ソーシャルアプローチによる業務システム再構成手法の提案

粂照宣[†]　田中昌弘[†]　松尾昭彦[†]

ビジネスの変化に応じた業務システムが求められているが，既存システムを変更することは容易ではない．そこで，利用者の操作ログを用いたWebAPI自動作成手法とソーシャルアプローチを用いた操作ログ解析手法の組合せにより，既存システムの機能を活用した業務システムの構築を容易にする手法について提案する．

Restructuring Business System By Using Social Approach

Terunobu Kume[†] Masahiro Tanaka[†] and Akihiko Matsuo[†]

Nowadays, business environment has been changing quickly and dynamically. And it is necessary to change to business systems with this change. But it is difficult to change existing business system. So, We propose that automatically generate a WebAPI by using user's operation. And moreover We describe that how to generate useful WebAPI for user by using social approach.

1. はじめに

企業を取り巻くビジネス環境はめまぐるしく変化しており，そして，ビジネス環境にあわせて業務も変化し続けている．この変化する業務に迅速に対応するためにも，変化にあわせた業務システムを用意していく必要がある．しかし，変化のスピードが早すぎるために，一からシステム設計・開発を行っていたのでは，業務システムを立ち上げるまでの時間がかかりすぎ迅速に対応することができない．

Web 全盛の時代となり，企業システムを Web アプリケーションとして作成するのが当たり前となっている現在，業務システムを変化に柔軟に対応させるために，開発を行わずに既存の業務システムを組み合わせ，新たな業務システムを生み出すマッシュアップという技法が注目されている．その中でも特に企業内の業務システムを使いマッシュアップすることをエンタープライズマッシュアップと呼び，クラウド時代の新しいプログラミングスタイルとして期待されている．

このエンタープライズマッシュアップを実現するためには，業務システムが各機能をWeb経由で呼び出せるようなWebAPIと言われる外部インターフェースを持っている必要がある．しかしながら，既存の業務システムは古くから利用されているものが多く，ほとんどの業務システムにはWebAPIが用意されていない．エンタープライズマッシュアップを実現するためには，既存業務システムにWebAPIを追加する必要がある．しかし，既存業務システムに対して，簡単に新たな機能を追加することは難しい．そこで，既存業務システムに手を加えることなくWebAPIを追加し，エンタープライズマッシュアップを実現できるようにする方法が必要となっている．

本稿では，この問題を解決するために，利用者が業務実行時に送信する通信リクエストとレスポンス(操作ログ)を用いて，既存業務システムに変更を加えることなくWebAPIを半自動的に作成するWebAPI自動作成手法を提案する．また，提案手法を利用者にとってより使いやすいものにするための手法として，ソーシャルアプローチを用いた利用者の操作ログの分析方法について提案する．

2. 課題

業務システムは業務に関連する情報を効率的に取得するために活用されており，ビジネス環境の変化にあわせて業務システムを変化させていくことは，ビジネスを効率的かつ円滑に進める上で重要なものである．しかしながら，近年のめまぐるしいビジネス環境の変化が早く，既存業務システムの改修が間に合わないという状況であり，ビジネスに支障をきたしている．そこで，迅速に変化に対応するために，業務システムにマッシュアップ手法の適用することで，既存システムを活用しながら変化にあわ

[†] 株式会社富士通研究所
FUJITSU LABORATORIES LTD.

せた業務システムを構築することが期待されている．このような業務システムにマッシュアップ手法を用いることをエンタープライズマッシュアップと呼んでいる．しかし，エンタープライズマッシュアップを行うためには，対象業務システムに WebAPI が必要であり，既存業務システムには WebAPI が無いということが問題となっている．

そこで，既存業務システムに WebAPI を作成し，エンタープライズマッシュアップを実現可能にすることが課題となっている．

3. 関連研究

エンタープライズマッシュアップはビジネスの世界でも注目を集めており，調査会社の Forrester Research によれば，2013 年までに 7 億ドル市場に成長すると見込まれている[1]．2009 年には，Open Mashup Alliance(OMA)[2]という企業向けマッシュアップ業界団体も発足し，エンタープライズマッシュアップのための記述言語として Enterprise Mashup Markup language(EMML)を策定し，そのリファレンス実装も公開されている．また，別の団体として Enterprise MashUps(EMU)[3]という団体も存在し，Enterprise Mashup Summit という会合を毎年開催し盛況を集めている．

