**RC23829 (C0512-004) December 14, 2005**
**Computer Science**

# IBM Research Report

## Aspect-Oriented Business Process Modeling

**Jian Wang, Jun Zhu, Haiqi Liang**
IBM Research Division
China Research Laboratory
HaoHai Building, No. 7, 5th Street
ShangDi, Beijing 100085
China

**Ke Xu**
Tsinghua University

**Research Division**
**Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich**

# Aspect Oriented Business Process Modeling

Jian WANG, Jun ZHU, Haiqi LIANG, Ke XU
*IBM China Research Lab, Tsinghua University*
*{wangwj, zhujun, lianghq}@cn.ibm.com, xk02@mails.tsinghua.edu.cn*

## Abstract

*Model-driven approach has been regarded by many researchers and practitioners as an efficient and practical way for well handling large scale and complex IT systems. However, model composition and decomposition as an existing way to reduce model complexity isn't enough to fully deal with large scale and complex cases. We need a dynamic model componentization structure to support the flexible composition, decomposition and refactoring of models. This paper proposes an approach to aspect-oriented business process modeling, which introduces a set of dynamic model componentization technologies to business process modeling practices. The following key contributions of this work are introduced in detail: the concept and definition of aspectual process as the base for dynamically structuring business process model; an end-to-end aspect-oriented business process modeling method covering key phases including aspect definition, identification, extraction, assembling and weaving and key technologies to support the above method. A prototype system, together with an end-to-end scenario as sample is also presented.*

## 1. Introduction

With the needs for global resource and information consolidation and the continuous advancing of information technologies, today's business is now asking for more and more support from IT systems. As a result, IT systems are growing larger and larger, and introducing more and more functions to cover enterprise business execution from various aspects, i.e. operating, producing, planning, controlling, auditing and so on. IT system is usually built with hundreds thousands of codes, and inevitably causing a lot of problems in building, changing, maintaining and reusing them.

Model-driven approach has been regarded by many researchers and practitioners as the future way for well handling large scale and complex IT systems. MDD [1] (Model-Driven Development), proposed by OMG on 2001, has been widely studied and practiced in many cases, and proved itself to be a promisingly effective approach, especially in the following perspectives: formalized business semantics to achieve mutual understanding across teams, backgrounds and scenarios; automatic model transformation across layered models to facilitate job distribution and role-based development; formally captured inter-model relationship to facilitate the change, maintenance and reapplying of target system after it is delivered.

However, with MDD gradually got widely adopted, model composition and decomposition as an existing way to reduce model complexity isn't enough to fully deal with large scale and complex cases. Today's modeling methods rely on a formalized meta-model (i.e., MOF [2] for UML [3]) to organize and present modeling elements within a model. The model organization structure, usually specified by the meta-model in a fixed way, thus becomes the dominant model componentization scheme and inevitably determines the way in which the model is understood, created, manipulated and reused. This fixed model componentization scheme has generated the following problems:

1) IT system normally includes the concerns or requirements from different stakeholders. But the modeling elements to represent them, in existing modeling methods, are usually contained in one single model in a crosscutting way (as illustrated in Fig. 1). Taking security, auditing, compliance, logging, and recovery in business models into consideration are necessary for building a real executable IT solution, whereas such a comprehensive model in an auditor's view will be too hard to understand because so many irrelevant things are added. One subsequent problem of crosscutting modeling elements is that it makes the modification of existing system difficult to handle and control, because every minor change has to be made on top of a complete model together with many other intricate aspects.

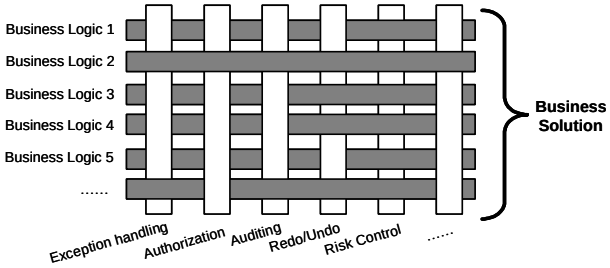


**Fig. 1 multiple concerns scattering in business processes make process models complex.**

2) In the lifecycle of building model-driven solutions, models are widely used as the vehicle to convey project participants' ideas and working results. The fixed

componentization structure of the models, from this point, also brings challenges to the work distribution and team collaboration among project members; models have to be created, exchanged and shared under this structure.

3) There are many common features (such as exception handling and auditing) in each solution that ought to be represented separately but cannot be supported by current model driven approach. These features are usually associated with main logics and features in a crosscutting way. They cannot be represented as a single model unit in most existing model scheme. This results in redundancy and overlapping of similar features across different solution functions.

4）One key advantage of model driven approach is that the resulting models, as formal representations of business knowledge or solution design, are more likely to be understood and reused in future scenarios. Since the current reuse at model level are largely restricted by the fixed asset model structure, reusable parts in models cannot be easily refactored and extracted to make them tailored for use in other situations.

