

2層構造による Web アプリケーションの Web サービス変換

高 橋 健 一^{†1} 紫 合 治^{†2}

Web アプリケーションを再利用し Web サービスとして公開することは、新しい Web サービスの開発コストの削減に大変有効である。本論文では、ページ遷移をベースとした、セッション情報の伝播による Web アプリケーションの Web サービス変換方式を提案する。ページ遷移単位で変換された Web サービスをビジネスプロセス記述言語 (BPEL) でつなぐことにより、Web アプリケーションの提供するユースケース単位での Web サービス公開と、拡張や変更柔軟に対応可能な Web サービス変換を実現した。さらに、本アーキテクチャを実装した H2W-Runtime と、開発環境である H2W-IDE を開発した。また、実際に運用されている 4 種の Web アプリケーションに対して適用し、Web サービスを自動生成することができた。

Converting Web Application to Web Service by Two Layers

KENICHI TAKAHASHI^{†1} and OSAMU SHIGO^{†2}

Developing Web Services from scratch is costly task. Converting Web Applications to Web Services are considered one of the most effective methods for Developing Web Services. This paper describes a framework named H2W that can be used for constructing Web Service Wrappers from existing Web Applications. We propose a page-transition-based decomposition model and page access model with context propagation. We have successfully developed Web Service wrapper applications for four real world Web Applications.

1. はじめに

近年、サービス指向アーキテクチャ (SOA) の実現技術として Web サービスがさかんに研究され、実用段階を迎えつつある。Web サービスは、XML ベースの標準プロトコルを用いることで高い相互運用性を実現することを目的としている。しかしながら、Web サービスの開発に必要な知識は多く、導入には多くのコストがかかることが問題とされている。

一方、システム開発コストの削減の手段として、既存システムの再利用が注目されている。運用中のシステムのアプリケーションロジックを再利用することで、実装やテストにかかるコストを削減することができる。

そこで我々は、現在様々なシステムが Web アプリケーションとして開発・利用されていることに注目し、既存 Web アプリケーションの再利用によって Web サービスを開発することで、Web サービスの開発コストを抑えることができると考えた¹⁾。

Web マイニングの分野では、Web アプリケーションの出力する情報を再利用するために、HTML から重要なデータを抽出する Web ページラッピングと呼ばれる技術がさかんに研究され、多くの成果をあげている。Kuhlins ら²⁾ は、テキストマイニングの調査と、LAPIS による Web ページラッピング生成の提案を行っている。Wang ら³⁾ は、ラッピングの生成法として、あるスキーマに従って動的に生成される複数の WEB ページからそのスキーマを導出し、適切なデータを抽出する方式を提案している。Gupta ら⁴⁾ は、非有用コンテンツを削除することで、結果的に有用なコンテンツを抽出するためのラッピング生成法を提案している。また、Leander ら⁵⁾ は、Web 上のコンテンツからデータを抽出するツールを調査し、6 種類に分類・評価を行った。上記のような Web ページラッピングの技術を用いることで、HTML から重要な情報を抽出することは十分に可能となっている。

また、Noga ら⁶⁾ は、単一のリクエストによって得られた Web ページからデータを抽出し Web サービスで提供可能な方式を、藤原⁷⁾ は、複数の Web アプリケーションと Web サービスの出力を合成し、1 つの Web サービスの出力へと変換する方式を提案している。

^{†1} 株式会社永和システムマネジメント
Eiwa System Management, Inc.

^{†2} 東京電機大学情報環境学部
School of Information Environment, Tokyo Denki University

我々は、複数の Web ページの移動を含むような Web アプリケーションから、必要な情報を出力する Web サービスを作成するためのアーキテクチャを精練することに焦点を当て、Web ページラッパを応用した Web サービス変換アーキテクチャの提案と、実装したフレームワークの開発を行った。

本論文では、まず 2 章で Web アプリケーションの特性に関する分析を行う。次に 3 章で汎用的な Web サービス変換アーキテクチャを示し、4 章で我々が開発したフレームワークについて述べる。5 章では、実際に運用されている 4 つの Web アプリケーションに対して行った Web サービス変換実験と実験による評価を、6 章で今後の課題を、最後に 7 章でまとめを述べる。

