
UNIVERSITY
OF WOLLONGONG
AUSTRALIA

University of Wollongong
Research Online

Faculty of Engineering - Papers (Archive) Faculty of Engineering and Information Sciences

2007

TiCoBTx-Net: a model to manage temporal consistency of service oriented business collaboration

Haiyang Sun
Macquarie University

Jian Yang
Macquarie University

Weiliang Zhao
University of Wollongong, wzhao@uow.edu.au

Jianwen Su
University Of California

http://ro.uow.edu.au/engpapers/5018

Publication Details

Sun, H., Yang, J., Zhao, W. & Su, J. (2007). TiCoBTx-Net: a model to manage temporal consistency of service oriented business collaboration. IEEE Transactions on Services Computing, 5 (2), 207-219.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

*Ti*CoBTx-Net: A Model to Manage Temporal Consistency of Service-Oriented Business Collaboration

Haiyang Sun, Jian Yang, *Member*, *IEEE*, Weiliang Zhao, and Jianwen Su, *Senior Member*, *IEEE*

Abstract—Business collaboration is about coordinating the flow of information among organizations and linking their business processes into a cohesive whole. Collaborative business processes are time critical within and across organizations and can become unreliable due to temporal inconsistency where processes cannot execute according to the agreed temporal policies. It is necessary to have a mechanism to manage temporal consistency in service-oriented business collaboration. In this paper, we propose a model named **Ti**med **Cho**reographical **B**usiness **T**ransaction **N**et (*Ti*CoBTx-Net) based on Hierarchical Colored Petri Net for individual business participants to specify and manage the temporal consistency in business collaboration. A series of temporal polices are formalized and checked in *Ti*CoBTx-Net to enforce the temporal consistency at design time and runtime. A verification mechanism is also developed to clarify the status of temporal inconsistencies. Finally, the implementation details of the proposed mechanism is provided.

Index Terms—TiCoBTx-Net, temporal consistency, temporal policies, business collaboration, web service.

1 Introduction

Business collaboration is about coordinating the flow of information among organizations and linking their business processes into a cohesive whole. A collaborative business process operates in a distributed environment involving multiple parties with dynamic availability, and a large number of heterogeneous sources with evolving contents [1]. A consistent outcome is expected within and among the parties involved. The service technologies as the foundation to build up consistent business collaboration have an aim to allow effective composition of discrete services or processes into end-to-end service aggregation on a global scale. A consistent business collaboration is supported by the service systems through the creation of alliances between service providers, each offering services to be used or syndicated with other external services [2]. As a result, services become the building blocks of collaborative business processes [3].

Inconsistency in business collaboration can be caused by many reasons, e.g., structure incompatibility (the participating processes do not bear consistent process logic) or behavior nonconformance (the behaviors of participating processes are not acted as agreed by others). In this paper, we mainly focus on temporal inconsistency in business collaboration. Business collaboration is time critical within and across organizations. For example, a delay will cause cascading delays that can affect multiple business participants. Deadlocks may happen for the mutual waiting of each participant's results in collaboration. Temporal inconsistency occurs since the participating processes in business collaboration fail to coordinate the temporal policies within and across business participants, i.e., specific service in the participating processes cannot execute in a properly temporal manner.

Temporal polices are developed by individual business participants to restrict when the web services in participating processes can start and finish. When services are involved in business collaboration, the following must be guaranteed:

- All services in the common participating process can successfully execute within their available temporal intervals. If there exists a service that cannot execute within its available temporal interval, then the temporal policies of the service are not matched with those of other related services in the common participating process in business collaboration.
- When a service is interacting with another service of a collaborative partner, the temporal policies of these services must be matched with each other. These services must start and finish within their available temporal intervals when they are interacting.

The coordination on temporal policies within and across participating processes is critical to ensure the smooth execution of the business collaboration. Deadlock caused by temporal inconsistency in business collaboration must be avoided.

For example, in Fig. 1, Processes A and B are syndicated by services in different organizations, respectively, and interactions can be identified between 1) Services A and B, and 2) Services A+ and B+. Messages are transferred between services in the sequences 1 to 4. We can observe the occurrence of deadlock caused by temporal inconsistency as follows: The start time of Service A is missing and the

- *H. Sun, J. Yang, and W. Zhao are with the Department of Computing, Macquarie University, Sydney, NSW 2122, Australia. E-mail: {haiyang.sun, jian.yang, weiliang.zhao}@mq.edu.au.*
- *J. Su is with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 93106-5110. E-mail: su@cs.ucsb.edu.*

Manuscript received 20 Apr. 2010; revised 29 Sept. 2010; accepted 25 Dec. 2010; published online 2 Feb. 2011.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSCSI-2010-04-0066. Digital Object Identifier no. 10.1109/TSC.2011.5.

1939-1374/12/$31.00 © 2012 IEEE Published by the IEEE Computer Society

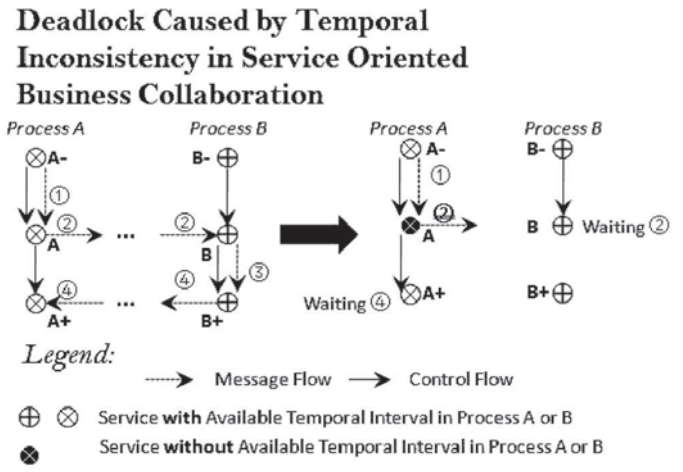

Fig. 1. Deadlock example.

service cannot interact with Service B in collaborative partner when the previous service—Service A^-—in common participating business process is finished, i.e., message 2 cannot be delivered to Service B. The next available service—Service A^+—in the participating process is ready to start, i.e., receiving message from Service B^+, the next service after Service B in collaborative partner. Service B^+ cannot be initiated if Service B does not finish. A deadlock occurs because Service A^+ is waiting message from Services B^+ and B needs message from Service A. We can observe from the above example, that although Service A^- finishes execution within its available temporal interval, the Service A syndicated with Service A^- still cannot be initiated since its available temporal interval has passed. This could happen if the temporal policies coordination on Services A and A^- in the common participating process is missing.

Managing temporal consistency in business collaboration is challenging in the service-oriented environment

- Temporal inconsistency can occur inside the organization, can occur in the service interaction between partners, and can occur due to the violation of temporal policies at a specific service. It is necessary to develop a mechanism to consider all the activities, services, and interactions together for defining and managing temporal consistency in business collaboration.
- Temporal consistency management is difficult in the peer-based collaborative business process that has multiple participating business processes running independently and exchanging asynchronous messages among them. It has the following features: (1) Temporal policies are defined by individual peer-based business participants and cannot be controlled by others. Each participant can only manage temporal policies coordination from one organization point of view in business collaboration. (2) Temporal policies are hidden inside the organization boundary of an individual participant. Temporal policy coordination can only be performed on an organization's internal business processes, service interfaces, and protocols, as well as the interfaces and protocols of its collaborative partners.
- Temporal consistency management in business collaboration is required not only at design time but also at runtime. The management at design time ensures that the temporal policies coordination at collaborative business is carried out before hand. However, it is extremely difficult to know all possible temporal parameters of participants, e.g., start time or finish time, before hand in the dynamic, distributed, loosely coupled environment. Therefore, the runtime dynamic temporal policy coordination becomes essential to enforce temporal consistency for collaborative business process.

It is critical to develop models and mechanisms to effectively check and enforce temporal consistency during business collaboration. A number of research have been carried out to deal with temporal consistency in collaborative business process [6], [7], [14], [15], [28]. Existing models have the limitations either being constructed from a centralized global view which includes all the detailed information of participants, or simply specifying business collaboration without considering details of internal business process. In [19], [20], [22], [23], [24], [25], multiple methods are proposed to manage temporal consistency. However, it is still lacking of a solution that includes algorithms for the runtime dynamic temporal policy coordination.

In this paper, we propose a model named **Ti**med **Cho**reographical **B**usiness **T**ransaction **N**et (*Ti*CoBTx-Net) for individual business participants to specify and manage the temporal consistency in business collaboration. The proposed model is based on Hierarchical Colored Petri Net [4], [5], [8], [9], [16], [17], [26]. Temporal policies are processed in *Ti*CoBTx-Net to enforce the temporal consistency at both design time and runtime. The business collaboration is modeled by the *Ti*CoBTx-Net controlled by two types of execution policies as behavior policies (based on business process logic) and temporal policies (based on temporal logic). These policies are employed to guarantee the correct operation of business collaboration. The Hierarchical Timed Computation Tree Logic (*Hi*TCTL) is proposed to formalize the temporal policies and specify when and where these temporal policies can be effective on *Ti*CoBTx-Net. Algorithms are developed to check if temporal policies are matched with each other in business collaboration considering related internal activities, web services, and communication tasks. The verification mechanism is devised to detect the status of temporal inconsistency in business collaboration.

Our approach addresses the temporal inconsistency issue in business collaboration. The existing approaches are normally design time-based solutions and they have an inherent limitation to cover the dynamic temporal dependencies of involved services in a business collaboration. The proposed work presented in this paper is beyond the existing approaches [6], [7], [28] due to its following main contributions:

1. The proposed approach can provide a generic solution to address the temporal inconsistency issue in business collaboration by considering internal activities, web services, and communication tasks;
2. The proposed *Ti*CoBTx-Net model has the capability to put the concerns of temporal inconsistency at the design time and the runtime under the same umbrella;

