2009－SE－163 (10)
2009／3／18

アプリケーション連携システムのスクリーン・スクレイピングを用いたデバッグシステム

西村紅美† 塚本享治‡
東京工科大学大学院バイオ・情報メディア研究科メディアサイエンス専攻†
東京工科大学大学院バイオ・情報メディア研究科‡

インターネット上にあるアプリケーションの多くは、複数の Web ページから構成されている。アプリケーションから情報を取得するためには、HTML の構造の問題から、必要な部分のみを抽出するスクリーン・スクレイピングが必要である。
スクリーン・スクレイピングを用いてアプリケーションの連携とデバッグを簡単に行うための統合テストシステムを開発したので報告する。このシステムは、情報抽出変換とアプリケーションへのアクセス、デバッグ情報の出力から構成される。そして、開発したシステムを用いていくつかのアプリケーションの連携を行い、有効性を検証した。

Debugging System Using Screen Scraping for Cooperation of Applications

†Kumi Nishimura, ‡Michiharu Tsukamoto
†Media Science Program, Graduate School Bionics, Computer and Media Science, Tokyo University of Technology
‡Graduate School Bionics, Computer and Media Science, Tokyo University of Technology

Internet applications consist of the plural Web pages. The Screen scraping Technique is used to extract necessary parts from HTML pages. However the pages include a lot of layout information, and the page layout are changed frequently. I developed integrated testing system for cooperating applications. This system consists of information extraction used on screen scraping technique, accessing applications, debugging and logging. This paper also reports some experimental application system with effective results.

1. はじめに

インターネット上のアプリケーションには、Web サービスや複数の Web ページから構成される Web アプリケーションがある。これらのアプリケーションを連携する方法として、Web サービスが公開している API などを用いるマッシュアップ技術が注目されているが、多くは Web サービスを対象としたものである。

Web アプリケーションは様々なものがあるが、人間が Web ブラウザを使って利用することが前提であるために連携させることは難しい。Web ページである HTML は仕様に沿っていないものが多く、特定の情報のみを抽出することが難しいためである。また、Web アプリケーションの HTML の構造などが頻繁に変更されるため、その都度デバッグが必要となるが、デバッグは難しい。アプリケーションは通信プロトコルやバックエンドアプリケーションなどとネットワーク上で相互運用され、バグの再現性がないためである。

そこで、Web アプリケーションを対象に、Web ページから特定の情報を抽出するスクリーン・スクレイピングを用いて、アプリケーションの連携とデバッグを簡単に行うための統合テストシステムを開発した。このシステムは、アプリケーションへのアクセス、Web ページからの情

報抽出、通信ログの出力から構成される。アプリケーションの Web ページをインターフェースに用いることで、デバッグをより簡単に行えることを目指した。

2. アプリケーション連携で発生する問題点

2.1. スクリーン・スクレイピングで発生する問題点

アプリケーション連携を、スクリーン・スクレイピングを用いて行おうとすると以下の問題がある。

(1) Web ページである HTML は仕様に沿っていないものが多く、しかも曖昧であるため、HTML から特定の情報抽出は難しい。

(2) レイアウトのための情報が非常に多く、たくさんの広告が埋め込まれているものもあり、特定の情報を取得する構造に適していない。

(3) HTML は Web ブラウザで表示することが前提であり、頻繁にレイアウトなどが変わる。そのたびにデバッグが必要となる。

2.2. アプリケーションの組み合わせの問題点

アプリケーションによっては、以下の情報やコンテンツを用いている場合がある。また、Web ページにフォームを含む場合、パラメータの設定も必要となることが多い。これらを機械的に組み合わせることは難しい。

(1) クッキー情報
クライアントに一時的にデータを書き込んで保存させる。フォームを用いた認証などに使われる。

(2) 隠しフィールドを含むパラメータ
HTML の〈form〉要素の子要素〈input type="hidden"〉を用いて、パラメータを与えながらページ遷移などを行う。

