Using Petri Nets and Component Software Technologies to Develop a Distributed Workflow System

Stephen J.H. Yang* and Chyun-Chyi Chen
*Department of Computer and Communication Engineering
*National Kaohsiung First University of Science and Technology, Taiwan
*E-mail: jhyang@ccms.nkfu.edu.tw
Department of Computer Science and Information Engineering
National Central University, Taiwan
E-mail: chyun@selab.csie.ncu.edu.tw

Abstract

The aim of this paper is to present the ability of Petri nets and component software technologies to face a kind of distributed workflow systems problems. These problems are characterized by collaborating between intra-originations and inter-origination. First, we will introduce how to modeling workflow by Petri nets. A Petri net is graphic formal method. We use Petri nets to modeling workflow that can be analysis workflow properties. Finally, we try to build a modeling architecture. This modeling architecture cites Petri nets and component software technology to modeling workflow in every organization. We called component architecture specification (CAS). The CAS model provides an effectively way to deal with large and complexity in the application of formal model. We focus on the workflow modeling by CAS.

Keywords: workflow, Petri Nets, business process, process definition, component software.

1. Introduction

In 1970's, it was a hard dream for office automation. Companies hope to reduce manpower and workload, but many other information technology movements did not achieve its promise. Many researchers make effort in office automation area, it has been advancement. Today's an enterprise or organization is facing various challenges and pressures such as market competition, globalization economics, virtual corporation, distributed sub-companies, business process reengineering, reduction of cost, and rapid development of new products and services. They need new information techniques to reduce processing time, allocate resource

efficiently, improve performance, and shorten product's time to market. New information technologies come with the tide of fashion. We can find an important information technique to solution this problem that is workflow management technique. The main purpose of a workflow management system is the support of the definition, management, execution, coordination, control, and monitor of complex business processes logic [8]. Furthermore, workflow management system supports large and heterogeneous distributed execution environments where sets of interrelated tasks can be carried out in an efficient and closely supervised fashion. Numerous application domains exits today that use workflow management techniques in their day-to-day activities for controlling their business processes. These areas include transaction in banking, flexible manufacturing, global logistics, virtual corporation, health care, enterprise resource planning, finance, electronic commerce, and so on. We have found workflow techniques become very important for enterprises and organizations.

In the early years, it is a dream to specify process of workflow into application programs in order to satisfy certain requirements of office procedures. Today's business processes are also subject to frequent changes. The range of products and services inside organization has increased, also the lifetime of products and services has decreased. With the evolution of the computer technology, workflow has experienced numbers of shifts in changes. These are not dream, thanks to the progress of communication, information and object-oriented technologies, workflow system has been able to support decentralized organizational units through graphical interfaces and a workflow engine to manage distributed tasks and resources on different locations [13]. Enterprises or organizations can obtain many benefits by using workflow technologies. For example, workflow technology improves the access of information by scattering organizational information among various users. Thus, workflow technology provides opportunities for process change. Since workflow systems force organizations to examine and define their business processes, it is the ideal time to consider business process reengineering. Workflow technology also provides opportunities for organizational change. Workflow technology can model the entire organization as business processes and provide manager information regarding organizational changes in order to operate efficiency in today's world. These problems are characterized by collaborating between intra-originations and inter-origination. We present a CAS based component for modeling systems. A component specification provides a basis for distributed, large, and complexity system that enables one to develop a reusable and redefined system. And Workflow has played an important role that provides back-end services to response front-end requirements in the age of electronic commence. Therefore, more and more venders have invested in the development of workflow products. These includes ActionWorkflow System of Action Technologies; IBM's Flow Mark; Visual WorkFlow of FileNet; InConcert produced

by Xsoft (a division of Xerox Corp); FormFlow of Delrina; Regatta of Fujitsu (currently incorporated into ICL's TeamWARE); SAP Business Workflow by SAP; HP's WorkManager; OPEN/workflow of WANG and so on [16]. At the same time, there are many research projects about workflow management systems are undergoing including Exotics of IBM Almaden Research Center; Mentor of University of Saarland, Germany; ObjectFlow of DEC; TriGflow of University of Linz, Austria; TransCoop of Technical Research Center of Finland; Meteor of the University of Georgia and so on [16]. There is a problem. These products are incompatible and no standards to enable these workflow products to cooperation. Until 1993, the Workflow Management Coalition (WFMC) was established to make efforts in development workflow standard. WFMC provides a common "Reference Model" of workflow management systems to identify workflow management system's characteristics, terminology and components, and also enables individual specifications to be developed within the context of an overall model for workflow systems [14]. Thus, all products all workflow products can achieve a level of interoperability through the use of common standard for various functions.

