ライフサイエンス統合データベース開発運用

「標準化に向けた主要ウェブサービスの調査と

ドキュメントの作成」

調査報告書

平成 19 年 12 月 28 日

(株)三菱総合研究所

目次

1. 調査の目的

本調査は、「ライフサイエンス統合データベース開発運用」における「共通基盤技術開発」を行うにあたり必要とされる、ウェブサービスの現状調査とドキュメント作成を目的としている。

ライフサイエンスにおけるデータベースは、多様かつ巨大になりつつあり、これらを統合したデータベースを構築するにあたり、実際に既存のデータベースを受け入れるだけでなく、ウェブサービスにより分散環境のまま利用可能な形で統合するシステムを構築する必要に迫られている。しかし、国内外で提供されているウェブサービスはドキュメントが十分に整備されているとはいえない。

このため、それぞれのサービスの機能と使用方法を十分に調査することで、効果的な統合化を行うためのドキュメントと基礎資料を作成する。

2. 調査の概要

現在、世界中のバイオインフォマティクスのセンターで様々なウェブサービスが公開されており、BioMOBY のように相互運用を目的としたレポジトリも開発されてきているが、主要なサービスがすべて網羅されているとは言えない状況である。また、各サービスでやりとりされる生物学データは、フォーマットの違いなどにより簡単には相互利用できない状況であること、サービス毎にメソッドの命名規則が異なることやドキュメントが不足しているといった問題があり、各サービスを組み合わせたワークフローの構築は非常に困難である。

そこで、主要なウェブサービスについて、各サービスで提供されている機能を調査し、それぞれのメソッドの解説と使用方法をドキュメント化するとともに、メソッドの目的ごとに階層的な分類を行う。この結果から、各サーバの全てのメソッドを統一的に扱うために破綻しない命名規則を検討する。さらに、各サービスでやりとりされるデータフォーマットの中で共通化できるものの種類を検討しデータ型の標準化を図る。また、ウェブサービスのメソッドの機能的な階層分類と、まだウェブサービス化されていない既存のバイオインフォマティクスのアプリケーションの機能的な階層分類について比較を行い、現状のウェブサービスにおいてどのようなサービスが欠けているかを明らかにする。これらの調査を通じて、今後の統合ウェブサービス構築に必要となる基礎資料を作成する。また、これらの調査結果は一般利用者が既存のウェブサービスを直接利用する際に参考となる有用なドキュメントとして公開する。

具体的には以下の項目を実施する(**図 1** 参照)。

① ウェブサービスプロバイダーの選定

ウェブサービスのメソッド情報を収集、調査する対象プロバイダーの選定を行う。

選定基準は、主要機関であること、汎用的なメソッドを多く提供していること、調査全体としてバイオインフォマティクス分野の全体をカバーできること、などとする。

② ウェブサービス記述言語ファイル(WSDL)の収集

ウェブサービスの機能等を調査するためのリソースとして、①で選定したウェブサービスプロバイダーで提供されているウェブサービスの WSDL ファイルを収集する。

③ ドキュメントページの収集

②で取得した WSDL にはウェブサービスのメソッド機能が十分に記述されていない場合が多いことから、メソッドの機能や利用方法を解説したウェブページを可能な限りで収集する。

④ メソッド情報の整理

②と③で収集した WSDL ファイルとウェブページから、メソッド情報を表形式で整理する。

⑤ メソッドの分類

④で整理したメソッド情報をもとに、メソッドの機能を大まかに分類する。

⑥ メソッド命名規則の提案

⑤のメソッド機能分類結果をもとに、メソッドの機能分類カテゴリを定義し、命名規則を決める。その際、⑤の既存メソッドの現状を反映するとともに、類似した試みが行われている BioMOBY などのプロジェクトを調査し、連携の可能性なども含めて、参考にする。

⑦ メソッド入出力データの分類

⑤及び⑥で機能分類したメソッドに対し、ドキュメント整備の必要性の観点から優先順位を付け、優先順位の高いメソッドに絞りその入出力データの分類、整理を行う。なお、優先順位付けについては、適宜担当者の意見を仰ぐこととする。

⑧ メソッド入出力データの標準化

⑦のメソッド入出力データの分類結果をもとに、プロバイダー間で異なる構造、表記がなされているものを標準化するための仕様案を提案する。その際、⑥と同様に、類似した試みが行われている BioMOBY などのプロジェクトを調査し、連携の可能性なども含めて、参考にする。

⑨ メソッド実行テスト

⑦の優先順位付けで絞り込まれたメソッドに対し、適当な入力データサンプルを用意し、メソッドの実行テストを行い、実行の可否などの結果を⑩のドキュメント作成でまとめる。なお、実行テスト時の使用言語は、Ruby、Perl、Java、Python のいずれかとし、いずれかの言語で実行が失敗した場合はこの中の別の言語でも試すこととする。

⑩ メソッドドキュメントの作成

⑥で作成したメソッド命名規則、⑧で作成したメソッド標準入出力データと、個別メソッド間の対応関係表を作成するとともに、メソッドの機能、使用方法、実行の可否についての解説をまとめる。

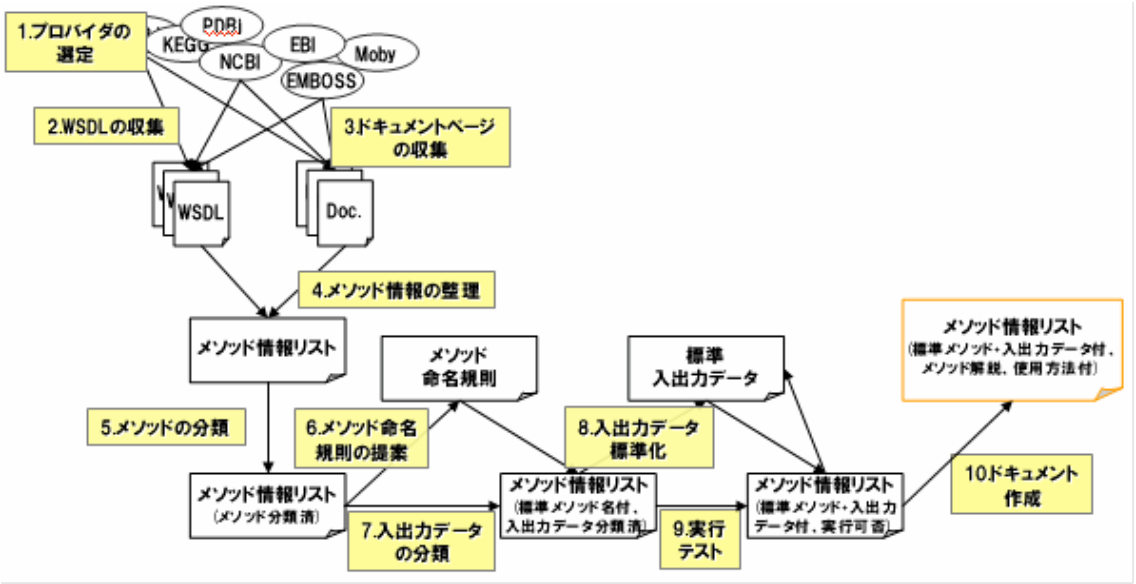
1.プロバイダの選定
PDBJ
KEGG
NCBI
EBI
Moby
EMBOSS
2.WSDLの収集
3.ドキュメントページの収集
WSDL
Doc.
4.メソッド情報の整理
メソッド情報リスト
メソッド命名規則
標準入出力データ
メソッド情報リスト（標準メソッド・入出力データ付，メソッド解説，使用方法付）
5.メソッドの分類
6.メソッド命名規則の提案
8.入出力データ標準化
メソッド情報リスト（メソッド分類済）
メソッド情報リスト（標準メソッド名付，入出力データ分類済）
メソッド情報リスト（標準メソッド・入出力データ付，実行可否）
10.ドキュメント作成
7.入出力データの分類
9.実行テスト

図 1 調査の流れ

3. 調査結果の詳細

3.1 ウェブサービスプロバイダーの選定

ウェブサービスのメソッド情報を収集、調査する対象プロバイダーの選定を行う。選定基準は、主要機関であること、汎用的なメソッドを多く提供していること、調査全体としてバイオインフォマティクス分野の全体をカバーできること、などとした。

その結果、以下のウェブサービスプロバイダーを第一選定プロバイダーとした。後述の作業においては、さらにこれらのプロバイダーから絞込み、詳細な調査を行う。

表 1 ウェブサービスプロバイダーの選定

プロバイダー名	URL
DDBJ	http://xml.nig.ac.jp/
KEGG	http://www.genome.jp/kegg/soap/
PDBj	http://www.pdbj.org/
BIND	http://soap.bind.ca/
Basis	http://www.basis.ncl.ac.uk/
EBI	http://www.ebi.ac.uk/
EMBL	http://www.ebi.ac.uk/xembl/
EMBOSS	http://www.ebi.ac.uk/soaplab/emboss4/
GoMiner	http://discover.nci.nih.gov/gominer/
MITBroadInst	http://www.broad.mit.edu/webservices/genecruiser/services/Annotation?wsdl
Moby	http://www.biomoby.org/
NCBI	http://www.ncbi.nlm.nih.gov/entrez/query/static/esoap_help.html
Sanger	http://services.sanger.ac.uk/pfamWebService/services/pfamWebService?wsdl
ZLab	http://zlab.bu.edu/zlab/gene.shtml
myGrid	http://www.mygrid.org.uk/

3.2 ウェブサービス記述言語ファイル(WSDL)の収集

ウェブサービスの機能等を調査するためのリソースとして、3.1 で選定したウェブサービスプロバイダーで提供されているウェブサービスの WSDL ファイルを収集した。なお、Moby で提供されているウェブサービスの WSDL は URL 指定で直接入手できないため、Moby Client プログラムを利用して取得した(詳細は 3.6 を参照)。

3.3 ドキュメントページの収集

3.2 で取得した WSDL にはウェブサービスのメソッド機能が十分に記述されていない場合が多いことから、メソッドの機能や利用方法を解説したウェブページを可能な限り収集した。

3.4 メソッド情報の整理

3.2、3.3 で収集した WSDL ファイルとウェブページから、メソッド情報を表形式で整理した。整理した表は、「調査結果データ」に一部記載するとともに、納品物の電子媒体に全体を保存した。

ここでは、収集した WSDL などから得られたウェブサービスのメソッド数の集計結果を記載する(**表 2**、**表 3** 参照)。

これらの数値や、国内外区分、提供されているサービスの種類、来年 2 月に予定している会議出席者などの情報を考慮して、表の薄青に塗りつぶしたプロバイダーについて、3.5 メソッドの分類を行うことにした。

表 2 メソッド数の集計(全体)

プロバイダー名	portType 数	operation 数
DDBJ	25	205
KEGG	1	72
PDBj	4	34
BIND	1	12
Basis	1	8
EBI	26	235
EMBL	1	1
EMBOSS	209	1045
GoMiner	1	1
MITBroadInst	1	10
Moby	1600	2044
NCBI	4	30
Sanger	1	5
ZLab	7	33
myGrid	19	437

合計	3501	6216
合計(重複除去)	1645	1998

※薄青背景色は、3.5 メソッドの分類以降の調査対象プロバイダー

表 3 メソッド数の集計(Moby 内プロバイダー)

プロバイダー名	portType 数	Operation 数
Moby/CDK.SF.NET	1	1
Moby/NNN.NNN.NNN	1	1
Moby/antirrhinum.net	12	12
Moby/arabidopsis.info	133	133
Moby/arabidopsis.med.ohio-state.edu	1	1
Moby/arabidopsis.org	5	5
Moby/arexdb.org	2	2
Moby/asrp.cgrb.oregonstate.edu	4	4
Moby/atidb.org	17	17
Moby/bar.utoronto.ca	2	2
Moby/bioinfo.genopole-toulouse.prd.fr	68	68
Moby/bioinfo.icapture.ubc.ca	100	100
Moby/bioinfo.inibap.org	7	7
Moby/bioinfo.mpimp-golm.mpg.de	1	1
Moby/biomoby.inibap.org	1	1
Moby/biomoby.renci.org	120	120
Moby/bioserv.rpbs.jussieu.fr	19	19
Moby/bioweb.pasteur.fr	1	1
Moby/castor.brc.mcw.edu	9	9
Moby/ccgb.umn.edu	2	2
Moby/cnio.es	26	44
Moby/cshl.edu	4	4
Moby/de.mpiz_koeln-mpg	1	1
Moby/edu.purdue.discoverypark.ionomics	1	1
Moby/eva.mpg.de	5	5
Moby/gabi.rzpd.de	4	4
Moby/genome.imim.es	43	43

