RO-012

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

大友 浩照† 大谷 真‡
Hiroaki Otomo Makoto Oya

1. はじめに

自律型 Web サービス（Autonomous Web Services 以後 AWS）[4]では、異なるビジネスプロセスモデル（BPM）を持つシステム間で取引を行う際、動的モデル協調（Dynamic Model Harmonization 以後 DMH）を用いて互いのモデルを協調変形させることで Web 上での商取引を可能としている。AWS の根幹技術である DMH のアルゴリズムは事前研究[1][2]で提案され、[3]でプロトタイプの実装が行われ検証がなされている。また[5]において DMH の改良及び DMH のアルゴリズムの改良が提案されている。本論文は[5]の提案をさらに拡張し、３システム以上での DMH 及びそのアルゴリズムを提案する。

2. 自律型 Web サービス(AWS)

Web サービスは現在、Web 上のシステム間で行なう商取引の基盤技術として広く利用されている。しかし従来の Web サービスで商取引を行う場合、サービス全体に対して事前に取引モデル(その取引の手順等）が詳細に取り決められていなければならない。そして関連するシステムはその取引モデルに従って動作するように構築、運用されていなければならない。そのため、自由に開発されたシステムでは当該サービスに参加することができない。

これに対し AWS では、サービス全体での取引モデルの事前取り決めを不要とし、Web 上でシステムが遭遇した際、個々のシステムの取引モデルを動的に協調させ取引可能な形に変形する。そしてこの協調変形したモデルに従いアプリケーションを駆動させることで実際の取引を可能とする。こうして自由に開発され独自の取引モデルを持つシステム間でもより柔軟な商取引を行えるようにすることが AWS の最大の目的である。

3. 動的モデル協調（DMH）

各システムはビジネスプロセスモデル（BPM）を独自に定義することができる。BPM とは、そのシステムが処理可能なビジネスプロセスの流れを定義したものであり、前述での取引モデルに相当する。BPM の詳細は４節で述べる。図 1 に DMH の動作の概略を示す。

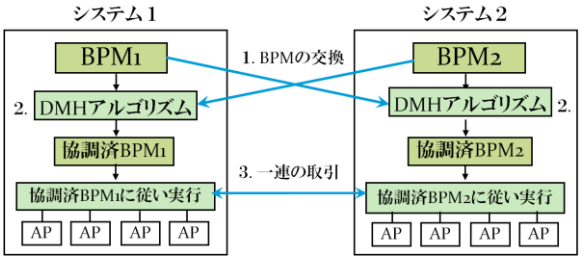

図 1 DMH

システムが遭遇した際、お互いの持つ BPM を交換し(図 1-1)、DMH アルゴリズムを用いて取引相手の BPM に合わせて自身の BPM を取引が可能な形に動的に変形する(図 1-2)。そして変形した BPM（協調済 BPM）に従いシステムはアプリケーションを駆動させ取引を実行する(図 1-3)。これにより異なる BPM を持つシステム間での商取引を可能としている。

4. ビジネスプロセスモデル（BPM）

AWS における BPM とは、取引における処理（オペレーション）の実行順序を定義したものである。BPM は次のように形式定義がなされている。

BPM＝(O,B)
O：オペレーション（o）の有限集合
o = (format, pattern)
format：メッセージのフォーマット
pattern：送受信パターン（”In” or “Out”）
B = (S, I, L, δ)
S：状態の有限集合
I ∈ S ：初期状態
L ⊆ S ：終了状態
δ: O×S→P(S) 状態関数(注 P(x)は x の冪集合)

B は O を入力とした非決定性オートマトンである。

図 2 に見積を依頼し、その結果を受け、発注を行う。または取引を中止する手順を表す BPM の例を示す。

† 湘南工科大学大学院工学研究科電気情報工学専攻
‡ 湘南工科大学

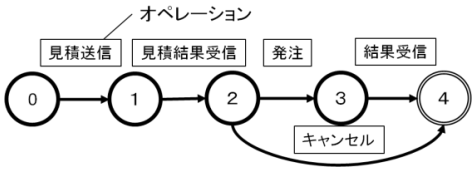

図 2　発注を行う BPM 例

BPM は状態遷移表の形で記述する方法と正規表現による記述方法との２種類の表現定義がなされており、AWS ではオートマトンの範囲に限定し、XML を用いて BPM を記述することができる。

