BTx-Net: A Token Based Dynamic Model for Supporting Consistent Collaborative Business Transactions

Haiyang Sun Jian Yang
Department of Computing
Macquarie University
Sydney, NSW 2109
{hsun,jian}@ics.mq.edu.au

Abstract

Business transaction is about coordinating the flow of information among organizations and linking their business processes, associated with the solutions to ensure the eventual generation of the consistent outcomes. One important issue here is how to maintain consistency for each partner in the collaborative business transaction as well as consistency for the transaction as a whole during the long running business process in the presence of failures and concurrent activities. Existing models and protocols by designing static process control can not support the dynamic behavior of the collaborative business transaction in the complex loosely coupled business environment. In this paper, we present a novel transactional model, ***BTx-Net****, which is based on Hierarchical CPN to manage consistency for business transaction. Quite different from traditional CPN, two types of tokens, Management Token and Application Token, are used in BTx-Net. The correlation of both tokens is used for verifying consistency in business transaction. We believe that BTx-Net provides a theoretical basis for describing and managing collaborative business transaction consistencies.*

1. Introduction

Integrating business applications requires coordinating the flow of processes and information among organizations and linking their support and information systems together into a cohesive whole. Business integration requires loose coupling between distributed systems so that they can interoperate more freely across the Internet and that the collaboration can be established in a highly dynamic fashion and on-demand basis. Nevertheless, such business collaboration is prone to unreliability and inconsistency especially conducted in a loosely coupled distributed environment such as web services [1]. For example, in automotive industry, an ordering process is a business collaboration involving multiple business partners—customer, dealers and manufacturers, and communicating through the autonomous and heterogeneous business applications that include quote inquiry and processing purchase order, etc.

The ordering process begins with a quote inquiry broadcasting from a customer. After receiving the inquiry, dealers will validate the status of the customer. A quote will return if the customer is valid. Then the customer will choose the dealer whose quote feedback is satisfactory. The selected dealer will receive the purchase order from the customer. After checking the stock based on the purchase order, an order acknowledgement together with invoice and payment details will be sent to customer. After the payment is received, the dealer will deliver the car.

We can observe the following features for this collaboration:

1. The ordering process may last days even weeks. Locking "vehicles" long time for a customer who puts an inquiry is impractical. However, without locking mechanism, one vehicle may be "purchased" by two concurrent transactions.
2. Customer, Dealer, and Manufacturer are peer organizations with their own business rules, policies, and internal processes that are agnostic to each other. However, one fault in a collaborative transaction may require recovery support from several partners. Since the processes in individual organization are autonomous, it is difficult to coordinate and manage the recovery process.
3. The asynchronous interactions among Customer, Dealer, and Manufacturer can easily yield inconsistent outcome. For example, the

2007 IEEE International Conference on Services Computing (SCC 2007)
0-7695-2925-9/07 $25.00 © 2007 IEEE

customer may prepare fund after sending purchase order without waiting for the order acknowledgement from the dealer. However, the dealer may decide not to accept the order, which can cause inconsistent order status in these two parties.

Therefore, business collaboration is required to provide transactional support for cross-organizational business interactions in the loosely coupled service environment. Such collaboration with transactional support is called collaborative Business Transaction (BTx), which is a consistent change in the state of business that is driven by well-defined business functions [1]. With transactional support, the business collaboration can: (1) *from system point of view*, guarantee that the overall system consistency is maintained and data integrity is not compromised.[2] (2) *from business point of view*, ensure that the explicitly shared trade objectives among business entities are achieved and the agreed upon conclusion among such entities is reached.[3]

The existing technology such as inter-organizational workflow and Service Oriented Computing (SOC) only provide minimum support for creating reliable business transaction. All of them handle business transactions in terms of services or activities. The limitations of these approaches lie in the fact that

1) The internal activities of a business process are hidden behind the business protocols, and they are not part of the collaborative business transaction management. In other words the current transaction standards only specify coordination elements and it is up to the individual organization to handle the rest. However in the business collaboration world, problem can occur anytime, anywhere. For example, inconsistency can occur inside the organization, can occur in the message exchange between partners, and can occur because of the violation of time constraints. Therefore a cohesive mechanism is needed to link all the activity, service, and interaction together to define and manage consistency in business collaboration.

2) The control flow and message flow are wired together in most workflow languages and service composition language e.g., BPEL. Generally speaking, control flow is determined by business policies and rules, and message flow should be governed by control flow at run time. Once these two flows are mixed together and hard coded in the process specification during design time, we are facing two serious problems:

- We cannot predicate and specify every possible situation in the context of business collaboration, which may depend on the emerging situation during runtime, especially when faults and exceptions occur. Therefore it is not flexible to predefine control flow for collaboration.
- Message flows between activities within an organization and services cross organizations. When facing faults or exceptions, according to existing models and protocols that offer compensation [4], the committed service or activity will be undone, and the relevant messages are transferred reversely to erase their effects. However, it is not clear how to handle those application messages within the committed services which have been passed to other organizations.

