B-003

アドバイスの衝突問題を解決する拡張ウィーバの提案

A proposal of extended weaver for resolving aspect's advice conflict

沖津 直樹† 小柳 和子‡
Naoki Okitsu Kazuko Oyanagi

1. はじめに

ソフトウェア構築技法の１つである SOA が最近注目を浴びている。このアプローチを取ればサービスの実装を意識せずシステムをデザインすることができるため、開発が容易になると言われている[1]。 WS-BPEL は Web サービスを統合・連携する為のワークフロー言語であり、「SOA の中核をなす標準技術」[1]であると考えられている。また、商用でもオープンソースでも多くの実装が提供されている。[2]は、WS-BPEL のプロセスファイル中に現れる横断的関心事をモジュールとして扱うための仕組みが無いという問題を指摘している。そして、この問題を解決するためのコンセプトを示し、コンセプトを実装した AO4BPEL[2]というウィーバを提案している。[2]は、WS-BPEL を拡張した初めての論文である。この論文では、アスペクトの織り込み先は WS-BPEL のアクティビティであり、織り込むアスペクトは WS-Security、WS-RM、WS-TX 等で提供されているサービス品質に関わる関数である。 AspectJ[3]では織り込みの際、Java のオブジェクトに焦点を当てている。フィールドのリードやライト、メソッドのコール、コンストラクタコールなどをジョイン・ポイントとし、そこに アスペクトを織り込む。一方、AO4BPEL は織り込みの際 WS-BPEL のアクティビティに焦点を当てており、この点が AspectJ を中心とする多くの既存研究と違う点である。 本稿では、「アドバイスの衝突」という古典的な課題から AO4BPEL を捉え直している。

その結果、AO4BPEL には以下の問題があることが分かった。

問題１	プロトコルが定まっているアドバイスをセットで使用する仕組みが考慮されていない
問題２	アドバイスの実行順番を再定義するための仕組みが考慮されていない

この問題点を解決する為、本稿では順番定義の為の処理フローを提案する。

解決策１	「順番定義の為の処理フロー」

本提案方式は AO4BPEL というウィーバの拡張であり、WS-BPEL のワークフロー設計者がアドバイスの配置をする際の負荷を軽減することを狙いとしている。本提案方式を使うことで WS-BPEL のワークフロー設計者は、 WS-Security や RSA 暗号方式などサービス品質の層に関する知識を持たなくてもアドバイスの衝突問題を回避しつつ、配置作業を楽に実施出来ることが分かった。また、本稿では説明していないが、WS-RM とメッセージの配送保証、WS-TX と更新処理の同期に関する知識が無くても、同様に WS-BPEL の設計者の作業負担が軽減されることが分かった。

本稿は次のように構成されている。２章では AO4BPEL の概要を説明する。ここでは AO4BPEL が扱っている仕様、提案されているアドバイスタイプ、織り込み位置、衝突が起きた際の対策などを紹介する。３章では、AO4BPEL を「アドバイスの衝突」問題から捉え直した際に出てくる問題点を２つ指摘する。４章では、３章で述べた問題を解決するためのコンセプトを示す。５章で関連研究に触れ、６章でまとめる。

2. AO4BPEL の概要

2.1 AO4BPEL で扱っている Web サービス仕様

AO4BPEL は、WS-BPEL、WS-Security、WS-RM、WS-AT、WS-BA、WS-Coordination を研究対象としている。本稿では、WS-BPEL、WS-Security について扱う。WS-BPEL はワークフローベースのオーケストレーション言語である。オーケストレーションとは、Web サービスを連携・統合するものである。この言語を使って Web サービスを調整するための制御ロジックを記述することができる。この制御ロジックをアクティビティという。アクティビティには、receive、invoke、reply などがある。

また、WS-Security は署名や暗号化された要素を SOAP のヘッダ部やボディ部にどのように格納するかというメッセージ構造を定めた仕様である。RSA 暗号方式で旅行代理店（送信側）と A ホテル（受信側）の間をやりとりする場合、SOAP メッセージに対する署名、暗号化、セッション鍵の暗号化の処理の流れは図１の通りになる[4]

†株式会社 NTT データ　NTT DATA Corporation
‡情報セキュリティ大学院大学
Institute of Information Security

