# Correctness of aspect-oriented business process modeling

Xu Wang
*School of Economics, Yunnan University, Kunming, China, and*
Xuan Zhang, Tong Li, Junhui Liu and Qingyi Chen
*School of Software, Yunnan University, Kunming, China and*
*Key Laboratory of Software Engineering of Yunnan, Kunming, China*

Received 22 April 2016
Revised 4 October 2016
1 December 2016
28 April 2017
Accepted 15 May 2017



## Abstract

**Purpose** – Business process models, while primarily intended for process documentation, communication, and improvement, are often also used as input for developing process-oriented software systems (Ouyang *et al.*, 2009). Ensuring correctness, handling complexity, and improving reusability and maintainability of business process models are important for all these goals. The purpose of this paper is to propose an aspect-oriented business process modeling and correctness controlling method based on Petri nets to satisfy these goals.

**Design/methodology/approach** – The aspect-oriented paradigm provides a proper mechanism to modularization, and thus reduces the complexity of models, and also improves reusability and maintainability. However, weaving aspects into base processes may bring in mistakes or errors. To ensure correctness of modeling, this paper presents a formal approach to modeling aspect-oriented business processes and a method to ensure modeling correctness. Petri net is used as the process modeling language and its analysis techniques are applied to analyze the correctness of modeling. Two types of correctness, specifically, aspect-aspect correctness and base-aspect correctness are analyzed. A real banking process model is studied systematically in the case study to evaluate the approach and the performance assessments are conducted to show the cost and effect of the approach.

**Findings** – Designing aspect-oriented business process models help organizations reusing the model elements to reduce redundancy of their model repository, improving their maintainability, and supporting them to adapt to the changes of business requirements with flexible modeling. It is important to stress that the correctness of business process modeling is important in ensuring the quality of the models, especially in the safety-critical business domains, such as financial business domain.

**Originality/value** – In this paper, separation of concerns is used to separate the cross-cutting activities and core activities in accordance with the different functions of these activities, and an approach to modeling aspect-oriented business processes is proposed. First, the cross-cutting activities are encapsulated as aspects, while core business activities are modeled as base processes. Then, according to the correctness requirements of business process models, based on the weaving mechanisms of aspect-oriented approach, weaving correctness is defined. Weaving correctness controlling methods between multi-aspects and between aspects and base processes are designed. Errors or mistakes of aspect-oriented business process modeling are prevented during the procedure of modeling to ensure error-free business process modeling.

**Keywords** Aspect-oriented modelling, Business process modelling, Correctness, Petri nets

**Paper type** Research paper

## 1. Introduction

As it becomes increasingly common for organizations to work in a process-oriented manner, single organizations may be dealing with collections of hundreds or thousands business process models. Examples of such collections include 735 models in the BIT Process Library, 604 models in the SAP Reference Model, around 600 models in Dutch municipalities, around 250 models in the IBM's Insurance Application Architecture, and 6,000 + models in Suncorp's process model repository for insurance (Dijkman *et al.*, 2012). In such large collections of process models, the cross-cutting concerns were modeled as the integral parts of the business processes. The tangling of the cross-cutting concerns and base processes leads to this redundancy and less reusability.

Cross-cutting concerns, such as compliance, auditing, business monitoring, accounting, billing, authorization, privacy, and separation of duties, are highly repeated in different business process models. Traditional modeling languages do not provide appropriate means

Business Process Management Journal
Vol. 24 No. 2, 2018
pp. 537-566

1463-7154
DOI 10.1108/BPMJ-04-2016-0083

to model such concerns in an organized manner. When modeling cross-cutting concerns using state of the art language, two problems have been observed (Anis *et al.*, 2010). First, model elements that address certain cross-cutting concerns are often scattered across different business processes. Keeping their consistency when one of these concerns is changed is difficult. Second, business process models for the same purpose but in different projects can be reused due to the fact that they often share the same set of base business processes but have different concerns. For example, banks have same business processes, such as deposit, withdrawal, and loan, but different for authorization or accounting, etc. Mixing base processes and cross-cutting concerns make the business processes less reusable and flexible. Furthermore, business processes should well match the developments and changes of business requirements. This business requirements change may make problems even more challenging.

Separation of concerns (SoC) is one of the useful methods to solve these issues. The concept of SoC was first proposed by Dijkstra (1976) and Parnas (1972). Its key idea is to identify different concerns early and separate them by encapsulating them in appropriate modules (Schauerhuber *et al.*, 2006). Multiple SoC approaches have been proposed, such as composition filters (Aksit *et al.*, 1992), subject-oriented programming (William and Ossher, 1993), adaptive programming (Lieberherr, 1996), aspect-oriented programming (AOP) (Kiczales *et al.*, 1997), and multi-dimensional separation of concerns (Tarr *et al.*, 1999). The common techniques share the same nature: modularizing cross-cutting concerns into aspects with the advice invoked at the specified join points of base models. However, in the aspect-oriented mechanism, aspects may affect base models or affect other aspects. If the effects are adverse, it will destroy desired correctness properties or even the conceptual integrity of business processes. Therefore, when using the aspect-oriented paradigm, ensuring the correctness of business process models is important. In this paper, the correctness of aspect-oriented business process modeling is analyzed.

Becker *et al.* (2000) presented six guidelines to ameliorate the quality of business process models. One of the basic guidelines is the guideline of correctness. Their guideline of correctness has got two facets: the syntactic and the semantic correctness. A model is syntactic correct if it is consistent and complete against the meta-model the model is based on. Semantic correctness postulates that the structure and the behavior of the model are consistent with the real world. Afterward, most correctness definitions are based on Becker's correctness definition.

In this paper, based on Becker's definition and the characteristics of the aspect-oriented modeling, we designed an aspect-oriented business process modeling meta-model first and then defined aspect-aspect correctness and base-aspect correctness. An aspect-oriented business process modeling meta-model is a formal tool which is used to define aspect-oriented business processes. Based on the weaving mechanisms of aspect-oriented approach, aspects can interact with each other or with base processes. According to the correctness requirements of business process models, weaving correctness between multi-aspects and between aspects and base processes are designed. Correctness is based on a mechanism to avoid problems before their occurrences (Blair *et al.*, 2005). Therefore, aspect-aspect correctness is when multiple aspects are integrated at the same join points they must be woven together based on their interdependencies to keep correct relations among them. Base-aspect correctness is when aspects are woven into base models, there is no execution problem, such as halt or raise an exception, while the base models can execute properly in isolation.

The rest of this paper is structured as follows. In Section 2, the related work is introduced and compared with our work. Section 3 presents an approach for aspect-oriented business process modeling. Section 4 discusses aspect-aspect correctness based on the interdependencies among aspects. Section 5 uses Petri net properties to analyze

base-aspect correctness and proposes correct base-aspect weaving operations. Section 6 uses four banking business processes to test whether the approach really ensures correctness. Finally, Section 7 concludes and outlines future work.

## 2. Related work

Aspect-orientation provides a new way of modularization by clearly separating cross-cutting concerns from core concerns. Some proposals to aspect-oriented business process modeling have already provided different concepts, notations, and maturity.

In 1999, Odgers and Thompson (1999) first combined the concepts of business process management and AOP to present aspect-oriented process engineering (ASOPE) which enabled the creation of flexible and dynamic business processes. ASOPE was proposed for the construction of specialized business processes which were customized by combining the views of all participants. Park *et al.* (2007) considered that the business rules are cross-cutting concerns that should be distinguished from core business processes. Thus, they presented a rule-based aspect-oriented programming framework where business rule aspects contained in business processes can be effectively separated and executed. Pourshahid *et al.* (2009), Amyot and Mussbacher (2011) and Amyot (2013) proposed to use Aspect-oriented User Requirements Notation which was an aspect-oriented extension to Use Case Maps and Goal-oriented Requirement Language to implement aspect-oriented business process improvement. Afterward, modularization of business processes becomes a topic which has been attracting the attention of some researchers, such as Charfi and Mezini (2007), Charfi *et al.* (2010), Cappelli *et al.* (2010), Santos *et al.* (2012), and Leite *et al.* (2016). Charfi and Mezini (2007) and Charfi *et al.* (2010) first extended the aspect-oriented method to Business Process Model and Notation (BPMN) and Business Process Execution Language (BPEL) and proposed AO4BPMN and AO4BPEL. Cappelli *et al.* (2010) used the aspect-oriented concepts to the design of business processes and suggested heuristics for aspect identification. Then, Santos *et al.* (2012) listed initial thoughts on possible solutions for five open issues on aspect-oriented business process modeling. These open issues are aspects identification, elements used in the models, levels used to modularize business process models, assignment of aspects to organizational actors, and ways that an aspectized model can be generated or visualized. As to the issue of "assignment of aspects to organizational actors," Leite *et al.* (2016) represented the actor involved in process and aspect ownership as an instantiation of the *i** strategic actor by combining the aspect-oriented approach and the *i** strategic actor model.

Some of these researchers, such as Odgers and Thompson (1999), Pourshahid *et al.* (2009), Amyot and Mussbacher (2011), Amyot (2013), and Santos *et al.* (2012) focused on establishing concepts of aspect-oriented extensions to the business process modeling. Some of the others, such as Park *et al.* (2007), Charfi and Mezini (2007), Charfi *et al.* (2010), and Cappelli *et al.* (2010) proposed the detailed approaches to modeling. Leite *et al.* (2016) argued the importance of clearly established process ownership from the organizational perspective. To our knowledge, none of them considered the correctness when extending aspects to the business process model. Correctness is important for business process models since detecting process anomalies before process models are put into operation can help reduce high costs of breakdown, debugging, and fixing during runtime. A process anomaly here is an improper design that causes execution errors (Bi and Zhao, 2004).

### *2.1 Correctness of business process models*

In business process modeling, quality is referred to a set of attributes that are related to a business process model. Correctness is one of these attributes. Bi and Zhao (2004) formally defined the syntax and semantics of process logic and transform the problem of verifying the correctness of process models into a problem of determining the validity of logic argument forms. Mendling (2008) identified theoretical arguments on why structural

metrics should be connected with error probability and provide an empirical validation of this connection. Dijkman *et al.* (2008) proposed a mapping from BPMN to formal language Petri nets for enabling the static analysis of BPMN models. In order to ensure the correctness of the Petri net-based process models, van der Aalst *et al.* (2008) proposed a framework for staged correctness-preserving configuration of reference process models both with respect to syntax and behavioral semantics. Afterward, they specifically aimed at checking that a configuration step preserves the structural properties of workflow nets (van der Aalst *et al.*, 2010). Laue and Mendling (2010) defined metrics that quantify the (un) structuredness of a process model, the degree of structuredness, and the unmatched connector count to predict error probability.

Our work is inspired by these approaches but it is targeted at aspect-oriented business process models. We deal with aspect-oriented models and aspect-aspect correctness and base-aspect correctness need special attention.

