特別研究報告書

言語資源のラッピング支援システムの開発

指導教員　　石田 亨 教授

京都大学工学部情報学科

後藤 雅樹

平成19年2月9日

言語資源のラッピング支援システムの開発

後藤 雅樹

内容梗概

現在，インターネット上には様々な言語資源が存在する．これらの連携を目標とし，言語グリッドプロジェクトが進められている．既存の資源を言語グリッドで利用する場合は，資源ごとにラッパーと呼ばれるプログラムを作成し，資源の種類ごとに共通のインタフェースを外付し Web サービス化する．例えば，Web ページで提供される用例対訳集のラッパーは，検索要求に応じてページから対訳情報を切り出し，インタフェースに沿う形式に整形してユーザに返す．

言語資源に対しラッパーを作成する作業を，ラッピングと呼ぶ．現在のラッピングには効率的でない作業が多く，グリッド上の資源の早期充実の妨げとなっている．また前提とする知識も多く，利用したい資源を自らラッピングしようと考えるエンドユーザにとって，参入の障害となっている．

この問題を解決するため，本研究ではエンドユーザにも効率的なラッピングが行える支援システムを開発する. 本システムでは，言語資源の中でも困難な作業を含む，Web ページで提供される用例対訳のラッピングに焦点を当てる. 支援システムの開発に際し，以下の 3 つの課題が生じる．

1. 言語資源のラッピング作業の負荷箇所の特定
   言語資源のラッピング作業を効率的に支援するためには，作業のどの部分が時間的コストを要するか，また技術的にエンドユーザの負担となるかを把握しなければならない．その上でラッピング作業全体内の支援すべき部分を見定め，支援方法を決定する必要がある．
2. プログラミング開発環境に関する知識の不要化
   現状の言語資源のラッピングは，Web サービスプログラミング作業であり，専用の開発環境の使用を前提としている．そのため，エンドユーザがラッピングを行う際には，開発環境の操作を学習し，コードなどの編集にも慣れる必要がある．こういったエンドユーザの負荷の軽減が必要となる．
3. 対訳を切り出すルールの容易な記述
   Web ページで提供される資源のラッパーには，ページから対訳を切り出すためのルールを記述する．しかし既存のラッパー生成支援システムが採用

するタグ名に基づくルールでは，切り出しの難しい資源が存在する．このため本研究では，柔軟な記述が可能な，正規表現に基づくルールを採用する．しかしこのルールの記述には，HTML ソースの参照，正規表現の記述など，エンドユーザに負荷の掛かる作業が存在し，支援が要求される．

本研究では，ラッピング作業の分析に基づき上記の課題の解決を図った．まずラッピング経験の蓄積から作業手順を標準化した. これと実際の作業のビデオ記録から，ラッピングのタスクモデルを整理し，下位作業ごとの所要時間を得た．このモデルの分析と所要時間から，支援すべき部分を特定した．

次に，支援システムの大まかなアーキテクチャを決定した．サーバクライアントモデルを採用し，コーディング処理や Web サービス化処理をサーバで行うことで，クライアント側ではプログラミング開発環境が不要となり，エンドユーザに負荷の少ない設計が可能となった．

切り出しルールの記述では，HTML ソースの参照，正規表現の記述などの操作を，ブラウザやテキストエリア上で，エンドユーザの使い慣れたマウス操作を用いて行えるようにした．これにより，高い表現力を持つ切り出しルールを，エンドユーザが容易に記述することが可能となる．

本研究の貢献は以下の３点となる．

1. ラッピング作業の分析に基づいた負荷箇所の特定
   現状のラッピング作業の分析から，環境の導入，ラッパー検証テストの記述，正規表現などの記法の学習といった，エンドユーザの負荷となる作業を洗い出した．またラッピング作業のビデオ記録から，環境の導入，検証テストの記述，切り出しルールの記述，ファイルの記述におけるミスタイプなど，ラッピング作業者にとって時間を要する作業を特定した．
2. プログラミング開発環境に関する知識の不要化
   Web サービス作成に必要な環境をサーバに置くことにより，エンドユーザがプログラミング開発環境の学習をしなくともラッピングを行えるようになった．
3. 柔軟な情報切り出しルール記述支援の提案
   現状の作業に基づき，高い表現能力を持つ切り出しルールの記述を支援するユーザインタフェースを開発した．マウスでの範囲選択による HTML 取得や，選択肢からのルール修正が可能となる．その結果，切り出しルールの記述に関するユーザの学習負荷が軽減される．

Development of a Wrapping Support System for Language Resources

Masaki GOTOU

Abstract

Today, there exist a lot of online language services including bilingual dictionaries, parallel-texts and machine translators. The Language Grid is a language infrastructure on which people can combine language services as they need.

Language resources on the grid must implement standard interfaces, otherwise be wrapped with a web service called 'wrapper' that provides standard interfaces. 'Wrapping' means an activity to generate a wrapper for an existing resource. For example, a wrapper for a language resource provided on web pages works as below. It gets a request, extracts language data from the pages, and finally returns formed data.

Wrapping a resource is a time-consuming and skill-requiring activity, which is a hurdle for the widespread use of the grid. Moreover, it loses end-user's motivation to add resources they need to the grid. In order to solve this problem, this research proposes a Wrapping Support System for Language Resources. In developing the system, the following problems must be considered.

1. Identifying tasks to support

   To support wrapping activities efficiently, it is necessary to identify which parts of the activity are time-consuming or skill-requiring, and decide which part and how to support.

2. Elimination of required knowledge about programming environments

   Currently, programming development environments are necessary for language resource wrapping. For this reason, end-users have to study the manner of operation, and adapt to edit source codes, setting files etc. These hurdles for end-users shall be eliminated.

3. Supporting of describing data extraction rule

   Rules to extract parallel texts from web pages are described in the wrapper for the parallel text pages. Some of existing wrapping support systems use a tag name based rules, but there exist resources hard to deal with using those rules. Now therefore, my system adopts regular expression based

rules to deal with such resources. However, end users should have a lot of trouble writing that type of rules, which should be supported.

This research worked on the above issues basing on the analysis of observed wrapping activity. To start with, I described the outline of the standard wrapping process as a manual, and video recorded a real work in order to get the detailed task model of the process and got the time required for each part of the process. Using the result, I found out which part of wrapping to support.

In regard to the elimination of programming environment, my system adopts the client/server model. The server implements the coding and building functions. This means no programming environment is needed on the client side, which eliminates end-user training on the programming environment.

To support describing data extraction rule, I designed an user interface with which users can get the corresponding HTML source code from a rendered content, and can describe regular expression based rules from the code. All these can be accomplished by simple mouse action. This allows users to describe flexible extraction rules easily. The contribution of this research is as follows.

1. Identifying problematic processes based on the wrapping activity analysis
   Through the analysis of wrapping activity, this paper identified tasks which seem to be hurdles for end users, in particular, introduction of programming environment, describing wrapper verification tests and study of notation of regular expression etc. In addition, time-consuming tasks such as introduction of programming environment, describing wrapper verification tests and data extraction rules, and discovery of typing error in setting files, are observed through the analysis of the video record.
2. Wrapping environment without programming environment
   With the web service development environment put on the server, end users can do wrapping with no study on programming environment.
3. Support for writing flexible data extraction rule
   I proposed an user interface that helps users to write flexible extraction rule. With it, users can get HTML source with simple mouse operations, and refine a rule with selecting options. As a result, the task of writing extraction rule got easier.

言語資源のラッピング支援システムの開発

目次

第 1 章　はじめに　1

第 2 章　Web サービスラッピング　3

2.1　言語資源の Web サービスラッピング 3

2.1.1　用例対訳集 3

2.1.2　ラッピング作業 3

2.2　既存のラッピング支援手法 6

第 3 章　ラッピング作業の分析　8

3.1　タスクモデルの作成 8

3.1.1　ラッピング環境の準備 9

3.1.2　Java のプロジェクト，コードの作成 10

3.1.3　ラッパーの検証テストの記述 10

3.1.4　設定ファイルの記述 11

3.1.5　ラッパーの Web サービス化，検証テストの実行 12

3.2　作業時間の分析 13

第 4 章　サーバ型の開発環境　14

4.1　開発環境の操作負荷 14

4.2　システムアーキテクチャ 15

第 5 章　データの切り出しルール作成の支援　16

5.1　切り出しルールの作成負荷 16

5.2　切り出しルールの生成手法 18

第 6 章　ラッピング支援システム　20

6.1　システム全体の構成 20

6.2　データの切り出し支援部の実装 22

第 7 章　考察　24

第 8 章　おわりに　25

謝辞　27

参考文献 **27**

付録 **A-1**

A.1 言語資源ラッパーに関するファイル A-1

A.1.1 ラッパーテンプレート A-1

A.2 作業分析に関する資料 A-4

A.2.1 作業の所要時間の詳細なデータ A-4

A.3 支援システムの主なソースコード A-5

A.3.1 HTML 取得支援のためのサーバ側処理 A-5

A.3.2 HTML 取得支援のための JavaScript A-6

A.3.3 クライアント側 HTML 取得支援画面 A-8

A.3.4 プロジェクト支援のためのサーバ側処理 A-13

A.3.5 ルール修正支援のためのサーバ側処理 A-16

A.3.6 切り出し結果確認支援のためのサーバ側処理 1 . . . . A-18

A.3.7 切り出し結果確認支援のためのサーバ側処理 2 . . . . A-21

A.3.8 クライアント側 MainWindow A-24

第1章　はじめに

現在インターネット上には辞書や機械翻訳といった言語サービスが多く存在する．言語サービスとは，言語解析や機械翻訳などの言語処理機能と，用例対訳や辞書などの言語資源の総称である．これらを連携させることを目標とし，言語グリッドプロジェクト [1] が進められている．言語グリッドとは，インターネット上の言語サービスを自由に組み合わせ，新たな言語サービスを産み出す枠組である．さらに，エンドユーザが新たな言語サービスをグリッド上に追加することで，自らの必要とするサービスを作り出すことができる．

言語グリッドでは，言語サービス間の連携を容易にするため，言語サービスに汎用的に適用可能なインタフェースを定義している．しかし既存の言語サービスの多くは，言語グリッドで定めたインタフェースに適合していない．そういった言語サービスをグリッド上で利用可能とするためには，サービスごとにラッパーと呼ばれる Web サービスを作成し，特定のインタフェースを外付けする必要がある．ラッパーはユーザからの要求を受けて対応する言語サービスにアクセスし，得た情報をインタフェースに沿う形式に整形してユーザに返すプログラムである．言語サービスに対し，それを処理するラッパーを作成する作業をラッピングと呼ぶ．

現在このラッピング作業は，ラッピングボランティアと呼ばれる作業者により行われている．その主な目的は，まずその部品として使えるサービスを充実させ，言語グリッド上での新たな言語サービスの開発を促すことである．一方で，自身のタスクに合わせた言語サービスを開発するため，エンドユーザがラッピング作業を行う場合も考えられる．しかしこのような作業者は，プログラミングや言語グリッドに関する専門的な知識や経験を持つとは限らない．

これまで数人のラッピングボランティアにより 50 程度の言語サービスについてラッピングを行った．ところがそこで行われた工程には，多くの時間を要する効率的でない作業が存在した．また専門的な知識を要するなど，エンドユーザの負荷となりそうな作業も多く存在した．

ラッピングは Web サービスプログラミング作業であるため，作業のためにはプログラミング開発環境を導入し，その操作に慣れる必要がある．これはエンドユーザにとって大きな負荷となる．

また特に言語サービスが HTML で提供されている場合，ラッパーは HTML

からの言語情報の切り出しを行うが，その切り出し方法を記述する作業が困難であった．早期に言語サービスを充実させ，エンドユーザが新たなサービスをグリッドに追加できるよう，ラッピング作業における人間の作業量を減らし，専門的な前提知識を不要にする必要がある．

このような動機から本研究では，言語資源の中でもラッピングが困難であった，HTML で提供される用例対訳集のラッピング作業を支援するシステムの開発を目的とする．このため，以下の課題に取り組む．

1. 言語資源のラッピング作業の負荷箇所の特定
   言語資源のラッピング作業を効率的に支援するためには，作業のどの部分が時間的コストを要するか，またエンドユーザの負担となるかを把握しなければならない．その上でラッピング作業全体内の支援すべき部分を見定め，支援方法を決定する必要がある．
2. プログラミング開発環境に関する知識の不要化
   現状の言語資源のラッピングでは，プログラミング開発環境の使用を前提としている．そのため，エンドユーザがラッピングを行う際には，この開発環境の操作を学習し，コードなどの編集にも慣れる必要がある．これらのエンドユーザの負荷の軽減が必要となる．
3. 対訳を切り出すルールの容易な記述
   Web ページで提供される資源のラッパーには，ページから対訳を切り出すためのルールを記述する．W4F[2] や WSGen[3] といった既存のラッパー生成支援システムでも，ルール記述に対する支援は行われている．しかしこれらが採用するタグ名に基づくルールでは，切り出しの難しい資源が存在する．このため本研究では，柔軟な記述が可能な，HTML ソースコードに基づく，正規表現を利用したルールを採用する．しかしこのルールの記述には，HTML ソースの参照，正規表現の記述など，エンドユーザに負荷の掛かる作業が存在するため，適切な支援を考える必要がある．

以降，本稿の構成は次のとおりである．まず，第 2 章で現状のラッピング作業，既存のラッピング支援システムについて述べる．次に，第 3 章では，ラッピング作業を分析し，タスクモデルと作業時間から支援すべき部分を決定する．第 4 章では，Web サービス開発環境に関するエンドユーザの負荷を軽減するための支援方法を考える．第 5 章では，用例対訳データの切り出しルールの支援方法を考察する．第 6 章で，開発したシステムの構成とユーザインタフェース

を示す．第 7 章は考察である. 第 8 章でまとめを行い，本稿を締めくくる．

第 2 章　Web サービスラッピング

支援システムを考える準備として，本章では現在のラッピング作業が何を成果物とし，どのような環境で行われているかを述べる．また，既存のラッピング作業支援環境と，その問題点についても触れる．