2. Web アプリケーションについて

Web アプリケーションとは、利用者が Web ブラウザを用いて Web サーバと通信を行う対話型のネットワークアプリケーションである。本章では、図 1 のようなページ構成のショッピングアプリケーションを例とし、Web アプリケーションの特性について分析を行う。

2.1 ページ遷移

Web アプリケーションを利用するユーザから見た場合、アプリケーションを構成する最小単位は「Web ページの移動」である。これは、ユーザが行う「リンクのクリック」や「ボタンを押す」という操作によってサーバに「リクエスト」が送られ、サーバが新たなページを「レスポンス」として返却する一連の流れによって実現されている。

本論文では、この Web ページの移動を「ページ遷移」と呼ぶこととする。図 1 での各矢印が「ページ遷移」に対応する。

2.2 Web アプリケーションの提供するサービス

Web アプリケーションのユーザは「ページ遷移」を繰り返しながら、Web アプリケーションの提供する

機能を利用する。図 1 の例では、ログインページからトップページへ、カタログページから商品を選択してショッピングカートへ追加、最後に注文確定画面へといったページ遷移の組合せで「商品購入」機能が利用できる。このような、ある決まった「ページ遷移」の組合せによって提供される機能を、Web アプリケーションの「サービス」と呼ぶこととする。

1 つの Web アプリケーションが、複数のサービスを提供することは明らかである。図 1 のようなシンプルなアプリケーションであっても、前述の「商品購入」というサービスのほかに、商品情報ページまでのフローによって「商品検索」というサービスが提供されていると考えることができる。

2.3 セッション情報

現実レベルの Web アプリケーションでは、アクセスコントロールの仕組みが不可欠である。図 1 のようなアプリケーションでは、初めにログインしなければ他のページにアクセスできないように設計する必要がある。しかしながら、Web アプリケーションを利用するためのプロトコルである HTTP プロトコルは、ステートレスなプロトコルであり、状態を保存することができない。

このような課題を解決するために、Web アプリケーションでは Cookie を用いた「セッション」という概念を導入している。Web サーバは、初回アクセスの際にクライアントに一意の ID を振ることで、以降のリクエストがどのクライアントから送信されたものなのかを識別することが可能となる。

ここで、セッションという概念を「次のページに遷移するために必要な情報」と考え、セッション情報という概念を導入した。ページ遷移に必要な情報として考えられるのは、まず URL (Uniform Resource Locator) である。また、動的なパラメータを送信するために利用される HTML の FORM 要素には、ユーザの入力に関係なく送信される hidden 要素がある。ユーザの入力に関係なく送信される情報はセッション情報という意味合いが強いと考え、URL、hidden パラメータ、Cookie の 3 つの情報をセッション情報の要素と定義した。

3. Web サービス変換アーキテクチャの提案

本章では、Web アプリケーションと Web サービスの違いについて述べ、それを吸収するための Web サービス変換アーキテクチャの提案を行う。

3.1 Web アプリケーションと Web サービス

Web サービスと Web アプリケーションを比較した

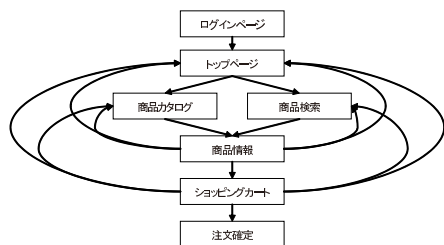


図 1 ショッピングアプリケーションのページフロー例
Fig.1 Possible navigation of a web application.

表 1 Web アプリケーションと Web サービスの比較

Table 1 Comparison of web application and web service.

	Web アプリケーション	Web サービス
プロトコル	HTTP	SOAP
ドキュメント	HTML	XML
利用者	人間	システム
システム特性	対話型	バッチ的

際の違いを表 1 に示す．ここで，特に重要なのは「システム特性」の違いである．

Web アプリケーションは，複数回のページ遷移により結果が得られれば満足である．初めに住所や名前，商品名，個数などすべての情報を入力させるショッピングサイトがないように，複数の画面で必要な情報をそのつど入力させるインタラクティブなシステムであるといえる．一方で，利用者が他システムである Web サービスでは，「1 回の処理で結果を取得できる」ということが重要であり，バッチ的なシステムであるといえる．

