アスペクト指向ソフトウェアアーキテクチャの文書検査に関する研究 ー動的振る舞い図の検査ツールの設計と実現ー

2004MT078 長 大介，2004MT124 山本 明利
指導教員　沢田 篤史

1　はじめに

我々の研究室では，組込みソフトウェアのためのアスペクト指向ソフトウェアアーキテクチャスタイル (E-AoSAS++) を提案し，その記法を，UML の記法を基に提案している．UML は広く利用されており，開発者の間でアーキテクチャ記述の理解が容易だと考えており，UML を記述するツールが豊富に存在し，実用性が高いと考える．

問題点として，UML の記法と E-AoSAS++ のアーキテクチャ記法との間には乖離があるので，UML の記法で記述したアーキテクチャが E-AoSAS++ の記法を満たさない場合がある．また，アーキテクチャ記述が E-AoSAS++ の記法に従っているか否かを検査する手段が確立していない．

この問題を解決するために，我々の研究室では，UML の記法で記述されたアーキテクチャが，E-AoSAS++ の記法に従っているかを検査をするための，一連のツールを開発している．

本研究では，これらの一連のツールのうち，UML の記法で記述された E-AoSAS++ の動的振る舞い図が，その記法を満たしているか否かを自動的に検査をするツールを提案する．これにより，E-AoSAS++ の記法を満たすアーキテクチャ記述の支援が可能となる．

本研究では，E-AoSAS++ の動的振る舞い図の検査内容を整理し，検査ツールの設計と実現をおこなった．その結果，特定の UML ツールで記述されたアーキテクチャが形式を満たしているか否かの検査の自動化を可能にした．

2　組込みソフトウェアのためのアスペクト指向ソフトウェアアーキテクチャスタイル (E-AoSAS++)

E-AoSAS++ は，組込みソフトウェアのためのアーキテクチャスタイルである．E-AoSAS++ では，組込みソフトウェアアーキテクチャを，並行に動作する状態遷移機械 (CSTM) の集合として定義している．CSTM が互いにイベントを送りあい，連動し動作することでソフトウェアの機能を実現する．

2.1　並行に動作する状態遷移機械 (CSTM)

CSTM 内部の要素を，以下に示す．

並行処理アスペクト　CSTM を並行に動作させる処理をおこなう．

状態遷移アスペクト　状態遷移機械に関する処理をおこなう．

アプリケーションロジックアスペクト　CSTM が保持するデータに関する処理をおこなう．

アスペクト間記述 (Inter Aspect Description (IAD))　各アスペクト間の関連を記述する．

アスペクト間記述 (遷移時のアクション)　状態遷移アスペクトとアプリケーションロジックアスペクト間のアスペクト間記述を，遷移時のアクションと呼ぶ．CSTM が状態遷移したさいのアクションを記述する．

動的振る舞い図を記述するさい，状態遷移アスペクトと遷移時のアクションを記述する．

2.2　コンポジットコンポーネント

E-AoSAS++ では，アーキテクチャの構成を切替えの手段として，コンポジットコンポーネントが定義されている．

コンポジットコンポーネントは，各機能を実現するコンポーネントと，それらの active/sleep 状態を管理するポリシーで構成される．ポリシーは，管理下のコンポーネントの active/sleep 状態の管理と切替えをおこなう．また，ポリシーには管理下のコンポーネントをどのように取り扱うかを記述することができる．

3　E-AoSAS++ の動的振る舞い図の記法

E-AoSAS++ の動的振る舞いは，次に挙げるものがある．

- CSTM の状態遷移機械と状態遷移
- CSTM の状態遷移時のアクション
- CSTM の active/sleep 状態

本節では，これらの記法について述べる．

我々の研究室では，E-AoSAS++ に基づいたソフトウェアアーキテクチャの記法を，UML の記法を基に提案している．UML は広く利用されており，開発者の間でアーキテクチャ記述の理解が容易だと考えるからである．

3.1　動的振る舞い図で記述する図の相互関係

E-AoSAS++ のアーキテクチャの記述では，静的構造図をコンポーネント図を用いて記述し，CSTM の状態遷移機械をステートマシン図を用いて記述し，遷移時のアクションをシーケンス図を用いて記述する．

