Inter-Enterprise Business Document Exchange

Jingzhi Guo

Department of Computer and Information Science
Faculty of Science and Technology, University of Macau
Av. Padre Tomás, Pereira, S.J., Taipa, Macau, Tel: +853-397 4890

Email: jzguo@umac.mo

ABSTRACT

Electronic business document interoperation is the cornerstone of business process integration. An essential issue for business document interoperation is to maintain semantic consistency of the exchanged business documents between any two autonomous business communities, where the document sender and receiver have no misunderstanding in using the exchanged documents. Existing approaches to resolving this issue either adopts document standards to map heterogeneous document elements or applies business ontologies to mediate inconsistent document elements. While these approaches are effective in certain degree, the issues of limited flexibility and evolvability in using standards and the lack of accuracy in using ontologies to mediate document elements must be explored and resolved. This paper proposes a Collaborative Document Exchange (CODEX) approach to resolving the issues. In this approach, structures and concepts of business document are separated and layered in CODEX framework. Structures provide the commonality of business documents through classified concept identifiers while concepts support particularity of business documents through collaboration.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Web-based interaction*; H3.5 **[Information Storage and Retrieval]**: Online Information Services – Sharing Data; Web-based services.

General Terms

Theory, design

Keywords

Business document, semantic integration, semantic consistency maintenance, document exchange, document interoperation, concept collaboration, information sharing

1. INTRODUCTION

Electronic business documents interoperation is the cornerstone of business process integration [18] and is an important topic for electronic commerce [17]. An essential issue for business document interoperation is how a business document can be *faithfully* sent and received between any two autonomous business communities through computers, and the document sender and receiver have not misunderstood the uses of the exchanged documents. This faithfulness requires not only the consistent document structures but more importantly also the consistent meaning-level interpretation of document contents between the sender and the receiver. Without such faithfulness, interoperation cannot be established between interaction parties or at most the interaction is based on a wrong meaning interpretation.

ICEC'06, August 14–16, 2006, Fredericton, Canada.

For example, supposing that Party A has a business document RequestForQuote(RequestorName, RequestorAddress, RequestProductList(Refrigerator(color, price(currency, value, unitScale)), PriceTerm)), and Party B also has a document Inquiry(InquirerName, InquirerAddress, ProductItems(fridge, microwave oven, …), TermsOfPrice(FOB, New York)). Now Party A broadcasts its RequestForQuote document to all its possible vendors. Party B as one of the potential vendors has received Party A's business document, but it cannot understand what Party A is talking about.

Why can Party B not understand Party A's document? Several reasons can be listed:

1. B has different document name from A, i.e. RequestForQuote vs. Inquiry.
2. B has different document structure from A, i.e. Party A and Party B structure document elements in a different way. In a result, they have different control functions to read a given document.
3. B has different document elements from A to indicate the same meanings of elements, e.g. RequestorName vs. InquirerName, and RequestProductList vs. ProductItems.
4. B has different product name from A to refer to a same product, e.g. refrigerator vs. fridge.
5. B has different method from A to represent a product, e.g. Party A uses a tree structure to represent refrigerator while Party B simply put products in a list.

These differences make Party B even unable to receive Party A's document when the document arrives, and have become the issues that need to be solved.

With current technologies and practices, the above issue 1 (*document vocabulary interoperation issue*) may be resolved through using ontologies of semantic web to mediate the inconsistent semantic terms between document names, e.g. ontological mapping [19]. However, how to have those vendors who have no document vocabulary/ontology mapping to receive an unknown document is still an issue. The issue 2 (*document structure interoperation issue*) may be resolved through a common document message structure

supported by common document schemas (e.g. BizTalk public schema [2], cXML procurement documents [4], RosettaNet schema [25], ebXML business documents [5]). Nevertheless, there is still an open issue in a broad sense. Can all documents in the world be commonly structured? Indirectly, can we find a way to accurately transform heterogeneous document structures? For the issue 3 (*document element interoperation issue*), the Universal Document Element Framework (UDEF) has proposed a solution [30], but its element set is limited, and difficult to cope the dynamic change of document requirements beyond standard document elements. The issue 4 and 5 (*product data interoperation issue* [7][20][9]) belongs to product data integration research field, which is discussed in the research of ontology mapping (e.g. [20]), thesaurus linking (e.g. [16]) and product concept transformation (e.g. [10]). Currently, existing product vocabulary approaches are diverse in standardization [6][32], mediation [14] and collaborative concept mapping [11]. There is still no consensus on resolving this issue. In the following of this paper, we refer all the above issues as the *semantic consistency issue of electronic business documents* between distributed and autonomous business parties.

To resolve the semantic consistency issue of business documents, this paper provides an approach called COllaborative Document EXchange (CODEX). Primarily, CODEX approach applies the layer design thoughts from both semiotic concept analysis [26][1] and communication protocol such as Open Systems Interconnection (OSI). This reflects in our novel separation of a business document into the layers of document structure and document semantics, where a document structure is meaningless with regard to the interpretation of a business document and is only a conveyor for conveying document semantics. With this thought, any investigated document has the underlying structure layer and the higher semantics layer that denotes the structure layer. Thus, with regard to the whole expected CODEX system, there are several layers: each higher layer presents the semantics that is conveyed in its lower structure layer such that "semantics → structure (semantic → structure (semantics → structure (…)))". This design thought not only provides CODEX with a good modular design for reusing lower structure layer but also well explained what really a metadata is (not simply data about data but semantics on structure and as a whole to be another structure for its higher layer semantics). Secondly, based on the layer design thought, our approach adopts many valuable taxonomy practices for term classification such as UNSPSC [32], ecl@ss [6] and UDEF [30]. Thirdly, we largely apply the collaboration thought in the process of term classification, which makes the term/concept classification accurate and semantically consistent between multiple autonomous business partners.

As a design principle, we take a trichotomy of *systems*, *designers* and *users*: the systems provide tools and mechanisms for representing a consistent document structure model; the designers (viewed as a type of knowledge workers[1]) resolve semantic conflicts that systems are not capable of; and the users simply use the systems automatically. By this trichotomy, the semantic inconsistency incurred from the human interpretation can only be resolved in the semantic document design level, not in the lower level of CODEX system operations or in the higher user level for electronic business document exchange.

In the remainder of this paper, we describe the CODEX approach. First, we introduce CODEX framework, then we define an abstract document structure model. In Section 4, we implement this abstract model in XML language. Section 5 provides the CODEX system architecture and exemplifies the automatic document exchange on this architecture. Section 6 discusses some related works. The final section concludes the paper, lists the contributions and limitations of CODEX approach, and proposes some future works.

2. CODEX FRAMEWORK

Following the above design principle, a simplified CODEX framework can be illustrated in Fig. 1, which includes four layers of communication, document structure, semantics collaboration and document exchange.

The *communication layer* is a business document transport layer between sender and receiver. In this paper, we apply XML SOAP [8] for document transports. Each business document is embedded in the body of a SOAP message. The *document structure layer* is for document structure modeling. There are two types of document structures are modeled in this layer: one is *business document structure* and the other is *collaboration document structure*. The former will be *conceptualized* (i.e. to give meaning) by higher collaboration layer as a semantic business document, and transported in the lower communication layer. A *conceptualized business document* (also called as *document template*) has been semantically given with the meanings of document name and the inside semantic document elements during collaboration. The purpose of this document is to contain both human- and machine-understandable document meanings for both senders and receivers during document exchange. The latter is *conceptualized* by higher collaboration layer as an understandable collaboration document for mediating common understanding between collaborative designers of autonomous and heterogeneous semantic communities [23]. The purpose is to record semantics agreements between collaborative designers.

