Answering an Inquiry from Heterogeneous Contexts

Jingzhi Guo[1], Zhuo Hu[1], Grigoris Antoniou[2] and Chi-Kit Chan[1]
1. Department of Computer and Information Science, University of Macau
Av. Padre Tomás, Pereira, S.J., Taipa, Macau, Tel: +853-397 4890
E-mail: jzguo@umac.mo
2. Computer Science Department, University of Crete, Greece
Institute of Computer Science, FORTH, Greece, Tel: +30 2810 250166
antoniou@ics.forth.gr

Abstract

In this paper we study the semantic consistency maintenance issue between heterogeneous contexts, that is, how an inquiry from an unknown user of an e-marketplace can be received and answered in a semantically consistent way by a firm that is not in the context of the user's e-marketplace. The proposed solution uses XPM to represent semantically consistent business concepts and adopts defeasible logic to reason with XPM document-oriented business rules for inquiring and offering. We motivate the approach with a real-world apartment rental problem, and explain it in architecture of collaborative business process design and automatic service provision. Finally, we report on an implementation specification within a hybrid human-agent framework.

1. Introduction

In last decade, the design of e-marketplace has experienced a rapid transformation from focusing on front-end web presence of products to emphasizing back-end business interoperation [10]. In this transformation, technologies like business standards (e.g. UNSPSC.org), ontology engineering [5][9] (e.g. OWL [4]) and semantic web [3] have pushed the development of e-marketplace, where sellers can search for buyers and buyers can search for products/sellers to make transaction deals.

In an e-marketplace, a trade process is often a complex process beyond a single search activity but involves a sequence of conditional activities of inquiry, offer, counteroffers, acceptance, and contracting. Contemporary approaches to building a trade process often introduce domain-wide business process standards such as BizTalk, BPML or BEPL that define trade processes using shared business standards of a single domain of business contexts. A challenging situation in real world is: not all firms participating in an e-marketplace adopt a same business process standard, especially SMEs. This implies an issue of heterogeneous business process integration: the numerous business processes of different firms need to be integrated for business interoperation.

The issue can be illustrated in the following example where Carlos, a user of e-marketplace (e.g. alibaba.com), wants to rent an apartment in a city with the requirement shown in Table 1 through his computer agent (e.g. a client software like TradeManager of trademanager.alibaba.com) provided by his participated e-marketplace.

Table 1: Rental Requirements from Carlos

1. Carlos is looking for an apartment of at least 45m^2 with at least 2 bedrooms. If it is on the 3rd floor or higher, the house must have an elevator. Also, pet animals must be allowed.
2. Carlos is willing to pay \$300 for a centrally located 45m^2 apartment, and \$250 for a similar flat in the suburbs. In addition, he is willing to pay an extra \$5 per m^2 for a larger apartment, and \$2 per m^2 for a garden.
3. He is unable to pay more than \$400 in total. If given the choice, he would go for the cheapest option. His 2nd priority is the presence of a garden; lowest priority is additional space.

Immediately, Carlos has several problems:

(1) *Incomplete solution range*. The city that Carlos wants to rent an apartment has more than 500 real estate agents that all provide rental services, but Carlos' agent can only talk to 50 of them because Carlos' e-marketplace only has 50 real estate members. This implies that Carlos has incomplete solution range among all possibilities (Problem 1).

(2) *Insufficient requirement representation*. Carlos' e-marketplace only supports simple inquiry format (e.g. Fig. 1), i.e. Carlos' agent can only provide a restricted inquiry form to Carlos. This form cannot fully represent Carlos' requirements of Table 1. This implies that Carlos has to reduce his requirements in order to make an inquiry through his e-marketplace (Problem 2).

(3) *Long processing time*. Carlos' rental inquiry in web form is posted to 50 real estate agent systems by his e-marketplace. Only 20 of them can automatically analyze and process Carlos' requirement. The others can only read and process manually because their systems lack web form analysis ability. This implies longer waiting time for Carlos (Problem 3).

(4) *Inconsistent semantics between e-marketplaces and their participated systems*. The concepts defined by the e-marketplace in web form may not be understandable by

those systems that have automatic web form processing abilities. For example, the concepts, shown in Fig. 2 corresponding to the web form of Fig. 1, are not interpretable by those real estate agent systems if they are not provided by the e-marketplace. This implies that the semantic inconsistency of used concepts exists between the e-marketplace and the real estate agent systems (Problem 4).

Fig. 1: Example inquiry sheet for renting house

```
<form name="n" id="n" method="get" action="/cgi-bin/rsearch">
    <input name="a" id="a" value="s" type="hidden">
        // What are concepts of "a" and "s"?
    <input name="cu" id="cu" value="fn-rea" type="hidden">
        // What are concepts of "cu" and "fn-rea"?
    <input name="s" id="s" value="qld" type="hidden">
        // What are concepts of "s" and "qld"?
    <input name="ss" id="ss" value="" type="hidden">
    ……
</form>
```

Fig. 2: Concepts used in web form of Figure 1

These four problems illustrate that, without a proper mechanism, Carlos' inquiry cannot be effectively sent to the potential offerers for effective processing. These problems constitute the *semantic consistency issue between heterogeneous business concepts*, which may lead to fewer, wrong or even none of rental offers from the existing real estate agent systems.