2.1　言語資源の Web サービスラッピング

本研究では言語資源，その中でも HTML で提供される用例対訳集のラッピング作業を対象とする．

2.1.1　用例対訳集

用例とはある言語で記述された単語や文章である．用例対訳集とは，用例に対し，そのいくつかの言語における翻訳表現が列記されたものを指す．図 1 の例[1)]では，病院や薬局で使用される用語やフレーズについて，日本語と英語での用例対訳が列挙されている．

多くの対訳集においては，このような HTML ページが複数集まって 1 つの Web サイトを構成している．図 2 は，図 1 を含むサイトの構成であり，言語ごと，使用場面ごとに分類された複数の HTML ページを持つ．言語グリッドでは，このような複数のページをまとめて 1 つの用例対訳集と考える．

1熱がある	フィーバー **Fever**
2痛みがある	ペイン **Pain**
3吐き気	ヴァミッティン **Vomiting**

図 1: Web ページとして提供される用例対訳の例

2.1.2　ラッピング作業

用例対訳集を言語グリッド上で利用するには，言語グリッドの標準インフェース ParallelTextService に合わせる必要がある．表 1 は標準インタフェースが用

1) 三菱ウェルファーマ株式会社　病院・薬局で使う外国語会話集 (http://di.m-pharma.co.jp/foreign/index.html)

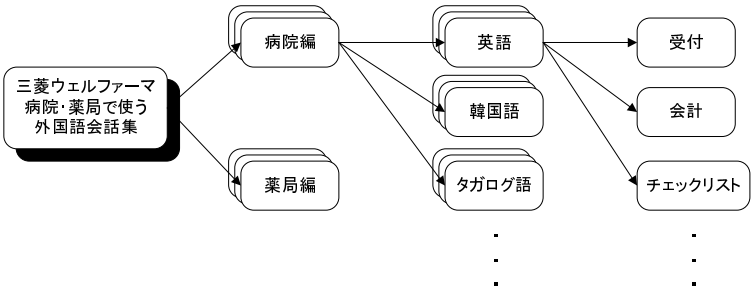

図 2: 複数のページで構成される用例対訳の例

例の検索のために提供するメソッドの仕様である．

表 1: 用例対訳の標準インタフェース

メソッド名：	search
パラメータ：	sourceLang: String – 対訳の元言語
	targetLang: String – 対訳の対象言語
	source: String – 対訳を検索する文字列
	searchMethod: String – 検索方法(“string-exact”, “prefix”, “suffix”, “substring”, “regex”のいずれか).
戻り値：	ParallelText[] – 検索結果の配列

クラス名：	PARALLELTEXT
プロパティ：	source: String – 対訳元文字列
	target: String – source に対応する対訳文字列

用例対訳のラッパーとは，図 3 のように資源ごとに作成され，標準インタフェースを外付けで提供する Web サービスである．ラッパーを作成することで，他の言語サービスは用例対訳集に対し，標準インタフェースによる検索要求を行うことが出来るようになる．上記のようなラッパープログラムを作成する工程が，ラッピング作業である．標準インタフェースに従う限り，検索処理などの実装方法は作業者に任せられている．ただし現在，標準インタフェースは Java 言語のコードで与えられているため，多くのラッパーはこのコードを利用して Java 言語により実装されている．またこれまでに，過去に蓄積された用例対訳集ラッパーの実装から，共通でして現れる処理のライブラリ化を行った．加えて，このライブラリを用いたラッパーの作成例をテンプレートとして作成した (付録 A.1.1)．このテンプレートを用いれば，クラス名などを書き換えることで，ほぼコーディングを行うこと無くラッパーが作成できる．

一方で，対訳データの切り出し対象となる HTML ページや，ページからの用例対訳の切り出しルールなどはラッピング対象により異なる．ライブラリを用

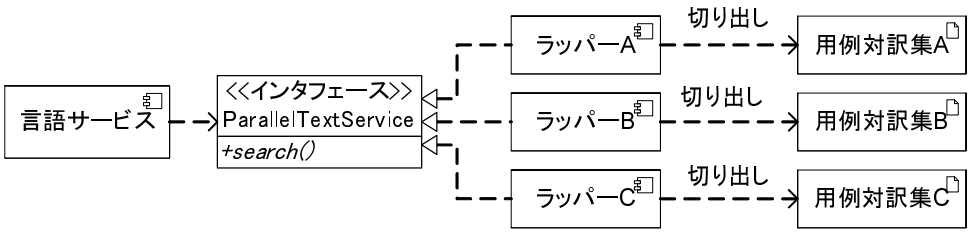

図 3: 言語グリッドの標準インタフェースとラッパー

いる場合，このような処理は，表 2 のような設定ファイルとしてコード外に記述される．この中で切り出しルールは，用例対訳ページの HTML ソースコードに基づき，切り出したい部分の周辺のソースを指定することで記述する．ルール記述には正規表現を含むことができる. 例えば図 1 のページの言語情報は現在,

```
[0-9]*(\\:ja:.+?)</b>.+?<b>(\\:en:.+?)</b>
```

といったルールで記述されている.

ライブラリを用いたラッパーの構成を図 4 に示す.

表 2: 設定ファイルの主な構造

type	:資源のタイプを記述．ここでは"ParallelText_HTML"
metaDataFile	:作成者など，資源のメタデータを記述.
default?	:資源の情報を記述．resource で項目の欠如が有った場合，これが参照される.
langPath*	:対応言語
content?	:HTML ページ群の内容の記述
charSet?	:ページの文字コード
address*	:各ページの URL
scraper?	:切り出しルールの記述
resources*	:資源の情報を記述．子要素は”default”と同様.

(末尾の記号は要素の出現数　?は 0 または 1 つ，*は 0 以上の任意の数，無印は 1 つ)

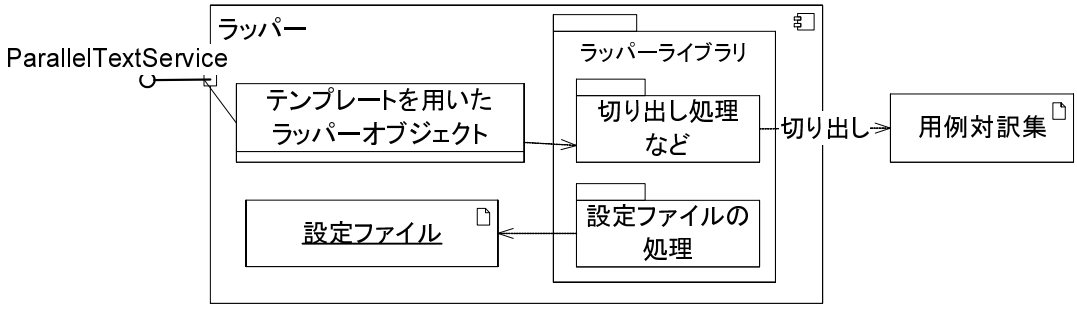

図 4: HTML 用例対訳ラッパーのアーキテクチャ

ライブラリとテンプレートを用いた場合，ラッピング作業は必要ファイルの

作成，設定ファイルの記述を中心として行われる．その後，各コードをビルドし，Web サービス化する．この他に，作成されたラッパーが標準インタフェースに準拠しているかを検証するテストの記述と実行も求められている．このライブラリを用いた用例対訳集のラッピング作業が，現在マニュアルの形で標準化されている．本稿ではこの標準的な手順に基づき支援システムを開発する．

2.2 既存のラッピング支援手法

Web ページで提供される用例対訳ラッパーの主な機能は，Web ページからルールに従い情報を切り出し，整理するというものである．現在，このような機能をユーザの必要に応じて作成するための手法は既に提案されている．これらは，それぞれにルールの表現，ルールの作成法，支援の度合いが異なっている．

1 つのアプローチとして，WIEN[4] や WHISK[5]，STALKER[6] など，HTML ソースコードに基づく切り出しルールを用い，結果を特定形式の文字列として出力するものが提案されている．ただし第 5 章で詳述するように，このようなルールの記述はユーザへの負担が大きい．このため，上記の既存手法ではルールはユーザが記述せず，[4] の提案するラッパー帰納と呼ばれる手法を用いている．ラッパー帰納は帰納学習に基づく手法である.1 つのページと，そのページに対する切り出し部分をユーザが訓練例として与えることで，切り出しに用いるルールが自動的に推測される．

他には，W4F[2] や WSGen[3] など，ルール記述から Web サービスの作成までを総合的に支援する環境が開発されている．W4F は，切り出した情報から必要な形式で Java オブジェクトを生成できる.WSGen は同様に，指定されたインタフェースを備えた Web サービスの生成を総合的に支援する．これらの支援環境では共通して，切り出しルールを DOM(Document Object Model) に基づく HTML タグの階層構造により記述する．例えば W4F で，図 1 から対訳を切り出すルールは図 5 のように記述される．対訳がテーブルで記述されている場合など，タグの階層構造で切り出し部分が記述できるならば，HTML ソースコードに基づくルールよりも簡潔な記述ができる．このためこれらの手法では，ルール判断の支援はするものの，採用するルールの決定および記述はユーザが行う．

これらの支援環境を言語資源のラッピングに適用する場合，切り出しルール作成における問題，および支援の網羅性の問題が存在する．

- 切り出しルール作成における問題

```
parallelTexts = html.body.table[1].tr[*](
(
        .td[0].font[0].b[0].txt   ,match /[０-９]+(.+?)/   //Japanese
        .td[1].b[0].txt           ,match /(.+?)/           //English
)
```

図 5: W4F における図 1 のページからの切り出しルール

上記の 2 通りのルール記述には，それぞれに問題が存在する．

HTML ソースに基づくルール記法の問題点は，作成時の負荷である．ラッパー帰納を用いれば記述作業は省略できる，しかし依然として 1 つのページに対し，切り出し部分を訓練例として与える必要がある．ところが用例対訳集では，1 つのページあたり数十〜数百の対訳が記述されている場合が多い．このため，訓練例を与える作業がユーザにとって大きな負荷となる．

一方，DOM に基づく記法は 2 つの問題点を持つ．まず，タグ内からの切り出しには，依然として/[０-９]+(.+?)/のような正規表現の記述が必要となる点である．この部分の記述は支援されていない．

次に，タグの属性を切り出しルールに使用できない場合がある．例えば図 6 のソースを持つ資源から，“英語”-“日本語” を切り出さず，“AA”[1)]-“学力年齢” などを切り出すようなルールの記述が困難である．

```
<TR>
    <TD align="center" width="180">英　語</TD>
    <TD align="center" width="180">日本語</TD>
    <TD align="center">意　味</TD>
</TR>
<TR>
    <TD align="left">AA</TD>
    <TD align="left">学力年齢</TD>
    <TD align="left">標準学力検査における学力水準年齢。</TD>
</TR>
<TR>
```

図 6: DOM に基づくルールで切り出しが困難な例

- 支援の網羅性の問題

  既存の手法で支援されるのは，切り出しルールの作成のみ，またはそこからの Web サービスの作成までである．ところが言語グリッドのラッピングでは，資源について説明するためのメタデータの記述，インタフェース仕

[1)] achievement age の略

様を満たしていることを検証するテストの記述など，特有の作業を行う必要がある．また対訳の切り出し結果に関しても，“完全一致” や “正規表現” といった検索条件によって，異なる処理を行わなければならない．これらの要求は，既存手法の単純な適用では満たされない．

第3章　ラッピング作業の分析

本章では，支援対象となるラッピングシステムを詳細に分析し，具体的な支援方法を決定する．現在ラッピング作業は，数人によるラッピング経験に基づいて標準的手順がマニュアル化されている．そこで，まず筆者がこの標準的手順に従い実際に用例対訳資源[1)]のラッピング作業を行い，過程を記録した．

この作業のラッピング対象は，様々な場面での日本語-トンガ語の会話が，各ページ 20〜50 対訳程度，７ページからなる用例対訳集である．作業はマニュアルを参照しながら行い，延べ 2 時間程度の作業時間で行った．

ただし，ラッピング環境の準備作業は，40 分程度の作業を，上とは別に記録した．また，実際には数時間〜数日程度かかるであろう，ユーザ登録待ちの時間は準備作業の所要時間外とした．

まず 3.1 節では，作業記録に基づき標準的手順を下位のタスクに分解し，詳細なタスクモデルを得る．それぞれのタスクをエンドユーザの負荷の面から考察し，支援を必要とする問題状況を特定する．

続く 3.2 節では，記録から得た時系列データを基に，時間的コストの大きい作業を特定する．ただし筆者の作業に基づいているため，得られた記録はエンドユーザのものでなく，作業に慣れた者の参考値となる．

3.1　タスクモデルの作成

以下の分析では，マニュアルの記述を作業のモデルとし，細部はこの作業記録によって得られた操作系列を参考とした．

ラッピング作業は図 7 のように，準備作業，およびラッパーの作成と検討を目的とする 4 つの下位作業に分けられる．以下の各項では，それぞれの下位作業に注目し考察していく．

[1)] 楽しいトンガ語 http://www.daito.ac.jp/~nakamoto/tongago.html

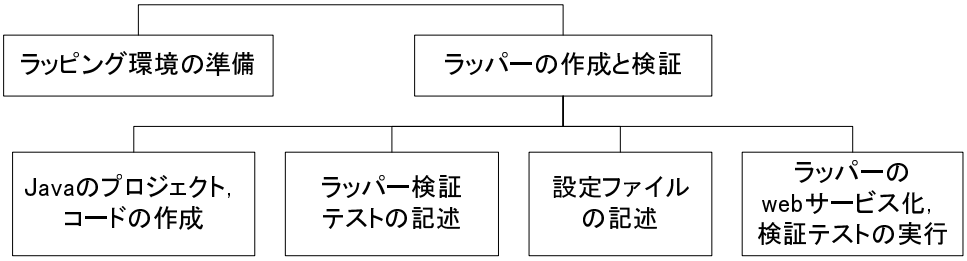

図 7: ラッピングの下位作業

3.1.1 ラッピング環境の準備

ラッピング環境の準備作業は，図 8 のような 3 つの下位作業からなる．

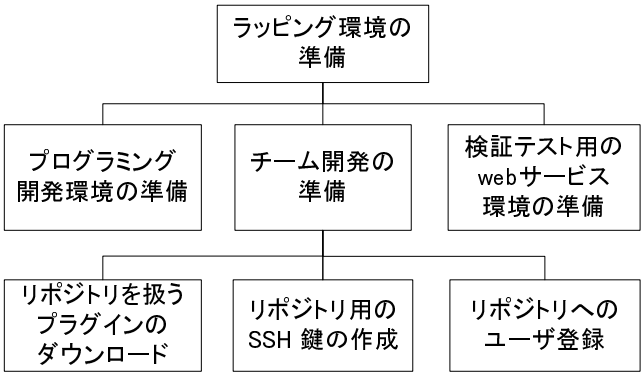

図 8: ラッピング環境の準備作業の下位作業

Web サービス開発環境の準備，検証テスト用の Web サービス環境の準備作業には，ソフトウェアのダウンロードおよびインストール作業が含まれる．多くのソフトウェアのインストール作業はエンドユーザにとって負担となり得る．

