アスペクトの優先度を考慮したアプリケーション仕様に基づくアスペクトの織込みに関する研究

M2012MM037 澤田 康太
指導教員：野呂 昌満

1 はじめに

アスペクト指向プログラミングは，横断的関心事をアスペクトとして分離し，モジュール化して記述する手法である．代表的なアスペクト指向言語に AspectJ がある．アスペクトには，特定の処理と，処理が実行される箇所を記述する．明示的にメッセージ送信コードを記述しないことで，疎結合を実現する．

アスペクト指向プログラミングにおいて，アスペクトの動作順序によって実行結果が異なる干渉問題がある．アスペクトの干渉が発生することで，予期しないプログラムの動作を引き起こす可能性がある．干渉問題を解決するために，プログラム依存グラフを用いて干渉を検出する研究では，アスペクトの干渉を検出後，アスペクトに織込み順序を指定することで解決できることを示している [2]．AspectJ ではアスペクトの織込み順序を指定する仕組みが提供されている，システム全体に対してのみアスペクトの優先順位を指定することが可能であるが，サブシステムごとに優先順位を指定する仕組みが提供されていない．我々は，様々なアプリケーションの開発の事例から，システム全体に対してのみアスペクトの優先順位を指定するだけでは不十分であると考えた．アスペクトに織込み順序を指定する枠組みを提供することで，干渉問題を解決できる．

本研究の目的は，アスペクトの優先順位を定める枠組みを提供することで，アスペクトの干渉問題を解決することである．非機能特性は一般的にアスペクトとして抽出できることから，非機能特性に対する要求の優先順位に着目することで，アスペクト間の優先順位も決定できると考えた．しかし，一般に大規模なシステムでは，非機能特性間の優先順位を大域的に定めた場合，サブシステムの非機能特性の優先順位に矛盾が生じる可能性がある．したがって，優先順位を定めるスコープを定義する必要がある．

本研究では，非機能特性間に優先順位を指定することで，アスペクトの織込み順序を決定する枠組みを示す．非機能特性間の優先順位を決定することで，系統的にアスペクトの織込み順序を決定できるという仮説を立てる．システム全体対して非機能特性間の優先順位を定めることで，サブシステムにおいて非機能特性間の優先順位に矛盾が生じる可能性があると仮説を立てる．非機能特性間の優先順位を矛盾無く定めるために，スコープの定義を行なう．提案した枠組みが適用可能か検証するために，飛行船制御システムを用いて事例検証を行なう．

本研究の成果として，非機能特性間の優先順位に基づいて，アスペクトの織込み順序を決定することで，アスペクト干渉問題を解決できた．

2 背景技術

本章では，横断的関心事をアスペクトとして分離して実現するアスペクト指向プログラミングと，アスペクトの干渉を検出する方法を示しているプログラム依存グラフを用いたアスペクト干渉の検出について挙げる．

2.1 アスペクト指向プログラミング

アスペクト指向プログラミングは，横断的関心事をアスペクトとして分離し，モジュール化するプログラミングである．アスペクト指向プログラミングに用いるアスペクト指向言語には，“ApectsC++”，“AspectJ”，“AspectR”，“Hyper/J” などがある．本研究では，一般に記述形式が受け入れられており，幅広く利用されている AspectJ を対象とする．AspectJ では，以下の三つの構文要素を用いて実現する．

- ジョインポイント (join point)
- ポイントカット (point cut)
- アドバイス (advice)

プログラム実行における特定の時点をジョインポイント，複数のジョインポイントをひとまとめにしたものをポイントカット，ポイントカットに関連付ける処理のことをアドバイスと呼ぶ．ジョインポイントの例には，メッセージ送信箇所やフィールドへの代入箇所，参照箇所がある．アドバイスはアスペクトの織込む順序をジョインポイントの実行の直前 (before), 直後 (after), 置き換え (around) を用いて定める．AspectJ では，アスペクトの織込み順序を定めることができる “declare precedence” 記述が用意されている．アスペクトの織込み順序を定める記述を用いることで，予期しないアスペクトの織込み順序を解決することが可能である．図 1 は，アスペクトの織込み順序を定める記述を用いたアスペクトコードの例である．“declare precedence” に対して左に記述されているものほど優先順位が高く，“AspectA” が織込まれた後に “AspectB” が織込まれることを示している．AspectJ で提供されている

```
public aspect Ordering{
declare precedence : AspectA , AspectB;
}
```

