マルチメディア通信と分散処理　95－5
（1999．11．18）

Webブラウザ操作の記録・再生システム

青木 義則　安藤 史郎　中島 周
日本アイ・ビー・エム（株）東京基礎研究所

概要
本稿では、Webブラウザ上での操作を記録・再生するシステムの仕組みについて説明する。本システムは、ユーザのWebブラウザ上での操作を検出し、イベント列としてテキスト・ファイルに保存する。また、Webブラウザを制御することにより、記録されたイベント列を再生することも可能である。本技術を用いることにより、Webサイト上でのユーザの振る舞いを検出し、実際の操作を再現することが可能となる。また、記録した操作列を利用した自動実行型のプレゼンテーション・パッケージをユーザに提供することも可能となる。本システムはJava実行環境を持つ通常のWebブラウザ上で動作する。また、インターネット上の通常のWebページに対して適応できるため、操作の記録用の特殊なページを準備する必要はない。

Web Operation Recording and Playback System

Yoshinori Aoki, Fumio Ando, and Amane Nakajima
IBM Research, Tokyo Research Laboratory

Abstract
This paper describes mechanisms for recording and playing back Web browser operations. A recorder detects a user's operations on a Web browser and saves them as an event sequence called a scenario. The recorder also plays back the scenario by controlling an actual Web browser. The recorder not only enables a Web site owner to gather users' behavior at his or her Web site, but also makes it easy to create a Web-based automatic presentation scenario, which can be played back later. The recorder runs in a Java-enabled Web browser. Users do not have to prepare the Web contents to be recorded; they can work with ordinary Web pages.

1. はじめに

近年のWorld Wide Webの普及により、Webブラウザはネットワーク環境における標準的なユーザ・インタフェースとなりつつある。Webブラウザは、ネット上の情報を閲覧するためだけでなく、オンライン・モール等のWebベースのアプリケーションを利用するためのインタフェースとしても利用されている。Webサイトの運営者にとって、Webサイトの集客率の向上は非常に重要な課題である。そのため、Webサーバのログを解析してユーザの挙動を分析するためのソフトが商用化されている。しかし、サーバ・ログから取得できる情報は、タイムスタンプ、アクセスのあったURL、ブラウザの種類などのわずかな種類のみである。そのためユーザの詳細な行動を分析するには不十分である。ユーザの行動を理解する最も良い方法の一つは、ユーザのブラウザ上での操作を直に観察することである。

本稿では、ユーザのWebブラウザ操作の記録・再生システムの仕組みについて説明する。本システムは、ページの読み込み、フォームの入力操作、

マウスポインタの移動等の操作イベント列を検出し、テキスト・ファイルに保存する。また、本システムは実際にWebブラウザを制御することにより、記録されたイベント列を再生することができる。本システムにより、Webサイトの運営者はユーザの操作データを収集できるようになる。また、Webブラウザを用いてユーザの操作を再現することも可能となる。しかし、ユーザの操作情報の取得はプライバシーの問題を含んでいるため、本システムの適応にあたってはユーザの同意を得るなど、運用には十分注意する必要がある [3]。

2. 本システムの特徴

本システムは、Document Object Model (DOM) [2] のイベントを記録する点に特徴がある。これまでにもOSレベルでのイベント・フッキングによりユーザの操作イベントを検出するシステムが提案されている [1]。その手法ではウィンドウ・システムでのイベントは検出できるが、ページ読み込みやリンク・クリックなどのWebブラウザ固有のイベントを検出することはできない。

本システムはJavaアプレットとしてWebブラウザ上で動作するため、ユーザのPCに特殊なソフトウェアをインストールする必要はない。また、インターネット上の既存のHTMLコンテンツに対して適応できるため、操作記録のために特殊なコンテンツを用意する必要もない。

ロータス社のスクリーンカムはイメージ・ベースの記録システムであり、画面のキャプチャによる操作の記録・再生を可能とする。画面の解像度などの環境に依存するが、通常１分程度の操作記録データは数メガバイトになる。一方、本システムは操作イベント列の記録システムであり、１分程度の操作記録データは10キロバイト程度である。記録データの収集・配信をインターネット経由で行う場合、データ・サイズを小さくすることが非常に重要になる。

