

Enforcing Business Collaboration Consistency in Business Transaction Net

Haiyang Sun and Jian Yang

Department of Computing, Macquarie University, Sydney, NSW2109, Australia
 {hsun, jian}@ics.mq.edu.au

Abstract

Managing business collaboration is about coordinating the flow of information among organizations and linking their business processes into a cohesive whole. A consistent outcome is expected within and among involving organization. However, inadequate coordination on the autonomous, heterogeneous, and long-lasting cross-organizational business processes can lead to inconsistent execution in loosely-coupled environment, e.g., Service Oriented Computing (SOC). Therefore, a model that describes business collaboration to enforce consistency for each partner as well as consistency for the collaboration as a whole becomes essential. Previously we proposed a Business Transaction Net(BTx-Net) as a theoretical model to describe business collaboration during design time and manage consistency at run-time execution. In this paper, we will further explore the BTx-Net's enforceability for managing consistency by performing Transactional Management Actions(TMAs) which are needed to be taken during the occurrence of runtime faults and exceptions.

Keywords: Business Collaboration, Consistency Enforcement, Business Transaction Net

1 Introduction

Business collaboration is prone to inconsistency especially when it is conducted in a loosely-coupled distributed environment such as SOC [10]. We refer to *Consistency* with regard to business collaboration as that the status and result of an involving organization's behavior in the business collaboration are what the organization and its collaborators expect.

1.1 A Motivating Example

Let us take an example in the dealer-automotive industry. The ordering process begins with a quote

inquiry broadcasted from a Customer. After receiving an inquiry, Dealers will validate the status of the Customer. A quote will be returned if the Customer has a valid status. Then the Customer will choose a Dealer who offers the best deal. The selected Dealer will receive a purchase order from the Customer. After checking the stock, the Dealer will send the Customer an order acknowledgement together with invoice and payment details. Concurrently, the Dealer will require the Logistics company to arrange the transportation. Once the Dealer receives the payment from the Customer and an acknowledgement of the delivery schedule from Logistics company, the vehicle will be transported to the Customer. The Logistics company will then notify the Customer for the delivery of the vehicle.

1.2 Consistency Enforcement

As shown from the above example, we can observe the following features of this collaboration: (1) **Customer**, **Dealer**, and **Logistics Company** are peer organizations with their own business rules, organization policies and internal processes that are agnostic to each other. (2) The asynchronous interactions among each organization can easily generate inconsistent outcome.

Therefore, the business collaboration is more prone to inconsistent outcomes than single process due to its peer based nature. Furthermore, proper completion of a business collaboration depends not only on correct behavior of one organization, but also the behaviors of the participants observed through complex asyn- and synchronous interactions. Accordingly, the business collaboration requires consistency enforcement: (1) from system point of view, guaranteeing that the overall system consistency is maintained and data integrity is not compromised [3]. (2) from business point of view, ensuring that the explicitly shared trade objectives among business entities are achieved and the agreed upon conclusion among such entities is

reached. [11].

Research has been carried out in the area of business collaboration. However the available approaches have the following limitations:

- Current collaboration standards only specify coordination elements and leave the individual organization to handle the rest, e.g., WS-CDL.
- Control flow and message flow are wired together in most workflow languages and service composition language e.g., WS-BPEL.

In our project, a Business Transaction Net (BTx-Net) was developed based on Hierarchical Colored Petri Net (HCPNs) [12]. It models business collaboration from one organization point of view, which has following features: (1) BTx-Net models the collaborating process in terms of three layers: *internal business process*, *service interface* and *business protocol*. The *internal business process* will provide functional support for service to achieve its objectives. The *service interface* will expose the detailed necessary service information to other collaborating services to achieve service interactions. The *business protocol* is an agreement on the content and sequence of service interactions between collaborating services. BTx-Net uses three subnets to model each layer and links them together to represent individual organization's behavior in the collaboration. (2) BTx-Net separates the control flow from the message flow so that the control flow can dynamically direct the message flow according to the properties of the message at run time, and detect and prevent the advent of the unexpected messages by ensuring the correlation of the elements of control flow and message flow in BTx-Net.

