モデル駆動型アーキテクチャを用いたアスペクト指向ソフトウェアアーキテクチャからのコード生成に関する研究
− プラットフォームを Java 言語にして −

2004MT037 金森 亮太　　2004MT044 川島 寛之
指導教員　沢田 篤史

1　はじめに

ソフトウェア開発において，ソフトウェアアーキテクチャからプログラムコードを自動生成することで，コーディング量の削減によるプログラミングの労力を省力化することが一般的に行われている．プログラムコードを自動生成する手法のひとつにモデル駆動アーキテクチャ (以下, MDA)[1] がある．MDA では，プラットフォームに独立なモデル (以下，PIM) を定義し，PIM をプラットフォームに依存したモデル (以下，PSM) に変換することで，様々なプラットフォームのプログラムコードの生成が可能である．

われわれの研究室では，組込みソフトウェア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 (以下, E-AoSAS++) を提案している．E-AoSAS++ では組込みソフトウェアのアーキテクチャを並行に動作する状態遷移機械の集合として規定する．並行処理や状態遷移などの関心事 (コンサーン) をアスペクトとして適切にモジュール化することで，再利用性，柔軟性の高いソフトウェアを作成することが出来る．E-AoSAS++ に基づくソフトウェア開発の問題点として，E-AoSAS++ に基づき記述したアーキテクチャからプラットフォームコードへの自動生成方法の考察が行われていない点が挙げられる．E-AoSAS++ に基づいたアーキテクチャを実現したプログラムコードには定型的なパターンが現れるので，定型箇所を自動的に生成するコード生成ツールを作成することで，コーディング量を大幅に削減できると考える．

本研究の目的は, MDA の概念を用いて, E-AoSAS++ に基づき記述したアーキテクチャから対象プラットフォームを言語とした PSM のプログラムコードを自動生成するツールの考察と実現である．我々は特に対象プラットフォーム言語を Java として開発を行った．研究手順を以下に示す．

1. E-AoSAS++ に基づくプラットフォームコードの設計
2. 自動生成可能箇所，ユーザ記述箇所の明確化
3. PIM から PSM へのモデル変換の考察
4. コード生成ツールの作成

2　E-AoSAS++

E-AoSAS++ では組込みソフトウェアのアーキテクチャを並行に動作する状態遷移機械 (以下，CSTM) と複数の CSTM の構成を管理する PolicyCSTM の集合として規定する．CSTM はイベントを受理することで動作し，複数の CSTM が連動動作することで組込みソフトウェアの機能を実現する．E-AoSAS++ は，並行処理や状態遷移などの組込みソフトウェアにおける横断的に存在するコンサーンをアスペクトとしてモジュール化する．コンサーンは組込みソフトウェア全体に存在するグローバルコンサーンと，組込みソフトウェアの特定のコンポーネントに存在するローカルコンサーンに分類される．

2.1　グローバルコンサーン

E-AoSAS++ では並行処理，状態遷移のコンサーンをグローバルコンサーンと規定する．E-AoSAS++ に基づく組込みソフトウェアでは，コンサーンをアスペクトとしてモジュール化した並行処理アスペクト，状態遷移アスペクトと，各 CSTM 特有の機能を実現するコンポーネント群であるアプリケーションロジックアスペクトによって CSTM と PolicyCSTM を構成する．各アスペクトの関連を図 1 に示す．並行処理アスペクトは他の CSTM から受信したイベントを保持し，順にイベントを状態遷移アスペクトに送信する．状態遷移アスペクトは並行処理アスペクトから受理したイベントによって CSTM の状態を遷移させる．アプリケーションロジックアスペクトは，状態が遷移した際のアプリケーションロジックを実行する．

