I3E'2005 in Poznan
Day1: October 26, 2005

Dynamic Model Harmonization between Unknown eBusiness Systems

Makoto Oya[1] and Masumi Ito[2]

[1] Shonan Institute of Technology, Japan
(formerly, Hokkaido University, Japan)

2 Hokkaido University, Japan

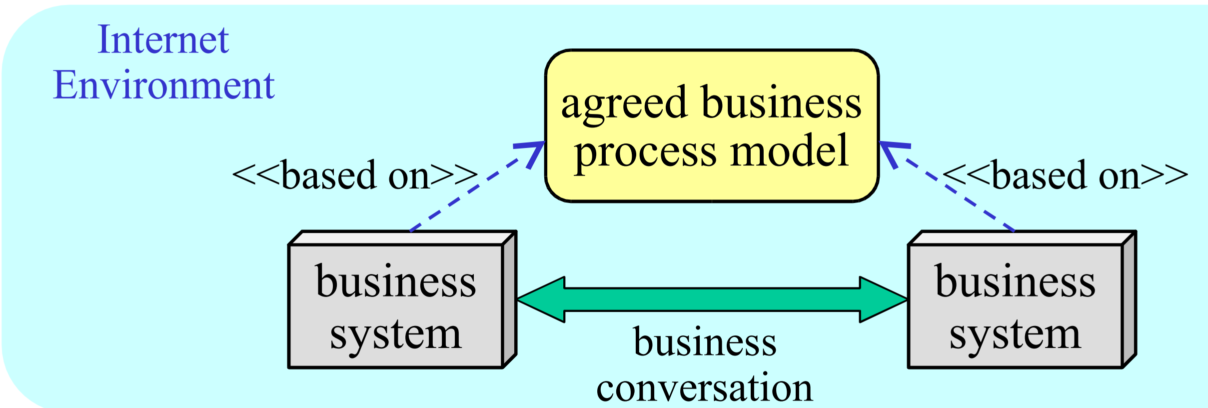
Agenda

- Introduction
- Concept of "Model Harmonization"
- Model Description
- Model Harmonization Algorithm
- Experiment
- Discussion on Business Resource Definition Matching
- Conclusions

Introduction

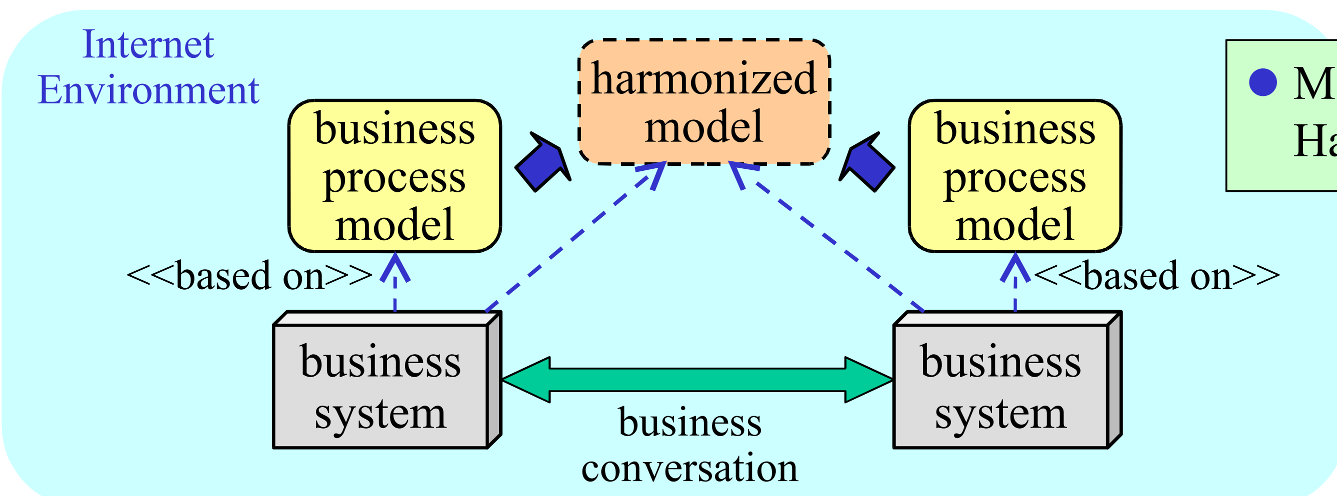
- Web Services
  - from simple request-response to message-based conversation
  - SOAP, WSDL, ebXML messaging, WS-Reliability, BTP
  - Key issue: alignment of business process model among eBusiness systems
- Business process modeling technologies
  - WS-BPEL, WS-CDL, BPNM, BPML, ...
- Usual assumption
  - systems share an agreed business process model
  - not always be satisfied
- Approach in this research
  - systems do not share complete business process models
  - dynamically harmonize business process models
  - to enable business conversation among independent and autonomous business systems using their best efforts

