2008 International Conference on Computer Science and Software Engineering

Towards a Framework for Reasoning about Aspect Weaving Impact

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

***Abstract*—This paper presents a framework for reasoning about the semantic impact of aspect weaving at the level of early design modeling. The framework is based on semantic consistency between a model and its projection in the woven model. If a weaving preserves the semantic consistency between the model and its projection, then it has no impact on the model. The underlying formalisms are Process Algebras. Firstly, notations for aspect weaving are given. Then, semantic preserved weaving is defined, through which the semantic impact of aspect weaving can be reasoned about. Understanding the impact of weaving can aid developers in foreseeing unintended aspect impacts and increase the reliability of the software, which is especially vital for aspect oriented system refinements.**

Keywords- aspect weaving; impact; reasoning about; process algebra; early design

I. INTRODUCTION

To achieve better separation of crosscutting concerns, aspect-oriented concepts are currently introduced in all phases of the software development life cycle. At the design level, several aspect-oriented approaches[1][2][3][4] have been proposed to modularize cross-cutting concerns.

However, aspects should be used with care as superimposing aspects on software modules may cause side effects, sometimes in a harmful way that is unexpected[11]. Nowadays, to promote understanding the effects of aspects, many methods for promoting modular reasoning or aspect interactions have been proposed for the programming level[7][8][9][10]. The programming techniques cannot be used immediately for the design level because they rely on the operational specification of the complete behavior as given by the code, while designs abstract from these details. At the design level, some researches focused on checking the impact of weaving on the desired system properties through model checking[11][12]. However, they provide no way for reasoning about the overall semantic impact of weaving on the base model that it applies to.

This paper presents a framework for reasoning about the semantic impact of aspect weaving at the level of early design modeling. The framework is based on semantic consistency between a model and its projection in the woven model. The underlying formalisms are Process Algebras. Firstly, notations for aspect weaving are given. Then, semantic preserved weaving is defined.

The rest of the paper is organized as follows: In section 2, the framework is outlined. Then, the aspect weaving and semantic impact of weaving is defined in section 3 and section 4 separately. Finally, section 5 describes the conclusion.

II. OUTLINE OF THE FRAMEWORK

At early design level, designs for concerns are mainly based on models. The design model for core concern is called the base model, while designs for the crosscutting concerns are aspect models. In essence, the weaving of an aspect model and a base model is a certain composition of the two models. In the woven model, the two models interweave with each other according to certain methods or rules. The original semantics of a model would be altered in the woven model. Here, we interpret the semantics as the behavior of the model. Therefore, through checking consistency between a model's semantics in the woven model and its original semantics, impact of weaving on the model can be got. Grounded on this, our framework is outlined in Fig.1.

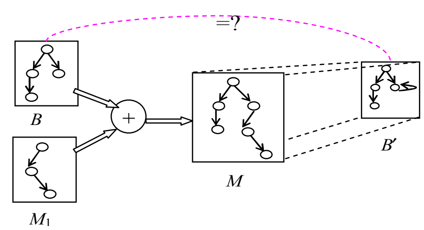

Figure 1. Outline of the framework.

In Fig.1, model M results from weaving of aspect model M_1 into the base model B. B' are projections of M on B. Here, the projection represents a model's behavior in the woven model. If B is semantic consistent with B', then the weaving is semantics-preserved. A semantics-preserved weaving has no impact on the base model that it applies to, which is the basis for reasoning.

III. NOTATIONS

A. Related PA Notions and Notations

The underlying formalisms of the framework is Process Algebra[5].

***Definition* 3.1** The collection of process terms of the Process Algebra is generated by the following grammar:

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

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

Semantically, a PA process term corresponds to a Labeled Transition System (LTS).

***Definition* 3.2 [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.

Notation 3.1. For a labeled transition system <S, A, T>, we use the following notations:

$-s \xrightarrow{a} s'$ for every transition $(s, a, s') \in T$; and

$-s \overset{w}{\Rightarrow} s'$ for $s \xrightarrow{\tau^m} s'' \xrightarrow{w} s''' \xrightarrow{\tau^n} s'$ where $m, n \in \mathrm{N}$; and

$-\hat{a} = a$ for $a \neq \tau$, $\hat{a} = \varepsilon$ for $a = \tau$ where $a \in A$.