Based on the above discussion, we propose a transaction model that (1) separates the control flow from the message flow so that the correctness of message type and sequencing can be verified. Furthermore control flow can direct message flow at runtime based on message properties, organization policies and business rules; (2) divides a business collaboration into three layers with a uniformed representation of transactional support for all the layers so that activities of internal business process and interactions among organizations can be linked together to enforce consistent transaction outcome.

The purpose of the proposed model is for enforcing and managing consistency when faults and exceptions occur. Due to space limit, concurrency issues and exception handling will not be discussed in the paper.

The rest of paper is organized as follow. In section 2, a detailed BTx-Net model is introduced including the formal specification. In section 3, we mainly discuss the related work. A conclusion and discussion of future work is presented in the last section.

2. BTx Net-A Business Transaction Model

BTx-Net is a token based business transaction model that specifies the dynamic behaviors of business transaction at design time and manages the consistency and reliability at runtime. It is developed based on Hierarchical Colored Petri Net (HCPN) [5-6] that will be briefly introduced here.

Petri Net is a net theory advanced by Dr. Petri in 1962 [7]. The advantages of its graphically and mathematically founded modeling formalism with various algorithms for design and analysis [8] make it a good candidate for modeling business transactions.

2007 IEEE International Conference on Services Computing (SCC 2007)
0-7695-2925-9/07 $25.00 © 2007 IEEE

The Petri net (see Fig.1) transfers tokens from one place (P0) to another (P1) through a transition (T0) by using firing rules following the arcs between place and transition. Colored Petri Net (CPN) is an extension of Petri net in which tokens are assigned with values. In business transaction, various types of messages can be transferred within or cross organizations. Therefore, the message types can be represented as colored tokens in CPN. Another extension of Petri Net is Hierarchical Petri Net (HPN) in which different views in supporting different levels of abstraction and refinement can be specified. In this paper, we call it *net refinement* or *refinement* when a transition or place can be represented as one or more HPNs.

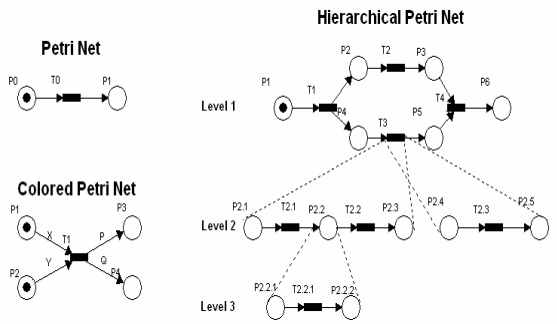

Fig. 1 Various types of Petri Nets

Based on the concept of HCPN, we develop a token based dynamic business transaction model to enforce consistency of business transaction, called ***Business Transaction Net (BTx-Net)***.

2.1 The Structure and Execution Policy of BTx-Net

In BTx-Net a collaborating service is separated into three levels relevant to the stratified structure of web service in business collaboration in terms of internal process, service interface, and business protocol. Each of them is a subnet of the whole BTx-Net:

- At execution level, internal business tasks (activities) are formed as an *Exe-subnet*.
- At abstract level, the input/output messages of the operations defined for service, and the control ability that can transfer messages to and from other subnets form an *Abs-subnet*.
- At communication level, different communication patterns such as *request-response, notify*, form a *Com-subnet.*

There are two types of tokens that are operated within a BTx-Net, the colored Application-Oriented Token (AO-Token) and the Management-Oriented Token (MO-Token), which movements correspond to message flow and control flow respectively. Each MO-token is correlated to a specific AO-token. Here we explain the firing rules for the movement of tokens of the model:

(1) *Token movement at individual level:* An AO-Token can move in each level *iff* the correlated MO-Token in each level exists and moves with it.
(2) *Cross-level token movement:* Every MO-Token splits first into specific level at beginning, moves only within them and meets with other MO-Tokens from other levels when the transaction is successfully terminated. The AO-Token moves between levels determined by the *refinement function* in BTx-Net.
(3) *Cross-organization token movement:* the MO-Token can not move out of the organization, which follows the principle of peer-to-peer collaboration, i.e., no central control exists. However the AO-Token can be exchanged between organizations for the purpose of business collaboration. The MO-Token will be unbound from the correlated AO-Token when it moves out of the organization and bound with it again when the correlated AO-Token returns.