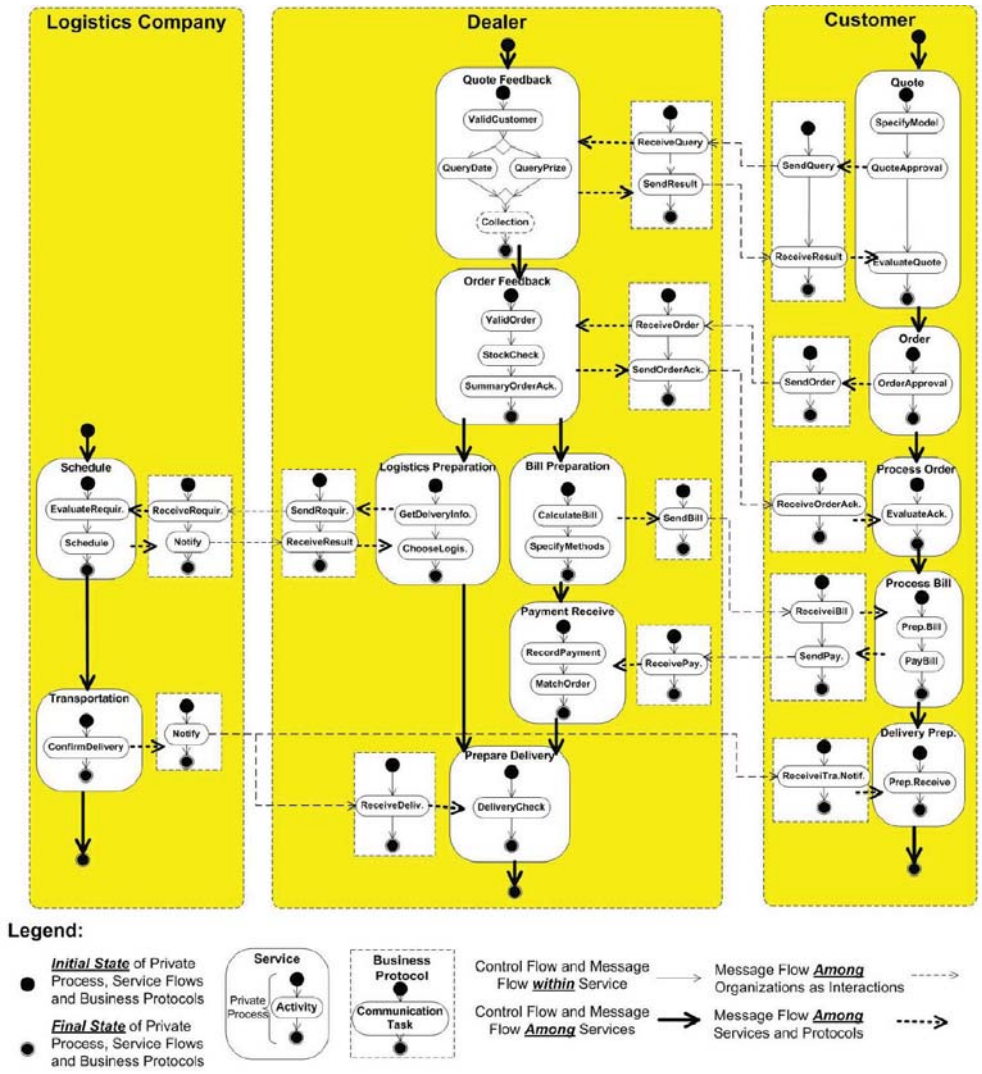

Fig. 2. Ordering process scenario.

3. The work provides the practical way to identify and rationalize the temporal policies based on the Hierarchical Timed Computation Tree Logic;
4. Verification mechanism is provided to clarify the status of temporal inconsistency at runtime.

The rest of paper is organized as follows: Section 2 provides a motivating example to illustrate the temporal inconsistency. Section 3 describes **Ti**med **Cho**reographical **B**usiness **T**ransaction Net which models the business collaboration with temporal semantics. Section 4 presents how temporal policies are specified and rationalized by Hierarchical Timed Computation Tree Logic. Section 5 describes the algorithms to perform policy check on business collaboration and the mechanism to clarify the status of the temporal inconsistency. Section 6 gives the implementation details of the proposed approach. Section 7 reviews some related work. Concluding remarks are presented in Section 8.

2 Motivating Example

Let us take an example in dealer-automotive industry. The ordering process begins with a quote inquiry broadcasting from a **Customer**. After receiving an inquiry, **Dealers** will validate the status of the **Customer**. A quote will be returned if the status of **Customer** is valid. 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 acknowledgment with invoice and payment details. Concurrently, the **Dealer** will require the **Logistics Company** to arrange the transportation. Once the **Dealer** receives both the payment from the **Customer** and 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.

Fig. 2 illustrates the ordering process. Each organization in the collaboration is depicted as a set of *services* (round corner rectangles in Fig. 2), e.g., *Order Feedback* service in **Dealer**, or *Quote* service in **Customer**. A *service* is a business application unit to provide functions to the operation of the whole business collaboration; the `activities` are included in an individual service in Fig. 2 to provide support for the functions of the service. For instance, `Specify Model` and `Quote Approval` are two sequential activities enclosed in *Quote* service to support its function-prepare quote. The `communication tasks` support business interactions among organizations and they are represented by dash lined rectangles. For example, the `Receive Query`

and `Send Result` are two communication tasks to support the interactions between the *Quote Feedback* service of **Dealer** and the *Process Quote* service of **Customer**. For illustrative purpose, in Fig. 2, control flows and message flows within a service are represented as *thin* solid line arrows. Control flows and message flows between services are represented as *thick* solid line arrows. The message flows are represented as *thin* dash line arrows when the messages are between organizations and represented as *thick* dash line arrows when the messages are between the service and its protocol. (See *Legend* in Fig. 2).

From the above example, we can observe the following features that make the management on the temporal consistency difficult in the business collaboration:

- The *Order Feedback* Service in **Dealer** is supported by the internal activities, `Validate Order`, `Stock Check`, and `Summary Order Acknowledgement`; it needs to communicate with *Order* Service in **Customer**. Managing temporal consistency need not only consider the match of temporal policies of web services-*Order* Service in **Customer** and *Order Feedback* Service in **Dealer**. It is also necessary to take the temporal policies of internal activities and associated communication tasks into account.
- All temporal policies of web services syndicated within **Dealer** are defined by **Dealer** and they cannot be dominated by other business participants. Managing temporal consistency in business collaboration requires each involving business participant to guarantee the temporal consistency from single organization's point of view. Furthermore, since web services are peer-to-peer, the temporal policies for internal activity in **Customer** cannot be controlled by the **Dealer**. The Dealer can only manage the temporal consistency within and across organization by identifying the temporal policies of web services, internal activities, and communication tasks in **Dealer**, and those of web services and associated communication tasks in **Customer**.
- Temporal consistency management should be taken at both design time and runtime. For example, the finish time of *Order Feedback* service is 13:00 pm, and the finish time of *Quote Feedback* service is 15:00 pm. At design time, the two services can be syndicated together if the *Quote Feedback* service finishes before 13:00 pm when the *Order Feedback* service is still available to work. If the finish time of *Quote Feedback* service is at least 4 hours later than the start time of *Quote Feedback* service, managing temporal consistency between the *Quote Feedback* service and *Order Feedback* service can only be operated at runtime, depending when the *Quote Feedback* service starts in reality.

The start time, finish time, and execution duration of the service, internal activity and communication task are nondeterministic in an individual business participant. For instance, the earliest start time of *Order Feedback* service in business collaboration should be immediately after sending out the quote result, and the latest start time is seven days later than the finish time of *Quote Feedback* service. Hence, we specify the start time and finish time of each service, internal activity, and communication task with an earliest time and latest time, and a minimum execution duration. The lower bound of the available temporal interval represents the Earliest Start Time ($\mathcal{EST}$) and the upper bound of the available temporal interval is the Latest Finish Time ($\mathcal{LFT}$). The Latest Start Time ($\mathcal{LST}$) is calculated as the start time when the service, internal activity, or communication task reaches the Latest Finish Time after minimum execution duration. If the service, internal activity or communication task is started at the Earliest Start Time, after minimum execution duration, it reaches the Earliest Finish Time ($\mathcal{EFT}$). We do not need to define the maximum execution duration of each web service since it is the time difference between $\mathcal{EST}$ and $\mathcal{LFT}$. The actual execution duration at runtime therefore must longer than the minimum execution duration.

Based on the above-observed features, the temporal inconsistency may occur in service-oriented business collaboration as

- **Across participant process**. If the **Dealer** requires to receive the payment within one week after issuing bill to **Customer**, then Latest Start Time of `Receive Payment` task in **Dealer** should be $(LST_{receivePayment} = completeTime_{issueBill} + 7days)$. The temporal policy in **Customer** emphasizes that the payment will be prepared earliest 14 working days later than receiving the bill, i.e., the Earliest Start Time of `Send Payment` task in **Customer** is $EST_{sendPayment} = completeTime_{issueBill} + 14\ days$ (We ignore the message transfer time here). The Earliest Finish Time of `Send Payment` task $EFT_{sendPayment}$ is later than the Earliest Start Time of `Send Payment` task $EST_{sendPayment}$ (minimum execution duration is required to finish the `Send Payment` task). The Earliest Start Time of `Send Payment` task in **Customer** $EST_{sendPayment}$ is also later than the Latest Start Time of `Receive Payment` task in **Dealer** $LST_{receivePayment}$. Due to $EFT_{sendPayment} \geq EST_{sendPayment} > LST_{receivePayment}$, the `Receive Payment` task is not available when the `Send Payment` task finishes, though as early as possible (If LST of `Receive Payment` is passed, the task cannot be initiated any more). The temporal policies coordination is failed between the services of *Payment Receive* in **Dealer** and *Bill Process* in **Customer**.
- **Within participant process**. The *Prepare Delivery* service in **Dealer** may not be able to work within its available temporal interval (14:00-16:00), since the previous service—*Payment Receive* service—is finished at its latest finish time 17:00. The *Prepare Delivery* service is not available when the *Payment Receive* service is finalized. The temporal policies coordination on *Payment Receive* service and *Prepare Delivery* service does not succeed in the ordering process of **Dealer**.

3 *Ti*CoBTx-Net Model—A Collaboration Model with Temporal Semantics

*Ti*CoBTx-Net model as an infrastructure is developed based on Hierarchical Colored Petri Net to describe the business

collaboration from one organization point of view to catering for the feature of "no central control" in the service environment. The model will employ temporal semantics to identify and operate temporal policies.

3.1 Structure of *Ti*CoBTx-Net

A *Ti*CoBTx-Net is specified for individual participants to understand the behavior of their business partners. It consists of two components: the participant's process and the publicly visible part of its collaborators. A collaborative process in each participant is separated into three layers as shown in Fig. 2, each of which is a subnet of *Ti*CoBTx-Net. The *Execution (Exe) subnet*, *Abstract (Abs) subnet*, and *Communication (Com) subnet* correspond to the generic stratified structure of web service in business collaboration in terms of internal business process, service interface, and business protocol

- At execution level, internal business activities form *Exe-subnet*.
- 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*.
- At communication level, different communication patterns such as request-response, notify, form a *Com-subnet*.

The publicly visible part of other business partners that can be observed by participants are separated into two subnets in *Ti*CoBTx-Net: *External Communication (ExCom) subnet* and *External Abstract (ExAbs) subnet* that represent the business protocol and public interfaces of web services in other business partners, respectively.

3.2 Execution Policy of *Ti*CoBTx-Net

In *Ti*CoBTx-Net, the movements of Application-Oriented Token (AO-Token) correspond to the message flow. AO-Token can move within and across organizational boundary. The movement of AO-Token is controlled by the execution policy of a specific participant. The execution policies related with AO-Token in *Ti*CoBTx-Net are categorized into two types:

- *Behavior Policy* identifies `"where AO-Token should go."` It guides the AO-Token movement from one service to another service, or from one internal activity to another one, based on the business process logic. In *Ti*CoBTx-Net, it is graphically modeled as lines to link net elements, e.g., places or transitions.
- *Temporal Policy* identifies `"when AO-Token should go"` and `"How long AO-Token can go."` For example, an AO-Token can only be fired at a specific transition during a time interval, or the AO-Token can only start to fire after a specific time.