図 1 declare precedence の記述例

アスペクトの織込み順序を定める記述は，システム全体に対してアスペクトの織込み順序を定めることが可能である．一方，サブシステムに対して個々にアスペクト織込み順序を定義することができない．図 2 は，アスペク

トの織込み順序を定める記述の仕様を表したものである．我々は，様々なアプリケーションの開発の事例から，シ

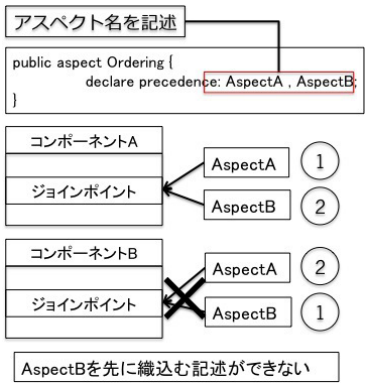

図 2　アスペクトの織込み順序を定める記述の仕様

ステム全体でアスペクトの優先順位を指定するだけでは不十分であると考えた．本研究では，サブシステムごとに対するアスペクトの優先順位を考慮して，織込み順序を定める枠組みを提供する．本研究におけるシステムの定義は，“一つの目標を達成するためのコンポーネントの集合で構成されたソフトウェア全体” とし，サブシステムの定義は，“システムを構築しているコンポーネント” とする．

2.2　アスペクト干渉の検出

平井ら [2] は，アスペクト指向プログラミングにおいて，アスペクトの衝突と干渉問題の検出方法を提案した．アスペクトの衝突とは，“同一のジョインポイントに複数のアドバイスが存在すること” と定義している．アスペクト干渉とは，“衝突したアドバイスの実行順序の違いによって，全体としての実行結果に違いが生じること” と定義している．図 3 は，アスペクト干渉の例である．x = 1

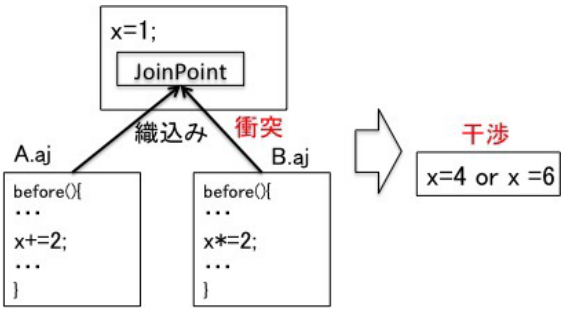

図 3　アスペクト干渉の例

という値を持つオブジェクトに対して x + = 2 を行なう “AspectA” と，x * = 2 を行なう “AspectB” を織込む．“AspectA” が先に織込まれる場合，実行結果は x = 6 となる．“AspectB” が先に織込まれる場合，実行結果は x = 4 となり，織込み順序によって実行結果が異なることが確認できる．平井らは干渉問題を解決するために，プログラム依存グラフを用いて干渉を検出する指針を示した．検出することにより，衝突しない記述に変更するか，アスペクトに織込み順序を指定することで干渉を解決できる．我々は，アスペクトに織込み順序を指定することで干渉を解決できる点に着目し，アスペクトに織込み順序を指定する枠組みを提供することで，干渉問題を解決できると考えた．