Now we can explain how consistency can be enforced for business transaction through token movements in the stratified service infrastructure based upon the control of intra- and inter-organizational message transferring:

- For the inter-organizational message transferring, the MO-Tokens ensure that the returned AO-Token from other parties is what the organization is expecting. For example, it is not acceptable that the dealer receives the payment message as an AO-Token before accepting the order, because the relevant management token still resides in the *accept order* service, which is a service shall be executed before the *payment* service. Therefore the payment message will be abandoned as an unexpected message. There exists two types of interactions:
  1). for synchronous interaction, the MO-Token will wait for the return of the corresponding AO-Token and examine it to see if it is what the MO-Token expects.
  2). for the asynchronous interaction, the MO-Token needs to examine the AO-Token from asynchronous interaction based on the stored information on previous matched AO-Token.

2007 IEEE International Conference on Services Computing (SCC 2007)
0-7695-2925-9/07 $25.00 © 2007 IEEE

For example the MO-Token needs to correlate the received invoice to the matching purchase order.

- For intra-organizational message movement, the firing rules defined based on organization polices can guarantee that the tasks or operations associated with the AO-tokens movement within and between levels are executed as expected. For example it is impossible to send order final acknowledgement through the communication level for a service when the organization is still processing the order in the *order feedback* service to decide whether the order should be accepted.

2.2 BTx-Net Specification

In this section, we will define business transaction in terms of BTx-Net, its subnets and the refinement rules for the token movement between subnets.

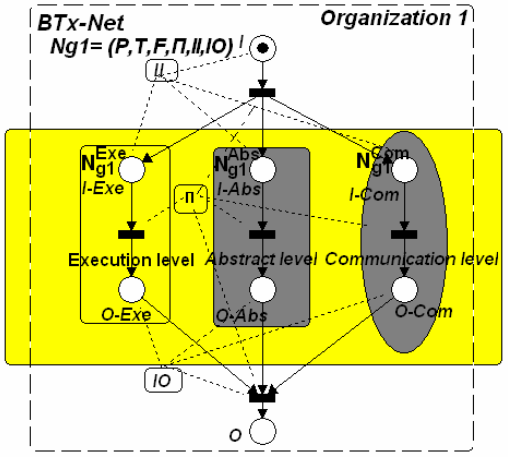

Fig. 2 Overview of BTx-Net

Fig. 2 depicts the business transaction from one organization point of view. Π represents the functions such as the operations or tasks and net refinement functions for each subnet. *II* and *IO* are sets of input and output places for each subnet, N_{gl}^{Exe}, N_{gl}^{Abs} and N_{gl}^{Com} representing the *Exe-subnet*, *Abs-subnet*, and *Com-subnet* respectively.

Definition 1. A BTx-Net in one organization g1 is a tuple N_{gl}=(P, T, F, Π, *II*, *IO*), Where:

- P is a set of places graphically represented as circle in Fig. 2. P_{Exe}, P_{Abs}, and P_{Com} are sets of places at each sub level. $P=P_{Exe}\cup P_{Abs}\cup P_{Com}$, where $P_{Exe}\cap P_{Abs}\cap P_{Com}=\varnothing$.
- T is a set of transitions graphically represented as dark bar in Fig. 2, where: T_{Exe} T_{Abs} and T_{Com} are sets of transitions at each level. T_τ is a set of empty transitions for transferring, distributing, and collecting tokens. $T=T_{Exe}\cup T_{Abs}\cup T_{Com}\cup T_\tau$, where $T_{Exe}\cap T_{Abs}\cap T_{Com}\cap T_\tau=\varnothing$, and $P\cap T=\varnothing$.
- $F\subseteq(P\times V\times T)\cup(T\times V\times P)$ is the flow relation between places and transitions, where V are the sets of variables V= {x,y,….} to represent the colored AO-tokens and MO-Token.
- Π is a group of functions at the three sub levels of a collaboration, which also includes refinement function, MO-token checking function, and colored token map function.
- *II, IO* are the sets of in and out places of BTx-Net and their subnets including $II=\{I, i^{Exe}, i^{Abs}, i^{Com}\}$, and $IO=\{O, o^{Exe}, o^{Abs}, o^{Com}\}$

A BTx-Net is consists of four parts: beginning and ending part, execution level part, abstract level part and communication part. Here below we shall explain the token movement between three levels in BTx-Net. The MO-Token movements at these three levels are independent from each other. However, they are linked together by the movement of AO-Token based upon the *refinement function* at each level.