In our previous work [12], the structure and the execution policy of BTx-Net was introduced, and an algorithm for the normal execution of BTx-Net was discussed. In this paper, instead of describing the normal execution of business collaboration, we will focus on how collaboration consistency can be enforced by performing necessary actions with regard to the occurrence of runtime faults and exceptions. We call this type of actions *Transactional Management Actions* (TMAs). *Transition* is a term originated from classical Petri Net which is intuitively used to model the specific functions in process-oriented petri-net-based models. In this paper, we use *Transition* to model the functions of each layer, e.g., internal tasks in private business process layer, or communication tasks in business protocol layer. Furthermore, we will include the TMAs used by BTx-Net to enforce consistency in the transitions to further enhance BTx-Net's capability in dealing with runtime faults and exceptions. A cohesive mechanism

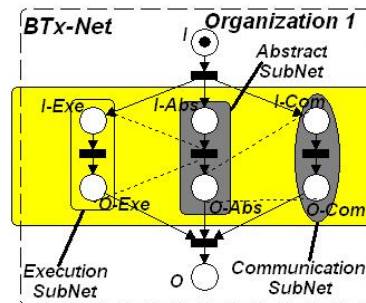


Figure 1. BTx-Net

is thus presented in this paper by coordinating TMAs at each specific layer. Finally, a running example will be introduced to explain how transitions can be used to implement and coordinate TMAs at various layers.

The rest of paper is organized as follows. In section 2, the specification of BTx-Net model is presented, especially the structure and function of transitions are explored. The discussion on related work is presented in section 3, followed by conclusion and future work in section 4.

2 The Specification of BTx-Net model

A BTx-Net models the behavior of an individual participant in a business collaboration. In this paper, we will elaborate BTx-Net based on Dealer's point of view only.

2.1 The Structure of BTx-Net Model

As shown in Fig. 1, there are three levels in BTx-Net corresponding to the generic stratified structure of web service in business collaboration in terms of internal business process, service interface, and business protocol [9]. The relevant three levels are execution level, abstract level, and communication level: (1) At execution level, internal business tasks are formed as an *Exe-subnet*. (2) At abstract level, the input/output messages of the operations defined for the service and the control ability that can transfer messages to and from other subnets form an *Abs-subnet*. (3) At communication level, different communication patterns such as request-response, notify, etc, form a *Com-subnet*.

BTx-Net is developed based on Petri Nets [2], we will use the Petri-Net based terms to model business collaboration. In Petri Net, *place* and *transition* are two key components. They are linked by the *flow relations*. If the *places* before the *transition* have accumulated enough tokens, then the *transition* will be

enabled to pass the tokens to the next *places*. In BTx-Net, we use *places* to model the state of the execution in business collaboration, and use *transitions* to model the function of tasks at each specific subnet. For example, internal activities and communication tasks at the execution subnet and the communication subnet are both modeled as *transitions*. In abstract subnet, *transition* is modeled as service interface. The *flow relations* in BTx-Net represent the flow specification generated at design-time to direct control flow and message flow. In addition to the basic terms of Petri Net, we introduce several new terms implemented in BTx-Net, e.g., *Group functions* and *Input and output places*. The *Group functions* are a series of functions acting on *places* and *transitions* to describe the business collaboration from one organization point of view and to enforce collaboration consistency. The *Input and output places* are two sets of special places indicating the initial and final state of the BTx-Net and its subnets. The token type will be specified in section 2.2.1. We shall present the formal definition of the BTx-Net based on the Petri Nets as follows:

Definition 1 A BTx-Net in one organization G_i is a tuple $NG_i=(P_{G_i}, T_{G_i}, F_{G_i}, \Pi_{G_i}, II_{G_i}, IO_{G_i})$, Where:

- P_{G_i} is a set of places graphically represented as circle in Fig. 1. $P_{G_i}^{Exe}$, $P_{G_i}^{Abs}$, and $P_{G_i}^{Com}$ are sets of places at each subnet. $P_{G_i}=P_{G_i}^{Exe}\cup P_{G_i}^{Abs}\cup P_{G_i}^{Com}$, where $P_{G_i}^{Exe}\cap P_{G_i}^{Abs}\cap P_{G_i}^{Com} = NULL$.
- T_{G_i} is a set of transitions graphically represented as dark bar in Fig. 1, where: $T_{G_i}^{Exe}$, $T_{G_i}^{Abs}$ and $T_{G_i}^{Com}$ are sets of transitions at each level. $T_{G_i}^\tau$ is a set of empty transitions for transferring, distributing, and collecting tokens.

