

Formal Models for Architecture Aspects and Their Weaving

Chunhua Yang^{1,2}

¹ School of Computer Science and Technology, Shandong University, Jinan, China

² School of Information Science and Technology, Shandong Institute of Light Industry, Jinan, China
jnych@126.com

Haiyang Wang

School of Computer Science and Technology, Shandong University, Jinan, China
why@sdu.edu.cn

Abstract—Aspect-oriented concepts are currently introduced in early stages of software development to achieve better separation of concerns. However, at the architecture level, there exists no strict model for aspects and their weaving, which makes it difficult for analyzing and reasoning about the semantic problems introduced in the composition of the aspects and the base system. We present a formal model to specify architecture aspects. Its underlying formalism is Process Algebra. In the model, an aspect is specified as an extended architecture model, in which aspect components encapsulate the function of the aspect while aspect connectors encapsulate the weaving logics of the aspect. The separation of weaving logics can promote reuse. Then, we give a formal definition for aspect weaving. The definition builds the structural and behavioral relationship between the woven and wove models, which lays foundations for future semantic analysis and reasoning. An example illustrates the notions and models.

Index Terms—formal methods, aspect architecture, aspect weaving, aspect oriented software development, aspect oriented modeling, process algebra

I. INTRODUCTION

Aspect-oriented concepts are currently introduced in early stages of the software development life cycle with the aim of reducing complexity and enhancing maintainability already early on. On the design level, several approaches[1][2][3][4] have been proposed to modularize, represent and compose cross-cutting concerns using aspect-oriented techniques. Generally, they define weaving as certain composition rules, e.g. composition relationship[5], signature-based composition[6], aspect evaluation rules[4], etc.

However, there are no rigorous underlying model for specifying aspects and interpreting the relationship

between the woven model, the base model and aspect models, which make it difficult for analyzing and reasoning about the semantic problems introduced in the composition of the aspect and the base system[7]. Moreover, current approaches provide no support for expressing weaving characteristics. As a matter of fact, different aspects may have distinct structural, behavioral or weaving characteristics. For example, given the same join point, a logging aspect and an encryption aspect have distinct weaving characteristics. The logging aspect generally does not affect the control and data flow of the join point, whereas the encryption aspect requires that the control of the join point flow through the encryption aspect and the data of the join point be altered. Such characteristics, we call them *weaving logics*, are vital to the weaving of aspects, they should be modeled explicitly and strictly thereby.

We present a formal aspect architecture model to specify architecture aspects. The model is based on component oriented software architecture and Process Algebra theory[8]. In the aspect architecture model, special connectors named *aspect connectors* are used to specify aspect weaving logics. Furthermore, we give the formal definition of aspect weaving, which defines the structural and behavioral relationship between the woven and wove models (i.e. the base model and the aspect model).

This paper is structured as follows: In section 2, related notions are given and the based architecture model and the aspect architecture model are defined. In section 3, aspect weaving is defined. Then, an illustrative example is given in section 4. Thereafter, section 5 relates our work to other approaches. Finally, Section 6 gives the conclusion and the future work.

II. ASPECT MODELS

Models and notions in the paper are based on such an idea(see Figure 1): The base architecture model represents the initial design derived from the business concerns, while the aspect architecture model depicts one aspect derived from a crosscutting concern. The

This research was supported in part by the National Natural Science Foundation of China under Grant No.60673130, the Jinan Science & Technology Bureau under Grant No.051014, and the Department of Science & Technology of Shandong Province under Grant No.2006GG2201009.

Corresponding author: Chunhua Yang

crosscutting relationship between the aspect and the base architecture is modeled in a tuple $CRel$. Given a base

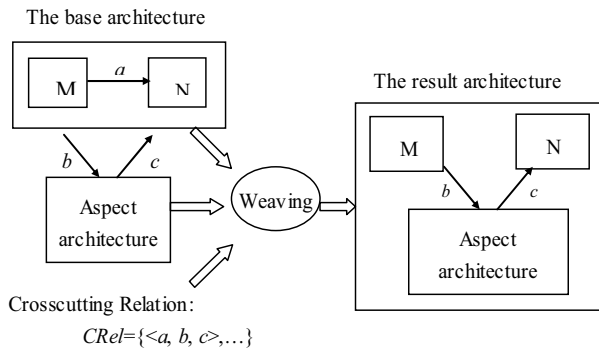


Figure 1. The main idea.

architecture model, an aspect architecture model, and the crosscutting relation $CRel$, aspect weaving is an operation to compose them to form a result architecture model.

A. A brief introduction to Process Algebra

The underlying formalisms of the model is Process Algebra[8]. In the section, we give important notions.

Process Algebra Terms. The set of process terms of the process algebra is generated by the following syntax:

$$P ::= 0 \mid \alpha.P \mid P + P \mid P \parallel P \mid P/L \mid P[f] \mid K$$

where a belongs to an action set A which includes a distinguished action τ for unobservable activities, $L \subseteq A - \{\tau\}$, f is a relabeling function, and K is a constant possessing a defining equation of the form $K \triangleq P$.

In the syntax above, the null term “0” is the term that cannot execute any action. The action prefix operator “ $\alpha.$ ” denotes the sequential composition of an action and a term. The hiding operator “ $/L$ ” makes some of the executed actions belonging to L unobservable. The relabeling operator “[f]” changes each executed action a into $f(a)$. The alternative composition operator “ $+$ ” expresses a nondeterministic choice between two terms. The parallel composition operator “ \parallel ” expresses the concurrent execution of two terms.

A PA process corresponds to a Labeled Transition System (LTS).

Labeled Transition System. A labeled transition system (LTS) is a triple (S, A, T) , where S is a set of states, A is a set of actions, $T \subseteq S \times A \times S$ is a transition relation.

B. The base architecture model

The base architecture model is a software architecture model. According to the notion of software architecture[9], it is composed of connected components and connectors.

Components provide the locus of computation. A component is described by an interface and behavior. An interface consists of actions that represent operations that can be accessed outside. The behavior described the inner computer of the component. Connectors are special-purpose components. Their architectural roles are to connect together components. They specify interactions among components.

