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.

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

下條　彰† 福田　将之† 井垣　宏† 中村　匡秀†

† 神戸大学　〒657–8531 神戸市灘区六甲台町 1–1
E-mail: †{shimojo,fukuda}@ws.cs.kobe-u.ac.jp, ††{igaki,masa-n}@cs.kobe-u.ac.jp

あらまし　ばらばらに記録されたライフログを集約・連携 (マッシュアップ) することで，付加価値の高いサービスを実現することが可能である．先行研究において我々は，異なるライフログを横断的に扱うための標準データモデルを提案している．本稿では，先行研究の提案に基づき，各サービスが持つライフログデータを標準データに変換し，そのデータを利用するためのマッシュアップ API を実装する．具体的には，標準データモデルの物理データ設計，標準データモデルへの変換手法，マッシュアップ API の設計・実装を行う．また，実装した API を実際のライフログサービスのマッシュアップに適用し，開発効率および性能の面から評価を行う．

キーワード　ライフログ, マッシュアップ, 標準データモデル, サービス API, サービス連携

Implementing Data Conversion and Aggregation APIs for Efficient Mash-up of Heterogeneous Life Logs

Akira SHIMOJO†, Masayuki FUKUDA†, Hiroshi IGAKI†, and Masahide NAKAMURA†

† Kobe University　Rokkoudai-cho 1–1, Nada-ku, Kobe, Hyogo, 657–8531 Japan
E-mail: †{shimojo,fukuda}@ws.cs.kobe-u.ac.jp, ††{igaki,masa-n}@cs.kobe-u.ac.jp

Abstract　Aggregating multiple data from different lifelog services can create sophisticated value-added services. In our previous work, we have proposed a common data model to achieve efficient data mashups of heterogeneous lifelogs. Based on the previous work, this paper presents a method that converts the proprietary lifelog data into the common data, and then implements *mashup APIs* for retrieving the converted data. For this, we first conduct the physical data design for the common data model, and then propose the data conversion method. Finally, we design and implement the mashup APIs. We also evaluate the feasibility and the performance of the mashup APIs through the application to actual lifelog services.

Key words　lifelog, mash up, common data model, service API, service cooperation

1. はじめに

近年，人間の行動をデジタルデータとして記録に残す**ライフログ**が注目を集めている．Web 上にはいくつものサービスが存在し，様々な種類のライフログを Web 上で記録・共有できるようになっている．具体例として，ブログから始まり，つぶやきを記録する “Twitter” [1]，写真・映像を共有する “Flickr” [2]，毎日の食事を記録する “FoodLog” [3]，GPS レシーバーで移動経路情報 (緯度，経度) を記録する “GARMIN” [4]，睡眠時間を記録する “ねむログ” [5] 等が存在する．

現在こうしたサービスにおいては，その「記録」と「利活用」はそのサービス内に閉じた形で行われている．しかしながら，様々なサービスでばらばらに記録されたライフログを，集約・連携 (**マッシュアップ**と呼ぶ) する事で，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．

既存のライフログ・サービスの中には，マッシュアップ用の API やブログパーツを用意しているものも存在し，外部プログラムからログデータにアクセスすることができる．しかしながら，これら API の仕様はサービス毎に異なっており，連携するサービスの組み合わせ毎に異なるプログラムロジックが必要となる．マッシュアップの効率化，生産性を向上するには，ある程度標準的なデータモデルおよびサービス API が望まれる．

先行研究 [6] において我々は，異なるライフログを横断的に扱うための標準データモデルを提案している．このモデルでは，5W1H の観点から異なるライフログを分析し，特定のサービスに依存しないデータ項目を抽出，依存データは意味を解釈せず

に埋め込むというアプローチをとっている．具体的なデータ項目は日付，時刻，場所，ユーザ，コンテンツ，アプリケーション，デバイス等となっている．しかし，先行研究では論理的なモデルを定義したにとどまっている．物理的なデータフォーマットや，データ変換の方法，アクセスのための API は未実装であり，その有効性はまだ十分に検証されていない．