$$T_{G_i} = T_{G_i}^{Exe}\cup T_{G_i}^{Abs}\cup T_{G_i}^{Com}\cup T_{G_i}^\tau, \quad \text{where}$$

$$T_{G_i}^{Exe}\cap T_{G_i}^{Abs}\cap T_{G_i}^{Com}\cap T_{G_i}^\tau = NULL, \text{ and } P_{G_i}\cap T_{G_i} = NULL.$$
- $F_{G_i}=(P_{G_i}\times V\times T_{G_i})\cup(T_{G_i}\times V\times P_{G_i})$ is the flow relation between places and transitions, where V is the sets of variables to represent the tokens.
- Π_{G_i} is a group of functions at the three subnets of a collaborating organization, which includes refinement functions, MO-Token exam functions, and Colored token map functions etc. Here we briefly introduce refinement functions. See [12] for the details of other functions in each subnet. With regard to the **refinement function** from Abs-subnet to Exe-subnet and Com-subnet, $R^{Abs} : T^{Abs} \cup P^{Abs} \rightarrow L$ is a refinement formula on the transition and place in abstract level to connect with other two levels.

$L=\{g(x),\{e(x),N_{g_1}^{Exe}:N_{g_1}^{Com},r(x)\}\} x\in V$. $g(x)$ is a function to evaluate token that is an input of a transition or a place and decides which subnet shall be activated. $e(x)$ and $r(x)$ are the guard functions of corresponding subnet to evaluate whether a subnet is available to be activated and exited.

- II_{G_i}, IO_{G_i} are the sets of input/output places of BTx-Net and their subnets including $II_{G_i}=\{I_{G_i}^{Exe}, I_{G_i}^{Abs}, I_{G_i}^{Com}\}$, and $IO_{G_i}=\{O_{G_i}, O_{G_i}^{Exe}, O_{G_i}^{Abs}, O_{G_i}^{Com}\}$

Here we shall show how to construct a BTx-Net from the motivating example. Firstly, we model the business application as *services* and link them as Abs-subnet. The internal activities and communication tasks are linked and modeled as private business process (Exe-subnet) and business protocol (Com-subnet) respectively. Through the *Refinement functions* in BTx-Net shown in **Definition 1**, the three subnets are linked together to simulate how the *services* work to (1) provide operational function to implement the application, and (2) interact with collaborators to facilitate the business collaboration. Since transitions simulate the behaviors of the BTx-Net, in the following we will present the formal definition of transitions at each subnet based on which the three subnets can be linked and business collaboration consistency can be enforced.

Definition 2 A transition in Exe-subnet of G_i is a tuple $T_{G_i,\nu}^{Exe}=(\nu, \kappa_{In}, \kappa_{Out}, \gamma, \lambda, \vartheta, \varrho, \chi)$, where:

- ν is the ID of the internal activity, e.g T_j ,
- κ_{In} and κ_{Out} represent the input message and output message for the transition,
- γ is a boolean variable of time constraints on the transition, such as $\gamma=True$ if T_1 is executed at the appointed time interval,
- λ is the function operated on the input message and generating the output message,
- ϑ is the list of TMAs used in the transition to deal with runtime faults and exceptions, e.g., $\langle Retry, Compensate \rangle$,
- ϱ is the possible transition in Abs-subnet which the transition in Exe-subnet may link to.
- χ is the set of possible following transitions in Exe-subnet when this transition is finished. (\emptyset means that the transition represents the last activity used to support one specific service.)

For example, the first activity in Dealer-*validate customer status* in Fig. 2 can be written as $T_{Dealer.T_1}^{Exe} = (T_1, \{\text{Customer Name \& Address}\}, \{\text{Customer status}\}, \{\text{True}\}, \{\text{Query(In:Customer Name, In:Address, Out:Customer status)}\}, \langle \text{Retry, Alternatives} \rangle, \{\text{Quote Feedback Service}\}, \{T_1, T_2 \& T_3, \emptyset\})$. It means that the activity of *validating customer status* is labeled as T_1 in Exe-subnet at *Quote Feedback* service in Dealer. The output message *customer status* is generated through function *Query()* by parameterizing the input message *Customer Name* and *Address*. The activity is ready to be activated now. If the activity is failed, it may *retry* firstly. An *alternative* way may be used (e.g. requiring Customer to register firstly) if the *retry* still does not succeed. The activity belongs to the *Quote Feedback* service and the next activity following this one can be T_1 , or T_2 followed by T_3 as normal execution, or none as alternative way to notify the Customer that it fails to query the status.