Moby/heaven.mshri.on.ca	1	1
Moby/iant.toulouse.inra.fr	3	3
Moby/icapture.mrl.ubc.ca	2	2
Moby/icapture.ubc.ca	2	2
Moby/imb.uq.edu.au	1	1
Moby/inb.bsc.es	211	358
Moby/inblosam.com	1	1
Moby/itdinges.mine.nu	2	2
Moby/llama.med.harvard.edu	16	16
Moby/mapman.mpimp-golm.mpg.de	4	4
Moby/metnetdb.org	2	2
Moby/mips.gsf.de	45	45
Moby/mmb.pcb.ub.es	37	37
Moby/moby.ucalgary.ca	5	5
Moby/mpiz-koeln.mpg.de	5	5
Moby/orygenesdb.cirad.fr	4	4
Moby/oryzataglines.cirad.fr	1	1
Moby/plantenergy.uwa.edu.au	6	6
Moby/plantgdb.org	1	1
Moby/samples.jmoby.net	5	5
Moby/sapporo.genomics.purdue.edu	4	4
Moby/schematikon.org	8	8
Moby/scri.ac.uk	1	1
Moby/seedgenes.org	1	1
Moby/sgn.cornell.edu	2	2
Moby/sirius.brc.mcw.edu	1	1
Moby/soaplab.icapture.ubc.ca	200	200
Moby/ssg.uab.edu	2	2
Moby/statgen.reversetest.authority	1	1
Moby/test_toppred.toulouse.inra.fr	2	2
Moby/testing.bioinfo.genopole-toulouse.prd.fr	4	4
Moby/testing.org	1	1

Moby/tropgenedb.cirad.fr	10	10
Moby/ualberta.ca	85	85
Moby/urgi.versailles.inra.fr	21	21
Moby/vbd.crop.org.cn	1	1
Moby/workflow.renci.unc.edu	5	5
Moby/www.bcgsc.bc.ca	1	1
Moby/www.bioinformatica.ucb.br	6	6
Moby/www.bioinformatics.nl	9	9
Moby/www.ciat.org	1	1
Moby/www.cnb.uam.es	217	496
Moby/www.cnio.es	6	6
Moby/www.cs.purdue.edu	1	1
Moby/www.gbif.org	7	7
Moby/www.generationcp.org	1	1
Moby/www.hgvbase.org	9	9
Moby/www.iris.irri.org	18	18
Moby/www.mncn.csic.es	2	2
Moby/www.nias.affrc.go.jp	12	12
Moby/www.rri.sari.ac.uk	1	1
Moby/www.sdsc.edu	5	5
Moby/www.tigr.org	5	5
Moby/www.transvar.org	1	1
Moby/www.visualgenomics.ca	1	1
Moby/www.weigelworld.org	4	4

※薄青背景色は、3.5 メソッドの分類以降の調査対象プロバイダー

3.5 メソッドの分類

3.4 で整理したメソッド情報をもとに、メソッドの機能を大まかに分類した。分類結果の詳細は納品物の電子媒体に納めるので、ここでは主なメソッド分類の operation 数を**表 4** に記載する。

表 4 主要分類に含まれるメソッドの数

主なメソッド分類	分類された operation 数
Analysis > Calculate Nucleic Composition	41
Analysis > Calculate Nucleic Property	31
Analysis > Calculate Protein Property	73
Convert > Convert Sequence	108
Database Search	768
Display > Display Graph	33
Display > Display Multiple Alignment	10
Homology Search > Blast	8
Homology Search > FASTA	40
Motif Search	170
Multiple Alignment	93
Pairwise Alignment	192
Parsing	217
Phylogenetic Analysis	265
Predict > Repeat Prediction	23
Predict > Gene Finding	23

3.6 メソッド命名規則の提案

3.5 のメソッド機能分類結果をもとに、メソッドの機能分類カテゴリを定義し、命名規則を決めた。その際、3.5 の既存メソッドの現状を反映するとともに、類似した試みが行われている BioMOBY などのプロジェクトを調査し、連携の可能性なども含めて、参考にした。

まず、BioMoby プロジェクトの調査結果をまとめる。

BioMoby(http://www.biomoby.org/)では、サービスクラス及びサービス入出力オブジェクトクラスのオントロジーを定義、参照することで、サービス発見やサービス連携を容易にする仕組みを構築している(**図 2** 参照)。

従来のウェブサービスにおいては、ユーザは、自分の要求を満たしてくれるサービスを提供しているプロバイダーをウェブ上から探し出し、プロバイダーごとに異なるリクエスト方法を調べた上で、リクエストを行う必要があった。しかし BioMoby を使うことで、ユーザは、Moby Central と呼ばれる、各プロバイダーのサービス情報を管理しているサーバーを介してサービスの探索を行うことができる。また BioMoby で提供されているクライアントプログラムを利用することで、統一した形式でサービスの実行も可能となる。

処理の流れは、まずサービスプロバイダーが提供したいサービスを Moby Central

に登録する(Step1)。その際、そのサービスのクラスや、入出力オブジェクトのクラスを、Service Type Ontology や Data Type Ontology をもとに定義し、統一化しておく。次に、ユーザは、BioMoby ホームページからダウンロードしたクライアントプログラムを利用して、サービスクラス名や入力クラス名などをキーにしてサービス探索を行い(Step2)、サービスインスタンス情報を取得する(Step3)。最後に、ユーザは再びクライアントプログラムを利用して、サービスプロバイダーに実行リクエストを出し(Step4)、プロバイダー側からの結果を受け取る(Step5)。

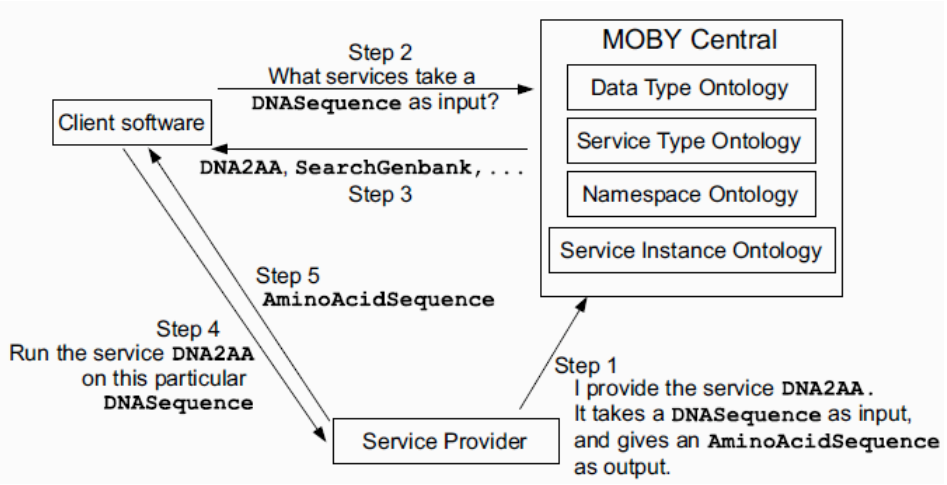

図 2 BioMoby のシステム構成

(出典:Seahawk: moving beyond HTML in Web-based bioinformatics analysis. P.MK. Gordon & C.W.Sensen, BMC Bioinformatics (2007))

図 3 に BioMoby クライアントプログラム(perl 版。SOAP::Lite を利用)を利用したサービス発見プログラム例を示す。このプログラムの中では、findService を使って、プロバイダーURI やサービスメソッド名、入出力オブジェクト名などをもとにサービスの検索を行っている。検索で得られる情報としては、そのサービスのプロバイダーURI、メソッド名、メソッドの解説、入出力データの名前・オブジェクトクラス・名前空間、そしてパラメータ情報がある。パラメータ情報には、その名前、メソッドクラス(同期/非同期の区分)、上下限値、デフォルト値、選択肢がある。例えば、ホモロジー検索メソッドの場合、選択できるデータベース名を得るためには、パラメータ database(メソッドごとに名前は異なる場合がある)の選択肢を参照すれば良い。

```
#!/usr/bin/perl


use MOBY::Client::Central;
```

```
$ENV{'MOBY_SERVER'} = "http://moby.ucalgary.ca/moby/MOBY-Central.pl";
$ENV{'MOBY_URI'}    = "http://moby.ucalgary.ca/MOBY/Central";

#---------------- サービス検索のためのキー指定 ----------------#
$authURI     = "inb.bsc.es";                  # プロバイダーURI（任意）
$serviceName = "getAminoAcidSequence";        # サービスメソッド名（任意）

#--------------- findService ---------------#
my $m = MOBY::Client::Central->new;
my ($serv_instances, $reg) = $m->findService(authURI => $authURI, serviceName =>
$serviceName);

if (scalar(@$serv_instances)<1){
    exit;
}
my $SI = shift @$serv_instances;

printf "Service Name: %s¥n",$SI->name;
printf "Authority: %s¥n",$SI->authority;
printf "Description: %s¥n",$SI->description;
printf "Sync/Async: %s¥n",$SI->category;

#--------------- Input ---------------#
my $inputs = $SI->input;
foreach my $input (@$inputs){
    if ($input->isSimple){
       printf "Simple Input Name: %s¥n", $input->articleName;
       printf "Simple Input Type: %s¥n", $input->objectType
       my $namespaces = $input->namespaces;
       foreach my $namespace (@$namespaces){
            printf "Simple Input Namespace: %s¥n",$namespace;
       }
```

```
    } elsif ($input->isCollection){
       printf "Collection Input Name: %s¥n",$input->articleName;
       my $simples = $input->Simples;
       foreach my $simp (@$simples){
           printf “Included Input Type: %s¥n”, $simp->objectType;
          my $namespaces = $simp->namespaces;
          foreach my $namespace (@$namespaces){
               printf “Included Input Namespace: %s¥n”,$namespace;
          }
    }
}

#--------------- Output ---------------#
my $outputs = $SI->output;
foreach my $output (@$outputs){
    if ($output->isSimple){
       printf “Simple Output Name: %s¥n”, $output->articleName;
       printf “Simple Output Type: %s¥n”, $output->objectType
       my $namespaces = $output->namespaces;
       foreach my $namespace (@$namespaces){
            printf “Simple Output Namespace: %s¥n”,$namespace;
       }
    } elsif ($output->isCollection){
       printf "Collection Output Name: %s¥n",$output->articleName;
       my $simples = $output->Simples;
       foreach my $simp (@$simples){
           printf “Included Output Type: %s¥n”, $simp->objectType;
          my $namespaces = $simp->namespaces;
          foreach my $namespace (@$namespaces){
               printf “Included Output Namespace: %s¥n”,$namespace;
          }
    }
```

```
}


#--------------- Parameters ---------------#
my $secondaries = $SI->secondary;
foreach my $secondary (@$secondaries){
    next unless $secondary->isSecondary;
    printf “Parameter Name: %s¥n” $secondary->articleName;
    printf “Parameter Type: %s¥n”, $secondary->datatype;
    printf “Parameter Max: %s¥n”,$secondary->max;
    printf “Parameter Min: %s¥n”,$secondary->min;
    printf “Parameter Default: %s¥n”,$secondary->default;
    $enum_line,$_->description
    foreach my $enum (@{$secondary->enum}){
       printf “Parameter Enumeration: %s¥n”,$enum;
    }
}
```

図 3 サービス発見のための BioMoby クライアントプログラム例

BioMoby のサービスの中には、同期実行と非同期実行のメソッドが含まれており、そのいずれであるかは、前述のサービス発見プログラムの中で利用しているように[Service Instance]->category で参照可能である。この値が単に”moby”の場合は同期実行メソッド、”moby-async”の場合は非同期メソッドである。なお、Moby Central サーバーが移動することもあり得るので、その場合はプログラム内で記述しているように環境変数 MOBY_SERVER、MOBY_URI などを指定しておく必要がある。

