3A-7

# 自律型 Web サービス（AWS）の原理と実装

大谷　真†

湘南工科大学†

## 1. はじめに

インターネット内の複数のサイト間での電子商取引などのビジネスメッセージ交換を考えてみよう．従来の Web サービスでは，BPEL に代表されるように，関連サイト全部のビジネスプロセスの流れ（ビジネスプロセスモデル；BPM）をあらかじめ詳細に取り決めておき，関連するサイトはそれに従って動作するように開発されていなければならない．自発的かつ自由に開発されたシステムはその取引に参加できない．自律型 Web サービス（AWS；Autonomous Web Services）は，自由に作られたサイト間でもビジネスプロセスが実行できることを目標としている．ここでは中心技術である動的モデル協調（DMH）の原理および AWS ミドルウェアの実装について，これまでの研究結果概略を述べる．

## 2. 動的モデル協調(DMH)

### 2.1 ビジネスプロセスモデル（BPM）

AWS では，BPM $M$ は，$M$= ($O$, $B$)と形式定義する．$O$ はそのシステムが持っているオペレーション $op$ = (*pattern*, *format*) の集合である．*pattern* はそのオペレーションが受信（'input'）であるか送信（'output'）であるかを示す．*format* は受け渡されるメッセージの形式を示す．$B$ は振る舞い（オペレーション実行の流れ）であり，一般には $O$ をアルファベット集合とする有限状態機械，すなわち $B$ = ($S$, $\lambda$, $F$, $\Phi$)である．ここに，$S$ は状態の集合，$\lambda(\in S)$は開始状態，$F(\subset S)$は終了状態の集合，$\Phi$は遷移関数である．ただし現時点では，$B$ は非決定性オートマトンの範囲（すなわち$\Phi$: $S \times O \rightarrow S$）に限定している．なお BPM は WSDL の自然な拡張形になるように工夫されている．$O$ は WSDL の interface または portType に相当し，*pattern* は MEP に対応している．AWS ミドルウェアでは XML を使って BPM の外部表現を定義している[4].

### 2.2 DMH の原理

DMH は次の 4 ステップで構成される．

①システム $Z_1$ と $Z_2$ は，各々の BPM $M_1$ = ($O_1$, $B_1$) と $M_2$ = ($O_2$, $B_2$)を公開（アクセス可能）しておく．

②$Z_1$ と $Z_2$ は，ビジネスプロセスを実行する時点で動的に，互いの BPM を交換する．

③それぞれのシステムは，相手システムから受け取った BPM と自システムの BPM を突き合わせて，両者が協調動作するように自システムの BPM を変形する．（これを DMH アルゴリズム，変形した BPM のことを協調済 BPM と呼ぶ．）

④すなわち変形が成功したら，協調済 BPM に従ってアプリケーションを駆動し，ビジネスプロセスに必要な一連のメッセージ交換を開始する．

### 2.3 DMH アルゴリズム

DMH アルゴリズムは[1]で最初に提案されその後改良が重ねられ[2][3]，2 システム間の DMH が完成した[4]．また 3 システム以上の間の DMH アルゴリズムも[5]で提案されている．アルゴリズムの概略は以下のとおりである：

- format マッチング関数 tMatch()を使って，$O_1$ と $O_2$ のオペレーションの対応関係を決定する．
- その結果から 2 オートマトンの直積をとる．
- 初期状態から到達不能または終了状態に到達不能な遷移パスなどを取り除く．
- 自システムの部分の射影をとる．

## 3. AWS ミドルウェア

2008 年から開発開始し[6]，メッセージング層の改良[7]，DMH 層の改良，フレームワーク層の並行処理化[9]を経て，2011 年度に完成した．

AWS ミドルウェアの実装上の基本方式はモデル駆動型 AP 実行である[3][4]．DMH の結果 BPM が変形されるが，それに合わせてアプリケーションの制御の流を自動的に変化させるために提案された．アプリケーションを各オペレーションに対応したメソッド(APS メソッド)の集まりとして記述しておき，協調済 BPM の状態遷移に応じて，ミドルウェアが自動的に対応する APS メソッドを実行する．

AWS ミドルウェアは次の 3 つのソフトウェア層から構成されている：

- 動的モデル協調層：協調済 BPM を生成.
- フレームワーク層：モデル駆動型 AP 実行制御.
- メッセージング層：メッセージングの実行.

### 3.1 動的モデル協調層

動的モデル協調層はクラス DMH として実装されている(図 1)．AP は DMH オブジェクトを生成した後，自 BPM の URL(図では URL1)と相手システムの URL(URL2)を，DMH オブジェクトに連絡しておく．一般に URL1 はローカルファイルで URL2 はインターネット上の URL だが，必ずし

---

Autonomous Web Services (AWS): Principles and Implementation

† Makoto Oya, Shonan Institute of Technology

もそれに限らない．DMH オブジェクトには tMatch()の実装クラスも登録しておく．DMH の実行を harmonize()で指示すると，DMH オブジェクトは 2 つの BPM を取得して DMH アルゴリズムを実行する．動的協調に成功すれば協調済 BPM オブジェクトが BPM クラスのインスタンスとせて作成され，フレームワーク層に渡される．

