「マルチメディア，分散，協調とモバイル
(DICOMO2014)シンポジウム」 平成26年7月

既存 Web アプリケーションの操作記録を用いたスマート端末向け Web アプリケーション生成方式

大木憲二†1　　栗原英俊†1　　山本里枝子†1

スマートフォンなどのスマート端末から PC 向けに構築済みの既存 Web システムを使う機会が増えてきているが，スマート端末の PC とは異なる特性のためにそのままでは満足する操作性を得られないことが多い．一方，スマート端末に対応するために PC 向けに正常に動いている既存システムに手を入れることには，リグレッション発生などのリスクを伴う．

そこで本論文では，ブラウザから既存システムを操作した際に得られる HTTP 通信や画面の記録情報を用いて，既存システムの修正を行うことなしに，スマート端末向け Web アプリケーションを生成する手法を提案する．本手法では，何度アクセスしても変わらない項目については静的な HTML コンテンツとして生成を行う事でキャッシュが効くようになる．また，利用者とのインタラクションを行う必要が無い処理を，単一のアダプタに集約して既存システムへアクセスすることで，通信数の削減による効率性向上効果や，画面数の削減による利用者の入力手番数削減効果が得られる．これらの効果によってスマート端末により適した状態で既存システムの利用が可能となる．

1. はじめに

近年，スマートフォンやタブレットなどのスマート端末が急速に普及してきている．企業内においてもスマート端末が持つ高機能性やその可搬性を活かす事でワークスタイルの変革が期待されており，導入が進みつつある．

企業においてスマート端末を利用する際には，企業内のシステムへアクセスするケースが頻繁に生じる．例えば，営業担当者が対面商談において企業内システムを用いてその場で見積書を作成して顧客に対して表示するケースや会社に戻れない際に企業内システムを用いて，出張先や自宅から出張報告を行うケースなどがある．

このような利用ケースを実現するために，企業内システムをスマート端末に対応するための開発要求が高まっているが，この開発に際して次のような問題がある．

(1) **スマート端末利用者に適した操作性**

スマート端末は, 従来のデスクトップ PC やノート PC と比較して入力操作の困難さや画面サイズの小ささ，通信の不安定さといった様々な制約を抱えている．多くの既存の企業内システムはデスクトップ PC やノート PC からアクセスする事を前提として作られているため，スマート端末の制約を考慮しないと使い勝手の悪いシステムとなり，スマート端末を用いるメリットが損なわれてしまう．

また，スマート端末向けのアプリケーションを開発する際のコンセプトとしてモバイルファーストが謳われつつある[]．モバイルファーストにおいては，PC よりもモバイルからの利用を想定してシステムを開発するという手段の観点だけではなく，その際に利用者が本当に必要とすることにフォーカスすることでモバイルの制約を乗り越えたり，モバイル固有の機能を積極活用したりする事の必要性について述べている．これは PC 向け既存システムがあった場合においても，スマート端末対応を行った際に守られるべき考え方である．

(2) **既存システム改修の困難さ**

デスクトップ PC やノート PC 向けに正常稼働している既存システムに対して，新たな機能要求もなくスマート端末対応のためにこれらを改修することは無駄な工数の発生や改修によるリグレッションの発生などのリスクが伴う．現在稼動している環境を極力変えないまま，スマート端末対応させることが求められる．

(1)の問題への対応として, 既存の Web アプリケーションの画面をスマート端末に適した画面に変換するサービスが既に存在している[6][7]．これらのサービスでは，HTML で記述された単一の画面を事前或いはアクセス時に変換する方式を採っているが，それ以上の最適化を行おうとするとサーバ側の改修が必要となってしまう．

(2)について，既存の企業内システムに手を入れられない事が同様に問題となる分野としてエンタープライズ・マッシュアップが挙げられる[2]．マッシュアップとは既存の Web サービスや Web アプリケーションのコンテンツを組み合わせて新たなシステムを開発する手法であり，一から開発する場合に比べてコストの削減が図れる．エンタープライズ・マッシュアップではその対象を企業内の Web システムにも広げる事で企業内の情報を統合する事を目的としている．エンタープライズ・マッシュアップにおいては，SOAP や REST などのマッシュアップを行うための API が用意されてなく, 認証や画面遷移などを伴う既存 Web システムを再利用したい場合の対応方法が課題となっている．

†1　(株)富士通研究所
Fujitsu Laboratories Ltd.

