{OFX} = (\text{payment}_{INTERNAL} + \sum \text{GST amount for payment}_{OFX} * \text{payment}_{INTERNAL}) \quad \text{* Exchange Rate ("EUR", OFF\_CUR,DATE\_OF\_TRANSACTION)} \qquad (1)$$

In the COIN framework, the mediation formulas are translated into logical expressions of the COIN theoretical model [1].Later these expressions are implemented in prolog and evaluated by an abduction engine implemented in the same language [13]. The following describes the logical representation of the formula (1) for this example.

The formula below describes a non-commutative mediation of *paymentType* object depending on its modifier *paymentScheme*, in this case hold the values "*noTax*" and "*withTax*". The *Ctxt* defines the destination context. The conversion in simple terms would be to retrieve the Rate for the tax "GST" from the elevated relation '*OFX_TAX_TYPES_p'* which is an elevation mapped to relation '*OFX_TAX_TYPES'* under *OFX* Context (The destination context in this case) and utilizes in the tax calculation. The *value* predicate in the formula defines a value of a particular semantic object under a certain context.

```
cvt(noncommutative,paymentAmt,_O,paymentScheme,Ctxt,"notax",Vs,"withtax",Vt) ⇐
      value(TaxName,Ctxt,"GST"),'OFX_TAX_TYPES_p'(TaxName,_,Rate),
      value(Rate,Ctxt,RR),
      (Vtemp is RR * Vs),
      (Vt is Vs + Vtemp).
```

Further, this resembles an *Equational ontological heterogeneity* addressed in [5], which is a clear example of differences in the two ontologies for OFX and internal contexts. But the ontological conflict has been transformed into a contextual heterogeneity by way of matching the definitional equations as in [5].

Then, this tax adjusted payment needs to be mediate to the currency of *OFX* context. This requires in a *dynamic modifier* to extract the currency value depending on the official currency in the incorporated country of the payee's bank as given below.

$$\begin{array}{lll} \mathit{OFF\_CUR}_{OFX} = \mathit{Currency}_{OFX}(\mathit{payment}) & \Longleftarrow & \mathit{AID = Payee\ Account\ of\ Payment}_{INTERNAL} \\ \mathit{BRANCH}_{OFX} & \Longleftarrow & \mathit{Branch\ of\ Account\ AID}_{OFX} \\ \mathit{BANK}_{OFX} & \Longleftarrow & \mathit{Bank\ of\ BRANCH}_{OFX} \\ \mathit{COUNTRY}_{OFX} & \Longleftarrow & \mathit{country\ of\ Incorporation\ of\ BANK}_{OFX} \\ \mathit{OFF\_CUR}_{OFX} & \Longleftarrow & \mathit{official\ currency\ of\ COUNTRY}_{OFX} \end{array} \qquad \mathbf{(2)}$$

The following logical representation describes how the value of modifier *currency* for *paymentAmount* is obtained for *OFX* context dynamically through the relationships between semantic objects.

```
modifier(paymentAmt,_O,currency,ofx,M) ⇐   ( attr(_O,paymentRef,Payment),
attr(Payment,payeeAct,Account),
                              attr(Account,location,Location),
                              attr(Location,bank,Bank),
                              attr(Bank,countryIncorporated,Country),
                              attr(Country,officialCurrency,M))).
```

For example the predicate *attr (Payment,payeeAct,Account)* defines the attribute relationship '*payeeAc*' between the *Payment* and *Account* semantic objects. This relation can be mapped to underlying relationships in different contexts as shown in the following logical representation.

```
attr(Payment,payeeAct,PayeeAcct)⇐
                        ('INTERNAL_PAYMENT_p'(Payment,_,_,_,_,_,PayeeAcct,_).
attr(Payment,payeeAct,PayeeAcct) ⇐
                        ('OFX_PAYMENT_p'(Payment,_,_,_,_,_,PayeeAcct,_).
```

The two statements correspond to how the attribute relation *payeAcct* has been elevated to two elevation relations with their attributes, mapped in *INTERNAL* and *OFX* contexts.

This will ensure that the correct currency would be extracted for the OFX side. But now the current exchange rate needs to be applied in order to get the actual value in that currency represented in OF from that of internal representation (i.e USD to EURO).

This is achieved through a conversion function as following. We use an elevation axiom called *olsen_p* that is mapped from a *Cameleon* [12] data source that extracts the current exchange rate from a configured URL . The resultant output of the conversion function would be the converted payment amount.

```
%% rule for currency conversion-----.
rule(cvt(commutative,paymentAmt,O,currency,Ctxt,Mvs,Vs,Mvt,Vt),
(olsen_p(Fc, Tc, Rate, Date),
     value(Fc, Ctxt, Mvs),
     value(Tc, Ctxt, Mvt),
     value(Rate, Ctxt, Rv),
     currentDate_p(CurDate),
     value(CurDate, Ctxt, DateValue),
     value(Date, Ctxt, DateValue),
     multiply(Vs, Rv, Vt))).
```

Account type code - This is represented as heterogeneity in enumerated data types in defining the account type codes in the three contexts. Table 3 summarizes the enumerated data mapping in the three contexts. Since there can be more than two types of financial standards, rather than having mappings between each standard , we adopt a 'Indirect conversion with ontology inference' strategy [13] where we represent the different account types in the ontology itself and providing mapping between the context independent ontology's enumerated type and the context sensitive type codes. The context model would then map each security type context construct into its corresponding security type ontology construct.

Therefore usage the above mapping from *INTERNAL* to *OFX* would be,

$$\begin{aligned} Account\_type_{OFX}(\ Account\_type_{INTERNAL}(\text{'CHK'})) \Leftarrow\ & ONTOLOGY\_TYPE_{INTERNAL} = \text{'CHKA' [table INTERNAL]} \\ & ONTOLOGY\_TYPE_{NONE} = \text{'CHKA' [table Ontology]} \\ & ONTOLOGY\_TYPE_{OFX} = \text{'CHKA' [table OFX]} \\ & OWN\_TYPE\ (\text{'CHK'})_{OFX} = \text{'CHECKING' [table OFX]} \end{aligned} \quad (3)$$

Table 5.2: conflicts in Internal and OFX contexts

Ontology : Account types
Table Ontology

ONTOLOGY_TYPE	Description
CHKA	Checking account
SVGA	Savings account
MNYMRTA	Money Market Account
CRLINEA	Credit Line Account

Mapping between Internal and Ontology
- Table INTERNAL

ONTOLOGY_TYPE	OWN_TYPE
CHKA	CHK
SVGA	SVG
MNYMRTA	MNYMRT
CRLINEA	CRLINE

Mapping between OFX and Ontology
- Table OFX

ONTOLOGY_TYPE	OWN_TYPE
CHKA	CHECKING
SVGA	SAVINGS
MNYMRTA	MONEYMRKT
CRLINEA	CREDITLINE

Mapping between IFX and Ontology
- Table IFX

ONTOLOGY_TYPE	OWN_TYPE
CHKA	DDA
SVGA	SDA
MNYMRTA	MMA
CRLINEA	CDA

5.5 Internal Schema vs. IFX

After looking at some of the interoperability issues between internal context and OFX, now we would delve into the newer standard, IFX, which has more features and detailed representations.

Table 4 shows the different types of heterogeneities. The conflicts of *account type*, *date format*, *phone number format* and *currency types* are similar to the OFX scenarios. The new conflicts are the extended conflicts identified in *payment amount* and introduction of *invoice* related conflicts.

Table 5.3: Conflict between Internal and IFX contexts

Conflict	**Internal Ontology**	**IFX Ontology**	**Mapped modifier (Refer ontology)**
Payment amount	Net amount	Net + $\sum$ tax amount + $\sum$ Fees	*PaymentScheme*
Payment due date format	European format	US format	*DateFormat*
Payment due date Style	dd/mm/yyyy 03/03/2003	Yyyy-mm-dd 2003-03-03	*DateStyle*
Account type code	SVG,MNYMRT,CRLINE, CHK etc	SDA,MMA,CCA,DDA etc	*AccountCodeScheme*
Invoice Amount	Net amount	Net + $\sum$ tax amount + $\sum$ Fees	InvoicePayment-Scheme
Currency type (Exchange rate)	"GBP"	Currency of country of incorporation of payee bank	*Currency*
Phone number format	415.445.4345	1-415-4454345	*PhoneNumberS-cheme*

Both IFX and OFX handle complex business payment transactions for business customers. This requires incorporating multiple invoice details attached to the payment aggregates when both the biller and customer are business entities. The older OFX provides a basic mechanism of incorporating invoice details like invoice discounts, line items in invoices etc. But the newer IFX extends this by providing more elaborate aggregates constituting different tax schemes as well as fees (late fees, FoRex fees) etc that are applicable to invoice.

Mediating Invoice Amount

Each payment can have at least one invoice aggregate that represent the different invoices paid through a particular invoice. In an internal schema the invoice amount might be represented as the net amount, where the taxes and fees would be aggregated when the bill is presented or invoiced. But the IFX context, the Invoice amount constitute of the various taxes and fees that could be added to the net amount.

The mediation between the two invoice amounts represents an Equational ontological conflict (EOC) [5] that would be resolved through introduction of a set of modifiers that would match the two different definitional equations. Each invoice would have multiple fees .i.e. an invoice would have FoRex, late payment fees, import fees as well as multiple taxes like GST, withholding taxes etc
Therefore the relationship between the two definitional equations for invoice amount would be:

$$\begin{aligned} \mathit{InvoiceAmount}_{IFX} = {} & \mathit{InvoiceAmount}_{internal} + \\ & \sum (\mathit{InvoiceAmount}_{internal} * \mathit{FeeRate}_{IFX}) + \sum (\mathit{FixedFee}_{IFX}) + \\ & \sum (\mathit{InvoiceAmount}_{internal} * \mathit{TaxRate}_{IFX}) + \sum (\mathit{FixedTax}_{IFX}) + \end{aligned} \quad (4)$$

The current COIN framework does not facilitate aggregate functionality, where all the applicable fees and taxes would not be SUM'ed up dynamically. A workaround strategy is to define modifiers for each tax and fee type and associate its applicability with contexts and exhaustively define which fees and taxes that could be applicable under a certain context.

Let's say we executed the query `'select INVOICE_AMOUNT from INTERNAL_INVOICE'` in *IFX* context where the relation `INTERNAL_INVOICE'` is defined for *internal* context. Let's take an example where a particular invoice with GST and IMPORT as tax components and LATE

FEES and DELIVERY as Fees in addition to the selling price. Therefore in IFX context all these needs to be considered in addition to the face value represented in the INTERNAL context.

Table 5 summarizes the heterogeneities that exist in the two contexts.

Cost component	Internal	IFX
GST	No	Yes *(percentage)*
Late Fees	No	Yes *(fixed amount)*
Import Taxes	No	Yes *(percentage)*
Delivery Fees	No	Yes (*Fixed)…..*

Table 5.4: additional fees and taxes in IFX

The COIN mediation framework mediates these ontological conflicts that persist between the two formulas which is represented by the following datalog.

```
answer('V23'):-
            'IFX_TAX_TYPES'("GST", 'V22', 'V21'),
            'V20' is 'V19' * 'V21',
            'IFX_TAX_TYPES'("IMPORT", 'V18', 'V17'),
            'V16' is 'V19' * 'V16',
            'V15' is 'V20' + 'V17',
            'IFX_FEES_TYPES'("LATE", 'V14', 'V13'),
            'V12' is 'V19' + 'V13',
            'IFX_INVOICE_FEES'("DELIVERY", 'V11'),
            'IFX_INVOICE_FEES'("LATE", 'V11'),
            'IFX_INVOICE_TAXES'("IMPORT", 'V11'),
            'IFX_INVOICE_TAXES'("GST", 'V11'),
            'INTERNAL_INVOICE'('V11', 'V10', 'V19', 'V9', 'V8', 'V7', 'V6'),
            'IFX_FEES_TYPES'("DELIVERY", 'V5', 'V4'),
            'V3' is 'V19' * 'V4',
            'V2' is 'V12' + 'V3',
            'V1' is 'V15' + 'V2',
            'V23' is 'V19' + 'V1'.
```

Then the following shows the mediated SQL query automatically generated by the COIN mediation framework considering all the conflicts associated between *internal* and *IFX* contexts:

```
Select (internal_invoice.INVOICE_AMOUNT +  (((internal_invoice.INVOICE_AMOUNT *
ifx_tax_types.AMOUNT) + ifx_tax_types2.AMOUNT) + ((internal_invoice.INVOICE_AMOUNT +
ifx_fees_types.AMOUNT) + (internal_invoice.INVOICE_AMOUNT * ifx_fees_types2.AMOUNT))))
        from   (select 'GST', TYPE, AMOUNT from ifx_tax_types
        where  TAX_NAME='GST') ifx_tax_types,
       (select 'IMPORT', TYPE, AMOUNT from ifx_tax_types
        where  TAX_NAME='IMPORT') ifx_tax_types2,
       (select 'LATE', TYPE, AMOUNT from ifx_fees_types
        where  FEES_NAME='LATE') ifx_fees_types,
       (select 'DELIVERY', INVOICE_NO from ifx_invoice_fees
        where  FEE_NAME='DELIVERY') ifx_invoice_fees,
       (select 'LATE', INVOICE_NO  from ifx_invoice_fees
        where  FEE_NAME='LATE') ifx_invoice_fees2,
       (select 'IMPORT', INVOICE_NO from ifx_invoice_taxes
        where  TAX_NAME='IMPORT') ifx_invoice_taxes,
       (select 'GST', INVOICE_NO from   ifx_invoice_taxes
        where  TAX_NAME='GST') ifx_invoice_taxes2,
       (select INVOICE_NO, PAYMENT_ID, INVOICE_AMOUNT, DESCR, INVOICE_DATE,
        DISCOUNT_RATE, DISCOUNT_DESC
        from   internal_invoice) internal_invoice,
       (select 'DELIVERY', TYPE, AMOUNT from   ifx_fees_types
        where  FEES_NAME='DELIVERY') ifx_fees_types2
        where  ifx_invoice_fees.INVOICE_NO = ifx_invoice_fees2.INVOICE_NO
        and    ifx_invoice_fees2.INVOICE_NO = ifx_invoice_taxes.INVOICE_NO
        and    ifx_invoice_taxes.INVOICE_NO = ifx_invoice_taxes2.INVOICE_NO
        and    ifx_invoice_taxes2.INVOICE_NO = internal_invoice.INVOICE_NO
```

Some readers may have so far considered that identifying and resolving semantic heterogeneity is a small matter of handling date formats, currency exchange, and other accounting conventions. We observe now that the net effect and accumulation of such small matters makes the programmer's task impossible. A programmer not equipped with the COIN mediation system must devise and type the above query. A programmer using the COIN mediation system can type the original query: '*select INVOICE_AMOUNT from INTERNAL_INVOICE*' in *IFX* context and rely on COIN to automatically mediate the query. The application gains in clarity of design and code, as well as in scalability. The sharing of domain knowledge, context descriptions, and conversion functions improve the knowledge independence of the programs and their maintainability.