### *2.2 Aspect-oriented correctness*

Constantinides *et al.* (1999) proposed an aspect-oriented design pattern which is called the aspect moderator pattern. This pattern made use of a moderate class to moderate the functional behavior together with different aspects of concerns by handling their interdependencies. The moderator class defined the order of activation of the aspects to coordinate the interaction between aspects. In AspectJ, the ordering of advices at the same join points is resolved with respect to the relative precedence of the aspects in which the advice is defined. The correctness problem in AspectJ can be resolved by reordering aspects using dominates modifier (Kiczales *et al.*, 2001). Douence *et al.* (2002, 2004) presented two general aspect independence properties (strong independence and independence w.r.t. a program) and the approach to analyze them statically. Based on the static analyses, they proposed some useful commands for correctness problem resolution. Pawlak *et al.* (2005) also showed that the correctly ordering of the aspects is the way to ensure the aspect-aspect correctness. Therefore, they presented a language called CompAr (for composing around advice) which helps the programmer to find and validate the right composition order in a rigorous way. Nagy *et al.* (2005) proposed a general and declarative model for defining constraints upon the possible compositions of aspects at the same join points. They distinguished between two main categories of constraints: ordering constraints and control constraints. Ordering constraints specify a partial order upon the execution, while control constraints specify conditional execution. Durr *et al.* (2005) defined the semantics of advices in terms of operations on an abstract resource model. After analyzing all advices at the same join points, errors or mistakes can be detected based on the required and disallowed sequences of operations on these resources. Kniesel (2009) and Kniesel and Bardey (2006) showed that a large class of aspect interference results from incorrect or incomplete weaving. Based on this concept, they developed a formal automated solution for weaving interaction detection and resolution. Their approach consists of a three-step procedure: creating a graph of potential weaving interaction, analyzing correctness, and creating an execution plan that is guaranteed to preserve correctness and completeness. Dinkelaker *et al.* (2012) proposed a formal approach to detect and resolve feature interactions. Their approach is based on formalism for aspect-oriented state machines (AO-FSM) and language implementation AO4FSM based on finite-state machines and essential behavioral models. They used AO-FSM to intercept, prevent, and manipulate events that cause errors or mistakes.

Summarizing these studies, aspect-aspect correctness in aspect-oriented methods has been studied by all these people for many years. We can conclude that aspect-aspect correctness is determined by the right interdependent aspects. The way to ensure these kinds of correctness is to analyze the interdependencies among aspects first, and then weave them based on the interdependencies.

We have also found that the correctness is important both in business process modeling and aspect-oriented methods, but there is no research on the correctness of aspect-oriented business process modeling. In our work, correctness of aspect-oriented business process modeling will be defined and analyzed based on the correctness requirements from business process modeling and the weaving mechanism of the aspect-oriented modeling. The controlling of the aspect-aspect correctness will be adapted from the aspect-oriented correctness methods. Furthermore, correctness controlling methods to ensure base-aspect correctness will be proposed.

In order to provide the detailed business processes modeling approach and control the correctness of modeling, we proposed to use aspect-oriented Petri nets to model business processes and provide correctness requirements when aspects are woven. Petri nets are well-founded process modeling techniques. A business process specified in Petri net languages has a clear and precise definition due to the fact that the semantics of the Petri nets have been formally well defined (van der Aalst, 1998). Also, Petri nets benefit from a rich body of theoretical results, analysis techniques, and tools. They have been extensively applied to the formal verification of business process models. These features make Petri nets suitable for establishing a formal foundation for business process model (van der Aalst *et al.*, 2008). In the following, related researches on aspect-oriented Petri nets and correctness analysis are presented.

### *2.3 Aspect-oriented Petri nets*

Xie and Shatz (2000) blended the Petri net notation with object-oriented features to develop a domain-specified formal notation called state-based object Petri nets (SBOPN). SBOPN can be seen as an object-interpreted form of colored Petri net and is used to model every single aspect and aspect weaving. To solve the problems of losing readability and traceability of design decisions when modeling in Petri nets, Roubtsova and Aksit (2005) proposed aspect Petri net notation. This notation used classical Petri nets to model aspects separately and a logical language to specify rules of aspect weaving. Aspect Petri net was defined by following the general principles of the aspect-oriented approach to software development. Xu and Nygard (2006) proposed aspect-oriented predicate-transition Petri net (PrT net) which was incorporated the fundamental features of AOP into PrT net. Each Petri net-based aspect was an encapsulated entity of introductions, pointcuts, and advices. An introduction, represented by a PrT net, modeled the functionality common to the advices in the aspect. Pointcuts could be transitions, predicates, and arcs in base nets. An advice, represented by a PrT net, specified the operations to be applied to the join points that match the pointcuts. They also formalized a weaving process for integrating aspects into base nets. Based on Xu's aspect-oriented PrT nets (Xu and Nygard, 2006) and Nagy's *et al.* (2005) aspect relations definitions, Guan *et al.* (2008) proposed solutions to resolve the problems about aspect-aspect correctness that may exist among the aspect nets. Molderez *et al.* (2012) presented the aspect-oriented extension to Petri nets. They defined stereotypes for places and transitions, and introductions in an advice. Stereotypes for places were used to specify how the input and output places of a join point should be bound. Stereotypes for transitions were used to indicate a proceed transition. Introductions were used to define places that can be shared among advice. Ouyang *et al.* (2009) applied Petri net analysis techniques to statically check the semantic conditions on the BPMN model and translated BPMN models into languages used by software system developers, such as BPEL.

All these studies focused on the approach of modeling aspect-oriented Petri nets. Only Guan *et al.* (2008) analyzed the aspect-aspect correctness. As mentioned above, analysis and controlling of base-aspect correctness should also be considered.

In the following, we first present an aspect-oriented modeling approach based on Petri nets to model business processes. Then, analyzing the correctness in aspect-oriented business process modeling and propose correctness controlling methods.

## 3. Aspect-oriented business process modeling

In modeling aspect-oriented business processes, a meta-model which includes a number of different formal components is defined. These components are the base process, join point, advice, aspect, and weaving mechanism.

### *3.1 Base process*

A base process is a Petri net that is modeled by the core activities of the business processes. Cross-cutting concerns in business processes, such as compliance, auditing, business monitoring, accounting, billing, authorization, privacy, and separation of duties are separated from the base processes:

*Definition 1.* (Base process). A base process is a 4-tuple $N = (C, A; F, M_0)$, where:

(1) $(C, A; F)$ is a Petri net without isolated elements, $A \cup C \neq \varnothing$.

(2) $C$ is a finite set of conditions; $\forall c \in C$ is called a condition.

(3) $A$ is a finite set of activities; $\forall a \in A$ is called an activity; the execution of $a$ is called that $a$ is fired or enabled.

(4) F is a finite set of flow relations, $F \subseteq (C \times A) \cup (A \times C)$.

(5) $M \subseteq 2^C$ is a set of case, $2^C$ is the power set of $C$, $M_0 \in M$ ($M_0 \subseteq C$) is an initial case of $N$.

(6) $\forall a \in A$, $\exists m \in M$, such that $a$ has concession in $m$.

In the above Definition 1, a condition is a place in Petri nets, an activity is a transition in Petri nets, and a flow relation is an arc in Petri nets. The definition of conditions, activities, and flow relations is for a better understanding of business processes.

### *3.2 Join points*

In the aspect-oriented programming, a join point is a well-defined position in the base program where additional behavior can be attached (Blair *et al.*, 2005). At the modeling level, a join point represents a well-defined element in the base model where an advice can be introduced. In this paper, join points in the base processes are the elements of a Petri net:

*Definition 2.* (Join point). Given a based process $N = (C, A; F)$, a join point $j$ is a 2-tuple $j = (jp, e)$ where $jp$ is a condition $jp \in C$, or an activity $jp \in A$, or a flow relation $jp \in F$ of $N$, $jp \in JP$, $JP \mapsto \{C, A, F\}$. $e$ is the weaving type of an aspect. Weaving type is categorized into before type, after type, around type, iteration type, and concurrency type which are described as 0, 1, 2, 3, and 4, respectively.

We use the enumeration to define $jp$ and describe their syntactic construction in Extended Backus-Naur Form:

$$jp = \text{ConditionList},\ \text{ActivityList},\ \text{FlowList}$$

$$\text{ConditionList} = N.c | N.c, \text{ConditionList}$$

$$\text{ActivityList} = N.a | N.a, \text{ActivityList}$$

$$\text{FlowList} = N.f | N.f, \text{FlowList}$$

In the above construction, $N.c \in N.C$, $N.a \in N.A$, $N.f \in N.F$.

*3.3 Advices and aspects*

An advice specifies how to augment or constrain base processes. Each advice is defined with a set of join points which determine the position where the aspects are attached. When a join point is matched, the advice is executed. The corresponding relations between an advice and a base process are a sequence, selection, iteration, or concurrence. Petri nets are used to define the functions of advices:

*Definition 3.* (Advice) An advice is a 5-tuple $v = (C, A; F, A_e, A_x)$, where:

(1) $(C, A; F)$ is a Petri net where $C$ is a set of conditions, $A$ is a set of activities, $A \cup C \neq \varnothing$, $F$ is a set of flow relations of $(C, A; F)$, $F \subseteq (C \times A) \cup (A \times C)$.

(2) $A_e$, $A_x \subseteq A$ are the entrance activity set and the exit activity set of the advice, respectively.

The difference between a base process and an advice is that the advice is not indispensable to the business requirements to reach its goal and thus is represented as a Petri net with no cases (the definition of the case is defined in Definition 1), which means the activities in an advice are not enabled. They can only be fired when the aspect is woven into the base process.

An aspect is an encapsulated entity of join points and advices:

*Definition 4.* (Aspect). An aspect is defined by a 2-tuple $s = (v, J)$, where:

(1) $v$ is an advice, which is the augmentation or constraint for the base processes $N$.

(2) $J$ is a set of join points. $\forall j \in J$ is a join point which represents the weaving position $jp$ and weaving type $e$ of an advice $v$.

*3.4 Weaving mechanism*

The weaving of aspects and base processes is specified by a weaving mechanism. There are two types of weaving in the weaving mechanism. Asymmetric weaving weaves aspects into base processes, while symmetric weaving weaves multi-aspects together when these aspects have same join points (Schauerhuber *et al.*, 2007). Given a base process $N$ and a set of aspects $S = \{s_1, s_2, \ldots, s_x\}$, $(x > 0)$. The weaving mechanism for weaving $S$ into $N$ is defined as the following steps:

Step1. Initialization: creating a weaving plan to store the join points of all aspects. Table I shows an example.

In this weaving plan, $s_1$ and $s_2$ have the same join point $jp_2$ and the same weaving type 1 (before weaving type in Definition 2), as shown encased in a dashed circle. These aspects will be woven together before weaving them into the base process. If these aspects have the

| Join points / Aspects | $jp_1$ | $jp_2$ | … | $jp_x$ |
|---|---|---|---|---|
| $s_1$ | 0 | 1 | | |
| $s_2$ | | 1 | | 2 |
| … | | | | |
| $s_y$ | | | | 3 |

**Table I.**
Weaving plan

same joint point but different composition type (as shown encased in a solid circle), they will be woven into the base process separately. Therefore, we have the following two steps.

Step2. When several aspects are woven at the same join point with the same weaving type, the execution orders of these aspects and the interdependencies among them are very important. Thus, the interdependencies among these aspects should be analyzed first and then based on the interdependence relations to weave them together. If there is no interdependence among these aspects, they can be woven in concurrent relations for higher efficiency.

Step3. After all aspects at the same join points with the same weaving types are woven together, base-aspect weaving weaves these aspects into the base processes. In this weaving, base-aspect correctness is analyzed and controlled.