The key reason for all above problems is that we are greatly constrained by the fixed model componentization architecture defined in the meta-model. To fully address the above challenges, we need a dynamic model componentization structure to support the flexible composition, decomposition and refactoring of models. In the programming domain, the same problem also exists and there have been quite some efforts on addressing this problem, among which aspect-oriented programming [4] or separation-of-concern [5,6] technologies are the most related ones. AspectJ [7] builds itself on top of Java programming language to support the specification of common programming functions (e.g. logging); CME [8] (Concern Manipulation Environment) provides an integrated framework to support the extraction and weaving of code level concerns in different stages of software development. These technologies are proved to be effective in reducing code level complexities, and hence triggering the research on resolving the complexity problem in business process modeling domain through this concern separation concept.

There have been several related works, such as AO4BPEL[9], that focus on migrating existing technology from executable code level to executable process script. However in business modeling domain, things are quite different: model plays a more important role in building business solutions than in software development lifecycle. More focuses should be laid on higher level models to attack the complexity raised by concerns from multiple stakeholders. Model simplification technologies should be combined with end-to-end model-driven lifecycle to make sure that they are truly consumable, and adhere to the solving of real business problems, as listed in(but surely not limited to) the following scenarios:

- Refactoring an existing model from another viewpoint, so that the model componentization structure is tailored for a specific usage scenario.
- Extract common features as a separated unit and weaving them with main business logic during the solution development process.
- Extracting a solution pattern from existing solution implementation as a reusable asset, and then reuse it in future cases by weaving it with other functional parts.
- Flexibly making changes to existing solutions, removing existing features and replacing them with newly developed ones.
- Building a new feature and then plugging the feature to main model by weaving.

In this paper, we will provide detailed description to one research work in IBM China Research Laboratory (CRL) aiming at addressing the fixed model componentization challenge. Our efforts mainly focus on business process modeling by introducing a set of dynamic model componentization technologies. The key contribution of this paper includes: introducing the concept and definition of aspectual process as the base for dynamically structuring business process models; proposing an end-to-end aspect-oriented business process modeling method which covers key phases including aspect definition, identification, assembling, and weaving; some initial theoretical work done as the base for correct aspect-oriented business process decomposition and composition. Above technologies are implemented in a prove-of-concept prototype system on top of CME and WSU (WBI Service Utility [10], a.k.a. Model Blue, a lightweight business process modeling tool developed by IBM CRL), and based on the prototype we implemented the four business scenarios as described above.

The rest part of this paper is organized as follows. The key concepts we used (including aspectual process) are introduced in section two, followed by the end-to-end aspect oriented business process modeling approach described in steps in section three. We then introduce in section four the key enabling technologies for our approach, i.e., model query language, extraction and weaving method. The prototype system, together with an end-to-end scenario as sample, is described in section five. Section six concludes this paper with summary and possible future work.

## 2. Basic Concepts

The approach of business process modeling proposed in this paper is based on the concept of *aspectual process*, which is an extension of the general concept of *aspect* and *concern* in aspect-oriented programming (AOP) area.

### 2.1. Concern and Aspect

In general, a concern is what users may be interested in. In programming domain, a concern is a problem that a program tries to solve. Programmers write programs in order to address one or more core concerns (such as credit card billing, or sending email), whereas cross-cutting concerns form aspects of a program that do not relate to the core concerns directly, but which proper program execution nevertheless requires.

Separation of concerns (SOC) forms an important goal in program design. When programmers simply insert calls to cross-cutting concerns (such as logging, object persistence, etc.) in the source code when needed, the resulted program leads to a highly-coupled system and is difficult to change, because every time programmers change a feature of these cross-cutting concerns, they may need to recompile a lot of source files and check a lot of calls for consistency. And every time programmers change the signature of an operation, they have to change all calls to that operation, again touching many separate source files. Cross-cutting concerns are those parts, or aspects, of the program that in standard design mechanisms end up being scattered across multiple program modules, and tangled with other modules. Isolating these cross-cutting concerns is the essential idea in Aspect-Oriented Programming (AOP). AOP allows the programmer to create cross-cutting concerns as first-class program modules.

### 2.2. Aspectual Process and Its Visualization

Extending the concept of aspect and concern, this paper introduces a new concept of aspectual process. An aspectual process is basically a kind of representation of business process from certain specific concern perspective. It can be a piece of business process, a group of business process fragment or even a complete business process. A business process typically fulfills specific business requirements by integrating functions of different business units and resources, while an aspectual process clearly defines a specific view of a business process. For example, an aspectual process can represent how a certain resource is consumed solely, or how a crosscutting business function such as auditing is done, or what a single employee does in the overall process. Each aspectual process depicts a facet of a business process. A global view of the business process can be obtained by weaving the aspectual processes together. For example, the following shows some sample aspectual processes in a general '*Purchase Order Handling*' business process:

- The aspect that depicts activities of the financial department;
- The aspect that depicts activities of the manager in the financial department;
- The aspect that depicts how a certain resource (such as printer) is consumed in the purchase order handling process;
- The aspect that depicts how a certain standard (e.g. ISO9000) is ensured in order handling.