5.6 Some insight to conflicts analysis between internal and SWIFT contexts

The SWIFT protocol is mainly involved in inter-bank cross border transactions. It uses globally unique identifiers for bank code like BIC, BEI. For e.g. the BIC code comprise of concatenation of bank code, country code and location code (defined by ISO 9362), compared to just a bank code representation used in internal schema. This peculiar heterogeneity requires in a non-commutative building up of a composite bank identifier when mediating from *internal* to *SWIFT* context. Following represents a logical formula for the mediation for the concatenation. The predicate notations were discussed in a previous example.

```
cvt(noncommutative,bankLoc,O,idType,Ctxt,"single",Vs,"composite",Vt) ⇐
            ('SWIFT_BANK_BCI_p'(BANK, LOC, COUNTRY),
             value(BANK,Ctxt,Vs),
             value(LOC,Ctxt,Locc),
             value(COUNTRY,Ctxt,Countryc),
             (Vtemp is Vs + Locc),
             (Vt is Vtemp + Countryc))).
```

Usage of sub contexts

Under *SWIFT* context, depending on whether the transaction is between financial institutions inside the EU or outside, a bank handling fee is credited to the payment amount. This can be modeled using the *sub context* concept of COIN. A sub context would derive all the super context based modifier values while having specialized modifier values for extended features. The following logical formulas denote how this can be modeled in COIN.

```
is_a(swift_intraEU,swift)
is_a(swift_outsideEU,swift)
```

Then a query like '*select amount from payment'* in *outsideEU* context called on a relation defined for internal context would be resolved by adding the handling charges on top of the local applicable tax (inherited from *SWIFT* context) as denoted in the following mediated datalog. Observe the usage of *SWIFT_CHARGE_TYPES* elevated relation to obtain the charges.

```
answer('V15'):-
        'INTERNAL_PAYMENT'('V14', 'V13', 'V12', 'V11', 'V10', 'V9', 'V8', 'V7'),
        'TAX_TYPES'("GST", 'V6', 'V5'),
        'V4' is 'V5' * 'V12',
        'V3' is 'V12' + 'V4',
        'SWIFT_CHARGE_TYPES'("outsideEU", 'V2', 'V1'),
        'V15' is 'V1' + 'V3'.
```

Following shows the corresponding mediated SQL query

```
select
(swift_charge_types.AMOUNT+(internal_payment.AMOUNT_NET+(ofx_tax_types.AMOUNT*
internal_payment.AMOUNT_NET)))
from   (select PAYMENT_ID, FROM_ACCT_ID, AMOUNT_NET, GST_PERCENT,
OTHER_TAX_PERCENT, PAYEE_ID, PAYEE_ACCOUNT, DUE_DATE
     from   internal_payment) internal_payment,
    (select 'GST', TYPE, AMOUNT
     from   ofx_tax_types
     where  TAX_NAME='GST') ofx_tax_types,
    (select 'outsideEU', TYPE, AMOUNT
     from   swift_charge_types
     where  TAX_NAME='outsideEU') swift_charge_types
```

Note that although datalog and prolog representations and used internally within COIN and shown in this paper, the actual COIN system provides a user-friendly interface so that a user need not know anything about these internal representations.

Confusion in Interpretation

Before the introduction of its new message set, SWIFT MT 103 series, the already obsolete SWIFT MT 100 message was interpreted in different ways by the vendors. Since SWIFT deals mainly with inter-bank transactions, this aggravate to a bigger issue. The charging option field of a MT 100 message was interpreted differently by vendors, when it was not mentioned in a message. Vendors treated in three different ways.

1. Borne by the beneficiary
2. Share the charges.
3. And even sometimes by the creditor.

The resultant would be that the wrong party would have paid while the other party benefiting from the transaction and sometimes double accounting and paying the intermediary twice.

This in fact is a major reason for SWIFT to change the older version with a newer and more robust set of messages.

5.7 SWIFT vs. OFX

As we have seen, SWIFT use ISO defined Bank Identification Code (BIC). But in OFX the usage of bank identification number is quite dependent on the country of usage. The *From bank* and *To bank* identification in a payment message depends on the standards adhered in one country. Table 6 summarizes some examples.

Country	Interpretation
BEL	Bank code
CAN	Routing and transit number
CHE	Clearing number
DEU	*Bankleitzahl*
ESP	*Entidad*
FRA	*Banque*
GBR	Sort code
ITA	*ABI*
NLD	Not used (field contents ignored)
USA	Routing and transit number

Table 5.5: Account location identification differences

Therefore in an instance where there is a direct interaction between SWIFT and OFX, depending on the country of usage, a particular mapping needs to be applied. In order to model such relationship sub contexts needs to be defined for each country under OFX sub contexts as shown in fig 5.10.

```
is_a(ofx_USA,ofx)
is_a(ofx_GBR,ofx)
is_a(ofx_CAN,ofx)
```

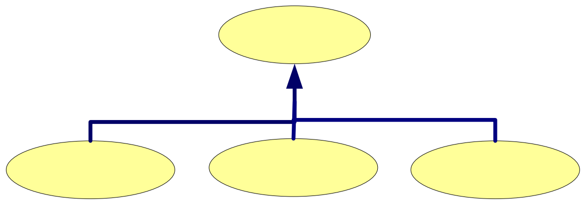


Fig. 5.10

Then a conversion function for a bank identification code between `ofx_USA` and `swift` would look like the following.

```
cvt(commutative,bankLoc,O,idType,Ctxt,"ofx_USA",Vs,"swift",Vt) ⇐
            ('SWIFT_BANK_BCI_p'(BANK, LOC, COUNTRY),
            ('OFX_USA__BANKID_p'(OFX_ID, SWIFT_LOC),
             value(OFX_ID,Ctxt,Vs),
             value(BANK,Ctxt,BID),
             value(SWIFT_LOC,Ctxt,BID),
             value(LOC,Ctxt,Locc),
             value(COUNTRY,Ctxt,Countryc),
             (Vtemp is BID + Locc),
             (Vt is Vtemp + Countryc))).
```

Here the country specific bank identification # (In this case the Bank routing and transit number used in *ofx_USA* context is mapped to a bank ID using the mapping elevation '`OFX_USA__BANKID_p'`. Then the corresponding *SWIFT* elevation `'SWIFT_BANK_BCI_p'` is used to map and compose the SWIFT based composite bank location identifier composing location and country details.

5.8 Temporal Heterogeneities

For now we looked at ontological and contextual heterogeneities. But there might be scenarios where there are such conflicts depending on temporal issues as given in table 7. There has been some work in progress carried on resolving temporal issues [34].

Property	Internal Schema	OFX	IFX	SWIFT 103/103+
Price	Net	Net + tax of 5% on and before 2000 , Net + tax of 2% after 2000	Net + tax of 5% on and before 2000 , Net + tax of 2% after 2000	(Net + tax of 5% on and before 2000 , Net + tax of 2% after 2000) + inter-bank charges.
Currency	FFR on and before 2000, EUR after 2000.	Currency of country of incorporation of payee bank	Currency of country of incorporation of payee bank	Explicitly mentioned- ISO 4217

Table 5.6: temporal heterogeneities

Scenario 1: temporal conflict in payment

Before January 1st 2004, in the *internal* context the payment amount was represented as the net amount without any tax component.

But in for OFX schema it needed to be represented as:

$$Payment_{OFX} = + (VAT\% \text{ before } 2004) * Payment_{INTERNAL} + defence\_tax_{OFX}\%$$

Scenario 2: change in Currency

Before December 1st 2003, all payments of the *internal* schema was represented in Deutsche Marks (DM) .But after the introduction of *Euro*, all the monetary amounts were represents using *EURO*. Since the OFX schema represents *moneyAmounts* using currency of incorporated country of the payee's bank, on the due date of the payment, the currency value needs to be denominated from DM to respective currency before December 31st 2003 and from *EUROs* after that.

```
If ( dueDate < 12-01-2003 ){
    $moneyAmount_{OFX} = moneyAmount_{internal} *$
    $ExchangeRate(currency_{PAYEE\_BANK\_INCORPORATIO\_COUNTRY}, currency_{DM})$
}
else{
    $moneyAmount_{OFX} = moneyAmount_{internal} *$
    $ExchangeRate(currency_{PAYEE\_BANK\_INCORPORATIO\_COUNTRY}, currency_{EURO})$
}
```

Before introducing temporal modifiers, such a currency modifier would be represented in COIN as following.

```
rule(modifier(paymentAmt,_O,currency,internal,M),(cste(currency,M,internal,"FR"))
).
```

But when temporal modifiers been included to the framework the additional predicate called *containObj* was incorporated. The following code denotes how the temporal heterogeneity is modeled using the same currency modifier.

```
%% - before 2004
modifier(paymentAmt,O,currency,internal,M):-containObj([bottom, 2003], O),
cste(currency, M, internal, "DM").

%% after 2004
modifier(paymentAmt,O,currency,internal,M):-containObj([2004, top],O),
cste(currency, M, internal, "EUR").
```

5.9. Hierarchical schemas Vs. COIN

AS explained in chapter 3, COIN is modeled and designed for mediating between relational data sources. The mediation framework uses an abduction framework to resolve conflicts among relational sources. To a certain extend this has been deviated by using Cameleon [12] where the hierarchical markup structure of HTML was mapped a relational schema. This has certain limitations where it can't handle other generic markup language structures as well as cannot be used directly with the mediation framework. But in the context of mediating financial standards that constitute of complex hierarchical data structures in XML and other mark-up languages, COIN lacks in directly interfacing with them. Figure 5.11 depicts a possible interaction between the relational mediation framework and the XML message standards. Therefore in order facilitate direct interaction and mediation among these disparate standards the following need to be incorporated.

1. Translation from hierarchical representation to relational model and vice versa.
2. In addition to handling semantic heterogeneities, there is a need to mediate between the different hierarchical mappings with each other. There can be two ways of addressing this issue.
   a. Each hierarchical structure (for e.g. the invoice sub tree and likewise of an IFX, OFX message) could be mapped to a set of semantic types in the corresponding Domain ontology. This would be scalable when the number of standards increases. But the representational mapping in the domain ontology should be very much flexible in handling various intricate conflicts among the representation mechanism. This would be much more complex and dynamic compared to the domain ontology reference mechanisms addressed in previous financial mediation case studies involving enumerated data types as in [13].
   b. Each standard would be having a mapping to each other standard that it directly communicates. This would grow in an uncontrollable manner if the set of standards increase or there are frequent modifications and version changes in standards resulting in changing all the dependent mappings.

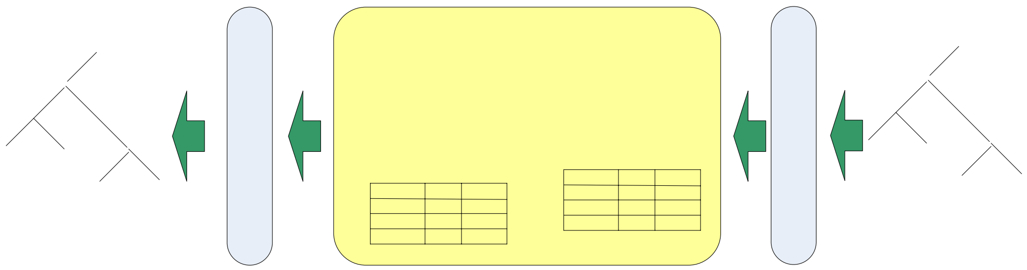


Fig. 5.11: COIN and hierarchical models

One strategy in facilitating the interaction is to extend the current Cameleon architecture to facilitate complex hierarchical models. But the current Cameleon model only handles HTML to relational mapping but not the reverse. Therefore the following extensions need to be incorporated into Cameleon.

1. More generic markup language modeling (i.e. XML) and transformation functionality rather than only handling HTML.
2. Modeling more complex hierarchical structures as in actual financial messages involving EBPP.
3. Transformation from a relational domain model /mediation framework to a hierarchical model.
4. This would definitely involve in some performance bottlenecks that need to be addressed separately.

A more ambitious approach would be in replacing the relational model based mediation /abduction framework with an all new semantic web related methodology like W3C's emerging OWL [35]. This would alleviate the need for the two transformation layers as shown in the above figure. Even though this would result in lesser complexity it requires additional mediation and abduction framework built on a methodology which is still in its infancy compared to the well tested and proved relational model. Further if COIN needs to mediate relational as well as hierarchical data sources, interoperability and combined abduction /mediation would not be feasible unless relational data sources were mapped back to hierarchical representation and vice versa. Therefore careful consideration should be given in adopting such a strategy.

6. COIN Metadata Manager

6.1 Introduction

Context Mediation Framework is a complex piece of software that requires understanding on the modeling and representation of the domain ontology and the declarative conversion functions that constitute the crux of a mediation application. Writing the mediation code in prolog based language running up to several pages without the help of a application template or modeling paradigm is a mere nightmare for the application developer. Therefore the aim of this User centric graphical COIN Metadata manager is to alleviate the trouble in understanding complex prolog code and give the freedom to model the required Domain focusing on the business domain rather than worrying about how to code in prolog. The previous work done in this area [28, 29] have been focused on web based textual interfaces that are naïve and difficult to model and understand. The focus of this effort is to provide a user friendly graphical modeling methodology to the COIN user. The application is divided into two parts.

1. The Web based (JAVA Applet) graphical modeling tool.
2. The stand-alone graphical modeling tool.

Figure 6.1 denotes the overall architecture of this tool from the user interface up to the application code generation. As depicted the framework supports in modeling the domain ontology sources, contexts, elevation axioms, and attribute and modifier definitions as well as generate the corresponding prolog application file for the application under consideration.

