Vol.2012-DPS-152 No.8
Vol.2012-GN-85 No.8
Vol.2012-EIP-57 No.8
2012/9/13

エンドユーザを対象としたマッシュアップフレームワークの提案

野村聡太郎[†1]　　小板隆浩[†2]

本稿では，エンドユーザにとって容易なマッシュアップの実現を目的とする．PC やスマートフォンが普及し多くのエンドユーザが Web サービスを日常的に利用するに伴い，ユーザのニーズは多様化している．日常生活において一つの Web サービスのみでユーザの多様なニーズを満たすことは難しい．複数の Web サービスを連携させ，新たな Web サービスを作るマッシュアップは，多様なニーズに対処するための有効な手段である．しかし，現状のマッシュアップはプログラミングスキルや専門知識を必要とするため，コアユーザーが利用するにとどまっている．そこで，エンドユーザでも実現できるマッシュアップフレームワークを提案し，有用性を示す．

Proposal of Mashup Framework for End Users

SOTARO NOMURA[†1]　TAKAHIRO KOITA[†2]

This paper proposes a new mashup framework for end users and discusses the usability of the proposed framework. Recently, wide variety of requirements for web services arises because many users use web services for daily life. However, only one web service cannot satisfy the requirement due to the variety of requirements. To satisfy the requirement, web service mashup is important which coordinates multiple web services to meet the requirements. Mashup framework is the one of the software environment to make mashup easily for developers. The challenge of this study is to develop a new mashup framework for end users with a new mashup model and modules. The model and the modules are implemented in the proposed framework.

1. はじめに

Web サービスが普及し，PC やスマートフォンなどを通して多くのユーザが Web サービスを日常的に利用するに伴い，ユーザのニーズは多様化している．しかし，一つの Web サービスのみでは多様なニーズを満すことは難しい．多様なニーズに対して，複数の Web サービスを組み合わせ新たなサービスを作りだすマッシュアップは多様なニーズに対処するための有効な手段である．例えば，マッシュアップは旅館の口コミ情報を地図上にプロットしたいというニーズに対して，地図サービスの GoogleMaps[1]に旅館検索サービスの楽天トラベル[2]を連携することでニーズを満たすサービスを実現することができる．このようにユーザが自由に Web サービスを連携するマッシュアップを実現できればユーザ自身でユーザのニーズを満たすことが可能となる．

しかし，マッシュアップはプログラミングスキルや Web サービスの専門知識が必要であり，プログラミングスキルや専門知識を持たないユーザには実現することが難しい．本稿では，プログラミングスキルや Web サービスの専門知識を持たないユーザをエンドユーザと定義し，プログラミングスキルや専門知識を持つユーザをコアユーザと定義する．これまでマッシュアップの容易な実現を支援するための様々なマッシュアップフレームワークが実現されてきた．マッシュアップフレームワークとは Web サービスや機能毎に定義されたモジュールをモデルにしたがって連結可能とするソフトウェアの枠組みを指し，次の様な特性を持つ．

Web サービスと機能のモジュール化：
モジュールとは 1 つの処理の単位である．マッシュアップフレームワークでは必要な Web サービスや機能をモジュール化して扱う．モジュールを複数組み合わせマッシュアップを行う．モジュール化することで技術的な側面を隠蔽する．

マッシュアップのモデル化：
マッシュアップのモジュール間の連携を，マッシュアップフレームワークにおけるモデルとして表現し，単純化することで容易な理解を可能とする．

しかし，既存マッシュアップフレームワークは様々な専門知識を必要とするため，コアユーザが利用するにとどまっており，エンドユーザが利用することは難しい．本稿ではエンドユーザがマッシュアップを容易に実現可能とするようなマッシュアップフレームワークの実現を目的とする．プログラミングスキルや専門知識が必要となるマッシュアップをエンドユーザが使えるようにするためモジュールとマッシュアップモデルについて検討し，エンドユーザ向けのマッシュアップフレームワークの実現を目指す．

†1 同志社大学大学院 理工学研究科
Guraduate School of Science and Engineering, Doshisha University
†2 同志社大学 理工学部
Faculty of Science and Engineering, Doshisha University

Vol.2012-DPS-152 No.8
Vol.2012-GN-85 No.8
Vol.2012-EIP-57 No.8
2012/9/13

2 章で現状のマッシュアップと課題を述べ, 3 章でエンドユーザ向けフレームワークの設計方針を述べ，4 章で提案フレームワークの概要について述べ，5 章でまとめる．

