Webサービス連携のためのフロー記述方式

†東 真樹 †小泉 寿男 †佐藤 智哉
† 東京電機大学大学院理工学研究科

現在, Webサービスを用いたアプリケーションの開発において複数のWebサービスを連携することにより，新たなアプリケーションを構築する様々な手法が提案されている．その一つにBPEL4WSのように，フロー記述言語とそのプロセス文章を解釈・実行を行えるエンジンによってWebサービスの連携を実現させる方法がある．本稿では，スクリプト言語からBPELプロセス文書と生成される複合WebサービスのWSDLを生成できるシステムを提案することにより，BPELベースの複合Webサービスを用いたアプリケーション開発の支援を狙いとする．

The flow description system for Web Service cooperation

†Masaki Higashi †Hisao Koizumi ‡Tomoya Sato
† Department of Computers and Systems Engineering, Graduate School of Tokyo Denki University ‡Mitsubishi Electric Corporation

An information system architecture, which builds new applications by the integration and cooperation of multiple Web services, is now proposed in the field of application development based on Web services. In the Web services technologies, there is a method of realizing cooperation of a Web services with the flow description language like BPEL4WS, and the method can perform interpretation and execution for the process text. This paper proposes an application development system using the compound Web services based on BPEL which generate process document and WSDL of the compound Web services from a script language.

1．はじめに

近年のネットワーク技術の進歩により，分散環境上で提供されるアプリケーション（サービス）を相互連携し，新たなシステムの構築を低コスト，短期間で実現しようという考えのもとWebサービスが登場した．Webサービスは，標準化されたXMLメッセージを使用してネットワーク経由でアクセス可能にし，相互運用性を保つようになっている．その複数のWebサービスを連携し後悔したものが複合 Web サービスである[1]．Webサービスは，サービス情報を共通的なXMLの記述方式で記述される．そのサービス情報を示すサービス記述には，メッセージフォーマットなどを含む[2]．現在では複数の Web サービスを連携させて，新たなアプリケーションを構築するという手法が将来，ソフトウェア開発の面で有効になるのではないかという期待のもと，システム実行時に複数のWebサービス同士が連携し合って複合 Web サービスを構成するようなシステムの構築手法の開発が目指されており[3]，環境の基盤となるソフトウェアや構築手法，開発環境などの研究が行われている段階であり，標準化に向けて技術開発が進められている[4]．

複合 Web サービスではサービス同士が柔軟に連携しシステムを構成することが望まれる．Webサービスを連携する際に，そのフローをプログラム中にコーディングする必要があるため，柔軟性に問題がある．そこで，特定のプラットフォームに依存しないで，サービスのフローを記述できるフレームワークとしてXMLベースのビジネスプロセス記述言語が提案されている[1]．ビジネスプロセス記述言語として代表的なものが，BPEL4WS（Business Process Execution Language for Web Services）である．

しかしながら，BPELを用いて複合Webサービスを構成する際にフローの記述方式がまだ柔軟でない点がある．本稿ではBPELによって複合Webサービスを構成する際のサービスのフローを記述する言語を提案し，BPELを用いたアプリケーション構築を支援する．

2．複合WebサービスとBPEL4WS

現在Webサービスの連携について，連携のための専用言語を開発し，その言語を解釈，実行するエンジンを構築することによって複合 Web サービスを実現する研究が行われBPELが開発された．

Web サービスではこれまで，SOAP/WSDL によりサービス・インタフェースの記述が標準化された[5]が，個々のプログラムをWebサービスとして呼び出す

には別途ラッピングが必要であった．そのため，複数のWebサービスを組み合わせて，すぐ使えるWebサービスとして利用することが出来なかった．そこでBPELでは図1のように一連のプロセス(処理の流れ)をアクティビティ（処理単位）を繋げたフロー（遷移グラフ)として構成することによって複合Webサービスを実現している［6].