Most related researches of workflow could be classified into process definition modeling and analysis, activity coordinating and scheduling, workflow system architecture and design, and development methodology. Of all the workflow techniques, process definition is one of the kernel parts of workflow techniques. It defines necessary information related to business process, such as the information of starting and completing conditions, constituent tasks, rules for navigating between activities, user tasks to be undertaken, applications that may be invoked and relevant data that may need to be referenced, and the resulting process definition will be executed by the workflow management system. Thus, the integrity and accuracy of process definition will affect the result of execution. In this paper we present a Petri-nets-based formal framework for modeling workflows. Therefore, a formal specification provides a basis for formal proofs that enables one to develop a higher confidence in the correctness of the workflow.

The rest of this paper is organized as follows. We will explain what business process is in section 2. Section 3 represents how to utilize our Petri-nets-based approach to model business processes. In section 4, we propose a component architecture specification (CAS) to model workflow process and a E-commerce example how to use CAS. Section 5 concludes this paper with our future research.

2. Business Process

In workflow management systems, business process is a set of one or more procedure(s) or activity(s), which realize business objectives or policy goals such as an insurance claims process, an order process, or a loan process. In other word, a business process *B* can be

represented as a set of procedure or activity pb_1, pb_2, …, pb_n, together with their order of invocation and information flow. Even though workflow management techniques are able to reduce manual efforts and to provide enterprises with automatic environments, but these techniques may not be suitable for all business processes. Since the concept of workflow is originally used in solving management problems of business processes, it is adapted for a business process, whose activities are allocated, scheduled, routed, managed, and executed automatically. Business processes suitable for workflow management are usually characterized with properties such as automation, monitoring, repeatability, predictability, integration, and so on. In contrast, workflow management mechanism will not be suitable for business process characterized with simple, rarely used, or needs many manual works. Once we can identify business processes suitable for workflow management techniques, the next issue is to decide using which method and how to model these business processes. In the following, we will present our Petri-nets-based approach for representation control flow, information value, and temporal of business process as following in next section.

3. Defining Business Process Using Petri Nets

We used to transform business processes directly into logical steps, such as Petri Nets, event flow, state transition diagram, etc [1, 10]. In the following, we will describe how to modeling business process using our Petri-nets-based approach.

3.1 Classical Petri Nets Representation Control Flow

3.1.1 Introduction to Classical Petri Nets

A Petri net (classical Petri nets) is a particular kind of directed graph [11]. A typical Petri net has transitions, places, directed arcs, and tokens in which rectangles represent transitions and circles represent places as shown in Figure 1. A transition tj is said to be enabled if each input place pi of tj has at least w(pi,tj) tokens, where w(pi,tj) is the weight of arc from pi to tj. A net is said to be an ordinary net if all of its arcs are weighted as one. In this paper, we assume all nets are ordinary. An enabled transition may fire any time when it is enabled. The firing of a transition is instantaneous and will remove one token from each of its input places and put one token into each of its output places [11]. Due to various system modeling and analysis, Petri nets have been extended to many variations, such as time Petri nets, temporal Petri nets, Coloured Petri nets (CP-nets), predicate/transition nets (PrT-Nets), hierarchical Coloured Petri nets (hierarchical CP-nets), and stochastic Petri nets.

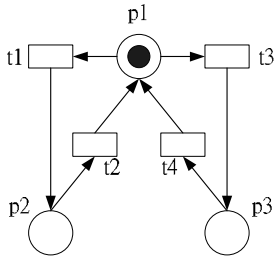

Figure 1. An example of Petri nets

Define 3.1. A Petri net is a 4-tuple PN=(P, T, F, M_0) where

(1) P is a finite set of places

(2) T is a finite set of transitions

(3) $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs

(4) M_0 is an initial marking

3.1.2 Workflow Primitives and Routings

In order to eliminate the difference between the business description and the software specification, unearthing common language understood by users and developers is imperative. Each symbol and semantic within the language must be defined clearly and intuitive for users. Petri net is a well-defined formal semantics with graphical natural. Petri net has applied in various domains, such manufacturing, industry, software engineering, and so on. Petri nets use friendly visual notations, and translate the requirements of users into software specifications more precisely. Petri nets even provide much mathematical formalism for properties analysis to software specifications. Within Petri nets modeling elements, some extended and tailored notations are suitable to represent the process definition of workflow. In the following, we will illustrate how to make use of Petri-net-based approach to specify process definition. Next, we would like to discuss what kinds of business process are suitable for workflow.

In order to capture the characters of a business process, classical Petri nets are useful to represent these. A classical Petri net can show the order of each business process behavior. Within the classical Petri nets model elements, this support to describe the dynamic behavior of business process. Whereas classical Petri nets emphasize the flow of control from business process (activity) to business process (activity). A classical Petri nets is very useful in modeling the process definition of the workflow and in describing the behavior that contains a lot of parallel processing. Each activity can be followed by another activity. This is essential for business processes.