図 4 に同期実行のためのクライアントプログラム例を示す。実行のためにまずサービス発見時と同様にプロバイダーURI、メソッド名をもとにサービスインスタンスを取得しておく必要がある。サービスインスタンスが見つかれば、次に retrieveService を使って WSDL ファイルを取得し、MOBY::Client::Service->new によってサービスプロバイダーにアクセスする。実行リクエストは、入力データ(、必要ならばパラメータ)を指定し、execute によって発信する。入力データ、パラメータの指定方法は同じで、入力/パラメータ名とその値の組を並べればよい。例えば、サンプルプログラムの場合は、入力データ名は、”input”、そのデータ値は”<Object namespace=’Swiss-Prot’

id=’P08641’/>”である。このデータ値の記述は、BioMoby で設定されているオブジェクト指定方法で、そのオブジェクトクラス(この場合は“Object”)と、名前空間(“Swiss-Prot”)、ID(“P08641”)が基本要素となり、入力データに配列データ自体を指定するような場合はさらに配列を埋め込むようにする(図 5 参照)。

```
#!/usr/bin/perl

use MOBY::Client::Central;
use MOBY::Client::Service;
$ENV{'MOBY_SERVER'} = "http://moby.ucalgary.ca/moby/MOBY-Central.pl";
$ENV{'MOBY_URI'}    = "http://moby.ucalgary.ca/MOBY/Central";

$authURI     = "inb.bsc.es";
$serviceName = "getAminoAcidSequence";
$articleName = "input";
$inputdata    = "<Object namespace='Swiss-Prot' id='P08641'/>";

my $m = MOBY::Client::Central->new;
my ($serv_instances, $reg) = $m->findService(
                                                authURI => $authURI,
                                                serviceName => $serviceName);

my $SI = shift @$serv_instances;

my $wsdl = $m->retrieveService($SI);

my $SERV = MOBY::Client::Service->new(service => $wsdl);

@XMLinputlist = ([$articleName, $inputdata]);

my $result = $SERV->execute(XMLinputlist => ¥@XMLinputlist);
```

```
printf "%s¥n",$result;
```

図 4 サービス同期実行のための BioMoby クライアントプログラム例

図 5 に非同期実行の場合の BioMoby クライアントプログラム例を示す。非同期実行と同期実行の方法の違いは、まず呼び出すライブラリが異なる。非同期実行の場合は、MOBY::Async::Serve を使い、サービスプロバイダーにアクセスアクセス後、submit → poll → result の順に実行する。その際に、受け渡す変数として、$EPR と@queryIDs があり、それぞれはエンドポイント(サービス提供先)の URL、クエリーID を示す。現在のところ、BioMoby において非同期実行環境が完成しているかは不明であり、一部完成しているドキュメントから本プログラム例を記述したが、実行の際にエラーが発生し、実行完了していない。

```
#!/usr/bin/perl

use MOBY::Client::Central;
use MOBY::Async::Service;
$ENV{'MOBY_SERVER'} = "http://moby.ucalgary.ca/moby/MOBY-Central.pl";
$ENV{'MOBY_URI'}    = "http://moby.ucalgary.ca/MOBY/Central";

$authURI     = "inb.bsc.es";
$serviceName = "runNCBIBlastpXML";

my $inputname1  = "sequence";
my $inputvalue1 = '
<AminoAcidSequence namespace="Swiss-Prot" id="P08641" >
<Integer namespace="" id="" articleName="Length">887</Integer>
<String namespace="" id="" articleName="SequenceString">
MGRRWGSPALQRFPVLVLLLLLQVCGRRCDEAAPCQPGFAAETFSFSVPQDSVAAGRELGRVSFAACSGRPWAVYVPT
DTRFKVNGDGVVSTKRPLTLYGRKISFTIYAQDAMGKRHSARVTVGRHRHRRHHHNHHLQDTTPAVLTFPKHDPGFLR
RQKRDWVIPPISCLENHRGPYPMRLVQIKSNKDKESKVYYSITGQGADSPPVGIFIIERETGWLEVTEQLDREKIDRY
TLLSHAVSASGQPVEDPMEIIITVMDQNDNKPVFIKEVFVGYIEENAKPGTSVMTVNATDADDAVNTDNGIVSYSIVS
```

```
QQPPRPHPQMFTIDPAKGIISVLGTGLDRETTPNYTLIVQATDQEGKGLSNTATAIIEVTDANDNIPIFNPTMYEGVV
EENKPGTEVARLTVTDQDAPGSPAWQAVYHIKSGNLDGAFSIITDPSTNNGILKTAKGLDYETKSRYDLVVTVENKVP
LSVPITLSTASVLVTVLDVNEPPVFVPPIKRVGVPEDLPVGQQVTSYTAQDPDRDMRQKITYRMGSDPAGWLYIHPENG
IVTATQPLDRESVHAINSTYKAIILAVDNGIPDTTGTGTLLLLLQDVNDNGPTPEPRSFEICSRQPEKQILSIVDKDLP
PHTYPFKAALEHGSSNNWTVEIRGQDELAMGLKKELEPGEYNIFVKLTDSQGKAQVTQVKAQVCECEGTAKNCERRSYI
VGGLGVPAILGILGGILALLILLLLLLLFARRRKVEKEPLLPPEDDMRDNVYNYDEEGGGEEDQDYDLSQLHRGLDARP
EVIRNDVAPPLMAAPQYRPRPANPDEIGNFIDENLKAADTDPTAPPYDSLLVFDYEGGGSEATSLSSLNSSASDQDQDY
DYLNEWGNRFKKLAELYGGGEDDE
</String>
</AminoAcidSequence>';

my $inputname2  = "expect_threshold";
my $inputvalue2 = '<Value>0.01<¥Value>';

my $m = MOBY::Client::Central->new;
my ($serv_instances, $reg) = $m->findService(
                                              authURI => $authURI,
                                              serviceName => $serviceName);

my $SI = shift @$serv_instances;

my $wsdl = $m->retrieveService($SI);

my $SERV = MOBY::Async::Service->new(service => $wsdl);

$SERV->silent(0);

@XMLinputlist = ([$inputname1, $inputvalue1,
                  $inputname2, $inputvalue2]);

#my $response = $SERV->execute(XMLinputlist => ¥@XMLinputlist);
```

```
my ($EPR, @queryIDs) = $SERV->submit(XMLinputlist => ¥@XMLinputlist);

my @status = $SERV->poll($EPR, @queryIDs);

printf "status: %s¥n",@status[0];

my @response = $SERV->result($EPR, @queryIDs);

$SERV->destroy($EPR);
```

図 5 サービス非同期実行のための BioMoby クライアントプログラム例

次に Moby Central に登録されているサービス及びオブジェクトクラスの現状をまとめる。現在登録されているクラス情報を取得するには、仕様上 Moby クライアントプログラムを利用すれば良いはずであるが、現在何らかの理由(オントロジーサーバの不備と考えられる)により取得できない。そのため、本調査では、BioMoby のオントロジーデータを参照している別のサイト Remora (http://lipm-bioinfo.toulouse.inra.fr/remora/)から取得し、代用した。また、BioMoby の仕組みは、さまざまな研究グループが加工、活用している。そのうちの 1 つ MOWServ (http://www.inab.org/MOWServ/)では、BioMoby のサービス及びオブジェクトクラス階層を加工し、より分かりやすい階層構造を提供しているので、このサイトの情報も比較対象として利用した。

図 6、**図 7** に BioMoby、MOWServ それぞれに登録されているサービスクラスの階層構造を示す。また、**図 8** に BioMoby に登録されているサービスクラスのうち、実際にサービスインスタンスに付与され利用されているクラスを示す。BioMoby と MOWServ のサービスクラス及びその階層構造の比較から分かることは、MOWServ のサービスクラス構造は、BioMoby を参考にしており、そのうち実際利用されることの多いクラスのみの残し、あまり使われることのないクラスを間引いて作成している点である。一方、BioMoby のサービスクラス構造は相対的に複雑になっており、特にトップ階層にフラットに登録されているクラスが多い。