また，チーム開発の準備のためにも多くの手順が必要となる．リポジトリを扱う環境をインストールし，また専用のソフトウェアにより暗号鍵を生成，それを言語グリッドのリポジトリ管理者に送信し，ユーザ登録を待つ必要がある．このため，作業が開始できるまでに登録待ちの期間が生じる．

上記の問題状況を整理すると以下のようになる．

- ラッピング作業の開始に，多くのソフトウェアの導入が必要
- チーム開発のために，ユーザ登録を待つ期間が発生

3.1.2 Javaのプロジェクト，コードの作成

Javaのプロジェクト，コードの作成作業は，図9のような3つの下位作業からなる．

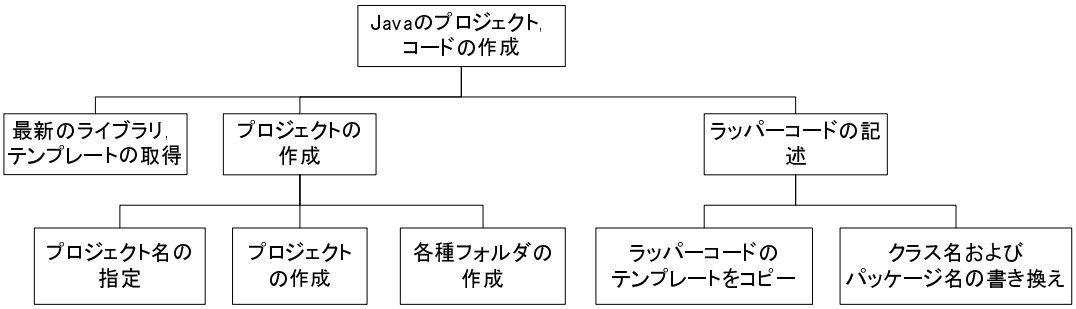

図9: Javaのプロジェクト，コードの作成の下位作業

最新のライブラリ，テンプレートの取得は，作成するラッパーに最新の仕様を反映させるために必要な作業である．

続くプロジェクトの作成で必要となるのは，単純なマウス操作に加え，プロジェクト名，フォルダ名の入力といった単調なキー入力である．最後のラッパーコードの記述でも，必要となるのはコピー&ペーストなどを含む操作である．これは，個々の作業の難易度は高くないが，単調な作業の連続となるため，利用者にとっての負荷は大きいと考えられる．Javaのプロジェクト，コードの作成作業における問題状況は以下である．

- 人間が行わなくても良い単調な作業が多い

3.1.3 ラッパーの検証テストの記述

ラッパーの検証テストの記述作業は，図10のような2つの下位作業からなる．コードの作成は，テストコードのテンプレートのコピーと，クラス名の書き換えという単調作業である．

テストの記述では，まずある文字列をサンプルとして選ぶ．次にラッパーの対応する全ての言語，検索方法でその文字列の検索を行うコードを書く．2.1節で示したインタフェースに基づいた適切な結果が返らない場合には，エラーを通知するよう記述する．

つまりテストの記述のために，作業者が正しい検索結果を把握する必要がある．このため，言語資源を構成する全てのページをブラウジングし，手動で検索を行い，どのような内容の対訳が，いくつ存在するかを記録する必要がある．

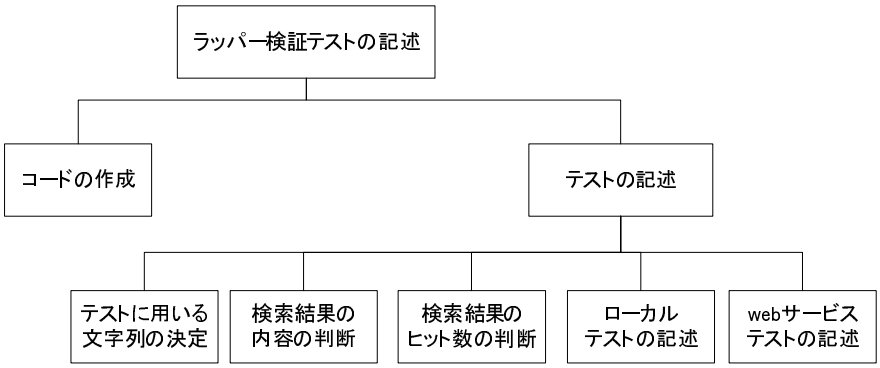

図 10: 設定ファイルの記述を構成する作業

その後に，テスト項目と正しい結果をコードとして記述する．この作業は，テンプレート内の文字列や対応言語，数値を書き換えることで行える．ただし，言語グリッドでは言語を言語名でなく，ISO639 で規定される言語コードで記述している．そのため例えば，対応言語で “日本語-英語” は “ja-en” と記述する，といった専門の知識が必要となる．

ラッパー検証テストの記述作業における問題状況は以下である

- テスト内容の決定のため，多くのブラウジングと記録が必要
- 言語コードの専門知識が必要となる

3.1.4 設定ファイルの記述

設定ファイルの記述作業は，図 11 のような 4 つの下位作業からなる．

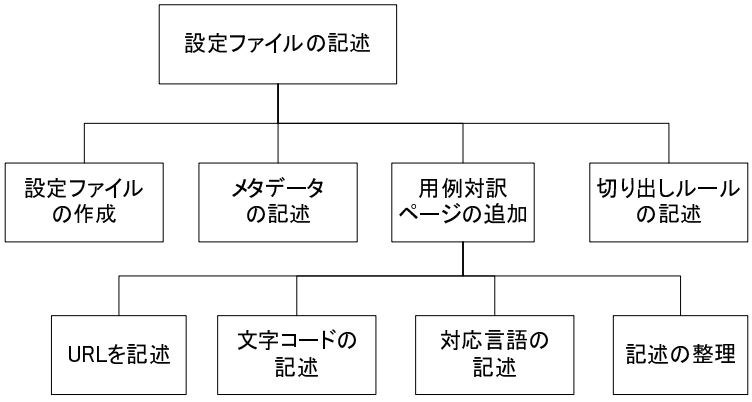

図 11: 設定ファイルの記述を構成する作業

設定ファイルの作成は，テストコードのテンプレートのコピーと，クラス名の書き換えという単調作業である．メタデータの記述は，言語資源のブラウジ

ングにより資源の作成者（○○語協会など）などを判断し，テンプレートを書き換えることで行われる．

用例対訳ページの追加では，まず追加するURLを設定ファイルに記述し，ソースなどを参考に文字コードを記述する．次に，ページ内の対訳が対応している言語の言語コードを記述する．また設定ファイルでは，切り出しルールや対応言語が同じページをまとめて簡潔に記述できる．よって，ページ追加に際し設定ファイルの記述を整形した方が良い場合などがある．この整形作業には，設定ファイルの書式に慣れねばならず，エンドユーザには負担となる．

ラッパーが言語資源からデータを切り出すためのルールも，設定ファイルの中に記述する．この作業については，特に第5章で考察する．

設定ファイルの記述作業における問題状況は以下である

- 言語コードの専門知識が必要となる
- 設定ファイルの記法の専門知識が必要となる

3.1.5　ラッパーのWebサービス化，検証テストの実行

ラッパーのWebサービス化，検証テストの実行作業は，図12のような４つの下位作業からなる．

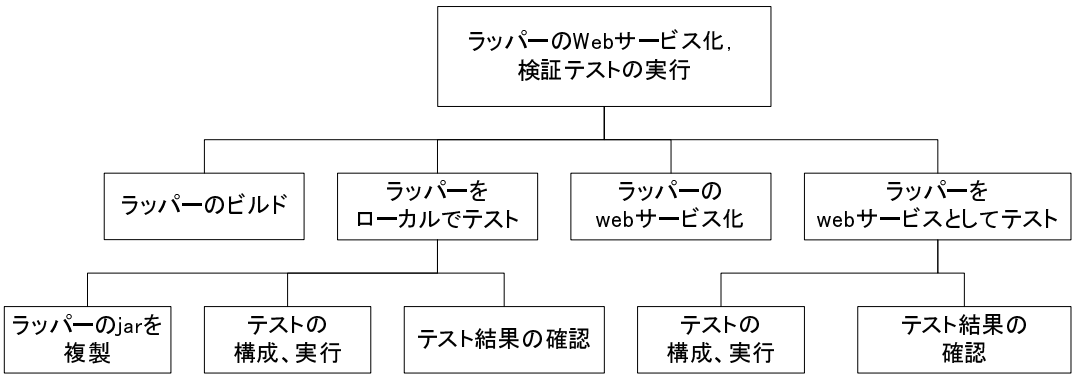

図12: ラッパーのWebサービス化，検証テストの実行を構成する作業

ラッパーのビルド作業は，ビルド用のスクリプトをコピーし，ラッパー名を書き換えて実行するという単調作業である．

こうして作成したラッパーをテストするには，まずテスト用のプロジェクトに，ビルドされたラッパーを複製する．次にテストを構成し，実行する．これらの工程は，クラス名などの書き換え以外は単調なマウス操作である．

テストごとの検証結果は，プログラミング環境の専用のウィンドウなどに表示され，成否の判断は容易である．

ラッパーの Web サービス化，テストの構成，実行，結果判断も，同様の単調作業である．

- 人間が行わなくても良い単調な作業が多い

3.2 作業時間の分析

本節では，ラッピング作業の効率化のため，どの作業に時間的コストが掛かっているかを特定する．

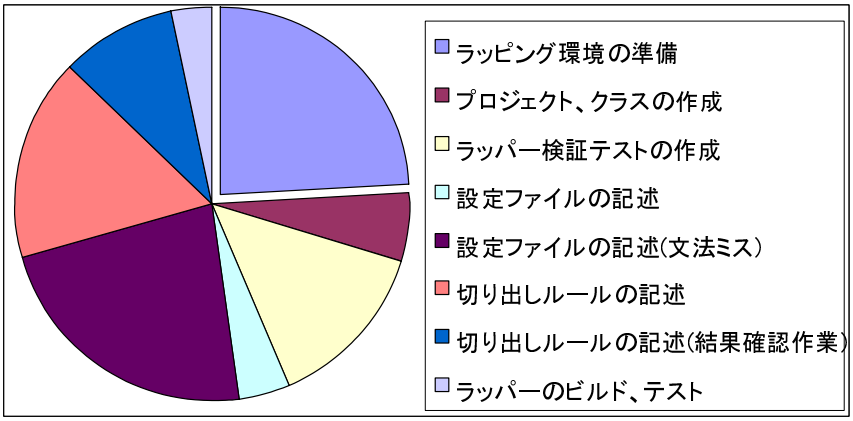

図 13: 各下位作業の所要時間比率

ここでは，作業記録から得た時系列データを用いた．ただしサンプルである筆者はラッピング作業を多く経験しているので，以下はエンドユーザのラッピングでなく，慣れた者のラッピングでの問題状況の考察である．各下位作業の所要時間の比率を図示したのが図 13 である．

設定ファイルの記述 (文法ミス) の項は，設定ファイル内の単語や記述位置の間違いによりラッパーが正常に動かず，その原因の特定までに費やした時間である．切り出しルールの記述，その結果の確認作業も，設定ファイルの記述から分けて示してある．

この図から分かる，ラッピング作業で特に時間を要する作業は以下である

- ラッピング環境の準備作業
- ラッパーの検証テストの作成作業

- 設定ファイル記述作業時の文法ミスの発見
- 切り出しルールの作成，その結果確認作業

第4章　サーバ型の開発環境

現在のラッピング作業は，作業者のマシンに Web サービス開発環境を導入し，その上で行われている．本章では開発環境の使用がエンドユーザにもたらす負荷を考察し，それらの負担を軽減すべく支援システムのアーキテクチャを決定する．

4.1　開発環境の操作負荷

第3章で分析したラッピング作業は，3.1.1 節の作業で準備する Web サービス開発環境で行っている．開発環境などの導入時の負荷は第 3 章で考察した．このような開発環境は，様々なファイルの管理に始まり，プログラムのコーディングやデバッグなど，プログラマの作業を総合的に支援すべく設計されている．このせいで，エンドユーザが作業を行う場合，作業中にも以下のような負荷が想定される．

第一に，開発環境の多機能なユーザインタフェースによる負荷である．開発環境ではラッピング作業に不必要な機能も多く提供されており，画面上から呼び出せるようになっている．このため，ラッピング作業中に多くの操作の中から実行すべき操作を呼び出せるようになるためには，ある程度の学習を要する．また，学習後も誤って不必要な処理を実行してしまうといった操作ミスが発生しやすい．

第二に，見慣れないファイルを直接扱う事による負荷である．ラッピング作業には，ラッパーおよび検証テストのコード，設定ファイルといったファイルを直接書き換える作業が多く含まれる．これらの作業は単調であるが，数十行に及び関数名や要素名が羅列されているテキストファイルから，書き換えるべき場所を発見し，書き換えるのはエンドユーザにとって負荷となり得る．また 3.2 節の所要時間の図で見たように，慣れた者でさえファイル記述作業でミスを犯す場合もある．エンドユーザの作業では更にミスが発生しやすいと考えられる．

最後に，ライブラリの更新に伴う負荷について述べる．ラッパーから呼び出されるライブラリや，ビルドに必要なスクリプトなどは開発環境と同じマシン

に存在する必要がある．現在，開発環境が作業者のマシンに存在するため，随時最新のライブラリなどをわざわざ言語グリッドの共有リポジトリから取得しなければならない．3.1.2 節では，プロジェクト作成時に更新作業が必要なことを書いているが，実際にはこの作業はライブラリの仕様が変更されるたびに必要となる．これは現在，ラッピング用のライブラリの仕様が確定しておらず，ライブラリの更新に伴いラッパーの修正が必要となる場合が有るためである．