図 1 に，各図の相互関係を示す．

静的構造図で記述するすべての CSTM に対して，それ

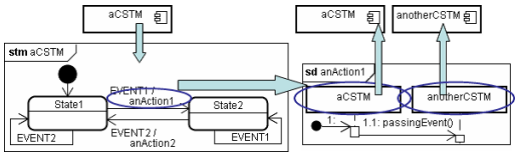

図 1　動的振る舞い図の相互関係

ぞれ 1 つのステートマシン図を記述する．

ステートマシン図に含まれる状態遷移に付与されたアクションに対して，対応するシーケンス図を記述する．

シーケンス図を用いて，遷移時のアクションで発生するイベント通知の記述をおこなう．シーケンス図のライフラインには，静的構造図で記述されている CSTM を用いる．

3.2　CSTM の状態遷移機械に関する記法

CSTM の状態遷移機械の記述には，UML のステートマシン図を用いる．記述例を図 2 に示す．

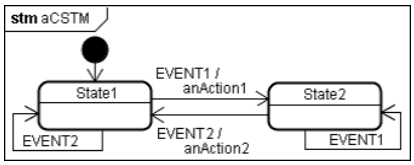

図 2　CSTM の状態遷移機械の記法

CSTM の状態遷移機械の記述には，UML の状態，状態遷移，初期疑似状態を用いる．

UML では，状態に，状態名のほかに内部アクティビティや内部遷移などを記述できるが，E-AoSAS++ の動的振る舞い図では，状態名のみを記述する．また，状態遷移には，イベント名，アクション名，ガード条件を記述できるが，E-AoSAS+ では，ガード条件は記述しない．

3.3　CSTM の状態遷移時のアクションに関する記法

CSTM の状態遷移時のアクションは，UML のシーケンス図を用いて記述する．その記述例を図 3 に示す．

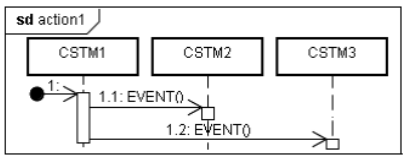

図 3　CSTM の状態遷移時のアクションの記述例

遷移時のアクションを，UML のライフライン，メッセージ，メッセージ終了点を用いて記述する．ライフラインは CSTM を示し，メッセージは CSTM 間のイベント通知を示す．

このシーケンス図の最初のメッセージとして，名前のないメッセージ終了点からのメッセージを記述する．これは，CSTM の初期 active/sleep 状態を定義するシーケンス図と視覚的に区別する目的で用いる．

基準となる CSTM

E-AoSAS++ の動的振る舞い図には，ステートマシン図の状態遷移のアクションに対応して 1 つのシーケンス図が記述される

シーケンス図のイベント通知元は，すべて対応する CSTM で一致していなければならない．

図 3 では，`CSTM1` のアクション `action1` を記述しているので，2 つのイベント通知元は `CSTM1` で一致している．

ポリシーでない CSTM の遷移時のアクションの記述では，`active` と `sleep` のイベントを記述することはできない．

3.4　CSTM の初期 active/sleep 状態の記法

CSTM の初期 active/sleep 状態の記述にもシーケンス図を用いる．その記述例を図 4 に示す．

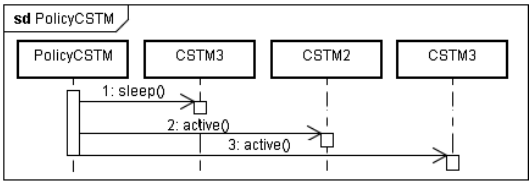

図 4　CSTM の初期 active/sleep 状態の記法

このシーケンス図を，それぞれの PolicyCSTM に対して 1 つ，PolicyCSTM が管理しているすべての CSTM に対して，`active` または `sleep` のイベントを通知する形で記述する．

4　E-AoSAS++ の動的振る舞い図の検査

本研究では，動的振る舞い図の検査を，次の 2 種類に分けて提案する．

- アーキテクチャ記述が形式を満たしているかの検査 (形式検査)
- アーキテクチャが誤りを含んでいないかの検査 (誤り検査)

本章では，検査すべき事項をこの 2 つに分類して述べる．

4.1　動的振る舞い図の形式検査

形式検査では，3 節で説明した記法通りに記述されているかを検査する．

4.2　動的振る舞い図の誤り検査

動的振る舞い図の形式検査をおこなっても，アーキテクチャが記述ミスを含んでいる場合がある．

ステートマシン図とシーケンス図において記述ミスの可能性がある点を検査することで，開発者の記述ミスを指摘し，アーキテクチャ記述の支援をおこなうことができる．

無効なイベント通知の検出

シーケンス図のメッセージとして記述されたイベント通知のうち，通知先 CSTM によって受理されることのないイベント通知は無効である．

無効な状態遷移と無効な状態の検出

無効な状態遷移とは，ステートマシン図の状態遷移のうち，状態遷移のイベント通知がいずれのシーケンス図にも記述されていない状態遷移を指す．また，この状態遷移によってのみ到達する状態は無効である．

5 E-AoSAS++ の動的振る舞い図の検査ツール

本研究では，次の 2 つのツールを作成する (図 5).