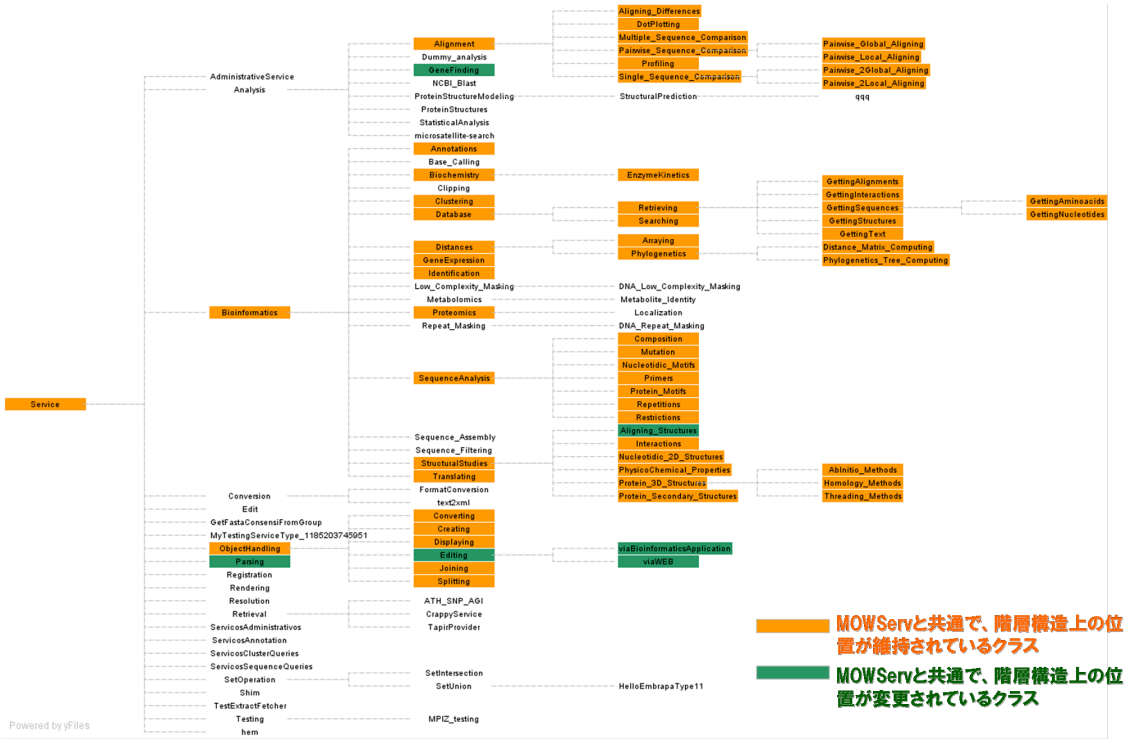

図 6 BioMoby のサービスクラス階層構造

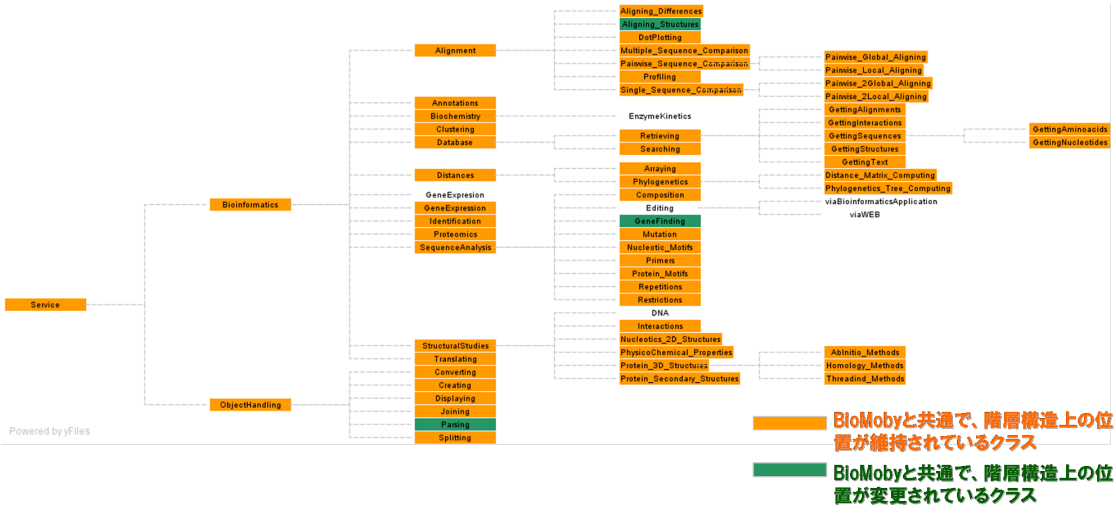

図 7 MOWServ のサービスクラス階層構造

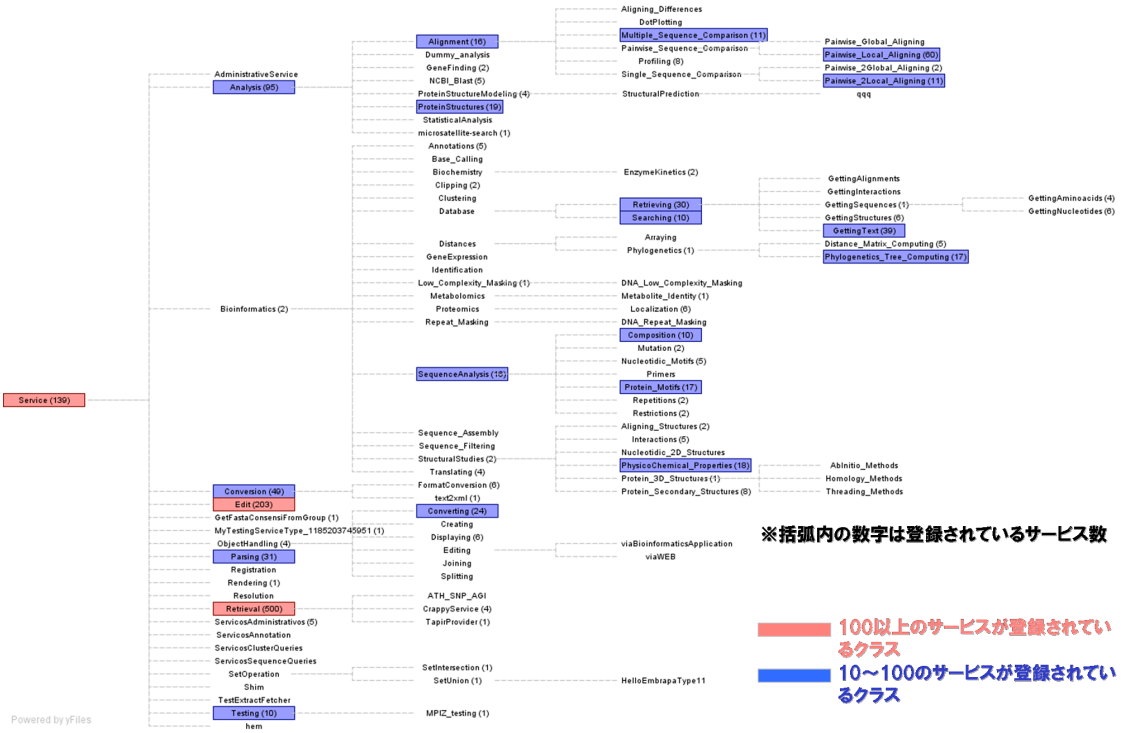

図 8 BioMoby において実際に利用されているサービスクラス

図 9、**図 10** に BioMoby、MOWServ それぞれに登録されているオブジェクトクラスの階層構造を示す。また、**図 11** に BioMoby に登録されているオブジェクトクラスのうち、実際にオブジェクトインスタンスに付与され利用されているクラスを示す。オブジェクトクラス階層についてはサービスクラス階層構造以上に複雑でフラットになっている。但し、配列オブジェクトのようにコンセンサスのとりやすいクラスについては比較的きちんと整理されている。一方、MOWServ のオブジェクトクラスは、サービスクラス構造と同様に、BioMoby のオブジェクトクラス構造を間引いて作成したものである。比較的簡略化されているものの、それでもトップ階層や text-formatted 階層下などがフラットにクラスが並べられている。メソッドの解析結果やデータベースファイルなどはこの text-formatted の下にぶら下がるが、サービス連携のためにそれらを整理、関連付ける試みはあまり行われていない印象を受ける。

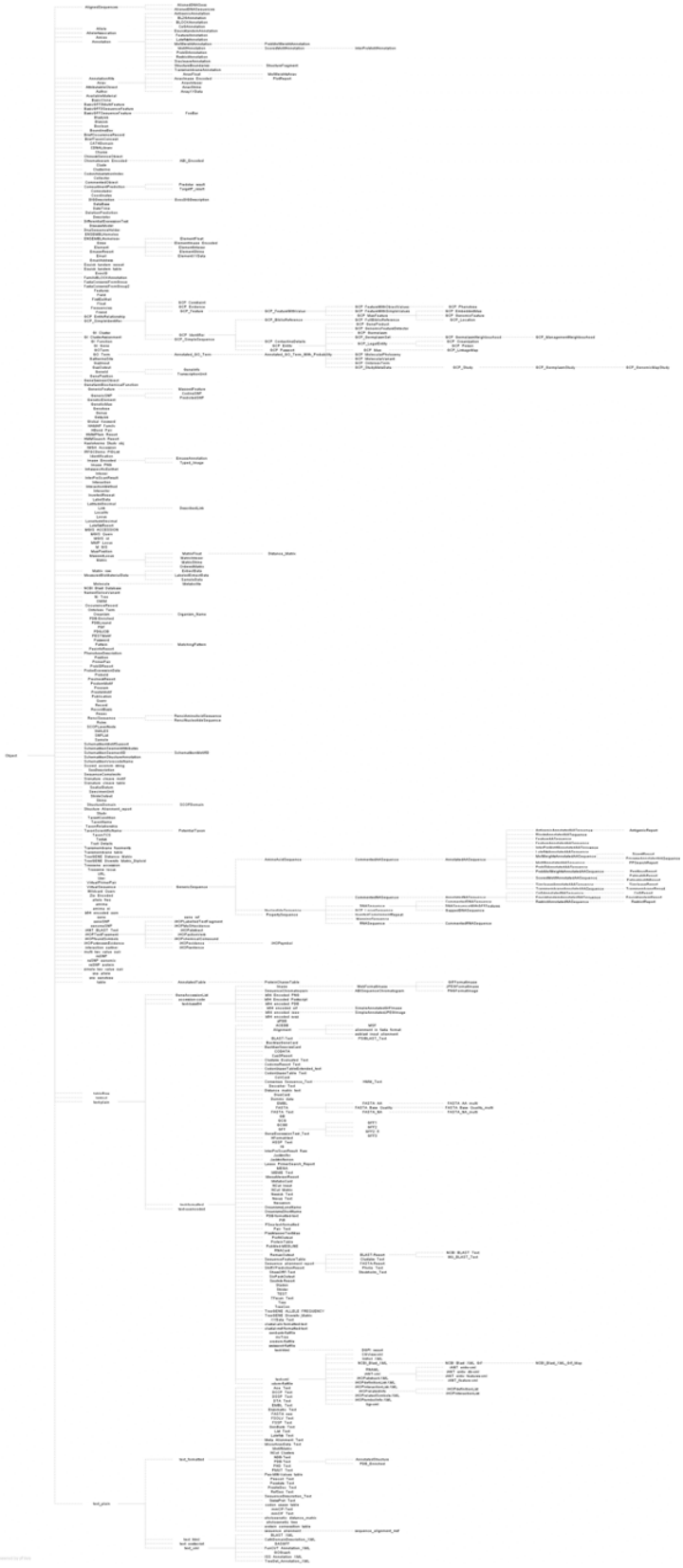

図 9 BioMoby のオブジェクトクラスの階層構造

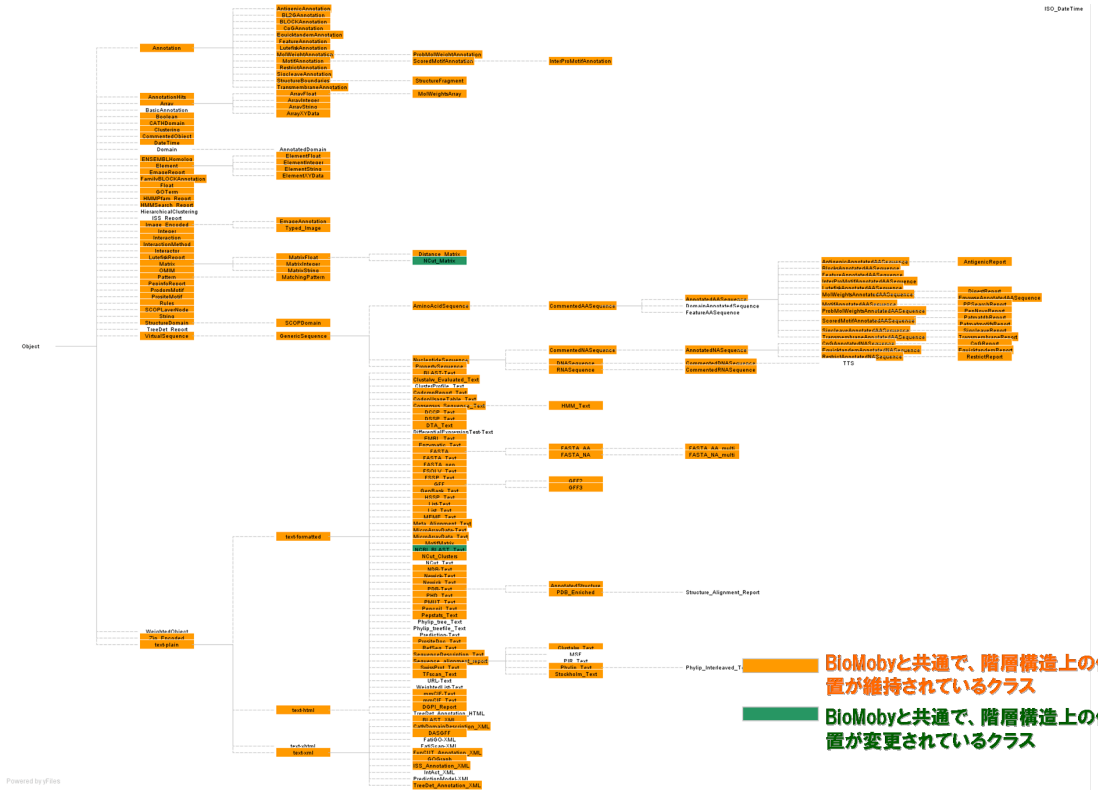

図 10 MOWServ のオブジェクトクラス階層構造

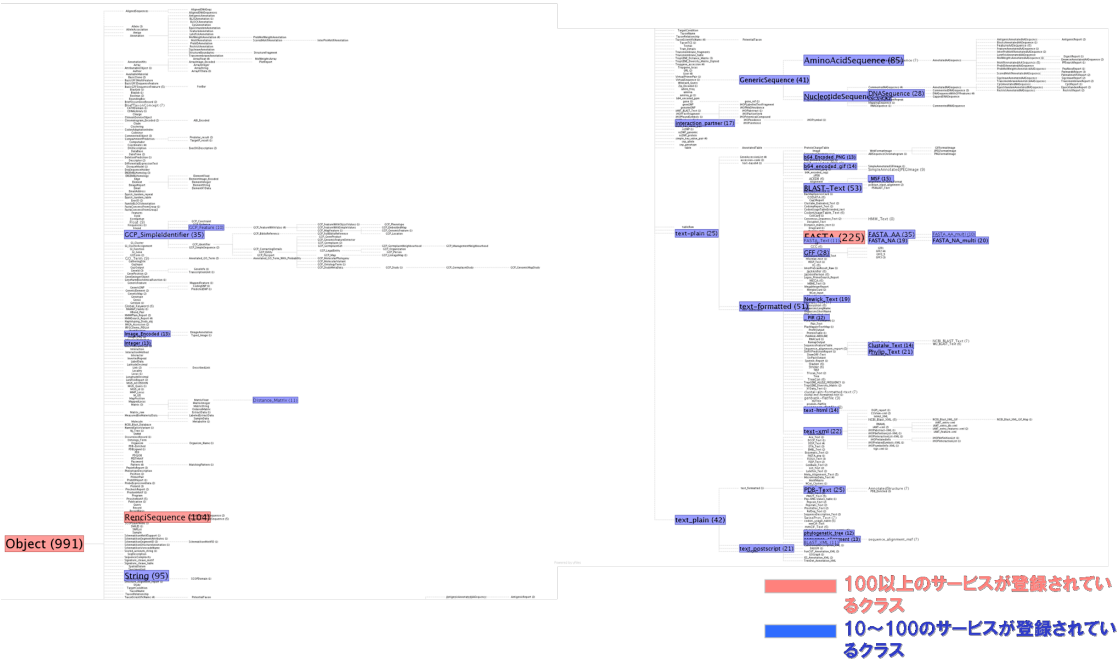

図 11 BioMoby において実際利用されているオブジェクトクラス

次に BioMoby のオブジェクト定義方法についてあらためて検討する。オブジェクトは､オブジェクトクラス（､データベースに登録されている場合はさらに名前空間と ID)によって定義されることになっている。オブジェクトクラスは、前述の図 9 に記載のものである。一方、名前空間、ID は、データベース名とその ID に当たるものであり、ユーザ固有のオブジェクトを定義するためには、あまり有効ではない。そのため、メソッドの入出力がどのような意味のどのような形態のものであるかを定義するには､オブジェクトクラスと、メソッドごとに個別に付けられた「名前」しか利用できない。例えば、ホモロジー検索メソッドの場合、問い合わせ配列のオブジェクトクラスとしては、moby:GenericSequence 下のクラスを指定する。しかしその配列とメソッドとの関係の意味は、メソッドごとに個別に振られた”query”や”sequence”などの「名前」に頼るしかなく、機械処理する上での課題として残る。

