Analyzing The Influences of Aspect Weaving on Software System Behavior

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

Abstract—The aspect-oriented paradigm can help promoting system modularity and maintainability by separating crosscutting concerns as aspects. However, weaving an aspect may introduce undesired impacts on the original system behavior. We present an approach to analyzing the influences of an aspect weaving on the base model. The analysis is based on the behavioral equivalence between the base model and its projection in the woven model. An example shows the effectiveness.

Index Terms—aspect influences, aspect interference, aspect weaving, detection, aspect-oriented software development

I. INTRODUCTION

The aspect-oriented paradigm[1] was first introduced in 1990s to support the modularization of the crosscutting concerns of a system. Currently, it has been extended to the early modeling stage of the software development lifecycle with the aim of reducing complexity and enhancing maintainability early on [4, 5]. An aspect-oriented software system model generally consists of the base model and a set of aspects. The base model encapsulates the business concern, while aspects encapsulate crosscutting concerns [5]. The two types of models are developed separately and combined together through a process of weaving.

However, the separate development of aspects may introduce semantic problems in the process of weaving [6, 7, 8, 14]. The weaving of a new aspect may introduce undesired impacts on the behavior of the base model that it applies to[8], which threaten the reliability of the software.

Many attempts have been made to check the aspect weaving influences at programming level [6, 7, 8, 9] and the requirements level [10]. In this paper, we confine the discussion to the early design level of the software lifecycle. In contrast to other researches on the impacts of aspect weaving on desired system properties[11, 12, 13] at the design level, our approach focus on weaving influences on the behavior of a model. The behavior means the trace of the model’s execution instead of system properties. We present an approach to detect the influences of aspect weaving on the base model that it applies to. Its underlying formalisms are PA algebras.

The rest of the paper is organized as follows: In section 2, the approach is presented. Thereafter, section 3 describes an example. Section 4 summarizes the conclusions and future work.

II. THE APPROACH

The influences of an aspect weaving on the base model can be detected by comparing the consistency between its original behavior and the actual behavior in the woven model. As Process Algebra [2] is a popular tool for modeling software system behavior[3]. Moreover, its notion of behavior equivalence makes is feasible for comparing the semantic relationship between two models in our approach. Therefore, our approach has Process Algebra as the underlying formalism. In this section, we firstly give a brief introduction to PA. Then we define the base model, aspects and aspect weaving. Lastly, we define the influences of aspect weaving on the base model.

A. An introduction to PA

Assuming an infinite collection A of names, the set $\bar{A} = \{ \bar{a} | a \in A \}$ is the set of complementary names (or co-names for short). In the paper, names are interpreted as input actions and co-names as output actions. Let $Act = A \cup \bar{A} \cup \{ \tau \}$ be the set of actions, where action $\tau$ is a distinguished unobservable action or inner action.

Definition 1 The collection of process terms of the Process Algebra is generated by the following grammar:

\[ P ::= 0 \| a.P \| P + P \| P[\lambda_f \ P[\lambda_g] \ K] \]

where $a$ is an action in Act, $f$: Act$\rightarrow$Act is a relabelling function, $L \subseteq Act^{\tau}$ is a set of labels, and $K$ is a constant possessing a defining equation of the form $K \perp P$.

The semantics for PA is defined in the standard operational style by means of a set of axioms and inference rules. The result of the application of the operational semantic rules is a Labeled Transition System, where states are in correspondence with process terms and transitions are labeled with actions.

Definition 2 A labeled transition system (LTS) is a tuple $(S, Act, T, s_{init})$, where $S$ is a set of states which include an initiate state $s_{init}$, Act is a set of actions, $T \subseteq S \times Act \times S$ is a transition relation. We write $s \rightarrow s'$ for a transition $(s, a, s') \in T$.

Let $P$ and $Q$ be processes. We write $P \Rightarrow Q$ iff there is a (possibly empty) sequence of $\tau$ -labeled transitions that leads from $P$ to $Q$. (If the sequence is empty, then $P = Q$.) For each action $a$, we write $P \Rightarrow^a Q$ iff there are processes $P'$ and $Q'$ such that $P \Rightarrow^a P' \Rightarrow^\epsilon Q' \Rightarrow^\epsilon Q$. 