Using these formal components in the meta-model, a woven aspect-oriented business processes are defined formally in the following:

*Definition 5.* (Aspect-oriented business process). An aspect-oriented business process is defined by a 4-tuple $W = (C, A; F, M_0)$ when a base process $N$ is woven by a set of aspects $S = \{s_1, s_2, \ldots, s_x\}$, $(x > 0)$:

(1) $W.C = N.C \cup s_1.C \cup s_2.C \cup, \ldots, \cup s_x.C$.

(2) $W.A = N.A \cup s_1.A \cup s_2.A \cup, \ldots, \cup s_x.A$.

(3) $W.F = N.F \cup s_1.F \cup s_2.F \cup, \ldots, \cup s_x.F$.

(4) $W.M_0 = N.M_0$.

In the following, we analyze the correctness of aspect-oriented business process modeling first and then propose correctness controlling method for preventing mistakes or errors in modeling.

## 4. Aspect-aspect weaving

Jointly deployed aspects may interact with each other. If not treated properly, interactions can give rise to interferences. Interferences are interactions that violate specified constraints (Kniesel, 2009; Kniesel and Bardey, 2006). The constraints of aspect-aspect interaction specify that the aspects must exhibit the specified effect and every effect must be applied only at the join points. For instance, logging activities of field accesses must be effective for all the matching field accesses in the woven process, while activity added by an aspect for the purpose of confirming field accesses must not show up where there are no filed assesses. Therefore, the errors or mistakes of aspect-aspect weaving are caused by the interferences among aspects. In the following section, we propose a method of weaving aspects together correctly.

### *4.1 Aspect-aspect interdependence*

Aspect interference is caused by the interdependencies among aspects. In 1966, Bernstein (Bergmans, 2003) stated a sufficient condition for the independence of two programs. He defined four types of dependence: true dependence, anti-dependence, output dependence, and control dependence. Bernstein's dependence relations are used to analyze concurrency between two programs. The presence of dependence implies that two programs cannot be executed concurrently. The fewer the dependencies are, the greater the concurrency is. In our previous research (Li, 2008), Bernstein's theory is extended to analyze interdependencies among activities to capturing concurrency in modeling software evolution processes. In this paper, Bernstein's dependence analysis is used for analyzing aspect-aspect interdependences.

In order to define dependence relations between aspects, we first define the input and output data set of the advice in an aspect:

*Definition 6.* (Input data set and output data set). Input ($v$) denotes the input data set of an advice $v$ and output ($v$) denotes the output data set of the advice $v$. Except for input ($v$) and output ($v$), all of the other data of the advice $v$ are local and have meaning only within the advice $v$.

*Definition 7.* (Data dependence). Let $s_1$ and $s_2$ be two aspects, for the advice $v_1$ of $s_1$ and the advice $v_2$ of $s_2$, suppose $v_1$ is executed before $v_2$:

(1) $s_2$ is true dependent on $s_1$ iff output ($v_1$) $\cap$ input ($v_2$) $\neq \varnothing$, which is denoted by $s_1\ \delta\ s_2$.

(2) $s_2$ is anti-dependent on $s_1$ iff output ($v_2$) $\cap$ input ($v_1$) $\neq \varnothing$, which is denoted by $s_1\ \bar{\delta}\ s_2$.

(3) $s_2$ is output dependent on $s_1$ iff output ($v_1$) $\cap$ output ($v_2$) $\neq \varnothing$, which is denoted by $s_1\ \delta^o\ s_2$.

True dependence, anti-dependence, and output dependence are defined as data dependence. Data dependence specifies that the execution of an advice should precede the execution of other advices.

Control dependence expresses conditional execution dependencies between advices. It specifies that an advice is conditionally executed depending on the execution result of another advice:

*Definition 8.* (Control dependence). Let $s_1$ and $s_2$ be two aspects, for the advice $v_1$ of $s_1$ and the advice $v_2$ of $s_2$, $s_2$ is control dependent on $s_1$, denoted by $s_1\ \delta^c\ s_2$, if whether $s_2$ can execute is determined by the execution result of $s_1$.

Data dependence and control dependence are two constraints of aspect-aspect interaction. In order to ensure the correctness of aspect-aspect weaving, it is essential to ensure that all aspects of data dependencies are woven in the order of dependencies and that all aspects of control dependencies are woven according to the control relations:

*Definition 9.* (Aspect-aspect correctness). Let $s_1$ and $s_2$ be two aspects in a set of aspects $S = \{s_i\}_{i \in \{1,\ldots,x\}}$, the correctness of aspect-aspect weaving is defined as follows:

(1) If the dependence between $s_1$ and $s_2$ is data dependence, they are woven in the order of data dependence.

(2) If the dependence between is control dependence, they are woven according to the control dependence.

For analyzing and describing the preceding dependence relations between aspects intuitively, an aspect dependence graph is constructed:

*Definition 10.* (Aspect dependence graph). An aspect dependence graph is a 2-tuple $G = (S, R)$ where $S = \{s_i\}_{i \in \{1,\ldots,x\}}$ is a set of aspects; $R \subseteq S \times S$ is an arc set; $R = \{\delta^d, \delta^c\}$ where $\delta^d$ is data dependence and $\delta^c$ is control dependence.

Figure 1 shows the examples of data dependence and control dependence. In the figure, "*d*" denote data dependence and "*c*" denotes control dependence.


**Figure 1.**
Dependences between two aspects

According to the dependence analysis, aspects at the same join points can be constructed into an aspect dependence graph and then transform this dependence graph into a Petri net. This Petri net represents an aspect that is woven correctly by the aspects at the same join points.

*4.2 Correct aspect-aspect weaving*

The following transformation rules are used to transform an aspect dependence graph into a Petri net. The result Petri net is an aspect that is woven by the aspects at the same join points based on the dependence relations.

Rule 4.1. Let $G = (S, R)$ be an aspect dependence graph and $I = (C, A; F)$ be a Petri net. Each aspect in $S$ is transformed into an activity in $A$.

Rules 4.2. Let $G = (S, R)$ be an aspect dependence graph and $I = (C, A; F)$ be a Petri net. If $R(s_i, s_j) = \delta^d$, then arc $(s_i, s_j)$ is transformed into a conditions $c_{ij}$ in $C$, arcs $(a_i, c_{ij})$ and $(c_{ij}, a_j)$ in $F$.

As shown in Figure 2, when weaving aspect $s_i$ and $s_j$, if $s_j$ is woven sequentially behind $s_i$, $s_j$ can exhibit its specified effect because $s_j$ has the input from $s_i$.

Rules 4.3. Let $G = (S, R)$ be an aspect dependence graph and $I = (C, A; F)$ be a Petri net. If $R(s_i,s_j) = \delta^c$ and $R(s_i, s_k) = \delta^c$, then two arcs $(s_i, s_j)$ and $(s_i, s_k)$ are transformed into a condition $c_i$ in $C$, arcs $(a_i, c_i)$, $(c_i, a_j)$ and $(c_i, a_k)$ in $F$.

If only one aspect $s_j$ is control dependent on aspect $s_i$, then the control dependence is still transformed into the selection structure. The difference is that the activity $a_k$ is replaced by a virtual activity. Virtual activity does nothing but transfers tokens.

Rules 4.4. Let $G = (S, R)$ be an aspect dependence graph and $I = (C, A; F)$ be a Petri net. If $R(s_j, s_i) = \delta^c$ and $R(s_k, s_i) = \delta^c$, then two arcs $(s_j, s_i)$ and $(s_k, s_i)$ are transformed into a condition $c_i$ in $C$, arcs $(a_j, c_i)$, $(a_k, c_i)$ and $(c_i, a_i)$ in $F$.

As shown in Figure 3(a), when weaving aspect $s_i$, $s_j$ and $s_k$, if they are woven in selection relation, the execution of $s_j$ and $s_k$ is determined by $s_i$, which means $s_i$ control the effect of them at the join points. Figure 3(b) shows the same correctness of aspect-aspect weaving.

## 5. Base-aspect weaving

Base-aspect weaving weaves the aspects into the base processes. Petri net is used not only as the modeling language for the specification of business processes but also for the correctness analysis techniques. In comparison with other process models, the major characteristics of Petri nets are as follows (Reisig, 1985):

(1) Causal dependencies and interdependencies in some set of events may be represented explicitly.

(2) Models may be represented at different levels of abstraction without having to change the description language.

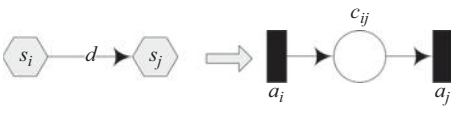



**Figure 2.**
Transformation of data dependences

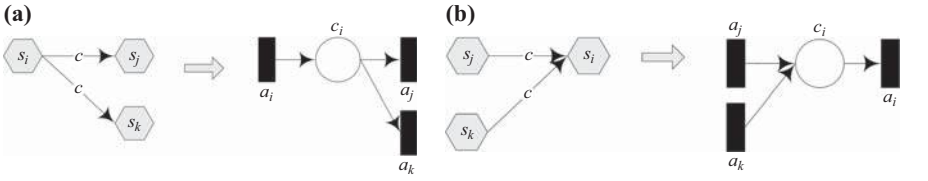



**Figure 3.**
Transformation of control dependences

(3) Petri Net representations make it possible to verify system properties and to do correctness proofs in a specific way.

Thus, according to the properties of Petri nets, when an aspect is woven into a base process, the errors or mistakes that may occur can be defined as the structural or dynamic property problems of Petri nets. Structural property of an aspect-oriented business process is determined by the structure of the process model. When an aspect is woven into a base process, if structural property problems arise, there must be static structural flaws existed in the woven model. Refer to the structural properties of Petri nets, correctness requirements for structural properties of the aspect-oriented process are no side condition, no isolated node, no deadlock, and no trap. When an aspect-oriented business process executes, its dynamic properties should be discussed. Dynamic properties are relevant to the case of the process model (the definition of the case is defined in Definition 1). Refer to the dynamic properties of Petri nets, correctness requirements for dynamic properties of the aspect-oriented process are safe, contact-free, persistent, and liveness.

### 5.1 Correctness analysis of base-aspect weaving

When an aspect is woven into a base process, errors or mistakes only affect the activities around the join point. Thus, in this paper, we define weaving process segment:

*Definition 11.* (Weaving process segment). Let $s = (v, J)$ be an aspect and $N = (C, A; F, M_0)$ be a base process. Suppose that $s$ is woven into $N$ at join point $jp \in JP$, a weaving process segment $L = (C', A'; F')$ is defined as follows:

(1) $L.A' = \{a | a \in {}^{\bullet}JP \cup JP^{\bullet} \cup JP \cup \mathrm{dom}(JP) \cup \mathrm{cod}(JP) \wedge a \in N.A\} \cup v.A_e \cup v.A_x$

(2) $L.C' = \{c | c \in L.{}^{\bullet}A' \cup L.A'^{\bullet} \wedge c \in N.C \cup v.C\}$

(3) $L.F' = \{f | f \in (L.C' \times L.A') \cup (L.A' \times L.C')\}$

A weaving process segment is a Petri net where the transitions are the activities around join points and the entrance activities and exit activities of the aspect. It is part of a woven aspect-oriented business process and the other part of the woven aspect-oriented business process does not need to be analysis because the errors or mistakes can only exist in this process segment. For example, an aspect $s = (\{c_{s1}, c_{s2}\}, \{a_{s1}\}; \{(c_{s1}, a_{s1}), (a_{s1}, c_{s2})\}), (N.a_{n1}, 2))$ is woven into a base process $N = (\{c_{n1}, c_{n2}, c_{n3}\}, \{a_{n1}, a_{n2}\}; \{(c_{n1}, a_{n1}), (a_{n1}, c_{n2}), (c_{n2}, a_{n2}), (a_{n2}, c_{n3})\}, \{c_{n1}\})$, the weaving process segment is $L = (\{c_{n2}, c_{n3}\}, \{a_{s1}, a_{n2}\}; \{(c_{n2}, a_{n2}), (a_{n2}, c_{n3}), (c_{n2}, a_{s1}), (a_{s1}, c_{n3})\})$, which is encased in the dashed rectangle in Figure 4.

