

國立交通大學

資訊科學與工程研究所

碩士論文

BPMN* 與工作流程管理系統之整合

Integrating BPMN* with a WfMS

研究生：羅皓仁

指導教授：王豐堅 教授

中華民國九十五年九月

BPMN* 與 工 作 流 程 管 理 系 統 之 整 合
Integrating BPMN* with a WfMS

研 究 生：羅皓仁

Student：How-Jen Lo

指 導 教 授：王豐堅

Advisor：Feng-Jian Wang

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文



A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年九月

國立交通大學

研究所碩士班

論文口試委員會審定書

本校 資訊科學與工程 研究所 羅皓仁 君

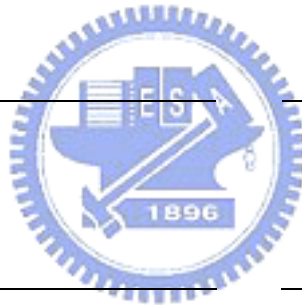
所提論文：

BPMN* 與工作流程管理系統之整合

Integrating BPMN* with a WFMS

合於碩士資格水準、業經本委員會評審認可。

口試委員：



指導教授：_____

所 長：_____

中華民國九十五年 月 日

BPMN* 與工作流程管理系統之整合

研究生：羅皓仁

指導教授：王豐堅 博士

國立交通大學

資訊科學與工程研究所

新竹市大學路 1001 號

摘要

在現今的商業流程環境中，要在不同型態的流程管理系統之間交換流程定義是件困難的事，再加上大多數的建構語言都過於偏向設計者的專業技術領域，使得其所牽涉到的各相關領域人士在溝通與建立共識上亦產生困難。為了解決此類問題，標準化的程序語言 BPMN 被發展出用以描述商業流程規格；然而 BPMN 被設計為只提供流程相關的建構，其他如組織架構、資料模型等方面的規格定義卻付之闕如。這些方面的規格描述，與 BPMN 定義的整合，都有其確實的必要性。本篇論文提出了一個結合修訂之 BPMN 的概念模型，與其應用的商業流程開發方法，接著，把所開發的商業流程定義轉換為 Agentflow 可執行的規格。

Keywords: 工作流程、商業流程、BPMN、商業流程模型建構

Integrating BPMN* with a WfMS

Student: How-Jen Lo

Advisor: Dr. Feng-Jian Wang

Institute of Computer Science and Engineering

National Chiao Tung University

1001 Ta Hsueh Road, Hsinchu, Taiwan, ROC

Abstract

In current business process environments, it is difficult to transfer business process specifications between WfMSs based on different paradigms. The participants of a business process consist of people from very different domains, while most of the modeling languages are designed for technical developers and thus hinder the communication and mutual understanding among developers and stakeholders. BPMN is a standardized process language designed to conquer these problems. However, BPMN supports only the modeling concepts that are applicable to business processes. Several practical issues for business purposes are out of scope of BPMN, such as organization structures, data models. It is necessary to do the corresponding specifications and integrate them with those in BPMN. This thesis proposes a conceptual business process model with a modified notation of BPMN, and a methodology to develop a business process. The specification is then translated into an Agentflow execution instance.

Keywords: workflow, business process, BPMN, business process modeling

誌謝

本篇論文的完成，首先要感謝我的指導教授王豐堅博士兩年來不斷的指導與鼓勵，讓我在軟體工程及工作流程的技術上，得到很多豐富的知識與實務經驗。另外，也非常感謝我的畢業口試評審委員吳毅成博士、郭譽申博士以及曾淑峰博士，提供許多寶貴的意見，補足我論文裡不足的部分。

其次，我要感謝實驗室的學長姊們，有博士班慶鴻學長參與討論，以及其他學長姐在研究生涯上的指導與照顧，讓我學得許多做研究的技巧，得以順利的撰寫論文。

最後，我要感謝我的家人，由於有你們的支持，讓我能心無旁騖地讀書、作研究然後到畢業。由衷地感謝你們大家一路下來陪著我走過這段研究生歲月。



Table of Contents

摘要.....	I
Abstract.....	II
誌謝.....	III
Table of Contents	IV
List of Figures	V
Chapter 1. Introduction.....	1
Chapter 2. Background.....	4
2.1. Introduction of BPMN	4
2.2. Agentflow and CA-PLAN	8
2.2.1. Overview of Agentflow	8
2.2.2. Introduction of CA-PLAN.....	10
2.3. Coloured Petri Net	12
Chapter 3. Business Process Model.....	15
3.1. BPMN-PLAN	15
3.2. BPMN*	19
Chapter 4. Developments with BPMN-PLAN	25
4.1. Development methodology	25
4.2. Transferring algorithm	28
4.2.1. Phase 1 : Recompose collaboration process	29
4.2.2. Phase 2 : Transform process	31
4.2.3. One-to-one mapping	41
Chapter 5. Case study	43
5.1. Requirement specifications	43
5.2. System design	45
Chapter 6. Conclusion and future works	57
Reference	58

List of Figures

Figure 2_1 BPMN core objects architecture	7
Figure 2_2 Agentflow system architecture	9
Figure 2_3 Conceptual framework of PLAN	10
Figure 2_4 Cooperative framework of Agentflow system.....	12
Figure 2_5 CPN definition	13
Figure 2_6 Transition rule	14
Figure 3_1 BPMN-PLAN meta-model.....	16
Figure 3_2 BPMN-PLAN sub-models and relationships	18
Figure 3_3 Explicit and implicit start event example	20
Figure 3_4 BP Event and BP Message Flow example.....	21
Figure 3_5 Example of uncontrolled gateway	22
Figure 3_6 Example of BP Message Flow usages.....	24
Figure 4_1 BPSDM	27
Figure 4_2 Join start and end events	33
Figure 4_3 Example of start event transformation	34
Figure 4_4 Example of end event transformation	35
Figure 4_5 Example of interrupted routines	36
Figure 4_6 Example of interrupted routines expanding	37
Figure 4_7 Example of inclusive gateway transformation	37
Figure 4_8 Example of event gateway transformation	38
Figure 4_9 Example of halt event transformation	40
Figure 5_1 Trading process	44
Figure 5_2 BPD after first phase	47
Figure 5_3 Goods redeployment	48
Figure 5_4 Join start events	49
Figure 5_5 Join end events	50
Figure 5_6 Expand interrupt routines	50
Figure 5_7 Expand gateways.....	51
Figure 5_8 Transform halt events.....	52
Figure 5_9 BPD after second phase.....	53
Figure 5_a Mapped PTD	56

Chapter 1. Introduction

Workflow at its simplest is the movement of documents or tasks through a work process with automation for achieving certain goals. It is the operational aspect of work procedures: how works are structured, who perform them, what their relative order is, how they are synchronized, how information flows to support the tasks and how tasks are tracked. In order to improve the performance of task processing, managing frequent changes of plans, as well as handling exceptions, *Workflow Management System (WfMS)* comes into existence.

Different vendors implement workflows in different ways, which may raise the difficulties in transferring specifications between WfMSs based on different paradigms. Many efforts have been spent by both industry and academia in attempting to define a common representation of a business process. Moreover, a WfMS works for people from different domains, while most of the workflow notations are designed for technical developers and thus hinder the communication and mutual understanding among developers and stakeholders. The development of a workflow can be divided into many phases. It is vital for the development process to have a systematic and efficient way to pass the acquired information to the next phase seamlessly.

Business Process Modeling Notation (BPMN) [1] is a process language which satisfies these demands. It is developed by *Business Process Management Initiative (BPMI)* as a standardized graphical notation for modeling business processes. The primary goal of BPMN is to provide a set of notations that are readily comprehensible to all business users, including business analysts, technical developers, and managers. Through standardizing the common modeling notations, the difficulties addressed

above are reduced. BPMN is more superior with providing a set of notations for better descriptions and enhancing the capabilities of traditional business process notations by inherently handling *business to business (B2B)* business process concepts. Note that in this thesis, the terms "workflow" and "business process" are considered identical.

However, BPMN supports only the modeling concepts that are applicable to business processes. Several practical concerns for business purposes are out of scope of BPMN. For instance, BPMN does not contain the notation for organization structures and data models. These corresponding notations and related semantics are necessary to be constructed and integrated with BPMN.

To make up the deficiency, one simpler way is to integrate BPMN with existent WfMS. *Agentflow* [2], which is the workflow platform in this thesis. It is developed by our laboratory that is mainly based on *Cooperative Agentflow Process LANguage (CA-PLAN)* [3] [4] and several prior researches. It has been implemented in practical *Business Process Management (BPM)* environment for years and consequently proofs the importance of its features.

In this thesis, we propose a modification of BPMN, i.e. *BPMN**, for clarifying several conflicts and obscure notations in the latest publication of BPMN. The *BPMN** is then integrated with CA-PLAN to build a conceptual model, *BPMN-PLAN*. In *BPMN-PLAN*, *BPMN** provides a standardized notation for CA-PLAN, while CA-PLAN makes up the rest deficiencies. In addition, *BPMN-PLAN* specification can be transferred to a CA-PLAN specification to take advantages of editing tools and execution environment of *Agentflow* system. To indicate the work, we present a corresponding development methodology and a transferring algorithm.

The rest of the article is organized as follows. Section 2 introduces BPMN, execution platform with frameworks, and related process modeling/analysis languages. Section 3 depicts BPMN-PLAN and the modified notations of BPMN, section 4 presents the development methodology and the transferring algorithms, and section 5 provides a simple case study to show the whole development process. Finally, conclusion and future works are made in the section 6.



Chapter 2. Background

2.1. Introduction of BPMN

BPMN is an evolving standard of process logic specifications for the business process management languages such as BPEL(BPEL4WS) [5] and *Business Process Modeling Language (BPML)* [6]. BPMN standardizes a modeling notation by applying many different modeling concepts and viewpoints. BPMN defines both the graphical notation and the semantics of a so-called *Business Process Diagram (BPD)*. BPMN is based on flowcharting techniques and consists of four categories: *Flow Object*, *Connecting Object*, *Swimlane*, and *Artifact*, and two modularization elements: *Process* and *BPD* (A BPD is also used to represent a definition instance of BPMN).

- Flow Objects are the core objects to define the process behavior. It contains *event*, *activity*, and *gateway*.
 - Events occur during the execution of a business process. They affect the flow and usually have a cause or an impact. BPMN discusses the events those affect the sequence or timing of activities of a process only. BPMN further categorizes events into three main types: start, intermediate, and end events.
 - Activities are works being performed. An activity can either be atomic or non-atomic (compound), which is represented by task and sub-process accordingly. BPMN predefines three basic types of sub-processes: embedded, independent, and reference sub-processes; and seven basic types of tasks: service, receive, send, user, script, manual, and reference tasks.
 - Gateways are used as decisions within a process for constraining the behaviors of divergence and convergence of the flows (e.g. two at most to

be chosen among the outgoing flows). There are four types of gateways defined in BPMN: *exclusive*, *inclusive*, *complex*, and *parallel gateways*.

- Connecting objects are the connectors among flow objects and artifacts. Their common attributes are "source" and "target", which represent upstream and downstream of a connecting object. A connecting object is a *sequence flow*, a *message flow*, or an *association*.
 - Sequence flows are used to show the conditions and the order in which flow objects will be performed or examined.
 - A message flow is used to show the communication among organizations, that is, communication within an organization is not included.
 - An association is used to associate artifacts to a flow object as input or output and activity to an intermediate event as compensation.
- Swimlane is used to help partitioning and organizing processes in a BPD. They must be used beyond the process composition for collaboration choreography. It is a *Pool* or a *Lane*.
 - A Pool generally represents a participant in the process, while in practical cases, a specific business entity or domain (e.g. an organization). It is used to divide a BPD into local and remote domains, and accordingly differentiates between local and remote processes.
 - A Lane is a sub partition of a Pool and is used to organize and categorize flow objects within a Pool which are logically related to each other (e.g. same department).
- Artifacts provide modelers with the capability of showing additional information about a process, in a process diagram, to help understanding the process. All of the artifacts have no directly effect on the process behaviors. It shall be modeler

who decides whether to include them for more information or to omit for reducing cutters. The Artifact is a *data object*, a *group*, or an *annotation*.

- A data object is used to represent the data instance manipulated by the process, including both electronic and physical.
- A group is a visual mechanism to group objects of a process informally for documentation or analysis purposes. It is out of the tree structure of business process.
- An annotation is a mechanism for modelers to provide textual description for the reader in a diagram (e.g. a paragraph of comment).
- A Process is generally an activity performed within an organization. In BPMN a process is depicted as a graph, which may be a group of activities and the controls that sequence them. Processes are intrinsically hierarchical that may be defined at any level, from an enterprise-wide process to a simple task performed by a single person. Low-level processes may be grouped together to achieve a business goal.
- A business process is a set of activities that are performed within an organization or among organizations. A business process may involve processes from distinct organizations and communication among them, namely *choreography*. In BPMN, a business process is described by a BPD. Compared with a process, a BPD may depict organizational domain definitions with swimlanes and not contain other BPDs.