Fig. 3 shows a *Ti*CoBTx-Net of a **Dealer** to answer the *Quote Request* from a **Customer**. The *Ti*CoBTx-Net of the **Dealer** is composed of two components, the three subnets to represent the **Dealer**'s process and two subnets as the public part of the **Customer**. Colored AO-Token representing different messages, e.g., B and D, are moved in *Ti*CoBTx-Net.

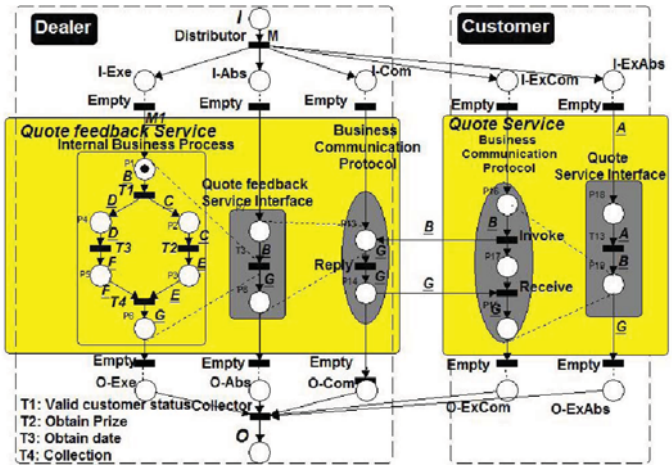

Fig. 3. Part of *Ti*CoBTx-Net of **Dealer**.

3.3 Specification of *Ti*CoBTx-Net

Definition 1. *A TiCoBTx-Net is a tuple* $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{R}, \delta, \lambda, \pi, \mathcal{II}, \mathcal{IO})$, *where*

- $\mathcal{P}$ *is a set of places graphically represented as circles.* $\mathcal{P}^{Exe}$, $\mathcal{P}^{Abs}$, $\mathcal{P}^{Com}$, $\mathcal{P}^{ExC}$, *and* $\mathcal{P}^{ExA}$ *are sets of places at each subnet.*
- $\mathcal{T}$ *is a set of transitions graphically represented as dark bars in Fig. 3, where:* $\mathcal{T}^{Exe}$, $\mathcal{T}^{Abs}$, $\mathcal{T}^{Com}$, $\mathcal{T}^{ExC}$, *and* $\mathcal{T}^{ExA}$ *are sets of transitions at each level.* $\mathcal{T}^{\tau}$ *is a set of empty transitions for transferring AO-Tokens.*
- $\mathcal{F} = (\mathcal{P} \times \mathcal{V} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{V} \times \mathcal{P})$ *is the flow relation between places and transitions, where* $\mathcal{V}$ *is a set of variables* $\mathcal{V} = \{x, y, \ldots\}$ *to represent the colored tokens.*
- $\mathcal{R}$: $\mathcal{P} \cup \mathcal{T} \rightarrow \mathcal{L}$ *is a refinement formula on a transition or place to connect other subnets.* $\mathcal{L} = \{g(x), \{e(x), \mathcal{Y}, r(x)\}\} x \in \mathcal{V}$. $g(x)$ *is a function to evaluate the AO-Token enabled at a specific transition or arrived at an individual place, and decides which subnet/s shall be initiated. Sometimes, multiple subnets* $\mathcal{Y}$ *can be activated simultaneously.* $e(x)$ *and* $r(x)$ *are the guard functions of corresponding subnet* $\mathcal{Y}$ *to evaluate whether or not the subnet is available to initiate and exit.*
- δ: $\mathcal{P} \cup \mathcal{T} \rightarrow \{\mathcal{EST}, \mathcal{LST}, \mathcal{EFT}, \mathcal{LFT}\}$ *is a function on a transition and place to illustrate the start time and finish time, where* $\mathcal{EST}$ *and* $\mathcal{LST}$ *represent the Earliest Start Time and Latest Start Time that each place and transition can be fired, and* $\mathcal{EFT}$ *and* $\mathcal{LFT}$ *represent the Earliest Finish Time and Latest Finish Time that each place and transition can stop firing.*
- λ: $\mathcal{P} \cup \mathcal{T} \rightarrow \widetilde{T}i$, *where* λ *is used to represent the arrive time when the token is deposited at each place and transition.*
- π: $\mathcal{P} \cup \mathcal{T} \rightarrow Q$ *is an interval function representing the minimum execution duration of a place or transition.*
- $\mathcal{II}, \mathcal{IO}$ *are the sets of in and out places of* TiCoBTx-Net *and their subnets, including* $\mathcal{II} = \{\mathcal{I}, i^{Exe}, i^{Abs}, i^{Com}, i^{ExC}, i^{ExA}\}$ *and* $\mathcal{IO} = \{\mathcal{O}, o^{Exe} o^{Abs} o^{Com}, o^{ExC}, o^{ExA}\}$.

$\mathcal{P}$, $\mathcal{T}$, $\mathcal{II}$, and $\mathcal{IO}$ define the basic elements—Place and Transition—in *Ti*CoBTx-Net; while $\mathcal{F}$ and $\mathcal{R}$ are used to design the business process logic within and across subnet, respectively. δ, λ, and π describe the temporal logic in *Ti*CoBTx-Net, e.g., the earliest start time of a web service.

4 Temporal Policies-Enforcement of Temporal Consistency

Temporal policies in *Ti*CoBTx-Net must be coordinated in business collaboration because services are syndicated or interacted with other web services from different participants. A web service could be bound with multiple temporal policies which are depended on the status of business collaboration. A logic named Hierarchical Timed Computation Tree Logic is developed in this section to rationalize which temporal policies should be performed based on the status of *Ti*CoBTx-Net.

4.1 Formal Syntax of the Temporal Policies

A web service, as well as internal activity and communication task, called as an event in this paper, bears specific available temporal interval when it executes. An event is associated with a minimum execution duration π. For an event, there are

- The lower bound of the available time interval is the Earliest Start Time of the event.
- The upper bound of the available time interval is the Latest Finish Time of the event.
- The Earliest Finish Time of the event is the time when the event executes minimum duration π from the Earliest Start Time of the event, i.e., $\mathcal{EFT} = \mathcal{EST} + \pi$.
- The Latest Start Time of the event is the time when the event reaches its Latest Finish Time after executing minimum duration π, i.e., $\mathcal{LST} = \mathcal{LFT} - \pi$.

The above four temporal parameters constrain how each event cooperates with other events in business collaboration. Temporal policy coordination is carried out through matching the four temporal parameters at design time and runtime.

4.1.1 Temporal Policies for Single Subnet

Two types of time are taken into account in temporal policies, message transfer time and task execution time. We use α to indicate the real message transfer time at place P, while β is used to denote the real message execution time at Transition T. Θ represents the real start time of an event and Δ represents the real finish time of an event.

Temporal Policy 1. $\pi(p_j) \leq \alpha$, where π on place p_j denotes the minimum message transfer time.

Temporal Policy 2. $\pi(t_j) \leq \beta$, where π on transition t_j is the minimum message execution time.

Through above Temporal Policies 1 and 2, both real message transfer time α and real message execution time β must be greater than the setup minimum execution duration, respectively.

Based on the basic four temporal parameters of each event, $\mathcal{EST}$, $\mathcal{LST}$, $\mathcal{EFT}$, and $\mathcal{LFT}$, we deduct two types of temporal parameters at design time and runtime denoted by $DT\_^*$ and $\mathrm{R}\_^*$, where $*$ represents the basic four temporal parameters. The basic temporal parameters of each event are set up when the event is created independently. At design time, when each event is syndicated with other events within common business process or interacted with other events, e.g., web services, in participating processes, the earliest and the latest of start time or finish time of each event can be affected by the other cooperated events. For example, at design time, an event is not able to start at its developed earliest start time, and it can only be initiated after all previous time-related events based on business logic have ideally started at their earliest start times and executed at minimum execution durations. When messages are transferred at runtime, the earliest and the latest of start time and finish time of each event may be modified based on the real-time information. For instance, a Runtime Earliest Start Time ($\mathcal{R}\_\mathcal{EST}$) depends on the runtime finish time of another event. Temporal policies may be required to restrict the start time and finish time of each event at both design time and runtime (See Temporal Policies 3 and 4).

Temporal Policy 3. $\mathcal{EST} \leq \mathcal{DT}\_\mathcal{EST} \leq \mathcal{R}\_\mathcal{EST} \leq \Theta \leq \mathcal{R}\_\mathcal{LST} \leq \mathcal{DT}\_\mathcal{LST} \leq \mathcal{LST}$, where the real start time of an event Θ must be between $\mathcal{EST}$ and $\mathcal{LST}$.

Moreover, Θ as the real start time of an event should be restricted by the Design Time Earliest Start Time ($\mathcal{DT}\_\mathcal{EST}$) and Design Time Latest Start Time ($\mathcal{DT}\_\mathcal{LST}$) when the event is involved in business collaboration at design time. The $\mathcal{DT}\_\mathcal{EST}$ of each transition is the maximum of $\mathcal{EFT}$ of all message transfers before the transition, while the $\mathcal{DT}\_\mathcal{LST}$ is as same as $\mathcal{LST}$. The $\mathcal{DT}\_\mathcal{EST}$ and $\mathcal{DT}\_\mathcal{LST}$ of each place bear similar semantics. $\mathcal{DT}\_\mathcal{EST}$ and $\mathcal{DT}\_\mathcal{LST}$ can be calculated as follows: p_j $(j = 1..n)$ are preplaces of a transition and t_j $(j = 1..n)$ are transitions before a place, where

$$\begin{aligned}
&\mathcal{DT}_\mathcal{EST}_{t_i} = \max_{j=1}^{n}(\mathcal{EST}(p_j) + \pi(p_j)), j < i,\\
&\mathcal{DT}_\mathcal{LST}_{t_i} = \mathcal{LST}_{t_i},\\
&\mathcal{DT}_\mathcal{EST}_{p_i} = \max_{j=1}^{n}(\mathcal{EST}(t_j) + \pi(t_j)), j < i,\\
&\mathcal{DT}_\mathcal{LST}_{p_i} = \mathcal{LST}_{p_i},
\end{aligned}$$