WFMC defined six primitives to model business logical steps [14]. In this paper, we

adopt classical Petri nets to specify these six primitives because classical Petri nets support the modeling of workflow activity, transition, condition, synchronization, parallelism, iteration, etc. We specify workflow activity by means of place notation of classical Petri nets and workflow transition by means of transition and arc notation of classical Petri nets. This specifies is difference Aalst's specify. Our specify look workflow activity trigger as rule. Figure 2 shows how classical Petri nets are corresponded to the six workflow primitives defined by WFMC. AND-join primitive expresses that two or more parallel threads meet into a single thread and the synchronization bar may only be crossed to next workflow activity when all input workflow activities on the bar have been triggered. AND-split primitive expresses that a single thread split into two or more threads and the output workflow activities attached to the synchronization bar are triggered simultaneously. OR-join primitive expresses that when two or more alternative workflow activities branches re-converge into a single thread without any synchronization. OR-Split primitive expresses that when a single thread makes a decision upon which branch to take when encountered with multiple workflow branches. Branches between activities can be guarded by conditions. If guards validate, the transitions close to them are triggered to next workflow activities. Iteration primitive expresses that a workflow activity cycle involves the repetitive execution of workflow activity until a condition is met. Causality primitive expresses that two or more workflow activities are executed in a sequential form without any join or split.

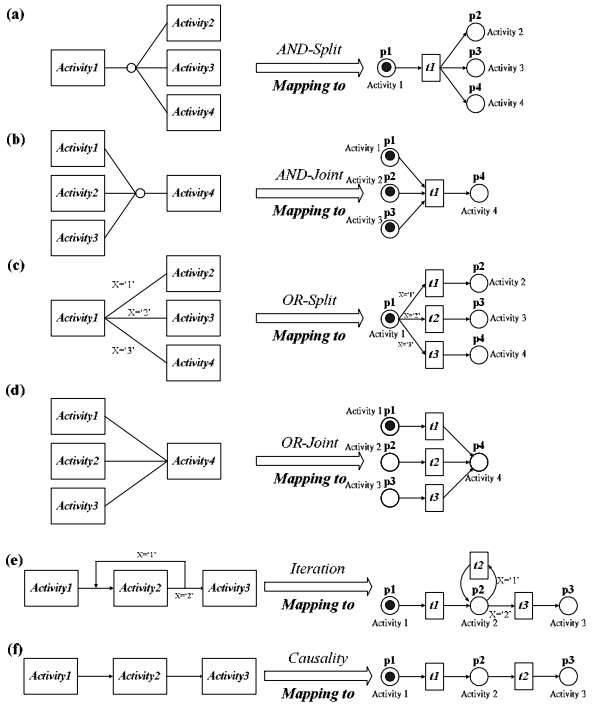

Figure 2. Workflow primitives specified by Petri nets

With the above six workflow primitives specified by Petri nets as shown in Figure 4, we can further to define four processes routing, which are sequential, conditional, parallel, and iterative routing [1, 8]. In workflow process, the four routing can be used to model any business process workflow and business process workflow can be used to model enterprise workflow. The results are show in Figure 3. Sequential routing is used to deal with causal relationships between activities. For example, three activities A, B, and C are executed sequentially. Figure 3.a shows how to use Causality workflow primitive to model sequential routing. Parallel routing is used when the ordering of activity execution is not of concern. For example, three activities A, B, and C are executed and the order of their execution is arbitrary. Figure 3.b shows how use AND-split and AND-join workflow primitives to model parallel routing. Conditional routing is used when instances need to be considered and those instances may depend on the workflow attributes. For example, in Figure 3.c one of three activities B, C and D are executed and one of execution is depend on the workflow attributes whether satisfy condition X='1', X='2' and X='3'. Figure 3.c shows how use OR-split and OR-join workflow primitives to model conditional routing. Iterative routing is used to deal with activity which need to execute one or more than one times. Figure 3.d shows how to use iteration workflow primitive to model iterative routing.

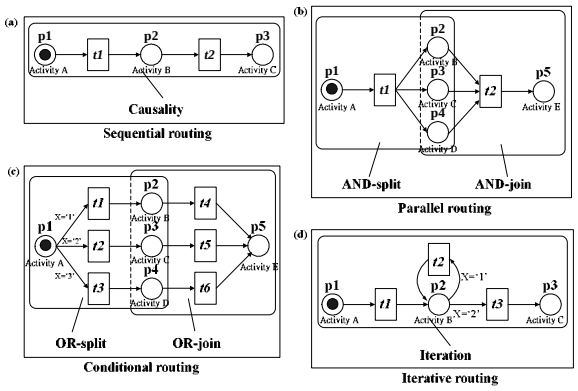

Figure 3. Process routing presented by workflow primitives

3.2 Color Petri Nets Representation of Information Value

From information aspect, Color Petri nets are useful to represent information of actors, roles, organizational units, and relevant data for business processes. These information objects can be seen as classes with relevant attributes in color token. We user color token which carry attributes to represent a workflow instances as shown in Figure 4. In Figure 6, there are two workflow instances in Loan request workflow. The color token attributes include name for apply loan, amount of loan, annual income. Case 1 of workflow instance represents William apply loan amount $3000 and William has annual income $70000. Case 2 of workflow

instance represents Jennify apply loan amount $5000 and Jennify has annual income $30000.