Corresponding author: Chunhua Yang

© 2009 ACADEMY PUBLISHER
AP-PROC-CS-09CN001

234
For each action \(a\), we use \(\hat{a}\) to stand for \(\varepsilon\) if \(a = \tau\), and for \(a\) otherwise.

**Definition 3** [Weak Bisimulation] A binary relation \(R\) over the set of states of an LTS is a weak bisimulation iff whenever \(s_1, R s_2\) and \(a\) is an action:

- if \(s_1 \xrightarrow{a} s_1'\), then there is a transition \(s_2 \xrightarrow{a} s_2'\) such that \(s_1' R s_2'\);
- if \(s_2 \xrightarrow{a} s_2'\), then there is a transition \(s_1 \xrightarrow{a} s_1'\) such that \(s_2' R s_1'\).

Two states \(s\) and \(s'\) are observationally equivalent (or weakly bisimilar), written \(s \approx s'\), iff there is a weak bisimulation that relates them.

**B. Definition of models**

We define the base model and advice models as labeled transition systems.

**Definition 4** A base component \(\text{base} = (S, \text{Act}, T, \text{sinit})\) is a LTS.

**Definition 5** A base model \(m = (S, \text{Act}, T, \text{sinit})\) is a LTS that is a parallel composition of the base components \(e_1, \ldots, e_n\), i.e. \(m = e_1 \parallel \cdots \parallel e_n\) \((n \geq 1)\).

**Definition 6** [Join Point] Given a base component \(Bc = (S, \text{Act}, T, \text{sinit})\), a join point \(jp\) is a state \(s \in S\) or a transition \((s, a, s') \in T\) where \(a\) is an observable action. If a join point is a transition, we call it transition join point. Otherwise, we call it state join point.

According to observations on principles of some aspects proposed in the literature[1,5,14], we identify three advice types:

- **sequential**: A sequential advice applies to a transition join point. The advice should run completely before (or after) a transition join point. In other words, the advice begins to run when the join point is active. After the advice stops, i.e. arrives at a final state, the join point executes (or continues) immediately.

- **branched**: A branched advice applies to both a transition join point and a state join point. The advice executions before (or after) the transition join point. If the conditions set by the advice satisfy, then the advice runs to a true final state and the transition join point executes immediately. Otherwise, the advice runs to a false final state and the state join point becomes active.

- **synchronized**: A synchronized advice is an advice that applies to transition join points. Execution of a synchronized advice and the base model are concurrent according to the synchronization rules: a transition join point synchronizes with a predefined observable transition of the advice.

**Definition 7** [Advice] An advice \(ad = (\text{Type, Beha})\) consists of a type \(\text{Type}\) and a behavior \(\text{Bhea}\), in which \(\text{Type} = \{\text{sequential, branched, synchronized}\}\) and \(\text{Bhea} = (S, \text{Act}, T, \text{sinit}, \text{Sfinal})\) is an extended LTS. An extended LTS introduces a final state set \(\text{Sfinal}\) to the base LTS. A final state is a state from which no transitions direct. Moreover, \(\text{Beha}.\text{Sfinal} = \emptyset\) for \(\text{Type} = \text{sequential, branched}\), \(\text{Beha}.\text{Sfinal} = \emptyset\) for \(\text{Type} = \text{sequential}\).

**Definition 8** [Aspect] An aspect \(Ap = \{jp_1, ad_1\}, \ldots, \{jp_n, ad_n\}\) \((n \geq 1)\) is a set of pairs of an advice and a join point set where the advice would apply.

