特別研究報告書

ドメインオントロジを用いたWebサービス連携の半自動化

指導教員　　石田 亨 教授

京都大学工学部情報学科

境 智史

平成19年2月9日

ドメインオントロジを用いたWebサービス連携の半自動化

境 智史

内容梗概

現在，数多くのWebサービスが提供され始めている．Webサービスはコンピュータを対象として構築されており，機械可読なサービスを提供している．Webサービスはそれ自身で一つのコンポーネントとみることができるため，複数のWebサービスを連携することで，新しいサービスを構築することができる．

Webサービス連携を行うためには，それぞれのWebサービスのインタフェースに基づいて，連携プロセスを作成する必要がある．そこで，サービスの種類によってインタフェースを標準化することにより，同種類のサービスを実現するWebサービス同士を変更することが容易になると考えられる．これにより，連携を行う際に使用するワークフローの作成コストは軽減されるものの，連携プロセス作成にはまだ多くの過程が含まれ，そこでの作業は人手を要するものが多い．この問題を解決するために，ユーザとシステムが協調作業を行うことで半自動的に連携プロセスを作成するアプローチをとる．Webサービス連携を半自動化するには，以下の2つの課題が存在する．

1. 多数の利用可能なサービスからの適切なサービスの選択
   サービス連携を行う際には，利用可能なサービスの中から，適切なサービスを選択する必要がある．また，利用したいサービスを提供するWebサービスが存在しない場合には，利用したいサービスに近いサービスを代用することによって，ワークフローの作成を行う．しかし，代用するサービスとして，同様の機能を提供できるようなサービスを選択する必要がある．
2. Webサービスを入れ替える際の，入出力変数の再割り当て
   ワークフローを作成する際，サービス連携が適切な結果を出力するように，実行するワークフローを修正する必要がある．修正の方法として，ワークフローの制御構造の変更は行わずに，実行するWebサービスの変更を行う方法がある．その場合，変更するWebサービスのインタフェースが同じであれば，入出力の変数割り当てを変更する必要はない．しかし，インタフェースが異なる場合は，サービスの入出力変数の割り当てを変更する必要がある．

本研究では，上記の問題点を解決するために，ドメインオントロジを利用す

る．ドメインオントロジとは，あるドメインの中にあるWebサービスに関する情報を記述したものである．情報とは，サービスやサービスが使用する変数の概念階層，さらに，サービスとメッセージタイプ間の関係や，メッセージタイプと変数間の関係を指す．また，サービス階層において，サブクラスのサービスは全てスーパークラスのサービスを実行することが可能である．このドメインオントロジを用いることにより，システムがサービスの選択や変数の変更に関して，提案を行うアーキテクチャを構築する．

まず，ドメインオントロジのサービスの概念階層を用いて，ユーザが指定したサービスクラスのサブクラスの探索を行い，それらのクラスのサービスを実現するWebサービスをデータベースから検索することで，ユーザが利用したいサービスを実現するWebサービスの提案を行う．さらに，実行するWebサービスが提案されない場合には，上位のクラスのサブクラスを探索することで，範囲を広げて探索を行い，代用可能なサービスを提案する．Webサービスの変更を行う際の変数割り当ての変更に関しては，変更前と変更後で使用する変数を探索し，変更する可能性がある変数対に対して，変数の階層グラフ上での距離を計算し，その距離がもっとも小さいものを変更する変数対として提案を行う．

本研究における主な貢献は以下の2点である．

1. 実行するWebサービス選択の半自動化
   ドメインオントロジのサービス階層を用いて，ユーザが指定したサービスを実現するWebサービスの提案，さらに，適切なWebサービスが存在しない際の代用サービスの提案を行うアーキテクチャを構築した．そして，それらの提案の中からユーザが選択したWebサービスを連携プロセスに反映するシステムの実装を行った．
2. 実行するWebサービス入れ替えに伴う変数割り当て変更の半自動化
   ドメインオントロジの変数階層グラフの距離を用いることで，Webサービス変更に伴う変数割り当ての変更に関する提案を行うアーキテクチャを構築した．ユーザが選択した新しいWebサービスに変更を行う際に，入出力変数の変数対の提案を行い，その提案にユーザが修正を加えたものを連携プロセスに反映するシステムの実装を行った．

上記2点の貢献により，Webサービス選択，Webサービスを変更する際の変数割り当て変更に関して，半自動化を行うことが可能となった．

Semi-automatic Web Service Composition using Domain Ontology

Satoshi SAKAI

Abstract

Various Web services are available now. The Web services provide a machine-readable service. Since a Web service can be regard as one component, we can make the new service by a composition of two or more Web Services.

It is necessary to make a coordinated process based on the interface of each Web service for the Web service composition. It is thought that standardizing the interface according to the kind of services makes it easy to change the same kind Web service. Although standardizing reduces the cost of making workflow used in a Web service composition, many works still remain in making the process of the composition and most of these works need human intervention. To solve this problem, I take the approach to make a workflow semi-automatically by the cooperation work of the system with the user. The following problems exist in making a Web service composition semi-automatically.

1. `Selection of an appropriate service from various services`
   It is necessary for a Web service composition to select an appropriate service from available services. Moreover, if an executable Web service does not exist in the service to be executed, the workflow is made by substituting service similar to it. However, it is necessary to select the alternative service that can provide a similar function.
2. `Reassignment of the I/O variable in replacing a Web service`
   In making a workflow, it is necessary to correct the workflow for a service composition to output the appropriate result. The correction method includes the method of changing the Web service executed without changing the control structure of the workflow. In the case of changing a Web service for new one that has the same interface, the I/O variable assignment needs not to be changed. However, in the case of changing it for new one that has another interface, the I/O variable assignment needs to be changed.

In this research, to solve the above problems I use a domain ontology. The domain ontology is a description of the information about Web services that

exist in a certain domain. The information includes concept hierarchies of services and variables and relations of services and message types and relations of message types and variables. Moreover, in the service hierarchy, all subclass services can execute a service of a super-class. With this domain ontology I construct the architecture that proposes Web services in the selection and variable reassignment in replacing Web service.

First, a system proposes the Web services that achieve the service the user wants to execute by searching for subclasses of the service class the user specified in a service hierarchy of the domain ontology and retrieving Web services that achieve the service of these classes from the database. Moreover, if a appropriate Web service is not proposed, the system expands search range by searching for the subclass of a high-ranking class and it proposes the services that can be substituted. In the case of variable reassignment, it searches for the variables used by the previous and new services and calculates the distance of variable pairs with possibility of change on a variable hierarchy graph. Then it proposes variable pair with the smallest distance.

The main contributions in this research are the following two points.

1. `Semi-automatically selection of an executed Web service`

   I constructed the architecture proposing Web services that achieve the service the user specified and proposing similar services in the case an appropriate Web service does not exist by using a service hierarchy of a domain ontology. And I implemented the system reflecting the Web service the user had selected from these proposals in a process of a composition.

2. `Semi-automatically variable reassignment in replacement of a Web service`

   I constructed the architecture proposing variable reassignments in replacing a Web service by using the distance on a variable hierarchical graph of a domain ontology. I implemented the system proposing change pairs of the I/O variable, and reflecting the one the user added the correction in a process of a composition in changing to a new Web service the user selected.

It is possible to select a Web service and change assignments in replacement of a Web service semi-automatically by these contributions.

ドメインオントロジを用いたWebサービス連携の半自動化

目次

第1章　はじめに

これまで，Web 上では，数多くの Web アプリケーションが公開されてきた．しかし，これらのアプリケーションは，人間が直接利用することを前提として，Web ブラウザを介してサービスを提供している．そのため，コンピュータが内容を解釈することは不可能であり，コンピュータが Web アプリケーションを利用することは難しい．

この問題を解決するために，Web サービスが存在する．Web サービスは機械可読なサービスを提供しており，サービス指向アーキテクチャを具現化するものとして注目されている．現在，さまざまな Web サービスが利用できるようになりつつある．

Web サービスは，それ自身を一つのコンポーネントとみることができるため，複数の Web サービスを利用することにより，サービスを連携させることが可能となる．そこで，サービス連携によって新たに有用なサービスを生み出す技術に期待が高まっている．現在，Web サービスを連携するワークフローの作成は BPEL [1] という言語を用いられることが多い．BPEL は XML に基づいた Web サービス連携を記述するワークフロー記述言語である．現在では，テキストエディタを用いて直接編集するのではなく，GUI を用いて，ワークフロー図を編集することにより，BPEL の編集を行う．

Web サービスの連携を行う際には，実行する Web サービスのインタフェースに基づいて連携プロセスを記述する必要がある．そこで，Web サービスをサービスの種類ごとにインタフェースの標準化を行うことによって，同じサービスを提供するサービス同士で，実行する Web サービスを変更することは容易になると考えられる．例としては，言語グリッド [2] というプロジェクトが挙げられる．このプロジェクトではインタフェースの標準化にしたがって，言語資源の Web サービス化を行っている．標準化により，ワークフロー作成のコストは軽減されるが，連携プロセス作成にはまだ多くの過程が含まれ，そこでの作業は人手を要するものが多く，以下のようなコストが依然存在する．

- 多数の利用可能なサービスから適切なサービスを選択するコスト
  Web サービス連携を行うには，利用可能なサービスの中から利用するサービスを選択する必要がある．また，利用したいサービスに実行可能な Web サービスが存在しない場合，利用したいサービスに近いサービスを代用す

ることによって，ワークフローを作成する．しかし，その際には，同様の機能を提供できるようなサービスを選択する必要がある．

- ワークフロー中で実行する Web サービスを入れ替える際に，入出力変数を再び割り当てるコスト
  ワークフローを作成する際，サービス連携が適切な結果を出力するように，修正を加える必要がある．修正の方法としては，Web サービスを実行する順番を変えるような流れ自体を変更する方法と，実行する Web サービスを変更して修正を行う方法が存在する．実行する Web サービスを変更する場合，同じインタフェースを持ったサービスに変更を行うのであれば，入出力変数の割り当てを変更する必要はない．しかし，異なったインタフェースを持つサービスに変更を行う場合には，入出力変数の割り当てを変更する必要がある．

既存の Web サービス連携を行う方法として，2 つの研究を紹介する．1 つ目は，Web サービスを OWL-S を用いて記述し，その OWL-S の中のサービスプロファイルという記述をもとに次に実行する Web サービスの絞り込みを行う研究 [3] である．この研究では，サービス連携を行う際，次に実行する Web サービスは，それまでに実行された Web サービスの出力を用いることから，サービスプロファイルを用いて，次に実行される Web サービスの入力を見ることで，次に実行することができる Web サービスの絞り込みを行う．2 つ目は，OWL-S で記述された Web サービスを，OWL-S の推論と，情報検索における類似度のマッチングを用いることで，Web サービスを発見するという研究 [4] である．この研究では，OWLS-MX と呼ばれる Hybrid Matchmaker を用いて，Web サービスを発見する．

本研究では，ドメインオントロジを用いることにより上述の問題点を解決する方法を提案する．ドメインオントロジとは，あるドメインの中のサービスに関する情報を記述したものである．例えば，言語サービスを記述するためのオントロジに関する研究 [5] がある．この研究では，言語サービスの自動合成や効率的なラッパー生成を目的として，OWL [6] を用いた言語サービスのオントロジを提案している．

本研究で使用するドメインオントロジには，サービスの概念階層，ドメイン中のサービスが使用する変数の概念階層，また，サービスと使用するメッセージタイプの関係，メッセージタイプと変数の関係を記述する．また，サービスの概

念階層において，サブクラスのサービスはすべて，スーパークラスのサービスを実行することが可能である．ドメインオントロジを用いることにより，Web サービスの選択，Web サービスの変更に伴う入出力変数の割り当て変更の半自動化を行っていく．

具体的には以下の 2 つの機能によって，上述の問題点の解決を図る．

1. ドメインオントロジのサービス階層を用いた，Web サービスの提案
   ドメインオントロジのサービスの概念階層を用いてユーザが指定したサービスクラスのサブクラスに相当するサービスクラスを探索する．そして，見つかったサブクラスを実現することができる Web サービスをユーザに提案することにより，より効率のよい Web サービス選択を行う．また，実行する Web サービスが存在しない場合には，ユーザが指定したサービスの上位のクラスから探索を行うことで範囲を広げた Web サービスの提案を行う．
2. Web サービス入れ替えに伴う入出力変数の割り当て変更の提案
   ユーザが実行する Web サービスを入れ替える際に，ドメインオントロジのサービス，メッセージタイプ，変数の関係を用いることにより，変更前のサービスと変更後のサービスで使用する変数を探索する．そして，割り当ての変更を行う可能性のある変更前の変数と，変更後の変数の対に対して，ドメインオントロジの変数の概念階層グラフにおける距離を計算する．それらの距離のもっとも小さいものを割り当ての変更対として，提案を行う．その提案にユーザが修正を加えることで，割り当ての変更を確定する．さらに，作成されたワークフローをたどることにより，割り当ての変更を行う．

本稿では，第 2 章で Web サービス連携を行う際に必要となるワークフロー作成の現状について述べる．上記の問題を解決するために，第 3 章では，ドメインオントロジを用いたワークフロー作成の半自動化の手法を述べ，第 4 章で，それらの方法を用いた実装と動作について述べる．最後に，第 5 章で本研究のまとめを行う．また，本稿では，抽象的な Web サービスをサービスと記述し，実行可能な Web サービスのみを Web サービスと記述する．

第 2 章　Web サービス連携

2.1　Web サービス連携手順

現在，ワークフローの作成は以下の手順で進められている．

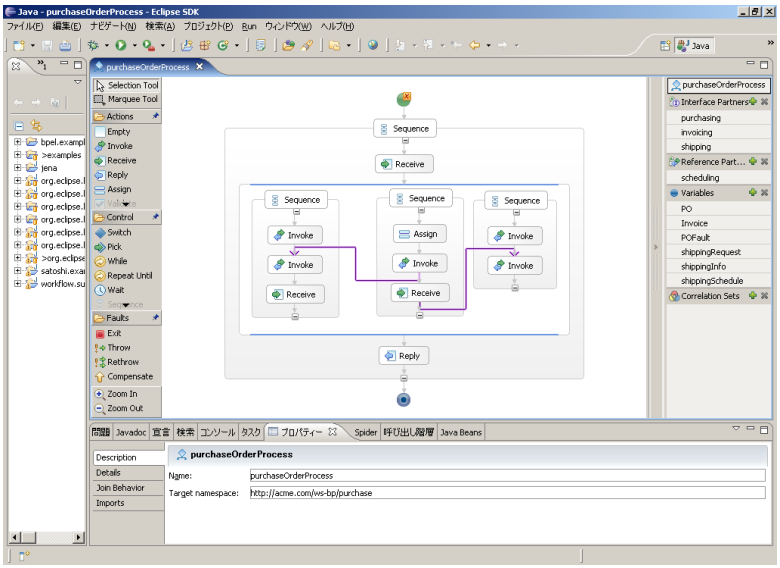

図 1: Eclipse BPEL Designer Editor

1. 実行したい Web サービスを選択

   Web サービス一覧の中から，サービス連携の中で実行する Web サービスを選択する．現在，Web サービスを公開しているサイトも増えてきており，たとえば，XMETHOD [1)]などの Web サービスを提供しているサイトから，Web サービス連携を行う際に必要な Web サービスを検索する．さらに，企業・組織内が利用する Web サービスのデータベースを持っており，その中から Web サービスを検索することもある．

2. Web サービスの実行フローを作成

   連携する際の Web サービスの実行順序や，制御構造などを記述する．BPEL では，Web サービスの呼び出しや，順列実行，分岐などをアクティビティという形で利用してワークフローを作成する．実際には図 1 のような GUI 上でアクティビティを用いてワークフロー図を作成することにより BPEL の記述を行う．

3. Web サービスの入出力の対応付け

   変数に値を割り当て，サービス間の入出力を対応付ける．先ほど作成したワークフローの中の Assign アクティビティを定義する．Assign アクティビティとは，プロセスの中で使用する変数を更新するもので，プロセス変数から別のプロセス変数へのコピー，Xpath 式によるプロセス変数の更新な

[1)] http://xmethods.net/

どを行う．このアクティビティを記述することで，Web サービスの入出力の変数を割り当てる．

4. 作成したワークフローの修正
   連携が期待通りの結果を出力するように，ワークフローの制御構造の変更や，ワークフロー中で呼び出される Web サービスを入れ替える．また必要に応じて 1，2，3 の手順を繰り返し，実行フローや入出力の対応付けに対し変更を行う．

2.2 Web サービス連携の問題点

Web サービス連携を行うワークフローを作成するには，人手で行う作業が多いために，大きなコストがかかっている．特に大きなコストがかかっている点としては以下の 2 点が挙げられる．

1. 実行したい Web サービスの選択の困難さ
   現在，実行したい Web サービスを検索する際，利用したいサービスの種類で検索を行う．しかし，利用したいサービスを実現するが，異なる種類のサービスとして登録されているために，検索が困難なことがある．また，利用したいサービスに適切な Web サービスが存在せず，代用するサービスを選択しなければならない場合に，利用したいサービスと同様の機能を提供するサービスを探さなくてはならない．例えば，ユーザが翻訳サービスを利用したいと考えていたとする．そのときに，翻訳サービスには適切な Web サービスが存在せず，機械翻訳サービスに適切なサービスが存在するという場合があり，そのときには，翻訳サービスではなく機械翻訳サービスを探さなくてはならず，コストがかかる．また，翻訳サービスが存在しない場合に，代用するサービスとして，用例対訳サービスを選択する場合には，まず，代用となるサービスが用例対訳サービスであるということを判断しなくてはならない．
2. ワークフローの修正を行う際の Web サービス入れ替えの困難さ
   利用者の要求は多様であるため，Web サービス連携が期待通りの動作を行うには，ワークフローの修正が欠かせなくなってきている．そこで，実行する Web サービスの変更を行う際，同じインタフェースを持った Web サービスに変更する場合には，入出力の変数の割り当てを変更する必要はないが，異なったインタフェースに変更する場合には，入出力割り当てを変更する必