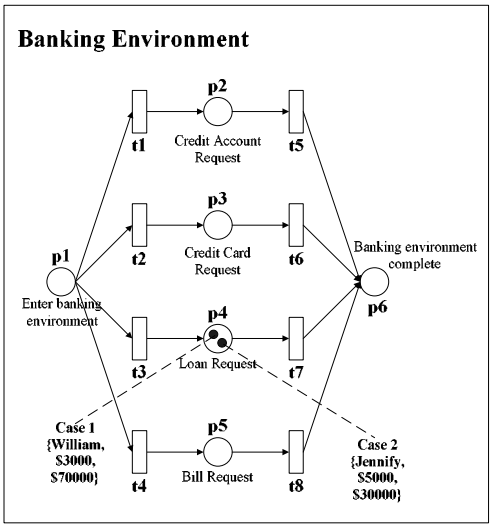

Figure 4. Workflow instances represented by color Petri Nets

Define 3.2. A color Petri net is a 5-tuple PN=(Σ, P, T, F, M_0) where

(1) Σ is a finite set of non-empty types, called color token sets

(2) P is a finite set of places

(3) T is a finite set of transitions

(4) $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs

(5) M_0 is an initial marking

3.3 Time Petri Nets Representation of Temporal

A workflow is complete or partial automation of a business process, in which participants involve in a set of activities according to certain procedural rules and constrains. The successful completion of the process often depends on the correct scheduling of the activities. Commercial workflow systems are usually rather limited in their ability to specify temporal conditions for each individual activity or for the global plan [3]. So far we have shown how to model the control flow and workflow instance using a classical Petri nets and color Petri nets. In order to timing constrains in a workflow, we need to extend Petri nets. As a simple example shown in Figure 5, consider activity A must occur within a_1 to b_1 business time, activity B must occur within a_2 to b_2 business time, activity C must occur within a_3 to b_3 business time, respectively.

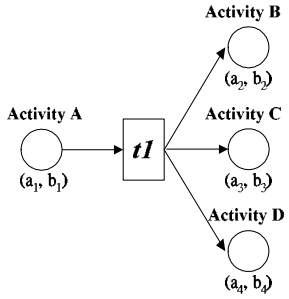

Figure 5. Workflow timing constrain represented by time Petri Nets

Define 3.3. A P-time Petri net is a 5-tuple PN=(P, T, F, M_0, SI) where

(1) P is a finite set of places

(2) T is a finite set of transitions

(3) $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs

(4) M_0 is an initial marking

(5) SI: $P \rightarrow Q^+ \times (Q^+ \times \infty)$is mapping called static interval

3.4 A Petri-nets-based approach for Business Process Modeling

Nowadays, workflow systems elements of business process including control flow, information value, and temporal. Therefore, classical Petri nets, color Petri nets, or time Petri nets not satisfy to representation business process. In order to representation business process, we extend Petri nets notation that called color-time Petri nets (CTPN). In Figure 6, there are two workflow instances in Loan request workflow. The color token attributes include name for apply loan, amount of loan, annual income. Case 1 of workflow instance represents William apply loan amount $3000 and William has annual income $70000. Case 2 of workflow instance represents Jennify apply loan amount $5000 and Jennify has annual income $30000. The Enter banking environment activity occur immediately, Credit Account Request activity occur within 3 to 9 business time, Credit Card Request activity occur within 2 to 4 business time, Loan Request activity occur within 1 to 4 business time, Bill Request occur within 2 to 6 business time, Banking environment complete activity occur immediately, respectively.