This paper aims to reduce the effect of semantic consistency issue by proposing a novel approach to heterogeneous business process integration, where business concepts in terms of facts and rules used in heterogeneous processes can be collaboratively designed by concept designers (i.e. rule makers or knowledge engineers), automatically transformed by automated agents, and easily used by business users. This approach regards a business process as a conditional sequence of automated actions on a set of business documents collaboratively created, where business concepts in terms of facts and rules are made. It is called as *Collaborative Document-Oriented Rule Making* (CoDORM) approach.

Due to space limitations, CoDORM approach only demonstrates how semantic consistency can be maintained in Carlos' example by answering the rental inquiry, and how this inquiry and its reasoned offer can be composed as a set of instantiated collaborative concepts, which can be transformed into a set of business rules that logic can automatically handle. By so, this paper contributes a designed business inquiry/offer system between the inquirers and offerers.

The rest of the paper is arranged as follows: Section 2 describes CoDORM approach. Section 3 proposes the implementation specification. Section 4 discusses some related work. Finally, a conclusion with a contribution list and future work are provided.

2. CoDORM Approach

CoDORM applies the technologies for collaborative conceptualization (please refer to the introduction article[1]) and defeasible reasoning (please refer to the introduction article[2]) to design the system. It has four design principles: (1) *Flexibility*: the systems shall be flexible to add new participated systems. (2) *Semantic consistency*: the systems shall be able to maintain semantic consistency of business concepts between participated systems. (3) *Easiness*: the systems shall be easy to use for participants. (4) *Automatic*: the systems shall be automatic to process the incoming inquiries and generate the outgoing offers.

2.1. CPDASP Model

Collaborative process design and automatic service provision (CPDASP) is a model of collaborative designing and using semantically consistent business processes between business process providers and e-marketplace participants so that business process users can obtain desired transaction results. For example, Carlos as a user can correctly make his rental inquiry based on his rental requirement in Table 1 to receive his desired rental offer.

A *business process* is a sequence of conditional actions. Each action is an *action concept*, which has an action sender and an action receiver. The action acts based on an action logic particular to this action. This action logic acts on a set of objects, where some objects provide the input for the action logic and some objects receive the output of the action logic. These objects are consumed by both action sender and action receiver. Generic actions can be diagrammed in Fig. 3.

[1] Collaborative Conceptualization, http://www.sftw.umac.mo/~jzguo/resource/collaborativeConceptualization.html.

[2] Defeasible Reasoning, http://www.sftw.umac.mo/~jzguo/resource/defeasibleReasoning.html.

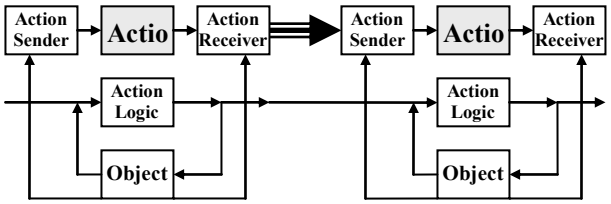

Fig. 3: Generic Actions in a Business Process

The CPDASP involves several steps as follows:

(1) *Common business process providers* (BPP) for e-marketplaces collaboratively design document-oriented business processes on a *peer-to-peer* (P2P) collaboration network to leverage heterogeneous business contexts, in the way of collaborative designing (a) *common business vocabularies* V(X), (b) *common business document templates* D(X) using V(X), and (c) *common business process patterns* P(X) using both V(X) and D(X).

(2) *E-marketplace facilitators* (EMp) *localize* V(X), D(X) and P(X) into their own personalized e-marketplace V'(X), D'(X) and P'(X) to satisfy their own preferences through a collaborative mapping on a *dominator-to-follower* (D2F) [7] collaboration network such that:

- $\forall t' \in V'(X)$ and $\forall t \in V(Y) \subseteq V(X) \bullet t' =_{sem} t$;
- $\forall d' \in D'(X)$ and $\forall d \in D(Y) \subseteq D(X) \bullet d' =_{sem} d$;
- $\forall p' \in P'(X)$ and $\forall p \in P(Y) \subseteq P(X) \bullet p' =_{sem} p$.

The notation "$=_{sem}$" means semantically equivalent, for example, "refrigerator" is semantically equivalent to "fridge" after collaborative agreement.

(3) All *firms* (e.g. real estate agents) (FIRM) again localize or directly use the e-marketplace V'(X), D'(X) and P'(X) as firm-based V''(X), D''(X) and P''(X) to satisfy their own personalized enterprise information systems through a collaborative mapping on a *dominator-to-follower* (D2F) collaboration network such that:

$\forall t'' \in V''(X)$ and $\forall t' \in V'(Y) \subseteq V'(X) \bullet t'' =_{sem} t'$;
$\forall d'' \in D''(X)$ and $\forall d' \in D'(Y) \subseteq D'(X) \bullet d'' =_{sem} d'$;
$\forall p'' \in P''(X)$ and $\forall p' \in P'(Y) \subseteq P'(X) \bullet p'' =_{sem} p'$.

