Web エージェントシステム MiSpider における継続的実行について

On Continuous Execution of Web Agent System MiSpider

大囿 忠親, 深萱 裕二郎, 伊藤 孝行, 新谷 虎松 †
Tadachika OZONO, Yujiro FUKAGAYA, Takayuki ITO, Toramatsu SHINTANI

† 名古屋工業大学大学院情報工学専攻
Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology
ozono@nitech.ac.jp, {fukagaya,itota,tora}@ics.nitech.ac.jp

本研究では，Web ページを介した智の収集・利用を促進するために，Web ページを誰にでも容易に書き込み可能なメディアとして位置づけ，それを Web Paper と呼ぶ．より高度な Web Paper の実現には，Web ブラウザ上の限られた計算資源やセキュリティに関連する制約を考慮した上で，エージェントを実装するための技術が必要である．本研究では，Web ページ上を実行環境とするソフトウェアエージェントとして MiSpider を実装した．本論文では，MiSpider のアーキテクチャや，Web ブラウザ上でのエージェントの実装技術について述べる．特に，異なる HTTP のセッション間で処理を継続的に実行する仕組みについて述べる．

1 はじめに

本研究では，高度な Web の実現を目指して Web Paper と呼ぶ技術を開発している [1]．Web Paper を実現するために，Web ページ上で動作する Javascript 言語に基づくエージェントシステムとして MiSpider を実装した [2]．Web ページ上の知的な支援をエージェントとしてカプセル化することで，Web ページ閲覧中にサービスを動的にプッシュしたり，単に 1 人のユーザの活動を支援するだけではなく，複数のユーザにまたがる作業の支援を可能にする Web ページの実現も可能になる．

ユーザが複数のページを閲覧する場合，それらのページ間でも継続的なサービスが提供されることが好ましい．Potter は，Web 閲覧におけるセッションを記憶媒体に保存する手法を提案しているが，Javascript の状態保存は考慮していない [4]．ブラウザとサーバ間は自由に通信できない，かつ，ユーザのページ遷移等のタイミングは予測困難なので，Javascript の継続的なサービスの実現が困難であった．本論文では，ユーザによる非同期な要因によるサービス実行中断においても，サービスの継続を可能にするための手法を提案する．以降，Web ページと Web ブラウザを，単に，ページ，ブラウザと記述する．

本論文の構成について述べる．2 では，MiSpider の実装言語である Javascript について説明する．3 では，Web Paper 上でのエージェントについて議論し，そのアーキテクチャについて考察する．4 では，本エージェントの実装方法について述べる．最後に 5 を本論文のまとめとする．

2 Javascript 言語における継続的実行

2.1 Javascript における多重実行

Javascript 言語は，ブラウザ上において利用可能な最も一般的なスクリプト言語である．Javascript 言語の文法は Java 言語に類似しているが，これらの言語は全く異なる言語である．

eval 関数は，Javascript 言語のソースプログラムを解釈実行する組込み関数である．

変数に関数を代入することが可能である．本論文では，代入された関数を関数オブジェクトと呼ぶ．Javascript 言語では，次の 2 種類の定義方法がある．

```
1: function f(x, y) { ...
2: var f = function(x, y) { ...
```

左側の数字は行番号である．上記 1 と 2 の両方とも関数名 f で引数 x,y の関数を表す．2 の var f は変数 f の宣言であり，f には関数オブジェクトが格納される．

次のような関数 g を考える．

```
3: var a = 1;
4: var b = 2;
5: var g = function() { f(a,b) ...
6: g();
```

3,4 行目で，変数 a,b にはそれぞれ値 1,2 が束縛される．5 行目では関数 g が定義されており，g 内では関数 f を引数 a,b で呼び出している．6 行目では関数 g が評価され，このとき変数 a,b の値が f で利用される．