(3) JavaScript
Web ページに動きや対話性を付加する。Ajax など、JavaScript を用いた対話型 Web アプリケーションの実装が増えてきている。

(4) Flash
アニメーションなどを用いた Web コンテンツ。双方向性を持たせたものもある。

2.3. デバッグで発生する問題点

2.1 で述べたように、アプリケーション連携には頻繁にデバッグが必要である。

デバッグのためには、テストコードを作成し、それに基づいてテストとデバッグを行うのが一般的である。しかし、頻繁にデバッグを行うためには、テストコードをその都度作成しなおさなければならない。

特に、インターネット上のアプリケーションの場合、バグの再現性がなく、デバッグは難しい。インターネット上のアプリケーションのバグには、Web ページの HTML 構造の問題から表示できないといったものから、接続先にアクセスできないといったものなど様々な原因があるためである。

3. 解決方法

2 で述べた問題を解決するために、アプリケーション連携のための統合テストシステムを開発した。

(1) Web ページからの情報抽出変換に、研究室で開発したマッシュアップツール[1]を用いた。

(2) クッキー情報やパラメータの設定のために HTTP Client[2]を用いた。

(3) デバッグを行うために、バグの再現性があるかどうかログを利用して判断するフィルターを作成した。デバッグのインターフェースにアプリケーションの Web ページを利用した。

4. アプリケーション連携システム

4.1. システムの基本動作

開発したシステムは、パラメータの設定を自動で行い、Web ページである HTML から情報抽出を用いてあらたなアプリケーションにリクエストを行うといったアプリケーションの連携とデバッグを簡単に行うことを目指した。

このシステムは以下の 4 点の機能から構成される。（図 1）

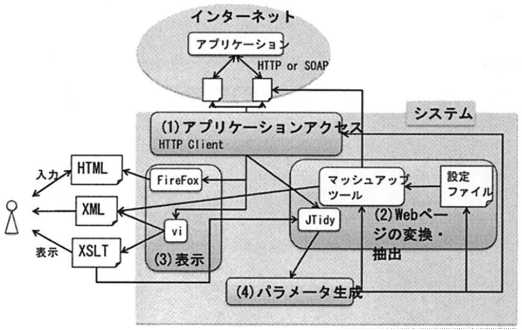

図 1 システムの動作

(1) アプリケーションアクセス
HTTP Client を用いて、アプリケーショ

ンへのアクセスのインターフェースを作成した。

(2) Web ページおよび抽出結果と XSLT の表示
Web ページおよび抽出結果の表示には Web ブラウザ Mozilla Firefox を、XSLT および通信ログの表示には vi を利用した。

(3) Web ページの変換・抽出
HTTP Client を用いて取得した Web ページを変換・抽出するために、JTidy と XSLT およびマッシュアップツールを用いた。

(4) パラメータの生成
フォームを含む Web ページから、XSLT を用いて変換・抽出し、パラメータを生成する。

4.2. アプリケーションアクセス

アプリケーションにアクセスするためには、2.2 で述べたとおり、フォームを利用した認証でクッキー情報が必要となるなど、Web ブラウザと同様の振る舞いが必要となる。そこで、HTTP Clint を用いる。HTTP Clint は、Web ブラウザと同等の機能を持つ Java ライブラリであり、クッキー情報の保持や、パラメータの設定、通信ログを出力することが可能である。しかし、HTTP Client は JavaScript および Flash には未対応である。