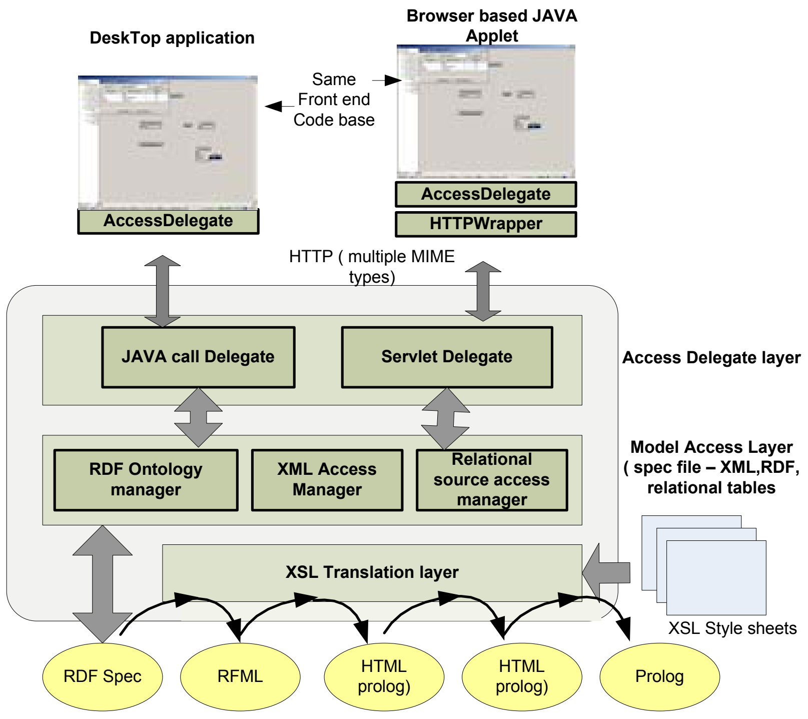


Fig 6.1 Overall Architecture

6.2 Overall Architecture

The overall architecture basically comprises of four modules. The classes associated in each layer is denoted in class diagram in fig 6.2

1. **Application and java Applet loaders.**

   This module is responsible in loading the main Metadata manager screens in Applet and Stand-Alone contexts. Specifically in the applet context, the Loader ensures to load the application as a singleton without requiring re-fetching of the code base for each application load.

   In Standalone context, the *AppLoader* would provide the option for the user to select the application to model. This will load the main Metadata Manager. In comparison the *AppletLoader* is linked with the existing web based textual Interface. The applet loaded in a particular application would be loading the Metadata Manger dynamically from the HTTP Session context represented in the web server.

2. **User Interface modeling modules comprising the web based and stand-alone graphical modules.**

   Both Applet based as well as Application based system uses the same UI model and classes. But depending on the context the respective loader would configure the properties of how to fetch data. The Graphical Interface provides the following features.

   i. Graphically model the semantic types, attributes and modifiers of the Domain under consideration providing a drag and draw/ aligning feature.
   ii. Define and graphically display the context hierarchy.
   iii. Extract Metadata from relational data bases and XML/TEXT data source like Cameleon to create sources for the application.
   iv. Model Elevation axioms for the sources created in step (iii).
   v. Define context based dynamic and static modifiers.
   vi. Contextual Attribute definition.

3. **Resource connection layer.**

   Comprising of the class *AccessDelegat,* this acts as the delegate for backend communication. This is implemented in line with the *Delegate / Controller* Design pattern used in software engineering design practices. The AccessDelegate hides and abstracts the backend resource access mechanism to the front end code, namely the graphical Interfaces.

   If the application is running under Stand-alone context, the access Delegate directly accesses the back end resource access code running in the same JAVA virtual machine.

   But if it's running in Applet Context, the *AccessDelegate* used a wrapper class called *HTTPWrapper* to connect to a JAVA Servlet running in COIN Application server sending HTTP GET and HTTP POST requests to get and send data respectively, where in turn the Servlet would use the backend resource access code running in its virtual machine. Refer figure 6.2. The back end Servlet *ServletDelegate* delegates the resource access to resource access layer classes. All this is different strategies are transparent to the application user in both contexts. The following code snippet shows how the access delegate calls 'generateApplication function, through the HTTPWrapper or a direct JAVA call depending on the application context (Standalone or Applet).

```
public void generateApplication(String application,HashMap poss){
    if( clientType.equals(Constants.CLIENT_APP)) {
      rdfm.generateApplication(application,poss);
    }
    else if(clientType.equals(Constants.CLIENT_BROWSER) ){
       System.out.println(" ogign to generate " + application   );
                 HttpWrapper.putResource(HttpWrapper.baseurl + "?" +
                 Constants.request + "=" + Constants.GEN_APP +
                 "&" + Constants.application  + "=" + application,poss);
}
```

Correspondingly, the following code segment shows how the HTTPServlet submits results back to the Servlet that intern would use the resource access classes to update the back ed data structures.

```
public static void putResource(String url,Object value){
    Object ret = null;
    URL studentDBservlet = null;
    URLConnection servletConnection = null;

    try{
      studentDBservlet = new URL( url );
    }
    catch(MalformedURLException  e){
      System.out.println(" malformed URL" + e.getMessage() );
    }
    try{
     servletConnection = studentDBservlet.openConnection();
      servletConnection.setRequestProperty ("Content-Type", "application/octet-
      stream");
      servletConnection.setDoInput(true);
      servletConnection.setDoOutput(true);

      ObjectOutputStream outputToServlet = new
      ObjectOutputStream(servletConnection.getOutputStream());

     outputToServlet.writeObject(value);
     outputToServlet.flush();………………
```

4. Resource and Data access layer.

The main interface that is exposed by this layer is the *RDFManager* class. This is developed inline with *Manager* Design pattern where it delegates and dispatches the request to the relevant sub modules but keeps them transparent to the invoker and facilitates in adding different modules to the resource access APIs yet making it transparent.

This layer is responsible in providing the following services.

i. Load Domain Ontology, source, elevation and context information for RDF files (*RDFManager, RDFWriter*)
These sub modules extract and loads the Domain ontology and the relevant ontological information to a memory hierarchy to be served to the Metadata manager interfaces. The loading is done from the RDF Files pertaining to the schema.

ii. Generates the RDF application files after application has been modified. (*RDFManager, RDFWriter*)

iii. Generates RuleML, RFML , prolog files form the generated application (*RDFManager , XSLTransformer, PrologGenerator*)

iv. Extracts metadata from Relational as well as XML/TEXT data sources (*RDFManager , DBMetaDataExtractor*)

v. Provides the system properties and configuration data for defining resource parameters, folder and file paths. For more information refer 6.2.

JAVA is used as the main programming language and uses JAVA 2D Graphics, JAVA Swing, JAVA Servlet and JSP API, JAVA net and Applet programming, RDF, XML and XSL API.

6. 3 Graphical Domain Ontology modeling

One of the main deliverables of the project is the graphical modeling of the Domain ontology. It provides a user friendly way if visualizing the domain including the semantic types, their modifiers, Attributes and Relations (is-a, modifier, attribute) among them as shown in figure 6.3. The user points and clicks to create semantic types and even delete/ modify them. As the user prefers he can place the objects by dragging them to the appropriate positions. This is modeled inline with professional modeling tools like MS Visio where the user is given the option to lay out the ontology or design. The positional information and rendering data is stored transparently when the Ontology files are generated. Therefore when the application is reloaded the previously stored positional information would be used to reconstruct the Ontology Layout on screen.

User can modify /add and delete the attributes and modifiers for each semantic type. Saving a Semantic Type with its super semantic type, attributes and modifiers would perform the following actions.

1. Semantic type would be added to the domain ontology data structures.
2. New semantic type would be added to the display context on the clicked location with the attributed and modifiers and their Domain and range given inside the semantic type.
3. The Domains of the attributes and modifiers as well as the super semantic type would be linked with different notations.

Then the user can modify the semantic type(s) as well as dragging to appropriate place in the drawing pane.

The following small code segment shows how a Semantic Type is added and rendered in the Graphical panes.

```
public  void drawSemancitType(int X,int Y,SemanticType sc){
  // Graphical renderer for semantic type
  semanticTypeRenderer renderer = new semanticTypeRenderer(sc.semanticName );
  renderer.semantictype = sc;
  HashMap attributes = sc.attributes ;
  if(sc.attributes .size() > 0){
    Iterator itr = attributes.values() .iterator() ;
    // add the attribute
    while(itr.hasNext() ){
      renderer.addAttribute((Attribute)itr.next() );
    }
  }
  Insets insets = DrawingPanel.getInsets();
  Dimension size = renderer.getPreferredSize();
   semanticTypeRenderMap.put(sc.semanticName , renderer) ;

 // add to the drawing pane …
  DrawingPanel.add(renderer)  ;
     System.out.println(renderer.semantictype.attributes .size());
renderer.setPreferredSize(new Dimension(getWidth(sc) + 10,20 + 20
*renderer.semantictype.attributes .size() )) ;
  // set bounds for the palen object for semantic type
  renderer.setBounds( X + insets.left,Y + 10 + insets.top,
renderer.getPreferredSize().width, renderer.getPreferredSize().height);
```

And part of the rendering section called asynchronously for repainting the drawing context.

```
  Iterator itr = semanticTypeMap.values() .iterator();
 SemanticType type = null;
 while( itr.hasNext() ){
```

```
type = (SemanticType)itr.next(); // for all semantic types

if(semanticTypeRenderMap.containsKey(type.semanticName) ){
    renderer =  (semanticTypeRenderer)semanticTypeRenderMap.get(type.semanticName );

   renderer.show() ;
   // draw the lines for relationships
   drawLinesForSemanticType(type,gr);
```

6.4 Contexts

The COIN framework mediates the heterogeneities between different contexts. For example in the financial standards addressed previously, the different standards were represented in different contexts. The context hierarchy might grow down to several levels where the sub contexts would inherit all the modifiers and conversion functions from its super contexts. Modeling contexts is a vital part of the Metadata Manager. Figure 6.4 denotes how the Metadata Manager graphically models the contexts with their relationship with the super contexts. When a context is created and given the super context type, the context is automatically inserted to the correct position in the graphical context model, making the required relationships.

This model can be used in order to visualize the static modifiers created for each context in the modifier management sub module. It enables the application modeler to view the relationships between the contexts as well as the contextual values they hold.

6.5 Metadata Extraction for Source creation

In previous efforts in modeling the Domain Ontology, the user has to manually enter the relation name, column names, data types etc when he wants to create a SOURCE for the application. This is one error-prone area where user can enter wrong information and would be difficult to debug. Therefore this Metadata Manager provides a Interface for the following meta data extraction.

1. logs in to the schema of a required database Server (This can be Oracle, MS SQL , MySQL , Sybase like commercial databases – where depending on the user selection the database access drivers would be loaded dynamically) and extract the table and view Metadata comprises of the following
    a. Relation type (tables , views etc)
    b. Column names of each relation
    c. Database specific Column types (i.e NUMBER, VARCHAR etc in Oracle).
    d. Column properties – Part of Primary Key, Uniqueness, Nullable etc

Further the user can give a table/view filter string to filter out the table/view names that he want to view for adding relations to the COIN while narrowing down the search in a large schema. Refer fig 6.7.

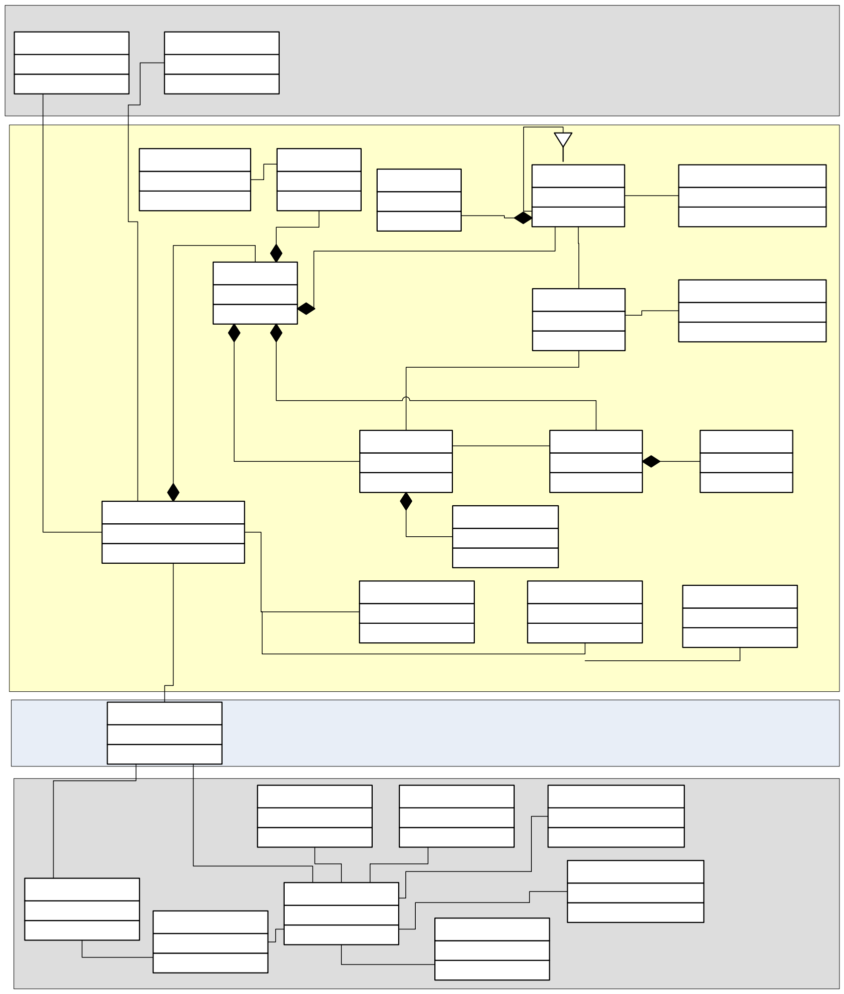


Fig 6.2. Overall Class Diagram showing the main classes

The following code segment shows how the schema Metadata was extracted from a given schema in Oracle.