作業者のマシンで Web サービス開発環境を使用することに伴う問題状況を整理すると以下のようになる

- 多機能なインタフェースから，必要な操作を選択しなければならない
- エンドユーザの馴染みの無いファイルを直接操作しなければならない
- 随時ライブラリなどの更新が必要となる

続く 4.2 節では，これらの問題状況を支援すべくシステムのアーキテクチャを決定する．

4.2 システムアーキテクチャ

4.1 節で示した，Web サービス開発環境の使用による負荷を軽減するためには，以下の要求を満たすように支援システムを設計すればよい．

- 必要な操作に必要十分なユーザインタフェースを提供する
- 使用者がファイルを直接操作する必要が無い
- ライブラリなどを使用者が更新する必要が無い

これらを満たすため，支援システムにサーバクライアントモデルを採用し，現在利用者のマシンに置かれている Web サービス開発環境をサーバに置くことを考えた．このようにすれば，ライブラリなどの更新もこのサーバの環境に対して行うだけでよい．加えて，サーバの開発環境を利用したラッパー作成を行うクライアントアプリケーションを考える．このアプリケーションに，エンドユーザに負荷の少ないユーザインタフェースを持たせることで，上記の要求を満たすことを考える．

またサーバ側にも，クライアントから Web サービス開発環境を呼び出すための窓口でとなるアプリケーションを用意する．サーバアプリケーションは，他にも，ラッピングの作業ごとに様々な支援機能も提供する．

ここまでに述べた事をまとめ，支援システムの大まかなアーキテクチャを図 14 に示す．詳細な実装は 6.1 節で述べる．

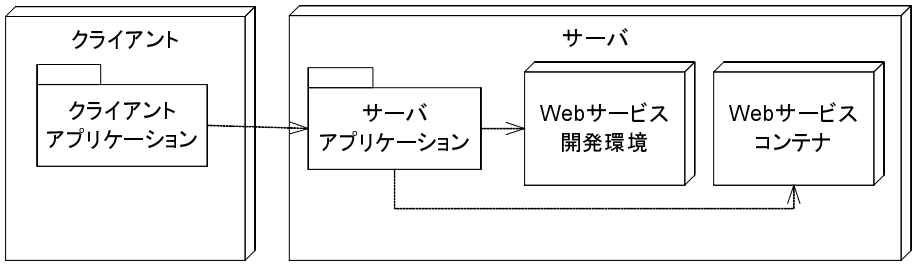

図 14: サーバクライアント型支援システムのアーキテクチャ

第 5 章　データの切り出しルール作成の支援

5.1　切り出しルールの作成負荷

図 15 は，言語資源の切り出し作業のモデルである．これらの作業は，対訳集を構成する各 Web ページを設定ファイルに追加するたびに繰り返される．

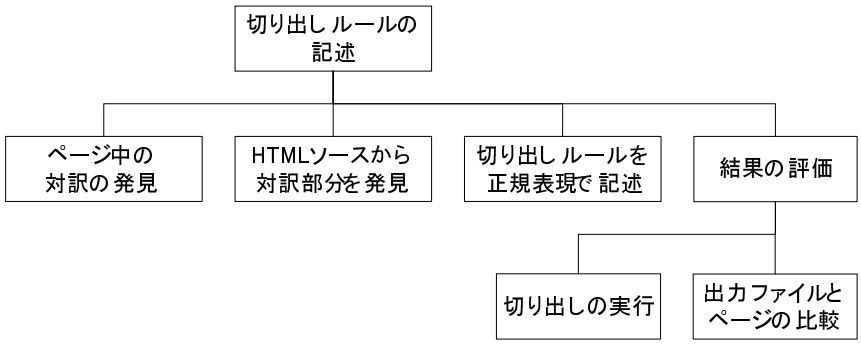

図 15: 切り出しルール作成を構成する作業

まずユーザは，追加したページを眺めて対訳を発見する．

続いて HTML ソースに基づく切り出しルールを記述する．このためには，まず対訳の周りの HTML ソースを得る必要がある．この作業は，ブラウザから “ソースの表示” などを選択してソースをテキストエディタで開き，発見した対訳を手作業で検索することで行われる．HTML ソースをテキストエディタで扱うという作業は，エンドユーザにとって不慣れなものと考えられる．

続いて同様に，他のいくつかの対訳周辺の HTML ソースを得る．そのように得た複数の HTML ソースが似通っていた場合，似通った部分を正規表現の記述によって一般化し，似た構造の対訳を一括して切り出すルールを記述する．例えば，図 16 の左のソースから日本語部分を切り出すためには，1，2，3，といっ

た具体的な数字を，連続する数字を表す正規表現に置き換えて一般化し，同図右の切り出しルールを作成する．

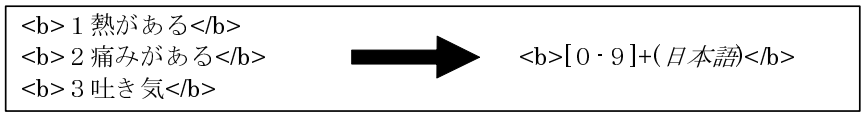

図 16: 切り出しルールの一般化例

しかし，エンドユーザにとって正規表現の知識は一般的でなく，記述に際しての学習が負担になると考えられる．また慣れたユーザにとっても，3.2 節で見たようにこの作業は時間を要する．このため，正規表現の記述作業量を軽減すべく支援が望まれる．

切り出し方法を記述した後は，結果を確認するために切り出しを試行する．この際，切り出し結果は図 17 のような “：” 区切りの文字列の羅列として出力される．これをブラウザでの表示と見比べながら，切り出しの過不足を判断する．

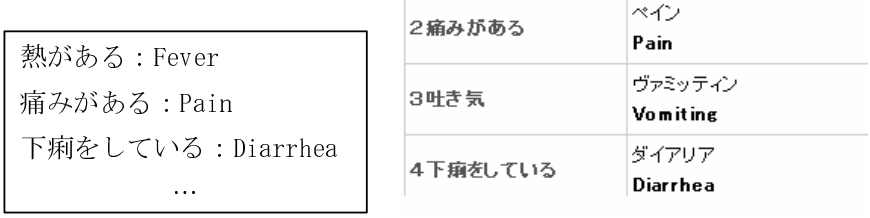

図 17: 出力ファイルとブラウザでの表示の比較

整理すると，切り出しルールの記述における問題状況は以下である．

- テキストエディタでの HTML の検索という，ユーザに不慣れな作業
- 正規表現の記述という，ユーザが不慣れで，かつ時間を要する作業
- 切り出し結果の確認で，直感的でない比較を必要とする作業

次の 5.2 節では，これらの問題状況への支援方針をシステム要求として定める．

5.2　切り出しルールの生成手法

第 5 章節で示した問題状況に対応し，支援システムのシステム要求を記述すると以下のようになる．

- HTML ソースを，ユーザの慣れたインタフェースで参照できる
- 切り出しルールを，エンドユーザが慣れたインタフェースで記述できる
- 切り出し結果の確認を視覚的に行えるインタフェースを備える

以下，これらをどのように実現するかを考える．

まず一点目に対しては，マウス操作での指示からのソース取得を考える．ユーザは図 18 のように，ブラウザ上での範囲選択により，対応する HTML ソースを得られる．このようなユーザインタフェースを用いれば，ユーザはマウスのドラッグという慣れた操作のみでこの作業を行える．

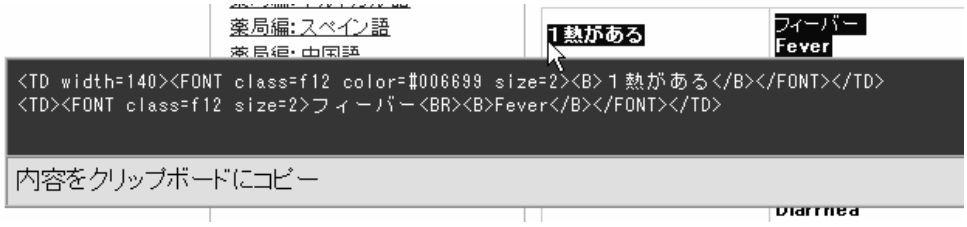

図 18: HTML ソースの参照作業の支援

次に二点目に対しては，図 19 のようにマウス操作のみでテキストを操作できるインタフェースを考える．

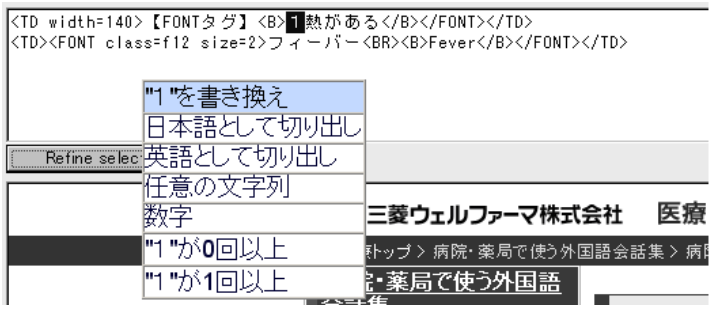

図 19: 切り出しルールの記述支援

このインタフェースでは，切り出しルールに用いる正規表現の多くを占める，ワイルドカードや繰り返しの記述を，簡単なマウス操作で可能にする．例えば選択範囲のテキストを，その種類に対応して “数値” “任意の文字列” といった

ワイルドカードで置き換えることにより，切り出しルールの一般化が可能となる．置き換えに用いるワイルドカードは図 20 のような階層で与える．例えば <a href=”top.htm”>を選択した状態でワイルドカードへの置き換え操作を行うと，置き換え候補として “A タグ”，“タグ”，“任意の文字列” の 3 種類のワイルドカードが提示される．また，文字列の繰り返しを表す正規表現も多く用いられた. これを簡単に記述するため，“○○が 0 回以上” といった記述の挿入も可能とする. 加えて，“日本語として切り出す” のような切り出し部分の記述も同様のマウス操作で行えるようにする．これにより，エンドユーザの慣れたマウス操作により切り出しルールを得られるようにし，既存手法では対応できない言語資源の切り出しルールも容易に記述できるようにした.

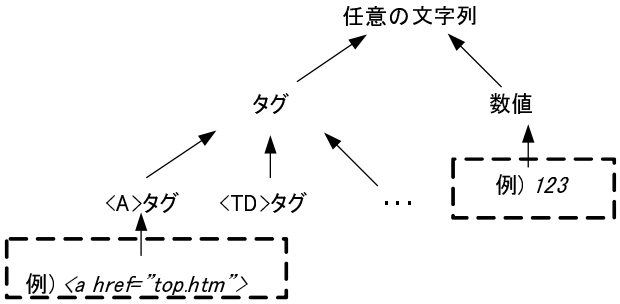

図 20: ルールの一般化に用いる階層の全体

最後の三点目に対しては，切り出し結果を図 21 のようにブラウザ上でハイライト表示し，過不足の確認を視覚的行えるようにする．図 21 では，フォーマットが統一されていないせいで，“LUX syndrome-軟骨毛髪低形成症候群” という対訳が切り出されていないことが一目で確認できる. また，“(呼)” のような，切り出すべきでないジャンル名などの部分が切り出されていることが確認できる. これらはルールを修正する際の手助けとなる.

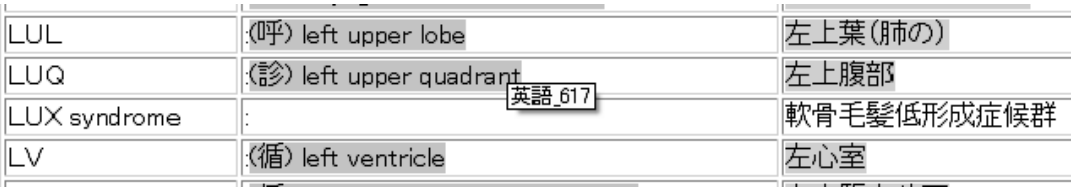

図 21: 切り出し結果の確認

第6章　ラッピング支援システム

この章では，第4章および第5章での設計を基に実装した支援システムの構成について述べる．

6.1　システム全体の構成

第4章で示したクライアントアプリケーション，サーバアプリケーションの全体的な構成を示したのが図22である．

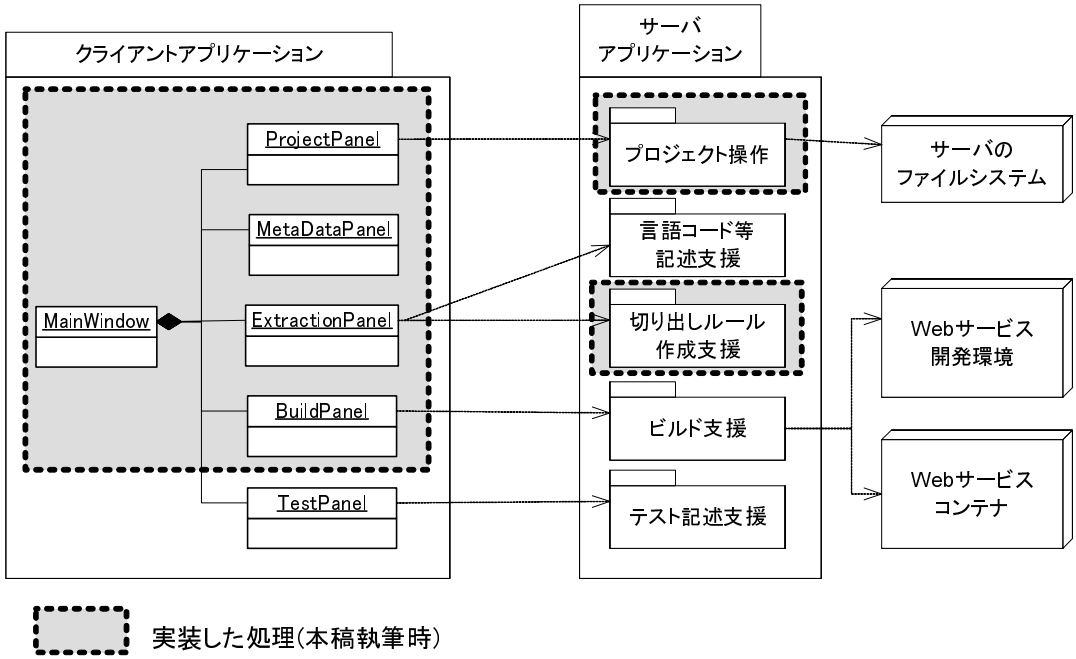

図22: 支援システム全体の構成

クライアントアプリケーションは，Webブラウザ上での動作を想定しJavaScriptでの実装を考えた．これにより，ラッピング作業に必要なソフトウェアがWebブラウザのみとなり，作業を始める際のソフトウェアのインストールが不要となり，準備作業でのエンドユーザの負荷が抑えられる．また，支援システム自体に更新が有った場合にも，ユーザはシステムのアップデートなどを行うこと無く対応できる．

