社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

マッシュアップAPIを用いた異なるライフログサービスの連携

鎌田　早織† 　坂本　寛幸† 　井垣　宏† 　中村　匡秀†

† 神戸大学　〒657–8501 神戸市灘区六甲台町 1–1
E-mail: †{kamada,sakamoto}@ws.cs.kobe-u.ac.jp, ††{igaki,masa-n}@cs.kobe-u.ac.jp

あらまし　異なる種類のライフログを効率的に集約・連携 (マッシュアップ) するために，先行研究において我々は，ライフログのための標準データモデルと，ライフログデータを横断的に検索・取得するマッシュアップ API を提案している．これらの有効性を評価するために，本研究では提案するマッシュアップ API を用いて実際のライフログサービスをマッシュアップするアプリケーションを構築する．具体的には，Twitter, Flickr, GARMIN Connect の 3 つのライフログサービスを連携し，ウォーキング中のつぶやきと写真データを，歩いたコースの地図上にオーバーレイするサービス LifeLogMaps を開発する．ライフログサービスが公開している API を用いた場合と，マッシュアップ API を用いた場合の二通りの開発を行い，両者の比較評価を行う．

キーワード　ライフログ　マッシュアップ API　標準データモデル　評価

Integrating Heterogeneous Lifelog Services Using Mashup APIs

Saori KAMADA†, Hiroyuki SAKAMOTO†, Hiroshi IGAKI†, and Masahide NAKAMURA†

† Kobe University　rokkoudaityou 1–1, nada-ku, Kobe, Hyogo, 657–8501 Japan
E-mail: †{kamada,sakamoto}@ws.cs.kobe-u.ac.jp, ††{igaki,masa-n}@cs.kobe-u.ac.jp

Abstract　In order to integrate heterogeneous lifelog services efficiently, we have previously proposed the lifelog common data model and the mashup APIs. To evaluate their feasibility, in this paper we construct a practical mashup application, called LifeLogMaps, using the proposed mashup APIs. More specifically, LifeLogMaps aggregate three kinds of data (texts, pictures, trails) recorded during a walking exercise onto a single map, by integrating three lifelog services: Twitter, Flickr and GARMIN connect. We conduct two types of development processes with and without the proposed APIs. We then evaluate the advantage and limitation of the proposed method against the conventional mashup development.

Key words　lifelog, mashup, API, common data model, evaluation

1. はじめに

近年，人間の日常生活における行動をデジタルデータとして記録する**ライフログサービス**が注目を集めている．インターネット上では，Web 技術を応用した様々なライフログサービスが公開・提供されている．具体例として，ユーザのつぶやきを記録する “Twitter” [1]，写真を記録する “Flickr” [2]，ランニング／ウォーキングしたコースや心拍数，歩数を記録する “GARMIN Connect” [3] などがある．

各自がばらばらに記録したライフログを，集約・連携（**マッシュアップ**と呼ぶ）することで，より付加価値の高い情報サービスへと発展させることが期待できる．例えば，GARMIN Connect と Flickr をマッシュアップすれば，ウォーキング中に撮影した写真をコースログの上に重ねることができ，より視覚的なトレーニング振り返りサービスが実現できる．

こうしたマッシュアップを支援するために，既存のライフログサービスの中には，外部のプログラムからライフログデータにアクセスするための **API (Application Program Interface)** を公開しているものもある．しかし，これらの API には統一した標準が無くサービスごとにばらばらである．したがって，連携するアプリケーションの組み合わせごとに異なるプログラムロジックが必要となり，マッシュアップの開発効率 (生産性) が低下してしまう．

ライフログの効率的なマッシュアップを実現するために，我々は先行研究において，ライフログのための**標準データモデル** を提案している [4]．標準データモデルでは，ライフログに必要なデータ項目をアプリケーションに独立なものと依存するものに分類することで，様々なライフログサービスに中立な論理データモデルを構築している．日付や時刻，ユーザ名，場所等，ライフログの種類に強く依存しないデータは**標準スキーマ**によっ

て定義し，つぶやきや写真画像の URL 等，ライフログの内容に関するデータはアプリケーション毎の**固有スキーマ**によって外部定義する構造となっている．

また最新の成果 [5] では，様々なライフログを標準データモデルへ変換し，それらを検索・取得するための汎用的な API（**マッシュアップ API** と呼ぶ）を設計・実装している．これにより，多種多様なライフログのデータを，プログラムから統一的な方法で横断的にアクセスすることが可能となる．