図 2 通信ログ

4.3. Web ページからの情報抽出

Web ページから情報を取得するためには、HTML から必要な情報のみ抽出するスクリーン・スクレイピングが必要である。HTML は仕様に沿っていないものが多く、レイアウトのための情報も多い。そこで、取得した Web ページの HTML を、JTidy[3]を用いて XHTML に変換することで仕様に曖昧な部分を解消する。しかし、それだけではまだ不十分である。以下の例は、CPU の商品名と価格を表す HTML (図 3) および XML (図 4) である。

```
−<div class="parts_right">
   <p class="parts_name01">INTEL</p>
 −<p class="parts_name02">
    <strong>Celeron 430 BOX</strong>
  </p>
 −<p class="parts_comment01">
    Celeron 430 [1.80GHz/L2:512kB] Conore-L FSB:800MHz LGA775[35W] リテールBOX
  </p>
  <p class="parts_comment02">BX80557430 Coreマイクロアーキテクチャ [Signle-Core] 65nmプロセス</p>
 −<form method="get" action="/order/cart/cart.php">
  −<table bgcolor="#ff0000" cellspacing="0" cellpadding="1" border="0">
   −<tr>
    −<td>
       <input value="203304" name="ngds_no" type="hidden" />
       <input value="203304" name="ncrd_goods_no" type="hidden" />
       <input value="regist" name="action" type="hidden" />
       <input value="0735858190685" name="rem" type="hidden" />
       <input value="Celeron 430 BOX" name="goods" type="hidden" />
       <input value="Celeron 430 BOX" name="zenkaku" type="hidden" />
       <input value="3600" name="unit" type="hidden" />
       <input value="0" name="datasource" type="hidden" />
       <input value="B" name="limit" type="hidden" />
      −<table bgcolor="#ffffff" cellspacing="2" cellpadding="3" border="0">
       −<tr>
        −<td>
          −<div class="bto_list_price_all">
             <span class="priceyen_red">通販価格 ¥</span>
             <span class="price_red">3,600</span>
             <span class="price_zei">(税込)</span>
           </div>
         </td>
        </tr>
```

図 3 CPU の商品名と価格 (HTML)

```
<?xml version="1.0" encoding="UTF-8"?>
<pc-koubou>
        <cpu>
                <maker>INTEL</maker>
                <name>Celeron 430 BOX</name>
                <price>3,669</price>
        </cpu>
</pc-koubou>
```

図 4 CPU の商品名と価格 (XML)

この HTML では、広告やフォームの隠しフィールド、レイアウトのための情報が多くあり、特定の情報を取得するような構造ではない。

この問題を解決するために、XHTML に XSLT を用いて変換し、情報を整理したのち必要な部分のみ抽出する。

4.4. マッシュアップツール

このツールは、アプリケーションにアクセスし、スクリーン・スクレイピングを用いて情報を XML として取得する Java ライブラリである。アプリケーションがもつパラメータの設定は XML で記述する。(図 5)

```
<ms:mashup xmlns:ms="http://www.tl-dev.net/2007/mashup">
<ms:input>
    <ms:param name="search_word" type="text" description="検索する言葉" />
    <ms:param name="scope" type="select" description="検索範囲" value="japanese">
        <ms:option name="world" value="" />
        <ms:option name="japanese" value="lang_ja" />
    </ms:param>
    <ms:param name="count" type="text" description="検索結果の数" value="50" />
    <ms:param name="safetiy" type="select" description="" value="on">
        <ms:option name="on" value="on" />
        <ms:option name="off" value="off" />
    </ms:param>
    <ms:param name="offset" type="text" value="0" description="オフセット" />
</ms:input>
<ms:variable name="search_result">
    <ms:request protocol="GET"
     request_encoding="UTF-8"
     response_encoding="UTF-8"
     response_form="html"
     url="http://www.google.com/search">
        <ms:param name="hl" value="ja" />
        <ms:param name="lr" value="$scope" />
        <ms:param name="ie" value="UTF-8" />
        <ms:param name="oe" value="UTF-8" />
        <ms:param name="q" value="$search_word" />
        <ms:param name="num" value="$count" />
        <ms:param name="safe" value="$safetiy" />
        <ms:param name="start" value="$offset" />
    </ms:request>
</ms:variable>
```