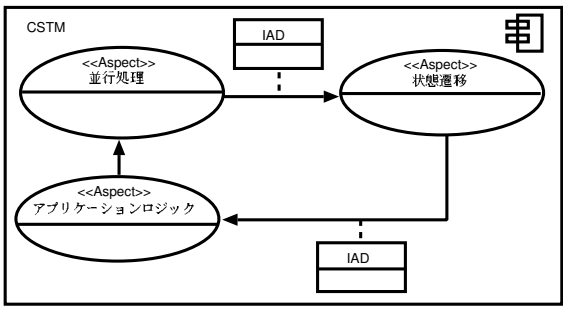

図 1　アスペクト間の関連

2.2　ローカルコンサーン

ローカルコンサーンは特定のコンポーネントに横断的に現れるコンサーンである．E-AoSAS++ では次のようなコンサーンをローカルコンサーンとして規定している．

- 例外処理 :故障などの例外的な状態で使用される処理
- 耐故障性処理 :故障などが起きたときに正常な動

作を保証する処理

- 実時間処理 :時間計測に関する処理

E-AoSAS++ ではローカルコンサーンを PolicyCSTM と CSTM で構成した CompositeCSTM で実現する．PolicyCSTM は CSTM の実行状態を管理する CSTM であり，管理している CSTM の active，sleep 状態を切替えることで，ローカルコンサーンを実現する．

3 コード生成ツールの実現

3.1 コード生成ツールの概要

E-AoSAS++ ではソフトウェアアーキテクチャを CSTM の集合として記述する．アーキテクチャの静的構造をコンポーネント図を用いて記述し，各 CSTM の状態遷移と振舞いはそれぞれステートマシン図とシーケンス図を用いて記述する．本研究で提案するコード生成ツールの処理の流れを図 2 に示す．ツールは，E-AoSAS++ に基づいて記述された情報を基に言語独立な PIM を作成する．その後，作成された PIM から各プラットフォーム言語に依存した PSM に変換してプログラムコードを生成する．なお，PIM は既に本研究室で提案されているものを用いる．

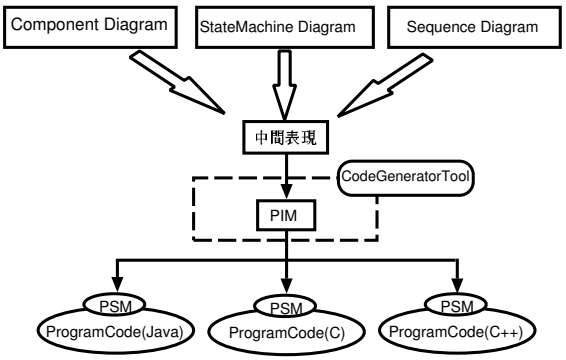

図 2　コード生成の流れ

3.2 プラットフォームコードの設計

本節では，CSTM の各アスペクトを構成するコンポーネント群について述べる．

3.2.1 並行処理アスペクト

並行処理アスペクトは CSTM の並行実行に関するアスペクトであり，CSTM が扱うイベントの追加，排出を行う Queue と，並行実行を実現する Thread，他の CSTM とのインターフェースの役割を担う Concurrency のコンポーネント群で構成される．

3.2.2 状態遷移アスペクト

状態遷移アスペクトは CSTM の状態の遷移に関するアスペクトであり，並行処理アスペクトと状態遷移アスペクト間のアスペクト間記述を表す PolicyStateTransition コンポーネント，状態を管理する StateTransition コンポーネント，各状態を表す State のコンポーネント群で構成される．

3.2.3 アプリケーションロジックアスペクト

アプリケーションロジックアスペクトは CSTM ごとの振舞いに関したアスペクトであり，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表す PolicyApplicationLogicAspect コンポーネント，各イベントに対応した処理を行う Action コンポーネント群とデータの操作を行う ApplicationLogic のコンポーネントで構成される．

3.3 プラットフォーム非依存モデル (PIM)

コード生成ツールではコンポーネント図，ステートマシン図，シーケンス図からの情報を基に作成された抽象構文木を PIM とする．PIM のノード構成を図 3 に示す．PIM は E-AoSAS++ に基づいたアーキテクチャに現れる CSTM の階層構造と，3.2 節で示した各 CSTM のコンポーネントの構成を表現している． 3.2 節で示したコンポーネント群に対応するノード群の持つデータを表 1 に示す．各ノードはそれぞれプログラムコードの生成に必要な情報を含んでいる．

