社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－DBS－142 (6)
2007－ＦＩ－ 87 (6)
2007／5／31

マッシュアップを簡単に実現するメタCGIとそのアーキテクチャ

森 雅生[†] 中藤哲也[††] 廣川佐千男[††]

複数の異なるコンテンツやサービスから新たなウェブサービスを再構成して展開するマッシュアップが注目されている．これはXMLデータ取り扱いやPerl，PHP，JavaScriptなどの専門的な技術を駆使して実装される．本稿はウェブサービスの検索に重点をおいたマッシュアップをサーバサイドで簡単に実行できるアーキテクチャを提案する．

A meta-CGI for light-weight implementations of Mashups and its architecture

Masao Mori ,[†] Tetsuya Nakatoh [††] and Sachio Hirokawa[††]

Mashup is a remarkable concept that combines more than one content and services into an integrated web service. Its implementation is done with expertises, – Perl, PHP, JavaScript and XML – for instance. We propose a new architecture of a meta-CGI which enables us to construct Mashups on server side.

1. 序　　文

インターネットでアクセス可能な学術論文データベースやe-コマース商品情報サイト，交通機関の路線検索データベースなどはウェブをインターフェースに持つデータベースである．これらはディープウェブ[1]やウェブデータベースなどと呼ばれ，系統だった膨大な量の情報が内包されていることからその価値が注目されてきた．これまでのウェブデータベースはCGIを通してのみ内包されているデータベースにアクセスすることができないので，出力されるHTMLを解析してラッパーを作成する研究がなされて来ている[5)6)7)]．

一方，新しいコンテンツのありかたとして，データベースの所有者が情報を一般に積極的に公開し，共有しようと言う風潮が高まりつつある(Web2.0)．そこではSOAPやREST，XML-RPCなど公開されたプロトコルを用い，ウェブサービスで展開しやすいデータ形式(XMLやJSONなど)で情報がやりとりされている．この様な形態で情報を提供するサイトはウェブサービスと呼ばれている．上述のウェブデータベースもHTML解析をするラッパープログラムとセットにすれば広義のウェブサービスとみなすことができる．

複数のウェブサービスを連携させることをマッシュアップと呼ぶ．マッシュアップではブラウザ上でのログイン処理やウェブサービスのアカウントに依存した情報登録なども行われる．すなわちウェブサービスからクライアントへ情報を提示するだけでなく，相互にメッセージを交換する．よってマッシュアップは，クライアント側とウェブサービス側のそれぞれの処理を考慮しなければならない．本稿では，ウェブサービスが提供する情報の検索機能の連携に焦点をしぼり，サーバ側での処理のみで行うマッシュアップのアーキテクチャを提案する．

ウェブサービスでの検索は，一つのブラウザを一つのウィンドウで表示させ，検索結果から度々次のキーワードを選び出し，次のサイトのクエリボックスにコピーすると言った検索の定型的な反復作業が行われるであろう．本研究の動機は，これらの反復作業を簡易な方法で記述し，ウェブサービスを連携させて検索を実行する枠組と検索結果の効率的な表示方式を求めることであった．著者らは学術論文ウェブデータベース群のウェブラッパーの自動生成と機能合成の研究[9)]を行い、一般にXML形式をベースにしたウェブサービスとの連携も同様のアーキテクチャが有効であることを示した[3)4)]．

クライアント側でマッシュアップを実現するとき，AJAXによる実装が広く使われている．これにはJavaScriptに精通していなければならないが，マッシュアップのJavaScriptプログラムを簡易に記述するための仕組みについての研究がある[8)]．大量の検索結果を処理するため，連携を実現するスクリプトがサーバ側で実装される点で本稿の研究とはアプローチが異なる．

以下、本論文の構成は2章でアーキテクチャの概要を述べ，3章ではシステムの実装について解説する．本研究は検索に力点を置くマッシュアップであり，これに関する入出力制御の重要な特徴を4章で解説する．

2. アーキテクチャの概要

本稿で提案するアーキテクチャを *Personally Scripting*

† 九州大学大学評価情報室
Office for Information of University Evaluation
†† 九州大学情報基盤研究開発センター
Research Institute for Information Technology