Javascript 言語では，ページの表示時のページ生成処理，または，イベント駆動での短い時間のプログラム実行が設計上想定されている．プログラム実行時はユーザからのインタラクションを受け付けないので，推論処理のような実行時間の長い処理を扱うと，ユーザからはブラウザが停止した様に見えるため，好ましくない．より高度な機能を実現するためには，プログラムをユーザへ悪い影響を与えずに長時間実行できるだけでなく，多重実行の仕組みも必要となる．

Javascript 言語には，多重実行の実現に流用可能な組込み関数として setTimeout(P,T) 関数と setInterval(P,T) 関数がある．P を文字列，T を整数値としたとき，setTimeout 関数は，T ミリ秒後に eval(P) を実行する関数であり，setInterval 関数は，T ミリ秒毎に eval(P) を実行する関数である．また P が関数オブジェクトでも P を実行する．

Javascript 言語では，setTimeout を利用することで，末尾再起処理の多重実行が容易に実現可能である．例えば，次の様に記述すればよい．

```
 7: function p(x,y) {
 8:   x1 = x + 1; y1 = y + 1;
 9:   var q = function(){p(x1,y1)}
10:   setTimeout(q,10);
11: }
```

数 x,y を取る関数 p は，x と y の値を増加させるだけの関数である．9,10 行目が多重実行を考慮した末尾再起の記述例である．9 行目では，p(x1,y1) を呼び出す関数オブジェクトが生成されており，10 行目で setTimeout 関数で q を 10 ミリ秒後に実行する様にしている．q は setTimeout の後には，実行終了しているため，別の関数を実行できる．すなわち，関数 p を多重実行することが可能になる．10 行目の setTimeout 関数の第 2 引数の値を調整することで，複数のプログラミング間のスケジューリングも可能である．

2.2 Javascript における実行状態の保存

プログラムの実行状態の保存に関連して，プログラムのマイグレーションやそれの応用であるモバイルエージェントシステムの研究が行われてきた．マイグレーションやモバイルエージェントに関する研究は古くから行われているが，Web ページ上のエージェントシステムを作成する目的で Javascript に関するマイグレーションに関する研究は行われていない．

Javascript では実行時スタックにアクセスする手段が用意されていない．実行状態の保存には，プログラム変換による方法か，プログラム中に実行状態の保存箇所を陽に記述する方法が考えられるが，今回は簡単な後者に基づいて実現する．

Javascript 言語には，Object と呼ばれるデータ型が存在する．Object 型は，配列と辞書の両方の機能を持つ．例えば，Object が格納された変数 a のとき，a[1] = 2 が配列の 1 番目の要素に 2 を代入することが可能になり，a を配列として扱える．また，a[”key1”] = 2 とすれば，a を辞書として扱うことができ，この場合，項目 key1 の値が 2 という意味になる．また，別の記述方法として，a.key1 = 2 と記述することも可能である．末尾再起に基づくエージェントは大まかには次の様に記述すれば良い．Javascript には key1 などの名前を列挙する仕組みもあるので，env の内容を直列化するのも容易である．

```
12: var env = new Object();
13: function p(env) {
14:  エージェントの処理
15:   var q = function(){p(env)}
16:   setTimeout(q,10);
17: }
```

12 行目では，オブジェクト env を生成している．エージェントは，実行環境として保存したいデータを env 中に格納することにする．env.x = 1 のように，可読性の高い形式で記述可能なため，エージェントの処理を記述する際に支障がない．15,16 行目が，再実行のための処理である．

3 Web エージェント MiSpider

Web Paper におけるエージェントである MiSpider が満たすべき性質について議論する．MiSpider は，ページ上で動作するエージェントとサーバ上で動作するエージェントが協調してサービスを提供する．ここでは，次のような制約がある．1）ページ上のエージェントとサーバ上のエージェントの通信には制約がある．2）ページ上のエージェントは計算資源が限

られている．3）サーバ上のエージェントは計算資源はあるがユーザと直接インタラクションできない．

これらの制約を考慮すると，ページ上のエージェントは，ユーザに対するセンサ及びアクチュエータの様な役割を担当し，サーバ上のエージェントはページ上のエージェントを遠隔操作する役割を担当するのが良いと考えられる．制限された通信環境において，適切に協調するためのプロトコルが必要となるが，本論文では省略する．