AppletLoader

```
if( dataBaseType.equals(Constants.ORACLE ) ){
    dmd = (OracleDatabaseMetaData)con.getMetaData();
    if( dmd == null){
      System.out.println(" no meta data available");
      return relations ;
    }
    else{
      ResultSet rs = dmd.getSchemas(); // schema abstraction
      ResultSet rs1 = null;
      if (schema != null){ // always need a schema
```

1

1

Mc

```
      while (rs.next()) {

      if(schema.equalsIgnoreCase(rs.getString(1))){
        // get tables for the given schema(s)
        rs1 = dmd.getTables(null, rs.getString(1), "%", null);
        while (rs1.next()) {
             relations.put(new String(rs1.getString(3)) , new String(rs1.getString(4)));
             // add name and type of relation.
   ……………………
```

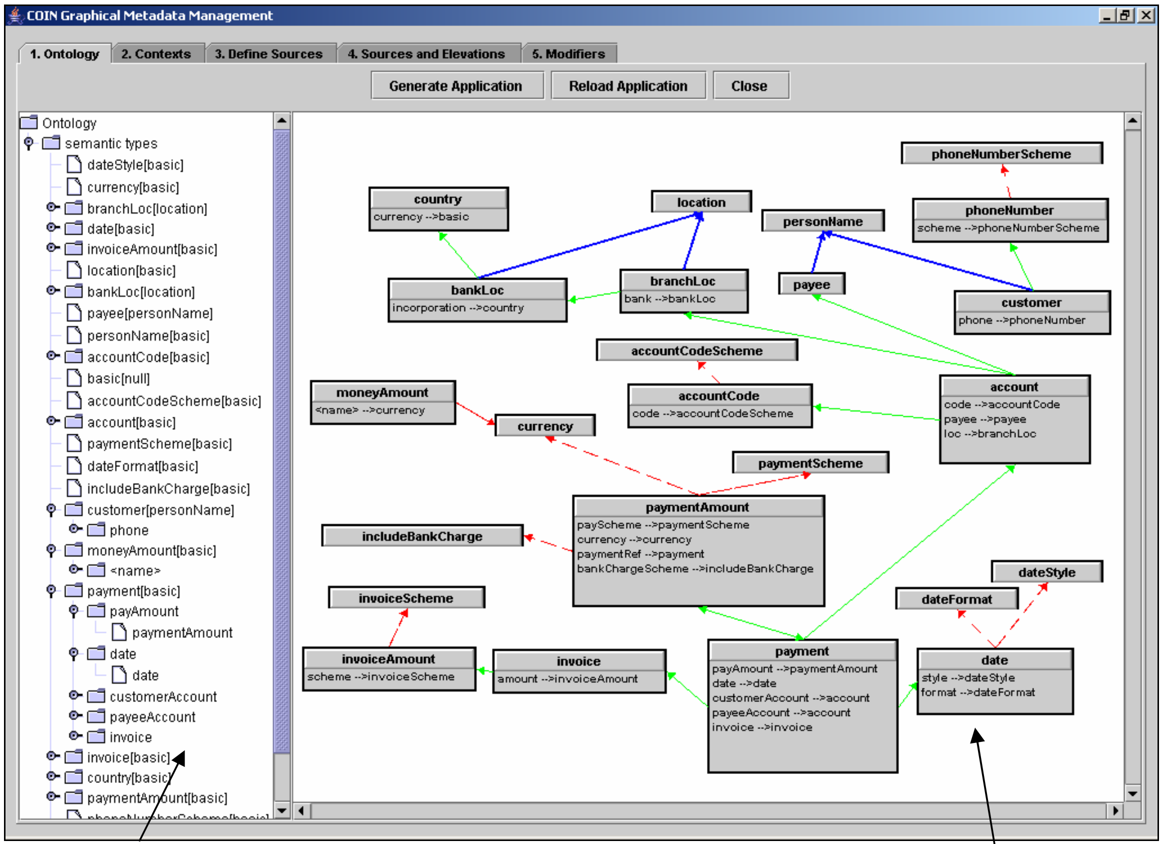
And the following shows a condensed version of how Metadata for a particular selected relation is extracted.

```
  public Vector getTableColumnMetaData(String tableName){
   Vector columns = new Vector();

   if (dataBaseType.equals(Constants.ORACLE) ){
   try{
    ResultSet rs =  dmd.getColumns(null,schema.toUpperCase(),tableName,"%");
    while(rs.next() ){  // get each column.
     if (rs.getString(18).equals("NO") ){
        isNull = rs.getString(18);
     }
     else{
        isNull = "YES";
     }
     columns.add(new ColumnData(rs.getString(4),rs.getString(6),isNull) );
    }……
```

2. When the user is using the stand-alone system, he can select a Cameleon spec file and extract the spec name and columns of the web source defined in the spec file. The applet based system can only be used if the Applet is signed with a digital certificate that is trusted by the host system since generally an Applet runs in a san boxed environment in its virtual machine.
   For either of the source types, the user can define the respective COIN data types for each column and their uniqueness in COIN's context, namely STRING or NUMBER. Also user can define which columns need to be added to the COIN relation.
3. If the user is unable to connect to the database servers or Cameleon Spec folder, user can select manual option and create a relation and relation type, column names, their types and primary key associativity.

As shown in the architecture for the Meta data access layer in fig 6.5, Relational Source Access Manager module is responsible for extracting metadata from different databases while the XML/TEXT data access Manager extracts XML or Cameleon spec file metadata. The Metadata Filter layer is responsible in filtering the data as per the selected schema and table name filters specified by the user.



Ontology Tree

Fig 6.3. Domain ontology in Metadata Manager

Graphical model

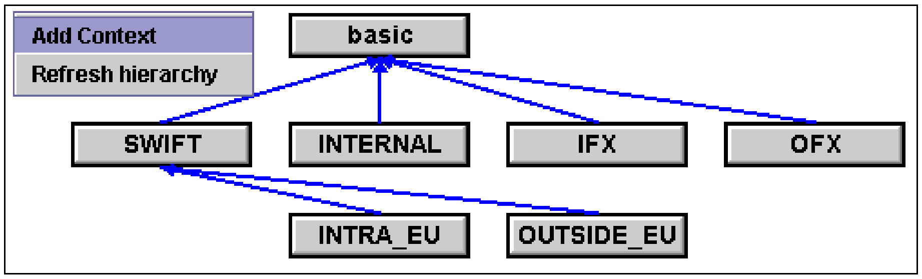


Fig 6.4. Context Hierarchy

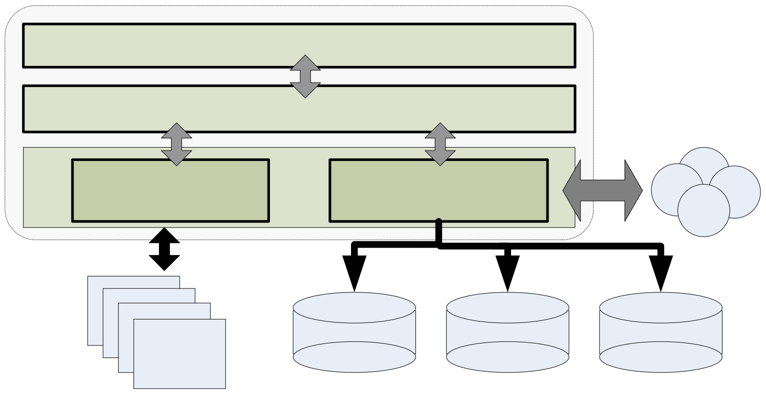
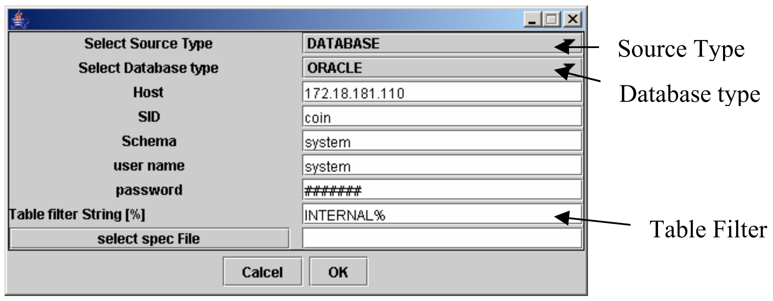
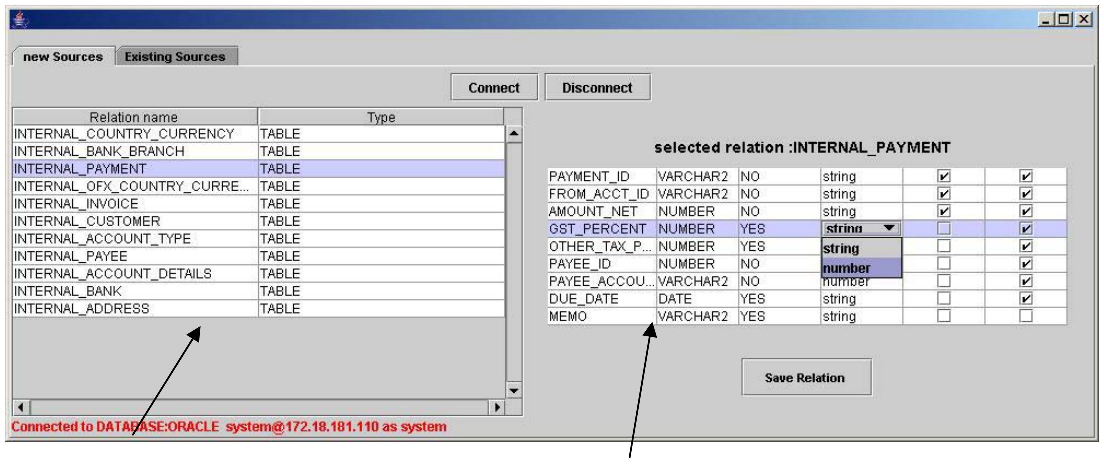


Fig 6.5. Metadata Access



Select Source Type	DATABASE
Select Database type	ORACLE
Host	172.18.181.110
SID	coin
Schema	system
user name	system
password	#######
Table filter String [%]	INTERNAL%
select spec File	

Fig. 6.6 Selecting a source



Relation name	Type
INTERNAL_COUNTRY_CURRENCY	TABLE
INTERNAL_BANK_BRANCH	TABLE
INTERNAL_PAYMENT	TABLE
INTERNAL_OFX_COUNTRY_CURRE...	TABLE
INTERNAL_INVOICE	TABLE
INTERNAL_CUSTOMER	TABLE
INTERNAL_ACCOUNT_TYPE	TABLE
INTERNAL_PAYEE	TABLE
INTERNAL_ACCOUNT_DETAILS	TABLE
INTERNAL_BANK	TABLE
INTERNAL_ADDRESS	TABLE

Connected to DATABASE:ORACLE system@172.18.181.110 as system

selected relation :INTERNAL_PAYMENT

PAYMENT_ID	VARCHAR2	NO	string	✓	✓
FROM_ACCT_ID	VARCHAR2	NO	string	✓	✓
AMOUNT_NET	NUMBER	NO	string	✓	✓
GST_PERCENT	NUMBER	YES	string	☐	✓
OTHER_TAX_P...	NUMBER	YES	string	☐	✓
PAYEE_ID	NUMBER	NO	number	☐	✓
PAYEE_ACCOU...	VARCHAR2	NO	number	☐	✓
DUE_DATE	DATE	YES	string	☐	✓
MEMO	VARCHAR2	YES	string	☐	☐

Relation names and type (view, table)

Selected relation: The columns, data types, is NULL, selectable COIN data type, select PK cols and check columns to add to COIN

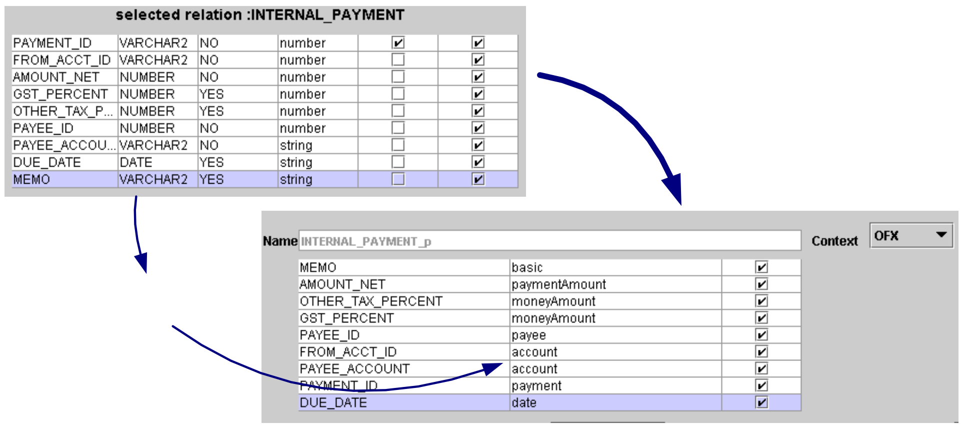
Fig 6.7. Extracted metadata for a relational table

6.6 Elevation Axioms

After creating the resource, the next step is to define the elevations of those resources to particular contexts. This mapping can be done in a manner where the user can select an existing COIN relation and a context, and define the columns applicable to the Elevation. For each column in the elevation axiom, the user can define the COIN semantic type mapping for elevation and whether he wishes to add the column to an elevation as shown in fig. 6.8. The user has been facilitated with viewing the corresponding column properties of the resource while creating the elevations resulting is lesser errors. Also quick context switching between the elevation creation and Ontology model would help in deciding on the appropriate semantic type for an elevated column.

