Achieving Transparent Integration of Information, Documents and Processes

Jingzhi Guo
Faculty of Science and Technology, University of Macau,
Av. Padre Tomás, Pereira, S.J., Taipa, Macau. Tel: +853 3974890 Email: jzguo@umac.mo

Abstract

While progress is made in the individual integration of business information, documents and processes, the issue of how to vertically integrate these individually integrated results needs to be resolved. This paper proposes a novel TRANSCODE approach to the issue. The approach represents business information, business documents and business processes in three TRANSCODE Structures, which are then implemented, conceptualized and reified in an integrated and flexible 3-layer TRANSCODE Model.

1. Introduction

Business interoperation is an important research topic in electronic business [4], and has been studied in the integration fields of heterogeneous business information [2], heterogeneous business documents [5], and heterogeneous business processes [6]. These researches all concern a non-trivial issue, that is, two business entities are difficult to interoperate with each other to fulfill their shared task due to the autonomous and heterogeneous computing environments. Traditionally, solutions to this problem are individually resolved in different levels. While all these solutions contribute to their individual research realms, an interesting question is asked: how the integrated business information could be effectively used in business documents and how the interoperable business documents should be effectively utilized in business processes?

An integrated solution to integrating business information, documents and processes is important [5]. It can increase the reuse of existing business integration results and save labor costs in business reengineering.

This paper aims to propose a novel TRANSCODE approach to vertically integrate business information, documents and processes. It is *transparent* because it allows the integrated business information to be openly used in business document integration, which further to be openly utilized in business process integration. Through this approach, the reuse of integrated results is available.

Contributions of the paper are: (1) TRANSCODE representation, which represents information, documents and processes of businesses in three independent domains but could be transparently referenced, and (2) a three-layer TRANSCODE model, in which each domain could be autonomously designed in its own way.

In the following, Section 2 depicts TRANSCODE representations. Section 3 presents a 3-layer TRANSCODE model for vertical integration solution. The final section summaries the paper and provides the future work. In addition, to compensate for the length limitation of this paper, a longer article can be found in www.sftw.umac.mo /~jzguo/pages/pub/Transcode.pdf.

2. TRANSCODE Representation

This section introduces the TRANSCODE representations to represent correlated integration domains of information, documents and processes of businesses.

Business information is the fundamental information of a company such as products, assets, people and organization. It specifies the *basic knowledge* of a company. It has the following characteristics:

- Unit of concept
- Hierarchically divisible
- Uniquely identifiable
- Strongly grouped
- Possibly reified
- Strongly typed reification
- Numeric value scalar
- Numeric value unit
- Conversion functions for scalar and unit

Definition 1 (*BusinessInformationDomain*): *BID*

Given a *businessinformationdomain* *BID*, then *BID* is a tuple *BID* = (*C*, *V*, *AN*, *IID*, *G*, *CO*, *VAL*, *DT*, *CVT*), where:

C is set of concept structure symbols.

V is a set of meaningful concept vocabularies such that *V* takes *C* as its form (i.e. *C* is syntactic structure) and *V* is the meaning conveyed in *C*. The *V* consists of the vocabularies of product information *R*, business documents *D*, business processes *P*, organization resources *O*, and other vocabularies V_1, V_2, …, V_n such that $V = \{R, D, P, O, V_1, V_2, \ldots, V_n\}$.

AN $\subset C$ is the symbol of annotation (i.e. denotation).

IID $\subset C$ is the symbol of unique concept identifier such that $AN \xrightarrow{\text{determine}} IID$.

G $\subset C$ is the symbol of concept group.

CO $\subset C$ is the symbol of connotation.

VAL $\subset C$ is the symbol of concept value structure paired with *C* such that *C* takes *VAL*, notated as $C \rightarrow VAL$.

DT $\subset VAL$ is the symbol of data types of values.

CVT $\subset VAL$ is the symbol of *conversion* functions for scalars, units and numeric values.