**C. Definition of aspect weaving**

According to the advice types, we define five operators \((\mathcal{Z}_{\text{seq}}, \mathcal{Z}_{\text{bran}}, \mathcal{Z}_{\text{syn}})\) to weave the sequential advices, branched advices, and synchronized advices respectively. Moreover, we use \(M = \mathcal{Z}_{\text{op}}(Bc, jp_x, ad)\) to denote the advice weaving, where:

- \(M\) is a LTS that represents the woven model derived from the weaving; and
- \(\mathcal{Z}_{\text{op}}\) is the weaving operator, \(ad\) is an advice, \(Bc\) is the base component that \(jp_x\) belongs to; and
- \(jp_x = s \rightarrow s\) stands for a transition join point for a sequential advice; and
- \(jp_x = \{s \rightarrow s', \text{sfalse}\}\) stands for a pair of a transition and a state join point for a branched advice; and
- \(jp_x = \{s \rightarrow s', \text{sfalse}\}\) stands for a set of pairs of a join point and the corresponding synchronized transition in \(ad\) for a synchronized advice.

In the following definition, it is assumed that the aspect and base model have no identical names.

Fig.1 illustrates the two types of sequential advice weaving. Take the before-sequential weaving for instance (see Fig.1(2)). The weaving process is implemented through the following steps:

1. introduce a new state \(s_{\text{new}}\) to \(Bc\) and replace the transition \(s \rightarrow s'\) with \(s_{\text{new}} \rightarrow s'\);
2. introduce a \(\varepsilon\)-labeled transition \(s \rightarrow t\) from state \(s\) of \(Bc\) to the initial state \(t\) of \(ad\);
3. introduce a \(\varepsilon\)-labeled transition \(t_{\text{final}} \rightarrow s_{\text{new}}\) from the final state \(t_{\text{final}}\) of \(ad\) to the new state \(s_{\text{new}}\) of \(Bc\).

Note: the \(\varepsilon\)-labeled transition \(P \rightarrow Q\) means \(P = Q\).

Formally, let \(M = \mathcal{Z}_{\text{seq}}(Bc, s \rightarrow s', ad)\), then:

- \(M.S = Bc.S \cup \{s_{\text{new}}\}\cup ad.\text{Bhea}.S, M.S_{\text{init}} = Bc.\text{sinit}, M.\text{Act} = Bc.\text{Act} \cup \text{ad.}\text{Act}\), and
- \(M.T = (Bc.T - \{s \rightarrow s'\} \cup \{s_{\text{new}} \rightarrow s', s \rightarrow t, t_{\text{final}} \rightarrow s_{\text{new}}\} \cup ad.\text{Bhea}.T\).

The process of the after-sequential weaving is similar to the before-sequential weaving, which is as shown in Fig.1(3).

Given a branched advice \(ad\) which a true final state \(t_{\text{final}}\) and a false final state \(t_{\text{final}}'\), the before-branched weaving is specified as follows. The after-branched advice
weaving can be achieved by analogy. Formally, let \( M = \angle_{\text{base}} (BC, \{ a \rightarrow s', s_\text{false} \}, ad) \), then:

- \( M.S = BC.S \cup \{ s_\text{new} \} \cup \text{ad}.Beha.S \), \( M.S_{\text{init}} = BC. s_{\text{init}} \), \( M. Act = BC. Act \cup \text{ad}. Act \), and

- \( M.T = (BC.T \cdot \{ \text{new} \rightarrow s' \} \cup \{ \text{new} \rightarrow s' \}, s_{\text{init}} \cdot t_1 \rightarrow s_{\text{new}} \cdot t_f \rightarrow s_{\text{false}} \), \( \text{ad}. \text{Beha}.T \).

As for a synchronized advice's weaving, it is implemented by parallel composition of PA. Formally, let \( ad \) is a synchronized advice and \( M = \angle_{\text{syn}} (BC, \{ <s_1 \rightarrow s'_1, t_1 \rightarrow t'_1, \ldots, <s_n \rightarrow s'_n, t_n \rightarrow t'_n > \}, ad) \), then:

- \( BC. Act = BC. Act \cup \{ b_1, \ldots, b_n \} \), \( BC.T = (BC.T \cdot \{ <s_1 \rightarrow s'_1, t_1 \rightarrow t'_1, \ldots, <s_n \rightarrow s'_n, t_n \rightarrow t'_n > \}, \text{ad} \).