Definition 3 A transition in Abs-Subnet of G_i is a tuple $T_{G_i,\varsigma}^{Abs} = (\varsigma, \kappa_{In}, \kappa_{Out}, \gamma, \psi, \vartheta, \chi, \beta)$ where:

- ς is the service ID,
- κ_{In} and κ_{Out} are sets of the input and output messages of this service,
- γ is a boolean variable of time constraints on the service,
- ψ is the communication types for each input and output messages, such as {request-response, solicit-response, notify, one way}
- ϑ is the list of TMAs used in the transition to deal with runtime faults and exceptions, e.g., $\langle \text{Retry} \rangle$,
- χ is the sets of possible following services,
- β is the sets of logic of internal activities.

For instance, the *Quote Feedback* service in Dealer can be written as $T_{Dealer.S_1}^{Abs} = (S_1, \{\text{Quote Enquiry}\}, \{\text{Quote Result}\}, \{\text{True}\}, \{\text{Solicit-response}\}, \langle \text{Retry, Alternative} \rangle, \{S_2, S_{10}\}, \{T_1 \mapsto T_2 \mapsto T_3, \text{OR } T_1\})$. It means that the *Quote Feedback* service in Dealer is the initial service in ordering process identified as S_1 . It is in the active status at this moment. It provides *solicit-response* interaction pattern by inputting *quote enquiry* and outputting *quote result*. If the quote can not be found, the service will *retry* it first. If it fails again, the service will use alternative means, such as activating service S_{10} to require that the Customer should register first (S_{10} is a *Customer Register* service which is not described in motivating scenario but exists in

the collaboration as an alternative application following the *Quote Feedback* service). If the service is executed normally, the service S_2 will be activated. No other service can be executed after S_1 . The service interface can be realized by two internal business processes: (1) a sequence of activities from T_1 to T_3 or (2) T_1 only. The service can be achieved by any one of them (depending on what the situation is at the time of executing).

Definition 4 A transition in Com-Subnet of G_i is a tuple $T_{G_i,\alpha}^{Com} = (\alpha, \delta, \kappa_{In}, \kappa_{Out}, \gamma, \vartheta, \varrho, \chi)$ where:

- α is the communication behavior ID
- δ is the communication behavior type, e.g. {Invoke, Receive, Reply},
- κ_{In} and κ_{Out} are sets of the input and output messages of this communication behavior,
- γ is a boolean variable of time constraints on this communication behavior,
- ϑ is the list of TMAs used in the transition to deal with runtime faults and exceptions, e.g., $\langle \text{Compensate} \rangle$,
- ϱ is the possible service the transition may belong to,
- χ is the sets of possible following communication behavior.

The communication tasks in Dealer which belong to *Quote Feedback* service can be written as $T_{Dealer.C_1}^{Com} = (C_1, \{\text{Reply}\}, \{\text{Quote Enquiry}\}, \{\text{Quote Result}\}, \{\text{True}\}, \langle \text{Retry} \rangle, \{\text{Quote feedback service}\}, \emptyset)$. It means that the transition represents two communication tasks in the synchronous business interaction to receive *quote enquiry* first and then send *Quote result*. The communication tasks are available to operate now. If it fails to reply, it will continue to *retry* processing the message until it is successful. No other communication behaviors belonging to *Quote Feedback* service exist after this one.

Based on the structure and function of transitions at each specific subnet in the BTx-Net model defined above, we can observe that:

- each independent specific subnet representing different layer in business collaboration is linked together within one organization according to the structure of each transition. The transitions also encapsulate TMAs in dealing with runtime faults and exceptions. Therefore, it is possible for the

organization to enforce business collaboration consistency among each layer by performing TMAs at each subnet if an error occurs.

- the Abs-subnet and Com-subnet can be observed by other collaborators' BTx-Net. The TMAs of each transition at these two subnets can also be observed by collaborators. Consequently, the enforceability on collaboration consistency among organizations can be achieved if the TMAs' behavior in one organization can be supported by collaborators' BTx-Net behaviors. For example, the consistency enforcement among organizations is realized if the Customer understands that the *Register* service in Customer should be activated once the Dealer rejects the quote enquiry due to the absence of the customer information. In this case, the Customer's "understanding" of TMAs- "Alternative" specified in Dealer's *Quote Feedback* service guarantees that both parties will behave as expected.