一方，研究分野では Hoyer.V らがロングテール的な利用者のニーズをエンタープライズマッシュアップにより解決することを提案している[4]．ロングテール的な利用者のニーズとは，業務システムに対して誰もが要求する機能ではなく，一部の限定された利用者が業務システムに対して修正を要求するようなものを意味している．つまり，大多数の利用者が要求するものであれば，業務システムを修正する費用対効果は高くなるが，一部の利用者のみの場合には費用対効果が低くなる．そこで，このような要求に対応する場合にエンタープライズマッシュアップを利用するとで開発費用を抑えることができ，費用対効果が高くなる上に素早く要求を実現することができることを報告している．

また，Jhingran.A も同様に，既存の業務システムだけでは，目的にあった情報を取得することができないことを課題とし，既存システムに対してエンタープライズマッシュアップ手法を適用し，既存業務システムを組合せて必要な情報を取得するための方法について報告している．[5]

市場動向や関連研究でも示される通り，エンタープライズマッシュアップは企業がビジネスを推進していく上で，既存業務システムに対して，追加開発を行わずに目的の業務システムを素早く作りあげることができるので注目を集めているが，どの報告においても，業務システムに既に WebAPI が存在していることを前提としており，本稿の課題で述べたような既存業務システムの問題点については言及していない．

そこで，本稿では，既存業務システムに外部インターフェースである WebAPI を構築する手法について議論する．

4. 従来手法

既存業務システムに WebAPI を作成する方法として，三つの方法が考えられる．

1. 既存業務システムを修正する
2. システムが持っているデータを直接参照・更新する
3. 利用者が画面から行う操作を再現し，結果を画面から得る

まず，既存業務システムを修正して WebAPI を作成する場合には，設計・開発・テストというフェーズを実施する必要があり，修正を行うために時間とコストがかかる．また，業務システムの一部として外部サービスを利用している場合があり，直接システムを変更することができないこともある．次に，システムが持っているデータを直接参照・更新する場合には，データベーススキーマの統合や既存業務システムへの影響を検証する必要や，既存業務システムに対して修正が必要となるために，どちらの方法を使っても多くの時間とコストが必要となることが問題である．

そこで，利用者の画面操作を再現し，目的の結果を得るという方法が既存業務システムに影響を及ぼすことがなく，最もマッシュアップに適した方法であると言える．

図 1　既存業務システムの課題

しかしながら，利用者が画面から行う操作といっても，一つの業務を実行する際の画面遷移と各画面で利用者が実施する画面操作は多岐にわたり，これらの処理をプログラムで再現することは非常に困難である．そこで，このような，業務実行中の画面遷移と各画面での操作を自動的に記録し，その記録を元に処理を再現するような仕組みが必要となってくる．

5. WebAPI自動作成システム

本稿では利用者が業務を実行した際の通信データを記録し実行可能にする手法を提案する．この手法では業務を実行した際にブラウザからサーバへ送信される通信データを記録・解析し，実行可能にすることで簡単に実行結果を得られるようになる．

5.1 通信データ記録方法

通信データを記録するために，図 2 に示すような，ブラウザから送信されるリクエストと業務システムから送られてくるレスポンスを記録するリクエスト中継サーバを設置し，業務システムを実行する際には必ずリクエスト中継サーバを経由するようにする．

図 2 通信記録取得方法

これにより，リクエスト中継サーバで全てのリクエストを記録することができる．リクエスト中継サーバは，ほぼ Proxy と同様で利用者端末から送信されてきたリクエストを業務システムにそのまま送信し，業務システムから送られてきたレスポンスをそのまま利用者端末に送信する．その際に，中継した全てのリクエストを記録し，これを部品として蓄積しておく．

5.2 操作再現方法

部品化された処理は，エンタープライズマッシュアップを行う際の WebAPI として活用され，図 3 に示すように，単純に部品として蓄積されている操作記録を呼び出すだけで，部品実行サーバが処理を行い目的の結果を取得できるようになる．

図 3 部品実行処理

これにより，エンタープライズマッシュアップを実現するために必要な WebAPI を既存システムに対して修正を行うことなく，簡単に作成することが可能となる．

5.3 業務システム特有の問題

前述の方法を使い業務システムの操作を記録すれば，目的の業務を簡単に部品化し活用することができる．しかしながら，業務システムには以下のような特徴があり，単純に利用者の通信データを再実行することができない．