図 5 アプリケーションのパラメータ設定

4.5. フォームを含む Web ページのパラメータの設定

Web ページにフォームが含まれている場合、フォーム固有のパラメータを持つことが多い。たとえば、検索サイトでは、検索対象言語や表示件数、検索対象とするサイトまたはドメインなどである。アプリケーションによっては、ブラウザ上に表示されているものの他に、隠しフィールドとして設定しているものもある。そこで、対象の Web ページをいったん取得し、スクリーン・スクレイピングを用いてパラメータを抽出し、それを用いた。

フォームにはテキストボックスやラジオボタンなど、複数の種類がある。それらの情報は、HTML の〈form〉要素および子要素に記述されている。また、〈form〉要素にはサーバに送るデータ形式と遷移先が記述されている。フォームのパラメータの種類は表 1 のとおりである。

表 1 フォームのパラメータ

要素名			属性名
〈form〉			method
			action
	〈input〉		type
			name
			value
	〈select〉		name
		〈option〉	value
	〈textarea〉		name
			value

そこで、これらの要素を、XSLT を用いて抽出して用いることで、パラメータを設定した。(図 6)

```
<xsl:template match="form">
    <root>
        <action><xsl:value-of select="@action"/></action>
        <params><xsl:apply-templates
        select="child::*[input]"/></params>
    </root>
</xsl:template>
<xsl:template match="input">
    <param>
        <xsl:attribute name="type">
            <xsl:value-of select="@type"/>
        </xsl:attribute>
        <xsl:attribute name="name">
            <xsl:value-of select="@name"/>
        </xsl:attribute>
        <xsl:attribute name="value">
            <xsl:value-of select="@value"/>
        </xsl:attribute>
    </param>
</xsl:template>
```

図 6 〈input〉要素部分の XSLT

5. アプリケーション連携とデバッグ

5.1. アプリケーション連携

アプリケーションの HTML 構造などは頻繁に変更される。そのため、連携対象のアプリケーション側の変更があった場合、そのままでは Web ページを取得する事ができない。そこで、連携対象のアプリケーションに対してデバッグを行った後、アプリケーション連携を行う。これ以降、英語の Web ページを検索サイトで検索し、結果を翻訳サイトで日本語に翻訳するシステムを例に進めていく。このシステムの連携の流れは以下のとおりである。

(1) 検索サイトのキーワード検索の Web ページを表示する。
(2) 検索サイトのキーワード検索の Web ページからキーワードを入力する。
(3) 検索結果の Web ページからタイトルと文章を抽出する。
(4) 翻訳サイトに抽出した情報と英訳のパラメータを入力する。
(5) 翻訳結果を表示する。

アプリケーション連携の記述は、以下のコードを書く形である。

```
public class Processer {
    public void GoogleTest(String[] args) {
        Process process = new Process();
        process.setUrl("http://www.google.co.jp/");
        process.setParam(args);
        //process.execute();
        String result = process.execute();
        //ShowPage show = new ShowPage();
        //show.getPage(result);
        //System.out.println(result);
        return result;
    }
    public void ExciteTest(String[] args) {
        Process process = new Process();
        process.setUrl("http://www.excite.co.jp/world/");
        process.setParam(args);
        //process.execute();
        String result = process.execute();
        //ShowPage show = new ShowPage();
        //show.getPage(result);
        //System.out.println(result);
        return result;
    }
    public void Show(String[] args){
        Processor processor = new Processor();
        ShowPage show = new ShowPage();
        show.getPage(processor.ExciteTest(processor.GoogleTest(args)));
    }
}
```

図 7 アプリケーション連携の記述

アプリケーション連携時には、システムはブラックボックスであり、ユーザからは検索サイトのキーワード検索の Web ページと翻訳サイトの翻訳結果の Web ページのみが表示される。（図 8）