この解決策として既存の Web システムの前に Web アダプタを配備する方法がある．アダプタ上で既存の Web システムへリクエスト送信処理やレスポンス加工処理を行うことで，マッシュアップを行うクライアントアプリケーションに対して扱いやすいインタフェースを，既存 Web システムに手を加えることなく提供することが可能となる．アダプタは，本来既存 Web システムに対してブラウザが発生させる通信を再現する必要があるため開発に手間が掛かるものであるが，利用者がブラウザから既存の Web システムを操作した際に発生した通信記録を用いて，アダプタの開発を効率化する手法が提案されている[1]．しかしながら，[1]で生成される API は必ずしもスマート端末上で動くアプリケーションに対して適切に動作するというわけではない．例えば[2]は単一の API を生成することを前提としているが，途中で利用者のインタラクションが必要な場合には API を複数にする必要がある．

そこで本論文では，既存の Web システムへの操作から採取した HTTP 通信や画面の記録情報を用いて，既存システムにアクセスするためのアダプタと，それに対応するスマート端末向けアプリケーションを生成する方式を提案する．また，生成するために必要な作業を支援するための開発環境についても言及する．

本論文の残りの構成は以下の通りある．2 章では，我々が提案する生成方式として目指すべくモバイル Web アプリケーションの構成について説明する．3 章では，既存の Web アプリケーションの操作から得られる情報と生成するために付与が必要な情報，そして 2 章で述べた構成への変換する方法について述べる．4 章では，3 章の作業を実施に必要となる開発支援機能及び，開発プロセスについて述べる．5 章では本論文で説明した方式の試作を社内のシステムに試行した結果について述べる．6 章で関連研究について触れ，7 章でまとめと今後の課題について述べる．

2. 生成するスマート端末向け Web アプリケーションの構成

我々の方式で目指すべくスマート端末向け Web アプリケーションの構成について説明する．

2.1 スマート端末における PC 向け Web アプリケーション利用時の課題

Android や iPhone などの近年のスマート端末には標準でフルブラウザが搭載されているため，多くの場合 PC 向けに作られたコンテンツをそのまま表示出来るが，それは必ずしもスマート端末から使いやすいものになるとは限らない．ここでは，特に本方式で生成するアプリに影響を与える既存アプリ利用時の２つの課題について説明する．

(a) 通信の効率化

PC は社内などの安定した LAN 環境で使われることが多いが，スマート端末は外出先などから 3G や LTE などのモバイル通信を用いた不安定かつ低速な環境で使われることが多い．安定したネットワーク環境にあることを想定して構築された大量のネットワーク通信を伴うシステムなどはスマート端末からでは使いづらいものとなる．スマート端末からの利用を想定する場合には，通信の効率性に十分配慮したシステムが求められる．

(b) 手番数の削減

PC 向けのシステムはキーボードとマウスを使うことを想定して入力が多くかつ複雑なものがたくさん存在しているが，画面サイズが小さくかつキー入力が不得手なタッチ操作を基本とするスマート端末からこのようなシステムを使うには PC 以上の労力を要することになる．また，スマート端末は電車に乗っているときなどの隙間時間を利用して使われる機会が多い．そのためオフィスで使うことが多い PC 向けシステムよりも最小限の手番で手早く操作できるシステムが求められる．

2.2 HTML5 を用いた RIA 方式

RIA[10]とは Web ブラウザなどのクライアントの機能を活かした柔軟なインタフェースをもつ Web アプリケーションのことである．従来ウェブブラウザ上で RIA を実現するためには Flash や Java アプレットなどの拡張機能を用いる必要があったが，マッシュアップが流行ったころに使われだした非同期通信を行う API である Ajax や W3C が 2013 年に勧告した HTML5 の各種 API を用いることで拡張機能なしに RIA を実現できるようになった．

W3C は Mobile Web Application Best Practices[9]の中でモバイル Web アプリ開発において考慮すべき事項とそれを実現するための方法について HTML5 の機能と対応付けながら説明している．RIA を採用することで，従来のサーバサイド主体の Web アプリケーションに比べて特に以下の点で，2.1 節で述べた課題の解決に結びつくメリットがあると考えられる．

- クライアント側の静的なリソース量を増やせるので，その分だけクライアント側のキャッシュが効く．(a)
- サーバ側のモデル変更を伴わない限りクライアントとサーバ間の通信が不要になるので，通信量や通信回数の削減が見込める．またそれに伴い，画面数も減らすことができる．(a) (b)

以上より，本論文では HTML5 を用いた RIA 方式としてアプリを生成することを目標とする．

2.3 アダプタを介するアクセス方式

本方式では既存の Web システムへの操作をアダプタとして生成・配備して，スマート端末のアプリ画面からはアダプタを介してアクセスすることで既存システムの修正を不要とする方式を採る．この概要図を図 1 に示す．

また，アダプタを介することは，既存システムに影響を与えなくなるメリットに加えてアダプタ上で通信内容をカスタマイズできるという特徴がある．2.2 節で RIA 方式の

必要性について述べたが，既存の PC 向けに構築されている Web アプリケーションの多くは RIA 方式になっているわけではないため，アダプタ上でその差異を吸収することが可能となる．