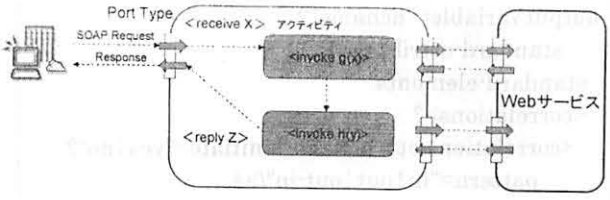

図1 BPEL4WSの構成

BPEL4WSは，複数のサービスを連携させて新しいサービスとして公開するためのビジネスプロセス記述言語である．BPELではアクティビティと呼ばれる制御構文によって記述され，sequence, while, switchといったアクティビティとflow, linkなどのフローグラフ記述のためのアクティビティを組み合わせることによってビジネスプロセスを記述する．BPELプロセス文書はそれ自身がWebサービスとして公開され，receiveアクティビティによってプロセスが呼び出しのリクエストを受け取る．その後各アクティビティによってビジネスプロセスが実行され，replyアクティビティによって結果を返すことになる．

3．Webサービス連携のためのフロー記述方式

3．1．サービス連携フローの記述支援

BPEL4WSでは利用するサービスの定義から必要な情報を取得してからBPELプロセス文書を記述してやらなければならないため，記述者はWebサービスの技術仕様について理解している必要がある．また，BPELプロセス文書はBPELエンジンによってWebサービス（BPELサービス）として公開されるが，このサービスのWSDL定義も手作業で記述する必要があり，直感的なフローの記述が難しくなっている．

本稿ではBPELによって実現される複数Webサービス連携の動作とサービス定義を記述するフロー記述言語および，フロー言語に変換可能なスクリプト言語を開発し，BPELを中心とした複数Webサービスの連携によるアプリケーションの構築手法を提案する．

本研究におけるフロー記述言語は，本来BPELによって記述される複合Webサービスのフロー，およびBPELサービスのWSDL定義を同時に記述できる．特に，BPELによるWebサービス連携の特色とも言えるpartnerやserviceLinkTypeといった要素の記述を統合した形で定義している．これによって，BPELよるWebサービス連携のための開発基盤を提供する．また，このフロー記述言語によって記述されたBPELサービスのフロー記述を解析・実行することのできるアプリケーションを構築する．

また，スクリプト言語ではJavaをベースに開発を行い，一般的なオブジェクト指向のプログラム言語同様，定義などを宣言しその後，if, whileといった命令によってフローを記述することを可能とする．そして，フロー記述言語への変換アプリケーションによりフロー記述言語に変換する．このアプリケーションによってビジネスプロセスのフロー記述が複合Webサービスによるアプリケーション開発のインターフェースにすることができる．

3．2．サービス連携フロー記述言語

本手法は，XMLによって定義されたフロー記述言語およびその記述，記述された文書の解析，実行可能なプロセスであるBPELプロセス文書への変換，BPELプロセス文書の実行という流れを持つ．また，実行可能なBPELプロセス文書のための実行エンジンとしてBPWS4Jを利用する．

3．2．1．フロー記述言語の言語要素

フロー記述言語では，従来BPELによるWebサービスの連携を行う際にプロセス文書，複合WebサービスのWSDLの二つを同時に記述することができる．

本手法で提案するフロー記述言語は，言語要素は役割によってBPELと同様に構造アクティビティ，基本アクティビティおよびその他の言語要素に分けられる．

(1) 構造アクティビティ

構造アクティビティはサービスのフロー構造を記述するための要素である．本手法におけるフロー記述言語の構造アクティビティは，BPELにおける構造アクティビティと同じ構造になっており，並列実行，繰り返し，条件分岐，イベントによる実行などといった制御構文を定義している．

・flow

並行実行と，linkとの組み合わせによるグラフ構造のフローの作成に用いる．

```
<flow standard-attributes>
  standard-elements
    <links>?
      <link name="ncname">+
    </links>
  activity+
</flow>
```

・switch

case要素とotherwise要素を用いて条件付分岐を実現するために用いる．

```
<switch standard-attributes>
    standard-elements
    <case condition="bool-expr">+
        activity
    </case>
    <otherwise>?
        activity
    </otherwise>
</switch>
```

・sequence

一つ以上のアクティビティを順次実行する．

```
<sequence standard-attributes>
  standard-elementsactivity+
</sequence>
```

・pick

あるアクティビティの発生に関連したアクティビティを実行する．

```
<pick createInstance="yes|no"? standard
-attributes>
```

```
 standard-elements
 <onMessage service="ncname"
 portType="qname"
      operation="ncname" variable="ncname"?>+
  <correlations>?
   <correlation set="ncname" initiate
    ="yes|no"?>+
  </correlations>activity</onMessage>
 <onAlarm (for="duration-expr" | until
  ="deadline-expr")>*
  activity
 </onAlarm>
</pick>
```