BioMoby におけるメソッドの命名方法についてまとめる。BioMoby ではサービスプロバイダーが自ら Moby Central にサービスを登録することになっている。そのため、基本的にプロバイダーごとに命名方法は異なっている。しかし、基本的には次のような傾向がある。

- 基本的にそのメソッドの機能を動詞、対象物などを並べて表現することが多い。
- その場合、ワード間の区切りが分かるように、”_”でつなぐか(“_”以外に記号で区切った例はない)、ワード先頭を大文字、それ以降を小文字にして表現している。
- 動詞として、”get”や”run”、”parse”、”convert”が用いられる。
- 対象物には、メソッドの入力や出力、使用するデータベース、ツール名などが表現されている場合が多い。
- 入力の表現は、その形式(例えば「文字列」)を表現するのではなく、メソッドに対する役割や解析上の意味(例えば､「検索キーワード」や､「ゲノム配列」)が表現されている。但し、データ形式変換などのメソッドの場合、変換元/先のデータ形式(例えば､「FASTA」など)が記述される。
- 著名なツールを利用している場合は、ホモロジー検索などの一般名でなく、BLAST のような具体的なツール名で記述している。

このような傾向を示す主な例とそのパターン出現数(母数は 1456)をまとめる。

主なメソッド 命名パターン	パターン 出現数	パターン例
Get ～	483	

Get ～ By ～	280	“Get [出力] by [入力]” となっている場合が多い。 getGeneInformationByEntrezGeneID カテゴリ: Retrieval 入力: geneid - a Simple of type moby:Object 出力: gene_info - a Simple of type moby:gene_ref getKeggPathwaysByKeggID カテゴリ: Retrieval 入力: keggID - a Simple of type moby:Object 出力: pathways - a Collection of moby:Object getUniprotIdentifierByGeneName カテゴリ: Retrieval 入力: geneName - a Simple of type moby:Object 出力: ids - a Collection of moby:Object getUniprotIdentifiersByKeyword カテゴリ: Retrieval 入力: keyword - a Simple of type moby:Object 出力: ids - a Collection of moby:Object
Get ～ From ～	74	“Get [出力] from [入力]” となっている場合が多い。 getConservedDomainLabelFromDomainId カテゴリ: Conversion 入力: id - a Simple of type moby:Object 出力: labels - a Collection of moby:Object getJpegFromAnnotatedImage カテゴリ: Parsing 入力: annotatedImage - a Simple of type moby:SimpleAnnotatedJPEGImage 出力: image - a Simple of type moby:b64_encoded_jpeg
Get ～ Of ～	11	“Get [出力] from [入力]” となっている場合が多い。
Get ～ For ～	9	“Get [出力] from [入力]” となっている場合が多い。
Get ～ As ～	4	“Get [出力] As [出力形式]”となっている。 getKeggPathwayAsGif カテゴリ: Conversion

		入力: pathway - a Simple of type moby:Object 出力: image - a Simple of type moby:b64_encoded_gif
Get ～ In ～	3	“Get [出力] In [データベース]”となっている。 getCrossReferencesInUniprot カテゴリ: Retrieving 入力: id - a Simple of type moby:Object 出力: ids - a Collection of moby:Object
Get ～ With ～	2	
Get ～ To ～	1	
Run ～	266	
Run ～ From ～	67	“Run [ツール] From [入力]”となっている場合が多い。
Run ～ Against ～	11	“Run [ツール] Against [DB/入力]”となっている場合が多い。
Run ～ For ～	4	“Run [ツール] For [入力]”となっている場合が多い。
Run ～ To ～	2	“Run [ツール] To [目的]”となっている。
Run ～ With ～	1	“Run [ツール] With [入力]”となっている。
Parse ～	22	
Parse ～ From ～	11	“Parse [出力] From [入力]”となっている場合が多い。
Parse ～ With ～	1	“Parse [出力] With [入力]”となっている。
Extract ～	11	
Extract ～ From ～	2	“Extract [出力] From [入力]”となっている。
Convert ～	10	
Convert ～ 2 ～	6	“Convert [入力] 2 [出力]”となっている。
Convert ～ To ～	3	“Convert [入力] To [出力]”となっている。
Retrieve ～	7	“Retrieve [出力]”となっている場合が多い。
Search ～	6	
Search ～ By ～	1	“Search By [入力]”。何を検索し出力するのか分からない。

Search ～ From ～	1	“Search [出力] From [入力]”となっている。
Search ～ In ～	1	
seqret ～	196	
seqret ～ 2 ～	192	“seqret [入力] 2 [出力]”となっている。
MOBYSHound ～	34	“MOBYSHound [入力] From [出力]”となっている。
from ～	32	“from [入力] to [出力]”となっている。
genefarm ～	13	genefarm 以降は”get～”などのような表現。(以下同様)
moby ～	10	
Phytoprot ～	6	
GBIFGet ～	6	
SUBA ～	6	
Evoc ～	5	
plantsp ～	5	

BioMoby における以上のようなメソッド命名規則の傾向を参考に、今後現れるであろうメソッドを命名する際に破綻が起こらないように配慮すると、以下のような命名ルールが望ましいと考えられる。

- メソッド名は、メソッドの①機能カテゴリ、②入力、③出力、④利用するツール、⑤データベースの情報が一目で分かるように表現するために、①から⑤を表現した動詞、名詞及びそれらをつなぐ前置詞から構成する。
- その際、ワード間の区切りが分かるように、”_”でつなぐ。
- 機能カテゴリは動詞表現し、”get”や”run”、”parse”、”convert”など、直感的に分かるワードを用いる。
- BLAST など著名なプログラムを利用している場合は、それが分かるようにメソッド名の中に入れる(“run_blastn”など)。
- メソッド名の入力、出力を表す部分は、メソッドに対する役割(query キーワードなど)や解析上の意味(ゲノム配列(短に ACGT から成る DNA 配列ではない)など)を表す表現を用いる。なお、入力、出力の形式については、基本的にメソッド命名の中に含めず、メソッドの入力及び出力の定義の部分で記述する(「3.8 メソッド入出力データの標準化」において後述)。

具体的な表記パターン例を挙げる。

- “Get_[出力]_by_[入力(キー)]” ([入力(キー)]=“ID”,”Keyword”など)

- “Get_[出力]_in_[データベース]
- “Get_[出力]_as_[出力形式]” ([出力形式]にはオブジェクトクラスを記述)
- “Get_[出力]_by_[入力(キー)]_in_[データベース]_as_[出力形式]”
- “Run_[解析ツール]_from_[入力(解析データ)]”
- “Run_[解析ツール]_against_[データベース]”
- “Run_[解析ツール]_from_[入力(解析データ)]_against_[データベース]”
- “Parse_[出力]_from_[入力]
- “Convert_[入力]_to_[出力]”

以上の BioMoby 調査から、本調査対象のウェブサービスメソッドについては、次のように整理するのが良いと考えられる。

- サービスクラスの付与には、可能な限り MOWServ のクラスに従い、不足しているクラス、構造があれば追加する。
- オブジェクトクラスの付与については、現状では MOWServ クラスに従い、連携を図っていくことが望ましい。しかし、解析結果クラスの分解、整理などに今後取り組んでいかなければサービス連携は難しい。
- メソッドの命名規則については、前述のようなルールに従って、そのメソッドの機能カテゴリ、入力、出力、利用ツール、データベースが何か、人が読んで理解できるように工夫するのがよいと考えられる。

上記提案のうち、サービスクラスの付与について、3.5 の独自分類を MOWServ クラスに整理し直す作業を行った。本作業は、メソッドごとに再分類するのではなく、既に設定した独自分類に対し、MOWServ クラスの対応表を作成することで実施した。**図 12** に対応表から抜粋したものを示す(詳細は「調査結果データ」あるいは納品物電子媒体を参照)。

1	Service Category	#operations	BioMoby Service Category
2	Analysis	185	Analysis
3	Analysis > Calculate Nucleic Composition	41	Composition
4	Analysis > Calculate Nucleic Composition > Calculate Codon Usage	6	Composition
5	Analysis > Calculate Nucleic Composition > Calculate Codon Usage > EMBOSS	6	Composition
6	Analysis > Calculate Nucleic Composition > Calculate Codon Usage > EMBOSS > cusp	6	Composition
7	Analysis > Calculate Nucleic Composition > Calculate GC Contents	5	Composition
8	Analysis > Calculate Nucleic Composition > Calculate GC Contents > EMBOSS	5	Composition
9	Analysis > Calculate Nucleic Composition > Calculate GC Contents > EMBOSS > geecee	5	Composition
10	Analysis > Calculate Nucleic Composition > EMBOSS	10	Composition
11	Analysis > Calculate Nucleic Composition > EMBOSS > cai	5	Composition
12	Analysis > Calculate Nucleic Composition > EMBOSS > chips	5	Composition
13	Analysis > Calculate Nucleic Property	31	PhysicoChemical_Properties
14	Analysis > Calculate Nucleic Property > EMBOSS	21	Composition
15	Analysis > Calculate Nucleic Property > EMBOSS > codcmp	11	Composition
16	Analysis > Calculate Nucleic Property > EMBOSS > cusp	5	Composition
17	Analysis > Calculate Nucleic Property > EMBOSS > syco	5	Composition
18	Analysis > Calculate Protein Property	73	PhysicoChemical_Properties
19	Analysis > Calculate Protein Property > Calculate Accessibility	3	PhysicoChemical_Properties
20	Analysis > Calculate Protein Property > Calculate Charge	5	PhysicoChemical_Properties
21	Analysis > Calculate Protein Property > Calculate Solvation	1	PhysicoChemical_Properties
22	Analysis > Calculate Protein Property > EMBOSS	56	PhysicoChemical_Properties
23	Analysis > Calculate Protein Property > EMBOSS > charge	6	PhysicoChemical_Properties
24	Analysis > Calculate Protein Property > EMBOSS > findkm	11	EnzymeKinetics
25	Analysis > Calculate Protein Property > EMBOSS > hmoment	5	PhysicoChemical_Properties
26	Analysis > Calculate Protein Property > EMBOSS > iep	5	PhysicoChemical_Properties

図 12 本業務独自サービス分類と MOWServ/BioMoby サービスクラスの対応(抜粋)

この本調査対象ウェブサービスの BioMoby サービスクラスへの再分類結果から、本調査対象にはどのようなサービスクラスがどれくらい含まれているかを俯瞰することができる。**図 13**、**図 14** に 3.5 で分類対象とした全サービスの分布状況を示す。また、**図 15**、**図 16** にそのうちの国内プロバイダーDDBJ、KEGG、PDBj のサービス分布状況を示す。

両図から分かるように、本調査対象プロバイダーがカバーするサービス機能は多岐に渡る。特に、配列解析クラス(SequenceAnalysis)の網羅性は高い。またアライメント(Alignment)、データベース検索(Database)クラスも多い。一方遺伝子発現(GeneExpression)や代謝(Metabolomics)クラスなどがない。今後調査対象を広げ、広範囲の分野をカバーしていくことが必要である。

国内 3 プロバイダー(DDBJ、KEGG、PDBj)については、主にデータベース検索やそれに伴う配列解析ツールが一部含まれているという状況であり、今後データベースサービスをより活用していくためには、それらをつなく解析ツールサービスや解析結果のパースサービスの充実が必要である。