- 形式検査をおこない，中間形を生成するツール (形式検査と中間形生成ツール)
- 中間形から誤り検査をおこなうツール (誤り検査ツール)

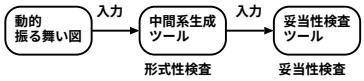

図 5　作成するツール

本研究では, UML ツールの一例として, SparxSystem 社の UML ツール“ Enterprise Architect ”[5] (以下，UML ツールと記述) を用いた.

5.1 形式検査と中間形生成ツールの設計

形式検査と中間形生成ツールの入力の決定

本研究では，UML ツールからダイアグラム情報を他のアプリケーションへ渡す手段として，次の 2 つの方法を検討した.

- UML ツールに用意されている外部アプリケーションを開発するための API を用いて，UML ツール用の外部アプリケーションを作成する.
- UML ツールの，ダイアグラムを XML 出力する機能を用いて XML 文書を出力し，出力された XML 文書を解析する.

本研究では，後者の方法を採用した．その理由は，UML ツール用の外部アプリケーションが対応しているプラットフォームの制限があるからである．また，作成した外部アプリケーションを実行するコンピュータに，その UML ツールがインストールされていなければならない．これらの制限は，生成する中間形を利用する一連のツールの利便性を欠く原因となってしまう.

形式検査と中間形生成ツールの出力の決定

このツールは，XML 文書を解析し，形式検査の後に中間形を構築し出力する.

中間形の設計

E-AoSAS++ の動的振る舞い図では，個々の図が大きな役割を担う.

本ツールでは，図という単位を保持するために，ステートマシン図を状態と状態遷移の集合とし，シーケンス図を CSTM とイベント通知の集合とした．また，動的振る舞い図をステートマシン図とシーケンス図の集合とした (図 6).

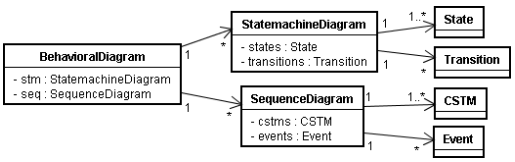

図 6　中間形の設計

ID によるモデル要素の識別

UML ツールでは，すべてのモデル要素に，一意に識別するための ID が付けられている．本研究では，以下のモデル要素の ID を用いて，すべての図とモデル要素の識別をおこなう.

- ステートマシン図，シーケンス図
- 状態，ライフライン (CSTM)
- 状態遷移，メッセージ (イベント通知)

ステートマシン図のクラス設計

ステートマシン図に含まれる要素には次のものがある.

状態 (State)　状態は，状態名と ID を持つ．また，開始状態であるかを区別できる必要がある.

遷移 (Transition)　遷移は，ID，遷移元状態と遷移先状態，イベント名とアクション名を持つ.

すべての CSTM は異なるステートマシンを持つことから，ステートマシン図は名前と ID，ステートマシンを所有する CSTM を持つ.

以上を基にしたクラス設計を図 7 に示す.

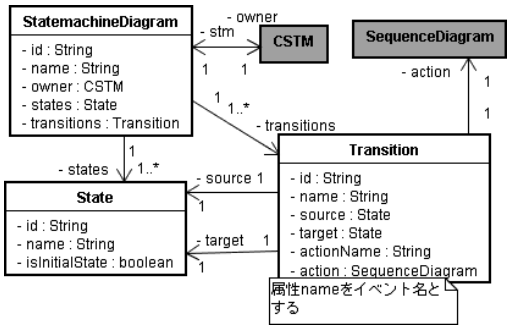

図 7　ステートマシン図のクラス設計

シーケンス図のクラス設計

シーケンス図に含まれる要素には次のものがある.

ライフライン (CSTM)　CSTM は，CSTM 名と ID を持つ．CSTM は，PolicyCSTM であるかどうか，初期状態が active 状態か，sleep 状態かを区別できる必要がある.

メッセージ (Event)　イベントは，イベント名と ID，通知元 CSTM と通知先 CSTM を持つ.

以上を基にしたクラス設計を図 8 に示す.

5.2 誤り検査ツールの設計

動的振る舞い図の誤り検査をおこなうツールでは，形式検査と中間形生成ツールが出力する中間形に対して誤り検査をおこなう．誤りが見つかった場合，該当箇所と誤りの内容の通知をおこなう.