(4) Each participated e-marketplace facilitator creates *user-based computer agents* (UPA), providing document-oriented user interface (UI) to users (e.g. Carlos) based on the designed V'(X), D'(X) and P'(X) for providing business services. For example, Carlos can use his computer agent to make inquiries and receive offers.

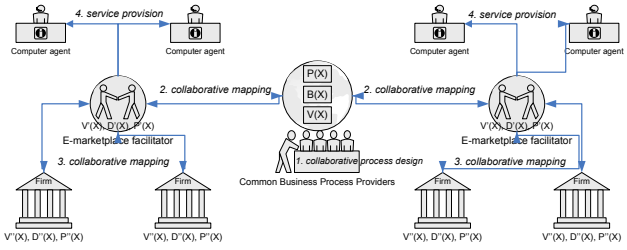

Fig. 4: CPDASP Model

The CPDASP model is diagrammed in Fig. 4, where the key business processes are inquiring and offering processes. An *inquiring process* is initiated from computer agent and ends at local firms. An *offering process* is initiated from local firms, aggregated, compared and selected at e-marketplace facilitators, and finally ends at computer agents. The CPDASP task is to ensure that both inquiring and offering business processes can be fulfilled.

2.2. Document-Oriented Rule Making

Document-oriented rule making (DORM) describes how business rules and facts can be turned into XML PRODUCT MAP (XPM) documents [6] during the business document template design by concept designers and during the business document instantiation by users (e.g. Carlos).

XPM-based rule making is important, because it can effectively check semantic inconsistency during document transformation. It is novel for document exchange in business process running.

2.2.1. Facts Classification

In order to maintain semantic consistency between heterogeneous representations of different contexts and block out the ambiguous concepts that may not be semantically consistent between business processes, DORM distinguishes between consistent facts and ambiguous facts on real-world facts in rule making.

- *Consistent facts* (TF), which are mutually agreed concepts between interaction parties without semantic consistency issue. In this paper, the consistent facts are meaningful concepts and derived from the mutual agreement in P2P and D2F collaboration.
- *Ambiguous facts* (PF), which are not mutually agreed concepts between interaction parties with possible semantic consistency issue, i.e. NOT(TF).

TF proof guarantees the consistent uses of concepts between heterogeneous business processes.

2.2.2. Categories of Consistent Facts

To effectively prove that a fact is semantically consistent, we classify facts (F) into four categories.

- *Basic fact* (BF), which is a noun-form *class concept* (i.e. not reified) in terms of a category, a class or an abstract phenomenon. For example, terms of a product vocabulary are basic consistent facts.
- *Composite fact* (CF), which is a noun-form composite concept. It is composed by a set of logically related BFs. Any document template or its instance is a CTF.
- *Action fact* (AF), which is a verb-form action concept that composes an action. For each AF, the action has its fixedly designated action sender, action receiver, action logic, and targeted objects.
- *Instantiated fact* (IF) is a *reified concept* that associates with a *class concept*. Without associating a class concept, it is a *null fact* (NF) referring an independent symbol or literal without context and is a meaningless

representation by its own. For example, "33", "red" and possible computing formulas.

2.2.3. Proof by Concept Checking Mechanism (CCM)

One responsibility of DORM is to prove whether BF, CF, AF and IF are semantically consistent facts BTF, CTF, ATF and ITF carried by a business process in the CPDASP system scope. To prove it, a CCM is created to check semantic consistency in CPDASP model, such that:

$$btf(p) \bullet \text{for each } p \in CCM_{btf}(CPDASP);$$
$$ctf(p) \bullet \text{for each } p \in CCM_{ctf}(CPDASP);$$
$$atf(p) \bullet \text{for each } p \in CCM_{atf}(CPDASP);$$
$$itf(p) \bullet \text{for each } p \in CCM_{itf}(CPDASP);$$

where the check is to prove whether the concept *p* is *collaboratively created* (i.e. consistent) in CPDASP model.

2.2.4. Rule-Based and Document-Oriented Process

We represent a business process as a rule-based and document-oriented process as follows:

$$a_1(d_1) \Rightarrow \ldots \Rightarrow a_i(d_i) \Rightarrow \ldots \Rightarrow a_n(d_n),$$

in which a_i is an action fact *af* belonging to a process (*proc*) and d_i is a composite fact *cf* (here a *document*) on which the action a_i acts. Both *af* and *cf* and must be proved as consistent *atf* and *ctf*.

Since a document is a composite fact that composes a set of logically related basic facts *bf*, these basic facts and their instantiated facts *if* must be proved as consistent facts of *btf* and *itf*. If any of them is not proven, the process will block the unproven facts with a consequence of either aborting or continuing the process, depending on the predefined procedural rules and the user-defined rules directly appeared in the document d_i.