Web Paper におけるエージェントが知的にユーザを支援するためには，複数エージェント間の通信，永続的なエージェントの実行，ページ内の環境でタスクを実行するためのセンサとアクチュエータ，そして，適切な言語処理系などを考慮する必要がある．また，ページ上で動作するという制約から，ブラウザがページを読み込むたびに，エージェントのプログラムをブラウザに読み込ませる必要があるので，プログラムのサイズも重要である．本節では，エージェントの，通信，永続性や継続性，そして外部とのインタラクションについて考察し，これらを実現する上で適切なアーキテクチャを提案する．

3.1 通信

ブラウザ上ではセキュリティの向上を目的として，さまざまな制約が課せられている．通信に関しても制約されており，Javascript を用いて自由に通信をすることができない．ある程度，自由な通信を実現する方法として，フォームを利用可能であるが，フォームを用いた場合，ページが変わる，すなわち，現在表示中のページが破棄されてしまい，エージェントが存続できない．また，ユーザインタフェースの観点からも，エージェントの通信毎にページが読み変わることも好ましくない．ページの更新を伴わない，エージェント間通信の実現が必要である．

3.2 永続性・継続性

ページを閉じたときや，検索サイトなどでフォームを送信した後も，ページエージェントがページが変わる前の状態で継続的に動作することが可能になれば，より効果的な支援が行える．本研究では，このような性質を，エージェントの永続性と呼ぶ．新たにページが読まれたときに，エージェントの状態を再構築するためには，ページが閉じられる，またはフォームを送信する直前のページエージェントの状態を保存し，関連のページが新たに開かれたときに，その状態を再構築するための仕組みが必要である．一般的に，対象とするページは，エージェントの存在を仮定して作成されているわけではないので，エージェントの状態をページ内のフォームを利用して送ることはできない．また，ブラウザに情報を残すための手段としてクッキーが挙げられるが，クッキーだけでは新たなページの読み込み時にページエージェントのプログラムコードまで再現することはできない．何らかの手法で，サーバにエージェントの状態を保存する方法を実現する必要があるが，そのためには前節で議論したような通信機能の実現が必要不可欠である．

3.3 外部とのインタラクション

ここでは，エージェントの外部とのインタラクションを考えるが，特に，ブラウザを動作環境とするエージェントのセンサとアクチュエータについて議論し，他のエージェントとのインタラクションについては議論しない．エージェントの外部とは，ページ，ブラウザ，そしてユーザとする．

センサは，ユーザとのインタラクションや，ページ内のオブジェクトの状態変化を扱う必要がある．アクチュエータは，ページ内のオブジェクトの操作や，ユーザへのサービスの提供のための機能を扱う必要がある．ページを表すデータは，ページを表す HTML が記述されたテキストとして，または，Document Object Model (DOM) にとして構築されたオブジェクトとして扱うことが可能である．センサやアクチュエータは，これら両形式での対応が必要である．

3.4 アーキテクチャ

本エージェントは，ベースエージェントとページエージェントから構成される．図 1 は，本エージェントの構成を表している．ページエージェントとはページ上で動作するエージェントであり，ページ上のデータやユーザに対してサービスを提供する．ベースエージェントとは Web サーバ上で動作するエージェントであり，ページエージェントのタスク実行をサーバ側から支援する．ベースエージェントは，複数のページエージェントと連携するが，あるページエージェントと連携するベースエージェントは一意に定まる．

