1D4-01

セマンティクス導入によるユーザ指向 Web サービス連携支援

User-Oriented Web Service Composition with Semantics

杉本 悠樹 Yuki Sugimoto　村上 陽平 Yohei Murakami　荒井幸代 Sachiyo Arai　石田亨 Toru Ishida

京都大学大学院情報学研究科社会情報学専攻
Department of Social Informatics, Kyoto University

Current Web Service composition technologies can not meet to satisfy a great variety of user's requests because of their server-oriented specification. In this paper, we propose the user-oriented architecture that realizes a flexible and reusable Web Service composition. It consists of two aspects: applying '*state transition description; scenario*' to reflect user's request as s/he likes, and *'Semantic Web service description'* to make the created scenario understandable documents for the agents. The former makes a service description more intuitive than a workflow, especially for representing the concurrent processes. The latter enables agents to find the suitable service compositions and compose them flexibly for each user's situation.

1. はじめに

Web 上に散在するアプリケーションを活用するための Web サービス技術の進歩が近年目覚ましく，企業を中心として標準化が進められている．これら技術により，Web サービスの公開，利用が可能となり，複数の Web サービスを連携させて複雑なサービスを作成するための言語仕様の開発が現在進んでいる．

しかしながら，このような Web サービスを連携させるための技術は，既存のサービスをコンポーネントとして利用することで新たなサービスを低コストで作成するといったサービスプロバイダ側の効率だけを重視しており，ユーザの視点に立ったものではない．Web 上に分散したサービスを自由に組み合わせ，多様なサービスを作り出すという Web サービスの趣旨を考慮すれば，ユーザ指向のサービス作成を支援する Web サービス連携手法もまた重要である．

そこで本論文では，ユーザ指向の Web サービス連携を実現するために以下の課題に取り組む．

1)　サービス連携プロセスの記述支援

ユーザが自らの状況に合わせたサービス連携を直感的に記述できるような容易な記述形式と，その記述に従ってサービスを起動するエージェントが必要である．また，類似した連携プロセスをパターン化することで，利用する Web サービスを選択するだけで連携プロセスを記述できる仕組みを開発する．

2)　サービス連携プロセスの形式化

サービスプロバイダがユーザの作成した連携プロセスの事例を再利用し，他のユーザの状況に合った連携サービスを自動合成できるように，連携プロセスの記述をエージェントの解釈できる形式に変換する技術を開発する．

以下本論文では，我々のアプローチの概要について述べる．次に，そのアプローチを実現するために用いたインタラクション設計言語 *Q*[Ishida 2002]と，我々が開発した Web サービス連携カード，Web サービス連携フォーマライザについて，それぞれ説明する．

2. セマンティクス導入による Web サービス連携

本章では，Web サービスを支える基盤技術，および，Web サービス連携に関する既存技術を述べた上で，我々のアプローチの概要を説明する．

2.1 既存の Web サービス連携技術

Web サービスの利用は，目的に合ったサービスの発見，サービス利用のためのインタフェース情報の取得，サービスとの通信の 3 段階から成り，それぞれ，Web サービスのディレクトリサービスである UDDI (Universal Description, Discovery and Integration)，Web サービスのインタフェース記述言語である WSDL (Web Service Description Language)，そして通信プロトコルである SOAP (Simple Object Access Protocol)によって支援される．

これらの Web サービス基盤技術を用いた Web サービスを連携させるための言語として，Business Process Execution Language for Web Services (BPEL4WS) や Web Service Choreography Interface (WSCI)など，XML をベースとしたワークフロー記述言語が一般的である．

BPEL4WS では，Web サービス呼び出しの実行，データの操作，障害の通知，またはプロセスの終了などのさまざまなアクティビティを作成し，それらを結び合わせることによって，複雑なプロセスを作成することができる．しかし，サービス連携中の各サービスは静的にバインドされているため，要求に応じて動的に Web サービスを組替えることはできない．[Sirin 02]

WSCI では，Web サービスの内部プロセスの定義や実装について言及せず，サービス間で交換されるメッセージのフローを記述することで，複数の Web サービスの連携プロセスを記述できる．