5.　従来の DMH アルゴリズム

各システム P_1, P_2 をそれぞれ P_1={BPM_1}, P_2={BPM_2} とする。(P_1 = 自社のシステム、P_2=取引相手のシステム)

DMH アルゴリズムの定義は以下のとおり。

DMH(BPM_1, BPM_2) → 協調済 BPM

引数１に自社の BPM_1、引数２に取引相手の BPM_2 を指定し、協調済 BPM が出力される。

DMH アルゴリズムを以下に示す。

① 全システムで統一的な UBPM を生成する。
UBPM＝(Ou,Bu)
Bu = (Su, Iu, Lu, δu)
Bu は Ou を入力とした非決定性オートマトン。

② 遷移オペレーション集合 Ou :
$Ou = \{(o_1, o_2) \mid o_1 \in O_1 \ \wedge \ o_2 \in O_2 \ \wedge \ Match(o_1, o_2) \neq False\}$
Match()は整合性の検査をする関数であり整合性があると判定できれば True,整合性がないと判定できれば False,どちらとも判定できなければ Unknown を返す。詳細は省略する([6]参照)。

③ 状態の集合 Su :　$Su = S_1 \times S_2$

④ 初期状態 Iu :　$Iu = I_1 \times I_2$

⑤ 終了状態 Lu :　$Lu = L_1 \times L_2$

⑥ 遷移関数 δu :　$\delta u(o_1, o_2),(s_1,s_2)) \rightarrow (\delta(o_1,s_1) \times \delta(o_1,s_1) \mid o_1 \in O_1 \ \wedge \ o_2 \in O_2 \ \wedge \ s_1 \in S_1 \ \wedge \ s_2 \in S_2)$

⑦ 到達不能な状態、遷移を削除する。

⑧ UBPM の射影として協調済 BPM(HBPM)を生成する。
HBPM = (Oh, (Sh, Ih, Lh, δh))
$Oh = \{ o \mid (o_1, o_2) \in Ou \rightarrow o1 \}$,
$Sh = \{ s \mid (s_1, s_2) \in Su \}$ $Ih = I_1$,
$Lh = \{ s \mid (s_1, s_2) \in Lu \}$
$\delta h(o, s) = \cup^X \cup^Y \ \delta u((o_1, o_2), (s_1, s_2))$
[X : $(s_1,s_2,\ldots,s_n) \in Su$ を満たす全ての s, Y : $(o_i, o_j) \in Ou$ を満たす全ての o]

⑨ 重複する遷移をまとめ不要な状態を削除する。

図 3 は第４節、**図 2** の BPM_1 と、見積を受信し、見積結果を送信、その後受注を行う BPM_2 に対しシステム P_1 が DMH アルゴリズムを適用した例である。オペレーションの下に書いてあるのは送受信パターンである。また(見積送信,見積受信),(見積結果受信,見積結果送信),(発注,受注),(結果受信,結果送信)はそれぞれ Match()=True となるオペレーションであり他の組合せは全て False とする。

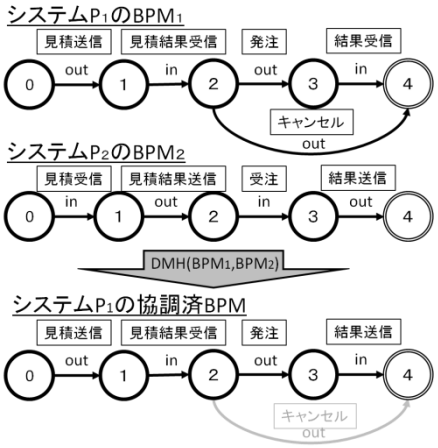

図 3 DMH の適用例

適用した結果、BPM_1 からオペレーション「キャンセル」が無くなり BPM_2 に合わせ縮退されている。このように DMH アルゴリズムは自身の BPM を相手の BPM に合わせ縮退するものである。