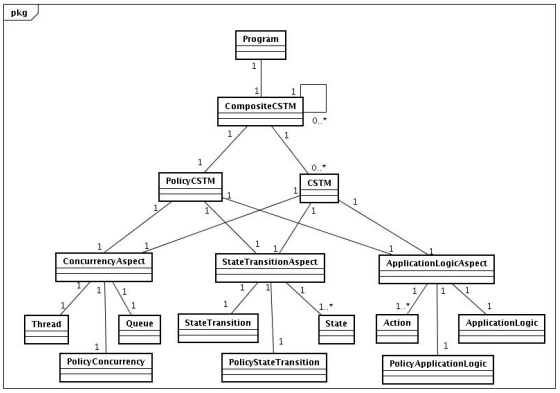

図 3　PIM

表 1　ノードが保持している情報

ノード名	保持しているデータ
PolicyConcurrency	CSTM 名，初期メタ状態
Thread	CSTM 名
Queue	CSTM 名
PolicyStateTransition	CSTM 名
StateTransition	CSTM 名，状態名， 初期状態名，イベント名
State	CSTM 名，遷移前の状態名， イベント名，遷移後の状態名
PolicyApplicationLogic	CSTM 名，状態名， イベント名，アクション名
Action	CSTM 名，アクション名， イベント名，他の CSTM 名
ApplicationLogic	CSTM 名

3.4 プラットフォーム依存モデルの設計 (PSM)

Java 言語を PSM とした E-AoSAS++ のプラットフォームコードでは，各アスペクトを構成するクラスの汎用的な属性と操作をライブラリとしてあらかじめ用意した．

3.4.1 並行処理アスペクト

並行処理アスペクトの構成を図 4 に示す．並行処理アスペクトは Queue，Thread，Concurrency クラスで構成される．Queue クラスは Java 標準ライブラリの Vector クラスを用いることで，入出力方式を FIFO とした可変長配列のキューを実現し，Thread クラスは Java 標準ライブラリの Thread クラスのサブクラスとすることで並行実行を実現している．Concurrency クラスは他の CSTM と Thread クラス，Queue クラスとのインターフェースの役割を担う．また並行処理アスペクトではすべての CSTM における共通の記述をライブラリ化することで，生成したい定型コードの記述を簡略化した．

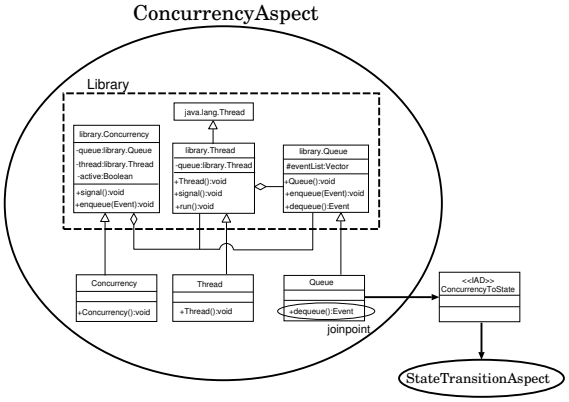

図 4　並行処理アスペクト

3.4.2 IAD(並行処理→状態遷移)

並行処理アスペクトと状態遷移アスペクトをつなぐアスペクト間記述では，Queue クラスのイベント排出をジョインポイントとし，アドバイスでは Queue から排出されたイベントを状態遷移アスペクトに対して送信する．

3.4.3 状態遷移アスペクト

状態遷移アスペクトの構成を図 5 に示す．状態遷移アスペクトは CSTM の状態の遷移に関するアスペクトであり，デザインパターン [2] の State パターンを用いて実現する．状態遷移アスペクトは状態を管理する StateTransition クラスと，状態遷移機械のひとつの状態を表す ConcreteState クラス，そのインターフェースである State クラスから構成される．

3.4.4 IAD(状態遷移→アプリケーションロジック)

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をつなぐアスペクト間記述では，StateTransition クラスの Trans メソッドにおいてイベントを受けて状態が遷移した際をジョインポイントとし，アドバイスではイベントに対応する処理を実行する．