C is said to *be implemented* to convey a specific concept vocabulary $V_i \in V$ iff *C* is a tuple *C* = (*AN*, *IID*, *G*, *CO*,

IEEE International Conference on e-Business Engineering (ICEBE'06)
0-7695-2645-4/06 $20.00 © 2006 IEEE

VAL, *DT*, *CVT*) such that $V_i \xrightarrow{\text{is assigned to}} C$, notated as $V_i = C(AN, IID, G, CO) \rightarrow VAL(DT, CVT)$, where:

- *C*(*AN*, *IID*, *G*, *CO*) is called an *implemented conceptstructure* on *V*, simply notated as *C*.
- *VAL*(*DT*, *CVT*) is called an *implemented concept valuestructure* for *C*(*AN*, *IID*, *G*, *CO*), simply notated as *VAL*.

- An instance of an implemented concept structure $c \in C$ is a *conceptualization* of *C* iff all *an*, *iid*, *g*, $co \subseteq c$ respectively take their particular values such that $an \rightarrow value$, $iid \rightarrow value$, $g \rightarrow value$, $co \rightarrow value$ such that *c*(*value*(*an*), *value*(*iid*), *value*(*g*), *value*(*co*)), where $iid \in IID$, $an \in AN$, $g \in G$ and $co \in CO$. A conceptualization *c* of *C* is called as a *concept*, which is a concept in a vocabulary V_i such that $c \in V_i$.

- When all $c \in V_i$ is classified through their *iid*s on the vector concept tree (1, *i*, …, *i*) [1], we say V_i is a *classifiedvocabulary* .
- Recursively, if all $V_i \in V$ is classified through V_i(*value*(*iid*), *value*(*an*), *value*(*g*), *value*(*co*)) on the vector concept tree (1, *i*, …, *i*), we say *V* is a resource tree in the *BID* domain.

- A concept *c* = *c*(*value*(*an*), *value*(*iid*), *value*(*g*), *value*(*co*)) is *reified* iff its paired implemented concept value structure $val \in VAL$ is *instantiated* as *val*(*value*(*dt*), *value*(*cvt*)), and the *val* takes a particular value *value* such that $c \rightarrow value(val) \rightarrow value$.

For example, a piece of specifically conceptualized and reified business information (i.e. a reified concept) can be in the form of *c*(r.52.14.15.1.3.1, currency of price, scalarType, 0)→*val*(string, *Currency*)→ USD.

Business document is a composite concept of many business concepts, such as purchasing order. It specifies the *compositeknowledge* of a company such that how a document concept is a composed from multiple vocabularies. It has the following characteristics:

- Unit of concept composition
- Uniquely identifiable document elements
- Hierarchically arranged document elements
- External concept referenced
- Computing function
- Document concept vocabulary

Definition 2 (*BusinessDocumentDomain*): *BDD*

Given a business document domain *BDD*, then *BDD* is a tuple *BDD* = (*DOC*, *D*, *T*, *E*, *EV*, *IID*, *AN*, *G*, *CO*, *RID*, *VAL*,*DT* , *FN*), where:

DOC is document structure symbol.

D is a document vocabulary such that for any particular document name $d \in D \in V \in BID$, $doc \in DOC$ as structure conveys the meaning of $d \in D$.

T is document type symbol for specifying that the document is either conceptualized or reified.

E $\subset DOC$ is element concept structure symbol.

EV is a set of meaningful element vocabularies such that *EV* takes *E* as its syntactic structure and *EV* is the meaning conveyed in *E*.

IID, *AN*, *CO*, $G \subset E$ are the symbols of document element concept identifier, annotation, connotation and concept group.

RID $\subset E$ is external concept identifier symbol referenced to the external concepts such that $RID \rightarrow IID$.

VAL $\subset E$ is the concept value structure symbol of document element.

DT ,*FN* $\subset VAL$ are symbols of data types and computing functions.

E is said to be *implemented* to convey a particular document element vocabulary $EV_i \subseteq EV$ iff *E* is a tuple *E* = (*IID*, *AN*, *CO*, *G*, *RID*, *VAL* , *DT* , *FN*) such that $EV_i \xrightarrow{\text{is assigned to}} E$, notated as $E(IID, AN, CO, G, RID) \rightarrow VAL(DT, FN)$, where:

- *E*(*IID*, *AN*, *CO*, *G*, *RID*) is called *implementedelementconcept* structure on EV_i, simply notated as *E*.
- *VAL*(*DT*, *FN*) is called the implemented element value structure for *E*(*IID*, *AN*, *CO*, *G*, *RID*), simply notated as *VAL*.