2. 既存のマッシュアップと課題

既存のマッシュアップでは，コアユーザが開発したものを不特定多数のエンドユーザが利用することが多い．エンドユーザにとって Web が身近になり, ニーズが多様化する中，今後はエンドユーザも手軽にマッシュアップできることが望ましい．手軽なマッシュアップとはエンドユーザが提供されるモジュールを用いて，容易にニーズを満たすサービスを作り出すことと定義する．手軽なマッシュアップを実現するためには，以下の 2 点が重要となる．

- プログラミングスキルや専門知識を必要としない手軽さを有していること
- 様々なサービスを連携できる柔軟性を有していること

次節以降，既存のマッシュアップの課題について述べる．

2.1 既存のマッシュアップフレームワーク

代表的なマッシュアップフレームワークの例として，Plagger[3]，Yahoo!Pipes[4]，MicroSoftPopfly[5]，Intel@MashMaker[6]，iGoogle[7]などが挙げられる．図 1 に既存のマッシュアップフレームワークの分類を示す．図 1 では，既存のマッシュアップフレームワークを手軽さと柔軟性の観点から分類している．

例えば, Plagger は様々な機能を持った Perl モジュールを組み合わせ，マッシュアップを行う．モジュールの連結には Perl を用いてプログラミングで行うプログラミングモデルである．プログラミングモデルでモジュールの連携を実現するため柔軟性が高いが，十分なプログラミングスキルと専門知識が必要となり，エンドユーザにとっては利用が難しい．

また， Plagger とは異なり，柔軟性は低いが GUI ベースでプログラミングスキルを必要とせずマッシュアップが行えるフレームワークも実現されている．Yahoo!Pipes は，RSS フィードの読み込みや正規表現，並び替えなどの操作や Web サービスの操作をおこなう様々なモジュールの処理の入出力をパイプで繋ぎマッシュアップを行うデータフローモデルのマッシュアップフレームワークである．

Yahoo!Pipes と同様に Microsoft Popfly は Web サービスを 1 つのモジュールとして扱い，モジュールの処理の入出力を繋ぐデータフローモデルを採用している．これらのデータフローモデルのマッシュアップフレームワークは手軽さと柔軟さを適度に実現している．しかし，文献[8]で指摘されているように，エンドユーザは入出力の整合性を考慮する必要があり，データフローの概念を理解するためにはある程度のプログラミングスキルや専門知識が必要となる．

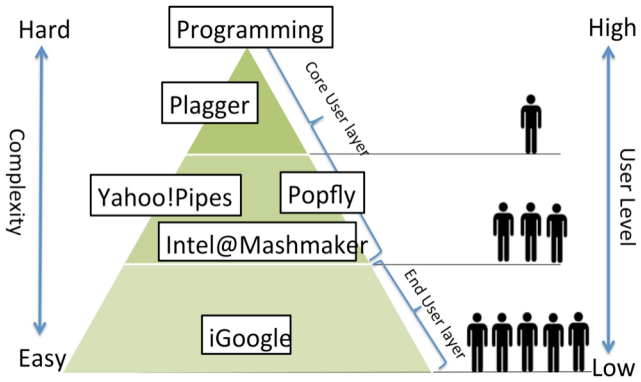

図 1：マッシュアップフレームワークの分類

また，データフローモデルとは異なるフレームワークを利用したマッシュアップツールとして Intel@MashMaker が挙げられる．Interl@MashMaker では 1 つの Web サービスをモジュールとして扱い, 現在閲覧している Web ページの情報に対しモジュールを埋め込むことができるガジェットモデルのマッシュアップフレームワークである．しかし，現在閲覧しているページの構造について理解が必要であり，エンドユーザが利用するには難しい．同様なモデルを採用している iGoogle は Web サービスを 1 つのモジュールとして扱い，1 つのページに表示するガジェットモデルで，モジュールをドラック＆ドロップするだけなので非常に手軽ではあるが，あらかじめコアユーザが開発したモジュールしか利用できず，エンドユーザが様々なサービスを柔軟に連携させることは難しい．

このように，様々なマッシュアップフレームワークが実現されているが，プログラミングや専門知識が必要となるか，利用が限られてしまうため，エンドユーザが利用することが難しい．