要がある．現在，インタフェースの標準化に基づいたWebサービスを提供するプロジェクトが存在する．このプロジェクトでは，同種類のサービスは同じインタフェースを持つため，同種類のサービス変更の際には入出力変数の割り当てを再び行う必要は無い．しかし，異なる種類のサービスはインタフェースが違うため，異なるサービスに変更を行った際には，入出力変数の割り当ての変更を行わなくてはいけない．また，この入出力変数の割り当ての変更は人手で行わなくてはならないので，コストが大きい．例えば，医療用語を適切に訳せる翻訳サービスが存在しない場合，医療用語の部分だけでも訳すことができる対訳サービスが存在すれば，利用者にとっての価値は大きい．そこで，翻訳サービスの代用として用例対訳サービスを用いる場合，それぞれのWebサービスのインタフェースが異なっているために，入出力の変数の割り当てを変更する必要がある．翻訳サービスが使用していたMessageのsourceという変数への割り当てを，用例対訳が使用するMessageのsourceという変数への割り当てに変更する必要がある．また，翻訳サービスが使用するMessageのtranslateReturnという変数をほかの変数に割り当てていたものを，用例対訳が使用するMessageのsearchReturnというパートのsearchReturn/parallelText/element/targetという変数がほかの変数に割り当てるように変更を行わなければならない．このような変更は利用者にとって煩雑な作業であり，時間を要する．また，誤って不適切な割り当てを指定するなどの問題も生じる．

第3章　ドメインオントロジを用いた提案手法

3.1　OWLを用いたオントロジ

本研究では，Webオントロジ記述言語であるOWL [6] によって記述されたドメインオントロジを利用する．ドメインオントロジを作成するために，以下のような3種類のクラスを定義した．

- Serviceクラス
  ドメイン中のサービスの種類を，それぞれをひとつのクラスとして記述する．そして，このサービスクラスに階層関係を持たせる．
- Messageクラス
  Webサービスを実行するときに使用するメッセージを記述する．

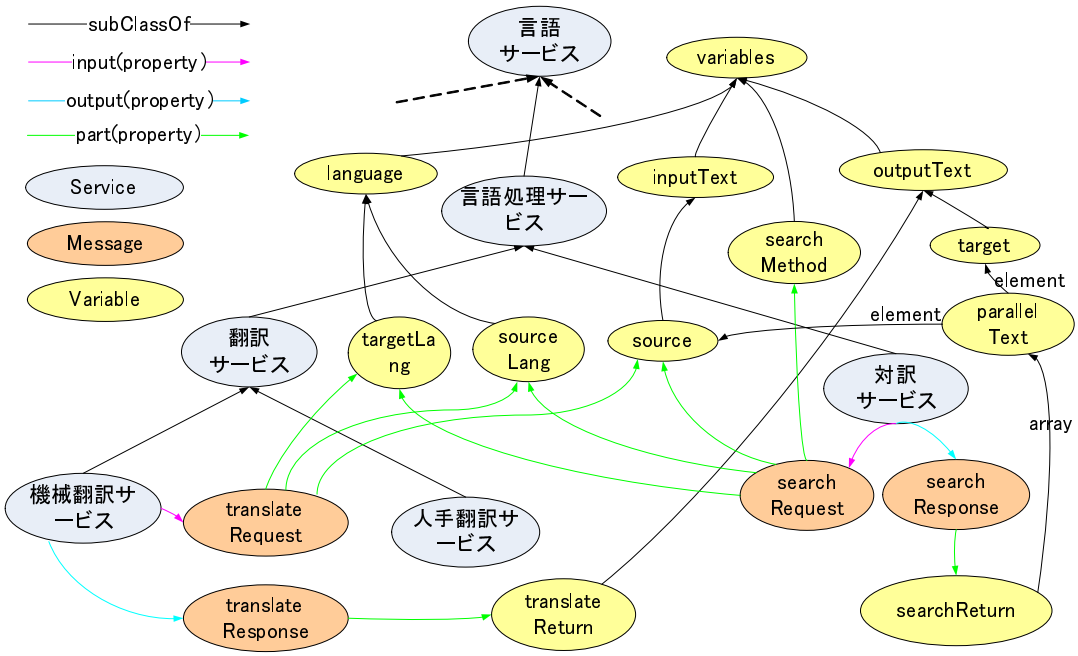

図 2: ドメインオントロジの例の一部

- Variable クラス
  ドメイン中の Web サービスが使用するメッセージの中で用いられている変数全てをこのクラスを用いて記述する．

ドメインにおけるサービスのオントロジを作成するために，本研究では，以下のようなプロパティを用意した．

- subClassOf プロパティ
  Service クラス群と Variable クラス群の階層関係を記述するために使用する．このプロパティをたどることにより，クラス間の概念階層を求め，サービスの提案や変数の提案を効率化する．
- input プロパティ
  domain は Service クラス，range は Message クラスである．Service クラスのインスタンスの Web サービスが入力として必要とする Message クラスを指定する．
- output プロパティ
  domain は Service クラス，range は Message クラスである．Service クラスのインスタンスの Web サービスが出力として必要とする Message クラスを指定する．

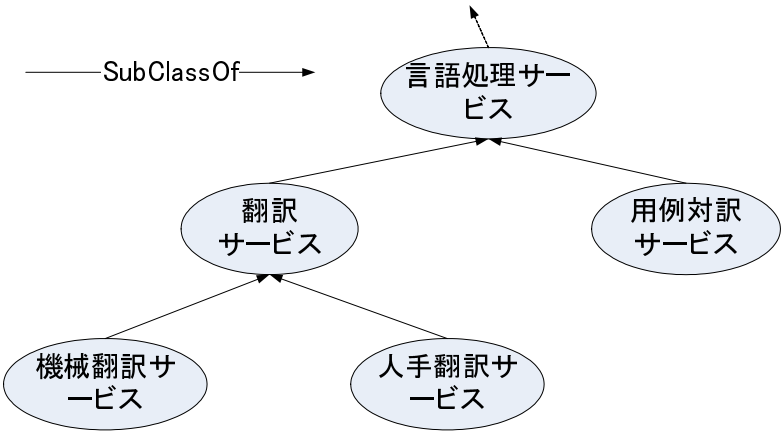

図 3: サービス階層の例

- part プロパティ
  domain は Message クラス，range は Variable クラスである．Message で使用する変数を指定する．
- array プロパティ
  domain，range ともに Variable で，入出力変数の中で配列を定義する際に使用する．
- element プロパティ
  domain，range ともに Variable で，入出力変数の中で構造体を定義する際に使用する．

ドメインオントロジは，これらのクラスとプロパティを用いて，そのドメインの専門家が作成する．言語サービスに関して作成したドメインオントロジの例が図 2 である．言語サービスにはいくつかのサブクラスが存在するが，その中の言語処理サービスに関して詳細に示した．これらのクラスとプロパティを用いて作成されたオントロジを用いて，Web サービス選択の半自動化と実行サービス変更に伴う入出力変数割り当て変更の半自動化を行う．

3.2 Web サービス選択の半自動化

Web サービスの連携を半自動化するために，本研究では，ひとつの機能として，Web サービス選択の半自動化という機能を提案する．これは，ワークフロー作成の際に，ユーザが利用したいサービスの種類を選択することにより，システムが Web サービスを提案する機能である．この機能を実現するために，以下

```
<WebServices>
  …
  <WebService>
  <Name>JServer</Name>
  <WSDL>file:/C:/wsdlfile/JServer.wsdl</WSDL>
  <ServiceClass>MachineTranslation_Service</ServiceClass>
  <OperationName>translate</OperationName>
  <Location>http://localhost:8080/wrapper_current/services/JServer
  </Location>
  </WebService>
  …
</WebServices>
```

図 4: Web サービスデータベースの例

のような手法を用いる．

- サービスクラスの概念階層の利用
  サービスクラス間のサブクラスプロパティを用いて，サービスの階層を探索する．まず，ユーザが利用したいサービスクラスのサブクラスを探索することで，ユーザが利用したいサービスを実行することができるサービスの種類全てを探索する．さらに，ユーザが指定したサービスの上位のサービスに探索範囲を広げることにより，より多くのサービスの種類を探索することが可能となる．
  図 3 にサービス階層の例を示した．この例の場合，翻訳サービスのサブクラスである機械翻訳サービスや，人手翻訳サービスは翻訳サービスを実行することができる．つまり，ユーザが利用したいサービスとして翻訳サービスを指定した場合，翻訳サービス，機械翻訳サービス，人手翻訳サービスを実現する Web サービスの提案を受けることで，ユーザが実行したい Web サービスを選択することができる．また，それら 3 つのサービスに実行できる Web サービスが存在しないときは，探索の範囲を一つ上位の言語処理サービスに広げることで，言語処理サービスとそのサブクラスである，用例対訳サービスを実現する Web サービスの提案を受けることができ，代用サービスをこれらの Web サービスの中から選択することができる．
- Web サービスデータベースの検索
  Web サービスデータベースの検索を行う．Web サービスデータベースには，

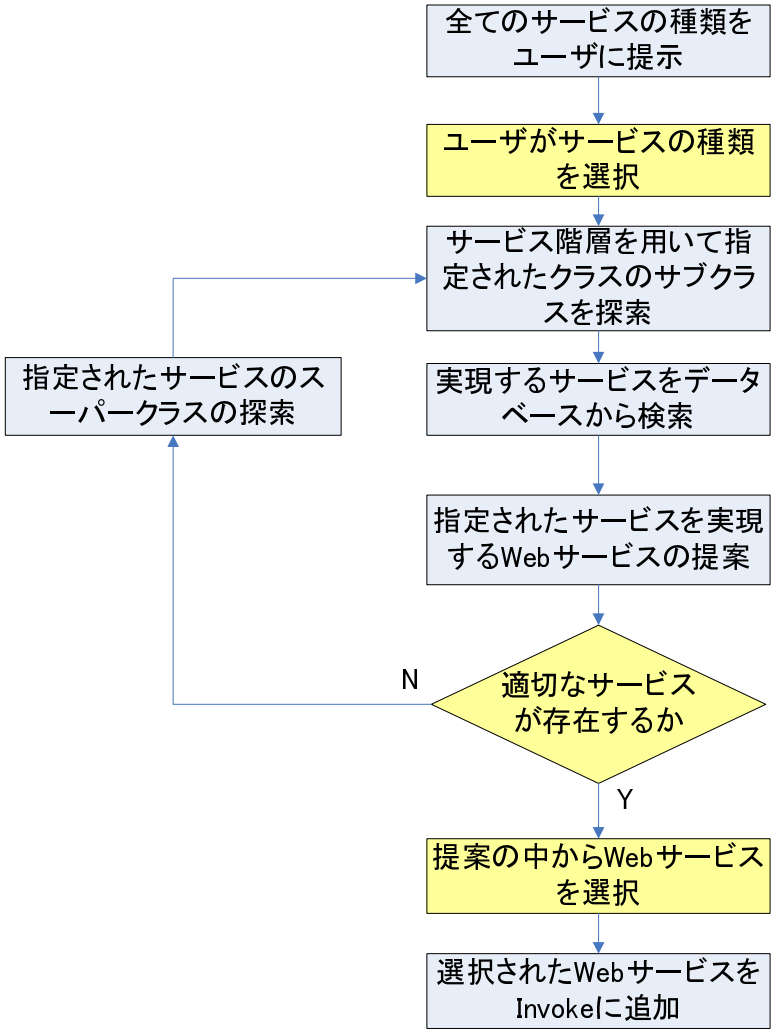

図 5: Web サービス選択半自動化のフロー

Web サービスそれぞれが，どのサービスのクラスに属していて，どのようなオペレーションを持っているのかを記述している．サービスクラスの種類を用いてこのデータベースを検索することにより，そのサービスを実行することができる Web サービスを発見することができる．本研究で使用した XML ファイルの例を図 4 に示した．

Web サービスの選択を半自動的に行うためのアルゴリズムフローを図 5 に示した．水色の処理はシステムが行い，黄色の処理はユーザが行うことによって処理を進めていく．まず，ユーザにドメイン中に存在するサービスの種類すべてを提示する．その中から，ユーザが利用したいサービスを選択することで，ユーザが利用したいサービスを確定する．次に，ユーザが利用したいサービス

クラスのサブクラス全てを探索することで，ユーザが利用したいサービスを実行することができるサービスを得る．それらのサービスを実現することができるWebサービスをデータベースから検索し，得られたWebサービス群をユーザに提案する．ユーザは，得られた提案に対し，適切なサービスが存在するかを判断する．存在しない場合は，指定されたサービスの一つ上位のサービスに探索範囲を広げて，再びサブクラスの探索に戻る．存在した場合は，提案の中から，ユーザが実行したいWebサービスを選択する．そのサービスをワークフローのInvokeとして追加を行う．これら一連の流れを行うことにより，半自動的にWebサービスの選択を行うことができる．

3.3 Webサービス入れ替えに伴う変数割り当て変更の半自動化

すでにワークフローが完成した状態で，実行するWebサービスを変更する際，3.2節の手法を用いて，Webサービスの提案を再び行い，選択を行う．ここで，サービスの選択には以下の２つのパターンが存在する．

- 以前のサービスと同じインタフェースを持つサービスに変更
- 以前のサービスと異なるインタフェースを持つサービスに変更

同じインタフェースを持つサービスに変更する場合には，同じMessageを使用すればよいため，変数の割り当てに変更を加える必要はない．しかし，異なるインタフェースを持つサービスに変更を行う場合は使用するMessageが異なるため，入出力の変数の割り当てを再び行う必要がある．

そこで，本研究では，２つめの機能として，サービス変更の際に，システムが変更すべき変数の割り当てを提案する．その機能を実現するために以下のような手法を用いる．

- 変数の探索
  まず，サービスがどのような変数を用いているのかを調べる必要がある．そこで，3.1節のオントロジを用いる．まず，サービスのinputプロパティとoutputプロパティを用いて，そのサービスがどのMessageを使用するのかを探索する．さらに，Messageクラスのpartプロパティを探索することで，変数サービスがどのような変数を使用しているのかを調べる．この際，arrayやelementプロパティも同様に探索する．
- 変数割り当て変更対の計算
  サービスを変更した際，変更前のサービスと，変更後のサービスで同様の変

数を用いている場合には割り当ては，変更前と同様に行う．変更前のサービスと変更後のサービスで異なる変数を用いていた場合，変更前のサービスで用いていた変数を変更後で新たに使用される変数に割り当てを変更する．このとき，変更を行う変数がどちらかに 2 つ以上ある場合は，Variable の階層における，変数同士の距離を利用して，変更対の決定を行う．距離とは，階層において subClassOf プロパティを枝として，変数対を結ぶことができる最小の枝の数のことである．この距離がもっとも小さいものを，変更する対とする．例えば図 2 のドメインオントロジについて，機械翻訳サービスから用例対訳サービスに変更を行ったとする．その際，出力変数として translateReturn から，target または source に変更する必要がある．このときに，このオントロジ上での，translateReturn から target までの距離は 2 である．また，translateReturn から source までの距離が 4 であるので，距離が小さい target に変更を行うことを提案する．

- 変更する Message の変数割り当ての箇所の決定
  先ほどの 2 つの手法を用いて，サービス変更に伴う変数の割り当ての変更を決定した．しかし，実際に割り当ての変更を行う際に，どこを変更すればよいのかを調べる必要がある．そこで，入力の変数に関しては，Web サービスの変更を行うところからワークフローをさかのぼっていき，その変数に割り当てを行ったところに変更を加える．出力の変数に関しては，ワークフローをたどっていき，その変数が新たな変数を割り当てられるまでに変数を使われるところに変更を加える．

Web サービス変更における半自動化のアルゴリズムフローを図 6 に示した．水色の処理はシステムが行い，黄色の処理はユーザが行うことによって処理を進めていく．まず，図 5 のフローを用いてユーザが新しい Web サービスを選択する．そこで，変更する Web サービスが，変更前の Web サービスと同クラスのサービスであるかをそれぞれのサービスクラスを参照することで，判断する．同クラスのサービスの場合は，標準化によって Web サービスのインタフェースが同じであるので，入出力変数の割り当てを変更せずに，実行する Web サービスの変更のみを行う．変更前のサービスを別のクラスのサービスに変更を行う場合，Web サービスのインタフェースが異なるために，入出力変数の割り当ての変更を行う必要がある．そこで，変更前のサービスと変更後のサービスそれぞれに対して，使用する変数を調べる．変更前のサービスが使用していた変数