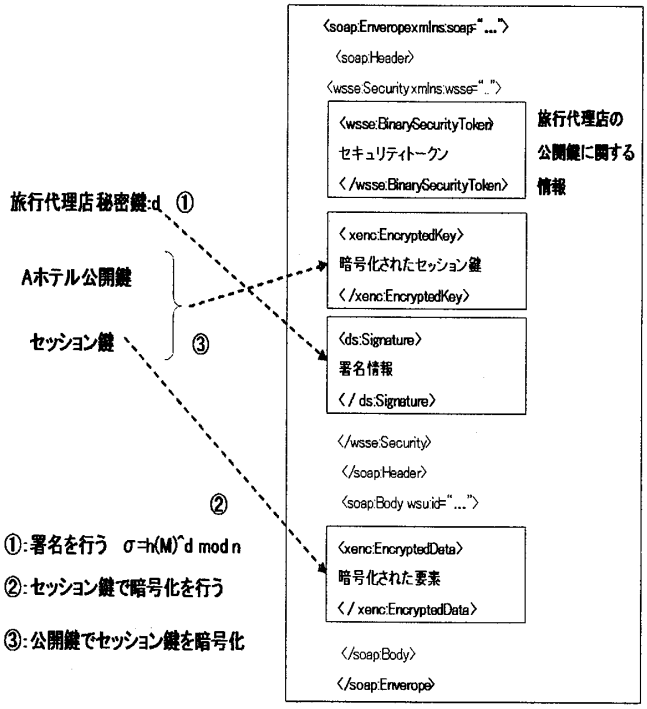

図１　WS-Security の適用

2.2 アスペクトの織り込み方法

AO4BPEL は、WS-BPEL の実装である BPWS4J エンジンを拡張したものである。BPWS4J エンジンでは、アクティビティに関して default、activate、enabled、running、complete の５つのライフサイクルが定められている[5]。アスペクトを織り込む際には、これらのライフサイクルを利用する 。running の段階はアクティビティのタスクが実際に実行されている状態である。complete の段階はアクティビティのタスクが完了した状態である。

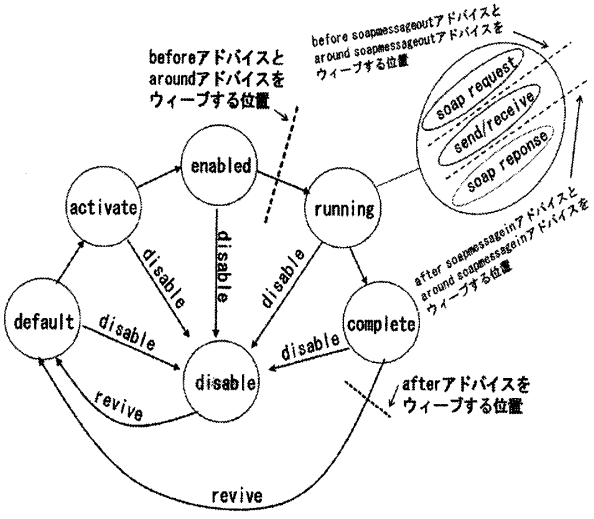

図２　アドバイスの織り込み位置

[2, pp93]には表１に示す７つのアドバイスタイプが提案されている。

アドバイスタイプ	織り込み位置
before	アクティビティが running の段階に入る手前
around	
before soapmessageout	アクティビティは running の段階に入っており、外に向かって送り出す(outgoing の)SOAP メッセージが生成されたタイミング
around soapmessageout	
after soapmessagein	アクティビティは running の段階に入っており、外から入ってきた(incomming の)SOAP メッセージを受信したタイミング
around soamessagein	
after	アクティビティが complete の段階を経た後

表１　アドバイスタイプと織り込み位置

2.3 衝突の定義

図３を見るとジョイン・ポイントとして invoke アクティビティが指定されているのが分かる。invoke や receive、reply は WS-BPEL のアクティビティである。本稿では、「衝突」とは、同一のジョイン・ポイントに対して複数のアドバイスを織り込む状況であると考えて話を進める。アスペクトは、ポイントカット、アドバイスタイプ、アドバイスの３つから構成される。この分類は、[6]の PA(Pointcuts and Advice)に基づくものである。