2.2 BTx-Net Execution

2.2.1 Execution Policy of BTx-Net Model

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

- Token movement at individual level: An AO-Token can move within each level *iff* the correlated MO-Token in each level exists and moves with it.
- Cross-level token movement: Every MO-Token is split into each level at the beginning of the business collaboration and converged at the end. Such split MO-Token can only move within the specific level at runtime. However, the AO-Token moves between levels according to the *refinement functions* defined in BTx-Net.
- Cross-organization token movement: The MO-Token can not move out of the organization. However the AO-Token can be exchanged between organizations for the purpose of business collaboration.

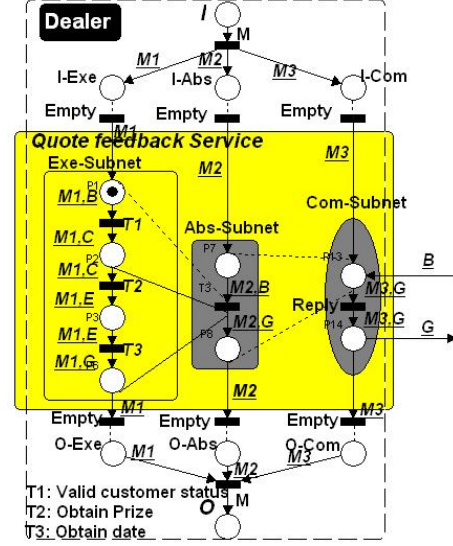


Figure 2. BTx-Net of Dealer

2.2.2 An Example of BTx-Net's Execution

In this section, we will present a running example to explore the structure and function of the transitions at each specific subnet in BTx-Net model. The example will be discussed by following the movement of AO-Token within one organization. The AO-Token will be firstly received by the Com-subnet and eventually sent out of the organization after processing by the transitions at each subnet. Through this example, we can observe how transitions work at each specific subnet and how they can be used to deal with runtime faults and exceptions.

Due to space limit, we only provide an example on how the *Quote Feedback* service is constructed and how it is linked with other services. Fig. 2 depicts a BTx-Net of Dealer in normal execution derived from the motivating scenario. $M_1 \sim M_3$ represent the MO-Tokens, and $B \sim G$ denote the AO-Tokens. The control flow of this business process is identified by the movement of MO-Token. Similarly, the message flow can be represented by the transferring of AO-Tokens. We can find that the *Quote Feedback* service in Dealer includes one service interface, one service communication behavior (*reply*), and three internal activities (Validate customer status(T_1), Obtain price(T_2), Obtain date(T_3)). The internal activities are terminated if T_1 is finished or T_3 is achieved which is shown as the solid line between the Exe-subnet and Abs-subnet (The dash line shows the initiation of the internal business process). The three transitions representing internal activities are listed below:

