AWS ミドルウェアの研究 -アプリケーションフレームワーク層-

高木良輔†　塚本修也†　木村泰輔†　大谷真†
湘南工科大学情報工学科†

1. はじめに

自律型 Web サービス(AWS)のメッセージング基盤は、send-receive 型の非同期メッセージ送受信を実現する。しかし、アプリケーション作成にはイベント駆動型のフレームワーク機能が必要である。しかし現在のところ、このようなフレームワークは Web サービスメッセージングについては存在しない。

本研究では上記フレームワークについて方式検討・設計を行い、その後、機能と制御方式確認のためのプロトタイプを開発し、評価した。

2. AWS の構成 -フレームワーク層-

(1)AWS の構成

AWS は動的モデル協調層(MH 層)、アプリケーションフレームワーク層(AF 層)、自律型メッセージング層(MS 層)の 3 つの層で構成されている。

(2)フレームワーク層

図 1.に AF 層の構成を示す。

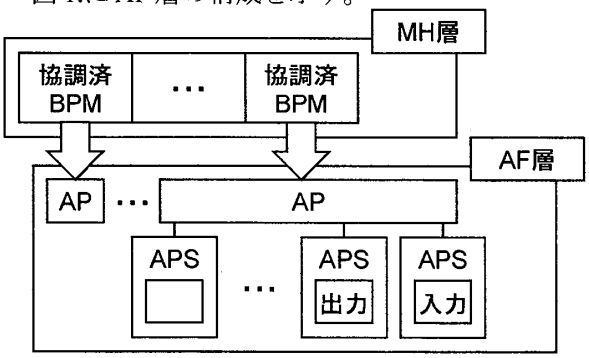

図 1.AF 層の構成

AF 層は MH 層から協調済 BPM を受け取る。受け取った協調済 BPM はアプリケーション(AP)に組み込まれる。AP は複数の場合もある。AP の中には、モデルに応じたアプリケーションセグメント(AP セグメント)が存在する。AP セグメントは 1 つの出力または入力に対応する商取引処理を実行する。AF 層は協調済 BPM から状態遷移に合わせたオペレーションを順次受け取り、実行する AP セグメントを選択する。選択された AP セグメントが出力の場合は取引相手にデータを送信し、入力の場合はデータを受信する。データの送受信には MS 層の API を使用する。

3. 実現方式

3.1 開発方針

(1) 1 つの長期セッション全体を 1 つのアプリケーションオブジェクトとする。

1 つの長期セッションを 1 つのオブジェクトとし、一連の商取引を 1 つのオブジェクトだけで扱えるようにした。これにより、複数の AP セグメントを 1 つのオブジェクトに入れることができ、AP セグメント間でのデータのやり取りは、アプリケーション内のフィールドを介して自然に受け渡しができる。

(2) 1 つの AP セグメントは 1 つの Java メソッドに対応させる。

1 つの AP セグメントに 1 つの Java メソッドを対応させ、それをユーザメソッドと呼ぶ。ユーザメソッドはユーザの BPM によって異なるので、ユーザメソッドの名前・引数は可変で、メソッドの数も自由にした。

(3)モデルとプログラムの独立性を確保する。

オペレーションとそれに対応したユーザメソッドとの関連を XML で記述した構成ファイルをユーザが作成する。これにより BPM の作成とユーザメソッドなどのプログラミングを分離し、独立性を確保した。

3.2 解決方法

図 2.に BPM の例を、図 3.にアプリケーションの例を、図 4.に構成ファイルの例を示す。

```
<operation name="見積依頼" format="form.est">
    <pattern>output</pattern>
</operation>
<operation name="見積結果" format="result.est">
    <pattern>input</pattern>
</operation>
```

図 2.BPM の例

```
public class User extends AWSFramework {
    public void sendEst(EstForm ef) {
        // 見積内容を ef にセットする
    }
    public void receiveEst(EstResult er) {
        // 見積結果が er にセットされている
    }
}
```

図 3.アプリケーションの例

Study on AWS middleware -Application Framework Layer-
† Ryosuke Takagi, Makoto Oya, Masaki Ito, Shuya Tsukamoto, Taisuke Kimura, Shonan Institute of Technology

```
<options>
  <operation name = "見積依頼">
    <method>sendEst</method>
    <parameter>EstForm</parameter>
  </operation>
  <operation name = "見積結果">
    <method>receiveEst</method>
    <parameter>EstResult</parameter>
  </operation>
</options>
```