図23に，支援システムのユーザインタフェースを示す．ユーザインタフェースは，大枠となるウィンドウ(MainWindow)と，その上に配置された複数の操

作画面 (〜Panel) で構成される．以下，図 23 の構成を説明する．

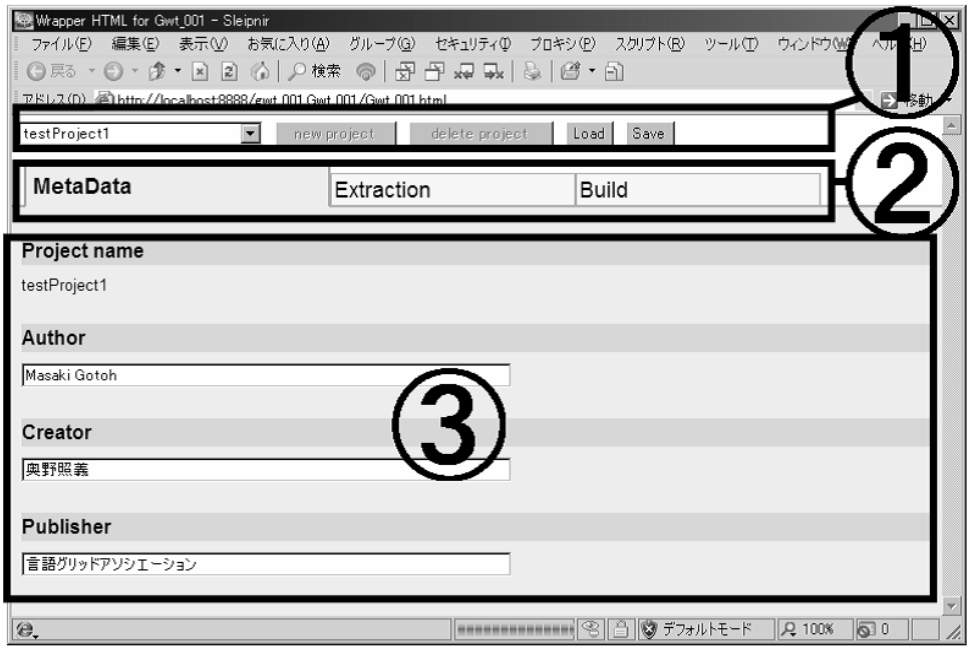

図 23: 支援システムのユーザインタフェース

1. プロジェクトを扱う ProjectPanel
   サーバに存在するラッパープロジェクトのリスト，新規プロジェクトの作成ボタン，プロジェクトの消去ボタン，および作成中のプロジェクトのセーブ，ロードボタンからなる．
   サーバ側には，これと対応してプロジェクト操作の窓口となるサブシステムが置かれている．このサブシステムは，ProjectPanel からの要求を受けてファイルシステムにアクセスし，プロジェクトに関するフォルダやファイルの作成を行う．これにより，ラッパーを作成する際の，プロジェクト作成などの作業が省略される．
2. 作業の切り替えを行うタブ
   それぞれの操作画面 (〜Panel) はラッピング作業の各下位作業にほぼ対応しており，ユーザはタブ操作で画面を切り替えることで，行う作業を選択できる．
3. 下位作業に対応した作業画面
   この部分に，MetaDataPanel 他，下位作業に対応する操作画面が表示される．

図 23 に表示されているのは，メタデータの記述作業に対応する MetaDataPanel である．この画面では，テキストボックスから各メタデータの記述を行うようになっている．これにより，エンドユーザが操作の選択に迷う事無く，またファイルを直接参照する事無くラッパーコードや設定ファイルの書き換えを行える．この画面同様，全ての〜Panel はラッピング作業中の操作に対応するテキストボックスやボタンなどを主な部品として構成されており，ラッピング作業に不要な操作はユーザインタフェースから取り除かれている．

以下は，図 23 で選択されていない作業画面の働きである．

ExtractionPanel と切り出しルール作成支援については，6.2 節で詳述する．

BuildPanel は，ラッパーをビルドし，Web アプリケーション化する一連の処理を呼び出す画面である．この画面上のボタンによる指示を受け，サーバ側に置かれたビルド支援モジュールが，開発環境上でビルド，Web サービス化スクリプトを実行する．作成されたラッパーはサーバの Web サービスコンテナに配備される．これにより，ビルド，Web サービス化作業における単調作業が省略できる．

TestPanel はテストの記述を行なうインタフェースであるが，現段階では未設計である．

6.2 データの切り出し支援部の実装

ExtractionPanel と，対応する切り出しルール支援モジュールの構成を図 24 に示す．

ExtractionPanel は，第 5 章で述べた HTML の参照，ルールの記述，切り出し結果の確認，の 3 種の支援に対応するユーザインタフェースで構成される．サーバ側でも，これら支援を行うサーブレットが用意され，クライアントからの Post でのリクエストに応じてレスポンスを返す．以下，これらのユーザインタフェースとサーブレットにより，どのように支援が実現されるかを述べる．

HTML の参照は以下のように支援される．利用者が HTML 取得ボタンを押すことで，アドレス欄の URL が HTMLCodeSupport に送信される．HTMLCodeSupport はこれを受けて対応する Web ページを取得し，図 18 のような支援を提供する JavaScript コードを埋め込み，クライアントに返す．利用者は HTMLFrame に表示された HTML に対する範囲選択により，対訳に対応する

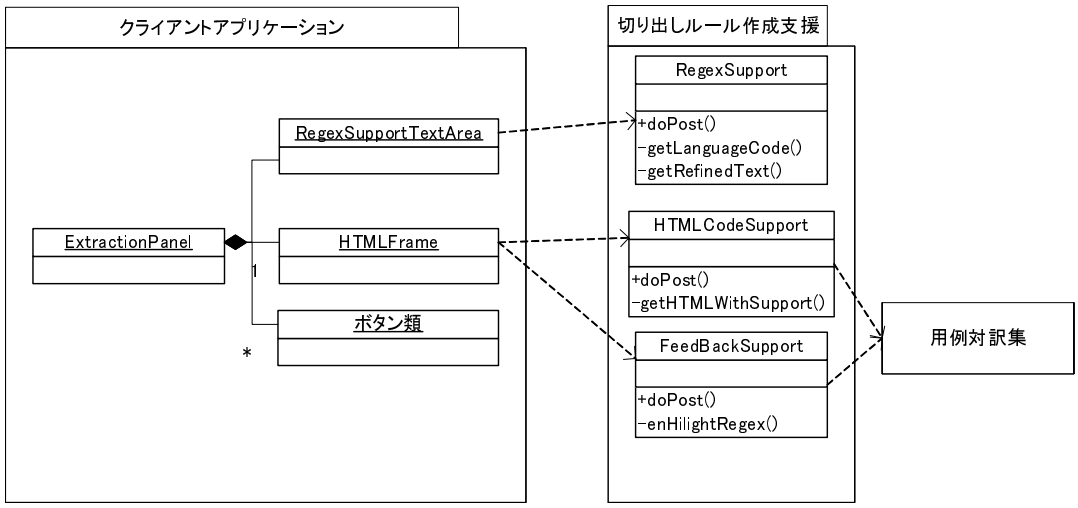

図 24: 切り出しルール作成支援部の構成

HTML ソースを得られる．

ただし本稿執筆時の実装では，この機能は InternetExplorer(以下 IE)6 以降にのみ対応している．また，範囲選択により得られる HTML は IE の HTML パーサによりタグの大文字小文字，タグ内の属性などが整形されたものである．よって，範囲選択で得られる HTML と実際のコードを一致させるためには，事前に対象となる HTML を IE の HTML パーサにより整形しておく必要がある．このため，第 7 章で行う実験は，実際には整形後の HTML に対して行っている．

切り出しルールの記述は以下のように支援される．利用者は RegexSupport-TextArea のテキストから，対訳として切り出したい，または一般化したい部分を選択する．その状態でルール洗練ボタンを押すことにより，選択部分の文字列，および既に記述されているそのページの対応言語が RegexSupport に送信される．RegexSupport は，受け取った対応言語から”～語として切り出し”といった選択肢を返す．また，選択文字列が図 20 内のワイルドカードにで表せる場合には，そのワイルドカードを一般化候補として返す．これをクライアント側で表示することで，図 19 の操作を実現する．

最後の，切り出し結果の確認は以下のように支援される．利用者が結果の確認ボタンを押すことで，アドレス欄の URL および RegexSupportTextArea 内の切り出しルールが FeedBackSupport に送信される．FeedBackSupport はこれを受けて対応する Web ページを取得し，切り出ルールに基づき対訳部分を特定する．そして対訳部分に図 25 のような，ハイライト表示，ラベルの表示を行う HTML

コードを埋め込み，クライアントに返す．返されたページはHTMLFrameに図17のように表示される．利用者はこれを見て，切り出し結果の過不足を判断できる．

```
<font title="ラベル名" style="background-color:ハイライト色 ">
ハイライト対象文字列
</font>
```

図25: 切り出し結果のハイライト表示，ラベルの表示を行うHTMLコード

第7章　考察

切り出しルールの作成支援部を実装し，複数の用例対訳集に対しルール作成実験を行うことで，提案手法の有効性を考察した．

ただし，この実験で用いる資源は，6.2で述べた実装上の都合により，InternetExplorerのHTMLパーサによりタグの大文字小文字，タグ内の属性の順序などを事前に統一して用いている．

2.2節で，タグの属性を用いない既存手法による切り出しが困難とした例[1)]に対して，提案手法では以下の手順でルールが得られた．

1. 切り出したい部分を選択し，以下のHTMLを得る

```
<TD align=left>ability grouping</TD>
<TD align=left>能力別学級編成</TD>
```

2. 得たHTMLを修正し，以下のように切り出したい部分を指定する

```
<TD align=left>【英語として切り出し】</TD>
<TD align=left>【日本語として切り出し】</TD>
```

3. ルールが一般的すぎ，期待した結果が得られていないと判断する．
4. より広い範囲を選択し，以下のHTMLを得る

[1)] LD用語集 (英和) (http://www.amy.hi-ho.ne.jp/yamaokash/ldword01.htm)

```
<TD align=left>ability grouping</TD>
<TD align=left>能力別学級編成</TD>
<TD align=left>一般的能力や学業成績により、等質をグルーピングしたクラス編成。</TD>
```

5. 切り出す部分を指定し，関係の無い部分をワイルドカードに修正する．

```
<TD align=left>【英語として切り出し】</TD>
<TD align=left>【日本語として切り出し】</TD>
<TD align=left>【任意の文字列】</TD>
```

6. 切り出し結果の確認

   切り出すべき部分が正確に切り出されていると判断する．

以上のように，この例ではHTMLの取得，ルールの書き換え，結果の判断のプロセスを2度繰り返すことで，適切なルールを得られた．

この例を含め12例の用例対訳集について同様にルール取得を試みた．

既存手法での切り出しが困難であり，提案手法で切り出しが行えたものは2例であった．2例とも，タグの属性情報を利用する必要がある資源であった．

既存手法での切り出しが容易であり，提案手法が適用できなかったのは3例であった．この原因は，資源がJavaScriptを用いており埋め込んだコードが動かない，という実装上の問題の他，提案手法で簡潔に記述できるルールが，正規表現と比べて表現力に劣るという点が大きかった．

どちらの手法でも切り出しが行えたものが2例であり，両手法で切り出し困難なものが5例であった．5例において，既存手法で生じた困難は，主にタグ内の文字列切り出しのための正規表現の記述であった．対して，提案手法で生じた困難の多くは，提案手法で記述できるルールの表現力不足によるものであった．

以上より，タグ内の属性情報を利用する必要がある場合には提案手法が，その他の場合には既存手法が優位であると推測できた．

第8章　おわりに

本稿では，言語資源のラッピングにおいて，効率的でない作業，エンドユーザの負荷となる作業が存在し，言語サービス充実の妨げとなっている事を問題

とし，ラッピング作業を支援するシステムを設計した．

開発に際しては，言語資源のラッピング作業の負荷箇所の特定，プログラミング開発環境に関する知識の不要化，対訳を切り出すルールの容易な記述という３点の課題に取り組んだ．

具体的には，まず蓄積されたラッピング経験から得られた標準的な作業手順を分析し，タスクモデルとして整理した．次にラッピング作業をビデオ記録し，下位作業ごとの所要時間を得た．このモデルと所要時間から，支援すべき部分を特定した．

次に，支援システムの大まかなアーキテクチャを決定した．サーバクライアントモデルを採用し，コーディング処理やWebサービス化処理をサーバで行うことで，クライアント側ではプログラミング開発環境が不要となり，エンドユーザに負荷の少ない設計が可能となった．

切り出しルールの記述では，HTMLソースの参照，正規表現の記述などの操作を，ブラウザやテキストエリア上で，エンドユーザの使い慣れたマウス操作を用いて行えるようにした．これにより，広い表現力を持つ切り出しルールを，エンドユーザが記述することが可能となる．

本研究の貢献は以下の３点となる．

1. ラッピング作業の分析に基づいた負荷箇所の特定
   現状のラッピング作業の分析から，環境の導入，ラッパー検証テストの記述，正規表現などの記法の学習といった，エンドユーザの負荷となる作業を洗い出した．またラッピング作業のビデオ記録から，環境の導入，検証テストの記述，切り出しルールの記述，ファイルの記述におけるミスタイプなど，ラッピング作業者にとって時間を要する作業を特定した．
2. プログラミング開発環境に関する知識の不要化
   Webサービス作成に必要な環境をサーバに置くことにより，エンドユーザがプログラミング開発環境の学習をしなくともラッピングを行えるようになった．
3. 柔軟な情報切り出しルール記述支援の提案
   現状の作業に基づき，広い表現能力を持つ切り出しルールの記述を支援するユーザインタフェースを開発した．マウスでの範囲選択によるHTML取得や，選択式のルール記述が可能となり，切り出しルールの記述に関する利用者の負荷が軽減された．

今後の課題としては，本稿で提案した切り出しルールの記述支援において，ユーザの負荷を低く保ったまま表現力を上げることが挙げられる．

また切り出しルールの記述だけでなく，支援システムの全体的な有用性を評価するため，システムを実装し，まずラッピング作業経験者，続いてエンドユーザによるラッピングを実際に行うことが望まれる．

謝辞

研究の機会を与えてくださり，熱心にご指導下さいました石田亨教授に深く感謝いたします．また，研究方針について日頃からご指導，助言を下さいましたNICTの村上陽平氏，共に言語資源のラッピング成果を蓄積し，研究の基盤を築いて下さった言語資源ラッピングプロジェクトの皆様に感謝致します．最後に，様々なご指導を下さいました松原繁夫助教授はじめ，石田研究室の皆様にもここで感謝の意を示させて頂きます．

参考文献

[1] Ishida, T.: Language Grid: An Infrastructure for Intercultural Collaboration, *IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06)* (2006).