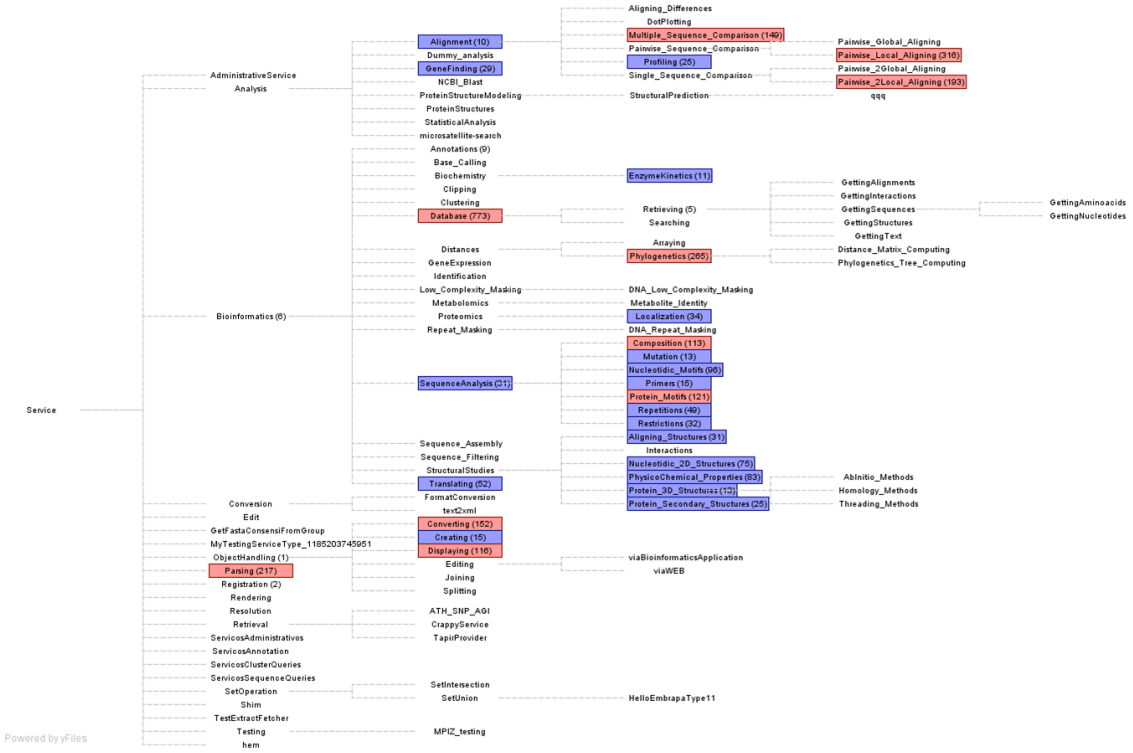

図 13 本調査対象ウェブサービスの BioMoby クラス分類

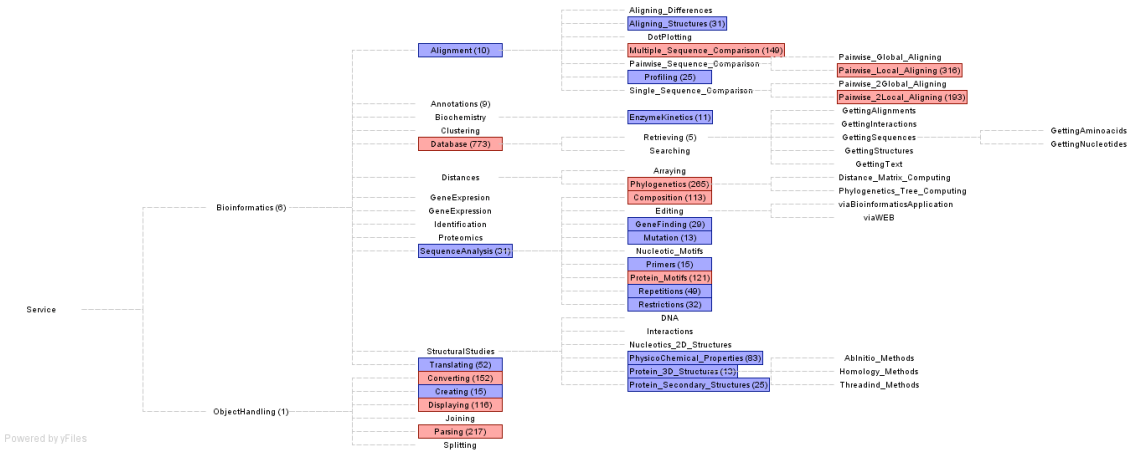

図 14 本調査対象ウェブサービスの MOWServ クラス分類

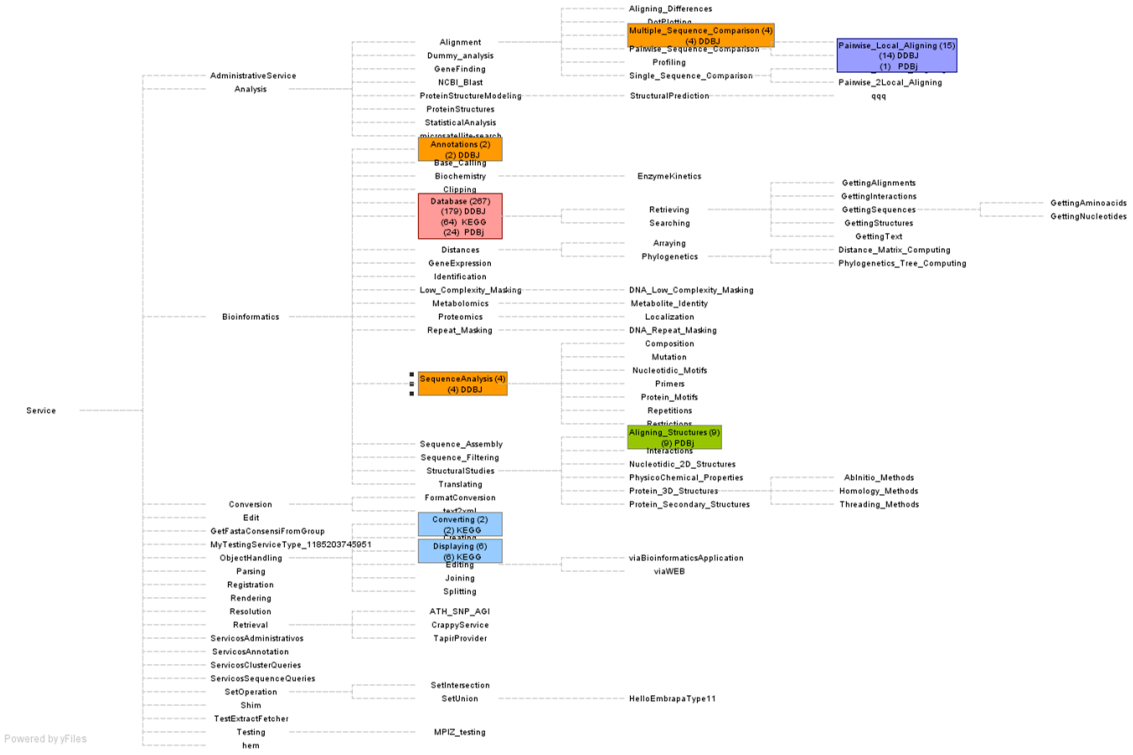

図 15 本調査対象のうち国内サービス(DDBJ、KEGG、PDB j ）の BioMoby クラス分類

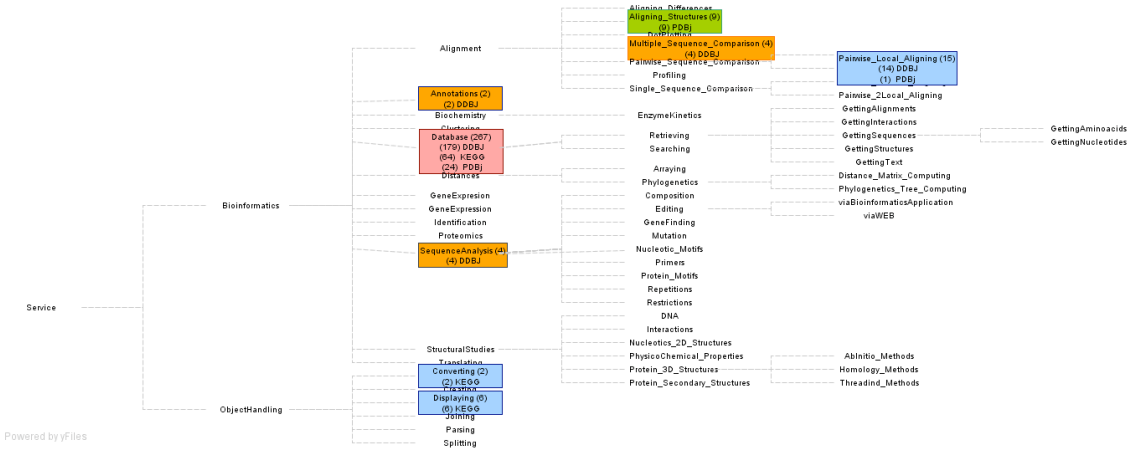

図 16 本調査対象のうち国内サービス(DDBJ、KEGG、PDBj)の MOWServ クラス分類

3.7 メソッド入出力データの分類

3.5 及び 3.6 で機能分類したメソッドに対し、ドキュメント整備の必要性の観点から優先順位を付け、優先順位の高いメソッドに絞りその入出力データの分類、整理を行った。

打ち合わせの過程において、3.6 の結果などをもとに優先順位の高いものは国内 3 プロバイダー(DDBJ、KEGG、PDBj)であるとのことから、これら 3 プロバイダーで提供されているメソッド(約 300 種)を入出力データの分類対象とした。

本作業は、打ち合わせの過程において、BioMoby のオブジェクトクラスをもとに分類することになったため、3.8 の標準化をまず行い、その標準化ルールをもとに分類した。

3.8 メソッド入出力データの標準化

3.7 のメソッド入出力データの分類結果をもとに、プロバイダー間で異なる構造、表記がなされているものを標準化するための仕様案を提案する。その際、3.6 と同様に、類似した試みが行われている BioMOBY などのプロジェクトを調査し、連携の可能性なども含めて、参考にした。

本作業は、打ち合わせの過程において、BioMoby のオブジェクトクラスをもとに分類することになったため、3.7 をやる前に実施し、その標準化ルールをもとに 3.7 の分類作業を行った。

MOWServ のオブジェクトクラスの特徴をあらためてまとめる。MOWServ のオブジェクトクラス構造は、**図 10** に示すような複雑なものである。このうち、配列オブジェクト(VirtualSequence)は比較的整理されている。一方、解析結果オブジェクトは、主に text-formatted 下に収められており、それがどのようなサブオブジェクトから構成されているかなどの分解は行われておらず、各解析ツールの結果オブジェクトがフラットに並べられているのが現状である。

BioMoby ではこのオブジェクトクラスを使って、入出力やパラメータの統一化を図ろうとしている。具体的には、このオブジェクトクラスと個別名(“query”など)、データベースエントリの場合、名前空間、ID によって、その意味(“アライメントした結果”など)と形式(“FASTA 形式”など)が定義される。しかし、オブジェクトクラスには、形式だけを定義したものと、意味だけを定義したもの、その両方が定義されたものが混合しており、統一性が取れていない。また、オブジェクトクラスでは意味が定義されていないものは「個別名」で人はその意味を理解するが個別名に付け方にルールはなく、統一化が必要である。

以上のことから、本作業では BioMoby のオブジェクトクラスを利用するものの、オブジェクトクラス・個別名・名前空間・ID 間における形式・意味定義の役割分担を新たに設定した。

オブジェクトクラス：	形式を指定することで自動的に意味が規定されるのを避ける必要はないが、基本的にオブジェクトの形式を表現する。
個別名：	オブジェクトのメソッド機能や、解析上の意味を表現する。例えば検索キーワードや、検索配列、ゲノム配列(短に ACGT からなる DNA 配列ではない)。
名前空間、ID：	ID と組みにして、データベースエントリを識別する。

3.9 メソッド実行テスト

3.7 の優先順位付けで絞り込まれたメソッドに対し、適当な入力データサンプルを用意し、メソッドの実行テストを行い、実行の可否などの結果を 3.10 のドキュメント作成でまとめる。

- 概要

本調査における実行確認においては、海外の主要なウェブサービスサイトである EBI(European Bioinformatics Institute: http://www.ebi.ac.uk) と NCBI(National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov)に焦点をあてることとした。これらのサイトは、世界的にも主要なライフサイエンス系ウェブサイトであること、ウェブサービスの提供は始められたばかりで、実用に耐えられるサービスかの判断が必要なこと、利用するために必要なドキュメント類が少ないこと等からこれら 2 つのサイトが提供しているウェブサービスについて調査を行い、各メソッドの実行確認を行った。