3. 記録・再生の仕組み

本システムは、ユーザの操作イベントを記録するレコーダと、記録されたイベント列を再生するプレーヤで構成される。記録できるイベントの種類には、ページ読み込み、スクロール、ウィンドウ・サイズの変更、ウィンドウ位置の変更、フォームの入力操作、マウスの動き、マウスのクリックがある。記録データは、ユーザが訪問したページのURL、各ページでの操作イベント列を含む。

3.1. 操作の記録

レコーダはユーザの操作イベントを記録する。本システムでは、イベントを発生させるためのプログラムはプロキシ・サーバでHTMLファイルに埋め込まれる。図１に、本システムの概要を示す。レコーダはJavaアプレットとして実装されており、ユーザは最初にレコーダ・アプレットを含むWebページをブラウザに読み込む。レコーダ・アプレットは新しいブラウザのウィンドウを開く。以降、このウィンドウをコンテンツ・ウィンドウと呼ぶ。ユーザはコンテンツ・ウィンドウを使って通常のブラウジングと同様にWebページを閲覧することができ、コンテンツ・ウィンドウ上での操作がレコーダによって記録される。このとき、ユーザは埋め込みエンジンを持つプロキシ・サーバを通してWebページをコンテンツ・ウィンドウに読み込む必要がある。埋め込みエンジンは、ユーザの操作を検出するためのイベント・ハンドラとJavaScriptプログラムをHTMLファイルに埋め込む。

3.1.1. イベント・ハンドラの挿入

HTMLファイルにイベント・ハンドラを設定する手法には、２つの手法がある。１つは、以下に示す様にHTMLタグでイベント・ハンドラを指定する方法である。以降、この手法をタグ方式と呼ぶ。

```
<IMG SRC="XXX" onClick="notify()">
```

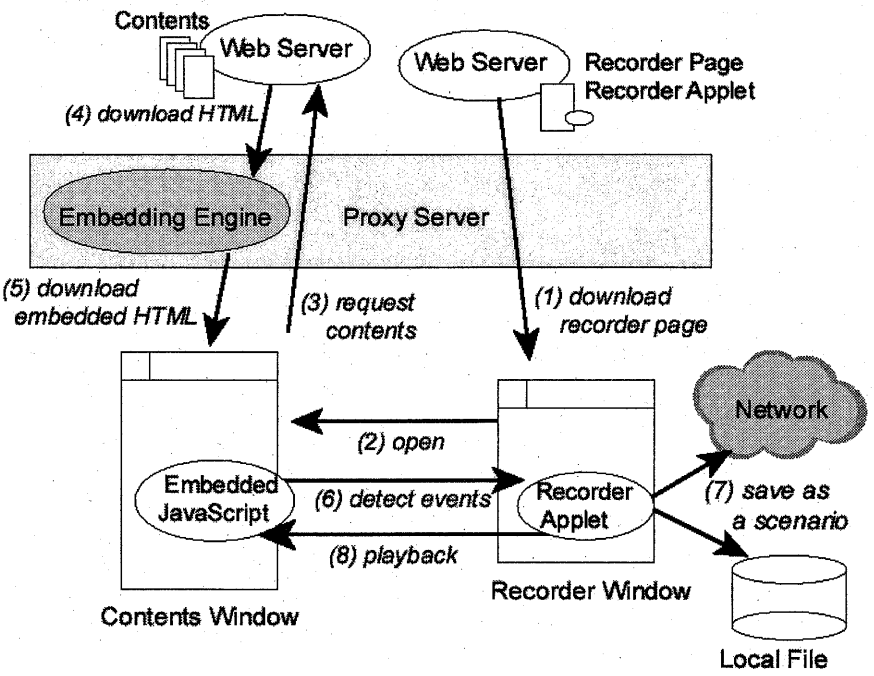


図1 システム概要

もう１つは、以下に示すように埋め込まれたJavaScriptプログラムでイベントハンドラを設定する方法である。以降、この手法をスクリプト方式と呼ぶ。

document.images[0].onClick=notify;