[2] Sahuguet, A. and Azavant, F.: WysiWyg Web Wrapper Factory (W4F).

[3] HUY, H. P., KAWAMURA, T. and HASEGAWA, T.: How to Make Web Sites Talk Together - Web Service Solution.

[4] Kushmerick, N.: Wrapper induction: efficiency and expressiveness, *Artif. Intell.*, Vol. 118, No. 1-2, pp. 15–68 (2000).

[5] Soderland, S.: Learning Information Extraction Rules for Semi-Structured and Free Text, *Mach. Learn.*, Vol. 34, No. 1-3, pp. 233–272 (1999).

[6] Muslea, I., Minton, S. and Knoblock, C.: A hierarchical approach to wrapper induction, *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, New York, NY, USA, ACM Press, pp. 190–197 (1999).

付録

A.1　言語資源ラッパーに関するファイル

A.1.1　ラッパーテンプレート

```
/*
 * $Id$
 *
 * langrid ライセンス条項
 */
package {packageName};

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

import jp.go.nict.langrid.language.Language;
import jp.go.nict.langrid.service_1_0.InvalidParameterException;
import jp.go.nict.langrid.service_1_0.LanguagePairNotUniquelyDecidedException;
import jp.go.nict.langrid.service_1_0.ServiceConfigurationException;
import jp.go.nict.langrid.service_1_0.UnknownException;
import jp.go.nict.langrid.service_1_0.UnsupportedLanguagePairException;
import jp.go.nict.langrid.service_1_0.paralleltext.ParallelText;
import jp.go.nict.langrid.service_1_0.paralleltext.SearchFailedException;
import jp.go.nict.langrid.service_1_0.typed.SearchMethod;
import jp.go.nict.langrid.wrapper.common.iniFileUtil.IniFunctions;
import jp.go.nict.langrid.wrapper.common.iniFileUtil.paralleltext.HTML_ServiceDescription;
import jp.go.nict.langrid.wrapper.paralleltext.AbstractParallelTextService;
import jp.go.nict.langrid.wrapper.paralleltext.HTMLParallelTextService;

/**
* HTML で提供される用例対訳資源をラッピングする場合のテンプレート.
*
* @author Author: {authorName}
*/

public class {className} extends AbstractParallelTextService
{
/**
 * 対訳データを格納します.
 */
private Collection<String[]> texts = new ArrayList<String[]>();

/**
```

```
 * 利用可能な言語を格納します.
 */
private Map<Language, Integer> languages  = new HashMap<Language, Integer>();

/**
 * コンストラクタで発生した例外のフラグです.
 */
private ServiceConfigurationException exceptionFlag = null;

/**
 * ini ファイルに対応するオブジェクト
 */
private HTML_ServiceDescription sd = null;

/**
 * LocalParallelTextService のコンストラクタです.
 * ※改変、対訳ペアのファイル名として、自動生成された.dat を指定
 * また、dat ファイルの生成もコンストラクタ内で行う。
 */
public {className}()
{
try
{
//ini ファイルを読み込み、ini ファイルを表すオブジェクト、サービスを表すオブジェクト
を生成
sd = HTMLParallelTextService.getServiceDescription(this.getClass().getResourceAsStream("resource

// 対応言語対の指定
this.setSupportedLanguagePairs(IniFunctions.integrateLanguagePathes(sd));
}
catch(ServiceConfigurationException e)
{
// 例外は doSearch 時に処理
this.exceptionFlag = e;
}
}

/**
 * オーバーライドした search．コンストラクタ内で例外が発生したかどうかを判断する処理
が入る．
 */
public ParallelText[] search(String sourceLang, String targetLang,
String source, String searchMethod
)
throws InvalidParameterException, LanguagePairNotUniquelyDecidedException
```

```
, SearchFailedException, ServiceConfigurationException
, UnknownException, UnsupportedLanguagePairException
{
// コンストラクタ内で ini ファイルの処理に関する例外が発生していた場合の通知処理
if (exceptionFlag != null) {
throw exceptionFlag;
}
return super.search(sourceLang,targetLang,source,searchMethod);
}


/* (非 Javadoc)
 * @see jp.go.nict.langrid.wrapper.paralleltext.AbstractParalleltextService#doSearch(jp.go.nict
 */
@Override
protected Collection<ParallelText> doSearch(Language sourceLang,
Language targetLang, String source, SearchMethod searchMethod)
throws InvalidParameterException, SearchFailedException,
ServiceConfigurationException, UnknownException
{
return HTMLParallelTextService.doSearchImpl(sourceLang,targetLang,source,searchMethod,
this.getClass(),sd,texts,languages);
}
}
```

A.2　作業分析に関する資料

A.2.1　作業の所要時間の詳細なデータ

No.	作業名	作業のゴール	時間
1	JDKのセットアップ	ラッピング環境の準備	0:09:00
2	Eclipseのセットアップ	ラッピング環境の準備	0:04:53
3	Eclipseの初期設定	ラッピング環境の準備	0:04:19
4	Subversionのインストール	ラッピング環境の準備	0:05:29
5	鍵の作成	ラッピング環境の準備	0:03:39
6	リポジトリに接続(推定)	ラッピング環境の準備	0:03:00
7	プロジェクトのチェックアウト(推定)	ラッピング環境の準備	0:02:00
8	Tomcatのインストール	ラッピング環境の準備	0:03:16
9	Tomcat関係のファイルの修正(推定)	ラッピング環境の準備	0:03:00
10	ページの概観	プロジェクト、クラスの作成	0:01:00
11	プロジェクトの作成	プロジェクト、クラスの作成	0:00:51
12	パッケージ作成	プロジェクト、クラスの作成	0:01:39
13	クラス作成・記述	プロジェクト、クラスの作成	0:05:15
14	テスト作成	ラッパー検証テストの作成	0:02:00
15	テスト記述	ラッパー検証テストの作成	0:08:15
16	Ini作成	設定ファイルの記述	0:00:54
17	資源提供者などの判断	設定ファイルの記述	0:01:24
18	資源提供者などの記述	設定ファイルの記述	0:01:19
19	文字コードの記述	設定ファイルの記述	0:01:30
20	iniの整形	設定ファイルの記述	0:00:46
21	URLの追加	対訳の抽出部の記述	0:00:34
22	webページから対訳の発見	対訳の抽出部の記述	0:00:05
23	HTMLから対訳部分の発見	対訳の抽出部の記述	0:00:10
24	iniの記述	設定ファイルの記述	0:00:49
25	正規表現1の記述	対訳の抽出部の記述	0:04:36
26	抽出テスト、文法ミスの修正	設定ファイルの記述(文法ミス)	0:02:38
27	抽出テストの実行、結果の確認	設定ファイルの記述(文法ミス)	0:00:15
28	正規表現1の修正	設定ファイルの記述(文法ミス)	0:02:26
29	抽出テストの実行、結果の確認	設定ファイルの記述(文法ミス)	0:00:15
30	文法ミスの修正	設定ファイルの記述(文法ミス)	0:01:09
31	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:13
32	正規表現1の修正	切り出しルールの記述	0:00:19
33	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:23
34	正規表現1の修正	切り出しルールの記述	0:02:16
35	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:52
36	正規表現1の修正	切り出しルールの記述	0:01:21
37	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:10
38	正規表現1の修正	切り出しルールの記述	0:02:00
39	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:14
40	正規表現1の修正	切り出しルールの記述	0:00:49
41	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:01:02
42	webページから対訳の発見	切り出しルールの記述	0:00:03
43	HTMLから対訳部分の発見	切り出しルールの記述	0:00:20
44	URLの追加、iniの整形	切り出しルールの記述	0:02:03
45	正規表現2の記述	切り出しルールの記述	0:02:40
46	抽出テストの実行、文法の修正	設定ファイルの記述(文法ミス)	0:00:51
47	address->adress	設定ファイルの記述(文法ミス)	0:28:46
48	正規表現2の記述	切り出しルールの記述	0:01:20
49	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:01:39
50	正規表現2の記述	切り出しルールの記述	0:02:46
51	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:59
52	正規表現2の記述	切り出しルールの記述	0:01:42
53	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:34
54	正規表現2の記述	切り出しルールの記述	0:01:53
55	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:54
56	正規表現2の記述	切り出しルールの記述	0:00:40
57	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:11
58	正規表現2の記述	切り出しルールの記述	0:03:39
59	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:03:14
60	正規表現2の記述	切り出しルールの記述	0:01:16
61	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:29
62	URLの追加	切り出しルールの記述	0:00:37
63	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:57
64	URLの追加	切り出しルールの記述	0:00:35
65	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:02:35
66	URLの追加	切り出しルールの記述	0:00:23
67	抽出テストの実行、結果の確認	切り出しルールの記述(結果確認作業	0:00:42
68	テストの修正、結果の確認	ラッパー検証テストの作成	0:00:57
69	Commonのビルド	ラッパーのビルド、テスト	0:00:30
70	ラッパーのビルド	ラッパーのビルド、テスト	0:01:26
71	Jarの設定	ラッパーのビルド、テスト	0:02:12
72	Jarテストの実行	ラッパーのビルド、テスト	0:00:53
73	webサービステストの作成	ラッパー検証テストの作成	0:01:55
74	webサービステストの記述	ラッパー検証テストの作成	0:08:58
75	webサービステストの実行	ラッパーのビルド、テスト	0:00:17

A.3　支援システムの主なソースコード

A.3.1　HTML 取得支援のためのサーバ側処理

```
package support.htmlsource;

import java.io.IOException;
import java.net.URLDecoder;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import jp.go.nict.langrid.wrapper.common.ContentUtil;

import org.json.JSONObject;

public class HTMLSupportPost extends HttpServlet{

private static final String onLoadMethod = "onload=\"checkBrowser()\"";
private static final String HTMLDetectCodeName = "HTMLDetectCode";

protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws IOException {
String HTMLDetectCodeStr = ContentUtil.getLocal(
this.getClass().getResourceAsStream(HTMLDetectCodeName));

String jsonData = URLDecoder.decode(request.getParameter("jsonData"),"UTF-8");
//TODO delete
System.out.println("HTMLSupport data:"+jsonData);

String urlStr;
String chrStr;
String content;

try {

JSONObject reqObj = new JSONObject(jsonData);

response.setContentType("text/html;charset=UTF-8");
request.setCharacterEncoding("UTF-8");

//init URL
urlStr = reqObj.getString("url");
//init charset
chrStr = reqObj.getString("charSet");
```

```
content = ContentUtil.getHttp(urlStr, chrStr);

} catch (Exception e1) {
response.getWriter().println("Error in HTMLSupportPost: \n"+e1.toString());
return;
}

//body の onLoad プロパティを記述
StringBuffer refinedContent = new StringBuffer(content.replaceFirst("< *[Bb][Oo][Dd][Yy]",
"<body "+onLoadMethod));

//body の閉じタグの次に，実際の HTML 取得支援処理を記述
//実際にはパース処理が必要だが，省略 (body タグ内で">"が出現するとエラー)
int indexAfterBodyTag = refinedContent.indexOf(">",refinedContent.indexOf("<body"));
refinedContent.insert(indexAfterBodyTag+1,HTMLDetectCodeStr );

response.getWriter().println(refinedContent);
};
}
```

A.3.2 HTML 取得支援のための JavaScript

```
<script language="JavaScript"><!--

myID = "myCursor";                    // DIV タグで付けた ID

defaultNeighborHTML = "範囲を選択してください";
defaultToClip = "";

mouseDown = 0; //マウスが押下されている間は 1

function checkBrowser(){
   isIE = document.all;

   if (isIE){            // IE?
      document.all["neighborHTML"].innerHTML = defaultNeighborHTML;
      document.all["toClip"].innerHTML = defaultToClip;
      document.onmousedown = function(){mouseDown=1};
      document.onmouseup = function(){mouseDown=0};
      document.onmousemove = myMoveIE;
      document.onmousewheel = myMoveIE;
   }else{
      alert("Sorry... this system is for IE only.");
   }
}
```

```
function myMoveIE(){ // IE でマウスが動いた
  myObj=document.all[myID].style;

  var selectedHTML = document.selection.createRange().htmlText;

  if(selectedHTML.length>0){
   document.all["neighborHTML"].innerHTML = "<xmp>"+selectedHTML+"</xmp>";
  }else
    document.all["neighborHTML"].innerHTML = defaultNeighborHTML;

  if(selectedHTML.length>0){
   document.all["toClip"].innerText = "内容をクリップボードにコピー";
   if(mouseDown == 0) return; //マウスが押下されておらず，文字列が選択されていれば
欄を移動しない
  }else{
   document.all["toClip"].innerText = defaultToClip;
  }

  myObj.left = 8 + document.body.scrollLeft + "px";
  myObj.top  = 8 + window.event.clientY + document.body.scrollTop + "px";
 }

function clipIE(){
textWithXmp = document.all['neighborHTML'].innerHTML
textToClip = textWithXmp.substring(5,textWithXmp.length-6);
clipboardData.setData("Text",textToClip);
}
// --></script>
<div id="myCursor" style="position:absolute; z-index:1;" width="100%">
<table bgcolor="#333355" border="1" cellpadding="6" cellspacing="0" width="100%">
<tr>
<td>
<font id="neighborHTML" color="#FFFFFF" ;>
</font>
</td>
</tr>
<tr>
<td bgcolor="#DDDDFF" style="cursor:pointer" onmouseover="this.style.background='#BBBBDD';"
  onclick = "clipIE()" onmouseout="this.style.background='#DDDDFF'">
<font id="toClip" color="#000000">
</font>
</td>
</tr>
</table>
```

```
</div>
```