- \( M = BC \upharpoonright ad \).

The weaving of an aspect is achieved by weaving its advice in turn. Given an aspect \( A \) and a base model \( B \), we use \( B \triangleleft A \) to denote the process of weaving \( A \) to \( B \) in the following sections.

\[ \text{Definition 9} ] \text{[Projection]} \]

Given a base model \( B = (S, [A, T, s_{\text{init}}]) \) and a woven model \( M = (S', [A', T', s'_{\text{init}}]) \) which satisfies \( \text{Act} \subseteq \text{Act}' \cup \{ \tau \} \), then the projection of \( M \) on \( M \), written as \( \nabla_B^M \), is a LTS \( (S', \text{Act}_{\text{init}}, T', s'_{\text{init}}) \), where

- \( S' = S \), \( \text{Act}_{\text{init}} = \text{Act}_{\text{init}} \cup \{ \tau \} \), \( s'_{\text{init}} = s'_{\text{init}} \), and
- \( T' = \{ \{ s_1, a, s_2 \} | <s_1, a, s_2 > \in T' \wedge (if \ a' \in \text{Act} \cup \{ \tau \} then a = a', otherwise a = \tau) \} \).

According to the definition of projection, actions of the base model are observable while others are invisible. Therefore, projection \( \nabla_B^M \) represents the actual behavior of the pattern \( B \) in the woven model \( M \).

The influence of an aspect weaving on the base model is evaluated according to the semantic relationship between the base model and its projection in the woven model.

\[ \text{Definition 10} \]

Given a base model \( B \), an aspect \( A \), then the weaving \( B \triangleleft A \) does not influence the base model \( B \) iff \( B = \nabla_B^{B \triangleleft A} \).

III. Example

Consider a property listing subsystem (or PLS for short) in an online real estate system. When a seller has a house for sale, he or she should list the property information for publicity through a broker. The work flow is as follows: A broker inputs the property information to the property listing system. Then system verifies the correctness of the provided information. If the verification result is ok, the information is saved to the DB for listing publicity and the broker receives an accepted response. Otherwise, the broker receives a refused response.

Fig.2 is the definition of the base model of the property listing system. The base model PLS is a parallel composition of three base components LDB, PFM and PL.

\[ \begin{align*}
\text{PLS} &= (\text{PL} | \text{PFM} | \text{LDB}) \{ \text{failed}, \text{ok}, \text{verify} \}
\text{PL} &= \text{infoin}. \text{verify}. \text{PL}'
\text{PL}' &= \text{ok}. \text{accepted}. \text{save}. \text{PL} + \text{failed}. \text{refused}. \text{PL}
\text{PFM} &= \text{verify}(\text{ok} + \text{failed}). \text{PFM}
\text{LDB} &= \text{save}. \text{LDB}
\end{align*} \]

\[ \text{Timing1} = \text{starttimer} . \text{endtimer} . \tau. \text{\text{\texttt{\texttt{\texttt{\texttt{\texttt{totaltime}}}}}}}. \text{Timing1} \]

\[ \text{Timing1} = \text{starttimer} . \text{endtimer} . \tau. \text{\texttt{\texttt{\texttt{\texttt{\texttt{totaltime}}}}}}. \text{Timing1} \]

\[ \begin{align*}
\text{PL} &= \text{infoin}. \text{starttimer}. \text{verify}. \text{PL}'
\text{PL}' &= \text{ok}. \text{accepted}. \text{endtimer}. \text{save}. \text{PL} + \text{failed}. \\
\text{refused}. \text{PL}
\text{PFM} &= \text{verify}(\text{ok} + \text{failed}). \text{PFM}
\text{LDB} &= \text{save}. \text{LDB}
\text{Timing1} &= \text{starttimer}. \text{endtimer} . \tau. \text{\texttt{\texttt{\texttt{\texttt{\texttt{totaltime}}}}}}. \text{Timing1}
\text{PLS}_W &= (\text{PL} \mid \text{PFM} \mid \text{LDB} | \text{Timing1}) \backslash \{ \text{failed}, \text{ok}, \text{save}, \text{verify}, \text{starttimer}, \text{endtimer} \}
\end{align*} \]

Figure 2. The base model of the example

Now, an aspect is designed to count the average time of the process of successful property listing. Moreover, it is desired that the original system behavior should be preserved after the aspect is woven.