HTMLファイルの作者が既にイベント・ハンドラを設定していた場合、作者によるオリジナルのイベント・ハンドラと、埋め込みエンジンによって設定されるイベント・ハンドラの両方が実行されなくてはならない。埋め込みエンジンが設定するイベント・ハンドラの処理内容は、検出したイベントをレコーダ・アプレットに通知するだけなので、オリジナルのイベント・ハンドラの処理との間で矛盾が生じることはない。オリジナルのイベント・ハンドラ"org()"とイベント記録のためのイベント・ハンドラ"notify()"の両方を実行するためには、タグ方式において埋め込みエンジンは以下のようにイベント・ハンドラを挿入する必要がある。

<IMG SRC="XXX" onClick="notify();org()">

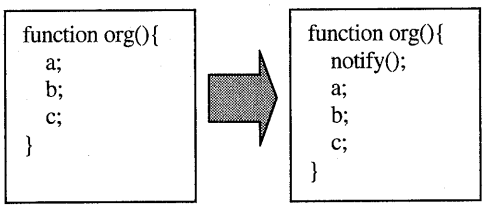
スクリプト方式では埋め込まれたJavaScriptプログラムは、以下の手順でイベントハンドラを設定しなくてはならない。

(1) オリジナルのイベント・ハンドラを関数オブジェクトとして取得する。

(2) 関数オブジェクトを図２(a)に示すような文字列オブジェクトに変換する。

(3) 文字列操作により、図２(b)に示すようにオリジナルのイベント・ハンドラに操作記録のためのイベント・ハンドラ notify() を挿入する。

(4)　文字列オブジェクトを関数オブジェクトに変換し、イベント・ハンドラとして設定する。



(a) Original Event Handler (b) Modified Event Handler

図2 イベント・ハンドラの挿入

インターネット上のHTMLファイルの中にはHTMLの文法に厳密に従っていないものも存在する。例えば、BODYタグが書かれていなかったり、TABLEタグの終了タグが書かれていないHTMLファイル等が存在する。タグ方式を用いた場合、書かれていないタグに対してはイベント・ハンドラを設定することができないため、全てのイベントを記録することができない。一方、スクリプト方式を用いた場合、ブラウザが正しく表示できるHTMLファイルに関しては、タグが正しく書かれていなくてもDOMのオブジェクトとしてJavaScriptでイベント・ハンドラを設定できる。また、タグ方式では、埋め込みエンジンは各HTMLタグを正しく認識しなくてはならない。そのため埋め込みエンジンの実装は複雑になる。一方、スクリプト方式ではJavaScriptファイルを埋め込むだけなので、HTMLファイルを完全にパースする必要はない。以上の理由から、スクリプト方式の方が操作イベントの記録には適していると言える。

3.2. 操作の再生

図1のレコーダ・アプレットはプレーヤとしての機能も持ち、記録された操作を再生することができる。レコーダ・アプレットは、JavaScriptの機能を利用して、実際のブラウザ（コンテンツ・ウィンドウ）を制御することにより再生を行う。例えば、ウィンドウのサイズ変更操作を再生するには、JavaScriptメソッドwindow.resizeTo(x, y)を用いる。

3.2.1. URL遷移の再生

URL遷移を再生するには、（1）ページ読み込みのタイミング、（2）フォーム提出の扱い、（3）フレームの扱い、の3点に注意しなければならない。

本システムでは、新しいページの読み込みはloadイベントとして記録される。しかし、loadイベントはページの読み込みが終了したときに発生するため、再生時にはloadイベントのタイム・スタンプではなく、新しいページに移るときに発生するunloadイベントのタイム・スタンプをもとにページの読み込みを開始する必要がある。