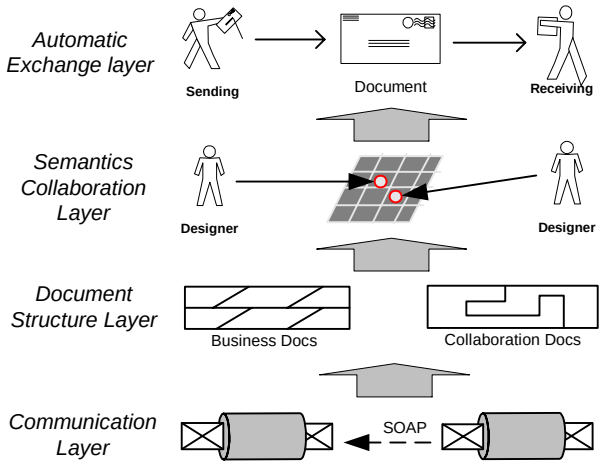

Fig. 1: Simplified CODEX Framework

Above the document structure layer is the *semantics collaboration layer*. Document designers in here resolve their semantic consistency issue in both document element creation and document composition. In this layer, the *semantics* of a term/concept refers

[1] Knowledge workers: "Their main value to an organization is their ability to gather and analyze information and make decisions that will benefit the company. They are able to work collaboratively with and learn from each other" [24]

to the meaning of a term in a *vocabulary*, which specifically refers to a concept/term set, for example, a set of document names, document elements or product terms. The top layer is the *automatic exchange layer*, which is a user layer. In this paper, we assume that semantic consistency issue can be resolved in the lower three levels. Thus, all documents and their inside terms/concepts can be semantically understood between sender and receiver. Therefore, users can automate the business document exchange without misinterpreting the exchanged business documents.

Since using XML SOAP as message transport structure is well-known and the automatic document exchange in user layer is based on the lower two layers, the rest of this paper only focuses on the discussion of document structure layer and semantics collaboration layer provided in the CODEX framework.

3. DOCUMENT STRUCTURE MODELING

The core of document structure layer is the abstract document structure model, which governs the way of enabling the implementation of this structure into the exchangeable documents in communication layer and the conceptualization of the structure into semantic documents in semantics collaboration layer. In this section, we will first briefly introduce some background knowledge of semiotic concept analysis. Based on this knowledge, we will model two types of document structures: a *business document structure model* that represents the general structure of a business documents (e.g. invoice, order sheet, etc.), and *collaborative document structure model* that represents some general structures of collaborative documents (e.g. a mapping document for various designers to work on).

3.1 Introduction to Semiotic Concept Analysis

A great influence on the following CODEX solution design is the semiotic concept analysis. According to Sausure's dyadic representation model ([26]:67), a representation (a sign) consists of two parts: a structure or a form (a signifier) and a concept or a meaning (signified), where for a given representation, structure takes the forms of representation and means nothing but just holds and conveys the concept. The model in Barthes' *orders of signification* ([1]:114:115) further explains that the structure and concept of a representation as a whole can again becomes a structure (signifier) that is denoted by a concept (signified), where the inside concept of this denoted structure is a connotation of the higher level concept. Thus, a concept is recursive, which denotes a same level structure and connoted by a lower level (inside) concept that again denotes its own level structure. This forms a concept tree naturally associated with a hierarchical structure with many levels. Thus, when we refer to a business document (e.g. inquiry sheet), the name "inquiry sheet" is a concept that denotes a document structure. Inside of the document structure, we have many lower level document elements, which may be the concepts of "inquirer", "address" and "product list". Each of them denotes its own structure. If we line the same level concepts in a sequence and mark them with sequential number, then we obtain a *concept tree* (1, *i*, …, *i*) [10] where "1" identifies the root concept to denotes the document structure of "inquirer", and inside concepts can be identified as 1.1, 1.2, …, 1.i to denote the structures of "1.1→address" and "1.2→product list". When any vocabulary is notated in this way, we could obtain a uniquely identified vocabulary tree, which is layered metadata (i.e. semantics about semantics or structure about structure, just choosing as one like) and can be referenced in other vocabularies.

3.2 Business Document Structure Model

A business document structure model describes the structure relationship within the scope of business document representation. It regulates the data relationship of business document structure.

Definition 1: *Business Document Structure Model* (BDSM)

A BDSM is described as a tuple such that *BDSM* = (*D*, *E*, *IID*, *AN*, *T*, *V*), where:

- *AN* is a set of annotation/definition of a term/concept.
- *IID* is a set of unique term/concept numeric identifier such that $iid \in IID \leftarrow an \in AN$ (interpreted as *iid* is uniquely annotated by an *an*) and *IID* = (1, *i*, …, *i*) is an *internal concept tree* defined in, or alternatively $iid_a \leftarrow iid_b$, which means a unique term/concept identifier in one vocabulary can be uniquely annotated by another term/concept identifier in another vocabulary.
- *D* is a taxonomy of business documents such that *D* is a tree structure and $d \in D$ is a document node in document taxonomy *D*, where $d \leftarrow iid$ (interpreted as an *iid* uniquely identifies a document *d*).
- *T* is a set of predefined data types (e.g. string, number, etc.)
- *V* is a set of values such that $V \leftarrow T$ (interpreted as the term/concept value *V* is instantiated by data type *T*).
- *E* is a set of document elements such that there exist some $e \in E$ in a document *d* where all *e* in this *d* is organized in a tree, and $e \leftarrow iid$ (interpreted as for each *e* in a document *d*, *e* is uniquely identified by an *iid*), and optionally $v \in V \leftarrow_{opt} e$ (interpreted as *e* optionally has term/concept value *v* if $\exists e \in d$ is a reified document).

Intuitively, BDSM can be illustrated in a diagram shown as in Fig. 2, where a set of documents $\{d_i \mid i \in 0\ldots n\} \in D$ is classified in a

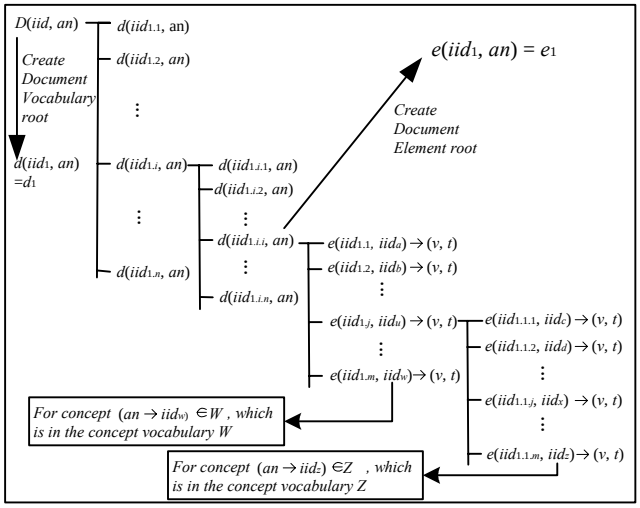

Fig. 2: Illustration of BDSM

tree *D* rooted from a document $d_1 \in D$ created by *D*, and a set of document elements $\{e_j \mid j \in 0\ldots m\} \in d_i$ forms a another tree rooted from a document element $e_1 \in d_i$ created by d_i and each e_j optionally has a typed value $e_j \rightarrow (v \in V, t \in T)$ if d_i is reified.

3.3 Functions and Benefits of Using IID

For a given document vocabulary, the function of IID in BDSM is to provide the programmable identifier structure for how the fu-

ture designed terms/concepts can be dynamically identified based on the concept tree *IID* = (1, *i*, …, *i*) when an annotation/definition is given to the document name of a given document. Similarly, a document element name is also assigned a dynamic term/concept identifier *iid* based on the rule of IID. The difference is that the annotation/definition of a document element identifier iid_e can be another term/concept identifier iid_x which is defined in another vocabulary *X*, which may or may not be designed in a similar way.

The introduction of IID-ed term/concept identifier written in a way of a numeric tree path (e.g. 1.52.14.15.1) fully eliminates the ambiguity and heterogeneity in the arbitrary use of human designed term/concept identifiers such as UNB, UNH and BGM used in EDI. It has increased the accuracy in mapping heterogeneous terms/concepts between contextually different semantic communities. The second benefit is that heterogeneous terms/concepts of different natural languages or dialects can be easily mapped without the need to consider whether the other participants are using other languages. Another extra benefit is that since an *iid* is a regular numeric tree path expression, it can immediately find out its given position in a vocabulary and know who are its parent and ancestors. For example, 1.52.14.15.1 refers to 1:ProductCatalogue(52:domestic appliances and consumer electronic products(14:domestic appliances(15:kitchen appliances(1:refrigerators)))). With this contextual knowledge, when we are given the unique identifier 1.52.14.15.1, we can not only know what 1.52.14.15.1 refers to but also do all its ancestors such as 1.52.14.15 referring to "kitchen appliances" and 1.52.14 referring to domestic appliances". This is extremely useful to infer whether two seemingly same annotations are really identical in meanings if we need to decide whether two terms/concepts are from the same context.