Usual assumption



- MDA
  - CIM-PIM-PSM
- Repository
- Business process description lang.
- ...

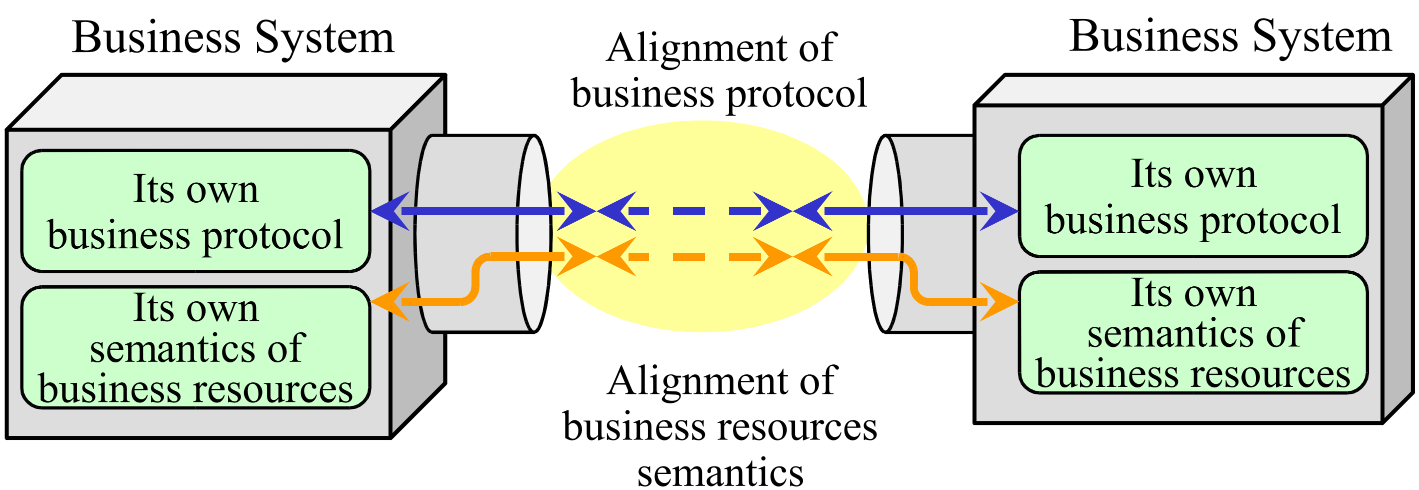
Approach in this research



- Model Harmonization

Two basic issues

- Alignment of business protocol — main focus
- Alignment of semantics of business resources in messages — separate issue



request: payment => shipment

request: shipment => payment
or payment => shipment

Concept of "Model Harmonization"

- Focus on *exposed models*.
  - models exposed by systems to other systems.
- An exposed model consists of an *interface model* and a *behavior model*.
- A type of business element in messages is specified by a *BRD* (*business resource definition*).
  - product id, price, invoice, shipment notification, ...
  - specified by a *BRD identifier*, typically URI.
- *Model Harmonization* is a process to automatically adjust exposed models between encountering systems.

Model Description

- Formal definition of (exposed) models

$M = (IM, BM)$
 IM : An interface model defined below.
 BM : A behavior model defined below.

$IM = (U, T, D)$
 U : A finite set. (Operations)
 $T : U \rightarrow \{ I, O, (I, O), (O, I) \}$
 $D : U \times \{ I, O \} \rightarrow \{ (BRD, \ldots, BRD) \}$

$BM = (B)$
 B : A regular expression on U.