A.3.3 クライアント側 HTML 取得支援画面

```
package wrapSupport.client;

import java.util.ArrayList;

import com.google.gwt.core.client.GWT;
import com.google.gwt.http.client.URL;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.rpc.ServiceDefTarget;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.ClickListener;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FocusListener;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.Hidden;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.MouseListener;
import com.google.gwt.user.client.ui.NamedFrame;
import com.google.gwt.user.client.ui.PopupPanel;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.VerticalPanel;
import com.google.gwt.user.client.ui.Widget;

public class RegexSupportPanel extends Composite{

private static TextRefineServiceAsync trs =
   (TextRefineServiceAsync) GWT.create(TextRefineService.class);
static {
   ((ServiceDefTarget) trs).setServiceEntryPoint(
   "http://localhost:8080/GWT_001Server/WordRefiner");
}

static String tinyRegexPostServletPath =
"http://localhost:8080/GWT_001Server/tinyRegexUtilPost";

private final long serialVersionUID = 1L;
private static final String defaultText = "(default)";

private final VerticalPanel vp = new VerticalPanel();
private final HorizontalPanel hp = new HorizontalPanel();
```

```
private final TextArea ta = new TextArea();
private final Button refButton = new Button();
private final Button clrButton = new Button();
private URLInfoGrid parentRegexPanel;
private NamedFrame parentPreviewFrame;
private ExtractionPanel parentExtractionPanel;
private FormPanel hlButtonPanel ;
private final HorizontalPanel hlButtonhp = new HorizontalPanel();
final private Button hlButton = new Button();

private final int DATA_INDEX_JSON = 1;
private final int DATA_INDEX_REGEX = 2;

public RegexSupportPanel(ExtractionPanel pPanel){
//set parent
parentRegexPanel = pPanel.getExtractionRegexPanel();
parentPreviewFrame = pPanel.getPreviewFrame();
parentExtractionPanel = pPanel;
hlButtonPanel = new FormPanel(parentPreviewFrame);

ta.setText(defaultText);
ta.setCharacterWidth(150);
ta.setHeight("100px");

//デフォルト表示でフォーカスを得ると，内容をブランクに
ta.addFocusListener(new FocusListener(){

public void onFocus(Widget arg0) {
if(ta.getText().equals(defaultText))
ta.setText("");
}
public void onLostFocus(Widget arg0) {
if(ta.getText().equals(""))
ta.setText(defaultText);
}

});


refButton.setText("Refine selected text.");
//クリックで選択範囲の書換
refButton.addClickListener(new ClickListener(){

public void onClick(Widget sender) {
```

```
if(ta.getSelectionLength() > 0){

PopupRegexSupport prs = new PopupRegexSupport();

int left = refButton.getAbsoluteLeft()+100;
int top = refButton.getAbsoluteTop();

//ポップアップの内容を支援サービスから取得
ArrayList optList = initPopupList();

prs.setPopupPosition(left, top);
prs.show();
}
}
});


clrButton.setText("Clear");
//クリックでテキストエリアの消去
clrButton.addClickListener(new ClickListener(){
public void onClick(Widget sender) {
ta.setText(defaultText);
}
});

//init preview regex button
hlButton.setText("Preview tinyRegex");
hlButtonPanel.setMethod(FormPanel.METHOD_POST);
hlButtonPanel.setEncoding(FormPanel.ENCODING_URLENCODED);
hlButtonPanel.setAction(tinyRegexPostServletPath);
hlButton.addClickListener(new ClickListener(){
public void onClick(Widget arg0) {

((Hidden)hlButtonhp.getWidget(DATA_INDEX_JSON)).setValue(
URL.encode(
parentExtractionPanel.getExPanel_top_left().getPreviewJson()));

((Hidden)hlButtonhp.getWidget(DATA_INDEX_REGEX)).setValue(
URL.encode(ta.getText()));

hlButtonPanel.submit();

}
});
hlButtonhp.add(hlButton);
```

```
hlButtonhp.add(new Hidden("jsonData"));
hlButtonhp.add(new Hidden("tinyRegex"));
hlButtonPanel.add(hlButtonhp);

hp.add(refButton);
hp.add(clrButton);
hp.add(hlButtonPanel);

vp.add(ta);
vp.add(hp);

initWidget(vp);
}

private ArrayList initPopupList(){
ArrayList rtn= new ArrayList();

//TODO 実装
for(int i=0;i<5;i++){
HTML aHTML = new HTML("option No."+i);
rtn.add(aHTML);
}

return rtn;
}

private class PopupRegexSupport extends PopupPanel {

VerticalPanel vp = new VerticalPanel();

public PopupRegexSupport() {
super(true);

String[] refinedWordList;
//TODO 言語コードの取得処理を実装
String[] languageCodes = {"ja","en"};
trs.getRefinedWords(ta.getSelectedText(),languageCodes, new AsyncCallback(){

public void onFailure(Throwable arg0) {
addRefinedWords(new String[]{"Error in getting refined words."});
}

public void onSuccess(Object arg0) {
String[] resultStr = (String[]) arg0;
if(resultStr.length==0)
```

```
addRefinedWords(new String[]{"(No refined words.)"});
else
addRefinedWords(resultStr);
}

});

//リストからの popup の初期化処理
//対象文字列を popup に追加
HTML targetWordHTML = new HTML();
targetWordHTML.setText("\""+ta.getSelectedText() + "\"を書き換え");
targetWordHTML.setStyleName("wsup-exPanel-popupTarget");
vp.add(targetWordHTML);

vp.setBorderWidth(1);
add(vp);

setStyleName("wsup-rsPanel");
}

private void addRefinedWords(String[] refinedWordList){
//変更先の文字列の追加
for(int i=0;i<refinedWordList.length;i++){
final HTML refinedWordHTML = new HTML();
refinedWordHTML.setText(refinedWordList[i]);
refinedWordHTML.setStyleName("wsup-exPanel-popupRefined");
refinedWordHTML.addMouseListener(new MouseListener(){

public void onMouseDown(Widget sender, int arg1, int arg2) {
//選択範囲を書換
ta.setText(
ta.getText().substring(0,ta.getCursorPos())
+ "【"+refinedWordHTML.getText()+"】"
+ ta.getText().substring(
ta.getCursorPos()+ta.getSelectionLength(),ta.getText().length())
);
}

public void onMouseEnter(Widget sender) {
sender.setStyleName("wsup-exPanel-popupRefinedOver");

}

public void onMouseLeave(Widget sender) {
sender.setStyleName("wsup-exPanel-popupRefined");
```

```
}

public void onMouseMove(Widget sender, int arg1, int arg2) {
}

public void onMouseUp(Widget sender, int arg1, int arg2) {
}

});
refinedWordHTML.setHeight("20px");
vp.add(refinedWordHTML);
}
}
}

}
```

A.3.4 プロジェクト支援のためのサーバ側処理

```
package support.project;



import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.util.ArrayList;

import jp.go.nict.langrid.wrapper.common.ContentUtil;
import support.regex.feedback.RegexEnhilighter;
import wrapSupport.client.ProjectService;
import wrappingSupport.conductor.WrappingConducter;

import com.google.gwt.json.client.JSONArray;
import com.google.gwt.json.client.JSONObject;
import com.google.gwt.json.client.JSONString;
import com.google.gwt.user.server.rpc.RemoteServiceServlet;

public class ProjectFacade  extends RemoteServiceServlet implements ProjectService {

private static final String workspacePath = "c:\\workspace" ;
```

```
private static final String projectPreffix = "com.product.support" ;
private static final String[] neededFiles
= {"description.json"};

public String[] getProjectList() {

final ArrayList<String> validProjectNames = new ArrayList<String>();

//list up all projects on the server workspace.
final File workspace = new File(workspacePath);
final File[] files = workspace.listFiles();

for(int i=0;i<files.length;i++){
if(isValid(files[i]))
validProjectNames.add(getProjectTail(files[i]));
}

// TODO 自動生成されたメソッド・スタブ
return validProjectNames.toArray(new String[]{});
}

private boolean isValid(final File file) {
//make sure the file is a project
if(!file.isDirectory())
return false;

//check the directory name
if(!file.getName().matches("com\\.product\\.support\\.[a-z][a-zA-Z0-9_]+"))
return false;

//check the directory has important files
final String filePath =file.toString();
for(int i=0;i<neededFiles.length;i++){
final File aFile = new File(filePath+"\\"+neededFiles[i]);
if(!aFile.exists())
return false;
}

// TODO 自動生成されたメソッド・スタブ
return true;
}

private String getProjectTail(final File file){
final String projectName = file.getName();
if(projectName.contains("."))
```

```
return projectName.substring(projectName.lastIndexOf(".")+1);
else return "error: bad project name";
}

public String getProject(final String projectName) {
final File descriptionFile = new File(
workspacePath+"\\"+projectPreffix+"."+projectName+
"\\description.json");

final StringBuffer buff=new StringBuffer();

try {
final BufferedReader br = new BufferedReader(
new InputStreamReader(new FileInputStream(descriptionFile),"UTF-8"));

String line;
while((line = br.readLine())!=null)
buff.append(line);
} catch (final IOException e) {
return null;
}

return buff.toString();
}

public String buildProject(final String projectName){
final String progress = "buildProject called";

try {
WrappingConducter.createWrapperFromName(
projectPreffix+"."+projectName,
projectName.substring(0,1).toUpperCase()+projectName.substring(1));
} catch (final Exception e) {
e.printStackTrace();
return e.toString();
}

return progress;
}

public void putProject(final String projectName, final String jsonFile) {

final File descriptionFile = new File(
workspacePath+"\\"+projectPreffix+"."+projectName+
"\\description.json");
```

```
PrintWriter toFile;
try {
toFile = new PrintWriter(new OutputStreamWriter(
new FileOutputStream(descriptionFile),"UTF-8"));
toFile.print(jsonFile);
toFile.flush();
toFile.close();
} catch (final Exception e) {
return;
}
}

public String getRegexFeedBack(final JSONObject aUrlSet){

//get content
final String aUrl = ((JSONString)aUrlSet.get("url")).stringValue();
final String chrStr = "s-jis";
String content;
try {
content = ContentUtil.getHttp(aUrl,chrStr);
} catch (final Exception e) {
return("error: "+e.toString());
}

//ignore items
final JSONArray ignores = (JSONArray)aUrlSet.get("ignore");
for(int i=0;i<ignores.size();i++){
content = content.replaceAll(
((JSONString)ignores.get(i)).stringValue(), "");
}

//enhilight regex
final String aRegex = ((JSONString)aUrlSet.get("regex")).stringValue();
final String result=RegexEnhilighter.enhilight(content, aRegex);

return result;
}
}
```

A.3.5 ルール修正支援のためのサーバ側処理

```
package support.regex;


import java.util.ArrayList;
```

```
import wrapSupport.client.TextRefineService;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

public class WordRefiner  extends RemoteServiceServlet implements TextRefineService {

public String[] getRefinedWords(String targetWord, String[] languageCodes) {

System.out.println("getRefinedWords called");

ArrayList<String> rtn = new ArrayList<String>();

// 対訳として切り出す選択肢を返す
for(int lc=0;lc<languageCodes.length;lc++){
rtn.add(
getLanguageNameFromLanguageCode(
languageCodes[lc])+"として切り出し");
}
// 一般化の選択肢を返す
rtn.addAll(getWildCards(targetWord));
// 繰り返しの選択肢を返す
rtn.addAll(getRepeats(targetWord));

return rtn.toArray(new String[]{});
}

private String getLanguageNameFromLanguageCode(String languageCode){
//TODO 言語コードの仕様に基づき実装

if(languageCode.equals("ja"))
return("日本語");
else if(languageCode.equals("en"))
return("英語");
else if(languageCode.equals("to"))
return("トンガ語");
else return(languageCode);
}

private ArrayList<String> getWildCards(String aWord) {
ArrayList<String> rtn = new ArrayList<String>();

//TODO もっと詳細なワイルドカードを作成できるように

rtn.add("任意の文字列");
```

```
//数値かどうか
if(aWord.matches("[0-9０-９]++"))
rtn.add("数字");

//タグかどうか
if(aWord.matches("<[^/][^>]*>")){
rtn.add("タグ");
rtn.add(aWord.split("[< ]")[1]+"タグ");
}

//閉じタグかどうか
if(aWord.matches("</[^>]*>")){
rtn.add("閉じタグ");
}

//アルファベットかどうか
if(aWord.matches("[a-zA-Z]+?"))
rtn.add("アルファベット");

return rtn;
}

private ArrayList<String> getRepeats(String aWord) {
ArrayList<String> rtn = new ArrayList<String>();

//TODO もっと詳細なワイルドカードを作成できるように

rtn.add("\""+aWord+"\""+"が０回以上");
rtn.add("\""+aWord+"\""+"が１回以上");

return rtn;
}
}
```