3　アスペクトの織込み順序を定める枠組みの提案

本章では，非機能特性間の優先順位に着目することで，アスペクトの優先順位を定める枠組みを示す．

3.1　アスペクトの優先順位を決定する指針

非機能特性は一般的にアスペクトとして分離できる点に着目する．図 4 はアスペクト干渉が発生する例である．ジョインポイントを “TestMethod” のメッセージ送信とし，アスペクトは引数をチェックする．図 4 において，引数 name が NULL だった際に，“ExceptionHandling アスペクト” が “Logging アスペクト” より先に織込まれた場合，出力は “name = Exception” となる．一方，“Logging アスペクト” が “ExceptionHandling アスペクト” より先に織込まれた場合，出力は “name = NULL” となる．“ExceptionHandling アスペクト” は引数の値をチェックし，異常だった際に引数に代替の値を代入することから，例外処理を実現する．例外処理を実現することにより，成熟性を満たしている．“Logging アスペクト” は引数の値をログ出力することから，解析性を満たしている．ユーザが解析性より成熟性の優先順位を高く望まなければ，例外処理された実行結果を得ることができない．非機能特

```
public class Test {
    public void TestMethod(String name){
            ・・・
    }
}
```

```
public aspect ExceptionHandling {
    if( name == NULL ) {
        name = Exception ;
    }
}
```

```
public aspect Logging {
    System.out.println("name = " + name );
}
```

図 4　アスペクト干渉の発生するコード例

性に関するアスペクトの織込み順序によってアスペクト干渉が発生していることに着目し，“非機能特性に優先順位を定めることで，アスペクトの織込み順序を定めることができる” という仮説を立てる．本研究では，ISO9126 で定義された非機能特性を対象とする．着目する非機能特性は組込みシステムに必要だと考えられている以下の 4 つとする．

- 障害許容性 (fault tolerance)
- 時間効率性 (time behaviour)
- 解析性 (analyzability)
- 成熟性 (maturity)

本研究では，非機能特性とアスペクトは一対一で対応するものとする．

3.2 非機能特性間の優先順位を定めるスコープの定義

一般に大規模なシステムでは，非機能特性間の優先順位を大域的に定めた場合，サブシステムに対する非機能特性の優先順位に矛盾が生じる可能性がある．飛行船制御システムにおいて，飛行船制御システムに対する非機能特性の優先順位は，成熟性より時間効率性の方が高いとする．この時，モーター部分に対する時間効率性と成熟性は時間効率性の方が優先順位が高く，センサー部分では成熟性の方が優先度が高い場合，モーターとセンサーにおける非機能特性間の優先順位が異なる．サブシステムに対する非機能特性間の優先順位の矛盾を解決するために，優先順位を定めるスコープを定義する必要がある．本研究では，ユースケースをスコープの単位とする．ユースケースとは，利用者の視点からシステムをどう扱うかを示したものである [1]．利用者の要求する機能特性やシナリオなどにより，選ばれるユースケースは異なる．我々は，非機能特性間の優先順位もユースケースに関連する要素になると考えた．図 5 は，ユースケースに対して定められた非機能特性間の優先順位と，アーキテクチャにおける非機能特性間の優先順位の対応関係である．図 5 が示している表は，ユースケースに対して定められた非機能特性の優先順位を示している．ユースケースに対して定められた非機能特性間の優先順位は，ユースケースに基づいて構築されたアーキテクチャ上のコンポーネントに対する非機能特性間の優先順位と対応する．

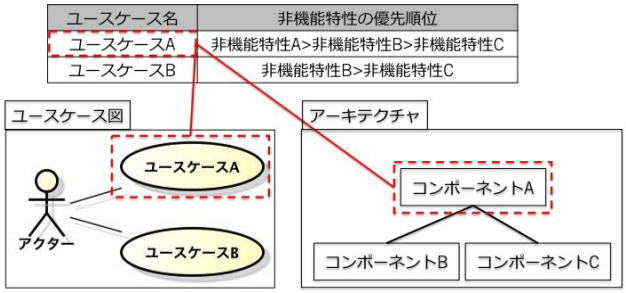

図 5 ユースケース図とアーキテクチャの対応関係

3.3 アスペクトの織込み順序の決定による干渉問題の解決

本研究では，アスペクトの織込み順序を定めることでアスペクトの干渉問題を解決する．ユースケースをスコープの単位とすることで，ユースケースに関連するコンポーネントを捉えることができる．ユースケースに対して定められた非機能特性間の優先順位に基づいて，アスペクトの織込み順序を定める．図 6 は非機能特性間の優先順位に基づいてアスペクトの織込み順序の決定する例である．“アスペクト A” は “非機能特性 A” を満たし，“アスペクト B”，“アスペクト C” も同様に “非機能特性 B”，“非機能特性 C” を満たす．優先順位表において “非機能特性 A” が最も優先順位が高いことから，“コンポーネント A” に対して織込まれるアスペクトは “非機能特性 A” を満たす “アスペクト A” が一番目に織込まれる．“非機能特性 B” は “非機能特性 A” より優先順位が低いことから，“アスペクト A” の次に “アスペクト B” が織込まれる．アスペクトに対して優先順位を定めて織込むことで，予期しない織込み順序を防ぎ，アスペクトの干渉の発生を防ぐことができる．