In addition, to make it convenient for discussion, we assume that:

1. A relabeling function f is denoted as $f = \{a \mapsto b, \ldots\}$, where "$\mapsto$" represents "is relabeled as" ; and
2. Operations on PA terms are also applicable for LTSs. For example, suppose LTS M_1 and M_2 correspond to two PA terms P_1 and P_2. Then, "$M_1 \| M_2$" represents the parallel composition "$P_1 \| P_2$"; and
3. For a LTS $M = e_1 \| \ldots \| e_n$ for $n \geq 1$, assume that $M = (S, A, T)$ and $e_i = (S_i, A_i, T_i)$ for $i \in (1..n)$. Then, each state $s \in S$ is denoted as a tuple $<s_1, \ldots, s_i, \ldots, s_n>$ in which $s_i \in S_i$. Moreover, for any action $a \in A$, if $a \in A_i$ and $a \in A_j$, then action a is called a *synchronized* action.

B. Definition of Models

At the initial design level, software architecture provides a model of the system. Therefore, not only the base model but also aspect models can all be abstracted as software architecture. Behaviorally, software architecture can be modeled as a labeled transition system[6]. Architecture elements can be interpreted as PA terms, while connections among elements are interpreted as parallel compositions of PA terms.

***Definition* 3.3 [Model Element]**. A model element e is a labeled transition system (S, A, T) where $A = A^I \cup A^V \cup \{\tau\}$:

1. A^I is an interface action set;
2. A^V is an observable action set which includes all actions in A^I and some inner actions;
3. τ represents any unobservable inner actions.

***Definition* 3.4[Model]**. A model m is a labeled transition system (S, A, T) that is a parallel composition of model elements $e_1, \ldots, e_n$, i.e. $m = e_1 \| \ldots \| e_n$ ($n \geq 1$).

In *Def.*3.3 and *Def.*3.4, the action set is defined as $A = A^I \cup A^V \cup \{\tau\}$ which aims to distinguish between interface actions and observable actions. In the notion of aspect weaving, all observable actions in a base model are candidate positions where an aspect would insert. Such observable actions include interface actions and some inner actions. In the following sections, we use "$M.A^I$" or "$M.A^V$" to indicate action set A^I or A^V in model M.

For example, in Fig.2(1), model B is the base model of an imaginary system. Element E_1 sends requests (represented as action j) to element E_3 for certain services, and element E_2 is a communication component. Observable action j and k are candidate join points. Model M_1 is a model of an encryption/decryption aspect, whose description is depicted in the box labeled M_1. Aspect M_1 integrates two advices: encryption and decryption. Advice encryption inputs data to be encrypted through interface action a, then outputs encrypted data through interface action b. Similarly, Advice decryption inputs data to be decrypted through interface action c and then outputs decrypted data through interface action d. Moreover, there exists implicated constraints between the two advices, i.e. decryption should and must execute after encryption.

C. Definition of Aspect Weaving

Given a base model and an aspect model, aspect weaving is to build the behavioral crosscutting relation between the two models. Such a crosscutting is essentially the *caller-callee* relationship between join points and its corresponding advices.

We define a *join point* as an observable action in the base model because an observable action that activates state transitions is a basal observable point in execution of the base model. For example, in Fig.2(1), advice encryption of aspect M_1 would inserts before the request flows into the communication element E_2, while advice decryption would inserts after the request flows out of E_2. Thereby, action j and k are join points for aspect M_1.

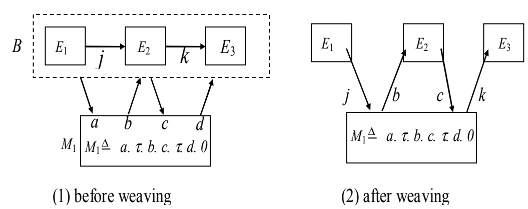

Figure 2. Illustration of aspect weaving.