Definition 2. BTx-Net-*Execution level* is a tuple of N_{gl}^{Exe}= (P_{Exe},T_{Exe}, F, λ, *IK*, *IΛ*, i^{Exe}, o^{Exe}) Where:

- P_{Exe},T_{Exe}, F, i^{Exe}, o^{Exe} are introduced as above.
- λ : $T_{Exe}\rightarrow I\Gamma(V,\Sigma)$ is a formula called *transition guard* which specifies the relation between the input token and output token within the transition. IΓ means list of formulas Σ operated on the token variables V in transition. λ represents the real groups of internal tasks of the service. They work as follows:
  1) Firstly, the formulas in λ representing the internal tasks can be fired *iff,* for x∈V d∈D-- a sets of tokens, ξ(x)=d where ξ: V→D, and φ[ξ] =True where φ is a evaluation formula which denotes the result of the evaluation φ: ξ→Boolean.
  2) Secondly, λ(t)={x+y=z, x•y=w} (where Σ={+,•} representing the internal tasks) with t∈T, x,y,z,w∈V, and there exist $\{s_1,x,t\}\in F$ and $\{s_2,y,t\}\in F$ with $s_1,s_2\in P$ as input, and $\{t,z,s_3\}\in F$ and $\{t,w,s_4\}\in F$ with $s_3,s_4\in P$ as output. λ(t) specifies the relation between input token variables (x, y) and output token variables (w, z).
- IK: $T_{Exe}\cup P_{Exe}\rightarrow$Boolean is a formula called *MO-token guard* that evaluates whether or not a transition or a place should transfer the AO-token to next place or transition. If one AO-token in specific transition or place and the value of IK

(T∪P) is false, it means that there is no existence of relevant MO-token and the AO-token is not expected. Otherwise, the AO-Token is matched with the MO-Token.

- IΛ: $T_{Exe}(Time) \cup P_{Exe}(Time) \rightarrow Boolean$ is a group of time formula called *time guard*. If a time constraint set in transition or place is evaluated as False, then the token will self-terminate or be transferred based on the token movement policy.

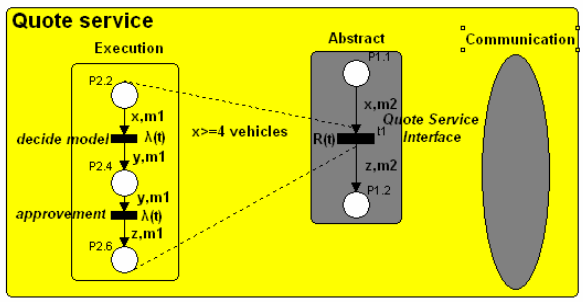

Fig. 3 Refinement from abstract level to execution level

Definition 3: BTx-Net-*Abstract level* is a tuple of N_{gl}^{Abs}= (P_{Abs}, T_{Abs}, F, IK, IΛ, R^{Abs}, i^{Abs}, o^{Abs}), where:

- P_{Abs}, T_{Abs}, F, IK, IΛ, i^{Abs}, o^{Abs} are defined as above.
- R^{Abs}:$T_{Abs} \cup P_{Abs} \rightarrow L$ is a refinement formula on the transition and place in abstract level to connect other two levels. L={g(x),{e(x), N_{gl}^{Exe} : N_{gl} ,r(x)}} $x \in V^{Com}$. g(x) is a function to evaluate the token that is an input of a transition or a place and decides which subnet shall be initiated. Sometimes multiple subnets can be activated simultaneously. e(x) and r(x) are the guard functions of corresponding subnet to evaluate whether or not the subnet is available to initiate and exit.

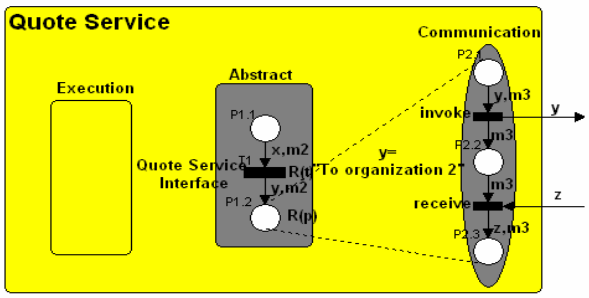

Fig. 4 **Refinement from abstract level to communication level**

Let us take the example in Fig. 3, the *quote service* will be activated when it receives the AO-token *x* with MO-Token m_2. The AO-token will be moved into execution level to execute the tasks. The transition at abstract level needs to decide which subnet to be activated according to the AO token *x*. For example, if x>="4 vehicles" and the guard function in refinement net e(x)=true, then the "*decide order*" with "*approval*" activities will be chosen. AO-Token *x* will be unbound from the MO-Token-m_2 at abstract level and bound with MO-Token-m_1 at execution level. When the "*approval*" task completes and r(x)=true, the AO token *z* will be returned to the transition at abstract level with a new value. Fig. 4 is a similar diagram except that the refinement process is from abstract level to communication level for service communication. Due to space limit, we are not going to discuss it in detail.