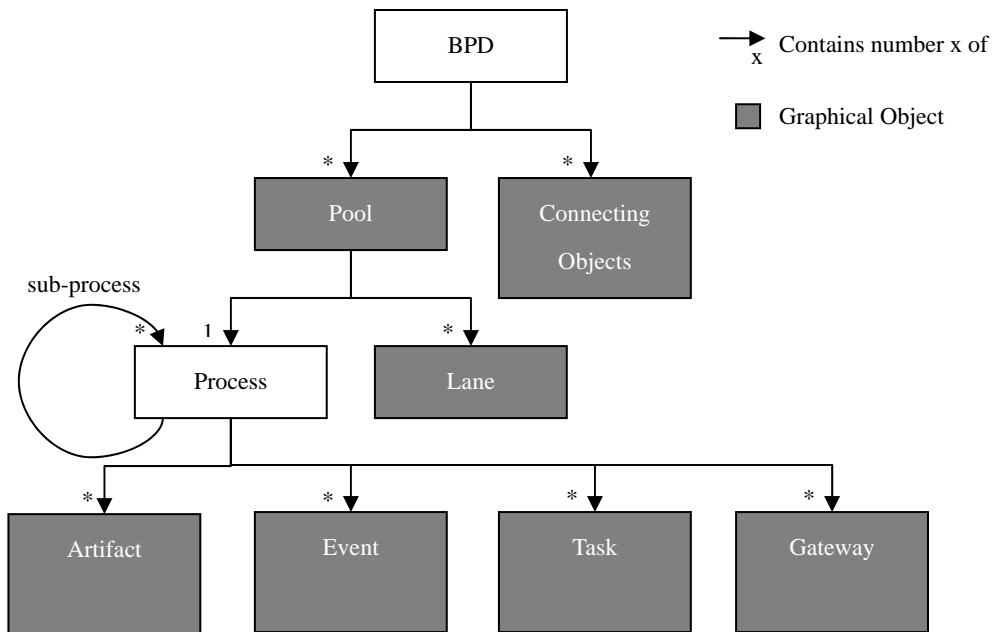


Figure 2_1 BPMN core objects architecture

BPMN provides three types of sub-models for modeling collaborations: *private (internal) business process*, *abstract (public) process*, and *collaboration (global) process*.

- A private business process is internal to an organization and is generally referred as workflow or local process.
- An abstract process represents the interactions between a private business process and processes of other participants. It includes the types of activities a local process uses to cooperate with remote services and the message sequences used for synchronization. An abstract process is also called a public interface of a process and its activities can be deemed the touch-points between participants.
- A collaboration process depicts the whole interactions between participants. It is defined as a sequence of activities that represent the interactions between the participants involved. The collaboration process can be shown as two or more

abstract processes communicating with each other. A collaboration process is also considered a global process and may be mapped to languages such as *ebXML* or *WS Choreography*.

BPMN also supports the process composition (aka block composition) that a process may aggregate one or more lower level processes recursively. In this way, the process definition may be highly abstracted and reusable. A large and complicated process can be modeled in top-down or bottom-up approaches.

2.2. Agentflow and CA-PLAN

2.2.1. Overview of Agentflow

Agentflow system developed by our laboratory is a typical WfMS. It provides workflow designers an integrated developing environment during design time, and a centralized server for automated process enactment, process control, execution monitoring, database management, and supporting workflow participants during run time. It consists of *Workflow Engine*, *Process Aid Software engineering Environment (PASE)*, *Process Definition Environment (PDE)*, and *Agenda*. Besides, Agentflow can be roughly divided into a three-tier architecture containing client/user interface, server/network connections, and database.

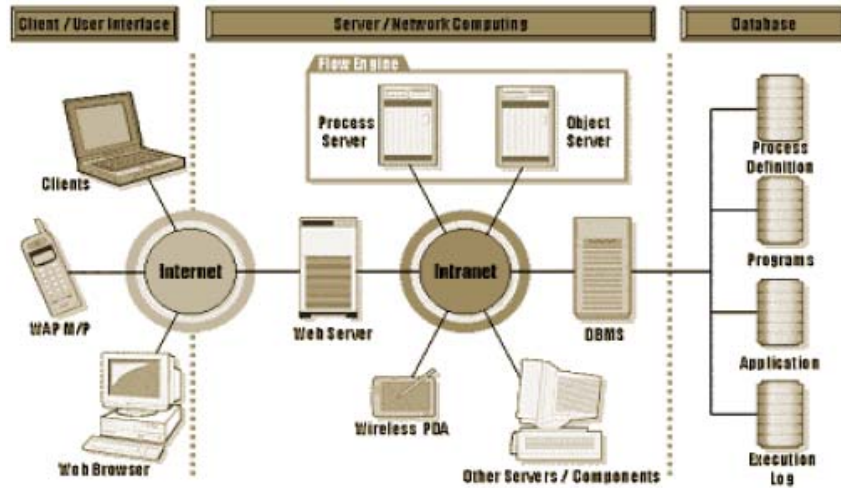


Figure 2_2 Agentflow system architecture

- client/user interface

PDE provides tools for modeling each view of a business process separately, including an Organization Designer for establishing the organization structure, an e-form Designer for designing the electronic document layout, and a Process Designer for modeling process logic and managing project modularizations. Agenda is the main part of the user interface during execution time, which is the client-side program of Agentflow system. Through Agenda, users can access personal task lists, initiate processes, invoke related programs, and track processes.

- server/network connections

The workflow server dispatches tasks to their users. It adopts the network centric computing concept.

- database

As for the Database Server or DBMS, it consists of data storage, electronic

documents, and process definitions, and it can integrate other databases.

Agentflow system provides two major software servers: PDE server is for an access control mechanism for process modules and also a process *definition repository* for storing process definitions; the other is PASE server, which includes a *runtime repository* for all the workflow run time data storage and records. The PASE provides an environment for process enactment, process monitor, and execution.

2.2.2. Introduction of CA-PLAN

The software process model, *Process LANguage (PLAN)* [7] [8] is the predecessor of CA-PLAN. It is used to model the software development project with how activities are performed by the people (roles), supplied with the resources, and lead to the product (artifact).

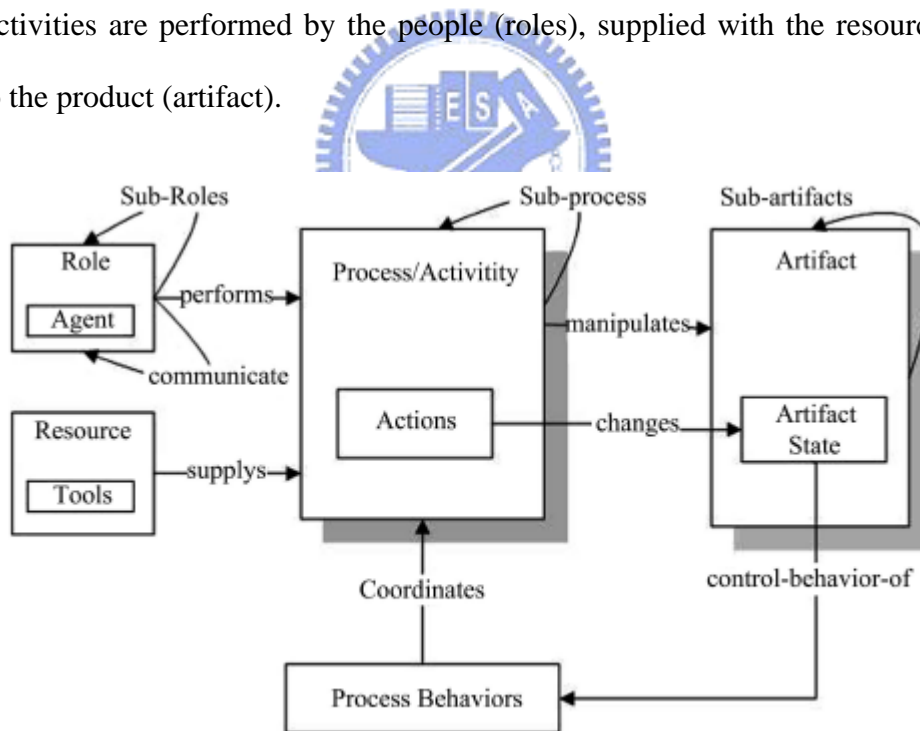


Figure 2_3 Conceptual framework of PLAN

CA-PLAN is an extension of PLAN that enables the modeling of inter-organizational business processes with *Remote Call Process (RCP)*, a

mechanism based on *Remote Procedure Call (RPC)* and a process model of PLAN. RCP works in a service-oriented aspect, allows the changes of services dynamically, and thereby adapting competitive business environment. RCP is assumed to run under a homogeneous environment of Agentflow systems, and the constructed system can thus take advantage of the services of Agentflow system, such as process monitor, version control, artifact, script, etc. Within heterogeneous environments, a service can be modified to adapt the reference specifications such as WF-XML. Each of the component systems can provide sets of services for cooperators and the underlying communications are handled by RCP transparently, which makes the cooperative process across organizations simpler, faster, and more flexible.

A cooperative framework of Agentflow system is proposed by CA-PLAN, which provides communication interfaces and mechanisms on PDE and PASE servers. A RCP registry is a process service repository for deployment of information of services provided by the corresponding organization. A WfMS can therefore find the available process services by looking up in the RCP registry.

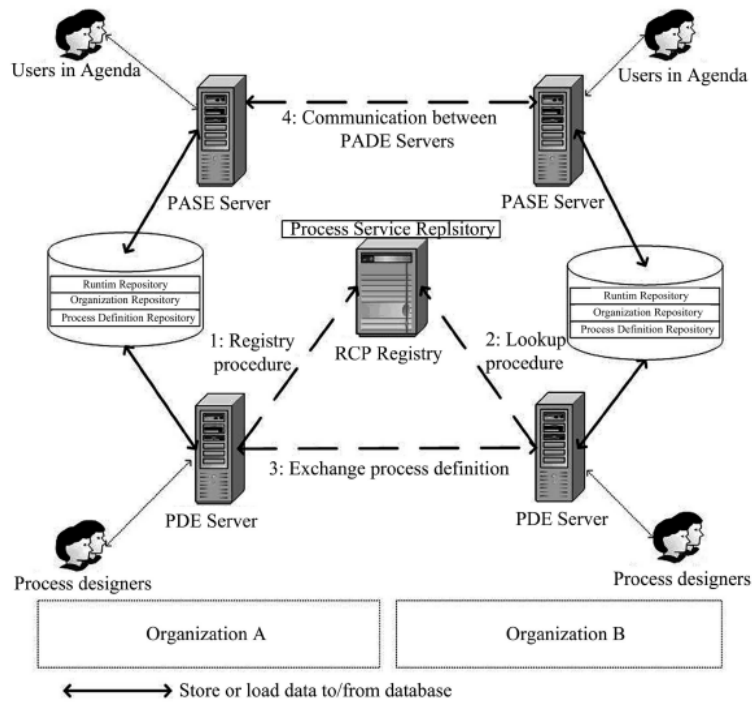


Figure 2_4 Cooperative framework of Agentflow system

In CA-PLAN, a process transition diagram (PTD) is used to depict the behavior of a process. It consists of a *start point*, an *end point*, *processes* (nodes), and *transitions* (links). The start and end points are where a process starts and finishes, the processes are works to be done, and the transitions are the relationships between the processes connected. Most of the execution behaviors are process-related and therefore are embedded in the processes.

2.3. Coloured Petri Net

Petri Net [9] is a formal and graphical oriented language that is suitable for design, specification, simulation and verification of systems with concurrency or competition (e.g. communication protocols, distributed systems, workflow analysis and VLSI chips).

Definition 3.3: A CP-net is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ where:

- (i) Σ is a finite set of non-empty **types**, also called colour sets.
- (ii) P is a finite set of **places**.
- (iii) T is a finite set of **transitions**.
- (iv) A is a finite set of **arcs** such that:
 - $P \cap T = P \cap A = T \cap A = \emptyset$.
- (v) N is a **node** function. It is defined from A into $P \times T \cup T \times P$.
- (vi) C is a **colour** function. It is defined from P into Σ .
- (vii) G is a **guard** function. It is defined from T into expressions such that:
 - $\forall t \in T: [Type(G(t)) = \mathbb{B} \wedge Type(Var(G(t))) \subseteq \Sigma]$.
- (viii) E is an **arc expression** function. It is defined from A into expressions such that:
 - $\forall a \in A: [Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$
where p is the place of $N(a)$.
- (ix) I is an **initialisation** function. It is defined from P into closed expressions such that:
 - $\forall p \in P: [Type(I(p)) = C(p)_{MS}]$.