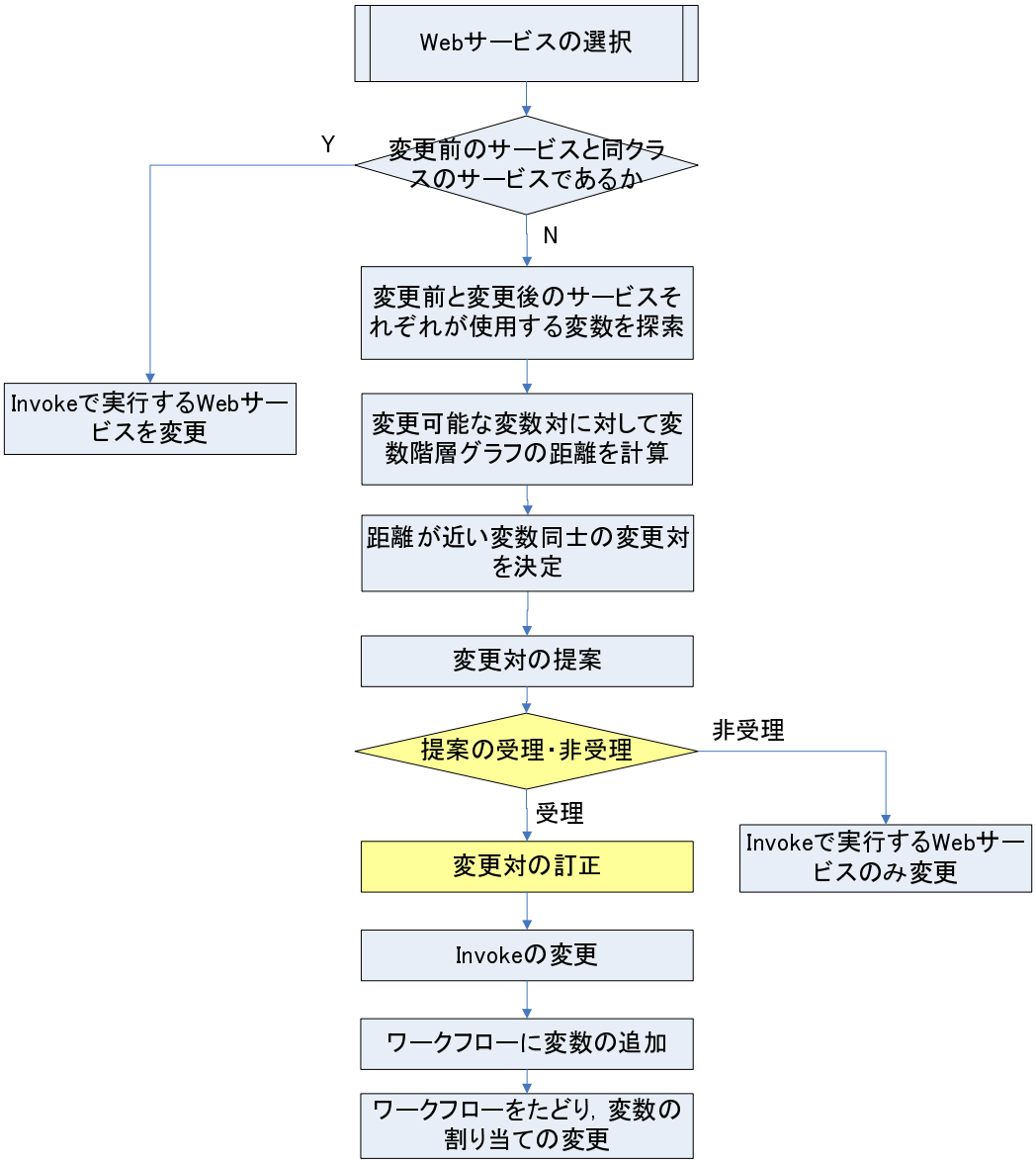

図 6: Web サービス変更時の変数割り当て半自動化のフロー

の割り当てを変更後のサービスが使用する変数に変更する必要があるため，変更する可能性がある変数対に対して，ドメインオントロジ中の変数階層グラフの距離を計算する．そして，全ての変数対に対して，距離の小さいものから，変更対としていく．全ての変数に関して，変更対が決定したら，ユーザに変更対の提案を行う．ユーザが提案を受理しない場合は，Web サービスの変更のみを行い，変数割り当ての変更は行わない．ユーザが提案を受理する場合，ユーザが提案の修正を行う．変更対の修正が終わったら，まず，Invoke の Web サービス

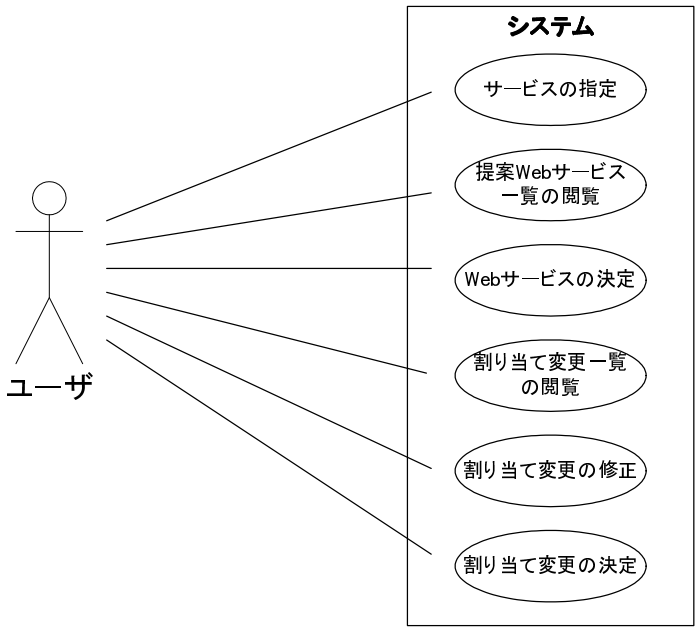

図 7: ユースケース図

を変更し，変更を行った Invoke からワークフローをさかのぼって入力変数の割り当てを変更し，また，変更を行った Invoke からワークフローをたどることで，出力の変数割り当ての変更を行う．これら一連の流れを行うことにより，Web サービス変更に伴う変数割り当てを半自動的に行うことができる．

第 4 章　実装と動作

4.1　システムの実装

4.1.1　システム構成

本研究で実装するシステムのユースケース図を図 7 に示した．システムは以下のような機能をユーザに提供する．

- 全サービスの種類の中から利用したいサービスの指定を行う機能
- システムが提案する Web サービスを一覧する機能
- 提案された Web サービスの中から実行したいものを選択する機能
- Web サービスを変更する際に，システムが作成した変数割り当て変更対の提案を一覧する機能
- システムが提案する割り当て変更対の修正を行う機能
- システムが提案した割り当ての変更の受理，または拒否する機能

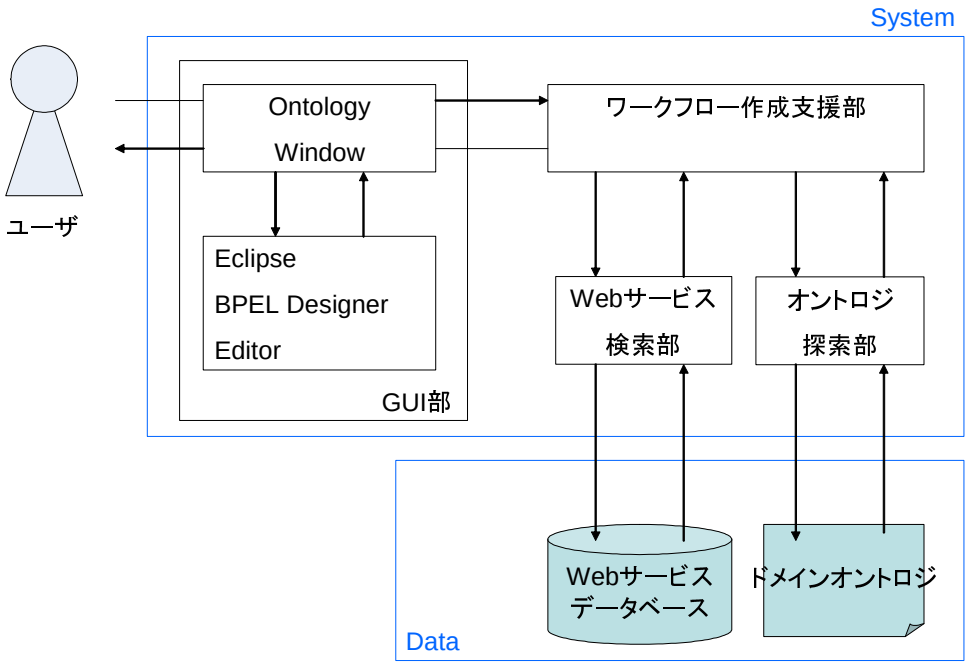

図 8: システム構成図

以上の機能を満たすシステムの構成図を図 8 に示した．個々のモジュールは以下のようになっている．

- GUI 部
  GUI 部は 2 つの部分で構成されており，BPEL エディタ本来の GUI は Eclipse のプラグインとして提供されている BPEL Designer Editor に改良を加えることによって作成する．また OntologyWindow に関しては，GUI を用いて，ユーザにサービスの提案や，変数変更対の提案を行う部分である．この OntologyWindow がワークフロー作成支援部の機能を用いて，提案を行う．さらに，その提案をユーザが受理した場合には，その結果を BPEL エディタ本来の GUI に渡す．BPEL エディタ本来の GUI は受け取った値を BPEL に反映するように改良を行う．
- ワークフロー作成支援部
  Web サービス検索とオントロジ探索を利用して，Web サービスの提案と入出力変数割り当て変更の提案の 2 つの機能を実現しＧＵＩ部に提案を渡す部分である．
- Web サービス検索部
  ワークフロー作成支援部より指定されたサービスの種類を実現する Web

サービスをデータベースの中から検索し，該当するWebサービスの情報をワークフロー作成支援部に返す.

- オントロジ探索部
  オントロジ探索部は2つの機能を持つ．1つ目の機能は，指定されたサービスの全てのサブクラスを探索し，それらのサービスのクラスをワークフロー作成支援部に返す．2つ目の機能は，指定されたサービスの入出力の変数のクラスを探索し，それらの変数のクラスをワークフロー作成支援部に返す．オントロジ探索部はJena [2)]を用いて実装を行う.
- Webサービスデータベース
  Webサービスの情報を保持するXML文書.
- 言語サービスオントロジ
  ドメインオントロジを記述したOWL文書

4.1.2 処理の流れ

Invokeで実行するWebサービスの追加，さらに，Invokeで実行するWebサービスの変更とそれに伴う入出力変数の変更を行う際のシーケンス図をそれぞれ図9，図10に示した.

まず，Invokeで実行するWebサービスの追加におけるシステムの処理の流れを説明する．以下のステップを行うことにより処理を行う.

1. サービスの種類の提案
   サービスの全種類の要求がGUIにくると，ワークフロー作成支援部にサービスの全種類の要求を行う．そこで，ワークフロー作成支援部は，オントロジ探索部に全てのサービスクラスを要求する．オントロジ探索部は，まず，オントロジ文書を読み込み，オントロジのグラフ情報を読み込む．そして，そのグラフ情報を探索し，存在する全てのサービスクラスを返す．ワークフロー作成支援部は，それらのクラスを，文字列として，GUIに返すことで，ユーザに全てのサービスの種類を提示する.
2. Webサービスの提案
   ステップ1で提案したサービスの中から，ユーザが選択したサービスを実現するWebサービスを提案するため，以下の3ステップを行う.
   (a) サブクラスの探索

[2)] Jena - A Semantic Web Framework for Java

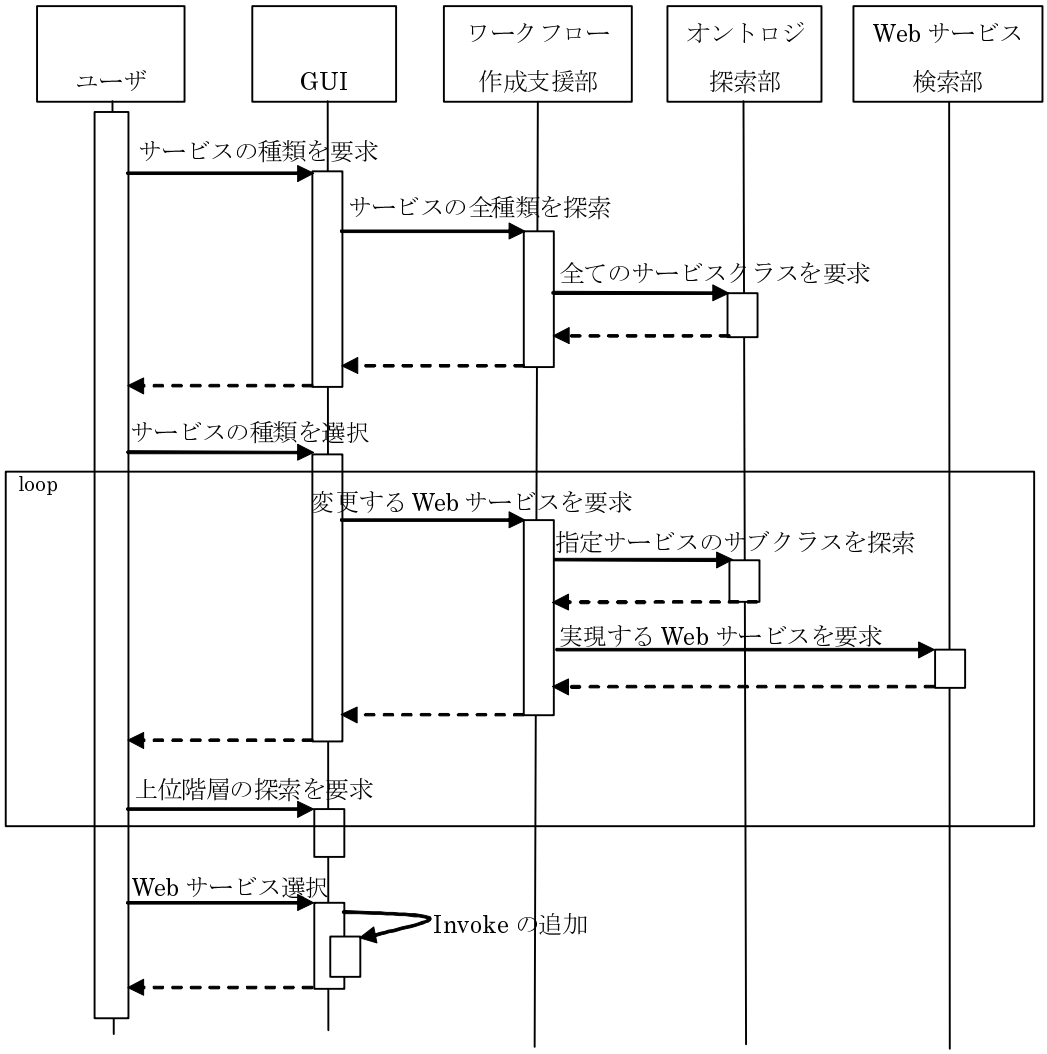

図 9: Invoke を追加する際の処理の流れ

ワークフロー作成支援部がオントロジ探索部に，ユーザが選択したサービスクラスの全てのサブクラスの探索を要求する．それを受けて，オントロジ探索部は，ステップ 1 で読み込んだオントロジのグラフ構造の中から，指定されたサービスクラスを探索する．そして，そのサービスクラスの subClassOf プロパティをたどっていくことにより，指定されたサービスクラスの全てのサブクラスを探索する．それらをワークフロー作成支援部に返す.

(b) Web サービスの検索

ワークフロー作成支援部は先ほどのステップで得たユーザが指定したサービスクラスの全てのサブクラスを Web サービス検索部に与える.

