Vol.2018-SE-199 No.17
2018/7/19

アスペクト指向モデルにおける アスペクト間の関係のプロパティについて

野田夏子[†1]　岸知二[†2]

概要：我々が提案したアスペクト指向モデリングメカニズムでは，関心事に対応するアスペクトとアスペクト間の関係を分離することにより，アスペクトの独立性を高め，柔軟なソフトウェア構造を構成することができる．しかし一方で，アスペクト間の関係を定義するルールの設計を間違うと，モデル全体の振る舞いが意図しないものになる可能性がある．

本稿では，意図しない振る舞いを生じうる関係のプロパティについて分析し，こうしたプロパティの検証の可能性について考察する．

Properties of Aspect Relationships in Aspect-Oriented Models

NATSUKO NODA[†1]　TOMOJI KISHI[†2]

1. はじめに

ビジネス環境や技術環境の変化が一層激しくなる中，ソフトウェアの要求の変更がますます増え，不確実性も増している．そうした中，ソフトウェアをそうした変化や不確実性に耐えられる柔軟な構造にすることが重要となっている．そうした背景を踏まえ，我々は独立性の高いコンポーネントを柔軟に構成することを目指した，アスペクト指向モデリングメカニズムの提案を行ってきた[1][2]. さらに現在，そのモデリングメカニズムを踏まえたモデル駆動環境の開発を進めている[4]. 本アスペクト指向モデリングメカニズムでは，関心事に応じて定義されるアスペクトを，ルールによって関連付けて組み合わせる方法をとっており，アスペクトを独立性高く定義できるとともに，ルールの変更により柔軟な組み合わせができる[3].

一方，アスペクト内に定義されている振る舞いが，ルールによって他のアスペクトと関連づけられることによって意図しない振舞いになるなどの状況を引き起こすこともある．すなわち，アスペクトとルールの分離することで，全体の振る舞いの直感的な理解が妨げられることがありうる.

本稿では，本アスペクト指向モデリングメカニズムにおけるそうした全体の振る舞いに関わる性質を概観するとともに,望ましくない状況の可能性を検出する方法について，初期の検討を行う．以下，2 章で本アスペクト指向モデリングメカニズムの概要を説明し，3 章で本メカニズムに基づくモデルにおいてアスペクト間の関係により引き起こされる得る問題を概観する．4 章ではそうした問題を検出する方法についての初期の考察を行い，将来的な可能性についても議論する．5 章で本稿を締め括る.

2. アスペクト指向モデリングメカニズム

本アスペクト指向モデリングメカニズムの概要を以下に示す.

2.1 モデル要素の意味

以下に主要なモデル要素を示す.

- アスペクト：特定の関心事に関わるモデル要素をモジュール化する単位．名前空間の一種であり，並行動作の単位ではない.
- クラス：アスペクト中に定義されるモデル要素で，その関心事に関わる概念などを表す. クラスの振る舞いはステートマシン図で状態遷移として定義される.
- オブジェクト：クラスのインスタンス．本稿では 1 クラス 1 オブジェクトの状況に限定する.
- 状態遷移：オブジェクトの振る舞いは，対応するクラスのステートマシン図で状態遷移として定義される. なお各遷移には識別のために名前がつけられるものとする.

2.2 アスペクト関連ルール

アスペクトを関連付けるために，以下のルールがある.

- イベント導入ルール：あるアスペクト中で遷移が発火した際に，他のアスペクトにイベントを導入するルール．例えば “Aspect1.Class1:trans1 -> event^Aspect2.Class2” というルールは，アスペクト Aspect1 中のクラス Class1 に定義された状態遷移の遷移 trans1 が発火すると，アスペクト Aspect2 中のクラス Class2 にイベント event が導入されることを意味する.
- 条件参照ルール：あるアスペクトの状態遷移が発火するときに，他のアスペクトのクラスの状態をガード

†1 芝浦工業大学
Shibaura Institute of Technology
†2 早稲田大学
Waseda University

条件として参照するルール．例えば“Aspect1.Class1:trans1[Aspect2.Class2@State1]”というルールは，Aspect1 中のクラス Class1 に定義された状態遷移の遷移 trans1 は，アスペクト 2 中のクラス Class2 が状態 State1 にあるときに限って発火することを意味する．

2.3 動的意味

各アスペクト中のオブジェクトの動作意味は，通常のオブジェクト指向の動作意味に準じており，各オブジェクトはイベントキューを持ち，発生したイベントはそこに蓄えられ，run-to-completion に従って動作する．

