九州大学学術情報リポジトリ
Kyushu University Institutional Repository

# マッシュアップの軽量実装のための提案

森，雅生
九州大学大学評価情報室

中藤，哲也
九州大学情報基盤センター

廣川，佐千男
九州大学情報基盤センター

https://hdl.handle.net/2324/3628

出版情報：2007-03
バージョン：
権利関係：

# マッシュアップの軽量実装のための提案

森　雅生† 中藤　哲也†† 廣川佐千男††

† 九州大学 大学評価情報室
†† 九州大学 情報基盤センター

**あらまし** 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 や JavaScript などの技術 で実 装されるが，専門的な知識が無くては自らのマッシュアッ プサイトを作ることが困難である．本稿では，著者らが開発を進めている PSM(Personally Scripting Meta-CGI search architecture) について述べる．PSM はウェブデータベースの機能合成の研究から始まったものであり，機能合成 を実装するための簡易スクリプト言語として実装中である．

**キーワード** ウェブサービス, ウェブデータベース, マッシュアップ, Web2.0

# A Light-Weight Implementation of Mash-Ups

Masao MORI†, Tetsuya NAKATOH††, and Sachio HIROKAWA††

† Office of Information for University Evaluation, Kyushu University. Fukuoka, Japan.
†† Communication and Computing Center, Kyushu University. Fukuoka, Japan

## 1. 序　文

近年ソーシャルブックマークサイトやネット資源ののタグ情報，商品の口コミ情報など，不特定多数のユーザから収集した情報がデータベース化されウェブサービスとして登場している．これらの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展開するマッシュアップが注目されている．この新しいウェブサービス群の特徴として，内包するデータベースにアクセスするための API が用意されており，これらを使ってプログラムを作成すればデータを一括で取得したり変更することが可能である．これらのウェブサービスやウェブデータベースの連携は AJAX や REST といった既存の技術の組み合わせで実現できるが，より少ない予備知識と手軽さでの連携機能の実現の研究が行われている [7].

著者らは入出力に構造をもつウェブデータベースの機能合成の研究を進めてきた [3]．この研究はウェブデータベースの統合 ( [10]) やオンデマンドでメタサーチを生成する ( [11], [8]) といった課題が動機であった．たとえば論文検索データベースは，著者名検索やタイトル検索など属性を持つクエリに対し検索結果の論文の著者名やタイトル，出版時期などのデータを構造をもつレコードのリストを出力として返す．すなわち，このデータベースは論文情報に関するレコードの入出力機械と見なすことができる．本稿は，このようなウェブデータベース群を機能結合することを目的とした PSM(Personally Scripting Meta-CGI search architecture) を提案する．このアーキテクチャはウェブサービスやウェブデータベースの結合に関する情報のみを記述するだけで連携をおこなうことができる．　実装中のウェブサービスである．ウェブデータベースにプログラムでアクセスする場合は出力される HTML を解析してウェブラッパーを作成しなければならなかった ( [4], [5], [6]) が，上述のようなウェブサービスは API が用意され，返されるのは XML 形式のデータであるので HTML の解析が不要である．本稿で提案される機能合成とは，異なるウェブデータベース間で同じ属性の出力と入力を渡すことを言う．機能合成されるウェブサービスおよびウェブデータベースは，API が用意されているものを対象とする．

## 2. アーキテクチャ

この研究で提案されるアーキテクチャ *PSM* (Personally Scripting Meta-CGI search architecture) の特徴は次の 4 点である．

（1） ウェブサービスや検索出力が向けられるブラウザなど，入出力が行われるすべての要素を**コンポーネント**と呼び，一様な対象と見なして入出力の**機能合成**(functional compsition) が定義される．

（2） コンポーネントにはウェブサービス・コンポーネント，出力コンポーネント，パイプ・コンポーネントの 3 種類がある．

（3） 出力コンポーネントの出力はブラウザからの入力（ここでは主にアンカータグへのクリック）であり，出力されている語をクエリとしてウェブサービス・コンポーネントへ送る機

能である．実行している CGI 自身を呼ぶ**セルフ・リンク** *(Self link)* と，ほかの URL を指定するアンカーとある (図では単に Link と表記する).

（4） ユーザは 3 種類のコンポーネント群の合成を簡単なスクリプト言語を使って記述する．それに従って逐次的に CGI が生成される．

## 3. コンポーネントと合成

この研究では，ウェブサービスから得られる XML タグを属性とすることで，XML データを表データととらえる．以下に定義されるウェブサービスの抽象化「コンポーネント」では，受け渡されるデータは「表」であり，行を**レコード**，属性を**チャネル**と呼ぶ.