Different from aspects in programming, an aspectual process is basically a business process or a fragment of business process. It needs to be modeled in a visualized way just like a business process model. As an example illustrated in Fig.2, we design an intuitive presentation for aspectual processes. In this visualization, an aspectual process is considered as a self-contained meaningful process. Aside from normal modeling elements in a generic business process model, an aspectual process also includes a boundary box with the aspect 's name and multiple *losing/gaining control* (captioned with *L* and *G* respectively) nodes. The boundary box encapsulates all modeling elements in the aspectual process. On the boundary, several interaction points can be modeled as aspect's interfaces interacting with other processes or aspectual processes. The semantics of "*losing control* " nodes is "releasing a control token to other aspects through interaction point, and then executing its following nodes". The semantics of "*gaining control*" is "waiting an incoming control token, and executing its following nodes once it is received". In Fig.2, the dashed lines stand for the directed control token flows.

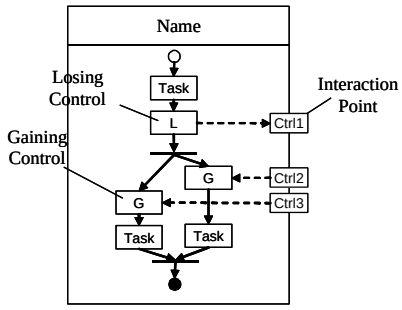


**Fig. 2 Visualization of Aspectual Process**

## 3. AOBPM Approach Overview

Based on the concept of aspectual process, we propose an aspect-oriented business process modeling (AOBPM) approach targeting at manipulating and componentizing business process model by leveraging aspect technology. The AOBPM approach is illustrated in Fig.3 as a state chart.

There are the following key actions in the approach: aspect definition, aspect identification from existing process models, aspect extraction as a self-contained process, aspect assembling, aspect weaving, aspect creation and aspect modification. These actions make aspects transit from one aspect state to another. For different usage scenarios, the performance path can be different. For example, to separate/delete an aspect from an existing process, actions are performed along the following path: aspect definition **à** aspect identification

à aspect extraction; to add a new feature to an existing process, we need: aspect definition à aspect creation à aspect assembling à aspect weaving; to refactor an existing process from a concern point, we need: aspect definition à aspect identification à aspect extraction à (aspect modification) à aspect assembling; to modify and reuse an aspect, we need: aspect modification à aspect assembling à aspect weaving. The following subsections will explain several main steps in AOBPM approach in detail.

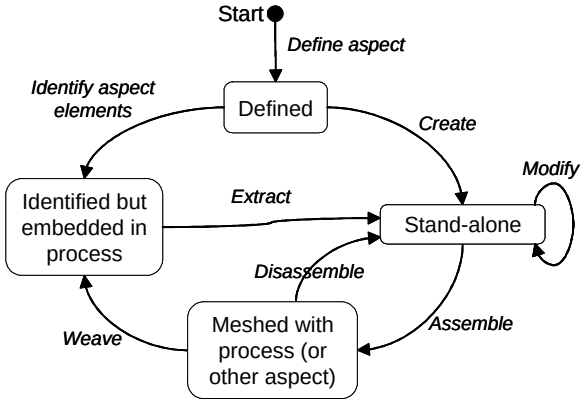


**Fig. 3 AOBPM Approach**

### 3.1 Aspect Definition

Most AOBPM usage scenarios start from aspect definition. The definition of an aspect is critical to aspect element identification, extraction and reuse. In this step, AOBPM users ought to carefully think what their interests are, and they should be able to articulate what the aspect is and what its scope and granularity is in a formal specification. It is mainly a human thinking action.

AOBPM allows user to define aspects at different granularities. The definition of aspects in fine granularity should be concrete and specific enough to be directly describable with the concepts or terms in business process models. It enables aspect element identification simply via query statements or manual selections. For example, an aspect can be defined by all tasks performed by a specific role (e.g. auditing department). For aspects in coarse granularity, they don't need to be described with the concepts in process model, but they must be finally defined with other fine granularity aspects. For example, an aspect can also be defined as all tasks that make the business process compliant with ISO9001. Its definition, however, should be further refined with the definition of other aspects.

### 3.2. Aspect Identification

A crosscutting aspect scatters in process model. If we need to extract an aspect from existing process models after aspect definition, all model elements covered by the aspect definition should be pinpointed. This procedure is called aspect identification. There are two approaches to identify related model elements: automatic selection by query engine and one-by-one selection by user. A flexible and powerful query mechanism is critical to make AOBPM approach practical and consumable. Section 4.1 and 4.2 will present our query mechanism and semantic-based query extension. For elements that can not be searched out due to the lack of semantics and clear definition, they have to be selected manually.

### 3.3. Aspectual Process Extraction

In identification step, all the corresponding modeling elements are identified. Aspectual process extraction will repack all identified aspect elements in a consistent or self-contained view so that it can be understood and reused. "losing/gaining control " nodes and interactions points will be added into the original and extracted business processes so as to maintain their completeness. The extraction result is an "aspectual process " with internal structure (indicating its internal behavior) and external interaction points (indicating how it interacts with its external environment).