DOC is said to be *implemented* to convey a particular document *D* iff *DOC* is a tuple *DOC* = (*IID*, *AN*, *T*, *E*), notated as *DOC*(*IID*, *AN*, *T*, *E*), where *IID* is the symbol of document concept identifier $IID \subset D$ and *AN* is denotation of document.

- An implemented element structure $e \in E$ is a *conceptualization* of *E* iff *e*(*value*(*iid*), *value*(*an*), *value*(*co*), *value*(*g*), *value*(*rid*)), where $iid \in IID$, $an \in AN$, $co \in CO$, $g \in G$, $rid \in RID$. This *e* is called as *documentelement concept*.

- An implemented document structure $doc \in DOC$ is *conceptualized* iff $\forall e \subseteq E$ is conceptualized, and *IID* is instantiated to a particular $iid \in d \in D$, and *T* take a particular value "template" such that *doc* = *doc*(*value*(*iid*), *value*(*an*), "template", {*e*}) This *doc* is called a *document template*.

- An *e* is *reified* iff *e*(*value*(*iid*), *value*(*an*), *value*(*co*), *value*(*g*), *value*(*rid*)) → *val*(*dt*, *fn*), where $val \in VAL$, $dt \in DT$ and $fn \in FN$.

- A document template *doc* is *reified* iff $\forall e \subseteq doc$ are reified and *T* takes the value "instance" such that *doc* = *doc*(*value*(*iid*), *value*(*an*), "instance", $\{e \rightarrow val\}$).

For example, a simple conceptualized PurchaseOrder document template can be:

```
doc(iid="d.1.2"an="purchaseorder"t=   "template")(
   e(iid="e.1",an="Shi pTo",co="many", g="address",
       rid="addr345"),
   e(iid="e.2",an="items",  co="many",g="product",
   rid="prod23"))
```

where document term *d* is (*iid*: d.1.2 *an*: purchase order) and *rid=addr345* and *rid=prod23* point to the address

concept and product concepts defined in BID domain for users to reify the document in reification time.

Business process is a sequence of conditional operations on a set of business documents. It dynamically specifies the intra- and inter-activities of organizations as *activitypatternknowledge* [3]. Given a set of documents, a conditional operation on one document in different context may produce different resulting documents and trigger different conditional operations on them. These triggering conditions constitute different activity patterns between heterogeneous semantic communities and are the issue of *businessprocessinteroperation* . For example, an operation SendQuote may send QutationSheet to receivers, where some may trigger operation ReceiveQuote if they understand the incoming SendQuote on QuotationSheet and some may simply ignore it if not.

This following devises the *documentbased* business process in a *businessprocessdomain* (BPD).

Definition 3 (*BusinessProcessDomain*): *BPD*

Given a business process domain *BPD*, the *BPD* is a tuple *BPD* = (*PROC*,*P* , *O*,*IID*, *AN*, *VIS*, *DID*, *SND*, *RCV*, *S*, *LID*, *LOGIC*, *COND*, *DV*), where:

PROC is process structure symbol.

P is a process vocabulary such that for any particular process name $p \in P \in V \in BID$, $proc \in PROC$ as structure conveys the meaning of *p*.

O $\subset PROC$ is process operation structure symbol.

IID, *AN*, *VIS*, *DID*, *SND*, *RCV*, *S*, *LID* $\subset O$ are the symbols of identifier (*IID*), annotation (*AN*), and operation visibility (*VIS*) of the operation *O*, document identifier in processing (*DID* = *IID* of $DOC \in BDD$), sender's address (*SND*), receiver's address (*RCV*), process operation status (*S*), and proposed document processing logic identifier (*LID*), where visibility *VIS* has the status such as "public", "private" and "partner" to restrict the nature of the operation *O*, and process operation status *S* has status of "arrived", "acknowledged", "processed" and "sent".

LOGIC is the symbol of a document processing logic identified by *LID*, in the form of a computing logic.