Figure 2_5 CPN definition

Coloured Petri Net (CPN) [10] [11] is an extension of Petri Net and is completely backward compatible. Similar to Petri Net, CPN contains places, transitions, directed arcs and tokens as well. The characteristic of CPN is that each of the tokens carries a data value which belongs to a given color (type) - in contrast to the "uncolored" and indistinguishable tokens of Petri Nets. A token and its value are manipulated by transitions, which are fire-able if its input places contain sufficient tokens with colors that match the arc expressions functions.

Definition 3.6: A step Y is **enabled** in a marking M iff the following property is satisfied:

$$\forall p \in P: \sum_{(t,b) \in Y} E(p,t) \langle b \rangle \leq M(p).$$

We then say that (t,b) is enabled and we also say that t is enabled. The elements of Y are concurrently enabled (when $|Y| \geq 1$).

When a step Y is enabled in a marking M_1 it may **occur**, changing the marking M_1 to another marking M_2 , defined by:

$$\forall p \in P: M_2(p) = (M_1(p) - \sum_{(t,b) \in Y} E(p,t) \langle b \rangle) + \sum_{(t,b) \in Y} E(t,p) \langle b \rangle.$$

M_2 is **directly reachable** from M_1 . This is written: $M_1 [Y \rangle M_2$.

Figure 2_6 Transition rule

However, there are problems when modeling with Petri Net or high level Petri Nets (e.g. CPN); they do not support for some functions [12] (e.g. workflow patterns [13] [14] involving multiple instances, synchronizations). Moreover, using CPN for modeling industrial applications usually results in huge and complex diagrams. Although these processes can be decomposed into subnets recursively with high level Petri Nets, they are still too complicated for human understanding, and do not mention the burden of tracking the tokens by manpower.



Chapter 3. Business Process Model

In this section, BPMN-PLAN is introduced first. It is a conceptual business process model based on BPMN and CA-PLAN. Then, BPMN* is described to make up the process behavior sub-model of BPMN-PLAN.

3.1. BPMN-PLAN

Our experience of applying Agentflow and CA-PLAN in real world BPM environments indicates that the following features are essential for a practical business scenario:

- Organization structure

A complete organization structure needs to be created in the system to keep the organization's information dynamically. The structure may also be used in authority inheritance or work delegation.

- Artifact states

A product (artifact) can be represented with the states of the process, and the states might be applied for analysis.

- Electronic documentation

Our primary part of process automation is the electronic form (e-Form), which significantly saves paper and manpower cost. Also, the handling of e-Form is faster than traditional paper-based operations and efficient in maintenance. The e-Form can also be an inter-medium between developers and stakeholders during

requirements elicitation or verification.

- Service-oriented collaboration mechanism

Rather than end-to-end, the collaboration between organizations is based on the client-server service model. The transitions and message exchanging between them are simplified and compressed as service processes.

Therefore *BPMN-PLAN* is proposed. It is an extension model of BPMN which can be used to describe the requirements addressed above.

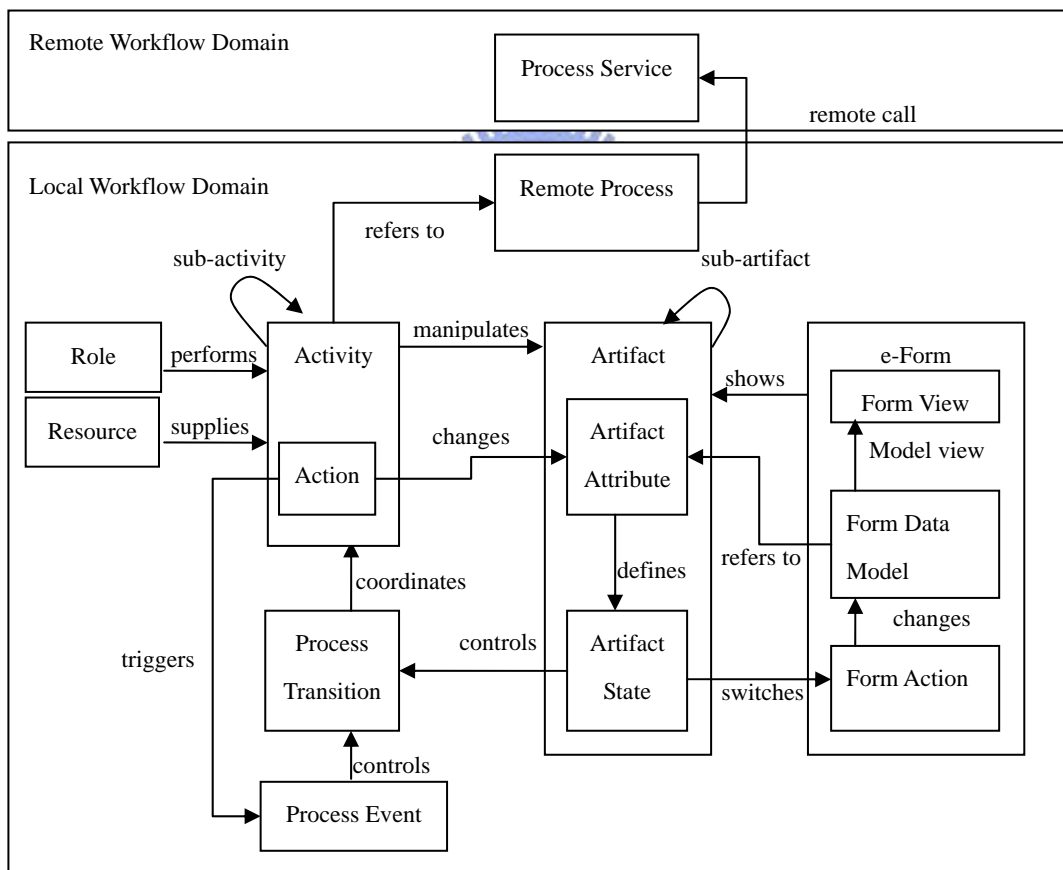



Figure 3_1 BPMN-PLAN meta-model

A *workflow domain (WD)* is classified as the smallest business entity in

collaboration of BPMN-PLAN. An organization can be divided into several sub-organizations of which each belongs to one or more WDs. Each domain contains its own process definition repository and runtime repository for WfMSs within. For the organization, a WfMS can be deemed as the integration of the local WfMSs in the same domain.

An activity manipulates one or more artifacts by reading or modifying its states. It might be defined as a computer program or a human activity. The *process event* is drawn out from process transitions for abstracting time stamps of business process. It is explicitly designated and expected to happen during runtime. The process event controls the process behaviors such as start, interruption, halt, recovery, and termination.



The collaboration mechanism of BPMN-PLAN is based on RCP mechanism, where a collaboration (inter-organization) process is partitioned into several private (intra-organization) processes. A *process service* is a mechanism that packs a set of processes. It can be used to provide service for process in other WDs they cooperative with. A *remote service* (process) is used when a local process attempts to cooperate with the processes in another WD; the local process uses remote service to refer to the process services provided by other WDs.

An artifact stands for the abstractions of data instances within the business process. Semantically, an artifact can be deemed as a finite state automaton where the starting state represents the instantiation of the artifact and the final state represents its deletion. Besides representing life cycle of artifacts, artifact states can simplify influences of data instances on process behaviors (e.g. Boolean expression of artifacts can be used as pre-condition and post-condition of activities).

An e-Form is a group of artifacts. For example, an e-Form contains a set of paper documents. The *form data model* refers to and manages access controls of artifacts associated with the form. The *form view* depicts the representation of the form data model. An e-Form might have more than one presentation (e.g. JSP or Applet). The *form action* describes the interaction between users and the form.

Resources are any type of entities other than above, for example, hardware machine, software component, laborer, etc.

A BPMN-PLAN model contains the following sub-models: *Process Behavior model*, *Role model*, *Artifact model*, *Form model*, and *Resource model*. Each of them provides distinct perspective for a business process. And an additional mechanism is given to bind related sub-models into a project module.

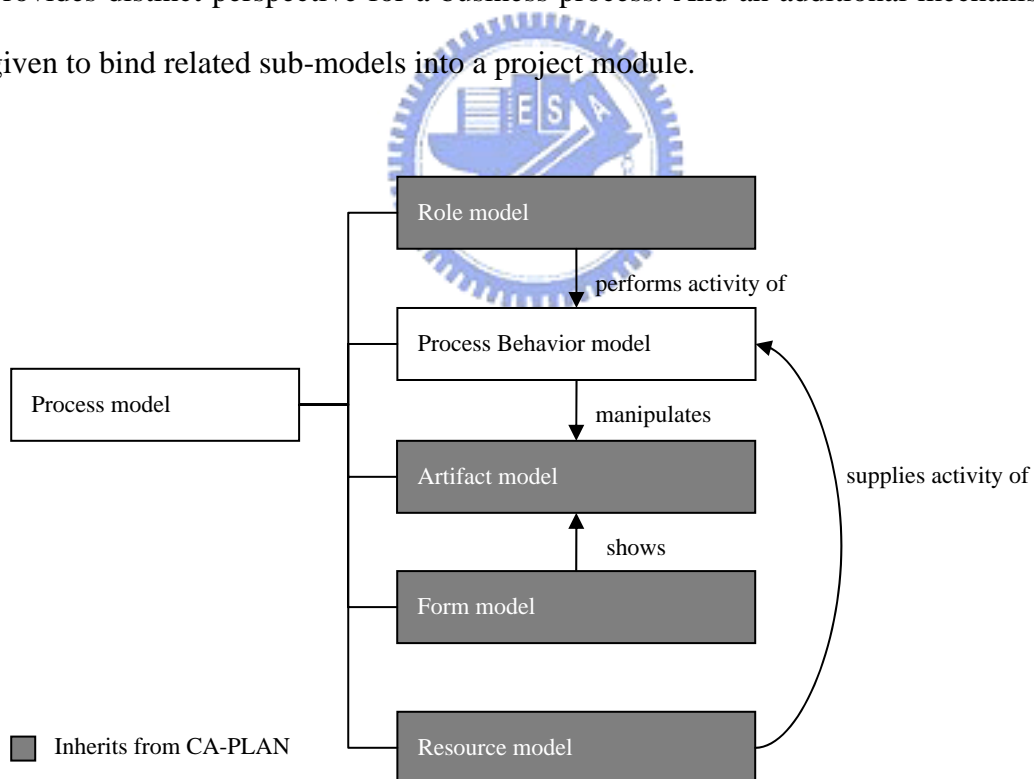


Figure 3_2 BPMN-PLAN sub-models and relationships

- Process Behavior model specifies the activity, process event, and process

transitions. We use a modified BPMN, BPMN*, for the modeling notation of Process Behavior model. The details of BPMN* will be discussed in next chapter.

- Artifact model provide an artifact tree that specifies aggregation relationship among the artifacts and an artifact state transition diagram for analysis on artifact state correctness [15].
- Form Model defines the data expressions, user access controls and artifact manipulations.
- Role Model illustrates the organizational properties of the process. It provides information about the performers of activities or the hierarchy of roles.
- Resource model consists of the information required to manage a process, and types and quantities of required resources to enable a process to accomplish its jobs.



3.2. BPMN*

To reduce the ambiguity within BPMN and simplify the development process, several constraints and clarifications are added. The modified notation is named as BPMN*, which is still conformity with BPMN. In this paragraph, the modified notations are described, while the directly inherited ones are left out.

The name of each category in BPMN* is extended with prefix "BP" to differentiate from the terms we used in CA-PLAN (e.g. the term "BP Artifact" differs from the artifact in CA-PLAN).

- **BP Flow Object**

○ BP Event

In BPMN, there are three types of events: start, end, and intermediate events. An intermediate event may behave in two ways, which shall be defined separately. In BPMN*, the BP Event associated with a BP Activity is called an *interrupt event*, while the rest of the intermediate events are called *halt event*. An interrupt event is associated with an interruption handling routine that reacts to the designate triggers. A halt event is placed in normal flows to halt the flow and represents an expected delay or waiting. The halted flow is not recovered until the designate trigger occurs.

In addition, for the readability of a process diagram, each process in BPMN* must explicitly designate start events and end events. For example, the diagram in lower part in figure 3_3 has an implicit start event that is forbidden in BPMN*.

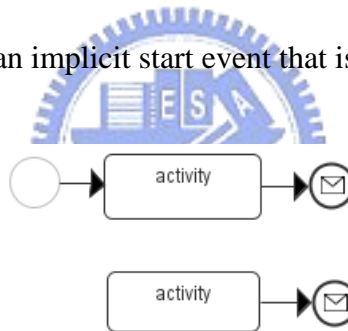


Figure 3_3 Explicit and implicit start event example

○ BP Activity

In BPMN, an activity of sending or receiving messages is also defined as an event. These definitions for event and activity concepts are confused. It is clarified in BPMN* that when the source or target of message flow is an activity, there is corresponding sending or receiving event implicitly. For example, both two diagrams in figure 3_4 are equivalent. (Since the idea of the end event results is to summarize the process, the upper one is recommended).