### 3.4. Aspectual Process Assembling

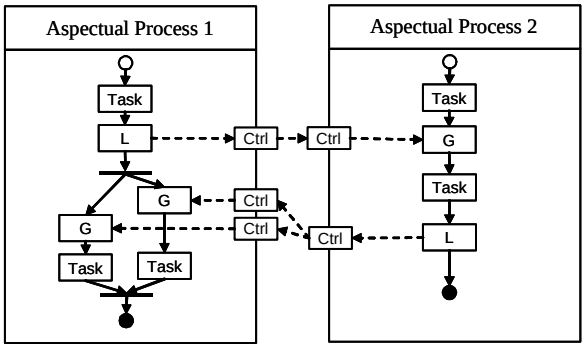


**Fig. 4 Assembled Aspectual Process**

If we want to add certain aspectual process to another aspect or process, we need assemble aspects together to build a complete business process which reflects all critical and typical aspects (see Fig.4). Assembling is realized by linking corresponding interaction point pairs. Among these aspectual processes, the exchange of control tokens represents the logical integration of aspectual processes. We can manually build the interaction relationships among those aspectual processes. If business semantics are recorded in exchanged tokens, it is also possible to automatically connect each pair by semantics.

### 3.5. Aspect Weaving

The assembled business process depicts different aspects as separated aspectual processes and visualizes their relationships clearly. However, the assembled result is not a general business process and a weaving algorithm is needed to merge them together into one single "aspectual process" that represents the combination of all related aspects.

## 4. Key Enabling Technologies

The previous section provides an overview description about our AOBPM approach, which follows the principle of 'separation of concern ' and 'divide and conquer ' in business process modeling by modeling and weaving of aspectual processes. In this section several key enabling technologies evolved in our approach are explained in detail, including aspect identification by process query, process extraction, process assembling and weaving, and the further optimization for simplifying the process extraction / weaving results. It must be emphasized here that we have laid enough theoretical groundwork so as to implement a "reasonable" process extraction / weaving approach and result optimization mechanism. More specifically, the correctness of the above techniques is ensured by applying a set of model transformation rules that are deducted from the weak-bisimulation analysis of our formalized business process model with the process algebra of CCS [11]. Consequently, the business process models before and after applying these techniques are guaranteed to be equivalent by their observable behavior to the external environment. Due to the size limitation of this paper, this part of theoretical support will be addressed in detail in another incoming paper.

### 4.1. Aspect Identification by Process Query

Aspect definition and identification are the very first two steps of our AOBPM approach. Typically a user can identify the critical/dominant aspects either by manually selecting different elements in a business model (i.e. activities, documents, resource, etc) or through an intelligent query mechanism which automatically returns all business elements that satisfy specific query conditions. Our query engine is built upon the extension of Pattern Unified Matcher (PUMA) [12] in CME, which enables the automatic identification of business elements, their relations and even their different semantics in business process models.

Originally in CME, a common meta-model for various software applications is provided based on which PUMA can search for its desired results. The common meta-model can be briefly concluded in the left part of Fig.5 below. All external application models are represented in CME with two basic concepts: concerns and the hierarchical structure of its units. Each unit is further associated with an artifact definition which contains the location of where the unit is physically stored. Various relations can be defined among different units. In order to realize the aspect oriented business process modeling, an abstract business process model is extended based on this common structure as illustrated in the right part of Fig.5. The abstract process model contains the most usual business model constructs (e.g. *Activity, Fork, Join, Decision, Merge, Timer, Information Artifact, Collaboration Entity,...*) and their possible properties (e.g. *role, purpose, cost, date* ……). Furthermore, to enhance the flexibility for the dynamic creation of user-customized model constructs, these extended units can be generated "on-the-fly" according to user 's own configuration of his/her query targets in a business process model. This is especially important for unit like properties since different users usually concerns about different information in the business process model.

