3J-1

委譲型 AWS フレームワークの実現方法

二宮良太† 大友浩照† 大谷真†
湘南工科大学†

1. はじめに

AWS(自律型 Web サービス)フレームワークの目的は、独立に定義したビジネスプロセスモデル（BPM）を取引開始時に協調変形を行い、その BPM に従って、取引を行うためのアプリケーションを実行することである。従来の AWS フレームワークでは、アプリケーションがフレームワークの機能を利用するには AWSFramework クラスを継承する必要がある。このため、アプリケーションは独自の上位クラスを置くことが出来ない問題があった。本研究では既存の AWSFramework クラスにデリゲーションパターンを適用することで、この問題を解決した。

2. AWS ミドルウェア

AWS ミドルウェア[1]とは、独立に定義された BPM を持つシステムが遭遇したときに、互いの BPM を交換し、協調変形を行うことで、取引が実行出来るようにすることを目的としている。AWS フレームワークは AWS ミドルウェアの一部である。AWS フレームワークでは、BPM に従って定義された取引を行うためのアプリケーション（以降 APS と呼ぶ）を実行制御する。APS にはシステムが持つ入出力オペレーションに対応するメソッド（APS メソッド）が定義される。フレームワークは BPM の状態変化に応じて APS メソッドを実行することで取引を実行する。

3. 従来の AWS フレームワークの問題点

問題点 1：従来の AWS フレームワークでは APS クラスは AWSFramework クラスを継承する必要があるため、独自の上位クラスを置くことができない。したがって、既存の AWSFramework クラスにデリゲーションパターン[2]を適用する必要がある。このデリゲーションパターンは実際の処理を他のクラスへと移すこと(委譲)ができる。

問題点 2：AWSFramework クラスは、オペレーションの実行制御機能を提供している。具体的には、次に実行可能なオペレーションの取得と、次に実行するオペレーションを設定する機能の 2 つである。単純にデリゲーションパターンを適用すると、この機能へアクセスできなくなってしまう。したがって、オペレーションの実行制御機能へアクセスするためのオブジェクトを介して制御する必要がある。これら 2 つの問題する委譲型 AWS フレームワークを開発した。

Delegation pattern applied to Framework for AWS Middleware
†Ryota Ninomiya, Hiroaki Otomo, Makoto Oya, Shonan Institute of Technology

4. 委譲型フレームワーク

4.1. フレームワークの構成

図 1 に AWS フレームワークの構成を示す。各実線は各オブジェクトの関係を示す。これらのうち APS 開発者が開発するクラスは Main と APS の実装である MyAPS クラスである。今回新たに追加した APSController クラスと OperationContoller クラスは AWS フレームワークの一部である。APSController クラスは MyAPS オブジェクトを実行制御する。OperationController クラスはオペレーション実行制御を行う。また、Manager クラスは APSController オブジェクトの起動を行う。Manager クラスはスレッドプールを保持しており、execute メソッドを実行することで、APSController オブジェクトがスレッドプールに追加される。従来の AWSFramework クラスと異なり、MyAPS は APS メソッドのインタフェースを実装するようにした。今回の変更に伴い、AWSFramework クラスは廃止した。

4.2. 本提案の API

今回新たに作成した、APSController のメソッド一覧を表 1 に示す。APS オブジェクトを設定するには setAPS メソッドを使用する。Config ファイルを指定するには setConfig メソッドを使用する。この Config ファイルには各入出力オペレーションに対応する APS クラスのメソッド名やパラメータの定義を記述する。データフォーマットの整合性を確認するためのメソッド(dMatch メソッド)を設定するには setDataMatcher を実行する。BPM オブジェクトを設定するには setHBPM メソッドを実行する。オペレーションの実行制御用オブジェクトを取得するには getOperationController メソッ

ドを実行する。

次に、APS メソッドのインタフェースを以下に示す。

void メソッド名(データコンテナクラス)

データコンテナクラスには各入出力オペレーションで使用するデータを格納するクラスを指定する。

さらに、オペレーション実行制御用のメソッド一覧を表 2 に示す。次に実行可能なオペレーションの一覧を取得するには getNextOperation を実行する。次に実行するオペレーションの設定を行うには setNextOperation を実行する。