・empty
空のアクティビティ．何も実行しないことを明示的に示すために用いる．

```
<empty standard-attributes>
  standard-elements
</empty>
```

・scope
アクティビティの実行範囲を規定する．

```
<scope          variableAccessSerializable="yes|no"
standard-attributes>
 standard-elements
 <variables>? ... </variables>
 <correlationSets>? ... </correlationSets>
 <faultHandlers>? ... </faultHandlers>
 <compensationHandler>? ...
  </compensationHandler>
 <eventHandlers>? ... </eventHandlers>
 activity
</scope>
```

(2) 基本アクティビティ
基本アクティビティはフローの中でのサービスの呼び出し制御を記述するための要素である．
・receive
partner によって定義された operation に対してメッセージを受け取る．

```
<receive service="ncname" portType="qname"
    operation="ncname"
    variable="ncname"? createInstance="yes|no"?
    standard-attributes>
 standard-elements
 <correlations>?
  <correlation set="ncname" initiate="yes|no"?>+
 </correlations>
</receive>
```

・reply
receive に対応する応答としてメッセージを送信する．

```
<receive service="ncname" portType="qname"
    operation="ncname"
    variable="ncname"? createInstance="yes|no"?
    standard-attributes>
 standard-elements
 <correlations>?
  <correlation set="ncname" initiate="yes|no"?>+
 </correlations>
```

・invoke
partner によって定義された operation の呼び出しを行う．

```
<invoke service="ncname" portType="qname"
    operation="ncname"
    inputVariable="ncname"?
outputVariable="ncname"?
    standard-attributes>
 standard-elements
 <correlations>?
  <correlation set="ncname" initiate="yes|no"?
      pattern="in|out|out-in"/>+
  </correlations>
 <catch faultName="qname" faultVariable
    ="ncname"?>*
  activity
 </catch>
 <catchAll>?
  activity
 </catchAll>
 <compensationHandler>?
  activity
 </compensationHandler>
</invoke>
```

・assign
変数の作成，値の変更を行う．各アクティビティで変数の変更が行われなかった場合に用いる．

```
<assign standard-attributes>
 standard-elements
 <copy>+
  from-spec to-spec
 </copy>
</assign>
```

・wait
一定期間，あるいはある期限まで待機する．

```
<wait (for="duration-expr" | until
    ="deadline-expr")
    standard-attributes>
 standard-elements
</wait>
```

・throw
障害の通知．後述の faultHandler と同時に用いることで障害に対する処理を記述する．

```
<throw faultName="qname"
     faultVariable="ncname"?
     standard-attributes>
 standard-elements
</throw>
```

・terminate
フローを中断する．

```
<terminate standard-attributes>
 standard-elements
</terminate>
```

・compensate
scope を指定し，その有効範囲に対する補正を行う．

```
<compensate scope="ncname"?
    standard-attributes>
 standard-elements
</compensate>
```

(3) その他の言語要素その他の言語要素は，フローの中で用いられる変数や呼び出すサービスの定義などの記述，あるいは構造アクティビティ，基本アクティビティの動作を細かく記述するための要素である．

・serviceFlow
フロー記述文書を定義する．

・service
BPEL4WS における role 定義や serviceLink Type 定義, partner などといった定義を記述するために用いる要素である．

```
<services>
 <service standard-attributes>+
  <role standard-attributes>+
   standard-elements
  </role>
 </service>
</services>
```

・link
flow アクティビティと同時に用いることでグラフ構造のフローを記述する．

・source, target
flow, link によるグラフ構造を記述するために用いる．

・faultHandler, compensationHandler
faultHandler は例外処理，compensationHandler はフローの実行の補正を行うために用いる．

・onMessage, onAlarm
アクティビティの発生タイミングを指定するために pick アクティビティ内で用いる．

・case, otherwise
switch アクティビティで条件付分岐を実現するために用いる．case は指定した条件で分岐を行い，otherwise はデフォルト分岐となる．

・correlationSet
receive, reply, invoke といったアクティビティの相関関係を定義する．

```
<correlationSets>
 <correlationSet standard-attributes>+
  <property standard-attributes/>+
 </correlationSet>
</correlationSets>
```

・variable
メッセージを保存するためにフロー内で利用する変数の定義を行う．

・delete
アクティビティを削除するために用いる [3].