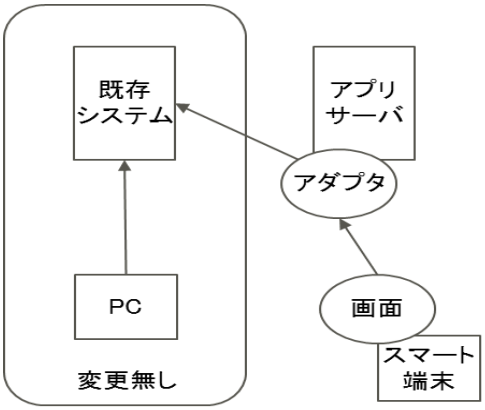

図 1 アダプタを介するアクセス方式の概要図

3. スマート端末向け Web アプリケーションの生成方式

本方式では，既存の Web システムへの操作から採取した HTTP 通信や画面の記録情報を元に，スマート端末アプリ用のアダプタと，それに対応するスマート端末向け画面を生成する．本方式は以下の 4 つのステップで構成される．

1. 既存システム操作記録の採取
2. 生成に利用する画面項目と生成タイミングの決定
3. 通信記録パラメタの変数化と分類化
4. アダプタと画面の生成

これらのステップの流れを図 2 に示す．

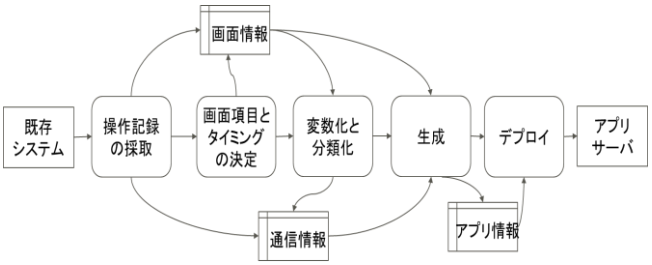

図 2 アプリ生成ステップ

まずステップ 1 で既存システムから通信情報と（通信情報に含まれる）画面情報を採取する．続いてステップ 2 では，得られた画面情報に対してスマート端末で用いる画面項目や，その項目を実行時にどのタイミングで生成するかを決定し，画面情報に付加する．ステップ 3 では得られた通信情報に対してパラメタの変数化と分類化を行い，通信情報に付加する．ステップ 4 でこれらの情報を用いてアダプタと画面からなるアプリを生成する．生成したアプリをアプリサーバ上に配備することでスマート端末からの利用が可能となる．各ステップの詳細について説明する．

3.1 既存システム操作記録の採取

このステップでは，開発者が既存の Web システムを操作するなどして発生させた HTTP 通信内容の採取を行う．取得する内容を以下のように定義する．

connection = [(req(1), res(1)),...,(req(N), res(N))]

req(i) = (url, [param_list])

res(i) = (http_code, html_body}

HTTP 通信内容 connection はリクエスト情報 res とレスポンス情報 res の組から構成され，リクエスト情報にはアクセス先の url とリクエストパラメタ param_list が含まれる．レスポンス情報にはリクエストの結果をあらわす http_code と HTML で記述された画面情報 html_body が含まれる．

3.2 生成に利用する画面項目と生成タイミングの決定

このステップでは，2.2 節で述べた RIA 方式のアプリを生成するために，記録時に得られた通信情報に含まれる画面情報に対して，まず画面項目に分割し，各画面項目の生成タイミングを決定する．

HTML の画面項目への分割については素朴に HTML を解析してタグ毎に分類する方法や，より高度には VIPS[8] など意味のある単位に分割する手法が知られている．本方式では具体的な分割アルゴリズムは問わないが，いずれかの手法を用いて画面情報 html_body から画面項目の集合 frag(1),…,frag(N)を得られるようにする．

画面項目の生成タイミングは，ステップ 4 でアプリを生成するときと，アプリ生成後に実際に既存システムにアクセスするときが考えられる．既存の典型的な Web アプリケーションは，クライアントから送られた HTTP リクエストに対して，サーバ側でプログラムの実行によって動的に HTML の生成を行い HTTP レスポンスとして返却する．つまり，このようなアプリをそのまま再現しようとした場合の生成タイミングは後者となる．しかしながら，動的に生成される画面に含まれる情報には何度アクセスしても変わらない箇所も多く含まれており，このような情報は実行時に毎回プロラム処理によって生成するのではなく，HTML として静的に保持する方が性能面で優位である．本方式では両者を区別してアプリ生成時に異なる方法を採る．

各画面項目の生成タイミングが静的か動的かを判定する方法のひとつとして，既存システムから複数回の同一操作記録を取得して，両者の画面項目情報の同一性を確認し，同一の場合には静的であると判断する方法が考えられる．

また上記の他に，画面項目をスマート端末に用いないという選択肢も考えられる．例えば PC で普段見慣れている操作方法や利用に影響を与えない画像などを省くことは．通信料削減の観点で重要となる．

3.3 通信記録パラメタの変数化と分類化

このステップでは，記録時に得られた通信情報をアダプタとして再現可能とする方法について決定する．

パラメタの変数化とは，得られた通信情報のパラメタ情報などの一部を変数化して，実行時にアダプタが呼ばれた際にアダプタのリクエスト情報を変数に代入することで既存システムを呼び出せるようにすることである．例えば ID やパスワードなどの利用者認証情報は利用者毎に異なるため，ステップ 1 で採取した値をそのまま用いるのではなく，変数化を施して実行時に値を決定しなければいけない．この詳細については[1]で述べられており，本方式でも同様に手法を適用する．