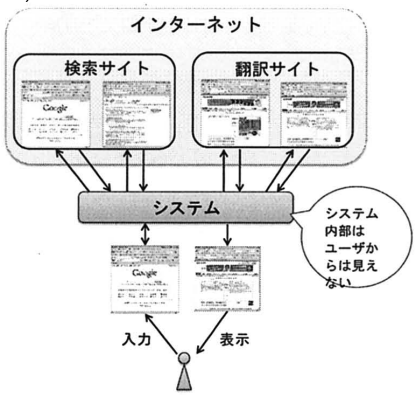

図 8 アプリケーション連携時

5.2. デバッグ

もし、連携中に Web ページが取得できないなどのエラーが起こった場合、システム内部のエラーが発生した箇所がユーザに表示される。たとえば、検索サイトの検索結果の Web ページの変換・抽出結果が正しくなかったとする。システムは、キーワード検索の Web ページと検索結果の Web ページを Fire Fox で、両方の Web ページに対応する抽出結果の XML、XSLT および通信ログを vi で表示する。（図 9）

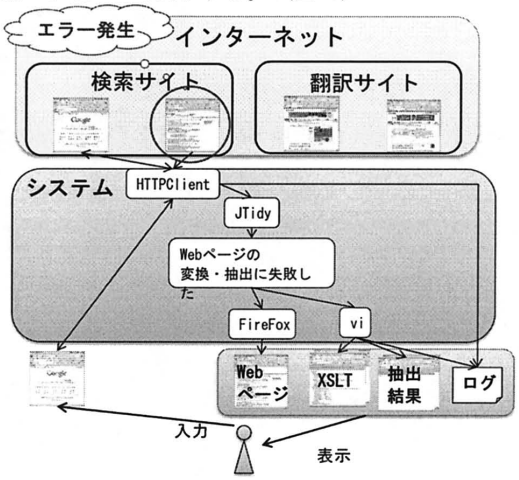

図 9 デバッグ時-Web ページ変換・抽出エラー-

システムに用意したデバッグ時のチェックポイントは以下のとおりである。

(1) Web ページが取得できない場合
URL の変更やネットワークトラブルなどで Web ページが取得できない場合は、エラーメッセージと通信ログをそのまま表示する。図 10 の例では、検索サイトのキーワード検索の Web ページが取得できない場合である。

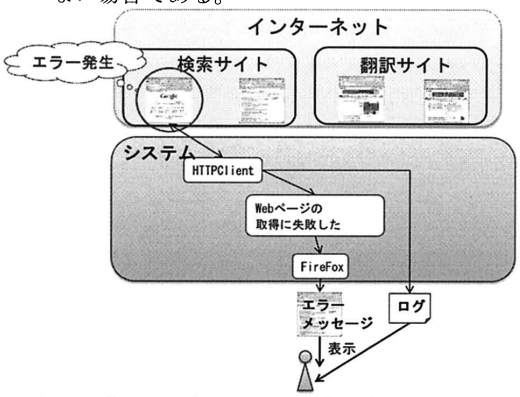

図 10 デバッグ時-Web ページ取得不可-

(2) Web ページの変換・抽出に失敗した場合
前述のとおりである。（図 9）

(3) パラメータが間違っている場合（Web ページにフォームを含むとき）
たとえば、検索結果を翻訳サイトに入力するときにエラーが起こった場合である。このとき、システムは、検索サイトの検索結果の Web ページ、抽出結果および XSLT と、翻訳サイトの原文入力の Web ページ、抽出結果および XSLT と、通信ログを表示する。（図 11）