そこで本稿では，各サービスが持つライフログデータ (**オリジナルデータ**) を標準データに変換し，そのデータを利用するためのアクセス API を実装する事を目的とする．データ変換を行う手法としては，まずはじめに各サービスの API を利用し，オリジナルのデータを読み出す．次にサービス固有のデータフォーマットと標準データモデルのデータ項目との間のマッピングを定義し，データの変換を行う．変換されたデータをファイルとして書き出し，蓄積する．この変換作業を様々なライフログサービスに対して行うことで，異なるライフログを統一的な形式で保存するリポジトリが構築できる．

アクセス API では，日付，時刻，場所，ユーザ，アプリケーション，デバイスから構成されるクエリに基づき，標準データモデルのリポジトリを検索し，クエリにマッチするライフログを与えられた形式で出力する．検索はライフログサービスの種類に依存せず，統一的な方法によるデータ取得が可能となる．これにより，マッシュアップの大幅な効率化，生産性向上が期待できる．最後に，実装したデータモデル・アクセス API を実際のライフログサービスのマッシュアップに適用し，その有効性を検証・評価する．

2. 準　　備

2.1 ライフログ

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*) として残すこと」[3] 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，自分発見や気づき，さらには意思決定，行動支援に役立てようという試みである．ただし一言で life といっても様々なものが考えられる．日記，心のつぶやき，見た風景，行った場所，移動した経路，生体情報（脈拍，歩数など）など，バリエーションは多岐にわたる．

現在インターネット上では，目的に応じた様々なライフログサービスが展開されている．しかしながら，ライフログの記録と利用はサービスごとに閉じており，様々なライフログのデータはサービスごとにばらばらに管理され，ネットワーク上に分散して蓄積されている．

2.2 ライフログのマッシュアップ

ばらばらに蓄積されたライフログを，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，“FoodLog” と “からだログ” を連携することで，食生活の実態と結果を細かく分析できるだろう．また，“Twitter” と “ねむログ” を連携することで，考え事と睡眠時間の相関が取れるかもしれない．複数のライフログの集約・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**と呼ぶことにする．

こうしたマッシュアップを見越して，ライフログ・サービス

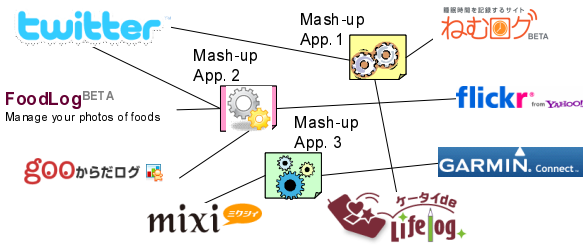

図 1　従来のライフログ・サービスの集約・連携

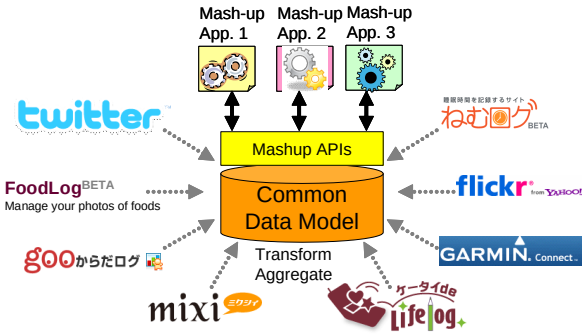

図 2　提案するライフログ・サービスの集約・連携

の中には，外部サービスから自身のライフログへのアクセス手段を，サービス API や Web 部品 (ブログパーツ等) という形で公開しているものも存在する．

2.3 従来のマッシュアップ開発における課題

図 1 に現在のライフログのマッシュアップ形態を示す．図中，Mash-up App. はマッシュアップ・サービスを，線は各ライフログ・サービスへのアクセスをあらわす．マッシュアップ・サービスから，各ライフログ・サービスへのアクセスは，2.2 で述べた API やブログパーツを利用して実現される

現在，こうした API や API によって得られるライフログのデータ構造には統一的な標準が無く，サービスごとにまちまちである．つまり，**ライフログへのアクセス手段がサービスに強く依存**している．したがって，マッシュアップの開発においては，連携するライフログ・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る．