3.4.5 アプリケーションロジックアスペクト

アプリケーションロジックアスペクトの構成を図 6 に示す．アプリケーションロジックアスペクトは CSTM ごとの振舞いに関するアスペクトであり，Action クラス，ConcreteAction クラス，ApplicationLogic クラスから構成される．アプリケーションロジックアスペクトでは

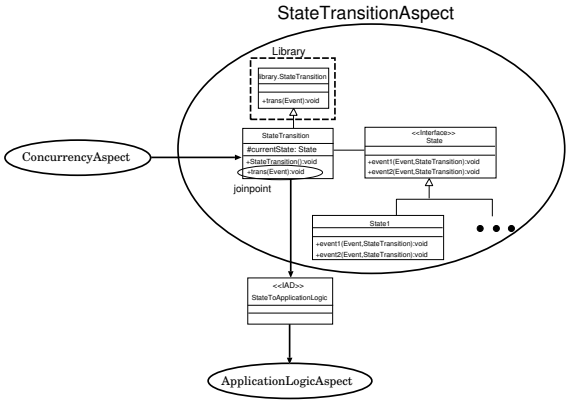

図 5　状態遷移アスペクト

デザインパターン [2] の Command パターンを用いて実現している．アプリケーションロジックアスペクトはイベントに対応した処理を ConcreteAction クラスとしてカプセル化し，Action クラスのサブクラスとしている．ApplicationLogic クラスは各 CSTM が保持するデータおよび，データへの操作を行うメソッドを記述する．

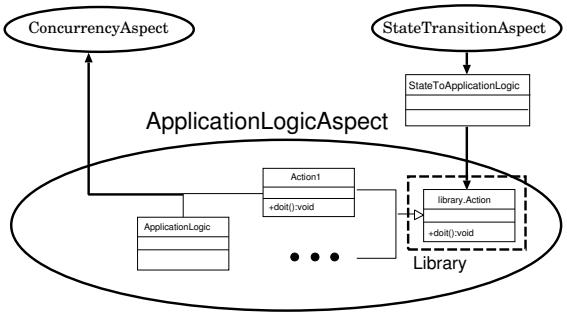

図 6　アプリケーションロジックアスペクト

3.5 モデル変換論理の設計

コード生成ツールは，抽象構文木として表された PIM をルートから順にたどりながら，各ノードにおいて対応する PSM のクラスに変換している．PIM のノードと PSM のクラスの対応を図 7 に示す．プログラムコードには，定型コードに加えて PIM のノードから取得した状態名やイベント名などの状態遷移機械ごとの情報が必要である．対応する PIM のノードから情報を取得することでコードの自動生成を行う．

例として図 8 を用いて，PIM の StateTransition ノードから PSM の StateTransition クラスへのモデル変換について説明する．StateTransition クラスは状態遷移機械の状態を管理するクラスであり，State パターンを用いたプログラムコードである．このプログラムコードをすべて自動生成するためには，定型コードに加えて状態遷移機械名とそれに対応したステートマシン図に現れる状態名，イベント名，初期状態名が必要となる．PIM ノードには生成に必要なデータが含まれているの

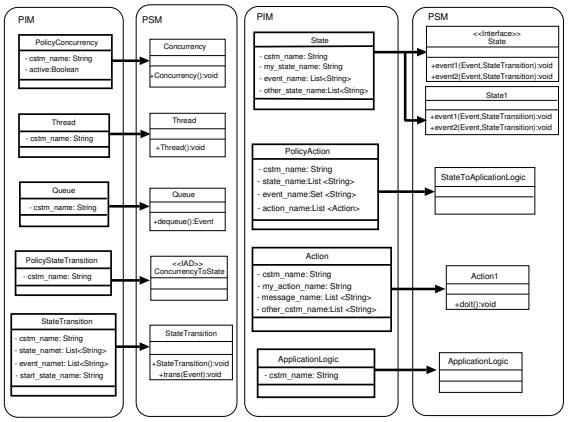

図 7　PIM と PSM の対応関係

で，PIM からデータを取得することでプログラムコードの生成を可能とする．図 8 の場合，宣言部分やコンストラクタ，case 文において PIM の情報が複数必要となるので，状態名などを含んだリストから一つずつ要素を取り出して，右下のように複数行のコードをブロックとしてまとめて生成することで，プログラムコードを生成している．