パラメタの分類化とは，パラメタ情報を以下の 2 種に分けることである．この情報はステップ 4 で生成するアダプタの粒度に影響する．

1. 利用者入力を要さないパラメタ
2. 利用者入力を要するパラメタ

利用者入力を要さないパラメタとは，実行しても常に値が同じ固定値であるか，既存システムのあるレスポンスを得たあと，次のリクエストを送るために必要なパラメタ値を自動的に抽出可能であり，利用者の入力を必要としないものを指す．例えば input タグの hidden 属性値から発生するリクエストパラメタは，利用者が直接値を眼にすることなくブラウザがリクエストとして送信するものであり，この分類に属する．

逆に利用者入力を要するパラメタとは，利用者が入力値を与えないと決定することが不可能なものを指す．例えばログイン時に与える ID とパスワードなどの認証情報や物品購入を行う際の購入対象物の値などは利用者とのインタラクションなしに判断することが不可能であるため，そこから発生するリクエストパラメタはこの分類に属する．

判定について以下の方法を用いることができる．

- ある通信のリクエストパラメタ値がその直前のレスポンスデータに含まれており、かつそれが利用者入力可能なフォーム部品だった場合
- 複数回の同一操作記録を取得した際に、入力可能なフォーム部品コンテンツの一致性を判定し、一致しなかった場合

あるリクエストに対して利用者入力を要するパラメタを含まない場合は，利用者に対してレスポンスを返す必要無しに既存システムの再現を続行することができる．

3.4 アダプタと画面の生成

ステップ 1 から 3 までで得られた情報を元にアダプタと画面の生成を行う．生成手続きを以下に示す．

```
p_idx = 0
w_idx = 0
page[p_idx] = generate_page(PTEMPLATE)
adapter[w_idx] = generate_adapter(WTEMPLATE)
for con to connections do
  page[p_idx].regist(con.res.static_fragments)
  adapter[w_idx].regist(con, con.res.dynamic_fragments)
  if exist_user_input(con.req.params) then
    p_idx++
    w_idx++
    page[p_idx] = generate_ adapter(PTEMPLATE)
    adapter[w_idx] = generate_ adapter(WTEMPLATE,)
  endif
endfor
```

この手続きではステップ 1 で採取した通信内容 connections を先頭から順にステップ 2 とステップ 3 で行った設定を確認していき，page と adapter のリストを生成する．

con.res.static_fragments は通信内容に含まれる画面のうち，ステップ 2 で生成タイミングが静的であると指定した画面項目の集合である．この項目をテンプレートに基づいて用意したページに挿入する．一方，生成タイミングが動的であると指定した画面項目の集合である con_res.dynamic_fragments については，既存システムへのアクセスをしないと取得できないため，adapter 側で取得するようにする．

exist_user_input()はステップ 3 で述べた利用者入力を要するパラメタが含まれるかどうかを判定する関数である．もし含まれていない場合は同じ adapter で次の既存システムへの通信を行うようにすることで複数の通信をひとつに集約する．ページについても同様に複数の画面を集約する．もし含まれている場合は新たなページやアダプタとして生成を行う．

この手続きで生成される画面 page とアダプタ adapter は以下のように動作する．

1. システムは登録された静的画面項目を伴う page[i]を表示する
2. page[i]は利用者の送信アクションによって Ajax を用いて adapter[i]を呼び出す．
3. adapter[i]は登録されている通信に基づいて既存システムを呼び出す．
4. adapter[i]は既存システムの結果と登録された動的画面項目を伴う値をレスポンスとして返す．
5. page[i]は受け取った値をローカルストレージに格納して，DOM 変更か URL 変更によって page[i+1]に遷移する．

これらのうち，2 と 5 については PTEMPLATE 内に予め

JavaScript コードとして埋め込んでおき，3 と 4 については Java のようなサーバサイド処理記述言語として埋め込んでおく．

また 5 については，URL を同一にしたまま DOM 書き換えで実現する方法と別の URL に遷移する方法の 2 種類の実現方法が考えられる．前者は無駄な通信の発生や同じ JavaScript ロジックの読み込みを防げるため応答性で優れるが，1 つのソースに大量の情報を持つことになるので画面数が多い場合にメモリ効率の問題が起きるためアプリに併せてどちらの方式を採るか検討する必要がある．

4. 開発環境

本手法を実施するための開発環境について述べる．3 章の各ステップは全て自動的に行えるものではなく，人による確認や操作も必要となる．また，3 章のステップ 2 で画面項目のスマート端末への転用について述べたが，実際には転用するだけでなく画面全体の UI 調整が必要である．本開発環境では主にこの 2 点を支援する事を目的とする．本開発環境の構成図を図 3 に示す．