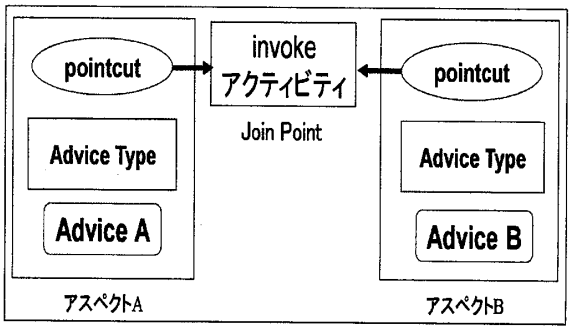

図３　衝突の定義

2.4 衝突発生時の対策

衝突が発生した時の対策として AO4BPEL では表２に示すように２種類の対策が述べられている[2, pp81]。

衝突の種類	対策
アドバイスタイプが同一である場合	Advice A と Advice B のアドバイスタイプが同じであった場合は、order 属性をアスペクト文書内に付ける。
アドバイスタイプが異なる場合	Advice A と Advice B のアドバイスタイプが異なる場合は、アドバイスタイプの優先順位が高いものから先にアドバイスを実行する。 高 ↕ 低 before around before soapmessageout around soapmessageout around soapmessagein after soapmessagein after

表２　衝突発生時の対策

3. 問題提起

3.1 銀行送金シナリオ

次に本稿で使用するシナリオを説明する。本稿では図４に示す銀行送金シナリオを使用する。

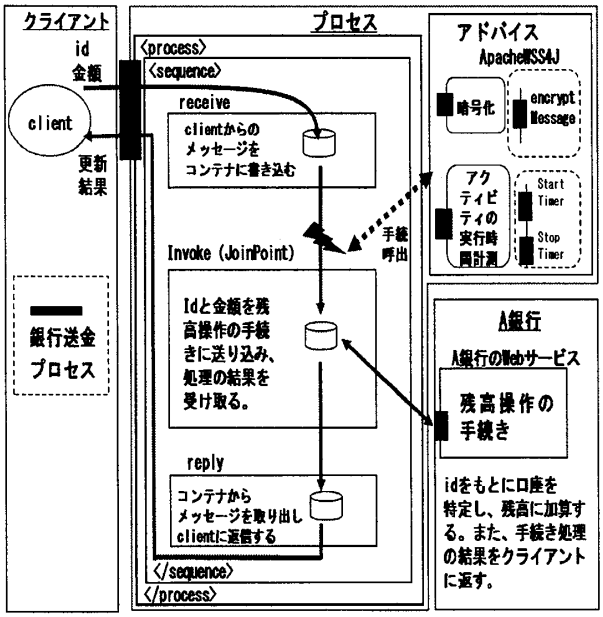

図４　銀行送金シナリオ

図４の銀行送金シナリオの内容は以下の通りである。

- クライアントは、プロセスに id と送金したい金額を送る。
- プロセスにおいて、金額に対する署名を行う。
- プロセスにおいて、セッション鍵で金額を暗号化する。
- プロセスにおいて、セッション鍵を A 銀行の公開鍵で暗号化する。
- プロセスは、署名、暗号化、セッション鍵の暗号化の処理が済んでいる SOAP メッセージを invoke アクティビティで A 銀行に送り込む。
- A 銀行側で、復号を行う。
- A 銀行側で、復号したメッセージから id、金額を取り出して口座の更新処理を行う。
- プロセスは、更新処理結果を reply アクティビティでクライアントに返信する。
- プロセスは、invoke アクティビティの処理時間を計測する。

3.2 問題１　プロトコルが定まっているアドバイスをセットで使用する仕組みが考慮されていない

銀行送金シナリオでは invoke アクティビティに暗号化のアドバイスを織り込もうとしている。RSA 暗号方式でプロセスと A 銀行の間のやり取りを行う場合は、図５に示す通り暗号化のアドバイスの他に署名、セッション鍵の暗号化のアドバイスも配置する必**要がある。** [2] では、アドバイスをセットで配置したり、取り外したりということについての考慮がされていない。