Component. A component $c=(I, Beha)$ consists of an interface I and behavior $Beha$. The interface I is a set of observable actions, i.e. $I=\{a_1, \dots, a_n\}$ where $a_j \neq \tau$ for $j \in \{1..n\}$. The behavior $Beha$ is a PA term that corresponds to a labeled transition system (S, A, T) , where $A=I \cup \{\tau\}$ in which τ represents any unobservable inner action.

Connector. A connector $con=(I, Beha)$ has the same form as a component, though it has distinct roles in the architecture.

Before define the software architecture, we give an auxiliary definition “ \Rightarrow ”.

Given action a_1 and a_2 , if their execution should synchronize, then it is called that there exists connection between a_1 and a_2 , which is denoted as $a_1 \Rightarrow a_2$.

Software Architecture. A software architecture $sa=(I, Beha, Elem, Cfg)$ consists of a interface I , Behavior $Beha$, an element set $Elem$, and a topology set Cfg , where

- 1 $Elem=\{e \mid e \text{ is a component or a connector}\}$ and $Elem \neq \emptyset$;
- 2 $Cfg=\{\langle e_i, e_k, e_j, a \rangle \mid e_i, e_j \in Elem, a_k \in e_i.I, a_l \in e_j.I, e_i, a_k \Rightarrow e_j, a_l\}$;
- 3 $I=\{a \mid a \in e, I \wedge e \in Elem \wedge \neg \exists \langle s, a \rangle \in Cfg \forall \langle a, s \rangle \in Cfg\}$;
- 4 $Beha=(e_1.Beha \parallel \dots \parallel e_n.Beha)(f)$, where $e_1, \dots, e_n \in Elem$, f is a relabeling function. For each $cfg_i=\langle e_i, a_k, e_j, a \rangle \in Cfg$, f assigns it a unique name, i.e. $(e_i, e_k \rightarrow n, e_j, a_l \rightarrow n) \in f$ where n is a new name. Note here “ \rightarrow ” represents “is relabeled as”.

C. The aspect architecture model

An architecture aspect is specified as an extended architecture model-*aspect architecture model*. An aspect architecture model is composed of *aspect components* and *aspect connectors*. The former provide the locus of aspect functions, while the latter specify aspect weaving logics.

Aspect Component. An *aspect component* $acom=(I, Beha)$ consists of an interface I , behavior $Beha$. The elements of $Beha$ have the same form as of components despite that aspect component provide the locus of functions of an aspect.

Aspect Connector. An *aspect connector* $ac=(BI, AI, Beha)$, consists of a *base interface* BI , an *aspect interface* AI and behavior $Beha$, where:

- 1 BI is a non-empty set of observable actions;
- 2 AI is a non-empty set of observable actions;
- 3 $Beha$ is a PA term that corresponds to a labeled transition system (S, A, T) , where $A=BI \cup AI \cup \{\tau\}$ in which τ represents any unobservable inner action.

An aspect connector is a special connector that is the medium between an aspect component and a join point that the aspect component will crosscut. The *base interface* of the aspect connector would connect with the join point, while the *aspect interface* would connect with interface of the aspect component.

Aspect Architecture. An aspect architecture asa is a tuple $(I, Beha, Elem, Cfg)$, where the following conditions hold:

- 1 $Elem = CElem \cup ACElem$, where $CElem = \{e \mid e \text{ is an aspect component}\}$, $ACElem = \{e \mid e \text{ is an aspect connector}\}$, and satisfies $|CElem| \geq 1 \wedge |ACElem| \geq 1$;
- 2 I is an interface, and satisfies $I = \bigcup_{ac \in ACElem} ac.BI$;
- 3 $Cfg = \{ \langle e_i, a_k, e_j, a_l \rangle \mid (e_i \in ACElem \wedge e_j \in CElem \wedge e_i, a_k \in e_i.AI \wedge e_j, a_l \in e_j.I) \vee (e_i \in CElem \wedge e_j \in ACElem \wedge e_i, a_k \in e_i.I \wedge e_j, a_l \in e_j.AI) \wedge ((e_i, a_k \Rightarrow e_j, a_l) \vee (e_j, a_l \Rightarrow e_i, a_k)) \}$;
- 4 $Beha = (e_1.Beha \parallel \dots \parallel e_n.Beha)(f)$, where $e_1, \dots, e_n \in Elem$, f is a relabeling function. For each $cfg_i = \langle e_i, a_k, e_j, a_l \rangle \in Cfg$, $(e_i, a_k \rightarrow n, e_j, a_l \rightarrow n) \in f$. Note here “ $a \rightarrow n$ ” represents that action a is relabeled as n .

Aspect architecture is composed of an aspect component and aspect connectors. The base interfaces of aspect connectors comprise the interface of the aspect architecture. Aspect components connect to the outside only through the aspect connectors. The interface of an aspect component connects to the aspect interface of an aspect connector, which constitutes the configuration of the aspect architecture.

D. Types of aspect connectors

Aspect connectors are mechanisms for specifying the weaving logics of aspects, through which we can get the semantics of the relationship between aspects and the join points. Different aspects can adopt the same type of aspect connectors provided that they have the same weaving logics.

Given a join point that an aspect will be inserted, the behavior of aspects is to add some constraints to the join point. According to the control flow relationship between aspect interfaces and the join points, the weaving of aspects can be categorized as *parallel*, *sequential*, *choice* weaving.

Parallel weaving superimposes an observational control flow (See Fig. 2(a)), which do not change the control flow and data flow of the join point. In real worlds, aspects such as logging and tracing generally adopts this way. Sequential weaving requires the control of the join point pass through the aspects before they continue flowing along the original route (see Fig. 2(b)). Typically, for example, the weaving of encryption aspect is a sequential weaving. Choice weaving shown in Fig. 2(c) would change or interrupt the route of the join point. The access control aspect is an example of choice weaving.

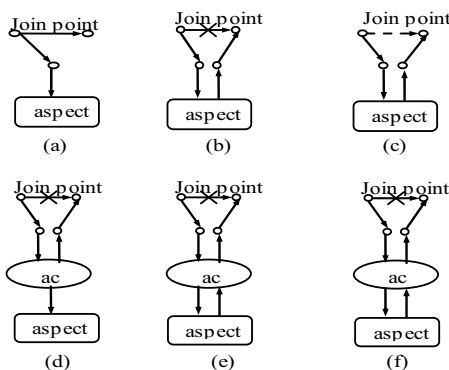


Figure 2. Weaving types and aspect connectors.