例えば，似たようなライフログ検索を行いたくても，アプリケーションが違えば全く異なる手段で呼び出さなければならない．このことは，マッシュアップの開発効率（生産性）を下げる原因となる．また，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つながる．

2.4 先行研究：標準データモデル

前節で述べた課題を解決し，ライフログの高度かつ柔軟なマッシュアップを支援するため，我々は先行研究 [6] 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 2 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，サービスに強く依存しない標準的なデータモデル (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

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	説明
WHEN	<date>	日付
	<time>	時刻
WHO	<user>	ユーザ
	<party>	共に行動したユーザ
	<object>	対象ユーザ
WHERE	<location>	場所
HOW	<application>	ライフログサービスを実現するアプリケーション
	<device>	ログ記録に用いたデバイス
WHAT	<content>	ログの内容
	<ref_schema>	外部スキーマへの参照
WHY	-------	必要なら<content>内へ

義され，様々なライフログへの標準的かつ横断的なアクセスが可能となる．マッシュアップ・サービスはこれらマッシュアップ API を用いて開発される．ライフログへのアクセス手段がサービスに強依存しなくなるため，2.3 で述べた課題が解決できる．

先行研究 [6] では，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，サービスに強く依存するものと非依存なものに分類する．そして，サービスに依存しないデータ項目を並べて，標準データモデルの論理的なスキーマを定義している (表 1 参照)．標準データモデルでは，表 1 の When, Who, Where, How に関するデータ項目については，ライフログサービスの種類に依らず存在する標準的なデータ項目としている．一方，What, Why に関するデータはサービスに依存するため，内容を解釈せずにオリジナルデータをそのまま<content>内に保存する．このデータの解釈は，サービス毎に与えられる外部スキーマに任せるものとし，その参照を<ref_schema>で保持する．

2.5　現状の課題

標準データモデルの実用化にむけて，まだ達成されていないいくつかの課題が存在する．本研究では特に以下の未解決課題に着目する．

課題 P1：標準データモデルの物理データ設計が未決定である．例えば<date>や<time>に代入するデータは協定世界時 (UTC) にするのかそれともローカル時間 (日本標準時など) にするのか，という問題が生じてくる．

課題 P2：標準データモデルの変換が未決定である．標準データモデルとオリジナルデータのデータマッピングやデータ表現形式の変換方法を決定しなければならない．

課題 P3：標準データモデルのためのマッシュアップ API が未設計・未実装である．標準データモデルをどのように検索し，どのような形式で取得するか．またどのような API が必要で引数には何が必要かを決定し、実装しなければならない．

3. 提 案 手 法

3.1　研究の目的とスコープ

前節で述べた課題を解決するために，本研究では各ライフログ・サービスが持つオリジナルデータを標準データモデルに変換・集約し，そのデータを検索・取得するためのマッシュアップ API を実装することを目的とする．

ここで研究のスコープを明確にするため次の目標を設定する．

目標 G1：標準データモデルの物理的なデータ設計を行うこと．

目標 G2：オリジナルデータと標準データモデルのデータマッピング，データ変換方式を定義すること．

目標 G3：アクセス API の仕様を決定し，実装すること．

3.2　標準データモデルの物理データ設計

表 1 の標準データモデルの各項目について，物理的なデータ形式を決める．データ形式の決定においては，特定のサービスや実装プラットフォームに，なるべく依存しないように決定すべきである．これによって 3.1 の G1 を解決できる．

3.2.1　WHEN に関するデータ項目

<date>や<time>を表すデータ形式としては，協定世界時 (UTC) を用いることにする．ライフログサービスによっては，記録した地域のローカル時間になっていることがある．しかし，データの中立性を高めるため，標準データモデルにおいては，これらを全て UTC に変換し，保持するものとする．

- <date> YYYY-MM-DD (例：2010-03-05)
- <time> hh:mm:ssZ (例：12:34:56Z)