この違いを吸収するためには，1 回の SOAP リクエストに含まれるパラメータを複数の HTTP リクエストの適切な適切な箇所へ挿入し，HTTP レスポンスとして返却される複数の HTML に横断的に存在する SOAP レスポンスとして返却すべき情報を抽出・合成することが必要となる．

つまり，Web アプリケーションを Web サービスへと変換するためには，プロトコルとドキュメントの相互変換機能と，システムの違いを吸収するためのワークフロー処理機能を備えた変換システムが必要となる．

3.2 変換プロキシによる Web サービス変換

まず我々は，プロトコルとドキュメントの違いに注目した．Web サービスは SOAP プロトコルを使用して XML ドキュメントを提供し，Web アプリケーションは HTTP プロトコルを使用して HTML ドキュメントを提供する．これらを相互に変換するプロキシソフトウェアを導入することで，WEB アプリケーションの Web サービス変換が実現可能であると考えた．(図 2)

この方式は，非常にシンプルな Web サービス変換，1 回の SOAP リクエストが 1 回の HTTP リクエストに相当する場合には十分適用可能であると考えられる．

しかしながら，1 回の SOAP リクエストが複数回の HTTP リクエストに対応する場合は非常に複雑になる．プロキシ側で複数の HTTP リクエストへの分解処理を行えば，変換プロキシの処理が煩雑なものになり，サービスクライアントが送信する SOAP リクエストに分解処理の規定を記述すれば，SOAP メッセージ

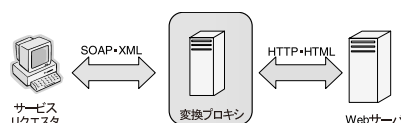


図 2 変換プロキシによる Web サービス変換

Fig. 2 Converting web application to web service using a proxy.

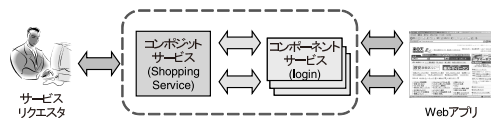


図 3 2 層構造による Web サービス変換

Fig. 3 An example of shopping web service by two layers.

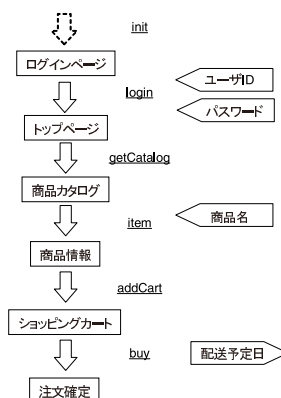


図 4 ショッピングアプリケーションの画面遷移と入出力パラメータ

Fig. 4 An example navigation of shopping web application.

ジは煩雑なものとなる．

3.3 2 層構造による変換アーキテクチャ

3.2 節の課題を解決するため，ページ遷移に対応する「コンポーネントサービス」と，Web アプリケーションのサービス単位に，複数のコンポーネントサービスを組み合わせた「コンジョイントサービス」の 2 種類の Web サービスによって変換するアーキテクチャを提案する (図 3)．以下でそれぞれの詳細について述べる．

3.4 コンポーネントサービス

コンポーネントサービスは，Web アプリケーションのページ遷移に対応する Web サービスである．図 1 のショッピングアプリケーションの例では，ログインページからトップページへのページ遷移が，login のような 1 つのコンポーネントサービスとなる．

図 1 の「商品購入」を行うまでのページ遷移とコンポーネントサービスの関係を図 4 に示す．図中の下線が引いてある部分が，コンポーネントサービスの名

前となる．また，コンポーネントサービスには，Web ページの遷移と同様に，入力パラメータと返却値が存在する．入力パラメータには，login サービスに対してのユーザ ID やパスワードなどが相当する．返却値の例としては，注文確定画面へと遷移する buy サービスにおける，配送予定日が相当する．

3.5 コンポジットサービス

コンポジットサービスは，複数のコンポーネントサービスを Web アプリケーションの提供するサービス単位にまとめたものである．図 4 の例では，init, login, getCatalog, item, addCart, buy というフローを Shopping サービスのようなコンポジットサービスとして定義する．