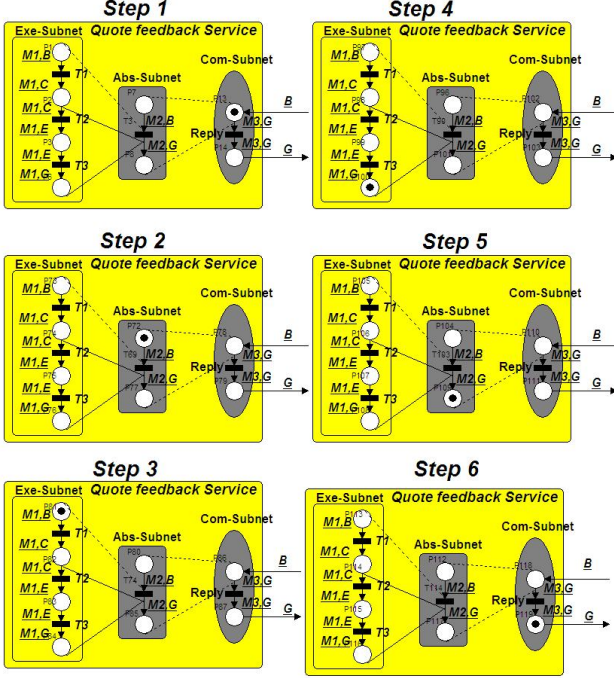


Figure 3. Steps of BTx-Net Execution

1. $T_{Dealer.T_1}^{Exe} = (T_1, \{\text{Customer Name \& Address}, \{\text{Customer status}\}, \{\text{True}\}, \{\text{Query}(\text{In:Customer Name, In:Address, Out:Customer status})\}, \langle \text{Retry}, \text{Alternatives} \rangle, \{\text{Quote Feedback Service}\}, \{T_1, T_2 \& T_3, \emptyset\})$
2. $T_{Dealer.T_2}^{Exe} = (T_2, \{\text{Vehicle model \& vehicle color}\}, \{\text{Prize}\}, \{\text{True}\}, \{\text{Query}(\text{In:Vehicle model, In:vehicle color, Out:Prize})\}, \langle \text{Retry}, \text{Alternatives} \rangle, \{\text{Quote Feedback Service}\}, \{T_2, T_3\})$
3. $T_{Dealer.T_3}^{Exe} = (T_3, \{\text{Vehicle model \& vehicle color}\}, \{\text{quote date}\}, \{\text{True}\}, \{\text{Query}(\text{In:Vehicle model, In:vehicle color, Out:quote available date})\}, \langle \text{Retry}, \text{Alternatives} \rangle, \{\text{Quote Feedback Service}\}, \{T_3\})$

Now let us exam how this service can communicate with other collaborators using transitions in service interface and business protocol layer. The relevant transitions for such two layers are listed as follow:

- $T_{Dealer.S_1}^{Abs} = (S_1, \{\text{Quote enquiry}\}, \{\text{Quote result}\}, \{\text{True}\}, \{\text{Solicit-response}\}, \langle \text{Retry}, \text{Alternative} \rangle, \{S_2, S_{10}\}, \{T_1 \mapsto T_2 \mapsto T_3, \text{OR } T_1\})$.
- $T_{Dealer.C_1}^{Com} = (C_1, \{\text{Reply}\}, \{\text{Quote enquiry}\}, \{\text{Quote result}\}, \{\text{True}\}, \langle \text{Retry} \rangle, \{\text{Quote feedback service}\}, \emptyset)$.

After defining the structure of the transitions constructing *Quote Feedback* service, let us take a look at how the service is executed based upon the runtime information. Here we will look into how transitions can be grouped together to contribute to the operation of the BTx-Net when facing runtime faults and exceptions.

Step 1: AO-Token in the Com-subnet (Receive) Let us assume that the Customer is sending an AO-Token B as quote enquiry to the Dealer. $T_{Dealer.C_1}^{Com}.\delta$ is $\{\text{Reply}\}$ which means that the transition $T_{Dealer.C_1}^{Com}$ in Dealer represents two communication tasks *receive* and *send* as in the business protocol. If relevant MO-Token exists in the place $\bullet T_{Dealer.C_1}^{Com}$ (a place just before the transition $T_{Dealer.C_1}^{Com}$) and the content of the expected input message $T_{Dealer.C_1}^{Com}.\kappa In$ equals to the AO-Token, then the AO-Token B can be accepted as the expected input message. That the service identifier $T_{Dealer.C_1}^{Com}.\rho$ is $\{\text{Quote Feedback service}\}$ indicates that the AO-Token B needs to be transferred to *Quote Feedback* service interface $T_{Dealer.S_1}^{Abs}$. We assume that the time constraint is not violated as $T_{Dealer.C_1}^{Com}.\gamma = \text{True}$. However, if the transition $T_{Dealer.C_1}^{Com}$ fails to receive this message due to communication faults, it will *Retry* to contact the collaborator to re-send the message.

Step 2: AO-Token in the Abs-subnet (Receive) Once the AO-Token B is transferred to the transition $T_{Dealer.S_1}^{Abs}$ in Abs-subnet, the $T_{Dealer.S_1}^{Abs}$ will examine if the relevant MO-Token exists and the required input message $T_{Dealer.S_1}^{Abs}.\kappa In$ equals to the AO-Token. If everything is satisfied, the transition $T_{Dealer.S_1}^{Abs}$ will move the AO-Token B into its internal business process by calling the internal business process identified as $T_{Dealer.S_1}^{Abs}.\beta$.