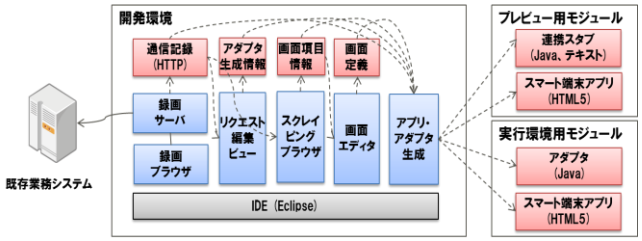

図 3 開発環境の構成図

録画ブラウザと録画サーバは，ステップ 1 に相当するものである．録画サーバは録画ブラウザと既存システムのプロキシとして機能しており，開発者が録画ブラウザ上で既存システムを操作した際に発生する通信情報を取得する．取得した情報は図 4(a)のように発生した通信順に格納する．

スクレイピングブラウザは，ステップ 2 で述べた画面項目に関する情報の決定を支援する機能である．画面例を図 4 の(b)に示す．本開発環境ではパースされた HTML を開発者が画面上で直接静的であるか静的であるかを選択できるようにしている．選択された項目は後で情報を取得できるように XPath として保持される．また，画面項目を選ばないことも可能であり，その項目は画面生成時に無視される．もしその項目がリクエストパラメタに紐付いている場合は，自動的にステップ 3 の利用者入力を要さないパラメタで述べた固定値の扱いとなり，通信集約の対象とすることができる．

リクエスト編集ビューは，ステップ 3 で述べた通信記録の変数化と分類化を支援する機能である．画面例を図 4 の(b)に示す．記録した通信のパラメタに対して自動判定機能を適用後，変数化の有無や分類の確認や修正を行うことが可能である．

画面編集エディタは，画面の調整を目的として，画面項目を生成予定の画面テンプレートに配備した後で画面項目の配置順序を入れ替えたり，画面項目に対して変換関数を適用したりするための機能である．画面エディタには図 5(a)に示す HTML ソースコードを直接表示するビューや図 5(b)に示すステップ 2 で切り出した項目を一覧化するアウ