Adopting aspect connectors, the above discussed weaving logics are specified as behavior of aspect connectors. Thus, aspect weaving can be implemented in a unified way, i.e. passing the control flow of the join point through the aspect connector which decides the direction of the flow according to its behavior. Fig. 2(d)(e)(f) illustrates the weaving mechanisms based on aspect connectors.

Corresponding to types of weaving, we identify three types of aspect connectors, i.e. *parallel*, *sequential*, and *choice*. There is no doubt that types of aspect connectors are far more than the listed. For complicate aspects, we can define or combine these types to form new ones.

III. ASPECT WEAVING

A. Definition of aspect weaving

Given a base architecture that represents the initial design of the system and aspect architectures that represents the design of related aspects, aspect weaving is used to combine them to form a whole architecture.

Join Point. We define the *join point* as an interaction between two elements in the base architecture model. Structurally, the join point is a topology of the base architecture. Behaviorally, it is an action that the topology induces.

Crosscutting Relation. Given a set JP of join points, an interface I of an aspect architecture, define $CR = \{ \langle jp, a_1, a_2 \rangle \mid jp \in \langle s, t \rangle, a_1 \in I, a_2 \in I, s \Rightarrow a_1, a_2 \Rightarrow t \}$ as the crosscutting relation over JP and I .

In Fig.3, i and o are actions of an aspect architecture interface. Structurally, topology $\langle s, t \rangle$ is a join point, while action a that the topology induces is the behavioral join point. $\langle \langle s, t \rangle, i, o \rangle$ is a structural crosscutting relation tuple, whereas $\langle a, i, o \rangle$ is its behavioral counterpart.

Given a crosscutting relations CR , define an auxiliary function $Con(CR) = \{ \langle s, a_1 \rangle, \langle a_2, t \rangle \mid \langle \langle s, t \rangle, a_1, a_2 \rangle \in CR \}$.

Aspect Weaving \oplus_w . Given a base architecture Bsa , an aspect architecture Asa , a set JP of join points, a crosscutting relation CR over JP and $Asa.I$, then define operation \oplus_w as the weaving operation that combines Bsa and Asa . The operation inputs Asa , Bsa , JP and CR , and outputs an architecture rsa that satisfies:

- 1 $rsa.I = Bsa.I$;
- 2 $rsa.Elem = Bsa.Elem \cup Asa.Elem$;
- 3 $rsa.Cfg = (Bsa.Cfg - JP) \cup Bsa.Cfg \cup Con(CR)$;
- 4 $rsa.Beha = (e_1.Beha \parallel \dots \parallel e_n.Beha)(f)$, where $e_1, \dots, e_n \in rsa.Elem$, f is a relabeling function. For each $cfg_i = \langle e_i, a_k, e_j, a_l \rangle \in rsa.Cfg$, f assigns it a unique name; The aspect weaving operation is denoted as $rsa = \oplus_w(Bsa, Asa, JP, CR)$.

For example, given the following conditions (as shown in Fig.3(a)):

- $$Bsa.Elem = \{M, N\};$$
- $$Bsa.cfg = \{ \langle M, s, N, t \rangle \},$$
- $$Bsa.Beha = (M.Beha \parallel N.Beha)(f_1), \text{ where } f_1 = \{ s \rightarrow a, t \rightarrow a \};$$
- $$Asa.Elem = \{P, Q\};$$
- $$Asa.cfg = \{ \langle P, u, Q, v \rangle \},$$
- $$Asa.I = \{i, o\},$$

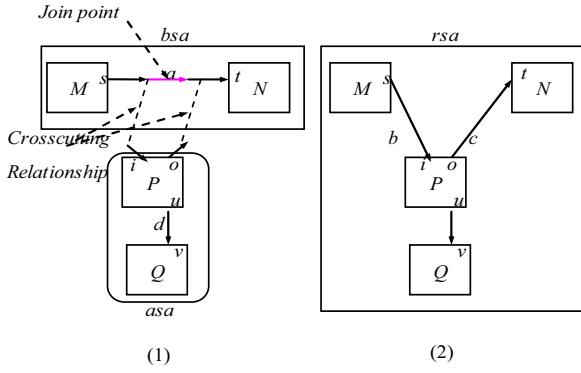


Figure 3. An example of aspect weaving.

$Asa.Beha=(P.Beha||Q.Beha)(f_2)$, where $f_2=\{u \rightarrow d, v \rightarrow d\}$;
 $JP=\{<M.s, N.t>\}$; $CR=\{<(M.s, N.t), asa.i, asa.o>\}$.

Let $rsa=\oplus_w(Bsa, Asa, JP, CR)$. Then, according to the definition of aspect weaving, we can get:

$$rsa.cfg=\{<M.s, asa.i>, <asa.o, N.t>, <P.u, Q.v>\};$$

$$rsa.Beha=(M.Beha||N.Beha||P.Beha||Q.Beha)(g),$$

$$g=\{s \rightarrow b, i \rightarrow b, o \rightarrow c, t \rightarrow c, u \rightarrow d, v \rightarrow d\}. \quad (1)$$

B. Behavior weaving

In the aspect weaving definition, behavior of the result architecture $rsa.Beha$ is derived from its configuration $rsa.Cfg$. In this section, the relationship between behavior of the base architecture $bsa.Beha$ and that of aspect architecture $asa.Beha$ will be built.

We redefine crosscutting relations from the point of view of Process Algebra.

Crosscutting Relation. Given an architecture model Bsa and an aspect architecture model Asa , lts_1 is the LTS of $Bsa.Beha$, and lts_2 is the LTS of $Asa.Beha$, a relation $CRel \subseteq lts_1.Act \times lts_2.Act \times lts_2.Act$ is a crosscutting relation iff $CRel = \{<jpa, aspa_1, aspa_2> | (jpa \in lts_1.Act \wedge jpa \text{ is a join point}) \wedge aspa_1 \neq aspa_2 \wedge (aspa_1, aspa_2 \in lts_2.Act) \wedge (aspa_1, aspa_2 \in Asa.I)\}$.