URLの遷移はフォームを提出したときにも発生する。フォームの提出にはGETとPOSTという2種類の方式がある。POSTの場合、レコーダがloadイベント発生時に記録するURLだけではフォームの提出を正しく再生できない。しかし、フォーム提出にGETとPOSTのどちらの方式を用いるかは、HTMLファイル中に指定されている。そのため、プレーヤは記録データにsubmitイベントを発見した場合、JavaScriptの機能を利用して実際にフォームを提出する必要がある。

ユーザが複数のフレームで構成されたWebページ（以降、マルチ・フレームと呼ぶ）を読み込んだ場合、loadイベントはそれぞれのフレームで発生する。図3(a)に示すマルチ・フレームの例では、ユーザには3つのフレームしか見えないが、実際には図3(b)に示すように5つのフレームで構成されている。ここで、図3(b)のフレーム1のように、全フレームを含むフレームをルート・フレームと呼ぶことにする。ユーザがマルチ・フレームのページを読み込んだ場合、親フレームのloadイベントは、そのフレームが含む全ての子フレームのloadイベントが発生した後に発生する。そのため、図3(a)のページを読み込んだ場合、

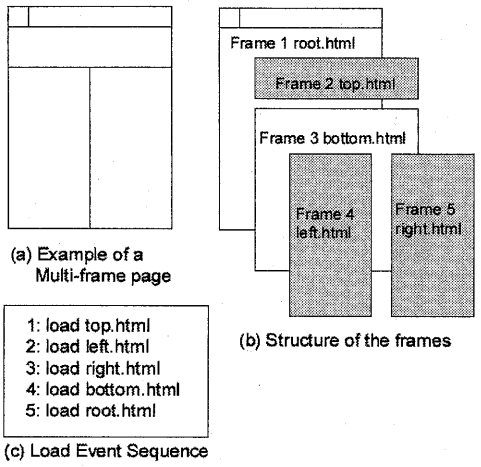


図3 マルチ・フレームの例

図3(c)に示す順序でloadイベントが5つのフレームで発生する。しかし、プレーヤは図3(c)の順序でページ読み込みを再生するとエラーを発生する。例えば、1行目のloadイベントを再生しようとした場合、フレーム1を読み込むまでtop.htmlの読み込み対象となるフレーム2は存在しないためエラーを発生する。同様の理由で、図3(c)の例では1～4行目までのloadイベントの再生でエラーを発生する。この例のように、親フレームの読み込みにより発生する子フレームのloadイベントは再生する必要はない。しかし、全フレームが読み込まれた後に子フレームで発生するloadイベントは記録する必要がある。例えば、図3(b)においてフレーム4がメニューのページになっており、フレーム4のリンクをクリックするとフレーム5に対象ページが読み込まれるような場合、フレーム5で発生するloadイベントは記録・再生されるべきである。そこで、loadイベントが発生したとき、埋め込まれたJavaScriptプログラムはそのloadイベントが親フレームの読み込みによって発生した場合はレコーダ・アプレットに通知せず、それ以外の理由で発生した場合はレコーダ・アプレットに通知する。

3.2.2. 時間管理

図4(a)に記録時の操作イベントと時間の関係を、図4(b)に再生時の操作イベントと時間の関係の例を示す。読み込み時間（T_1）は、HTTPリクエストからWebページの読み込み完了までの時間である。読み込み時間はネットワーク環境やCPU等に依存しており、再生時にレコーダ・アプレットで制御することはできない。そのため、記録時の読み込み時間（T_1）と再生時の読み込み時間（T_3）は異なる。再生時には、レコーダ・アプレットはページ読み込みが終了するのを確認してから再生を再開しなくてはならない。そうしないと、操作対象オブジェクトがブラウザに読み込まれる前に操作イベントを再生しようとしてエラーを発生させてしまう危険性がある。一方、読み込み時間以外のイベント間の時間は制御できる。そのため、レコーダ・アプレットは各イベントのタイム・スタンプに基づいて操作イベントを再生することにより、T_2とT_4を等しくする。