A.3.6　切り出し結果確認支援のためのサーバ側処理１

```
package support.regex.feedback;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLDecoder;
import java.util.ArrayList;
import java.util.HashMap;
```

```
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import jp.go.nict.langrid.service_1_0.ServiceConfigurationException;
import jp.go.nict.langrid.util.StreamUtil;
import jp.go.nict.langrid.wrapper.common.ContentUtil;

public class TinyRegexUtilPost extends HttpServlet{

private static final String cacheFolder="c:/cache/";
private static final boolean EnableCache=false;

protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws IOException {

String jsonData = URLDecoder.decode(request.getParameter("jsonData"),"UTF-8");
String tinyRegex = URLDecoder.decode(request.getParameter("tinyRegex"),"UTF-8");

String urlStr;
String chrStr;
String[] ignores;

try {

JSONObject reqObj = new JSONObject(jsonData);

response.setContentType("text/html;charset=UTF-8");
request.setCharacterEncoding("UTF-8");

//init URL
urlStr = reqObj.getString("url");
//init charset
chrStr = reqObj.getString("charSet");
//init ignores
ArrayList<String> ALIgnore = new ArrayList<String>();
JSONArray JAIgnore = reqObj.getJSONArray("ignore");
```

```
for(int i=0;i<JAIgnore.length();i++)
ALIgnore.add(JAIgnore.getString(i));
ignores = ALIgnore.toArray(new String[]{});

} catch (JSONException e1) {
response.getWriter().println("Error in RegexUtilPost: \n"+e1.toString());
return;
}

if (urlStr == null
|| chrStr == null) {
response.getWriter().println("usage: url=...&chr=...&rgx=...");
return;
}

String content="";

//TODO キャッシュの処理
//OutOfDate 処理や、複数アドレスへの対応
int readFlag=0;
if (EnableCache) {
File cacheFile = new File(cacheFolder + "cache.htmlbak");
StringBuffer contentBuffer = new StringBuffer();
if (cacheFile.exists()) {
BufferedReader r = new BufferedReader(StreamUtil
.createUTF8Reader(new FileInputStream(cacheFile)));
if (r.readLine().equals(urlStr)) {
String line;
while ((line = r.readLine()) != null) {
contentBuffer.append(line+"\n");
}
content = contentBuffer.toString();
readFlag = 1;
}
}
}
try {
if (readFlag == 0)
content = ContentUtil.getHttp(urlStr, chrStr);
} catch (ServiceConfigurationException e) {
response.getWriter().println(
"Error in RegexUtilPost: \n" + e.toString());
return;
}
```

```
if (readFlag == 0)
ContentUtil.setLocal(cacheFolder + "cache.htmlbak",
urlStr+"\n"+
content,
"UTF-8");

//無視する文字列の処理
if(ignores!=null){
for (int i=0;i<ignores.length;i++){
content = content.replaceAll(ignores[i],"");
}
}

String result=RegexEnhilighter.enhilight(content, makeRegexFromTinyRegex(tinyRegex));

response.getWriter().println(result);
};

protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException {

System.out.println("TinyRegexUtilPost.doGet called.");
response.getWriter().println("TinyRegexUtil:please call via POST");
};

private String makeRegexFromTinyRegex(String tinyRegex){
//TODO delete
System.out.print("tinyRegex:"+tinyRegex);
//【xxx として切り出し】をラベル xxx に置換
String labelReplaced = tinyRegex.replaceAll("\\+", "\\\\+").replaceAll("【([^】]+?)
として切り出し】", "(\\\\:$1:.+?)")
;

//ワイルドカードを置換
labelReplaced = labelReplaced.replaceAll("【任意の文字列】", ".*?")
.replaceAll("【閉じタグ】", "</[^>]*>")
.replaceAll("【\"(.+?)\"が 0 回以上】", "(?:$1)*?")
.replaceAll("【\"(.+?)\"が 1 回以上】", "(?:$1)+?");

return labelReplaced;
}
}
```

A.3.7　切り出し結果確認支援のためのサーバ側処理２

```
package support.regex.feedback;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class RegexEnhilighter {

public static String enhilight(String content,String regex){

//TODO delete
System.out.print("regex:"+regex);

//content = content.replaceAll("\n","");
StringBuffer rtnStr=new StringBuffer(content.length()*2);

ArrayList<String> tagRegexList = new ArrayList<String>();

Map<Integer,String> numToTag = new HashMap<Integer,String>();

//グループ番号とタグを対応付ける
int tagOrd=0;
Pattern pattern = Pattern.compile("\\(\\\\:(.*?):.*?\\)");
Matcher matcher = pattern.matcher(regex);
tagRegexList.add("");
while(matcher.find()==true){
tagOrd++;

tagRegexList.add(tagOrd,matcher.group(0));

numToTag.put(tagOrd,matcher.group(1));
}

//タグ情報を消去し refinedRegex に
String refinedRegex = regex.replaceAll("\\\\:(.*?):","").replaceAll("\r\n", "\n");
String refinedRegex2 = "<TD width=\"47%\" height=35><FONT face=\"ＭＳ 明朝\"><FONT lang=JA></FON
"<P><FONT face=\"ＭＳ 明朝\">(.+?)</FONT></FONT></P></TD>\n" +
"<TD width=\"53%\" height=35><FONT face=\"ＭＳ 明朝\"><FONT lang=JA></FONT>\n" +
"<P><FONT face=\"ＭＳ 明朝\">(.+?)</FONT></FONT></P></TD></TR>";

//TODO delete
System.out.print("\nrefinedRegex:\n"+refinedRegex);
System.out.print("\nrefinedRegex2:\n"+refinedRegex2);
```

```
if(refinedRegex.equals(refinedRegex2))
System.out.print("equal!!");
int i=0;
System.out.println("\n**********");
while(refinedRegex.charAt(i)==refinedRegex2.charAt(i)&&
i>refinedRegex.length()){
System.out.print(refinedRegex.charAt(i));
i++;
}
System.out.println("\n char "+i+"is"+
new Integer(refinedRegex.charAt(i))+" and "+new Integer(refinedRegex2.charAt(i)));

//正規表現を用いてマッチング
Pattern regexPattern = Pattern.compile(refinedRegex,Pattern.DOTALL);
Matcher regexMatcher = regexPattern.matcher(content);
int count = 0,oldIdx=0,nowIdx=0;
while(regexMatcher.find()==true){

count++;

//マッチした文字列
String machedStr=regexMatcher.group(0);

//マッチした文字列にタグ情報を付与
int groups = regexMatcher.groupCount();
StringBuffer taggedText=new StringBuffer();
int startIdx=regexMatcher.start();
nowIdx=startIdx;

for(int groupNum=0;groupNum<groups;groupNum++){
taggedText.append(content.substring(nowIdx,regexMatcher.start(groupNum+1)));
taggedText.append(addTag(regexMatcher.group(groupNum+1),
numToTag.get(groupNum+1)+"_"+count,groupNum+1));
nowIdx=regexMatcher.end(groupNum+1);
}
taggedText.append(content.substring(nowIdx,regexMatcher.end()));

rtnStr.append(content.substring(oldIdx,startIdx).concat(taggedText.toString())+"\n");
oldIdx = startIdx+machedStr.length();

}
if(count==0){
//TODO delete
System.out.println("no matches");
return content;
```

```
}else
rtnStr.append(content.substring(oldIdx));

return rtnStr.toString();
}

private static String addTag(String word, String tag, int colorNum) {

String[] tagSkeltons={
"<Table border=\"0\" bgcolor=\"{color}\"><Tr><Td>{tag}:{word}</Td></Tr></Table>"
,"<font title=\"{tag}\" style=\"background-color:{color}\">{word}</font>"
,"<div id=\"{tag}\" style=\"background-color:{color}\" onClick=\"enDraggable('{tag}')\">{word}</

String[] colorArray = {"#ffbbbb","#bbffbb","#bbbbff"};

String color = colorArray[colorNum%colorArray.length];

return tagSkeltons[1].replace("{color}",color).replace("{word}",
word.replaceAll("<","&lt;").replaceAll(">","&gt;")).replace("{tag}",tag);

}

}
```

A.3.8 クライアント側 MainWindow

```
package gwt_001.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.core.client.GWT;
import com.google.gwt.json.client.JSONArray;
import com.google.gwt.json.client.JSONObject;
import com.google.gwt.json.client.JSONParser;
import com.google.gwt.json.client.JSONString;
import com.google.gwt.user.client.Command;
import com.google.gwt.user.client.DeferredCommand;
import com.google.gwt.user.client.HTTPRequest;
import com.google.gwt.user.client.ResponseTextHandler;
import com.google.gwt.user.client.Timer;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.WindowResizeListener;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.rpc.ServiceDefTarget;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.ChangeListener;
```

```
import com.google.gwt.user.client.ui.ClickListener;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.TabPanel;
import com.google.gwt.user.client.ui.VerticalPanel;
import com.google.gwt.user.client.ui.Widget;

/**
 * Entry point classes define <code>onModuleLoad()</code>.
 */
public class MainWindow implements EntryPoint,WindowResizeListener {

static String projectNotSelected = "---";

private static ProjectServiceAsync ps =
   (ProjectServiceAsync) GWT.create(ProjectService.class);
static {
   ((ServiceDefTarget) ps).setServiceEntryPoint(
   "http://localhost:8080/GWT_001Server/projectFacade");
}

final private HorizontalPanel topMenu = new HorizontalPanel();
final private VerticalPanel vp = new VerticalPanel();
final private Button createButton = new Button();
final private Button deleteButton = new Button();
final private Button loadButton = new Button();
final private Button saveButton = new Button();
final private TabPanel tp = new TabPanel();
final private ListBox lb = new ListBox();
private String projectNow = projectNotSelected;

//MainWindow 上に配置するパネル
//プロジェクトを扱うもの
//タブで切り替えるもの
MetaDataPanel mdp = new MetaDataPanel();
TestPanel tstp = new TestPanel();
ExtractionPanel ep = new ExtractionPanel();
BuildPanel bp = new BuildPanel();

public void onModuleLoad() {
```

```
vp.setHeight("99%");
vp.setWidth("100%");

//init listbox
lb.setWidth("200px");
lb.setMultipleSelect(false);
initProjectList();
lb.addChangeListener(new ChangeListener(){
public void onChange(Widget arg0) {
loadProject();
}
});

//init loadButton
loadButton.setText("Load");
loadButton.addClickListener(new ClickListener(){
public void onClick(Widget arg0) {
loadProject();
}
});

//TODO implement
//init createButton
createButton.setText("new project");
createButton.setEnabled(false);
//init deleteButton
deleteButton.setText("delete project");
deleteButton.setEnabled(false);

//init saveButton
saveButton.setText("Save");
saveButton.addClickListener(new ClickListener(){
public void onClick(Widget arg0) {
saveProject();
}
});

//init top menu
topMenu.add(lb);
topMenu.add(createButton);
topMenu.add(deleteButton);
topMenu.add(loadButton);
topMenu.add(saveButton);
topMenu.setCellWidth(saveButton, "100%");
topMenu.setStyleName("wsup-root-top");
```

```
vp.add(topMenu);

// Create a tab panel with three tabs, each of which displays a different
// piece of text.
tp.add(mdp, "MetaData");
tp.add(tstp, "Test");
tp.add(ep, "Extraction");
tp.add(bp, "Build");

// Show the 'bar' tab initially.
tp.selectTab(0);

tp.setStyleName("gwt-TabBar");
tp.setWidth("100%");

vp.add(tp);
vp.setStyleName("wsup-root");
vp.setCellHeight(topMenu, "40px");
vp.setCellVerticalAlignment(tp, HasVerticalAlignment.ALIGN_TOP);

// Add it to the root panel.
RootPanel.get().add(vp);

//Hook the window resize event, so that we can adjust the UI.
    Window.addWindowResizeListener(this);
DeferredCommand.add(new Command() {
      public void execute() {
        onWindowResized(Window.getClientWidth(), Window.getClientHeight());
      }
    });
}

private void initProjectList() {

ps.getProjectList(new AsyncCallback(){

public void onFailure(Throwable arg0) {
Window.alert("Error in getting project list.");
lb.addItem("(error)");
}

public void onSuccess(Object arg0) {
String[] resultStr = (String[]) arg0;
if(resultStr.length==0)
```

```
lb.addItem("(no project)");
else
lb.addItem(projectNotSelected);
for (int i=0;i<resultStr.length;i++)
lb.addItem(resultStr[i]);
}});
}

public void onWindowResized(int width, int height) {

if(height>400)ep.setBottomHeight(height-400);

if(width>650)ep.setTopLeftWidth(width-10);
}

private void onProjectSelected(){

String selectedProject = lb.getItemText(lb.getSelectedIndex());

if(selectedProject.equals(projectNotSelected)){
Window.alert("Please select a project.");
return;
}

//when a project selected
projectNow = selectedProject;

ps.getProject(projectNow ,new AsyncCallback(){

public void onFailure(Throwable arg0) {
Window.alert("Error in getting project description.");
lb.addItem("(error)");
}

public void onSuccess(Object arg0) {
String resultStr = (String) arg0;

reload((JSONObject)JSONParser.parse(resultStr));
}});
}

private void loadProject(){
String selectedProject = lb.getItemText(lb.getSelectedIndex());

if(!projectNow.equals(projectNotSelected)
```

```
&& !selectedProject.equals(projectNotSelected)){
if (!Window.confirm("Load project\"" + selectedProject
+ "\" and discard change ?"))
return;
}
onProjectSelected();
}

private void saveProject() {
JSONObject saveJObjTop = new JSONObject();

saveJObjTop.put("projectName", new JSONString(projectNow));
saveJObjTop.put("metaData", mdp.getMetaDataPanelData());
saveJObjTop.put("extraction", ep.getExtractionPanelData());
saveJObjTop.put("build", bp.getBuildPanelData());

ps.putProject(projectNow, saveJObjTop.toString(),
new AsyncCallback() {

public void onFailure(Throwable arg0) {
Window.alert("Error in putting project description.");
lb.addItem("(error)");
}

public void onSuccess(Object arg0) {
((Label) topMenu.getWidget(0)).setText(projectNow
+ " had been saved succesfully.");

Timer t = new Timer() {
public void run() {
((Label) topMenu.getWidget(0)).setText("");
}
};

// Schedule the timer to run once in 5 seconds.
t.schedule(5000);
}
});

}

private void reload(JSONObject jObjTop) {

projectNow = ((JSONString)jObjTop.get("projectName")).stringValue();
mdp.reloadPanel((JSONObject)jObjTop.get("metaData"), projectNow);
```

```
ep.reloadPanel((JSONArray)jObjTop.get("extraction"));
bp.reloadPanel((JSONObject)jObjTop.get("build"), projectNow);
}
}
```