The main idea of the algorithm design of the consistent fact proof of a document-oriented process is as follows:

```
Input a1(d1):
   atf(a1) ⇒ { //prove outmost action layer, a1∈CCMatf(CPDASP)
   ctf(d1) ⇒ { //prove document layer, d1∈ CCMctf(CPDASP)
   {btf(ci) ⇒ itf(ci, pi)} // recursively prove the inner layer concepts,
   }}                          // ci∈ CCMbtf(CPDASP) and ci∈
CCMitf(CPDASP)
```

2.2.5. XPM Representation of Facts and Rules

In this part, we describe the representation of facts and business rules in XPM. The detailed specification of XPM can be found in [6]. We also show how to convert them into defeasible logic-like syntax.

An XPM document is very simple and consists of a set of concepts that can be represented as follows:

<concept *iid* = "" *an* = "" *cof* = "" *ct* = "" *refTo* = "">
  <value *pr* = "" *dt* = "" *fn* = ""></value>
</concept>

where <concept> represents a concept in which the attributes *iid* is the unique concept identifier, *cof* is the parent concept *iid*, *ct* is the type of the concept that defines how to process the sibling concepts such that whether they have relations of "single choice", "partial selection" and "group" based on lower level concept computational results, *refTo* is the referenced concept *iid* from other concept vocabulary, and <value> represents the instance structure of the <concept> in which *pr* denotes how the instance looks like, *dt* is the data type of instance and *fn* is the processing method of instance. In collaborative concept design stage, the particular value of <value> is not specified. It is instantiated only in the use stage.

In XPM, <concept> has several variant notations in order to differentiate vocabularies, documents and processes with each other. They are <voc> and <concept> in vocabulary, <doc> and <elemon> in document, and <proc>, <act>, <op> , <logic>, etc. in process.

When an XPM is received by a receiver, it is parsed and validated against XPM specification. XPM can easily represent all types of business concepts in both collaborative design stage (as templates) and automatic use stage (as instances). To utilize defeasible reasoning capability, XPM documents can be converted to defeasible logic-like syntax, following the *convertion rules* (cr):

cr_1: xpm: concept(*iid*) ⇒ fact(*iid*)
// *iid* attribute to be a fact and no <value> instance.

cr_2: xpm: concept(*iid*, *v*) ⇒ fact(*iid*(*v*))
// *iid* attribute to be a fact with <value> instance.

cr_3: xpm:concept(iid_2){concept(iid_1)}, $iid_1 \Rightarrow iid_2$
// Single child concept $\{iid_1\}$ as antecedent and parent concept iid_2 as conclusion.

cr_4: xpm: concept(iid_2, *v*){concept(iid_1, *v*)}, $iid_1(v) \Rightarrow iid_2(v)$.
// Single child reified concept $\{iid_1\}$ as antecedent and parent reified concept iid_2 as conclusion.

cr_5: xpm:concept(iid_p, *v*){concept(iid_1, *v*), …, concept(iid_n, *v*)}, $iid_1(v)$, …, $iid_n(v) \Rightarrow iid_p(v)$.
// Multiple children reified concepts $\{iid_1, \ldots, iid_n\}$ are antecedents and parent reified concept iid_p is conclusion .

r_6: xpm: concept(iid_r, *rank*) < concept(iid_s, *rank*) ⇒ $iid_r(rank) > iid_s(rank)$.
// Less ranking number in concept (i.e. higher ranking), higher priority in superiority relation.

In summary, DORM uses XPM document to represent business rules to disambiguate business semantics.

2.3. Reasoning with Defeasible Logic

2.3.1. Inquiring/Offering Process in Reasoning

In CPDASP model, a business process proceeds forward on a concept supply chain [6], where a reified business document is transformed across heterogeneous contexts by procedural mapping rules. For the inquiring and offering process, the particular process steps can be as follows:

Step1: the inquiry document (e.g. Fig. 5), is sent by the inquirer from the *user's computer process agent* (UPA) and

arrives at destinations of *firms' systems* (FIRM) in offerers' understandable forms through the path of *e-marketplace facilitator* (EMp) and *common business process provider* (BPP) (optional).

Step 2: each FIRM, after receiving, needs to analyze and reasons the received inquiry document whether it can make offer based on (1) the requirements shown in the received inquiry document, (2) the offerer' business rules, and (3) the offerer's available stocks. All FIRM uses the defeasible reasoning to produce offers.

Step 3: Upon the offers are ready, all FIRM send back the offers automatically to UPA in following alternatives:

Alternative 1: directly sending back to UPA via EMp if FIRM and UPA are within the same EMp.

Alternative 2: indirectly sending back to UPA via EMp via BPP if FIRM and UPA are not within the same EMp.