Θ as the start time of an event should starts between $\mathcal{R}\_\mathcal{EST}$ and $\mathcal{R}\_\mathcal{LST}$, where $\mathcal{R}\_\mathcal{EST}$ and $\mathcal{R}\_\mathcal{LST}$ represent Runtime Earliest Start Time and Runtime Latest Start Time. $\mathcal{R}\_\mathcal{EST}$ and $\mathcal{R}\_\mathcal{LST}$ are deducted based on the real event execution time, which can only be identified at runtime. $\mathcal{R}\_\mathcal{EST}$ for each transition is the maximum time of $(\lambda(p_j) + \pi(p_j))$, $j = 1..n$. Through this way, we can repeatedly calculate the $\mathcal{R}\_\mathcal{EST}$ of each transition at runtime for each step of message movement among the events in business collaboration. The process is similar to $\mathcal{R}\_\mathcal{LST}$ as that of $\mathcal{LST}$. The $\mathcal{R}\_\mathcal{EST}$ and $\mathcal{R}\_\mathcal{LST}$ of each place bear similar semantics. They can be calculated as follows: p_j $(j = 1..n)$ are all preplaces of a transition and t_j $(j = 1..n)$ are all transitions before a place. $\lambda(p_j)$ and $\lambda(t_j)$ are token arrive times in place p_j and transition t_j, respectively, representing the start time of message transfer and execution, where

$$\begin{aligned}
&\mathcal{R}_\mathcal{EST}_{t_i} = \max_{j=1}^{n}(\lambda(p_j) + \pi(p_j)), j < i\\
&\mathcal{R}_\mathcal{LST}_{t_i} = \mathcal{LST}_{t_i},\\
&\mathcal{R}_\mathcal{EST}_{p_i} = \max_{j=1}^{n}(\lambda(t_j) + \pi(t_j)), j < i\\
&\mathcal{R}_\mathcal{LST}_{p_i} = \mathcal{LST}_{p_i}.
\end{aligned}$$

Temporal Policy 4. $\mathcal{EFT} \leq \mathcal{DT}\_\mathcal{EFT} \leq \mathcal{R}\_\mathcal{EFT} \leq \Delta \leq \mathcal{R}\_\mathcal{LFT} \leq \mathcal{DT}\_\mathcal{LFT} \leq \mathcal{LFT}$, where the real finish time of each event Δ should be between $\mathcal{EFT}$ and $\mathcal{LFT}$. Moreover, Δ needs to comply with the constraints at design time, where it must finish between Design Time Earliest Finish

Time $\mathcal{DT}\_\mathcal{EFT}$ and Design Time Latest Finish Time $\mathcal{DT}\_\mathcal{LFT}$. $\mathcal{DT}\_\mathcal{EFT}$ of the transition is deducted based on its $\mathcal{DT}\_\mathcal{EST}$, where the transition starts at the $\mathcal{DT}\_\mathcal{EST}$ and only executes minimum execution duration. $\mathcal{DT}\_\mathcal{LFT}$ of the transition is the same as the $\mathcal{LFT}$ which is the upper bound of the available temporal interval of the transition. The $\mathcal{DT}\_\mathcal{EFT}$ and $\mathcal{DT}\_\mathcal{LFT}$ of each place bear similar semantics. $\mathcal{DT}\_\mathcal{EFT}$ and $\mathcal{DT}\_\mathcal{LFT}$ for place and transition can be calculated as follows: p_j ($j = 1..n$) are all preplaces of a transition and t_j ($j = 1..n$) are all transitions before a place, where

$$\begin{aligned}
\mathcal{DT}_\mathcal{EFT}_{t_i} &= \mathcal{DT}_\mathcal{EST}_{t_i} + \pi(t_i),\\
\mathcal{DT}_\mathcal{LFT}_{t_i} &= \mathcal{LFT}_{t_i},\\
\mathcal{DT}_\mathcal{EFT}_{p_i} &= \mathcal{DT}_\mathcal{EST}_{p_i} + \pi(p_i),\\
\mathcal{DT}_\mathcal{LFT}_{p_i} &= \mathcal{LFT}_{p_i}.
\end{aligned}$$

Δ is the finish time of an event and it must be between $\mathcal{R}\_\mathcal{EFT}$ and $\mathcal{R}\_\mathcal{LFT}$, where $\mathcal{R}\_\mathcal{EFT}$ and $\mathcal{R}\_\mathcal{LFT}$ represent Runtime Earliest Finish Time and Runtime Latest Finish Time. $\mathcal{R}\_\mathcal{EFT}$ is deducted from $\mathcal{R}\_\mathcal{EST}$, and $\mathcal{R}\_\mathcal{LFT}$ is the same as $\mathcal{LFT}$, where

$$\begin{aligned}
\mathcal{R}_\mathcal{EFT}_{t_i} &= \mathcal{R}_\mathcal{EST}_{t_i} + \pi(t_i),\\
\mathcal{R}_\mathcal{LFT}_{t_i} &= \mathcal{LFT}_{t_i},\\
\mathcal{R}_\mathcal{EFT}_{p_i} &= \mathcal{R}_\mathcal{EST}_{p_i} + \pi(p_i),\\
\mathcal{R}_\mathcal{LFT}_{p_i} &= \mathcal{LFT}_{p_i}.
\end{aligned}$$

Temporal Policy 5. $\mathcal{EFT}^*_{t_j} \leq \mathcal{LST}^*_{t_i}$, $j < i$, where $\mathcal{EFT}^*_{t_j}$ represents Earliest Finish Time of an event, e.g., $\mathcal{EFT}$, $\mathcal{DT}\_\mathcal{EFT}$, and $\mathcal{R}\_\mathcal{EFT}$, while $\mathcal{LST}^*$ is Latest Start Time of next event, e.g., $\mathcal{LST}$, $\mathcal{DT}\_\mathcal{LST}$, and $\mathcal{R}\_\mathcal{LST}$. This policy ensures that the next event must be available to start when the previous event finishes as early as possible. Otherwise, when the previous event finishes, the next event cannot be initiated since its latest start time has been passed.

4.1.2 Temporal Policies for Multiple Subnets

This kind of policies restricts the token movement across the subnets. For example, the $\mathcal{EST}^*$ of a transition t^{Abs}_j in Abs-subnet should not be earlier than the $\mathcal{EST}^*$ of a transition t^{Exe}_i in Exe-subnet where t^{Exe}_i and t^{Abs}_j are linked by refinement function $\mathcal{R}$ in *Ti*CoBTx-Net. The reason is that the earliest start time of an internal activity which supports specific service must be earlier than the earliest start time of the supported service. $\mathcal{LFT}^*$ of a transition t^{Abs}_j in Abs-subnet must not be later than the $\mathcal{LFT}^*$ of a transition t^{Exe}_i in Exe-subnet. It is impossible for the service to be finished later than the latest finish time of its supporting internal activity. Temporal Policies 6 and 7 formally describe above restrictions.

Temporal Policy 6. $\mathcal{EST}^*_{t_i} \leq \mathcal{EST}^*_{t_j}$ if $t_i \in \mathcal{T}^{Exe}$ and $t_j \in \mathcal{T}^{Abs}$ are linked by the refinement function.

Temporal Policy 7. $\mathcal{LFT}^*_{t_i} \geq \mathcal{LFT}^*_{t_j}$ if $t_i \in \mathcal{T}^{Exe}$ and $t_j \in \mathcal{T}^{Abs}$ are linked by the refinement function.

4.2 Hierarchical Timed Computation Tree Logic (*Hi*TCTL)

There may be a big number of temporal policies under different circumstances that are involved in business collaboration. For example, if $\mathcal{DT}\_\mathcal{EST}$ of *Order Feedback* service in **Dealer** is after 15:00 pm, the $\mathcal{R}\_\mathcal{LFT}$ of *Payment Receive* service in **Dealer** will be seven days more than the real finish time of *Order Feedback* service; otherwise, it will be six days more only. In this case, there are two temporal policies in *Payment Receive* service to restrict the real finish time Δ, i.e., $\Delta_{PaymentReceive} \leq \mathcal{R}\_\mathcal{LFT}_{OrderFeedback} + 7\ days$ and $\Delta_{PaymentReceive} \leq \mathcal{R}\_\mathcal{LFT}_{OrderFeedback} + 6\ days$. It is necessary to build up a formal way to effectively and efficiently rationalize the temporal policies based on different status of *Ti*CoBTx-Net.

In *Ti*CoBTx-Net, the state s of each subnet is formulated as $\mathrm{s} = (\mathrm{m}, \mathrm{c})$, where m is a marking and c is a clock valuation, $\mathrm{c} : \mathrm{m} \rightarrow \widetilde{Dur}$ representing the duration elapsed in this marking. Note, marking is used in petri-net-based process model to describe the status of token movement, i.e., marking is changed with the movement of token. In this way, state s becomes a continuous state where time is taken into account. For example, $s_1 = (m_1, 1\ \text{second})..s_n = (m_1, n\ \text{seconds})..s_{n+1} = (m_2, 1\ \text{second})$, where marking m_1 is changed to m_2 after n seconds, i.e., token is moved from m_1 to m_2 after n seconds. Based on this background, we introduce a novel Hierarchical Timed Computation Tree Logic operated on *Ti*CoBTx-Net to rationalize the proposition of temporal policies based on the state of *Ti*CoBTx-Net. For example, if ϕ is an atomic proposition of temporal polices, e.g., $\Delta_{PaymentReceive} \leq \mathcal{R}\_\mathcal{LFT}_{OrderFeedback} + 7\ days$, and s is a state of the specific subnet, ϕ can be used at specific state of the subnet when $\{\phi | \phi : s \rightarrow \{true\}\}$. Here, below we will elaborate the *Hi*TCTL and its semantics.

Hierarchical Timed Computation Tree Logic is developed based on Timed Computation Tree Logic (TCTL) by embedding hierarchical semantics. Timed Computation Tree Logic is an extension of Computation Tree Logic Star (CTL*). Computation Tree Logic star is a temporal logic to specify both liner (LTL) and branching (CTL) property. The syntax of CTL* is given by the following grammar:

$$\begin{aligned}
&\psi_i \equiv false \mid \phi \mid \neg\psi_i \mid \psi_i \wedge \psi_i \mid \forall\psi_j \mid \exists\psi_j,\\
&\psi_j \equiv \psi_i \mid \neg\psi_j \mid \psi_j \wedge \psi_j \mid \mathcal{X}\psi_j \mid \psi_j\mathcal{U}\psi_j \mid \mathcal{F}\psi_j \mid \mathcal{G}\psi_j.
\end{aligned}$$

In the grammar, ϕ stands for an atomic proposition of temporal policy. $\forall$ ("for all path") and $\exists$ ("there exists a path") are path quantifiers, where $\mathcal{U}$ ("until") and $\mathcal{X}$ ("next") are temporal operators. $\mathcal{F}$ ("eventually") and $\mathcal{G}$ ("all future states") are also temporal operators, but can be represented by $\mathcal{U}$ ("until") and $\mathcal{X}$ ("next"), e.g., $\mathcal{F}\psi = \{\text{true}\mathcal{U}\psi\}$ and $\mathcal{G}\psi = \neg\mathcal{F}\neg\psi$. The path quantifiers and temporal operators will be introduced below.