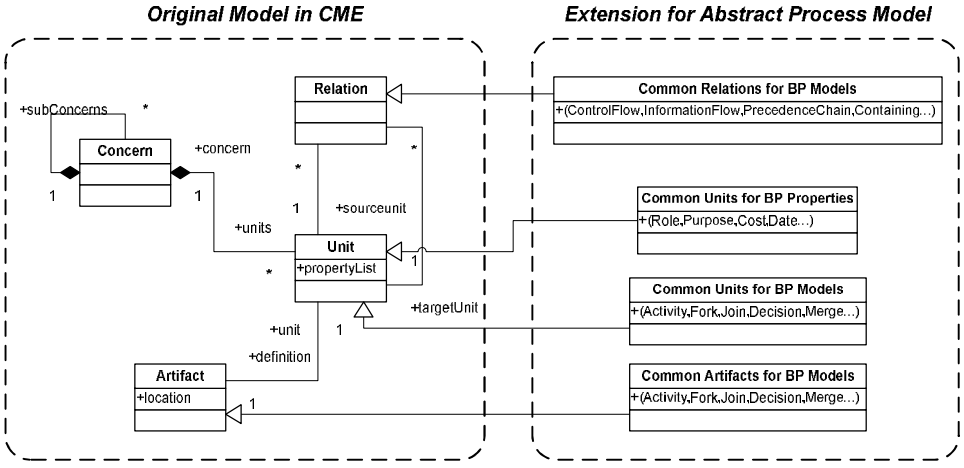


**Fig.5 Extended Abstract Process Model for Aspect Identification**

$CommonBPUnitType ::= Process \mid SubProcess \mid Activity \mid Decision \mid Fork \mid Join \mid Merge$
$ActivityLane \mid DataLane \mid InformationArtifact \mid CollaborationEntity$
$CommonBPProperty ::= Cost \mid Date \mid Time \mid Purpose \mid Role \mid Type \mid Value \mid Duration \mid Version \mid Id$
$CommonBPRelation ::= ControlFlowTo$ /*The control flow relation between two units
$InformationFlowTo$ /*The information flow relation between two units
$PrecedenceChain$ /*A chain of units that is precedent to a specific unit
$Containment$ /*The containment relation among units
$ExtendedQueries ::= IdenQuery \mid RelationshipQuery \mid ContainmentQuery \mid SemanticQuery$
$IdenQuery ::= CommonBPUnitType \mid CommonBPProperty\ \ Op\ \ Value$
/*Identify and Return units in BP models by the specification of its value
$RelationshipQuery ::= relationship\ CommonBPRelation(IdenQuery_{source}, IdenQuery_{target})$
/*Return units in BP models that satisfies the specific CommonBPRelation
$ContainmentQuery ::= containment(CommonBPUnitType, IdenQuery, [Recursive])$
/*(Recursively) Search for units in BP models that are not only of specific
CommonBPUnitType but also contain units that are identified by IdenQuery
$SemanticQuery ::= semantic\ [CommonBPUnitType]\ IdenQuery$
/*Semantically identify units in BP models by the specification of its value

**Fig.6 Query Syntax for Business Process Models**

A comprehensive list of the syntax for querying about model elements and their relations in our extended abstract process model is summarized in Fig.6.

***CommonBPUnitType*** and ***CommonBPProperty*** are the basic predicates for constructing our queries. In the above syntax specification, "***Op***" indicates the general binary logical relation for "equality", "less than", "more than", and etc. Wildcards like "*" and "?" are supported in the specification of "***Value***"s. Using the query "***containment(Activity, Role=role1)***" that will be used in the next section as a sample, it consists of a reserved key word "***containment***", the specification of its target ***CommonBPUnitType*** "***Activity***" and the unit that each target "***Activity***" is supposed to contain: "***Role=role1***". With this query, all Activities that are performed by the role of "***role1***" can be retrieved.

## 4.2. Intelligent Aspect Identification with Semantics

In Fig 6, there is a special "***SemanticQuery***" with key word "***semantic***" in the query syntax specificiation This means that in our query language, units in business process models can not only be explicitly identified by specifying its value or wildcards, but also be indirectly deduced from the semantics of business model elements. Querying business models with semantics enables the intelligent identification of aspects with the query engine. To implement this important feature, a general ontology repository of the most commonly used synonyms of terms in specific business domains [1] is maintained for the query engine. Whenever a semantic query is triggered with the key word of "***semantic***", the query engine will first search for business units that are the exact match of the expected values, it will then access the ontology repository and try to identify other qualified units that are semantically equivalent to the expected value.

For example, with the query of "***semantic Activity Purpose=PayMoney***", not only activities with purpose of "***Pay Money***" are returned, but also activities with purpose of "***Pay Cash***" or "***Pay Bill***" will also be identified.

## 4.3. Aspectual Process Extraction

The main purpose of aspectual process extraction is to separate the target business aspects from original business process model, and keep the extracted aspectual process as a meaningful self-contained process. The extracted process is supposed to represent a specific aspect of the original business process model. In Fig 7, a simple scenario is used to provide an intuitive understanding of the overall aspectual process extraction procedure. In the sample scenario, it is expected that the aspect of ***security operations*** (annotated with an asterisk mark) will be separated from the original business process model to help business consultants get a clearer understanding of the core businesses. The steps of the extraction procedure are explained as following:

***Step1:*** Identify the to-be extracted business elements in the original process "$\boldsymbol{P}_{original}$" and insert losing / gaining

[1] As in our case, we focus mainly on the business domain of banking.

control pairs with unique IDs before and after these elements respectively.