In this step, EMp or EMp/BPP (if alternative 2) need to defeasibly reason on the received offers for best offers for UPA based on the original requrements in the inquiry document (*see* example in Fig. 5). After best offers are made, the merged offer result is sent to UPA by EMp through or not through BPP.

It is obvious that the key issue in the inquiry/offering business process is the the defeasible reasoning on the concept supply chain in the CPDASP model but still needing to accurately maintainining semantic consistency.

2.3.2. Product Offer Reasoning Model

Our solution to the above issue has three aspects: First, for each XPM document received by FIRM, EMp or BPP, a consistency check must firstly be made by the concept checking mechanism (CCM) (*see* Section 2.2.3) to ensure all concepts in a received document are semantically consistent between UPA, FIRM, EMp and BPP. Second, the XPM concepts, rules, offers and decisions are extracted and translated into defeasible logic facts and rules based on the conversion rules described in Section 2.2.5. Third, best offers are generated applying defeasible reasoning.

In the following, a general *product offer reasoning model* is proposed on defeasible logic to generate best offers. The reasoning model has the following steps:

(1) *Positiveness As Success (PAS) for inclusion*, which finds out the successful *possible product set* (PPS) that satisfies the test of *subject product* (SUB) in product sources. The query rule is:

R1: Query(SUB) $\rightarrow$ PPS(SUB).

(2) *Negation As Failure (NAF) for exclusion in inquirer's view*, which finds out the *reduced product set* (RPS) in PPS(SUB) by applying the defeasible rule of NAF [2] by excluding the product set, matched with the negated required *true facts & rules* (TFR) in the inquiry document, from PPS(SUB). The exclusion rules are:

R2: $\neg TFR_i(PPS(SUB)) \Rightarrow \neg RPS(SUB)$;

for all TFR_i not matched in PPS(SUB), they are not in RPS(SUB).

R3: (R2 > R1) $\Rightarrow$ RPS(SUB);

for all in PPS(SUB) not matched with $TFR_i(PPS(SUB))$, they are excluded from PPS(SUB) thus to produce RPS(SUB).

(3) *Negation As Failure (NAF) for exclusion in offerer's view*, which finds out the *offerable product set* (OPS) in RPS(SUB) by applying NAF rule by excluding the product set, matched with the true *non-offerable facts & rules* (NFR) set by the offerer, from RPS(SUB). The exclusion rules are:

R4: $NFR_i(RPS(SUB)) \Rightarrow \neg OPS(SUB)$;

for all NFR_i matched in RPS(SUB), they are not in OPS(SUB).

R5: (R4 > R3) $\Rightarrow$ OPS(SUB);

for all in RPS(SUB) matched with $NFR_i(RPS(SUB))$, they are excluded from RPS(SUB) thus to produce OPS(SUB).

(4) *Best offers generation by ranking offerable products*, which finds out the *best offerable set* (BOS) of products by computing *ranking results* (RNK) of OPS(SUB) for each *ranking criterion* (CRI) applying *superiority relation* >. The ranking rule is:

R6: $CRI_p(SUB, RNK_i) > CRI_p(SUB, RNK_j) \Rightarrow$
$CRI_p(SUB, RNK_i)$.

R7: $CRI_p(SUB, RNK_i) > CRI_q(SUB, RNK_i) \Rightarrow$
$CRI_p(SUB, RNK_i)$.

For the same ranking criterion, lower ranking number in higher ranking sequence is superior to the higher ranking number in lower ranking sequence. For the different ranking criteria, higher priority criterion is superior to the lower priority criterion.