**コンポーネント**とは，複数の異なる**チャネル**を持った入出力機械である．形式的に述べると，入力チャネル $i_1, \cdots, i_n (n \geqq 1)$ のうち $p$ 個のチャネル $i_{k_1}, \cdots, i_{k_p} (1 \leqq p \leqq n)$ に入力が行われたときに，出力チャネル　$o_1, \cdots, o_m (m \geqq 1)$ から $q$ 個のチャネル $o_{l_1}, \cdots, o_{l_q} (1 \leqq q \leqq m)$ よりデータが $q$ 対になって $r$ 個出力する機械のことを言う．図式化すると以下のようになる．

$$
\begin{aligned}
\text{入力}\quad & (x_1, \cdots, x_p) \\
\text{出力}\quad & (y_{1,1}, \cdots, y_{1,q}), \\
& (y_{2,1}, \cdots, y_{2,q}), \\
& \cdots, \\
& (y_{r,1}, \cdots, y_{r,q})
\end{aligned}
$$

このとき出力は $(r, q)$ 型の表となっている．出力の各行を**レコード** と呼ぶ．以下に挙げるようにコンポーネントには 3 種類ある。

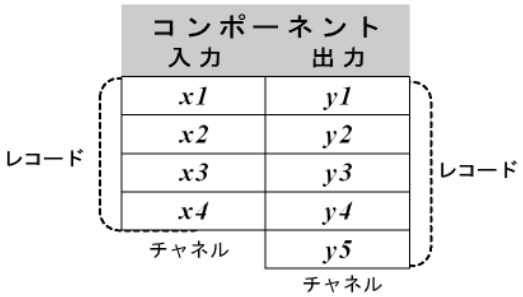


図 1　コンポーネントの枠組み

### 3.1 ウェブサービス・コンポーネント

ウェブサービスと API を通して通信するコンポーネント．入力はウェブサービスへの検索クエリであり，出力はウェブサービスからの検索結果である．通常ウェブサービスから返ってくるデータ形式は属性名などをタグにした XML 形式であるが，コンポーネントはデータをレコードのリストに抽象化して取り扱うことにする．

図 2 は `amazon.com` において公開されているウェブサービスの API で，商品検索 (音楽C D) の機能の一部をコンポーネントとして図式化したものである．入力チャネル `ItemSearch` に入力されたキーワードに対し，検索でヒットした商品情報が 4 つの属性を持つレコード (`artist, album, ASIN, URL`) のリストとして出力される．それぞれ，アーティスト名，アルバム名，商品番号，商品紹介の URL となっている．

図 2　例：Amazon ECS コンポーネント

### 3.2 出力コンポーネント

**出力コンポーネント**に入力されるのはレコードのリストであり，ブラウザに表形式で表示されるデータである．入力チャネルの数は受け取るレコードの内容に応じて可変である．また，出力コンポーネントからの出力は *CGI* **リンク**であり，他のコンポーネントへ入力する機能として実現される．

図 3 の出力コンポーネントは，語と URL(`word, URL`) のリストを受け取り，属性 `word` からは次のコンポーネントへ送り，`URL` からはそれ自身のリンクへつなぐ．`word` のリンクは実行している CGI 自身をよび (*Self link*)，指定されたコンポーネントへ実行を移すことを表す.

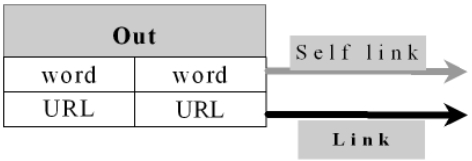


図 3　例：出力コンポーネント

### 3.3 パイプ・コンポーネント

パイプ・コンポーネントは，入力されたレコードのリストを並べ替え（ソート）たり，複数のコンポーネントからの同種のレコードのリストを併合（マージ）するなど，データの成形や簡単な統計情報（ヒストグラムの作成）などを行う.

図 4 の左は 3 組のレコード $(\mathtt{title}_i, \mathtt{author}_i, \mathtt{source}_i), (i = 1, 2, 3)$ を併合して 1 組のレコードのリストにするパイプ・コンポーネント．右は (`aritist, album`) に入力された 2 つの語を 1 つの文字列に連接して出力する.