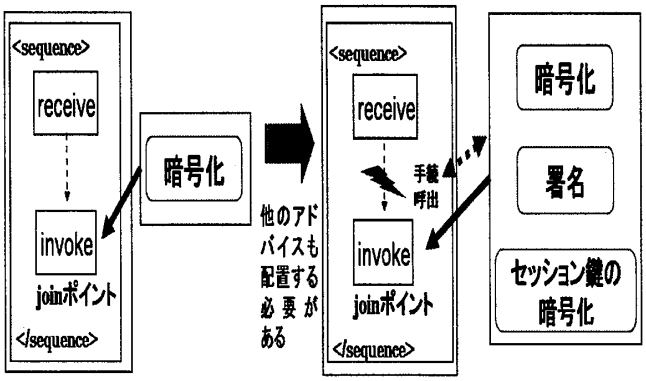

図５　セットで配置が必要なアドバイス

図６の右側では「アドバイスタイプが同一であるアドバイス間の干渉」が発生していることが分かる。この為、order 属性に順番を付ける必要がある。なお、暗号化、署名のアドバイスタイプは [2, pp131, Table8.1] では before soapmessageout となっていたので、その定義にならっている。

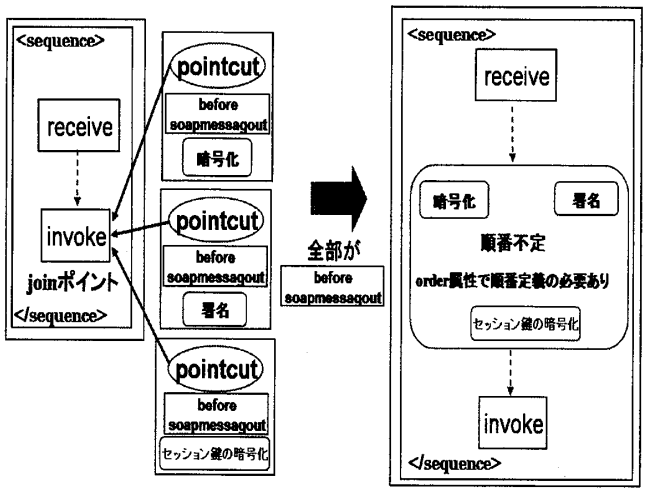

図６「アドバイスタイプが同一であるアドバイス間の干渉」の発生

では、order 属性に正しい順番を定義するにはどうしたら良いであろうか。このためには RSA 暗号方式と WS-Security の仕様についての理解が必要である。しかし、これには専門知識が必要であり、WS-BPEL のワークフロー設計者が知識を持たない場合は正しい順番を定義することは難しい。これらの知識を持たなければ、アドバイスが３つ必要だということさえ分からないであろう。[2] はこの点に関して、サポートの仕組みが考慮されていない。

図１を参考にすると署名⇒暗号化⇒セッション鍵の暗号化の順で、しかも３つをセットにしてアスペクトを配置する必要がある。そうしないと、署名が付いていない、メッセージの暗号化がされていない、セッション鍵が平文のまま送信されてしまう、といういずれかのセキュリティ上の問題が発生してしまう。

3.3 問題２　アドバイスの実行順番を再定義するための仕組みが考慮されていない

「アドバイスタイプが異なるアドバイス間の干渉」が発生した場合、手続きの優先順番については、表２で説明した定義がされている。しかし、この定義に従うと、銀行送金シナリオでは invoke アクティビティの正確な処理時間が計測できないという問題が起こることがあり得る。

図７において、暗号化のアドバイスタイプは before soapmessageout となっており、アクティビティの実行時間計測のアドバイスタイプは around となっている[2, pp86]。[2, pp81]の定義に従うとアクティビティの実行時間計測を先に行い、次に暗号化を行うという順番になる。

しかし、これでは不都合が生じる。なぜなら、暗号化の処理の時間までアクティビティの実行時間計測に含めてしまっているからである。

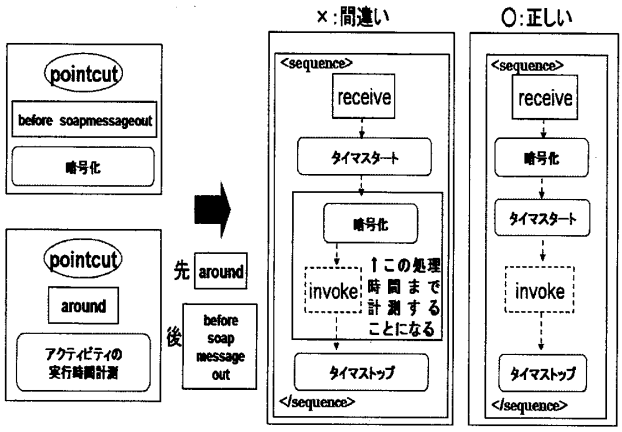

図７　順番を再定義する必要性

[2]における定義では意図した要件が満たせないことが起こり得るため、WS-BPEL のワークフロー設計者が明示的にアドバイスの順番を再定義できる仕組みが必要である。