3．2．2．フロー記述文書の記述

Web サービスのフロー記述には XML を使用する．主にビジネスプロセスを記述するフローモデルのみを中心にした仕様を目的としていることと，実行可能なプロセスへの変換を行うことから，フローモデルの記述は BPEL4WS をベースに拡張した言語定義とした．本システムにおけるフロー記述言語が構成する要素はアクティビティと呼ばれる制御構文要素, Web サービスの場所，名前，機能，与える引数，フローで利用する変数などであり，主にアクティビティの配置の仕方でフローを定義することになる．

3．3．スクリプト言語によるフロー記述

本方式で提案するスクリプト言語では，一般的に普及しているプログラミング言語のように，条件分岐，繰り返し，中止といった処理の記述方法を定義し，その処理内容を解析して，フロー記述言語で定義されている各言語要素に変換することにより BPEL で実行可能なプロセス文書の記述を行う．

スクリプト言語の記述の構成としては以下のようにする．まず，連携するサービスの名前やメソッド，データ型などといった Web サービス利用に必ず必要になる要素を宣言する．それらを元にスクリプト言語で定義されている記述方式に従い，構成したい複合 Web サービスのフローを記述していく．

スクリプト言語によって記述されたプログラムを元に，以下のような流れでフロー記述言語に変換する．

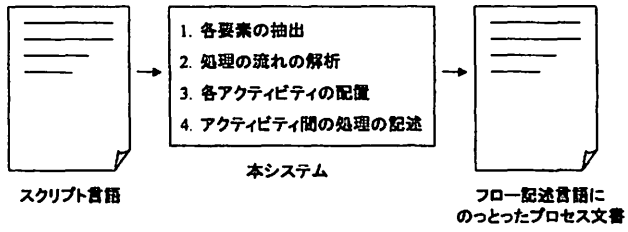

図 2　プロセス文書の記述

スクリプト言語からフロー記述言語を生成したのち，実装環境によって生成された BPEL プロセス文章は BPEL 上で複合 Web サービスとして公開され，各サービスの呼び出しを行っていく．

4．システムの実装

ここでは，本稿で提案した方式を用いたプロトタイプシステムを構築し，その動作についての評価方法を述べる．

4．1．構築システム

まず，サービスプロバイダとして予約サービス，検索サービス，清算サービスといった複数のサービスを構築しアプリケーションサーバに配置する．そして，サービスリクエスタはそれらのサービス群からサービスを複数選び出し，本システムを元にアプリケーションを構築する．このときサービスのフローは“Pattern Based Analysis of BPEL4WS “[6]で提唱されているワークフローパターンを元に構築する．

4．2．実装環境

以下のようなツールを用いてシステムの実装を行う．

表 1 使用ツール

役割	ツール名
OS	WindowsXP Professional
SOAP プロセッサ	Apache Axis1.2
プログラム実行環境	J2SDK1.4.2
XML パーサ	Xerces2.6.2
BPEL エンジン	BPWS4J2.0
Servlet エンジン	Apache Tomcat4.1
DataBase	MySQL4.0

4．3．システム構成

関連仕様，構築環境を元にスクリプト文書により複合 Web サービスを実現するシステムを構築する．

本システムにおける実装は，XML，Java を中心とした技術で構成する．本システムの構成要素は，各Ｗｅｂサービス，スクリプト言語からフロー記述言語への変換エンジン，フロー記述言語から実行可能なプロセス，複合ＷｅｂサービスのＷＳＤＬを生成するプログラムから構成される．