3.4 Collaborative Document Structure Model

A collaborative document structure model describes the structure relationship within the scope of designers' collaboration representation. In another word, it represents how the collaboration results can be mediated between semantically inconsistent parties.

Collaboration, in general, has three modes: *peer-to-peer collaboration* (P2P), *dominator-to-follower collaboration* (D2F) and *requestor-to-answerer collaboration* (R2A). P2P means that all collaborators are equal. They negotiate with each other to resolve their semantic inconsistencies in designing terms/concepts for a vocabulary. D2F means that there are some collaborators that are strong enough in positions (i.e. dominators). They design the terms/concepts for a vocabulary and has other collaborators (i.e. followers) to use or map their designed terms/concepts. This collaborative mode is similar to the process of standardization and adoption. R2A means that some collaborators (i.e. requestors) are incapable of designing the useful terms/concepts (e.g. terms/concepts that can be accepted everywhere) and then request some capable collaborators (i.e. answerers) to design useful terms/concepts in a vocabulary. These three collaborative modes lead to the different structure models for representing collaborative relationships. Besides the above three different collaborative modes, the different execution times for collaboration (i.e. *synchronous collaboration* (SYNC) and *asynchronous collaboration* (ASYNC)) also affect the design of structure model for representing collaboration relationships. In this subsection, we will describe an integrated model to concern the above different collaboration modes.

3.4.1 Generic CDSM

Definition 2: *Collaborative Document Structure Model* (CDSM)

A CDSM is described as a tuple *CDSM* = (*AN*, *T*, *CVT*, *IID*, *AID*, *C*, *P*, *U*, *CΓ*, *LΓ*, *CU*, *LU*), where:

- *AN* is a set of annotations/definitions exactly as defined in Definition 1.
- *IID* is a set of unique term/concept numeric identifier exactly as defined in Definition 1.
- *AID* is a set of alias identifiers corresponding to IID.
- *T* is a set of predefined data types exactly as defined in Definition 1 for constraining values.
- *CVT* is a set of conversion functions that convert one data type to another data type (e.g. 1 pair = 2 piece or 1 dozen = 12 pieces, and 1 dozen pair = 12*2 piece).
- *C* is a set of concepts such that *C* = (*IID*, *AN*, *T*, *CVT*), where $AN \rightarrow IID$ and $(T, CVT) \rightarrow$ value.
- *P* is a set of vocabulary/document designers.
- *U* is a subset of *P*, called a *semantic community* [23] such that for any two U_1 and U_2, $U_1 \cap U_2 = \varnothing$.
- *CΓ* is a P2P collaboration mechanism.
- *LΓ* is a D2F collaboration mechanism.
- *CU* is a subset of *P*, called *common semantic community*, such that given any two semantic communities U_1 and U_2, there exists a P2P collaboration mechanism *CΓ* that makes $CU=\{U_1, U_2\}$, where for all concepts C_1 designed by U_1 and for all concepts C_2 designed by U_2, there is a set of shared concepts *C*, called *common concepts*, such that C_1 and C_2 *semantically equivalent* to *C*, notated as $CCMAP(C_1, C)$ and $CCMAP(C_2, C)$. Since $AN \rightarrow IID$, the concept sharing relationship can also be notated as $CCMAP(IID_1, IID)$ and $CCMAP(IID_2, IID)$. The designers in *CU* are called *common concept designers CP*.
- *LU* is a subset of *P*, called *local semantic community*, such that given a semantic community $CU_x \subseteq CU$ and a semantic community $U_y \not\subset CU$, there exists a D2F collaboration mechanism *LΓ* that makes $LU=\{CU_x, U_y\}$, where for all concepts C_1 designed by U_y and for some common concepts C_2 designed by CU_x, there is a mapping relationship $LCMAP(C_1, C_2)$, where C_1 is semantically equivalent to C_2. The C_1 is called *local concepts*. Since $AN \rightarrow IID$, the concept mapping relationship can also be notated as $LCMAP(IID_1, IID_2)$. The designers in *LU* are called *local concept designers LP*.

The above CDSM has depicted the collaboration relationships of P2P in a common semantic community and D2F in a local semantic community. These collaboration relationships are represented in the form of concept storage method. For example, given a common semantic community with two common concepts CU_1: C_1(1.52.14.15.1, refrigerator) and CU_2: C_2(1.52.14.15.1, 电冰箱), and two local semantic communities with two local concepts LU_1: C_3(xyz1, fridge) and LU_2: C_4(XG2, 雪柜), since we have P2P collaboration relationship between CU_1 and CU_2, the semantic consistency between "refrigerator" and "电冰箱" can be maintained and uniquely identified by "1.52.14.15.1". Also since we have D2F collaboration relationship between CU_1 and LU_1 and between CU_2 and LU_2, we have maintained semantic consisten-

cies between "refrigerator" and "fridge" with *lcmap*(1.52.14.15.1, xyz1) and between "电冰箱" and "雪柜" with *lcmap*(1.52.14.15.1, XG2). With this representation structure, it allows heterogeneous semantic communities to work together to design semantically consistent concepts for incremental vocabulary creation.

3.4.2 Synchronous and Asynchronous Collaboration

CDSM model is an evolvable model, i.e. common concepts and local concepts are constantly ADDED and DELETED to adapt to the new customer requirements during both synchronous and asynchronous collaboration. This poses a significant challenge of how to maintain semantic consistency between a *common document* (i.e. a set of common concepts) and many *local documents* (i.e. many sets of local concepts) in a dynamic environment. This subsection first redefines the local document structure and then provides the mechanism and principle for the two primitive operations ADD and DELETE to cope with this challenge.

Definition 3: *Local Document Structure*

Given a *common document structure D*(*ComIID*, *AN*, *<others>*), then a *local document structure* is defined as *D*(*LocIID*, *ComIID*, *AN*, *<others>*) such that *comIID→LocIID*, where *ComIID* is a concept tree (1, *i*, …, *i*) for a common document element identifiers and *LocIID* is the set of local document element identifiers.

With this definition, a local document is always a concept subset of a common document and partially retains the document tree structure of the common document. Thus, a local-common concept map *LCMAP*(*ComIID*, *LocIID*) between common document and local document is, in fact, locally maintained in the local part of a local semantic community *LU* in *D*(*LocIID*, *ComIID*, *AN*, *<others>*).

Now, suppose that, in a common document, a *comIid*∈*ComIID* is added (e.g. add 52 from 1.14.15 to 1.52.14.15), the ADD result must also reflect in all local documents. Obviously, we have two cases for ADD operation: synchronous ADD and asynchronous ADD. To support the both types of ADD in a flexible way, we apply the state-of-the-art flexible notification technique discussed in [28] and create a *one-way buffer mechanism*, where two buffers are built: one outgoing buffer (OB) for common document as output and one incoming (IB) for local document as input. Two buffers OB and IB are causally ordered based on the sequence of the ADD operations on common document such that IB=OB. If the local document is online, then IB is immediately operated on the local document in a synchronous way. If it is offline, then the ADD operations are accumulated in OB until it becomes online to