2.2 既存のマッシュアップフレームワークの課題

既存のマッシュアップフレームワークは，モジュールの詳細な設定部分や連携方式などのユーザインタフェースにおいてプログラミングスキルや専門知識を必要とするため難しいものとなり，また，様々な機能を実現する柔軟性を求めてもエンドユーザにとって難しいものとなり，コアユーザが利用するにとどまっているのが現状である．目標とするフレームワークの課題はエンドユーザが手軽に使える範囲で，これらのトレードオフを十分に考慮し，最大限の柔軟性を持たせることである．次章でその方針について述べる．

3. 設計方針

エンドユーザ向けマッシュアップフレームワークの実

Vol.2012-DPS-152 No.8
Vol.2012-GN-85 No.8
Vol.2012-EIP-57 No.8
2012/9/13

現にあたり，必要要件について述べる．既存のマッシュアップフレームワークの課題とユーザインタフェースに必要とされる要件[9]を参考にし，以下の必要要件を挙げる．

シンプルなデザイン：
インタフェースにおいて情報を詰め込みすぎないことが望ましい．余分な情報が関連する情報と競合して，相対的に視認性を減少させてしまう．モジュールの詳細な設定部分やインタフェースはシンプルなものとすることが望ましい．

システムと実世界の調和：
システムはシステム指向の表現ではなく，ユーザに馴染みのある表現，用語，コンセプトを用いて，ユーザが理解できる表現を用いることが望ましい．

解りやすいユーザビリティ：
プログラミングを必要としない環境を作るため，システム利用のためのオブジェクトや動作，オプションは可視化して，操作方法や情報を覚えなくても，見てわかるようになっていることが望ましい．モジュールの連携方式においてはプログラミングモデルやデータフローモデルは避けることが望ましい．

4. 提案フレームワーク

提案するフレームワークについて概要を述べる．提案するフレームワークでは，手軽さと柔軟性を実現するため，イベント駆動型モデルを導入する．既存のマッシュアップフレームワークでは，全てのサービスに対応するために，エンドユーザにとって複雑なモデルとなっているが，エンドユーザが最も多く利用するマッシュアップでは，イベント駆動型モデルによりエンドユーザ向けの実世界との調和を実現し，ほとんどのマッシュアップに対応することが可能である．また，シンプルなデザインと解りやすいユーザビリティを実現できる．以下に，必要要件とイベント駆動型モデルの関連について述べる．

4.1 シンプルなデザイン

既存のマッシュアップフレームワークは専門的なパラメータ設定など専門知識が必要となりエンドユーザにとって難しい．イベント駆動型モデルでは，すべてイベントを中心にモデルを構築するため，ごく少数のイベントとパラメータのみでモデルを構築することが可能である．本フレームワークではモジュールのモデルを次のように定義する．各モジュールは独立した 1 つの Web サービスを扱っており，モジュールは以下のインタフェースを提供する．

図 2：フレームワーク概要

ActiveEvent
各モジュールの機能を通して起こるそのモジュールの状態の変化を指す．この状態の変化に応じて他のモジュールに対してメッセージを送る．また，メッセージと同時に結果となるデータも送る．

PassiveAction
自身以外のモジュールから何らかのメッセージを受け取って起こす自身の機能を指す．メッセージと共にモジュールの機能を起動させるためのデータを含む．

Rule
ActiveEvent，PassiveAction に付随するルールである．例えば，天気を扱うサービスのであれば，天気情報を表示するという ActiveEvent に対して，晴れの時，雨の時などを決める．

これらのインタフェースのみでエンドユーザが利用する多くのマッシュアップに対応可能となる．また，GUI を通して各モジュール間の ActiveEvent と PassiveAction を連携させることにより，ユーザに解りやすいインターフェースも実現しやすくなる．図 2 にフレームワークの概要を示す．

4.2 システムと実世界の調和

従来のマッシュアップフレームワークではデータ指向の表現を用いているものが多かった．しかし，本フレームワークのモジュールではイベント駆動型のモデルを中心として，ActiveEvent と PassiveAction がそれぞれ Web サービスを連携させる．例えば天気モジュールであればイベントとして「雨の時」「晴れの時」といった実世界に近い表現が可能であり，メールモジュールでも「メールを送る」というエンドユーザが理解しやすい表現を用いることができる．