Therefore end and intermediate (halt) event constraints are applied to send and receive tasks accordingly.

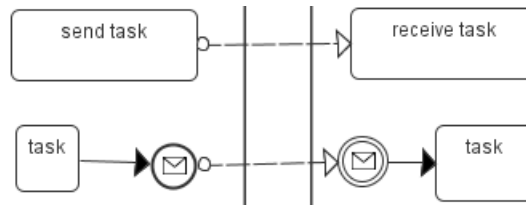


Figure 3_4 BP Event and BP Message Flow example

The pre-condition and post-condition based on additional attributes are added into each BP Activity, which constrain the starting and the normal finishing (since there may be an interruption) conditions accordingly.

o BP Gateway

In BPMN, there are four explicit and one implicit gateway, which are not distinctively differentiated from each other. Moreover, a contradiction in BPMN specification (exclude gateway used in merging in p.75 & p.119 [1]) is admitted by the author.



In BPMN*, gateways are reclassified as *exclusive* (data-based XOR), *event* (event-based XOR), *inclusive* (OR), *parallel* (AND), *uncontrolled*, and *complex*. An uncontrolled gateway is abstracted from BPMN, which represents none of control capabilities of other gateways. In other words, an uncontrolled gateway is a divergence or a convergence without centralized controls. It is classified as a gateway because of the essence of BP Gateways, that all splitting and joining of the flows shall be under the control of gateways.

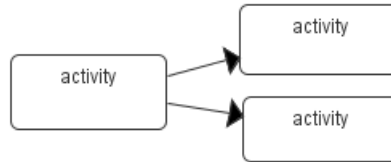


Figure 3_5 Example of uncontrolled gateway

Each of the six types of gateways may function as split (branch) or join (merge); they are described in the table 3_1. Note that the term "incoming flow" or "outgoing flow" here is used to represent a token flow.

Table 3_1 BPMN* gateways

	Split control	Join control
Exclusive	The first of the outgoing flows whose condition is met is chosen. A default choice may be set while none is satisfied.	The first arrived incoming flow is taken as merge result, while others are blocked.
Event	The first outgoing flow whose associated event is triggered is chosen.	
Inclusive	No control is applied on the outgoing flows. A default choice may be set while none is satisfied.	The inclusive join may easily cause structure problems and is excluded from BPMN*.

Parallel	No control is applied on the outgoing flows. Note that all outgoing flows are expected to be activated.	All the incoming flows are chosen to merge. Note that all incoming flows are expected to be activated.
Uncontrolled	No control is applied on the outgoing flows.	No control is applied on the incoming flows.
Complex	One or more outgoing flows are chosen (activate) based on the "outgoing condition" of complex gateway.	One or more incoming flows are chosen to merge based on the "incoming condition" of complex gateway.

- **BP Swimlane**

Each process is contained with a BP Pool, which represents a distinct domain. For each BPD of the collaboration sub-models (i.e. private, abstract, and collaboration process), there shall be at least one BP Pool specified for local domain. In our cases, a BP Pool is used to indicate a WD.

As for the BP Lane, it is preserved for attribute assignments and annotations.

- **BP Connecting Object**

A BP Message Flow may be considered as a control flow which synchronizes the processes among business entities. It is named *synchronous flow* and definitely specifies the sequences of processes among business entities. For example, in the upper part of figure 3_6 represents the execution sequences: send task, remote process, then receive task.

Otherwise, it is used to merely represent the communicative behavior of

processes. In this case, it is named as *communication annotation*. For example, in the lower part of figure 3_6 does not define the order between local and remote process.

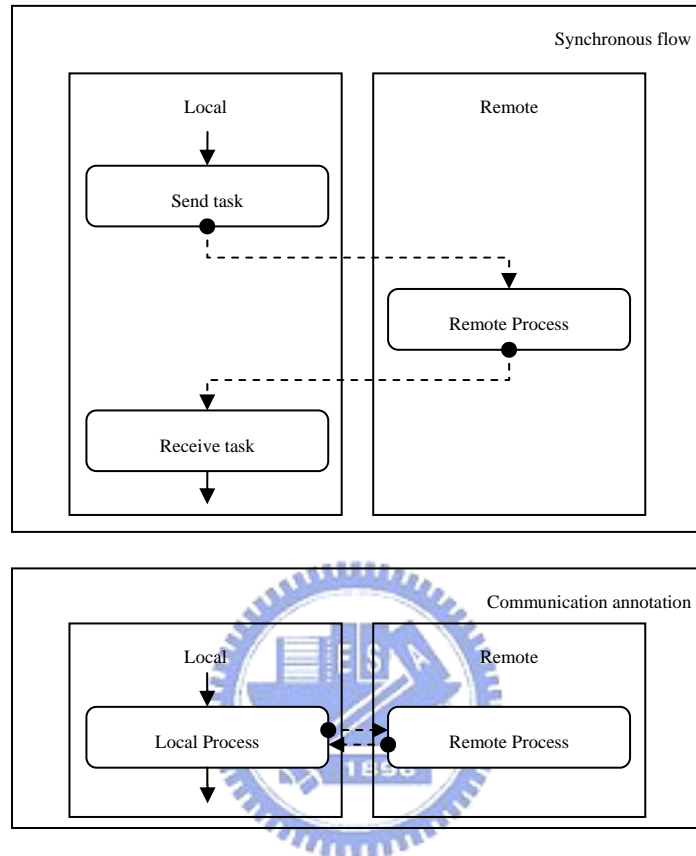


Figure 3_6 Example of BP Message Flow usages

As for the rest of BP Connecting Objects, i.e. BP Sequence Flow and BP Associations are remaining unchanged from BPMN.

- **BP Artifact**

No modification is made; they inherit the properties from BPMN objects directly.

Chapter 4. Developments with BPMN-PLAN

4.1. Development methodology

We developed a development methodology with BPMN-PLAN, named *Business Process Management Software Development Methodology (BPMSDM)*, as shown in figure 4_1. The topmost block is the requirement specification phase. The requirements are specified according to the five models in BPMN-PLAN, which are process behavior, reference data instance, e-form, role, and resource specifications. The process behavior specifications are specified with BPMN*.

After the requirements are specified, they are analyzed in the specification analysis phase to ensure the completeness and the consistency. Completeness means that all requirements are covered by the specifications, and consistency indicates that there is no conflict between the specifications. If there is no problem, the specifications are used as the input of next phase. In case a problem occurs, the specifications are modified accordingly.

In the system design phase, the Process Behavior model is defined first. It is a more detailed definition of the process behavior specification, which is obtained in requirement specification phase. For instance, the artifact association, activity per-condition and post-condition assignment, etc. In Process Behavior model, where a business process is defined in terms of business component objects and their interactions. To obtain an executable model, i.e. CA-PLAN definition, the Process Behavior model is transferred to the Activity model of CA-PLAN, and the rest sub-models are specified according to requirement specifications. The transferring process also creates the design templates of the four other sub-models. These

templates are used to help to build the sub-model definitions with the editing tools in Agentflow system, based on the corresponding requirement specifications obtained in first phase. For example, role templates are created from the participant information defined in Process Behavior model, and they are used to construct one or more Role models with the role specifications.

After all the sub-models of CA-PLAN are designed, they are analyzed to ensure the business process is executable, which indicates structural correctness [16] and satisfaction of resource constraints [17]. If there is any conflict in the CA-PLAN definition, the development is rolled back to system design or requirement specification phase.

The development ends up with a complete business process definition in CA-PLAN, which is executable on an Agentflow platform. The definition is then taken into testing phase, execution phase, and so forth, which are out of the scope of this thesis and are not discussed.

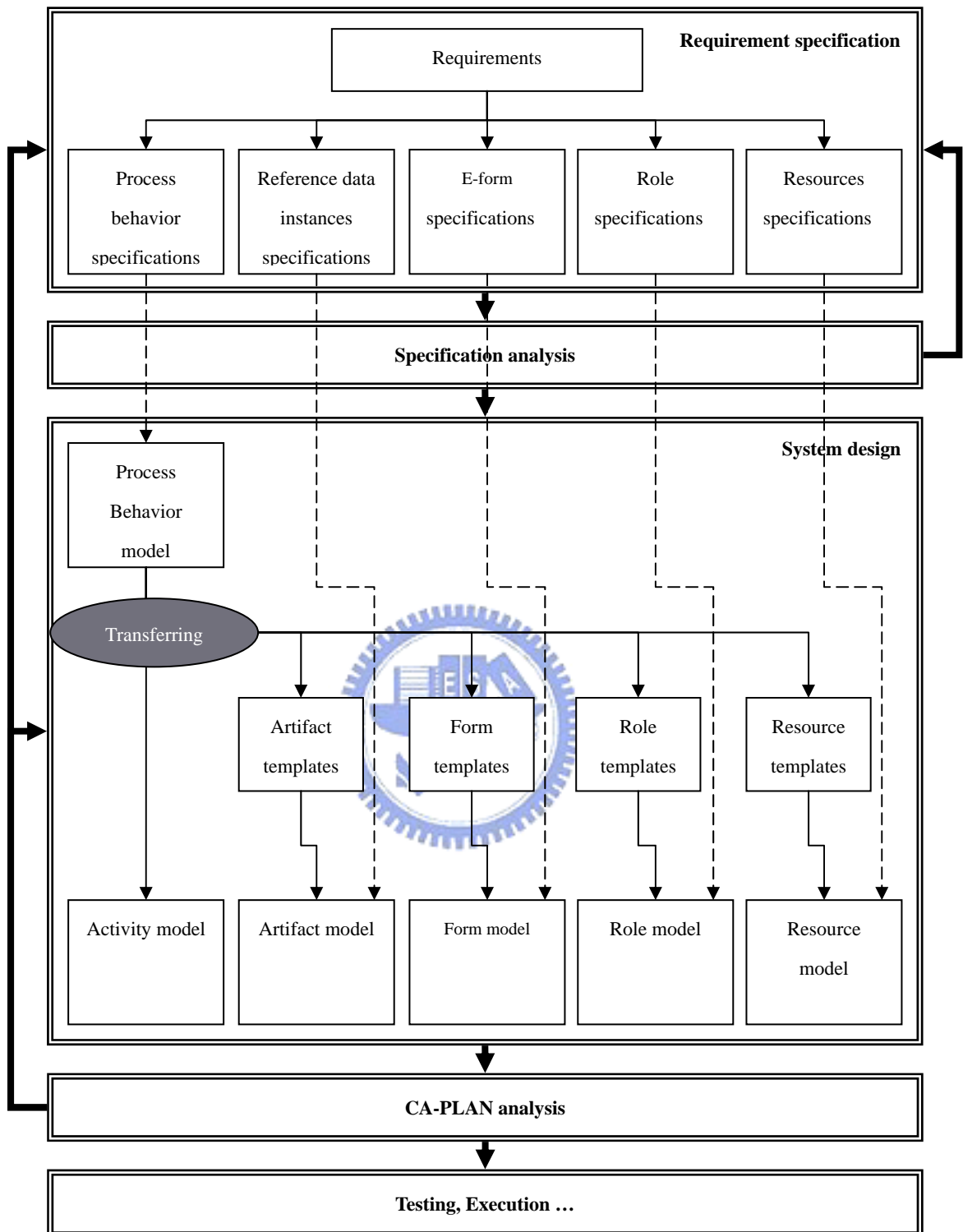


Figure 4_1 BPSDM

4.2. Transferring algorithm

In this section, we present an algorithm for transferring a BPMN* definition to a PTD definition of CA-PLAN.

As the comparison in the table below, BPMN* has richer notations than PTD. The objects they have in common are compose-able processes (BP Activities), conditional transitions (BP Sequence Flows), start and end point (start event, end event).

Table 4_1 BPMN* versus PTD

BPMN* object	PTD object
BPD	Project
Multiple start and end events (with triggers or results)	Single start and end point (without triggers and results)
Intermediate event	Events are embedded in processes
BP Activity	Process
BP Gateway	None (each transition is independent from others)
BP Sequence Flow (with condition)	Transitions (represented by pre-condition and post-condition of connected processes)
BP Message Flow	None
BP Association	None
BP Pool, BP Lane	None

BP Artifact	None
-------------	------

The transferring algorithm 4_1 works in three phases. First, in line A01, it recomposes the inter-organization processes into intra-organization processes. In line A02, it transform specific components in the BPD. After these two phases, we shall have a "compressed" BPD whose components are ready for the one-to-one mapping to PTD at the last phase in line A03. The details including the input and the output of each phase will be put in the following subsections.

Algorithm Mapping BPMN* (B);	
Input:	B (BPD of BPMN*)
Output:	B (PTD of CA-PLAN)
A00 begin	
A01	Recompose collaboration process (B);
A02	Transform Process (B.process);
A03	One-to-one Mapping (B);
A04 end	

Algorithm 4_1 Transferring BPMN*

4.2.1. Phase 1 : Recompose collaboration process

Since the collaborations in CA-PLAN are depicted in a service-oriented approach, we presume that the inter-organization process in BPMN* is defined in the format of call and return with BP Message Flows. A *caller* is a process which requests a remote service by sending a segment of message, and a *waiter* is a process which waits for the remote service to be done and accepts the reply. For clarity, we define that a caller is the source of the call message flow, and a waiter is the target of the return message flow. Both processes own the parent process in common according to

process composition.