6.7 Modifier Management

One of the core features of the COIN framework is, defining modifiers for different contexts. Modifiers can be defined statically as well as dynamically. Static modifies are easy to define. For e.g. the following code segment denotes a statically defined for *currency* modifier for *paymentAmt* in *internal* context.



selected relation :INTERNAL_PAYMENT

PAYMENT_ID	VARCHAR2	NO	number	✓	✓
FROM_ACCT_ID	VARCHAR2	NO	number	☐	✓
AMOUNT_NET	NUMBER	NO	number	☐	✓
GST_PERCENT	NUMBER	YES	number	☐	✓
OTHER_TAX_P...	NUMBER	YES	number	☐	✓
PAYEE_ID	NUMBER	NO	number	☐	✓
PAYEE_ACCOU...	VARCHAR2	NO	string	☐	✓
DUE_DATE	DATE	YES	string	☐	✓
MEMO	VARCHAR2	YES	string	☐	✓

Name INTERNAL_PAYMENT_p Context OFX

MEMO	basic	✓
AMOUNT_NET	paymentAmount	✓
OTHER_TAX_PERCENT	moneyAmount	✓
GST_PERCENT	moneyAmount	✓
PAYEE_ID	payee	✓
FROM_ACCT_ID	account	✓
PAYEE_ACCOUNT	account	✓
PAYMENT_ID	payment	✓
DUE_DATE	date	✓

Fig 6.8: Elevation Axioms creation

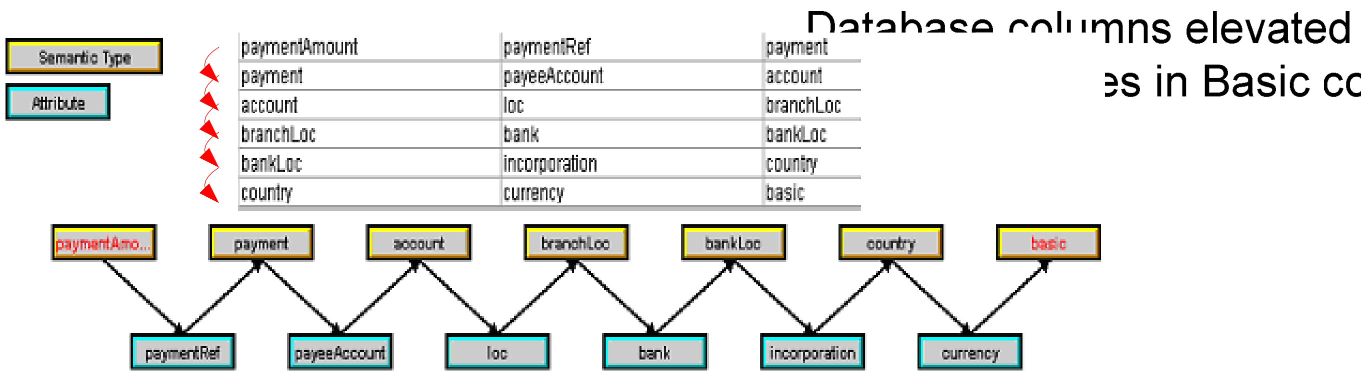
```
rule(modifier(paymentAmt,_O,currency,internal,M),(cste(currency,M,interna
l,"GBP"))).
```

But defining a dynamic modifier is tricky using prolog. A simple dynamic modifier for the same *paymentAmt* for *OFX* context would be as following.

```
rule(modifier(paymentAmt,_O,currency,ofx,M),(  attr(_O,paymentRef,Payment)
,attr(Payment,payeeAct,Account),
attr(Account,location,Location),
attr(Location,bank,Bank),
attr(Bank,countryIncorporated,Country),
attr(Country,officialCurrency,M))).
```

Defining such modifiers spanning several attribute relations manually might results in errors. To facilitate dynamic modifier creation, functionality was added to the Metadata Manager to define the modifiers graphically. Figure 5.9 depicts how a dynamic modifier is modeled. The *paymentAmount* is the semantic type and corresponding modifier value is the basic semantic type mapped to the *currency* attribute of *Country* semantic type.

The graphical structure shows how the relationship propagates from *paymentAmount*, *payment*, *account*, *branchLoc*, *bankLoc*, *country* semantic types and attribute relationships.



paymentAmount	paymentRef	payment
payment	payeeAccount	account
account	loc	branchLoc
branchLoc	bank	bankLoc
bankLoc	incorporation	country
country	currency	basic

Fig6.9: Dynamic modifier creation

6.8 Attribute Management

Another aspect of the creating an application in COIN is to create and define Attributes and the relevant mapping in a context. In some instances, an attribute relation might span across multiple relations propagated through the referential integrities. Creating such attributes using prolog would be error prone and painstaking. The Attribute management module facilitates in creating and defining the attribute relations in a graphical and tabular manner.

For e.g if the *currency* relationship in the EBPP domain might differ in different contexts. In OFX context the currency attribute would like the following:

```
rule(attr(Country,officialCurrency,Currency),

('COUNTRY_CURRENCY_p'(X,Currency),'OFX_BANK_p'(_,_,Country),
value(Country,ofx,U),value(X,ofx,U))).
```

But using the graphical metadata manager this could be mapped to three types of relationships as given in the table below.

Fig 6.1: Attribute Mapping types

Type	Example
Map to DOMAIN	Country attribute of 'OFX_BANK_p'
Map to RANGE	Currency attribute of ('COUNTRY_CURRENCY_p'
Map ELEVTION	X attribute of 'COUNTRY_CURRENCY_p' = Country attribute of 'OFX_BANK_p'

Using these three mapping a tabular representation can be envisioned by the user while viewing the automatic graphical model generated by the metadata Manager. The *Map to DOMAIN* can be used to map the Semantic type of the Domain of the attribute to a elevated column while '*Map to RANGE'* can be used to map a elevated column to the Range of the attribute. Any referential relationship can be modeled through 'Map to ELEVATION' mapping. The graphical representation for the above prolog based attribute definition would look like as in fig 6.10. This representation can be modeled to any extension by using the above three types of mappings.

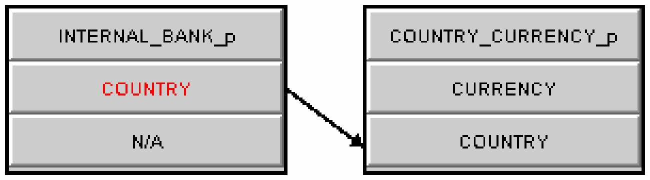


Fig 6.10: graphical Attribute modeling

6.9 Multiple Ontology generation & representation mechanisms

The main mechanism of storing the domain ontology is W3C's Resource description framework (RDF). Back end *RDFManager* creates the required source, ontology and context files for the application. Then these can be translated into multiple formats including RuleML[REF] , RFML [REF] as well as non markup language like prolog. We use XML Style sheets (XSL / XSLT) to convert the RDF to different formats as shown in figure 6.5. The final transformation step translates the HTML based prolog file to a plain prolog file that would be deployed automatically in the application space of COIN.

The interfacing between a stored application and the user interface is done through RDF files rather than the end result of prolog files. This is due to couple of reasons.

1. The Metadata manager is modeled using Object Oriented Technologies. The ontological components and definitions are modeled using classes.
2. Resource description Framework is quite a rich language in modeling ontologies even though there are richer definitions like DAML+ OIL,OWL {REFER}. RDF provides a similar concept to an OOP's class, attribute and Object definitions.
3. It's easy to model and map between an Object oriented framework and a RDF Schema through transformation between Objects and XML based hierarchy.
4. Complex XML parsing and modeling APIs can be utilized in manipulating (adding, modifying and deleting) RDF elements which are XML Elements and attributes.
5. Using Extensible Style Sheets (XSL, XSLT) it is easy to transform one markup language based model to another. Therefore facilitates interoperability among different schema modeling mechanisms.
6. Relative to prolog, the hierarchical modeling capability in RDF like in any other markup base languages makes modeling and programming easy while providing easy transformation to other formats using existing tools.

But since the COIN Abduction and mediation engine relies on a prolog based application, The Application Generation generates the prolog based application file as the final step. As an example refer figure 6.11. For a small representative comparison on how a semantic type is modeled in RDF and Prolog in the Application generation routines.

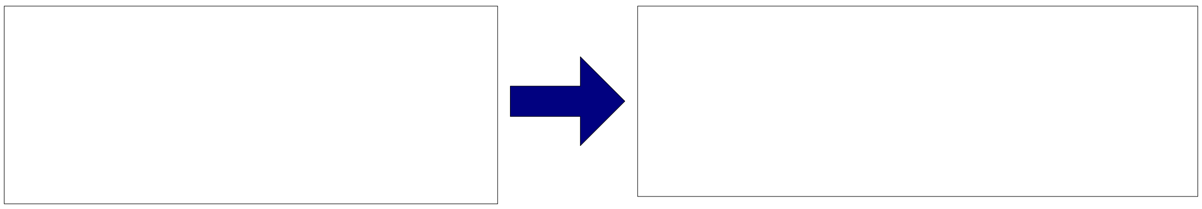


Fig 6.11: RDF Vs Prolog

The following code segment shows the high level invocation of the XSL/XSLT transformations and prolog file generation. The transformation is handled by a *Factory* class that uses the XSL/XSTL style sheet to carry out the transformation.

```
public void tranaformDoc(String application ){
 try{
   TransformerFactory tFactory = TransformerFactory.newInstance();

   // 1. RDF to ruleML
        Transformer transformer = tFactory.newTransformer(new
        StreamSource(RDFManager.xslbase + RDFManager.rdf2ruleml ));
        File in = new File(RDFManager.resouceBase + "application" + application + ".rdf");
       // transform
        transformer.transform(new StreamSource(in), new StreamResult(new
        FileOutputStream(RDFManager.ruleMLresouceBase + "application" + application +
        ".ruleml")));

  // 2. ruleml to rfml
      transformer = tFactory.newTransformer(new StreamSource(RDFManager.xslbase +
      RDFManager.ruleml2rfml  ));
      in = new File(RDFManager.ruleMLresouceBase + "application" + application + ".ruleml");
     // transform
```

```
      transformer.transform(new StreamSource(in), new StreamResult(new
      FileOutputStream(RDFManager.rfmlresouceBase + "application" + application + ".rfml")));

  // 3. rfml to prolog html

      transformer = tFactory.newTransformer(new StreamSource(RDFManager.xslbase +
      RDFManager.rfml2htmlprolog  ));
      in = new File(RDFManager.rfmlresouceBase + "application" + application + ".rfml");
     // transform
      transformer.transform(new StreamSource(in), new StreamResult(new
      FileOutputStream(RDFManager.prologHTMLresouceBase + "application" + application +
      ".html")));

  // 4. prologhtml to prolog
// special class for generating non markup based prolog from a Markup based representation
     prologGenerator gen = new prologGenerator();
      gen.parseHTMLtoProlog(application,RDFManager.prologHTMLresouceBase + "application" +
      application + ".html");
      ……………………………………………
```

6.9 Performance Enhancement

Since the Metadata modeler constitutes a significant code base and needs to be downloaded, the following performance enhancements are utilized.

The Metadata Application is loaded as a *Singleton* into the browser JAVA virtual Machine. Therefore the code base would only be downloaded to the browser once during a session and stored in the virtual machine sandbox. As and when user selects different applications to manage, the *AppletLoader* calls the *loadApplication* of the metadata application to invoke a HTTP get to extract the Domain Ontology from the Resource access layer running in a Web/JAVA Servlet Server. This would refresh the graphical interfaces and data structures with the data from the new application.

6.10 Extendibility of the framework

The architecture and design was done in a manner that would facilitate extension and adaptation. Several key decisions were taken in making the system modular and less coupling with the rest of the framework. This would facilitate in extending the Metadata manager to incorporate more functionality without worrying about the interdependencies and couplings.

1. **Access Delegate Framework** – An *Access delegate* design pattern was utilized to access the backend data for the GUI engine. The two access methods, HTTP and direct java object call were encapsulated behind the access Delegator, and exposed a well defined set of interface methods to load application properties and create application objects. This would ensure that any type of access mechanism can be incorporated with the same graphical engine without changing the interface and vice versa.
2. **Data Access object (DAO) framework** – The data access mechanisms for RDF , RuleML ,RFML and prolog were encapsulated with in separate *manager* classes that exposed APIs for accessing and modifying the file contests. This provides a file representation-independent mechanism for interfacing with the application files.
3. **Dynamic resource factory** – The metadata extraction module is dynamic in the sense that it would extract the required metadata in a data source dependent manner, yet encapsulate those differences to the user. The user only has to select the database source and login details and the table filtering strings. The underlying implementation would dynamically load the drivers and extract the metadata for the given database instance. This would be true for both relational sources as well as textual /XML based repositories.

7. Conclusion and Future work

I identified different contextual, ontological and temporal heterogeneities that exist in different financial messaging standards. I showed that indeed mediation between these is not a trivial task, yet a critical and important to the globalization of the financial industry. Further I show that the best answer is to have a mediation service that provides automatic and a transparent mediation without engineering new standards.

I have shown that the approach in COIN is capable of mediating the different heterogeneities that exist in different financial standards and internal contexts of Financial Institutions. My approach in modeling a business domain and mapping different contextual representations and values through a declarative manner proves extensibility, flexibility and the user-friendliness in the COIN framework. One aspect we haven't totally explored is how temporal conflicts are handled and mediates in COIN. Current work done [34] addresses how semantic conflicts are mediated in the presence of temporal heterogeneities. This could be applied in the analysis of temporal issues arising among different financial standards as well as among different versions.

We are looking at how to extend the current COIN technology in handling of aggregates (i.e. SUM, AVG, COUNT, etc.). In another aspect we are delving into understand more complex scenarios and issues faced by financial institutions in using different standards that would enable us to further extend the scope of the COIN framework and technology. Also the current abduction framework does not handle non-orthogonal modifier modeling. This refers to in prioritizing the order in which a set of modifiers that can be applied to a semantic type. Consider a small example of mediating a payment amount from USD to EURO. And in the destination context (EURO), an additional fixed fee of 10EURO need to be added to come up with the final payment amount. But if the conversion functions apply this addition to the USD amount and then perform the conversion, the result would be wrong. Current COIN framework cannot prioritize such modifier applicability for the moment. So this opens up another avenue for research.

Further, COIN mediation framework is based on a relational model. But financial standards uses hierarchical structures like XML. Thus in order to mediate between such sources an additional layer of translation/abstraction is needed that would map from the hierarchical structures to a relational model and vice versa. One aspect would be to extend a framework like HTTP Wrapper Cameleon [12] to support more generic and complex hierarchical structure representations. This would facilitate in utilizing the same COIN abduction engine for mediating relational sources as well as hierarchical sources/models.

While collecting and analyzing conflicts in standards, I was faced with great difficulties in finding the correct, appropriate examples and scenarios. Even after consulting with US and Singapore based institutions, companies and forums it was difficult to gather the required information. If this type of research to be fully successful a significant assistance is required for the industry.

In the Graphical Metadata Management frontier I managed to develop a System that provides a true user friendly, easy to learn graphical modeling tool to define the domain ontology as well the context hierarchies. The graphical modeling approach differs from previous efforts where by providing a relatively similar look and feel represented by modern modeling tools like Rational Rose and Microsoft Visio software. Further the user has been provided with a tool to log in and extract metadata related to different sources comprising relational databases, Text and XML files. These relive the user in looking at a particular schema and manually enter the source details. Further the application provides an option to generate the underlying application through the interface and deploy on the COIN abduction framework reducing several steps in configuration. I managed to extend the system to run as a stand alone application as well as a web based model providing the same features and look-and-feel.

Further work is carried out in order to facilitate the creation of Conversion functions for the underlying modifiers and elevations. This would make the Graphical Metadata manager a fully fledged interface for managing and creating applications for COIN, making it feasible to be used in commercial ventures.

References:

[1] A.Firat ."Information Integration using Contextual Knowledge and Ontology Merging", PhD Thesis, MIT,2003

[2] C.H. Goh, S.Bressan.S.Madnick,M.Siegel, "Context Interchange :New Features and Formalisms for the Intelligent Integration of Information", ACM TOIS, vol. 17,pp 270-293,1999.

