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

$$P ::= 0 | a.P | P + P | P | P/L | P[f] | K$$

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

In the syntax above, the null term “0” is the term that cannot execute any action. The action prefix operator “$a.$” denotes the sequential composition of an action and a term. The hiding operator “/” makes some of the executed actions belonging to $L$ unobservable. The relabeling operator “[f]” changes each executed action a turned into $f(a)$. The alternative composition operator “+” expresses a nondeterministic choice between two terms. The parallel composition operator “||” 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 $Baha$. The interface $I$ is a set of observable actions, i.e. $I=\{a_1, ..., a_n\}$ where $a_j \neq \tau$ for $j=1..n$. The behavior $Baha$ is a PA term that corresponds to a labeled transition system $(S, A, T)$, where $A=A \cup \{\tau\}$ in which $\tau$ represents any unobservable inner action.

**Connector.** A connector $con=(I, Beha, Elem, Cfg)$ consists of a set of interfaces $I$, Behavior $Baha$, an element set $Elem$, and a topology set $Cfg$, where

1. $Elem=\{e \mid e \text{ is a component or a connector}\} \cup \{\emptyset\}$
2. $Cfg=\{<e_1, a_1, e_2, a_2> | e_1, e_2 \in Elem, a_1 = a_2, e_1 = e_2\}$
3. $I=\{a_1 \in Elem, I = \{a_1 \in Cfg \} \cup \{\emptyset\}$
4. $Baha(e_1, Beha[\ldots|e_n, Beha(f)](f)$, where $e_1, ..., e_n \in Elem, f$ is a relabeling function. For each $cfg <e_1, a_1, e_2, a_2> \in Cfg$, $f$ assigns it a unique name, i.e. $(e_1, e_2 \rightarrow n, e_1, e_2 \rightarrow m) \in f$ 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 $Baha$. The elements of $Baha$ have the same form as of components despite that aspect component provide the locus of functions of an aspect.

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

1. $BI$ is a non-empty set of observable actions;
2. $AI$ is a non-empty set of observable actions;
3. $Baha$ is a PA term that corresponds to a labeled transition system $(S, A, T)$, where $A=BI \cup AI \cup \{\tau\}$ in which $\tau$ 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 $asa=(I, Beha, Elem, Cfg)$, where the following conditions hold:
1. \( E \) is an aspect component, \( A \) is an aspect connector, and satisfies \( |E| \geq 1 \wedge |A| \geq 1 \).

2. \( I \) is an interface, and satisfies \( I = \bigcup_{E \in A \cup E} BI \).

3. \( Cfg = \{<e, a, a', a_0>| (e, e \in ACElem \land e, e, e \in AI) \lor (e, e, e \in ACElem \land e, e, e \in AI) \} \).

4. \( B = (e, B, ... | e, B, B)(f) \), where \( e, e \in E, f \) is a relabeling function. For each \( cfg = <e, a, a_0> \), \( e, a, a_0 \in Cfg, e, a, a_0 \in E, f, a, a_0 \in f \).

Aspects 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 architecture. Behaviorally, an 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 architecture, 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.

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 complicated 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 \( CR = \{<p, a, a_0>|<p, a, a_0> \in JP, a_1 \in I, s \to a, a_2 \to \} \) as the crosscutting relation over \( JP \) and \( I \).

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

Given a crosscutting relations \( CR \), define an auxiliary function \( Com(CR) = \{<s, a, a_0>|<s, t, \alpha, \beta> \in CR \} \).

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

\[ \text{rsa}.I = \text{Bsa}.I; \]
\[ \text{rsa}.E = \text{Bsa}.E; \]
\[ \text{rsa}.C = \text{Bsa}.C \cup \text{Asa}.C \cup \text{Con}(CR); \]
\[ \text{rsa}.B = (\text{Asa}.B, \text{Bsa}.B)(f) \], where \( e_1, ..., e_n \in \text{rsa}.E \).

The aspect weaving operation is denoted as \( rsa = \oplus (Bsa, Asa, JP, CR) \).