The inessential remote information is removed, that is because the internal details of a remote service are not concerned within RCP mechanism, and all of them are regarded as "black box" services.

In algorithm 4_2, we find each pair of BP Message Flows (call and return) and their local contact processes, i.e. caller and waiter. If both the caller and waiter are the same process (e.g. the lower part of figure 3_6), it is identified as a pair of communication annotations and bypassed. Otherwise, a pair of synchronous flows (e.g. the upper part of figure 3_6), and the caller and waiter is replaced with a virtual process named "independent process" in line 05. Pre-condition and post-condition of the independent process are same as the remote process. After we are done with the pair of BP Message Flows, they are removed from the BPD in line 07.

The rest non-pair BP Message Flows are considered as communication annotations and are removed in line 09. At the last of the procedure, all remote BP Pools and their associated processes are removed in line 10.

```

Algorithm Recompose collaboration process (B);
Input:    B (BPD of BPMN*)
Output:   B (BPD without BP Swimlanes, BP Message Flows)

00  begin
01      while B contains any pair of BP Message Flows do {
02          find a pair of BP Message Flows 'M1' and 'M2'
03          find caller, waiter 'O1', 'O2' of M1, M2
04          if O1 NOT equals O2 {
05              Transform independent process (O1, O2);
06          }
07          remove M1, M2 from B
08      }
09      remove non-pair BP Message Flows
10      remove remote domains information from B
11  end

```

Algorithm 4_2 Procedure Recompose collaboration process

4.2.2. Phase 2 : Transform process

As prior addressed, there are several notations in BPMN* which has no correspondent in PTD. In this phase, components of these notations are transformed into other BPMN* components which have correspondents in PTD. For example, there is no corresponding PTD notation for the halt event, and thus the halt events in BPD are transformed into the listen tasks.

Therefore, the process is transformed step by step into a "compressed" process whose components are one-to-one corresponding to PTD.

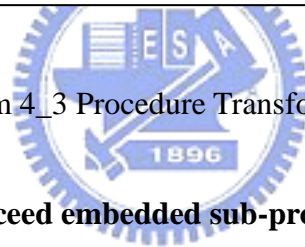
```

Algorithm Transform Process (pc);
Input:    P (process or sub-process)
Output:   P (“compressed” process)

00  begin
01      for each embedded Sub-Process ESP in P do {
02          Transform Process (P.ESP);
03      }
04      Remove ignorable commentaries (P);
05      Join start and end events (P);
06      Expand interrupted routines (P);
07      Expand gateways (P);
08      Transform halt events (P);
09      Arrange sequence flow conditions (P);
10      Duplicate references (P);
11  end

```

Algorithm 4_3 Procedure Transform Process



Lines 01-03 Recursively proceed embedded sub-processes

At the beginning of the procedure, each embedded sub-process of current process is proceeded in advance. Thereby the algorithm recursively works from the bottom to the top level of process composition.

Line 04: Remove ignorable commentaries

The commentaries such as BP Associations and BP Artifacts are redundant. They are removed after validation.

Line 05: Join start and end events

In BPMN*, a start event in BPMN* is corresponding to a start point in PTD, and

an end event is corresponding to an end point. There might be more than one start or end event in a process, while they are not supported in CA-PLAN. To solve the incompatibility, in each process diagram, a start event with no trigger named as *initiate event* is added to represent the single start point, and an end event with no result named as *complete event* is added to represent the single end point. Additional BP Sequence Flows are also added to link together begin event and start events, or end events and complete event.

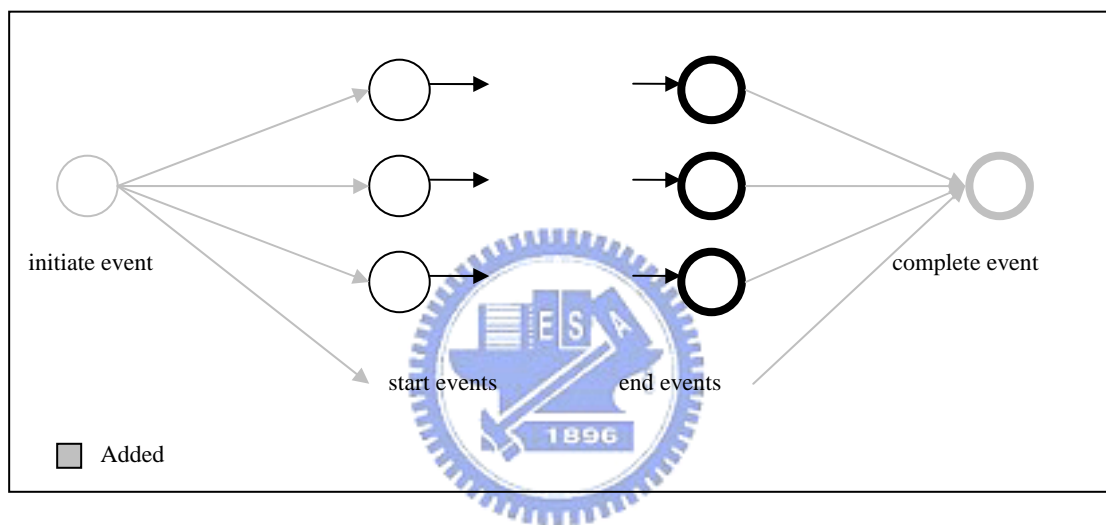


Figure 4_2 Join start and end events

After inserting the begin event, all the start events need to be transformed into halt events with corresponding triggers to obey the BPMN* semantics while preserving the "expecting" behavior. Table 4_2 indicates the transformations of start events. This transformation is only considered as a transit, the forbidden syntax of intermediate events in BPMN (e.g. a halt event with "multiple" trigger) are ignored here.

Table 4_2 Start events transformation

Start event	Action
None (no trigger)	It is omitted.
With trigger: BP Message, Timer, Rule, Link, or Multiple	It is transformed into a halt event with the corresponding trigger.

For example, in figure 4_3, there are two routines A and B. The A routine starts with a no trigger start event, while B routine starts with a link start event. After the initiate event is added, the start events are transformed into halt events.

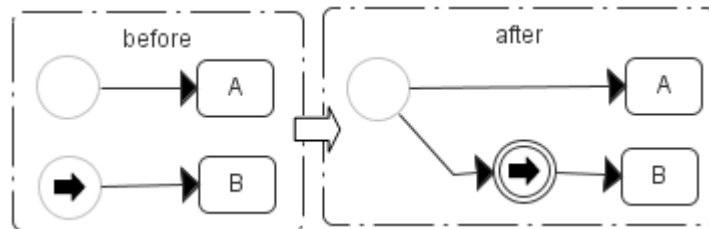


Figure 4_3 Example of start event transformation

The end events are transformed into the tasks, which perform the corresponding actions. Table 4_3 indicates the transformations of end events. The post-condition of each transformed task is set to the completion of the action.

Table 4_3 End events transformation

End event	Action
None (no trigger)	It is omitted.
Message trigger	It is transformed into a send task.
With trigger: Error, Cancel, Compensation, or Link	It is transformed into a (inter-processes message) send task.
Terminate trigger	It is transformed into a task with system

	call that terminates current process.
Multiple trigger	It is transformed into an embedded sub-process with corresponding tasks only.

In figure 4_4, two routines B and C end with results "message" and "none" accordingly. After the complete event is added, the end event with message result is transformed into a task that sends message, while the end event with no result is omitted.

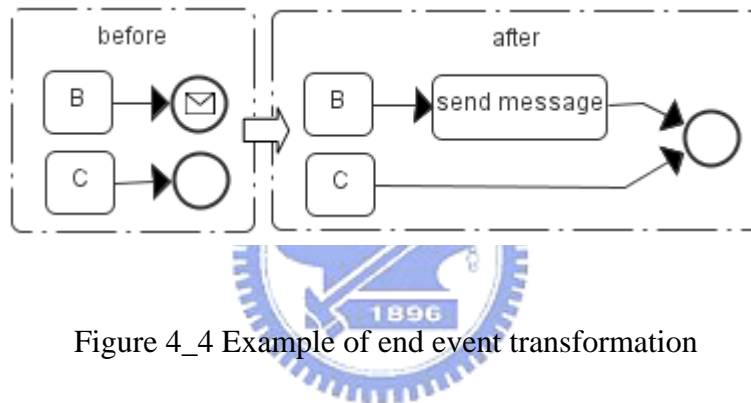


Figure 4_4 Example of end event transformation

Line 06: Expand interrupted routines

An *interruptible activity* is a BP Activity attached with at least one interrupt event. A *complete routine* is the subsequent routine of normal completion of the interruptible activity, and interrupt events and their subsequent routines are grouped as *interrupt handling routines*. For example, the gray area of figure 4_5 is an interrupt handling routine that is designated to be triggered by a cancel event.

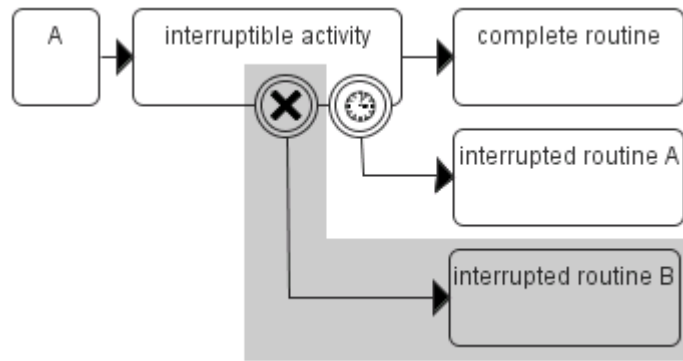


Figure 4_5 Example of interrupted routines

In CA-PLAN, an atomic process is considered as a transaction which can be disrupted by messages (e.g. cancel an assigned work which is out of date). When an interruptible process is interrupted, an interrupt handling routine is executed and then the control is given back to the interruptible process. The problem is that PTD does not support the modeling of interrupt handling routines, and therefore it is necessary to expand the interruptible activities to fit in with PTD.

A halt event with link trigger is added to the complete routine to represent the normal completion of the interruptible activity. The interrupt handling routines are rearranged to become the outgoing flows of the interruptible activity. To decide the subsequent flow among them, a deferred choice pattern (event gateway) is applied after the completion of interrupted activity. Thereby the branch among complete routine and interrupt handling routines are decided according to the completion state of the interruptible activity, i.e. the first occurrence of event (e.g. normal completion or interrupted).

For example, the process diagram in figure 4_5 is expanded to the one in figure 4_6. If the interruptible activity completes normally, the complete routine is activated. Otherwise, when timer or cancel event occurs first, the interrupted routine A or B will

be activated accordingly.

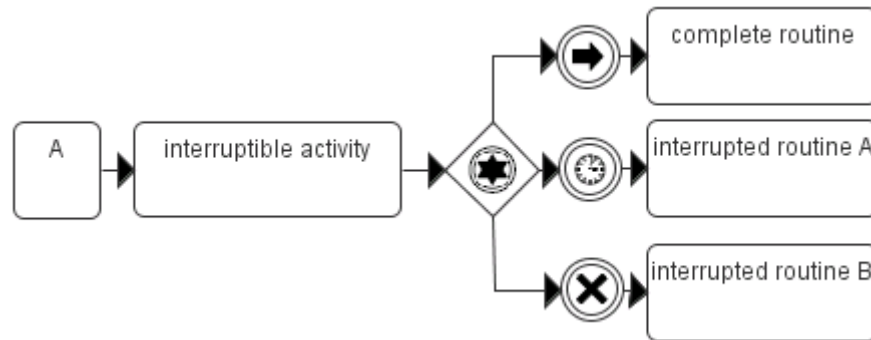


Figure 4_6 Example of interrupted routines expanding

Line 07: Expand BP Gateways

BP Gateways centrally control the convergence and divergence behaviors of connected BP Sequence Flows, which correspond to transitions in PTD. Since there is no corresponding concept in PTD, the BP Gateways are expanded.

The idea of expanding a BP Gateways is to scatter its constraints over the connected BP Sequence Flows. A *dummy task* which performs nothing is used to replace the BP Gateway, and the constraints of the BP Gateway are converted to its pre-condition or post-condition. In this way, the dummy task constrains the incoming or outgoing flows as BP Gateway does.