2.2 ユーザ指向の Web サービス連携

本論文では「連携プロセスの記述支援」，「連携プロセスの形式化」という二段階により，ユーザ指向の Web サービス連携を実現する．

まず，「連携プロセスの記述支援」の段階では，インタラクションのプロトコルを記述する際に最も一般的である状態遷移表現に基づいて，ユーザ自身に Web サービスの連携プロセスを記述させることを試みる．ここで記述された連携プロセスは，ユーザと Web サービスの間を仲介するエージェントに与えられ，エージェントはその記述に従ってユーザの代わりに複数の Web サービスを利用する．なお，一般に，インタラクションのプロトコルの記述には状態遷移表現が適しているといわれ，これを採用することで，ユーザは自らの状況を考慮した連携サービスの構築が容易になる[Kawamura 2000]．このアプローチの具体的な

連絡先：〒606-8501 京都府京都市左京区吉田本町 京都大学大学院情報学研究科 石田研究室，075-753-5396

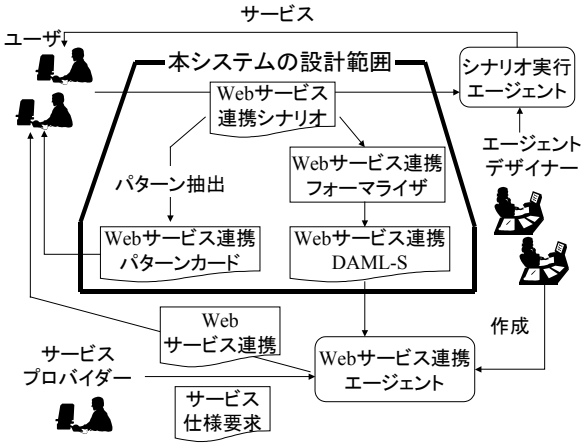

図 1: システム概略

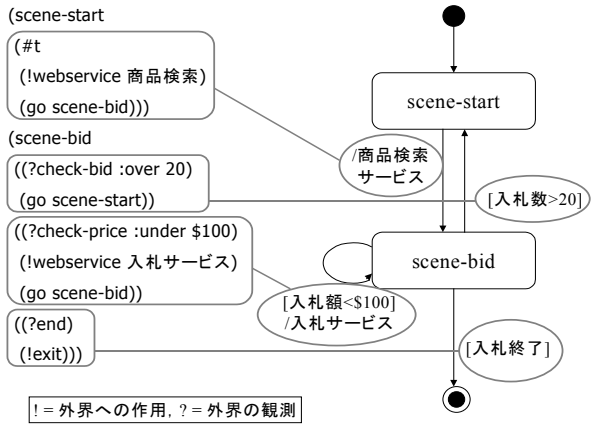

図 2: *Q* シナリオ

方法として，エージェントに仕事を依頼するための言語であるシナリオ記述言語 *Q* を用いる．

さらに，記述された複数の連携プロセスからインタラクションのパターンを抽出し，パターンカードとして整理することで，ユーザに複雑なプロセス間のインタラクションを意識させることなく，利用する Web サービスの選択だけで連携プロセスの記述が可能になる．

次に「連携プロセスの形式化」の段階では，サービスプロバイダのエージェントが，さまざまなユーザの状況に合った連携サービスを自動合成できるように，ユーザの作成した連携プロセスの事例を収集し形式化を試みる．これ対しては Web サービスに関連するセマンティック Web[TBL 2002]の技術を導入し，ユーザの記述した連携プロセスをエージェントに理解できる形式に変換する．具体的には，第1段階でユーザが記述した連携プロセスを，記述論理表現に基づいたプロセス記述言語である DAML-S[Ankolokar 2001]のプロセスモデルへと変換する．この変換システムをフォーマライザと呼ぶ．