しかしながら，提案する標準データモデルやマッシュアップ API が実際のマッシュアップ開発の効率化にどれだけ貢献するかは，まだ評価されていない．

そこで本研究では，マッシュアップ API を用いて実際のマッシュアップアプリケーションを構築し，その効果を評価することを目的とする．より具体的には，Twitter，Flickr および GARMIN Connect の 3 つの異なるライフログをマッシュアップし，ウォーキング中のつぶやき (Twitter で記録) と写真データ (Flickr で記録) を歩いたコース (GARMIN で記録) の地図上にオーバーレイするサービス **LifeLogMaps** を開発する．開発においては，これら 3 つのライフログサービスが個別に公開している**固有 API** を用いる場合と，提案するマッシュアップ API を用いる場合の 2 通りで開発し，両者の比較・評価を行う．

2. 準　　備

2.1 ライフログサービス

ライフログとは「人間の行い (life) をデジタルデータとして記録 (log) に残すこと」である．人間の日常行動を記録として残しておき，生活の振り返りや改善，さらには意思決定，行動支援に役立てようとする試みである [6]．

ネットワーク技術の進歩により，現在インターネット上には様々な**ライフログサービス**が公開され，人気を集めている．これらのライフログサービスは，それぞれ記録する内容に特化した情報や機能を持ち，ユーザは自分の用途に応じてサービスを選択，好きな時にライフログを記録する．本稿では様々なライフログサービスの中から，特に以下の 3 種類をとりあげる．

Twitter [1]:　「今なにをしているか」や「何を思っているか」を自分専用ページから短く発言する（**つぶやく**）ことで，記録していくサービスである．個々のつぶやきは，コミュニティ内の仲間にリアルタイムに配信される．また，つぶやきの履歴は時間順の一覧（タイムライン）で表示して見ることができる．手軽に更新でき，小さなブログのようなサービスということから「ミニブログ」などと呼ばれることもある．

Flickr [2]:　自分で撮影したデジタル**写真画像**をネットワーク上のストレージにアップロードし，保存・管理することができるサービスである．Flickr 上で公開された写真は URL で管理され，複数人で共有可能である．また，タイトル情報や写真へのコメントを残すこともできる．さらに，タグと呼ばれる任意のキーワードを写真に付与することで，膨大な画像データベースを効率的に検索できるようになっている．

GARMIN Connect [3]:　腕時計型の高感度 GPS レシーバーで，自分が移動した**経路情報**（緯度，経度，高度）を記録するトレーニング用途のサービスである．記録した経路情報は PC やネットワークにアップロードして，地図上に移動経路を表示することが可能である．また，日付やラップごとに，消費カロリー，走行時間，走行距離等を計算して管理する．オプションの FoodPod や Heart Monitor を併用することで，歩数や心拍数も記録できる．

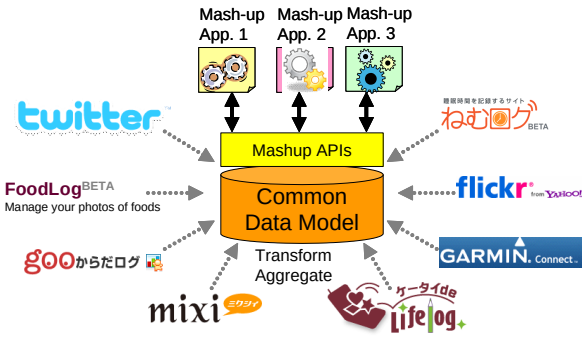

図 1　ライフログサービスの新しい集約・連携方式

2.2 ライフログのマッシュアップ

異なるライフログサービスで取得されたばらばらのライフログを，ネットワーク越しに集約・連携することで，単独で利用する以上の価値が生まれる可能性がある．例えば，Flickr と Twitter を連携することで,「写真をつぶやきで説明する」サービスを実現できる．また，GARMIN と Flickr を連携することで，ランニング中に撮影した写真をコース上に重ねて地図表示するサービスも可能である．

このように，複数のライフログの連携による付加価値創造を，本稿では**ライフログのマッシュアップ**と呼ぶ．ライフログサービスの中には，マッシュアップ用途を見越して，外部プログラムから自身のデータへアクセスするための **API** (Application Program Interface) を公開しているものもある．