図 1　PSM サイト

Meta-CGI search architecture (PSM) と呼ぶ．本章では PSM アーキテクチャの概要を実装したサーバの挙動を例にとって紹介する．PSM は単一のウェブサーバ (PSM サーバ) で実行される．PSM サーバにアクセスすると，テキストボックスなど入力インターフェースが表示され 3，ウェブサービス群を連携させるスクリプト (**マッシュアップ・スクリプト**) を記述することができる．CGI 生成のボタンが押されると PSM サーバは記述されたスクリプトに従って**マッシュアップ** *CGI* を生成し，PSM サーバ自身で実行する．マッシュアップ CGI は PSM サーバ内で保存されるので，一度作成したマッシュアップ CGI をいつでも実行できる．連携するウェブサービスは事前に PSM サーバにモジュールを定義しなければならない．モジュールの定義方法は 3 章で述べる．

ユーザは PSM サーバでマッシュアップ CGI を生成するとき，次の 3 つを知っていれば良い．

(1)　ウェブサービスの入出力情報の構造．
(2)　ウェブサービスの出力からの次クエリの選択．
(3)　検索結果からの表示したい項目の選択．

ウェブサービスで提供される詳細検索は，いくつかの属性に分けられたクエリの組合せを入力として受け付け，検索結果としていくつかの属性を組み合わせた構造を持つ情報が出力される．たとえば，音楽情報配信ウェブサービス *Rhapsody*☆を見てみる．「アーティスト名」「アルバム名」の属性の語句の組み合わせが検索クエリであり，「アーティスト名」「アルバム名」「曲名」などの属性が一つの組 (**レコード**) となり，レコードのリストが検索結果として出力される．また，Amazon.com のウェブサービスは「キーワード検索」と「製品番号検索」のいずれかを検索クエリに選ぶことができる．このように，ユーザは利用するウェブサービスがどの様な情報を提供しているかを知っておく必要がある (1)．このようなウェブサービスの入出力機構に備わっている属性を**チャネル**と呼ぶ．入力の属性を**クエリ・チャネル**，出力の属性を**リザルト・チャネル**と呼ぶことにする．

☆ `http://www.rhapsody.com/`

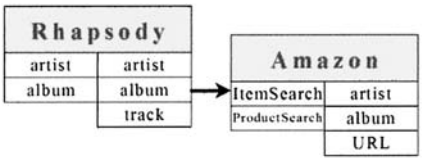

図 2　ウェブサービスと機能結合の模式図

ウェブサービス間のクエリ受渡しをウェブサービスの**機能結合**と呼び，その定義を記述したものを**マッシュアップ・スクリプト**(スクリプトと略) と呼ぶ (2)．各種ウェブサービスのチャネルの性質を踏まえた上で，ユーザはウェブサービス間で受け渡される情報のセマンティクスを認識しなければならない，すなわち意味のある機能結合をスクリプトに記述する．先程の Rhapsody と Amazon の例を挙げると，Rhapsody ウェブサービスから得られた結果のリザルト・チャネル「アルバム名」にある語句を，Amazon ウェブサービスのクエリ・チャネル「製品番号検索」へ渡すのはあまり現実的ではない．「キーワード検索」のクエリ・チャネルに渡すのが自然である．図 2 はこの 2 つのウェブサービスの模式図である．左右の矩形列がそれぞれクエリ・チャネル，リザルト・チャネルを表している．また，矢印は 2 つのウェブサービスの機能結合の模式図である．ウェブサービス Rhapsody はクエリ・チャネルに `artist` と `album`，リザルトチャネルに `artist`, `album`, `track` を持つ．ウェブサービス Amazon は CD 情報に特化している．クエリチャネルは `ItemSearch`(キーワード検索) と `ProductSearch`(製品番号検索) リザルト・チャネルが `artist`, `album`, `URL`(詳細情報 URL) をもつ．機能結合の定義はウェブサービスのクエリ・チャネルとリザルト・チャネルを矢印 `->`でつないで表記する．例えば，模式図 2 の様な機能結合を定義するスクリプトは次のようになる．

```
Rhapsody.album -> Amazon.ItemSearch,
```