6 考察

6.1 検査内容に関する考察

動的振る舞い図を想定される記述誤りを次に挙げる.

1. ステートマシン図
   (a) 記述形式の間違い

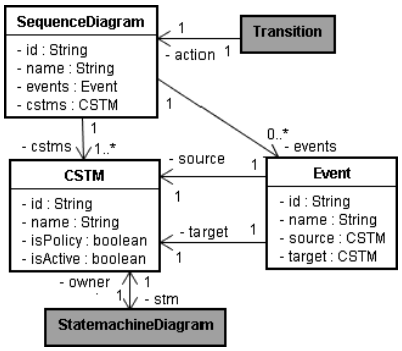

図 8　シーケンス図のクラス設計

(b) 状態遷移の遷移元や遷移先の記述間違い
(c) 状態遷移時のイベントの記述間違い
(d) 状態遷移時のアクション名の記述間違い

2. シーケンス図
   (a) 記述形式の間違い
   (b) シーケンス図の名前の記述間違い
   (c) イベントの送信元 CSTM や送信先 CSTM の記述間違い
   (d) イベント名の記述間違い
   (e) イベント通知のシーケンスの順番の記述間違い

これらの誤りのうち，本研究で提案する検査で次の誤りは発見可能である．

1. ステートマシン図
   (a) 記述形式の間違い
   (b) 状態遷移のイベントに対応したメッセージが無い
   (c) 状態遷移時のアクション名に対応するシーケンス図が無い
2. シーケンス図
   (a) 記述形式の間違い
   (b) シーケンス図の名前に対応する状態遷移のアクションが無い
   (c) イベントの通知先 CSTM がイベントを受理しない

その他の発見できない記述誤りは，記述されたアーキテクチャを実現したアプリケーションを実行し，予期しない結果が得られることでしか発見できない．実行結果を予測することは，記述検査の範囲を越える検査である．

以上より，今回提案する検査内容は妥当であると考える．

6.2 中間形の妥当性に関する考察

本研究で作成した中間形生成ツールを，われわれの研究室で提供されているソースコード自動生成ツールで利用することで妥当性の検証をおこなった．

ソースコード自動生成ツールにおいて，ソースプログラムに必要な情報がすべて出力できたので，本研究で設計した中間形および中間形生成ツールは妥当であると考える．

6.3 ツールの汎用性に関する考察

本研究で作成した中間形生成ツールは，特定の UML ツールが出力する XML 文書を対象とした検査ツールである．それ以外の UML ツールが出力する XML 文書の検査をおこなうことができない．

この問題の解決策として，XMI[4] にしたがった XML 文書から中間形を生成するツールを作成することが考えられる．

本研究で XMI を採用しなかった理由は，XMI の最新である XMI2.1 において，ダイアグラムという単位が提供されないことによる．本研究では，遷移時のアクションとシーケンス図を，名前を一致させることで対応付けているので，ダイアグラムという単位がない XMI を利用することはできない．

XMI を利用するためには，ダイアグラムの名前ではない別の方法でステートマシン図とシーケンス図を対応付けする必要がある．

7 おわりに

本研究の成果として，次の 2 点を挙げる．

- E-AoSAS++ の動的振る舞い図の記述検査の内容を整理した．
- UML ツール“ Enterprise Architect ”で記述された動的振る舞い図の形式検査と，中間系の生成を可能にした．

今後の課題として，次を挙げる．

- 動的振る舞い図の記述検査ツールの作成．
- UML ツールに依存しないツールの作成．

謝辞

本研究を進めるにあたり，熱心な御指導をいただいた沢田篤史教授，野呂昌満教授，有益なアドバイスをくださった蜂巣吉成先生，大学院生のみなさんに深く感謝いたします．また，いつも励まし合い頑張ってきた沢田研究室，野呂研究室，蜂巣研究室のみなさんに感謝いたします．

参考文献

[1] Brett McLaughlin, ”Java & XML 第 2 版”, 2002.5
[2] Martin Fowler, ”UML モデリングのエッセンス 第 3 版”, 2005.6
[3] OMG, ”UML2.0 仕様書”, 2006.11
[4] Timothy J. Grose, Gary C. Doney, Stephen A. Brodsky, ”Mastering Xmi: Java Programming With Xmi, Xml, and Uml”, 2002.4
[5] Sparx Systems Japan, ”Enterprise Architect”, http://www.sparxsystems.jp/ (2008.1 accessed)
[6] 株式会社オージス総研 オブジェクトの広場編集部, ”その場でつかえる しっかり学べる UML2.0”, 2006.1