このように，エージェントを 2 つの異なるエージェ

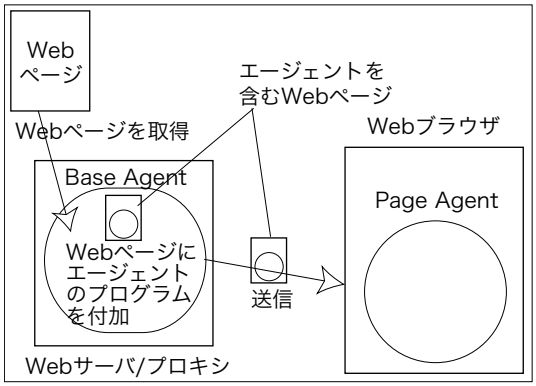

図 1: エージェントの構成

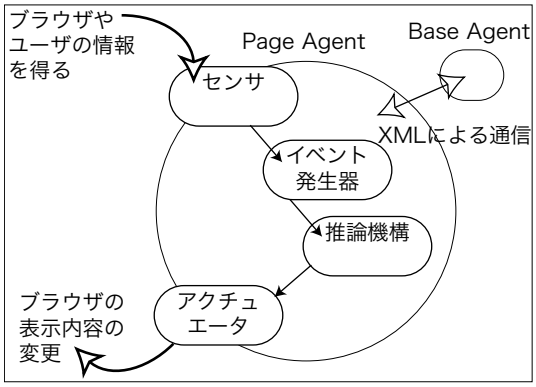

図 2: ページエージェントの構成

ントから構成するのは，ブラウザのさまざまな制限 (通信など) を超えて，より効果的なサービスを提供するためである．本構成により，エージェント間の通信が可能になり，またページ上のエージェントの永続性が実現可能になる．

3.5 ページエージェント

図 2 は，ページエージェントの構成を表している．ページエージェントは，センサ，イベント発生器，推論機構，そして，アクチュエータから構成される．ページエージェントを定義するには，ページ内の情報の変化の取得やページ内容の操作のような低レベルの処理と，低レベルにおけるデータを高レベルの処理で扱えるようにする処理，そして推論のような高レベルの処理をそれぞれ定義する必要がある．ページエージェントの定義は，ページエージェント内の 4 つのモジュールを定義することで実現される．すなわち，センサ定義，アクチュエータ定義，イベント定義，そして推論定義から構成される．センサ定義とアクチュエータ定義は，ページエージェントの低レベルの処理を定義する．センサ定義は，ブラウザ上の情報の変化やユーザからのインタラクションをイベントに変換する．アクチュエータ定義は，高レベルの処理を低レベルの処理に変換し，ページエージェントの外界に対して変化を与える手続きを定義する．イベント定義は，マウスクリックのような低レベルのイベントを，ユーザからの情報要求の様なより高レベルなイベントに変換する手続きを定義する．推論定義では，ページエージェントの推論機構を決定するための定義をする．

4 実装

4.1 通信

ブラウザのセキュリティ上の制約から，ページエージェント間の直接的な通信は不可能である．ページエージェント間の通信は，ベースエージェントを介して行う必要がある．ベースエージェント同士の通信に関しては通信相手の制限は少ないので，ページエージェントとベースエージェント間の通信が確立されれば，ページエージェント間の通信が実現される．

ページの更新を伴わない，ページエージェントとベースエージェント間の実現方法について述べる．本システムでは，ブラウザから XML 文章を読み込むための組込み関数である xmlHTTPRequest 関数を利用する．近年では，AJAX[?] という名前で知られており，Web 上の非同期な．ベースエージェントに対して，送信メッセージを HTTP の POST として送信し，受信メッセージを XML 文章中として受信することで送受信を行う．

4.2 永続性と継続的な実行

ページエージェントの永続性を実現するには，ページの更新前にページエージェントの状態を保存する方法と，ページの更新後にページエージェントを再構築する方法を実現する必要がある．ページの更新タイミングの取得には，HTML の標準仕様で定義されている，ページが閉じられるときに活性化されるイベントハンドラを利用する．ページエージェントの状態の保存は次の 3 ステップで構成される．(s1) ページエージェントの状態をメッセージ通信で送信可能なデータ形式で表現，(s2) メッセージ通信でページエージェントのデータをベースエージェントに送信，(s3) ベースエージェント内のページエージェントのデータベースを更新しページエージェントの再

