The Computational Group Concepts in Business Document Exchange

Jingzhi Guo and Xin Guan
Department of Computer and Information Science, University of Macau
Av. Padre Tomás, Pereira, S.J., Taipa, Macau, Tel: +853-397 4890
E-mail:{jzguo, ma46601}@umac.mo

Abstract

Computational group concepts (e.g. Total = Quantity × Unit Price) in business document exchange are very important concepts. However, semantic consistency issue exists in document exchange such that how the computational relationship in a computational group concept can be represented, conceptualized, reified and interpreted between heterogeneous business document contexts. This paper has discussed this issue and proposed a public operation concept strategy as the solution to representing, designing and implementing the computational group concepts.

1. Introduction

In the business document exchange, there is a computational group concept that is often not well understood. What is a computational group concept? Simply, it could be a sum of product value or a formula concept value that cannot be separately discussed. For instance, given a "price" concept that is exchanging, how should its member concepts of currency, value and unit be processed such that the currency or unit change will be reflected in the change of the amount of value?

Indeed, in the real world of business exchange, the computational group concept is complex. First, when a firm creates a business document template, some concepts have to be grouped and their computational relationships have to be explicitly represented. For example, a blank invoice has a "total amount", which possibly represents the sum of the sub-totals of all involved product value. Second, given that we have built the "total amount" group and represented the computational relationships between its group members in a document template, how should the group concept "total amount" be reified at sender's side when instance data are provided? Third, when a real invoice is sent to the business partner, how could the recipient interpret the received invoice and regenerate it in a semantically consistent way using the recipient's local invoice template?

Granted the above complexity, we are motivated to investigate the details of computational group concepts for the phases of business document representation and exchange.

It is noticed that the autonomy of business document exchange has an important effect on how to handle computational group concept. For example, if the senders and receivers of the business documents are sitting in a same context and using the same document application, the operations on the group concepts may possibly be designed and implemented in a same library attached to the document application. This is similar to the standard XML schema implementations [2], where default functions such as "ID", "DATE", "TIME", etc. are included in the schema implementation. For a single user application like Microsoft Excel, locally available operations are well sufficient. However, when senders and receivers are in different contexts and are autonomous, i.e. they have different document templates and reification methods, a single "one-fit-all" operation library is often not feasible.

In this paper, we assume that senders and receivers of the business documents are in different business contexts. The heterogeneity assumption poses a great challenge to handle computational group concepts, where operations on group concepts cannot be designed and implemented in a single document application due to the heterogeneous nature of involved business documents.

This paper aims to propose a strategic solution, called *Public Operation Concept* (POC), to the issue of handling computational group concepts between heterogeneous business documents of distributed contexts.

In the following, Section 2 provides examples to introduce problems. Section 3 proposes the strategies for problem solving. Section 4 discusses the business model implied in the strategy. Section 5 details the design process of computational group concepts. Section 6 implements the POC approach. Section 7 discusses and concludes the paper with the highlight of contributions.

2. Examples, Problems and Analysis

To introduce the problem we are concerning, we first present two real invoices shown in Fig. 1 in English (say Inv1) and Fig. 2 in Chinese (say Inv2), which both are electronically stored but printed out in different presentation styles. We suppose that our task is to build a global invoice exchange system between distributed and heterogeneous business document contexts disregarding

0-7695-3003-6/07 $25.00 © 2007 IEEE
DOI 10.1109/ICEBE.2007.15

their natural languages used, presentation style formats, and data extraction purposes.