To build the caller-callee relationship, connections between join points and the corresponding interface actions of advices in the aspect model should be rebuilt. Such a process can be implemented through the relabeling operation in PA. Moreover, to make it convenient for discussion, it is assumed that only join points and the corresponding advice interface actions can be relabeled. Furthermore, join points can only be relabeled as its connected advice interface action, and vice versa.

For example, in Fig.2(1), advice *encryption* can be inserted to join point j through connecting j with the corresponding advice interface actions a and b, which can be described as the following two relabeling operations:

$$a \mapsto j,\ E_2.j \mapsto b.$$

Noted that here "$E_2.j|\rightarrow b$" is used to assure the action j of E_2 is relabeled because j is a synchronized action between E_1 and E_2.

Essentially, such relabeling operations are micro codes for weaving an advice. Through a list of such codes, aspect weaving is implemented. The following *Def.*3.6 defines aspect weaving formally.

***Definition* 3.5 [unambiguous].** Given any two action set A_1 and A_2 and a relabeling relation $f \subseteq A_1 \times A_2$, then f is *unambiguous* iff $\forall (a_1|\rightarrow b_1),(a_2|\rightarrow b_2) \in f$, whenever $a_1 = a_2$ then $b_1 = b_2$.

***Definition* 3.6 [Weaving $\angle_f$].** Given a base model B, an aspect model M_1, and a relabeling relation $f \subseteq (B.A^V \times M_1.A^I) \cup (M_1.A^I \times B.A^V)$, then $(B||M_1)[f]$ is the *weaving* of aspect M_2 to M_1 iff f is *unambiguous*, which is denoted as $B\angle_f M_1$.

In *Def.* 3.6, it is assumed that any action names in the base model and the aspect model are divisible. Relation f is a set of relabeling codes which satisfies $f \subseteq (JP \times I) \cup (I \times JP)$, where $JP \subseteq B.A^V$ is a set of join points and $I \subseteq M_1.A^I$ is a set of interface actions of advices in aspect M_1. Moreover, in the relabeling relation f, its relabeling codes are order sensitive. In addition, any join point in $B.A^V$ can not be relabeled as two more actions in $M_1.A^I$, i.e. a relabing relation f should be *unambiguous*. This aims to assure the feasibility of weaving.

According to such a definition, the weaving as shown in Fig.2 (2) can be defined as:

$B\angle_f M_1$, where $f=\{a|\rightarrow j, E_2.j|\rightarrow b, d|\rightarrow k, E_2.k\rightarrow c\}$.

IV. Definition of Semantic Impact of Weaving

***Definition* 4.1 [Projection].** Given models $M_1=<S_1, A_1, T_1>$, $M_2=<S_2, A_2, T_2>$, and $M=<S, A, T>$ in which $M=M_1\angle_f M_2$. Then, define $M/(A-(A_1\text{-dom}f \cup A_1')-\{\tau\})$ as the *projection* of M on M_1 which is denoted as $\nabla^{M}_{f\ M_1}$, where $A_1'=\{f(a)|\ a \in \text{dom}f \cap A_1\}$.

Note that here "$(A_1\text{-dom}f \cup A_1')$" represents actions that belong to M_1 in the woven model M because some actions in "$A_1 \cap \text{dom}f$" have been relabeled as actions in A_1' after the weaving.

The projection of a model M on M_1 makes actions not belonging to M_1 unobservable. In other words, from the behavior projection $\nabla^{M}_{f\ M_1}$, we can see the behavior of M_1 in M.

Take the example shown in Fig.2 for illustration. From the description of aspect model M_1, its action set $A_1=\{a, b, c, d, \tau\}$, $A_1 \cap \text{dom}f=\{a, d\}$, $A_1'=\{j, k\}$, $A_1\text{-dom}f \cup A_1'=\{j, k, b, c, \tau\}$, so we have:

$$\nabla^{B\angle_f M_1}_{f\ M_1} = (B\angle_f M_1)/(Act(B\angle_f M_1)-(A_1\text{-dom}f \cup A_1')-\{\tau\})$$
$$= (B\angle_f M_1)/(Act(B\angle_f M_1)-\{j, k, b, c\}-\{\tau\}).$$

From the projection, we can see the behavior of aspect model M_1 in the woven model. Here the term $Act(B\angle_f M_1)$ refers to the action set of model $B\angle_f M_1$.