以上を実現するため設計したシステムを図 1 に示す．本システムは，図 1 の太線部分を実現する．

3. サービス連携記述のためのユーザ支援

本論文では，ユーザ指向の Web サービスの連携プロセスを作成する目的として，連携プロセスの記述法と，その記述からパターンを抽出し，これを参照するための手法，そして連携プロセスの記述を Web サービス連携エージェントの入力用に形式化する手法の三つを提案する．一つ目の，ユーザによる Web サービスの連携プロセスの記述法として，既存の Web サービス記述言語のようなワークフローではなく，状態遷移を用いる．ここでは，シナリオ記述言語 *Q* を用いて状態遷移を記述することによって，サービスの実行主体であるエージェントの動作を，より直感的に表現することを可能にする．一方，二つ目に対しては，作成されたサービス連携プロセスから Web サービスの連携パターンを抽出し，そのパターンの参照を可能にするために，IPC(Interaction Pattern Card)を導入し Web サービス連携カードを作成する．

3.1 シナリオ記述言語

シナリオ記述言語 *Q* は，エージェントの動作を規定するシナリオを記述する言語である．シナリオはエージェントの内部状態の設計ではなく，エージェントが外界とどのようにインタラクションするかを規定するものである． 図 2 にシナリオの例を示す．これは，複数の Web サービスを用いてオークションの入札を行うシナリオの例である．以下に *Q* 言語の特徴を示す．

(1) シナリオ

シナリオは，関数とは異なり，状態遷移表現を記述する．図 2 の左側がシナリオ記述の例で，図 2 右の状態遷移図に対応づけられる．このシナリオは，Web サービスを用いたオークションでの入札サービスを表わしている．シナリオは，ガード付コマンドである複数の状態から成り立っており，次状態への遷移を表わす go 文を組み合わせることで状態遷移表現を実現する．

(2) キュー・アクション

キューは“?”で始まる文であり，エージェントへの外界観測の依頼をあらわす．”!”から始まるアクションは外界への作用の依頼である．キュー，アクション共に命令や関数と違い，エージェントへの依頼を表わしており，これらの解釈実行はエージェントに任されている．それゆえシナリオには実行内容の詳細を規定する必要がなく抽象度が高いため，ユーザにとって読みやすいものとなる．本論文の実装では，Web サービスの実行は”!webservice”文にパラメータとして WSDL ファイルの URI，サービスに対する操作(operation) ，入力値を指定することで実行される．

(3) ガード付コマンド

ガード条件による遷移を実現するのが，ガード付コマンドである．ガード付コマンドは，キューから始まる複数の節からなる．ガード付コマンドが実行されると，エージェントは各節の最初にあるキューを同時観測し，いずれかが発火されるとその節の後続するキュー，アクション群が実行される．ガード付コマンドは”guard”という記述から始まるが，状態として用いられる場合は，状態名によって代用することが出来る．

シナリオ記述による Web サービス連携記述支援では，Web サービス実行というアクションがあり，そのパラメータとしてどの Web サービスをどのように実行するかを指定した．すなわち，この設計は，ユーザが WSDL を理解できることを前提とし，Web サービス連携記述を支援するものとなっている．これは，一つのアクションでさまざまな Web サービスの起動が可能となるよう，汎用性のある設計にしたためである．汎用性よりも記述の容易さを優先するのであれば，エージェントデザイナーがエージェン

カードID	1	カード名	ユーザ入力	カード定義名	Webサービス連携カード	エージェント名	ユーザ入力エージェント
Webサービス連携実行（繰り返し）	メッセージ受信	送信元		メッセージ名		アクション	
		ユーザ		住所情報			
		ユーザ		終了通知		（繰り返し終了）	
	Webサービス実行	WSDL		操作		入力	返信メッセージ名
	メッセージ送信	送信先				メッセージ	
		サービス実行エージェント				住所情報	
Webサービス連携終了	メッセージ送信	送信先				メッセージ	
		サービス実行エージェント				終了通知	