T is a mapping from each operation to its *message exchange pattern*. (abstracted from WSDL2.0 Part2.)

Each *BRD* (business resource definition) is given by a *BRD identifier*.

- Example of exposed model

U = { PriceQuery, GetPrice, Order, Response, Bill, PaymentNotice, Delivery, Acceptance }
T(PriceQuery)=O, T(GetPrice)=I, T(Order)=O, T(Response)=I,
T(Bill)=I, T(PaymentNotice)=O, T(Delivery)=I, T(Acceptance)=O
D(PriceQuery,O)= 'ns:price.ask', D(GetPrice,I)='ns:price.ans',
D(Order,O)='ns:form.order', T(Response,I)='ns:form.order_response',
D(Bill,I)='ns:form.bill', D(PaymentNotice,O)='ns:notice.conf',
D(Delivery,I)='ns:notice.shipment', D(Acceptance,O)='ns:notice.conf'
B = (SEQ,
    (OPT, (SEQ, PriceQuery, GetPrice)),
    Order, Response,
    (OPT, (SEQ, Bill, PaymentNotice, Delivery, Acceptance)))

- Example mapping to WSDL style

```
<definitions name= "Model_PS" xmlns:ns="http://www.examp.org/brd008/">
   <message name="Message1">
      <part name="pt_1" element="ns:price.ask"/>
   </message>
   <message name="Message2">
      <part name="pt_1" element="ns:price.ans"/>
   </message>
   . . . . .
   <portType name="Model_PS_interface">
      <operation name="PriceQuery">
         <output message="Message1">
      </operation>
      <operation name="GetPrice">
         <input message="Message2">
      </operation>
      . . . . .
   </portType>
   <behavior>
      <SEQ>
         <OPT>
            <SEQ> "PriceQuery" "GetPrice" </SEQ>
         </OPT>
         "Order" "Response"
         <OPT>
            <SEQ> "Bill" "PaymentNotice" "Delivery" "Acceptance" </SEQ>
         </OPT>
      </SEQ>
    </behavior>
</definitions>
```

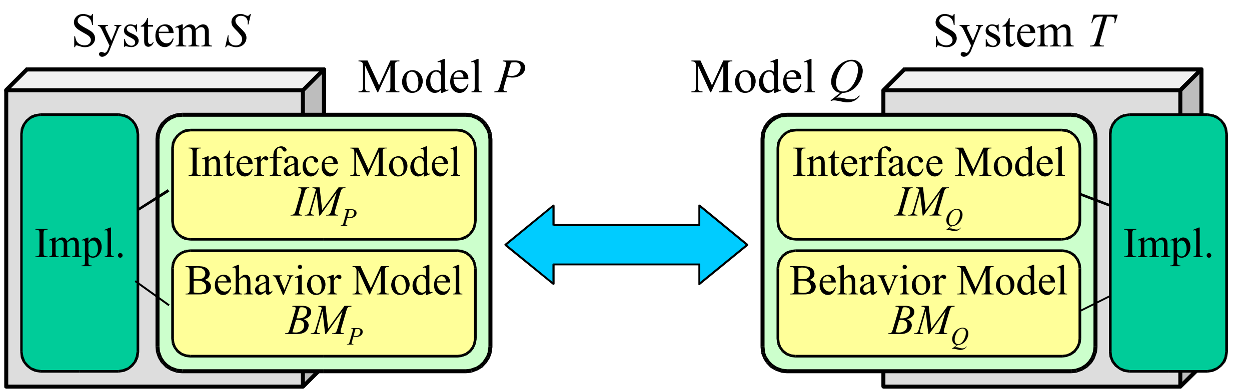
- Example of exposed model

U = { PriceQuery, GetPrice, Order, Response, Bill, PaymentNotice, Delivery, Acceptance }
T(PriceQuery)=O, T(GetPrice)=I, T(Order)=O, T(Response)=I,
T(Bill)=I, T(PaymentNotice)=O, T(Delivery)=I, T(Acceptance)=O
D(**PriceQuery**,O)= '**ns:price.ask**', D(**GetPrice**,I)='**ns:price.ans**',
D(Order,O)='ns:form.order', T(Response,I)='ns:form.order_response',
D(Bill,I)='ns:form.bill', D(PaymentNotice,O)='ns:notice.conf',
D(Delivery,I)='ns:notice.shipment', D(Acceptance,O)='ns:notice.conf'
B = **(SEQ,**
(OPT, (SEQ, PriceQuery, GetPrice)),
Order, Response,
(OPT, (SEQ, Bill, PaymentNotice, Delivery, Acceptance)))