コンポジットサービスは，フロー内にあるすべてのコンポーネントサービスに必要なパラメータを，入力パラメータとして受け取る．図 5 の Shopping サービスでは，ユーザ ID とパスワード，商品名が入力パラメータとなる．コンポジットサービスは，各パラメータが必要となるコンポーネントサービス呼び出す際に，適切なパラメータを選択し，送信を行う．また，各コンポーネントサービスの返却値（一般には最後のサービスの返却値）をまとめ，コンポジットサービスの返却値とする．これにより，Web アプリケーションの提供する機能単位での Web サービスが提供可能となる．

3.6 セッション情報の伝播

各ページ遷移をコンポーネントサービスという独立したモジュールへと分解することにより，2.3 節で述べたセッション情報の伝播が課題となる．この課題を解決するため，セッション情報を 2 種類に分類し，それぞれの抽出・伝播方式について述べる．

1 種類目は，Web ページ（HTML）上に存在するものである．これは，URL とフォームのパラメータが該当するが，HTML 上にある情報は，すべて Web ページラッピングを応用することで抽出が可能である．

2 種類目は，Web ページ上にはなく，Web ブラウザが完全に隠蔽して処理しているタイプである．これは，Cookie が該当する．Cookie は HTTP プロトコルのヘッダに記述されており，プロトコルを解析することで抽出が可能となる．

したがって，HTTP プロトコルの解析+Web ページラッピングによってすべてのセッション情報を抽出することが可能である．

セッション情報を実際に送受信する必要があるのはコンポーネントサービスである．そこで，すべてのコンポーネントサービスは，固有の入力パラメータとは別に，セッション情報を入力として受け取ることとした．同様に，すべてのコンポーネントサービスはセッション情報を返却値とする．そして，コンポーネントサービスの順序関係を管理するコンポジットサービスが，コンポーネントサービスにおけるセッション情報を次のコンポーネントサービスへ送信することで，セッション情報の伝播が実現できると考えた．

4. Web サービス変換システム：H2W

我々は，3 章で提案した Web サービス変換アーキテクチャを実装した「H2W」を開発した．H2W プロジェクトの全体像を図 6 に示す．

H2W はコンポーネントサービスの実行環境である「H2W-Runtime」と，コンポーネントサービスの生成を行う「H2W-IDE」から構成される．ここで，我々はコンポジットサービスの実装として，Business Process Execution Language for Web Services (BPEL4WS) が利用できると考えた．BPEL4WS では，複数の Web サービスを順序付けした 1 つのビジネスプロセスとして定義することができ，各 Web サービスへのパラメータの送信，レスポンスの合成，レスポンスを利用したリクエストの送信など，3.2 節で述べたコンポジットサービスに必要な機能を網羅している．また，WebSphere や WebLogic では BPEL4WS の開発環

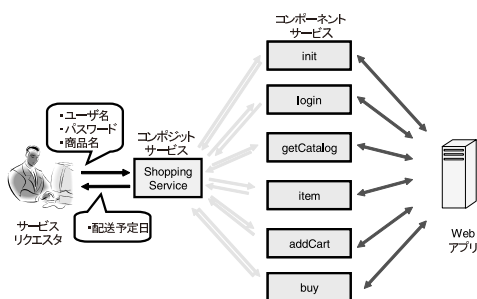


図 5 2 層構造によるショッピングサービスのイメージ
Fig. 5 Conversion by two layer structure.

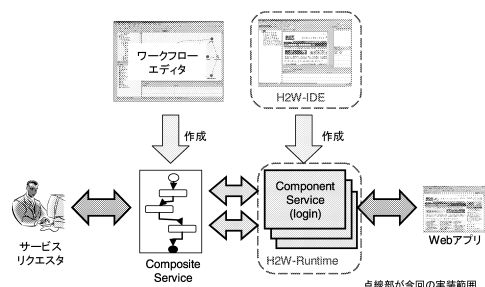


図 6 H2W プロジェクト全体像
Fig. 6 The whole H2W-Project image.

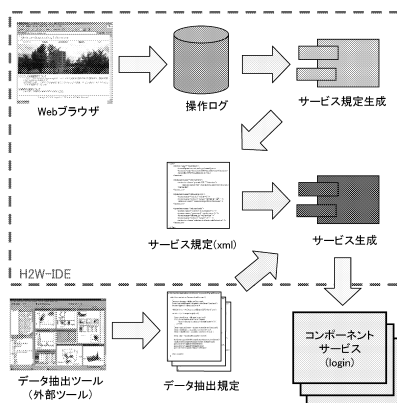


図 8 H2W-IDE の内部構成

Fig. 8 The structure of H2W-IDE.