Step 3 AO-Token in the Exe-subnet (Receive) When the AO-Token B is moved to the *Validate customer status* ($T_{Dealer.T_1}^{Exe}$), relevant checking will be made to ensure that the AO-Token is what the activity is expecting and that the activity is available to process the AO-Token. After extracting the *Customer Name \& Address* from the AO-Token and inputting them into the *Query()* function (an application function), AO-Token C is generated corresponding to *Customer status*, and the following transition $T_{Dealer.T_2}^{Exe}$ is then activated. However, if the query can not find the Customer status in the system, then according to $T_{Dealer.T_1}^{Exe}.\vartheta$ list of TMAs, a *Retry* action will be executed by activating T_1 in the *following activity set* identified as $T_{Dealer.T_1}^{Exe}.\chi$. If it fails again, an *Alternative* action in the second position in the list of TMAs $T_{Dealer.T_1}^{Exe}.\vartheta$ is performed by activating \emptyset in the *follow-*

ing activity set $T_{Dealer.T_1}^{Exe} \cdot \chi$.

Step 4: AO-Token in the Exe-subnet (Send) and Step 5 AO-Token in the Abs-subnet (Send)

Here we assume that the internal business process of *Quote Feedback* service is executed successfully. When the final transition $T_{Dealer.T_3}^{Exe}$ is achieved in the internal business process, the AO-Token G including the quote result will be returned to the transition $T_{Dealer.S_1}^{Abs}$. After checking the existence of the relevant MO-Token and if the AO-Token equals to the output message $T_{Dealer.S_1}^{Abs} \cdot \kappa_{Out}$, the service interface will return the AO-Token to the transition $T_{Dealer.C_1}^{Com}$ in business protocol since the *communication pattern identifier* $T_{Dealer.S_1}^{Abs} \cdot \psi$ equals to {Solicit-response} suggesting that a returned message is required. If the service fails, a relevant action in the TMAs list will be implemented. Based upon the properties of the returned AO-Token from internal business process, the next available service will be activated. For example, if the Customer exists, the transition $T_{Dealer.S_2}^{Abs}$ representing *Ordering* service will be activated. However, if the Customer does not exist, *Customer Register* service modeled as transition $T_{Dealer.S_{10}}^{Abs}$ will be executed.

Step 6: AO-Token in the Com-subnet (Send)

If the output message $T_{Dealer.C_1}^{Com} \cdot \kappa_{Out}$ =AO-Token and the MO-Token exists, then the AO-Token will be transferred to the communication level of BTx-Nets of the collaborating partner. Any failure will be processed and encapsulated in the communication level. As soon as the *reply* action is performed, relevant MO-Token in this service will move to the next service based upon the decision of transition $T_{Dealer.S_1}^{Abs}$ to continue the business collaboration.

2.3 The Advantages of BTx-Net Model

2.3.1 Consistent Message Transfer Support

Now we can explain how consistency can be enforced in business collaboration through token movements in the stratified service infrastructure based on the control of intra- and inter-organizational message transfer:

- For the inter-organizational message transfer, 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 that 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 **asynchronous interaction**, the MO-Token needs to examine the AO-Token from asynchronous interaction based on the stored information on previous matched AO-Token. 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 layers are executed as expected. For example it is impossible to send order final acknowledgement through the communication layer from a service when the organization is still processing the order in the *Order Feedback* service to decide whether the order should be accepted or not.

2.3.2 Consistent Collaboration Management Support

BTx-Net provides an infrastructure to maintain the consistency of individual organization as well as the whole business collaboration through token movements as discussed in the previous section. Between organizations collaboration consistency can also be enforced using TMAs defined in BTx-Net. For example, the service interface layer and business protocol layer in one organization (Dealer) can be observed by its collaborator (Customer). Therefore the TMAs in these two layers can be identified and examined by these two collaborating partners. If the Dealer can not proceed with executing *Quote Feedback* service due to lack of customer information, it will try "alternative" as specified in its TMA, which is *Customer Register* service (see the example in Section 2.2.2). This alternative service execution in Dealer will lead to alternative service execution at the Customer side as well through message token transferring between the Dealer and the Customer. As a result the *Register* service will be activated at the Customer side rather than normal execution-*Place Order* service in Customer. This is how consistency is enforced in the BTx-Net model.

3 Related Work

Research has been done in the area of modeling business collaboration. Petri Net is a widely used technique for business process modeling. We shall look into some of the representative work in the area.