4. 解決策

銀行送金シナリオを使って問題点を２つ指摘してきた。次にこれらの問題点を解決する為の考え方を提案する。本提案方式では順番定義の為の処理フローを提案する。

4.1　衝突と干渉の関係

提案方式が全ての干渉を網羅していることを示す為、衝突と干渉の関係を図８の通りに整理した。

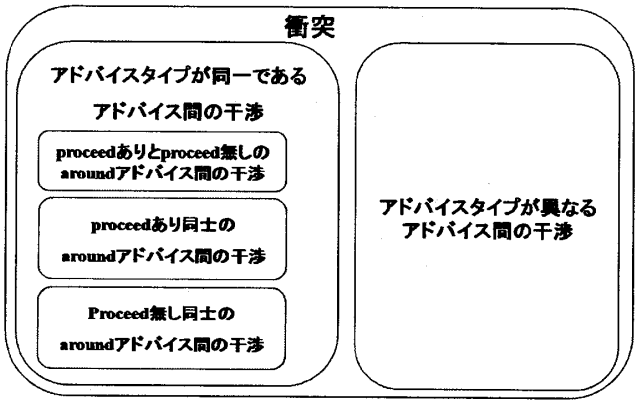

図８　衝突と干渉の関係

4.2　処理フローの中で使用されるDB

処理フローにおいては、以下の３つのDBを用意する。これらのDBは共通モジュール作成グループが用意し、WS-BPELのワークフロー設計者とシステムがこれらのDBを利用する。

DB名	概要
DB_グループ	【カラム】 以下の４つのカラムを保持している。 ①実行順番 ②アドバイス名 ③アドバイスタイプ ④アクティビティ名 【具体例】 表４、表５、表６ これらは[2, pp131]を参照し、筆者らが実行順番を付けた表である。
DB_優先度付け	【カラム】 以下の２つのカラムを保持している。 ①優先度 ②アドバイスタイプ 【具体例】 表７ 表７は表２に示されている考えを参照し、筆者らが優先度のカラムを設け、優先度付けをした表である。
DB_順番定義と検知	【カラム】 以下の６つのカラムを保持している。 ①最終実行順番 ②アドバイス名 ③プロセス名 ④アドバイスタイプ ⑤アクティビティ名 ⑥Proceedの有無 【具体例】 表８ 表８は、図８に整理した全ての干渉を検知する為、筆者らが各カラムを設けたものである。

表３　処理フローの中で利用するDB

実行順番	アドバイス名	アドバイスタイプ	アクティビティ名
1	署名	before soapmessageout	invoke/reply
2	暗号化	before soapmessageout	invoke/reply
3	セッション鍵の暗号化	before soapmessageout	invoke/reply

表４　DB_グループ（セキュリティのアドバイスのグループ化）

実行順番	アドバイス名	アドバイスタイプ	アクティビティ名
1	トランザクション開始	before	sequence/ flow/ scope
2	メッセージングアクティビティ登録	before soapmessageout	invoke/ reply
3	コミット	after	sequence/ flow/ scope

表５　DB_グループ（更新処理同期のアドバイスのグループ化）

実行順番	アドバイス名	アドバイスタイプ	アクティビティ名
1	ExactlyOnceの配送保証でシーケンスを開始	before	sequence
2	シーケンス番号の追加	around soapmessageout	invoke/ reply
3	ExactlyOnceの配送保証でメッセージを送信	around soapmessageout	invoke/ reply

表６　DB_グループ（配送保証のアドバイスのグループ化）

優先度	アドバイスタイプ
1	before
2	around
3	before soapmessageout
4	around soapmessageout
5	around soapmessagein
6	after soapmessagein
7	after