なお各国のローカル時間でデータを検索したいという要求に対しては，後述のマッシュアップ API の実装で補うものとする．

3.2.2　WHO に関するデータ項目

<user>を表すデータ形式としては，各ライフログサービスで用いられるアカウント名をそのまま用いることにする．アカウント名は，どのライフログサービスでも必ず存在するデータであり，また，ユーザ自ら覚えているデータであるからである．他にも，サービス固有のユーザ ID や実名などの候補が考えられるが，API によっては利用できなかったり，サービス実装に依存する形式になっていたりして，利用が困難である．<object>や<party>についても同様にアカウント名を用いることにする．

- <user> account_name (例：meruten)
- <party> account_name (例：petiTaro)
- <object> account_name (例 hirocell)

なお，同一ユーザが異なるサービスで異なるアカウント名を用いている場合には,「代表名」でつき合わせを行うための仕組みを後述のマッシュアップ API で考えるものとする．

3.2.3　WHERE に関するデータ項目

<location>を表すデータ形式としては，緯度経度情報や住所を用いることにする．緯度経度情報の場合，正確な位置をピンポイントで示すことが可能となる．一方，住所の場合，緯度経度情報よりも直感的にわかりやすく，後述のマッシュアップ API におけるクエリなどで場所を指定しやすいと推測できる．以上から<location>にはその両方を用いることにし，緯度経度情報の場合はその下の階層に<latitude>，<longitude>で表すこととする．一方，住所の場合は<geo>と表すことにした．

- <location>
  - <latitude> latitude (例：34.72631)
  - <longitude> longitude (例：135.235321)
  - <geo> address (例：兵庫県神戸市灘区六甲台町 1 − 1)

3.2.4 HOW に関するデータ項目

<application>を表すデータ形式としては，各ライフログサービス名をそのまま用いる．複数のライフログのマッシュアップを前提としていないオリジナルデータには元々データ内にサービス名が入っていない．そこで，Twitter のデータであれば application に “Twitter” と追加するようにした．

また<device>を表すデータ形式としては，使用した機器名またはアプリケーション名をそのまま用いることにする．画像を撮るのにカメラを使用したのであればそのカメラの製品名，ブログを記録するのにアプリケーションを使用したのであればそのアプリケーション名を用いる．

- <application> application_name (例：“Flickr”)
- <device> device_name (例： “Canon IXY DIGITAL L4”)

3.2.5 WHAT に関するデータ項目

<content>を表すデータ形式としては，各ライフログサービスから取得したデータを何も加工せずにそのまま用いる．これによりオリジナルデータの情報の欠落を防ぐことができる．

<content>の内容解釈はサービスごとに固有のスキーマを用いる．したがって，<ref_schema>に固有スキーマの参照 URL を用いる．

- <content> original_data (例：“<status> ··· ”)
- <ref_schema> schema_URL (例：“http:// ··· ”)

3.3 標準データモデルの変換

3.1 の G2 を達成すべく，オリジナルデータから標準データへの変換法を述べる．この変換は次の 3 つのステップで行う：

(Step 1) **データマッピングの定義**
(Step 2) **データ値の形式変換**
(Step 3) **オリジナルデータの保存**

3.3.1 Step1: データマッピングの定義

オリジナルデータの各データ項目を，表 1 の標準データモデルのどの項目へマッピングするかを定義する．通常，ライフログサービス毎に含まれるデータ項目数は異なり，同じ意味のデータでも項目名が異なる．よって，マッピング方法はライフログサービス毎に決定しなければならない．

表 2 に本研究で採用したマッピングの指針を示す．表の 1 列目の各標準データ項目に対し，オリジナルデータのどのようなデータ項目を対応させるべきかを 2 列目に書いている．3，4 列目は，マッピング例として Twitter および Flickr のオリジナルデータの項目を挙げている．

例えば，<date>および<time>については，「そのデータを生成した日 (時刻)」(例：Flickr の<dataken>) に相当するデータをマッピングする．ライフログサービスによっては「そのデータをサービスにアップロードした日 (時刻)」(例：Flickr の<FileModifyDate>) を持つものも存在し，こちらを候補にする考え方もあるがこれは採用しなかった．データ生成日 (時刻) はおおむねどのライフログサービスにも存在し，ライフログが本来持つべき実際の行動発生時間に近い時刻を表しているからである．