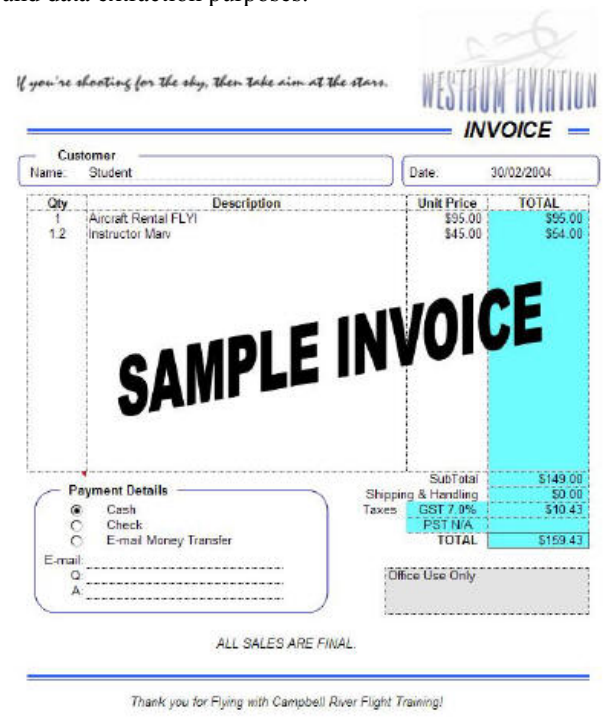

Fig. 1: Electronic Invoice Sample in English

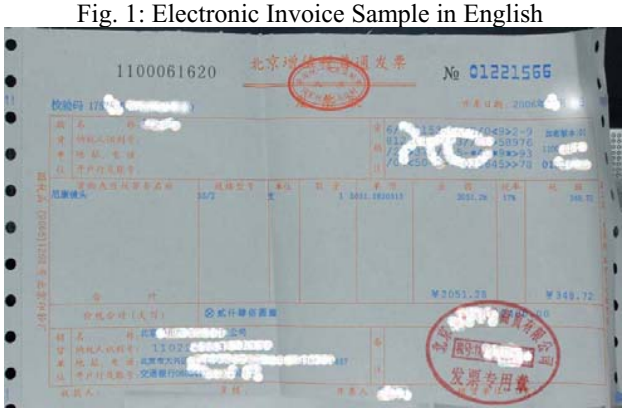

Fig. 2: Electronic Invoice Sample in Chinese

2.1. Computational Group Concepts in Examples

There are several computational group concepts in Fig. 1 and Fig. 2, for example, "payment detail", "(Product) Total or 金额", "SubTotal", "Taxes or 税额", "Total or 总额" and so on. In this paper, as an investigation illustration, we are interested in this type of computable group concepts. These concepts can often be represented by a computational formula, for example, "(Product) Total" = Qty × UnitPrice for all Description,, SubTotal = Sum("(Product) Total") and Total = SubTotal + Shipping and Handling + Taxes.

2.2. Problems

Through comparing Fig. 1 (Inv1) and Fig. 2 (Inv2), several problems can be identified:

(1) Inv1 and Inv2 have different computational group concepts, while they are the same in "(All Product) Total" = 金额 and SubTotal = (货物)金额, the other group concepts are different.

(2) In Inv1, Taxes = GST rate × 7.3% while in Inv2 Taxes (税额) = VAT rate × 17%. They have different tax categories and so the "Taxes" means differently.

(3) Inv1 has Shipping & handling while Inv2 does not include this item.

(4) Inv1 has Total = SubTotal + Shipping & Handling + Taxes while Inv2 has Total (价税合计) = (All Product) Total (金额) + 税额. These two group concepts are different.

By summarizing the above four problem phenomena, we can generalize a significant issue, that is, group concepts in heterogeneous document contexts have *inconsistent group concept composition* problem such that they may be homonyms or synonyms, have different formulas or may be orphan concepts only appeared in either senders or receivers. This issue leads to impossibility for business document exchange with correct interpretation between senders and receivers.

2.3. Analyses

Given the fact that Inv1 and Inv2 are generated in two different business contexts that are in different natural language environments and heterogeneous invoice applications, there are two possible solutions to resolve the inconsistent group concept composition problem.

(1) Set a mandatory rule that all invoice users apply the same invoice application with the same invoice format, semantic terms (including group concepts) and computational formula [1, 3].

(2) Build a mapping mechanism such that for all semantic terms used in Inv1, they can be semantically transformed into a new invoice that can be understood in the context of Inv2 [4, 8].

It is obvious that solution 1 is inappropriate in our problem domain because the invoice exchange between contexts of Inv1 and Inv2 requires a universal invoice standard and application [5]. It is not possible in reality. The solution 2 is worth considering. However, if this solution is adopted, some other problems have to be resolved:

(1) How to guarantee the group concepts in Inv1 and Inv2 are semantically consistent such that their computational relationships are exactly the same as in Inv1 and Inv2?

(2) How to guarantee that the recipient of Inv1 in Inv2 context correctly interpret the Inv1 such that the received value for regeneration can be validated?

In this paper, we adopt solution 2 and attempt our best effort to solve the above problems.