6.　３システム以上を用いた取引

現在の DMH では 1 対 1 での取引しか想定していない。しかし、実際の取引を行う際には複数のシステムが連携し取引を行うことも考えられる。[5]ではこの解決案として複数のシステムが連携して動作できるような DMH、及びそのアルゴリズムの提案がなされている。しかし、この提案は 3 社間での取引のみに限定したものであった。そこで第 7 節ではこれを拡張し、３社以上での取引に対応した DMH を示す。また第８節では３つ以上の BPM を協調変換することのできる DMH アルゴリズムを示す。

7.　３システム以上での取引を想定した DMH

システム P_1～P_n 間での DMH の動作を以下に定義する。

① あるシステムが Web 上で遭遇した際、取引に参加する全てのシステムと BPM を交換する（図 4-1)。

② DMH アルゴリズムを用いて自身の BPM を取引が可能な形に動的に変形する（図 4-2)。

③ 変形した BPM（協調済 BPM）に従いシステムは取

引の手順を進める（図 4-3）。

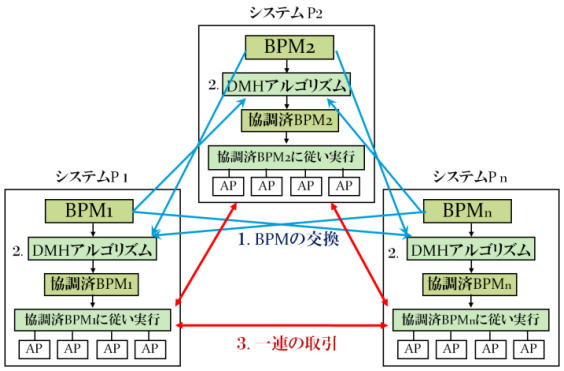

図 4 複数システムでの DMH

8. 3 つ以上の BPM を用いた DMH アルゴリズム

第 6 節を実現するにあたり、複数の BPM を用いたアルゴリズムを新たに定義する必要がある。従来の DMH アルゴリズムの定義を以下のように変更する。

各システム P_1～P_n をそれぞれ P_1={BPM_1}, P_2={BPM_2},.., P_n={BPM_n}とする。(n=システム数, P_1 = 自社のシステム、P_2～P_n =取引相手のシステム)

DMH アルゴリズムの定義を以下に変更する。

DMH(BPM_1, BPM_2,..,BPM_n) → 協調済 BPM

引数１に自社の BPM、引数２以降に取引相手の BPM を指定し、協調済 BPM が出力される。

また本提案では,メッセージのやり取りを行うシステムが複数存在している。そのため従来のオペレーションによる遷移ではそのオペレーションがどのシステムに対してメッセージのやり取りを行うものなのかの情報を持っていない。そこで BPM の定義を拡張し、協調後の BPM の各オペレーションに対して対になる相手のシステムの識別 ID を持たせることとする。BPM のオペレーションの定義を以下に変更する。

o = (format, pattern, systemId)
format：メッセージのフォーマット
pattern：送受信パターン（”In” or “Out”）
systemId = { システムの識別 ID }

以上を踏まえて DMH アルゴリズムを以下に示す。

① 全システムで統一的な UBPM を生成する。
UBPM＝(Ou,Bu)
Bu = (Su, Iu, Lu, δu)
Bu は Ou を入力とした非決定性オートマトン。

② 遷移オペレーション集合 Ou：
$Ou = \{(o_i, o_j) \mid o_i \in O_i \ \wedge \ o_j \in O_j \ \wedge \ Match(o_i, o_j) \neq False\} \quad (1 \leqq i,j \leqq n \ \wedge \ i \neq j)$

Match()は整合性の検査をする関数であり整合性があると判定できれば True,整合性がないと判定できれば False,どちらとも判定できなければ Unknown を返す。詳細は省略する([6]参照)。