以下では、これら海外２つのウェブサービスサイトの概要、メソッド実行確認結果を示す。なお、EBI のウェブサービスは、①本業務期間中にいくつかのサービスが追加されたこと、②BETA 版の提供で実質的に利用できないサービスが含まれていること、③ドキュメントが少なすぎてサービス、メソッドの利用方法が不明なものがあること、④メソッド実行時エラー等により実行できなかったこと、により実行確認ができないサービス、メソッドが存在する（詳細はそれぞれの実行確認結果等を参照)。

なお、本調査は 2007 年 11 月～12 月に行われたものであるが、この期間中においても、EBI のウェブサービスは、ドキュメントの充実化、サービス、メソッドの動作安定向上、実行時エラーの解消等の改善が行われており、日々サービスの充実、向上が図られているようであった。

- EBI のウェブサービス概要

EBI の HP のメニュバーの Tools タブに Web Services があり、これを選択すると図 9-1 のように EBI で提供しているウェブサービスについての情報が得られる。

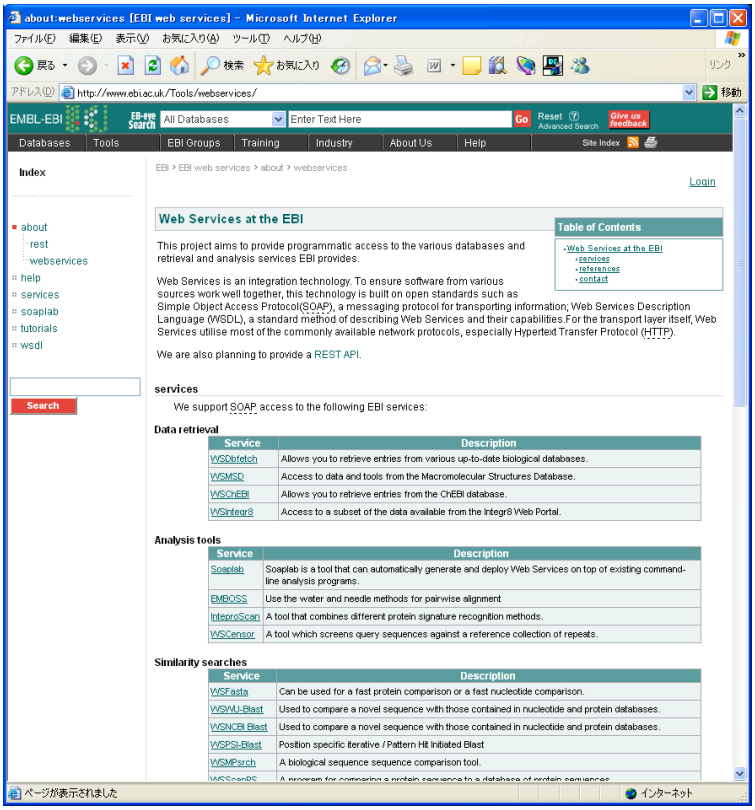

図 9-1：EBI 提供のウェブサービスについてのトップページ

2007 年 12 月現在、以下のウェブサービスが提供されている。

① Data retrieval

service	porttype	機能等
WSDbfetch	WSDBFetchServerLegacy	データベース名、ID を指定して当該エントリを取得することができる。エントリを取得する際のフォーマット名、出力形式名を指定することができる。ウェブサービスで利用できるデータベース名、フォーマット名、出力形式名リストを取得するメソッドも用意されている。
WSMSD	msd_soap_servicePortType	MSD(Macromolecular Structure

		Database)が提供しているデータやツールにアクセスすることができる。後述のSSMも含まれている。
WSChEBI	ChebiWebServicePortType	ChEBI(Chemical Entities of Biological Interest)情報を取得することができる。ChEBIオントロジーによる整理もされており、オントロジーによる関連（親、子）を参照することもできる。
WSIntegr8	Integr8	解読されたゲノムとこれらに関連するプロテオームについての情報を簡便、統合的にアクセスすることができる。用いているデータは次のようなものである。 DNA（EMBL, Genome Reviews, Ensembl) Protein(UniProt KB, IPI) 統計解析情報(InterPro, CluSTr, GOA)

② Analysis tools

service	porttype	機能等
Soaplab	各ツールごとにWSDLが定義されている。	EMBOSSのすべて（２０９個）のツールが提供されている。ツールごとにWSDLがある。
EMBOSS	WSEMBOSS	EMBOSSのウェブサービスである。本ウェブサービス内に提供しているツールのリストを取得するメソッドがあり、このメソッドの実行結果によれば141のツールが提供されている。こ

		れら 141 ツール名とツール実行時パラメータをウェブサービスのツール実行メソッドの引数としている。ツール実行時パラメータは、141 ツール共通の複合型となっており、97 属性で構成されている。各ツールで設定可能な属性は異なり、すべての属性がどのツールでも有効なわけではない。
InterproScan	WSInterProScan	いくつかのたんぱく関連解析ツールにより構成される InterProScan を実行することができる。ツールは以下の通りである。 blastprodom, hmmpanther hmmpir, hmmpfam, hmmsmart, signalp, hmmtigr, fprintscan, scanregexp, profilescan, superfamily, tmhmm
WSCensor	WSCensor	クエリ配列をリピートおよびマスクシンボルにしたがってスクリーニングすることができる。また、見つかったすべてのリピートについての分類情報が得ることができる。

③ Similarity searches

service	porttype	機能等
WSFasta	WSFasta	Fasta による配列検索を行うことができる。 ヒューリスティックなアルゴリズムを適用することで高速処理を実現している。
WSWU-Blast	WSWUBlast	WU-Blast による配列検索を行うことができる。ヒューリスティックなアルゴリズムを適用することで高速処理を実現している

WSNCBIBlast	WSNCBIBlast	NCBIBlast による配列検索を行うことができる。ヒューリスティックなアルゴリズムを適用することで高速処理を実現している
WSPSI-Blast	WSBlastpgp	PSI／PHI Blast による配列検索を行うことができる。位置特異的な繰り返し、特異的配列パターンによる配列検索を行う。
WSMPsrch	WSMPsrch	MPsrch による配列検索を行うことができる。Simth&Waterman アルゴリズムにより検索を行う。
WSScanPS	WSScanPS	ScanPS による配列検索を行うことができる。Simth&Waterman アルゴリズムにより検索を行う。

④ Multiple Alignment

service	porttype	機能等
WSClustalW	WSClustalW	ClustalW によりグローバルマルチプルアラインメントを行うことができる。系統樹情報を得ることもできる。
WSClustalW2	WSClustalW2	ClustalW のバージョン２を用いたものである。
WST-Coffee	WSTCoffee	T-Coffe によりマルチプルアラインメントを行うことができる。
WSMUSCLE	WSMuscle	Muscle によりマルチプルアラインメントを行うことができる。Muscle では、Log- Expectation による配列比較によるマルチプルアラインメントを行う。
WSKAlign	WSKalign	KAlign によりマルチプルアラインメントを行うことができる。KAlign は高速で正確なマルチプルアラインメントを行う。

WSMAFFT	WSMafft	Mafft によりマルチプルアラインメントを行うことができる。Mafft は非常に高速なマルチプルアラインメントを行う。 BETA 版による提供となっている。

⑤ Structural analysis

service	porttype	機能等
WSDaliLite	WSDaliLite	DaliLite によりペアワイズの構造比較を行うことができる。
WSSSM	msd_soap_servicePortType	SSM(Secondary Structure, Matching)は、３D のたんぱく質構造比較とアラインメント、比較した構造の最適 Cα アラインメント等を計算することができる。また、以下のような他のサービスとの連携も図られている。 MSD Motif, OCA, SCOP, GeneCensus, FSSP, 3Dee, CATH, PDBSum, SWISS-PROT, ProhoMap
WSMaxSprout	WSMaxsprout	MaxSprout は、たんぱく質の骨格および C-alpha trace から側鎖の座標を生成するための高速なデータベースアルゴリズムである。

⑥ Literature and ontologies

service	porttype	機能等
WSWhatizit	whatizit	Whatizit は、テキストマイニングを行うためのテキスト処理システムである。いくつか

		のパイプラインが用意されており、入力したテキストを選択したパイプラインにしたがって処理した結果を得ることができる。
WSCiteXplore	WSCitationImpl	citation データベースを検索することができる。
WSOntology Lookup	Query	オントロジー名（Gene Ontology 等）と ID（GO:0000022 等）から用語を検索したり、用語の一部からオントロジーを検索して用語を抽出することができる。また、用語の親や子を参照することもできる。オントロジーは 54 種ある。
WSSBO	SBOProvider	SBO(Systems Biology Ontology)の用語検索､親子参照等を行うことができる。
WSMIRIAM	MiriamProvider	バイオモデルの様々なアノテーション情報を管理するための（最低限の）ルールを定義し、これらリソースを参照できるようにしたシステムが MIRIAM であり、リソースのデータタイプ名、URI（Uniform Resource Identifier）の取得、実データエントリ URL の取得等を行うことができる。
Picr	AccessionMapperInterface	アクセッションまたはシークエンスから各種データベースへのマッピング情報の取得を

		行うことができる。 BETA 版による提供となっている。

• NCBI のウェブサービス概要

NCBI のウェブサービス HP は、NCBI のトップページに入り口を見つけることができず、http://www.ncbi.nlm.nih.gov/entrez/query/static/esoap_help.html を直接アクセスし、図 9-2 のような NCBI で提供している SOAP によるウェブサービスについての情報が得られる。

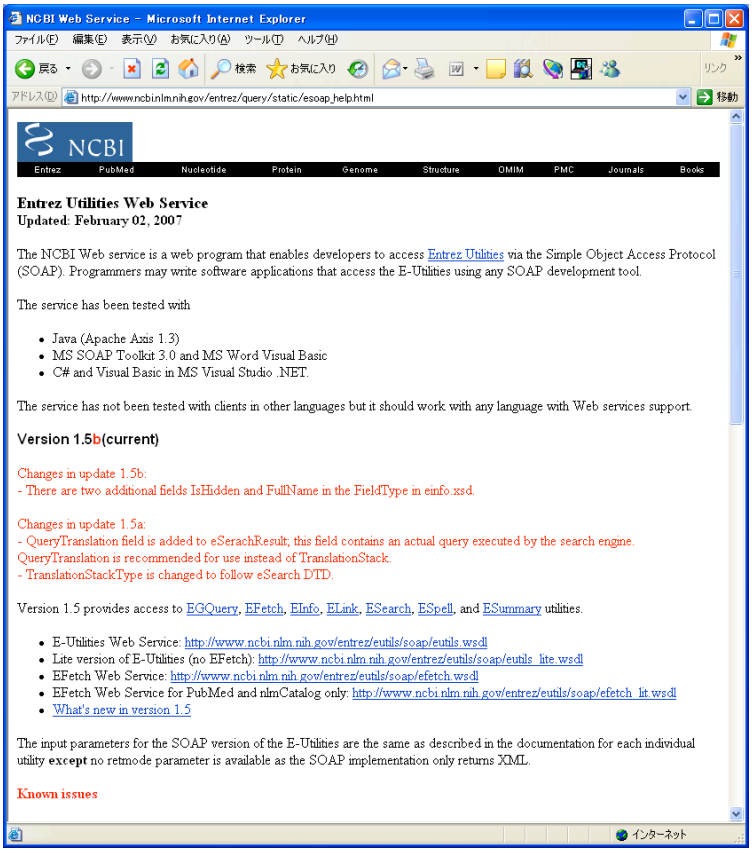

図 9-2：NCBI 提供の SOAP によるウェブサービスについてのトップページ

NCBI では、2007 年 12 月現在、以下に示す 4 つの WSDL を公開しているが、E-Utilities がすべての機能を持っているため、NCBI のウェブサービスについては、これについてのみ実行確認とドキュメント作成を行った。