3. Strategy of Business Document Exchange

We propose a novel business document exchange strategy, called *Public Operation Concept* (POC), to resolve the problems. Our strategy can be described in the following steps:

Step 1: collaboratively build common vocabularies for all involved parties, including all possible document element concepts, such that for each language different zone *i* or *j*, there is a corresponding common vocabulary cv_i or cv_j where cv_i semantically equals cv_j. With this step, document senders and receivers could have mutually interoperable semantic knowledge on terms *cv* used in the exchanged documents.

Step 2: collaboratively create common document templates for all involved parties using common vocabularies *cv*. With this step, both document senders and receivers have common document templates that are semantically interoperable.

Step 3: all possible computational group concepts are marked and default implementations are given for each group concept. The default implementation for each computational group is a common operation, which is called as a *common operation concept* that has a unique operation identifier. All the common operation concepts are published to a publicly available space with a unique web address, which is used as their namespace. The namespaces are used as the entry points to the access of computational group operations.

Step 4: each involved party creates *local document templates* according to the common document templates to generate their differentiated business documents and to maintain semantic consistency.

Step 5: for each computational group concept in local document templates, it can use common operation concepts or override them. The overridden operation concepts are called *local operation concepts* and are placed in another namespace of the individual party.

Step 6: when a document sender reifies a local document template, the computational concepts are reified by the given operation concepts.

Step 7: For a received business document, the receiver

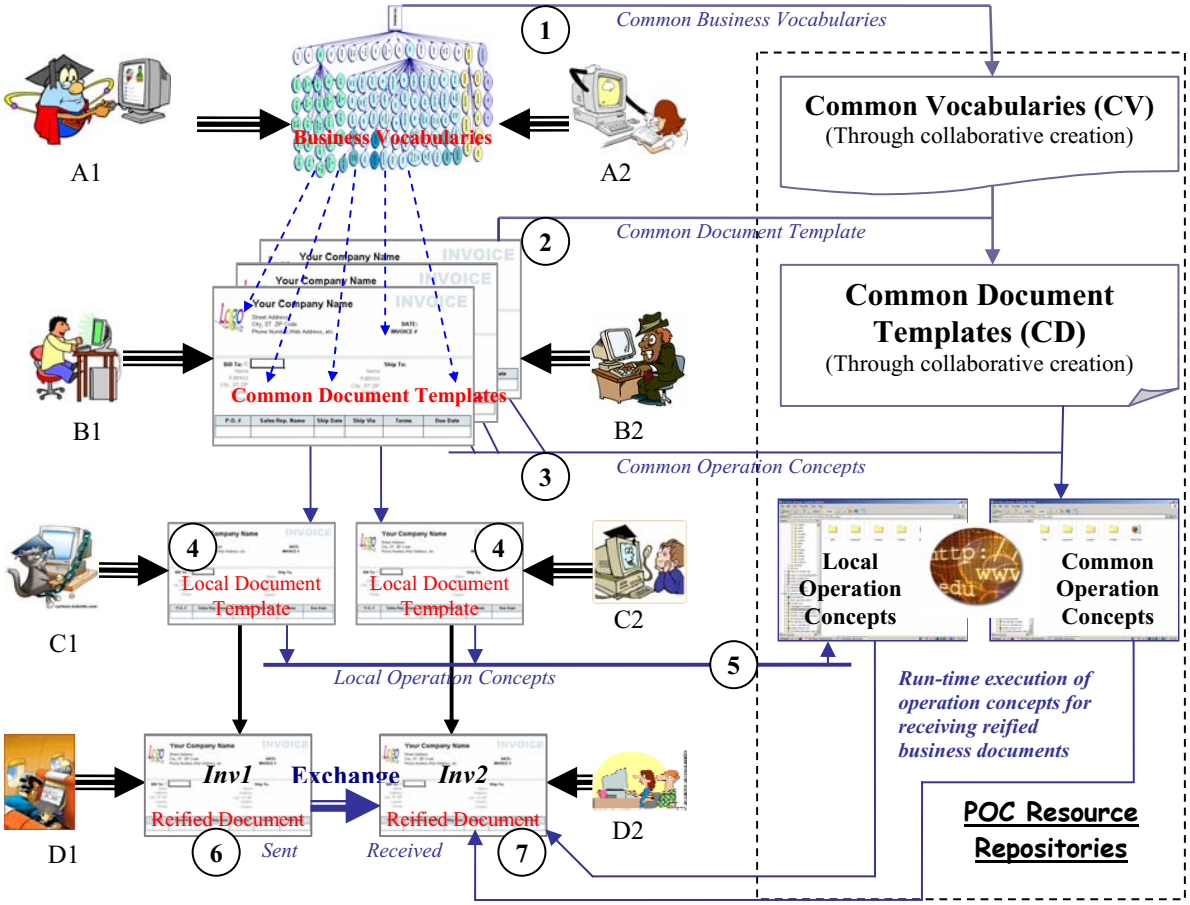

Fig 3: Illustration of Creating and Editing Group Concepts

re-generates the incoming document according to the receiver's local document template and validates the computational group concept value based on both operation concepts given by the operation namespaces.