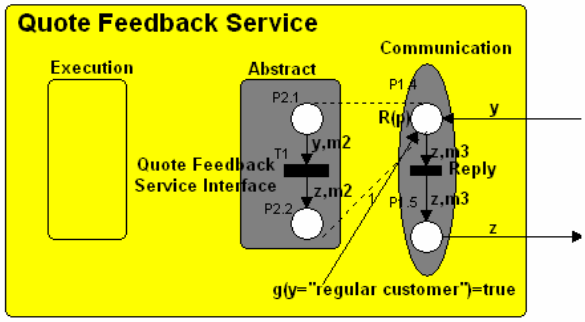

Fig. 5 Refinement from communication level to abstract level

Definition 4. BTx-Net-*Communication level* is a tuple of N_{gl}^{Com}= (P_{Com}, T_{Com}, F, IK, IΛ, R^{ComS}, i^{Com}, o^{Com}), where:

- P_{Com}, T_{Com}, F, IK, IΛ, i^{com}, o^{Com} are defined as above.
- R^{ComS}: $P_{Com} \rightarrow L$ is a refinement formula as similarly defined for abstract level. L={g(x),{e(x), N_{gl}^{Abs}, r(x)}} x∈V. However, only place can be refined at communication level.

Multiple subnets at abstract level can be refined from the communication level depending on which g(x) in each subnet can be satisfied. For example in Fig. 5, if g(y="regular customer")=true, then it only activates the "*quote feedback service interface*" at abstract level. The figure shows that in communication level, the MO-Token m_3 will be unbounded with AO-Token *y* when it is transferred to abstract level where the AO-Token y will be bounded with MO-Token m_2. When the AO-Token *z* is returned to communication level, the m_3 will be bounded with the AO-Token again.

The communication level has four fixed patterns for message exchange and three types of transitions. (see Fig 6) These four patterns are *request-response*, *solicit-response*, *notify* and *one way*. The three transitions are *invoke*, *receive*, and *reply*. For *request-*

response type, one service invokes other service first and receives the result. For *solicit-response* type, the service only replies an invocation. This two are synchronous interactions. In order to ensure the receiving message is what the service expects, MO-token is used to guarantee that the sent messages are matched with the received messages. When the AO-Token is sent out, the correlated MO-token will be transferred to "waiting" place (Place 2.2 in *Request-Response* pattern in Fig. 6). When the AO-token returns from other service, the *receive* transition will be activated if the result is what the service expected. The MO-Token from "waiting" place and the AO-token from other service place will be bound together again. Otherwise the transaction exception mechanism will be activated. *Notify* and *one-way* are two asynchronous interactions. When a service finishes *invoke* or *receive* activity, other services will be activated without waiting for the immediately reply or answer. MO-Token can also guarantee that the AO-Token in *receive* of *one way* pattern is the answer of *invoke* in *notify* pattern if the two types of tokens can be correlated.

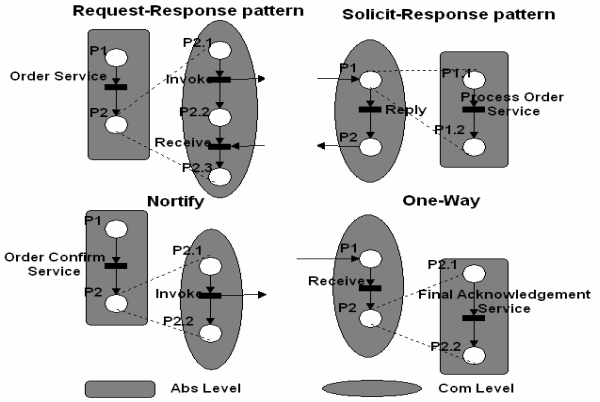

Fig. 6 Communication Patterns

2.3 Token Based BTx-Net Execution

In this section, we will illustrate how business transaction can be executed with the token control mechanism in BTx-Net for the automotive scenario (See Fig. 7). It has following features: (1) Organization 1 broadcasts the quote requirements. However, if there is no timely feedback received from others, for example like organization 3, then organization 1 will close the channel of the interaction with that organization, and moves the MO-Token without AO-Token. (2) The *quote* service will choose one of the internal tasks to execute based on the matching of MO-Token and AO-Token properties, such as "it will require manager approval if the vehicles cost more than $30000", or "directly decide the "car" models if it is less than $30000". (3) At beginning, the MO-Tokens reside at both internal task paths in *Quote Service*. However, only one path will be selected based upon the AO-Token property. For the unselected one, the MO-Toke will self-terminate. Therefore, it guarantees the consistency on the condition that the net structure in execution level in *quote service* begins with *and-slit* and finishes with *or-join*. Only part of the whole transaction is shown in Fig.7. Following are some pseudocode of the execution of organization 1:

G1:="Organization 1";
*x:=G1.**ApplicationToken.Create**();*
*m:=G1.**ManagementToken.Create**(); **Correlate** (m,x);*
While x **Not In** *O* **And** m **Not In** *O* **Do**
CASE
%AO-Token is executing at beginning%
\\1 **x In** *I* **And** m **In** *I* **do Transfer** (x, m, T_τ^{Dis}**);**
\\2 **x In** T_τ^{Dis} **And** m **In** T_τ **do**
begin
m_1*:=**Distribute** (m, "Exe"); **Transfer** (m_1, I^{Exe});*
m_2*:=**Distribute** (m, "Abs"); **Transfer** (m_2, I^{Abs});*
m_3*:=**Distribute** (m, "Com"); **Transfer** (m_3, I^{Com});*
***Transfer**(x ,I^{Abs})*
end;
%AO-Token is executing in Abstract Level%
\\3 x **In** Abs **And** m_1 **In** Exe **And** m_2 **In** Abs **and** m_3 **in** Com **do begin**
If m_2 **In** I^{Abs} **And** x **Not In** I^{Abs} **then transfer** (m_2, t_τ);
If m_2 **In** t_τ **And** x **Not In** t_τ **then transfer** (m_2, $t_\tau^\bullet$);
If IK(ϖ(x**))**=True **then**
Case
\\3.1 x **In** I^{Abs} **do transfer** (m_2,x,t_τ)**;**
\\3.2 x **In** t_τ **do transfer** (m_2,x, $^\bullet t_{Abs}$);
\\3.3 x **In** $^\bullet t_{Abs}$ **do transfer** (m_2,x,t_{Abs});
\\3.4 x **In** t_{Abs} **And** ϒ(x)=True **do**
begin
For i:=1 **to** n **do If** g_i(x)= **True and** e_i(x)=**True then** L=L+$R_i(t_{Abs})$**;**
If L≠∅ **then begin transfer**(x,L);ϒ(x):=**False**
end;
end;
\\3.5 x **In** t_{Abs} **And** ϒ(x)=**False do**
begin Transfer(x,m_2,$t_{Abs}^\bullet$); ϒ(x):=True **end;**
\\3.6 x **In** $t_{Abs}^\bullet$ **And** δ($t_{Abs}^\bullet$)=True **then**
begin
if ϒ(x)=True **then For** i:=1 **to** n **do**
if g_i(x)= **True and** e_i(x)=**True then** L=L+$R_i(P_{Abs})$;
if L≠∅ **then begin transfer**(x,L);ϒ(x):=**False**; **end;**
if ϒ(x)=**False then begin transfe**r (x,m_2,t_τ); ϒ(x):=**True end;**
end;