構築に備える．

ステップ (s1) では，ページエージェントの情報を次のような形式の XML 文章に変換する．

```
<pageagent>
  <meta>
    <name>エージェントの名前</name>
    <id>エージェントの ID</id>
    <base>ベースエージェントの URL</base>
  </meta>
  <body>
    エージェント内部データ
  </body>
</pageagent>
```

ページエージェントの情報は，大きく分けて meta で表されるページエージェントのメタ情報と，body で表されるページエージェントの定義及びデータ，の 2 つの部分から構成される．メタ情報としては，エージェントの名前や ID，そして，ベースエージェントの URL などが格納される．エージェントの定義には，ページエージェント内のプログラムやデータが格納される．

ステップ (s2) では，メッセージ通信機構を使ってステップ (s1) で得られた XML 文章をベースエージェントに送信し，ステップ (s3) でベースエージェントはページエージェントを表す XML 文章を保存し，再構築に備える．場合により，保存先はクッキーのような手段でもよい．

ページエージェントの再構築は，静的アプローチとプロキシアプローチでは可能である．動的アプローチでページエージェントを再構築する方法に関しては今後の課題である．ページエージェントの再構築は，特別な処理は無く，通常のページにエージェントのプログラムを付加するのと同様に，ページにエージェントのプログラムを付加するだけである．

実際にブラウザ上で動作させるためには，2 で述べた手法を利用することで継続的な実行を実現する．すなわち，エージェントのプログラムには，実行環境を保存するオブジェクトを導入し，末尾再起処理による多重実行および実行状態の保存を適用する．

4.3 環境とのインタラクション

ページエージェントのセンサとアクチュエータの実装について述べる．センサとアクチュエータは，ページエージェントの外界，すなわち，ページとユーザからのインタラクションを扱う．ページは，HTML で記述されている．ページエージェントは，HTML 文章をテキストとして操作することが可能である．ブラウザ内のページは，Document Object Model (DOM) におけるオブジェクトとしても操作可能である．DOM は，W3C により標準化された HTML 文章のオブジェクトとしての表現である．ページエージェントは，ページをオブジェクトとしても操作可能であり，同様に，ページ内のオブジェクト（テーブルなど）を自由に扱うこともできる．これにより Web Paper を容易に実現できる．ユーザからのインタラクションは，ページの変化として処理することも可能であるが，文字入力やマウスの移動・クリックなどのより低レベルな処理としても処理可能である．どのように外界とのインタラクションを行うかは，ページエージェントの定義により異なる．

5 おわりに

本論文では，知的で高度なページを実現するためのエージェントシステムである MiSpider における継続的な実行の方法と実装について述べた．イベント処理用に設計された Javascript 言語の組込み関数を適切に利用することで，Javascript プログラムの多重処理と実行状態の保存が可能なことを示し，これに基づくエージェントの実装方法について述べた．

本研究では，Web ページの知的化を目指しており対象としており，機械可読な Web ページの実現を目指す Semantic Web とは異なるが，Web ページの知的化という観点では，将来的に Semantic Web 技術も利用できると考える．

参考文献

[1] 西健太郎, 新谷虎松, 松尾徳朗, 田代慎治, 伊藤孝行, ” 既存 Web ブラウザを利用したオンライン編集可能な Web ページの実現,” 電気学会論文誌 (部門誌)C, 電気学会, Apr. 2005.

[2] Y. Fukagaya, T. Ozono, T. Ito and T. Shintani, “MiSpider: A Continuous Agent on Web Pages ,” WWW2005, pp.1008-1009, 2005.

[3] Ajax: A New Approach to Web Applications, http://www.adaptivepath.com/publications/essays/archives/000385.php

[4] S. Potter, J. Nieh, “WebPod: Persistent Web Browsing Sessions with Pocketable Storage Devices,” WWW2005, pp. 603-612. 2005.