アスペクトが上述したルールによって関連付けられた際の動作意味は以下のようになる．

- イベント導入ルールが与えられたときは，イベントに対応するアクションを付け加える．上記のイベント導入ルールの記述でいえば，アスペクト Aspect1 中のクラス Class1 の遷移 trans1 にイベント event に対応するアクションを追加する．実行時に trans1 が発火すると，Class2 に対応するイベントキューに event が追加される．
- 条件参照ルールが与えられたときは，記載された条件を遷移のガードに加える．上記の条件参照ルールの記述でいえば，アスペクト Aspect1 中のクラス Class1 の遷移 trans1 のガード条件に，アスペクト Aspect2 中のクラス Class2 の状態が State1 である，という条件をガードとしてつける．他にガード条件がある場合にはアンドをとる．

このように，いったんルールが与えられるとイベントがどのアスペクト中で発生したのか，参照するガード条件の状態がどのアスペクト中のものであるのか，ということとは関係なく，アスペクトが関連付けられることになる．

3. アスペクト間の関係により生じる問題

3.1 アスペクトの組み合わせで発生し得る問題

アスペクトは，前章で述べたように，関心事をモジュール化する単位であり，対応する関心事の視点から見たソフトウェア全体の射影である．他のアスペクトには依存せず，その内部の定義は自己完結している．したがって，注目している関心事の観点からは，単体でも動作可能なものである．

しかし，各コンポーネントが正常に動作したからといってソフトウェア全体の動作が保証できないのと同様に，アスペクト単体では正常に動作していても，アスペクト間の関係を定義し複数のアスペクトから構成したものに関しては，単体では起こらなかった振る舞いが生じる場合がある．

これらの振る舞いには，以下のようなものがある．

(1) アスペクト単体では動作していた動作が停まる

アスペクト間が関係を持つと，あるイベントが起こった場合に遷移が発火するかどうかが他のアスペクトのオブジェクトの状態によって変化することがあり得る．

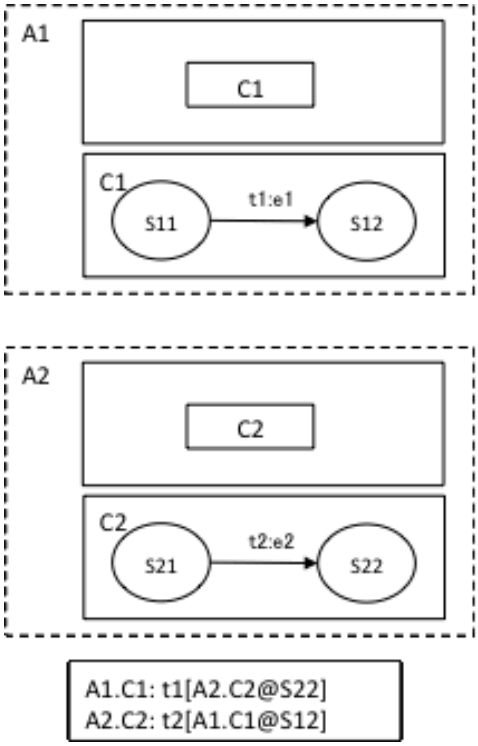

図 1 動作が停まる例

図 1 に示すモデルの場合，アスペクト A1，A2 のそれぞれは単独では対応するイベントの発生により状態が変わるが，図に示された 2 つのルールで A1, A2 を関連付けると，A1 中の C1，A2 中の C2 がそれぞれ S11，S21 にある場合，それぞれ S12，S22 に到達できず，動作が止まってしまう．つまり，C1 と C2 がデッドロックの状態に陥ってしまう．

(2) 単独では想定していなかった動作パスができる

アスペクト間が関係を持つと，他のアスペクトにおける状態の変化をきっかけとして遷移が起こり得る．したがって，単独で動作する場合とは異なるタイミングでイベントが発生し，そのことによりアスペクト単独での動作では想定していなかった動作パスができる場合がある．

図 2 に示すモデルの場合，アスペクト A1 単独の場合，C11 の正常な状態の変化の系列は，S111→S112→S113 である．しかし，アスペクト A1 と A2 を図に示すルールで関連付けると，そのルールによりイベントが導入されるタイミングによっては，S111→S113 という動作パスになってしまう．つまり，もし C11 の状態が S111 の時に，アスペクト A2 のクラス C2 において遷移 t2 が起こると，ルールによりイベント e2 がアスペクト A1 に導入され，クラス C11 の状態は S111 から S113 に変化してしまう．