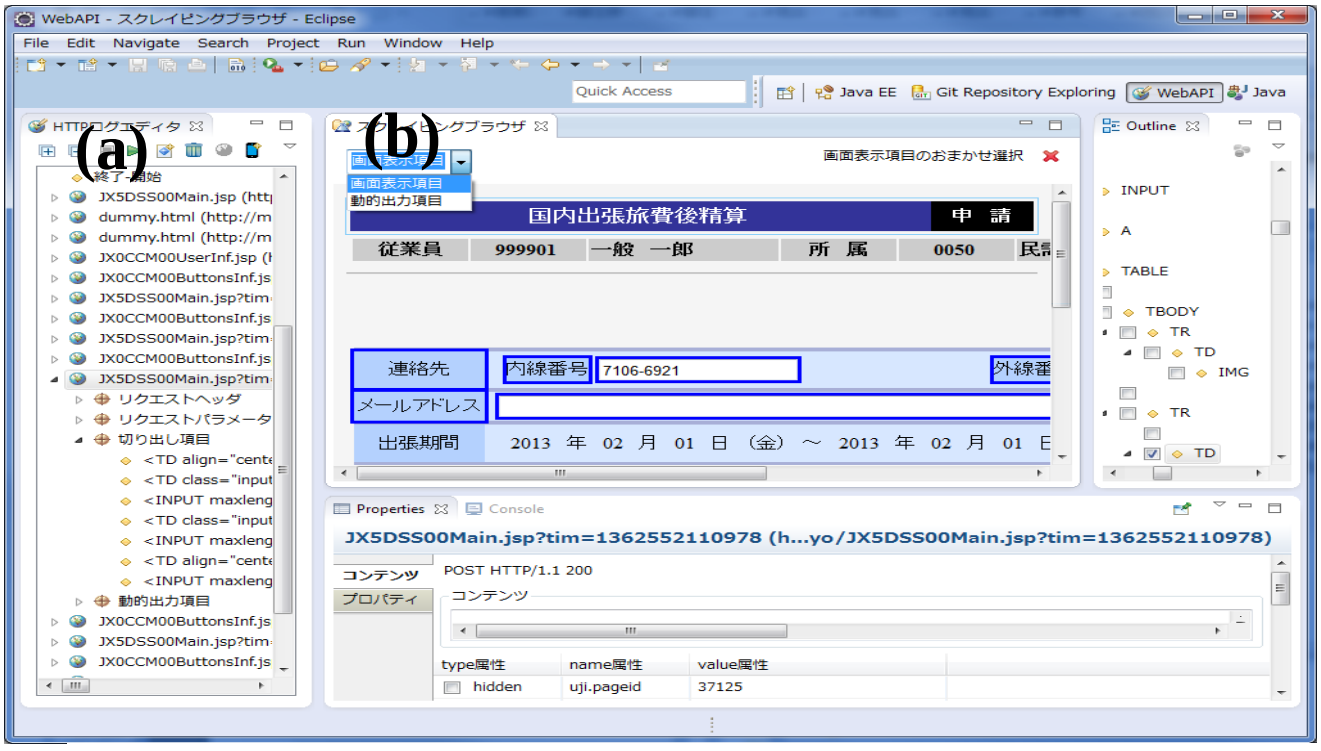

図 4 スクレイピングブラウザとリクエスト編集ビュー

トラインビューやレイアウトを直接確認する機能などがある．

これらの開発支援機能を用いて編集作業を行った後で生成機能を呼び出すことになるが，本開発環境ではプレビュー用のアプリと実際のアプリ 2 種類の生成機能を提供している．モバイル開発において UI の調整は非常に重要であり，一度確認のためにアダプタや画面を生成し，その後で画面を再度見直すといったプロトタイプ開発アプローチを採りたいケースがよく発生する．本開発環境が生成する実アプリはアダプタを介して稼働中の既存システムにアクセスを行うが，プレビュー用のアプリはアダプタの変わりにスタブにアクセスし，スタブは既存システムにアクセスせずに固定値を返す．本手法では既存システムの操作記録を得ることができるため，その値をスタブとして用いることができる．

本手法を用いた場合の開発手順の一案を図 6 に示す．開発者は録画ブラウザによる記録からプレビューアプリ生成までの各機能(図 6 の 2.2 から 2.5)を反復的に行うことで，振る舞いを確認しながら開発を行うことが出来る．

5. 適用事例

本論文で説明した方式の試作を社内のシステムに適用した結果と考察について述べる．

5.1 対象システムと機能

本試行では社内の事務処理を行うシステムを対象とする．

まずはスマート端末対応したい機能の決定を行った．

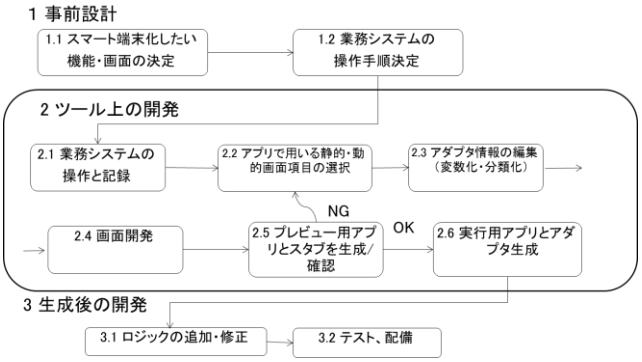

図 6 ツールを用いた開発手順

このシステムは様々な機能から構成されているが，その中でも特にスマート端末を使って外出先などから行えるようになると便利だと思われる，出張を行った際の旅費清算の申請処理を対応の対象とした．

図 7 は，PC から旅費清算の申請を行うフローの概略とスマート端末アプリ対応するために本手法で記録対象としたフローを示したものである．なお，対象システムはフレームやポップアップウィンドウを用いているが，ここでは利用者が実際に操作する画面の遷移を記載している．対象システムで旅費申請を行うためには，まずログインを行い，メニュー一覧から旅費申請を選択する．その後，数回の画面遷移を伴って出張期間やその内訳などを入力し，上司へ回送を行って申請完了となる．申請を終えたら最後にログアウトを行う．この一連のフローをスマート端末アプリ化する．