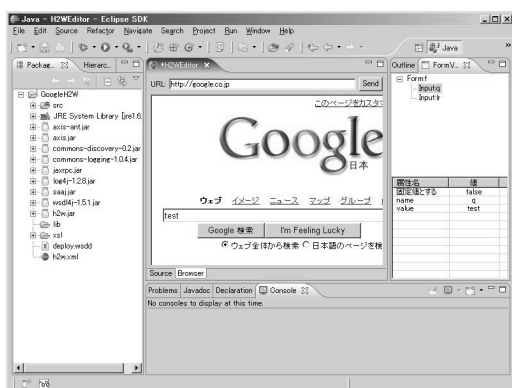


図 9 H2W-IDE スクリーンショット

Fig. 9 A screen shot of H2W-IDE.

4.2.1 サービス規定生成処理

本システムのスクリーンショットを図 9 に示す．中央の Web ブラウザに HTML が表示されると同時に，右のフォーム情報表示部に HTML 内の入力フォームの情報が表示される．利用者が Web ブラウザ上のある入力フォームにフォーカスを当てるとフォーム情報表示部が更新され，現在入力中のフォームの情報を確認できる．入力フォームの情報は，そのままコンポーネントサービスの入力パラメータとなるが，除外したいパラメータや，静的に固定したいパラメータをチェックすることも可能である．Web アプリケーションを最後まで操作した段階で，サービス規定生成処理を呼び出し，操作ログからサービス規定を生成する．

4.2.2 サービス規定仕様

サービス規定の例を図 10 に示す．

サービス規定には，変換する Web アプリケーション全体の情報を記述する service 要素と，1 つ 1 つのページ遷移（コンポーネントサービス）に対応するオ

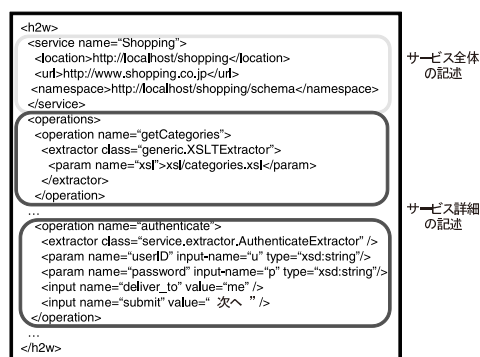


図 10 サービス規定例

Fig. 10 An example of component service rule.

ペレーション情報を記述する operation 要素の 2 種類を記述する．サービス全体の情報は，service 要素の name 属性としてサービス名を，service 要素の子要素 namespace の値としてサービスのネームスペースを，同様に子要素 location にサービスを公開する際の位置情報を，子要素 url に変換対象である Web アプリケーションの URL を記述する．

オペレーション情報では，まずオペレーション名（コンポーネントサービス名）が operation 要素の name 属性に記述される．operation 要素の子要素として，input 要素，param 要素，extractor 要素があるが，input 要素と param 要素はいずれも HTML のフォームへ入力するパラメータの情報である．input 要素として記述されたパラメータはつねに同じ値が Web サーバに送信される．param 要素として記述されたパラメータはサービス呼び出すクライアントが送信すべきパラメータである．また，extractor 要素ではコンテナ抽出プラグインのモジュール名を指定する．

4.2.3 サービス生成処理

サービス規定生成処理によって作成されたサービス規定をもとに，コンポーネントサービスを作成するサービス生成処理の流れについて述べる．

サービス生成処理では，まずサービス規定に含まれる入力パラメータをもとに，Web サービスのインタフェース記述である WSDL を生成する．次に，WSDL のオペレーションとデータ抽出規定の関連付けを行い，コンポーネントサービスとして動作する Java のソースコード（図 11）を出力する．最後に，デプロイ可能な形式へのパッケージングを行う．