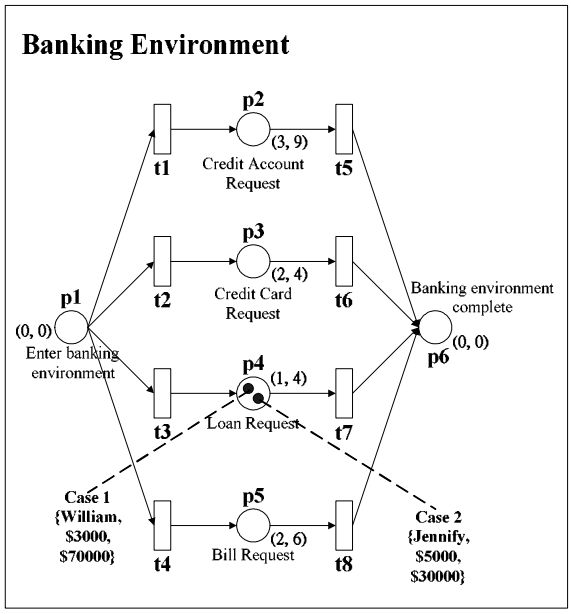

Figure 6. Business process by color-time Petri nets

Define 3.4. A color-time Petri net is a 6-tuple PN=(Σ, P, T, F, M_0, SI) where

(1) Σ is a finite set of non-empty types, called color token sets

(2) P is a finite set of places

(3) T is a finite set of transitions

(4) $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs

(5) M_0 is an initial marking

(6) SI: $P \rightarrow Q^{+} \times (Q^{+} \times \infty)$is mapping called static interval

4 Component Architecture Specification

In this section, we first give a conceptual of component architecture specification (CAS) model. Then we formalize the CAS notation. Finally, we give an E-commerce illustration to how use CAS to model.

4.1 Conceptual of CAS

Our goal is to develop both a rigorous approach to enhance the integrity of design and an evolutionary process to control complexity in system modeling and analysis. In recent year, information systems modeling become large, complexity, and distributed. In order to solve these problems, object-oriented and component-oriented styles of software development have addressed. Related to our approach, several structural Petri net models are proposed both to provide a mechanism for system composition and to manage complexity in modeling. These

include PROTOB [4], OBJSA nets [6], the Cooperative Objects Language [5], OPNets [9], HOOD nets [7], CmTPN [15] and C-net [2].

Our work has its basis from two research areas: color-time Petri nets and component software architecture. The first provide the formal basis for the notational and semantics system of CAS; and the second provides CAS with its conceptual basis. It is an emerging field with promising solutions for dealing with the rapidly changing requirements of workflow software applications. We use CAS to model a distributed system as a multi-leveled composition of components and their compositions must satisfy at every design level. We can redefine components in CAS sub-level. More specifically, a CAS model consists of two basic elements: components models, and inter-component connections. The component models describe the behavior and communication interface of the components. The inter-component connections specify how the components interact with each other. This way, the architecture can be viewed as a template for system composition, that is, sub-architecture of a component can be plugged-in the place of the corresponding specification to form an instantiated, recursively defined multi-level architecture model. We formalize the CAS notation as shown in Define 4.1.

A simple CAS model is illustrated in Figure 7. The high-level design has four components A, B, C and D. In a CAS model, a component (e.g. A) specification is described as a color-time Petri Nets. It consists of two parts: communication ports (denoted graphically by black rectangle) that describe features provided (input ports, e.g. PORT1) and required (output ports, e.g. PORT2 and PORT3) by A; a net that describes the time-dependent, operational behavior of A, that is, it defines the semantics associated with the ports. The communication between A is solely through the ports. A communication represents a channel of interaction between components. An interface specification allows any sub-architecture that conforms to the component specification to be plugged into the place of the specification to form a multi-level and more detailed or refined system architecture. The component D can be further refined at the next design level into the composition of components D1, D2, D3 and D4. This arrangement is critical to achieve the goals of incremental modeling described early. The component designs can be treated as block-boxes in construction, and understanding of a system’s architecture. As more detailed system architecture is constructed by decomposing one of the system’s components. Sub-architecture of component guarantees the satisfaction of system-wide requirements. Thus, each of these components will be subject to their own derived constrains.

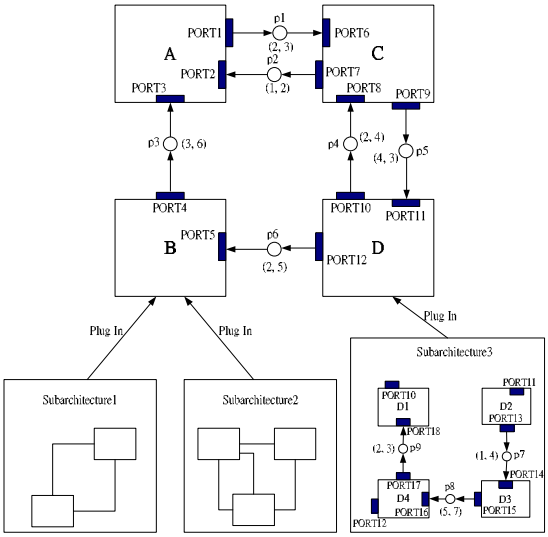

Figure 7. The modeling framework of CAS

4.2 An Overview of CAS Notation

In this section, we present the notation of CAS by the formal notation of color-time Petri Nets into above framework to form an integrated architecture model.