図 4.構成ファイルの例

図 2.ではオペレーションの名前とフォーマット、入力か出力かを定義している。図 3.に示すようにアプリケーションは AWSFramework クラスを継承して作成する。アプリケーション内の sendEst メソッドと receiveEst メソッドが図 2.の BPM で定義されている“見積依頼”と“見積結果”に対応していて、その対応関係は図 4.のように構成ファイルで定義するようにした。また、各メソッドのパラメータは構成ファイルに定義された EstForm クラスと EstResult クラスのオブジェクトである。これらは BPM で定義されている各オペレーションの format にアクセスするためのメソッドを持つ。(4.2 参照)

4. 実装

4.1 実装

(1)フレームワーク層の動作

図 5.にフレームワーク層の動作を示す。

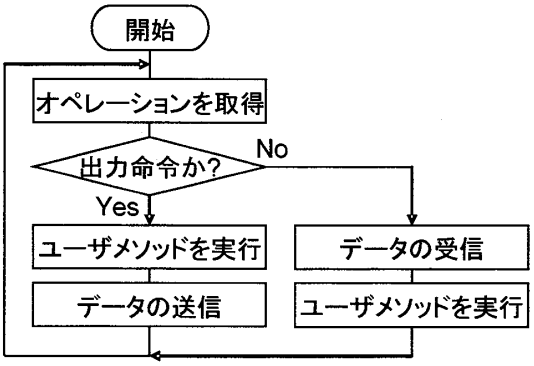

図 5.フレームワーク層の動作

まず、モデル協調層より渡された協調済 BPM のインスタンスからオペレーションを取得する。次に、そのオペレーションが出力オペレーションか入力オペレーションかを判断する。出力オペレーションだった場合はユーザメソッドを実行し、それによって得られたデータをメッセージング層に渡す。入力命令だった場合は受信したデータをメッセージング層から受け取り、そのデータをユーザメソッドのパラメータの一部としてユーザメソッドを実行する。その後、次のオペレーションを取得して一連の同じ動作を繰り返す。

(2)ユーザメソッドの動的実行

ユーザメソッドの実行にはまず、アプリケーションの起動時に構成ファイルを読み込んで、ユーザメソッドのメソッド名とパラメータとなるクラス、オペレーションとの対応をハッシュマップに格納する。そして、アプリケーション実行中に協調済 BPM より受け取ったオペレーションを使い、ハッシュマップからユーザメソッドのメソッド名とパラメータとなるクラスを引き当て、Java リフレクションを使うことで動的にユーザメソッドを実行することができる。

4.2 フォーマット

AWS ミドルウェアを使って取引相手とデータの送受信を行うが、送受信されるデータの書式は AWS ミドルウェアが決めるのではなく、ユーザが自由に決めることができる。しかし、そのデータ内容の解析方法を AF 層で用意しておくことはできない。この問題を解決する方法としてフォーマットという概念を利用した。フォーマットとは、送受信データの型のことで、送受信データに含まれる情報の形式・意味を定めたものである。このフォーマットに従って送信データを生成するメソッドと、受信したデータを取り込むメソッドをユーザメソッドの引数用クラスにユーザが実装する。ユーザが実装するメソッドを表 1.に示す。

表 1.ユーザが定義するメソッド

メソッド名	説明
generateMessageData	送信データを生成する
parseMessageData	受信データを取り込む

そして、送信オペレーションの実行時には generateMessageData を実行し、送信するデータを得る。受信オペレーション実行時には parseMessageData を呼び出して、受信されたデータを読み込むことができる。

5. まとめ

1 つの VLSession 全体を 1 つのアプリケーションオブジェクトとすることができた。1 つの AP セグメントに 1 つの Java メソッドを対応させることができた。モデルとプログラムの独立性を確保することができた。

本論文に述べた方式・仕様に従って、フレームワーク層のプロトタイプを作成した。同時に開発が行われた AWS モデル協調層・メッセージング層との連動テストを行い、目標機能が正しく動作することを確認した。これによって本論文で述べた方式の妥当性が検証できた。本研究は科研費(19500095)の助成を受けたものである。

参考文献

1. M. Oya, Autonomous Web Services Based on Dynamic Harmonization, IFIP I3E2008, Springer, ISBN:978-0-387-8590-2, pp.139-150, 2008

2. 伊東、塚本、木村、大谷、AWS ミドルウェアの研究、情報処理学会第 71 回全国大会、2009