[3] A.Bressan , C.H. Goh, "Answering Queries In Context", Proceedings of "Flexible Query Answering Systems". Third International Conference, FQAS, 1998, Roskild,Denmark.

[4] S.Madnick,A.Moulton,M.Siegel, "Semantic Interoperability in the Fixed Income Securities Industry: A Knowledge Representation Architecture for dynamic integration of Web-based information", HICSS,Hawai,2003

[5] S.Madnick, A.Firat, B.Grosof, "Knowledge Integration to overcome Ontological Heterogeneity: Challenges from Financial Information Systems",pp. 183-194,ICIS,Barcelona,Spain, 2002

[6] S.Madnick,A.Moulton,M.Siegel, "Context Interchange Mediation for Semantic Interoperability and Dynamic Integration of Autonomous Information Sources in the Fixed Income Securities Industry", (WITS), Barcelona, Spain, December 14-15, 2002, pp.61-66

[7] S.Madnick,S. Bressan, C.H. Goh, T. Lee, and M. Siegel "A Procedure for Mediation of Queries to Sources in Disparate Context", Proceedings of the International Logic Programming Symposium, October 1997

[8] S.Madnick, S. Bressan**,** C. Goh**,** N. Levina, A. Shah, M. Siegel ,"Context Knowledge Representation and Reasoning in the Context Interchange System" , *Applied Intelligence: The International Journal of Artificial Intelligence, Neutral Networks, and Complex Problem-Solving Technologies,* Vol 12, Number 2, September 2000, pp. 165-179

[9] Open Financial Exchange Specification OFX 2.0.2, Open Financial Exchange, http://www.ofx.net/ofx/de_spec.asp

[10]Interactive Financial Exchange –IFX version 1.5, IFX Forum, Inc, http://www.ifxforum.org/ifxforum.org/standards/standard.cfm

[11] Society for Worldwide Interbank Financial Telecommunication (S.W.I.F.T), Standard Release 2003, http://www.swift.com/index.cfm?item_id=5029

[12] S.Madnick, A. Firat and M. Siegel, "The Caméléon Web Wrapper Engine", *Proceedings of the VLDB2000 Workshop on Technologies for E-Services,* September 14-15, 2000

[13] S.Madnick, A. Moulton and M. Siegel "Semantic Interoperability in the Securities Industry: Context Interchange Mediation of Semantic Differences in Enumerated Data Types", *Proceedings of the Second International Workshop on Electronic Business Hubs: XML, Metadata, Ontologies, and Business Knowledge on the Web* (WEBH2002), Aix En Provence, France, September 6, 2002

[14] Kuhn, E., Puntigam, F., Elmagarmid A. (1991). Multidatabase Transaction and Query Processing in Logic, Database Transaction Models for Advanced Applications, Morgan Kaufmann Publishers.

[15] Litwin, W., Abdellatif, A. (1987), "An overview of the multi-database manipulation language MDSL". Proceedings of the IEEE, 75(5):621-632.

[16] Goh, C. H. (1997), "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems, MIT Ph.D. Thesis.

[17] Arens, Y., Knoblock, C., Shen, W. (1996). Query Reformulation for Dynamic Information Integration. Journal of Intelligent Information Systems 6(2/3): 99-130.

[18] Batini, C., Lenzerini, M., Navathe, S. B. (1986) "A Comparative Analysis of Methodologies for Database Schema Integration", ACM Computing Surveys 18(4): 323-364.

[19] Landers, T., Rosenberg, R (1982) "An Overview of MULTIBASE", International Symposium on Distributed Data Bases", 153-184

[20] Breitbart, Y., Tieman.L. (1984), "ADDS - Heterogeneous Distributed Database System", Proceedings of the Third International Seminar on Distributed Data Sharing Systems, 7- 24.

[21] Scheuermann, P., Elmagarmid, A. K., Garcia-Molina, H., Manola, F., McLeod, D.,Rosenthal, A., Templeton, M. (1990), "Report on the Workshop on Heterogeneous Database Systems" held at Northwestern University, Evanston, Illinois, December 11-13,

[22] Ahmed, R., De Smedt, P., Du, W., Kent, W., Ketabchi, M., Litwin, W.,, Rafii,A.,,Shan, M. (1991)." The Pegasus Heterogeneous Multidatabase System". IEEE Computer 24(12): 19-27.

[23] Collet, C., Huhns, M. N., Shen, W. (1991), "Resource Integration using a large knowledge base in Carnot", IEE Computer, 24(12):55-63.

[24] Kuhn, E., Ludwig, T. (1988), "VIP-MDBS: a logic multidatabase system", Proceedings of the first international symposium on Databases in parallel and distributed systems, p.190-201, December 05-07, Austin, Texas, USA.

[25] Litwin, W. (1992). "O*SQL: A language for object oriented multidatabase interoperability". In Proceedings of the Conference on IFIP WG2.6 Database Semantics and Interoperable Database Systems (DE-5) (Lorne, Victoria, Australia), D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, Eds. North-Holland Publishing Co., Amsterdam, The Netherlands, 119-138.

[26] Baral, C., Gelfond, M. (1994). "Logic Programming and Knowledge Representation", Journal of Logic Programming, 19,20:73-148.

[27] Kakas, A. C., Michael, A. (1995). "Integrating abductive and constraint logic programming", To appear in Proc. International Logic Programming Conference

[28] Graphical Metadata Management for the Context Mediation System - Usman Y. Mobin, Masters Thesis, Massachusetts Institute of Technology, 2002

[29] Metadata Representation and Management for Context Mediation – Philip Lee - Masters Thesis, Massachusetts Institute of Technology, 2003.

[30] Interoperability and Business Models for e-commerce, Man-Sze Li, IC Focus Ltd, 42 Clifton Road,

London N8 8JA, United Kingdom.
[31] Resource description Framework (RDF) - http://www.w3.org/RDF/
[32] The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformations, Harold Boley, Invited Talk, INAP2001, Tokyo, October 2001.
[33] Relational-Functional Markup Language (RFML) – Harold Boley, Markup Languages for functional logic programming, DFKI Gmbh
[34] Effective Data Integration in the Presence of Temporal Semantic Conflicts – working paper- Hongwei Zhu, Stuart E. Madnick, Michael D. Siegel- MIT Sloan School of Management.
[35] W3C OWL Web ontology language – Overview - http://www.w3.org/TR/2003/CR-owl-features-20030818

Acknowledgements

The author wishes to thank the following people whose help was immensely valuable for the completion of the research work.

- Professor Stuart Madnick, as the thesis supervisor – for his thoughtful and constructive advice on the direction, scope and guidance provided in carrying out the research as well as timely advice, and communication carried out with the author even with a significant time zone difference and virtual presence.
- Associate professor Tan, Kian Lee – as the NUS supervisor – for advising on the project requirements, guidelines and expectations and helping out on Singaporean side affairs related to project
- Dr. Stephane Bressan, for his advice, encouragement and help for carrying out the research and guidance provided in paper publishing and making contacts from local personal related to the research.
- Harry Hzu, Aykut Firat, Allen Moulton and Philip Lee, All from the context mediation group at Sloan School, MIT for providing help on COIN internals, existing systems as well for reviewing documents and presentations.