Given a triple $crel = \langle t, t_1, t_2 \rangle \in T$, define auxiliary functions $\varphi(t)=t$, $\varphi_1(t)=t_1$ and $\varphi_2(t)=t_2$ separately. In addition, let that $\varphi(T) = \bigcup_{\langle t_1, t_2 \rangle \in T} \varphi(t)$, $\varphi_1(T) = \bigcup_{\langle t_1, t_2 \rangle \in T} \varphi_1(t)$, and

$$\varphi_2(T) = \bigcup_{\langle t_1, t_2 \rangle \in T} \varphi_2(t).$$

Behavior Weaving \oplus_{bw} . Given two LTS lts_1, lts_2 , and a crosscutting relation $CRel$ on lts_1 and lts_2 , behavior weaving \oplus_{bw} is defined as $rlts = \oplus_{bw}(lts_1, lts_2, CRel)$, where $rlts$ is the result LTS that is achieved through the following two steps:

- Suppose that P_1 and P_2 be the PA terms of lts_1 and lts_2 respectively. For each $cr = \langle jpa, aspa_1, aspa_2 \rangle \in CRel$, execute the following rename operation on P_1 : Let P_M and P_N be the two corresponding process terms of P_1 sharing action jpa , and M and N are the source and the target architecture elements respectively that involve in action jpa . Then, rename action jpa of P_M as $aspa_1$, action jpa of P_N as $aspa_2$, i.e. P_1 becomes $(P_M(\varphi_1(jpa)) || P_N(\varphi_2(jpa)) || \dots)$. (Note: Here $P_1 \triangleq (P_M || P_N || \dots)$).

- After executing such rename operations on P_1 , the PA expression P_r of $rlts$ is the parallel composition of P_1 and P_2 , i.e. $P_r \triangleq P_1 || P_2$.

For example, suppose that there be two PA terms P_1 and A which are defined as follows:

$$P_1 \triangleq (P_M || P_N; P_M \triangleq \bar{a}.d.P_M; P_N \triangleq a.\bar{d}.P_N; A \triangleq b.\tau.\bar{c}.A.$$

The state transition graphs of lts_1 of P_1 and lts_2 of A are illustrated in Fig.4(1) and Fig.4(2). Let $CRel = \{<a, b, c>\}$. Now we evaluate $rlts = \oplus_{bw}(lts_1, lts_2, CRel)$ according to the definition of behavior weaving. Firstly, rename the action a of P_M as b and a of P_N as c , i.e. P_1 becomes $P_M(\varphi_1(a)) || P_N(\varphi_2(a))$. Then the P_r of $rlts$ is $P_r \triangleq P_M(\varphi_1(a)) || P_N(\varphi_2(a)) || A$. Fig.4(3) depicts the LTS $rlts$.

Behavior Weaving Semantics. Given LTS $lts_1 = \langle S_1, A_1, T_1 \rangle$, $lts_2 = \langle S_2, A_2, T_2 \rangle$, $rlts = \langle RS, RA, RT \rangle$, a crosscutting relation $CRel$ on lts_1 and lts_2 , and $rlts = \oplus_{bw}(lts_1, lts_2, CRel)$. Then, its semantics is as follows:

- $RA = (A_1 - \varphi(CRel)) \cup A_2$;
- $RS \subseteq S_1 \times S_2$, and for each state $s = \langle s_1, \dots, s_m \rangle$, $as_1, \dots, as_n \in RS$, and action $a \in RA$, the following conditions satisfy:
 - if $a \in A_1 - \varphi(CRel)$ and there exists a transition $\langle s_1, \dots, s_m \rangle \xrightarrow{a} \langle s'_1, \dots, s'_m \rangle \in T_1$, then there exists a transition $s \xrightarrow{a} s' \in RT$ where $s' = \langle s'_1, \dots, s'_m, as_1, \dots, as_n \rangle$;
 - if $a \in A_2 - \varphi_1(CRel) - \varphi_2(CRel)$ and there exists a transition $\langle as_1, \dots, as_n \rangle \xrightarrow{a} \langle as'_1, \dots, as'_n \rangle \in T_2$, then there exists a transition $s \xrightarrow{a} s' \in RT$ where $s' = \langle s_1, \dots, s_m, as'_1, \dots, as'_n \rangle$;
 - if $a \in \varphi_1(CRel)$, $\langle b, a, y \rangle \in CRel$, and there exists a transition $\langle s_1, \dots, s_m \rangle \xrightarrow{b} \langle s'_1, \dots, s'_m \rangle \in T_1$ where $s'_k = s_k$ except $s'_i \neq s_i$ and $s'_j \neq s_j$, and a transition $\langle as_1, \dots, as_n \rangle \xrightarrow{a} \langle as'_1, \dots, as'_n \rangle \in T_2$ where $as'_l = as_l$ except $as'_i \neq as_i$, then there exists a transition $s \xrightarrow{a} s' \in RT$ where $s' = \langle s'_1, \dots, s'_m, as'_1, \dots, as'_n \rangle$ in which $s'_k = s_k \wedge s'_i \neq s_i \wedge s'_j = s_j$ and $as'_l = as_l \wedge as'_i \neq as_i$; ($k \in \{1, \dots, m\} - \{i, j\} \wedge 1 \leq i, j \leq m; l \in \{1, \dots, n\} - \{l\} \wedge 1 \leq l \leq n$)
 - if $a \in \varphi_2(CRel)$, $\langle b, y, a \rangle \in CRel$, and there exists a transition $\langle s_1, \dots, s_m \rangle \xrightarrow{b} \langle s'_1, \dots, s'_m \rangle \in T_1$ where $s'_k = s_k$ except $s'_i \neq s_i$ and $s'_j \neq s_j$, and a transition

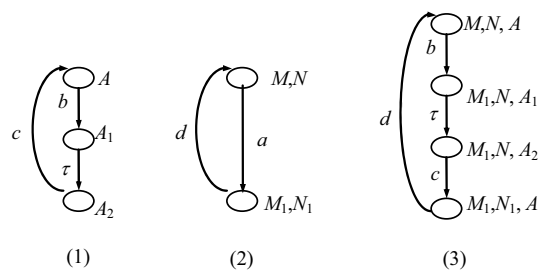


Figure 4. Illustration of aspect behavior weaving.