$\mathcal{M}$ is a tuple defined as $\mathcal{M} = (\mathcal{Z}, \mathcal{W})$, where $\mathcal{Z} = (\mathcal{S}, \twoheadrightarrow, s_0)$ is a state transition system in *Ti*CoBTx-Net, $\mathcal{S}$ is a set of states of all subnets, $\twoheadrightarrow$ is the relation between states, and s_0 is a set of initial states of all subnets; $\mathcal{W}$: $S \rightarrow 2^{pl}$ is valuation function which associates each state of subnets with the atomic proposition of temporal policy that the state can satisfy. Let us take Exe-subnet as an example. $s^e \in S^{Exe}$ is a state in Exe-subnet. $\Lambda(s^e)$ is the set of all execution paths in Exe-subnet starting from the s^e. $\rho^e = s^e \twoheadrightarrow s^e_1 \twoheadrightarrow s^e_2 \twoheadrightarrow s^e_3 \ldots\ldots$ is one of the execution path in $\Lambda(s^e)$. The formal semantics of CTL^* is given by the satisfaction relation $\models$ defined as follows:

- $\mathcal{M}, s^e \models \texttt{false}$,
- $\mathcal{M}, s^e \models \phi$ iff $\phi \in \mathcal{W}(s^e)$,
- $\mathcal{M}, c^e \models \neg\psi$ iff $\mathcal{M}, c^e \not\models \psi$ for $c^e \in \{s^e, \rho^e\}$,
- $\mathcal{M}, c^e \models \psi \cap \varphi$ iff $\mathcal{M}, c^e \models \psi$ and $\mathcal{M}, c^e \models \varphi$ for $c^e \in \{s^e, \rho^e\}$,
- $\mathcal{M}, s^e \models \forall\psi$ iff $\forall\rho^e \in \Lambda(s^e), \mathcal{M}, \rho^e \models \psi$,
- $\mathcal{M}, s^e \models \exists\psi$ iff $\exists\rho^e \in \Lambda(s^e), \mathcal{M}, \rho^e \models \psi$,
- $\mathcal{M}, \rho^e \models \psi$ iff $\forall s^e \in \rho^e, \mathcal{M}, s^e \models \psi$
- $\mathcal{M}, \rho^e \models \mathcal{X}\psi$ iff $\mathcal{M}, s_1^e \twoheadrightarrow s_\infty^e \models \psi$, $\{s_1^e, \ldots, s_\infty^e\} \in \rho^e$
- $\mathcal{M}, \rho^e \models \psi\mathcal{U}\varphi$ iff $\exists j \geq 0, \mathcal{M}, s_j^e \twoheadrightarrow s_\infty^e \models \varphi$, and $\forall 0 \leq i \leq j, \mathcal{M}, s_i^e \twoheadrightarrow s_j^e \models \psi$, $\{s_i^e, \ldots, s_j^e, \ldots, s_\infty^e\} \in \rho^e$.

Timed Computation Tree Logic extends CTL* with a time interval I. With this way, we cannot only reason if the proposition is true at a future state, but can also identify which state it is. For example, $F_I\psi$ means that ψ is "eventually" true after time I is past from current state. The grammar of TCTL is given as follows:

$$\psi \equiv false \mid \phi \mid \neg\psi \mid \psi \wedge \psi$$
$$\psi \equiv \forall(\psi\mathcal{U}_I\psi) \mid \exists(\psi\mathcal{U}_I\psi) \mid \forall\mathcal{X}\psi \mid \exists\mathcal{X}\psi \mid \forall\mathcal{F}_I\psi \mid \exists\mathcal{F}_I\psi \mid \forall\mathcal{G}_I\psi \mid \exists\mathcal{G}_I\psi.$$

Let us take Abs-subnet as example. $\widehat{\rho^a} = R^+ \rightarrow S^a$ is an execution path in $\Lambda(s^a)$ (s^a is a state in Abs-subnet), where $\widehat{\rho^a}(r) = s_i^a + \alpha$ is a continuous state in $\widehat{\rho^a}$ (α is a clock time), i.e., $r = \sum_{j=0}^{i-1} s_j^e + \alpha,\ i \geq 0$, and $0 \leq \alpha \leq v_i^a$ (v_i^a represents the time duration needed in state s_i^a). Since temporal operators $\mathcal{F}$ and $\mathcal{G}$ can be represented by $\mathcal{U}$ and $\mathcal{X}$, and the usage of temporal operator $\mathcal{X}$ is the same as its usage in CTL*, we only give the formal semantics of temporal operator $\mathcal{U}$ in TCTL

- $\mathcal{M}, s^a \models \forall(\psi\mathcal{U}_\mathcal{I}\varphi)$ iff $\forall\rho \in \Lambda(s^a), \exists r \in I$, $\mathcal{M}, \widehat{\rho^a}(r) \models \varphi$, and $\forall 0 \leq r' \leq r, \mathcal{M}, \widehat{\rho^a}(r') \models \psi$,
- $\mathcal{M}, s^a \models \exists(\psi\mathcal{U}_\mathcal{I}\varphi)$ iff $\exists\rho \in \Lambda(s^a), \exists r \in I$, $\mathcal{M}, \widehat{\rho^a}(r) \models \varphi$, and $\forall 0 \leq r' \leq r, \mathcal{M}, \widehat{\rho^a}(r') \models \psi$.

The temporal logic mentioned above are only suitable for the deduction within single subnet. *Ti*CoBTx-Net is developed based on the hierarchical petri net with five subnets. It is necessary to extend TCTL for 1) satisfying the requirements of hierarchical semantics and 2) preserving the temporal semantics in *Ti*CoBTx-Net. Hence, we develop a novel temporal logic named as Hierarchial Timed Computation Tree Logic. The syntax of *Hi*TCTL is given as follows:

$$\psi \equiv false \mid \phi \mid \neg\psi \mid \psi \wedge \psi,$$
$$\psi \equiv \forall_B(\psi\mathcal{U}_I\psi) \mid \exists_B(\psi\mathcal{U}_I\psi) \mid \forall_B\mathcal{X}\psi \mid \exists_B\mathcal{X}\psi \mid \forall_B\mathcal{F}_I\psi \mid \exists_B\mathcal{F}_I\psi \mid \forall_B\mathcal{G}_I\psi \mid \exists_B\mathcal{G}_I\psi,$$
$$\psi \equiv \forall_H \oplus_\mathcal{X} \psi \mid \exists_H \oplus_\mathcal{X} \psi \mid \forall_H \oplus_{\mathcal{F}_J} \psi \mid \exists_H \oplus_{\mathcal{F}_J} \psi \mid \forall_H \oplus_{\mathcal{G}_J} \psi \mid \exists_H \oplus_{\mathcal{G}_J} \psi \mid \forall_H(\psi \oplus_{\mathcal{U}_J} \psi) \mid \exists_H(\psi \oplus_{\mathcal{U}_J} \psi) \mid .$$

In *Hi*TCTL, the path quantifiers are classified into two categories: Branch quantifier ($\forall_B$ and $\exists_B$) and Hierarchical quantifier ($\forall_H$ and $\exists_H$). Branch quantifiers are used on future states in single subnets, while Hierarchical quantifiers are used across subnets to model the stratified structure of *Ti*CoBTx-Net. Another type operator named Hierarchical operator is also introduced to deduct proposition of temporal policy across subnets, e.g., $\oplus_X \oplus_{F_J}$, $\oplus_{G_J}$, and $\oplus_{U_J}$. For example, $\oplus_X$ means next upper subnet, while $\oplus_{F_J}$ represents there eventually exists one subnet within J upper levels, J$\in R^+$. Note, since the hierarchical operators for lower levels are the mirror objects of hierarchical operators for upper levels, we omit discussion on them.

Let $s^e \in S^{Exe}$ be a state in Exe-subnet, $s^a \in S^{Abs}$ be a state in Abs-subnet, $s^c \in S^{Com}$ be a state in Com-subnet, and $s^{ec} \in S^{ExCom}$ and $s^{ea} \in S^{ExAbs}$ be states in ExCom-subnet and ExAbs-subnet, respectively. Let $\Omega(s^e)$ be the set of all hierarchical paths from Exe-subnet starting from the s^e. $\pi^e = s^e \twoheadrightarrow s_2^a \twoheadrightarrow s_2^c \twoheadrightarrow s_4^{ec} \ldots\ldots$ is one of the hierarchical path in $\Omega(s^e)$, while $\pi^{e\prime} = s^e \twoheadrightarrow s_1^a \twoheadrightarrow s_1^c \twoheadrightarrow s_3^{ec} \ldots\ldots$ is another example of hierarchical path in $\Omega(s^e)$. Branch Quantifiers and Temporal Operators have been introduced in TCTL and CTL*; here, we only summarize the semantics of Hierarchial Quantifiers and Hierarchial Operators in *Hi*TCTL as follows:

- $\mathcal{M}, s^e \models \forall_H\psi$ iff $\forall\pi^e \in \Omega(s^e), \mathcal{M}, \pi^e \models \psi$,
- $\mathcal{M}, s^e \models \exists_H\psi$ iff $\exists\pi^e \in \Omega(s^e), \mathcal{M}, \pi^e \models \psi$,
- $\mathcal{M}, \pi^e \models \oplus_\mathcal{X}\psi$ iff $\mathcal{M}, s^a \twoheadrightarrow s^* \models \psi$, $\{s^a, \ldots, s^*\} \in \pi^e$
- $\mathcal{M}, \pi^e \models \psi \oplus_{\mathcal{U}_J} \varphi$ iff $\exists 0 \leq k \in J$, $\mathcal{M}, s^{e+k^{th}} \twoheadrightarrow s^* \models \varphi$, and $\forall 0 \leq i \leq k$, $\mathcal{M}, s^{e+i^{th}} \twoheadrightarrow s^{e+k^{th}} \models \psi$, $\{s^{e+i^{th}}, \ldots, s^{e+k^{th}}, \ldots, s^*\} \in \pi^e$.

$\oplus_{\mathcal{F}_J}$ and $\oplus_{\mathcal{G}_J}$ can be represented by $\oplus_\mathcal{X}$ and $\oplus_{\mathcal{U}_J}$. We will not give the detailed semantics of these Hierarchial Operators in this paper. In the formal semantics of Hierarchial Quantifiers and Hierarchial Operators in *Hi*TCTL, we use $s^{e+i^{th}}$ and $s^{e+k^{th}}$ to represent states that encapsulate the state s^e in the ith and kth upper subnets of the original Exe-subnet, and s^* as the final state in specific $\pi(s^e)$. Let us take several examples to see how *Hi*TCTL works at a specific state of individual subnet of *Ti*CoBTx-Net:

Example 1 ($\mathcal{M}, s^e \models \forall_H \oplus_{G_1} (\exists_B(\psi_1\mathcal{U}_{24hours}\psi_2))$)**.** At state of s^e in Exe-subnet, we can reason that for all Abs-subnets encapsulating this Exe-subnet (since G_J where $J = 1$, one upper level of Exe-subnet is considered), there exits a path in Abs-subnets where the proposition ψ_1 is hold 24 hours until ψ_2 to be true. ψ_1 and ψ_2 are the propositions of temporal policies enacted in TiCoBTx-Net, e.g., they may be regarding to define $\mathcal{LFT}$ of two sequential services in Abs-subnet.