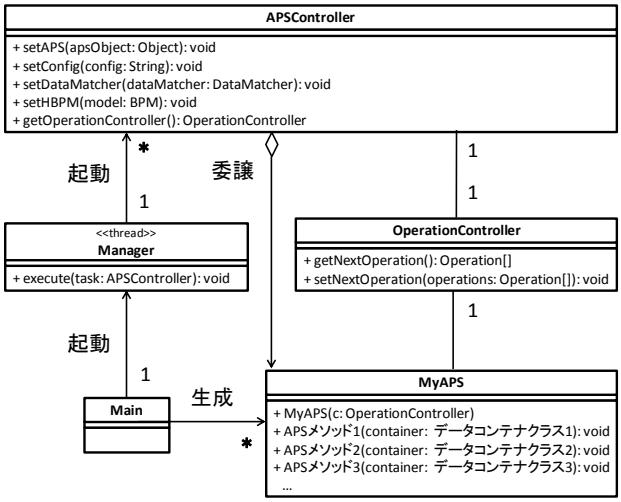

図 1 AWS フレームワークの構成

表 1 APSContoroller のメソッド一覧

メソッド	詳細
setAPS	APS オブジェクトを設定
setConfig	Config ファイルを設定
setDataMatcher	dMatch メソッドを設定
setHBPM	BPM オブジェクトを設定
getOperationController	オペレーション実行制御用オブジェクトを取得

表 2 オペレーション実行制御用のメソッド一覧

メソッド	詳細
getNextOperation	次に実行可能なオペレーションを取得
setNextOperation	次に実行可能なオペレーションを設定

4.3. APSController クラスの仕様

APS クラスはデータ構造やメソッドの定義は自由に定義される。このため、APSController クラスは事前に APS クラスのインスタンス化の方法や APS メソッド以外のメソッドの実行方法を与えることは難しい。そこで、setAPS メソッド実行することで、APSController オブジェクトが APS オブジェクトを保持するようにした。APSController オブジェクトは APS メソッドを実行するとき、保持している APS オブジェクトに対して実行（処理を委譲）する。

4.4. AWS アプリケーション

Main のプログラム例を図 2 に示す。このプログラムでは、Manager を事前にスレッドを割り当てて起動しておく。MyAPS オブジェクトが必要になっ時点で、APSContoroller オブジェクトと、MyAPS オブジェクトを生成する。このとき、APSController の getOperationController を呼び出すことで、オペレーション実行制御用オブジェクトを取得している。MyAPS はこのオブジェクトを介してオペレーションの実行制御を行う。次に、生成した APSController オブジェクトに対して、APS オブジェクトと BPM オブジェクト、さらに、今回追加した dMatch メソッドを実装するオブジェクトの設定を行う。従来のフレームワークでは継承したときに dMatch メソッドを実装する必要があったが、今回新たに追加した setDataMatcher メソッドで設定するよう変更した。最後に、APSController オブジェクトを execute メソッドで起動する。

```
m = new Manager();
(new Thread(m)).start();
whlie (1){
  c = new APSController();
  aps = new MyAPS(c.getOperationController());
  c.setAPS(aps);
  c.setHBPM(BPM オブジェクト);
  c.setDataMatcher(dMatch 実装オブジェクト);
  …
  m.execute(c);
}
```

図 2 Main プログラムの例

5. まとめ

今回既存の AWS フレームワークにデリゲーションパターンを適用することで、上位クラスを置くことが可能になった。また、オペレーション実行制御用オブジェクトによりデリゲーションパターンを適用することが可能となった。本研究は科研費(21500110)の助成を受けたものである。

6. 参考文献

[1] 二宮, 平本, 安齋, 大谷: AWS（自立型WEBサービス）ミドルウェアフレームワーク制御, 情報処理学会第73回全国大会, pp.1-707-708, 2011

[2] T.C.Lethbridge, R.Laganiere: Object-Oriented Software Engineering –Practical Software Development using UML and Java – 2nd Edition, 2005