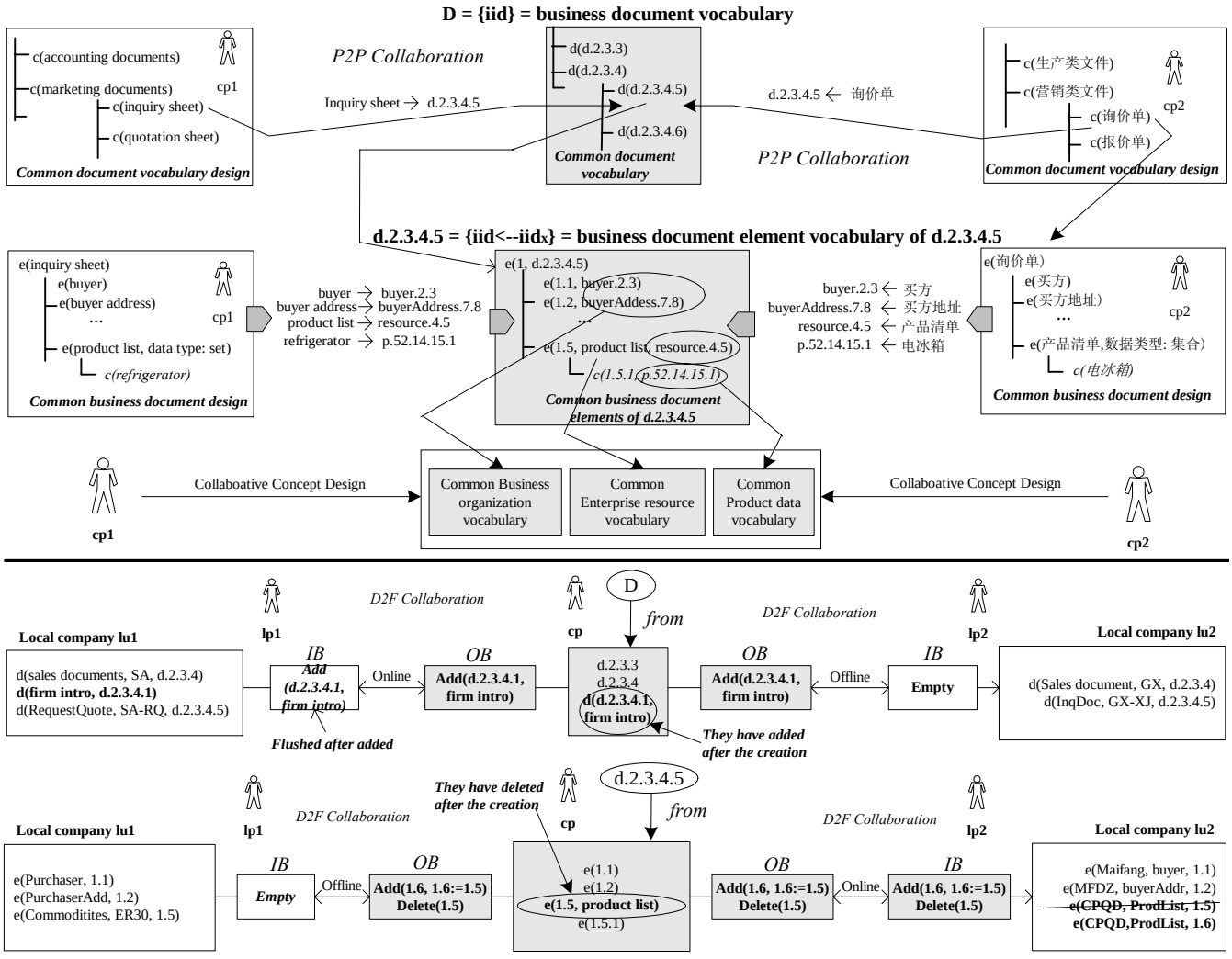

Fig 3: Illustration of CDSM

execute the asynchronous collaborative ADD operations.

For the DELETE operation, since the classification structure itself requires semantic consistency maintenance [12], before using the above buffer mechanism, a DELETE principle must be followed. That is, prior to deleting a $comIid_1 \in ComIID$, the deleted element concept must first be added to another node of $comIid_2 \in ComIID$ and then to execute the DELETE operation with one-way buffer mechanism, or a concept deletion must first be notified as "deleting concept" and can be finally deleted until it is no longer used in all local documents. The notice of "deleting concept" is an explicit request to local document designers that they should not use this element concept any more for further documentation and should append the deleting concept to other concepts.

The one-way buffer mechanism is a push approach, which maintains a very fine grain for notification message (in our case only the concept identifier *IID* and/or concept annotation *AN*) in a one-to-many manner. The DELETE principle is a guarantee of semantic consistency. The combined use of both ensures the requirements of evolvability and accuracy.

In the context of e-commerce applications, the cost of one-way buffer mechanism will not be high, because we can assume all e-commerce applications stay online. The "offline" happens only in emergency cases. This assumption is important because the fine grain notification message can almost neglect the buffer cost but the large buffer may require additional mechanisms and may cause network congestion. Thus, for e-commerce applications, the one-way buffer mechanism is flexible and allows millions of local documents to collaborate with a common document.

The discussion of implementing one-way buffer mechanism is out of the scope of this paper. In the following subsection, we will demonstrate the innovative CDSM model in a graphical way.

3.5 Graphical Demonstration of CDSM

Fig. 3 uses the motivation example of Section 1 to demonstrate how various concept designers in different semantic communities can work together to resolve their semantic discrepancies in business document design.

First, a group of common concept designers cp_1, $cp_2 \in CP$ collaborate with each other in P2P mode (we assume that common concept designers may use a common language of English or multiple languages). They achieve their consensus in the meanings of each business document name and represent these document names in numeric $iid \in IID$ (e.g. marketing document = 营销类文件 →d.2.3.4.5). Each *iid* is automatically generated based on the *IID generation rule* (i.e. identifier computed on the concept tree (1, *i*, …, *i*)). When differently expressed document names are converged to a same unique *iid*, they become interoperable between multiple semantic communities through this *iid*.

Second, given a document *iid*, the P2P common concept designers cp_1, $cp_2 \in CP$ now can design this *iid*-ed business document template (i.e. common business document d.2.3.4.5 shown in Fig. 3), which is a set of document elements aligned in a tree structure and identified by a set of {*iid*}. The tasks of document template designers are in three aspects: to collaboratively design the needed document elements (i.e. buyer↔买方, …, product list↔产品清单), to collaboratively align these elements in the mutually agreed tree structure (i.e. 1, 1.2, …, 1.5), and to collaboratively reference each document element with a field concept that is used for document element value instantiation (i.e. 1.1→buyer.2.3, …, 1.5→resource.4.5). This final step, in fact, associates each document element with an external concept that is again collaboratively designed for making agreements on concept definition, unique identifier, data type and possible conversion function applied (e.g. the vocabulary of common organization vocabulary, common enterprise resource vocabulary, and common product data vocabulary shown in Fig.3).

Third, after the document vocabulary and common document templates are designed, they can be used by local concept designers lp_1, $lp_2 \in LP$ in the way of D2F collaboration mode (we assume that local concept designers use a single language such as English or Chinese). In this collaboration mode, local designers can browse the document vocabulary through Internet program to create their own tailored and personalized document vocabulary and business document templates. They can substitute common document names and document elements in their own terms and can shrink the tree structure of the document vocabulary and document templates shown in the D2F collaboration between local concept designers *lp* and common concept designers *cp* (e.g. SA-RQ for d.2.3.4.5 in $lu1$ and GX-XJ for d.2.3.4.5 in $lu2$).

Fourth, an extremely important issue in designing vocabulary/document is to enable a common vocabulary/document to be dynamically revised as the business requirements or the expansion of the vocabulary/document. This issue often bothers many existing international product and document standards such as UNSPSC [32] and UDEF [30]. The CDSM model provides a one-way buffer mechanism and a DELETE principle to handle this problem. For example, the operation of adding a document/vocabulary term is first causally ordered in the outgoing buffer OB as sending message and then placed in the incoming buffer IB as receiving message. If the local vocabulary/document is online, the IB is immediately executed synchronously (e.g. *d*(firm intro, *d*.2.3.4.1) is added in lu_1 shown in Fig. 3). If it is offline, the ADD and DELETE operations are accumulated in OB until it is online to transfer into its IB. A DELETE operation is either as developed into a conditional MOVE operation (i.e. first ADD then DELETE) (e.g. DELETE(1.5) = ADD(1.6, 1.6:=1.5) AND DELETE(1.5) shown as in Fig. 3 for lu_2) or is a complete DELETE operation with a "deleting concept" notice to local concept designers (it cannot be deleted immediately until no local document uses it).