1. 認証・認可が必要なシステムである
2. セッション管理が含まれている
3. 次の画面のためにシステムが自動で埋め込む値がある

例えば，認証・認可が必要なシステムを実行した操作記録の場合，記録している通信データには，記録した利用者の認証情報などが含まれているために，本来見えてはいけないデータが結果に含まれてしまう可能性がある．そこで，認証・認可については記録した通信データから削除し，部品を呼び出した利用者のものに変更する必要がある．また，セッション管理があるシステムを実行した場合の操作記録では，記録時のセッション情報は無効になっているため，再実行時に改めて取得し，そのセッション情報を記録した全操作を実行する間利用する必要がある．さらに，ユーザが入力したものではなく,次の画面に必要な値を業務システムが自動的に埋め込む場合がある．本稿では，この自動的に埋め込まれた値をシステム固有値と呼ぶが，この値が実行の度に変化する場合があり，業務システムから戻ってくるレスポンスデータを分析し，その値を抽出する必要がある．

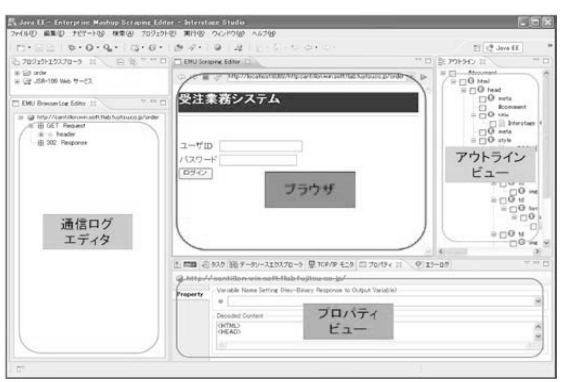

図 4 利用者 UI

この様な業務システム特有の問題を解決するために，図 4 に示すような UI を持ったクライアントアプリケーションを作成し，利用者自身で認証情報などを排除できるようにした．実際に WebAPI を作成するために，まず，このアプリケーションを起動し，中央にあるブラウザを使いながら業務を実行することで業務システムの必要部分を部品化することができる．次に，部品化されたデータを通信ログエディタ，プロパティビューを使い，認証情報，セッション情報などを選択しながら，ユーザ固有部分のデータを削除し，部品実行の際に利用者に要求する項目を利用者自身が選定していく．最後に，システム固有値をアウトラインビューを使い，記録している直前のレスポンスに含まれているデータから，どの値を利用するかを設定することで，記録した通信データを汎用化することができる．

5.4 リッチユーザインターフェース対応

近年の業務システムでは，利用者が使いやすくなるように UI に様々な手法が用いられており，その実現方法として Ajax が様々な業務システムに適用されている．このようなシステムでは，非同期通信により，画面とデータが分離されて送られてくる場合があり，データ形式も XML や JSON などの形式がある．本提案システムではこれを考慮し，非同期通信も記録するようにしている．記録した画面とデータのそれぞれを比較し，表示位置が特定可能な JSON 形式については，JSON データ選択時に画面のどの部分に表示されているものかを自動的に判定し利用者に提示することができる．この機能により，データと画面が分離されて送信されるような場合でも，利用者は目的のデータを簡単に抽出できる．

また，セッション管理などに良く用いられる Cookie については再実行時に自動的にシステムが抽出するようにしている．Cookie は通常 HTTPD のヘッダとして送信されてくるが，近年，JavaScript が多用されるようになり，JavaScript の中で Cookie を操作している場合がある．これに対しても判定可能なものを抽出することで，利用者の操作をなるべく簡略にするようにしている．

5.5 処理概要

今回提案するシステムのシステム構成を図 5 に示す．WebAPI の作成方法として，まず，利用者 A が部品化したい処理を実行する(1)．この際，常にリクエスト中継サーバを経由させることで，この利用者 A が操作を実行した際に発生する通信データおよびレスポンスを(2)で記録する．次に，記録データを(3)で実行した利用者 A が UI を使い汎用化し部品実行サーバに配置する(4)．以上の手順で既存業務システムに疑似的な WebAPI が作成される．利用者 B は部品実行サーバにある部品(4)にアクセスすると記録された操作を部品実行サーバが実行し同じ操作を実行した結果を取得することができ，この結果を他の業務システムとマッシュアップすることが可能となる．