③ 状態の集合 Su： $Su = S_1 \times S_2 \times \ldots \times S_n$

④ 初期状態 Iu： $Iu = I_1 \times I_2 \times \ldots \times I_n$

⑤ 終了状態 Lu： $Lu = \Pi(L_i \cup I_i) - \Sigma(L_i \cap \Pi I_j) \quad (1 \leqq i \leqq n, \ 1 \leqq j \leqq n \ \wedge \ i \neq j)$

⑥ 遷移関数 δu：$\delta u((o_i, o_j),(s_1,s_2,\ldots,s_n)) \rightarrow$
$(\delta(o_i,s_i) \times \delta(o_j,s_j) \times \delta(\varepsilon,s_m) \times, \ldots, \times \delta(\varepsilon,s_k) \mid o_i \in O_i \ \wedge \ o_j \in O_j \ \wedge \ s_i \in S_i \ \wedge \ s_j \in S_j \wedge \ s_m \in S_m \ \wedge \ s_k \in S_k) \quad (1 \leqq i,j,k,m \leqq n \ \wedge \ m \neq k \neq i \neq j)$

ここで $\delta(\varepsilon,s_x)$は状態 s_x から s_x への遷移を意味するものとする。（つまり状態を遷移させない）

⑦ 到達不能な状態、遷移を削除する。

⑧ UBPM の写像として協調済 BPM(HBPM)を生成する。
HBPM ＝（Oh, Bh）
$Oh = \{ o \mid (o_1, o_k) \in Ou \ \rightarrow o_1 \}$
Bh = (Sh, Ih, Lh, δh)
$Sh = \{ s \mid (s_1, s_2, \ldots, s_n) \in Su \}$
$Ih = I_1$, $Lh = \{ s \mid (s_1, s_2, \ldots, s_n) \in Lu \ \wedge \ s_1 \in L_i \}$
$\delta h(o, s) = \cup^X \cup^Y \ \delta u ((o_i, o_j), (s_1,s_2,\ldots,s_n))$
$\mid \vee o \neq (o1, ok) \rightarrow o = \varepsilon \ (1 \leqq i,j,k \leqq n \wedge i \neq j \neq k)$
[X : $(s_1,s_2,\ldots,s_n) \in Su$ を満たす全ての s, Y : $(o_i, o_j) \in Ou$ を満たす全ての o]

⑨ 重複する遷移をまとめ不要な状態を削除する。

8.1. 考え方

①は利用する BPM 全てを使った取引全体の振舞いを一時的に求めるため UBPM を定義しており。②～⑥は UBPM の値を実際に求める際の定義である。②では整合性のあるオペレーションの組を求める。③、④では各 BPM の S、I の直積を求める。⑤では、初期状態と終了状態の全組合せからどこか一箇所だけが終了状態かつ残りが初期状態の組合せを除外している。これにより全システムが同時に終了状態又はあるシステム２つ以上が終了状態で残りが開始状態である状態を終了状態 Lu としている。⑥では①で求めた Ou の要素から対応した状態への遷移を定義している。これらの操作で得られるオートマトンは各 BPM の終了状態 L に対し L=L∩I としたときの積オートマトンから一定の条件で抽出された部分オートマトンである。

⑧により射影をとることで自身の振舞いだけを抽出し BPM の縮退を行っている。つまり各システムの協調結果は他のシステムの協調結果に依存しない。また、このアルゴリズムでは２つで１組のオペレーションにより遷移する UBPM を構築しているため１度の遷移で２つ以上のシステムの状

態は変化しない。しかし本来の取引の動作としては、例として P_1 と P_2 がやり取りを行う際、同時に P_3 と P_4 もやり取りを行うことが考えられる。しかし、このような操作は P_1 と P_2 がやり取りを行った後 P_3 と P_4 がやり取りを行った場合と同じ状態に遷移することになる。そのため⑨で重複する遷移としてまとめられる。その結果本論文で定義したアルゴリズムと最終的な出力は変わらない。