It should be pointed out that for the e-marketplace facilitators (EMp), they only need to experience R6 and R7 to make ranking to find the best offers because their received offers from firms' systems are all offerable.

```
<xpm><head><proc iid = "p.2" an="inquire apartment" from="url" to="url"><o iid="p.2.1" an="inquire to receive" vis="public" did="d.2.1" lid="l.2.1"/></proc></head>
<body><doc name="house rental inquiry sheet" lang="en" refTo="d.2">
 <elemon iid="e" an="apartment" ct="atomic" refTo="voc:house-c3">
  <!--General requirements-->
  <elemon iid="e.1" an="size"><value dt="sqm" fn="LgAndEq">45</value></elemon>
  <elemon iid="e.2" an="bedrooms"><value dt="val" fn="LgAndEq">2</value></elemon>
  <element iid="e.3" an="Pet raising" refTo="act:RaisePet"><value dt="string" fn="IS">yes</value></element>
  <elemon iid="e.4" an="Price"><value dt="USD" fn="LessAndEq">400</value></elemon>
  <elemon iid="e.5" an="Floor"><value dt="val" fn="LgAndEq">3</value>
   <elemon iid="e.5.1" an="Elevator"><value dt="string" fn="IS">yes</value></elemon></elemon>
  <!--Specific requirements related to acceptable cost computation-->
  <elemon iid="e.6" an="Cost in central city" ct="comp"><value dt="USD" fn="Fomula">{e.6.2+e.6.3+e.6.4}</value>
    <elemon iid="e.6.1" an="Central city"><value dt="string" fn="IS">yes</value></elemon>
```

```
    <elemon iid="e.6.2" an="Cost of min size"><value dt="USD" fn="Eq">300</value></elemon>
    <elemon iid="e.6.3" an="Cost of extra size" ct="comp"><value dt="USD" fn="Formula">{e.6.2.1*e.6.2.2}</value>
      <elemon iid="e.6.3.1" an="Extra size"><value dt="sqm" fn="VAR">{x}</value></elemon>
      <elemon iid="e.6.3.2" an="Price"><value dt="USD" fn="val">5</value></elemon></elemon>
    <elemon iid="e.6.4" an="cost of garden size" ct="comp"><value dt="USD" fn="Formula">{e.6.3.1*e.6.3.2}</value>
      <elemon iid="e.6.4.1" an="Garden size"><value dt="sqm" fn="VAR">{x}</value>
      <elemon iid="e.6.4.2" an="Price"><value dt="USD" fn="val">2</value></elemon></elemon></elemon></elemon>
  <elemon iid="e.7" an="Cost in suburb" ct="comp"><value dt="USD" fn="Formula">{e.7.2+e.7.3+e.7.4}</value>
    <elemon iid="e.7.1" an="Suburb" ct="comp"><value dt="string" fn="IS">yes</value></elemon>
    <elemon iid="e.7.2" an="Cost of min size"><value dt="USD" fn="Eq">250</value></elemon>
    <elemon iid="e.7.3" an="Cost of extra apartment" ct="comp"><value dt="USD" fn="Formula">{e.7.3.1*e.7.3.2}</value>
      <elemon iid="e.7.3.1" an="Size"><value dt="sqm" fn="VAR">{x}</value></elemon>
      <elemon iid="e.7.3.2" an="Price"><value dt="USD" fn="val">5</value></elemon></elemon>
    <elemon iid="e.7.4" an="Cost of garden" ct="comp"><value dt="USD" fn="Formula">{e.7.4.1*e.7.4.2}</value>
      <elemon iid="e.7.4.1" an="Size"><value dt="sqm" fn="VAR">{x}</value></elemon>
      <elemon iid="e.7.4.1" an="Price"><value dt="USD" fn="val">2</value></elemon></elemon></elemon>
  <!--Inquirer's decision for priority-->
  <elemon iid="e.8" an="Cheapest in price" ct="rank">1</elemon>
  <elemon iid="e.9" an="Largest Garden size" ct="rank">2</elemon>
  <elemon iid="e.10" an="Largest Apartment Size" ct="rank">3</elemon></elemon>
</doc></body></xpm>
```

Fig. 5: Carlos' Rental Inquiry in XML Product Map (XPM)

3. Carlos' Inquiry Example

A concrete example of Carlos' apartment rental inquiry is provided to see how an inquiry can be answered using defeasible reasoning based on XPM files forwading between involved parties. The XPM example is shown in Fig. 5 and the defeasible reasoning process is shown in Fig. 6. For readability purpose, we use readable form of predicates to replace the *iid*-based predicates specified in Section 2.2.4 conversion rules for machine use.

R1: Query(X) → PPS(X) // X is apartment

// R2: $\neg TFR_i(PPS(X)) \Rightarrow \neg RPS(X)$, the following Vx is concept value

r2: bedrooms(X, Y), Y < Vx ⇒ ¬RPS(X) // Y is bedroom number

r3: size(X, Y), Y < Vx ⇒ ¬RPS(X) // Y is apartment size

r4: ¬pets(X) => ¬RPS(X)

r5: floor(X,Y), Y > Vx, ¬elevator(X) ⇒ ¬RPS(X) // Y is floor number

r6: cost(X,Y), Y > Vx ⇒ ¬RPS(X) // Y is Carlos' max acceptable cost

r7: size(X,Y), extraSize(X, E), garden(X,G), central(X) ⇒ cost(X, Y+E+Z) // Y, E are costs of apartment sizes, G is garden cost

r8: size(X,Y), exSize(X, E), garden(X,G), suburb(X) ⇒ cost(X, Y+E+Z)

R3: (r2 > R1, r3 > R1, r4 > R1, r5 > R1, r6 > R1) ⇒ RPS(SUB)

// R4: $NFR_i(RPS(SUB)) \Rightarrow \neg OPS(SUB)$

r9: maxCost(X,Y), offerable(X,Z), Y < Z ⇒ ¬RPS(X) // Y is Carlos' willing to pay max cost, Z is the offerer's minimum price to offer.

R5: (r9 > R3) ⇒ OPS(SUB)

// R6, R7 of Section 3.3.2 to best offers

r10: cheapest(X) => offer(X)

r11: cheapest(X), largestGarden(X) => offer(X)

r12: cheapest(X), largestGarden(X), largestApartment(X) => offer(X)

r11 > r10, r12 > r10, r12 > r11

Fig. 6: Reasoning of Best Offer on Carlos' Example

In this example, offers from real estate agents, which also set rules of minimum acceptable prices for their offers. These rules are not presented in the XPM inquiry document shown in Fig. 5 but in the rule database of offerers.