For example, given the following conditions (as shown in Fig. 3(a)): \( Bsa.E = \{M, N\}; \)
\( Bsa.C = \{<M.s, N.t>\}; \)
\( Bsa.B = (M.B, N.B)(f) \), where \( f = \{s \to a, t \to \} \);
\( Asa.E = \{P, Q\}; \)
\( Asa.C = \{<P.u, Q.v>\}; \)
\( Asa.I = \{i, o\} \).
After executing such rename operations on $P_1$, the PA expression $P$, of $rlts$ is the parallel composition of $P_1$ and $P_2$, i.e. $P = \| P_1 \parallel P_2$.

For example, suppose that there be two PA terms $P_1$ and $P_2$, which are defined as follows:

$P_1 = \{(a, b, c)\}$

$P_2 = \{(a, b, c)\}$

The state transition graphs of $lts_1$ of $P_1$ and $lts_2$ of $P_2$ are illustrated in Fig.4(1) and Fig.4(2). Let $CRel = \{(a, b, c)\}$. Now we evaluate $rlts = \Theta_{\Delta}(lts_1, lts_2, CRel)$ according to the definition of behavior weaving. Firstly, rename the action $a$ of $P_1$ as $b$ and $a$ of $P_2$, as $c$, i.e. $P_1$ becomes $P_1(b)$, then $P_2$ is the LTS of $P_2(b)$.

Behavior Weaving Semantics. Given LTS $lts_1 = (S_1, \Delta_1, T_1, \Delta_1)$, $lts_2 = (S_2, \Delta_2, T_2, \Delta_2)$, a crossing relation $CRel$ on $lts_1$ and $lts_2$, and $rlts = \Theta_{\Delta}(lts_1, lts_2, CRel)$, then its semantics is as follows:

1) if $a \in A_1 \cap \phi(CRel)$ and there exists a transition $s <_{\Delta_1} s'$ in $T_1$, then there exists a transition $s <_{\Delta_2} s'$ in $T_2$ such that $s' \in \phi(s)$.
2) if $a \in A_2 \cap \phi(CRel) \cap \phi(CRel)$ and there exists a transition $s <_{\Delta_1} s'$ in $T_1$, then there exists a transition $s <_{\Delta_2} s'$ in $T_2$ such that $s' \in \phi(s)$.
3) if $a \in \Delta_2 \cap \phi(CRel) \cap \phi(CRel)$ and there exists a transition $s <_{\Delta_1} s'$ in $T_1$, then there exists a transition $s <_{\Delta_2} s'$ in $T_2$ such that $s' \in \phi(s)$.
4) if $a \in \phi(CRel)$, then there exists a transition $s <_{\Delta_2} s'$ in $T_2$ such that $s' \in \phi(s)$.

Figure 4. Illustration of aspect behavior weaving.
Weaving.

As an example to illustrate the proposed notions and models.

A customer requires the system to search and return the catalog to the customer and send the order.

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.

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 $rsar \oplus \Omega (Bsa, Asa, JP, CR)$, then $rsar.Beha = \oplus \Omega (Bsa.Beha, Asa.Beha, BCR)$, where $BCR = \langle a, b, e \rangle$, $i, o \in CR$, $a$ is the name in $bsa.Beha$ 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. $Bsa.Beha = (M.Beha)[N.Beha](f)$

$\quad = M.Beha(s \rightarrow o)[N.Beha(t \rightarrow a)]$, $f_1 = \{s \rightarrow a, t \rightarrow a\}$;

Asa.Beha $= (P.Beha)[Q.Beha](f_2)$

$\quad = P.Beha(u \rightarrow d)[Q.Beha(v \rightarrow d)]$, where $f_2 = \{u \rightarrow d, v \rightarrow d\}$;

$\quad CR = \{\langle a, t, o \rangle, A, i, A, a \}$.