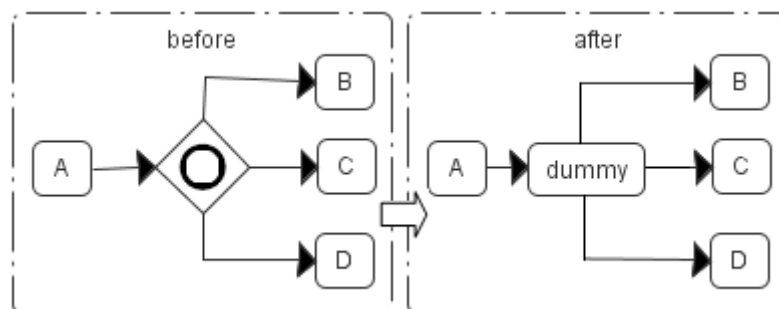


Figure 4_7 Example of inclusive gateway transformation

However, the event gateway used in branch function is unlike the others. It combines the behavior of halt event and exclusive gateway, that the decision of event gateway is delayed until the first of the designated BP Events is triggered. The event gateway is expanded to halt events with their triggers, and the exclusive constraint is associated with the outgoing flow condition of the halt events.

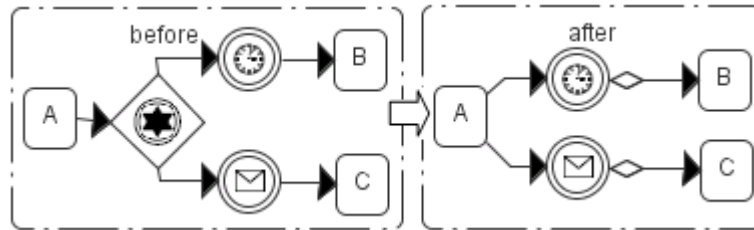


Figure 4_8 Example of event gateway transformation

The details of BP Gateways mapping is described in table 4_4 below.

Table 4_4 BP Gateways expanding

type\ function	Split	Join
Complex	Replace gateway with a dummy task, whose post-condition is outgoing condition of the gateway.	Replace gateway with a dummy task, whose pre-condition is incoming condition of the gateway.
Exclusive	Replace gateway with a dummy task, whose post-condition is exclusive choice of the outgoing flows.	Replace gateway with a dummy task, whose pre-condition is exclusive adopting of the incoming flows.

Event	Expand gateway to halt events, and its exclusive constraint is associated with each outgoing flow of the halt events, e.g. figure 4_8.	
Inclusive	Replace gateway with a dummy task.	Excluded from BPMN*.
Parallel		Replace gateway with a dummy task, whose pre-condition is set to wait for all the incoming flows.
Uncontrolled	No action	No action

Line 08: Transform halt events

The pause and recovery behaviors of halt events are implemented with the aid of inter-processes messaging. An expected trigger is expressed as a message, while *listen* (wait) and *notify* (trigger) processes are described by receive and send tasks accordingly. Halt events are therefore transformed into receive tasks, whose action is waiting for a trigger message. Note that the inter-processes messaging must be distinguished from the inter-organizational messaging, i.e. BP Message Flow in BPMN*.

Table 4_5 Halt events transformation

Halt event	Action
None (with no trigger)	It is omitted.

With trigger: Message, Error, Timer, Compensation, Rule, or Link	It is transformed into a corresponding receive task, whose post-condition is set to the completion of trigger.
--	--

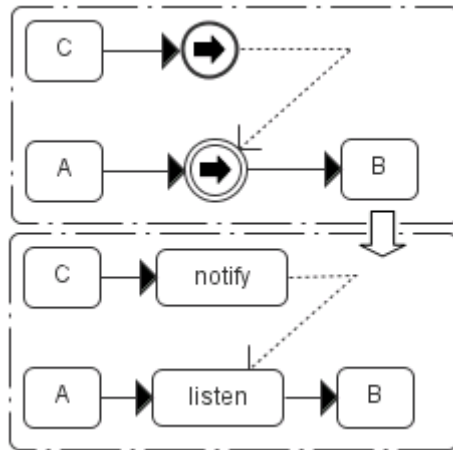


Figure 4_9 Example of halt event transformation

Line 09: Arrange BP Sequence Flow Conditions

In CA-PLAN, we can use a *transition* to represent the combination of pre-condition and post-condition of processes. To transform a BP Sequence Flows into a transition of PTD, the conditional constraints of each BP Sequence Flow are associated with the pre-condition and post-condition attribute of each connected BP Activity.

Line 10: Duplicate references

The reference activity is used to reference another sub-process or task. All the references share exactly the same behaviors and properties with the referent. The reference activities have no corresponding notations in PTD and each of them is replaced with a copy of the referent (i.e. referenced activity).

4.2.3. One-to-one mapping

The BP Activities are similar with processes in the CA-PLAN. They are mapped to corresponding processes of PTD in this step.

Table 4_6 indicates the relations between BPMN* BP Activities and PTD processes. In BPMN*, an independent sub-process can be used to represent invoking either a local process module or a remote service supported by data-mapping (e.g. RCP); it is mapped to a corresponding process accordingly. In comparison with an independent sub-process, a service task is used to represent message-based services rather than RCP (e.g. a web service). Tasks are mapped to atomic processes in PTD with appropriate programming constructs, whose detailed behavior is adjusted latter. The attributes associated with process are also used to generate design templates.

Table 4_6 BP Activities mapping

BP Activity	Corresponding process in PTD
Embedded sub-process	Compound-process
Independent sub-process	Call-process (local) or remote service (remote)
Reference sub-process, reference task	None (Transformed in procedure "Transform process", line 10)
Service task, receive tasks, send task, user task, script task, or manual task	Atomic processes

The rest of the BPMN* notations, i.e. initiate event, complete event, and BP

Sequence Flows, are mapped directly into corresponding notations of PTD. Table 4_7 indicates the relations among these notations.

Table 4_7 Miscellaneous notation mapping

BPMN* notation	Corresponding object in PTD
Initiate event	Start point
End event	End point
BP Sequence Flow	Transition



Chapter 5. Case study

In this section the development of a trading process is taken as example to demonstrate the methodology. This case concerns an on-line trading system that accepts customers' orders, redeploys and delivers ordered goods, interacts with a banking system, and notifies customer when the trading process is complete. The banking system is a cooperative system that provides service for the trading system to confirm or rebate the customer payment.

5.1. Requirement specifications

We define the process specification from the perspective of trading system. The orders from customers are presumed to be handled by front-end systems, such as a web shopping system. Thereby the interactions between the trading system and its customers can be simplified as events.

The whole trading process begins with receiving the order from a customer, which has been validated by a front-end system. The trading system then consults the bank for confirming the payment. If the remittance is not confirmed, a failed notification is sent to the customer and the process is terminated; otherwise the system starts to redeploy the ordered goods. In case the order is canceled during redeployment, the payment is rebated to the customer. As soon as redeployment completes successfully, the goods are delivered and the invoices are filed. At last the reply of successful delivery notification is sent to customer.

- Process behaviors

The process behavior is specified as BPMN* diagram in figure 5_1.

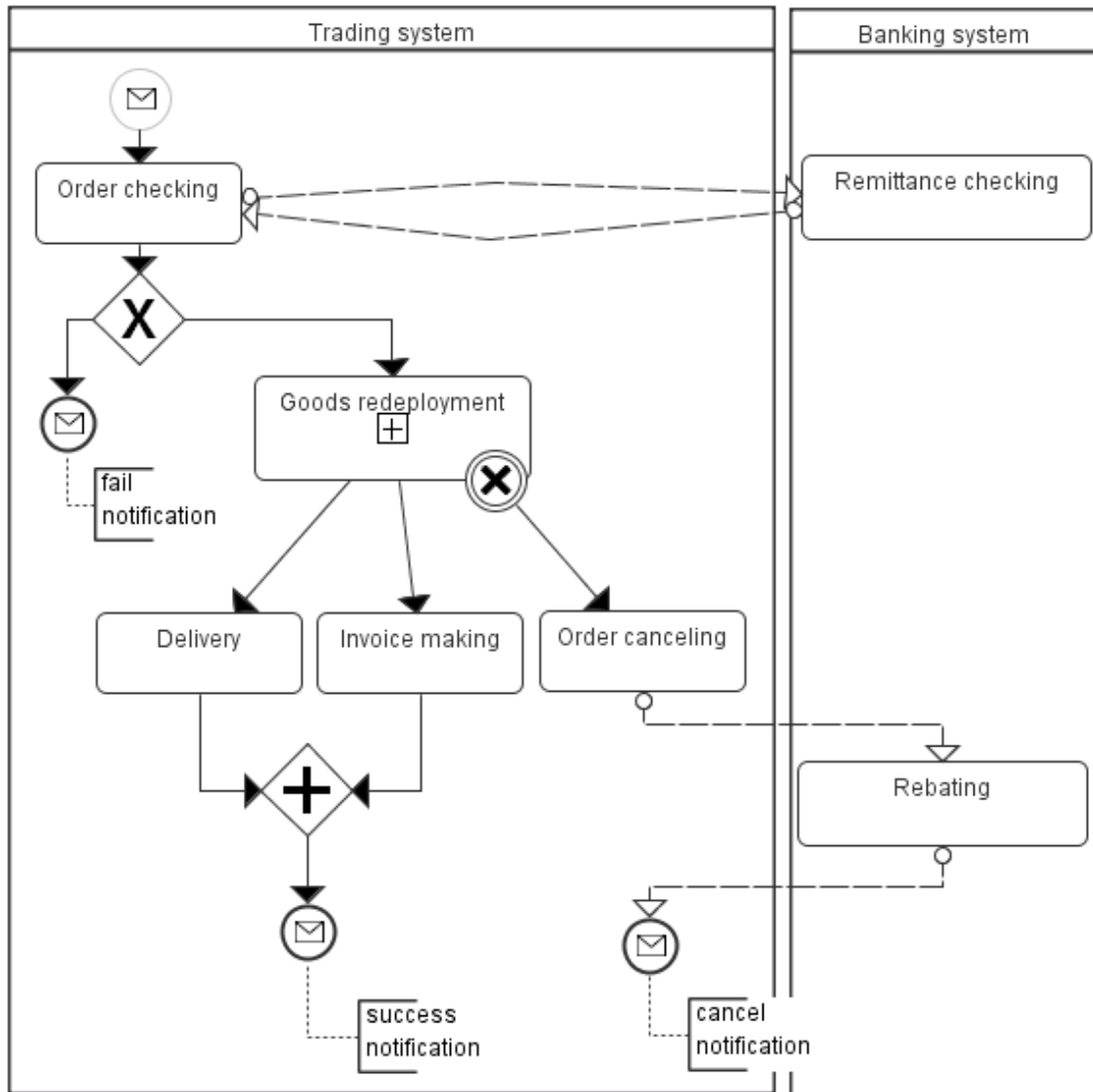


Figure 5_1 Trading process

- Reference data instances
 - Customer information
 - Deliver address, contact methods
 - Payment account, rebate account
 - Goods information
 - Goods ids, quantity, price
 - Invoices

- Process information
 - Checking result (boolean)
- E-Forms
 - Goods deployment form (GDF)
- Roles
 - Customer
 - Trading system
 - Banking system

5.2. System design

- **Process behavior modeling**

Table 5_1 Process events

Event	Trigger/ result
Incoming order (start event)	An order (message from the customer) is received.
Fail notification (end event)	The notification is sent.
Cancel interrupt (interrupt event)	The cancel notification is received during deployment.
Cancel notification (end event)	The notification is sent.
Success notification (end event)	The notification is sent.

Table 5_2 Process stages

Process stages	Start condition	End condition
Order checking	The order is received.	The order checking has been done.
Goods redeployment	The order checking result is confirmed. (checking result = T)	The goods have been deployed or the order has been canceled. (deployment result=T or F)
Order canceling	The redeployment is canceled.	The order has been canceled.
Rebating	The order is canceled.	The payment has been rebated.
Delivery	The goods are deployed.	The goods have been delivered.
Invoice making	The goods are deployed.	The invoices have been made.

- **Transferring**

For the first phase, the collaboration processes are recomposed. There are two remote processes defined for the trading in the banking system. One is the "remittance checking" process which interacts with the "order checking" process. This pair of BP Message Flows is considered as communication annotations and the "order checking" process is considered as a service task. The other is the "rebating" process executed after order canceling process. It is synchronized with an implicit send and receive event, which are transformed into an independent process named "rebating request".

The result of this phase is showed in figure 5_2.

Note that if the process behavior is specified with BPMN, the modeler will likely be confused with the execution sequences of local and remote processes.