同様の理由で，<location>についても「そのデータを生成した場所」を基準にマッピングする．

またオリジナルデータの項目の中には，標準データの項目から見て，複数のデータが一つの項目内に存在する場合もある．Twitter の<text>がその例である．この項目の中には，“つぶやき” の他に，“そのデータを生成した場所 (イマココ!)”，“行動の対象となったユーザの情報 (@ユーザ名)” および “共に行動したユーザ (群) の情報 (#グループ名)” などが存在する．このような場合，該当するデータ項目から必要な情報のみを抽出し，正しくマッピングしなければならない．

3.3.2 Step2: データ表現形式の変換

データ項目のマッピングが決定したら，マッピング元のオリジナルデータの値を，マッピング先である標準データの形式 (3.2) で定義済) へと変換する．

例として<date>，<time>の変換例を説明する．Twitter，Flickr におけるデータの生成日・時刻は以下の形式で表現されている．

- Twitter
  <created_at>Tue Aug 25 01:42:08 +0000 2009</created_at>
- Flickr
  <datetaken>2010-01-01 09:12:53</datetaken>

これらを 3.2.1 で定義した形式に変換するため，まず日付と時刻に分割する．さらに Twitter の場合は，月を英字表記から数字表記に変換し，“年”，“月”，“日” の順に並べ替え，ハイフン “-” で結ぶ必要がある．

3.3.3 Step3: オリジナルデータの保存

オリジナルデータは<content>にコピーし保存する．またオリジナルデータを定義する外部スキーマへのリンクを取得し，<ref_scheme>に入れる．これにより情報の欠落を防ぎ，データを操作するライブラリやサービスとの互換性を保っている．

例えば，Twitter から取得されるオリジナルデータは<status>という構造体オブジェクトであるが，この内容をそのまま<content>にコピーする．<ref_scheme>には，Twitter-API 仕様書への参照 http://apiwiki.twitter.com/Twitter-REST-API-Method%3A-statuses-user_timeline を記入し，<status>のスキーマ解釈を任せる．

3.3.4 オフライン変換とオンライン変換

各ライフログサービスのオリジナルデータから標準データへの変換には，**オフライン方式**と**オンライン方式**の 2 通りの実装方法が考えられる．オフライン方式は，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．一方オンライン方式はマッシュアップ API からのリクエストがくるたびに，目的のライフログサービスにアクセスし，データを取得・変換して返す．

今回の実装では，マッシュアップ API の負荷を軽くする目的として，オフライン方式を採用した．最新データへのアクセスが制限されてしまうデメリットがあるが，マッシュアップの開発者へは高速・軽量なデータ提供が可能になる．

3.4 マッシュアップ API

標準データモデルに変換されたライフログデータにアクセスするための以下のマッシュアップ API を提案する．これらは，標準データモデルのデータ項目に関するクエリを入力とし，ク

表 2　標準データとオリジナルデータのマッピング

標準データ	オリジナルデータ	例 (Twitter)	例 (Flickr)
<date>	そのデータを生成した日付	<created_at>	<datetaken>
<time>	そのデータを生成した時刻		
<user>	主体ユーザのアカウント名	<screen_name>	<ownername>
<party>	共に行動したユーザ (群) の情報	<text>(“＃グループ名”)	——
<object>	行動の対象となったユーザの情報	<text>(“@ユーザ名”)	——
<location>	そのデータを生成した場所	<text>(“イマココ!”)	<longitude>, <latitude>
<application>	ライフログサービス名	Twitter	Flickr
<device>	ログ記録に用いたデバイス	<source>	<Model>
<content>	オリジナルデータ全て	<status>	<photo id ···>
<ref_schema>	外部スキーマへの参照 URL	http://apiwiki.twitter.com/Twitter -REST-API-Method%3A-statuses-user_timeline	http://www.flickr.com/ services/api/