COND $\subset LOGIC$ is the symbol of document processing conditional result.

DV is the symbol of conditional value of *COND*.

- An process operation *O* is said to be *implemented* iff *O* is a tuple *O* = (*IID*, *AN*, *VIS*, *DID*, *SND*, *RCV*, *S*, *LID*), notated as *O*(*IID*, *AN*, *VIS*, *DID*, *SND*, *RCV*, *LID*) where *DID* identifies the incoming document and *LID* identifies processing logic *LOGIC*.

LOGIC is said to be *implemented* iff *LOGIC* is a tuple *LOGIC* = (*LID*, *DID*, *COND*, *DV*, *O*) such that $LOGIC(LID, DID, COND(DV)) \rightarrow O$, where *O* is the outgoing process operation.

PROC is said to be *implemented* iff *PROC* is a tuple *PROC* = (*IID*, *AN*, *O*) such that for (*IID*, *AN*) $\in P$, *PROC*(*IID*, *AN*, *O*).

- An implemented process operation $o \in O$ is said to be *conceptualized* iff all its elements are conceptualized such that *o*(*value*(*iid*), *value*(*an*), *value*(*vis*), *value*(*did*), *value*(*snd*) = EMPTY, *value*(*rcv*) = EMPTY, *value*(*s*) = EMPTY, *value*(*lid*)). This *o* is called as *processoperation concept*.

- An implemented business process $proc \in PROC$ is said to be *conceptualized* iff all $o \in O$ of *proc* is conceptualized. This *proc* is called as *processtemplate* .

- A conceptualized process is said to be *reified* iff one of its operation is triggered to process an incoming document and accordingly changes its status *S*.

For example, a conceptualized business offer process may include the following four process operations:

proc(iid="p.3" , an ="offerprocessing")
o(iid="p.3_1",an="RequestOffer",vis="public",
did="InquirySheet",snd,rcv,s,lid="processInquiry"),
o(iid="p.3_2",an="ProcessOffer",vis="private",
did="ReceivedInquiry",snd,rcv, s,lid="processOffer"),
o(iid="p.3_3",an="ProveOffer",vis="private",
did="UnprovedOffer",snd,rcv,s,lid="proveOffer"),
o(iid="p.3_4",an="MakeOffer",vis="public",
did="ProvedOffer",snd,rcv,s,lid="makeOffer") .

where each operation has an operation logic identified by *lid* to process the incoming business document identified by *did*. The processing triggers a forward operation in the sequence and produces an outgoing document.

In next section, we will describe the sharing relationship between the domains of BID, BDD and BPD in a tree-layer TRANSCODE model.

3. Three-Layer TRANSCODE Model

The three-layer TRANSCODE Model shown in Fig. 1 describes the knowledge sharing relationship, and states how the integrated business information can be shared in business document integration, which further be shared in business process integration. The key to understanding the Model is *structure*, *concept*, and the *relationship* between structure and concept [2].

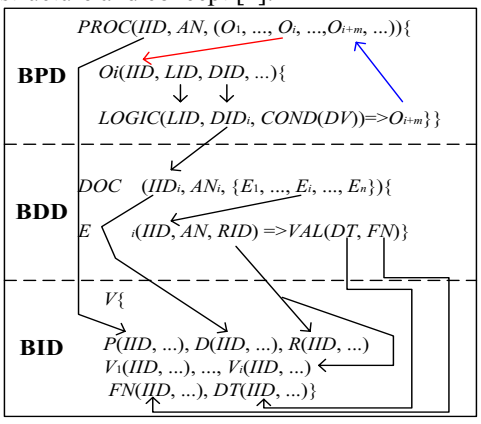

Fig. 1: A three layer TRANSCODE model

The Model consists of three layers. The bottom layer is *business information domain* (BID), where basic knowledge of business information is designed in a *vocabulary*

tree V, which consists of *concept vocabularies* (R, D, P, V_1, …, V_i, …, V_n) of products, documents, processes and others.