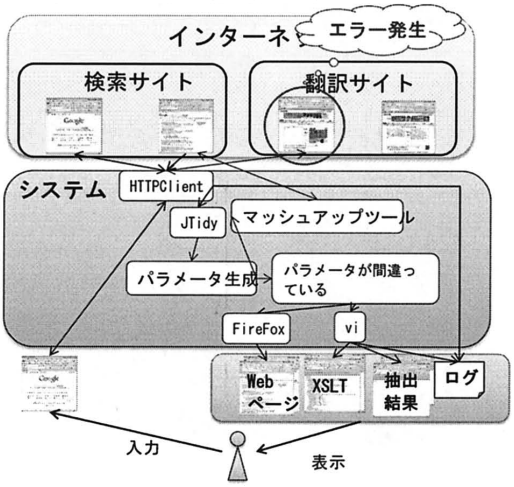

図 11 デバッグ時-パラメータ間違い-

6. 検証

6.1. 実行環境の構築

実際にアプリケーションの連携を行うために実行環境を構築し、2 つの連携パターンを用意した。行ったことは以下の 2 点である。

(1) アプリケーションの構築
JSF とマッシュアップツールを用いてアプリケーションを構築した。構築したアプリケーションは、フォームから入力したテキストを単純に表示するものと、Web サービスを利用するものである。利用した Web サービスは、Google、Excite 翻訳、NIKKEI NET である。

(2) Web ページを変換・抽出するための XSLT の作成
構築したアプリケーションの Web ページに対応する XSLT を作成した。

6.2. 検証したアプリケーションの連携パターン

6.1 の(1)で構築したアプリケーションの連携の順番は、「Google[2]、Excite 翻訳[5]」と「テキストを単純に表示するもの、NIKKEI NET[6]または Google、Excite 翻訳」の 2 パターンを用意した。(図 12)

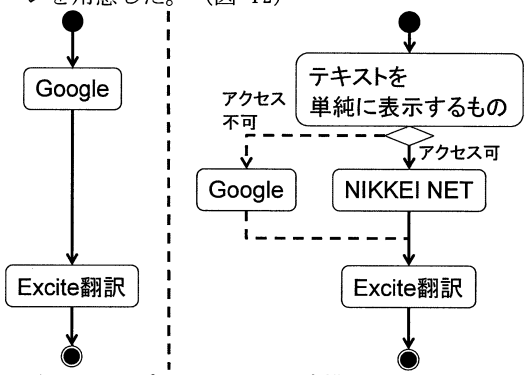

図 12 アプリケーション連携のパターン

(1) 2 アプリケーション連携
この連携は、Google でキーワード検索を行い、Excite 翻訳で翻訳する単純なものである。

(2) 3 アプリケーション連携
ここで用いる「テキストを単純に表示するもの」は、入力フォームに入力したテキストをそのまま表示する Web アプリケーションである。「テキストを単純に表示するもの」で検索するキーワードを表示し、NIKKEI NET で検索し、Excite 翻訳で翻訳する。NIKKEI NET にアクセスできないときは、Google で検索する。

7. 考察

7.1. 2 アプリケーション連携

(1) アプリケーション連携
最初の Google のキーワード入力の Web ページと Excite 翻訳の翻訳結果の Web ページを表示した。これにより、アプリケーション連携が正しく行われていると確認できた。 (図 13)

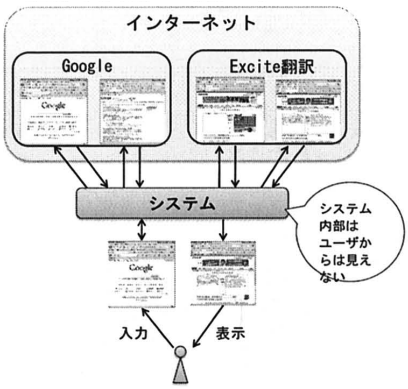

図 13 2 アプリケーション連携時

(2) デバッグ
Google のキーワード入力の Web ページの XSLT を変更した場合、Web ページを変換・抽出できないために、Web ページ、抽出結果、XSLT および通信ログを表示した。これにより、デバッグを行うことができた。