しかしながら，こうした API や API で得られるライフログのデータ構造には統一的な標準がなく，サービスごとにまちまちである．したがって，マッシュアップの開発においては，連携するサービスの組み合わせごとに異なるプログラムロジックが必要となっており，開発効率が悪い．また，サービスの API 仕様の改訂に伴ってプログラムの見直しが必要となるため，再利用性・信頼性を下げる要因にもなっている．

3. 先行研究：ライフログ標準データモデルとマッシュアップ API

3.1 ライフログ標準データモデル [4]

ライフログの効率的なマッシュアップを目的として，我々は先行研究 [4] においてライフログのための**標準データモデル**を提案している．図 1 に標準データモデルを用いたライフログのマッシュアップ形態を示す．各ライフログアプリケーションのデータは，何らかの形で，様々なライフログサービスに中立な標準データモデル (Common Data Model) に変換される．この標準データモデルの上では汎用的な API(**マッシュアップ API** と呼ぶ) が定義され，様々なライフログへの標準的かつ横断的

表 1　ライフログ標準データモデルのスキーマ

観点	データ項目	名称	物理データ形式	物理データ解釈
WHEN	<date>	日付	YYYY-MM-DD	データを生成した日付を UTC 時間で指定
	<time>	時刻	hh:mm:ss	データを生成した時刻を UTC 時間で指定
WHO	<user>	主体ユーザ	文字列	ログの主体となるユーザのアカウント名
	<party>	共に行動したユーザ	文字列	主体ユーザと協働するユーザ群の情報
	<object>	対象ユーザ	文字列	行動の対象となったユーザの情報
WHERE	<location>	場所 (経度・緯度指定)	<longitude>経度</longitude>	データを生成した場所の経度を浮動小数点で指定
		または	<latitude>緯度</latitude>	データを生成した場所の緯度を浮動小数点で指定
		場所 (住所指定)	<geo>住所</geo>	データを生成した場所の緯度を文字列で指定
HOW	<application>	アプリケーション	文字列	データを記録したサービスアプリケーション名
	<device>	デバイス	文字列	データを記録したデバイス
WHAT	<content>	ログ内容	任意形式	変換元のオリジナルデータをそのまま格納
	<ref_schema>	外部スキーマ	URI	ログ内容を解釈するためのスキーマへのリンク

なアクセスが可能となる．

文献 [4] では，標準データモデルを構築するために，ライフログを構成するデータ項目を 5W1H(What, Why, Who, When, Where, How) の観点から分析・抽出した後，サービスの種類に強く依存するものと非依存なものに分類している．サービスに依存しにくいデータ項目 (When, Who, Where, How) は，ライフログの標準データモデルのスキーマに組み入れる．一方，サービスに依存するデータ項目 (What, Why) は，内容を解釈せずオリジナルデータをそのまま保存する．このデータの解釈は，サービスごとに与えられる外部スキーマに任せるものとし，スキーマへの参照を保持する．

より最新の成果 [5] において我々は，標準データモデルの**物理データ設計**を行っている．表 1 に提案する標準データモデルのスキーマおよび物理データ形式，物理データ解釈を示す．さらに，ライフログ固有のオリジナルデータから標準データへの具体的な変換手順を提案している．データ変換の方法としては**オフライン変換方式**を採用している．これは，予めスケジュールされたタイミングで，バッチ処理を用いてライフログサービスからデータを一括取得・変換し，変換後のデータをリポジトリに蓄積しておく方式である．

3.2　マッシュアップ API [5]

さらに [5] では，標準データモデルへ変換されたライフログデータにアクセスするための 2 種類の**マッシュアップ API** を実装している．1 つ目の `getLifeLog()` は，標準データモデルに対して，日付，時刻，ユーザ，位置，サービス，デバイス，抽出データに関するクエリを指定し，クエリにマッチする全てのライフログを取得する API である．また，2 つ目の `getLifeLogNearestTime()` は，クエリに合致するライフログのうち，最も時間が近いライフログを 1 つだけ取得する API である．以下にその仕様を述べる．

■ `getLifeLog(date,time,user,location,application,select);`

・パラメータ

- `date:`　日付に関するクエリ
- `time:`　時刻に関するクエリ
- `user:`　ユーザに関するクエリ
- `location:` 場所に関するクエリ
- `application:` サービスに関するクエリ
- `select:` 抽出データに関するクエリ