In the following section, we will discuss the implementation of above discussed BDSM and CDSM models.

4. DOCUMENTING IN XML

The document structures modeled in Section 3 are abstract data models. To be usable for computer understanding in a semantic way, this section implements these models in XML language for semantic encoding because XML is a platform independent language and can be encapsulated in XML SOAP [8] for transparent message transport. Since we regard a document name vocabulary as a set of concepts, we can apply XML concept generation rules (1, *i*, …, *i*) to implement it and thus will not discuss it here. In this section, we focus on the implementation of BDSM and CDSM, which represent the document element relationship and designer collaboration relationship.

4.1 XML Business Document

An XML implementation of business document is to implement the abstract language *BDSM* = (*D*, *E*, *IID*, *AN*, *T*, *V*). Its requirements are firstly not to lose the simplicity of BDDM, secondly to

keep its evolvability of IID, and thirdly to enable the values of document elements to take strongly typed values. To meet these requirements, we define *XML Business Document* (*XBD*) in the following as a set of revised XML rules discussed in [10].

Rule 1 (*Document Element Connotation*): Assuming a document element $e \in E$ of *BDSM* is a concept tree node that is connoted by zero-to-many subtree nodes {*e*} and each document element *e* can optionally has a value $v \in V$ of BDSM, then it can be mapped onto an XML document tree defined in an XML DTD:

<!ELEMENT bd (e*)>

<!ELEMENT e (#PCDATA | e*)>

where the root element *bd* is a document level element identified by the currently selected business document $iid \in D$. The connotation *e** (a set of document element concepts) map onto a set of child nodes of business document element concepts. A child element concept is again connoted by a set of child element concepts until to leaf element concepts. Take the example of the Party A case of Section 1, we have:

```
<bd><!--RequestForQuote-->
    <e></e><!--RequestorName-->
    <e></e><!--RequestorAddress-->
    <e></e><!--Request-ProductList-->
        <e><!--Refrigerator-->
          <e></e><!--color-->
          <e><!--price-->
            <e></e><!--currency-->
            <e></e><!--value-->
            <e></e><!--unitScale-->
          </e>
    <e></e><!--PriceTerm-->
  </e>
</bd>.
```

Rule 2 (*Document Element Denotation*): Assuming that document elements $e \in E$ is defined in a tuple of $E = (IID, AN, T)$ from the model BDSM, then a document element *e* has attributes $iid \in IID$, $an \in AN$ or an external $iid \in$ (*any applicable vocabulary*) and an data type $t \in T$. They can be mapped onto an XML node with DTD definition as following:

```
<!ATTLIST bd
    d:iid ID #REQUIRED
    xmlns:d CDATA #REQUIRED
    xmlns:r CDATA “”>
<!ATTLIST e iid ID #REQUIRED
                an CDATA #REQUIRED
                r:iid CDATA “”
                r:t CDATA “”>
```

where the *xmlns:d* namespace points to the document name vocabulary, which uniquely identifies the whole document. The document element *iid* is dynamically generated during document conceptualization. The external *iid* that can substitute an *an* is from the namespace *xmlns:r* for containing external resources such as organization vocabulary, enterprise resource vocabulary, product data vocabulary and data type vocabulary. It must be noted that resource namespace is for *real* vocabularies that are used to validate the document elements in *conceptualization* of the DTD into document templates. Certainly, to increase processing speed, this namespace can be stored locally if possible. However, to provide the distributed design of business document templates, it is in principle placed anywhere on Internet for run-time validation of external concept identifiers. For example:

```
<bd d:iid="d.2.3.4.5"
      xmlns:d=“http://default-document”
      xmlns:r="http://resource">
   <e iid="e.1" an=“buyer” r:iid="buyer.2.3" r:t="string"/>
   <e iid="e.5" an=“product list”
                r:iid="resource.4.5" r:t=“set”/>
</bd>.
```

When the above semantically conceptualized document template is needed to take values for real use, the document experiences a process of *reification*, for example:

```
<bd d:iid="d.2.3.4.5"
      <!-- omitted namespaces, see above example -->
   <e iid="e.1" an=“buyer” r:iid="buyer.2.3" r:t="string">
     Haier
   </e>
   <e iid="e.5" an=“product list” r:iid="resource.4.5">
      <e iid="e.5.1" an=“product name” r:iid="p.52.14.15.1"
          r:t="string">Refrigerator </e></e>
</bd>.
```

Rule 3 (*Document Element Classifier*): Each document element has a unique identifier *iid* in its current document. This identifier is used to identify the corresponding element concept. The dynamic *iid* generation is based on the IID rule such that the document root node is “1” and its child nodes are 1.1, 1.2, …, 1.*n*. This rule classifies all *iid*s of a business document such that given a current $iid_{1.i...i}$ then its new sibling element identifier is $iid_{1.i...(i+1)}$ and its new child element identifier is $iid_{1.i...i.(i+1)}$.

Given the above Rules, during P2P collaboration between document designers, business document templates are designed with mutually agreed *AN* (annotation), *IID* (element concept identifier), *T* (data type for each element value) and the application of existing vocabulary namespaces. This collaboration is a process of *conceptualization* of an abstract business document structure (i.e. a DTD) into a semantically conceptualized business document template (i.e. filled with the document element concepts). When this template is further reified (i.e. the template is used and the document elements are given values, e.g. XYZ such that <e *iid*=“1.2.3””> XYZ </e>), we say that a business document template has a *reification*.

4.2 XML Collaboration Document

The key to implement CDSM is to build the collaboration relationship between collaborators between different semantic communities. To represent the collaboration relationship, we assume that all concepts *C* of different vocabularies involved in a business document (e.g. document name vocabulary, product data vocabulary, enterprise resource vocabulary, organization vocabulary and data type vocabulary) have been created elsewhere such that $C = (IID, AN, T, CVT)$, where $AN \rightarrow IID$ and $(T, CVT) \rightarrow$ value defined in Definition 2, and each *iid* identifies a concept $c \in C$ such that $iid \leftarrow c$.

4.2.1 P2P Collaboration

Based on this assumption, the requirements of implementation are to enable P2P collaboration in synchronous mode, and to provide the flexibility of collaborative concept editing in different autonomous semantic communities. To meet these two requirements, we define *XML Collaboration Document* (*XCD*) in a set of concept mapping rules.

Rule 4 (*P2P Collaboration*): Given any two P2P collaborative semantic community cu_1, $cu_2 \in CU$, then the collaborative editing relationship between cu_1 and cu_2 can be mapped onto two XML DTDs:

<Rule 1 + Rule 2 +
lock (yes | no) #REQUIRED><--GBD-->

Assuming that prior to using Rule 4, Rule 1 and Rule 2 are used to build P2P *common business document templates CBD* for all P2P collaborative semantic community cu_1 and cu_2. The DTD of Rule 4 thus builds a P2P global business document template *GBD*, where a lock attribute is additionally added to each document element. For each $cbd \in CBD$, its element identifier set {*iid*} is exactly the same as that of the *GBD*, except for the annotation/term definition language may different. When a common concept designer $cp \in CU$ wants to create new element concepts in leaf elements, s/he just switches the lock to "*yes*". If s/he edits a document element that is not a leaf element, after s/he locks the element, others cannot edit not only this element *e*:*iid* and but also all its child element *e*:*iids*, but others can lock any child elements to edit the annotation *e*:*an*, the external resource *r*:*iids* or the data type *r*:*t* (i.e. the attribute values of *r:iid* or *r:t*). This is because the annotation and the external concept identifiers do not affect the integrated *IID* structure of the being edited business document, i.e. the document template can be developed in an anticipated way. The lock including a dead lock prevention mechanism guarantees that no simultaneous editing will occur to induce side effect of semantic inconsistency between the collaborators *cps* of a common semantic communities *cu*.

If a common business document template *cbd* has already been mapped onto some local business document templates *LBD*, then an outgoing buffer mechanism *OB* must be built for supporting D2F collaboration. In the following, we describe the D2F collaboration in details.