The middle layer is the layer of *business document domain* (BDD), where the composite knowledge of business documents is designed in a set of *business document templates* such that $\forall doc \in DOC \subseteq$ BDD. Each template *doc* is identified by a document concept identifier $iid \in d \in D \in BID$, and consists of a set of document elements $\{e\} \subseteq E$. Any *e* has an element identifier $iid \in e$ and an external concept identifier $rid \in (r \cup o \cup v_i)$ ($r \in R$, $o \in O$ and $v_i \in V_i$ in BID) that semantically defines the meaning of element identifier such that $rid \rightarrow iid$. Thus, both documents and document elements reuse the already-defined concepts of vocabularies. However, these reuses are independent of document templates *doc* (*value*(*iid*), …) (e_1(*value*(*iid*), *value*(*rid*), …), …, e_i(*value*(*iid*), *value*(*rid*), …)), where each of them is conceptualized from the given document structure *doc*(*iid*, …)(e_1(*iid*, *rid*, …), …, e_i(*iid*, *rid*), …) $\in$ *DOC* in BDD.

The top layer is the layer of *business process domain* (BPD), where the activity pattern knowledge of business processes is designed in a set of *business process templates* such that $\forall proc \in PROC$. Each template *proc* reuses but is independent of the integration of documents

The above Model utilizes concept identifiers generated on *vector concept tree* (1, *i*, …, *i*) [1] to realize the sharing and integration of business information, business documents and business processes. It provides the features of the high reuse of individual integration results in BID and BDD and the flexibility of independent design of business information, documents and processes.

4. Conclusion

This paper has proposed a novel TRANSCODE approach to resolve the issue of the vertical integration of business information, documents and processes, where the most existing integration solutions only focus on individual aspect of the above. This approach firstly represents business information, documents and processes in three independent structures, and then aligns these structures in a 3-layer TRANSCODE Model. In this Model, business information domain provides basic knowledge of business organizations, business document domain provides composite knowledge of business document templates, and business process domain provides activity pattern knowledge of business process templates. The lower layer domain knowledge is reused and integrated into higher layer domain through unique concept identifiers. To test the feasibility of TRANSCODE Model, the Model has further been implemented in three XML specifications - XBI, XBD and XBP (*see* www.sftw.umac.mo/~jzguo/pages/ TRANSCODE_Specification.html). The examples there have demonstrated that the abstract TRANSCODE Model can be implemented, and the concept identifiers IID is an effective vehicle for semantically linking lower layer concepts with higher layer concepts for reuse.

In this paper, an important methodology for business integration is implied, that is, the separation of structure from concepts, and the separation of concepts from reification. The separation enables the flexible design and use of business integration systems.

This paper is a mature part of the ongoing research project for globally integrating semantically heterogeneous business information, business documents and business processes. It has built a core representation limited to a set of homogeneous business organizations. Some future work includes mapping the core representation onto ad hoc heterogeneous representations, the creation of conversion function library, and the contextual value translation between different natural languages.

5. Acknowledgements

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

6. References

[1] Guo, J. and C. Sun, "Context Representation, Transformation and Comparison for Ad Hoc Product Data Exchange", in: *Proc. of the 2003 ACM Symposium on Document Engineering*, ACM Press, 2003, pp. 121-130.

[2] Guo, J., Sun, C. and Chen, D., "Deconstruction and Reconstruction of Heterogeneous Electronic Product Catalogues for Semantic Interoperation", in*: Proc. of the 2004 IEEE Conf. on E-Commerce Technology* (*CEC'04*), IEEE Computer Society Press, 2004, pp. 333-336.

[3] Manheim, M. and M. Fritz, "Information Technology Tools to Support Virtual Organization Management: A Cognitive Informatics Approach", *Organizational Virtualness*, Sieber, P. and J. Griese (Eds.), 1998, pp. 137-153.

[4] 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.

[5] Omelayenko, B., Fensel, D. and Bussler, C., "Mapping Technology for Enterprise Integration", in: *Proc. of the 15th Int'l FLAIRS Conf.* , USA, 2002, pp. 419-424.

[6] Wombacher, A., Fankhauser, P., Mahleko, B. and E. Neuhold, "Matchmaking for business processes based on conjunctive finite state automata", *Int. J. Business Process Integration and Management* , Vol. 1, No. 1, 2005, pp. 3-11.