Web サービス検索部は与えられたサービスクラスを実現することができる Web サービスをデータベースから検索する．全てのサービスクラスに対して，検索が終了したら，ワークフロー作成支援部に，Web サービスの情報を返す．

(c) Web サービス一覧の提示
ワークフロー作成支援部は，先ほどのステップで得た Web サービスの情報を GUI 部の OntologyWindow に与え，その情報をユーザに提示する．

3. Web サービスの選択
提案された Web サービスの中に適切なサービスが存在する場合は，ユーザはその Web サービスを選択する．適切なサービスが存在しないとユーザが判断した場合，一つ上位のクラスをユーザが指定したとして，ステップ 2 の (a) に戻り再び提案を行う．
4. 選択された Web サービスの BPEL への反映
提案の中からユーザが選択した Web サービスを BPEL に反映する．その際に，まず，namespace，import，partnerLink の追加を行い，それらを用いて，Invoke の Web サービスを追加する．

次に，Invoke で実行する Web サービスの変更に伴う入出力変数の変更におけるシステムの処理の流れを説明する．以下のステップを行うことにより処理を行う．

1. 変更するサービスの提案
Invoke の追加のステップ 1，2，3 と同様の処理を行うことにより，ユーザに新しい Web サービスを提案し，ユーザが新しい Web サービスを選択する．
2. 変更前のサービスと変更後のサービスの比較
変更前の Web サービスのサービスクラスを Web サービスデータベースから検索し，その Web サービスと新しい Web サービスのサービスクラスを比較する．同じサービスクラスの Web サービスに変更する場合は，Invoke で実行する Web サービスを変更して終了．別のサービスクラスの Web サービスに変更を行う場合には，ステップ 3 に進む．
3. サービスの変数探索
変更前と変更後のサービスクラスをそれぞれ，オントロジ探索部に与える．オントロジ探索部は，与えられたサービスクラスのプロパティ値をたどる

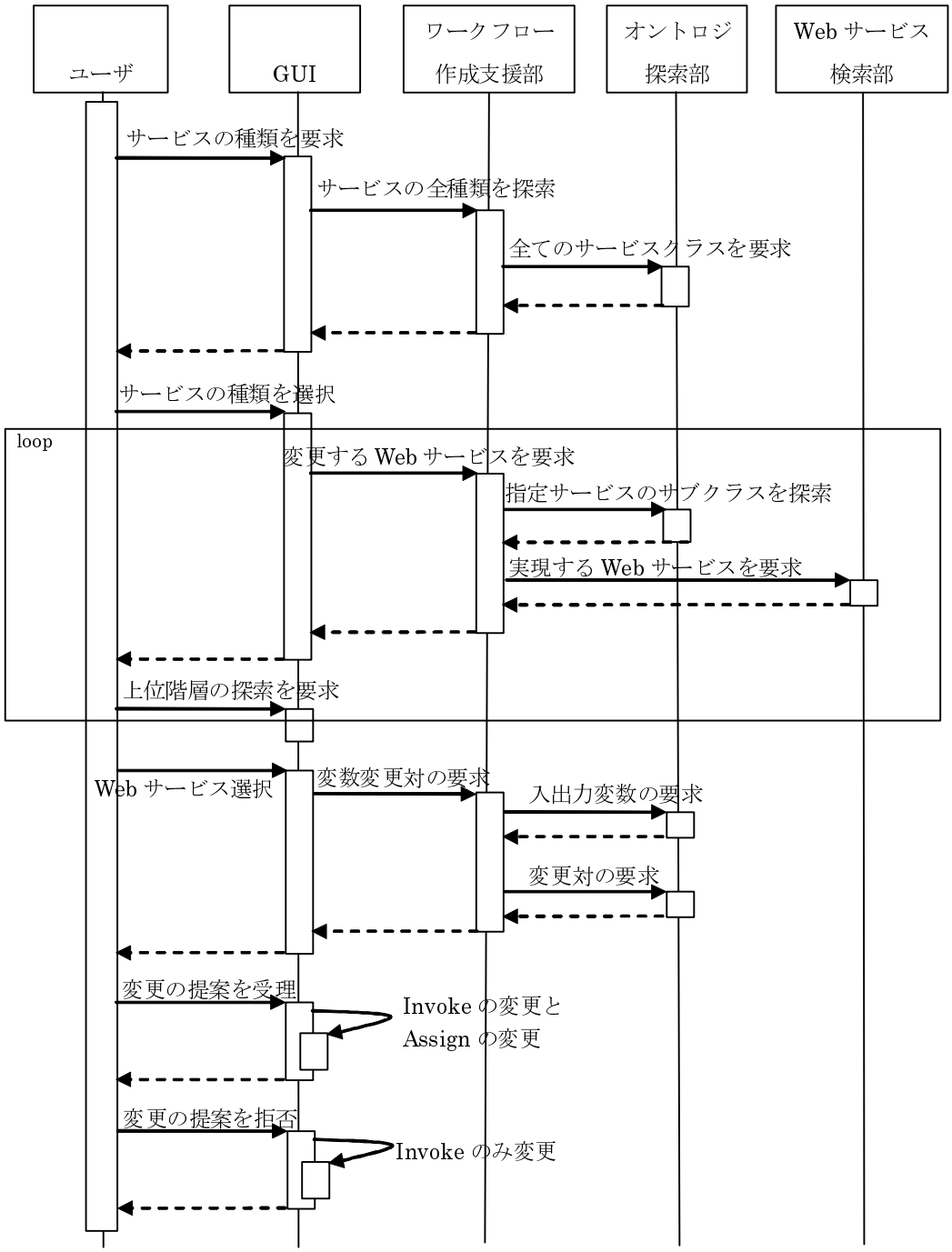

図 10: Web サービスを変更する際の処理の流れ

ことで，入出力変数を探索する．それぞれの Web サービスの入出力変数をワークフロー作成支援部に返す.

4. 変更する変数対の決定

ワークフロー作成支援部から，変更前で使用する変数と，変更後で使用す

る変数全てを，オントロジ探索部に与える．オントロジ探索部は，与えられた変数の変更が行われる可能性のある全ての変数対に対して，ドメインオントロジ中の変数階層のグラフ上の距離を求める．距離は，変数対のうち，変更前の変数から，subClassOf プロパティをたどり，変更後の変数に到達できる枝の数として計算する．計算した変更対のうち，もっとも距離が小さいものを変更対として，決定する．それらの変数対をワークフロー作成支援部に返す．

5. Assign の変更一覧の提示
   ワークフロー作成支援部は，先ほどのステップで得た変数変更の情報を GUI 部の OntologyWindow に与え，その情報をユーザに提示する．ここで，ユーザが，変数の変更を拒否した場合は Invoke の追加の処理の際のステップ 4 と同様の処理を行うことで Invoke の変更のみを行って終了する．受理する場合には，Reassignment ウィンドウを用いてユーザが変数変更対の修正を行いステップ 6 に進む．
6. Assign の変更の BPEL への反映
   Reassignment ウィンドウでユーザによって修正された変更対を GUI 部に渡す．それをもとに変数の割り当ての変更を行う．まず，Invoke の追加のステップ 4 と同様にして，Invoke で実行する Web サービスの変更を行う．そして，新たなサービスが使用する変数を BPEL に追加し，Invoke で使用する変数を追加した変数に変更する．変数割り当ての変更は，入力変数の割り当てを変更する場合には，ワークフローをさかのぼっていき，変更対の割り当てを行っているところを変更する．出力変数の場合には，次に，変更前に用いていた変数を用いて実行される Web サービスまでワークフローをたどり，変更対の割り当てを行っているところを変更する．

4.2 システムの動作

まず，Invoke の追加を行う際のシステムの動作を説明する．BPEL エディタ上で，Invoke を選択すると，エディタ上にその Invoke のプロパティ画面が表示される．その中から，Ontology ボタンをクリックすると，図 11 に示した Ontology ウィンドウが出現する．そのウィンドウの Ontology というテキストボックスに利用するドメインオントロジのファイルを指定し，さらに Database というテキストボックスに利用するドメインのデータベースファイルを指定する．そこで，

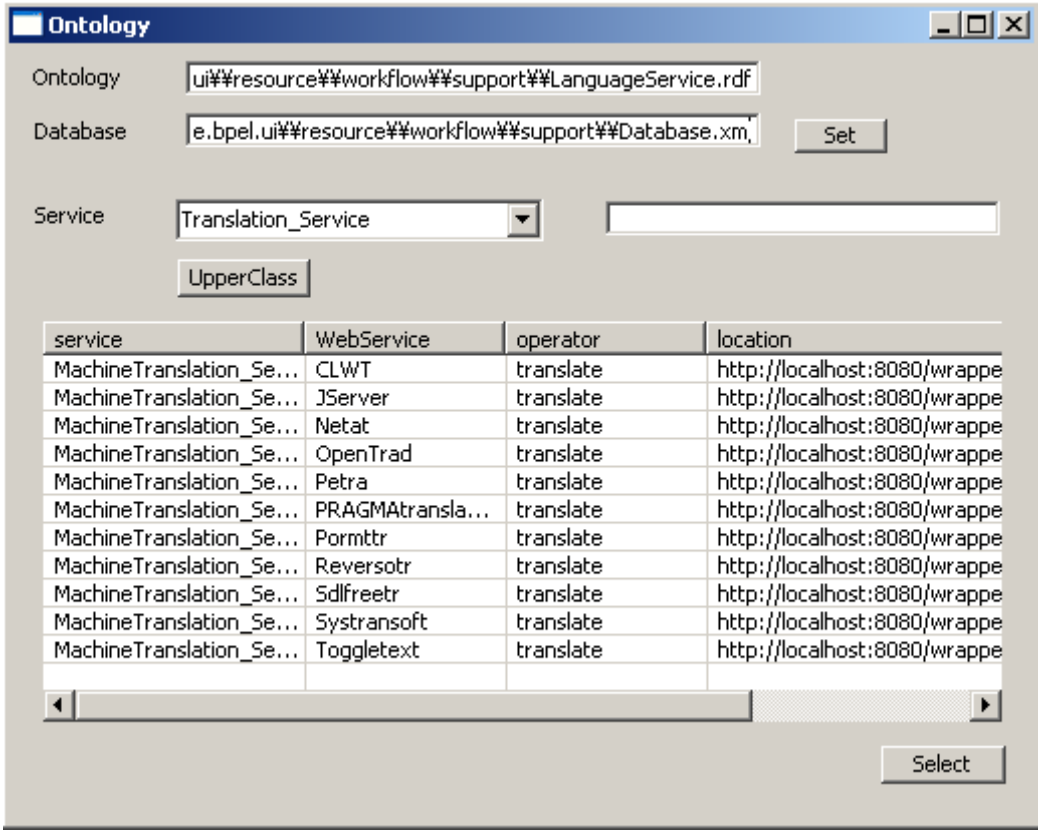

図 11: Translation_Service を実行する Web サービスの提案

Set というボタンをクリックすると，Service というプルダウンメニューにサービスの全種類が表示される．その中から，一つのサービスを選択すると，下のテーブルに，ユーザが指定したサービスを実現することができる Web サービスの一覧が表示される．ユーザはそこから，実行したい Web サービスを選択して Select ボタンをクリックすることで，BPEL にその Web サービスを反映することができる．

例えば図 11 では，図 2 で示したドメインオントロジを用いて，翻訳サービスを実行することのできる Web サービスの提案を行っている．ここで，UpperClass ボタンをクリックすることで，上位のクラスに範囲を広げて提案を行う．その様子を示したものが図 12 である．図 12 では，翻訳サービスの 1 つ上位の言語処理サービスをユーザが指定したとして，それを実行することができる Web サービスの提案を行っている．図 11 の提案と比べると，提案する Web サービスの数が増えていることがわかる．

次に，Invoke で実行する Web サービスの変更を行う際のシステムの動作を説

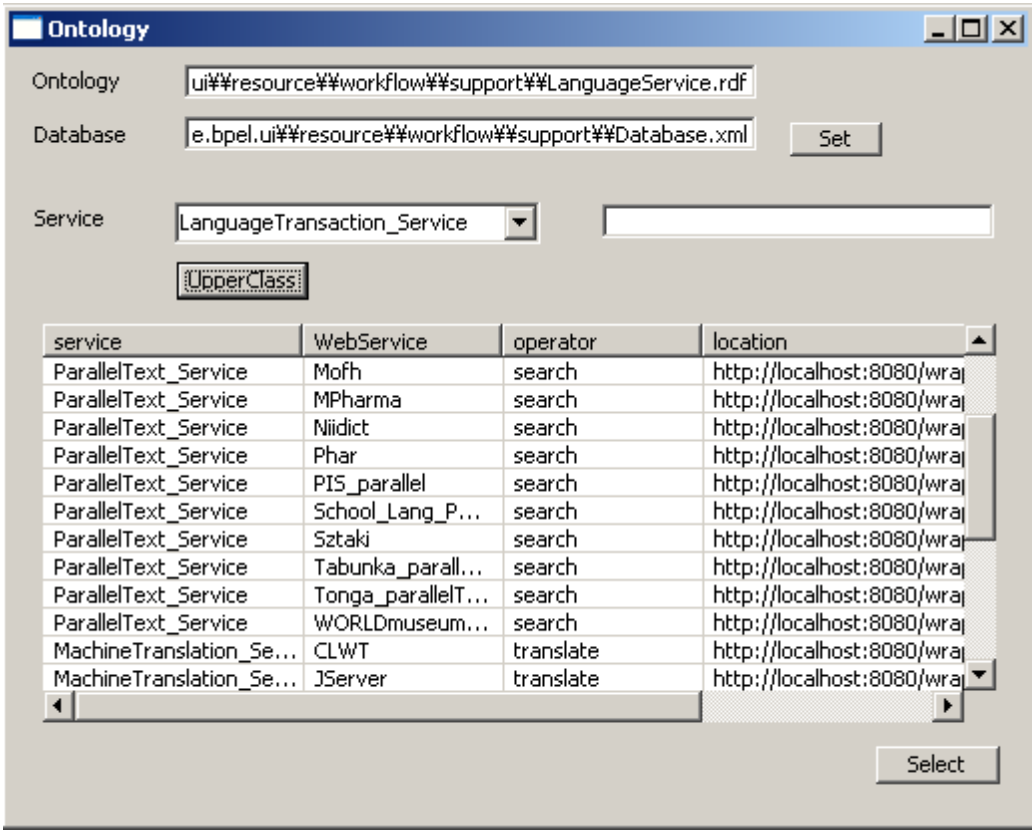

図 12: 範囲を広げた提案

明する．新しい Web サービスの提案を受けるまでは，先ほどの Invoke の追加の動作と同様にして行う．この際に，以前のサービスの種類が Service というプルダウンメニューの右側のボックスに表示される．ユーザがこのサービスと同じ種類のサービスを選択し Select ボタンをクリックした場合には，以前と同じ変数を用いて新しい Web サービスを実行するように BPEL に反映を行う．また，以前のサービスと異なる種類のサービスを選択して Select ボタンをクリックした際には，図 13 に示した Reassignment ウィンドウが出現する．ウィンドウの中のテーブルには変更を行う変数対の提案が表示される．I/O は入力か出力かを示していて，Previous には以前のサービスの変数を，New には変更するサービスの変数が示されている．図 13 では，以前の Web サービスで使用する入力メッセージの source，targetLang，sourceLang という変数の割り当てを，それぞれ新しい Web サービスの入力メッセージの source，targetLang，sourceLang への割り当て変更を提案している．また，以前の Web サービスの出力メッセージの translateReturn という変数を新しい Web サービスの searchReturn という

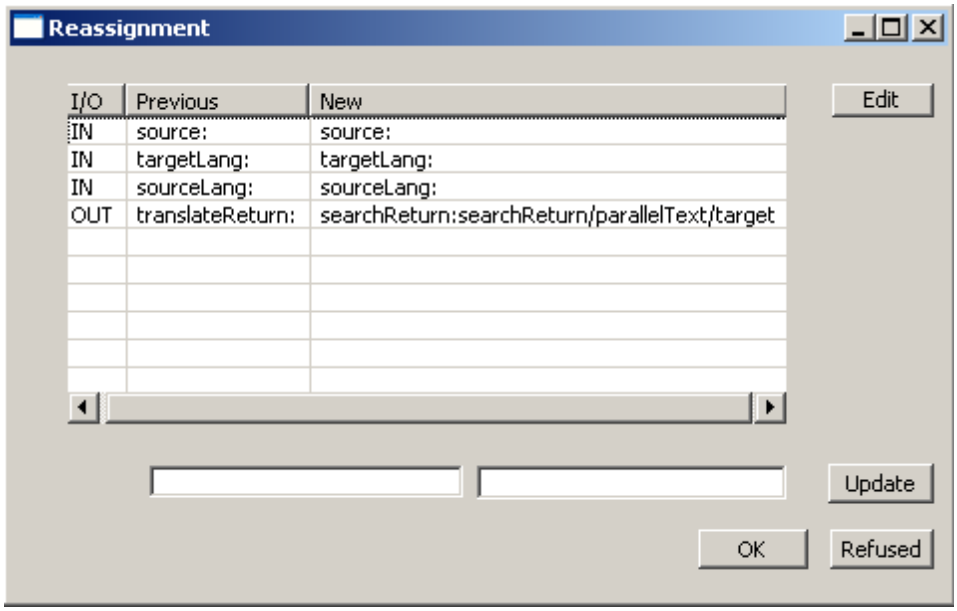

図 13: 変数割り当て変更の提案

パートの searchReturn/parallelText/target という変数への割り当てを変更している．

ユーザは，これらの一覧を見て，提案を受理するか拒否するかを判断する．その際に，ユーザは一覧から変更対を選択し，Edit ボタンをクリックすることで，下の 2 つのテキストボックスに変更対を表示させ，修正を行うことができる．修正が終わったら Update ボタンをクリックすることで，変更対の提案に変更を加えることができる．そして，提案の修正が終わったら，OK ボタンをクリックすることで，提案を確定し，Invoke で実行する Web サービスの変更と共に，ワークフロー中の Assign に対して，割り当ての変更を反映する．

第 5 章　おわりに

本研究では，Web サービス連携を半自動的に行う際に課題となる，以下の 2 点について議論を行った．

1. 多数の利用可能なサービスからの適切なサービスの選択

   ワークフローを作成する際に，サービス連携で実行する Web サービスを選択する必要がある．この際に，利用可能なサービスが多数存在する場合，その中から適切なサービスを選択することが必要となる．さらに，適切なサービスに実行可能な Web サービスが存在しない場合には，利用したいサービ

スに近いサービスを代用するが，その代用サービスを選択する必要がある．

2. ワークフロー中で実行する Web サービスを入れ替える際の，入出力変数の再割り当て

   サービス連携が期待通りの結果を出力するようにワークフローを修正する必要がある．その方法として，ワークフローの制御構造を変更する方法と，サービス連携の中で実行する Web サービスを変更する方法がある．Web サービスを変更して修正を行う場合には，Web サービスのインタフェースを確認する必要がある．同じインタフェースを持った Web サービスに変更を行う際には，入出力変数を変更する必要は無い．しかし，異なるインタフェースを持った Web サービスに変更を行う場合には，入出力の変数を変更する必要があり，それらの変数への割り当てを変更しなくてはならない．

以上の問題点を解決するために，本研究ではドメインオントロジを利用して，Web サービスの選択と Web サービス変更に伴う変数割り当ての変更を半自動化するアプローチを取った．ドメインオントロジ中のサービスの概念階層を用いてユーザが指定したサービスクラスの全てのサブクラスを探索し，それらのサービスを実現する Web サービスを検索することで，ユーザが実行したい Web サービスを提案する．また，適切な Web サービスが存在しない場合には，一つ上位のサービスを探索し，そのクラスのサブクラスを探索することで，範囲を広げた Web サービスの提案を行う．また，Web サービス変更に伴う入出力変数の変更に関しては，変更を行う可能性のある変数対のドメインオントロジ中での変数の概念階層グラフ上の距離を求め，その距離が小さいものを提案する変更対として計算を行う．

本研究における貢献は以下の 2 点である．

1. サービス連携中で実行する Web サービス選択の半自動化

   ドメインオントロジのサービス概念階層を用いて，ユーザが指定したサービスを実現する Web サービスの提案を行い，さらに，適切な Web サービスが存在しない場合には，一つ上位のクラスに探索範囲を広げることで，代用サービスの提案を行うアーキテクチャを構築した．それらの提案からユーザが選択した Web サービスを連携プロセスに反映するシステムを実装した．

2. Web サービスの入れ替えに伴う変数割り当て変更の半自動化

   ドメインオントロジの変数概念階層上の変数間の距離を用いることで，Web

サービスの変更に伴う変数割り当ての変更を提案するアーキテクチャを構築した．その提案をもとにユーザが修正を加えた割り当ての変更を連携プロセスに反映するシステムの実装を行った．

上記2点の貢献により，Webサービス選択，Webサービスを変更する際の変数割り当てに関して，半自動的に連携プロセスを作成することが可能となる．

今後の課題としては，本研究で構築したアーキテクチャにおけるシステムが行う提案の評価を行うことが挙げられる．また，Webサービスにおける前提条件や効果を含めたオントロジを用いることで，Webサービスの連携を容易にすることが考えられる．

謝辞

本研究を行うにあたり，熱心なご指導，ご助言を賜りました石田亨教授に厚く御礼申し上げます．また，日頃より，ご助言をいただきました独立行政法人情報通信研究機構の村上陽平氏に感謝いたします．そして，日頃から，さまざまなご助言をいただきました松原繁夫助教授はじめ，石田研究室の皆様に感謝いたします．

参考文献

[1] Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S. and Weerawarana, S.: Business Process Execution Language for Web Services. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

[2] Ishida, T.: Language Grid: An Infrastructure for Intercultual Collaboration, *IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06)*, pp. 96–100 (2006).