クエリ・チャネル，リザルト・チャネル，それぞれの属性群の組を**クエリ・レコード**および**リザルト・レコード**，または単に**レコード**と呼ぶ．ウェブサービスから出力されるのはリザルト・レコードのリスト，すなわち『**表**』である．ブラウザで閲覧したいリザルト・チャネルを選びだし，次のようにスクリプトに記述すれば，選んだリザルト・レコードのリストが表示される．

```
Amazon.artist:album:URL -> Output2,
```

「Output」＋「番号」はウェブサービスの出力をブラウザに「表」として表示するための機構を表わし，「**出力コンポーネント**」と呼ぶことにする．上の例では Amazon ウェブサービスから (アーティスト名，アルバム名，詳細情報 URL) のレコードリストが表として表示する指定を意味する．出力コン

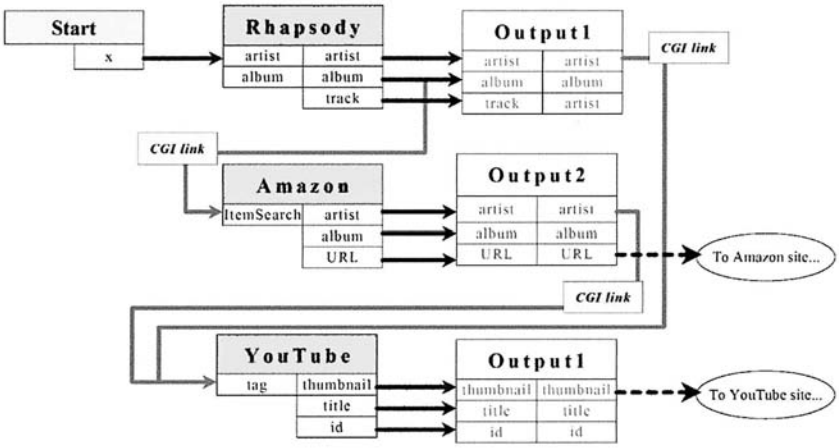

図 3　マッシュアップ・スクリプト模式図

ポーネントの詳細については次章で詳説する．

以上の 3 点をおさえればウェブサービスの機能結合をプログラムすることができるが，次の 2 つの機能を加える．

(4)　初期クエリの設定．

(5)　ブラウザ表示された語句をクエリとしてウェブサービスに渡すアンカータグの指定．

機能結合の**初期クエリ**を次の書式でスクリプトの冒頭に記述する (4)．

```
Start.x:y -> Rhapsody.artist:album,
```

スクリプトから生成される CGI が，初期クエリを入力するテキストボックスを表示する．上の例では 2 つのテキストボックス `x` と `y` が表示され，入力値がそれぞれ Rhapsody のクエリチャネル `artist` と `album` に渡される．

ブラウザに表示された検索結果の語句をウェブサービスへのクエリとするアンカータグをスクリプトで指定することができる．リンク先の主体はマッシュアップ CGI 自身であるが，当該の語句をクエリとして指定されたウェブサービスのクエリ・チャネルに渡すように制御される．スクリプトの記述は以下のようになる．

```
Rhapsody.artist:album:track -> Output1,
Output1.album -> Amazon.ItemSearch,
Amazon.artist:album:URL -> Output2,
```

まず，Output1 において Rhapsody からのレコード・リストが表示される．2 行目の `Output1.album` から `Amazon.ItemSearch` への機能結合は，Output1 に表示されたチャネル `album` にあたる文字列を Amazon のチャネル `ItemSearch` に渡すリンクを表示することを意味している．結果は Output2 に表示される．このリンク機能を *CGI* **リンク**と呼ぶことにする．

以上の 5 つのポイントに従い，スクリプトを作成すれば簡単にマッシュアップ CGI を生成することができる．実際にはウェブサービス，出力コンポーネントとの宣言も行うが，機能結合を記述するだけである．まとめると以下のような記述になる．このスクリプトの模式図を図 3 に示す．

```
use Rhapsody, Amazon, YouTube, Output1,
    Output2, Output3;
Start.x -> Rhapsody.artist,
Rhapsody.artist:album:track -> Output1,
Rhapsody.album -> Amazon.ItemSearch,
Amazon.artist:album:URL -> Output2,
Output1.artist* -> YouTube.tag,
Output2.artist* -> YouTube.tag,
YouTube.thumnail:title:id -> Output3
```