・クエリ：定数 (`""`で囲んだ文字列)，論理和 (`+`)，ワイルドカード (`*`) から構成される検索式．

・戻り値：クエリに合致するライフログデータのリスト (配列)．各ライフログデータは，表 1 の標準データ項目をメンバとして持つ構造体 (ハッシュ) で表現される．

■ `getLifeLogNearestTime(datetime, user, application);`

・パラメータ

- `date:`　日付に関するクエリ
- `time:`　時刻に関するクエリ
- `user:`　ユーザに関するクエリ
- `application:` サービスに関するクエリ

・戻り値：クエリに合致するライフログデータのうち，日時が最も近いデータ (1 件)

このマッシュアップ API は，perl 言語で書かれたライブラリとして実装されており，任意の perl プログラムにインクルードして使用することができる．現在このライブラリを，開発言語や実行プラットフォームに非依存な REST-Web サービスとして公開すべく開発中である．

4.　マッシュアップ API を用いたアプリケーション開発

本研究では，前節の提案手法の有効性を評価するために，具体的なマッシュアップアプリケーション LifeLogMaps を開発する．開発においては，マッシュアップ API を用いた場合と用いない場合の 2 通りの開発を実施し，両者の比較を行う．

4.1　LifeLogMaps

LifeLogMaps は，Twitter，Flickr および GARMIN Connect の 3 つの異なるライフログサービスからそれぞれのログデータを取得し，ウォーキング中に記録したつぶやき (Twitter)，写真 (Flickr)，歩いたコース (GARMIN Connect) を時系列で集約，Google Maps 上にオーバーレイして可視化するアプリケーションである．

LifeLogMaps のユーザは，GARMIN の GPS レシーバ，デジタルカメラ，携帯デバイス (PDA，携帯電話等) を持って，街

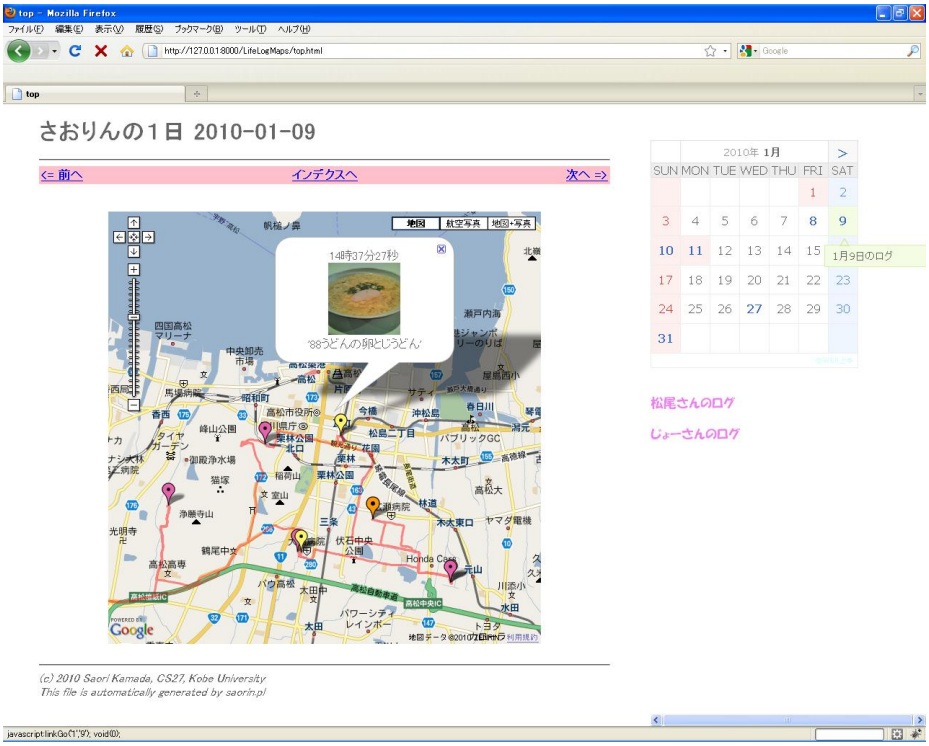

図 2　LifeLogMaps の画面

を散策する．散策中には，思いついたことを任意のタイミングで携帯デバイスを用いて Twitter に書き込む．また，自分の気に入った場所でデジタルカメラを用いて写真を撮影する．散策終了後，気に入った写真を Flickr にアップロードし，GPS データをレシーバから GARMIN Connect にアップロードする．LifeLogMaps は，Twitter, Flickr, GARMIN Connect のデータをダウンロードしてマッシュアップし，JSON 形式のファイルに出力する．このデータを JavaScript で読み込み，Google Maps 上に表示する．