Define 4.1. A component architecture specification is a 2-tuple CAS=(C, H) where

(1) C is a composition may correspond to a design level or the concept of sub-architecture, C=(C_1, C_2, …, C_i) and C_j=$\{C_{omp}, C_{onn}\}$ for j=1, 2, …, i, where

(a) C_{omp} is a set of components. $\forall C_{omp_i} \in C_{omp}$ is defined by CTPN. Let

$C_{omp_i}.PORTIN = \{t | t \in C_{omp_i}.T, \bullet t \cap C_{omp_i}.P = \phi\}$ is called the set of input ports of component C_{omp_i};

$C_{omp_i}.PORTOUT = \{t | t \in C_{omp_i}.T, t \bullet \cap C_{omp_i}.P = \phi\}$ is called the set of output ports of component C_{omp_i};

$C_{omp_i}.PORT = C_{omp_i}.PORTIN \cup C_{omp_i}.PORTOUT$ is called the set of ports of C_{omp_i};

Moreover, $\forall C_{omp_i}, C_{omp_j} \in C_{omp}$, $C_{omp_i}.P \cap C_{omp_j}.P = \phi$ $C_{omp_i}.T \cap C_{omp_j}.T = \phi$.

(b) C_{onn}=(P, T, F, M_0, SI) is connection, where

$C_{onn}.T \cap [\bigcup_{C_{omp_i} \in C_{omp}} C_{omp_i}.T \setminus C_{omp_i}.PORT] = \phi$;

$C_{onn}.P \cap [\bigcup_{C_{omp_i} \in C_{omp}} C_{omp_i}.P]$;

In composition C_j,

$$C_j.T \cap [\bigcup_{C_{omp_i} \in C_{omp}} C_{omp_i}.T] = C_{onn}.T;$$

$$C_j.P \cap [\bigcup_{C_{omp_i} \in C_{omp}} C_{omp_i}.P] = C_{onn}.P;$$

$$C_j.PORT = \bigcup_{C_{omp_i} \in C_{omp}} C_{omp_i}.PORT;$$

$$C_{omp_i}.EXTPORT = \{t | t \in C_j.PORT, \bullet T \cap C_j.P = \phi \vee T \bullet \cap C_j.P = \phi\};$$

(2) H: $C_{omp_i} \to C_j$, $i \neq j$, is a hierarchical mapping function, where $\forall C_j \in C$ and $\forall C_{omp_l} \in C_j.C_{omp}$,

$C_{omp_l}.PORT = C_j.EXTPORT$

For example, in the CAS model shown in Figure 7, C = (C_1, C_2),
$C_1.C_{omp}$ = {A, B, C, D}, $C_2.C_{omp}$ = {D1, D2, D3, D4}, H($C_1.C_{omp}$.D) = C_2,
$C_1.PORT$ = {PORT1, PORT2, …, PORT12},
$C_1.PORTIN$ = {PORT2, PORT3, PORT5, PORT6, PORT8, PORT11},
$C_1.PORTOUT$ = {PORT1, PORT4, PORT7, PORT9, PORT12},
$C_1.C_{omp}$.A.$PORT$ = {PORT1, PORT2, PORT3},
$C_1.C_{omp}$.A.$PORTIN$ = {PORT2, PORT3},
$C_1.C_{omp}$.A.$PORTOUT$ = {PORT1}
…. …. ….
$C_1.C_{omp}$.D.$PORT$ = {PORT10, PORT11, PORT12},
$C_1.C_{omp}$.D.$PORTIN$ = {PORT11},
$C_1.C_{omp}$.D.$PORTOUT$ = {PORT12, PORT13}
$C_2.PORT$ = {PORT10, PORT11, …, PORT18},
$C_2.PORTIN$ = {PORT11, PORT14, PORT16, PORT18},
$C_2.PORTOUT$ = {PORT10, PORT12, PORT13, PORT15, PORT17},
$C_2.EXTPORT$ = {PORT10, PORT11, PORT12} = $C_1.C_{omp}$.D.$PORT$,
$C_1.C_{omp}$.D1.$PORT$ = {PORT10, PORT18},
$C_1.C_{omp}$.D1.$PORTIN$ = {PORT18},
$C_1.C_{omp}$.D1.$PORTOUT$ = {PORT10},
…. …. ….
$C_1.C_{omp}$.D4.$PORT$ = {PORT12, PORT16, PORT17},
$C_1.C_{omp}$.D4.$PORTIN$ = {PORT16},
$C_1.C_{omp}$.D4.$PORTOUT$ = {PORT12, PORT17}

4.3 E-commerce illustration for CAS

We consider a workflow Scenario which involves four business partners: a customer, a producer and two suppliers. The customer orders a produce by sending an order for produce *a* to the producer. To produce the ordered product, the producer orders the products needed for production (*b* and *c*). Then the customer is informed that the order has been accepted. Supplier 1 produces products of type *b*, Supplier 2 produces products of type *c*. After both products have been delivered, they are assembled into a product of type *a* which is delivered to the customer. After delivery an invoice is sent which is then paid by the customer. Figure 8 models the interaction between the four business partners.