- E-Utilities
  すべての機能を包含したもの

- Lite version E-Utilities

  E-Utilities から Efetch 機能を除いたもの
- EFetch

  EFetch 機能のみ
- Efetch for PubMed and nlmCatalog

  EFetch 機能のうち PubMed と nlmCatalog に対応する機能だけにしたもの

この E-Utilities には、7 種のメソッドがある（WSDL には、.NET 用メソッドとそれ以外用のメソッドが別々に同機能のものが定義されており、計 14 のメソッド定義がある）。

method	機能等
run_eGquery	1 つの検索式で Entrez の各データベースを検索し、それぞれのヒット数を取得することができる。 http://www.ncbi.nlm.nih.gov/sites/gquery に対応するメソッドである。
run_eInfo	Entrez の各データベースについて、フィールド名、インデックス数、最終更新日、有効なリンクを取得することができる。
run_eLink	primaryID から関連するリンクの存在をチェックすることができる。
run_eSearch	検索により primaryID を取得することができる。
run_eSpell	対象データベースを検索する際のスペリング候補を取得することができる。
run_eSummary	primaryID からドキュメントのサマリーを取得することができる。
run_eFetch	データベースレコードを取得することができる。

・実行確認

EBI と NCBI のウェブサービスのメソッド実行確認結果一覧を以下に示す。実行確認の詳細については、次節「ドキュメント作成」で作成したドキュメント内にメソッドの機能等とともに記載した。

- EBI のウェブサービス実行確認結果

EBI のウェブサービスについて、下記一覧の備考欄に特別の記載がない場合には、perl(v5.8.7)+SOAP::Lite(0.60) @ Linux(SUSE10.0)の環境で実行確認を行った。

① Data retrieval

service	実行確認結果	備考等
WSDbfetch	◎	特に問題ない。
WSMSD	×	BETA 版でサービス提供している。Perl によるクライアントプログラムがダウンロードできるが、最初のメソッド(msdSSM)呼び出しでエラーとなる。
WSChEBI	◎	特に問題ない。
WSIntegr8	◎	特に問題ない。

② Analysis tools

service	実行確認結果	備考等
Soaplab	－	EMBOSS は本調査の対象外とした。
EMBOSS	－	EMBOSS は本調査の対象外とした。
InterproScan	○	メソッド doIprscan は実行できなかった。 入力パラメータ trlen, trtable, goterm, outformat の設定有効値、機能が不明である。
WSCensor	－	最近に追加されたウェブサービスであり、本調査の対象を検討する際にはなかったため実行確認調査を行っていない。

③ Similarity searches

service	実行確認結果	備考等
WSFasta	○	実行確認できた program は、fasta3, fastx/y3 である。fastf3, fasts3, ssearch は実行できなかった。 入力パラメータ histogram, outformat の設定有効値が不明である。
WSWU-Blast	◎	入力パラメータ echofilter, outformat の

		設定有効値が不明である。
WSNCBIBlast	◎	特に問題ない。
WSPSI-Blast	○	PHI-BLAST モードは実行できなかった。 入力パラメータ numal は WSDL 上には存在しないが有効である。 入力パラメータ sensitivity の設定有効値が不明である。 入力パラメータ appxml の機能、設定有効値が不明である。 入力パラメータ usagemode, pattern sequence は、PHI-BLAST 用であり設定値等の確認はできなかった。
WSMPsrch	◎	入力パラメータ sort の設定有効値が不明である。
WSScanPS	◎	入力パラメータ matrix は、WSDL 上では必須パラメータとなっているが、設定しなくても BLOSUM50 がデフォルト値として設定されており実行可能である。 入力パラメータ toponly の設定有効値が不明である。

④ Multiple Alignment

service	実行確認結果	備考等
WSClustalW	○	メソッド align, alignIds, tree は実行できなかった。 入力パラメータ score の設定有効値が不明である。 入力パラメータ gapdist, cpu のデフォルト値が不明である。
WSClustalW2	○	入力パラメータ score, iteration, clustering, numiter の設定有効値が不明である。

		入力パラメータ cpu のデフォルト値が不明である。
WST-Coffee	○	入力パラメータは WSDL 上は WSClustalW と同等であるが、有効なのは matrix, outorder, async, email である。
WSMUSCLE	○	入力パラメータ outputtree の設定有効値が不明である。 入力パラメータ outorder は WSDL 上には存在しないが有効である。
WSKAlign	×	ツール実行メソッドの戻り値(jobid)が空であり実行結果を取得できない。
WSMAFFT	○	メソッド align, alignIds は実行できなかった。 入力パラメータ pair の機能、設定有効値が不明である。 入力パラメータ localpair, globalpair, genafpair の機能が不明である。 入力パラメータ reorder による順序基準が不明である。

⑤ Structural analysis

service	実行確認結果	備考等
WSDaliLite	○	メソッド getIds は実行できなかった。 入力パラメータ chaind1, chaind2, outformat は設定有効値が不明である。
WSSSM	－	最近に追加されたウェブサービスであり、本調査の対象を検討する際にはなかったため実行確認調査を行っていない。なお、本ウェブサービスは、WSMSD の一部のようである。
WSMaxSprout	×	ツール実行メソッドの戻り値(jobid)が空であり実行結果を取得できない。

⑥ Literature and ontologies

service	実行確認結果	備考等
WSWhatizit	×	ツール実行メソッドの戻り値(jobid)が空であり実行結果を取得できない。 JAX-WS を利用するクライアントプログラムがダウンロードできるようになっており、JAX-WS 環境であれば実行できる可能性がある。
WSCiteXplore	×	ツール実行メソッドの戻り値(jobid)が空であり実行結果を取得できない。 JAX-WS を利用するクライアントプログラムがダウンロードできるようになっており、JAX-WS 環境であれば実行できる可能性がある
WSOntology Lookup	△	perl による実行環境では実行できなかったが、java(v1.4.2)+axis1(v1.4)+WSDL2Java の環境で実行確認を行うことができた。 メソッド getTermsByAnnotationData は、適切な設定値が不明であり実行結果を得ることができなかった。
WSSBO	○	perl による実行環境では実行できなかったが、java(v1.4.2)+axis1(v1.4)+本ウェブサービスサイトからダウンロードできるライブラリとクライアントプログラムで実行確認を行うことができた。
WSMIRIAM	◎	特に問題ない。 URN は世界的に登録は少なく、利用する際には URL を用いる必要がある。
Picr	－	最近に追加されたウェブサービスであり、本調査の対象を検討する際にはなかったため実行確認調査を行っていない。

－NCBI のウェブサービス実行確認結果

NCBI のウェブサービスについては、java1.4.2+axis1(v1.4)+WSDL2Java @

Linux(SUSE10.0)の環境で実行確認を行った。

method	実行確認結果	備考等
run_eGquery	◎	特に問題ない。
run_eInfo	◎	特に問題ない。
run_eLink	○	例として示されているクライアントプログラムでは、nucleotide から protein へのリンクを検索するものであるが、実行エラーとなる。Nucleotide を nuccore にした場合には、実行でき結果も得られた。
run_eSearch	◎	特に問題ない。 PubMed Central で実行確認を行った。
run_eSpell	◎	特に問題ない。 PubMed で実行確認を行った。 例として示されているクライアントプログラムでは、引数の順番が db, tool, term, email というものを仮定したものと思われるが、実際の引数の順序は、db, term, tool, email であり、引数の順序を変更することで、正しい結果を得ることができた。
run_eSummary	◎	特に問題ない。 nucleotide で確認を行った。
run_eFetch	◎	特に問題ない。 PubMed と taxonomy で実行確認を行った。

run_eFetch は、NCBI Entrez のデータベースエントリを取得するメソッドであるが、戻り値はひとつの複合型となっている。その複合型は、対象とするデータベースのデータをいくつかのカテゴリに分類し、それぞれのカテゴリに対応するような複合型が定義されている。したがって、取得するエントリは、これらカテゴリごとに定義された複合型の該当する属性に納められることとなる。以下に run_eFetch の戻り値（複合型）を構成する複合型を示す。

番号	複合型名	
1	NLMCatalogRecordsSetType	NLM カタログ情報
2	PubmedArticleSetType	PubMed 情報
3	pmc-articlesetType	PMC 情報
4	Mim-entries	MIM 情報
5	TaxaSetType	Taxonomy 情報
6	TseqSet	配列情報
7	GBSet	GenBank 情報
8	Bioseq-setType	配列および構造等の配列に関連する生物学的情報
9	Entrezgene-Set	Entrez 遺伝子情報
10	ExchangeSet	不明
11	IdListType	ID 情報(何の ID か不明)

次に取得するエントリのデータベースにより上記のどの複合型にデータが得られるかを以下に示す。また、いずれのデータベースエントリを取得しても、関連するデータベースへのリンク情報（ID 等）を得られる。なお、複合型については、データがない場合には、空が戻されることになっており、複合型へのアクセス時に空がどうかを判定した後に、メンバーへのアクセスを行う必要がある。

	1	2	3	4	5	6	7	8	9	10	11
pubmed		◎									
pmc			◎								
journals	◎										
omim				◎							
gene						○	○	○	○	○	
genome						○	○	○	○	○	
nucleotide						○	○	○	○	○	
nuccore						○	○	○	○	○	
nucest						○	○	○	○	○	
nucgss						○	○	○	○	○	
protein						○	○	○	○	○	
popset						○	○	○	○	○	
snp						○	○	○	○	○	

sequences						○	○	○	○	○	
taxonomy					◎						

3.10 メソッドドキュメントの作成

3.9 で実行確認できた以下のウェブサービスについて日本語によるドキュメント作成を行った。

- EBI
  - ① WSDbfetch
  - ② WSChEBI
  - ③ WSIntegr8
  - ④ InterproScan
  - ⑤ WSFasta
  - ⑥ WSWU-Blast
  - ⑦ WSNCBIBlast
  - ⑧ WSPSI-Blast
  - ⑨ WSMPsrch
  - ⑩ WSScanPS
  - ⑪ WSClustalW
  - ⑫ WSClustalW2
  - ⑬ WST-Coffee
  - ⑭ WSMUSCLE
  - ⑮ WSMAFFT
  - ⑯ WSDaliLite
  - ⑰ WSCiteXplore
  - ⑱ WSOntology Lookup
  - ⑲ WSSBO
  - ⑳ WSMIRIAM
- NCBI
  - ① eutils

ドキュメントの作成では、各メソッドの機能と入出力パラメータ、メソッドの入出力で用いられている複合型の属性とその内容、サンプル等について記述を行い、統合ウェブサービスの構築に資する内容となるように作成した。なお、作成したドキュメントは、「調査結果データ一式」に納めた。

4. 調査結果のまとめ

本調査では、国内外の主要なウェブサービスについて、各サービスで提供されている機能を調査し、それぞれのメソッドの解説と使用方法をドキュメント化するとともに、メソッドの目的ごとに階層的な分類を行った。この結果から、各サーバーの全てのメソッドを統一的に扱うために破綻しない命名規則を検討し、さらに各サービスでやりとりされるデータフォーマットの中で共通化できるものの種類を調査し、データ型の標準化方法を検討した。

ここでは、各調査・検討項目で挙がった、特に注目すべき結果をまとめる。

- 本調査では、SOAP で実現されている国内外のウェブサービスを網羅的に調査した。その結果、非常に多くのサービスが提供されており、種類も非常に多いことが分かった。
- しかし、比較的近年解析が行われ始め、まだ定型的なサービスとなりにくい分野のメソッドが不足していることが分かった。まずはこれらをサービス化し、他のサービスとの連携を図っていくことが必要である。
- 一方、国内の主要プロバイダーDDBJ、KEGG、PDB j については、データベース検索やその配列を利用したアライメントが大半であり、データサービス間をつなぐ解析サービスや解析結果のパースサービスなどの充実が望まれる。
- 国内外のウェブサービスの統一化を目的とした試みとしてはBioMobyが進んでおり、BioMoby で既に実現されている部分はそれを利用し、実現されていない部分は補充するとととともに、BioMoby グループや世界各国のグループととも連携し、統一化を図っていくことが望まれる。
- BioMoby で既に実現されている部分としては、サービスクラスの階層構造があり、これに従いつつ、不足している部分の補充、未整理部分の整理などを行っていくのがよい。
- 一方、BioMoby で実現されていない部分としては、オブジェクトクラスの整理が挙げられる。BioMoby ではオブジェクトクラスが定義されているものの、整理ができておらず、また最も課題となる解析結果のオブジェクト分解なども行われておらずサービス連携が難しい。また、オブジェクトの形式と意味をどのように定義するかも不十分に思われ、この部分で貢献していくことは重要かと思われる。
- サービスの実行テストを行ったが、実行できないサービスもいくつかあった。これはサービスプロバイダー側の問題とクライアントプログラム側の問題があり、関係者とも連絡をとり、少しずつでも解消していくことが必要である。