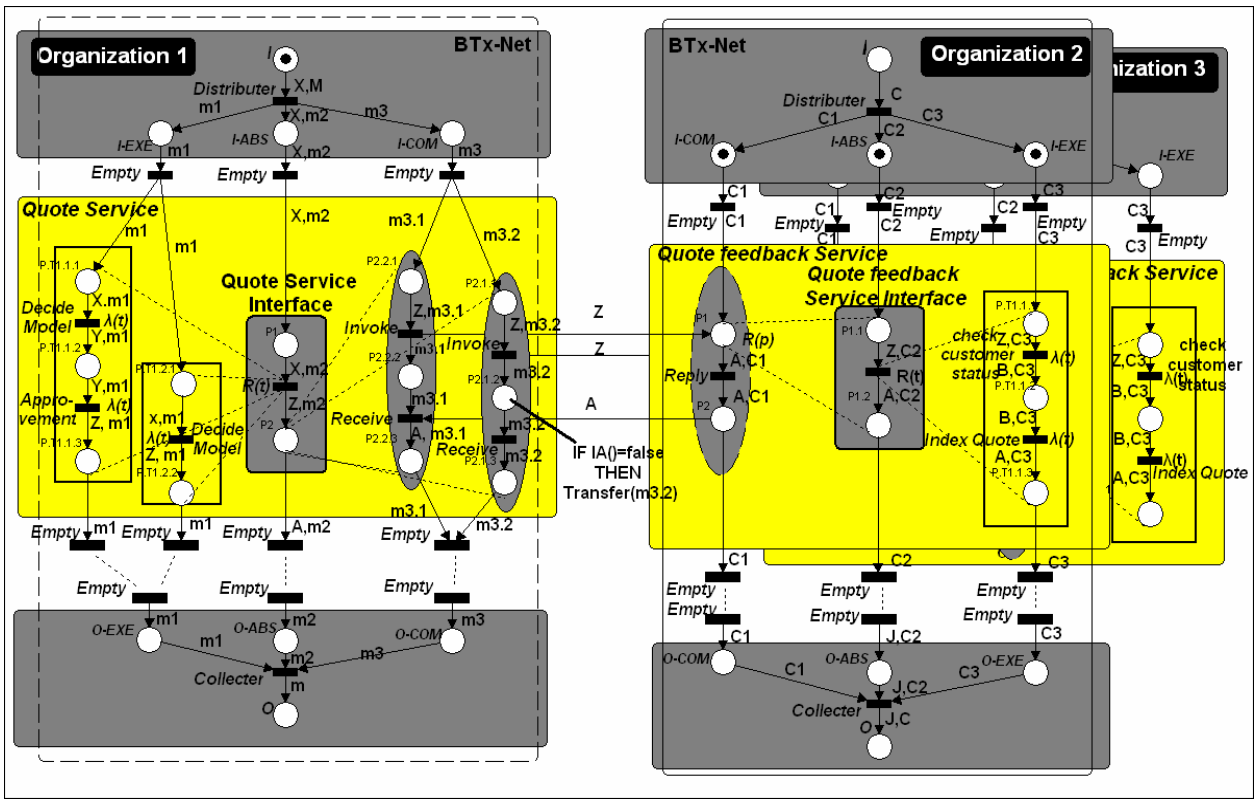

Fig. 7 Execution of two BTx-Nets within Collaborative Organizations

\\3.7 x **In** $t_{Abs}^{\bullet}$ **and** $\delta(t_{Abs}^{\bullet})$=**False do transfer**$(x,m_2,t_\tau)$
end;
end
%AO-Token is executing in Execution Level%
\\4 x **In** Exe **And** m_1 **In** Exe **And** m_2 **In** Abs **and** m_3 **in** Com **do begin**
If m_1 **In** I^{Exe} **then transfer** (m_1,t_τ);
If m_1 **In** t_τ **then transfer** $(m_1,t_\tau^{\bullet})$;
If x **Not In** $t_\tau^{\bullet}$ **And** m_1 **In** $t_\tau^{\bullet}$ **then self-terminate ();**
If IK(ϖ(x**))**=True **then**
Case
\\4.1 x **In** $^{\bullet}t_{Exe}$ **do transfer** (x,m_1,t_{Exe})
\\4.2 x **In** t_{Exe} **do begin** x:=λ(t); **transfer** $(x,m_1,t_{Exe}^{\bullet})$
end;
\\4.3 x **In** $t_{Exe}^{\bullet}$ **do if** r(x)=**True then begin**
transfer (m_1,t_τ)**; transfer** (x,L^{-1})**; end;**
end;
end;
%AO-Token is executing in Communication Level%
\\5 x **In** Com **And** m_1 **In** Exe **And** m_2 **In** Abs **and** m_3 **in** Com **do begin**
If m_3 **In** I^{Com} **then transfer** (m_3,t_τ);
If x **Not In** $t_\tau^{\bullet}$ **And** m_3 **In** $t_\tau^{\bullet}$ **then self-terminate ();**
If m_3 **In** t_τ **then transfer** $(m_3,t_\tau^{\bullet})$;
If IK(ϖ(x**))**=True **then Process Communication**
end;

%AO-Token is executing at ending part%
\\6 x **In** O^{Abs} //if available **And** m_1 **In** O^{Exe} **And** m_2 **In** O^{Abs} **and** m_3 **in** O^{Com} **do transfer(**$x,m_1,m_2,m_3,T_\tau^{Coll}$ **)**
\\7 x **In** T_τ //if available **And** m_1 **In** T_τ **And** m_2 **In** T_τ^{Coll} **and** m_3 **in** T_τ^{Coll} **do begin collect** (m_1,m_2,m_3,m)**; transfer** (m,x,O)**;end;**
END;
Annotation: ① **ϖ:**x→T∪P ② **ϒ:**x→Boolean (before refinement=True, after refinement=False)
③**δ**:$T_{Abs}^{\bullet}$→Boolean (public=True, private=False)
④$^{\bullet}$t are pre-places of t, $t^{\bullet}$ are post-places of t.

3. Related work

Research has been done in the area of specifying and managing reliable and consistent business transactions. We shall discuss some work in two areas: workflow and Service Oriented Computing (SOC).

In workflow community, the authors in [9] model the transactional workflow from atomicity support perspective using WorkFlow-Net [10]. Through reconstructing the process structure by adopting the atomicity pattern, the business process can satisfy the atomicity transaction requirement. In [11], the authors design a self-adaptive workflow system using Petri Net called *Recovery Net*. Recovery net simulates recovery

policy in business transaction when facing faults or exceptions. Obviously, in workflow community, the organization boundary is not respected with the assumption that a leading organization controls the execution of the workflow. Hence, it is not suitable for business transaction in the peer-to-peer environment.