Vol.2012-DPS-152 No.8
Vol.2012-GN-85 No.8
Vol.2012-EIP-57 No.8
2012/9/13

4.2 解かりやすいユーザビリティ

提案フレームワークでは，イベントとそれに関連するモジュールのみがユーザに提供される．実際のフレームワークでは，各モジュールはそれぞれのサービスに対応したイベントやモジュールアイコンをドラック＆ドロップして，ActiveEvent と PassiveAction をパイプで繋ぎ,マッシュアップを画面に表示される．

4.5 提案フレームワークの概要

提案フレームワークの構成について述べる．提案フレームワークは，イベントプロデューサー，イベントコンシューマ，イベントブローカーの 3 つの要素から成る．

イベントプロデューサー：
ActiveEvent が発行を管理する.例えば、送信ボタンや結果を表示した時などにイベントの発行を行う．

イベントコンシューマ：
イベントプロデューサーが発行したイベントの内容をもとに様々な処理を行う．

イベントブローカー：
各モジュールで発行されるイベントを分類しモジュールの組み合わせ判定を行う．

Web サービスはあらかじめイベントプロデューサーに登録されて，マッシュアップはイベントプロデューサーを中心に実現される. イベントプロデューサーは Web サービスのイベントを監視し，イベントが発生するとイベントブローカーにイベント発生を通知する．イベントブローカーは，通知されたイベントの条件や種類を判別し，必要ならばイベントコンシューマに通知する．最終的に，イベントコンシューマは，受け取ったイベント通知をもとに Web サービスのモジュール同士を連携させ，マッシュアップを実現する．

4.4 利用シナリオ

複数のモジュールを繋げて作るマッシュアップの例を挙げる．例えば，株の売買をサポートするマッシュアップを作るとする．サービスのモジュールとして株サービスのモジュールとメールサービスのモジュールを用意する．株サービスモジュールは 1 日の出来高がある閾値を超えるか超えないかを ActiveEvent として保持する．一方，メールモジュールには PassiveAction としてメールを送る機能を有する．株サービスモジュールにおいて，出来高が閾値を超えた場合，イベントが発生し，メールモジュールにイベントが通知される．通知されたイベントを受けて，メールモジュールではメールを送る．既存のマッシュアップフレームワークでは，利用されるデータやそれぞれの機能に対する知識が必要となるが，提案フレームワークではあらかじめ用意されたイベントのみを理解することによりマッシュアップを実現可能である．また，例えば，旅行をサポートするマッシュアップを作成する場合，旅行検索サービスモジュールとマップサービスモジュールを用意する．旅行検索モジュールは宿泊施設を検索することを ActiveEvent として有する．また，マップサービスモジュールでは結果を表示するという PassiveAction を保持する. この場合も同様に，検索というイベントを中心にマッシュアップを実現可能である．

このように，各モジュールの ActiveEvent と PassiveEvent を繋げることで簡単にマッシュアップをすることができ，また，各モジュールを組み替えることによって，エンドユーザでも手軽に柔軟なマッシュアップを実現することが可能である．

5. まとめと今後

エンドユーザが Web サービスを日常的に利用するに伴い個人のニーズは多様化している．多様なニーズに対してマッシュアップは有効であるが，マッシュアップは現状ではコアユーザが利用するにとどまっている．本稿ではエンドユーザが Web サービスごとのイベントやアクションを繋ぐことによって Web サービスを組み合わせ, 容易に利用できるようなイベント駆動型のマッシュアップフレームワークの提案をした．今後，開発環境の実装と共に，エンドユーザにとって必要となるサービスと，既存サービスにおけるイベント種別の検討を行い，ユーザビリティや性能面の評価を行う．

参考文献

1) GoogleMaps, https://maps.google.com/
2) 楽天トラベル, http://travel.rakuten.co.jp/
3) Plagger, http://plagger.org/
4) Yahoo!pipes, http://pipes.yahoo.com/pipes/
5) Microsoft Popfly, http://www.popfly.ms/
6) Intel@Mashmaker, http://software.intel.com/en-us/articles/intel-mash-maker-mashups-for-the-masses/
7) iGoogle, http://www.google.co.jp/
8) J. Wong and J. I. Hong, Making Mashups with Marmite: Towards End-user Programming for the Web, In the proc. of the 2007 conference on Human factors in computing systems (CHI), pp. 1435-1444, 2007.
9) ユーザビリティエンジニアリング–ユーザ調査とユーザビリティ評価実践テクニック，オーム社, 2005.