- Example mapping to WSDL style

```
<definitions name= "Model_PS" xmlns:ns="http://www.examp.org/brd008/">
   <message name="MessageI">
      <part name="pt_1" element="ns:price.ask"/>
   </message>
   <message name="Message2">
      <part name="pt_1" element="ns:price.ans"/>
   </message>
   . . . . .
   <portType name="Model_PS_interface">
      <operation name="PriceQuery">
         <output message="Message1">
      </operation>
      <operation name="GetPrice">
         <input message="Message2">
      </operation>
      . . . . .
   </portType>
   <behavior>
      <SEQ>
         <OPT>
            <SEQ> "PriceQuery" "GetPrice" </SEQ>
         </OPT>
         "Order" "Response"
         <OPT>
            <SEQ> "Bill" "PaymentNotice" "Delivery" "Acceptance" </SEQ>
         </OPT>
      </SEQ>
    </behavior>
</definitions>
```

Model Harmonization Algorithm

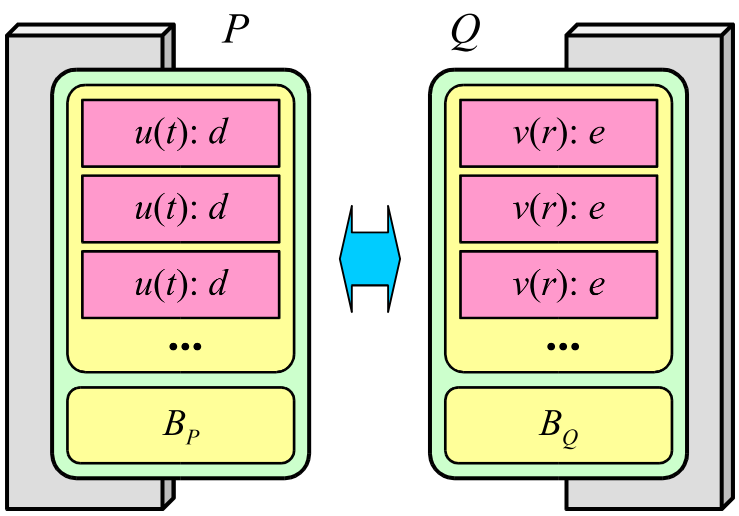


- Top-level algorithm

Step a : **exchange** P and Q between S and T
Step b : At S,
  reduce P adjusting to Q, using the model reduction algorithm
  suspend execution if the resulted model *newP* is ϕ
Step c : At T,
  reduce Q adjusting to P, using the model reduction algorithm
  suspend execution if the resulted model *newQ* is ϕ
Step d : **start** conversation between S and T using the reduced models