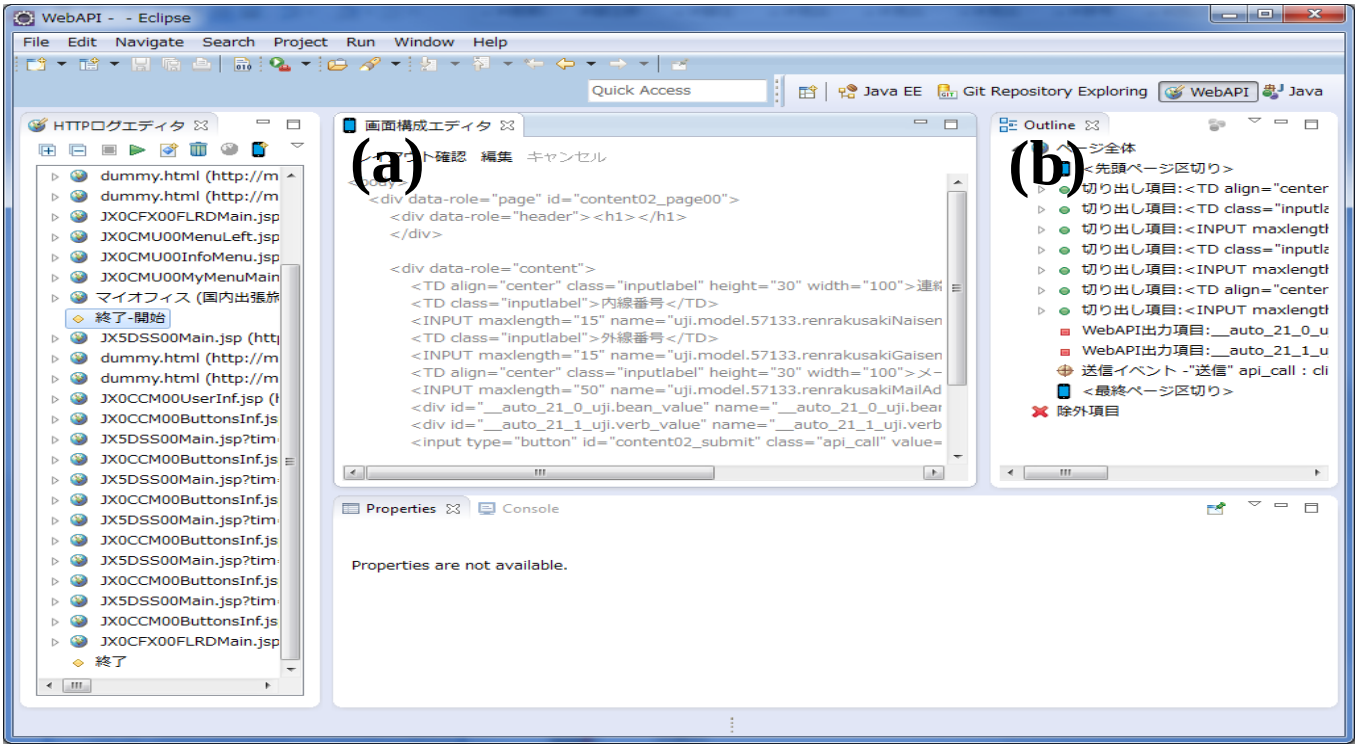

図 5 画面エディタ

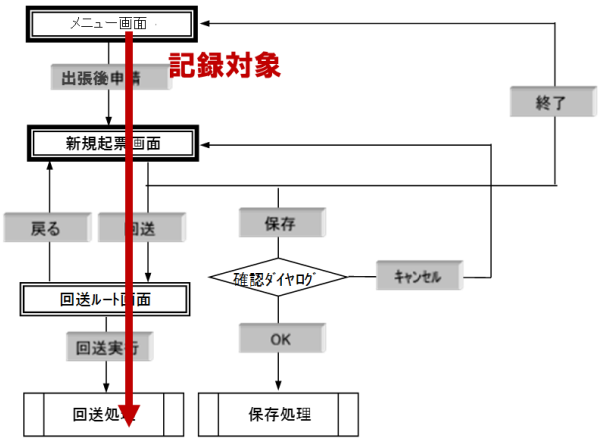

図 7　旅費清算システムのフロー

5.2 開発

4 章で述べた開発環境を使用し，図 6 の開発手順に従って開発を実施した．本開発環境が生成したコード量と全体の開発量を表 1 に示す．なお，本開発では 3rd-Party 製のライブラリを用いているが，それについては除外している．

表 1　生成/全体コード量と生成率

対象	機能	生成コード量	全体コード量	生成率
アプリ	画面(HTML)	130	391	33.2%
	クライアントロジック (JavaScript)	232	232	100%
アダプタ	業務システムアクセス(Java)	661	661	100%
	画面処理(Java)	68	237	28.7%

画面について既存の HTML 項目をそのまま用いるのではなく，モバイル向けの UI ライブラリを組み込むことで対応を行った．jQuery Mobile[11]などの HTML5 対応を謳うライブラリの部品は，HTML の文法に沿っていればライブラリの使用を意識しなくてもスマート端末に適した表示を行ってくれる．しかしながら，既存システムは正しい文法に沿っていない HTML で記述されていることも多いため，それを修正する作業が必要となる．これについては，HTML の生成に使用しているサーバサイドフレームワークなどに依存することも多いため，一度対応を行えば比較的横展開しやすいのではと考えている．

画面に関する処理以外は生成後に修正を行うことなく，動作するシステムを開発することができた．本システムには JavaScript 処理が用いられており，それが原因で動かなくなる懸念があったが，通信処理の再現で対応可能な範囲であった．

開発したアプリのスクリーンショットを図 8 に示す．

図 8 スマート端末向けアプリのスクリーンショット

5.3 生成したアプリケーションの特徴

対象システムを使用するときに発生する通信回数と入力手番数について，既存システムを直接利用する場合と生成アプリからアダプタを介して利用する場合とで比較した．

通信回数について，対象システムを利用する際には画面やアダプタ以外の画像などに対する通信も発生するが，対象システムにおいては画像を見栄え以外の目的に利用していないため，利用有無は業務の実行に影響を与えないため完全に主観的な問題になると判断して比較の対象外とした．また 3.4 節で述べた，生成する画面の遷移を DOM 書き換え方式にするか URL 遷移方式にするかで通信回数は異なるが，ここでは URL 遷移方式を採っている．比較結果を表 2 に示す．

表 2 通信回数の比較

	直接	アダプタ経由
画面	18	3
アダプタ	-	2
合計	18	5

直接アクセスする場合に発生する 18 回の通信は 3 章のステップ 1 で取得する connection そのものである．それがステップ 3 の分類化を経ることで，”ログイン処理”と”旅費清算入力処理とログアウト処理”の 2 つのアダプタにまとめることができる．アダプタに伴って画面数は 3 つとなり，通信数を 5 回に削減できた．