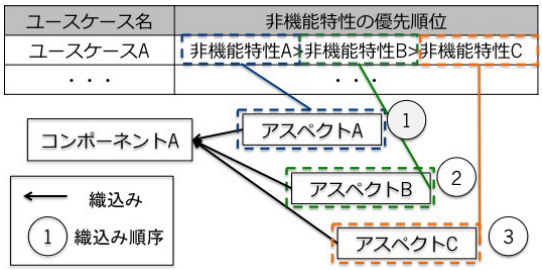

図 6 非機能特性間の優先順位に基づいたアスペクトの織込み順序の決定

3.4 複数のユースケースに関連するコンポーネントに対するアスペクトの織込み

ユースケースに基づいて構築されるコンポーネントは常に一対一とは限らず，一つのコンポーネントは複数のユースケースと関連する場合がある．この場合，本研究の枠組みを適応する際に，アスペクトの織込み順序に競合が発生する．図 7 は，アスペクトの織込み順序の競合と，競合を解決する方法を示したものである．ユースケース毎に定められた非機能特性間の優先順位に従って，アスペクトの織込みのポリシーを変更する．

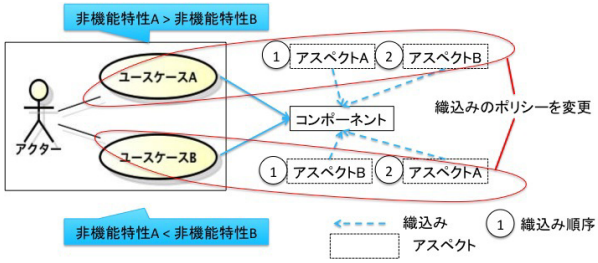

図 7 アスペクトの織込みポリシーの変更の例

4 事例検証

本章では，飛行船制御ソフトウェアを対象に事例検証を行なう．飛行船制御ソフトウェアで必要と考えられる非機能特性を以下の三つとした．

- 時間効率性
- 成熟性
- 解析性

図 8 は，事例に用いる飛行船制御ソフトウェアのユースケース図である．アクターが起動することで，飛行船はデータ取得を行ない，演算結果に基づき出力を行なう．図 9 は，各ユースケースに非機能特性間の優先順位を定めた優