Model Reduction Algorithm

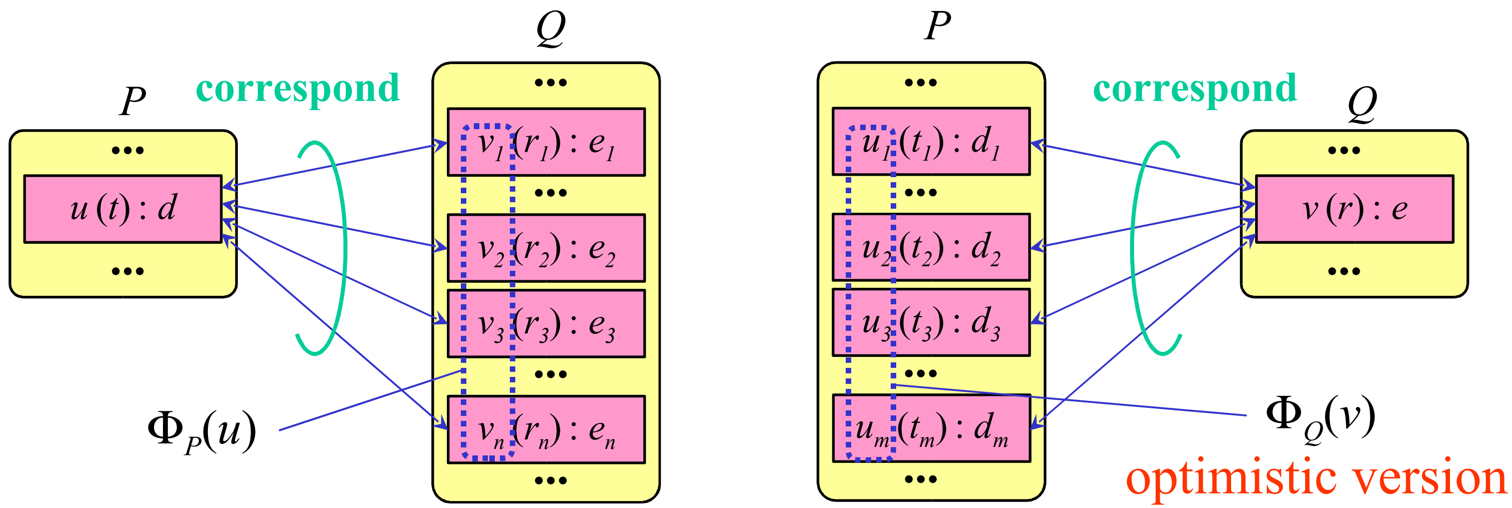
- Notations in this explanation
  - Models:
    - $P = (IM_P, BM_P)$, $IM_P = (U, T_P, D_P)$, $BM_P = (B_P)$
    - $Q = (IM_Q, BM_Q)$, $IM_Q = (V, T_Q, D_Q)$, $BM_Q = (B_Q)$
    - $T_P(u)$ and $T_Q(v)$ are either I or O, $D_P(u, T_P(u))$ and $D_Q(v, T_Q(v))$ have one BRD.
  - Illustration:



u ($\in U$) : an operation in P.
t ($=T_P(u)$) : its message exchange pattern, i.e., I or O.
d ($=D_P(u,T_P(u))$) : its BRD.

v ($\in V$) : an operation in Q.
r ($=T_Q(v)$) : its message exchange pattern, i.e., I or O.
e ($=D_Q(v,T_Q(v))$) : its BRD.

Step1: Find corresponding sets of operations, $\Phi_P(u)$ and $\Phi_Q(v)$



- *u* and *v* "correspond" when:

strict version

  - $t=O$ and $r=I$ and $match(d, e)=True$, or
    $t=I$ and $r=O$ and $match(e, d)=True$

optimistic version

$match(d, e)=(True$ or $Unknown)$
$match(e, d)=(True$ or $Unknown)$

- *match*() evaluates BRD matching:
  - $$match(BRD_{from}, BRD_{to}) = \begin{cases} True & \text{(if } I(BRD_{from}) \subset I(BRD_{to}) \text{ is detected.)} \\ False & \text{(if } I(BRD_{from}) \not\subset I(BRD_{to}) \text{ is detected.)} \\ Unknown & \text{(otherwise, i.e., neither is detected.)} \end{cases}$$

Note: *I*(*brd*) denotes all instances satisfying *brd*.

- match() is implemented in various ways.
  - A trivial implementation:

$$match(BRD_{from}, BRD_{to}) = \begin{cases} True & \text{(if } BRD_{from}=BRD_{to} \text{ and both are in the same name space.)} \\ Unkown & \text{(otherwise)} \end{cases}$$

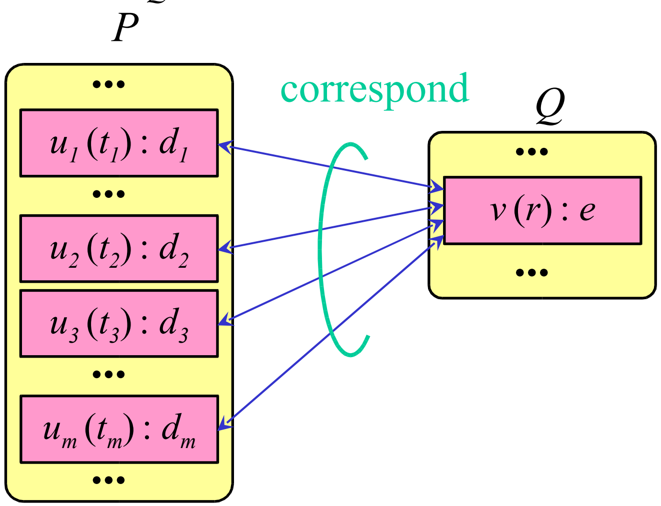
  - More refined implementations
    - will be discussed later.