This strategy is based on CONEX (for Step 1) [6] and CODEX (for Step 2) [7]. Step 3 to step 7 are novel, which has introduced a *public operation concept* (POC) approach to retrieve the common operation concepts and local operation concepts through using namespaces for positioning the implementation of the operation concepts and their run-time executions.

To have a better understanding of this strategy, Fig. 3 provides a graphical illustration of the strategy. In the figure, our task is that D1 sends a reified business document (say Inv1, *see our example in* [A7]) to D2 where D2 can fully understand in its own context (say the context of Inv2 with the different group concepts and natural language from Inv1 – *see our example in* [A8]). To fulfill this task, Fig. 3 divides related people in four categories: *common vocabulary (CV) designers* (CVD), *common document (CD) designers* (CDD), *local document (LD) designers* (LDD), and *reified document (RD) users* (RDU). With this classification, we have:

- CVD = (A1, A2), a P2P collaborative community [7] responsible for collaborative creation and editing common business vocabularies CV that can be used in all types of languages with semantic consensus (*see our examples in* [A1, A2]).
- CDD = (B1, B2), a P2P collaborative community [7] responsible for collaborative creation and editing common business document templates (i.e. blank documents) CD that can be used for heterogeneous business contexts but with semantic consensus (*see our examples in* [A3, A4]).
- LDD = (C1 | C2), any local document designer, which collaborates with CDD to form a D2F (dominant-to-follower) collaboration community [7] such that a D2F collaboration organization = (B1, C1) or (B2, C2). The result of D2F collaboration is that C1 or C2 has personalized the common document templates into local document templates LD that fit for its own business context (*see our examples in* [A5, A6]).
- RDU = (D1 | D2), any reified document user. It (e.g. D1) reifies local document templates by filling real concept values into reified document RD and sends them to other one (e.g. D2) for use (*see our examples in* [A7, A8]).

With this strategy, we wish to obtain the clear consistent semantic relationships between CV, CD, LD and RD without semantic discrepancies between CVD, CDD, LDD and RDU such that:

(1) $\forall ec \in$ CD, $ec \in cv \in$ CV such that $cv_i =_{\text{sem}} cv_j$. That is, for all document element concepts *ec* used in one common document template *cd*, they must come from one of common vocabularies *cv* belonging to CV, where any cv_i semantically equals any cv_j.

(2) $\forall ld \in$ LD, $ld \subseteq cd \in$ CD. That is, any local document *ld* is a partial common document *cd* in its natural language scope.

(3) $\forall$*groupConcept* $\in$ *ld*, *groupOperation* $\in$ *Namespace*. That is, for any group concept in a local document template *ld*, its corresponding group concept operation *groupOperation* must be implemented to stipulate the computational relationships between the underlying grouped concepts, and its implementation must be accessible as a public operation through a namespace.

(4) $\forall$*groupConcept* $\in$ *rd*←*ld*, *groupOperation* (*groupConcept*) $\in$ *namespace* $\in$ *ld*. That is, in the reification of local document *ld* to a reified document *rd*, any *groupConcept* be computed according to the predefined *groupOperation* in *ld*. Also, when receiving a reified document, the recipient must interpret the *groupConcept* according to the predefined *groupOperation* following the given *namespace* that implements *groupOperation*.

It is clear that if we can maintain the above relationships, we can guarantee that a sent reified business document can be safely received by a recipient in a semantically consistent way. It should be noted that this strategy does not guarantee that a received reified business document can be immediately interoperable with the recipient's standalone business system because a fully interoperable reified business document implies the following condition: both the sender and receiver has the exactly same document elements, concept grouping and underlying group concept computational behaviors. This requires a reconstruction process on the received reified business document. This reconstruction is beyond the discussion of this paper and should be investigated elsewhere, for example, legacy systems integration.

4. Business Model Implied

The POC strategy for handling computational group concepts in exchanging business documents between distributed and heterogeneous business contexts implies an important business model, which can be described in the following.