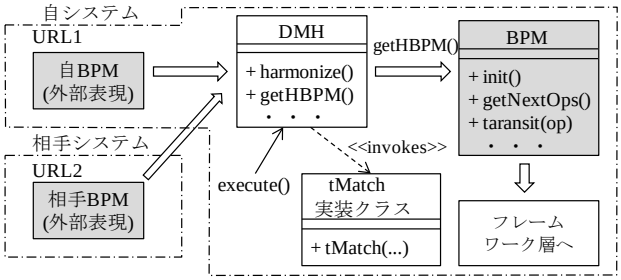


図 1 ：動的モデル協調層の構成

### 3.2 フレームワーク層

利用者は，APS メソッドの集まりである App クラスと Main クラスを開発する(図 2)．図 2 の一点鎖線内は，BP(ビジネスプロセス)インスタンス発生時に毎回インスタンス化され，その BP インスタンスが終了したときに削除される．インスタンスごとにスレッドプールを介して実行スレッドが割当てられ並行実行される．図 3 に Main クラスの典型的な処理概略を示す．

APSController オブジェクト ac は，協調済 BPM の状態を変化させながら適宜 app 内の対応メソッドを起動する．またそれに伴って発生するメッセージの送信と受信を VLSession オブジェクトを介してメッセージング層に依頼する．

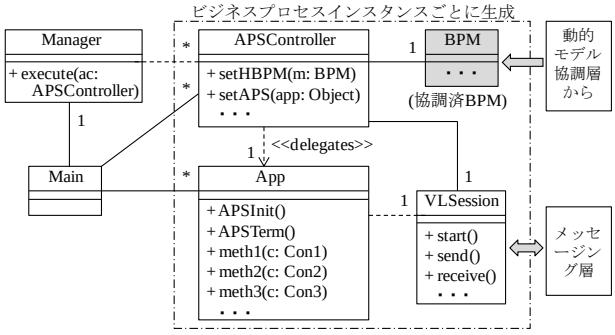


図 2：フレームワーク層の構成

```
. . .
dmh = new DMH();                    //(0)DMHオブジェクトを生成し，
dmhにURLなどを登録;                  //   各種セッテイングの後，
dmh.harmonize();                    //   モデル動的協調(DMH)を実行
. . .
m = new Manager();                  //(1)Managerオブジェクトを生成
while(1) {
  waitFor(BPインスタンス開始指示); //  BPインスタンスの開始を待つ
  ac = new APSComtroller();         //(2)APSControllerと
  app = new App(...);               //   アプリケーションをインスタンス化
  ac.setHBPM(dmh.getHBPM());        //(3)acに協調済BPMのコピーを登録
  ac.setAPS(app);                   //(4)acの委譲先としてappを登録
  m.execute(ac);                    //(5)BPインスタンスの実行を開始
}
```

図 3：Main クラスの典型的な処理概略

### 3.3 メッセージング層

HTTP 上でリライアブルなメッセージングを行うプロトコルエンジンである．ebXML メッセージング標準を簡略化したプロトコルを使い実装した．API は従来一般的であるメッセージチャネル共有では各システムの自律性が損なわれるため，VLSession と呼ばれる長期間持続可能あセッションを動的に生成する方式とした．下位システム構成は，両システムに Web サーバが存在する構成(図 4)と一方だけに存在する構成(図 5)の 2 つをサポートした[7].

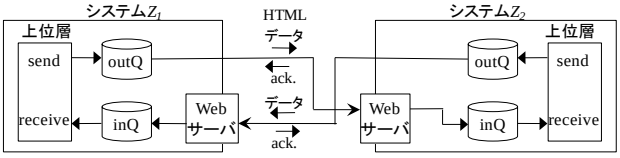


図 4：下位システム構成（対称型）

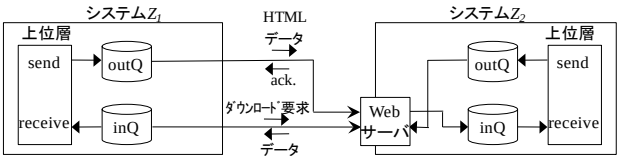


図 4：下位システム構成（非対称型）

## 4.まとめ

AWS の原理と実装を述べた．DMH アルゴリズムの妥当性は[1]以降様々な形で示されている．ミドルウェアの有効性についても[8]などで評価されている．本研究は科研費（21500110）の助成を受けたものである．

## 参考文献

[1]大谷,木下,嘉数: 自律的 Web サービスにおけるビジネスプロトコルの動的生成について, 電子情報通信学論文誌, vol.J87-D-I, no.8, pp.824-832 (2004).
[2]Oya, M. and Ito, M.: Dynamic Model Harmonization between Unknown eBusiness Systems, I3E 2005 IFIP, Springer, pp.389-403 (2005).
[3]Oya, M.: Autonomous Web Services Based on Dynamic Harmonization, I3E 2008 IFIP, Springer, pp.139-150 (2008).
[4]Oya, M., Ito, M. and Kimura, T.: Middleware for the Autonomous Web Services (AWS), I3E 2010, IFIP AICT341, Springer, pp.5-16 (2010).
[5]大友,大谷: 自律型 Web サービスにおける複数システム間での動的モデル協調, FIT2011 講演論文集第 4 分冊(査読付き), RO-012, pp.165-168 (2011).
[6]伊東,塚本,高木,木村,大谷: AWS ミドルウェアの研究 - アプローチと構成, 動的モデル協調層, アプリケーションフレームワーク層, 自律型メッセージング層-, 情報処理第 71 回全国大会, pp.1-509-516 (2009).
[7]木村,吉川,伊東,大谷: AWS メッセージング基盤改良の検討／非対称構成型メッセージング機能の実現, 情報処理学会第 72 回全国大会, pp.1-793-796, (2010).
[8]平本,木村,大友,大谷:実応用を想定した AWS ミドルウェアの評価, 情報処理第 73 回全国大会 pp.1-703-704 (2011).
[9]二宮,平本,安齋,大谷: AWS（自律型 WEB サービス）ミドルウェアフレームワーク制御, 情報処理第 73 回全国大会, pp.1-707-708 (2011).