4.2.2 D2F Collaboration

For the D2F collaboration between local designer $lp \in lu$ and a common designer $cp \in cu$, the collaboration relationship is different from P2P collaboration:

- Any local business document $lbd \in LBD$ in *lu* only takes a subset of the document elements *e* of the common business document *cbd* in *cu* such that $\{e \in lbd\} \leq \{e \in cbd\}$.
- The local document designer *lp* has no right to revise the common business document *cbd*. However, both *lp* and *cp* have the right to revise their own document elements, i.e. both are autonomous such that the representation of an $e \in cbd$ is different from an $e \in lbd$. No right to revise *cbd* from *lp* simplifies the collaboration but the mutual autonomy of *cp* and *lp* complicate the collaboration relationships.
- D2F collaboration must consider both asynchronous and synchronous collaboration models.

Based on the understanding of these differences, we define the following XML rule:

Rule 5 (*Local-to-Common Concept Mapping* "LCMAP"): Assuming both *lu* and *cu* have a business document template conceptualized from the DTD defined by Rule 1 and 2 such that local designer *lp* always copy and personalize the *cbd* into his/her own local *lbd* such that *lbd* semantically belongs to *cbd*, then there is a *local-to-common concept mapping LCMAP* between *lbd* and *cbd* such that *lcmap*(*cbd*, *lbd*). Since both *cbd* and *lbd* can semantically expressed as the set of concept identifier *IID*, we have *lcmap*(*ComIID*, *LocIID*).

Following the definition 2, we merge the local-to-common concept mapping relationship into the new DTD of local business document template by revising the DTD defined by Rule 1 and 2, such that:

```
<!ELEMENT lbd (e*)>
<!ELEMENT e (#PCDATA | e*)>
<!ATTLIST lbd
          locIid CDATA #REQUIRED
          comIid ID #REQUIRED>
<!ATTLIST e locIid ID #REQUIRED
          comIid #REQUIRED
          an CDATA #REQUIRED>
```

where the element concept mapping relationship between *cbd* and *lbd* is aligned into the same document element.

When a common designer *cp* edits a document element concept *e* in common business document *cbd*, it places the editing results in its outgoing buffer *OB*, which is received by the incoming buffer *IB* of a local business document *lbd* and is used to be executed on the *lbd*.

4.2.3 Comparing P2P and D2F Collaboration

Comparing the P2P and D2F collaboration[2], P2P collaboration has introduced the mechanisms of locking. Why a lock has to be used is that semantic level collaboration is different from the normal syntactic level collaboration, which aims only to present an identical document presentation state of multiple replicated documents (e.g. collaborative text document [29]) between multiple collaborative parties. Semantic level collaboration requires that the collaborative results must reflect the identical concept (i.e. meaning) interpretation of both document template structure evolution and the term concepts used in conceptualizing the document templates between multiple collaborators. Thus, the simultaneous revision to a same business document will generate unrecoverable semantic inconsistencies, which is not tolerable in semantic interoperation between business partners. In this sense, locks placed in a global business document template common to all P2P collaborators are desirable. Contrasting with D2F collaboration, since common document designers *cp* are in a dominant position, the local document designers *lp* have no rights to edit common business documents. Thus, *cp* has no need to issue a lock for any of its revising document element. However, it must have a mechanism to advise *lp* that an element is revised but still keeps *lp* in a semantic consistent way. This is why a one-way buffer mechanism is developed. This mechanism not only resolves asynchronous issue but also links heterogeneous semantic element representations in a persistent local business document template.

5. CODEX SYSTEM

This section describes CODEX system to the issue of semantic consistency maintenance (see Fig. 4).

5.1 System Architecture

The CODEX system is Web-based. Its architecture includes four types of participants: *business document service providers* for collaboration and transformation (BDSP), *common document*

[2] The R2A collaboration mentioned in Section 3.3 requires a very different collaboration mechanism and will not be discussed in this paper.

designers (CDD), *local document designers* (LDD), and *business document users* (BDU). All participants are connected and communicated through SOAP messaging, where the exchanging documents are embedded in the SOAP body. To facilitate the design collaboration between CDDs and LDDs and the document exchange between BDUs, the business document service providers BDSP provide two types of software engines: the document collaboration engine and the document transformation engine. The document collaboration engines include *common-to-common collaboration engine* (CCCE) and *local-to-common collaboration engine* (LCCE). The CDDs collaborate with each other in P2P collaborative mode on CCCE to design *common business documents* (CBD). During their collaboration, they can lock and check the collaboration status through the *global document with locks* (GBD) This document is kept in BDSP and coordinates the concurrent design of the element nodes of common document templates. In the design of common document templates, CDDs can also access to a large number of external vocabulary resources (Resource) such as product vocabulary, which are managed by Collaboration Manager of BDSP. The LDD and CDD collaborate with each in D2F collaborative mode on LCCE to design *local business documents* (LBD). Through their collaboration, *local-to-common concept maps* (LCMAP) are created to map local document elements and common document elements. The document transformation engines can be classified as *common-to-common transformation engine* (CCTE) and *local-to-common transformation engine* (LCTE). A *CCTE* automatically transforms one common *user business document* (BizDoc) received from *LCTE* into another common *BizDoc* based on the common IID. An *LCTE* automatically transforms a local *BizDoc* received from *local transformation user interface* (LTUI) into a common *BizDoc* based on the local-to-common concept maps in local business documents (LBD).

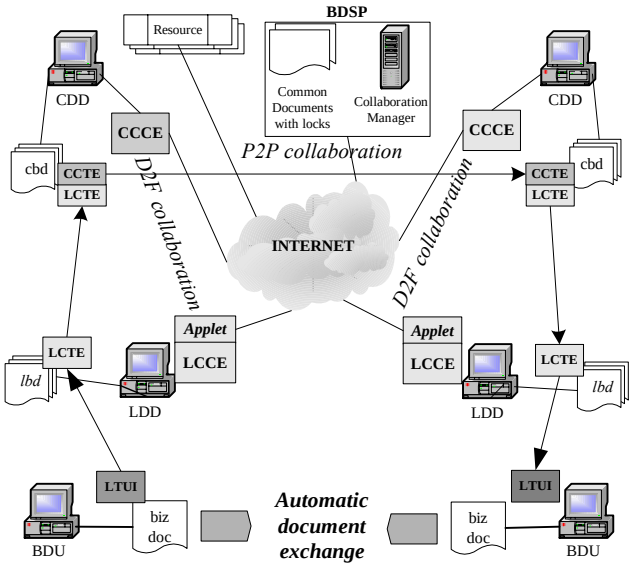

Fig. 4: CODEX Architecture

5.2 Automatic Document Exchange Example

We exemplify automatic document exchange ability of CODEX solution in Fig. 5 based on the system architecture of Fig. 4 and XBD and XCD languages designed in Section 4.

```
<?xml version='1.0' ?>
<!DOCTYPE bd SYSTEM "bd.dtd">
<bd d:iid="SA-RQ" xmlns:d="http://LU1/docs" xmlns:r="http://LU1/resources">
  <e iid="e.1" an="requestor Name" r:iid="R_NAME" r:t="string">Collins</e>
  <e iid="e.2" an="requestor address" r:iid="R_ADDR" r:t="string">New York</e>
  <e iid="e.5" an="product list" r:iid="P_LST" r:t="set">
    <e iid="e.5.1" an="kitchen fridge" r:iid="P356">
      <e iid="e.5.1.1" an="color" r:iid="P356.2" r:t="string">white</e>
      <e iid="e.5.1.2" an="price" r:iid="P354.3">
        <e iid="e.5.1.2.1" an="currency" r:iid="P354.3.1" r:t="scalar">USD</e>
        <e iid="e.5.1.2.2" an="unit value" r:iid="P354.3.2" r:t="value"> ? </e>
        <e iid="e.5.1.2.3" an="price unit" r:iid="P354.3.3" r:t="unit">piece</e>
      </e>
    </e>
    <e iid="e.5.2" an="microwave oven" r:iid="P363" r:t="string"/>
  </e>
</bd>
```