$\langle as_1, \dots, as_n \rangle \xrightarrow{a} \langle as'_1, \dots, as'_n \rangle \in T_2$ where $as'_i = as_i$

except $as'_i \neq as_i$, then there exists a transition $s \xrightarrow{a} s' \in RT$ where $s' = \langle s'_1, \dots, s'_m, as'_1, \dots, as'_n \rangle$ in which $s'_k = s_k \wedge s'_i = s_i \wedge s'_j \neq s_j$ and $as'_i = as_i \wedge as'_i \neq as_i$; ($k \in \{1, \dots, m\} - \{i, j\} \wedge 1 \leq i, j \leq m$; $l \in \{1, \dots, n\} - \{i\} \wedge 1 \leq l \leq n$).

Note that here it is supposed that s_i be the state of the process term of the source element that activates action b , and s_j be that of the target element.

Theorem 1. Given the base architecture Bsa , an aspect architecture Asa , a set JP of join points, and the crosscutting relation CR over JP and $Asa.I$, and $rsa = \oplus_w(Bsa, Asa, JP, CR)$, then $rsa.Beha \approx \oplus_{bw}(Bsa.Beha, Asa.Beha, BCR)$, where $BCR = \{ \langle a, b, c \rangle | \langle \langle s, t \rangle, i, o \rangle \in CR, a \text{ is the name in } bsa.Beha \text{ that assigned to } \langle s, t \rangle \}$.

The proof can be obtained from the definition of aspect behavior weaving, so it is omitted.

Reconsider the example as shown in Fig. 3.

$$\begin{aligned} Bsa.Beha &= (M.Beha || N.Beha)(f_1) \\ &= M.Beha(s \rightarrow a) || N.Beha(t \rightarrow a), \\ &\text{where } f_1 = \{s \rightarrow a, t \rightarrow a\}; \end{aligned}$$

$$\begin{aligned} Asa.Beha &= (P.Beha || Q.Beha)(f_2) \\ &= P.Beha(u \rightarrow d) || Q.Beha(v \rightarrow d), \\ &\text{where } f_2 = \{u \rightarrow d, v \rightarrow d\}; \end{aligned}$$

$$CR = \{ \langle (M.s, N.i), A.i, A.o \rangle \};$$

According to $CR = \{ \langle (M.s, N.i), A.i, A.o \rangle \}$, $f_1 = \{s \rightarrow a, t \rightarrow a\}$, we can get that $BCR = \{ \langle a, i, o \rangle \}$. According to the definition of behavior weaving, we have:

$$\begin{aligned} rsa.Beha &= \oplus_{bw}(Bsa.Beha, asa.Beha, BCR) \\ &= (M.Beha(s \rightarrow a) || N.Beha(t \rightarrow a), \\ &\quad (P.Beha(u \rightarrow d) || Q.Beha(v \rightarrow d), \{ \langle a, i, o \rangle \})) \\ &= (M.Beha(s \rightarrow a)(a \rightarrow i) || N.Beha(t \rightarrow a)(a \rightarrow o)) || (P. \\ &\quad Beha(u \rightarrow d) || Q.Beha(v \rightarrow d)) \\ &= (M.Beha || N.Beha || P.Beha || Q.Beha)(s \rightarrow i, \quad t \rightarrow o, \\ &\quad u \rightarrow d, v \rightarrow d) \quad (2). \end{aligned}$$

The expression (2) is equivalent with expression (1) in section 3.1 that resulted from the definition of aspect weaving.

IV. AN EXAMPLE

In this section, we introduce an e-commerce example [10] to illustrate the proposed notions and models. The e-commerce system includes the following functional requirements: A customer requires the system to browse catalogs and make orders, and the system will search and return the catalog to the customer and send customer orders to the shipping center. There are two non-functional requirements. One is that whenever the customer sends a purchase order request, it should be logged. The other is that before the customer sends a browsing catalog request, he needs permission.

A. The base architecture model of the example

According to above proposed architecture notion, we design the base architecture model of the e-commerce example as illustrated in Fig. 5.

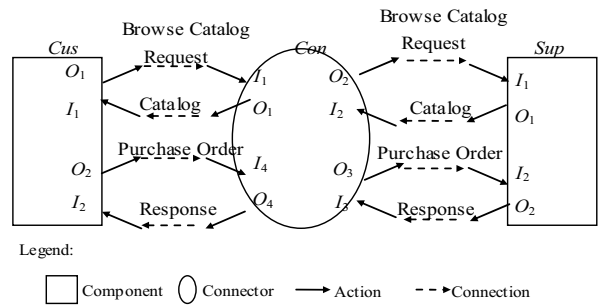


Figure 5. The base architecture model of the example.

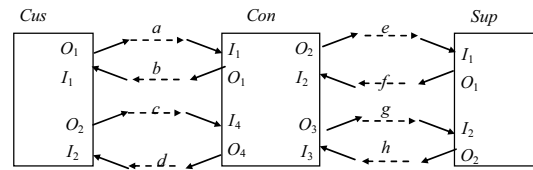


Figure 6. The flow graph of the e-commerce system behavior.

The model has two components Cus and Sup and a connector con . Component Cus and Sup encapsulate functions of the customer and the supplier respectively. Connector con provides the interaction between Cus and Sup . Take the Cus for instance, its interface has four actions I_1, O_1, I_2 and O_2 which represent sending browsing catalog request receiving catalog, sending order requests and receiving the order response respectively. Other elements of the model can be deduced from the Fig. 5.