このスクリプトから生成されたマッシュアップ CGI を実行した様子を図 4 に示す．

3. PSM の実装

PSM サーバは，ウェブサービスを連携させるマッシュアップ CGI を生成させるメタ CGI として perl で実装されている．この章では前述の PSM アーキテクチャに従って PSM をどの様に実装しているかを解説する．

3.1 ウェブサービス・コンポーネント

ウェブサービスの API に合わせた通信関数を定義したモジュールを準備する．ウェブサービス毎にこのモジュールを作成しなければならないが，この章で述べる通信モデルに基づけば定型的に作成することができる．ウェブデータベースの場合は，ラッパープログラムもモジュールの中に組み込む．マッシュアップ CGI では，ウェブサービスのモジュールからそれぞれオブジェクトを生成する．この perl オブジェクトを**ウェブサービス・コンポーネント**，または単に**コンポーネント**と呼ぶことにする．

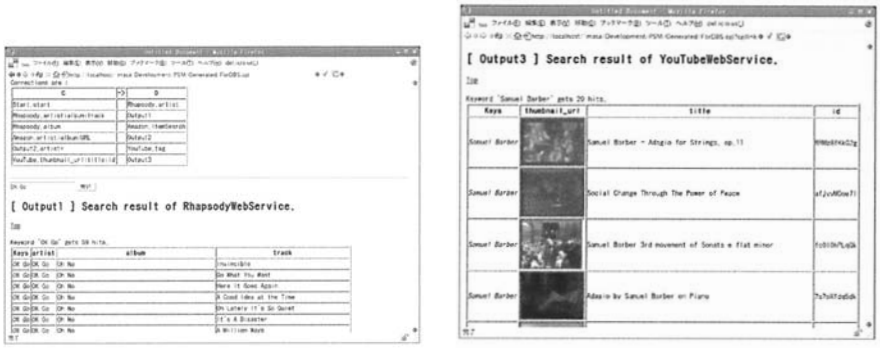

図 4　マッシュアップの実行図

3.2　機能結合の実装

コンポーネントはレコードを perl のハッシュに変換する．キーはチャネル名 (属性名) である．多くの場合，この変換は XML データから行われる．ウェブサービスの機能結合はコンポーネント間のレコード・リストの受渡しにより実現されるが，マッシュアップ CGI はコンポーネントからハッシュのリファレンスの配列を受け取り，結合が定義されているコンポーネントにこれを「そのまま」渡す．ここで，ウェブサービスへの入力クエリと検索結果の出力とのデータ構造について注意しなければならない．多くのウェブサービスの API では，一つのクエリ・レコードに対して複数のリザルト・レコード，すなわちレコード・リストが出力される．よって，そのまま渡したのでは「型」が合わない．PSM アーキテクチャでは，クエリとして渡すデータもレコード・リストに統一する．

3.3　入出力モデル

コンポーネントの入出力データ型をレコード・リストにすることにより，ブラウザへの出力先として記述していた出力コンポーネントも，入力に関してウェブサービス・コンポーネントと同等に取り扱うことができる．**出力コンポーネント**はクエリ・レコードのリストを渡されるとハッシュキーを属性名としたテーブルをブラウザに表示する．

```
Rhapsody.artist:album:track -> Output1,
```

この例では，artist,album,track が Rhapsody のリザルト・チャネルであるが，Output1 でも同じチャネル名で表示するので，出力コンポーネントのクエリ・チャネルは記述を省略する．出力コンポーネントのクエリおよびリザルト・チャネル名は，与えられるレコード・リストのチャネル名である．

出力コンポーネントから他のコンポーネントへの機能結合を CGI リンクと呼んだ．CGI リンクは，リンクの付いた文字列をクエリとしてコンポーネントに渡す処理を行う．クエリは，クエリ・レコードの単独要素リストで渡される．この文字列がいくつかのキーワードで構成され，コンマなどで区切られている場合がある．このときは Output1.aritst*のようにチャネル名の後にアスタリスクを付けることで，キーワード毎に CGI リンクを作成することができる．