Appendix

Appendix A: Sample prolog file for EBPP Domain

```
:- module_interface(application604).
:- export rule/2.
:- begin_module(application604).
:- dynamic rule/2.
rule(is_a(accountCode,basic),(true)).
rule(is_a(accountCodeScheme,basic),(true)).
rule(is_a(currency,basic),(true)).
rule(is_a(moneyAmount,basic),(true)).
rule(is_a(identifier,basic),(true)).
rule(is_a(personname,identifier),(true)).
rule(is_a(bankname,identifier),(true)).
rule(is_a(location,basic),(true)).
rule(is_a(bankLoc,basic),(true)).
rule(is_a(branchLoc,location),(true)).
rule(is_a(customer,basic),(true)).
rule(is_a(payee,basic),(true)).
rule(is_a(phoneNumber,basic),(true)).
%rule(is_a(numberScheme,basic),(true)).
rule(is_a(phoneNumberScheme,basic),(true)).
rule(is_a(account,basic),(true)).
rule(is_a(date,basic),(true)).
rule(is_a(dateFormat,basic),(true)).
rule(is_a(payment,basic),(true)).
rule(is_a(paymentAmt,basic),(true)).
rule(is_a(dateStyle,basic),(true)).
rule(is_a(exchangeRate, basic), (true)).
rule(is_a(paymentScheme, basic), (true)).
rule(is_a(countryName,basic), (true)).
rule(is_a(invoiceAmount,basic), (true)).

rule(attributes(basic,[]),(true)).
rule(attributes(accountCode,[]),(true)).
rule(attributes(accountCodeScheme,[]),(true)).
rule(attributes(currency,[]),(true)).
rule(attributes(moneyAmount,[]),(true)).
rule(attributes(identifier,[]),(true)).
rule(attributes(personname,[]),(true)).
rule(attributes(bankname,[]),(true)).
rule(attributes(location,[locName]),(true)).
rule(attributes(bankLoc,[bankName,countryIncorporated]),(true)).
rule(attributes(branchLoc,[bank,branchName]),(true)).
rule(attributes(customer,[name,phone]),(true)).
rule(attributes(payee,[name,phone]),(true)).
rule(attributes(phoneNumber,[]),(true)).
rule(attributes(phoneNumberScheme,[]),(true)).
rule(attributes(account,[type,customer,balance,location]),(true)).
rule(attributes(date,[]),(true)).
rule(attributes(dateFormat,[]),(true)).
rule(attributes(payment,[fromAccount,payee,payeeAct,dueDate]),(true)).
rule(attributes(paymentAmt,[paymentRef]),(true)).
rule(attributes(dateStyle,[]),(true)).
rule(attributes(exchangeRate, [txnDate, fromCur, toCur]), (true)).
rule(attributes(paymentScheme,[]),(true)).
rule(attributes(countryName,[officialCurrency]),(true)).
rule(attributes(invoiceAmount,[invoiceRef]),(true)).
rule(modifiers(basic,[]),(true)).
rule(modifiers(accountCode,[scheme]),(true)).
rule(modifiers(accountCodeScheme,[]),(true)).
rule(modifiers(currency,[]),(true)).
%rule(modifiers(moneyAmount,[]),(true)).
rule(modifiers(identifier,[]),(true)).
rule(modifiers(personname,[]),(true)).
rule(modifiers(bankname,[]),(true)).
rule(modifiers(location,[locationType]),(true)).
rule(modifiers(bankLoc,[idType]),(true)).
rule(modifiers(branchLoc,[]),(true)).
rule(modifiers(customer,[]),(true)).
rule(modifiers(payee,[]),(true)).
rule(modifiers(phoneNumber,[numberScheme]),(true)).
%rule(modifiers(numberScheme,[]),(true)).
rule(modifiers(phoneNumberScheme,[]),(true)).
rule(modifiers(account,[]),(true)).
rule(modifiers(date,[format,style]),(true)).
rule(modifiers(dateFormat,[]),(true)).
rule(modifiers(payment,[]),(true)).
rule(modifiers(paymentAmt,[paymentScheme,currency,includeBankCharges]),(true)).
rule(modifiers(dateStyle,[]),(true)).
rule(modifiers(exchangeRate, []), (true)).
rule(modifiers(paymentScheme,[]),(true)).
rule(modifiers(countryName,[]),(true)).
rule(modifiers(invoiceAmount,[invoiceScheme]),(true)).

rule(relation(oracle,'INTERNAL_PAYEE',ie,[['PAYEE_ID',number],['NAME',string],['PHONE',string]],
cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'INTERNAL_CUSTOMER',ie,[['CUST_ID',number],['NAME',string],['PHONE',strin
g]],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'INTERNAL_BANK',ie,[['BANK_NAME',string],['COUNTRY',string],['BANK_ID',
number]],cap([[0,0,1]],[])),(true)).
rule(relation(oracle,'INTERNAL_BANK_BRANCH',ie,[['BRANCH_ID',number],['BANK_ID',number],[
'BRANCH_NAME',string]],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'INTERNAL_ACCOUNT_TYPE',ie,[['ACCOUNT_CODE',string],['DESCRIPTION',
string]],cap([[1,0]],[])),(true)).
rule(relation(oracle,'INTERNAL_ACCOUNT_DETAILS',ie,[['ACCOUNT_ID',string],['CUSTOMER_I
D',number],['BRANCH_BANK_ID',number],['ACCOUNT_TYPE',string],['NET_BALANCE',number]],
cap([[1,0,0,0,0]],[])),(true)).
rule(relation(oracle,'INTERNAL_PAYMENT',ie,[['PAYMENT_ID',number],['FROM_ACCT_ID',numbe
r],['AMOUNT_NET',number],['GST_PERCENT',number],['OTHER_TAX_PERCENT',number],['PAYE
E_ID',number],['PAYEE_ACCOUNT',number],['DUE_DATE',string]],cap([[1,0,0,0,0,0,0,0]],[])),(true)).
rule(relation(oracle,'OFX_PAYEE',ie,[['PAYEE_ID',number],['NAME',string],['PHONE',string]],cap([[1,
0,0]],[])),(true)).
rule(relation(oracle,'OFX_CUSTOMER',ie,[['CUST_ID',number],['NAME',string],['PHONE',string]],cap(
[[1,0,0]],[])),(true)).
rule(relation(oracle,'OFX_BANK',ie,[['BANK_NAME',string],['BANK_ID',number],['COUNTRY',string
]],cap([[0,1,0]],[])),(true)).
rule(relation(oracle,'OFX_ACCOUNT_TYPE',ie,[['ACCOUNT_CODE',string],['DESCRIPTION',string]]
,cap([[1,0]],[])),(true)).
rule(relation(oracle,'OFX_ACCOUNT_DETAILS',ie,[['ACCOUNT_ID',string],['CUSTOMER_ID',numb
er],['ACCOUNT_TYPE',string],['BALANCE',number],['BRANCH_BANK_ID',number]],cap([[1,0,0,0,0]
],[])),(true)).
rule(relation(oracle,'OFX_PAYMENT',ie,[['PAYMENT_ID
',number],['FROM_ACCT_ID',string],['AMOUNT_NET',number],['GST_PERCENT',number],['OTHER
_TAX_PERCENT',number],['PAYEE_ID',number],['PAYEE_ACCOUNT',string],['DUE_DATE',string]]
,cap([[1,0,0,0,0,0,0,0]],[])),(true)).
 ([[1,0]],[])),(true)).
```

```
rule(relation(oracle,'CODE_MAP_INTERNAL_OFX',ie,[['INTERNAL_CODE',string],['OFX_CODE',str
ing]],cap([[0,0]],[])),(true)).
rule(relation(oracle,'ONTOLOGY_ACCOUNT_TYPES',ie,[['ONTO_CODE',string],['DESCR',string]],ca
p([[1,0]],[])),(true)).
rule(relation(oracle,'CODE_MAP_ONTOLOGY_INTERNAL',ie,[['INTERNAL_CODE',string],['ONTO
LOGY_CODE',string]],cap([[1,1]],[])),(true)).
rule(relation(oracle,'CODE_MAP_ONTOLOGY_OFX',ie,[['OFX_CODE',string],['ONTOLOGY_CODE',
string]],cap([[1,1]],[])),(true)).
rule(relation(oracle,'COUNTRY_CURRENCY',ie,[['COUNTRY',string],['CURRENCY',string]],cap([[0,0
]],[])),(true)).
rule(relation(oracle,'OFX_BANK_BRANCH',ie,[['BRANCH_ID',number],['BANK_ID',number],['BRAN
CH_NAME',string]],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'OFX_TAX_TYPES',ie,[['TAX_NAME',string],['TYPE',string],['AMOUNT',number]
],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'SWIFT_BANK_BCI',ie,[['BANK_ID',number],['LOCAITON_CODE',string],['COU
NTRY_CODE',string]],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'SWIFT_CHARGE_TYPES',ie,[['TAX_NAME',string],['TYPE',string],['AMOUNT',n
umber]],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'IFX_TAX_TYPES',ie,[['TAX_NAME',string],['TYPE',string],['AMOUNT',number]],
cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'IFX_FEES_TYPES',ie,[['FEES_NAME',string],['TYPE',string],['AMOUNT',number]
],cap([[1,0,0]],[])),(true)).
rule(relation(oracle,'IFX_INVOICE_FEES',ie,[['FEE_NAME',string],['INVOICE_NO',number]],cap([[1,1
]],[])),(true)).
rule(relation(oracle,'IFX_INVOICE_TAXES',ie,[['TAX_NAME',string],['INVOICE_NO',number]],cap([[
1,1]],[])),(true)).
rule(relation(oracle,'INTERNAL_INVOICE',ie,[['INVOICE_NO',string],['PAYMENT_ID',string],['INVO
ICE_AMOUNT',number],['DESCR',string],['INVOICE_DATE',string],['DISCOUNT_RATE',number],['
DISCOUNT_DESC',string]],cap([[1,0,0,0,0,0,0]],[])),(true)).


rule(relation(cameleon,
          olsen,
          ie,
          [['Exchanged', string],
           ['Expressed', string],
           ['Rate', real],
           ['Date', string]],
           cap([[0, 0, 0, 0]],
              [])), (true)).

%% elevations
rule('INTERNAL_PAYEE_p'(skolem(payee,_PAYEE_ID,internal,1,'INTERNAL_PAYEE'(_PAYEE_ID
,_NAME,_PHONE)),skolem(personname,_NAME,internal,2,'INTERNAL_PAYEE'(_PAYEE_ID,_NAM
E,_PHONE)),skolem(phoneNumber,_PHONE,internal,3,'INTERNAL_PAYEE'(_PAYEE_ID,_NAME,_
PHONE))),('INTERNAL_PAYEE'(_PAYEE_ID,_NAME,_PHONE))).
rule('INTERNAL_CUSTOMER_p'(skolem(customer,_CUST_ID,internal,1,'INTERNAL_CUSTOMER'(
_CUST_ID,_NAME,_PHONE)),skolem(personname,_NAME,internal,2,'INTERNAL_CUSTOMER'(_C
UST_ID,_NAME,_PHONE)),skolem(phoneNumber,_PHONE,internal,3,'INTERNAL_CUSTOMER'(_C
UST_ID,_NAME,_PHONE))),('INTERNAL_CUSTOMER'(_CUST_ID,_NAME,_PHONE))).
rule('INTERNAL_BANK_p'(skolem(basic,_BANK_NAME,internal,1,'INTERNAL_BANK'(_BANK_N
AME,_COUNTRY,_BANK_ID)),skolem(countryName,_COUNTRY,internal,2,'INTERNAL_BANK'(_
BANK_NAME,_COUNTRY,_BANK_ID)),skolem(bankLoc,_BANK_ID,internal,3,'INTERNAL_BAN
K'(_BANK_NAME,_COUNTRY,_BANK_ID))),('INTERNAL_BANK'(_BANK_NAME,_COUNTRY,_
BANK_ID))).
rule('INTERNAL_BANK_BRANCH_p'(skolem(branchLoc,_BRANCH_ID,internal,1,'INTERNAL_BA
NK_BRANCH'(_BRANCH_ID,_BANK_ID,_BRANCH_NAME)),skolem(bankLoc,_BANK_ID,internal
,2,'INTERNAL_BANK_BRANCH'(_BRANCH_ID,_BANK_ID,_BRANCH_NAME)),skolem(basic,_B
RANCH_NAME,internal,3,'INTERNAL_BANK_BRANCH'(_BRANCH_ID,_BANK_ID,_BRANCH_N
AME))),('INTERNAL_BANK_BRANCH'(_BRANCH_ID,_BANK_ID,_BRANCH_NAME))).
```

```
rule(is_a(internal,basic),(true)).
rule(is_a(ofx,basic),(true)).
rule(is_a(olsen_context, basic), (true)).
rule(is_a(ifx, basic), (true)).
rule(is_a(none, basic), (true)).
rule(is_a(swift,basic),(true)).
rule(is_a(swift_intraEU,swift),(true)).
rule(is_a(swift_outsideEU,swift),(true)).
rule(contexts([internal,ofx,olsen_context,ifx,none,swift,swift_intraEU,swift_outsideEU]),(true)).
rule(context(internal), (true)).
rule(context(ofx), (true)).
rule(context(olsen_context), (true)).
rule(context(ifx), (true)).
rule(context(none), (true)).
rule(context(swift), (true)).
rule(context(swift_intraEU), (true)).
rule(context(swift_outsideEU), (true)).

%% modifiers
%% for payment mode description
rule(modifier(paymentAmt,_O,paymentScheme,internal,M),(cste(basic,M,internal,"notax"))).
rule(modifier(paymentAmt,_O,paymentScheme,ofx,M),(cste(basic,M,ofx,"withtax"))).
rule(modifier(paymentAmt,_O,paymentScheme,swift,M),(cste(basic,M,swift,"withtax"))).
rule(modifier(paymentAmt,_O,paymentScheme,swift_intraEU,M),(cste(basic,M,swift_intraEU,"withtax")
)).
rule(modifier(paymentAmt,_O,paymentScheme,swift_outsideEU,M),(cste(basic,M,swift_outsideEU,"wit
htax"))).

%% includeBankCharges
rule(modifier(paymentAmt,_O,includeBankCharges,internal,M),(cste(basic,M,internal,"no"))).
rule(modifier(paymentAmt,_O,includeBankCharges,ofx,M),(cste(basic,M,ofx,"no"))).
rule(modifier(paymentAmt,_O,includeBankCharges,swift,M),(cste(basic,M,swift,"no"))).
rule(modifier(paymentAmt,_O,includeBankCharges,swift_intraEU,M),(cste(basic,M,swift_intraEU,"no")
)).
rule(modifier(paymentAmt,_O,includeBankCharges,swift_outsideEU,M),(cste(basic,M,swift_outsideEU,
"yes"))).

%% for currency type.for the moment hard coded to EUR and US$
rule(modifier(paymentAmt,_O,currency,internal,M),(cste(currency,M,internal,"GBP"))).

rule(modifier(paymentAmt,_O,currency,ofx,M),( attr(_O,paymentRef,Payment),attr(Payment,payeeAct,A
ccount),
attr(Account,location,Location),attr(Location,bank,Bank),attr(Bank,countryIncorporated,Country),
attr(Country,officialCurrency,M))).

%%for phone number scheme
rule(modifier(phoneNumber,_O,numberScheme,internal,M),(cste(phoneNumberScheme,M,internal,"DO
TSCHEME"))).
rule(modifier(phoneNumber,_O,numberScheme,ofx,M),(cste(phoneNumberScheme,M,ofx,"HYPENSCH
EME"))).

%% modifier for account code
rule(modifier(accountCode,_O,scheme,internal,M),(cste(accountCodeScheme,M,internal,"internal"))).
rule(modifier(accountCode,_O,scheme,ofx,M),(cste(accountCodeScheme,M,ofx,"ofx"))).
rule(modifier(accountCode,_O,scheme,ifx,M),(cste(accountCodeScheme,M,ifx,"ifx"))).

%% date format i.e US and UK
rule(modifier(date,_O,format,internal,M),(cste(dateFormat,M,internal,"US"))).
rule(modifier(date,_O,format,ofx,M),(cste(dateFormat,M,ofx,"UK"))).
```

```
%% date style
rule(modifier(date,_O,style,internal,M),(cste(dateStyle,M,internal,"dashUK"))).
rule(modifier(date,_O,style,ofx,M),(cste(dateStyle,M,ofx,"nodashUS"))).

%% the account location.
%rule(modifier(location,_O,locationType,internal,M),(cste(basic,M,internal,"branch"))).
%rule(modifier(location,_O,locationType,ofx,M),(cste(basic,M,ofx,"bank"))).

rule(modifier(bankLoc,_O,idType,internal,M),(cste(basic,M,internal,"single"))).
rule(modifier(bankLoc,_O,idType,ofx,M),(cste(basic,M,ofx,"composite"))).
rule(modifier(bankLoc,_O,idType,swift,M),(cste(basic,M,swift,"composite"))).

% invoice scheme
rule(modifier(invoiceAmount,_O,invoiceScheme,internal,M),(cste(basic,M,internal,"nominal"))).
rule(modifier(invoiceAmount,_O,invoiceScheme,ofx,M),(cste(basic,M,ofx,"nominaltaxfees"))).
rule(modifier(invoiceAmount,_O,invoiceScheme,ifx,M),(cste(basic,M,ifx,"nominaltaxfees"))).

%% the attr definitions. ---------------------------------------------------------
%% for exchange rate semantic type.
rule(attr(X, txnDate, Y), (olsen_p(_6313, _6314, X, Y))).
rule(attr(X, fromCur, Y), (olsen_p(_6351, Y, X, _6354))).
rule(attr(X, toCur, Y), (olsen_p(Y, _6390, X, _6392))).

%% attribute for payment and payment amount -----
%rule(attr(PaymentAmt,paymentRef,Payment),('INTERNAL_PAYMENT_p'(Payment,_,PaymentAmt,_,
_,_,_,_))).
%rule(attr(PaymentAmt,paymentRef,Payment),('OFX_PAYMENT_p'(Payment,_,PaymentAmt,_,_,_,_,_)
)).

%attrib for bank(location) and name relation - internal and ofx.--------------
rule(attr(BankLoc,bankName,BankName),('INTERNAL_BANK_p'(BankName,_,BankLoc))).
rule(attr(BankLoc,bankName,BankName),('OFX_BANK_p'(BankName,BankLoc,_))).

%attrib for branch and name-----------.
rule(attr(BranchLoc,branchName,BranchName),('INTERNAL_BANK_BRANCH_p'(BranchLoc,_,Branc
hName))).
rule(attr(BranchLoc,branchName,BranchName),('OFX_BANK_BRANCH_p'(BranchLoc,_,BranchName)
)).

%%atrib for branch , bank relation.-----------------
rule(attr(BranchLoc,bank,BankLoc),('INTERNAL_BANK_BRANCH_p'(BranchLoc,BankLoc,_))).
rule(attr(BranchLoc,bank,BankLoc),('OFX_BANK_BRANCH_p'(BranchLoc,BankLoc,_))).

%%attr for location for account -- for internal and OFX--------------
rule(attr(Account,location,Branch),('INTERNAL_ACCOUNT_DETAILS_p'(Account,_,Branch,_,_))).
rule(attr(Account,location,Branch),('OFX_ACCOUNT_DETAILS_p'(Account,_,_,_,Branch))).

%% payee account relationship
rule(attr(Payment,payeeAct,PayeeAcct),('INTERNAL_PAYMENT_p'(Payment,_,_,_,_,_,PayeeAcct,_))).
rule(attr(Payment,payeeAct,PayeeAcct),('OFX_PAYMENT_p'(Payment,_,_,_,_,_,PayeeAcct,_))).

%%  country of bank
rule(attr(Bank,countryIncorporated,Country),('INTERNAL_BANK_p'(_,Country,Bank))).
rule(attr(Bank,countryIncorporated,Country),('OFX_BANK_p'(_,Bank,Country))).
%% currency of country

rule(attr(Country,officialCurrency,Currency),('COUNTRY_CURRENCY_p'(X,Currency),'INTERNAL_
BANK_p'(_,Country,_), value(Country,internal,U),value(X,internal,U))).
rule(attr(Country,officialCurrency,Currency),('COUNTRY_CURRENCY_p'(X,Currency),'OFX_BANK_
p'(_,_,Country), value(Country,ofx,U),value(X,ofx,U))).
```

```
%rule(attr(Country,officialCurrency,Currency),('INTERNAL_COUNTRY_CURRENCY_p'(Country,Cur
rency))).
%rule(attr(Country,officialCurrency,Currency),('OFX_COUNTRY_CURRENCY_p'(Country,Currency))
).

%% invoice ref of invoice amount
rule(attr(InvAmt,invoiceRef,Invoice),('INTERNAL_INVOICE_p'(Invoice,_,InvAmt,_,_,_,_))).

%% for bank charge inclusion/exclusion
rule(cvt(noncommutative,paymentAmt,_O,includeBankCharges,Ctxt,"no",Vs,"yes",Vt),
    (value(FeeName,Ctxt,"outsideEU"),'SWIFT_CHARGE_TYPES_p'(FeeName,_,Rate),
    value(Rate,Ctxt,RR),
     (Vt is RR + Vs))).

%% for payment amount i,e tax and without tax
rule(cvt(noncommutative,paymentAmt,_O,paymentScheme,Ctxt,"notax",Vs,"withtax",Vt),
    (value(TaxName,Ctxt,"GST"),'OFX_TAX_TYPES_p'(TaxName,_,Rate),
    value(Rate,Ctxt,RR),
     (Vtemp is RR * Vs),
     (Vt is Vs + Vtemp))).

%conversion for invoice scheme for addition of fees and invices
%%% WORKING !!!!
rule(cvt(noncommutative,invoiceAmount,_O,invoiceScheme,Ctxt,"nominal",Vs,"nominaltaxfees",Vt),
    (attr(_O,invoiceRef,INVOIC_ID),value(INVOIC_ID,Ctxt,ID),
     value(TaxName1,Ctxt,"GST"),
'IFX_INVOICE_TAXES_p'(TaxName2,ID2), value(TaxName2,Ctxt,"GST"),
    value(ID2,Ctxt,ID),
     'IFX_TAX_TYPES_p'(TaxName1,_,TRate1),value(TRate1,Ctxt,TR1),
     (VtT1 is   Vs * TR1),
     % second tax -----------------
     value(TaxName3,Ctxt,"IMPORT"),
     'IFX_INVOICE_TAXES_p'(TaxName4,ID3),
     value(TaxName4,Ctxt,"IMPORT"),value(ID3,Ctxt,ID),
    'IFX_TAX_TYPES_p'(TaxName3,_,TRate2),value(TRate2,Ctxt,TR2),
     (VtT2 is   Vs * VtT2),
     (TOT_TAX is VtT1 + TR2),
     % now the fees -----------
      value(FeeName1,Ctxt,"LATE"),'IFX_INVOICE_FEES_p'(FeeName2,ID4),
      value(FeeName2,Ctxt,"LATE"),value(ID4,Ctxt,ID),
'IFX_FEES_TYPES_p'(FeeName1,_,FRate1),value(FRate1,Ctxt,FR1),
    (VtF1 is   Vs + FR1),
% second fee
    value(FeeName3,Ctxt,"DELIVERY"),'IFX_INVOICE_FEES_p'(FeeName4,ID5),
    value(FeeName4,Ctxt,"DELIVERY"),value(ID5,Ctxt,ID),
'IFX_FEES_TYPES_p'(FeeName3,_,FRate2),value(FRate2,Ctxt,FR2),
   (VtF2 is   Vs * FR2),  (TOT_FEES is VtF1 + VtF2),   (Vtemp is TOT_TAX + TOT_FEES),
 (Vt is Vs + Vtemp))).

rule(cvt(commutative,accountCode,_O,scheme,Ctxt,Mvs,Vs,Mvt,Vt), (code_map(Vs,Mvs,Mvt,Vt,Ctxt))).

%% ofx and internal
rule(code_map(Val,"internal","ofx",V,Ctxt), ( 'CODE_MAP_ONTOLOGY_INTERNAL'(Val,ONTO),
                              'ONTOLOGY_ACCOUNT_TYPES'(ONTO,_),
                              'CODE_MAP_ONTOLOGY_OFX'(V,ONTO))).

rule(code_map(Val,"ofx","internal",V,Ctxt), ( 'CODE_MAP_ONTOLOGY_OFX'(Val,ONTO),
                              'ONTOLOGY_ACCOUNT_TYPES'(ONTO,_),
                              'CODE_MAP_ONTOLOGY_INTERNAL'(V,ONTO))).
```

```
%% rule for currency conversion-----.
rule(cvt(commutative,paymentAmt,O,currency,Ctxt,Mvs,Vs,Mvt,Vt),
(olsen_p(Fc, Tc, Rate, Date),
        value(Fc, Ctxt, Mvs),
        value(Tc, Ctxt, Mvt),
        value(Rate, Ctxt, Rv),
        currentDate_p(CurDate),
        value(CurDate, Ctxt, DateValue),
        value(Date, Ctxt, DateValue),
        multiply(Vs, Rv, Vt))).


%% supplementary rules to get the date from system and match Exchange rate with olsen's date format.

rule(currentDate(Date), ({date(D), substring(D, 5, 3, Month), substring(D, 9, 2, Day), substring(D, 23, 2,
Year)}, month(Month, NumMonth), {concat_string([NumMonth, /, Day, /, Year], Date)})).
rule(month("Jan", 01), (true)).
rule(month("Feb", 02), (true)).
rule(month("Mar", 03), (true)).
rule(month("Apr", 04), (true)).
rule(month("May", 05), (true)).
rule(month("Jun", 06), (true)).
rule(month("Jul", 07), (true)).
rule(month("Aug", 08), (true)).
rule(month("Sep", 09), (true)).
rule(month("Oct", 10), (true)).
rule(month("Nov", 11), (true)).
rule(month("Dec", 12), (true)).

rule(currentDate_p(
        skolem(date, V, Ctxt, 1, currentDate(V))),
        (currentDate(V))).


rule(cvt(date,O,style,Ctxt,"dashUK",Vs,"nodashUS",Vt),
(substring(Vs,1,2,Date),substring(Vs,4,2,Month),substring(Vs,6,4,Year),
concat_string([Year,Month,Date],Vt))).

% other way
rule(cvt(date,O,style,Ctxt,"nodashUS",Vs,"dashUK",Vt),
(substring(Vs,1,4,Year),substring(Vs,5,2,Month),substring(Vs,7,2,Date),
concat_string([Date, / , Month, / , Year],Vt))).


% ---------------- BIC conversion -----------
%conversion for composite (BIC ) and simple bank ID
rule(cvt(noncommutative,bankLoc,O,idType,Ctxt,"single",Vs,"composite",Vt),
        ('SWIFT_BANK_BCI_p'(BANK,LOC,COUNTRY),
         value(BANK,Ctxt,Vs),
         value(LOC,Ctxt,Locc),
         value(COUNTRY,Ctxt,Countryc),
   (Vtemp is Vs + Locc),
   (Vt is Vtemp + Countryc))).

%----------- end BIC conversion ----------------
```