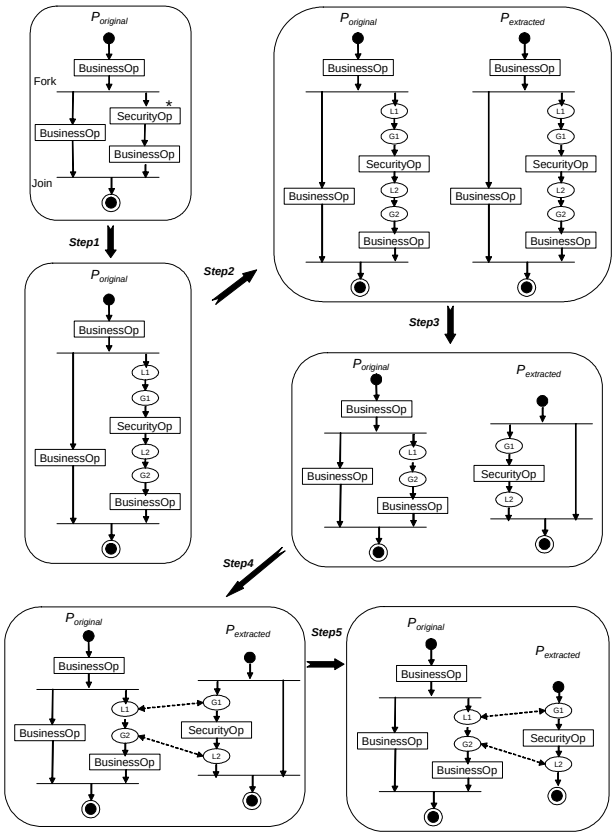


**Fig.7 A Simple Scenario for Aspectual Process Extraction**

For example, in Fig.7 two losing / gaining control pairs *L1/G1* and *L2/G2* are inserted before and after the activity "*SecurityOp*" respectively. These pairs will serve later as the potential interaction points for the process model be integrated with its external environment.

***Step2:*** Create a duplication of the original process "$P_{original}$" called "$P_{extracted}$", as the initial extraction result.

***Step3:*** Reduce "$P_{original}$" and "$P_{extracted}$", by applying the following rules.

For "$P_{original}$":

- Remove all model elements belonging to the extracted aspect;
- For the losing / gaining control pairs that are inserted before the to-be extracted business elements, remove the gaining control node in the pair;
- For the losing / gaining control pairs that are inserted after the to-be extracted business elements, remove the losing control node in the pair;

For "$P_{extracted}$":

- Remove all model elements that do not belong to the extracted aspect;
- For the losing / gaining control pairs that are inserted before to-be extracted business elements, remove the losing control node in the pair;
- For the losing / gaining control pairs that are inserted after to-be extracted business elements, remove the gaining control node in the pair;

Fig 7 (step3) shows the reduced results of "$P_{original}$" and "$P_{extracted}$" in the sample scenario according to the above rule.

***Step4:*** Connect each single losing control node and gaining control node in the two separated process by their unique ID.

In this way, we connect the sparsely scattered process fragments of the current extracted process results into a complete and self-contained aspect. The losing control node in the process model now indicates the generation of request tokens for calling external functions outside this aspect. On the other hand, the gaining control node thus blocks the execution of the current process model and waits until a request token is received from external side. By introducing the losing control and gaining control nodes with ID as identification, we are able to capture the inter-dependencies between the business process of "$P_{original}$" and "$P_{extracted}$" and preserve the completeness of our extracted results in terms of semantics.

***Step5:*** A rule driven optimization procedure is applied on both "$P_{original}$" and "$P_{extracted}$" to further remove or merge redundant structure nodes and control nodes.

For example, in fig7 (step5) the fork/join structure in "$P_{extracted}$" can be removed to simplify the extracted process model since an alternative process branch that actually does nothing is found in the structure.

As stated earlier, the correctness of the above aspectual process extraction procedure and optimization approach is ensured by proving the observable behavior equivalence between the original process model and our extraction results based on the weak-bisimulation analysis of the formalized business process model with CCS [11].

## 4.4. Process Assembling and Weaving

Process assembling and weaving enables the automatic business model reconstruction by integrating and reusing existing aspectual business process. We will also use a simple scenario here to provide an intuitive understanding of the complete assembling and weaving procedure as shown in fig 8. In the sample scenario, it is expected that the original business process can be conveniently reconstructed so as to support additional functionality of financial operation (which is modeled in another process fragment) in the end of the process. Similar to extraction, each correspondingly step of the assembling and weaving procedure is explained in detail as following:

***Step1***: Losing control and gaining control nodes are pinpointed on the to-be weaved aspectual processes as

their external interaction points either automatically by the operation of process extraction, or manually by users;

In the sample scenario, the financial process "$P_{financial}$" needs to be weaved into the original business process "$P_{original}$" to supplement its functionality. Firstly, the losing control and gaining control nodes L3, G4, G3, L4 are pinpointed in the two processes.

***Step2***: The dependencies among different aspectual processes are established by pairing of losing control and gaining control nodes with their unique IDs;

***Step3***: Each pair of losing control and gaining control nodes is merged to generate the parallel or synchronization relations between aspectual processes.

For example, in Fig.8 (step2), the merge of losing / gaining control pair is fulfilled by replacing each pair with additional Fork / Join structure nodes in the process models.