LifeLogMaps の概観を図 2 に示す．この画面は，ユーザ「さおりん」が 2010 年 1 月 9 日高松市内を散策した時のライフログを表している．画面には，歩いた軌跡を表すラインと，つぶやきや写真を撮った場所を示すマーカーが記されている．GPS の変移を線で結ぶことで歩いた軌跡を描く．つぶやいた場所や写真を撮った場所に置いたマーカーをクリックすると，吹き出しが表示され，つぶやきや写真を見ることができる．また，図の右側に表示されているカレンダーの日付をクリックすると，左側にその日のログを表す Map を表示できる．また，他のユーザの Map も表示することができる．

LifeLogMaps では，それ専用のライフログサービスを構築せず，3 つの**既存の**ライフログサービスを再利用して実現することに注意されたい．3 つのライフログサービスを 1 つのアプリケーションで集約・表示することで，個々のサービスをばらばらに利用するよりも，より付加価値の高い情報・サービスを提供できることがわかる．

4.2　マッシュアップ API を用いない従来型の実現方式

まず，マッシュアップ API を用いずに，ライフログサービスが個別に公開している**固有 API** を用いる従来型の実現方式を説明する．3 つのライフログサービスのデータを集約し，LifeLogMaps を実現する流れを図 3 に示す．マッシュアップは大きく 4 つのステップから構成される．以下に Step1〜4 の手順を述べる．

Step1 (固有 API を用いたライフログデータの取得): 各ライフログサービスが公開する API を用いて，ログデータを取得する．Twitter は，ユーザ名とパスワードを入力しておき，Twitter API の「user_timeline」を呼び出してデータを得る．Flickr は，APIkey と Secret 番号を入力しておき，Flickr API の「flickr.photos.search」を呼び出してデータを得る．GARMIN は，GARMIN Connect から日付とユーザでデータを検索，ログデータをエクスポートし，ファイルとしてダウンロードする．3 つのログファイルは全て XML で得られるが，データ形式はそれぞればらばらである．

Step2 (必要なデータ項目の抽出): Step1 で取得した 3 種類の XML データから，必要なデータ項目を抽出する．Twitter からは日付・時刻 (`<created_at>`) とつぶやき (`<text>`)，Flickr からは日付・時刻 (`<datetaken>`) と写真タイトル (`<title>`)，写真 URL(`url`) を，GARMIN からは日付・時刻 (`<Time>`) と緯度・経度 (`<Position>`) をそれぞれ抽出する．

Step3 (日付・時刻でマッシュアップ): Step2 で抽出した 3 つのサービスのデータを，日付・時刻でつきあわせてマッシュアップする．日付・時刻の表現形式，解釈はサービスごとに異なるので，必要なら変換を行い表現形式と意味の統一化をはかる．LifeLogMaps では時刻を日本標準時で管理したいが，Twitter や Flickr は世界標準時 (UTC) で管理しているので，日本時間に変換する．また，表現形式は，YYYY-MM-DD hh:mm:ss の形式に統一する．例えば Twitter であれば，「Sat Jan 09 06:11:55

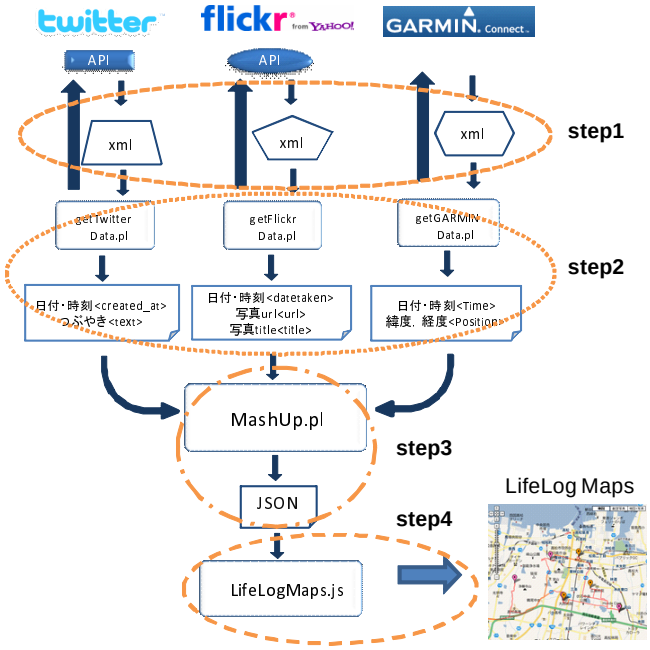

図 3　固有 API による実現方式 (従来方式)