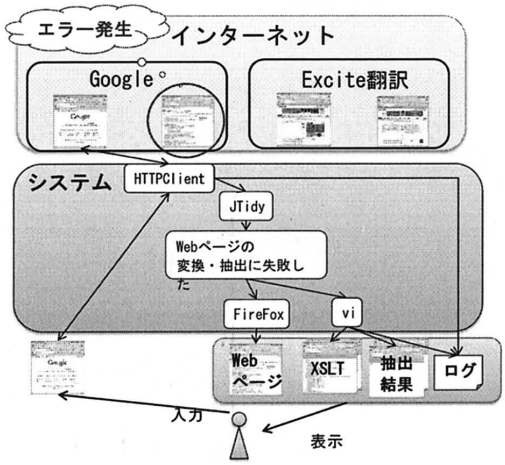

図 14 2 アプリケーションデバッグ時

7.2. 3 アプリケーション連携

7.1 と違うのは、連携先が分岐することである。

(1) NIKKEI NET にアクセスできる場合

① アプリケーション連携時

テキスト表示のキーワード入力の Web ページと Excite 翻訳の翻訳結果の Web ページを表示し、途中の Web ページは表示しなかった。（図 15）

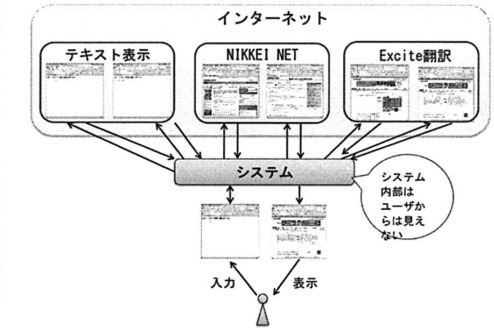

図 15 3アプリケーション連携時 1

② デバッグ時

該当する Web ページ、抽出結果、XSLT および通信ログを表示した。例として、NIKKEI NET にキーワードを入力するときにエラーが起こった場合をあげる。（図 16）

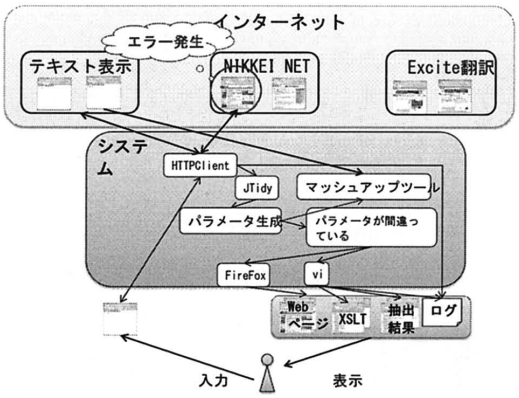

図 16 3アプリケーションデバッグ時 1

(2) NIKKEI NET にアクセスできない場合

① アプリケーション連携時

テキスト表示のキーワード入力の Web ページと Excite 翻訳の翻訳結果の Web ページを表示し、途中の Web ページは表示しなかった。（図 17）

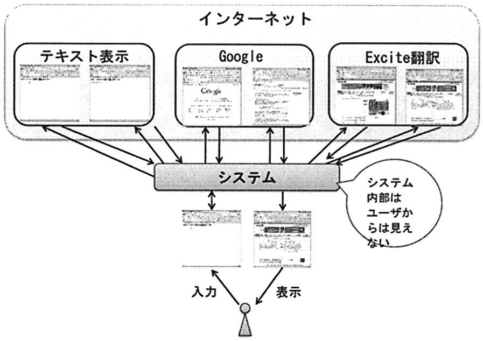

図 17 3アプリケーション連携時 2

② デバッグ時

該当する Web ページ、抽出結果、XSLT および通信ログを表示した。例として、Excite 翻訳に検索結果を入力し、翻訳結果を取得するときにエラーが起こった場合をあげる。（図 18）

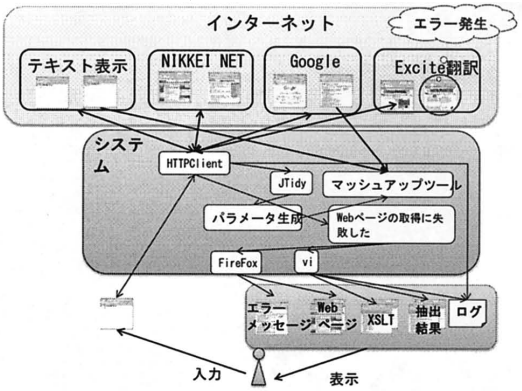

図 18 3アプリケーションデバッグ時 2

7.3. 今後の課題

現状では、デバッグの方法は Web ページの変換・抽出時に抽出結果が正しいかどうかのチェックを入れ、想定した結果でない場合は、該当する XSLT をエディタに表示するのみである。これを、GUI を用いるなど、より簡単にデバッグを行えるようなものにしていきたい。