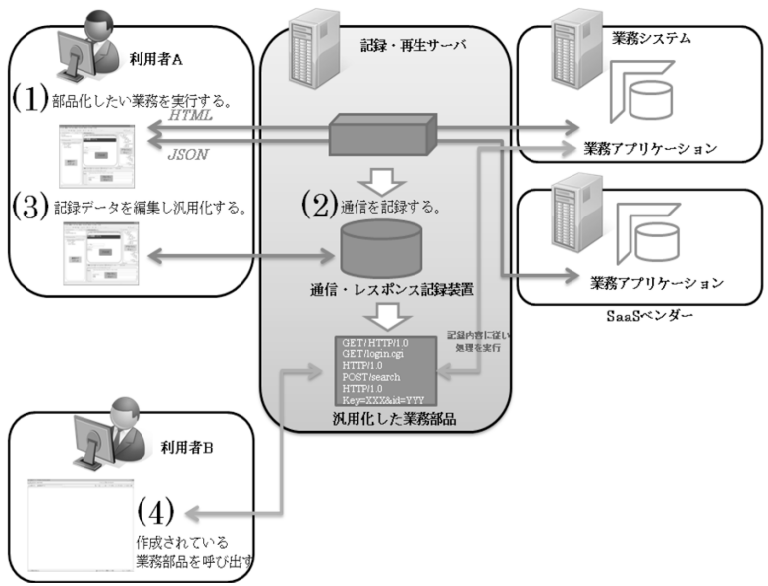

図 5 処理概要

6. 実験

本提案手法を用いて，実際の業務システムに対して WebAPI を作成する実験を行った．今回，実験対象とした業務システムは，財産番号管理システムである．このシステムは，企業が購入したパソコンなどの管理者，設置場所，購入金額，購入日などの情報を管理するためのシステムである．このシステムに対して財産番号から検索することで，購入品の管理者の情報取得することができる WebAPI を作成する．また，本手法を用いずに，同様の処理を手動で分析しプログラム化することで WebAPI 作成した場合と，本手法を用いた場合との作成時間の比較を行った．

表 1 実行時間

	提案手法	従来手法
時間(min)	20	240

表 1 に示した通り，今回提案した手法を用いた場合と従来通りにシステムに対して WebAPI を作成した場合を比較したところ，今回提案した手法を用いることで，WebAPI の開発時間を 1/10 以下に短縮することができた．

6.1 実験で判明した課題

本提案手法を用いることで，プログラムで WebAPI 作成を行うよりも開発時間を短縮できることが確認できた．しかし，WebAPI を作成しようとする業務システムによっては利用者はある程度システムの仕様を理解していないと汎用化することができない場合があった．

```
<html>
<script>
function jump() {
document.form.submit();
}
</script>
<body onload=jump();>
<form action=/menu/show.cgi method=get>
<input type=hidden name=identify value=U0001/>
</form>
</body>
</html>
```

図 6 レスポンスサンプル

提案手法では，システムが自動的に埋め込むシステム固有値を利用者に判断してもらうことで汎用的な業務部品を作成しているが，実際にシステムを利用するだけの利用者は，そのような値があることすら認識していない．そのため認証に関連する情報や画面の入力項目を汎用化することはできても，業務システムが次の処理のために利用しているシステム固有値などを汎用化することができないために，動作しない部品となってしまう場合がある．例えば，業務処理実行中に図 6 に示すようなレスポンスが返ってきた場合に，レスポンスに含まれている<input type=hidden name=identify value=U0001/>の”U0001”がシステム固有値であり状態により変化する値である．この場合ではシステムが利用者を識別するための値となっていて，その後の処理を実行するために利用されている．しかし，利用者にとっては自分が入力した値ではない上に，画面に表示されるものではない．そのために，そのデータの存在に気づかず，汎用化処理を行わない．その結果，汎用化後も記録時の値になってしまい部品が正しく動作しなくなってしまう．そこで，自動的にこのようなシステム固有値を判断するための仕組みが必要となる．

6.2 複数回実行による自動部品化

システム固有値を自動的に判断するためには一回の操作だけでは判断することができない，同様の処理を二回以上実行してもらい，その二つのリクエストとレスポンスを比較評価することで，一回目と二回目の違いから利用者が入力した値とシステム固有値を機械的に判定できるようにする．