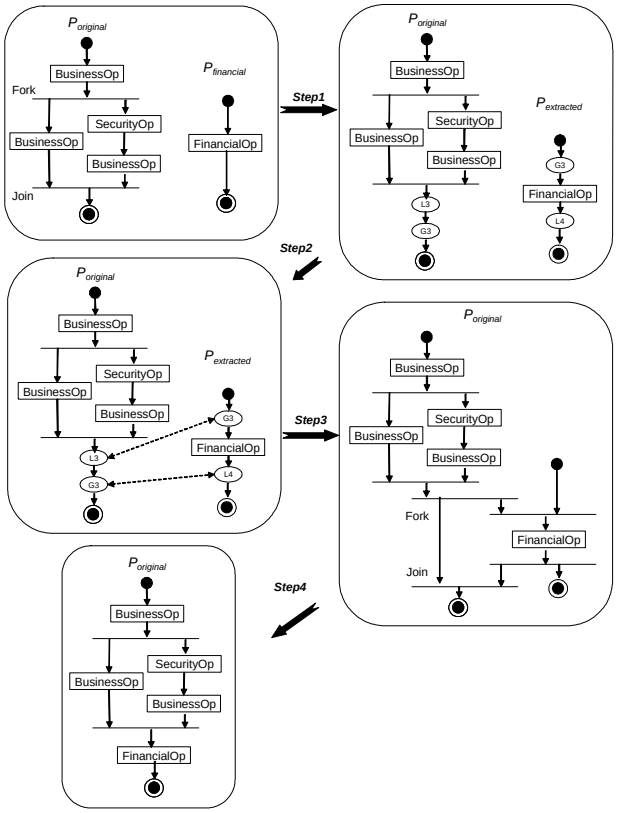


**Fig.8 A Simple Scenario for Business Process Weaving**

***Step4***: A rule driven optimization procedure is applied on the connected business process to further remove or merge redundant elements.

In the sample scenario, the final optimization operation removes the additional Start / Finish node of "$P_{financial}$", merges the redundant Fork and Join nodes in the original weaved process model.

It should also be noted here that the correctness of the above aspectual process assembling and weaving procedure and optimization approach is ensured by proving the observable behavior equivalence between the original process models and our weaving results based on the weak-bisimulation analysis of the formalized business process model with CCS [11].

## 5. Proof of Concept with a Prototype

To verify above concepts and technologies, we built a prototype system on top of CME and WSU for us to practice several useful business scenarios. In this section, one scenario will be used to demonstrate how we cab use aspect-oriented approach to flexibly manipulate business process models, i.e. refactoring an existing model from another meaningful viewpoint. The key objective of this scenario is to refactor an existing business process model from each individual role 's perspective. Usually, "role" is a typical property of business tasks. Business process models can be regarded as the combination operations performed by in different roles. Refactoring a business process from "role" perspective can help user identify the responsibility of each individual role, demonstrate the interaction relationship among roles, and enable workload analysis and optimization of efficiency. For other purposes, you can also refactor process from other viewpoints. In this scenario, there are two important phases: (1) refactor process by executing aspect definition, identification and extraction; (2) merge aspectual processes together by executing aspect assembling and weaving. Limited by paper size, we will use a fictive business process rather than a real one from customer cases.

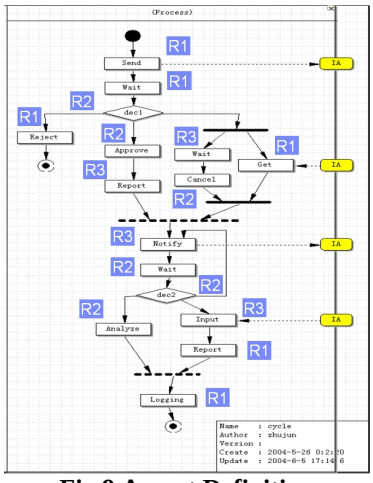


**Fig.9 Aspect Definition**

### I Aspect Definition

Before manipulating aspects, we should define which aspects we are interested in and what they are. In this example, we are interested in the tasks performed by certain roles. So I can define aspects by roles. In the following sample process, we marked business tasks with roles (R1, R2 and R3). We define all tasks performed by

role R1 as an aspect Role1, tasks performed by role R2, R3 as aspect Role2 and Role3.

- Aspect Identification

We can use two ways to do aspect identification: by query language or by manual selection. For example, in the following diagram, we use a query command "*containment(Activity, Role=role1)* " to find all business tasks done by 'role1'. The query result is listed in CME search result view, and also listed under the aspect "*Role1*".

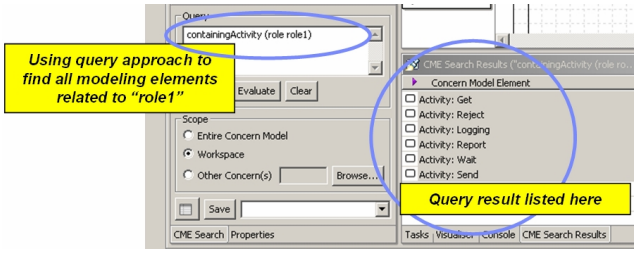


**Fig.10 Aspect Identification by Query**

- Aspect Extraction

Although we have identified all elements related with role1, they are still as separated elements. To make them more meaningful, aspect extraction operation can be performed to extract all aspect elements, transform them into a self-contained and reusable aspectual process format. The following diagram shows the extraction result. All tasks performed by role1 are linked together as an aspect of the whole business process from role1 's point of view. Some additional nodes are added, such as the losing-control node and the gaining control node. At these nodes, control tokens are sent to or received from other aspects. Those nodes can be considered as the interface of interaction with other aspects.