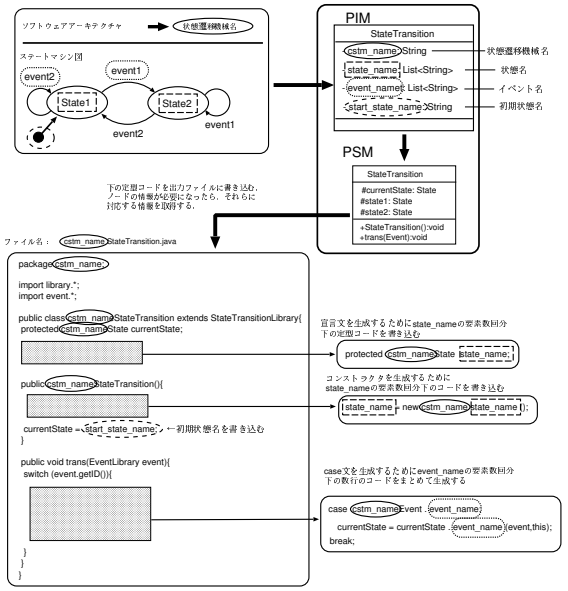

図 8　モデル変換の例

4　評価と考察

4.1　自動生成の評価

表 2 に本研究で作成したコード生成ツールの自動生成可能箇所を示す．表 2 に示すように並行処理アスペクト，状態遷移アスペクト，アスペクト間記述は定型コードに加えて PIM からの情報を組み合わせることで全て自動生成することが出来た．アプリケーションロジックアスペクトではデータの取り扱いに関しての記述がアプリケーションに依存するので，プログラムコードの一部のみ自動生成を行った．

表 2　自動生成可能箇所

構成要素	自動生成
並行処理アスペクト	○
状態遷移アスペクト	○
アプリケーションロジックアスペクト	×
IAD(並行処理→状態遷移)	○
IAD(状態遷移→アプリケーションロジック)	○

4.2　モデル変換における考察

他のプラットフォーム言語での実現方法について考察を行う．本研究では Java をプラットフォーム言語としたコード生成の際に，各 CSTM での共通の処理をライブラリとして用意しておき，それらを API として使用するプログラムコードのみを自動生成している．他のプラットフォーム言語においても，同様に API を作成し，それを使用するプログラムコードを自動生成することで実現できると考える．しかし，E-AoSAS++ は並行に動作する状態遷移機械の集合としているので，並行実行を実現できる環境が必要であると言える．また，C 言語などのオブジェクト指向言語以外の言語では，構造体をクラスとして定義するなど疑似的にオブジェクト指向を表現することで，E-AoSAS++ に基づく組込みソフトウェアを実現できると考える．

5　おわりに

本研究では，E-AoSAS++ に基づき構築したアーキテクチャから Java 言語を対象プラットフォームとする PSM のプログラムコードを自動生成するコード生成ツールを考察，実現した．今後の課題としてユーザ記述部分の省力化の解決などがあげられる．

謝辞

本研究を進めるにあたり熱心な指導をしていただいた，野呂先生，沢田先生，蜂巣先生，適切なアドバイスをしていただいた大学院生のみなさまに深く感謝します．また、いつも励ましあい研究を頑張ってきた野呂研究室，沢田研究室，蜂巣研究室のみなさまに感謝します．

参考文献

[1] Model Driven Architecture : OMG, http://www.omg.org/mda/.

[2] Erich Gamma, Richard Helm, Ralph Johnson, John Vissides : Design Patterns, Addison Wesley Longman(1995).

[3] 坂野　将秀：組込みソフトウェアのためのアスペクト指向アーキテクチャスタイルの提案 南山大学大学院数理情報研究科修士論文要旨集 (2006).