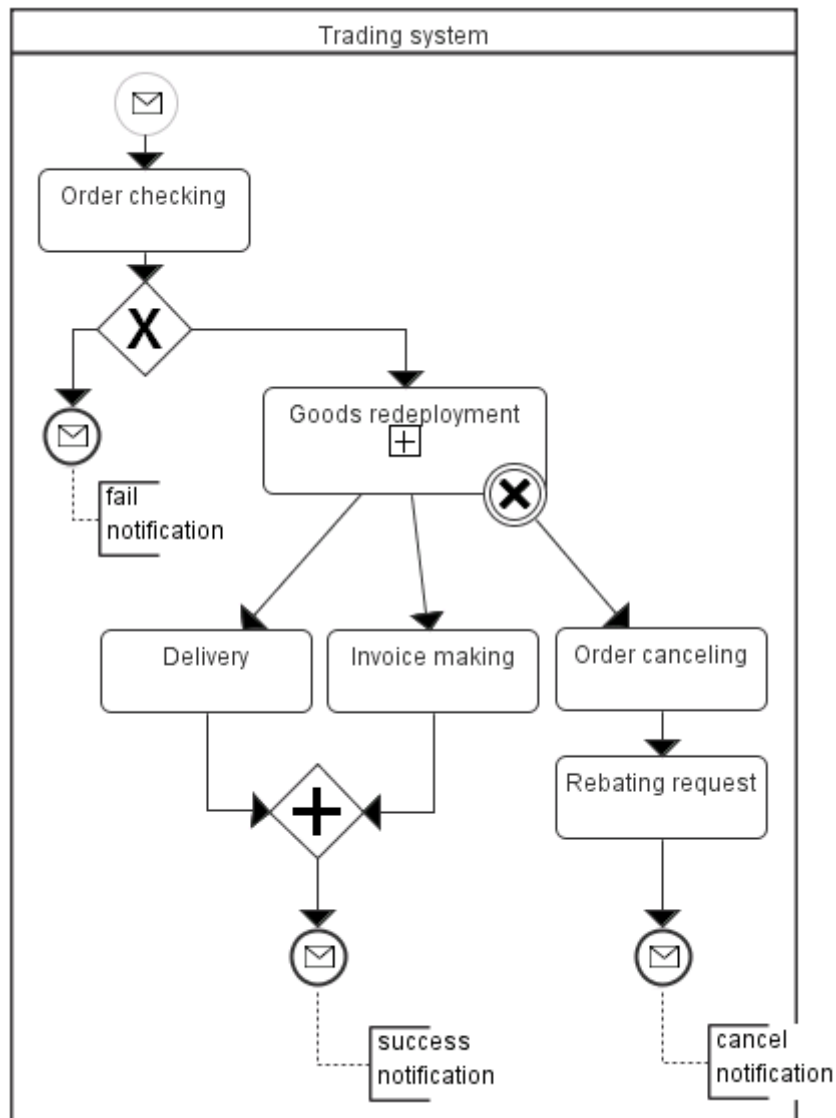


Figure 5_2 BPD after first phase

In the second phase, the embedded process, i.e. "goods redeployment" process, is proceeded in advance. It is expanded as in left part of figure 5_3, which contains a loop that keeps checking if all the ordered goods are in inventory. If all the ordered

goods are in stock, the redeployment process ends. Otherwise the process is halted to wait for replenishment notification.

The algorithm proceeds the redeployment process as follows.

1. Expand gateways

The exclusive gateway is replaced with a dummy task whose post-condition is the exclusive constraint of exclusive gateway.

2. Transform halt events

The halt event is transformed into a receive task which listens to each notification of replenishment.

3. Arrange sequence flow conditions

The conditions of BP Sequence Flows after the dummy task are associated with post-condition of the dummy Task.

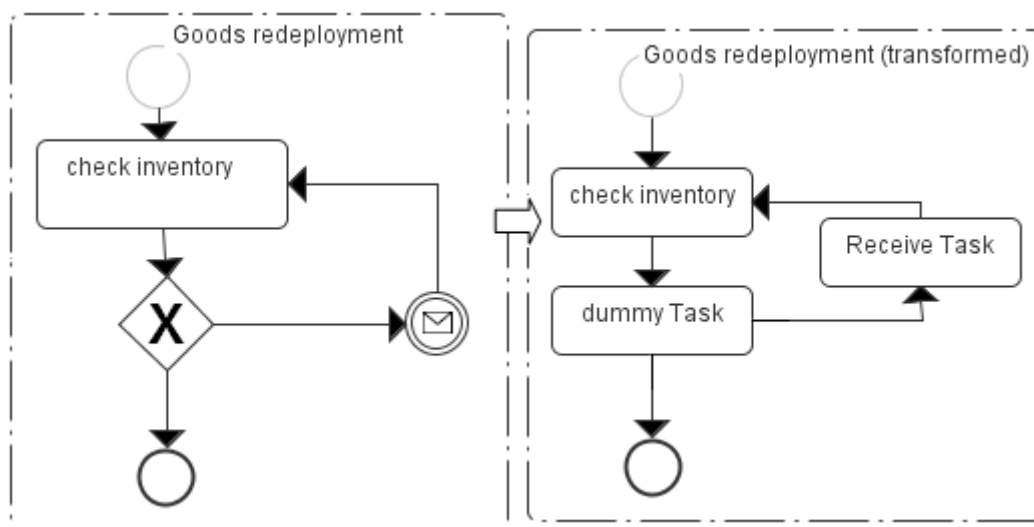


Figure 5_3 Goods redeployment

Now, the transferring of the rest sub-processes in trading process in figure 5_2 continues as follows.

1. Remove ignorable commentaries

All the three annotations and associations are removed.

2. Join start and end events

The start event with a message trigger is transformed into an initiate event and a message halt event as in figure 5_4.

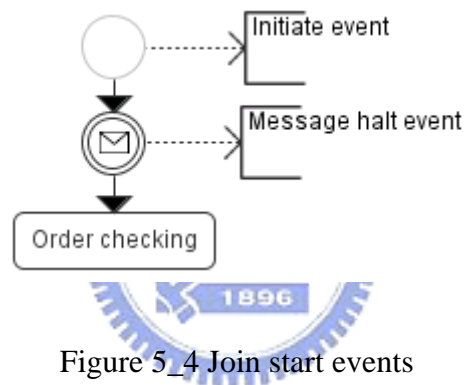


Figure 5_4 Join start events

There are three end events in the trading process, which represent "remittance confirmation failed", "trade successful completed", and "redeployment aborting". They are replaced with corresponding tasks and linked with BP Sequence Flows to the additional complete event as in figure 5_5.

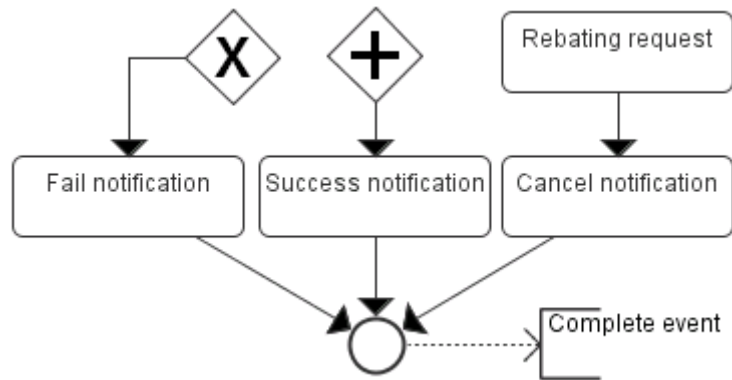


Figure 5_5 Join end events

3. Expand interrupted routines

The interrupted event associated with goods redeployment process is expanded to an event gateway as in figure 5_6.

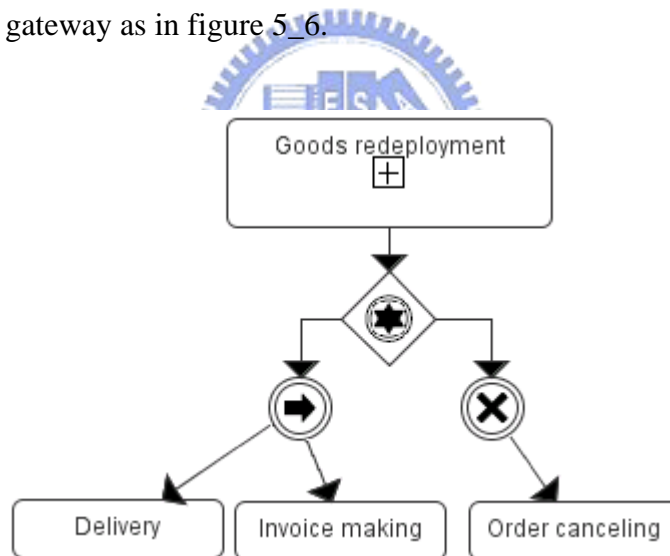


Figure 5_6 Expand interrupt routines

4. Expand gateways

The exclusive gateway, event gateway (generated in previous step), and parallel gateways are expanded as in figure 5_7 correspondingly. Note that the expansion of the event gateway generates a process variable, redeployment result, which

constrains the exclusive behavior of the branches.

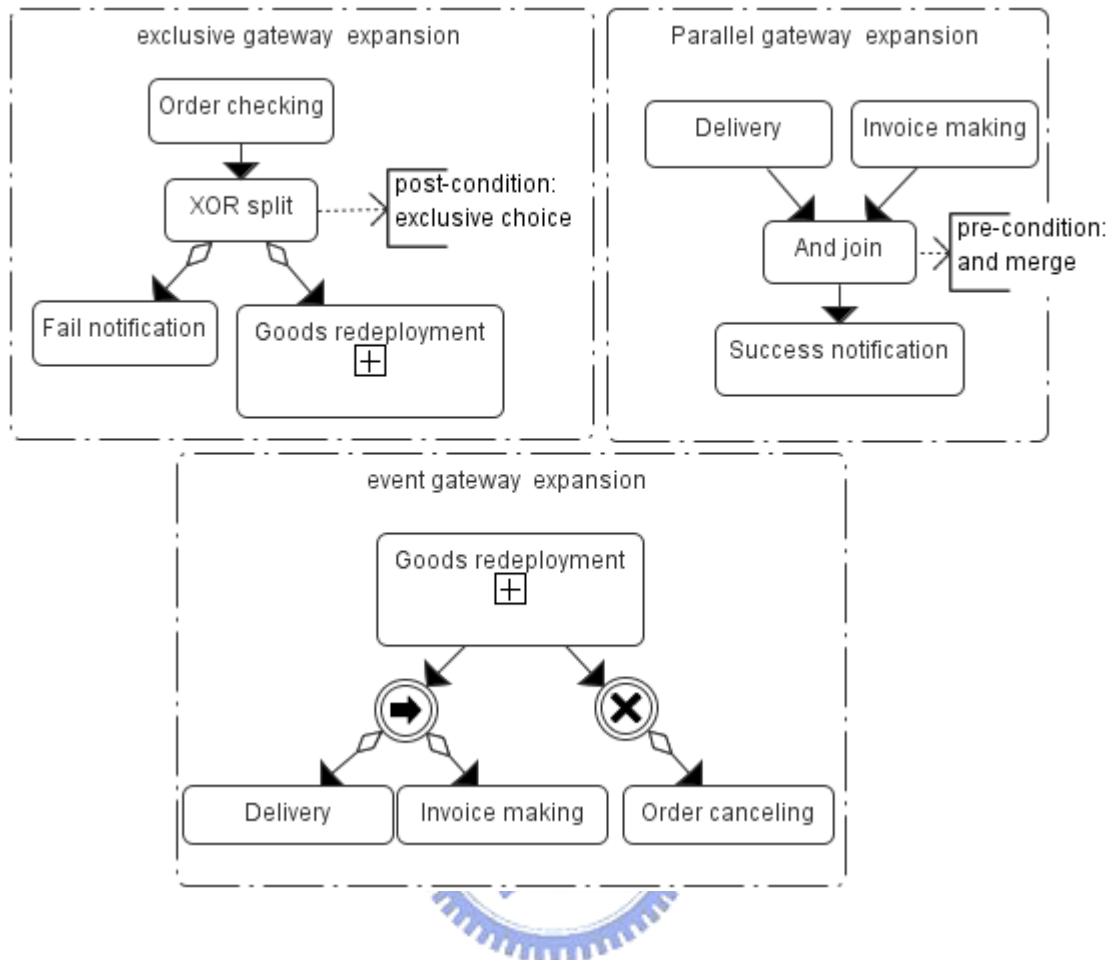


Figure 5_7 Expand gateways

5. Transform halt events

There are two halt events generated by expanding event gateway in step 3. They are transformed into two receive tasks which wait for the normal completion or cancelation trigger of goods redeployment process.

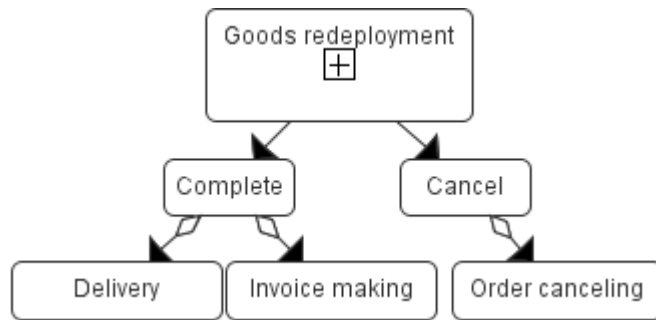
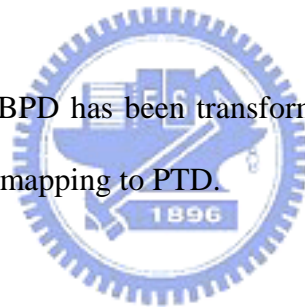


Figure 5_8 Transform halt events

6. Arrange sequence flow conditions

The condition of each BP Sequence Flow is scattered over the pre-condition of its downstream BP Activities and the post-condition of its upstream BP Activities.