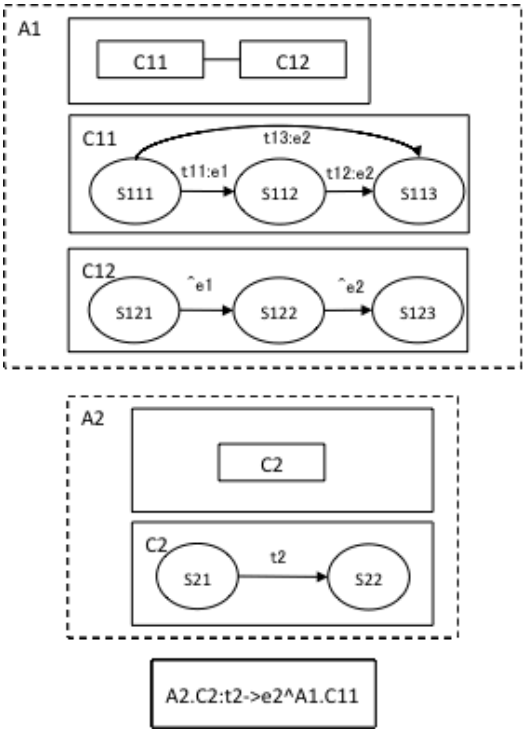

図 2 想定しない動作パスができる例

(3) **単独では達していた状態に達しない**

アスペクト間が関係を持つと，単独での動作の場合とは異なるタイミングでイベントが発生することになり，そのことにより単独の動作では達していた状態に達しないことが起こり得る．

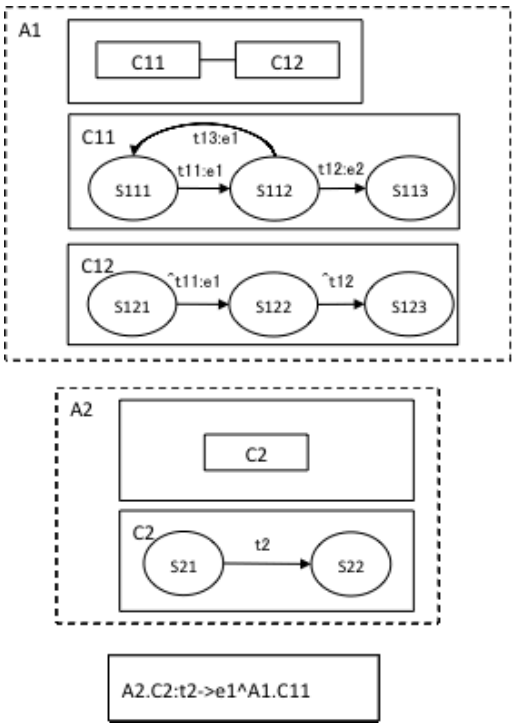

図 3 達していた状態に達しない例

図 3 に示すモデルの場合，アスペクト A1 単独では，イベントは t11→t12 という順序でしか発生しない．したがって，クラス C11 は状態 S113 に到達する．しかし，図に示すルールでアスペクト A1 と A2 を関連づけると，このルールによりアスペクト A1 のクラス C11 が状態 S112 の時にイベント e1 が導入される可能性があり，C11 の状態は S111 に戻ってしまって，S113 には到達できなくなる．

もちろん，これらが起こることが全て本当にソフトウェア上の問題になるというわけではない．例えば，ある状態に達しないことが設計上の意図であれば，状態に達しないこと自身は問題ではない．典型的には問題と判断され得る状況であるということを示している．

3.2 ルールの組み合わせで発生し得る問題

アスペクト間関係ルールは，各アスペクト内のクラスの状態や遷移，遷移を引き起こすイベントを構成要素として定義される．したがって，各クラスのステートマシン図のどれにも登場しない状態，遷移，イベントを指定したルールを書いても，当然正しく機能しない．

しかし，個々のルールは正しく機能していても，複数のルールを組み合わせた場合に競合を起こし得る場合がある．これらの競合として，以下のようなものがある．

(1) **条件参照ルールの競合**

複数の条件参照ルールが，同じクラス(同じアスペクト内の同じクラス)の同じ遷移の発火に関して，それとは異なる同一のクラスの異なる状態を参照する場合，それらのルールは競合する．

例えば，前章の図 1 のアスペクト A1，A2 について，(図 1 とは別の)以下の 2 つのルールがあったとする．

ルール 1　　A1.C1:t1[A2.C2@S21]
ルール 2　　A1.C1:t1[A2.C2@S22]