表７　DB_優先度付け（アドバイスタイプの優先度定義）

最終実行順番	アドバイス名	プロセス名	アドバイスタイプ	アクティビティ名	proceedの有無
1	署名	銀行送金プロセス	before soapmessageout	invoke	0
2	暗号化	銀行送金プロセス	before soapmessageout	invoke	0
3	セッション鍵の暗号化	銀行送金プロセス	before soapmessageout	invoke	0
4	アクティビティの実行時間計測	銀行送金プロセス	around	invoke	1

表８　DB_順番定義と検知（銀行送金シナリオ）

4.3　順番定義の為の処理フローの構成

順番定義の為の処理フローは、表９に示すようにⅠからⅣの処理から構成される。

No	処理名	概要
Ⅰ	セットで使用するアスペクトがあるかを調べる処理	【処理内容】 WS-BPELのワークフロー設計者があるアドバイスを「DB_順番定義と検知」に配置したら、システムはセットで使う別のアドバイスをWS-BPELの設計者に通知する。通知を受けたらWS-BPELのワークフロー設計者は、セットで使う別のアドバイスを「DB_順番定義と検知」に追加する。 【処理主体】 ①WS-BPELのワークフロー設計者 ②システム 【処理主体が使用するDB】 ①DB_グループ ②DB_順番定義と検知
Ⅱ-ⅰ	proceedありとproceed無しのaroundアドバイス間の干渉が発生しているかを調べる処理	【処理内容】 予期せぬ処理結果を防ぐため、アドバイスの配置作業を中止させる。 【処理主体】 ①システム 【処理主体が使用するDB】 ①DB_順番定義と検知
Ⅱ-ⅱ	アドバイスタイプが同一であるアドバイス間の干渉が発生しているかを調べる処理	【処理内容】 アドバイスタイプが同一であるアドバイスを並べ替える。 【処理主体】 ①システム 【処理主体が使用するDB】 ①DB_グループ
Ⅱ-ⅲ	proceedあり同士のaroundアドバイス間の干渉が発生しているかを調べる処理	【処理内容】 ジョイン・ポイントのアクティビティの実行を１回のみ行うようフラグを立てる。（AspectJは、この干渉が発生していた場合は自動的に入れ子構造になる。しかし、AO4BPELの場合はAspectJのように入れ子構造にして実行する仕組みについて明示されていない。本稿は、この点を明確にした。） 【処理主体】 ①システム 【処理主体が使用するDB】 ①DB_順番定義と検知
Ⅱ-ⅳ	proceed無し同士のaroundアドバイス間の干渉が発生しているかを調べる処理	【処理内容】 ジョイン・ポイントのアクティビティの実行を全てスキップするようフラグを立てる。 【処理主体】 ①システム 【処理主体が使用するDB】 ①DB_順番定義と検知
Ⅲ	アドバイスタイプが異なるアドバイス間の干渉が発生しているかを調べる処理	【処理内容】 表２[2,pp81]の定義に従ってアドバイスを並べ替える。 【処理主体】 ①システム 【処理主体が使用するDB】 ①DB_優先度付け
Ⅳ	実行順番を手動で調整する処理	【処理内容】 アドバイスの実行順番の最終調整を行う。（表２[2,pp81]の定義にそのまま従ってしまうと、銀行送金シナリオで示したように要件を満たせないことがある。その為、ここで最終調整を行う） 【処理主体】 ①WS-BPELのワークフロー設計者 【処理主体が使用するDB】 ①DB_順番定義と検知

表９　４つの処理で構成される処理フロー

4.4　提案方式の処理順番について

配置作業の効率化の為、Ⅰ、Ⅱ、Ⅲ、Ⅳの順番で処理することとした。この順番で処理することで、WS-BPELのワークフロー設計者が行うアドバイスの配置作業に漏れが無くなり、作業負荷の軽減が図れると期待している。

Ⅰの処理は、アドバイスをセット化したDBを使うことでアドバイスを効率的に全て揃える事を目的としている。その為、最初に行うこととした。

ⅡとⅢは両方とも干渉が発生しているかを調べる為の処理である。Ⅱの処理は、proceedありとproceed無しのaroundアドバイス間の干渉が発生しているかを早い段階で