+0000 2010」で取得し「2010-01-09 15:11:55」に直す．つきあわせの結果，Twitter の「つぶやき」と Flickr の「写真タイトル・URL」と GARMIN の「位置情報」が共通の「日付・時刻」で関連付けられる．最後にマッシュアップしたデータを JSON 形式で書き出す．

Step4 (LifeLogMaps の生成): Step3 でマッシュアップしたログデータを Google Maps API を使って可視化する．データ中の「位置情報」は，隣り合う座標を線で結び，地図上の軌跡とする．また，各「つぶやき」に対しては，紐付けられた位置情報の座標にマーカーを置き，その内容を表示するようにする．写真についても同様，紐付けられた座標にマーカーを置き，タイトルと写真 URL で示される写真を表示する．

4.3　マッシュアップ API を用いた実現方式

次に，提案するマッシュアップ API を用いた場合の実現方式を述べる．この方式では，3 種類のライフログサービスのデータは 3.1 で述べた標準データモデルに変換されており，3.2 のマッシュアップ API を用いてデータにアクセスする．ライフログサービスの違いを意識しなくて良いため，マッシュアップの流れは図 4 に示す通り，非常に単純なものとなる．

Step1´(ライフログデータの取得と関連付け): 与えられた日付・ユーザを元に `getLifeLog()` を実行して，Twitter で記録したその日の全ての「日付・時刻・つぶやき」のリスト (`@twitter`) を取得する．次に，このリストの各要素`$t` に対し，その時刻 (`$t->{time}`) を元に `getLifeLogNearestTime()` を実行し，GARMIN で記録した時間的に最も近いログ (`$t_garmin`) を検索し，これにつぶやき (`$t->{text}`) を関連付ける．Flickr の各写真情報についても同様に位置情報との関連付けを行う．最後に，GARMIN で記録したすべての「日付・時刻・位置情報」を加えて，LifeLogMaps で必要なデータを構成，JSON 形式で書き出す．