これらのルールは，それぞれ単独では正しく機能するルールである．しかし，アスペクト A2 中のクラス C2 について，当然それらが同時に S21 であり S22 であることはできないため，同時に用いられた場合には競合する．

(2) **イベント導入ルールの競合**

複数のイベント導入ルールが，同じクラスの同じ遷移の発火に対して，それとは異なる同一のクラスに対して異なるイベントを導入する場合，それらのルールは競合する．

例えば，前章の図 2 のアスペクト A1, A2 について，(図 1 とは別の)以下の 2 つのルールがあったとする．

ルール 1　　A2.C2:t2->e1^A1.C11
ルール 2　　A2.C2:t2->e2^A1.C11

これらのルールは，同じ振る舞いが引き起こす結果が異なるものであり，同時に用いられると競合する．

上記の競合も，前節で述べたアスペクトの組み合わせで

発生し得る問題と同様に，必ず実際の問題になるということではない．しかし，動作が止まったり，振る舞いが不定になったりする可能性が高く，典型的には問題となり得るものである．

4. 検証の可能性

本アスペクト指向モデリングにより，より柔軟な構造の実現ができる一方，前章で述べたようにアスペクト間の関係により予期しない振る舞い等の問題が生じ得る．したがって，こうした問題の可能性を検証できることが重要になる．本章では，検証の可能性について考察する．

4.1 ルールの静的なチェック

全体を構成するアスペクトとそれらを関連付けるアスペクト関連ルールの集合が与えられた時，それらのルール群中に 3.2 で述べたような競合を起こすルールの組があるかどうかは，ルールの静的なチェックにより発見できる．ルールは，状態やイベントを一意に特定できる名前を用いて記述しているため，ルールを構文に従って解析することで比較的簡単にこうしたチェックはできると考えられる．アスペクト指向モデル全体の検証の前に，まずはルール群について互いに競合しているルールがないかを検証することは重用であると考える．

4.2 モデルの振る舞いの網羅的な検査

ルールの競合は，ルールの静的なチェックで発見可能であるが，3.1 で述べたようなアスペクトの組み合わせで起こる問題はルールだけを見ていてもわからず，ルールにより変化する実際の振る舞いを検証しなければ発見できない．

3.1 で述べた振る舞いは，その多くがオブジェクトの並行動作により生じ得る，デッドロックやライブロック等の問題に帰着できると考えられる．このような問題は，例えばモデル検査を用いて，状態空間を網羅的に探索することによって検証することができる．

アスペクト関連ルールで表現されたアスペクト間の関連は，意味的には元々のオブジェクトの状態遷移に状態遷移の条件や状態遷移を引き起こすイベントを追加することに相当し，新たなオブジェクトが加えられるわけではない．したがって，ルールで表現された内容を，各オブジェクトの振る舞いに展開可能である．ルールを展開してしまえば，それらのオブジェクト群の振る舞いを，モデル検査で検査することが可能になると考えられる．

4.3 議論

前節で述べたような，ルールの内容を展開した後に全体の振る舞いを検査することは，検査の網羅性は高くなると考えられるが，一方で全体は大きくなり振る舞いの理解が困難になることが懸念される．

このような問題を回避するためには，例えばルール間の関連をチェックすることにより，互いに影響を受けるアスペクト群に分割するといったことが可能かもしれない．例えば，アスペクト A，B，C からなる全体を，ルールを分析し，その結果に従って例えばアスペクト A と B の統合，アスペクト B と C の統合のように分割して検査するのである．また，このような分割が可能になるようなモデリングの方法や，ルール自体のモジュール化といったことも検討する必要があると考える．

5. おわりに

本稿では，アスペクト指向モデリングメカニズムにおけるアスペクト間の関係に関わる振舞いに関わる性質と，問題となる状況を確認する手法についての考察を行った．これらの性質の厳密な定義や検証手法の検討は今後の課題である．

参考文献

[1] Noda, N. and Kishi, T.. Aspect-Oriented Modeling for Embedded Software Design. Proc. 14th Asia-Pacific Software Engineering Conference (APSEC 2007), Dec. 2007.
[2] Noda, N. and Kishi, T.. Aspect-oriented Modeling for Variability Management. Proc. 12th International Software Product Line Conference (SPLC 2008), Sep. 2008.
[3] 野田夏子，岸知二．柔軟なプロダクトラインアーキテクチャ設計に関する一考察．情報処理学会研究報告，vol. 2016, no. SE-191, 2016.
[4] 杉田郁人，野田夏子．アスペクト指向モデリング手法のためのモデル駆動開発環境の提案．情報処理学会研究報告，vol. 2017, no. SE-195, 2017.