調べ、発生していた場合は配置作業を中止することを目的としている。その為、Ⅲより先に行うこととした。

Ⅳの処理は最終調整を行うことが目的である。その為、最後に行うこととした。

4.5 提案方式の銀行送金シナリオへの適用

提案方式の表９を図４の銀行送金シナリオに適用した結果を図９に示す。

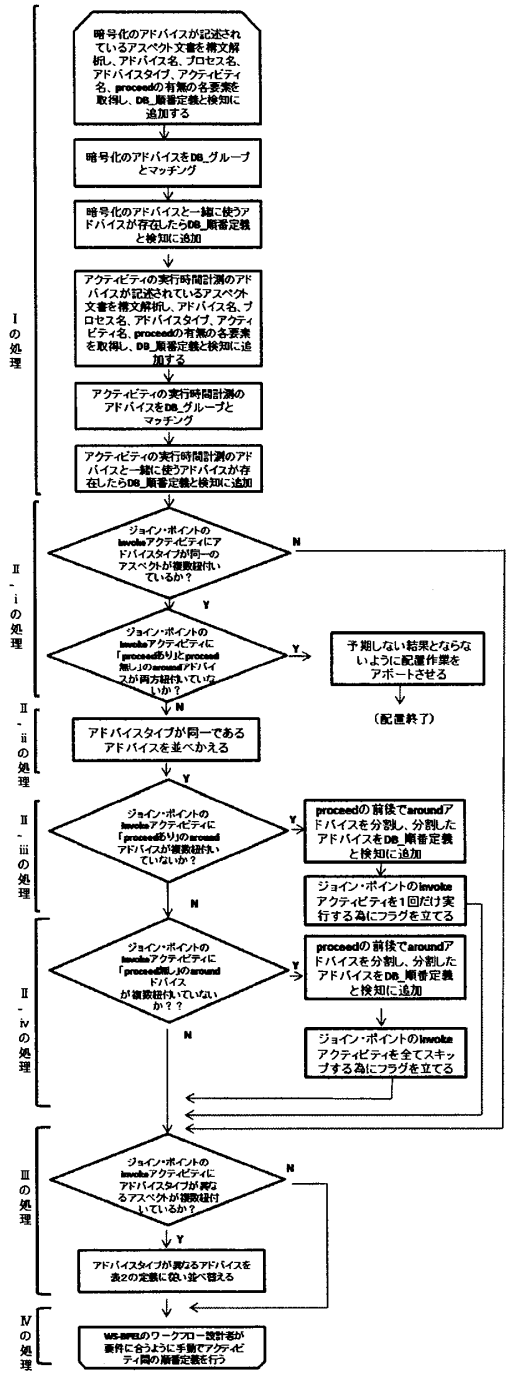

図９　銀行送金シナリオの処理フロー

5. 関連研究

AspectJ[3]では、アスペクトの織り込みを動的に行う研究も多く行われている。この言語ではジョイン・ポイントとして Java のオブジェクトに焦点をあてている。ジョイン・ポイントになるのは、フィールドのリードやライト、メソッドのコールなどである。もし、アドバイスの衝突が起きたら、declare precedence 句を使ってアスペクトの優先度を決めることができる。この宣言は開発者がプログラム中に記述するが、このやり方では開発者が手入力で行うため記述ミスが発生する危険があると思われる。記述ミスが起こっていても、不具合が発生した後でしか検知ができない。不具合が発生した後で、手続きの順番を決める人が再度順番を調べて、注意深く入力し直す必要がある。declare precedence 句を使ってアスペクトの優先度を決めることは、本提案方式の表９「Ⅳ．実行順番を手動で調整する処理」に相当するものであると思われる。しかし、「DB_グループ」に相当する考え方は知っている限りでは無いと思われる。

A04BPEL[2]は、本提案方式の基礎となった研究である。この言語の織り込み方法は動的ウィービングである。ジョイン・ポイントとしては、BPEL のワークフローコードを指定している。例えば、invoke アクティビティの或るオペレーションをジョイン・ポイントとすることができる。アドバイスの衝突が起きたときに備えて、order 属性を使うことやアドバイスタイプ間の優先度が決められている。しかし、衝突を検知する仕組みについての記述が不十分であり、アドバイスタイプ間の優先順も再定義する必要があることが分かった。order 属性を付けることは本提案方式の表９「Ⅳ．実行順番を手動で調整する処理」に相当するものである。しかし、本提案方式のようにセットでアドバイスを配置するという考えや順番を再定義する考えは[2]には示されていない。