Example 2 ($\mathcal{M}, s_1^a \models \forall_H \oplus_{G_0} (\forall_B\mathcal{G}_{50days}\psi_3)$)**.** At state of s_1^a in abs-subnet, there exist states in Abs-subnet (since G_J where J $= 0$, no upper level of Abs-subnet is considered) where we can reason that for all branches at Abs-subnet, there exists a proposition ψ_3 eventually hold within 50 days after the current state s_1^a. s_1^a is equivalent to the state where $\mathcal{DT}\_\mathcal{EST}$ of *Order Feedback* service in **Dealer** is confirmed after 15:00 pm and ψ_3 is the proposition of

temporal policy $\Delta_{PaymentReceive} \leq \mathcal{R}\_\mathcal{LFT}_{OrderFeedback} + 7\ days$. After 50 days, this proposition of temporal policy will be treated as false.

5 Detection Mechanism-Verification of Temporal Inconsistency

Temporal inconsistency occurs when the temporal policy coordination is failed in *Ti*CoBTx-Net. In this section, we propose algorithms to coordinate the temporal policy, named as Temporal Policy Match Check. If the match check is failed, a property named *time-embedded dead marking freeness* in *Ti*CoBTx-Net is introduced to verify the status of the temporal inconsistency.

5.1 Temporal Policies Coordination

Each service, as well as internal activity and communication task is associated with multiple temporal policies when they are created. When they are cooperated with each other in business collaboration, the coordination on temporal policies becomes necessary to ensure the successful business collaboration execution. It is not usual for these events to have precious temporal policy match. All of the involving services including the associated internal activity and communication task can be freely started and finished within their well-defined temporal intervals. We identify three statuses for the temporal policy coordination: 1) **Full Policy Match**. The service including the encapsulated internal activities and communication tasks can start and finish at any time within the available time interval; 2) **Fuzzy Policy Match**. The services, internal activities, or communication tasks can only execute within a more strict time interval to ensure the successful business collaboration execution; 3) **No Policy Match**. There exists at least one event which does not have time interval to execute when it cooperates with other events at design time or runtime. The business collaboration should avoid "No Policy Match." The policy match check is therefore developed as follows to measure the temporal policy coordination:

1. **Static Policy Check**. Static Policy Check is used at design time to ensure if the available time interval exists for each event in business collaboration. For example, the earliest start time of *Order Feedback* service in **Dealer** is designed at 8:00 am. When the service interacts with *Order* service in **Customer A**, the earliest start time of the *Order Feedback* service at design time ($\mathcal{DT}\_\mathcal{EST}_{OrderFeedback}$) has to be set up as 10:00 am since the start time of the available temporal interval of *Order service* in **Customer A** is 10:00 am. In this case, Static Policy Check ensures if the available time interval of the *Order Feedback* service in **Dealer** still exists after the earliest start time is modified at design time.
2. **Dynamic Policy Check**. The start time and finish time of each event in business collaboration are dynamic, e.g., the earliest start time of one event depends on the real finish time of another event. Dynamic Policy Check is needed to ensure that there does not exist "No Policy Match" for each event at runtime, i.e., available time interval of each event exists at runtime.

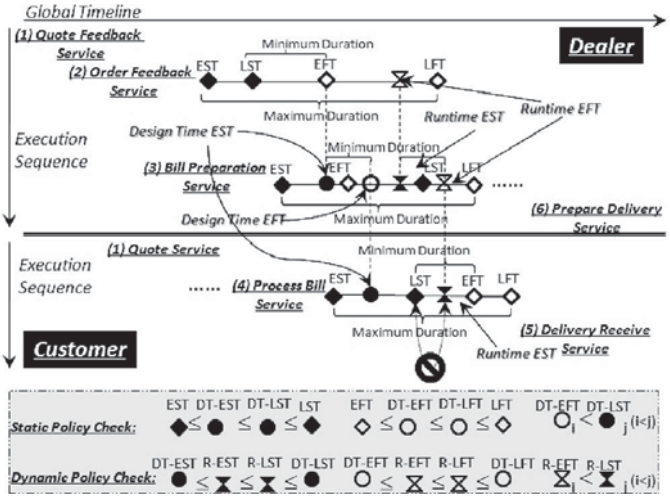

Fig. 4. Timeline of ordering process scenario.

Fig. 4 shows the temporal constraints of the ordering process scenario at the service level. Each service must work within its available temporal interval. Black diamond represents $\mathcal{EST}$ and $\mathcal{LST}$, and white diamond represents $\mathcal{EFT}$ and $\mathcal{LFT}$. It can be observed that the business collaboration is at the status of "Fuzzy Policy Match" and the Static Policy Check is successful. Black circle represents $\mathcal{DT}\_\mathcal{EST}$ and $\mathcal{DT}\_\mathcal{LST}$, while white circle is depicted as $\mathcal{DT}\_\mathcal{EFT}$ and $\mathcal{DT}\_\mathcal{LFT}$. Due to the $\mathcal{R}\_\mathcal{EFT}_{OrderFeedback}$ in **Dealer** is dynamic, e.g., it depends on the time of receiving order, at that time shown in Fig. 4, the *Process Bill* service in **Customer** cannot be able to start its function at runtime when *Prepare Bill* service in **Dealer** finishes, i.e., the latest start time of *Process Bill* service has passed. The Dynamic Policy Check is failed. Two black triangles represent $\mathcal{R}\_\mathcal{EST}$ and $\mathcal{R}\_\mathcal{LST}$, while two white triangles represent $\mathcal{R}\_\mathcal{EFT}$ and $\mathcal{R}\_\mathcal{LFT}$.

We design the Algorithm 1 for static policy check to examine the relationships of four basic temporal parameters with DT_*, i.e., $\mathcal{DT}\_\mathcal{EST}$, $\mathcal{DT}\_\mathcal{LST}$, $\mathcal{DT}\_\mathcal{EFT}$, and $\mathcal{DT}\_\mathcal{LFT}$. Algorithm 1 will repeatedly check the temporal policies 3 to 7 at each place and transition in *Ti*CoBTx-Net, to ensure the existence of the available time interval. Checks for temporal policies 1 and 2 are not included in Algorithm 1 since the execution duration of each place and transition are not related to the policy coordination at design time. Similarly, we design algorithm for dynamic policy check to examine the relationship of R_* with $DT\_^*$ (R_* represents $\mathcal{R}\_\mathcal{EST}$, $\mathcal{R}\_\mathcal{LST}$, $\mathcal{R}\_\mathcal{EFT}$, and $\mathcal{R}\_\mathcal{LFT}$). Due to space limit, we only provide the algorithm for static policy check as shown in Algorithm 1.

Algorithm 1. Algorithm for Static Policy Check
Input: $\mathcal{P}$, $\mathcal{T}$, $\mathcal{F}$, $\mathcal{EST}^*$, $\mathcal{LST}^*$, $\mathcal{EFT}^*$, and $\mathcal{LFT}^*$
Output: checkResult → {True, False}
Steps:
1: checkResult:= True; ob:=$\mathcal{I}$; obN:=$\emptyset$; obH:=$\emptyset$;
2: /*ob $\in \mathcal{P} \cup \mathcal{T}$; obN, obH $\subseteq \mathcal{P} \cup \mathcal{T}^*$/
3: **while** ob $\neq \mathcal{O}$ **do**
4: /***Temporal Policy 3***/
5: **if** Not ($\mathcal{EST}(ob) \leq \mathcal{DT}\_\mathcal{EST}(ob) \leq \mathcal{DT}\_\mathcal{LST}(ob) \leq \mathcal{LST}(ob)$) **then**
6: checkResult:= False
7: **end if**

```
/*Temporal Policy 4*/
if Not (EFT(ob) ≤ DT_EFT(ob) ≤ DT_LFT(ob) ≤ LFT(ob)) then
    checkResult:= False
end if
/*Temporal Policy 5*/
obN:=next(ob); i:=1;
for obN_i ∈ obN do
    if (EFT(ob) > LST(obN_i)) or (DT_EFT(ob) > DT_LST(ob)) then
        checkResult:= False;
    end if
    i := i + 1;
end for
/*Temporal Policy 6*/
obH:= Hierarchical (ob); j:=1;
for obH_j∈ obH do
    if (EST(ob) < EST(obH_j)) or (DT_EST(ob) < DT_EST(obH_j)) then
        checkResult:= False;
    end if
    j:=j+1;
end for
/*Temporal Policy 7*/
obH:=Hierarchical(ob); k:=1;
for obH_j∈ obH do
    if (LFT(ob) > LFT(obH_j)) or (DT_LFT(ob) > DT_LFT(obH_j)) then
        checkResult:= False;
    end if
    k := k + 1;
end for
end while
```

5.2 Verification Mechanism

In this section, we introduce a property named *time-embedded dead marking freeness* in *Ti*CoBTx-Net to clarify the status of the temporal inconsistency, i.e., to confirm in which step of *Ti*CoBTx-Net the temporal inconsistency occurs. The *labeled transitive matrix* L^*_{BP} [18] denotes the relationship between $\bullet t$ and $t\bullet$ based on transition t. (Note, $\bullet t = \{y \in \mathcal{P} | (y,x) \in \mathcal{F} \cap x \in \mathcal{T}\}$, $t\bullet = \{y \in \mathcal{P} | (x,y) \in \mathcal{F} \cap x \in \mathcal{T}\}$. $\bullet t$ and $t\bullet$ indicate the preplaces and postplaces of a transition, respectively.) We extend the transitive matrix by associating it with the result of temporal policy coordination in *Ti*CoBTx-Net, called *Labeled time-embedded transitive matrix*, and use it to detect the temporal inconsistency.

Definition 2. *A time-embedded transitive matrix*

$$L^t_{BP} = (A^-)^T Diag\left(\sum_{h=1}^{n} r_{t_n}, \sum_{h=1}^{n} r_{t_n}, \ldots, \sum_{h=1}^{n} r_{t_n}\right) A^+,$$

where $A^- = [a^-_{ij}]$ and $A^+ = [a^+_{ij}]$ are $n \times m$ matrix (n transitions, m places). T means transpose matrix. $x \in \mathcal{P}$, $y \in \mathcal{T}$.

$$a^-_{ij} = \begin{cases} 1 & (x,y)\ In\ \mathcal{F} \\ 0 & (x,y)\ Not\ In\ \mathcal{F} \end{cases}$$

$$a^+_{ij} = \begin{cases} 1 & (y,x)\ In\ \mathcal{F} \\ 0 & (y,x)\ Not\ In\ \mathcal{F} \end{cases}$$

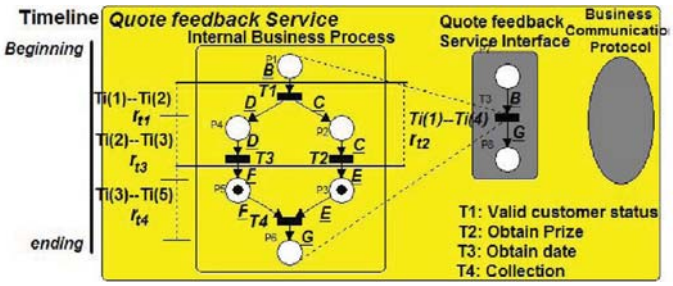

Fig. 5. Example for temporal consistency verification.