3.1. Collaborative Human-Agent Framework

Given the design principles of Section 2, CoDORM system needs to implement two business processes: a *rental inquiry process* from UPA (e.g. Carlos' personal agent) to FIRM, and a *rental offer process* from FIRM to EMp/BPP to UPA (*see* Section 2.3.1).

To realize these two processes, the CoDORM system is implemented to include four subsystems of UPA, FIRM, EMp and BPP in different locations. It provides automatic exchange of inquiries and offers, and fulfills the tasks of collaboratively designing and maintaining the semantic consistent business concepts in the forms of business vocabularies, documents and processes. It is implemented in a hybrid *collaborative human-agent framework*, which mixes with human and automated agents, shown in Fig.7.

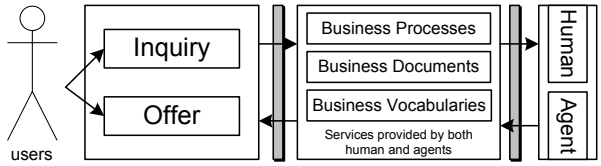

Figure 7: Hybrid Collaborative Human-Agent Framework

In this framework, *humans* are responsible for providing the human-related work, e.g. collaborative designing, editing, modifying, approving and publishing business concepts. *Automated agents* are responsible for non-human work, e.g. automatically analyzing, aggregating, mapping, matching and forwarding human-provided business concepts. Users of EMp, on the other hand, simply subscribe and use the services that both human and agents provide.

The benefit of responsibility separation between human and automated agents is that agents have no rights to make erroneous inferences without proven consistent facts and rules. It prevents semantic conflicts between users and designers.

3.2. System Architecture and Modules

The CoDORM systems adopt a distributed P2P/D2F collaboration architecture discussed in [8]. P2P refers to the peer-to-peer collaboration architecture, where BPP collaboratively design and maintain business concepts of

common BV (comVoc), *common BD* (comDoc) and *common BP* (comProc). D2F means *dominant-to-follower*, which is point-to-point collaboration architecture. In this architecture, different EMp localize comVoc, comDoc and comProc of BPP into their personalized *e-marketplace BV* (empVoc), e-marketplace *business BD* (empDoc) and *e-marketplace BP* (empProc). FIRM again localize empVoc, empDoc and empProc into their personalized *firm BV* (frmVoc), *firm BD* (frmDoc) and *firm BP* (frmProc).

All above-created common and local concepts constitute the business concepts (i.e. consistent facts of BTF, CTF and ATF). System components for the collaborative production of general concepts constitute *CoDORM Designer subsystem*.

At UPA (e.g. Carlos' UPA), users (e.g. Carlos) create and use reified concepts (e.g. inquiry/offer documents and processes). At EMp and FIRM sides, users are those people who process particular inquiries and offers. The system components related to creating and using reified concepts constitute *CoDORM User subsystem*.

Reified concepts of process actions and documents are sent from UPA to EMp. They are processed, stored and forwarded to FIRM, which again process, store and answer reified concepts back to EMp finally back to UPA. All these activities are not human-involved and are agents' tasks. The components for these activities constitute the *CoDORM Exchanger subsystem*.

The above subsystems are all distributed, that is, they shall be implemented in different locations of UPA, EMp, FIRM and BPP on Internet. Fig. 8 describes CoDORM architecture in four layers of collaboration, concept, logic (or structure) and messaging.

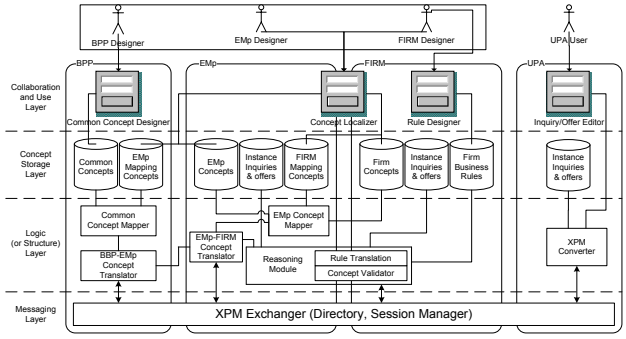

Fig. 8: CoDORM Architecture

- The *messaging layer* is the bottom layer and is responsible for exchanging business concepts between UPA, EMp, FIRM and BPP in XPM on HTTP protocol. It consists of *XPM Exchanger Module* in all locations. It consists of search directory of users, EMp, FIRM and BPP and Session Manager for managing interactions).

- The *logic (structure) layer* is responsible for translating the results of one subsystem to another through Concept Translator and Concept Mapper both appeared in BPP and EMp and reasoning inquiries for making offers through Reasoning Module. The Reasoning Module includes Rule Translator for translating between XPM concepts and defeasible logic rules, and Concept Validator that parses and validates the XPM representations.