## 4. PSM のスクリプト言語

PSM のスクリプト言語は合成するコンポーネント・チャネルの対を記述することで，機能合成を実現する CGI を生成する．以下で，$\mathtt{C}, \mathtt{D}$ をコンポーネント，$\mathtt{a}, \mathtt{b}, \mathtt{a}_1, \mathtt{a}_2, \mathtt{a}_3, \cdots, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3, \cdots$ をチャネルとし，$\mathtt{a}, \mathtt{b}$ と $\mathtt{a}_\mathtt{i}, \mathtt{b}_\mathtt{i}$ の対はそれぞれ同じ属性を持つものとする．コンポーネント `C` のチャネル `a` とコンポーネント `D` のチャネル `b` を合成するには

| Merge & Sort | |
|---|---|
| source1 | source |
| title1 | title |
| authors1 | authors |
| title2 | |
| authors2 | |
| source2 | |
| title3 | |
| author3 | |
| source3 | |

| Concat | |
|---|---|
| title | keyword |
| authors | |

図 4　例：マージ＆ソート および 連接

```
C.a -> D.b
```

また，コンポーネント C のチャネル $a_1, a_2, a_3$ とコンポーネント D のチャネル $b_1, b_2, b_3$ をそれぞれ合成するには

$$C.a_1{:}a_2{:}a_3 \text{ -> } D.b_1{:}b_2{:}b_3$$

と記述する．

合成形態は大別して 2 種類ある．メタ検索エンジンのように同種類のウェブサービス (たとえば論文検索データベース群) に同じクエリを与え，返された検索結果を列挙するような形態 (**同種マッシュアップ，***homogeneous mash-up*) と，あるウェブサービスの検索結果などから新しいキーワードを抽出し，種類の異なるウェブサービスのクエリとする形態 (**異種マッシュアップ，***heterogeneous mash-up*)．以下では，コンポーネント Start を，生成された CGI がユーザからブラウザを通して最初に受け取るクエリを出力するコンポーネントとする．図 5 と図 6 の様な図式を**マッシュアップ・ダイアグラム**(mash-up diagram) と呼ぶ．

### 4.1　同種マッシュアップの例

図 5 は 3 つの学術論文データベース (JSAI, IEICE, IPSJ) を機能合成したものである．それぞれのデータベースから著者，題名，出典情報が出力され「マージ＆ソート」コンｆポーネントに渡されている.「マージ＆ソート」コンポーネントは処理をした後，出力コンポーネントへデータを渡す．出力コンポーネントでは，著者と題名の 2 つの属性に独立した CGI リンクが定義されており，これらは最初の Start コンポーネントへ渡される．Tee-3 は入力のそのまま 3 つのチャネルにコピーして出力する．スクリプトは次のように記述される．MS はマージ＆ソート・コンポーネント，Out は出力コンポーネントを表す．

```
Start.keyword -> Tee-3,word,
Tee-3.word1 -> IPSJ.keywords,
Tee-3.word2 -> IEICE.keyword,
Tee-3.word3 -> JSAI.keys,
IPSJ.source:title:authors
-> MS.source1:title1:authors1,
IEICE.source:title:authors
-> MS.source2:title2:authors2,
JSAI.source:title:authors
-> MS.source3:title3:authors3,
MS.source:title:authors
-> Out.source:title:authors,
Out.title:authors
-> Concat.title:authors,
Concat.keyword -> Tee-3.word
```

### 4.2　異種マッシュアップの例

ウェブサービス *Rhapsody* は音楽情報を配信するサービスである．入力チャネル artist，album にアーティスト名やキーワードを入力すると，ヒットしたアーティストの名前，アルバム名，楽曲名，およびこのサイトにおける楽曲番号 rcID が返される．出力 1 にはアーティスト名とアルバム名の重複出力があるが，これは同じデータで Rhapsody にもう一度問い合わせをするものと，次の AmazonECS へのクエリにするものと 2 つの CGI リンクがあることを意味する．*AmazonECS* は図 2 と同じものである．Out1 と Out2 は出力コンポーネントである．図 6 のスクリプトの記述は次のようになる．

```
Start.keywords -> Rhapsody.album,
Rhapsody.artist:album:track:rcID
-> Out1.artist:album:track:rcID,
Out1.artist2 -> Start.artist
Out1.artist1:album1 -> Concat.word1:word2,
Concat.word -> AmazonECS.ItemSearch,
AmazonECS.artist:album:ASIN:URL
-> Out2.artist:album:ASIN:URL,
Out2.URL -> http://*****/
```