$r_{t_h}(h = 1, 2, \ldots, n)$ is a evaluation variable of temporal requirement as

$$|r_{t_h}| = \begin{cases} 1 & t_h \text{ is linked by specific places and temporal policy coordination on } t_h \text{ is succeeded.} \\ 0 & t_h \text{ is } \textbf{not} \text{ linked by specific places or temporal policy coordination on } t_h \text{ is failed.} \end{cases}$$

We also use L^{*t}_{BP}, a *labeled time-embedded transitive matrix* $(m \times m)$ to extend the original time-embedded transitive matrix in which r_{t_h} in L^t_{BP} is replaced by r_{t_h}/d_{t_h} in L^{*t}_{BP}, if r_{t_h} appears d times in the same column of L^t_{BP}. A *Ti*CoBTx-Net is time-embedded dead marking free **iff**

$$\exists t_h \in \mathcal{T},\ \mathcal{P}_j^{r_{t_h}}(time) = \sum_{i=1}^{m} \frac{r_{t_h}}{d_{t_h}} c_i \geq 1,$$

where $M^{*t}_k(time) = M_{k-1} \cdot L^{*t}_{BP}$, $M^{*t}_k(time_j) = \mathcal{P}_j(time) = \sum_{h=1}^{n} p_j^{r_{t_h}}(time)$, c_i represents the ith preplace of t_h.

Fig. 5 describes the internal business process of *Quote Feedback* service in **Dealer** to answer the *Quote* service in **Customer**. Each transition in this internal business process represents an individual activity and is associated with specific temporal policies. r_{t_h} is correlated with corresponding transition t_h as the evaluation variable of the temporal requirement. The $Ti(j)$ is a time spot to partition the time according to when the transition can be activated and terminated. We assume that the state of Exe-subnets at *Ti*CoBTx-Net of **Dealer** is $M_3 = [0,0,1,0,1,0]$ as t_4 is enabled by the tokens in p_3 and p_5 ($c_3 = 1$, $c_5 = 1$),

$$L^{*t}_{BP} = \begin{bmatrix} 0 & r_{t_1} & 0 & r_{t_1} & 0 & 0 \\ 0 & 0 & r_{t_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{r_{t_4}}{2} \\ 0 & 0 & 0 & 0 & r_{t_3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{r_{t_4}}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Hence, $M^{*t}_4(time_j) = M_3 \cdot (L^{*t}_{BP})_{ij}$. Since $M^{*t}_4(time_{1-5}) = 0$, we set $M^{*t}_4(time_6) = \mathcal{P}_6(time) = \sum_{h=1}^{4} \mathcal{P}_6^{r_{t_h}} = \frac{r_{t_4}}{2} \cdot (c_3 + c_5)$,

$$\mathcal{P}_6^{r_{t_4}}(time) = \frac{r_{t_4}}{2} \cdot (c_3 + c_5) \begin{cases} = 0\ if\ time < Ti(3)\ or \\ \quad time > Ti(5), then\ r_{t_4} = 0 \\ = 1\ if\ Ti(3) \leq time \leq Ti(5), \\ \quad then\ r_{t_4} = 1. \end{cases}$$

If **Dealer** can execute the activity "collection" (t_4) within time interval $Ti(3)$ and $Ti(5)$, then the temporal policies are

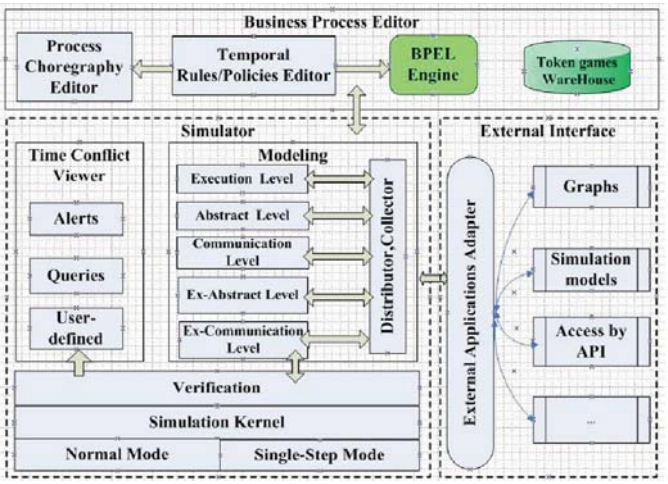

Fig. 6. System architecture of ***Ti***CoBTxS.

satisfied as $r_{t_4} = 1$ and eventually $\mathcal{P}_6^{r_{t_4}}(time) \geq 1$. The *Ti*CoBTx-Net of **Dealer** is time-embedded dead marking free at that moment. Otherwise, $r_{t_4} = 0$, the *Ti*CoBTx-Net is at the status of time-embedded dead marking and t_4 cannot be fired (See Fig. 5).

6 Implementation

The implemented *Ti*CoBTX-Net system is developed with JAVA/SWT. It consists of three components (See Fig. 6): 1) **Business Process Editor**: gets the input of a *Ti*CoBTx-Net through *Process Choreography Editor* or *BPEL engine*. The *BPEL engine* takes BPEL codes and converts them into the input of *Ti*CoBTx-Net. In our system, we adopt the existing technology to transfer the BPEL and Petri Net based model, e.g., [14]. Temporal policies are captured by *Temporal Rules/Policies Editor*. In the future, we will extend BPEL codes with temporal semantics and enrich the function of current BPEL engine in the system to input the temporal policies with temporal-aware BPEL codes. In order to simplify and reuse the configuration, *Token Games Warehouse* is used to store the execution policy for token movement. 2) **Simulator**: performs the verification. The *Modeling* works on tokens' movements between different subnets of various organizations. It can also clarify the status of temporal inconsistency. The *verification* enables users to choose to execute the simulation in either of two modes: `Normal Mode`, which means that tokens move continuously, or `Single-Step Mode`, which allows the user to capture every single step of token movements. The *Time Conflict Viewer* displays alerts for users. 3) **External Interface**: allows the system to be integrated with other applications. Interfaces for data importing and exporting are provided.

In this paper, we only use the Petri-net-based process model to describe the business collaboration and manage temporal consistency. In the future, the complexity of Petri-net-based process model will be touched upon with our research work stepping forward to the next phase.

Fig. 7 shows a system running status for the example shown in Fig. 3. Initially, a *Ti*CoBTx-Net of **Dealer** is constructed. Second, temporal policies are defined for each place and transition (See Fig. 8), while DT-* are calculated based on the design of temporal policies. Static policy check are enacted in *Ti*CoBTx-Net System based on DT-*. Dynamic policy check is performed at runtime. The temporal inconsistency is alerted after the verification.

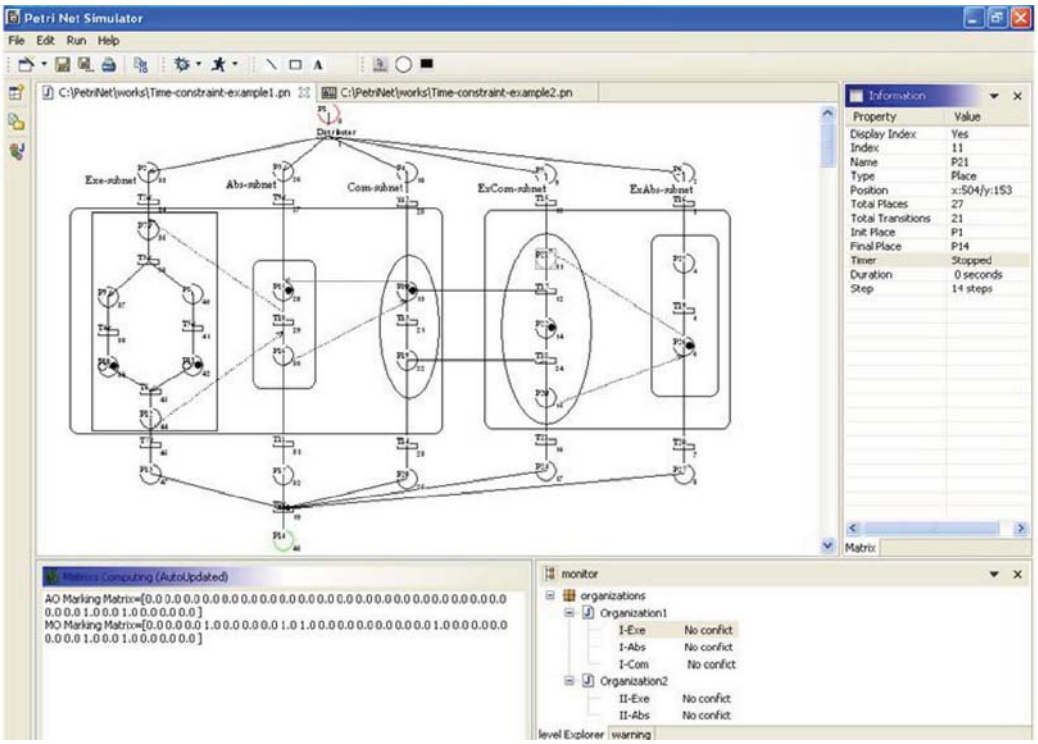

Fig. 7. Interface of *Ti*CoBTS.

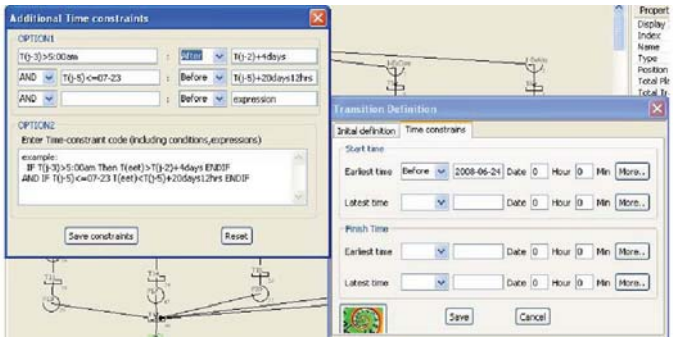

Fig. 8. Designing temporal policies.

7 Related Work

In this section, we will review the related work on managing temporal consistency in service-oriented business collaboration.

7.1 Business Collaboration Modeling

This section reviews some representative works of business process modeling based on Petri Net.

The works in [10], [11], [12], [13] develop a series of petri-net-based process models to perform web service behavior conformance checking. Different from our work, their petri-net process models focus on service level. We believe that their approaches cannot handle the complexity of business collaboration management without explicitly considering the internal activities and communication tasks. Our proposed model has the capability to consider the services, activities, and communication tasks under the same umbrella. Furthermore, their process models are designed for checking the conformance of web service behaviors instead of temporal constraints of involved events in business collaboration. The reported work in this paper focuses on the issue of temporal inconsistency in business collaboration. The proposed model is based on the hierarchical colored petri net with the adoption of the *Hi*TCTL for temporal policies.