- The *concept storage layer* is responsible for storing and retrieving various business concepts from both collaborative design and user creation. In this layer, there are Common Concepts collaboratively created by BPP, EMp Concepts, EMp Mapping Concepts and Instance Inquiries and Offers collaboratively created by EMp, Firm Concepts, FIRM Mapping Concepts, FIRM Business Rules and Instance Inquiries and Offers by FIRM, and Instance Inquiries and Offers by UPA.

- The *collaboration and use layer* is the highest layer, which is responsible for the collaborative creation and edition of common concepts in BPP, EMp concepts in EMp and FIRM concepts in FIRM, and for the use of EMp concepts in UPA.

Different roles interact with each other using their own user interfaces provided by the collaboration and use layer. In particular, BPP designers use Common Concept Designer as collaborative concept editor, EMp designers and FIRM designers use Concept Localizer to personalize their own concepts. FIRM also uses Rule Designers to develop their own business rules that are used for limiting the offers production in reasoning process. UPA users just simply use Inquiry/Offer Editor to read, write, send and receive the inquiries and offerns.

4. Related Works

The semantic consistency maintenance between heterogeneous contexts is a rather new research issue in the area of e-Commerce. Currently, few works could be found except in the authors' research groups. Thus, the theme of this paper mainly relates to CONEX [6] and CODEX [7], which are early works of the authors. They focus on how to represent heterogeneous concepts and how to apply collaboration method to maintain semantic consistency between heterogeneous contexts. In developing reasoning method for business rule inference, this paper applies the defeasible logic [1][2] as its foundation.

5. Conclusion

In this paper we studied the semantic consistency maintenance issue between heterogeneous contexts, that is, how an inquiry from an unknown user of an e-marketplace can be received and answered in a semantically consistent way by a firm that is not in the context of the user's e-marketplace. The proposed solution of this paper uses XPM of collaborative concept to represent semantically consistent business concepts and adopts defeasible logic to reason with XPM document-oriented business rules for inquiring and offering. We

motivated the approach with a real-world apartment rental problem and explained it in architecture of collaborative business process design and automatic service provision. We reported the implementation specification within a hybrid human-agent framework.

Our approach has advantages comparing with known solutions. (1) We do not rely on single shared domain vocabularies that cannot cope with the issue of semantic consistency maintenance, but a set of collaboratively designed and mapped cross-domain business concepts for enabling heterogeneous concept exchange. (2) The architecture we provide is collaborative, distributed, role-based and service-oriented. It is highly flexible for many semantically different systems to both join and leave without unfavorable consequences. (3) We introduce a proof mechanism to prove that all business concepts in reasoning for producing outgoing processes are all semantically consistent facts. Thus, we guarantee semantically consistent reasoning. (4) We use document-oriented rule making method by providing predefined document templates. It enables users to specify business rules in a simple web-based form, which again omits the need of programming in logic layer for each business process. (5) We use XPM to represent concepts. It helps the separation of structure representation from concept representation, which enhances the design reusability of both concept structures and concept annotations.

The main limitation of our work is: XPM has not been evolved to include reusable verb-formed concepts, i.e. templates of action logic have not been provided.

In future, we intend to extend the work in the following directions: (1) Provide a website that can physically demonstrate the approach. (2) Provide real common and local concept editors for people to localize their own local concepts in the demo website so that the real-world collaborative concept community can be established. (3) Publish a stable XPM specification for business concept design.

Acknowledgements

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

References

[1] Antoniou, G., Billington, D., Governatori, G. and M.J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic* 2, 2 (2001) 255 – 287.

[2] Antoniou, G. and A. Bikakis. DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web. *IEEE Transactions on Knowledge Data and Engineering* 19(2), (IEEE Computer Society, 2007) 233-245.

[3] Berners-Lee, T., Hendler, J. and O. Lassila. The Semantic Web. *Scientific American*, 284(5), (2001): 34-43.

[4] Dean, M. and G. Schreiber (Eds.). *OWL Web Ontology Language Reference* (2004). www.w3.org/TR/2004/REC-owl-ref-20040210/.

[5] Gruber, T. A Translation Approach to Portable Ontologies. *Knowledge Acquisition* 5(2), (1993)199-220.

[6] Guo, J. *Collaborative Concept Exchange*, VDM Publishing: Germany, 2008.

[7] Guo, J. Inter-Enterprise Business Document Exchange. In: *Proc. ICEC'06*, (ACM Press, 2006) 427-437.

[8] Guo, J. A Transparent Collaborative Integration Approach for Ad Hoc Product Data. In: *Proc. CEC'06/EEE'06* (IEEE Computer Society Press 2006).

[9] Keller, A. and M. Genesereth. Multivendor Catalogs: Smart Catalogs and Virtual Catalogs. *EDI Forum: Journal of Electronic Commerce* 9(3) (1996) 87-93.

[10] Segev, A., Wan, D. and C. Beam. Electronic Catalogs: a Technology Overview and Survey Results. In: *Proc. CIKM'95* (ACM Press, 1995) 11-18.