According to $CR = \{\langle M, s, N, d, A, i, A, o \rangle\}$, $f_3 = \{s \rightarrow a, t \rightarrow a\}$, we can get that $BCR = \{\langle a, i, o \rangle\}$. According to the definition of behavior weaving, we have:

$rsa.Beha = \oplus \Omega (Bsa.Beha, Asa.Beha, BCR)$

$\quad = M.Beha(s \rightarrow o)[N.Beha(t \rightarrow a)]$,

$\quad = \{P.Beha[(u \rightarrow d)(Q.Beha(v \rightarrow d)), (a, i, o)]\}$

$\quad = (M.Beha(s \rightarrow a)(a \rightarrow i))[N.Beha(t \rightarrow o)(a \rightarrow o)]$;

$\quad = P.Beha(u \rightarrow d)(Q.Beha(v \rightarrow d))$;

$\quad = (M.Beha)[N.Beha][P.Beha](Q.Beha)(s \rightarrow i, t \rightarrow a, u \rightarrow d, v \rightarrow d)$ (2).

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.

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. $Con$ provides the interaction between $Cus$ and $Sup$. Take the $Cus$ for instance, its interface has four actions $I_1, I_2, I_3$, and $I_4$ which represent sending browsing catalog request receiving catalog, sending order requests and receiving the order request 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. $I = \{Cus, Sup, Con\}$;
2. $Cfg = \{<Cus.O_1, Con.I_2>, <Cus.O_1, Cus.I_1>, <Cus.O_2, Con.I_2>, <Cus.O_2, Sup.I_1>, <Cus.O_2, Cus.I_2>, <Cus.O_2, Sup.I_2>, <Sup.O_2, Con.I_2>, <Sup.O_2, Cus.I_2>\}$;
3. $Bhea = (\{Bhea\}[\{Bhea\}][\{Sup.Bhea\}]$,

Fig.6 is the flow graph of the e-commerce system behavior.

Behavior of each element is as follows:

$Bhea.P_1$;

Sup.Bhea $= P_2$.

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 $aout$, 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.
the access control aspect), which contains an aspect component AccCtrl and an aspect connector Ac2. The aspect component AccCtrl inputs data and outputs the results through its interface aout and ain. 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. Asa1.I = {Ac1.I1, Ac1.O1};
   Asa1.Elem = {Ac1, Logging};
   Asa1.Cfg = {<Ac1, aout, logging. ain>};
2. Asa2.I = {Ac2.I1, Ac2.O1, Ac2.I2, Ac2.O2};
   Asa2.Elem = {Ac2, AccCtrl};
   Asa2.Cfg = {<Ac2, aout, AccCtrl. ain>,<AccCtrl. aout, Ac2. ain>};

Figure 7. Aspect architecture models of the example.

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:

<table>
<thead>
<tr>
<th>JPs</th>
<th>CRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPs1 = {&lt;Con O2, Con I2&gt;};</td>
<td>CRs1 = {&lt;Con O2, Con I2&gt;, Asa1.I1, Asa1.I2, O1, O2};</td>
</tr>
<tr>
<td>JPs2 = {&lt;Con O2, Sup I2&gt;;&lt;Sup O1, Con I2&gt;};</td>
<td>CRs2 = {&lt;Con O2, Sup I2&gt;,&lt;Con O2, Sup I2&gt;, Asa1.I1, Asa1.I2, O1, O2};</td>
</tr>
<tr>
<td>JPs3 = {&lt;Sup O1, Con I2&gt;;&lt;Asa1.I1, Asa1.I2, Asa2.O2&gt;};</td>
<td>CRs3 = {&lt;Sup O1, Con I2&gt;,&lt;Asa1.I1, Asa1.I2, Asa2.O2&gt;};</td>
</tr>
</tbody>
</table>

Firstly, the logging aspect is woven into the base model through operation rsa1=⊕a(Asa1, Asa2, JPs1, CRs1). Then, the access control aspect is inserted to the model rsa1 through operation rsa2=⊕a(rsa1, Asa2, JPs2, CRs2).

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

1. rsa1.I = Bsa.I = O;
   rsa1.Elem = Bsa.Elem ⊕ asa1.Elem
   = {Sup, Con, Logging, Ac1};
   rsa1.Cfg = (bsa.Cfg \{JPs1\}.asa1.Cfg \{Con(CR1)\});
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 resulting 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 weaving 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


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).