カードID	2	カード名	サービス実行	カード定義名	Webサービス連携カード	エージェント名	サービス実行エージェント
Webサービス連携実行（繰り返し）	メッセージ受信	送信元		メッセージ名		アクション	
		ユーザ入力エージェント		住所情報			
		ユーザ入力エージェント		終了通知		（繰り返し終了）	
	Webサービス実行	WSDL		操作		入力	返信メッセージ名
		http://webservices.eraserver.net/zipcoderesolver.wsdl		ShortZipCode		住所情報	ZIPコード
		http://www.xmethods.net/TemperatureService.wsdl		getTemp		ZIPコード	気温
	メッセージ送信	送信先				メッセージ	
		結果出力エージェント				住所情報 ZIPコード 気温	
Webサービス連携終了	メッセージ送信	送信先				メッセージ	
		結果出力エージェント				終了通知	

カードID	3	カード名	ユーザ入力	カード定義名	Webサービス連携カード	エージェント名	結果表示エージェント
Webサービス連携実行（繰り返し）	メッセージ受信	送信元		メッセージ名		アクション	
		サービス実行エージェント		住所情報 ZIPコード 気温			
		サービス実行エージェント		終了通知		（繰り返し終了）	
	Webサービス実行	WSDL		操作		入力	返信メッセージ名
	メッセージ送信	送信先				メッセージ	
		ユーザ				住所情報 ZIPコード 気温	
Webサービス連携終了	メッセージ送信	送信先				メッセージ	

```
(defscenario card2
 (&pattern ($addressinfo '())
           ($zipcode '())
           ($temperature '()))
 (scene-execution
  ((?receive :from ユーザ入力エージェント
             :message $addressinfo)
   (!webservice :wsdl
"http://webservices.net/zipcoderesolver.wsdl"
             :operation "ShortZipCodeResult“
             :input $addressinfo
             :result $zipcode)
   (!webservice :wsdl
“http://www.xmethods.net/TempService.wsdl”
             :operation “getTemp“
             :input $zipcode
             :result $temperature)
   (!send :to 結果表示エージェント
          :message ' ($addressinfo
                      $zipcode
                      $temperature))
   (go scene-execution))
  ((?receive :from ユーザ入力エージェント
             :message 'finish-scenario)
   (go scene-finish)))
 (scene-finish
  (#t
   (!send :to 結果表示エージェント
          :message 'finish-scenario))))
```

図 3: IPC カード(左)とカード 2 に対応するシナリオ

トのアクションまたはキューを特定の Web サービスにバインドすることで，ユーザは WSDL を意識することなく，シナリオを記述することができる．図 2 の例では，”!webservice 入札サービス”として呼び出していた入札サービスを，”!bid”というアクションにバインドさせることで，ユーザは WSDL を意識する必要がなくなる．

3.2 Web サービス連携カード

作成したシナリオから Web サービス連携時におけるエージェントの振る舞いのパターンを抽出し，他のユーザが容易に再利用あるいは参照するための機能として，Web サービス連携というドメインに限定した Web サービス連携カードを IPC (Interaction Pattern Card)により提供する．IPC とは，Q で記述したシナリオから抽出されたエージェントの振る舞いのパターンをカード形式で表現したものである．図 3 に Excel を用いて表示した IPC の例を示す．カードの各行各列には，当該ドメインにおけるエージェントと外界とのインタラクションのパターンが反映されている．各行はエージェントの状態，あるいは，アクションの種類を表わしており，各列は，各状態におけるアクションないしはアクションに与えるパラメータを指定する．

IPC コンバータによって IPC から Q のシナリオに変換することが可能であり，自分で作成した IPC，あるいは公開されている IPC のカード構造を利用してカードを完成させ，エージェントに実行させることが可能である．ある一つの Web サービスが利用不可能となった場合，代替サービスを見つけてシナリオを書き換えねばならないが，IPC カードとして整理されていれば，サービスの置き換えもスムーズに行うことが出来る．