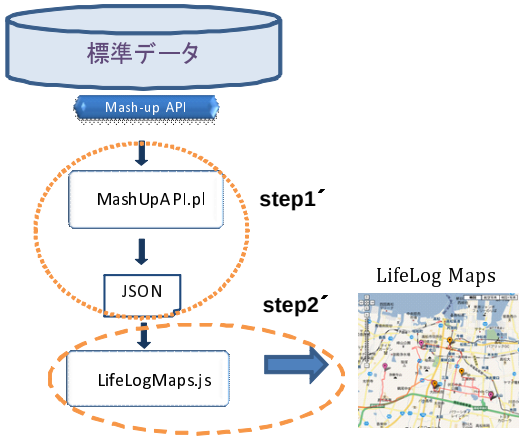

図 4　マッシュアップ API による実現方式 (提案方式)

```
#!/usr/bin/perl -w

require "mashupAPI.pl";  #マッシュアップAPIをインクルード

my @mashup;  #マッシュアップされたライフログデータを入れる配列
my @garmin;  #GARMINのライフログデータ用配列
my @twitter; #Twitterのライフログデータ用配列
my @flickr;  #Flickrのライフログデータ用配列

#---------------------Step 1: TwitterとGARMINのマッシュアップ -----------------------
# 2010-01-09のユーザ"saorin"のTwitterデータを取得(日付, 時刻, ユーザ, つぶやき)
@twitter = &getLifelog({date=> "2010-01-09", user => "saorin",
                application => "Twitter","select" => "date+time+usr+text"});

#各Twitterデータに一番時間の近いGARMINのデータを検索・取得(日付, 時刻, ユーザ, 位置)
foreach my $t (@twitter){
                my $t_garmin = &getLifeLogNearestTime({date=>$t->{date}, time=>$t->{time},
                                application=>"GARMIN", user => $t->{user},
                                select => "date+time+user+location"});
                $t_garmin->{text} = $t->{text}; #Twitterの呟きをGarminにひも付け
                push(@mashup, $t_garmin); #マッシュアップ要素とする
}

#---------------------Step 2:  FlickrとGARMINのマッシュアップ -----------------------
#2010-01-09のユーザ"saorin"のFlickrデータを取得(日付, 時刻, ユーザ, 写真URL, 写真タイトル)
@flickr = &getLifelog({date=> "2010-01-09", user => "saorin", application => "Flickr",
                "select" => "date+time+user+url_s+title"});

#各Flickrデータに一番時間の近いGARMINのデータを検索・取得(日付, 時刻, ユーザ, 位置)
foreach my $f (@flickr){
                my $f_garmin = &getLifeLogNearestTime({date=>$f->{date}, time=>$f->{time},
                                application=>"GARMIN", user => $f->{user},
                                select => "date+time+user+location"});
                $f_garmin->{url_s} = $f->{url_s}; #Flickrの写真URLとタイトルをGarminにひも付け
                $f_garmin->{title} = $f->{title};
                push(@mashup, $f_garmin); #マッシュアップ要素とする
}

#---------------------Step 3: GARMINのその日のすべてのデータを加える------------------
#2010-01-09のユーザ"saorin"のGARMINデータを取得
@garmin = &getLifelog({date=> "2010-01-09", user => "saorin", application => "GARMIN",
                "select" => "date+time+user+location"});

#マッシュアップ要素に入れる
push(@mashup, @garmin);

#マッシュアップデータをLifeLogMapsのJSON形式でファイル出力
my $xml = &outputByJSON(¥@mashup, "LifeLogMaps-saorin-2010-01-09.js");
```

図 5　マッシュアップ API を用いた perl 実装

Step2´(LifeLogMaps の生成): 4.2 で述べた Step4 と同じであるため，説明を省略する．

図 5 に Step1´の perl 言語による実装例を示す．マッシュアップ API を用いることで，およそ 30 行程度の短いプログラムで 3 種類のライフログサービスのマッシュアップが可能である．コード中，異なるライフログデータを同じ API 呼び出しで取得していること，また，取得したデータに対して共通のデータ項目でアクセスできていることに注意されたい．

5.　評　　価

従来型の開発方式とマッシュアップ API を用いた開発方式を，開発効率と実行効率の観点から比較し，評価を行う．

5.1　アプリケーションの開発効率

提案するマッシュアップ API を用いることで，マッシュアップアプリケーションの開発効率がどの程度向上したかを評価す

表 2　LifeLogMaps の開発効率の比較

開発方式	モジュール数	総コード行数	開発工数
固有 API	7	394 LOC	42 人日
マッシュアップ API	2	82 LOC	5 人日

表 3　LifeLogMaps の実行時間の比較

開発方式	入力データセット							
	1/8	1/9	1/10	1/11	1/16	1/27	1/29	1/31
固有API（秒）	3.9	6.4	4.0	3.8	4.2	5.0	4.6	3.9
マッシュアップAPI（秒）	44.7	43.3	20.7	17.4	45.3	8.6	13.2	45.8
処理したログサイズ(KB)	2411	1635	2150	1028	1695	346	699	1307

る．我々は perl 言語および JavaScript を用いて，LifeLogMaps を 4.2 および 4.3 で述べた 2 通りの方式で実装した．表 2 に，開発したモジュール数，総コード行数 (コメント除く実効行)，開発に要した工数をまとめる．提案マッシュアップ API を用いた場合，総行数で約 5 分の 1，開発工数で約 9 分の 1 の効率化を達成できた．

はじめに固有 API を用いた場合を説明する．図 3 に示したとおり，Step 1 で 3 つのサービスからデータを取得する必要がある．データの取得には各サービスが公開する API を用いたが，これらの利用方法を習得するのにまず時間がかかった．また Step2 では，各固有 API の仕様書を元に，オリジナルデータの項目と構造を理解し，データ抽出を行うモジュールをサービスごとに実装する必要があった．Step3 のマッシュアップでは，ばらばらであったデータの解釈，表現形式を統一化するのが煩雑であった．これらの作業はマッシュアップするサービスの種類が増えるたびに累計される．よって，より複雑なマッシュアップアプリケーションの開発では，プログラムの規模がさらに大きくなり，開発工数も増えていくと考えられる．