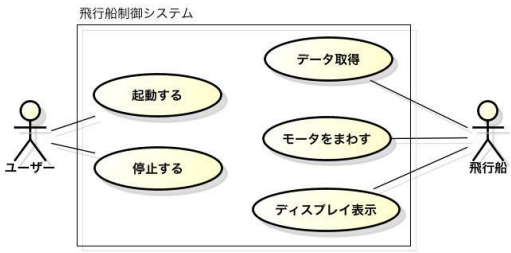

図 8 飛行船制御ソフトウェアのユースケース図

先順位表と，織込まれるアスペクトの関係を示した一部分である．“データ取得” は，指定された時間通りに機能する

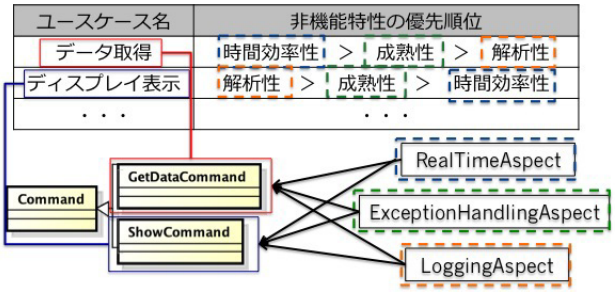

図 9 非機能特性間の優先順位表に基づくアスペクトの織込み順序

ことが一番に求められ，動作し続ける必要があることから，時間効率性，成熟性，解析性の順番に定めた．“ディスプレイ表示” は，データを視覚的に表示する機能なので，データ内容が確実に取得できるべきであり，処理時間は考慮しないことから，解析性，成熟性，時間効率性の順番に定めた．時間効率性，成熟性，解析性を満たすアスペクトはそれぞれ RealTimeAspect，ExceptionHandlingAspect，LoggingAspect とする．ユースケースに基づいて，コマンドパターンを適用してアーキテクチャを構築した．“データ取得” を実現する “GetDataCommand” に対して，非機能特性間の優先順位から，RealTimeAspect，ExceptionHandlingAspect，LoggingAspect の順番で織込む．同様に，“ディスプレイ表示” を実現する “ShowCommand” に対して，LoggingAspect，ExceptionHandlingAspect，RealTimeAspect の順番で織込む．ユースケースに対して非機能特性間の優先順位を定めることで，アスペクトの優先順位を定めることができた．

5 考察

本章では，アスペクトの優先順位を決定する指針の妥当性と，スコープの定義の妥当性について考察する．

5.1 アスペクトの優先順位を決定する指針の妥当性

本研究では，非機能特性間の優先順位に着目することで，アスペクトの優先順位を定めた枠組みを示した．アスペクトの干渉問題は，衝突をしない記述に変更するか，アスペクト間に優先順位を指定することによって解決できる．アスペクト間の優先順位を定める枠組みを示すことで，干渉問題の解決を目指した．アスペクト間の優先順位を定めるために．非機能特性は一般的にアスペクトとして抽出できる点に着目した．事例検証から，非機能特性に優先順位を定めることで，非機能特性を満たすアスペクト間の優先順位が系統的に決定できるという仮説が検証できた．アスペクトの織込み順序を決定することにより，干渉は発生しなくなる．非機能特性間の優先順位に基づいてアスペクトの織込み順序を定めることで，アスペクトの干渉問題を解決できた．今後，他の非機能特性についても同様に枠組みを適応できるか調査する必要がある．本研究の枠組みを適応するために，ジョインポイント単位でアスペクトの織込み順序を定めることができる仕組みを提供する必要がある．

5.2 スコープの定義の妥当性

非機能特性間の優先順位を定める範囲を定義するために，ユースケースをスコープの単位とした．大規模なシステムに対して大域的に非機能特性間の優先順位を定めると，サブシステムにおいて非機能特性間の優先順位に矛盾が発生する可能性がある．非機能特性間の優先順位の矛盾を回避するために，ユースケースをスコープの単位とした．利用者の要求する機能特性やシナリオなどにより，選ばれるユースケースは異なることから，非機能特性間の優先順位もユースケースに関連する要素として考えることができた．ユースケースに対して非機能特性間の優先順位を定めることによって，事例ではコマンドパターンに基づくコマンドに対するアスペクトの織込み順序を定めることができた．独立にアスペクトの織込み順序を定めることができるので，アスペクト間の優先順位に矛盾が発生することを防ぐことができた．今後，ユースケース以外についても同様にスコープ定義が可能か検証する必要がある．

6 おわりに

本研究では，アスペクトの織込み順序を定める枠組みを示すことでアスペクトの干渉問題の解決した．非機能特性間の優先順位に着目することで，系統的にアスペクトの織込み順序を決定する指針を示した．非機能特性間の優先順位を定める範囲を定義するために，ユースケースをスコープの単位とした．提案した枠組みが適用可能か検証するために，飛行船制御システムを用いて事例検証を行なった．今後の課題に，今回対象とした非機能特性以外について考慮すること，ジョインポイントごとにアスペクトの織込み順序を指定できる仕組みを提供すること，他の事例について同様に枠組みを適応できるか考察する必要がある．

参考文献

[1] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992.

[2] 平井考，丸山勝久，“プログラム依存グラフを用いたアスペクト干渉検出，” 情報処理学会第 153 回ソフトウェア工学研究会，vol.153，pp7-14，2006.