3.4　フィルタ・コンポーネント

テーブルであるレコード・リストで行われる簡単な統計処理や文字列処理をコンポーネントで実装している．

ソート　一番目に与えられたクエリ・チャネルに関して並べ替えを行う．辞書式順序 (SortL) や数値順序 (SortN) など．入出力の型は出力コンポーネントと同じ．以下は track での並べ替えである．

```
Rhapsody.track:artist:album -> SortL,
```

ヒストグラム　一つの属性に着目して，そこに現れる文字列の出現回数をレコードごとに数え挙げるコンポーネント．クエリ・レコードは注目する項目 (item) 一つで，リザルト・レコードは，文字列 (who)，出現回数 (num)，その棒グラフ (hist) である．

```
Rhapsody.artist -> Hist.item,
Hist.who:num -> Output,
```

Uniq　レコード・リストに現れる重複を削除する．データ型は出力コンポーネントと同じ．

```
Rhapsody.artist:album -> Uniq,
Uniq.artist:album -> Output,
```

4.　まとめと課題

ウェブサービスがマッシュアップの材料として提供するデータ形式は XML である．これにより，マッシュアップや Web2.0 的なウェブサービスの高い拡張性を持たせることができる．しかし，本稿で論じたウェブサービスの機能結合におけるレコード・リストは，抽象的なデータ構造でありデータ形式を特定するものではない．提案した PSM アーキテクチャでは，多くのウェブサービスが提供する XML データをレコード・リストに変換する方式をとり，これによってマッシュアップというものを次の観点からとらえることができた．

- ウェブサービス，出力コンポーネント，フィルタなどのコンポーネントは全てレコード・リスト上の関数とみなすことができる．

- 機能結合はコンポーネントの関数合成である.

また，得られた結果から新規に検索する仕組みは，CGI リンクとしてあらかじめスクリプトに定義されるが，入出力の観点から見るとこの機能も上の 2 つの枠組でとらえることができる．この枠組に基づけばウェブサービスの機能合成 (マッシュアップ・スクリプト) は，コンポーネントの合成 (チャネルの対応) のみを記述すれば十分であることが分かる．

全てのデータをレコード・リスト形式に変換することで，以下の問題点が考えられ，今後の課題である．

(1) XML データの変換にかかる処理時間のオーバーヘッド.
(2) ウェブサービスごとに XML の解析がに必要である.
(3) レコード・リストでは重複データが頻繁に出現する.

この研究で実装した PSM サーバは
`http://hyoka-inf.ofc.kyushu-u.ac.jp/~mori/research/PSM/Main/`
で公開している.

参 考 文 献

1) BRIGHTPLANET, Deep Web, White Paper (2000).
2) HEMENWAY, K. and CALISHAIN, T. *Spidering Hacks*, O'Reilly & Associates Inc. (Mar. 2003), ISBN-13 978-0596005771.
3) MORI, M., NAKATOH, T. and HIROKAWA, S. Functional Composition of Web Databases, Proceedings of ICADL2006, Lecture note in Computer Science 4312, Springer Verlag (2006).
4) MORI, M., NAKATOH, T. and HIROKAWA, S. A Light-Weight Implementation of Mash-Ups, Proceedings of Data Engineering Workshop 2007IEICE (2007).
5) NAKATOH, T., OHMORI, K. and HIROKAWA, S. A Report on Metadata for Web Databases, IPSJ SIG Technical Reports, 2004-ICS-138(17) (2004).
6) NAKATOH, T., OHMORI, K., YAMADA, Y. and HIROKAWA, S. COMPLEX QUERY AND METADATA, Proceedings of ISEE2003 (2003).
7) NAKATOH, T., YAMADA, Y. and HIROKAWA, S. Automatic Generation of Deep Web Wrappers based on Discovery of Repetition, Proceedings of the First Asia Information Retrieval Symposium (AIRS 2004) (2004).
8) YOKOYAMA, S., MATONO, A., PAHLEVI, S.M. and KOJIMA, I. A Framework for Modularization and Mashup of JavaScript Codes on Web2.0, *DBSJ Letters*, **5**, 3 (December 2006).
9) 中藤哲也, 大森敬介, 廣川佐千男「検索エンジンを部品とするエージェントの構成について」, The Third Joint Agent Workshop & Symposium (JAWS2004) (2004).