以下にその構成図を示す．

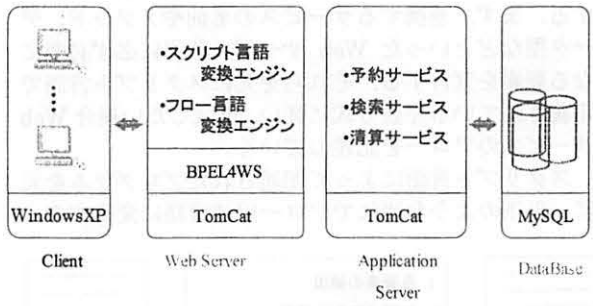

図３　プロトタイプシステムの構成

4．4．システムの処理の流れ

本システムの処理の流れは，まず利用するサービスの WSDL を元にそのサービスの情報を解析し，解析されたサービス情報を元にスクリプト言語により複合 Web サービスのフローを記述する．次にスクリプト文書を変換エンジンによりフロー記述言語に変換する．そして生成されたフロー記述文書を変換エンジンにかけ，BPEL で実行可能なプロセス文書と複合 Web サービスの WSDL を生成する．そのプロセス文書を BPEL エンジンに配置し，フローを解析実行することにより複合 Web サービスを実現する．

以下にその流れを図示する．

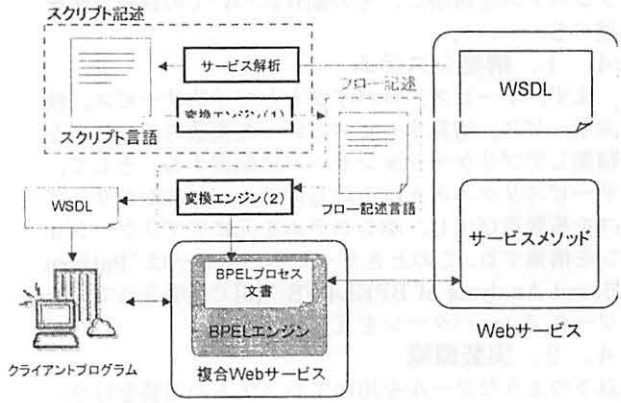

図４　処理の流れ

4．5．評価

”Pattern Based Analysis of BPEL4WS” [7]を参考にワークフローパターンおよびコミュニケーションパターンを設定する．これらのワークフローパターンについてスクリプト言語から変換されたプロセス文書を適用し，変換した BPEL プロセスが設定したパターンにマッチするかどうか，変換後の BPEL 文書について妥当性を検証する．文献[6]では，BPEL について代表的なワークフローパターンを記述できるかどうかを検証している．これらのワークフローパターンは順次実行や並列実行の他に，単純な送受信パターンなどであり，以下の 18 種類のワークフローパターンについてその動作を確認する．

- Sequence
- Parallel split
- Synchronization
- Exclusive choice
- Simple merge
- Multi choice
- Implicit termination
- Synchronizing merge
- MI without synchronization
- MI with a priori design time knowledge
- Deferred choice
- Interleaved parallel routing
- Chancel activity
- Cancel case
- Request/reply
- One-way
- Synchronous polling
- Message passing

5．まとめ

本稿では，フロー言語に変換できるエンジン，スクリプト言語の提案と実装およびその評価を行った．本方式を用いることによって，複合 Web サービスの構築支援が可能となるので，アプリケーションの開発効率が上がるといえる．今後は，スクリプト言語の処理定義をよりいっそう充実させることにより，様々なフローを記述することを可能とする．また，評価で用いた 18 種類のワークフローパターンによる評価だけで妥当なのかといったことの検討を行い，実際に実世界で使われるようなシステムに適用する場合の検証，またさらなるアプリケーションの構築支援のために，GUI の実装を行うことを課題としている．

参考文献

[1] 本 俊也「詳細 Web サービス構築」，ソフトバンクパブリッシング，2003

[2] 嶋本正，西本進，野間克司，福原信貴，「Java による Web サービス構築」，ソフトバンクパブリッシング，2002

[3] 佐藤智哉，小泉寿男，大川勉 「フロー記述言語による複数Webサービスの連携手法」，情報処理学会 第121回 マルチメディア通信と分散処理研究会 (DPS)，2005

[4] 藤田 悟，成田 雅彦，大場 みち子，鈴木 俊宏「Web サービスの標準化と相互接続性」，情報処理，Vol.45，No.12，pp.1272-1277，2004

[5] @IT http://www.atmarkit.co.jp/index.html

[6] P.Wohed, W.M.P.van der Aalst, M.Dumas, and A.H.M.ter Hofstede, "Pattern-Based Analysis of BPEL4WS," QUT Technical report,FIT-TR-2002-04, Queensland University of Technology, 2002