8.2. 適用例

発注を行うシステム P_1(小売業者)、入荷拠点 P_2(小売店舗)、出荷拠点 P_3(配送センター)、受注システム P_4(卸売業者)とし、 P_1 は商品の注文を任意回行った後、商品の入荷予定を送信する。その後入荷完了を確認したら支払い情報の通知を行う。P_2 は入荷予定を受信し、実際に商品が届いた後に商品を受け取ったことを通知。その後入荷確定の通知を行う。P_3 は出荷予定を受信し商品を発送、実際に商品が届いたことを確認後出荷確定の通知を行う。P_4 は注文を受けた後、商品の出荷予定を送信する。その後出荷完了を確認したら支払い情報の通知を受信する。以上のようなシステム(図 5)を用いて次の条件下での DMH アルゴリズムのシステム P_1 での適用例を図 6 に示す。

条件 : (注文,受注),(入荷予定通知, 入荷予定受信),(出荷予定通知, 出荷予定受信),(入荷確定,出荷確定) ,(入荷完了通知,出荷受領) ,(出荷完了通知,入荷受領), (支払確定,支払受領)の組合せのみが True。他の組合せは全て False とする。

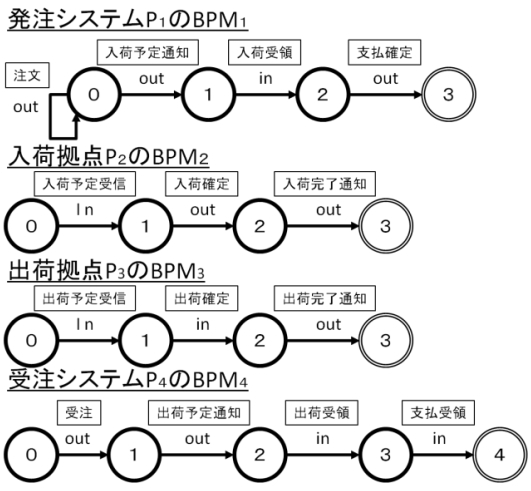

図 5　P_1～P_4 の BPM

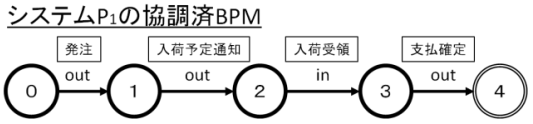

図 6　DMH 適用結果

DMH アルゴリズムを適用した結果、想定していた協調済 BPM を出力することができている。P_2～P_4 では協調後も手順が変わらないため省略しているが同じように協調できる事が確認できている。また、協調済後のオペレーションが対応するシステムとメッセージのやり取りを行うことで取引の手順を進めることが可能であることも確認できる。

9. まとめ

- 本論文ではまず第 7 節で 3 社以上の複数のシステムが連携し動作できるような DMH の動作を定義した。
- 次に第 8 節で前述の DMH を実現するために 3 つ以上の BPM を用いた DMH アルゴリズムを定義した。

以上より AWS の根幹技術である DMH の改良を行なった。今後の課題として 3 つ以上のシステムが連携し取引を行う事例を、実例から検証し、本提案の妥当性を示す必要がある。現在、実応用を想定したテストケースを作成し、AWS の評価を行っている最中である。また本提案では、従来とは違い複数のシステムを用いるためメッセージのやり取り相手が唯一ではない。そのため BPM 定義やメッセージングなど実装方式の一部変更が必要である。

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] M.Oya, and Ito: "Dynamic Model Harmonization between Unknown eBusiness Systems", IFIP I3E, Springer ISBN:0-387-28753-1, pp. 389-403, 2005

[3] M.Oya.: "Autonomous Web Services Based on Dynamic Harmonization", IFIP I3E, Springer, ISBN:978-0-387-8590-2, pp.139-150, Sep, 2008

[4] M.Oya et al.: "Middleware for the Autonomous Web Services (AWS)", IFIP I3E,Software Services for e-World, Springer, pp.5-16, Nov., 2010.

[5] 大友, 安斎,他,AWS ミドルウェア:動的モデル協調の改良 情報処理学会第 73 回全国大会, pp.1-705-706, 2011.

[6] 安斎,大友,他,AWS ミドルウェアにおける取引様式の互換性決定方式の研究, 情報処理学会第 73 回全国大会, pp.1-711-712, 2011.