Authors in [5] advocate modeling service orchestration based on Petri Net. They simply use *control place* to compose service and *orchestration place* to choreograph the interaction between organizations. However, they do not clearly identify how internal activities support the service interface composition and collaboration. The WS-Net [13] (Web Service Net) describes the web service components in three levels to simulate the business collaboration. But the work is lack of dynamic mechanism to govern the message flow. The control flows governing the application message that are interacted between intra- and inter-organization are both predefined. In [4], the authors link the algebraic meta-model to Petri Net representation for describing service composition. Unfortunately, the authors only present a mathematical formalism to describe the coarse-grained service composition without: (1) any introduction on the internal activity and external communication support and (2) any guarantee on message transferring consistency.

Several standards in SOC are also available for describing and managing business collaboration consistency. WS-BPEL [1] together with WS-C [8] and WS-T that includes Atomic Transaction [6] and Business Activity [7] provides some basic support for coordination and exception handling. WS-BPEL is a business process language that orchestrates services. WS-C and WS-T are two associated protocols to provide coordination and transaction support for WS-BPEL. However, they only specify business collaboration at abstract level by specifying coordination elements for service interface and leave the rest to the individual organization to handle.

As a summary, we need a language for orchestrating autonomous service as well as mechanism for coordinating and managing business collaboration in the context of SOC. However: (1) when error or exception occurs, we will not have the knowledge of how internal business process tasks contribute to the problem; or vice versa, how the external message exchange causes internal inconsistency, let alone handle the error. (2) The execution of business collaboration based on SOC is rigorously constrained by the static service control flow at design time and lack of dynamic mechanism to detect, verify, and manage message sequencing at runtime.

4 Conclusion and Future Work

Business collaboration requires support to maintain consistency in loosely-coupled environment. Existing models and protocols can not describe and manage business collaboration effectively. In this paper, we ex-

plore the structure and function of transitions at each specific subnet in BTx-Net model to enforce business collaboration consistency in dealing with runtime faults and exceptions.

In the future, how to dynamically execute TMAs based on runtime situations rather than TMAs list created at design time will be challenge. Therefore, a novel *Rule-Tokens* will be studied to choose actions from TMA based on runtime transaction constraints. The operator between AO-Token and Rule-Token will also be defined to facilitate the dynamical rule-driven business collaboration execution.

References

- [1] W. T. Committee. Web services business process execution language version 2.0 (ws-bpel), <http://docs.oasis-open.org/wsbpel/2.0/cs01/wsbpel-v2.0-cs01.html>. 2007.
- [2] C. Girault and R. Valk. *Petri Nets for System Engineering: A Guide to Modeling, Verification and Application*. Springer, 2003.
- [3] P. Greenfield, A. Fekete, J. Jang, and D. Kuo. Compensation is not enough. In *Proceedings of the Seventh IEEE International EDOC Conference*, pages 232–239, 2003.
- [4] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In *Proceedings of the 14th ADC*, pages 191–200, 2004.
- [5] M. Mecella, F. Presicce, and B. Pernici. Modeling e-service orchestration through petri net. In *Proceedings of The Third International Workshop on TES*, pages 109–134, 2002.
- [6] E. Newcomer and et al. Web services atomic transaction (ws-atomic transaction), <http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-pr-01.pdf>. 2006.
- [7] E. Newcomer and et al. Web services business activity (ws-business activity), <http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-pr-01.pdf>. 2006.
- [8] E. Newcomer and et al. Web services coordination 1.1 (ws-coordination), <http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-pr-01.pdf>. 2006.
- [9] E. Newcomer and G. Lomow. *Understanding SOA with Web Service*. Pearson Education, 2005.
- [10] M. P. Papazoglou. Web services and business transactions. *World Wide Web: Internet and Web Information Systems*, 6:49–91, 2003.
- [11] M. P. Papazoglou. A business-aware web services transaction model. In *Proceedings of the 4th ICSOC*, pages 352–364, 2006.
- [12] H. Sun and J. Yang. Btx-net: A token based dynamic model for supporting consistent collaborative business transactions. In *Proceedings of SCC*, pages 490–497, 2007.
- [13] J. Zhang, C. Chang, J. Chund, and S. Kim. Ws-net: a petri-net based specification model for web services. In *Proceedings of ICWS*, pages 420–427, 2004.