Step2: Remove unnecessary operations

- Remove operations not corresponding to any other operations, from behavior patterns B_P and B_Q.

Step3 (main part): Reduce behavior pattern

- Replace operations in a behavior pattern B_Q by operations in P.
- for all v in B_Q, replace v by $G(v)$. (call it B'_Q)



$$G(v) = u_1 \mid u_2 \mid \ldots \mid u_m$$

- Obtain $newB_P$ satisfying $L(newB_P) = L(B_P) \cap L(B'_Q)$.
  - Note: $L(R)$ is a set of sequences matching to a regular expression R.
- Suspend if $L(newB_P) = \phi$.

Step4: Generate final model.

Step 1 :

for each u **in** U **:**

$\Phi_P(u) = \{ v \mid T_P(u)=O$ **and** $T_Q(v)=I$ **and** $match(D_P(u,O), D_Q(v,I))$

or $T_P(u)=I$ **and** $T_Q(v)=O$ **and** $match(D_Q(v,O), D_P(u,I)) \}$

for each v **in** V :

$\Phi_Q(v) = \{ u \mid T_Q(v)=O$ **and** $T_P(u)=I$ **and** $match(D_Q(v,O), D_P(u,I))$

or $T_Q(v)=I$ **and** $T_P(u)=O$ **and** $match(D_P(u,O), D_Q(v,I)) \}$

$U' = \{ u \mid u \in U$ **and** $\Phi_P(u) \neq \phi \}$; $V' = \{ v \mid v \in V$ **and** $\Phi_Q(v) \neq \phi \}$

if $U' = \phi$ **or** $V' = \phi$ **then exit** (suspend)

Step 2 :

obtain B'_P **satisfying** $L(B'_P) = L(B_P) \cap (U')^*$

obtain B'_Q **satisfying** $L(B'_Q) = L(B_Q) \cap (V')^*$

if $L(B'_P) = \phi$ **or** $L(B'_Q) = \phi$ **then exit** (suspend)

Step 3 :

for each v **in** V' :

$G(v) = \varepsilon$

for each u **in** $\Phi_Q(v)$: $G(v) = G(v) + "|" + u$

replace all v **in** B'_Q **by** $G(v)$

obtain $newB_P$ **satisfying** $L(newB_P) = L(B'_P) \cap L(B'_Q)$

if $L(newB_P) = \phi$ **then exit** (suspend)

Step 4 :

$newU = \{ u \mid u \in U'$ **and** $(u$ **in** $newB_P) \}$

if $newU = \phi$ **then exit** (suspend)

$newT_P = T_P$ **restricted in** $newU$; $newD_P = D_P$ **restricted in** $newU$

$newIM_P = (newU, newT_P, newD_P)$; $newBM_P = (newB_P)$

$newP = (newIM_P, newBM_P)$

return $newP$

Experiment

- MD_REDUCE : an experimental implementation.
- Implements the model reduction algorithm.
- A model is represented by a Python object, independent of middleware platform. Mapping to WSDL style is supported.

- Execution example:
  - A buyer's system exposing a model *P* and a provider's system exposing a model *Q*.
  - The trivial *match*() is used.

Execution Example

- Before reduction

Model *P*

NameSpaces:
ns1=http://www.examp.org/brd008/,
ns2=http://www.mkik.org/brdef/
U, T, D:
PriceQuery[O]=ns2:price.ask, GetPrice[I]=ns2:price.ans,
Order[O]=ns2:form.order,
Response[I]=ns2:form.order_response,
AnsQuestion[O]=ns2:form.order,
Bill[I]=ns1:form.bill, PaymentNotice[O]=ns1:notice.conf,
Delivery[I]=ns1:notice.shipment,
Acceptance[O]=ns1:notice.conf
B:

```
(SEQ,
    (OPT,  (SEQ, PriceQuery, GetPrice)), Order,
    (CHO,
        Response,
        (SEQ, Response, AnsQuestion, Response),
        (SEQ,
            (CHO,
               Response,
               (SEQ, Response, AnsQuestion, Response) ),
            Delivery, Acceptance, Bill, PaymentNotice    )
)   )   )
```