After second phase, the BPD has been transformed from figure 5_2 into figure 5_9, which can be one-to-one mapping to PTD.



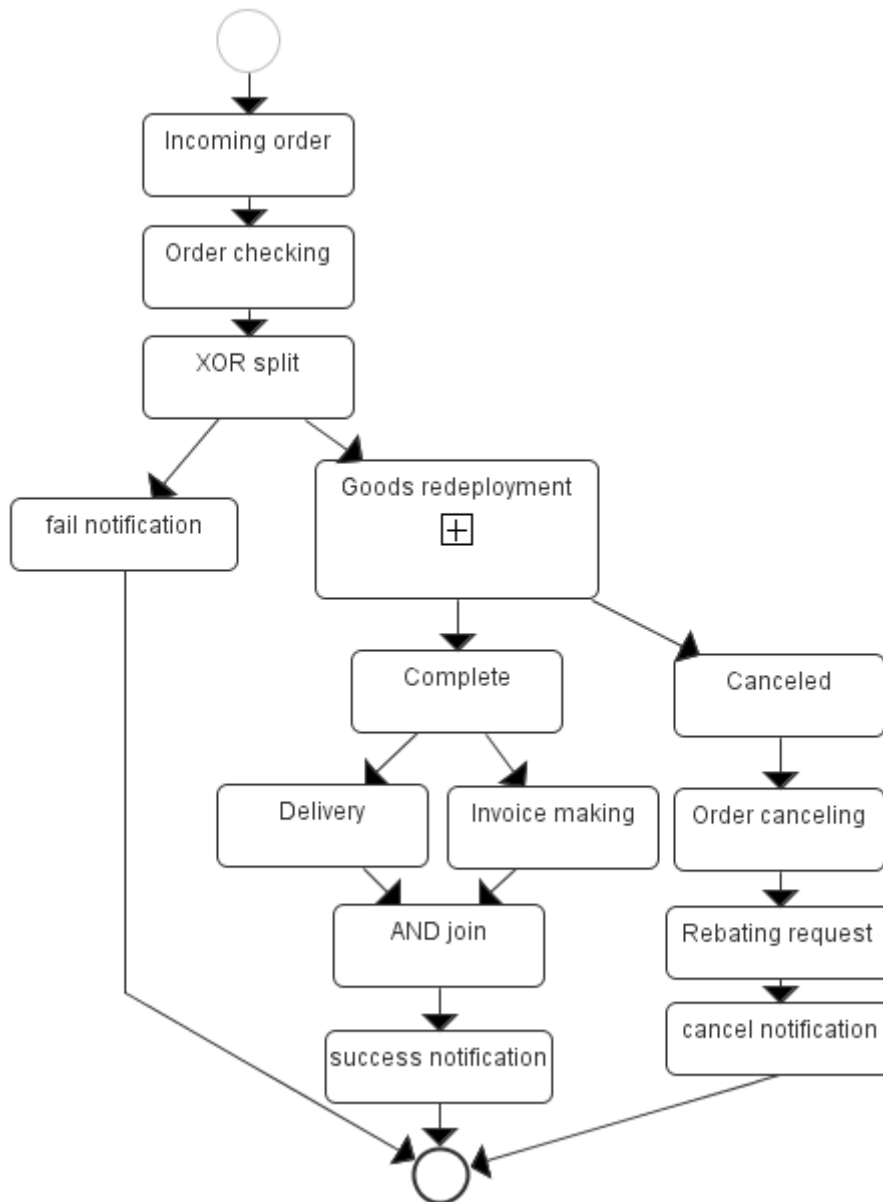


Figure 5_9 BPD after second phase

Table 5_3 BP Activity pre-conditions and post-conditions

Process (type)	Pre-condition	Post-condition
Incoming order (listen task)	None.	The order (message from the customer) has been received.

Order checking (task)	The order is received.	The order checking has been done.
XOR split (dummy task)	None.	The order checking result has been set to either true or false.
Fail notification (notify task)	The order checking result is false.	The notification has been sent.
Goods redeployment (embedded sub-process)	The order checking result is true.	The goods have been deployed. or The order is canceled.
Canceled (listen task)	None.	The redeployment result is false.
Order canceling (task)	The redeployment is canceled. and The redeployment result is false.	The order has been canceled.
Rebating request (independent sub-process)	The order is canceled.	The payment has been rebated.
Cancel notification (notify task)	None.	The notification has been sent.
Completed (listen task)	None.	The redeployment result is true.
Delivery	The goods are deployed.	The goods have been

(task)	and The redeployment result is true.	delivered.
Invoice making (task)	The goods are deployed. and The redeployment result is true.	The invoices have been made.
AND join (dummy task)	The goods are delivered. and The invoices are made.	None.
Success notification (notify task)	None.	The notification has been sent.

In the last phase, we perform a one-to-one mapping from BPMN* specification to PTD specification. This phase also creates design templates, which are not showed here.

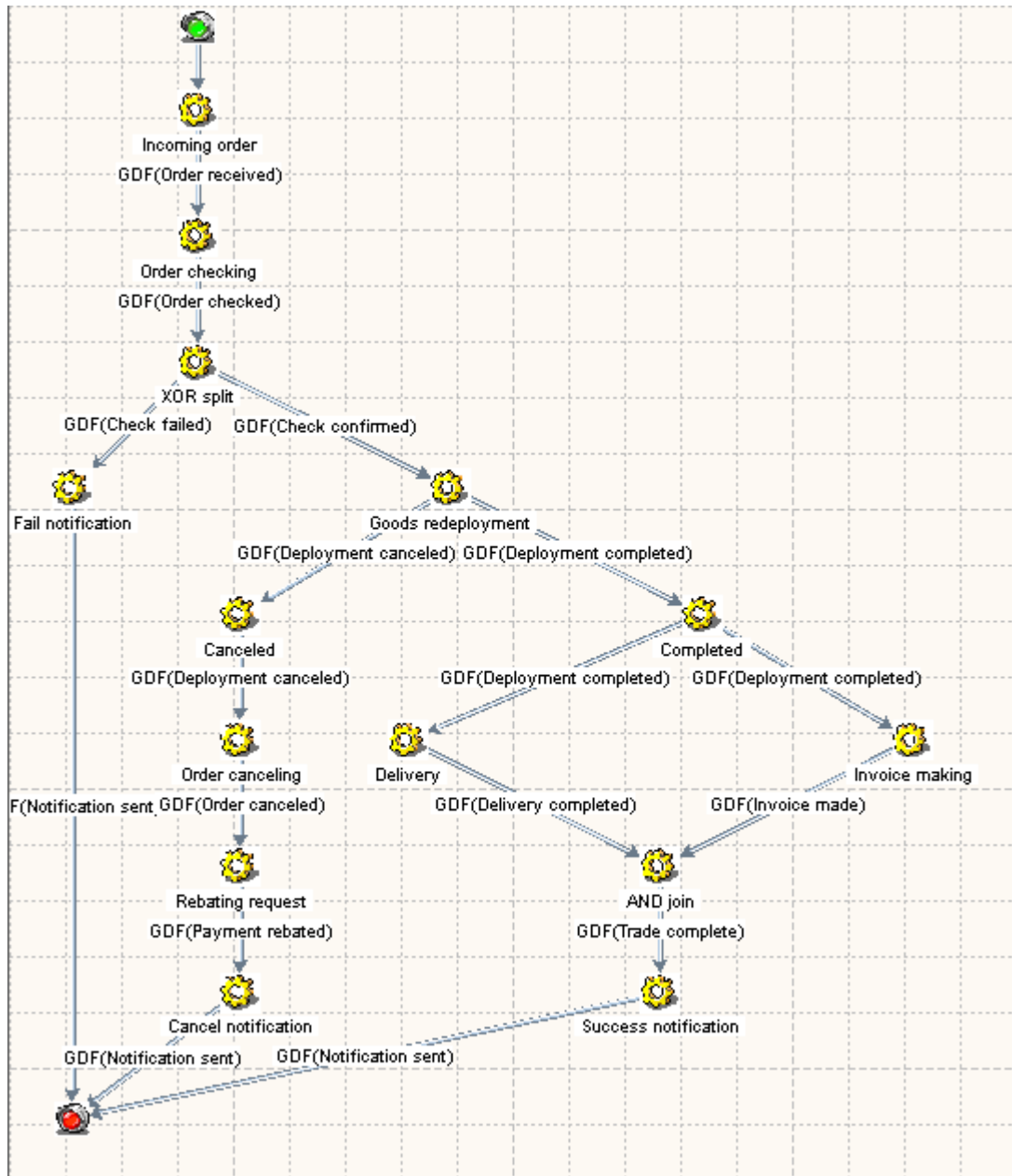


Figure 5_a Mapped PTD

Chapter 6. Conclusion and future works

This thesis presents a conceptual business process model by integrating CA-PLAN with a modified notation of BPMN, BPMN*. It also presents a methodology for business process development, and an interpretation algorithm for translating such a BPMN-PLAN specification into an Agentflow process definition. Finally, a case study is discussed to show the whole process.

The BPMN* decrease the ambiguity of the obscure notations within BPMN, and the BPMN-PLAN fulfills the deficiency of BPMN, thereby enabling designers to model practical business processes with BPMN. The development methodology then provides a systematic way to develop an executable business process specification with BPMN-PLAN for Agentflow system.

In the shortcoming, some notations of BPMN are excluded from BPMN*, which may reduce the expressive power of BPMN. For instance, the inclusive merge gateway is excluded from BPMN* for its uncertainty in design time.

Our laboratory is working on the construction of an editorial environment for BPMN*. After it is put into practice, the development of related techniques such as analyzing and testing for a BPMN* developed system will be our next topic. And since BPMN is still in evolving, BPMN* will need to be continuously studied to advance the compatibility in the future.

Reference

- [1] Object Management Group, "Business Process Modeling Notation (BPMN) Specification", Final Adopted Specification, February 2006
- [2] Flowring Technology Corp, Agentflow system, <http://www.flowring.com>
- [3] Shung-Bin Yan and Feng-Jian Wang, "A Cooperative Framework for Inter-Organizational Workflow system", In Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03), Nov. 2003
- [4] Shung-Bin Yan and Feng-Jian Wang, "CA-PLAN, an inter-organizational workflow model", International Journal of Automation and Computing 2 (2005) 195-207, September 2005
- [5] "Business Process Execution Language for Web Service", version 1.1, <http://www.ibm.com.developeworks/library/ws-bpel>, May 2004
- [6] Business Process Management Initiative, "Business Process Modeling Language", Nov 2002
- [7] Ming-Feng Chen, Bin-Shiang Liang, Feng-Jian Wang. "A Process-Centered Software Engineering Environment with Network Centric Computing", 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97), pp.234, 1997.
- [8] Yin-Shinn Chen, Feng-Jian Wang. An Editing System for Working Processes, 25th Annual International Computer Software and Applications Conference COMPSAC'01), pp. 332, 2001.
- [9] T.Muraya, "Petri Nets: Properties, analysis and applications", Proceedings of the IEEE, Vol.77, No.4,pp.541-580, 1989
- [10] K. Jensen. "Coloured Petri Nets: A High Level Language for System Design and Analysis" In G. Rozenberg, editor, Advances in Petri Nets 1990, volume 483 of Lecture Notes in Computer Science, pages 342–416. Springer-Verlag, Berlin, 1990.
- [11] K. Jensen. "Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use" Volume 1. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.
- [12] W.M.P. van der Aalst and A.H.M. ter Hofstede., "Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages" In K. Jensen, editor,

Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002), volume 560 of *DAIMI*, pages 1–20, Aarhus, Denmark, August 2002. University of Aarhus.

- [13] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow Patterns", BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000.
- [14] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Advanced Workflow Patterns", 7th International Conference on Cooperative Information Systems (CoopIS 2000), volume 1901 of Lecture Notes in Computer Science, pp.18-29. Springer-Verlag, Berlin, 2000.
- [15] Hsun-Jen Hsu and Feng-Jian Wang, "Using State Diagrams to Validate Artifact Specifications on Primitive Workflow Schema", Nov 2005
- [16] Lei Gong and Hai-yang Wang, "A method to verify the soundness of workflow control logic", Computer Supported Cooperative Work in Design, Volume 1, pp.284-388, May 2004.
- [17] Hongchen Li, Yun Yang and T.Y. Chen, "Resource constraints analysis of workflow specifications", The Journal of Systems and Software, Volume 73, Number 2, pp.271–285, Elsevier Science, October 2004.