図 3 左で示されているカードは Web サービス連携カードの例であり，複数のエージェントが協調して Web サービスを連携する際のインタラクションのパターンを表している．このパターンは，繰り返し行われる「Web サービス連携の実行状態」と「Web サービス連携の終了状態」の二状態から構成される．「Web サービス連携の実行状態」ではまず，他のエージェントからのメッセージを待ち受ける．メッセージ着信が確認出来次第，指定された Web サービスとその他のアクションを実行した後，別のエージェントにメッセージを送信し，再びメッセージ待ち受け状態に戻る．一方，「Web サービス連携の終了状態」では他のエージェントにメッセージを送信する．このようなカードを複数作成することで，マルチエージェント環境での Web サービス連携を構築することが出来る．図 3 では三体のエージェントにカードがそれぞれ割り当てられ，パイプライン処理によって Web サービス連携を実現しており，図 3 右は中央のカードに対応するシナリオを示している．

図 3 は，ユーザの入力した住所からその地域の ZIP コードと気温を導く複合サービスを， Web サービス連携カードを用いて表現したものである．各カードはユーザ入力エージェント，Web サービス実行エージェント，結果表示エージェントに割り当てられ，各エージェントはメッセージパッシングによるパイプライン処理を行っている．カード ID2 の例では，ユーザ入力エージェントから住所情報を受信すると次の「Web サービス実行」に処理が移る．「Web サービス実行」では，指定されたパラメータに従って複数の Web サービスが逐次実行される．カード 2 はひとつめのサービスにより住所情報から ZIP コードを獲得し，その ZIP コードを用いて，ふたつめのサービスから気温を得る．続いてエージェントは「メッセージ送信」に処理を移し，他のエージェントもしくはユーザにメッセージを送信し，繰り返しの始めに戻る．一方，「メッセージ受信」で終了通知を受信した場合は，繰り返し終了のアクションにより，Web サービス連携終了状態に遷移する． Web サービス連携終了状態では，終了通知を他のエージェントに送信し，処理を終了する．

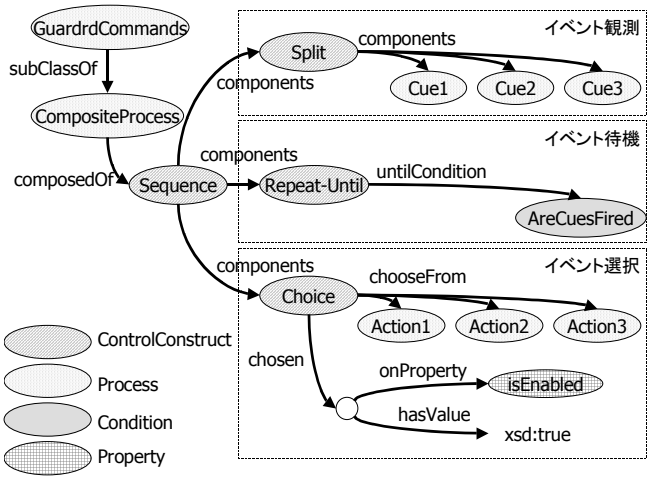

図 5: ガード付コマンド変換規則

3.3 Web サービス連携フォーマライザ

シナリオ記述及び Web サービス連携カードは Web サービス連携を作成しようとするユーザを個別に支援することで，ユーザ指向のサービス連携の実現を容易にするためのものである．一方，ユーザがそれぞれ独立に，毎回自分の状況に合った連携プロセスを記述することは，重複する連携プロセスを別々に記述する可能性が考えられ，極めて非効率である[Benetallah 2002]．そこで，さまざまな状況のユーザのニーズを反映した *Q* によるシナリオ記述を DAML-S へと変換してセマンティクスを付加することを考える．これによって，サービスプロバイダ側に Web サービスを動的に連携する仕組みを提供し[McIlraith 2001]，シナリオの再利用が可能になる．本節では，Web サービス連携において記述が困難であるとされる同時実行プロセスの実行シナリオを DAML-S のサービスモデルへ変換する方法を例としてとりあげて説明する．シナリオにおいて同時実行プロセスは，「ガード付コマンド」を用いて記述されている．ガード付コマンドは大きく分けて，以下のようなステップで逐次実行制御される．

1. 並行実行(Split)クラスを用いて，複数のイベント待ち(Cue)クラスの並行実行を開始する (イベント観測実行プロセス部)