Webページをブラウザに読み込んでいるとき、読み込みが完了したオブジェクトからブラウザ上に表示されるため、loadイベントが発生する前に操作をすることが可能となる。図4(a)では、操作AとBがページ読み込み前に記録された操作イベントである。再生時に操作AとBをタイム・スタンプに基づいて再生すると、操作対象オブジェクトがまだ読み込まれていない可能性があるため、エラーが発生するかもしれない。そのようなエラーを避けるために、レコーダ・アプレットは図4(b)に示すように、操作AとBをページ読み込み終了後に再生しなくてはならない。

3.2.3. 特殊再生

レコーダ・アプレットは以下の2つの機能を提供する。

- 再生速度調整機能
- 再生イベント選択機能

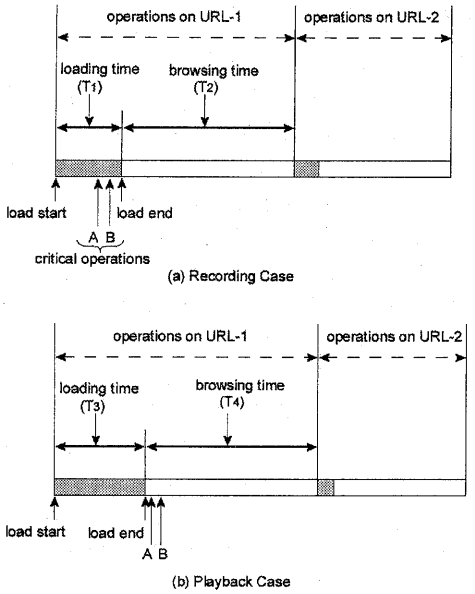


図4 時間管理

再生速度調節機能では、記録されたイベント間の間隔を調節することにより再生速度を変更する。ただし、3.2.2節で述べたようにページ読み込み時間は調節できない。

ユーザは再生イベント選択機能により、再生したいイベントの種類を選択することができる。

これらの機能を組み合わせることにより、ユーザは特定の操作のみを効率よく再生することができる。

3.2.4. マウス・ポインタの移動

レコーダ・アプレットは、コンテンツに埋め込まれたJavaScriptプログラムにより検出されるマウス移動イベントを記録する。しかし、JavaやJavaScriptではシステム・マウスを制御することはできない。そのため、マウス・ポインタの移動はイメージを用いて再生する。再生時にはDynamic HTMLの機能を用いてWebページ上に新しいレイヤを作り、その上にマウス・ポインタのイメージを貼り付けてマウス・ポインタの移動を再生する。

4. おわりに

本稿ではWebブラウザ上の操作イベントの記録・再生の仕組みを説明した。本システムの利点は、（1）通常のブラウザと通常のWebページに対して適応できること、（2）イメージ・ベースの記録システムに比べて記録データが小さいこと、（3）Java, JavaScriptで書かれているためクライアントマシンに特殊なソフトウェアをインストールする必要がないことである。

また、本システムには以下の3つの課題がある。（1）Javaアプレット、ActiveX、プラグ・イン等のWebページに埋め込まれたプログラム上の操作の記録・再生はできない。（2）フォームの提出はWeb上のサービスへの申し込みなどのトランザクション処理を発生させてしまう可能性があるため、実際にフォームを提出することなく正しく再生するための仕組みが必要である。（3）記録データはURLのみを含むため、Webページが更新された場合に正しく再生されない危険性がある。今後は上記課題について検討する必要がある。

参考文献

[1] Kobayashi, M., Shinozaki, M., Sakairi, T., Touma, M., Daijavad, S., and Wolf, C. "Collaborative Customer Services Using Synchronous Web Browser Sharing," in *Proceedings of CSCW '98* (Seattle, Washington, Nov. 1998), ACM Press, NY, pp. 99-108.

[2] W3C (World Wide Web Consortium). Document Object Model. Available at http://www.w3.org/DOM/.

[3] W3C. Platform for Privacy Preferences P3P Project. Available at http://www.w3.org/P3P/.