Model *Q*

NameSpaces:
NS=http://www.examp.org/brd008/,
mk=http://www.mkik.org/brdef/
U, T, D:
EstPrice[I]=mk:price.ask, AnsPrice[O]=mk:price.ans,
Preordered[I]=NS:notice.preorderID,
Order[I]=mk:form.order,
Question[O]=mk:form.order_response,
AcceptOrder[O]=mk:form.order_response,
DenyOrder[O]=mk:form.order_response,
Bill[O]=NS:form.bill, ConfPayment[I]=NS:notice.conf
Ship[O]=NS:notice.shipment, ConfDelivery[I]=NS:notice.conf,
B:

```
(SEQ,
    (CHO, (SEQ, EstPrice, AnsPrice), Preordered),
    Order,
    (REP, (SEQ, Question, Order)),
    (CHO,
        DenyOrder,
        (SEQ, AcceptOrder,
            (CHO, ( SEQ, Bill, ConfPayment, Ship, ConfDelivery),
                    ( SEQ, Ship, ConfDelivery, Bill, ConfPayment ) )  )
)    )
```

After reduction

Model *P*

NameSpaces:
ns1=http://www.examp.org/brd008/,
ns2=http://www.mkik.org/brdef/
U, T, D:
PriceQuery[O]=ns2:price.ask, GetPrice[I]=ns2:price.ans,
Order[O]=ns2:form.order,
Response[I]=ns2:form.order_response,
AnsQuestion[O]=ns2:form.order,
Bill[I]=ns1:form.bill, PaymentNotice[O]=ns1:notice.conf,
Delivery[I]=ns1:notice.shipment,
Acceptance[O]=ns1:notice.conf
B:

```
(SEQ,
    PriceQuery, GetPrice, Order,
    (CHO,
        Response,
        (SEQ, Response, AnsQuestion, Response),
        (SEQ,
            (CHO,
                Response,
                (SEQ, Response, AnsQuestion, Response) ),
            Delivery, Acceptance, Bill, PaymentNotice    )
)   )   )
```

Model *Q*

NameSpaces:
NS=http://www.examp.org/brd008/,
mk=http://www.mkik.org/brdef/
U, T, D:
EstPrice[I]=mk:price.ask, AnsPrice[O]=mk:price.ans,
Order[I]=mk:form.order,
Question[O]=mk:form.order_response,
AcceptOrder[O]=mk:form.order_response,
DenyOrder[O]=mk:form.order_response,
Bill[O]=NS:form.bill, ConfPayment[I]=NS:notice.conf
Ship[O]=NS:notice.shipment, ConfDelivery[I]=NS:notice.conf,
B:

```
(SEQ,
    EstPrice, AnsPrice,
    Order,
    (CHO,
        DenyOrder,
        (SEQ, Question, Order, DenyOrder),
        (SEQ,
            (CHO,
                AcceptOrder,
                (SEQ, Question, Order, AcceptOrder) ),
            Ship, ConfDelivery, Bill, ConfPayment    )
)   )
```

Simulation Results

- Intermediate protocol errors are decreased.

	without model reduction				with model reduction			
	System 1		System 2		System 1		System 2	
	total	error	total	error	total	error	total	error
Case 1	72	48	24	0	24	0	24	0
Case 2	12	8	4	0	4	0	4	0
Case 3	18	14	24	20	4	0	4	0
Case 4	16	14	18	16	2	0	2	0
Case 5	4	4	4	4	0	0	0	0

Discussion on Business Resource Definition Matching

- Model reduction algorithm is independent of *match*().
  - Even the trivial *match*() takes effect.
  - More refined *match*() will bring further effect.