- Three independent profit entities are identified, which are common business vocabulary designers (CVD). Given that all CVD are firms, they can organize a profitable P2P community specializing in designing common business vocabularies (CV) (*see our CV examples in* [A1, A2]) and allocate the revenues based on their contributions. The purchasers of the CV are the common document designers (CDD).

- Given all business document designers (CDD) are firms, they can form a profitable P2P community specializing in designing common business documents (CD) and common operation concepts (*see our CD examples in* [A3, A4]). What they have earned can be allocated according to their contributions. The purchasers of CD and operation concepts are local document designers.
- Local document designers (LDD) and reified document users (RDU) combined together are also firms, which actually exchange business documents. They first purchase the CD and operation concepts and then reify them for doing e-business (*see our LD and RD examples in* [A5, A6, A7, A8]).

This labor division and specialization implies a fully new *collaborative business model*, which will enable to vertically integrate existing industries and horizontally integrate heterogeneous e-business systems. In this model, the concept of *public operations* is integrated into the business interoperability activities with the perception of collaboration.

5. Computational Group Concept Design

Assuming that the POC strategy can work in the above mentioned collaborative business model, how could a computational group concept be particularly designed through a public operation concept? In this section, we will describe the POC approach to the design:

(1) In the stage of common document design, each *common group concept* or "cgc" in a common document (CD) is selected from common vocabulary (CV) and designed in a collaborative way for reaching common consensus. Each group concept has a *common operation concept* commonGroupOperation or "cgo", for example, cgo:oid="Oid5_1_5".

(2) In the stage of local document design, each *local group concept* or "lgc" in local documents (LD) is localized from corresponding common document (CD) and its "cgo" maybe personalized as *local group operation* localGroupOperation or "lgo", for example, lgo:oid = "O1234".

(3) In the stage of local document reification and exchange, all computational group concepts must be reified into concept values according to the computation of "cgo" and/or "lgo". The key is the requirement for consuming the public operations of the referenced "cgo" and/or "lgo" defined by namespaces, for example, xmlns:cgo="http://www.conex.em2i.org/papers/grpcpt/cgo.php".

Given the above stage-wise design, we provide the particular technical design of public operation concept.

5.1. Representation

Given a set of concepts C_1 , …, C_n of a common vocabulary VC such that they belong to a computational group concept $C_k \in$ VC in a business document BD, then we have:

```
Ck =
   C[iid, an, rid = "iidck" ct = "group",
     cgo:oid | lgo:oid = "OID"](
      C1 [iid1, ct = "atomicType"]{val1}
       …,
      Cn[iidn, ct = "atomicType"]{Valn}
   ){Valc}
```

In this representation, the group concept $C_k \rightarrow C$ has a set of concept attributes *iid* (group concept identifier), *an* (annotation of the group concept), *rid* (reference concept identifier iid_{ck} that equals to the one that C_k is identified in VC), *ct* (concept type and here as "group"), *cgo:oid* or *lgo:oid* (where *cgo*/*lgo* is the namespace of common/local group operation concepts and *OID* is the unique operation concept identifier/name in the namespace that introduces an operation), and a set of lower level concepts C_1 , …, C_n belong to this group where each of C_1, …, C_n here at least has a concept identifier iid_1, …, iid_n, a concept type *ct* to indicate whether the lower level concept is an atomic type concept or not a group concept, and a concept value Val_1, …, Val_n.

The operation *OID* implements the computational relationships between concepts C_1 , …, C_n belonging to the group concept C_k, such that:

OID(in *iid*[*n*], in *Val*[*n*], out *iid*[*n*], out *Val*[*n*])

where the input is an array of concept identifier *iid* and a corresponding array of concept values *Val* of C_1 , …, C_n and the output is the computational result array of the (*iid*, *Val*) pairs.

5.2. Design

Given the above group concept representation, document templates can be collaboratively designed into different languages, e.g. XML [*] . The collaborative document design is, in fact, a process of collaborative document editing, which relies on collaborative editors – P2P editor for common document editing and D2F editor for local document editing. This paper will not discuss the design of P2P and D2F collaborative editors, as it is beyond the paper scope.

The group operation concepts *cgo:oid* or *lgo:oid* can be implemented in different server-based script languages such as ASP, JSP, PHP or Perl with their Web addresses as the namespaces for *cgo* and *lgo* such that each

[*] Please be noted that since the operation of a computational group concept is only marked as a symbol in XML document and is implemented in a publicly accessible location through XML namespace, the richness of XML expressiveness problem has no effect on the POC strategy.

cgo:OID or *lgo:OID* triggers a group operation OID implemented in the server page. This design implements operation concepts in publicly available Web spaces and can be independently maintained and used without affecting the design and use of document templates and reifications.