2. 終了条件付き繰り返し(Repeat-Until)クラスを用いて，観測しているイベントのうち，少なくとも一つのイベントが発生するまで待機する (イベント待機プロセス部)

3. 選択実行(Choice)クラスを用いて，発生したイベントのうち一つを選択し，そのイベントをトリガとするアクション群プロセス(ActionSet)クラスを実行する(イベント選択実行プロセス部)

そこで，DAML-S で提供されている逐次実行(Sequence)クラスという制御構造クラスを用いて，イベント観測実行プロセス部，イベント待機プロセス部，そしてイベント選択実行プロセス部の逐次実行を行うコンポジットプロセス(CompositeProcess)クラスのサブクラスとしてガード付コマンドを変換する．図5に，イベント観測実行プロセス部，イベント待機プロセス部，そしてイベント選択実行プロセス部によって構成したガード付きコマンドの DAML-S 表現を，RDF の有向グラフで示す．

4. おわりに

本論文では，ユーザによる Web サービスの連携プロセス記述と，サービスプロバイダによる連携プロセスの自動合成というふたつの側面から，ユーザ指向のサービス連携支援を実現した．以下に本研究の貢献，および，今後の課題をまとめる．

1) Web サービス連携カードの開発

Web サービス連携を実行するエージェントの動作を，高い抽象度で，直感的に記述できる方法を提供した．また，サービス連携記述から Web サービス連携を行うエージェントの動作のパターンを抽出することで，代替サービスへの置き換えや，類似した Web サービス連携プロセスの記述を容易にした．

2) Web サービス連携フォーマライザの開発

ユーザが作成した Web サービス連携プロセスを Semantic Web 技術のひとつである DAML-S に変換する技術を開発した．これによって，サービスの連携プロセスにセマンティクスが付加され，エージェントに理解可能なものとなる．この結果，エージェントがこれらの連携プロセスを再利用して，他のユーザの状況に合わせた Web サービスを連携することを可能にした．

今後の課題として，Web サービス連携シナリオから Web サービス連携カードを生成する技術の開発，および，フォーマライザによって生成された DAML-S 記述を用いて Web サービスを自動的に連携させるエージェントの開発のふたつが挙げられる．現在 Web サービス連携カードから IPC コンバータによってシナリオを生成するコンバータはあるものの，複数のシナリオからインタラクションのパターンを自動的に抽出することは出来ない．また，セマンティクスの導入によってエージェント可読にした後のサービスの自動連携手法，さらに，エージェントにより生成された連携サービス全体のトランザクション処理についても考える必要がある．これらによって，より高度なユーザ指向の Web サービス連携の実現が期待できる．

参考文献

[Ishida 2002] Toru Ishida: Q: A Scenario Description Language for Interactive Agents. IEEE Computer, Vol.35, No. 10, 2002.

[Ankolokar 2001] A. Ankolokar et.al., DAML-S: Semantic markup for web services. *In Int. Semantic Web Working Symposium*, pages 411-430, 2001

[McIlraith 2001] Sheila A. McIlraith, Tran Cao Son, Honglei Zeng: Semantic Web Services. In IEEE Intelligent Systems (Special Issue on the Semantic Web), March/April 2001.

[TBL 2002] T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web. Scientific American, 284(5): pages 34-43, 2001.

[Sirin 2002] Evren Sirin, James Hendler, Bijan Parsia, "Semi-automatic Composition of Web Services using Semantic Descriptions." Accepted to "Web Services: Modeling, Architecture and Infrastructure" workshop in conjunction with ICEIS2003, 2002.

[Benetallah 2002] B. Benatallah, M. Dumas, M.-C. Fauvet, and F. Rabhi. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi, editors, Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, UK, Nov 2002.

[Kawamura 2000] T. Kawamura, T. Hasegawa, A. Ohsuga, S. Honiden, "Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems", Systems and Computers in Japan, John Wiley & Sons, Inc., Vol. 31, No. 13, pp. 42-56, 2000.