In the following, correctness requirements for structural and dynamic properties are defined. We assume that all the base process models and aspects have been proved to be satisfied with these structural and dynamic properties. The correctness analysis is only focused on the weaving process segment.

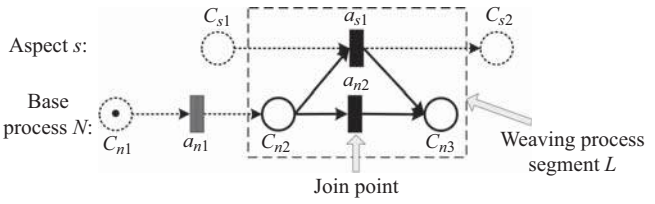



**Figure 4.** An example of weaving process segment

*5.1.1 Structural correctness*. Structural correctness of a business process model is determined by the topology structure properties of Petri nets (Bergmans, 2003):

*Definition 12*. (Structural correctness). Given an aspect $s=(v, J)$ and a base process $N=(C, A; F, M_0)$. When $s$ is woven into $N$, structural correctness of weaving process segment $L$ is defined as follows:

(1) If $\forall a \in L.A$ then ${}^{\bullet}a \cap a^{\bullet} = \emptyset$.

(2) If $\forall y \in L.C \cup L.A$ then ${}^{\bullet}y \cup y^{\bullet} \neq \emptyset$.

(3) If $C_1 \subseteq L.C$, there is no ${}^{\bullet}C_1 \subseteq C_1^{\bullet}$ or $C_1^{\bullet} \subseteq {}^{\bullet}C_1$.

In the Definition 12, the first correctness requirement means that an activity in the weaving process segment does not have side condition. If an activity has a side condition, it will not be fired forever. The second correctness requirement shows that isolated nodes are not allowed. The third correctness requirement means that the conditions in the weaving process segment do not have deadlock conditions (${}^{\bullet}C_1 \subseteq C_1^{\bullet}$) or trap conditions ($C_1^{\bullet} \subseteq {}^{\bullet}C_1$). A deadlock condition without token will not obtain token forever. A trap condition with token will not send the token out forever.

*5.1.2 Dynamic correctness*. Dynamic correctness of a business process model is relevant to the initial case of Petri nets (Reisig, 1985). In the following definition, we define dynamic correctness based on the case of Petri nets:

*Definition 13*. (Dynamic correctness). Given an aspect $s=(v, J)$ and a base process $N=(C, A; F, M_0)$. When $s$ is woven into $N$, dynamic correctness of weaving process segment $L$ is defined as follows:

(1) If $\forall a \in v.A$ and $\exists M \in \Sigma(N.M_0)$ then $M[a>$.

(2) If $\forall a \in L.A \cap N.A$, $\forall \sigma \in v.A^*$ and $\forall M \in \Sigma(N.M_0)$, $(M[a> \wedge M[\sigma > M') \rightarrow \neg M'[a>$ where $\sigma$ is an activity execution sequence of $N$.

(3) If $\forall c \in L.C$ and $B(c) = \min\{B | \forall M \in \Sigma(N.M_0): M(c) \leqslant B\}$ then $B(c)=1$.

(4) If $\forall c \in L.C$, $\forall a \in L.A$ and $\forall M \in \Sigma(N.M_0)$, there is no $c \in a^{\bullet} \cap M$ and ${}^{\bullet}a \subseteq M$.

Where $\Sigma(N.M_0)$ is a case set of $N$ that the initial case is $M_0$.

In the Definition 13, the first correctness requirement means that all activities in aspects are enabled. The second correctness requirement shows that the base process model is persistent which means that all the activities in the base process are enabled. Because the business processes in this paper are defined in elementary Petri net, conditions that are not safe or not contact-free are not allowed. Thus, the third correctness requirement means that all conditions in the weaving process segment are safe. The fourth correctness requirement shows that all conditions in the weaving process segment are contact-free. Safe and contact-free conditions make sure that the activities after these conditions can be enabled.

*5.1.3 Behavior correctness*. Proper modularization of multiple concerns into multiple aspects depends on the ability of aspects to interact with the base process in order to establish the desired overall behavior. When the aspects are woven into a base process, behavioral correctness is the behavior relations of the activities in the woven base process are consistent with the original behavior definitions in the base process specifications. Based on the definitions of the behavior relations of activities in the software evolution process proposed by Li (2008), the behavior of an aspect-oriented business process model is composed of four relations: sequence, selection, concurrency, and iteration. When aspects are woven into a basic process, behavior correctness postulates that these

behavior relations will not be changed. In the following, these behavior relations are defined as:

*Definition 14.* (Sequence behavior). Given a based process $N=(C, A; F, M_0)$, $a_\alpha$, $a_\beta \in A$, $M \in \Sigma(M_0)$, the behavior relation of $a_\alpha$ and $a_\beta$ is sequence behavior if:

(1) $M[a_\alpha >$ but $\neg M[a_\beta >$.

(2) For $\forall \sigma \in A^*$, $M[a_\alpha > M'[\sigma > M'' \rightarrow \neg M''[a_\beta >$.

If the length of $\sigma$ is 0, the behavior relation of $a_\alpha$ and $a_\beta$ is the direct sequence behavior, else is the indirect sequence behavior.

*Definition 15.* (Selection behavior). Given a based process $N=(C, A; F, M_0)$, $a_\alpha$, $a_\beta \in A$, $M \in \Sigma(M_0)$, the behavior relation of $a_\alpha$ and $a_\beta$ is selection behavior if:

(1) For $\forall \sigma \in A^*$, there are $M[a_\alpha >$ and $M[\sigma >$ or $M[a_\beta >$ and $M[\sigma >$.

(2) $M[\sigma > M_1[a_\alpha > M_2 \rightarrow \neg M_1[a_\beta > \wedge \neg M_2[a_\beta >$ but $M[a_\beta > M' \rightarrow \neg M'[\sigma \wedge \neg M'[a_\alpha >$ or $M[\sigma > M_3[a_\beta > M_4 \rightarrow \neg M_3[a_\alpha > \wedge \neg M_4[a_\alpha >$ but $M[a_\alpha > M'' \rightarrow \neg M''[\sigma \wedge \neg M''[a_\beta$.

If the length of $\sigma$ is 0, the behavior relation of $a_\alpha$ and $a_\beta$ is the direct selection behavior, else is the indirect selection behavior.

*Definition 16.* (Concurrency behavior) Given a based process $N=(C, A; F, M_0)$, $a_\alpha$, $a_\beta \in A$, $M \in \Sigma(M_0)$, the behavior relation of $a_\alpha$ and $a_\beta$ is concurrency behavior if:

(1) For $\forall \sigma \in A^*$, there are $M[a_\alpha >$ and $M[\sigma >$ or $M[a_\beta >$ and $M[\sigma >$.

(2) $M[\sigma > M_1 \rightarrow M_1[a_\alpha > \wedge M_1[a_\beta >$ and $M_1[a_\alpha > M' \rightarrow \neg M'[a_\beta >$ or $M_1[a_\beta > M'' \rightarrow \neg M''[a_\alpha >$, or $M[a_\alpha > M_2[\sigma > M_3 \rightarrow \neg M_3[a_\beta >$, or $M[a_\beta > M_4[\sigma > M_5 \rightarrow \neg M_5[a_\alpha >$.

If the length of $\sigma$ is 0, the behavior relation of $a_\alpha$ and $a_\beta$ is the direct concurrency behavior, else is the indirect concurrency behavior.

*Definition 17.* (Iteration behavior). Given a based process $N=(C, A; F, M_0)$, $a_\alpha$, $a_\beta \in A$, $M \in \Sigma(M_0)$, the behavior relation of $a_\alpha$ and $a_\beta$ is iteration behavior if:

(1) $M[a_\alpha >$ but $\neg M[a_\beta >$.

(2) For $\forall \sigma \in A^*$, $M[a_\alpha > M'[\sigma > M'' \rightarrow \neg M''[a_\beta > M'''[a_\alpha >$.

If the length of $\sigma$ is 0, the behavior relation of $a_\alpha$ and $a_\beta$ is the direct iteration behavior, else is the indirect iteration behavior.

Based on the preceding definitions of the behavior relations, when the aspects are woven into a basic process, the sequence, selection, concurrency, or iteration behavior relations between the base process activities should not be changed. Therefore, behavior correctness is defined as:

*Definition 18.* (Behavior correctness). An aspect is woven into a base process to keep the behavior correctness if and only if the behavior relations of the base process activities in the weaving process segment is consistent with the original behavior relations in the base process specification.

### *5.2 Correct base-aspect weaving*

All the possible base-aspect weaving structures can be summarized into ten types. They are before condition weaving, after condition weaving, around condition weaving, iteration weaving, concurrency weaving, before activity weaving, after activity weaving, around activity weaving, condition-activity flow weaving, and activity-condition flow weaving. Based on the correctness Definitions 12 and 13, following aspect weaving operations

are analyzed. Here, we assume that an aspect $s = (v, J)$ is woven into a base process model $N = (C, A; F, M_0)$:

(1) The join point of condition-activity flow weaving is defined as the flow relation that connects a condition and an activity. The weaving type *is around type*, as shown in Figure 5, $j = (N.(c, a), 2)$. The result of this weaving is to add the advice $v$ before the activity $a$ in the base process sequentially.

(2) Similar to the condition-activity flow weaving, activity-condition flow weaving adds the advice $v$ after the activity $a$ in the base process sequentially, as shown in Figure 6, $j = (N.(a, c), 2)$.

(3) Before condition weaving weaves the aspect before a condition join point, as shown in Figure 7, $j = (N.c, 0)$, the join point is condition $c$ in base process $N$ and the weaving type is before weaving. The result of this weaving is to add the advice $v$ before the condition $c$. Nevertheless, based on Definition 13, this weaving may cause a safe problem (the third correctness requirements in Definition 13) in $c$ or make the activities in aspect cannot be fired (the first correctness requirements in Definition 13). Thus, it is not allowed and we use activity-condition flow weaving to replace it as they have the same effect on weaving.

(4) The weaving position of the after condition weaving is opposite to the before condition weaving, as shown in Figure 8, $j = (N.c, 1)$, the join point is condition $c$ in base process $N$ and the weaving type is after weaving. This weaving may cause the persistent problem (the second correctness requirements in Definition 5.3) in the base process model. Thus, we use condition-activity flow weaving to replace it as they have the same effect on weaving.