LCTE

```
<?xml version='1.0' ?>
<!DOCTYPE bd SYSTEM "bd.dtd">
<bd d:iid="d.2.3.4.5" xmlns:d="http://CU1/docs" xmlns:r="http://CU1/resources">
  <e iid="e.1" an="buyer's name" r:iid="buyer.2.3" r:t="string">Collins</e>
  <e iid="e.2" an="buyer's address" r:iid="buyer.2.3.2" r:t="string">New York</e>
  <e iid="e.5" an="product list" r:iid="resource.4.5" r:t="set">
    <e iid="e.5.1" an="domestic refrigerator" r:iid="p.52.14.15.1">
      <e iid="e.5.1.1" an="color" r:iid="p.52.14.15.1.2" r:t="string">white</e>
      <e iid="e.5.1.2" an="price" r:iid="p.52.14.15.1.3">
        <e iid="e.5.1.2.1" an="currency" r:iid="p.52.14.15.1.3.1" r:t="scalar">USD</e>
        <e iid="e.5.1.2.2" an="unit value" r:iid="p.52.14.15.1.3.2" r:t="value">?</e>
        <e iid="e.5.1.2.3" an="price unit" r:iid="p.52.14.15.1.3.3" r:t="unit">piece</e>
      </e>
    </e>
    <e iid="e.5.2" an="microwave oven" r:iid="p.52.14.15.2" r:t="string"/>
  </e>
</bd>
```

CCTE

```
<?xml version='1.0' ?>
<!DOCTYPE bd SYSTEM "bd.dtd">
<bd d:iid="d.2.3.4.5" xmlns:d="http://CU2/docs" xmlns:r="http://CU2/resources">
  <e iid="e.1" an="询价方名称" r:iid="buyer.2.3" r:t="string">高林公司</e>
  <e iid="e.2" an="询价方地址" r:iid="buyer.2.3.2" r:t="string">纽约</e>
  <e iid="e.5" an="所询商品清单" r:iid="resource.4.5" r:t="set">
    <e iid="e.5.1" an="家用电冰箱" r:iid="p.52.14.15.1">
      <e iid="e.5.1.1" an="颜色" r:iid="p.52.14.15.1.2" r:t="string">白色</e>
      <e iid="e.5.1.2" an="价格" r:iid="p.52.14.15.1.3">
        <e iid="e.5.1.2.1" an="货币" r:iid="p.52.14.15.1.3.1" r:t="scalar">美元</e>
        <e iid="e.5.1.2.2" an="单价" r:iid="p.52.14.15.1.3.2" r:t="value">?</e>
        <e iid="e.5.1.2.3" an="价格单位" r:iid="p.52.14.15.1.3.3" r:t="unit">台</e>
      </e>
    </e>
    <e iid="e.5.2" an="家用微波炉" r:iid="p.52.14.15.2" r:t="string"/>
  </e>
</bd>
```

LCTE

```
<?xml version='1.0' ?>
<!DOCTYPE bd SYSTEM "bd.dtd">
<bd d:iid="XJD" xmlns:d="http://LU2/docs" xmlns:r="http://LU2/resources">
  <e iid="e.1" an="买方" r:iid="MF" r:t="string">高林公司</e>
  <e iid="e.2" an="买方地址" r:iid="MFD2" r:t="string">纽约</e>
  <e iid="e.5" an="商品" r:iid="SP" r:t="set">
    <e iid="e.5.1" an="冰箱" r:iid="SP686">
      <e iid="e.5.1.1" an="颜色" r:iid="SP686.1" r:t="string">白色</e>
      <e iid="e.5.1.2" an="价格" r:iid="SP686.2">
        <e iid="e.5.1.2.1" an="货币" r:iid="SP686.2.1" r:t="scalar">美元</e>
        <e iid="e.5.1.2.2" an="单价" r:iid="SP686.2.2" r:t="value">?</e>
        <e iid="e.5.1.2.3" an="单位" r:iid="SP686.2.3" r:t="unit">只</e>
      </e>
    </e>
    <e iid="e.5.2" an="微波炉" r:iid="SP710" r:t="string"/>
  </e>
</bd>
```

Fig. 5: Example of Automatic Document Exchange

Fig. 5 shows that an English RFQ in LU1 is automatically transformed into a Chinese RFQ in LU2. The transformation experiences LU1→CU1→CU2→LU2, where transformation engines LCTE and CCTE transform heterogeneous identifiers IID based on the CCP documents stored in CUs.

6. COMPARISON TO RELATED WORKS

In this section, some related works in semantic consistency maintenance for exchanging multiple inter-enterprise business documents will be compared, with special regard to their capability in achieving inter-enterprise interoperability in a distributed and autonomous environment.

The CODEX approach is based on the CONEX project [3], which proposed a collaborative concept exchange approach, focusing on maintaining semantic consistency between ad hoc product data through a set of collaboration procedures. The CODEX approach has inherited its collaboration thought but focuses on resolving semantic consistency issue in business document integration domain through formal modeling and one-way buffer mechanism besides locking. Special considerations of CODEX Framework are how an inter-enterprise business document can be represented in a set of classified document element identifiers, how existing classified vocabularies can be directly used, and how document

collaboration relationship can be represented to allow CODEX system implementation.

The CODEX approach differs from other related works. First, EDIFACT [31] is a single document standard, which is modeled and promoted for users to adopt in designing their documents. Technically, for EDIFACT, the semantic consistency maintenance between inter-enterprise business documents is limited to the trading partners that have used the same business document modeling standard. Outside of these trading partners, business document interoperation is not possible. More importantly, all semantic document term/concepts are the standard abbreviated types (i.e. alphabetic words), which have no clues to classify them for easily conceptualizing a business document template though EDIFACT provides annotations for each type. These types (e.g. UNB, UNH and BGM in D93A Quotes of EDI) are rigidly written and cannot be changed. The CODEX approach introduces well-designed IID on concept tree [10] to substitute standard abbreviated types, which has a clear hierarchical structure associated with corresponding annotations for web view during collaborative document element design. This overcomes the shortcoming of proprietary properties of EDIFACT and makes CODEX open and evolvable for ongoing collaborative editing of a given business document.

Second, Dublin Core [22][15] is a metadata standard that describes Web resources as document-alike objects. The primary purpose of Dublin Core is to enable Web resource discovery and thus aims to be a simple metadata standard for allowing unconstrained multiple views of metadata design on the same resources. While the metadata structure without the nesting ability limits Dublin Core to describe complex business documents, the unconstrained multiple view design limits Dublin Core to being only effectively workable in the resource discovery scope. This is because different view designs often have some understanding gap to a given resource if no solution is given to resolving the inconsistent semantics from multiple views. The CODEX approach focuses on the exchange of semantic consistent business document. It not only keeps simplicity of document element structure but also provides the nesting ability of document elements through the explicit IID structure of concept tree. It resolves the semantic consistency issue through collaboration engines, where collaborative designers can not only maintain their local views but also maintain semantic consistency between multiple views. Similar to Dublin Core, RDF (Resource Description Framework) or later OWL (Ontology Web Language), both standardized by W3C (www.w3.org) lacks the mechanism to avoid the semantic inconsistency when multiple design views are allowed from the distributed and autonomous design communities.

Third, UDEF [30] is a document element mapping standard that provides a set of document elements with mapping ability for integrating several standards. For example, UDEF has the mechanism to map the other document standards such as STEP [27] and X12/EDIFACT [33]. The mappable document elements are those synonymous elements such as “part no” in legacy PDM, “product part identifier” in EIA-836, “product/service ID” in X12 EDI and “part number” in STEP AP 203. The merit of UDEF is that it has applied an alphanumeric naming convention defined in ISO 11179 [13] to build document element hierarchy. This enables document elements to be used in a hierarchical way, which resembles the concept tree used in CODEX. Nevertheless, the hierarchical identifiers developed in UDEF are rather ad hoc and are not on a theoretical taxonomy base. This limits the UDEF’s ability in developing a more evolvable and flexible document integration solution. CODEX *IID* not only allows the elements of one document to be run-time classified in an IID concept tree but also enables its annotations to be referenced to many lower level IID-ed vocabularies. This adds the reusability of CODEX vocabularies and leaves the possibility for plugging-in other existing business vocabularies on markets.