アプリケーション連携の記述は、従来のテスト手法と同じテストコードを記述する形となっている。設定ファイルとしてスクリプトを別に用意し、そこに記述する方法に改善したい。

このシステムでは、使用している HTTP Client の仕様により、JavaScript および Flash には対応していない。Ajax などの動きのある Web ページも扱えるよう改良していく必要がある。

実験では、Web サービスを直接連携させずに

マッシュアップツールを間にはさんで連携させた。そのために、パラメータの変更があった場合には、マッシュアップツールの設定ファイルを手動で直すという手間ができてしまった。マッシュアップツールとこのシステムのパラメータの生成を連動させるような機能を付加し、Web サービスを連携させる場合のデバッグを簡単に行えるようにしなければならない。また、繰り返し処理などの複雑な連携にも対応していく必要がある。

8. 関連技術

アプリケーション連携の関連研究として、以下があげられる。これらは、マッシュアップ時にログが残るのみであり、デバッグを行うことができない。

- Yahoo! Pipes[7]
  Yahoo! が提供している GUI ベースのマッシュアップツールである。RSS、XML、JSON を対象にしており、HTML を扱うことはできない。「並び替え」や「置換」といった加工をパイプのようにつなぎ合わせて新たな RSS を生成する。
- Microsoft Popfly[8]
  Microsoft が提供している GUI ベースのマッシュアップツールである。Yahoo! Pipes と同じように、「並び替え」や「置換」といった加工のほか、Microsoft や Google などの Web サービスをつなぎ合わせてコンテンツを生成する。

デバッグの関連研究として、以下があげられる。

- JUnit[9]
  Java ベースの単体テスト用のフレームワークである。テストコードを作成して利用するため、仕様変更のたびにテストコードを作成しなおす必要がある。
- Selenium[10]
  Web ブラウザを使って Web アプリケーションをテストするツールである。HTML の表をテストコードに用いる。ページ遷移のテストに向いているが、統合テストには向いていない。

9. おわりに

以上のように、いくつかのアプリケーションに対して、アプリケーションの連携とデバッグを簡単に行う事ができた。これにより、頻繁に変更が行われるアプリケーションに対して、直接プログラムを書く場合に比べて柔軟に対応できる。

今後の課題として、スクリプトでのアプリケーション連携の記述や、JavaScript や Flash の対応があげられる。特に、JavaScript に対応することで、Ajax を利用しているアプリケーションを利用することが可能となる。対応するコンテンツの種類を増やして利用可能な Web アプリケーションの数を増やし、複雑なアプリケーションの連携を行えるようにすることで。汎用性が増すと考えられる。

なお、システムに利用した情報には著作権があるため、研究目的でのみ使用し、公開はしていない。

参考文献

[1] 山本, 小山, 杉田, 塚本, マッシュアップツールの開発とそれを用いた東京都防災マップの構築, 第 69 回全国大会論文, 2007
[2] HTTP Client, http://hc.apache.org/httpclient-3.x/
[3] JTidy, http://jtidy.sourceforge.net/
[4] Google, http://www.google.co.jp/
[5] Excite 翻訳, http://www.excite.co.jp/world/
[6] NIKKEI NET, http://www.nikkei.co.jp/
[7] Yahoo! Pipes, http://pipes.yahoo.com/
[8] Microsoft Popfly, http://www.popfly.com/
[9] JUnit, http://www.junit.org/
[10] Selenium, http://seleniumhq.org/