4.3 BPEL4WS によるコンポジットサービス

BPEL4WS を用いたコンポジットサービスのインタフェースは，リクエストパラメータは自身が呼び出すコンポーネントサービスの入力パラメータがすべて

```

public class Shopping {
    private static final String URI = "http://www.shopping.co.jp/";

    public Shopping() {
    }
    public ApplicationSession init() {
        ApplicationSession session = new ApplicationSession(URI);
        ApplicationSession.GET_METHOD, new Cookie(0), new HashMap<String, String>());
        return session;
    }

    public HTTPData[] getCategory(ApplicationSession session) {
        final framework.core.extractor.XSLTPayloadExtractor payloadExtractor
            = new framework.core.extractor.XSLTPayloadExtractor();
        payloadExtractor.set(XSLTFile("xslt/category.xsl"));
        return new ServiceStub().invoke(session, inputs, payloadExtractor);
    }
    ...
    public HTTPData[] authenticate(ApplicationSession session, String userID,
        String password, String deliver_to, String submit) {
        final service.extractor.AuthenticateExtractor payloadExtractor
            = new service.extractor.AuthenticateExtractor();
        final Map<String, String> inputs = new HashMap<String, String>();
        inputs.put("u", userID);
        inputs.put("p", password);
        return new ServiceStub().invoke(session, inputs, payloadExtractor);
    }
    ...
}

```

図 11 生成されたコンポーネントサービスの実装コード例

Fig. 11 An example of generated Java code for component service.

列挙されたものになり、レスポンスは各コンポーネントサービスのレスポンスを合成したものとなる。また、コンポーネントサービス呼び出す際には、直前に呼び出したコンポーネントサービスのレスポンスに含まれるセッション情報を、そのまま次のコンポーネントサービスの呼び出しパラメータに追加する。

5. 適用実験と評価

我々は、典型的な Web アプリケーションとして、ショッピング・コミュニティ・検索の 3 つを上げ、それらに東京電機大学で利用されている履修管理システムを加えた 4 つの Web アプリケーションに対し、H2W フレームワークを適用し Web サービスへと変換することに成功した。

ショッピングサイト「楽天」からは、ワイン購入サービスを、履修管理システム「ダイナミックシラバス」からは学生ごとの時間割とシラバスの合成サービスを、OSS コミュニティサイト「Sourceforge.net」からはコミュニティを指定してのバグ検索サービスを作成した。本章では、まず検索サイト「Google」を利用した「日本語ページ検索サービス」を作成する手順について述べる。次に、実験結果をふまえた本変換アーキテクチャと H2W プロジェクトの評価を行う。

5.1 Google を用いた Web サービスの作成

本プロジェクトを用いた Web サービス変換の手順は、まず H2W-IDE を用いてコンポーネントサービスを作成し公開する。次にコンポーネントサービスを作成するが、今回は BPEL4WS ではなく環境の整備が容易な Java プログラムによって実装した。最後に、コンポジットサービスを公開する。

以下に、手順の詳細を述べる。

- (1) コンポーネントサービスを作成するプロジェクト「GoogleH2W」を作成する。
- (2) サービス規定の雛形である「H2W Configuration」の作成を行うためのウィザードを開く。
- (3) サービスの名前「Google」と、Web アプリケーション URL「http://google.co.jp」と入力する。
- (4) サービス規定「h2w.xml」を開くことで専用のブラウザが開き、2 で入力した URL の Web ページが表示される。
- (5) 検索ワードを入力し、submit ボタンを押すことで、入力パラメータがサービス規定に自動的に追加される。
- (6) ページ遷移の際に現れるダイアログに、このページ遷移を行うオペレーション名を入力する。
- (7) 外部ツールによって作成した、Web ページラップの記述を「h2w.xml」に加える。
- (8) h2w.xml と Web ページラップの記述から war ファイルを生成する。
- (9) war ファイルアプリケーションサーバへデプロイする。
- (10) 次に、コンポーネントサービスの WSDL を Apache Axis のスタブ生成ツールで読み込み、スタブを利用したコンポジットサービスを Java 言語で作成する。
- (11) コンポーネントサービスのデプロイを行うことで「日本語ページ検索サービス」が公開できる。

以上の操作により、以下の 2 つのコンポーネントサービス init (トップページへの移動を行う)、searchJP (日本語での検索を行い、結果を取得する) を作成し、公開することができた。

また、コンポジットサービスとして検索したい語句を入力パラメータとし、日本語での検索結果を取得する「JPSearch」サービスを作成した。

5.2 評価

今回の適用実験による H2W-Runtime および、H2W-IDE の評価について述べる。

5.2.1 H2W-IDE による作業量の変化

今回の実験は、開発者自身が行ったためツールの評価は主観的なものである。

まず、作成時間に関して、H2W-IDE を使わずにコンポーネントサービスを策した場合が 4 つのアプリケーションで平均 8 時間、H2W-IDE を使用した場合で平均 3 時間となった。また、記述コード量は、ツールなしが 300 ステップのコードを記述したのに対して、H2W-IDE ではすべてのコードを自動生成するこ

表 2 H2W-IDE による作業量の比較

Table 2 Comparison of web service conversion costs by H2W-IDE.