5.3. Reification

Reification of a computational group concept refers to a procedure that a computational group concept consists of several related inner concepts that are given values through computation. For example, given a computational group concept:

Total[iid = "1234", rid = "1.2.3.4.5" an = "sum of tax and shipping expenses", ct = "Group", lgo:oid = "O1234"] (tax[iid="1234.1", ct="atomicGroup"] {Val="120"}, shippingExpense[iid="1234.2", ct = "atomicGroup"]{Val = "560"}){Val = "?"}.

Then:

Val(Total) = O1234([in] iid[1], [in] Val[1], [out] iid[0], [out] Val[0]) = 680.

More generally, the reification procedure of group concept can be written in the following:

```
(1)Given a document template:

ec1[iid1, ct=nonGroup](
 ec1.1[iid1.1, ct=group, lgo:oid=O11]{Val1.1}(
   ec1.1.1[iid1.1.1, ct=atomicGroup,
      lgo:oid=O111]{Val1.1.1},
   ec1.1.m[iid1.1.m, ct=atomicGroup,
      lgo:oid=O11m]{Val1.1.m}
 ),
 ec1.m[iid1.m, ct=atomicGroup,
    lgo:oid=O1m]{Val1.m},
 ec1.m+1[iid1.m+1, ct = nonGroup],
 ec1.m+n[iid1.m+n, ct = nonGroup])

(2) Reify the most inner level group concepts
marked by "atomicGroup" (The nonGroup concepts
are neglected) such that:

  //Calculate the value of a group concept
  For all siblings{
    if ec1...i has ct=atomicGroup or ct=Group,
    iid[i-1]=iid1...i and Val[i-1]=Val1...i;
  };
  Val(ec1...i-1)=OID1...i([in]iid[i-1], [in]Val[i-
    1], [out]Val[0]);

  Repeat;

(3) Calculate outer level group concepts in a
recursive way as (2).
```

Fig. 4: Group Concept Reification

Group concept reification provides an automatic computational result for all group concepts within a business document at sender's side.

5.4. Interpretation

When a business document is sent to a recipient, the procedure of group concept interpretation begins. Such interpretation is a re-computation of the received group concepts in a heterogeneous business context, and thus needing to understand how the sender reifies the group concepts for the sent document. The POC approach interprets the received document in the following way:

Step 1: Replacing all document elements of sender's document by the corresponding document elements understood by the recipient.

Step 2: Translate the document element values of sender's document, where atomic constant values, atomic unit value, atomic scalar value and atomic value are translated into the recipient's required contextual values.

Step 3: Validate the computational group concept values, where these values are recomputed according to the group operations.

In this interpretation process, the three steps of group concept re-computation guarantees: (1) heterogeneous document elements are translated conforming to the recipient's document context; (2) unit and scalar values are localized; (3) all computational values of atomic document element values and computational group concept values are re-computed and validated against the given atomic concept operations and group concept operations. This process can be diagrammed in Fig. 5.

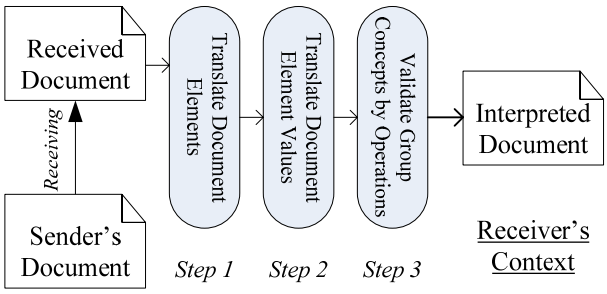

Fig. 5: Group Concept Interpretation

Related to Fig. 3, assuming the sender's document is the Inv1 of D1 and the expected interpreted document is the Inv2 of D2, then the above process will be:

Step 1: For all document elements $ec_1 \in$ Inv1 and $ec_2 \in$ Inv2, then $ec_1 \xrightarrow{translate\ to} ec_2$.

Step 2: For all document element $ec_1 \in$ Inv1 have their corresponding element values ev_1 in the context ctx_1 such that $ec_1 \rightarrow ev_1(\text{ctx}_1)$ and $ec_2 \in$ Inv2 have their context ctx_2, then $ev_1(ctx_1) \xrightarrow{translate\ to} ev_2(ctx_2)$ where $ec_2 \rightarrow ev_2$.