According to their definition, a model and its projection are two LTSs. Semantics between two LTSs with the same action set can be compared using weak bisimulation in PA. However, the action set of a model is not the same as that of its projection as some actions may be relabeled after aspect weaving. But, if we build the corresponding relationship between actions of the model and its corresponding relabeled actions in its projection, then we can check their semantic consistency (equivalence) by borrowing the idea of the weak bisimulation. The following definition of semantic consistency is based on such ideas, which is also illustrated in Fig.3.

***Definition* 4.2 [Semantic Consistency].** Given models $M_1=<S_1, A_1, T_1>$ and $M_2=<S_2, A_2, T_2>$, and a relabeling function f, then M_1 is (*external observational*) *semantic consistent with* M_2 according to f iff there is a binary relation R over S_1 and S_2 that satisfies:

— whenever $(s_1, s_2) \in R$ and $s_1 \xrightarrow{a} s_1' \in T_1$, then:

- $s_2 \overset{\hat{a}}{\Rightarrow} s_2' \in T_2$ for $a \in A_1\text{-dom}f$ or $a \in A_1 \cap \text{ran}f$, and $(s_1', s_2') \in R$;
- $s_2 \overset{\widehat{f(a)}}{\Rightarrow} s_2' \in T_2$ for $a \in A_1 \cap \text{dom}f$, and $(s_1', s_2') \in R$;

— whenever $(s_2, s_1) \in R$ and $s_2 \xrightarrow{a} s_2' \in T_2$, then:

- $s_1 \overset{\hat{a}}{\Rightarrow} s_1' \in T_1$ for $a \in A_1\text{-dom}f$ or $a \in A_1 \cap \text{ran}f$, and $(s_2', s_1') \in R$;
- $s_1 \overset{\widehat{f^{-1}(a)}}{\Rightarrow} s_1' \in T_1$ for $a \in \text{ran}f\text{-}A_1$, and $(s_2', s_1') \in R$.

Notation: M_1 is *semantic consistent with* M_2 according to f is denoted as $M_1 \overset{f}{\approx} M_2$.

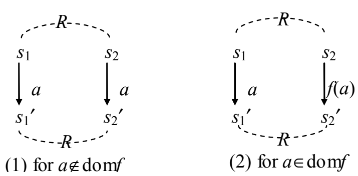

Figure 3. Illustration of semantic consistency.

However, we can not use the *Def.*4.2 immediately to check the consistency between a model and its projection in the woven model. Given a base model $M=e_1||\ e_2$, a synchronized action j, an aspect model A and $f=\{e_1.j|\rightarrow x,\ y|\rightarrow j\}$, then $M'=M\angle_f A=e_1[j|\rightarrow x]||\ e_2||\ A[y|\rightarrow j]$. State transitions graphs for model M and the woven model M' are as shown in Fig.4(1) and Fig.4(3) separately. We cannot use the *def.*4.2 to check the semantic consistency between M and M' as state $<s_1', s_2, s_2'>$ in M' has no corresponding states in M.

To overcome the problem, we make extensions for M by introducing a temporary state between $<s_1, s_2>$ and $<s_1', s_2'>$ as shown in Fig.4(2). The extension model M'' is semantic consistent with M. So, the semantic consistency detection between M and M' can be done instead between M'' and M'.

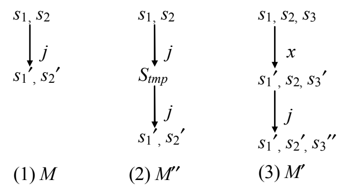

Figure 4. An example.