[3] Evren, S., James, H. and Bijan, P.: Semi-automatic Composition of Web Services using Semantic Descriptions, *In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003* (2003).

[4] Matthias, k., Benedikt, F. and Katia, S.: Automated Semantic Web Service Discovery with OWLS-MX, *Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (2006).

[5] Hayashi, Y.: Conceptual Framework of an Upper Ontology for Describing

Linguistic Services, *The First International Workshop on Intercultural Collaboration: IWIC 2007* (2007).

[6] Deborah, L., M. and Frank, van, H.: OWL Web Ontology Language Overview. W3C Recommendation, http://www.w3.org/TR/owl-features/ (2004).

付録

A.1 ソースコード

A.1.1 Support.java

```
package workflow.support;

import java.util.ArrayList;

import com.hp.hpl.jena.ontology.OntClass;

/**
 * ワークフロー作成支援を行うクラス
 * @author sakai
 */
public class Support {

    public DomainOntology domainOntology = null;
    public ArrayList<SearchWSRepository> repositoryList =
    new ArrayList<SearchWSRepository>();

    /**
     * コンストラクタ
     * @param domainOntologyFile ドメインオントロジのファイル名
     * @param repositoryFileList リポジトリファイルのファイル名の ArrayList<String>
     */
    public Support(String domainOntologyFile,
            ArrayList<String> repositoryFileList) throws Exception {

        try {

            domainOntology = new DomainOntology(domainOntologyFile);
            for (int i = 0; i < repositoryFileList.size(); i++) {
                SearchWSRepository wsr = new SearchWSRepository(
                        repositoryFileList.get(i));
                repositoryList.add(wsr);
            }
        } catch (Exception e) {
            throw e;
        }

    }

    /**
     * domainOntology の中に含まれるすべてのクラスの文字列を返す
     * @return ArrayList<String> AllClassesName
     */
    public ArrayList<String> getAllClasses() {

        ArrayList<String> AllClassesName = new ArrayList<String>();
        ArrayList<OntClass> AllClasses = domainOntology.searchAllClass();
        for (int i = 0; i < AllClasses.size(); i++) {
```

```
        AllClassesName.add(AllClasses.get(i).getLocalName());
    }
    return AllClassesName;
}

/**
 * domainOntology の中で、指定されたクラスのすべての
 * サブクラスの文字列を返します
 * @param rootClass サブクラスを調べたいクラス
 * @return ArrayList<String> SubClassesName
 */
public ArrayList<String> getSubClasses(String rootClass) {

    ArrayList<String> SubClassesName = new ArrayList<String>();
    ArrayList<OntClass> SubClasses = domainOntology
            .searchSubClass(rootClass);
    for (int i = 0; i < SubClasses.size(); i++) {
        SubClassesName.add(SubClasses.get(i).getLocalName());
    }
    return SubClassesName;
}

/**
 * スーパークラスを探索
 * @param serviceClass
 * @return
 */
public String getSuperClass(String serviceClass) {

    String superClassName = "";
    OntClass c = domainOntology.searchSuperClass(serviceClass);

    superClassName = c.getLocalName();

    return superClassName;

}

/**
 * クラスのリストに対応する ClassAndWSSet を返す
 * @param className
 * @return ArrayList<ClassAndWSSet> executableWSList
 */
public ArrayList<ClassAndWSSet> getWebServices(ArrayList<String> className) {

    ArrayList<ClassAndWSSet> executableWSList = new ArrayList<ClassAndWSSet>();
    for (int i = 0; i < repositoryList.size(); i++) {
        ArrayList<ClassAndWSSet> rl = repositoryList.get(i).search(
                className);
        executableWSList.addAll(rl);
    }
    return executableWSList;
}
```

```
/**
 * 指定されたクラスのすべてのサブクラスに対して、実現する WebService を返す
 * @param className
 * @return
 */
public ArrayList<ClassAndWSSet> getWSProposal(String className) {

    return getWebServices(getSubClasses(className));

}

/**
 * 変数変更の提案を返す
 * @param beforeServiceName
 * @param changeServiceName
 * @param InputOROutout
 * @return
 */
public ArrayList<VariableProposal> getVariablesPropasal(
        String beforeServiceName, String changeServiceName,
        String InputOROutout) {

    ArrayList<VariableProposal> proposal = new ArrayList<VariableProposal>();

    String IO = "Output";

    if (InputOROutout.equals("Input")) {
        IO = "Input";
    } else if (InputOROutout.equals("Output")) {
        IO = "Output";
    }

    OntClass beforeMessage = domainOntology.searchMessage(
            beforeServiceName, IO);
    OntClass proposalMessage = domainOntology.searchMessage(
            changeServiceName, IO);

    ArrayList<OntClass> beforeVariables = domainOntology
            .searchVariables(beforeMessage);
    ArrayList<OntClass> proposalVariables = domainOntology
            .searchVariables(proposalMessage);

    int dist = 0;
    while (beforeVariables.size() != 0 && proposalVariables.size() != 0) {

        OntClass delete1 = null;
        OntClass delete2 = null;

        dist = 10000;
        VariableProposal vp = null;
        for (int b = 0; b < beforeVariables.size(); b++) {
```

```
            for (int p = 0; p < proposalVariables.size(); p++) {

                if (domainOntology.distanceImpl(beforeVariables.get(b),
                        proposalVariables.get(p), 0, null) < dist) {

                    dist = domainOntology.distanceImpl(beforeVariables
                            .get(b), proposalVariables.get(p), 0, null);
                    vp = new VariableProposal(beforeMessage.getLocalName(),
                            beforeVariables.get(b).getLocalName(),
                            domainOntology.makeQuery(beforeMessage,
                                    beforeVariables.get(b)),
                            proposalMessage.getLocalName(),
                            proposalVariables.get(p).getLocalName(),
                            domainOntology.makeQuery(proposalMessage,
                                    proposalVariables.get(p)), IO);
                    delete1 = beforeVariables.get(b);
                    delete2 = proposalVariables.get(p);
                }
            }

        }
        proposal.add(vp);
        beforeVariables.remove(delete1);
        proposalVariables.remove(delete2);

    }

    return proposal;

}

/**
 * Web サービスデータベースからサービスクラスを検索
 * @param ServiceName
 * @return
 */
public String getServiceClass(String ServiceName) {
    String result = "";

    for (int i = 0; i < repositoryList.size(); i++) {
        result = repositoryList.get(i).searchServiceClass(ServiceName);
        if (!result.equals(""))
            return result;
    }

    return result;
}

/**
 * Web サービスデータベースから WSDL ファイルを検索
 * @param ServiceName
 * @return
 */
```

```
    public String getWSDLFile(String ServiceName) {
        String result = "";

        for (int i = 0; i < repositoryList.size(); i++) {
            result = repositoryList.get(i).searchWSDLFile(ServiceName);
            if (!result.equals(""))
                return result;
        }

        return result;
    }

}
```

A.1.2 DomainOntology.java

```
package workflow.support;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;

import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.Restriction;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;

/**
 * ドメインオントロジを利用したメソッド群
 * @author sakai
 */
public class DomainOntology {

    private OntModel serviceOnto;

    /**
     * @param file_name
     */
    public DomainOntology(String file_name) {

        serviceOnto = ModelFactory.createOntologyModel();
        InputStream in = null;

        try {

            in = new FileInputStream(file_name);

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
```

```
    serviceOnto.read(in, "");

}

/**
 * ドメインに含まれるすべてのクラスを返す
 * @return
 */
public ArrayList<OntClass> searchAllClass() {

    ArrayList<OntClass> result = new ArrayList<OntClass>();
    OntClass top = serviceOnto
            .getOntClass("http://www.owl-ontologies.com/Ontology1163589345.owl#Language_Service");
    result.add(top);
    for (Iterator each = top.listSubClasses(); each.hasNext();) {
        OntClass ss = serviceOnto.getOntClass(each.next().toString());
        result.add(ss);
    }

    return result;
}

/**
 * サービスクラスの文字列を与えると，オントロジを用いてサブクラスを探索する
 * @param rootClass サービスクラス
 * @return
 */
public ArrayList<OntClass> searchSubClass(String rootClass) {

    ArrayList<OntClass> result = new ArrayList<OntClass>();
    OntClass root = serviceOnto
            .getOntClass("http://www.owl-ontologies.com/Ontology1163589345.owl#"
                    + rootClass);
    result.add(root);

    for (Iterator each = root.listSubClasses(); each.hasNext();) {
        OntClass ss = serviceOnto.getOntClass(each.next().toString());
        result.add(ss);
    }

    return result;
}

/**
 * サービスクラスを与えると，スーパークラスを返す
 * @param className サービスクラス
 * @return
 */
public OntClass searchSuperClass(String className) {
    OntClass superClass = null;
    OntClass serviceClass = serviceOnto
            .getOntClass("http://www.owl-ontologies.com/Ontology1163589345.owl#"
                    + className);
```

```
    for (Iterator i = serviceClass.listSuperClasses(true); i.hasNext();) {
        OntClass c = (OntClass) i.next();

        if (!c.isUnionClass() && !c.isIntersectionClass()
                && !c.isComplementClass() && !c.isRestriction()
                && !c.isAnon()) {
            // superClass =
            // c.getModel().getGraph().getPrefixMapping()
            //.shortForm(c.getURI());
            superClass = c;
            return superClass;
        }

    }

    return superClass;

}

/**
 * サービスクラスと入出力のどちらかを与えると,
 * そのサービスが利用するサービスタイプを返す
 * @param serviceName サービスクラス
 * @param inOrOut Input または Output
 * @return
 */
public OntClass searchMessage(String serviceName, String inOrOut) {

    OntClass serviceClass = serviceOnto
            .getOntClass("http://www.owl-ontologies.com/Ontology1163589345.owl#"
                    + serviceName);

    OntClass message = null;

    for (Iterator i = serviceClass.listSuperClasses(true); i.hasNext();) {
        OntClass c = (OntClass) i.next();
        if (c.isRestriction() && checkProperty(c.asRestriction(), inOrOut)) {

            if (c.asRestriction().isAllValuesFromRestriction()) {
                message = (OntClass) c.asRestriction()
                        .asAllValuesFromRestriction().getAllValuesFrom()
                        .as(OntClass.class);
            }
            if (c.asRestriction().isSomeValuesFromRestriction()) {
                message = (OntClass) c.asRestriction()
                        .asSomeValuesFromRestriction().getSomeValuesFrom()
                        .as(OntClass.class);
            }
            if (c.asRestriction().isHasValueRestriction()) {
                message = (OntClass) c.asRestriction()
                        .asHasValueRestriction().getHasValue().as(
                                OntClass.class);
```

```
            }

        }
    }

    return message;
}

/**
 * メッセージクラスの中に含まれる、文字列の変数を探し出す。
 * 文字列の変数の ArrayList を返す
 * @param messageClass
 * @return
 */
public ArrayList<OntClass> searchVariables(OntClass messageClass) {
    ArrayList<OntClass> result = new ArrayList<OntClass>();

    for (Iterator i = messageClass.listSuperClasses(true); i.hasNext();) {
        OntClass c = (OntClass) i.next();
        if (c.isRestriction() && checkProperty(c.asRestriction(), "Part")) {

            if (c.asRestriction().isAllValuesFromRestriction()) {
                OntClass oc = (OntClass) c.asRestriction()
                        .asAllValuesFromRestriction().getAllValuesFrom()
                        .as(OntClass.class);
                result.addAll(searchStringORStrings(oc));
            }
            if (c.asRestriction().isSomeValuesFromRestriction()) {
                OntClass oc = (OntClass) c.asRestriction()
                        .asSomeValuesFromRestriction().getSomeValuesFrom()
                        .as(OntClass.class);
                result.addAll(searchStringORStrings(oc));
            }
            if (c.asRestriction().isHasValueRestriction()) {
                OntClass oc = (OntClass) c.asRestriction()
                        .asHasValueRestriction().getHasValue().as(
                                OntClass.class);
                result.addAll(searchStringORStrings(oc));
            }

        }
    }

    return result;
}

/**
 * restriction のプロパティ名が prop と等しいかを判断する
 * @param restriction
 * @param prop
 * @return
 */
public static boolean checkProperty(Restriction restriction, String prop) {
```

```
    boolean result = false;

    Resource resource = (Resource) restriction.getOnProperty();

    if (resource.getModel().shortForm(resource.getURI()).equals(":" + prop))
        result = true;

    return result;

}

/**
 * variable のクラスに Array プロパティが存在するかを調べる。
 * 存在する場合、Array プロパティの指すクラスを返し、
 * 存在しない場合は、null を返す。
 * @param variableClass
 * @return
 */
public static OntClass checkArray(OntClass variableClass) {
    OntClass arrayClass = null;

    for (Iterator i = variableClass.listSuperClasses(true); i.hasNext();) {
        OntClass c = (OntClass) i.next();
        if (c.isRestriction() && checkProperty(c.asRestriction(), "Array")) {

            if (c.asRestriction().isAllValuesFromRestriction()) {
                arrayClass = (OntClass) c.asRestriction()
                        .asAllValuesFromRestriction().getAllValuesFrom()
                        .as(OntClass.class);
            }
            if (c.asRestriction().isSomeValuesFromRestriction()) {
                arrayClass = (OntClass) c.asRestriction()
                        .asSomeValuesFromRestriction().getSomeValuesFrom()
                        .as(OntClass.class);
            }
            if (c.asRestriction().isHasValueRestriction()) {
                arrayClass = (OntClass) c.asRestriction()
                        .asHasValueRestriction().getHasValue().as(
                                OntClass.class);
            }
        }
    }

    return arrayClass;
}

/**
 * variable のクラスに Element プロパティが存在するかを調べる。
 * 存在する場合、Element プロパティの指すクラスの ArrayList を返し、
 * 存在しない場合は、null を返す。
 * @param variableClass
 * @return
```

```
 */
public static ArrayList<OntClass> checkElement(OntClass variableClass) {
    ArrayList<OntClass> elementClasses = new ArrayList<OntClass>();

    for (Iterator i = variableClass.listSuperClasses(true); i.hasNext();) {
        OntClass c = (OntClass) i.next();
        if (c.isRestriction()
                && checkProperty(c.asRestriction(), "Element")) {

            if (c.asRestriction().isAllValuesFromRestriction()) {
                elementClasses.add((OntClass) c.asRestriction()
                        .asAllValuesFromRestriction().getAllValuesFrom()
                        .as(OntClass.class));
            }
            if (c.asRestriction().isSomeValuesFromRestriction()) {
                elementClasses.add((OntClass) c.asRestriction()
                        .asSomeValuesFromRestriction().getSomeValuesFrom()
                        .as(OntClass.class));
            }
            if (c.asRestriction().isHasValueRestriction()) {
                elementClasses.add((OntClass) c.asRestriction()
                        .asHasValueRestriction().getHasValue().as(
                                OntClass.class));
            }
        }
    }

    return elementClasses;
}

/**
 * 文字列または，文字列のクラスを探す
 * @param variableClass
 * @return
 */
public static ArrayList<OntClass> searchStringORStrings(
        OntClass variableClass) {
    ArrayList<OntClass> stringClass = new ArrayList<OntClass>();

    if (checkArray(variableClass) != null) {
        stringClass
                .addAll(searchStringORStrings(checkArray(variableClass)));
    }

    if (checkElement(variableClass).size() != 0) {
        for (int i = 0; i < checkElement(variableClass).size(); i++) {
            stringClass.addAll(searchStringORStrings(checkElement(
                    variableClass).get(i)));
        }
    }

    if (checkArray(variableClass) == null
            && checkElement(variableClass).size() == 0) {
```

```
        stringClass.add(variableClass);
    }

    return stringClass;
}

/**
 * クエリを作成する
 * @param messageClass
 * @param to
 * @return
 */
public String makeQuery(OntClass messageClass, OntClass to) {
    String result = "";

    for (Iterator i = messageClass.listSuperClasses(true); i.hasNext();) {
        OntClass c = (OntClass) i.next();
        if (c.isRestriction() && checkProperty(c.asRestriction(), "Part")) {

            if (c.asRestriction().isAllValuesFromRestriction()) {
                OntClass oc = (OntClass) c.asRestriction()
                        .asAllValuesFromRestriction().getAllValuesFrom()
                        .as(OntClass.class);
                result = makeQueryImpl(oc, to);
                if (result != null) {
                    if (result.indexOf("_") != -1) {
                        result = result.substring(result.indexOf("_") + 1);
                    }
                    return result;
                }
            }
            if (c.asRestriction().isSomeValuesFromRestriction()) {
                OntClass oc = (OntClass) c.asRestriction()
                        .asSomeValuesFromRestriction().getSomeValuesFrom()
                        .as(OntClass.class);
                result = makeQueryImpl(oc, to);
                if (result != null) {
                    if (result.indexOf("_") != -1) {
                        result = result.substring(result.indexOf("_") + 1);
                    }
                    return result;
                }
            }
            if (c.asRestriction().isHasValueRestriction()) {
                OntClass oc = (OntClass) c.asRestriction()
                        .asHasValueRestriction().getHasValue().as(
                                OntClass.class);
                result = makeQueryImpl(oc, to);
                if (result != null) {
                    if (result.indexOf("_") != -1) {
                        result = result.substring(result.indexOf("_") + 1);
                    }
                    return result;
```

```
                }
            }

        }
    }

    return result;
}

/**
 * クエリの作成
 * @param from
 * @param to
 * @return
 */
public static String makeQueryImpl(OntClass from, OntClass to) {

    if (from.equals(to)) {
        return from.getLocalName();
    } else {

        if (checkArray(from) != null) {
            if (makeQueryImpl(checkArray(from), to) != null) {
                return from.getLocalName() + "/"
                        + makeQueryImpl(checkArray(from), to);
            }

        }

        if (checkElement(from).size() != 0) {
            for (int i = 0; i < checkElement(from).size(); i++) {
                if (makeQueryImpl(checkElement(from).get(i), to) != null) {
                    return from.getLocalName() + "/"
                            + makeQueryImpl(checkElement(from).get(i), to);
                }
            }
        }

    }

    return null;
}

/**
 * 変数の距離を求めるメソッド
 * @param oc
 * @param goal
 * @param d
 * @param from
 * @return
 */
public static int distanceImpl(OntClass oc, OntClass goal, int d,
        OntClass from) {
```

```
        if (oc.equals(goal))
            return 0;

        for (Iterator i = oc.listSuperClasses(true); i.hasNext();) {
            OntClass c = (OntClass) i.next();

            if (!c.equals(from)
                    && !c.isUnionClass()
                    && !c.isIntersectionClass()
                    && !c.isComplementClass()
                    && !c.isRestriction()
                    && !c.isAnon()
                    && !c.getModel().shortForm(c.getURI()).equals(
                            "rdfs:Resource")) {

                int e = distanceImpl(c, goal, d, oc);
                if (e >= 0)
                    return e + 1;
            }

        }

        for (Iterator i = oc.listSubClasses(true); i.hasNext();) {
            OntClass c = (OntClass) i.next();

            if (!c.equals(from)
                    && !c.isUnionClass()
                    && !c.isIntersectionClass()
                    && !c.isComplementClass()
                    && !c.isRestriction()
                    && !c.isAnon()
                    && !c.getModel().shortForm(c.getURI()).equals(
                            "rdfs:Resource")) {

                int e = distanceImpl(c, goal, d, oc);
                if (e >= 0)
                    return e + 1;
            }

        }

        return -1;
    }

}
```