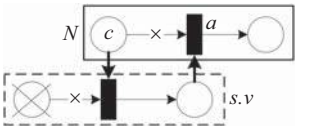



**Figure 5.**
Condition-activity flow weaving

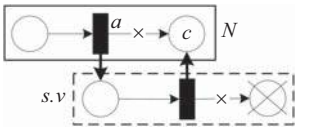



**Figure 6.**
Activity-condition flow weaving

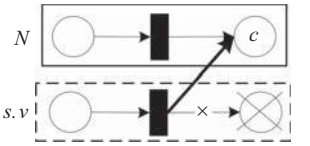



**Figure 7.**
Before condition weaving

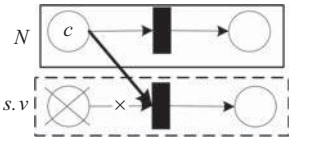



**Figure 8.**
After condition weaving

(5) Around condition weaving weaves the advice $v$ between the preset activities and the post activities of the join point condition $c$, as shown in Figure 9, $j = (N.c, 2)$. The result of this weaving is to insert the advice $v$ between two activities in the base process model.

(6) Before activity weaving weaves the aspect before an activity join point, as shown in Figure 10, $j = (N.a, 0)$, the join point is activity $a$ in base process $N$ and the weaving type is before weaving. The original intention of this weaving is to add restriction to the activity $a$ in the base process $N$ that the execution of $a$ must wait for the execution of the aspect. However, when the activity $a$ executes repeatedly in an iterative structure, this weaving will forbid its execution (the persistent problem in Definition 5.3). Therefore, we use condition-activity flow weaving to replace it as they have the same effect on weaving.

(7) After activity weaving is similar to the before activity weaving, but it has the opposite weaving type, as shown in Figure 11, $j = (N.a, 1)$, the join point is activity $a$ in base process $N$ and the weaving type is after weaving. This weaving may cause contact problem (the fourth correctness requirements in Definition 5.3) when $a$ is in an iterative structure. Therefore, it is replaced by the activity-condition flow weaving.

(8) Around activity weaving inserts the advice into the base process model to construct selection relation between the advice and the join point activities, as shown in Figure 12, $j = (N.a, 2)$. The join point activity $a$ may not execute if the advice is chosen to be executed. Thus, the advice must address its own requirements and also the requirements of the join point activity $a$.

(9) Iteration weaving weaves the advice into the base process to construct iterative relation with the join point activity, as shown in Figure 13, $j = (N.a, 3)$. When the join

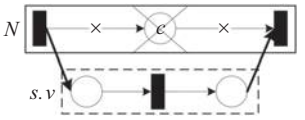



**Figure 9.**
Around condition weaving

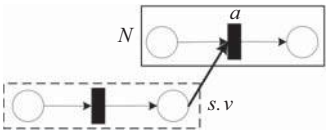



**Figure 10.**
Before activity weaving

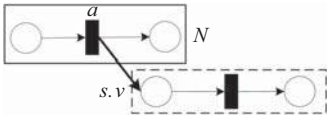



**Figure 11.**
After activity weaving

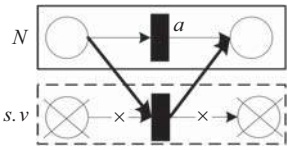



**Figure 12.**
Around activity weaving

point activity $a$ has iteration relation with another activity $b$, the added advice will have selection relation with $b$. similar to the around activity weaving, activity $b$ may not be chosen to be executed, thus, the advice must satisfy the requirements of the activity $b$.

(10) Concurrency weaving weaves the advice into the base process to construct concurrent relation with the join point activity, as shown in Figure 14, $j = (N.a, 4)$.

Based on the above analysis, only six weaving operations are used because the others violate the correctness definitions of weaving and can be replaced. These six weaving operations are condition-activity flow weaving, activity-condition flow weaving, around condition weaving, around activity weaving, iteration weaving, and concurrency weaving.

### *5.3 Aspect-oriented business process modeling tool*

For the aid of aspect-oriented business process modeling, trustworthy process aided tool (TPAT) is designed and developed in the open source software PIPE (Sourceforge, 2013) with plug-in technology. Based on the aspect-aspect weaving interdependence (see Section 4.2) and base-aspect weaving structure (see Section 5.2), TPAT is used to weave the aspects together first and then weave them into the base processes. The relationships of the TPAT modules are shown in Figure 15.

TPAT consists of two core components: the aspect-oriented extension component and the base process, aspect definition component. The aspect-oriented extension component consists of three core modules: aspect-aspect weaving, base-aspect weaving, and correctness definition. All the definitions and weaving data are written in a configuration file.

To model an aspect-oriented business process, TPAT works as follows and is illustrated in Figure 16.

- The base processes and the advices of the aspects are created in PIPE, as shown in Screen 1 and Screen 2.
- According to the definitions of the base processes and the aspects, a weaving plan is created and stored in a configuration file. This weaving plan is created for initialization of aspect-oriented extension, as introduced in weaving mechanism in Section 3.4. Then, by clicking TPAT in Screen 3, aspect weaving is started.
- For the aspects that are woven at the same join point with the same weaving type, abiding by the aspect-aspect correctness and the transformation rules, aspect dependence graphs are created first and then transformed into Petri nets.
- At each of the join points, according to the base-aspect weaving structure, aspects are woven into the base processes. Finally, new aspect-oriented business processes are created and stored, as shown in Screen 4.

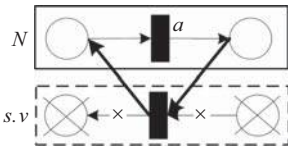



**Figure 13.**
Iteration weaving

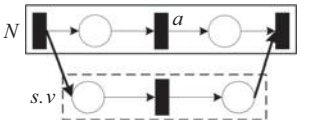



**Figure 14.**
Concurrency weaving

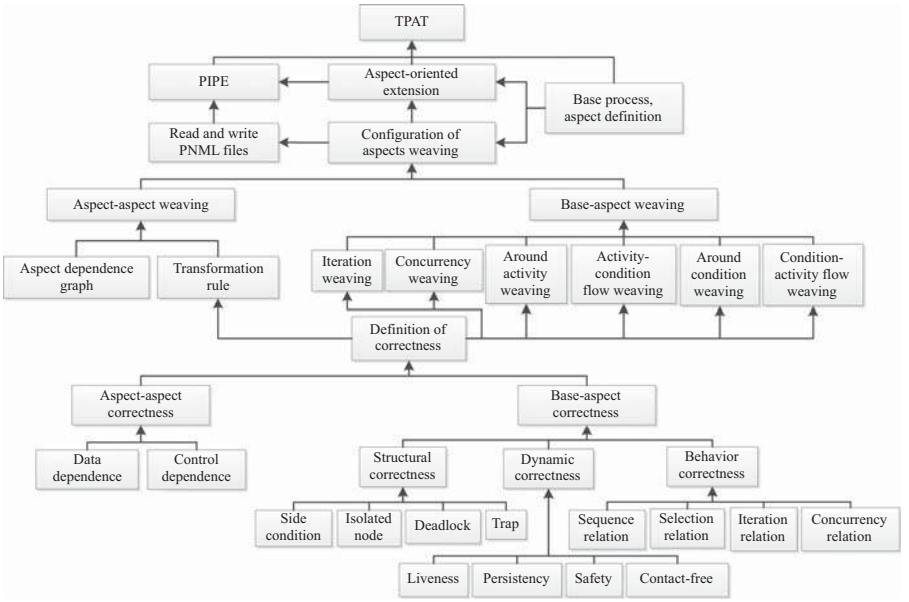



**Figure 15.**
TPAT module design

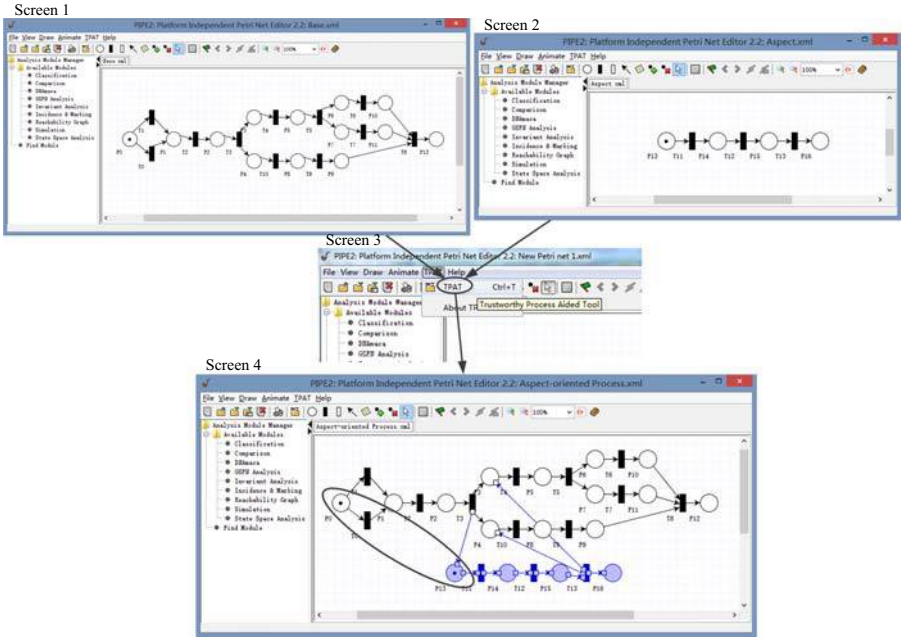



**Figure 16.**
Aspect-oriented business process modeling tool

## 6. Case study

The proposed method was applied to a systematic study on the modeling of a real-world banking process model, in order to prove the validity of the approach. The case study was conducted at the Yunnan Financial Engineering Institute and the aim was to improve the

maintainability of the banking processes and made the processes more flexible and effective. Since banks have little room for error, the correctness of their business processes becomes very important. Accordingly, the case study was designed in terms of the real needs and a project team was formed. The team consisted of the authors of this paper and some technical and management personnel from the local banks. With the support of the employees from the banks, aspect-oriented and correctness controlling method were used in the banking processes and the performance assessments were analyzed.

The model in the case study comprises a change asset deal process, a deal for speculation process, a bank draft deal process, and a letter of credit deal process. The change asset deal and the deal for speculation processes are taken from a bank which has more than 1,000 branches (Jalali, 2011). These two processes are integrated into one Petri net in Figure 17. $VA_i$ ($i \in \{1, 2, \ldots, x\}$, $x > 0$) in the process denotes the virtual activities which do nothing but passing tokens from one condition to another. Observing these two processes (see Figure 17), there are many cross-cutting activities which can be separated from the core activities.