エリに合致するライフログデータを標準データモデルの形式で出力するものである．

3.4.1　クエリに基づく複数ライフログの取得

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ライフログを取得する API である．

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

■引数

- 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 2 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．

上記の select を除く引数は，標準データモデルの各データ項目に対する検索条件を指定するものである．select は出力データの項目を絞るためのもので，SQL の SELECT の役割を果たす．以下にマッシュアップ API の呼び出し例を示す．クエリの表現には，perl のハッシュ表現を用いている．

使用例 1： “shimojo.akira” が 2010 年 03 月 01 日に “Twitter” と “Flickr” で記録したライフログデータを全て取得する．

```
getLifeLog({date        => ‘‘2010-03-01’’,
            user        => ‘‘shimojo.akira’’,
            application => ‘‘Twitter+Flickr’’})
```

使用例 2： 15 時台に “Flickr” で記録したライフログデータの中から，日時，ユーザ名，画像 (URL) とそのタイトルのみを抽出し，取得する．

```
getLifeLog({time        => ‘‘15:**:**’’,
            application => ‘‘Flickr’’,
            select      => ‘‘date+time+user
                             +url_s+title’’})
```

3.4.2　クエリに最も近いライフログの取得

上記の getLifeLog() は，クエリに合致する全てのライフログデータを返す汎用的なものである．しかし用途によっては，クエリに最も近いライフログデータを**一件だけ**欲しい場合も存在する．今回は日時のみに着目して，クエリに合致する日時的に最も近いデータを取得する API を考えた．将来的には，場所やユーザ，ログ行動の内容など，他の基準での最も近い検索を行う API も考えていきたい．

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

■引数

- datetime:　日時に関するクエリ
- user:　ユーザに関するクエリ
- application:　サービスに関するクエリ

■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うち，日時が最も近いデータ (1 件)．

使用例： “shimojo.akira” が “GARMIN” で記録したライフログデータの中で，“2010-03-01 12:34:56Z” に最も近いライフログデータを取得する．

```
getLifeLog({date        => ‘‘2010-03-01’’,
            time        => ‘‘12:34:56Z’’,
            user        => ‘‘shimojo.akira’’,
            application => ‘‘GARMIN’’})
```

3.5　マッシュアップ API の提供方式

提案するマッシュアップ API をどのような形で提供するかを考える．一つは，**プログラムライブラリとして配布**する方式である．開発においてはライブラリをインクルードするだけで，すぐにマッシュアップ API を呼び出せるようになる．アプリケーションからの呼び出しのオーバーヘッドも小さい．ライブラリ配布形式のデメリットとしては，バージョン更新のたびに配布し直しの必要があること，利用者の開発言語，実行環境に依存することが考えられる．

もう一つの提供方法は，マッシュアップ API を **Web サービスとして公開**することである．API を Web サーバに配置し，クエリおよびその応答はすべて XML でやり取りする．これにより，開発言語や実行環境に非依存な API 利用が可能となり，API の改訂・保守も容易となる．ただし，API 呼び出しのオーバーヘッドは大きくなる．開発者の用途，要求は様々であることから，理想的にはどちらの提供方法も実装すべきである．

4. 実　　装

4.1 標準データ変換プログラム

3.3 で提案した方式に基づいて，ライフログサービスのオリジナルデータを標準データに変換するプログラムを perl 言語で実装した．現在，Twitter，Flickr，GARMIN の 3 種類のライフログサービスに対応している．変換プログラムの規模はライフログサービスごとに異なるが，Twitter 用プログラムは約 300 行，Flick 用は約 470 行，GARMIN 用は約 160 行である．実装において用いた perl ライブラリは，XML::Simple, Net::Twitter，Time::Piece，Flickr::API 等である．

変換プログラムでは，各ライフログサービス固有の API でデータにアクセスし，提案方式で標準データモデルへと変換，XML 形式で出力する．現在オフライン変換方式で運用しており，スケジュールされた時間に変換プログラムを呼び出し，その日のライフログデータを標準形式に変換，結果を日付ごとの XML ファイルとしてリポジトリに保存している．