	ツールなし	H2W-IDE
作業時間 (時間)	8	3
記述コード量 (ステップ)	300	0

表 3 Google 検索サービスの呼び出し時間分析

Table 3 Comparison of web service performance.

	Google Web API	H2W
平均 (ms)	2,708	1,570
最小 (ms)	571	831
最大 (ms)	16,073	7,892

とができた (表 2)。

以上の結果から、H2W-IDE はコンポーネントサービスの生成に関して十分な作業量の軽減が実現できたと考える。また、利用者が行う作業はすべてブラウザとウィザードを用いたものであり、HTML や Web サービスに関する専門知識を必要としないことも確認した。

5.2.2 H2W-Runtime の適用範囲とパフォーマンス

今回行った 4 つの Web アプリケーションの Web サービス変換はすべて実現できたため、本方式は現実レベルの Web アプリケーション変換に十分有効であることが確認できた。実際の規模である楽天の例では、ページ遷移が 8 回あり、セッション管理はすべて想定していたとおり直前のレスポンスを次のリクエストに追加する方式で簡単に行うことができた。

次に、適用実験の際に作成した Google を用いた検索サービスと、Google の提供する Google Web API との検索時間測定実験のデータを表 3 に示す。H2W を用いた変換サービスについては、コンポジットサービス、コンポーネントサービスともにローカルホスト上で動作させている。サービスクライアントはどちらも Java 言語で作成し、プログラムの開始から検索結果の取得までの時間を測定した。連続して 10 回の検索を行い、それを 10 回繰り返すことで 100 回分のデータを計測した。

今回の適用実験により、本方式がパフォーマンスの面からも十分実用的であることが確認できた。実測結果では、応答時間の大半が Web アプリケーションの応答時間になっており、Web の応答時間が十分に速い場合でも H2W-Runtime の処理時間は全体の 10% 未満であった。本方式は、応答速度が優れているアプリケーションに対しても、その速度を損なわないで Web サービスを提供できる方法であるといえる。また、双方

ともに最小と最大で大きな差が生じているが、Google Web API ではネットワークの影響が大きいと考えられ、H2W に関しては初回アクセス時が最大時間になっており、クラスのロード時間が含まれていると考えられる。

6. 今後の課題

6.1 ページ遷移のないアプリケーションへの適用

近年、AJAX (Asynchronous JavaScript + XML) を代表とする HTML を動的に書き換えることで、ユーザビリティを向上させることができる技術が注目を集めている。このような Web アプリケーションは、バックグラウンドで非同期にサーバと通信を行い、表示されている HTML を直接書き換えているため、ページが書き換わるタイミングを検知することも、通信内容をハンドリングすることも難しい。これについては、Web アプリケーションのテスト・開発手法に関する研究^{8),9)}の成果を活用して Ajax アプリケーションのページの書き換えのタイミングが検知する方式を Web サービス変換に適用すること等を検討している。

6.2 UI フレームワークへの適用

現在の Web アプリケーション開発では、様々なフレームワークを利用することが一般的となっている。Java 言語での Web アプリケーション開発では、UI フレームワークとして Apache Struts や JSF (JavaServer Faces) が用いられることが多い。これらのフレームワークは、ページナビゲーションを設定ファイルに記述したり、命名規約としたりすることで Web アプリケーションの開発効率の向上に大きな効果を与えている。

我々の提案する方式は、ページナビゲーションと入力フォームをベースとしているため、UI フレームワークの設定ファイルや命名規約から、コンポーネントサービスの自動生成が可能になる可能性がある。

これが実現されれば、Web アプリケーションの開発がそのまま Web サービスの開発へとつながる。今後、UI フレームワークの調査と、自動生成の可能性について十分に検討する必要がある。