Formally, the base architecture model $Bsa = (I, Beha, Elem, Cfg)$, where:

- 1 $I = \emptyset; Elem = \{ Cus, Sup, Con \};$
- 2 $Cfg = \{ \langle Cus.O_1, Con.I_1 \rangle, \langle Con.O_1, Cus.I_1 \rangle, \langle Cus.O_2, Con.I_4 \rangle, \langle Con.O_4, Cus.I_2 \rangle, \langle Con.O_2, Sup.I_1 \rangle, \langle Sup.O_1, Con.I_2 \rangle, \langle Con.O_3, Sup.I_2 \rangle, \langle Sup.O_2, Con.I_3 \rangle \};$
- 3 $Beha \triangleq (Cus.Beha || Con.Beha || Sup.Beha)f;$
 $f = \{ Cus.O_1 \rightarrow a, Con.I_1 \rightarrow a, Con.O_1 \rightarrow b, Cus.I_1 \rightarrow b, Cus.O_2 \rightarrow c, Con.I_4 \rightarrow c, Con.O_4 \rightarrow d, Cus.I_2 \rightarrow d, Con.O_2 \rightarrow e, Sup.I_1 \rightarrow e, Sup.O_1 \rightarrow f, Con.I_2 \rightarrow f, Con.O_3 \rightarrow g, Sup.I_2 \rightarrow g, Sup.O_2 \rightarrow h, Con.I_3 \rightarrow h \}.$

Fig. 6 is the flow graph of the base architecture model. Behavior of each element is as follows:

$$\begin{aligned} Cus.Beha &\triangleq P_1; P_1 \triangleq \overline{O_1}.I_1.\tau.\overline{O_2}.I_2.P_1 \\ Con.Beha &\triangleq P_2; P_2 \triangleq I_1.\overline{O_2}.I_2.\overline{O_1}.I_4.\overline{O_3}.I_3.\overline{O_4}.P_2 \\ Sup.Beha &\triangleq P_3; P_3 \triangleq I_1.\tau.\overline{O_1}.I_2.\tau.\overline{O_2}.P_3. \end{aligned}$$

B. Aspect architecture models of the example

Logging and security is the typical crosscutting concerns, so we model them as aspects. The aspect architecture models for the logging and security aspect are shown in Fig. 7.

In Fig. 7, Asa_1 is the aspect architecture model of the logging aspect, which comprises an aspect component logging and an aspect connector Ac_1 . The aspect interface of Ac_1 is $acout$, which connects with the port ain of logging. The base interface of Ac_1 includes I_1 and O_1 , which compose the interface of Asa_1 . Similarly, Asa_2 is the aspect architecture model of the security aspect(i.e.

the *access control* aspect), which contains an aspect component *AccCtrl* and an aspect connector *Ac₂*. The

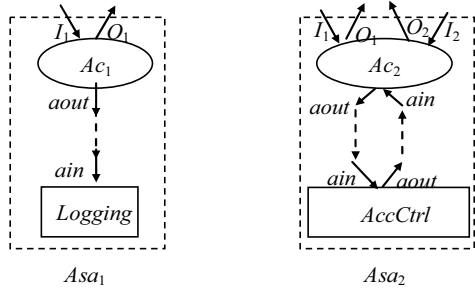


Figure 7. Aspect architecture models of the example.

aspect component *AccCtrl* inputs data and outputs the results through its interface *ain* and *aout*. The two interfaces connect to the aspect interface *aout* and *ain* of aspect connector *Ac₂* respectively. The base interface of the aspect connector *Ac₂* includes *I₁*, *O₁*, *I₂* and *O₂*, which form the base interface of the aspect architecture model *Asa₂*.

Formally, the aspect architecture models for *Asa₁* and *Asa₂* are listed as follows:

- 1 $Asa_1.I = \{Ac_1.I_1, Ac_1.O_1\};$
 $Asa_1.Elem = \{Ac_1, Logging\};$
 $Asa_1.Cfg = \{<Ac_1.aout, logging.ain>\};$
- 2 $Asa_2.I = \{Ac_2.I_1, Ac_2.O_1, Ac_2.I_2, Ac_2.O_2\};$
 $Asa_2.Elem = \{Ac_2, AccCtrl\};$
 $Asa_2.Cfg = \{<Ac_2.aout, AccCtrl.ain>, <AccCtrl.aout, Ac_2.ain>\}.$

Behavior of the two aspect architecture models and their elements is depicted in Table 1.

As shown in Table 1, the aspect connector *Ac₁* of the *logging* aspect is a parallel type, whereas *Ac₂* of the *AccCtrl* aspect is a choice type. From their behavior we can deduce the behavioral semantic relationship between aspects and the join points.

C. Aspect weaving

Now let's consider weaving the two aspects. According to the requirement, the logging aspect and the access control aspect should be inserted after a customer sends a purchase order and before the customer sends a browsing catalog request. So, join points and crosscutting relationship of the two aspects are listed as follows:

- $$JP_1 = \{<Cus.O_2, Con.I_4>\};$$
- $$CR_1 = \{(<Cus.O_2, Con.I_4>, Asa_1.I_1, Asa_1.O_1)\}.$$
- $$JP_2 = \{<Con.O_2, Sup.I_1>, <Sup.O_1, Con.I_2>\};$$
- $$CR_2 = \{(<Con.O_2, Sup.I_1>, Asa_2.I_1, Asa_2.O_1), <Sup.O_1, Con.I_2>, Asa_2.I_2, Asa_2.O_2)\}.$$

Firstly, the logging aspect is woven into the base model through operation $rsa_1 = \oplus_w(Bsa, Asa_1, JP_1, CR_1)$. Then, the access control aspect is inserted to the model *rsa₁* through operation $rsa_2 = \oplus_w(rsa_1, Asa_2, JP_2, CR_2)$.

According to the definition of aspect weaving, the description of the two woven models is as follows:

- 1 $rsa_1.I = Bsa.I = \emptyset;$
 $rsa_1.Elem = bsa.Elem \cup asa_1.Elem$
 $= \{Cus, Sup, Con, logging, Ac_1\};$
 $rsa_1.Cfg = (bsa.Cfg - JP_1) \cup asa_1.Cfg \cup Con(CR_1);$

TABLE I.