4.2 マッシュアップ API

3.4 で提案したマッシュアップ API を perl 言語で実装した．全体で約 400 行の規模であり，perl ライブラリとして利用可能である．リポジトリに蓄積された標準データの XML ファイルを検索し，該当するライフログデータを配列で返す．結果を XML ファイルにも出力可能である．現在このライブラリを REST-Web サービスとして公開すべく開発を進めている．

5. 評 価

提案法の有効性を確認するために，簡単なマッシュアッププログラムを作成して，開発効率性および実行効率性に関する評価を行う．ここでの例題は，Flickr で記録したユーザの写真 (動画) と Twitter で記録したつぶやきをマッシュアップし，「写真を撮影したときのつぶやき」を生成するアプリケーションを想定する．

まず，提案手法を用いない場合は，(1) Twitter，Flickr の各サービスの API を利用しオリジナルデータを取得，(2) データから必要なデータ項目だけを抽出，(3) 抽出したデータについて，日付と時刻をつき合わせてデータ統合，という 3 つのステップが必要となる．Twitter，Flickr それぞれの API の使い方の習得や，データ抽出，つき合せはすべて開発者が行わなければならない．

一方，3.4 に示すマッシュアップ API を用いた場合には，図 3 に示すシンプルなプログラムで実装できる．Twitter でも Flickr でも同じ方法でデータ検索が可能であり，検索結果に対しても同様に扱うことができるため，開発効率が大幅に向上したと考える．

```
#写真データの時間近辺のつぶやきを検索・取得し，マッシュアップする

require “mashupLib.pl”;  #マッシュアップAPIを呼び出す

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
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

#---STEP1. データの取得----------
#2010-03-01のユーザ"meruten"のFlickrのデータを取得
#抽出データ:(日付<date>, 時刻<time>, 写真URL<url_s>, タイトル<title>)
@flickr = &getLifelog({date=>"2010-03-01",user => "meruten",
      application => "Flickr",select => "date+time+user+url_s+title"});

#---STEP2. データのマッシュアップ----------
#各Flickrデータと同じ時刻(HH:MM)のユーザ"meruten"のTwitterデータを検索
#Flickrのデータ項目の中に<note>を追加
foreach my $f (@flickr){
    my $tme = $f->{'time'};
    $tme =~ s/¥d¥d$/**/;
    my $f_twitter = &getLifeLog({date=>$t->{date}, time=>$tme, user => $t->{user},
          application=>"Twitter", select => "date+time+user+url_s+title"});
    $t_flickr->{note} = $t->{text}; #Twitterの呟きをFlickrにひも付け
    push(@mashup, $t_flickr); #マッシュアップ要素とする
}

#マッシュアップデータをXML形式でファイル出力
my $xml = &outToXML(¥@mashup, "output.xml");
```

図 3　Twitter と Flickr のマッシュアップ

次に，プログラムの実行効率に関して評価を行う．現在，リポジトリには 3,600 件程度のライフログデータ<lifelog>が蓄積されている．このデータに対する図 3 のプログラムの実行時間は約 6.5 秒であり，オンラインでマッシュアップするには無視できないオーバーヘッドがかかることわかった．実行効率を上げるために，現在 XML ファイル単位で保存しているデータをデータベース化し，アクセスの高速化を図ることを検討中である．

6. おわりに

本稿では，異なるライフログサービスの集約・連携を支援するために，ライフログデータを標準データモデルに変換する手法を提案し，標準データにアクセスするためのマッシュアップ API を実装した．また，実際のマッシュアッププログラムを通して，マッシュアップ API の予備的な検証を行った．

今後の課題として，まずは対象となるライフログサービスを増やし，マッシュアップのバリエーションを広げることである．これによって，もっと大規模なマッシュアップアプリケーションに適用して有効性の評価を行いたい．また，Web サービスでの公開，データベースによる API の実行高速化も重大な課題である．

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] 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.
[4] Garmin Ltd., “Garmin”. http://connect.garmin.com/.
[5] BOstudio, Inc, “ねむログ”. http://www.nemulog.jp/.
[6] 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.