A.1.3 SearchWSRepository.java

```
package workflow.support;

import java.io.File;
import java.util.ArrayList;
```

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntClass;

/**
 * Web サービスデータベースを利用して， Web サービスの情報を検索するクラス
 *
 * @author s-sakai
 *
 */
public class SearchWSRepository {

    private String repository;
    private DocumentBuilderFactory dbfactory;
    private DocumentBuilder builder;
    private Document doc;
    private Element root;

    /**
     * @param repositoryFile
     *            repository の場所を指定するコンストラクタ
     */
    public SearchWSRepository(String repositoryFile) throws Exception {
        try {
            repository = repositoryFile;
            // ドキュメントビルダーファクトリを生成
            dbfactory = DocumentBuilderFactory.newInstance();
            // ドキュメントビルダーを生成
            builder = dbfactory.newDocumentBuilder();
            // パースを実行して Document オブジェクトを取得
            doc = builder.parse(new File(repositoryFile));
            // ルート要素を取得
            root = doc.getDocumentElement();

        } catch (Exception e) {
            throw e;
        }
    }

    /**
     * サービスのクラスの ArrayList を元に、Web サービスのデータベースを検索し、
     * そのクラスとそのクラスを実現する
     * サービスのリストの集合 ClassAndWSSet を返すメソッド
     * @param classList
     * @return ArrayList<ClassAndWSSet>
     */
    public ArrayList<ClassAndWSSet> search(ArrayList<String> classList) {
```

```
    ArrayList<ClassAndWSSet> result = new ArrayList<ClassAndWSSet>();
    NodeList nl = root.getElementsByTagName("WebService");

    for (int i = 0; i < classList.size(); i++) {
        String className = classList.get(i);
        ArrayList<WSData> list = new ArrayList<WSData>();

        for (int s = 0; s < nl.getLength(); s++) {

            Element service = (Element) nl.item(s);

            if (service.getElementsByTagName("ServiceClass").item(0)
                    .getFirstChild().getNodeValue().equals(className)) {
                String name = service.getElementsByTagName("Name").item(0)
                        .getFirstChild().getNodeValue();
                String serviceClass = service.getElementsByTagName(
                        "ServiceClass").item(0).getFirstChild()
                        .getNodeValue();
                String wsdl = service.getElementsByTagName("WSDL").item(0)
                        .getFirstChild().getNodeValue();
                String operation = service.getElementsByTagName(
                        "OperationName").item(0).getFirstChild()
                        .getNodeValue();
                String location = service.getElementsByTagName("Location")
                        .item(0).getFirstChild().getNodeValue();
                // String location = "";

                list.add(new WSData(name, serviceClass, wsdl, operation,
                        location));

            }

        }

        ClassAndWSSet set = new ClassAndWSSet(classList.get(i), list);

        result.add(set);
    }

    return result;
}

/**
 * ある Web サービスのサービスクラスを検索する
 *
 * @param ServiceName
 * @return
 */
public String searchServiceClass(String ServiceName) {

    String result = "";
    NodeList nl = root.getElementsByTagName("WebService");
```

```
        for (int s = 0; s < nl.getLength(); s++) {

            Element service = (Element) nl.item(s);

            if (service.getElementsByTagName("Name").item(0).getFirstChild()
                    .getNodeValue().equals(ServiceName)) {

                return service.getElementsByTagName("ServiceClass").item(0)
                        .getFirstChild().getNodeValue();

            }

        }

        return result;
    }

    /**
     * ある Web サービスの WSDL ファイルを検索する
     *
     * @param webServiceName
     * @return
     */
    public String searchWSDLFile(String webServiceName) {

        String wsdlFileUrl = "";

        NodeList nl = root.getElementsByTagName("WebService");

        for (int s = 0; s < nl.getLength(); s++) {

            Element service = (Element) nl.item(s);

            if (service.getElementsByTagName("Name").item(0).getFirstChild()
                    .getNodeValue().equals(webServiceName)) {

                return service.getElementsByTagName("WSDL").item(0)
                        .getFirstChild().getNodeValue();

            }

        }

        return wsdlFileUrl;
    }

}
```

A.1.4 OntologyWindow.java

```
package workflow.support;
```

```
import java.util.ArrayList;

import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.swt.widgets.TableItem;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.graphics.Rectangle;
import org.eclipse.swt.widgets.Text;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.Combo;


/**
 * OntologyWindow  ユーザとインタラクションを行う GUI
 * @author sakai
 *
 */
public class OntologyWindow {

    private Shell sShell = null;
    private Label l_ontology = null;
    private Text t_ontology = null;
    private Label l_service = null;
    private Button b_show = null;
    private Button b_select = null;
    private Label l_database = null;
    private Text t_database = null;
    private Table table = null;
    private Button b_upper = null;
    private String ontology = "";
    private String database = "";
    private String serviceName = "";  //  @jve:decl-index=0:
    private Support support;
    private ArrayList<String> sub;
    private ArrayList<ClassAndWSSet> cawss;
    private Combo combo_service = null;
    private Button b_setservice = null;
    private Text t_preservice = null;
    private String preServiceName = "";
    private String preServiceClass = "";
    private boolean sameClass = false;
    private String newServiceLocation = "";
    private String newServiceOperation = "";
    private String newServiceWSDLFile = "";
    private boolean hasVariable = false;
    private ArrayList<VariablePair> variableResult = null;
    private ArrayList<VariableProposal> proposalResult = null;
```

```
/**
 * コンストラクタ
 * @param hasVariable 変数が定義されていたら true
 */
public OntologyWindow(boolean hasVariable) {

    this.hasVariable = hasVariable;

}

/**
 * @param ServiceName
 */
public void popupOntologyWindow(String ServiceName) {

    preServiceName = ServiceName;

    Display display = Display.getDefault();
    createSShell();
    sShell.open();

    while (!sShell.isDisposed()) {
        if (!display.readAndDispatch())
            display.sleep();
    }

    return;
}

/**
 * This method initializes sShell
 */
private void createSShell() {

    sShell = new Shell(SWT.APPLICATION_MODAL | SWT.SHELL_TRIM);
    sShell.setText("Ontology");
    sShell.setSize(new Point(518, 415));
    sShell.setLayout(null);
    l_ontology = new Label(sShell, SWT.NONE);
    l_ontology.setBounds(new Rectangle(12, 9, 62, 13));
    l_ontology.setText("Ontology");
    t_ontology = new Text(sShell, SWT.BORDER);
    t_ontology.setBounds(new Rectangle(88, 8, 286, 17));
    l_database = new Label(sShell, SWT.NONE);
    l_database.setBounds(new Rectangle(13, 36, 55, 13));
    l_database.setText("Database");
    t_database = new Text(sShell, SWT.BORDER);
    t_database.setBounds(new Rectangle(88, 34, 286, 17));
    l_service = new Label(sShell, SWT.NONE);
    l_service.setBounds(new Rectangle(13, 78, 58, 15));
    l_service.setText("Service");
```

```
b_show = new Button(sShell, SWT.NONE);
b_upper = new Button(sShell, SWT.NONE);
b_upper.setBounds(new Rectangle(84, 107, 66, 19));
b_upper.setText("UpperClass");
b_select = new Button(sShell, SWT.NONE);
b_select.setBounds(new Rectangle(434, 350, 62, 19));
b_select.setText("Select");
table = new Table(sShell, SWT.MULTI | SWT.FULL_SELECTION);
table.setHeaderVisible(true);
table.setLinesVisible(true);
table.setBounds(new Rectangle(17, 138, 477, 200));

createCombo_service();
b_setservice = new Button(sShell, SWT.NONE);
b_setservice.setBounds(new Rectangle(391, 37, 46, 17));
b_setservice.setText("Set");
t_preservice = new Text(sShell, SWT.BORDER);
t_preservice.setBounds(new Rectangle(297, 78, 196, 17));
TableColumn col1 = new TableColumn(table, SWT.LEFT);

//set ボタンのアクション
b_setservice
        .addSelectionListener(new org.eclipse.swt.events
        .SelectionAdapter() {
            public void widgetSelected(
                    org.eclipse.swt.events.SelectionEvent e) {
                try {

                    ontology = t_ontology.getText();
                    database = t_database.getText();

                    ArrayList<String> repositoryList =
                    new ArrayList<String>();
                    repositoryList.add(database);

                    support = new Support(ontology, repositoryList);

                    preServiceClass = support
                            .getServiceClass(preServiceName);

                    t_preservice.setText(preServiceClass);

                    ArrayList<String> strs = support.getAllClasses();
                    for (int i = 0; i < strs.size(); i++) {
                        combo_service.add(strs.get(i));
                    }
                    combo_service.select(0);

                } catch (Exception e1) {

                }
            }
        });
```

```
//プルダウンテキストのアクション
combo_service
        .addSelectionListener(new org.eclipse.swt.events
        .SelectionListener() {
            public void widgetSelected(
                    org.eclipse.swt.events.SelectionEvent e) {

                try {

                    //TableItem item = new TableItem(table,SWT.NULL);
                    ontology = t_ontology.getText();
                    database = t_database.getText();
                    serviceName = combo_service.getText();

                    table.removeAll();
                    ArrayList<String> repositoryList =
                    new ArrayList<String>();
                    repositoryList.add(database);

                    support = new Support(ontology, repositoryList);

                    sub = support.getSubClasses(serviceName);

                    cawss = support.getWebServices(sub);

                    for (int i = 0; i < cawss.size(); i++) {

                        for (int s = 0; s < cawss.get(i).getWsList()
                                .size(); s++) {

                            TableItem item = new TableItem(table,
                                    SWT.NULL);
                            String[] data = {
                                    cawss.get(i).getWsList().get(s)
                                            .getServiceClass(),
                                    cawss.get(i).getWsList().get(s)
                                            .getName(),
                                    cawss.get(i).getWsList().get(s)
                                            .getOperationName(),
                                    cawss.get(i).getWsList().get(s)
                                            .getLocation()

                            };
                            item.setText(data);
                        }

                    }

                } catch (Exception e1) {

                    e1.printStackTrace();
                }
```

```
            }

            public void widgetDefaultSelected(
                    org.eclipse.swt.events.SelectionEvent e) {
            }
        });

//show ボタンのアクション
b_show.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e) {

        try {

            ontology = t_ontology.getText();
            database = t_database.getText();
            serviceName = combo_service.getText();

            table.removeAll();
            ArrayList<String> repositoryList = new ArrayList<String>();
            repositoryList.add(database);

            support = new Support(ontology, repositoryList);

            sub = support.getSubClasses(serviceName);

            cawss = support.getWebServices(sub);

            for (int i = 0; i < cawss.size(); i++) {

                for (int s = 0; s < cawss.get(i).getWsList().size(); s++) {

                    TableItem item = new TableItem(table, SWT.NULL);
                    String[] data = {
                            cawss.get(i).getWsList().get(s)
                                    .getServiceClass(),
                            cawss.get(i).getWsList().get(s).getName(),
                            "" };
                    item.setText(data);
                }

            }

        } catch (Exception e1) {
            e1.printStackTrace();
        }

    }
});

//upper ボタンのアクション
b_upper
        .addSelectionListener(new org.eclipse.swt.events
```

```
        .SelectionAdapter() {
            public void widgetSelected(
                    org.eclipse.swt.events.SelectionEvent e) {

                try {
                    serviceName = support.getSuperClass(serviceName);
                    combo_service.setText(serviceName);
                    table.removeAll();
                    ArrayList<String> repositoryList =
                    new ArrayList<String>();
                    repositoryList.add(database);

                    support = new Support(ontology, repositoryList);

                    sub = support.getSubClasses(serviceName);

                    cawss = support.getWebServices(sub);

                    for (int i = 0; i < cawss.size(); i++) {

                        for (int s = 0; s < cawss.get(i).getWsList()
                                .size(); s++) {

                            TableItem item = new TableItem(table,
                                    SWT.NULL);
                            String[] data = {
                                    cawss.get(i).getWsList().get(s)
                                            .getServiceClass(),
                                    cawss.get(i).getWsList().get(s)
                                            .getName(),
                                    cawss.get(i).getWsList().get(s)
                                            .getOperationName(),
                                    cawss.get(i).getWsList().get(s)
                                            .getLocation()

                            };
                            item.setText(data);
                        }

                    }

                } catch (Exception e1) {
                    e1.printStackTrace();
                }

            }
        });

//select ボタンのアクション
b_select
        .addSelectionListener(new org.eclipse.swt.events
        .SelectionAdapter() {
            public void widgetSelected(
```

```
                    org.eclipse.swt.events.SelectionEvent e) {
                TableItem[] serviceInfo = table.getSelection();

                serviceName = serviceInfo[0].getText(1);
                newServiceLocation = serviceInfo[0].getText(3);
                newServiceOperation = serviceInfo[0].getText(2);
                newServiceWSDLFile = support.getWSDLFile(serviceName);

                if (hasVariable) {

                    if (!preServiceClass.equals(serviceInfo[0]
                            .getText(0))) {

                        sameClass = false;

                        ArrayList<VariablePair> vp =
                        new ArrayList<VariablePair>();
                        ArrayList<VariableProposal> proposal = null;

                        proposal = support.getVariablesPropasal(
                                t_preservice.getText(), serviceInfo[0]
                                        .getText(0), "Input");

                        proposal.addAll(support.getVariablesPropasal(
                                t_preservice.getText(), serviceInfo[0]
                                        .getText(0), "Output"));

                        ReassignmentWindow sp = new ReassignmentWindow();
                        sp.popUpReassignmentWindow(proposal);

                        proposalResult = sp.getList();

                    } else {
                        sameClass = true;

                    }

                } else {

                }

                sShell.dispose();
            }
        });