***Definition* 4.3[Model Extension].** Given a model M=<S, A, T> and a relabeling function f, then an extension on a model M according to f is a model <S_e, A_e, T_e> that achieved by the following steps:

1 $A_e = A$;

2 For each action $a \in A$ and transition $s \xrightarrow{a} s' \in T$:
if a is a synchronized action, then introduce a state s_{tmp} that is different from any state in S, and $s \xrightarrow{a} s_{tmp} \in T_e$ and $s_{tmp} \xrightarrow{a} s' \in T_e$ and s, s_{tmp}, $s' \in S_e$;
else $s \xrightarrow{a} s' \in T_e$ and s, $s' \in S_e$.

Extension of M on M_1 according to f is denoted as $E_f(M)$.

***Definition* 4.4[Semantic Preserved Weaving].** Given a base model B, an aspect model M_1, and model $M=B\angle_f M_1$, then the weaving of $B\angle_f M_1$ is *semantic preserved* iff $E_f(B) \overset{f}{\approx} \nabla_{f\,B}^{M}$.

The semantic preserved weaving ensure the semantic consistency between the base model(or an aspect model) and its projection. So, it has mo impact on the base model and aspect model, which is the basis for detection and reasoning of impact of weaving.

Reconsider the example shown in Fig.2. Suppose that the base model B is depicted as follows:

$$B \triangleq E_1 \parallel E_2 \parallel E_3;\ E_1 \triangleq j.\,x\ 0;\quad E_2 \triangleq j.k.0;\quad E_3 \triangleq k.y.0.$$

Then, according to *Def*.4.4, we can get that $E_f(B) \overset{f}{\approx} \nabla_{f\,B}^{M}$, i.e. the encryption/decryption aspect has no impact on behavior of the base model.

V. Conclusion

The paper presents a framework for reasoning about semantic impact of weaving at earlier design level. The framework is based on semantic consistency between a model and its projection in the woven model.

As the underlying weaving model assumes that the relationship between aspects and the core be caller-callee relationship, the framework is applicable for aspects that own certain functions and provide auxiliary computation for the base model. Lots of aspects in real applications such as logging, tracing, counting, security, communication etc belong to such categories.

For complicate aspects that cannot be expressed in the proposed weaving model, if only the woven model and the consistency between a model and its projection can be defined, the framework can also be applicable. This is the future work.

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, S. Ghosh, “Aspect-Oriented Approach to Early Design Modeling”, *IEE Software*, 2004, 151(4):173-186.

[3] J. Pérez, I. Ramos, J. Jaén, P. Letelier, 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. 118–137. Springer-Verlag.

[5] 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.

[6] R. Allen, D. Garlan, “A formal basis for architectural connection”. *ACM Transactions on Software Engineering and M ethodology*, 1997, 6(3):213-249.

[7] G. Kiczales, M. Mezini, “Aspect-oriented programming and modular reasoning”, In *Proc. of the 27th International Conference on Software Engineering (ICSE'05)*, May 15-21, 2005, St. Louis, MO, USA, pp.49-58.

[8] J. Aldrich, Open Modules: Modular Reasoning about Advice. In: *Proc. 2005 European Conf. Object-Oriented Programming (ECOOP 05), LNCS 3586*, Springer, pp. 144-168, 2005.

[9] M. Rinard, A. Salcianu, S. Bugrara, “A Classification System and Analysis for Aspect-Oriented Programs”, In *Proc. of the ACM SIGSOFT 2004 Symposium on the Foundations of Software Engineering (FSE'04)*, Newport Beach, California, November 2004.

[10] H. Shinomi, T. Tamai, “Impact Analysis of Weaving in Aspect-Oriented Programming”, *In Proceedings of the 21st IEEE International Conference on Software Maintenance(ICSM’05),* Budapest, Hungary, Sept 25-30, 2005, pp. 657 – 660

[11] D. Xu, I. Alsmadi, and W. Xu, “Model Checking Aspect-Oriented Design Specification”, *In Proceedings of the 31st IEEE International Computer Software and Applications Conference (COMPSAC'07),* 2007, Beijing, China, pp.491 – 500.

[12] F. Mostefaoui and J. Vachon, “Design-level Detection of Interactions in Aspect-UML models using Alloy”, *Journal of Object Technology*, vol. 6, no.7, Special Issue: Aspect-Oriented Modeling, August 2007, pp.137–165.