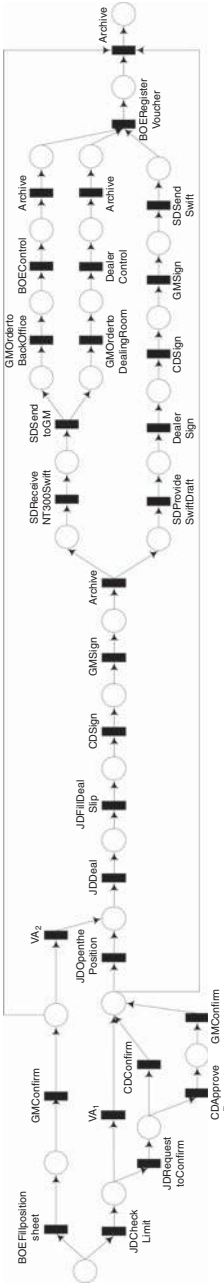
As introduced in Section 1, several concerns, such as compliance, auditing, monitoring, accounting, billing, authorization, privacy, and separation of duties, are highly repeated in different business process models and should be separated. Using the heuristics in Cappelli *et al.* (2010), confirmation, signature, control, check, and archiving are identified as cross-cutting concerns that should be separated from the core activities. Figure 18 shows the result of their separation. The left Petri net in Figure 18 shows the base process which is constructed by the core activities. The Petri nets, labeled from $s_1$ to $s_{11}$ in the right of Figure 18, represent the cross-cutting aspects that are separated from the base process.

Comparing two process models in Figures 17 and 18, the aspect-oriented process model reduces the complexity and makes the cross-cutting and core concerns more clearly. The maintainability of the whole process models is also increased since the modifications of the base process or aspects do not affect each other. In addition, SoC increases the reusability as the base process and aspects can be used in different bank branches or even different financial organizations. In the following, the reusability of the four banking processes is analyzed.

The change asset deal process makes a deal to exchange an amount of money from one currency to another in the banking domain. The whole process is triggered by a back office employee filling in a position sheet. Then, if the sheet is denied by the general manager (GM), the sheet is archived and the process is terminated. If the position sheet is approved, a junior dealer (JD) fills in a deal slip. After the deal slip is signed by the chief dealer (CD) and the GM, it will be archived and two parallel sets of activities are performed. In these two parallel sets of activities, the dealt amount of money is sent to the Swift department. On one hand, an employee of the Swift department provides a swift draft for sending the money. Then, the dealer, CD, and GM sign the swift draft. In parallel, an NT300 swift message is sent to the GM from the Swift department. The GM makes an order to the back office department. When the order has been controlled, the messages are archived and a back office employee registers a voucher in the accounting system. Finally, the deal is archived.

The deal for speculation process is begun at checking the limit on the amount of money that JD can deal. If the amount exceeds the limit, the JD needs permission from the CD or even the GM. If the CD or the GM denies the request, the process ends. Otherwise, JD proceeds the deal by opening a position. Afterward, the process continues in the same as the change asset deal process. Since most of the same base process and cross-cutting aspects of the change asset deal process are used in the deal for speculation process, these concerns could be reused.

In addition, the bank draft deal process and the letter of credit deal process could reuse the same base process and aspects. A bank draft is a check drawn on a bank's funds and is




**Notes:** BOE, back office employee; JD, junior dealer; CD, chief dealer; GM, general manager; SD, swift department

**Figure 17.**
Original banking business process model

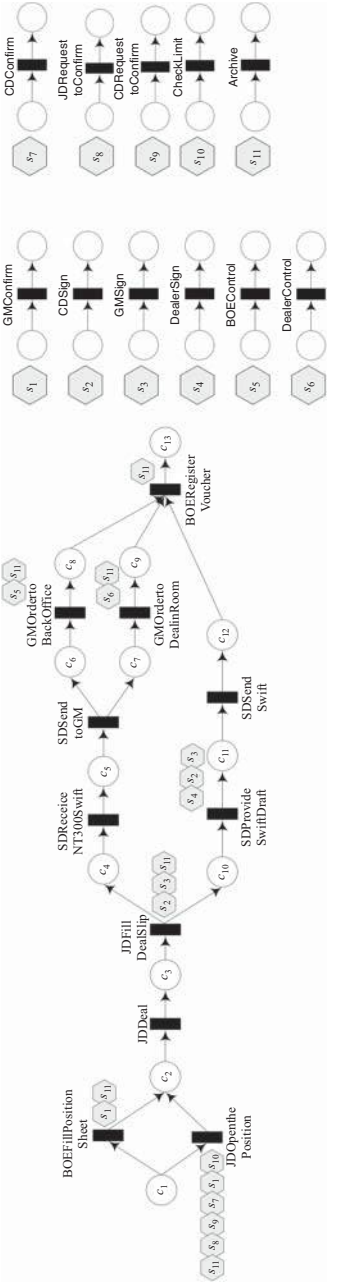



**Figure 18.**
Aspect-oriented business process models

guaranteed by the bank that issues it. The deal process of a bank draft begins at a drawer (the person or organization who wants to make a payment), submitting the application and providing the bank account information. The bank verifies the application. If the application is rejected, the process ends. Otherwise, the application is submitted to a JD to fill in a deal slip. The subsequent activities are the same as in the change asset deal process. However, the legitimation of a bank draft and the credit guarantee of the drawer are required. Comparing to the change asset deal process, more cross-cutting concerns, such as tracking and auditing are also needed. Therefore, the base process is reused but more aspects will be designed. The fourth process – letter of credit deal process is the process commonly for international trade but also used in domestic transactions (such as construction projects). The preceding base process and the cross-cutting concerns can also be reused in this process.

After defining the base processes and cross-cutting aspects, the banking processes can be modeled by aspect-oriented method by continually reusing similar base processes and cross-cutting aspects. SoC has lots of benefits, but as mentioned before, if multiple aspects are needed to be woven at the same join points, dependence relations between them should be analyzed and used to guide their weaving. When the aspects are woven into the base process, base-aspect correctness should be ensured by only using the correct base-aspect weaving operations in Section 5.2. In the following, the change asset deal process and the deal for speculation process are analyzed to show the details of aspect-oriented modeling and correctness controlling. Then, all the four processes are assessed in the reuse rate, the increased lines of code, and the correctness detection performance.

### *6.1 Aspect-aspect correctness*

As shown in Figure 17, there are seven join points in the base process. They are JDFillDealSlip, SDProvideSwiftDraft, GMOrdertoBackOffice, GMOrdertoDealingRoom, BOEFillpositionSheet, JDOpenthePosition, and BOERegisterVoucher. Six of these join points, except BOERegisterVoucher, have multiple aspects. As analyzed in Section 4, aspect interdependences can be divided into two categories: data dependence and control dependence. Aspects at JDFillDealSlip, SDProvideSwiftDraft, GMOrdertoBackOffice, and GMOrdertoDealingRoom only have data dependence relations. Aspects at BOEFillpositionSheet and JDRequesttoConfirm have both control and data dependence relations.

According to the dependence definitions and transformation rules in Section 4, data dependence analysis and transformation at JDFillDealSlip is illustrated in Figure 19, control and data dependence analyses and transformations at JDOpenthePosition are illustrated in Figure 20.

At join point JDFillDealSlip, after Junior Dealer fills the deal slip, CD signs the deal slip first and then GM can sign it. Archiving can only be executed after all these signatures finished. As shown in Figure 19, CDSign, GMSign, and Archive aspects are woven sequentially.

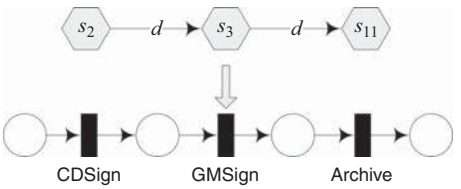



**Figure 19.**
Data Dependent Aspects at JDFillDealSlip

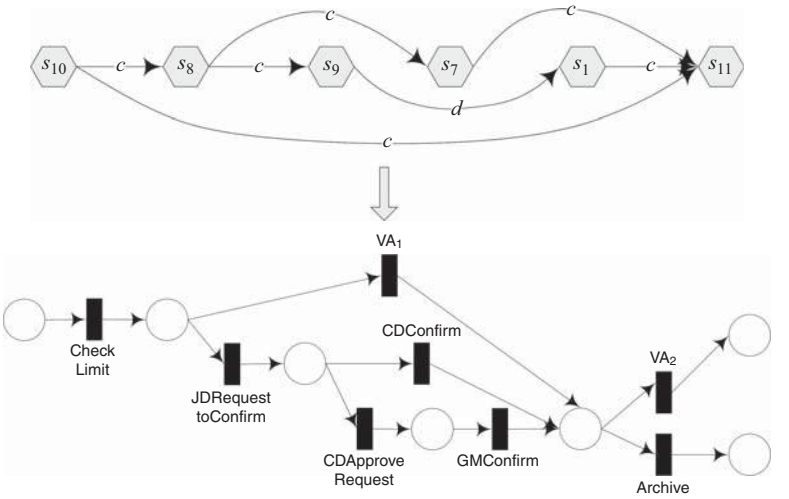



**Figure 20.**
Control Dependence Aspects at JDOpenthePosition

At join point JDOpenthePosition, as described above, the deal for speculation process is begun at JD to check the limit on the amount of money that JD can deal. If the amount exceeds the limit, JD needs permission from the CD or even the GM. As shown in Figure 20, Petri net transformed from the aspect dependence graph is conformity with this situation. In this Petri net, $VA_i$ ($i \in \{1, 2, \ldots, x\}$, $x > 0$) denotes the virtual activities which do nothing but for passing tokens from one condition to another condition.

If there is no interdependence between the aspects at the same join point, they can be woven in concurrent relation. After all aspects are woven at the same join point, these aspects will be woven into the base process. As mentioned above, base-aspect correctness should be controlled.

### *6.2 Base-aspect correctness*

When an aspect is woven into the base process model, correctness between the aspects and the base model should be controlled. The following Figures 21 and 22 show the examples. For simplicity, we only show the joint point of the base process model in the figures. In Figure 21, aspects $s_{11}$, $s_8$, $s_9$, $s_7$, $s_1$, and $s_{10}$ are woven together first (see Figure 20) and then woven into the base process model before join point activity JDOpenthePosition. If we use before activity weaving, as illustrated in Figure 21(a), the woven process model will never be executed because the aspect cannot be fired and the base process will never get the token from the aspect and cannot be fired. As discussed in Section 5.2, this weaving operation is replaced by the condition-activity flow weaving, as shown in Figure 21(b), the woven process can be fired and execute correctly.

After the aspects at join points BOEFillPositionSheet, JDOpenthePosition, and JDFillDealSlip are woven, the woven process model is illustrated in Figure 22. All the other aspects' weavings are the same.

The results of this case study show that aspect-oriented business process modeling has three advantages:

- The separation of the base processes and the cross-cutting concerns shows advantages in reuse and customization, and also help reduce the duplications of process models in the business process model library. The complicated problem of managing large number of similar models is resolved too.