実際の処理例として，図 7 に示すような，二人の利用者が同様の業務を実行した際の二つのリクエスト記録を比較する．

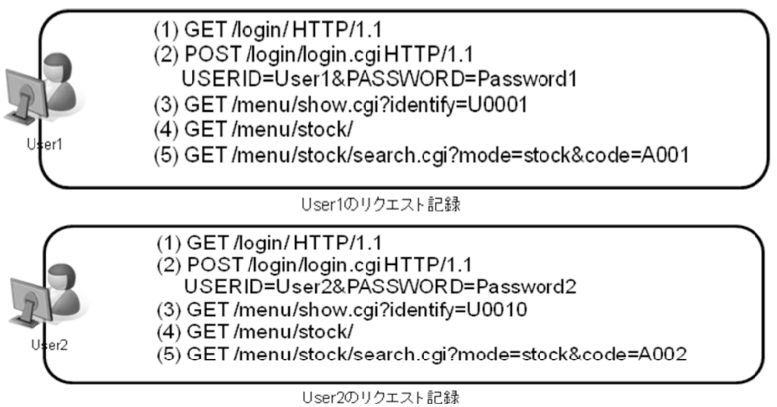

図 7 リクエストデータサンプル

二つのリクエストは同じ業務を実行しているので，基本的には同じリクエストデータとなるはずである．そこで，この二つのリクエストを使い，システム固有値，及び，利用者が入力した値を特定する手法について説明する．

まず，二つのリクエスト比較し，アクセスしている URI に違いがないことを確認する．次に，各リクエストからパラメータ名と値を抽出しておく．抽出したパラメータ名とパラメータ数を比較し違いがないことを確認する．

最後に各パラメータ値の比較を行う．パラメータ値に違いがない場合は，その値を固定値と判断し，違う場合にはシステム固有値もしくは利用者が入力したものであると判断する．

システム固有値もしくは利用者が入力した値と判断した場合には，該当リクエストを送信する直前のレスポンスを確認する．レスポンス中に該当パラメータ名で該当パラメータ値が存在する場合には，その値はシステム固有値であると判断し，再現する際には，直前のレスポンスから値を抽出するようにする．また，直前のレスポンスに該当パラメータ値を発見できなかった場合には，利用者が入力したものであると判断し，再現する際に利用者に対して値の入力を求めるようにする．

以上の手順を全てのリクエストに対して実行することにより図 8 に示すような汎用的な部品が作成される．

```
(1) GET /login/ HTTP/1.1
(2) POST /login/login.cgi HTTP/1.1
    USERID=変数1&PASSWORD=変数2
(3) GET /menu/show.cgi?identify="レスポンスから抽出"
(4) GET /menu/stock/
(5) GET /menu/stock/search.cgi?mode=stock&code=変数3
```

図 8 汎用化部品サンプル

これにより，利用者は記録された通信データを処理することなく自動的に汎用化部品を作成することができるようになる．

7. ソーシャルアプローチによる部品化

WebAPI 自動作成システムおよび自動汎用化手法により，完全自動で部品を作成できるようになる．そこで，本 WebAPI 作成システムを企業全体に適用し，全ての利用者が実行する業務処理操作を記録していくようにする．記録された情報を分析し，その中で多くの利用者が実行している共通部分を抽出し，汎用部品化することで，利用者の利用目的にあった部品を作成することができるようになる．また，抽出の方法を工夫し，部門などの単位でまとめることで，その部門の利用者が実行する際に必ず必要となる部門名などの情報を自動的に埋め込んだ形の部品を作成することも可能になる．つまり，本ソーシャルアプローチを用いることで，業務システムに必要となる WebAPI を簡単に作成できると同時に Hoyer.V らが報告したような，一部の利用者のみが必要とするような WebAPI も作成することが可能となる．

8. まとめと今後の課題

本稿では，利用者のリクエスト記録とレスポンスデータを使い，システム修正を行うことのできない既存の業務システムに対して，簡単に WebAPI を作成する手順とその手順の有効性についての実験を行った．また，実験の中で発生した，利用者がシステム固有値まで意識して操作を行っていないという問題に対して，複数の操作記録を使い機械的にシステム固有値を発見する手順を示した．最後に，ソーシャル的なアプローチを用いることで，より柔軟な業務システムを構築する方法を提案した．

今後は，本提案手法を様々な業務システムに適用し，その有効性について検証を行っていく予定である．

参考文献

1) G.Oliver Yang Ellen,D Mike,G Heidi,L Madiha,A: The Mashup Opportunity, Forrester Research, http://www.forrester.com/rb/Research/mashup_opportunity/q/id/44213/t/2 (2008).
2) Open Mashup Alliance(OMA), http://www.openmashup.org/.
3) Enterprise Mashups(EMU), http://emusummit.com/
4) Hoyer.V, Stanoesvka-Slabeva.K, Janner.T and Schroth.C: Enterprise Mashups: Design Principles towards the Long Tail of User Needs, Proceeding of IEEE International Conference on Service Computing, pp 601-602(2008)
5) Jhingran.A: Enterprise Information Mashups: Integrating Information, Simply, Proceedings of the 32nd international conference on Very large data bases table of contents, pp3-4(2006)