THE BEHAVIOR OF ARCHITECTURE ELEMENTS OF THE E-COMMERCE SYSTEM

| Elements | Behavior |
|------------------------|--|
| <i>logging</i> | $logging.Beha \triangleq logP; logP \triangleq ain.logP$ |
| <i>AccCtrl</i> | $AccCtrl.Beha \triangleq AccCtrlP$ $AccCtrlP \triangleq ain.\tau.aout.AccCtrlP$ |
| <i>Ac₁</i> | $Ac_1.Beha \triangleq Ac_1P; Ac_1P \triangleq I_1.\overline{aout}.O_1.Ac_1P$ |
| <i>Ac₂</i> | $Ac_2.Beha \triangleq Ac_2P$ $Ac_2P \triangleq I_1.aout.ain.(O_1.I_2.O_2 + O_2).Ac_2P$ |
| <i>Asa₁</i> | $(Logging.Beha Ac_1.Beha)(f_1)$ $f_1 = \{<Ac_1.aout, loga>, <Logging.ain, loga>\}$ |
| <i>Asa₂</i> | $(AccCtrl.Beha Ac_2.Beha)(f_2)$ $f_2 = \{<Ac_2.aout, aca>, <AccCtrl.ain, aca>, <AccCtrl.aout, acb>, <Ac_2.ain, acb>\}$ |

$$= \{<Cus.O_1, Con.I_1>, <Con.O_1, Cus.I_1>, <Con.O_4, Cus.I_2>, <Con.O_2, Sup.I_1>, <Sup.O_1, Con.I_2>, <Con.O_3, Sup.I_2>, <Sup.O_2, Con.I_3>, <Cus.O_2, Asa_1.I_1>, <Asa_1.O_1, Con.I_4>, <Ac_1.aout, logging.ain>\}$$

- $$rsa_1.Beha = (Cus.Beha || Sup.Beha || Con.Beha || logging.Beha || Ac_1.Beha)(f),$$
- where *f* is a relabeling function. For each $cfg_i = \langle e_i.a_k, e_j.a_l \rangle \in rsa_1.Cfg$, *f* assigns it a unique name.
- 2 $rsa_2.I = rsa_1.I = \emptyset;$
 $rsa_2.Elem = rsa_1.Elem \cup asa_2.Elem$
 $= \{Cus, Sup, Con, logging, Ac_1, AccCtrl, Ac_2\};$
 $rsa_2.Cfg = (rsa_1.Cfg - JP_2) \cup asa_2.Cfg \cup Con(CR_2);$
 $= \{<Cus.O_1, Con.I_1>, <Con.O_1, Cus.I_1>, <Con.O_4, Cus.I_2>, <Con.O_3, Sup.I_2>, <Sup.O_2, Con.I_3>, <Cus.O_2, Asa_1.I_1>, <Asa_1.O_1, Con.I_4>, <Ac_1.aout, logging.ain>, <Con.O_2, Asa_2.I_1>, <Asa_2.O_1, Sup.I_1>, <Sup.O_1, Asa_2.I_2>, <Asa_2.O_2, Con.I_2>, <Ac_2.aout, AccCtrl.ain>, <AccCtrl.aout, Ac_2.ain>\}$

$$rsa_2.Beha = (Cus.Beha || Sup.Beha || Con.Beha || logging.Beha || Ac_1.Beha || AccCtrl.Beha || Ac_2.Beha)(g),$$

where *g* is a relabeling function. For each $cfg_i = \langle e_i.a_k, e_j.a_l \rangle \in rsa_2.Cfg$, *f* assigns it a unique name.

The architecture resulted from weaving the two aspects, i.e. *rsa₂*, are shown in Fig.8. From the figure, we can see the altered configuration apparently.

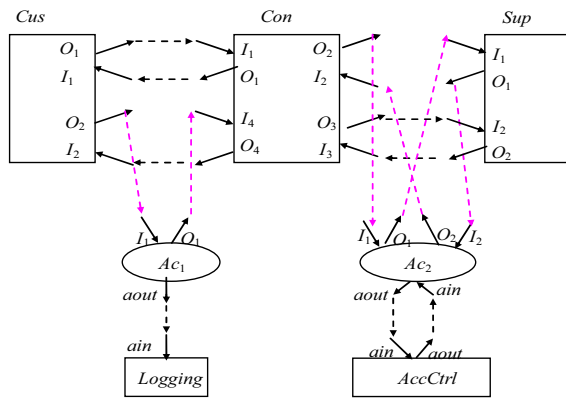


Figure 8. The result architecture model of the example.

V. RELATED WORK

Nowadays, design-level aspect weaving mainly takes on two forms, i.e. by merging aspect models into the base model[1][2] or by connecting corresponding components of the two models[3][4]. In the latter way, aspect models and the base model are divisible in the result model, whereas they are indivisible in the former way. We call them *coalescent* and *connective* ways respectively.

In the AOM approach proposed by R. France *et al.* [2], composition of the aspect and primary models relies on signature matching: A model element is merged with another if their signatures match[6]. In the theme approach proposed by Clarke *et al.* [1], composition of design models, i.e. themes, is specified with a composition relationship[5] which is based on name matching. In DAOP-ADL proposed by Pinto M. *et al.* [4], components are specified by a set of *provided* interfaces and *required* interfaces, whereas aspects are specified by a set of *evaluated* interfaces and *required* interfaces. Composition constraints are expressed in terms of a set of component composition rules and a set of aspect evaluation rules. In Ref.[3], Prez J. et al propose an architectural modeling approach based on aspects and components that use a component definition language to define architectural types at a high abstraction level and a configuration language to design the architecture of software systems.

Our weaving operation is based on the latter way, i.e. connective way. Moreover, the main differences between our weaving model and the above mentioned design-level related work is twofold: Firstly, underlying paradigms of our work are process algebra, whereas theirs are UML[1][2] or ADL[3][4]. Secondly, our weaving model builds the logic operation relation between the woven and wove models, whereas theirs do not build such a logic relation.

In addition to design-level weaving, source code level weaving has attracted more attention and many deep researches have been conducted on weaving model[13][14] or semantics[15]. At requirement level, J. Klein *et al.*[16] propose a semantic-based aspect waving algorithm for Hierarchical Message Sequence Charts(HMSCs). Source-level work similar to ours

includes: in Ref.[17], PA as a tool has been used in AOP field for modeling aspects and weaving; in Ref.[18], aspects are raised from code artifacts to mathematical entities (functions that transform programs) and an algebra are developed to model aspect composition. However, such works are applicable for AOP models.

As for the weaving logics, its separation has received certain attention from some work. M. Kande[11] proposed that crosscutting relationship should be captured and encapsulated in independent identities. Batista *et al.* [12] reflect on architectural connection and advocate that aspectual connectors should be used to implement composition of aspect components. However, they have not created strict models and concrete mechanisms for encapsulating the weaving logics.