Appendix B: Sample RDF file for EBPP generated automatically from COIN Metadata Manager

```
<?xml version="1.0" encoding="UTF-8" ?>
 <rdf:RDF xmlns:coin="http://localhost/appEditor/apps/rdf/coin_schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <coin:Ont_SemanticType rdf:ID="dateStyle">
  <coin:Ont_SemanticTypeName>dateStyle</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="currency">
  <coin:Ont_SemanticTypeName>currency</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="branchLoc">
  <coin:Ont_SemanticTypeName>branchLoc</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#location" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="date">
  <coin:Ont_SemanticTypeName>date</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="invoiceAmount">
  <coin:Ont_SemanticTypeName>invoiceAmount</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="location">
  <coin:Ont_SemanticTypeName>location</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="bankLoc">
  <coin:Ont_SemanticTypeName>bankLoc</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#location" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="payee">
  <coin:Ont_SemanticTypeName>payee</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#personName" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="personName">
  <coin:Ont_SemanticTypeName>personName</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
</coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="accountCode">
  <coin:Ont_SemanticTypeName>accountCode</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="basic">
  <coin:Ont_SemanticTypeName>basic</coin:Ont_SemanticTypeName>
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="accountCodeScheme">
  <coin:Ont_SemanticTypeName>accountCodeScheme</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
 <coin:Ont_SemanticType rdf:ID="account">
  <coin:Ont_SemanticTypeName>account</coin:Ont_SemanticTypeName>
  <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
  </coin:Ont_SemanticType>
```

```
<coin:Ont_SemanticType rdf:ID="paymentScheme">
 <coin:Ont_SemanticTypeName>paymentScheme</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="dateFormat">
 <coin:Ont_SemanticTypeName>dateFormat</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="includeBankCharge">
 <coin:Ont_SemanticTypeName>includeBankCharge</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="customer">
 <coin:Ont_SemanticTypeName>customer</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#personName" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="moneyAmount">
 <coin:Ont_SemanticTypeName>moneyAmount</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="invoice">
 <coin:Ont_SemanticTypeName>invoice</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="payment">
 <coin:Ont_SemanticTypeName>payment</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="country">
 <coin:Ont_SemanticTypeName>country</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="paymentAmount">
 <coin:Ont_SemanticTypeName>paymentAmount</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="phoneNumberScheme">
 <coin:Ont_SemanticTypeName>phoneNumberScheme</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="invoiceScheme">
 <coin:Ont_SemanticTypeName>invoiceScheme</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_SemanticType rdf:ID="phoneNumber">
 <coin:Ont_SemanticTypeName>phoneNumber</coin:Ont_SemanticTypeName>
 <coin:Ont_SemanticTypeParent rdf:resource="#basic" />
 </coin:Ont_SemanticType>
<coin:Ont_Attribute rdf:ID="bank">
 <coin:Ont_AttributeName>bank</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#branchLoc" />
 <coin:Ont_AttributeTo rdf:resource="#bankLoc" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="incorporation">
 <coin:Ont_AttributeName>incorporation</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#bankLoc" />
 <coin:Ont_AttributeTo rdf:resource="#country" />
 </coin:Ont_Attribute>
```

```
<coin:Ont_Attribute rdf:ID="code">
 <coin:Ont_AttributeName>code</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#account" />
 <coin:Ont_AttributeTo rdf:resource="#accountCode" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="payee">
 <coin:Ont_AttributeName>payee</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#account" />
 <coin:Ont_AttributeTo rdf:resource="#payee" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="loc">
 <coin:Ont_AttributeName>loc</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#account" />
 <coin:Ont_AttributeTo rdf:resource="#branchLoc" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="phone">
 <coin:Ont_AttributeName>phone</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#customer" />
 <coin:Ont_AttributeTo rdf:resource="#phoneNumber" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="amount">
 <coin:Ont_AttributeName>amount</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#invoice" />
 <coin:Ont_AttributeTo rdf:resource="#invoiceAmount" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="paymentAmount">
 <coin:Ont_AttributeName>paymentAmount</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#payment" />
 <coin:Ont_AttributeTo rdf:resource="#paymentAmount" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="payeeAcount">
 <coin:Ont_AttributeName>payeeAcount</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#payment" />
 <coin:Ont_AttributeTo rdf:resource="#account" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="currency">
 <coin:Ont_AttributeName>currency</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#country" />
 <coin:Ont_AttributeTo rdf:resource="#basic" />
 </coin:Ont_Attribute>
<coin:Ont_Attribute rdf:ID="payRef">
 <coin:Ont_AttributeName>payRef</coin:Ont_AttributeName>
 <coin:Ont_AttributeFrom rdf:resource="#paymentAmount" />
 <coin:Ont_AttributeTo rdf:resource="#payment" />
 </coin:Ont_Attribute>
<coin:Ont_Modifier rdf:ID="style">
 <coin:Ont_ModifierName>style</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#date" />
 <coin:Ont_ModifierTo rdf:resource="#dateStyle" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="format">
 <coin:Ont_ModifierName>format</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#date" />
 <coin:Ont_ModifierTo rdf:resource="#dateFormat" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="scheme">
 <coin:Ont_ModifierName>scheme</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#invoiceAmount" />
 <coin:Ont_ModifierTo rdf:resource="#invoiceScheme" />
 </coin:Ont_Modifier>
```

```
<coin:Ont_Modifier rdf:ID="code">
 <coin:Ont_ModifierName>code</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#accountCode" />
 <coin:Ont_ModifierTo rdf:resource="#accountCodeScheme" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="<name>">
 <coin:Ont_ModifierName><name></coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#moneyAmount" />
 <coin:Ont_ModifierTo rdf:resource="#currency" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="payScheme">
 <coin:Ont_ModifierName>payScheme</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#paymentAmount" />
 <coin:Ont_ModifierTo rdf:resource="#paymentScheme" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="currency">
 <coin:Ont_ModifierName>currency</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#paymentAmount" />
 <coin:Ont_ModifierTo rdf:resource="#currency" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="bankChargeScheme">
 <coin:Ont_ModifierName>bankChargeScheme</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#paymentAmount" />
 <coin:Ont_ModifierTo rdf:resource="#includeBankCharge" />
 </coin:Ont_Modifier>
<coin:Ont_Modifier rdf:ID="scheme">
 <coin:Ont_ModifierName>scheme</coin:Ont_ModifierName>
 <coin:Ont_ModifierFrom rdf:resource="#phoneNumber" />
 <coin:Ont_ModifierTo rdf:resource="#phoneNumberScheme" />
 </coin:Ont_Modifier>
```

Appendix C: Dynamic modifier representation for RDF and Prolog

RDF

```
 <coin:Ont_DynamicModifierValue rdf:ID="currency">
  <coin:Ont_ModifierName>currency</coin:Ont_ModifierName>
  <coin:Ont_ModifierFrom rdf:resource="#paymentAmount" />
  <coin:Ont_DynamicModifierContext rdf:resource="#OFX" />
  <coin:Ont_ModifierResolveSteps >
        <coin:Ont_ModifierResolveStep >
        <coin:Ont_StepAttributeDomain rdf:resource="#paymentAmount" />
                <coin:Ont_StepAttributeName rdf:resource="#paymentRef" />
        </coin:Ont_ModifierResolveStep >
        <coin:Ont_ModifierResolveStep >
                <coin:Ont_StepAttributeDomain rdf:resource="#payment" />
                <coin:Ont_StepAttributeName rdf:resource="#payeeAct" />
        </coin:Ont_ModifierResolveStep >
        <coin:Ont_ModifierResolveStep >
                <coin:Ont_StepAttributeDomain rdf:resource="#account" />
                <coin:Ont_StepAttributeName rdf:resource="#location" />
        </coin:Ont_ModifierResolveStep >
        <coin:Ont_ModifierResolveStep >
                <coin:Ont_StepAttributeDomain rdf:resource="#branchLoc" />
                <coin:Ont_StepAttributeName rdf:resource="#banklLoc" />
        </coin:Ont_ModifierResolveStep >
        <coin:Ont_ModifierResolveStep >
                <coin:Ont_StepAttributeDomain rdf:resource="#bank" />
                <coin:Ont_StepAttributeName rdf:resource="#incorpporatedCountry" /
        </coin:Ont_ModifierResolveStep >
        <coin:Ont_ModifierResolveStep >
                <coin:Ont_StepAttributeDomain rdf:resource="#country" />
                <coin:Ont_StepAttributeName rdf:resource="#officialCurrency" />
        </coin:Ont_ModifierResolveStep >
  </coin:Ont_ModifierResolveSteps >
 </coin:Ont_DynamicModifierValue>
```

PROLOG

```
rule(modifier(paymentAmt,_O,currency,ofx,M),( attr(_O,paymentRef,Payment),attr(Payment,pa
yeeAct,Account),
                        attr(Account,location,Location),
                        attr(Location,bank,Bank),
                        attr(Bank,countryIncorporated,Country),
                        attr(Country,officialCurrency,M))).
```