7. おわりに

本論文では、Web アプリケーションの Web サービス変換を実現するため、まず双方の技術的な違いの分析を行い「インタラクティブなシステムとバッチ的なシステム」という Web サービス変換における本質的な課題を明らかにした。

そして、その課題の解決法として、Web アプリケー

ションのページ遷移単位の変換と、それを Web アプリケーションの提供するサービス単位までまとめた粒度の大きなサービスを組み合わせるといふ「2 層構造による Web サービス変換アーキテクチャ」を提案し、実装を行った。

また、提案方式とその支援ツールの有効性を検証するため、現実レベルの 4 つの Web アプリケーションへの適用を行い、次の 3 点について十分有効であることが確認できた。まず、適用実験を行った性質の異なる 4 種の Web アプリケーションすべてに適用可能であったことから、アーキテクチャの有効性が確認でき、Google を用いた実験の結果より、実行時のパフォーマンスに関しても十分有効であるという結果が得られた。また、支援ツールによって作業時間とコーディング量が十分な軽減されたことも確認でき、ツールの有効性も確認された。

本方式は、Web アプリケーションの変更をいっさい必要としないことにより、Web アプリケーションの実装するビジネスロジック、セキュリティ機能をそのまま再利用することができるうえ、支援ツールによりプログラミングレスでの Web サービス開発が可能であり、Web サービス開発コスト削減に大きな効果があることが実証できた。

謝辞 H2W プロジェクトの開発に協力いただいた立堀道昭氏始め日本アイ・ピー・エム株式会社東京基礎研究所の皆様にご感謝の意を表する。

参 考 文 献

- 1) 高橋健一、紫合 治：Web アプリケーションの Web サービス変換，ソフトウェアエンジニアリング最前線 2006，pp.193-200 (2006)。
- 2) Kuhlins, S. and Tredwell, R.: Toolkits for Generating Wrappers, *Objects, Components, Architectures, Services, and Applications for a Networked World 2002*, No.1 (2002)。
- 3) Wang, J. and Lochovsky, F.H.: Data Extraction and Label Assignment for Web Databases, *ACM 2003* (2003)。
- 4) Gupta, S., Kaiser, G.E., Neistadt, D. and Grimm, P.: DOM-based Content Extraction of HTML Documents, *WWW 2003*, pp.207-214 (2003)。
- 5) Leander, A.H., Ribeiro-Neto, B.A., da Silva, A.S. and Teixeira, J.S.: A Brief Survey of Web Data Extraction Tools, *SIGMOD Record*, pp.84-93 (2002)。
- 6) Noga, M.L. and Volkel, M.: From Web Pages to Web Service with wal, *NCWS2003* (2003)。
- 7) 藤原克哉，中所武司，玉本英夫：Web サービス統合による自動記入エージェントの実現方式，情報処理学会論文誌，Vol.47, No.2 (2006)。
- 8) 立石考彰，宮下 尚，小野康一，斉藤 新：ページフローに着目した DHTML の自動検証，ソフトウェアエンジニアリング最前線 2006，pp.201-208，近代科学社 (2006)。
- 9) 福安直樹，満田成紀，鰐坂恒夫：DOM 操作による状態遷移に基づく Ajax アプリケーションの解析手法，ソフトウェアエンジニアリング最前線 2006，pp.209-216，近代科学社 (2006)。

(平成 19 年 3 月 27 日受付)

(平成 19 年 10 月 2 日採録)



高橋 健一 (正会員)

1982 年生。2007 年東京電機大学大学院情報環境学研究科情報環境工学専攻修士課程修了。同年 (株) 永和システムマネジメント入社。プログラミング言語 Ruby を用いたソフトウェア開発に従事。



紫合 治 (正会員)

1946 年生。1971 年東京電機大学電子工学科卒業。日本電気 (株) 入社後、中央研究所、ソフトウェア生産技術研究所、NEC アメリカ、インターネット技術研究所等で、ソフトウェア工学、インターネットセキュリティに関する研究開発やビジネス立上げに従事。2003 年より東京電機大学情報環境学部教授。情報処理学会 20 周年記念論文賞 (1980 年)、大河内記念技術賞 (1984 年) 受賞。日本ソフトウェア科学会会員。