- Our initial proposal on improvement of *match*():
  - BRDs are defined in various forms, sometimes in natural language.
  - Introduce a *BRD description* as a BRD's proxy, using Web Ontology technology.
  - A BRD description includes two parts.
    - *Binary relations*: relationship between BRDs such as equivalence, subsumption(inclusion), disjointness(exclusion)
    - *Structure* : structural characteristics of BRD
  - We proposed an initial algorithm for *match*() traversing Internet.

```
<owl:Class rdf:about=”http://www.examp.org/brd008/form.bill”>
  <owl:equivalentClass rdf:resource=“http://www.examp.org/brd008/form.seikyusho”/>
  <rdfs:subClassOf rdf:resource=“http://www.mkik.org/form.request”/>
  <owl:disjointWith rdf:resource=“http://www.examp.org/brd008/notice.shipment”/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=“http://www.struex.com/form.name”/>
    … … … … … …
  </rdfs:subClassOf>
</owl:Class>
```

Binary relations

Structure

Outline of a proposed BRD Matching Algorithm

Step 1 : **search** recursively BRDs referable from BRD_{from} using binary relations
sort BRDs into appropriate sets B_{fe}, B_{fs}, or B_{fd}.

Step 2 : **search** recursively BRDs referable from BRD_{to} using binary relations
sort BRDs into appropriate sets B_{te}, B_{ts}, or B_{td}.

Step 3 : **make a final decision** inspecting these six sorted sets.

Note: B_{fe}, B_{fs}, B_{fd}, B_{te}, B_{ts}, and B_{td} are sets of BRDs.

- B_{fe} as equivalent with BRD_f, B_{fs} as subsumed by BRD_f, and B_{fd} as disjoint with BRD_f
- B_{te} as equivalent with BRD_t, B_{ts} as subsumed by BRD_t, and B_{td} as disjoint with BRD_t

Details of the algorithm

<u>Step 1</u>:

$B_{fe} = \{from\}; B_{fs} = \phi ; B_{fd} = \phi$

$sort(from, B_{fe}, B_{fs}, B_{fd})$

<u>Step 2</u>:

$B_{te} = \{to\}; B_{ts} = \phi ; B_{td} = \phi$

$sort(to, B_{te}, B_{ts}, B_{td})$

<u>Step 3</u>:

if $B_{fe} \cap B_{te} \neq \phi$ **then return** T

if $B_{fs} \cap B_{te} \neq \phi$ **then return** T

if $B_{fe} \cap B_{td} \neq \phi$ **then return** F

if $B_{fs} \cap B_{td} \neq \phi$ **then return** F

if $B_{fd} \cap B_{te} \neq \phi$ **then return** F

if $B_{fd} \cap B_{ts} \neq \phi$ **then return** F

if $B_{fe} \cap B_{ts} \neq \phi$ **then return** U

if $B_{fs} \cap B_{ts} \neq \phi$ **then return** U

if $B_{fd} \cap B_{td} \neq \phi$ **then return** U

return U

$sort(b, E, S, D)$

if not $visible(b)$ **then return**

for each $c \in equivalentClass(b)$ **and** $c \notin E \cup S \cup D$ **:**

if $b \in E$ **then** add c to E; $sort(c, E, S, D)$

if $b \in S$ **then** add c to S; $sort(c, E, S, D)$

if $b \in D$ **then** add c to D; $sort(c, E, S, D)$

for each $c \in subClassOf(b)$ **and** $c \notin E \cup S \cup D$ **:**

if $b \in E$ **then** add c to S; $sort(c, E, S, D)$

if $b \in S$ **then** add c to S; $sort(c, E, S, D)$

for each $c \in disjointWith(b)$ **and** $c \notin E \cup S \cup D$ **:**

if $b \in E$ **then** add c to D; $sort(c, E, S, D)$

if $b \in S$ **then** add c to D; $sort(c, E, S, D)$

return

Note1: *visible(brd)* is a function to resolve if the description *brd* is referable.

Note2: Parameters E, S and D are used as "call by reference".

Conclusions

- The concept of "Model Harmonization" was proposed.
- The formal definition of an exposed model was given.
- The algorithm to harmonize exposed models was provided.
- Its appropriateness and effectiveness were proved through experimental implementation.
- BRD matching was discussed and initial solution was proposed.
- Future works include:
  - expansion of behavior pattern expression.
  - internal error handling
  - improvement of *match*().

END

Thank you very much.

oya@info.shonan-it.ac.jp
mas-ito@complex.eng.hokudai.ac.jp