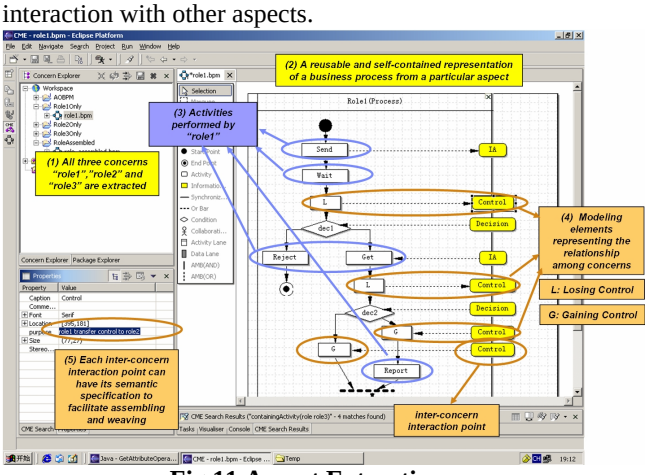


**Fig.11 Aspect Extraction**

Through the steps of aspect definition, identification and extraction, the aspect-oriented technology makes it possible to flexibly componentize business process model. It provides another view to make the process clear and enable independent change and reuse.

- Aspect Assembling

To integrate aspectual processes together, we can assemble them like the following figure and create the linkages between pairs of control tokens. Linkages can be automatically created with semantic matching or manually built. This diagram presents another perspective of the original business process model by role-based refactoring. Here, we only select role as a typical view, people can choose any other concerns for each specific case, such as location, time, financial dimension, and security-related aspects.

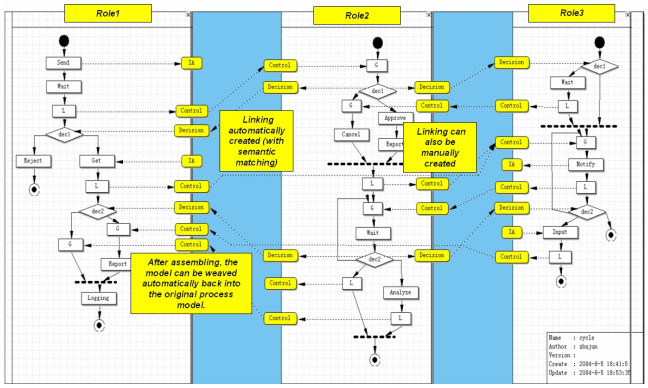


**Fig.12 Aspect Assembling**

- Aspect Weaving

Aspect weaving is a followup action to fuse the assembled process model into one complete process. This is an optional step for this scenario. If there is no change to any aspectual processes during all above steps, aspect weaving is expected to fully recover the original business process at the starting point.

## 6. Conclusion

This paper introduces an end-to-end aspect-oriented business process modeling method and related technologies to support dynamic model componentization. The concept of aspect is from programming domain (AOP) is introduced into business process modeling domain for for reducing complexity in business process models and improving model reusability. Contribution of this work can be summarized as following:

- It extends the method in CME for AOBPM, which covers the following key phases: aspect definition, identification, extraction, assembling, and weaving. AOBPM enables the flexible process model manipulation, such as model refactoring, componentization, adding new feature and model reuse, etc.
- It provides a novel presentation way to intuitively visualize aspectual business process models and their relationship.

- It extends the query language provided by PUMA query engine with process-related query and semantic-based query capability.
- It provides a set of approach for aspectual process extraction and weaving with CCS as its theoretical foundation.

Although we have built some scenarios for the purpose of showing ideas, there is still a long way to make this technology truly practical and consumable. We will continue the AOBPM work from both theoretical and practical perspectives in the future.

## 7. Reference


[1] OMG Model-Driven Architecture, http://www.omg.org/mda.

[2] Meta-Object Facility, http://www.omg.org/cgi-bin/doc?formal/2002-04-03.

[3] Unified Modeling Language, http://www.uml.org/#UML2.0

[4] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira, and J.-M. Loingtier. Aspect-Oriented Programming. In M. Aksit and S. Matsuoka, editors, Proceedings of the 11th European Conference on Object-Oriented Programming, volume 1241 of LNCS, pages 220{242, Jyv ÄaskylÄa, Finland, June 1997. Springer-Verlag.

[5] William Harrison, Harold Ossher, Stanley Sutton Jr., Peri Tarr, "Concern Modeling in the Concern Manipulation Environment ", IBM Research Report RC233443, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, September 2004.

[6] P. Tarr, H. Ossher, W. Harrison, and S. Sutton Jr., "N degrees of separation: Multi-dimensional separation of concerns", In 21st Int'l Conf. Soft. Eng, IEEE, 1999.

[7] AspectJ, http://eclipse.org/aspectj/

[8] Concern Manipulation Environment, Eclipse Technology Project, http://www.eclipse.org/cme/.

[9] A. Charfi and M. Mezini. "Aspect-Oriented Web Service Composition with AO4BPEL ", In Proceedings of European Conference on Web Services (ECOWS '04), Erfurt, Germany, September 2004.

[10] Jun, Zhu and etc. "Model-Driven Business Process Integration and Management: A Case Study with Bank SinoPac Regional Service Platform ", on IBM Journal of Research and Development issues in Asia, 2004

[11] Robin Milner: A Calculus of Communicating Systems, Springer Verlag, ISBN 0387102353, 1980

[12] Peri Tarr, William Harrison, Harold Ossher, "Pervasive Query Support in the Concern Manipulation Environment", IBM Research Report RC23343, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, September 2004.