O-015

サービスカスタマイズ向けウェブ自動操作ステム

北野 貴稔† 井口 圭一† 小山 和也†
Takatoshi Kitano Keiichi Iguchi Kazuya Koyama

1. はじめに

近年、企業の業務システムでは、業務の効率化が望まれる一方、内部統制の強化が求められている。

企業では、既に多くの社内システムを保有しており、様々な情報や機能が蓄積されているが、大企業では多数の利用者毎の細かな要件の違いがあり、企業の IT 部門が全て把握し、それをシステムに反映していくことはコスト対効果の面から難しい。そのため、コスト対効果という面においては、既存社内システムを無改造で機能改善や機能追加などのカスタマイズを、エンドユーザーが行うニーズが存在する。しかし、ユーザーがどのようなカスタマイズを行うかを把握できないため、統制上の課題があった。

従って、統制面を考えると、企業内においては、カスタマイズ内容はユーザや部門など利用者毎に設定できるのと同時に、適切なものがユーザーによって実行されることが保証され、それらのカスタマイズ定義は集中管理されることが望ましい。また、企業においては情報のアクセス権限などから個人の権限に応じてアプリケーション内容が変わるため、個人の権限に応じてカスタマイズを切り替える必要がある。

そこで本稿では、カスタマイズの中で、特に利用者毎の定型作業を自動化するカスタマイズを、サーバー側でプロキシ を介して Web アプリケーションのページに自動実行処理の起動処理を埋め込み、クライアント側でユーザー操作の自動実行をエミュレーションし行う事で、利用者の権限に応じて Web アプリケーションの操作手順を自動化するシステムを提案する。なお、本プロジェクトは Nagatuki プロジェクト[4]の研究活動の一部であり、本研究ではそのうち自動操作技術に関し詳細を述べる。

2. 既存技術の課題

既存システムを活用して外側でカスタマイズする技術として、既存システムの API 等を使うマッシュアップ技術が存在する。しかし、既存のアプリケーションが API 化されている必要があり、加えて API を呼び出す GUI を新たに作り直す必要があることから、カスタマイズの開発工数が膨らんでしまうという課題がある。

ウェブアプリケーションを外側から自動操作する技術には、WebMacros[1]のような HTTP 操作シミュレーション技術や、Checkhoot[2] 及び Koala[3] のようなクライアント側での GUI 操作エミュレーション技術、またはサーバーサイドで IE コンポーネントをサーバー側でのウェブ操作エミュレーション技術が存在する。HTTP 操作エミュレーション技術やサーバー側での操作エミュレーション技術では、サーバ上で Web ブラウザの動作をエミュレーションする形になるため、途中まで自動実行して、その後クライアントのブラウザに実行状態を移して手動操作を継続することができない。しかし、企業の業務システムでは、承認や業務上の判断が必要になるため、全体の操作の自動化ではなく、一部操作の自動化ができることが望ましく、従来技術には課題があるといえる。一方、クライアント側の GUI 操作エミュレーション技術では、自動操作後のユーザー操作が可能ではあるが、ユーザーが何の自動操作を実行するかを選択するため、何を実行するかを管理できず統制面で課題がある。

† （株）日本電気株式会社, NEC Corporation

3. カスタマイズ向けウェブ自動操作システム

3.1 提案システムアーキテクチャ

本稿では、サーバーサイドプロキシにより自動実行の起動処理を埋め込み、ブラウザ内で自動実行処理を動作させるカスタマイズをウェブアプリケーションに付加するアーキテクチャを提案する。この自動実行定義は、ユーザーの権限に応じてユーザー毎のマクロ定義を選択し実行させることで、権限に応じた自動操作を実行可能とする。サーバー側で自動実行の起動手段を埋め込む事で、複数のユーザーに対して一意にマクロ定義が決まる。

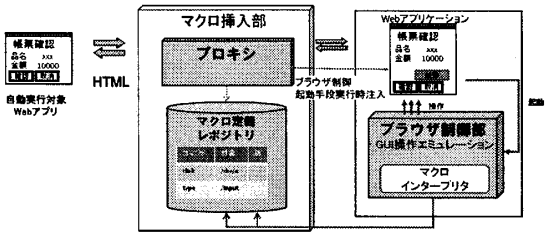

図 1. サービスカスタマイズ向けウェブ自動操作システム

本アーキテクチャにより、サーバーサイドプロキシでブラウザ制御手段を埋め込み、クライアント側で自動操作をすることで、自動操作後に実行状態を引き継ぐことが可能となる。さらに、マクロ定義をユーザーの権限に応じて読み込むことで、ユーザー権限に応じたカスタマイズを実行することが可能となる。

3.2 システム実装

提案システムでは、自動操作の起動手段を埋め込むサーバーサイドのプロキシ、ブラウザを操作し自動実行を行うブラウザ制御部、マクロ定義レポジトリの 3 つの主要モジュールから構成される。本システムは、プロキシにより自動実行起動手段を埋め込み(1)、ブラウザ制御部がブラウザ内で起動しマクロ定義をサーバーから読み込み(2)、マクロ定義に従いブラウザ制御部がブラウザにイベントを発行することにより動作する(3)。