一方，マッシュアップ API を用いた場合には，図 4 に示すとおり，実装すべき処理数が大幅に削減される．さらに，従来はサービスごとに異なっていたライフログデータの取得方法や表現形式は標準的な方法に統一されている．よって，一旦マッシュアップ API の使い方さえ覚えてしまえば，ライフログサービスの違いを気にすることなく，どのようにデータを付き合わせるかというマッシュアップ・ロジックのみに集中して開発が行える．その結果，実装するモジュール数，コード行数，開発工数の大幅な削減につながる．また，連携するライフログサービスの種類が増えても，API 側で対応している限り，データ取得・抽出に関するアプリケーション側の規模や開発コストは増加しない．このことは，開発するアプリケーションの規模や機能が増えるほど，マッシュアップ API の効果が大きくなることを示している．

5.2　アプリケーションの実行効率

それぞれの方式で実装した LifeLogMaps の実行時間を測定し，実行効率を比較する．評価に用いたデータは，2010 年 1 月に記録した 8 日間分のライフログデータである．各方式について，LifeLogMaps の JSON データ (図 3，図 4 の `JSON`) の生成にかかった時間を測定した．表 3 に結果を示す．表の各列は，データセット毎の処理時間を秒で表している．また，参考のため処理したライフログのサイズを KB で示す．

表 3 よりマッシュアップ API を用いる方が低速であり，約 3〜11 倍の実行時間がかかっている．マッシュアップ API が低速な理由として，現在の API の実装では，ライフログの標準データが XML ファイルで保存されており，それらのファイルを逐次検索するという設計になっていることが大きい．この処理時間を実行時のオーバーヘッドとして考えると無視できない．よって，マッシュアップ API を用いた実現方式では，リクエストのたびにコンテンツ (JSON) を動的生成するのではなく，あらかじめバッチ処理等でオフラインで生成しておく方式を採るべきである．マッシュアップ API の実用化に向けては，せめて固有 API に匹敵する応答時間を達成して欲しい．データベースやキャッシングを用いた高速化が必要であると考える．

5.3　マッシュアップ API の限界

現在のところ，提案するマッシュアップ API は，ライフログデータの検索と取得にしか対応していない．よって，固有 API に比べて各サービスに特化した細かい操作ができない．例えば，Twitter の固有 API は，つぶやきを投稿したり，指定ユーザを自分の friend にしたりできる．また，Flickr の API では，写真を投稿，変換，タグを管理するなどできる．しかしながら，これらの操作はマッシュアップ API の対象範囲外であり，開発アプリケーションで使用するには固有 API を用いるしかない．

マッシュアップ API で何をどこまで実現すべきかは，今後のより高度なマッシュアップの開発を通して考えていきたい．

6.　おわりに

本稿では，我々が提案しているライフログ標準データモデル，および，マッシュアップ API を評価するため，異なるライフログサービス (Twitter, Flickr, GARMIN Connect) を連携したアプリケーション LifeLogMaps を開発した．ライフログサービスの固有 API を用いた従来方式の場合と，マッシュアップ API を用いた提案方式の 2 通りで開発を行い，開発効率，実行効率の評価を行った．その結果，マッシュアップ API を用いた場合，開発工数，プログラム規模が大幅に削減できることがわかった．また，実行効率に関しては改善の余地があることがわかった．実用化に向けての今後の課題としては，マッシュアップ API の高速化と，検索・取得以外の操作による利便性の拡張が考えられる．

謝　　辞

この研究の一部は，科学技術研究費 (若手研究 B 20700027, 21700077) の助成を受けて行われている．

文　　献

[1] Twitter, Inc., “twitter”. `http://twitter.com/`.
[2] Yahoo, “Flickr”. `http://www.flickr.com/`.
[3] Garmin Ltd., “Garmin”. `http://connect.garmin.com/`.
[4] 中村匡秀，下條　彰，井垣　宏，“異なるライフログを集約するための標準データモデルの考察，” 電子情報通信学会技術研究報告，第 109 巻，pp.35–40，Nov. 2009.
[5] 下條　彰，福田将之，井垣　宏，中村匡秀，“異なるライフログをマッシュアップするためのデータ変換・集約アクセス api の実装，” 電子情報通信学会技術研究報告，pp.●●–●●，March 2010.
[6] 相澤清晴，“ライフログの実践的活用：食事ログからの展望，” 情報処理学会誌，vol.50，no.7，pp.592–597，July 2009.