## 5.　まとめと課題

本稿に先行する著者らの論文 [3] では，出力コンポーネントが今回提案しているパイプ・コンポーネントの役割を兼ねていたが，出力コンポーネントの役割を (1) 入力データに手を加えずそのままブラウザに表示することと，(2) 出力チャネルは CGI リンクになることの 2 点に制限し，パイプ・コンポーネントを導入することでアーキテクチャの内容が明確にした．

また，独立に動作するプロセスの結合の研究 [2] などがあるが，プロセスの結合ダイアグラムがサイクルを含む場合はシステムの停止性が問題となってくる．本稿で提案する PSM によるマッシュアップ・ダイアグラムでもサイクルを含むものを記述できる．しかし，PSM が想定する入出力はブラウザ・インターフェースを想定しているため，サイクルの一部に少なくとも 1 つの出力コンポーネントを含むという特徴があり，それによって停止性の問題は生じないことが保証される．

今後の課題として，出力コンポーネントにおいて，CGI リンクに渡す語が重複する場合，記述が煩雑になるがどのように回避できるか，ということ，どのようなパイプ・コンポーネントを準備すればよいか，の 2 点が挙げられる．

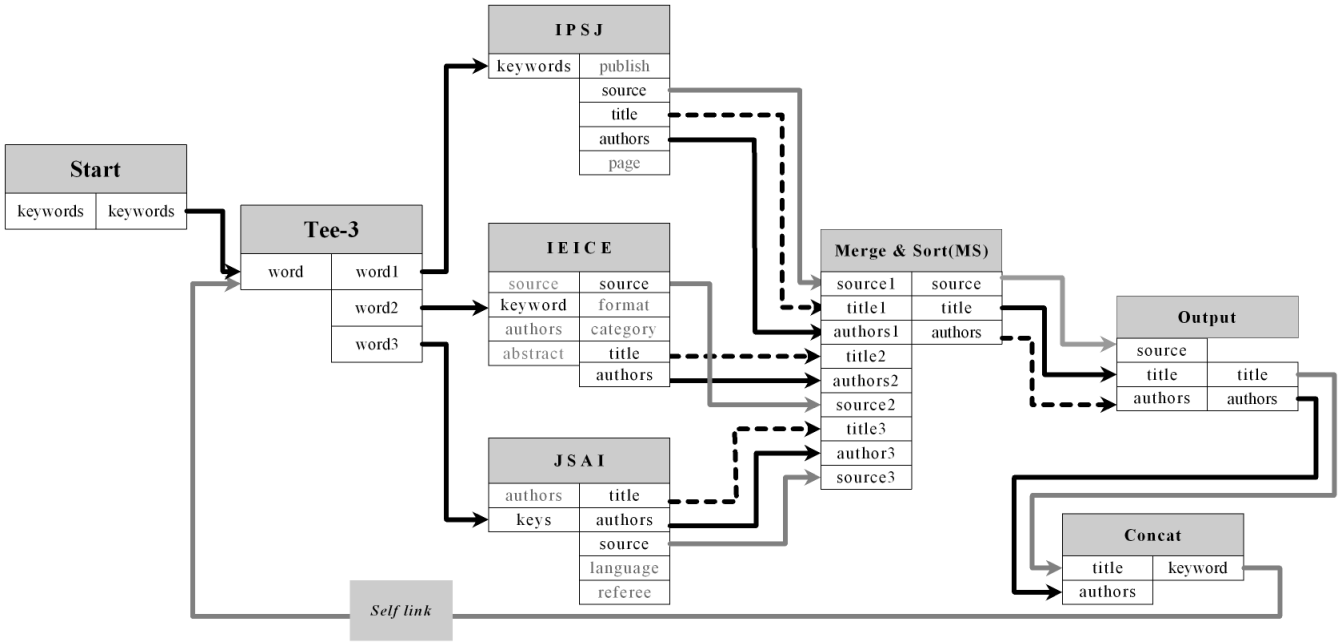


図 5　homogeneous な合成

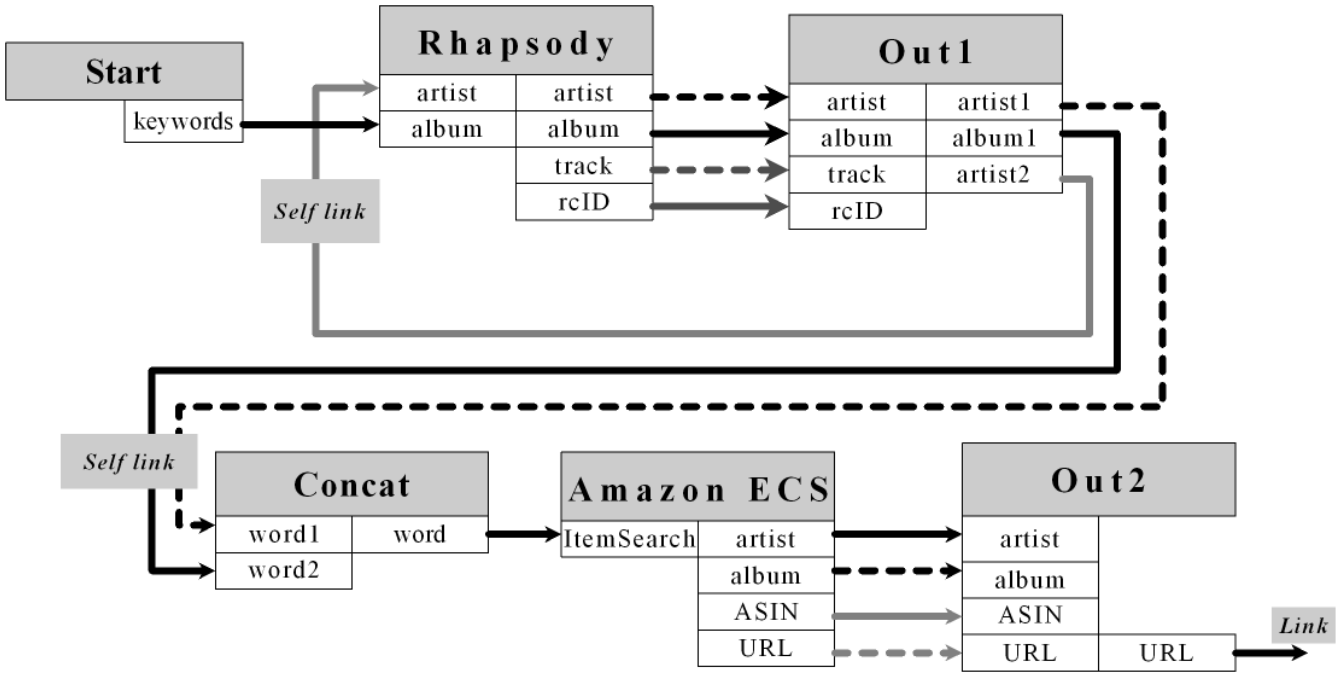


図 6　heterogeneous な合成

## 文　献


[1] BrightPlanet, *The Deep Web: Surfacing Hidden Value*, BrightPlanet White Paper, 2000.

[2] R. Milner, *Communicating and Mobile Systems: The π-Calculus*, Cambridge University Press, 1999.

[3] M. Mori, T. Nakatoh and S. Hirokawa, *Functional Composition of Web Databases*, Proceedings of ICADL2006, Lecture note in Computer Science 4312, Springer Verlag, Nov. 2006.

[4] T. Nakatoh, K. Ohmori and S. Hirokawa, *A Report on Metadata for Web Databases*, IPSJ SIG Technical Reports, 2004-ICS-138(17), pp. 95-98, 2004.