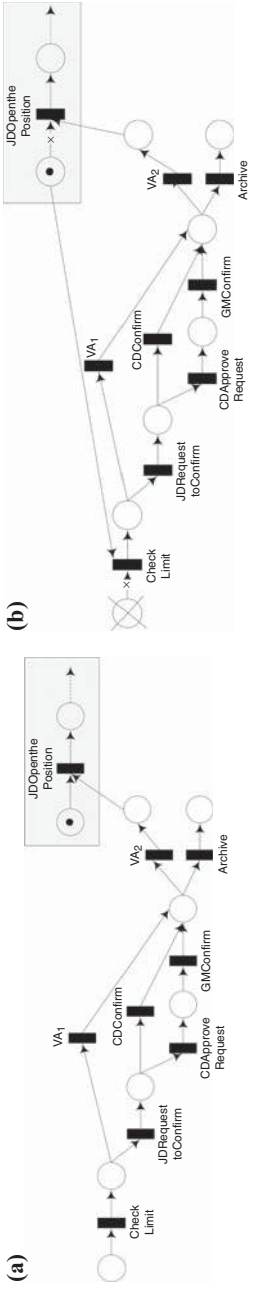



**Figure 21.**
Base-aspect correctness at join point JDOpenthePosition

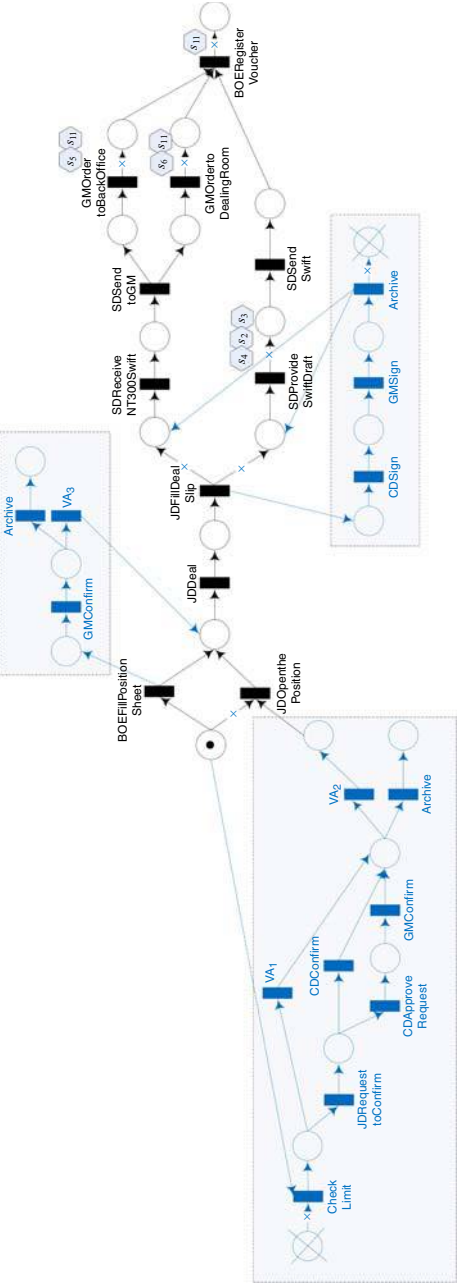



**Figure 22.**
Aspects are woven into the base process at three join points

- According to the different business applications requirements, aspect-oriented business process modeling shows its flexibility in weaving different aspects into different base processes on-demand. The resulting models are easy for process improvements.
- The correctness criteria for aspects weaving that are used in the procedure of business process modeling not only guarantee the correctness of business process models but also reduce the cost, labor, and time lost in correcting errors in process models.

*6.3 Performance assessment*

Through the preceding modeling, aspect-oriented banking process models are created. However, in practice, how much the reuse rate of process models will be increased by separating the cross-cutting activities and the core activities? Whether the correctness control in process modeling will affect the overall performance and efficiency of business process modeling? Will the aspect-oriented business process modeling increase the lines of code? These questions must be answered to demonstrate that it is rational practice to adopt the approach based on its pragmatic success. In the following, three metrics are designed to assess the performance and workload of the aspect-oriented business process modeling approach. They are the reuse rate, the increased lines of code, and the correctness detection performance.

In order to calculate the reuse rate, four processes in the banking business process model are assessed, as shown in Table II, $w_1$, $w_2$, $w_3$, and $w_4$ are aspect-oriented deal for speculation process, change asset deal process, bank draft deal process, and letter of credit deal process. The reuse rates shown in Table II are the comparison results of these four banking business processes. Since the base process of bank clearing in all banking business processes are reusable and this clearing process consists of nine core activities, the number of reusable activities in the base process is at least nine. For instance, in the deal for speculation process and the change asset deal process, except the different initial activities, all the other 14 activities are the same, therefore, the reuse rate of them is the highest.

Aspects in these four banking business processes include security aspect, log aspect, tracing aspect, archiving aspect, and audit aspect. Archiving aspect and audit aspect are both woven into four banking processes. In addition, based on the business requirements of these four banking processes, security aspect and log aspect are woven into the deal for speculation process and the change asset deal process, all the aspects are woven into the bank draft deal process, and the security aspect and tracing aspect are woven into the letter of credit deal process. Therefore, the reuse rates of these aspects are calculated and the results are shown in the last column of Table II.

Certainly, not all of the base processes can be reused. For instance, the loan process cannot reuse any base process of the preceding four banking processes because all their core activities are different. But, most aspects can be reused. Therefore, the aspect-oriented modeling approach can increase the reuse rate. However, extra data, such as the aspect

| Business process models | Reused activities in base processes | Reuse rate of base processes (%) | Reused activities in aspects | Reuse rate of aspects (%) |
|---|---|---|---|---|
| $w_1 w_2$ | 14 | 90 | 10 | 95 |
| $w_1 w_3$ | 11 | 58 | 10 | 83 |
| $w_1 w_4$ | 12 | 60 | 9 | 90 |
| $w_2 w_3$ | 10 | 58 | 10 | 88 |
| $w_2 w_4$ | 9 | 60 | 9 | 95 |
| $w_3 w_4$ | 11 | 48 | 9 | 84 |

**Table II.** Reuse rate

definitions and the aspect weaving configuration information may need more storage. In the following, the increasing of the lines of code is analyzed.

Table III shows the comparison analysis of the deal for speculation process model and four banking processes models. As shown in the first line of Table III, the increasing rate of code for the aspect-oriented deal for speculation process model is 33.8 percent. But the second line shows the total lines of code for all four banking process models decreased because of the reuse of the base processes and aspects. Therefore, although the use of the aspect-oriented modeling approach will increase the lines of code for the single model, but the reuse of the base processes and aspects will finally reduce the total lines of code.

For the third metrics, the correctness detection performance is used to evaluate the time consume of the correctness detection. Compared with the non-aspect-oriented business process modeling, correctness detection is executed in the procedure of aspect-aspect weaving and base-aspect weaving. Table IV presents a comparison of the correctness detection performance of the aspect-oriented and non-aspect-oriented modeling.

As seen from the comparison of the detection time performance in Table IV, the detection time for all models is decreased, in which the detection time of the bank draft deal process decreased more than the other processes. Two factors impact this result, one is the size of the model and another is the reuse rate. The size of the bank draft deal process model is relatively larger than the other process models and the complexity of the models is reduced more by separating cross-cutting concerns. In addition, the reuse rate also impacts the decrease of the detection time.

After the aspect-oriented modeling and performance assessments we discussed with all the participants in the project team to collect data about the impressions of building models and the approach. All participants highlighted that aspect-oriented approach provided a clear and explicit structure for their business processes and the separation of the base processes and aspects made the model reusable and the maintenance work easily. Another important highlight was the ensuring correctness of process models is very important for

**Table III.**
Increased lines of code

| Business process models | Evaluation metrics | Non-aspect-oriented business process model | Aspect-oriented business process model | |
|---|---|---|---|---|
| | | Lines of code | Increased lines of code | Increased rate (%) |
| $w_1$ | Lines of code | 1,597 | 815 | 33.8 |
| $w_1 w_2$ | Lines of code | 7,516 | 3,161 | 29.6 |
| $w_3 w_4$ | XML tags | 50 | 10 | 16.7 |

**Table IV.**
Correctness detection performance

| Business process models | Non-aspect-oriented busienss process model | Aspect-oriented busienss process model | | |
|---|---|---|---|---|
| | | Detection for aspect-aspect weaving | Detection for base-aspect weaving | |
| | Time (ms) | Time (ms) | Time (ms) | Increasing time (%) |
| $w_1$ | 2801.7 | 1259.1 | 1510.2 | −3.3 |
| $w_2$ | 2787.6 | 1171.7 | 1503.0 | −4.05 |
| $w_3$ | 3125.4 | 1024.2 | 1779.8 | −10.28 |
| $w_4$ | 2560.1 | 991.7 | 1477.9 | −3.54 |

their critical business processes. However, only four processes in this model were analyzed. In order to prove the methods could be applied in all banking context, more studies on other business processes in banks should be conducted in the future.

## 7. Conclusions

In the past, the cross-cutting concerns have been modeled as the integral parts of the business processes. This often leads to complexity, inflexibility, and less reusability. Aspect-orientation is a software engineering method that enables explicit separation and encapsulation of concerns (Mehmood and Jawawi, 2013). In this paper, based on the aspect-oriented concepts, an approach for aspect-oriented business process modeling is proposed. In the approach, Petri net is used not only as the design language for the specification of business processes but also as the analysis techniques that can effectively verify the correctness of business processes models.

The implications of the proposed approach include both the research and the practice perspectives.

In the research perspective, a method to modeling aspect-oriented business process is proposed. In this method, Petri net is used as the modeling language. Based on this modeling language, join point, advice, aspect, and weaving mechanism are defined. Furthermore, the correctness of business process modeling is addressed to ameliorate the quality of the models. For multiple aspects at the same join points, aspect-aspect correctness is controlled by analyzing the interdependencies among aspects. According to the interdependencies among aspects, a method of constructing aspect dependence graph is presented and the transforming from an aspect dependence graph to a Petri net is proposed. The transformed Petri net is the aspect that the aspect-aspect correctness has been controlled. When aspects are woven into base process models, weaving correctness of the aspects is analyzed by using structural and dynamic properties of Petri net. Based on the weaving correctness, six correct weaving operations are defined and used to control the base-aspect weaving correctness.

In the practice perspective, designing correct aspect-oriented business process models helps organizations reusing the model elements to reduce redundancy of their model repository and ensuring the correctness of their business processes. It also has the potential to support organizations to adapt to the changes of business requirements with flexible modeling. By doing so, they will have better maintainability for their business process models.

Our approach has made significant progress toward ensuring correctness of aspect-oriented business process modeling, positive results are also obtained in our case study, but further validation should be conducted in more variety of realistic cases and the effects on the economy and commercial should also be analyzed. In addition, many other Business Process Modeling Languages has been well-established, such as BPMN, UML 2 Activity Diagram (AD), Business Process Definition Metamodel, Event Driven Process Chain, Integrated DEFinition Method 3, and Role Activity Diagram (List and Korherr, 2006). Proposing a generic aspect-oriented business process modeling approach and ensuring correctness in these languages is another future work.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61502413, 61262025, 61379032, 61662085; Natural Science Foundation of Yunnan Province under Grant No. 2016FB106; Science Foundation of Yunnan Educational Committee under Grant No. 2015Z020; Science Foundation of Key Laboratory of Software Engineering of Yunnan Province under Grant No. 2015SE202; Data-Driven Software Engineering Innovative Research Team Funding of Yunnan Province; Software Engineering Innovative Research Team Funding of Yunnan University.

## References

Aksit, M., Bergmans, L. and Vural, S. (1992), "An object-oriented language-database integration model: the composition-filters approach", *Proceedings of the 6th European Conference on Object-Oriented Programming, Utrecht, June 29-July 3*, pp. 372-395.