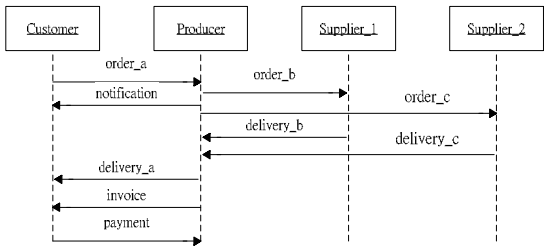

Figure 8. The interaction of E-commerce example

According above describe E-commerce, we use CAS model to construct this. Every partner in E-commerce example has its own private workflow, there is no need to agree upon one common process. Using CAS model to capture the profile of the operational model of E-commerce, we first present its CAS architecture in Figure 9, where every partner process represent a component. There are four business partners including a customer, a producer, and two suppliers. Therefore, top-level of CAS model is including four components for example of E-commerce. Between the four business partners have interaction. Between a customer partner and a producer partner have five interactions, such as order_a, notification, delivery_a, invoice, and payment. Between a producer partner and a supplier_1 partner have two interactions, such as order_b, and delivery_b. Between a producer partner and a supplier_2 partner have two interactions, such as order_c, and delivery_c. Therefore, between each component has communication connected. We model those interactions as communication linkages in CAS model. In this step, we have been modeled abstract view of E-commerce example by CAS model.

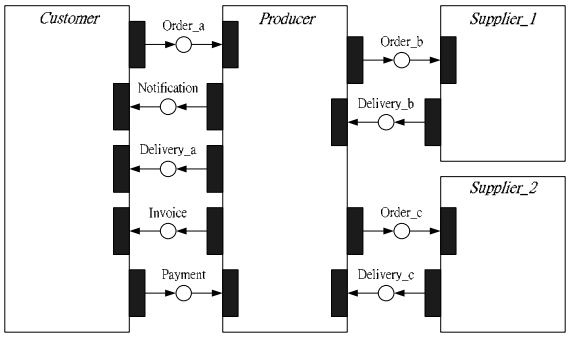

Figure 9. CAS architecture of the E-commerce example

We have been modeled E-commerce environment, but this model level not defined every partner operational behavior. In next step, we use Petri Nets to define the operational behavior of each component. Using the Petri net design methodologies, we use Petri Net model for a component based on its operations relationship among these operations. Figure 10 shows operational behavior of component Supplier_1. We further decompose the specification of Supplier_1 partner into a more detailed sub-architecture shown below. The supplier_1 partner will be processed following workflow processes. First, supplier_1 partner create a require task in Supplier_1_require process and receive order_b from producer partner, then supplier_1 partner transfer task into produce task in produce_b process. The supplier_1 partner must be check produce in check_produce_b workflow after produce_b process. If produce is enough that will transfer process into produce_b_OK process, other will transfer process into produce_b_NOK process. The supplier_1 partner will send delivery in send_del_b process after produce_b_OK process. Other partners can also decompose the specification into a more detailed sub-architecture. Figure 11 shows detail describe of E-commerce example by CAS.