[5] T. Nakatoh, K. Ohmori, Y. Yamada and S. Hirokawa, *COMPLEX QUERY AND METADATA*, Proc. ISEE2003, pp. 291-294, 2003.

[6] T. Nakatoh, Y. Yamada and S. Hirokawa, *Automatic Generation of Deep Web Wrappers based on Discovery of Repetition*, Proc. of the First Asia Information Retrieval Symposium (AIRS 2004), pp.269-272, 2004.

[7] S. Yokoyama, A. Matono, S.M. Pahlevi and I. Kojima, *A Framework for Modularization and Mashup of JavaScript Codes on Web2.0*,

[8] Z. Wu, V. Raghavan, C. Du, K. Sai C, W. Meng, H. He and C. Yu, *SE-LEGO: creating metasearch engines on demand*, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '03), 2003.

[9] *Project DAISEn: Directory Architecture for Integrated Search Engines*,
`http://daisen.cc.kyushu-u.ac.jp/`

[10] W. Wu, C. Yu, A. Doan and W. Meng, *An Interactive Clustering-based Approach to Integrating Source Query Interfaces on the Deep Web*, pp.95–106, SIGMOD Conference 2004.

[11] F. Liu, C.T. Yu and W. Meng *Personalized Web Search For Improving Retrieval Effectiveness*, pp.28–40, IEEE Trans. Knowl. Data Eng. 16(1), 2004.