AspectM[7]は、ユーザにモデル変換時の誤りを通知することができる仕組みである。この言語では、アスペクトの織り込みは静的に行っている。この言語で焦点をあてているのは Java のオブジェクトでありジョイン・ポイントとしてフィールドアクセスやメソッドのコールを指定できる。また、アドバイスの衝突が発生したときに備え、事前に正しい順番をメタモデルに登録しておくことが考えられている。これは本提案方式の表９「Ⅳ．実行順番を手動で調整する処理」に相当するものである。しかし、「DB_グループ」に相当する考え方は知っている限りでは無いと思われる。

[8]は「アスペクト間の合成方法を明示的に開発者が指示する方法の提案」である。これは、「同一時点で動作するアスペクトが複数あるとき、どのアスペクトがどのアスペクトより先、後という制約を記述していくことで順序を確定する」というものである。この考え方は、本提案方式における表９「Ⅳ．実行順番を手動で調整する処理」に相当するものである。しかし、「DB_グループ」に相当する考え方は示されていない。

6. まとめ

現在の AO4BPEL[2]は、「アドバイスの衝突」問題に関する考慮が十分に行われていない。この点について、以下に示す２つの問題があることを示した。

問題１	プロトコルが定まっているアドバイスをセットで使用する仕組みが考慮されていない
問題２	アドバイスの実行順番を再定義するための仕組みが考慮されていない

そして、これらの問題は①セキュリティ上の問題が発生する可能性がある、②定義されたアドバイスタイプの順番では意図した要件を満たせないことがあり得る、ということに繋がると指摘してきた。次いで、これらの問題に対する解決策を提案した。解決策として順番定義の為の処理フローを示した。

解決策１	「順番定義の為の処理フロー」

また、提案方式を銀行送金シナリオに適用した。適用にあたっては、サービス品質の層のプロトコルとして WS-Security を取り上げた。提案方式に従って「DB_順番定義と検知」、「DB_グループ」、「DB_優先度付け」を使っていくことで WS-BPEL のワークフロー設計者は WS-Security や RSA 暗号方式に関する知識が無くてもセキュリティ上の問題を回避し、意図した要件が満たせるようなアドバイスの順番定義が出来ることを示した。

参考文献

[1] 丸山不二夫, :” 連載 | グリッドと SOA からみる Web サービス標準技術 6　SOA の中核技術としての BPEL 入門(1)　BPEL はどのようにサービスを結合するか？”, Vol. 48 No2 pp191-pp199, IPSJMagazine, Feb. 2007

[2]Anis Charfi. :
“Aspect-Oriented Workflow Management”
VDM Verlag Dr. Muller, April. 2008

[3]Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, William .Griswold
” An Overview of AspectJ.”
In Proc. of the 15th European Conference on Object-Oriented Programming (ECOOP), volume 2072 of LNCS, pp327-pp353. Springer, June 2001.

[4] 佐藤史子, :” 連載 | グリッドと SOA からみる Web サービス標準技術 9　Web サービスセキュリティの最新動向　WS-Security と WS-SX(WS-SecureExchange)関連仕様(1)”, Vol. 48 No6
pp646-pp652, IPSJMagazine, June. 2007

[5] Francisco Curbera, Rania Khalaf, William Nagy, and Sanjiva Weerawarana. “implementing bpel4ws: The architecture of a bpel4ws implementation”. In GGF 10 Workshop on Workflow in Grid Systems, Berlin, Germany, March 2004.

[6]Hidehiko Masuhara and Gregor Kiczales. :
” A Modeling Framework for Aspect-oriented Mechanisms”. In Proc. of the 17th European Conference on Object-Oriented Programming (ECOOP), volume 2734 of LNCS, pp2-pp28. Springer, July 2003.

[7]前野雄作、鵜林尚靖 :” 拡張可能なアスペクト指向モデリングにおける織り合わせの検証”,
情報処理学会研究報告.
ソフトウェア工学研究会報告, Vol. 2007, No. 33(20070322) pp9-pp16

[8]Istvan Nagy, Lodewijk Bergmans, Mehmet Aksit:”Declarative Aspect Composition.”SPLAT Workshop held in conjunction with AOSD2004