Step 3: For all group concepts $gc_2 \in ec_{2g} \subseteq ec_2$ with corresponding operation concepts go_2, then the group concept value gv_2 of gc_2 is validated by go_2 such that $gv_2 = go_2(ev_{2g})$ where ev_{2g} is the group concept value of ec_{2g}.

6. System Implementation

Computational group concepts are implemented following XML Business Document developed in [7]. The difference is that the operation concepts are added to support the computational group concept design, reification and interpretation.

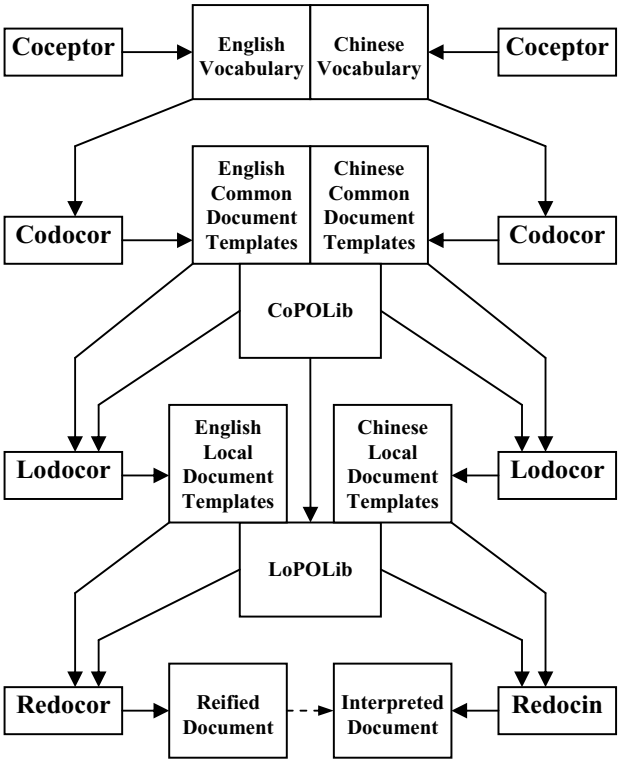

Fig. 6: System Architecture of POC Approach

Particular implementation steps are given in the following:

(1) *Common concept editor* (Coceptor) collaboratively designs common vocabularies in different natural languages in XML format, where common group concepts are designed.

(2) *Common document editor* (Codocor) collaboratively designs common document templates in different natural languages in XML format, where unique identifiers of common operation concepts are designed.

(3) *Common public operation library* (CoPOLib) is a library for common operation concepts implemented in PHP server pages with namespace like cgo:xmlns="URL", where common operations concepts cgo:OID are implemented in the PHP server page addressed by URL. So all operations are publicly accessible through namespace.

(4) *Local document editor* (Lodocor) designs local document templates in a particular natural language. Additionally, *local public operation library* (LoPOLib) is implemented for local operation concepts by overriding common operation concepts if they are not suitable.

(5) *Reified document editor* (Redocor) reifies XML documents so that the reified business documents can be created.

(6) *Reified document interpreter* (Redocin) interprets reified XML document so that the received reified business documents can be interpreted.

The above step-wise implementation of computational group concepts is described in Fig. 6, which is the system architecture of POC approach.

To illustrate how POC components work together to exchange reified documents, consider the sample scenario of sending Inv1 of D1 to D2 as shown in Fig.3. Suppose the understandable contexts of Inv1 and Inv2 are the following tables:

Table 1: Context of Inv1 (abstracted from Fig. 1)

```
Invoice(
Issuing Date;
Invoice Number;
Issuer(Address);
Customer(Address);
Items(
    Qty; Description; Unit Price; Total
    )
SubTotal;
Shipping & Handling;
Taxes(GST);
Total
)
```

Table 2: Context of Inv2 (abstracted from Fig. 2)

```
发票(
开票日期;
发票号;
销货单位(名称; 纳税人识别号; 地址; 电话;
 帐户信息
 );
购货单位(名称; 纳税人识别号; 地址; 电话;
 帐户信息
 );
货物及应税劳务清单(
     货物及应税劳务名称;
     规格型号;
     单位;
     数量;
     单价;
     金额;
     税率(增值税率);
     )
合计(货物总额; 应税总额);
价税合计
)
```

With the above invoice contexts and POC architecture, a reified invoice (say Inv1) generated in the Inv1 context

is able to be received and correctly interpreted in the Inv2 context if the POC approach is well implemented. To describe the invoice exchange process, we have implemented two common vocabularies (an English vocabulary - *see example in* [A1]) and a Chinese vocabulary – *see example in* [A2]), two common invoice templates (both in English – *see example in* [A3] and in Chinese – *see example in* [A4]) including a common public operation concept library, two local invoice templates (*see examples* in [A5, A6] that can be understood by Inv1 context and Inv2 context including their local public operation libraries.