(1) プロキシにより自動実行起動手段の埋め込み

プロキシは、HTTP プロキシとしてブラウザと Web アプリ間の通信を中継する。既存 Web アプリは必ずこのプロキシを介して利用し、プロキシでは該当ページに、ブラウザ制御部を起動する手段を、ボタンなどの形で自動実行対象のウェブアプリケーションに挿入する。図 2 は、ブラウザ制御部と起動手段を起動する JavaScript で記述されたスクリプトであり、プロキシ部はこのスクリプトをウェブアプリケーションに埋め込む。

```
<script type="text/javascript">
function runMacro(macroDefName) {
  openMacroControllerWindow(macroDefName);
}
</script>
<input type='button' class='submit'
value='　マ　ク　ロ　実　行　'
onclick='runMacro("<MACRO_NAME>");'/>
```

図 2. ブラウザ制御部起動手段スクリプトページ

(2) ブラウザ制御部のクライアントサイドでの起動

プロキシされたウェブアプリケーションに挿入されたブラウザ制御部の起動手段をユーザーが操作すると、ユーザーのブラウザの操作イベントに従って、ブラウザ制御部の新ウィンドウが新規に作成される。このとき、ブラウザ制御部はプロキシされたウェブアプリケーションと同一のドメインでウィンドウを作成する。これは、クロスドメインでブラウザ制御部がウィンドウを操作できるようにするためである

(3) ブラウザ制御部による自動実行

ブラウザ制御部では、JavaScript で実装された自動実行マクロ定義のインタープリタを備えており、起動時に起動手段で指定されたマクロ定義をマクロ定義レポジトリから読み込む。マクロ定義は、表 1 に示すように、コマンド・操作対象、操作対象に入力する値から構成され、マクロインタープリタにより解釈される。

表 1 自動実行マクロ定義

コマンド	対象	値
Type	/html/body/div[2]/textarea	kitano@example.com
Click	/html/body/div[2]/input	

インタープリタは、図 3 に示すように、マクロ定義が読み込まれると、そのマクロ定義に従い、ブラウザに表示されているウェブアプリケーションにフォーム値の入力やクリックなどの操作イベントを送ることで、自動実行操作を行う。操作イベント送信後にページの要素が表示されるまで待ち、次のコマンドを実行する。自動実行操作が完了すると、ブラウザ制御部のウィンドウが閉じ、ユーザーは継続してブラウザに対して操作をすることができる。

マクロ定義のスクリプトは、読み込む段階でユーザーの権限毎に定義を変えることができ、それによりユーザーごとに異なる自動化処理を実行することができる。

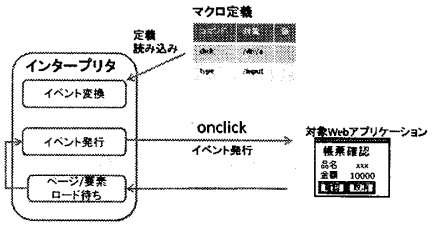

図 3. マクロインタープリタの動作

4. 考察

本アーキテクチャにより、サーバーサイドプロキシでカスタマイズを行うことで、統制を維持しながら元のウェブアプリケーションに無改造で自動実行操作カスタマイズを付与することが可能となり、さらにクライアント側で実行させることで、全体の作業のうちの一部分の処理のみを自動化させることが可能となった。

統制を維持しながらカスタマイズを行うには、クライアント側のブラウザ拡張で使用するマクロ定義を強制するといった方法も可能である。しかし、この手法ではブラウザ拡張定義のインストールおよび定義を強制する仕組みが必要となる。

一方、本アーキテクチャにはプロキシサーバーを使うことによる課題が存在する。ウェブアプリ内のリンク先をプロキシ内経由にしなければ、ブラウザの Same Origin Policy の制約によりブラウザ制御部でウェブアプリケーションを操作できないため、ウェブアプリケーションのリンクは全て同一プロキシを介するように書き換える必要がある。そのため、JavaScript などで生成される動的コンテンツ中の URL 書き換えは困難であるという点である。また、プロキシを介すことによるウェブアプリケーションにレスポンスを返す際の実行オーバヘッドである。

5. まとめ

本稿では、既存のウェブのアプリケーションを変更することなく、サーバーサイドプロキシを介して、ブラウザの自動制御起動手段と実行するマクロ名をウェブアプリケーションに注入することで、ユーザーが実行するマクロを制御しながら、ユーザー操作の自動実行を可能にする手法とそのシステムアーキテクチャについて述べた。

今後は、プロキシ方式の欠点を解消するため、実行オーバーヘッドの改善や動的 URL の書き換え部分の解析を進める予定である。

参考文献

[1] Alex Safonov, Joseph Konstan, John Carlis. Towards Web Macros: a Model and a Prototype System for Automating Common Tasks on the Web. Proceedings of the 5th Conference on Human Factors & the Web, Gaithersburg, MD, June 1999.

[2] Bolin, M., Webber, M., Rha, P., Wilson, T., and Miller, R.C. "Automation and customization of rendered web pages." Proc. UIST 2005, pp. 163-172.

[3] Little, G., Lau, T., Cypher, A., Lin, J., Haber, E., Kandogan, E. "Koala:Capture, Share, Automate, Personalize Business Processes on the Web." Proc. CHI 2007, pp. 943-946.

[4] 小山和也, 北野 貴稔, 井口圭一, "Nagatuki: サービスカスタマイズ向けマッシュアップ基盤", FIT 2008