On the other hand, SOC offers infrastructure support for business collaboration in a loosely coupled and peer-to-peer environment. The WS-Net [12] describes the web service components in three levels to simulate the business collaboration. In [13], the authors link the algebraic meta-model to the Petri net representation for describing the service composition. Several standards in SOC are also presented for describing the collaborative business transaction. BPEL4WS [14] is a business process language that orchestrates the services. WS-C[15] and WS-T[16-17] are two associated protocols to provide coordination and transaction support for BPEL. However, (1) the existing approaches in SOC for managing business transaction are based on service interface only. We will not have the knowledge of how internal business process tasks contribute to the exceptions; or vise versa, how the external message exchange causes internal inconsistency. Nevertheless such coarse representation of business collaboration coordination can not provide comprehensive transaction management support. (2) The execution of the business transaction based on SOC is rigorously constrained by the static service transactional control flow at design time and lack of dynamic mechanism to detect, verify, and manage message sequencing at runtime.

In order to overcome the problem mentioned above, we propose a token based transaction model BTx-Net to check and enforce consistency at runtime. Due to space limit, concurrency issues and error handling are not discussed in the paper.

4. Conclusion and Future Research Direction

Business collaboration requires transactional support in loosely-coupled environment. Existing models and protocols can not describe and manage collaborative transaction effectively. In this paper, we present a novel BTx-Net transaction model to describe the business transaction based upon the Hierarchical CPN, which has the following features: (1) We view business collaboration from three levels and utilize management token and application token mechanism to abstract the process control from message flow. (2) The communication patterns with the policy of MO-token movement ensure message matching within the synchronous and asynchronous business interactions, and prevent unexpected messages occurrence.

Currently we are extending the BTx-Net into ***Choreographical BTx-Net (CoBTx-Net)*** which views the global business transaction from individual organizational perspective to verify the reliability of the whole business transaction execution. Transactional MO-Token will be constructed to implement the flexible solutions on the detected unreliable issues.

5. References

[1] M.P. Papazoglou, "Web Services and Business Transactions". *World Wide Web: Internet and Web Information Systems*, 6, 2003, pp. 49–91

[2] G. Paul, F. Alan, J. Julian, and K. Dean, "Compensation is Not Enough", EDOC, 2003, pp.232.

[3] M.P. Papazoglou, "A Business-Aware Web Services Transaction Model". ICSOC, 2006, LNCS 4294, pp.352-364.

[4] H. Garcia-Molina, and K. Salem, "SAGAS", Int. Proc. of ACM SIGMOD Conference on Management of Data, 1987

[5] P. Huber1, K. Jensen1, and R.M.. Shapiro, "Hierarchies in Colored Petri nets", Advances in Petri Nets 1990, LNCS Volume 483, Springer Berlin, 1991, pp. 313-341.

[6] D.G. Stork, and R.J. van Glabbeek, "Token-controlled Place Refinement in Hierarchical Petri Nets with Application Active Document Workflow", ICATPN, 2002, pp. 394～413.

[7] C.A Petri, "Kommunikation mit Automaten". PhD Thesis, Institute fürinstrumentelle Mathematik, Bonn, 1962.

[8] C.Girault, and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Application, Springer*, Germany, 2003.

[9] W. Derks, J. Dehnert, P. Grefen, and W. Jonker, "Customized Atomicity Specification for Transactional Workflows", CODAS, 2001, pp. 140 – 147.

[10] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management", The Journal of Circuits, Systems and Computers 8(1), 1998, pp. 21-66

[11] R. Hamadi, and B. Benatallan, "Recovery Nets: Towards Self-Adaptive Workflow Systems", WISE, 2004.

[12] J. Zhang, C.K. Chang, J.Y. Chund, and S.W. Kim, "WS-Net: a Petri-net Based Specification Model for Web Services", ICWS, 2004, pp. 420 – 427.

[13] R. Hamadi, and B. Benatallan, "A Petri Net-based Model for Web Service Composition", ADC, 2004.

[14] D. Jordan, et al. "Business Process Execution Language for Web Service (BPEL4WS) 2.0", August 2006, http://docs.oasis-open.org/wsbpel/2.0/

[15] E. Newcomer, et al. "Web Services Coordination 1.1 (WS-Coordination)", August 2006, http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-pr-01.pdf.

[16] E. Newcomer, et al. "Web Services Atomic Transaction (WS- Atomic Transaction)", August 2006, http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-pr-01.pdf.

[17] E. Newcomer, et al. "Web Services Business Activity (WS-Business Activity)", November 2006, http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-pr-01.pdf.