手番数とは，利用者が画面遷移のためにボタン押下をしたり，フォームに情報を入力したりする数の総和である．スマート端末でもっとも負担が掛かるのは文字入力であるため，手番について文字入力とそれ以外の 2 つに分けて測定した．比較結果を表 3 に示す．

本システムでは文字入力を行わないと処理を行えない項目が多く削減数は最低限にとどまった．ここで削減できたものは主にステップ 2 で画面項目を選択しなかったものである．文字入力以外については多くの手番を削減できた．例えば画面数が減れば，それだけ遷移にボタン押下を行う

必要がなくなり，全体の作業が短縮できるようになる．

表 3 手番数の比較

	直接	アダプタ経由
文字入力	11	8
その他	18	4
合計	29	12

6. 関連研究

既存の Web アプリケーションの画面をスマート端末に適した画面に変換するサービスがある[6][7]. これらのサービスでは，HTML で記述された単一の画面を事前或いはアクセス時に変換する方式を採っている．このアプローチでは開発を全く行うこと無しにスマート端末からそれなりに使いやすいユーザインタフェースを生成することが可能であるため，対応工数面で優位である．しかしながら，効果は画面のみの部分最適化に留まり，通信効率を上げるためにはサーバ側も含めた改修も必要になる．

既存システムのユーザビリティを向上させるために，既存システムからを RIA 化する手法も提案されている[4][5]. これらの論文では既存システムからナビゲーションモデルを抽出し，それを元に RIA モデルに変換する手法を述べているが，彼らの目的は PC から使う際のシステムそのものをリッチにすることであるため，既存システムを改修することを前提としている．本論文では既存システムを修正することなくスマート端末向けにリッチな環境を提供することが可能になるが，その一方で本手法は利用者が既存システムを実際に操作して記録した範囲しか生成対象とできないため，既存システムのあらゆる機能をスマート端末したい場合には記録した内容をマージする開発作業などが加わり，多大な工数が掛かってしまうという課題がある．そのため目的に応じて彼らの手法を組み合わせる方法も考えられる．例えば，適用事例で示した旅費清算のようなニーズのある特定 1 機能だけを試行的にスマート端末対応して利用してもらい，明確に業務効率改善が期待できそうであれば既存システムの修正を伴う本格的なマイグレーションを実施するといった開発方法が考えられる．

7. おわりに

本論文では, PC 向け既存システムからスマート端末向けアプリケーションを生成する手法について提案した．本手法を適用することで，複雑なナビゲーションや画面を備える既存システムに対して，既存のシステムを修正することなく，少ない通信回数や手番数で処理を行えるスマート端末向けのシステムを開発できることを確認した．

今後の課題について，スマート端末対応可能とするシステムの拡充が挙げられる. 本手法は HTTP 通信のみを記録・再現の対象としているため, ブラウザ上で動く JavaScript の振る舞いやイベントが全体に影響を与える既存システムを対象とした場合にうまく動かなくなってしまう事がある．これを解決するためには，利用者とブラウザ間の操作や DOM の変更情報など, 通信以外の情報を記録・再現するための仕組みを検討する必要がある． また，3 章で述べた方式についてはまだ人手で決定しなければいけないものが多いため，これらをより自動化して作業を容易にしたり，人手で行った設定が整合性を満たすかどうかを検証したりする仕組みを検討することも課題である．

参考文献

1) M. Tanaka, T. Kume, A. Matsuo, "Web API Creation for Enterprise Mashup," IEEE World Congress on Services (SERVICES) 2011, pp.319-326, Washington DC, U.S., 4-9 Jul y 2011.
2) F. Rosenberg, R. Khalaf, M. Duftler, F. Curbera, and P. Austel, "End-to-end security for enterprise mashups," In Proceedings of the 7th International Joint Conference on Service-Oriented Computing (ICSOC) 2009, Stockholm. LNCS, pp. 389-403,
3) W. Luke. “Mobile first.” New York: A Book Apart, 2011. Print.
4) A. Mesbah , A. van Deursen , “Migrating Multi-page Web Applications to Single-page AJAX Interfaces” CSMR '07 Proceedings of the 11th European Conference on Software Maintenance and Reengineering Pages 181-190 IEEE Computer Society Washington, DC, USA ©2007
5) R. Rodríguez-Echeverría, J. María Conejero, P. J. Clemente, J. C. Preciado, F. Sánchez-Figueroa , “Modernization of legacy web applications into rich internet applications” ICWE'11 Proceedings of the 11th international conference on Current Trends in Web Engineering Pages 236-250 Springer-Verlag Berlin, Heidelberg ©2012
6) Duda Mobile, http://www.dudamobile.com/
7) bMobilized, http://www.bmobilized.com/
8) D. Cai, S. Yu, J. Wen and W. Ma. "Extracting Content Structure for Web Pages based on Visual Representation", in the Fifth Asia Pacific Web Conference (APWeb2003), 2003.
9) “Mobile Web Application Best Practices” http://www.w3.org/TR/mwabp/
10) Duhl J., “Rich Internet Applications”, IDC white papers, 2003, http://www.idc.com
11) jQuery Mobile, http://jquerymobile.com/