Fourth, using existing ontology to map existing document elements is another approach to achieving document interoperability [21]. In this approach, meta-ontology is used to map onto the synonymous document elements from multiple parties with conditions of concept equivalence in different contexts or homonyms in same context. The issue of this approach is that the mapping between meta-ontology and local document elements is static, which cannot cope with the changing requirements of meta-ontology. In CODEX approach, the common concept IID can be compared to the meta-ontology, which maps onto the local concept IID. The key difference is that the formation of common concept IID is a collaboration result, which permits collaborative parties to keep their language different concept definitions but maintain semantic consistency. Second, common concept IID is allowed to change through one-way buffer mechanism. The mapping of local concepts and common concepts is also a collaboration result, which constantly tracking the changes of common document element concepts that may affect the local document element concepts. Thus, the CODEX approach is a flexible, evolvable and accurate approach.

7. CONCLUSION

In this paper, we have addressed the issue of semantic consistency in inter-enterprise business document exchange with the CODEX approach. This approach has first been discussed in a novel 4-layer framework including layers, from bottom to top, of communication, document structure, semantics collaboration and automatic exchange. The communication layer is responsible for document message transport. The document structure layer is to provide document structure, which includes two structure models of business document structure model and collaboration document structure model. The semantics collaboration layer is to conceptualize the general document structure into semantic documents by both P2P and D2F collaboration. The collaboration relationship and the collaboration results are constrained in two XML specifications of XML Business Document (XBD) and XML Collaboration Documents (XCD). The top automatic exchange layer is a user layer responsible for automatically and routinely exchanging XBD business documents, without noticing the lower layers for document integration.

The CODEX approach to semantic consistency maintenance between inter-enterprise documents applies the thought of collaboration to design specific method for maintaining semantic consistency in business document template creation. This is an important contribution because it has eliminated the semantic consistency issue that metadata approaches cannot solve when they involve multiple view designs. Another contribution of CODEX approach is its document structure models based on the layered design thought. These models enable heterogeneous document concepts autonomously created but be uniquely identified and aligned through collaborative concept mapping structures. They resolve the flexibility issue of document standard approaches, where identifiers of document elements are rigidly predefined.

A limitation of CODEX approach is that, like almost all other integration approaches, its ability of integrating non-mappable

document elements of legacy documents is limited to providing the intersected concept similarity. Nevertheless, as a growing system, CODEX has excellent interoperability.

CODEX approach presented in this paper is still evolving. More stringent implementation level evaluation of this approach is required based on the future implementation of collaborative engines and transformation engines. Additional theoretical researches on business document resource classification, web business resource access, and context-based value translation for Multilanguage are needed to tackle the implementation level issues.

8. ACKNOWLEDGEMENTS

The work reported in this paper has been partially supported by University of Macau Research Grand.

9. REFERENCES

[1] Barthes, R., *Mythologies*, Hill and Wang, English version 1972.
[2] BizTalk, http://www.BizTalk.org.
[3] CONEX, http://www.cit.gu.edu.au/~jzguo
[4] cXML, http://www.cxml.org.
[5] ebXML, http://www.ebxml.org.
[6] ecl@ss, http://www.ecl@ss.de.
[7] Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M. and A. Flett, "Product Data Integration in B2B E-Commerce", *IEEE Intelligent Systems*, July/August 2001, 54-59.
[8] Gudgin, M., Hadley, M., Mendelsohn, N., Morear, J. and H. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation 24 June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624.
[9] Guo, J. and C. Sun, "Context Representation of Product Data", *ACM SIGEcom Ex-changes*, Vol. 4, No. 1, 2003, pp. 20-28.
[10] Guo, J. and C. Sun, "Context Representation, Transformation and Comparison for Ad Hoc Product Data Exchange", in: *DocEng'03: Proc. of the 2003 ACM Symposium on Document Engineering*, ACM Press, 2003, pp. 121-130.
[11] Guo, J., Sun, C. and D. Chen, "Transforming heterogeneous Product Concepts through Mapping Structure", in: *Proc. of the 2004 Int'l Conf. on Cyberworlds (CW 2004)*, IEEE Computer Society Press, Tokyo, Japan, November 18-20, 2004, pp. 22-29.
[12] Guo, J., Sun, C. and D. Chen, "Articulating Autonomously Distributed Electronic Product Catalogues for Constructing Dynamic CONEX Net", in: Proc. of IEEE Int'l Conf. on E-Commerce Technology for Dynamic E-Business, IEEE Computer Society Press, 2004, pp. 118-121.
[13] ISO 11179, http://metadata-standards.org/11179/.
[14] Keller, A. M., "Multivendor Catalogues: Smart Catalogues and Virtual Catalogues", *EDI Forum: Journal of Electronic Commerce* 9(3), 1996, pp. 87-93.
[15] Lagoze, C., "Keeping Dublin Core Simple: Cross-Domain Discovery or Resource Description?", D-Lib Magazine 7(1), 2001.
[16] Landry, P., "Multilingual Subject Access: The Linking Approach of MACS", *Cataloging & Classification Quarterly*, Vol. 37, No. 3/4, Haworth Information Press, 2004, pp. 177-191.
[17] Medjahed, B., Benetallah, B., Bouguettaya, A., Ngu, A. and A. Elmagarmid, "Business-to-business Interactions: Issues and Enabling Technologies", The VLDB Journal, 12, 2003, pp.59-85.
[18] Morschheuser, S. and H. Raufer, "Integrated Document and Workflow Management Applied to the Offering Processing of a Machine Tool Company", in: Proceedings of COOCS, Milpitas, CA, USA, 1995, ACM Press, pp.106-115.
[19] Obrst, L., Wray, R. and L. Howard, "Ontological Engineering for B2B E-Commerce", in: *Proceedings of ACM FOIS'01*, Ogunquit, Maine, USA, October 17-19, 2001, pp. 117-126.
[20] Omelayenko, B., "Integrating Vocabularies: Discovering and Representing Vocabu-lary Maps", in: *Proc. of The Semantic Web – ISWC 2002*, Horrocks and J. Hendler (Eds), LNCS 2342, Springer-Verlag Berlin Heidelberg, 2002, pp. 206-220.
[21] Omelayenko, B., Fensel, D. and Bussler, C., "Mapping Technology for Enterprise Integration", in: *Proc. of the 15th Int'l FLAIRS Conference*, Pensacola, FL, USA, 2002, pp. 419-424.
[22] Powell, A., Nilsson, M., Naeve, A. and P. Johnston, "DCMI Abstract Model", Dublin Core Metadata Initiatives, http://dublincore.org/documents/2005/03/07/abstract-model/.
[23] Robinson, M. and L. Bannon, "Questioning Representations", in: *Proceedings of ECSCW'91*, Amsterdam, 1991, pp. 219-233.
[24] Rogoski, R., "Knowledge workers top company assets", Triangle Business Journal, 14 (19), January 8, 1999, p 21.
[25] RosettaNet, http://www.rosettanet.org.
[26] Saussure, F., Course in General Linguistics, McGraw-Hill Book Company, 1966.
[27] STEP, http://www.steptools.com/library/standard/.
[28] Shen, H. and C. Sun, "Flexible Notification for Collaborative Systems", in: Proc. of CSCW'02, ACM Press, 2002, pp. 77-86.
[29] Sun, C., Jia, X., Zhang, Y., Yang, Y. and D. Chen, "Achieving Convergence, Causality, Preservation, and Intention Preservation in Real-Time Cooperation Editing Systems", *ACM Transactions on Computer-Human Interaction* 6(1), 1998, pp. 63-108.
[30] UDEF, http://www.udef.org.
[31] United Nations "EDI for Administration, Commerce, and Transport (EDIFACT)", 1987, http://www.unece.org/trade/untdid/welcome.htm, accessed 15 April 2003.
[32] UNSPSC, http://www.unspsc.org.
[33] X12, "EDI (Electronic Data Interchange) ANSI X12", http://www.x12.org.