Amyot, D. (2013), "Goal and aspect-oriented business process engineering", February, available at: www.cs.mcgill.ca/~joerg/SEL/AOM_Bellairs_2013_-_ Schedule_files/Daniel.pdf

Amyot, D. and Mussbacher, G. (2011), "User requirements notation: the first ten years, the next ten years (invited paper)", *Jounal of Software*, Vol. 6 No. 5, pp. 747-768.

Anis, C., Müller, H. and Mezini, M. (2010), "Aspect-oriented business process with AO4BPMN", in Anis, C., Müller, H. and Mezini, M. (Eds), *Modelling Foundations and Applications*, Springer, Berlin Heidelberg, pp. 48-61.

Becker, J., Rosemann, M. and von Uthmann, C. (2000), "Guidelines of business process modeling", in van der Aalst, W., Desel, J. and Oberweis, A. (Eds), *Business Process Management*, Springer, Berlin Heidelberg, pp. 30-49.

Bergmans, L. (2003), "Towards detection of semantic conflicts between crosscutting concerns", Workshop on Analysis of Aspect-Oriented Software, Darmstadt, July 21.

Bi, H.H. and Zhao, J.L (2004), "Process logic for verifying the correctness of business process models", *Proceedings of the Twenty-fifth International Conference on Information Systems*, pp. 91-100.

Blair, G.S., Blair, L., Rashid, A., Moreira, A., Araújo, J. and Chitchyan, R. (2005), "Engineering aspect-oriented systems", in Filman, R.E., Elrad, T., Clarke, S. and Aksit, M. (Eds), *Aspect-Oriented Software Development*, Addison-Wesley, Boston, MA, pp. 379-406.

Cappelli, C., Santoro, F.M., Leite, J.C.S.P., Batista, T., Medeiros, A. and Romeiro, C. (2010), "Reflections on the modularity of business process models: the case for introducing the aspect-oriented paradigm", *Business Process Management Journal*, Vol. 16 No. 4, pp. 662-687.

Charfi, A. and Mezini, M. (2007), "AO4BPEL: an aspect-oriented extension to BPEL", *World Wide Web Journal: Recent Advances on Web Services (Special Issue)*, Vol. 10 No. 3, pp. 309-344.

Charfi, A., Muller, H. and Mezini, M. (2010), "Aspect-oriented business process modeling with AO4BPMN", in Kühne, T., Selic, B., Gervais, M.P. and Terrier, F. (Eds), *Modelling Foundations and Applications*, Lecture Notes in Computer Science, Vol. 6138, Spinger, Heidelberg, pp. 48-61.

Constantinides, C.A, Bader, A. and Elrad, T. (1999), "An aspect-oriented design framework for concurrent systems", *Proceedings of the ECOOP'99 Workshop on Aspect-Oriented Programming, Lisbon*, pp. 302-311.

Dijkman, R., La Rosa, M. and Reijers, H.A. (2012), "Managing large collections of business process models: current techniques and challenges", *Computers in Industry*, Vol. 63 No. 2, pp. 91-97.

Dijkman, R.M., Dumas, M. and Ouyang, C. (2008), "Semantics and analysis of business process models in BPMN", *Information and Software Technology*, Vol. 50 No. 12, pp. 1281-1294.

Dijkstra, E.W. (1976), *A Discipline of Programming*, Prentice-Hall, Englewood Cliffs, NJ.

Dinkelaker, T., Erradi, M. and Ayache, M. (2012), "Using aspect-oriented state machines for detecting and resolving feature interactions", *Computer Science and Information Systems*, Vol. 9 No. 3, pp. 1045-1074.

Douence, R., Fradet, P. and Südholt, M. (2002), "A framework for the detection and resolution of aspect interactions", in Batory, D., Consel, C. and Taha, W. (Eds), *Generative Programming and Component Engineering*, Springer, Berlin Heidelberg, pp. 173-188.

Douence, R., Fradet, P. and Südholt, M. (2004), "Composition, reuse and interaction analysis of stateful aspects", *Proceedings of the 3rd International Conference on Aspect-Oriented Software Development, ACM*, pp. 141-150.

Durr, P., Staijen, T., Bergmans, L. and Akşit, M. (2005), "Reasoning about semantic conflicts between aspects", Proceedings of the 2nd European Ineractive Workshop on Aspects in Software, pp. 10-18.

Guan, L.W., Li, X., Hu, H. and Lu, J. (2008), "A Petri net-based approach for supporting aspect-oriented modeling", *Frontiers of Computer Science in China*, Vol. 2 No. 4, pp. 413-423.

Jalali, A. (2011), "Foundation of aspect oriented business process management", dissertation master's thesis, Stockholm University, Stockholm.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W.G. (2001), "An overview of AspectJ", *European Conference on Object-Oriented Programming*, pp. 327-353.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C.V., Loingtier, J.M. and Irwin, J. (1997), "Aspect-oriented programming", *Proceedings of the 11th European Conference on Object-Oriented Programming, Jyväskylä, June 9-13*, pp. 220-242.

Kniesel, G. (2009), "Detection and resolution of weaving interactions", in Rashid, A. and Ossher, H. (Eds), *Transactions on Aspect-Oriented Software Development*, Springer, Berlin Heidelberg, pp. 135-186.

Kniesel, G. and Bardey, U. (2006), "An analysis of the correctness and completeness of aspect weaving", *Proceedings of the 13th Working Conference on Reverse Engineering, IEEE*, pp. 324-333.

Laue, R. and Mendling, J. (2010), "Structuredness and its significance for correctness of process models", *Information Systems and E-Business Management*, Vol. 8 No. 3, pp. 287-307.

Leite, J.C.S.P., Santoro, F.M., Cappelli, C., Batista, T.V. and Santos, F.J.N. (2016), "Ownership relevance in aspect-oriented business process models", *Business Process Management Journal*, Vol. 22 No. 3, pp. 566-593.

Li, T. (2008), *An Approach to Modelling Software Evolution Processes*, Springer-Verlag, Berlin.

Lieberherr, K.J. (1996), *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*, PWS Publishing Company, Boston, MA.

List, B. and Korherr, B. (2006), "An evaluation of conceptual business process modelling languages", The 21st ACM Symposium on Applied Computing, Dijon, April 23-27, pp. 1532-1539.

Mehmood, A. and Jawawi, D.N.A. (2013), "Aspect-oriented model-driven code generation: a systematic mapping study", *Information and Software Technology*, Vol. 55 No. 2, pp. 395-411.

Mendling, J. (2008), "Metrics for process models-empirical foundations of verification, error prediction, and guidelines for correctness", in Meersman, R. and Tari, Z. (Eds), *Lecture Notes in Business Information Processing*, Springer-Verlag, Berlin Heidelberg, pp. 113-130.

Molderez, T., Meyers, B., Janssens, D. and Vangheluwe, H. (2012), "Towards an aspect-oriented language module: aspects for petri nets", *Proceedings of the 7th workshop on Domain-Specific Aspect Languages, ACM*, Vol 3, pp. 21-26.

Nagy, I., Bergmans, L. and Akşit, M. (2005), "Composing aspects at shared join points", *Proceedings of International Conference NetOjectDays, Lecture Notes in Computer Science*, pp. 69-84.

Odgers, B. and Thompson, S. (1999), "Aspect-oriented process engineering (ASOPE)", *Proceedings of the Workshop on Object-Oriented Technology*, Springer-Verlag, London, pp. 295-299.

Ouyang, C., Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M. and Mendling, J. (2009), "From business process models to process-oriented software systems", *ACM Transactions on Software Engineering and Methodology*, Vol. 19 No. 1, Article No. 2.

Park, R., Choi, H., Lee, D., Kang, S., Cho, H. and Sohn, J. (2007), "Knowledge-based AOP framework for business rule aspects in business process", *ETRI Journal*, Vol. 29 No. 4, pp. 477-488.

Parnas, D.L. (1972), "On the Criteria to be used in decomposing systems into modules", *Communications of the ACM*, Vol. 15 No. 2, pp. 1053-1058.

Pawlak, R., Duchien, L. and Seinturier, L. (2005), "CompAr: ensuring safe around advice composition", in Steffen, M. and Zavattaro, G. (Eds), *Formal Methods for Open Object-Based Distributed Systems*, Springer, Berlin Heidelberg, pp. 163-178.

Pourshahid, A., Mussbacher, G., Amyot, D. and Weiss, M. (2009), "An aspect-oriented framework for business process improvement", in Babin, G., Kropf, P. and Weiss, M. (Eds), *E-Technologies: Innovation in an Open World*, Springer, Berlin Heidelberg, pp. 290-305.

Reisig, W. (1985), *Petri Nets: An Introduction*, Springer-Verlag, Berlin.

Roubtsova, E.E. and Aksit, M. (2005), "Extension of Petri nets by aspects to apply the model driven architecture approach", *Preliminary Proceedings of the 1st International Workshop on Aspect-Based and Model-Based Separation of Concerns in Software Systems, Nuremberg*.

Santos, F.J.N., Cappelli, C., Santoro, F.M., Leite, J.C.S.P. and Batista, T.V. (2012), "Aspect-oriented business process modeling: analyzing open issues", *Business Process Management Journal*, Vol. 18 No. 6, pp. 964-991.

Schauerhuber, A., Schwinger, W., Kapsammer, E., Retschitzegger, W., Wimmer, M. and Kappel, G. (2006), "A survey on aspect-oriented modeling approaches", technical report, Vienna University of Technology.

Schauerhuber, A., Schwinger, W., Kapsammer, E., Retschitzegger, W., Wimmer, M. and Kappel, G. (2007), "A survey on aspect-oriented modeling approaches", Relatorio Tecnico, Vienna University of Technology, Vienna.

Sourceforge (2013) "PIPE2 (Platform independent petri net editor beta)", available at: http://pipe2.sourceforge.net/

Tarr, P.L., Ossher, H.L., William, H.H. and Sutton, S.M. (1999), "N degrees of separation: multi-dimensional separation of concerns", *Proceedings of the 21st International Conference on Software Engineering, Los Angeles, CA, May 16-22*, pp. 107-119.

van der Aalst, W.M.P. (1998), "The application of Petri nets to workflow management", *Journal of Circuits, Systems, and Computers*, Vol. 8 No. 1, pp. 21-66.

van der Aalst, W.M.P., Dumas, M. and Gottschalk, F. (2008), "Correctness-preserving configuration of business process models", in Fiadeiro, J.L. and Inverardi, P. (Eds), *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, Springer, Berlin Heidelberg, pp. 46-61.

van der Aalst, W.M.P., Dumas, M. and Gottschalk, F. (2010), "Preserving correctness during business process model configuration", *Formal Aspects of Computing*, Vol. 22 No. 3-4, pp. 459-482.

William, H.H. and Ossher, H.L. (1993), "Subject-oriented programming: a critique of pure objects", *Proceedings of the 8th Conference on Object-Oriented Programming Systems, Languages, and Applications, Washington, DC, September 26-October 1*, pp. 411-428.

Xie, X. and Shatz, S.M. (2000), "An approach to using formal methods in aspect orientation", *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pp. 263-269.

Xu, D.X. and Nygard, K. (2006), "Threat-driven modeling and verification of secure software using aspect-oriented Petri nets", *IEEE Transactions on Software Engineering*, Vol. 32 No. 4, pp. 265-278.

**Corresponding author**
Xuan Zhang can be contacted at: zhxuan@ynu.edu.cn