An aspect Timing1 is designed as shown in Fig.3. The Timing aspect has a synchronized Timing1 advice, which would execute the action starttimer at the join point ‘PL= infoin. verify.PL’ and execute the action endtimer at the join point ‘PL'=ok. accepted. save.PL’.

\[ \text{Timing1} = \text{starttimer} . \text{endtimer} . \tau. \text{\texttt{\texttt{\texttt{\texttt{totaltime}}}}}. \text{Timing1} \]

\[ \begin{align*}
\text{Timing1} &= \text{starttimer} . \text{endtimer} . \tau. \text{\texttt{\texttt{\texttt{\texttt{totaltime}}}}}. \text{Timing1}
\end{align*} \]

The woven model PLW resulted from weaving the Timing1 aspect is depicted in Fig.4, while the projection of the PLS W on PLS is PLW P which is shown in Fig.5.

\[ \begin{align*}
\text{PL} &= \text{infoin}. \text{starttimer}. \text{verify}. \text{PL}'
\text{PL}' &= \text{ok}. \text{accepted}. \text{endtimer}. \text{save}. \text{PL} + \text{failed}. \\
\text{refused}. \text{PL}
\text{PFM} &= \text{verify}(\text{ok} + \text{failed}). \text{PFM}
\text{LDB} &= \text{save}. \text{LDB}
\text{Timing1} &= \text{starttimer} . \text{endtimer} . \tau. \text{\texttt{\texttt{\texttt{\texttt{totaltime}}}}}. \text{Timing1}
\text{PLS}_W &= (\text{PL} \mid \text{PFM} \mid \text{LDB} | \text{Timing1}) \\
& \backslash \{ \text{failed}, \text{ok}, \text{save}, \text{verify}, \text{starttimer}, \text{endtimer} \}
\end{align*} \]

Figure 3. The Timing1 aspects

Through checking, the equivalence “PLS ≈ PLS W P” does not hold, i.e. the Timing1 aspect influences the base model.

Through analysis, there are deadlocks in the woven model after weaving the Timing1 aspect. To overcome the
problem, we design an alternative Timing2 aspect which consists of two advices TimeLog1 and TimeLog2 (see Fig.6). The two advices will log the time at the join points ‘PL = infoin. starttimer’, ‘verify .PL’ ‘PL’ =ok. accepted . endtimer . save .PL + failed. refusel.PL

\[ PFM = \text{verify}(\text{ok} + \text{failed}) \cdot PFM \]

\[ LDB = \text{save}.LDB \]

\[ \text{Timing1} = \text{starttimer}. \text{endtimer}. \tau. \tau.\text{Timing1} \]

\[ PLS\_W\_P = (PL \mid PFM \mid LDB \mid \text{Timing1}) \]

\{ failed, ok, save, verify, starttimer, endtimer \}

Figure 5. The projection of woven model PLS_W on PLS

\[ PL = \text{infoin. starttimer} \cdot \text{verify}.PL' \]

\[ PL' = \text{ok. accepted} \cdot \text{endtimer} \cdot \text{save} \cdot PL + \text{failed. refused}.PL \]

Through checking, the equivalence “PLS \approx PLS\_W\_P” holds, in which PLS\_W\_P is the projection of the woven model resulting from weaving the Timing2 aspect on PLS. Therefore, the base model stay unaltered after the weaving, i.e. the Timing2 aspect satisfies the requirements.

IV. CONCLUSION AND FUTURE WORK

In this paper, we have presented an approach to detect semantic influences of aspect weaving on the base model. Through comparing the semantic relationship between the base model and its projection, influences of the aspect weaving are detected. The approach can be used in estimating the correctness of aspects.

The future work includes: extend the scope of the research to influences of aspect weaving on certain desired behaviors.

REFERENCES