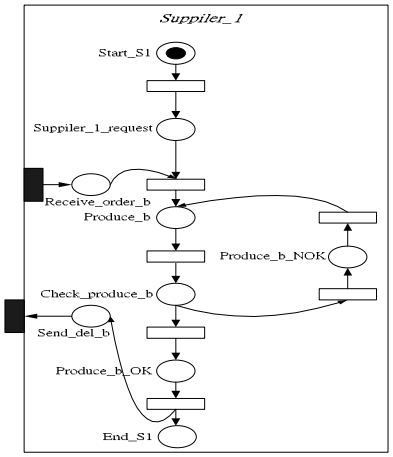

Figure 10. The operational model of the Supplier_1

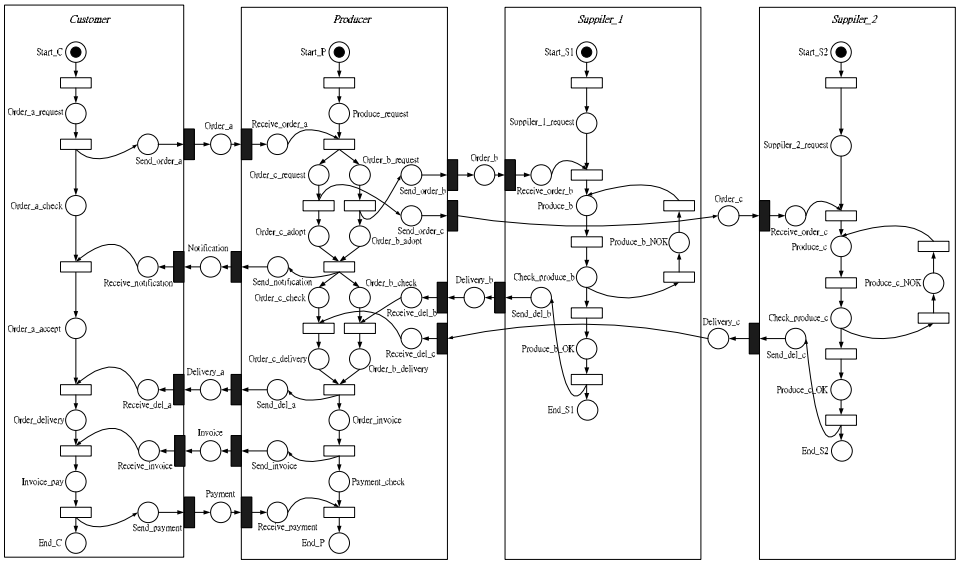

Figure 11. Detail describe of E-commerce example By CAS

5. Conclusions and Future Research

In this paper, we have presented a Petri nets approach to model business processes. Based on the requirement of the modeling and design of workflow process, we have presented a CAS-based incremental approach to architectural modeling, and illustrated the use of the approach to incrementally model a given E-commerce. We also proposed a deployment diagram to represent physical configuration of distributed activities. We will continue to develop workflow management system in our future research

References

[1] W.M.P van der Aalst, “Three Good Reasons for Using a Petri-net-based Workflow Management System,“ *Proc. Of the International Working Conference on Information and Process Integration in Enterprises (IPIC’96). Eds. S. Navathe and T. Wakayama, Camebridge, Massachusetts*, pp. 179-201, Nov. 1996.

[2] W.M.P. van der Aalst, K.M. van Hee and R.A. van der Toorn, “Component-Based Software Architectures: A Framework Based on Inheritance of Behavior,” *Science of Computer Programming*, vol. 42, no. 2-3, pp. 129-171, 2002.

[3] G. Alonso, D. Agrawal, A.E. Abbadi and C. Mohan, “Functionalities and Limitations of Current Workflow Management Systems,” *Research Report*, IBM Almaden Research Center, 1997.

[4] M. Baldasssri and G. Bruno, “PROTOB: An Object Methodology for Developing Discrete Event Dynamic System,” *High-Level Petri Nets: Theory and Application*, 1991.

[5] R. Bastide, C. Blanc and P. Palanque, “Cooperative Objects: A Concurrent Petri-net Based,

Objected-oriented Language," *Proc. of IEEE Int. Conf. on System, Man and Cybernetics*, vol. 3, pp. 286-291, 1993.

[6] E. Battiston, F. Cindio and G. Mauri, "OBJSA Nets: A Class of High-Level Nets having Objects as Domains," *Advances in Petri Nets, Lecture Notes on Computer Science*, CS 340, 1988.

[7] R.D. Giovani, "Petri Nets and Software Engineering: HOOD Nets," *Proc. of the 11th Int. Conf. on Application and Theory of Petri Nets*, pp. 123-138, June 1990.

[8] P. Lawrence, Workflow Management Coalition, "Workflow Handbook 1997," *Wiley and Sons Ltd, New York*, 1997.

[9] Y. Lee and S. Park, "OPNets: An Objects-Oriented High-Level Petri Net Model for Real-Time Systems," *The Journal of Systems and Software*, vol. 20, no. 1, pp. 69-86, 1993.

[10] Y. Lei and M.P. Singh, "A Comparison of Workflow Metamodels," http://osm7.cs.byu.edu/ER97/workshop4/ls.html, 1997.

[11] T. Murata, "Petri Nets: Properties, analysis and applications," in *Proceeding of The IEEE*, vol. 77, no. 4, pp. 541-580, 1989.

[12] Rational, and UML partners, "UML Summary version 1.1," http://www.rational.com/uml/resources/documentation/summary/index.jtmpl, 1997.

[13] J. Veijalanen, A. Lehtola, and O. Pihlajamaa, "Research Issues in Workflow Systems," October 2,1995.

[14] Workflow Management Coalition, "Workflow Management Coalition The Workflow Reference Model," Nov 1994.

[15] G. Bucci and E. Vicario, "Compositional Validation of Time-Critical Systems Using Communicating Time Petri Nets," *IEEE Trans. on Software Engineering*, vol. 21, no. 12, Dec 1995.

[16] S.J.H. Yang and Chyun-Chyi Chen, "A Petri-Nets-Based Approach for Workflow and Process Automation," *International Journal on Artificial Intelligence Tools*, vol. 8, no. 2, pp. 193-205, 1999.