These implementations make possible for a reified invoice produced in Inv1 context and transformed into another reified invoice in Inv2 context such that:

$$\begin{aligned}&\textit{Reified Inv1} \leftarrow \textit{Local English Invoice Template} \subseteq \textit{Common English Invoice Template} \\ &=_{Sem} \textit{Common Chinese Invoice Template} \supseteq \textit{Local Chinese Invoice Template} \\ &\rightarrow \textit{Reified Inv2}.\end{aligned}$$

Comparing with the reified Inv1 produced in Inv1 context (*see example in* [A7]) and the reified Inv1 received and interpreted in Inv2 context (*see example in* [A8]), we can see that heterogeneous concepts of invoices can be freely transformed if they maintain three properties of semantic consistency model proposed in [6], that is, structure mappability, semantic equivalence and common context. Structure mappability is achieved here through concept identifiers. Semantic equivalence between invoice templates is guaranteed through collaborative creation of invoice templates. Common context is provided through concept editors of Coceptor, Codocor, Lodocor and Redocor.

Particularly in this paper, the semantics of computational group concepts are collaboratively assigned in the stage of vocabulary creation, the operations of the group concepts are collaboratively designed in the stage of document template creation. These have guaranteed that the reification of a document can be successful in the sender's context and can be also successfully received and interpreted in the context of the receiver.

7. Conclusion

This paper has investigated the computational group concepts within business documents and their problems in business document exchange between heterogeneous business systems. It has found that semantic consistency between computational group concepts are often neglected because the existing understanding of them are to develop computational functions only applicable to the local applications, not for heterogeneous contexts.

To resolve the above issue, this paper proposed a novel Public Operation Concept approach, which design and build computational operations for the computational group concepts in a publicly available place through applying the namespace concept. Through this approach, computational group concept operations are no longer only tied to the local applications but able to be triggered and used in remote context of document receivers.

This paper has several contributions: the proposal of public operation on computational group concepts, the representation and design of group concepts for business documents, and the demonstration of XML-based implementation.

Future work of the POC approach is to enrich the common operation concept libraries and makes them available for public use.

Acknowledgement

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

References

1. AITF, http://www.iata.org/workgroups/airport-invoice-standards.htm.
2. Biron, P., Permanente, K. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/.
3. BQE Software, http://www.bqe.com/.
4. Chung, C. Y., Lieu, R., Liu, J., Luk, A., Mao, J. and P. Raghavan, "Thematic mapping - from unstructured documents to taxonomies", in: Proc. of CIKM'02, ACM Press, 2002, pp. 608-610.
5. Díaz, L., Wüstner, E. and P. Buxmann, "Inter-organizational document exchange: facing the conversion problem with XML", in: Proc. of SAC'02, ACM Press, 2002, pp. 1043-1047.
6. Guo, J., Integrating Ad Hoc Electronic Product Catalogues through Collaborative Maintenance of Semantic Consistency, PhD Thesis, Griffith University, Australia.
7. Guo, J., Inter-enterprise business document exchange, in: Proc. of ICEC'06, ACM Press, 2006, pp. 427-437.
8. UDEF, http://www.opengroup.org/udef/.

Appendix

A1. http://www.conex.em2i.org/papers/grpcpt/enVoc.xml
A2. http://www.conex.em2i.org/papers/grpcpt/cnVoc.xml
A3. http://www.conex.em2i.org/papers/grpcpt/enCDoc.xml
A4. http://www.conex.em2i.org/papers/grpcpt/cnCDoc.xml
A5. http://www.conex.em2i.org/papers/grpcpt/enLDoc.xml
A6. http://www.conex.em2i.org/papers/grpcpt/cnLDoc.xml
A7. http://www.conex.em2i.org/papers/grpcpt/enRDoc.xml
A8. http://www.conex.em2i.org/papers/grpcpt/cnRDoc.xml