Yang et al. [14] propose a method to transfer WS-BPEL [27] to Colored Petri-Nets (CPN) for verifying web service composition. A model based on Hierarchical CPN is introduced in [15] for detecting the reliability issues of web service workflow. These models have the limitation of being constructed from a centralized global view which requires the detailed information of all participants. This assumption is impracticable in the peer-based loosely coupled business collaboration environment.

7.2 Managing Temporal Consistency

Managing temporal consistency has attracted a lot of research recently. Here are the representative works.

Authors in [22], [23], [24] mainly focus on analyzing temporal compatibility from choreography perspective, i.e., check temporal policies among services of different organizations. Several temporal conflicts are identified in asynchronous web service interactions. However, their works can only work at service level, without taking internal activities and business protocols into account. Runtime temporal policy coordination is totally ignored.

Other researches spend much effort on analyzing temporal reliability based on flow-types web service orchestration. In [25], the authors use temporal logic based on BPMN-Q to ensure the compliance of rules and polices in web service orchestration. Authors in [19] present an approach to transform web service orchestration into a time-aware orchestration where temporal assessment and intervention logic are performed to ensure the reliability of service composition. In [20], the authors propose a verification approach on temporal reliability based on WS-BPEL. However, in these research works, coordination on temporal policies to ensure the temporal consistency is missing.

In [21], the authors present a framework for computing activity deadlines in workflow systems. The overall process deadline and external constraints are considered. This work targets workflow systems and it does not take cross-organizational service interactions into account.

There have many research efforts on managing temporal consistency in service-oriented business collaboration. However, it is still lacking of a comprehensive solution to coordinate temporal policies for business collaboration at both design time and runtime.

In this paper, we propose a novel process model, *Ti*CoBTx-Net, to manage temporal consistency in service-oriented business collaboration. The merits of *Ti*CoBTx-Net lie in

- The model has the capability to cover the internal business activities, web service interfaces, communication tasks, and the projection of public part of collaborative web services in other business partners under the same umbrella.
- The proposed approach provides a generic solution to capture the temporal requirements in the business collaboration by identifying temporal policies to be operated in *Ti*CoBTx-Net.
- Algorithms are developed for policy check based on *Ti*CoBTx-Net to enforce the temporal consistency at design time and runtime. A verification mechanism is provided to clarify the status of the temporal inconsistency.

8 Conclusion

A novel model has been proposed for managing temporal consistency in service-oriented business collaboration. The formal syntax and specification for temporal policies and *Ti*CoBTx-Net have been presented, as well as a new type of temporal logic, *Hi*TCTL, to reason temporal policies in *Ti*CoBTx-Net. The algorithms have been provided for checking temporal policies at both design time and runtime. The mechanism based on *Ti*CoBTx-Net has been proposed for clarifying the status of temporal inconsistency. The implementation details of the proposed mechanism have also been provided. The proposed approach provides a formal solution to address the challenging temporal inconsistency issue in the service-oriented business collaboration.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Service: Concept, Architectures and Applications*. Springer, 2004.

[2] E. Newcomer and G. Lomow, *Understanding SOA with Web Service.* Pearson Education, 2005.
[3] M.P. Papazoglou, *Web Services: Principles and Techonology.* Prentice Hall, July 2007.
[4] J. Tsai, S. Yang, Y. Chang, and E. Juan, "Verifying Timing Properties for Distributed Real-Time Systems Using Timing Constraint Petri Nets," *Proc. Int'l Computer Software and Applications Conf. (COMPSAC),* 1996.
[5] J. Toussaint, F. Simonot-Lion, and J. Thomesse, "Time Constraints Verification Methods Based on Time Petri Nets," *Proc. IEEE Workshop Future Trends of Distributed Computing Systems (FTDCS),* 1997.
[6] F.L. Tiplea and G.I. Macovei, "Timed Workflow Nets," *Proc. Seventh Int'l Symp. Symbolic and Numeric Algorithms for Scientific Computing (SYNASC),* 2005.
[7] S. Ling and H. Schmidt, "Time Petri Nets for Workflow Modelling and Analysis," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics (ICSMC),* 2000.
[8] T. Murata, "Petri Nets: Properties, Analysis and Application," *Proc. IEEE,* vol. 77, no. 4, pp. 541-580, Apr. 1989.
[9] C. Girault and R. Valk, *Petri Nets for System Engeneering: A Guide to Modeling, Verification and Application.* Springer, 2003.
[10] C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede, "WofBPEL: A Tool for Automated Analysis of BPEL Processes," *Proc. Int'l Conf. Service-Oriented Computing (ICSOC),* pp. 484-489, 2005.
[11] W.M.P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H.M.W. Verbeek, "Choreography Conformance Checking: An Approach Based on BPEL and Petri Nets," *The Role of Business Processes in Service Oriented Architectures,* Inderscience Enterprises, 2006.
[12] C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede, "Formal Semantics and Analysis of Control Flow in WS-BPEL," *Science of Computer Programming,* vol. 67, nos. 2/3, pp. 162-198, 2007.
[13] W.M.P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and E. Verbeek, "Conformance Checking of Service Behavior," *ACM Trans. Internet Technology,* vol. 8, no. 3, pp. 29-59, 2008.
[14] Y. Yang, Q. Tan, J. Yu, and F. Liu, "Transformation BPEL to CP-Nets for Verifying Web Service Composition," *Proc. Int'l Conf. Next Generation Web Services Practices,* 2005.
[15] Y. Yang, Q. Tan, Y. Xiao, J. Yu, and F. Liu, "Exploiting Hirachichical CP-Nets to Increase the Reliability of Web Services Workflow," *Proc. Symp. Application and the Internet,* 2006.
[16] P. Huber, K. Jensen, and R.M. Shapiro, "Hierarchies in Colored Petri Nets," *Proc. Int'l Conf. Applications and Theory of Petri Nets: Advances in Petri Nets,* pp. 313-341, 1990.
[17] D.G. Stork and R.J. van Glabbeek, "Token-Controlled Place Refinement in Hierarchical Petri Nets with Application Active Document Workflow," *Proc. 23rd Int'l Conf. Application and Theory in Petri Nets,* pp. 394-413, 2002.
[18] Y. Song and J. Lee, "Deadlock Analysis of Petri Nets Using the Transitive Matrix," *Proc. SICE Ann. Conf.,* pp. 689-694, 2002.
[19] H. Pichler, M. Wenger, and J. Eder, "Composing Time-Aware Web Service Orchestrations," *Proc. 21st Int'l Conf. Advanced Information Systems Eng. (CAiSE '09),* pp. 349-363, 2009.
[20] A. Ismail, J. Yan, and J. Shen, "Verification of Composite Services with Temporal Consistency Checking and Temporal Satisfaction Estimation," *Proc. 10th Int'l Conf. Web Information Systems Eng. (WISE '09),* pp. 343-350, Oct. 2009.
[21] J. Eder, E. Panagos, and M. Rabinovich, "Time Constraints in Workflow Systems," *Proc. 11th Int'l Conf. Advanced Information Systems Eng. (CAiSE '99),* pp. 165-280, June 1999.
[22] N. Guermouche and C. Godart, "Timed Model Checking Based Approach for Web Services Analysis," *Proc. IEEE Int'l Conf. Web Services (ICWS '09),* pp. 213-221, 2009.
[23] N. Guermouche and C. Godart, "Timed Properties-Aware Asynchronous Web Service Composition," *Proc. Int'l Conf. Cooperative Information Systems (CoopIS '08),* pp. 44-61, 2008.
[24] N. Guermouche and C. Godart, "Asynchronous Timed Web Service-Aware Choreography Analysis," *Proc. 21st Int'l Conf. Advanced Information Systems Eng. (CAiSE '09),* pp. 364-378, 2009.
[25] A. Awad, G. Decker, and M. Weske, "Efficient Compliance Checking Using BPMN-Q and Temporal Logic," *Proc. Sixth Int'l Conf. Business Process Management (BPM '08),* pp. 326-341, 2008.
[26] V. Kordic, *Petri Net: Theory and Applications.* I-Tech Education and Publishing, 2008.
[27] D. Jordan et al., *Business Process Execution Language for Web Service (BPEL4WS) 2.0,* http://docs.oasis-open.org/wsbpel/2.0, 2006.
[28] B. Benatallah, P. Chrzastowski-Wachtel, R. Hamadi, M. O'Dell, and A. Susanto, "HiWorD: A Petri Net-Based Hierarchical Workflow Designer," *Proc. Third Int'l Conf. Application of Concurrency to System Design (ACSD),* pp. 235-236, 2003.

Haiyang Sun received the BEcon degree in economic information management from Capital University of Economics and Business, Beijing, China, in 2002, the MIT degree in database management and administration from the University of Sydney, Australia, in 2005, and the MSc degree in computing from Macquarie University, Sydney, Australia, in 2008. He is currently working toward the PhD degree in computer science at the Department of Computing, Macquarie University, Sydney, Australia. His research interests include theory, mechanism, and infrastructure to support reliable and consistent business collaboration, service-oriented business collaboration modeling and integration, and petri nets applications.

Jian Yang received the PhD degree from the Australia National University in 1995. She is currently an associate professor of computing at Macquarie University, Sydney, Australia. She has been publishing in the area of database integration, data warehousing, query optimization, digital library, web services, and business process management. Her current work focuses on modeling and analysis of the QoS and trust aspects of services, and on business process management. She served/is serving on program committees of many conferences (e.g., ICDE, CAiSE, WISE, WWW, ICSOC, BPM, ICWS, SCC, etc.) and was the program committee cochair of ICSOC 2010. She is a member of the IEEE.

Weiliang Zhao received the PhD degree in computing from the University of Western Sydney in 2009. He is a research fellow at the Department of Computing, Macquarie University. Before joining Macquarie University, he worked at Australia and New Zealand Banking Group Limited as a programmer and at the Chinese Academy of Sciences as a researcher. His current research areas include trust management, security, and change management in service-oriented computing; quality of service analysis; security protocols and cryptographic techniques in e-commerce.

Jianwen Su received the BS and MS degrees from Fudan University and the PhD degree from the University of Southern California. He held visiting positions at INRIA and Bell Labs. He is currently a professor of computer science at the University of California, Santa Barbara. His publications concern database query languages, data models, scientific databases, workflow, spatial databases, formal verification, web services, and business process management. His current work focuses on modeling and analysis of web services and on business process management. He served/is serving on program committees of many conferences (e.g., PODS, VLDB, ICDE, ICDT, SSTD, WISE, WWW, ICSOC, BPM, etc.), was the general chair of the ACM SIGMOD 2001 conference, and was the program committee chair of ACM PODS 2009. He is a senior member of the IEEE and the IEEE Computer Society.