String[] cols = { "service", "WebService", "operator", "location" };
col1.setText(cols[0]);
col1.setWidth(130);
TableColumn col2 = new TableColumn(table, SWT.LEFT);
col2.setText(cols[1]);
col2.setWidth(100);
TableColumn col3 = new TableColumn(table, SWT.LEFT);
col3.setText(cols[2]);
```

```
    col3.setWidth(100);
    TableColumn col4 = new TableColumn(table, SWT.LEFT);
    col4.setText(cols[3]);
    col4.setWidth(300);

}

/**
 * This method initializes combo_service
 *
 */
private void createCombo_service() {
    combo_service = new Combo(sShell, SWT.NONE);
    combo_service.setBounds(new Rectangle(83, 77, 183, 17));

}

public String getServiceName() {
    return serviceName;
}

public void setServiceName(String serviceName) {
    this.serviceName = serviceName;
}

public boolean isSameClass() {
    return sameClass;
}

public void setSameClass(boolean sameClass) {
    this.sameClass = sameClass;
}

public String getNewServiceLocation() {
    return newServiceLocation;
}

public void setNewServiceLocation(String newServiceLocation) {
    this.newServiceLocation = newServiceLocation;
}

public String getNewServiceOperation() {
    return newServiceOperation;
}

public void setNewServiceOperation(String newServiceOperation) {
    this.newServiceOperation = newServiceOperation;
}

public String getNewServiceWSDLFile() {
    return newServiceWSDLFile;
}
```

```
    public void setNewServiceWSDLFile(String newServiceWSDLFile) {
        this.newServiceWSDLFile = newServiceWSDLFile;
    }

    public ArrayList<VariablePair> getVariableResult() {
        return variableResult;
    }

    public void setVariableResult(ArrayList<VariablePair> variableResult) {
        this.variableResult = variableResult;
    }

    public ArrayList<VariableProposal> getProposalResult() {
        return proposalResult;
    }

    public void setProposalResult(ArrayList<VariableProposal> proposalResult) {
        this.proposalResult = proposalResult;
    }




}
```

A.1.5 ReassignmentWindow.java

```
package workflow.support;

import java.util.ArrayList;

import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.swt.widgets.TableItem;
import org.eclipse.swt.SWT;
import org.eclipse.swt.graphics.Rectangle;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Text;

/**
 * ReassignmentWindow ユーザとインタラクションを行う GUI
 *
 * @author sakai
 *
 */
public class ReassignmentWindow {

    private Shell sShell = null;
    private Table table_reassignment = null;
    private Button b_edit = null;
```

```
private Text t_preassign = null;
private Text t_followassign = null;
private Button b_update = null;
private Button b_ok = null;
private Button b_refused = null;
private ArrayList<VariableProposal> list = null; // @jve:decl-index=0:
private String preInputMessage = "";
private String preOutputMessage = "";
private String newInputMessage = "";
private String newOutputMessage = "";

/**
 * ReassignmentWindow を起動する
 *
 * @param variablePairList
 */
public void popUpReassignmentWindow(
        ArrayList<VariableProposal> variablePairList) {

    list = variablePairList;

    Display display = Display.getDefault();

    createSShell();
    sShell.open();

    while (!sShell.isDisposed()) {
        if (!display.readAndDispatch())
            display.sleep();
    }

}

/**
 * This method initializes sShell
 */
private void createSShell() {
    sShell = new Shell(SWT.APPLICATION_MODAL | SWT.SHELL_TRIM);
    sShell.setText("Reassignment");
    sShell.setSize(new Point(478, 304));
    sShell.setLayout(null);
    table_reassignment = new Table(sShell, SWT.FULL_SELECTION);
    table_reassignment.setHeaderVisible(true);
    table_reassignment.setLinesVisible(true);
    table_reassignment.setBounds(new Rectangle(29, 18, 360, 171));

    String[] cols = { "I/O", "Previous", "New" };
    TableColumn col1 = new TableColumn(table_reassignment, SWT.LEFT);
    TableColumn col2 = new TableColumn(table_reassignment, SWT.LEFT);
    TableColumn col3 = new TableColumn(table_reassignment, SWT.LEFT);
    col1.setText(cols[0]);
    col1.setWidth(40);
    col2.setText(cols[1]);
```

```
col2.setWidth(160);
col3.setText(cols[2]);
col3.setWidth(160);

for (int s = 0; s < list.size(); s++) {

    TableItem item = new TableItem(table_reassignment, SWT.NULL);
    String io = "";
    if (list.get(s).getDirection().equals("Input")) {
        io = "IN";
        preInputMessage = list.get(s).getBefore().getMessage();
        newInputMessage = list.get(s).getProposal().getMessage();
    } else {
        io = "OUT";
        preOutputMessage = list.get(s).getBefore().getMessage();
        newOutputMessage = list.get(s).getProposal().getMessage();

    }
    String[] data = {
            io,
            list.get(s).getBefore().getVariable() + ":"
                    + list.get(s).getBefore().getQuery(),
            list.get(s).getProposal().getVariable() + ":"
                    + list.get(s).getProposal().getQuery() };
    item.setText(data);
}
b_edit = new Button(sShell, SWT.NONE);
b_edit.setBounds(new Rectangle(412, 18, 51, 17));
b_edit.setText("Edit");

t_preassign = new Text(sShell, SWT.BORDER);
t_preassign.setBounds(new Rectangle(70, 210, 157, 16));
t_followassign = new Text(sShell, SWT.BORDER);
t_followassign.setBounds(new Rectangle(234, 210, 155, 17));
b_update = new Button(sShell, SWT.NONE);
b_update.setBounds(new Rectangle(410, 209, 53, 20));
b_update.setText("Update");

b_ok = new Button(sShell, SWT.NONE);
b_ok.setBounds(new Rectangle(345, 242, 55, 20));
b_ok.setText("OK");

b_refused = new Button(sShell, SWT.NONE);
b_refused.setBounds(new Rectangle(411, 242, 52, 20));
b_refused.setText("Refused");

// edit ボタンのアクション
b_edit
        .addSelectionListener(new org.eclipse.swt.events
        .SelectionAdapter() {
            public void widgetSelected(
                    org.eclipse.swt.events.SelectionEvent e) {
```

```
                TableItem[] serviceInfo = table_reassignment
                        .getSelection();
                t_preassign.setText(serviceInfo[0].getText(1));
                t_followassign.setText(serviceInfo[0].getText(2));
            }
        });

// update ボタンのアクション
b_update
        .addSelectionListener(new org.eclipse.swt.events
        .SelectionAdapter() {
            public void widgetSelected(
                    org.eclipse.swt.events.SelectionEvent e) {
                TableItem[] items = table_reassignment.getItems();
                int selectionIndex = table_reassignment
                        .getSelectionIndex(); // 選択されている行位置
                String[] data = { items[selectionIndex].getText(0),
                        t_preassign.getText(), t_followassign.getText() };
                // TableItem 配列を取り出す
                if (selectionIndex == -1) {

                    return; // 何もしない
                } else {
                    String preVariable = "";
                    String preQuery = "";
                    String newVariable = "";
                    String newQuery = "";

                    if (t_preassign.getText().split(":").length == 1) {
                        preVariable = t_preassign.getText().split(":")[0];
                    } else if (t_preassign.getText().split(":").length == 2) {
                        preVariable = t_preassign.getText().split(":")[0];
                        preQuery = t_preassign.getText().split(":")[1];
                    }

                    if (t_followassign.getText().split(":").length == 1) {
                        newVariable = t_followassign.getText().split(
                                ":")[0];
                    } else if (t_followassign.getText().split(":").length == 2) {
                        newVariable = t_followassign.getText().split(
                                ":")[0];
                        newQuery = t_followassign.getText().split(":")[1];
                    }

                    if (items[selectionIndex].getText(0).equals("IN")) {
                        list.set(selectionIndex, new VariableProposal(
                                preInputMessage, preVariable, preQuery,
                                newInputMessage, newVariable, newQuery,
                                "Input"));
                    } else {
                        list.set(selectionIndex, new VariableProposal(
                                preOutputMessage, preVariable,
                                preQuery, newOutputMessage,
```

```
                                        newVariable, newQuery, "Output"));
                            }

                                items[selectionIndex].setText(data);
                            }

                        }
                    });

            // ok ボタンのアクション
            b_ok
                    .addSelectionListener(new org.eclipse.swt.events
                    .SelectionAdapter() {
                        public void widgetSelected(
                                org.eclipse.swt.events.SelectionEvent e) {

                            sShell.dispose();
                        }
                    });

            // refuse ボタンのアクション
            b_refused
                    .addSelectionListener(new org.eclipse.swt.events
                    .SelectionAdapter() {
                        public void widgetSelected(
                                org.eclipse.swt.events.SelectionEvent e) {

                            list = null;

                            sShell.dispose();
                        }
                    });

        }

        public ArrayList<VariableProposal> getList() {
            return list;
        }

        public void setList(ArrayList<VariableProposal> list) {
            this.list = list;
        }

    }
```

A.1.6 ClassAndWSSet.java

```
package workflow.support;

import java.util.ArrayList;

/**
 * @author sakai
```

```
 *ドメインの中のクラスとそれに対応する Web サービスリストのセットを格納するクラス
 */
public class ClassAndWSSet {

    private String ontClass;
    private ArrayList<WSData> wsList;

    /**
     * コンストラクタ
     * @param oc OntClass を格納
     * @param list クラスに対応する WebService のリストを格納
     */
    public ClassAndWSSet(String oc, ArrayList<WSData> list) {
        ontClass = oc;
        wsList = list;
    }

    public String getOntClass() {
        return ontClass;
    }

    public ArrayList<WSData> getWsList() {
        return wsList;
    }

    public void setOntClass(String ontClass) {
        this.ontClass = ontClass;
    }

    public void setWsList(ArrayList<WSData> wsList) {
        this.wsList = wsList;
    }

}
```

A.1.7 MessageAndVariable.java

```
package workflow.support;

/**
 * メッセージと変数，クエリを格納するクラス
 * @author sakai
 */
public class MessageAndVariableAndQuery {

    private String message;
    private String variable;
    private String query;

    /**
     * コンストラクタ
     * @param m メッセージ
     * @param v 変数
```

```
     * @param q クエリ
     */
    public MessageAndVariableAndQuery(String m, String v, String q) {
        this.message = m;
        this.variable = v;
        this.query = q;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getVariable() {
        return variable;
    }

    public void setVariable(String variable) {
        this.variable = variable;
    }

    public String getQuery() {
        return query;
    }

    public void setQuery(String query) {
        this.query = query;
    }

}
```

A.1.8 VariableAndQuery.java

```
package workflow.support;

import java.util.ArrayList;

import com.hp.hpl.jena.ontology.OntClass;

/**
 * 変数とクエリを格納するクラス
 * @author sakai
 */
public class VariableAndQuery {

    private OntClass variable;
    private ArrayList<OntClass> path;

    /**
     * コンストラクタ
```

```
     * @param v
     * @param p
     */
    public VariableAndQuery(OntClass v, OntClass p) {
        this.variable = v;
        this.path.add(p);
    }

    public ArrayList<OntClass> getPath() {
        return path;
    }

    public void setPath(ArrayList<OntClass> path) {
        this.path = path;
    }

    public OntClass getVariable() {
        return variable;
    }

    public void setVariable(OntClass variable) {
        this.variable = variable;
    }

}
```

A.1.9 VariablePair.java

```
package workflow.support;

/**
 * 変数対を格納するクラス
 * @author sakai
 */
public class VariablePair {

    private String direction;
    private String preVariable;
    private String folVariable;

    /**
     * コンストラクタ
     * @param d
     * @param pre
     * @param fol
     */
    public VariablePair(String d, String pre, String fol) {
        direction = d;
        preVariable = pre;
        folVariable = fol;
    }

    public String getFolVariable() {
```

```
        return folVariable;
    }

    public void setFolVariable(String folVariable) {
        this.folVariable = folVariable;
    }

    public String getPreVariable() {
        return preVariable;
    }

    public void setPreVariable(String preVariable) {
        this.preVariable = preVariable;
    }

    public String getDirection() {
        return direction;
    }

    public void setDirection(String direction) {
        this.direction = direction;
    }

}
```

A.1.10 VariableProposal.java

```
package workflow.support;

/**
 * 変数割り当ての提案を格納するクラス
 * @author sakai
 */
public class VariableProposal {

    private String direction;
    private MessageAndVariableAndQuery before;
    private MessageAndVariableAndQuery proposal;

    /**
     * コンストラクタ
     * @param b before
     * @param p proposal
     * @param d direction
     */
    public VariableProposal(MessageAndVariableAndQuery b,
            MessageAndVariableAndQuery p, String d) {

        direction = d;
        before = b;
        proposal = p;

    }
```

```
/**
 * コンストラクタ
 *
 * @param beforeMessage
 * @param beforeVariable
 * @param beforeQuery
 * @param proposalMessage
 * @param proposalVariable
 * @param proposalQuery
 * @param d
 */
public VariableProposal(String beforeMessage, String beforeVariable,
        String beforeQuery, String proposalMessage,
        String proposalVariable, String proposalQuery, String d) {

    direction = d;
    if (beforeVariable.equals(beforeQuery)) {
        before = new MessageAndVariableAndQuery(beforeMessage,
                beforeVariable, "");
    } else {
        before = new MessageAndVariableAndQuery(beforeMessage, beforeQuery
                .split("/")[0], beforeQuery);
    }

    if (proposalVariable.equals(proposalQuery)) {
        proposal = new MessageAndVariableAndQuery(proposalMessage,
                proposalVariable, "");
    } else {
        proposal = new MessageAndVariableAndQuery(proposalMessage,
                proposalQuery.split("/")[0], proposalQuery);
    }
}

public MessageAndVariableAndQuery getBefore() {
    return before;
}

public void setBefore(MessageAndVariableAndQuery before) {
    this.before = before;
}

public MessageAndVariableAndQuery getProposal() {
    return proposal;
}

public void setProposal(MessageAndVariableAndQuery proposal) {
    this.proposal = proposal;
}

public String getDirection() {
    return direction;
}
```

```
    public void setDirection(String direction) {
        this.direction = direction;
    }

}
```

A.1.11 WSData.java

```
package workflow.support;

/**
 * Web サービスデータベースに存在する
 * Web サービスのデータを格納するクラス
 * @author sakai
 */
public class WSData {

    private String name;
    private String serviceClass;
    private String wsdl;
    private String operationName;
    private String location;

    /**
     * コンストラクタ
     *
     * @param serviceName
     * @param sClass
     * @param url
     * @param operation
     * @param locate
     */
    public WSData(String serviceName, String sClass, String url,
            String operation, String locate) {

        serviceClass = sClass;
        name = serviceName;
        wsdl = url;
        operationName = operation;
        location = locate;

    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getOperationName() {
```

```
        return operationName;
    }

    public void setOperationName(String operationNames) {
        this.operationName = operationNames;
    }

    public String getServiceClass() {
        return serviceClass;
    }

    public void setServiceClass(String serviceClass) {
        this.serviceClass = serviceClass;
    }

    public String getWsdl() {
        return wsdl;
    }

    public void setWsdl(String wsdl) {
        this.wsdl = wsdl;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

}
```

A.1.12 BPEL Designer Editor に追加したメソッド群

```
    protected Composite createOntologyWidgets(Composite top, Composite parent) {
        FlatFormData data;

        final Composite composite = operationComposite =
        createFlatFormComposite(parent);
        data = new FlatFormData();
        if (top == null) {
            data.top = new FlatFormAttachment(0, IDetailsAreaConstants.VSPACE);
        } else {
            data.top = new FlatFormAttachment(top, IDetailsAreaConstants.VSPACE);
        }
        data.left = new FlatFormAttachment(0, IDetailsAreaConstants.HSPACE);
        data.right = new FlatFormAttachment(SPLIT_POINT, -SPLIT_POINT_OFFSET);
        composite.setLayoutData(data);

        ontologyLabel = wf.createLabel(composite, "Ontology:");
        ontologyText = wf.createText(composite, "", SWT.NONE);
        ontologyButton = wf.createButton(composite, "         Ontology       ",
                SWT.CENTER | SWT.DOWN);
```

```
// Provide Content Assist for the variables
OperationContentProvider provider = new OperationContentProvider();
ModelContentProposalProvider proposalProvider;
proposalProvider = new ModelContentProposalProvider(
        new ModelContentProposalProvider.ValueProvider() {
            public Object value() {
                return getInput();
            }
        }, provider);

RunnableProposal proposal3 = new RunnableProposal() {
    public String getLabel() {
        return "proposal";
    }

    public void run() {
        int direction = isInvoke ? ModelHelper.OUTGOING
                : ModelHelper.INCOMING;
        getCommandFramework().execute(
                new SetVariableCommand(getInput(), null, direction));
    }
};

proposalProvider.addProposalToEnd(new Separator());
proposalProvider.addProposalToEnd(getWSDLEdit());
proposalProvider.addProposalToStart(proposal3);

final FieldAssistAdapter contentAssist = new FieldAssistAdapter(
        ontologyText, fTextContentAdapter, proposalProvider, null, null);
//contentAsiist
contentAssist.setLabelProvider(new ModelLabelProvider());
contentAssist.setPopupSize(new Point(300, 100));
contentAssist.setFilterStyle(ContentProposalAdapter.FILTER_CUMULATIVE);
contentAssist
        .setProposalAcceptanceStyle(ContentProposalAdapter
        .PROPOSAL_REPLACE);
contentAssist.addContentProposalListener(getWSDLEdit());

contentAssist
        .addContentProposalListener(new IContentProposalListener() {

            public void proposalAccepted(
            IContentProposal chosenProposal) {
                if (chosenProposal.getContent() == null) {
                    return;
                }
                Operation oper = null;
                try {
                    oper = (Operation) ((Adapter) chosenProposal)
                            .getTarget();
                } catch (Throwable t) {
                    return;
```