VI. CONCLUSIONS AND FUTURE WORK

We created the aspect architecture models to specify architecture aspects. One of the main contributions of the models is to express weaving logics through independent aspect connectors. On the one hand, the separation of aspect connectors from aspect components can promote the reuse of aspect weaving logics. On the other hand, the encapsulation of weaving logics simplifies the implementation of weaving. Then, we defined weaving operations formally. The operation defines the structural and behavioral relationship between the base architecture model, aspect architecture model and the woven model, which lays basis for future reasoning on the semantic related problems. Moreover, the underlying formalism-Process Algebra make it convenient for future analysis on the characteristics of the architecture models.

The models proposed in the paper are suitable for aspects that own certain functions and provide auxiliary computation for the base model. Many aspects in real applications such as security, logging, communication etc belong to such categories and can be expressed by the model thereby.

The more complicate problems related to the weaving such as the weaving orders of multiple aspects are to be explored in our subsequent work.

ACKNOWLEDGMENT

The authors wish to thank Jinkui Hou, Xudong Lu, Shuaiqiang Wang, and Shihong Feng for their help and support in the work, and the reviewers for their comments. This work was supported in part by the National Natural Science Foundation of China under Grant No.60673130, the Jinan Science &Technology Bureau under Grant No.051014, and the Department of Science & Technology of Shandong Province under Grant No.2006GG2201009.

REFERENCES

- [1] S. Clarke and R. J. Walker, "Composition patterns: An approach to designing reusable aspects", *In International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001, pp.5-14.

- [2] R. France, I. Ray, G. Georg, and S. Ghosh, "Aspect-Oriented Approach to Early Design Modeling", *IEEE Software*, vol.151, Issue 4, pp.173-186, 2004.
- [3] J. Perez, I. Ramos, J. Jaen, P. Letelier, and E. Navarro, "PRISMA: Towards Quality, Aspect Oriented and Dynamic Software Architectures". In *Proceedings of 3rd IEEE International Conference on Quality Software (QSIC 2003)*, Dallas, Texas, USA, 2003, pp.59-66.
- [4] M. Pinto, L. Fuentes, J. M. Troya, "DAOP-ADL: An Architecture Description Language for Dynamic Component and Aspect-Based Development". In *Proceeding of the Second. International Conference on GPCE*, Erfurt, Germany, 2003, pp.118-137. Springer-Verlag.
- [5] S. Clarke, "Extending standard UML with model composition semantics", *Science of Computer Programming*, vol. 44 Issue 1: Elsevier Science, 2002, pp.71-100.
- [6] R. Reddy, R. France, S. Ghosh, F. Fleurey, and B. Baudry, "Model composition-a signature-based approach". In *Proceedings of Aspect Oriented Modeling (AOM) Workshop associated to MoDELS'05*, Montego Bay, Jamaica, October 2005.
- [7] P. Durr, T. Staijen, L. Bergmans, and M. Aksit, "Reasoning About Semantic Conflicts Between Aspects", In *Proceedings of the 2nd European Interactive Workshop on Aspects in Software (EIWAS)*, Brussels, Belgium, September 2005.
- [8] M. Bernardo, P. Ciancarini, L. Donatiello, "Architecting families of software systems with process algebras", *ACM Transactions on Software Engineering and Methodology*, 2002, 11(4):386-426.
- [9] R. Allen, D. Garlan, "A formal basis for architectural connection", *ACM Transactions on Software Engineering and Methodology*, 1997, vol.6 Issue 3, pp.213-249.
- [10] H. Gomaa, M.E. Shin, "Modeling complex systems by separating application and security concerns", In *Proceedings of the 9th IEEE International Conference on engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age (ICECCS'04)*, Florence, Italy, 2004, pp.19-28.
- [11] M. Kande, A concern-oriented approach to software architecture. Ph.D. Thesis, Lausanne, Switzerland: Swiss Federal Institute of Technology (EPFL), 2003.
- [12] T. Batista, C. Chavez, and A. Garcia, "Reflections on Architectural Connection: Seven Issues on Aspects and ADLs", In *Proceedings of the 2006 international workshop on Early aspects at ICSE (ICSE'06)*, Shanghai, China, 2006, ACM Press, New York, NY, 2006, pp. 3-9.
- [13] H. Masuhara, G. Kiczales and C. Dutchyn, "A Compilation and Optimization Model for Aspect-Oriented Programs", In *Proceedings of Compiler Construction (CC2003)*, LNCS 2622, 2003, pp.46-60.
- [14] M. Wand, G. Kiczales and C. Dutchyn, "A Semantics for Advice and Dynamic Join Points in Aspect-Oriented Programming", *ACM Transactions on Programming Languages and Systems*, vol. 26, No. 5, September 2004, pp. 890-910.
- [15] M.C. Skipper, Formal Models for Aspect-Oriented Software Development, [PhD Thesis], Imperial College London, 2004.
- [16] J. Klein, L. Hérouët, J.M. Jézéquel, "Semantic-based Weaving of Scenarios", In *Proceedings of the 5th international conference on Aspect-oriented software development(AOSD 06)*, March 20-24, 2006, Bonn, Germany, pp.27-38.
- [17] J. H. Andrews, "Process-algebraic foundations of aspect-oriented programming", In *Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns(REFLECTION'01)*, volume 2192, Springer-Verlag, Sept 2001, pp.187-209..
- [18] R. Lopez-Herrejon, D. Batory, C. Lengauer, "A disciplined approach to aspect composition", In *Proceedings of the 2006 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation(PEPM'06)*, January 9-10, 2006, Charleston, South Carolina, USA, pp. 68-77.

Chunhua Yang was born in Jinan in 1974. She received the BSc and MSc degree in Computer Science from Shandong University, Jinan, China in 1995 and 2002 respectively. She is a PhD candidate in the School of Computer Science and Technology at Shandong University. Her research interests include aspect oriented technologies, business process, and web services.

Haiyang Wang was born in Wendeng in 1965. He received the BSc and Msc degree in Computer Science from Shandong University, Jinan, China in 1985 and 1988 respectively, and the PhD degree from the Institute of Computing Technology of the Chinese Academy of Sciences. He is a Professor of the Department of Computer Science and Technology of the Shandong University, Shandong, China. His research interests include software and data engineering, computer supported cooperative work (CSCW) and business process management (BPM).