```
                }
                List list = basicCommandList(getInput(),
                        IGNORE_PARTNER_LINK, oper);
                CompoundCommand cmd = new CompoundCommand();
                cmd.getCommands().addAll(list);
                getCommandFramework().execute(cmd);
            }
        });

//ボタンの操作
ontologyButton.addListener(SWT.Selection, new Listener() {
    public void handleEvent(Event event) {

        String inV = inputVariableText.getText();
        String outV = outputVariableText.getText();

        Variable preInputV = ((Invoke) getInput()).getInputVariable();
        Variable preOutputV = ((Invoke) getInput()).getOutputVariable();

        boolean hasVariable = false;

        if (!inV.equals("") && !outV.equals(""))
            hasVariable = true;

        boolean hasPartnerLink = false;
        if (!partnerName.getText().equals(""))
            hasPartnerLink = true;

        OntologyWindow OW1 = new OntologyWindow(hasVariable);
        OW1.popupOntologyWindow(interfaceName.getText());

        //namespace に location の追加

        Map nameSpace = BPELUtils
                .getAllNamespacesForContext(ModelHelper
                        .getProcess(modelObject));
        Set set = nameSpace.keySet();
        Iterator iterator = set.iterator();

        boolean hasNameSpace = false;
        String location = OW1.getNewServiceLocation();
        Object object = null;
        while (iterator.hasNext()) {
            object = iterator.next();

            if (location.equals(nameSpace.get(object))) {
                hasNameSpace = true;
                break;
            }
        }

        String nameSpaceNumber = "";
```

```
if (hasNameSpace) {//namespace 登録済み

    nameSpaceNumber = object.toString();

} else {//namespace 未登録

    Process process = ModelHelper.getProcess(getInput());
    BPELUtils.setNamespacePrefix(process, location, "ns"
            + nameSpace.size());
    nameSpaceNumber = "ns" + nameSpace.size();

    //Import の追加
    Import imp = BPELFactory.eINSTANCE.createImport();

    imp.setImportType("http://schemas.xmlsoap.org/wsdl/");
    imp.setLocation(OW1.getNewServiceWSDLFile());
    imp.setNamespace(OW1.getNewServiceLocation());

    AddImportCommand cmd = new AddImportCommand(ModelHelper
            .getProcess(getInput()), imp);
    if (cmd.canDoExecute()
            && cmd.wouldCreateDuplicateImport() == false) {
        getCommandFramework().execute(cmd);
    }

}

//Import の追加
Import imp = BPELFactory.eINSTANCE.createImport();

imp.setImportType("http://schemas.xmlsoap.org/wsdl/");
imp.setLocation("yumiyumi");
imp.setNamespace("yatta!!!!!!");

AddImportCommand cmdy = new AddImportCommand(ModelHelper
        .getProcess(getInput()), imp);
if (cmdy.canDoExecute()
        && cmdy.wouldCreateDuplicateImport() == false) {
    getCommandFramework().execute(cmdy);
}

//partnerLink の追加
PartnerLink pl = BPELFactory.eINSTANCE.createPartnerLink();
//pl.setName("sss");

//PortType の選択
List elements = ModelHelper.getDefinitions(modelObject);

IStructuredContentProvider contentProvider = null;
GatedContentProvider portTypeContentProvider;
portTypeContentProvider = new GatedContentProvider(
        new PortTypeContentProvider());
```

```
CompositeContentProvider ccp = new CompositeContentProvider();
ccp.add(new PartnerLinkTypeContentProvider());
ccp.add(portTypeContentProvider);

contentProvider = ccp;

Object[] ee = contentProvider.getElements(elements);
PortType portType = null;

for (int i = 0; i < ee.length; i++) {
    if (ee[i] instanceof PortType) {

        if (((PortType) ee[i]).getQName().getLocalPart()
                .equals(OW1.getServiceName())) {
            portType = (PortType) ee[i];
            break;
        }
    }
}

PartnerLinkType plt = PartnerlinktypeFactory.eINSTANCE
        .createPartnerLinkType();
plt.setName(OW1.getServiceName() + "PLT");

Role role1 = PartnerlinktypeFactory.eINSTANCE.createRole();
role1.setName(OW1.getServiceName() + "Role");
role1.setPortType(portType);
plt.getRole().add(role1);

BPELEditor fEditor = ModelHelper.getBPELEditor(getInput());
Definition artifactsDefinition = fEditor
        .getArtifactsDefinition();
CompoundCommand cmd = new CompoundCommand();
// We add the import into the artifacts now,
//because code relies on diving through
// WSDL imports to find portTypes etc.

// 1. Add WSDL import to the artifacts file
//(again, this is a noop if already exists).
cmd
        .add(new AddWSDLImportCommand(artifactsDefinition,
                portType));

// 2. Create the new Partner Link Type in the artifacts file.
cmd.add(new CreatePartnerLinkTypeCommand(artifactsDefinition,
        plt));

fEditor.getCommandFramework().execute(cmd);

pl.setName(OW1.getServiceName() + "PL");
pl.setPartnerLinkType(plt);
pl.setPartnerRole(role1);
// ask for partner link type
```

```
List cmds = basicCommandList(getInput(), pl, null);
cmds.add(0, new AddPartnerLinkCommand(ModelHelper
        .getContainingScope(getInput()), pl));
//
CompoundCommand cmd1 = new CompoundCommand();
cmd1.getCommands().addAll(cmds);
getCommandFramework().execute(cmd1);

List oList = portType.getOperations();
Operation oper = null;
for (int i = 0; i < oList.size(); i++) {

    //if(((Operation)oList.get(i)).getName().equals
    //(OW1.getNewServiceOperation())){
    if (((Operation) oList.get(i)).getName().equals(
            OW1.getNewServiceOperation())) {
        oper = (Operation) oList.get(i);
        break;
    }
}

List list = basicCommandList(getInput(), IGNORE_PARTNER_LINK,
        oper);
CompoundCommand cmd2 = new CompoundCommand();
cmd2.getCommands().addAll(list);
getCommandFramework().execute(cmd2);

if (!hasPartnerLink) {

    //                  Input の変数の追加
    createVariable2(getInput(), OW1.getServiceName()
            + "Response", ModelHelper.INCOMING);

    //                  Output の変数の追加
    createVariable2(getInput(), OW1.getServiceName()
            + "Request", ModelHelper.OUTGOING);

} else {

    if (hasVariable) {

        if (OW1.isSameClass()) {

            CompoundCommand cmd3 = new CompoundCommand();
            cmd3.add(new SetVariableCommand(getInput(),
                    preInputV, ModelHelper.OUTGOING));
            cmd3.add(new SetVariableCommand(getInput(),
                    preOutputV, ModelHelper.INCOMING));
            getCommandFramework().execute(cmd3);

        } else {
```

```
                    //                          Input の変数の追加
                    createVariable2(getInput(), outV + "2"
                            + OW1.getServiceName() + "Response",
                            ModelHelper.INCOMING);

                    //                          Output の変数の追加
                    createVariable2(getInput(), inV + "2"
                            + OW1.getServiceName() + "Request",
                            ModelHelper.OUTGOING);

                    Process flow = ModelHelper.getProcess(getInput());
                    Sequence seq = (Sequence) flow.getActivity();

                    Variable newInputV = ((Invoke) getInput())
                            .getInputVariable();
                    Variable newOutputV = ((Invoke) getInput())
                            .getOutputVariable();

                    changeVariable(seq, OW1.getProposalResult(),
                            preInputV, preOutputV, newInputV,
                            newOutputV);

                }

            } else {

                //                      Input の変数の追加
                createVariable2(getInput(), OW1.getServiceName()
                        + "Response", ModelHelper.INCOMING);

                //                      Output の変数の追加
                createVariable2(getInput(), OW1.getServiceName()
                        + "Request", ModelHelper.OUTGOING);

            }

        }

    }
});

ontologyText.addListener(SWT.KeyDown, new Listener() {
    public void handleEvent(Event event) {
        if (event.keyCode == SWT.CR) {
            ontologyName = ontologyText.getText();
            findAndSetOperation(ontologyText.getText());
        }
    }
});

// end of content assist

//Form の位置決め
```

```
    data = new FlatFormData();
    data.right = new FlatFormAttachment(100, 0);
    data.top = new FlatFormAttachment(ontologyText, +2, SWT.TOP);
    data.bottom = new FlatFormAttachment(ontologyText, -2, SWT.BOTTOM);
    ontologyButton.setLayoutData(data);

    data = new FlatFormData();
    data.left = new FlatFormAttachment(0, BPELUtil.calculateLabelWidth(
            ontologyLabel, STANDARD_LABEL_WIDTH_SM));
    data.right = new FlatFormAttachment(ontologyButton, 0);
    ontologyText.setLayoutData(data);

    data = new FlatFormData();
    data.left = new FlatFormAttachment(0, 0);
    data.right = new FlatFormAttachment(ontologyText,
            -IDetailsAreaConstants.HSPACE);
    data.top = new FlatFormAttachment(ontologyText, 0, SWT.CENTER);
    ontologyLabel.setLayoutData(data);

    return composite;
}

void createVariable2(EObject ref, String name, int direction) {

    Variable variable = BPELFactory.eINSTANCE.createVariable();

    if (name == null) {
        name = plainLabelWordFor(direction);
    }

    // set name and type
    //variable.setName ( nameDialog.getValue() );
    variable.setName(name);

    Object type = ModelHelper.getVariableType(getInput(), direction);
    if (type != null && type instanceof Message) {
        variable.setMessageType((Message) type);
    }

    // create the variable and then set the input variable to it.
    CompoundCommand cmd = new CompoundCommand();
    cmd.add(new AddVariableCommand(ref, variable));
    cmd.add(new SetVariableCommand(getInput(), variable, direction));

    getCommandFramework().execute(cmd);
}

void changeVariable(Sequence seq, ArrayList<VariableProposal> list,
        Variable preInputVariable, Variable preOutputVariable,
        Variable newInputVariable, Variable newOutputVariable) {

    EList aList = seq.getActivities();
```

```
for (int i = 0; i < aList.size(); i++) {

    if (aList.get(i) instanceof Assign) {

        Assign assign = (Assign) aList.get(i);

        EList copyList = assign.getCopy();

        for (int c = 0; c < copyList.size(); c++) {

            Copy copy = (Copy) copyList.get(c);

            From from = copy.getFrom();

            if (from.getVariable().equals(preInputVariable)) {

                String partName = from.getPart().getName();
                for (int n = 0; n < list.size(); n++) {
                    if (from.getQuery() == null) {

                        if (list.get(n).getBefore().getVariable()
                                .equals(partName)) {

                            from.setVariable(newInputVariable);
                            Part part = null;
                            part = (Part) newInputVariable
                                    .getMessageType().getPart(
                                            list.get(n).getProposal()
                                                    .getVariable());
                            from.setPart(part);
                            if (!list.get(n).getProposal().getQuery()
                                    .equals("")) {
                                Query queryObject = BPELFactory.eINSTANCE
                                        .createQuery();
                                queryObject
                                        .setQueryLanguage(IBPELUIConstants
                                        .EXPRESSION_LANGUAGE_XPATH);
                                queryObject.setValue(list.get(n)
                                        .getProposal().getQuery());
                                from.setQuery(queryObject);
                            }

                        }
                    } else {

                        if (list.get(n).getBefore().getVariable()
                                .equals(partName)
                                && from.getQuery().getValue().equals(
                                        list.get(n).getBefore()
                                                .getQuery())) {

                            from.setVariable(newInputVariable);
                            Part part = null;
```

```
                part = (Part) newInputVariable
                        .getMessageType().getPart(
                                list.get(n).getProposal()
                                        .getVariable());
                from.setPart(part);
                if (!list.get(n).getProposal().getQuery()
                        .equals("")) {
                    Query queryObject = BPELFactory.eINSTANCE
                            .createQuery();
                    queryObject
                            .setQueryLanguage(IBPELUIConstants
                            .EXPRESSION_LANGUAGE_XPATH);
                    queryObject.setValue(list.get(n)
                            .getProposal().getQuery());
                    from.setQuery(queryObject);
                }

            }

        }

    }

}

if (from.getVariable().equals(preOutputVariable)) {

    String partName = from.getPart().getName();
    for (int n = 0; n < list.size(); n++) {
        if (from.getQuery() == null) {

            if (list.get(n).getBefore().getVariable()
                    .equals(partName)) {

                from.setVariable(newOutputVariable);
                Part part = null;
                part = (Part) newOutputVariable
                        .getMessageType().getPart(
                                list.get(n).getProposal()
                                        .getVariable());
                from.setPart(part);
                if (!list.get(n).getProposal().getQuery()
                        .equals("")) {
                    Query queryObject = BPELFactory.eINSTANCE
                            .createQuery();
                    queryObject
                            .setQueryLanguage(IBPELUIConstants
                            .EXPRESSION_LANGUAGE_XPATH);
                    queryObject.setValue(list.get(n)
                            .getProposal().getQuery());
                    from.setQuery(queryObject);
                }
```

```
            }
        } else {

            if (list.get(n).getBefore().getVariable()
                    .equals(partName)
                    && from.getQuery().getValue().equals(
                            list.get(n).getBefore()
                                    .getQuery())) {

                from.setVariable(newOutputVariable);
                Part part = null;
                part = (Part) newOutputVariable
                        .getMessageType().getPart(
                                list.get(n).getProposal()
                                        .getVariable());
                from.setPart(part);
                if (!list.get(n).getProposal().getQuery()
                        .equals("")) {
                    Query queryObject = BPELFactory.eINSTANCE
                            .createQuery();
                    queryObject
                            .setQueryLanguage(IBPELUIConstants
                            .EXPRESSION_LANGUAGE_XPATH);
                    queryObject.setValue(list.get(n)
                            .getProposal().getQuery());
                    from.setQuery(queryObject);
                }

            }

        }

    }

}

To to = copy.getTo();

if (to.getVariable().equals(preInputVariable)) {

    String partName = to.getPart().getName();
    for (int n = 0; n < list.size(); n++) {
        if (to.getQuery() == null) {

            if (list.get(n).getBefore().getVariable()
                    .equals(partName)) {

                to.setVariable(newInputVariable);
                Part part = null;
                part = (Part) newInputVariable
                        .getMessageType().getPart(
                                list.get(n).getProposal()
                                        .getVariable());
```

```
                to.setPart(part);
                if (!list.get(n).getProposal().getQuery()
                        .equals("")) {
                    Query queryObject = BPELFactory.eINSTANCE
                            .createQuery();
                    queryObject
                            .setQueryLanguage(IBPELUIConstants
                            .EXPRESSION_LANGUAGE_XPATH);
                    queryObject.setValue(list.get(n)
                            .getProposal().getQuery());
                    to.setQuery(queryObject);
                }

            }
        } else {

            if (list.get(n).getBefore().getVariable()
                    .equals(partName)
                    && to.getQuery().getValue().equals(
                            list.get(n).getBefore()
                                    .getQuery())) {

                to.setVariable(newInputVariable);
                Part part = null;
                part = (Part) newInputVariable
                        .getMessageType().getPart(
                                list.get(n).getProposal()
                                        .getVariable());
                to.setPart(part);
                if (!list.get(n).getProposal().getQuery()
                        .equals("")) {
                    Query queryObject = BPELFactory.eINSTANCE
                            .createQuery();
                    queryObject
                            .setQueryLanguage(IBPELUIConstants
                            .EXPRESSION_LANGUAGE_XPATH);
                    queryObject.setValue(list.get(n)
                            .getProposal().getQuery());
                    to.setQuery(queryObject);
                }

            }

        }

    }

}

if (to.getVariable().equals(preOutputVariable)) {

    String partName = to.getPart().getName();
    for (int n = 0; n < list.size(); n++) {
```

```
if (to.getQuery() == null) {

    if (list.get(n).getBefore().getVariable()
            .equals(partName)) {

        to.setVariable(newOutputVariable);
        Part part = null;
        part = (Part) newOutputVariable
                .getMessageType().getPart(
                        list.get(n).getProposal()
                                .getVariable());
        to.setPart(part);
        if (!list.get(n).getProposal().getQuery()
                .equals("")) {
            Query queryObject = BPELFactory.eINSTANCE
                    .createQuery();
            queryObject
                    .setQueryLanguage(IBPELUIConstants
                    .EXPRESSION_LANGUAGE_XPATH);
            queryObject.setValue(list.get(n)
                    .getProposal().getQuery());
            to.setQuery(queryObject);
        }

    }
} else {

    if (list.get(n).getBefore().getVariable()
            .equals(partName)
            && to.getQuery().getValue().equals(
                    list.get(n).getBefore()
                            .getQuery())) {

        to.setVariable(newOutputVariable);
        Part part = null;
        part = (Part) newOutputVariable
                .getMessageType().getPart(
                        list.get(n).getProposal()
                                .getVariable());
        to.setPart(part);
        if (!list.get(n).getProposal().getQuery()
                .equals("")) {
            Query queryObject = BPELFactory.eINSTANCE
                    .createQuery();
            queryObject
                    .setQueryLanguage(IBPELUIConstants
                    .EXPRESSION_LANGUAGE_XPATH);
            queryObject.setValue(list.get(n)
                    .getProposal().getQuery());
            to.setQuery(queryObject);
        }

    }
```

```
                    }
                }
            }
        }
    }
}
```