社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

KawaWiki: 人間とエージェントが協調する Semantic Wiki

河本　健作†　　北村　泰彦†　　YuriTijerino††

† 関西学院大学 理工学部　〒 669-1337 三田市学園 2-1
†† 関西学院大学 総合政策学部　〒 669-1337 三田市学園 2-1
E-mail: †{kkensaku,ykitamura}@ksc.kwansei.ac.jp, ††yuri@tijerino.net

あらまし　Wiki とは，ネットワーク上の不特定多数の人間同士がブラウザを介して協調しながら編集活動を行うことで，Web サイトを共同で制作するためのシステムである．エージェントが Wiki 上の情報を理解し，Wiki 上の編集活動に参加することができれば，整合性のとれたコンテンツの作成，Web 情報と Wiki 情報の統合など，従来の Wiki では不可能であった効果が期待できる．本論文では，Semantic Web を共通の情報表現として，エージェントと人間の協調を実現した新たな Wiki システムを提案する．

キーワード　Semantic Wiki，Semantic Web，WikiWikiWeb，エージェント

KawaWiki: Semantic Wiki Where Human and Agents Collaborate

Kensaku KAWAMOTO†, Yasuhiko KITAMURA†, and Yuri TIJERINO††

† School of Science and Technology, Kwansei Gakuin University　2-1 Gakuen, Sanda 669-1337, JAPAN
†† School of Policy Studies, Kwansei Gakuin University　2-1 Gakuen, Sanda 669-1337, JAPAN
E-mail: †{kkensaku,ykitamura}@ksc.kwansei.ac.jp, ††yuri@tijerino.net

Abstract　Wiki is a collaborative Web page authoring system. By using a Wiki system, we can build a Web site collaboratively by creating and updating Wiki pages through Web browsers. If agents can understand and update Wiki pages, they can build a consistent Web site and integrate Wiki pages with Web information. We propose a new Semantic Wiki system where human and agents can collaborate through Semantic Web.

Key words　Semantic Wiki, Semantic Web, WikiWikiWeb, Agent

1. はじめに

WikiWikiWeb（略して Wiki）とは，1990 年代に Ward Cunningham 氏によって考え出された Web ページ共同制作のためのシステムである [1]．ネットワーク上の不特定多数の人間同士がブラウザを介して他の参加者と協調しながら編集活動を行うことができる．Wikipedia に代表される従来のテキストベース Wiki ではユーザ同士が役割分担を行いながら，特定のトピックについての新たな Wiki ページを作成したり，他のユーザが作成した Wiki ページの一部を編集したり，誤りを修正したりしながらコンテンツが構築されている．

エージェントが Wiki 上の情報を理解して，Wiki 上の編集活動に参加することができれば，整合性のとれたコンテンツの編集が可能になる，他サイトの情報を統合することができるなど，従来の Wiki にはない効果が期待できる．

本論文では，Semantic Web を共通の情報表現とし，エージェントと人間の協調を実現した新たな Wiki システムを提案する．第 2 章では，関連研究として従来の Semantic Wiki を紹介する．第 3 章では，エージェントと人間が協調する Wiki の提案を行う．第 4 章では，本システムの実装について記述する．第 5 章では，スケジュール Wiki を例にして本システムの具体的な利用例を紹介する．最後に，第 6 章で本論文の結論を述べる．

2. Semantic Wiki

Semantic Web とは，W3C の Tim Berners-Lee 氏によって提唱され，次世代 Web システムの標準として期待されている枠組みである [2], [3]．Web 情報に意味タグを付け加えることで，コンピュータによる自動的な情報の収集や検索を正確でかつ効率的に処理することが可能になる．

Wiki の枠組みの上で Semantic Web 情報を取り扱ったものが Semantic Wiki である．現在，いくつかの Semantic Wiki が開発されており，Semantic Web オントロジを生成すること

を主な目的とするもの [4]，従来のテキストベース Wiki への Semantic Web 情報の付加を支援するもの [5] などが存在する．しかし，これらは Semantic Web 形式での情報発信や Semantic Web を利用した情報検索を主な目的とするものであった．本システムは Wiki 上でのエージェントと人間の協調を実現するシステムの実現を目指す．

3. エージェントと人間が協調する Wiki

従来の Wiki では，ユーザは自分のできる範囲で Wiki に貢献するという仕組みが成り立っている．自分の専門とする情報や知っている情報を基にして Wiki ページを記述することで，不特定多数のユーザ同士が協調し合いながら，Web ページの共同制作を行うことができる．また，ユーザは他のユーザが編集した Wiki ページの誤りを見つけてそれを修正したり，他の Wiki ページや Web 情報などの更新を基にして，Wiki ページの内容を書き換えて更新する作業を行う必要がある．

エージェントが Wiki 上の編集活動に参加し人間との協調を行うことで，以下のような利点が生まれる．

3.1 情報の自動更新

ユーザがひとつの Wiki ページの更新を行うと，エージェントが関連のあるすべての Wiki ページの内容を自動的に更新することができる．エージェントが Wiki に参加することで，エージェントが自動的に Wiki 上の情報検索を行い，得られた情報を基にして関連するすべての Wiki ページを書き換えることで，情報を更新することができるようになる．

例えば，研究室内で利用する Wiki において特定のプロジェクトの Wiki ページに，そのプロジェクトに所属しているメンバーの一覧を示す項目があるとする．従来のテキストベースの Wiki では，プロジェクトに新たなメンバーが参加したり，プロジェクトの所属メンバーに変更が生じた場合に，編集者が各個人ページから情報収集を行い，正しい状態にするためにすべての Wiki ページの更新を行う必要があった．一方，エージェントが Wiki に参加することによって，あるメンバーが自分の所属するプロジェクトを変更するだけで，エージェントが自動的にそれを検索し，プロジェクトページの所属メンバーの一覧の項目を更新することができる．

3.2 エージェントによる情報統合

Wiki 上の情報や Web 上の情報を統合し，ユーザのために様々な表示を行うことができる．従来のテキストベース Wiki では，Wiki 情報と Web 上の情報を参照したり統合するためには，Wiki ページ上から Web サイトへのリンクを作成したり，Web 上の情報をコピーして Wiki 上に直接書き込む必要がある．エージェントが Wiki に参加することで，エージェントが Web 上の情報を自動的に参照し，Wiki 上の情報と統合してユーザに表示することができる．また，常に Web 上の情報の監視を行うことで，情報が更新された際にそれらの情報を Wiki ページに反映させることができる．

例としては，Wiki 上に存在する各個人の予定と公式的な予定を統合してひとつのカレンダー上に表示することが挙げられる．また，随時更新される Web 上の祝祭日情報を統合して表示したり，旅行などの予定に対しては天気予報と併せて表示したりするといった利用が可能である．

3.3 意味的な整合性検証

Wiki 上の情報の整合性を検証し，その内容に矛盾が生じた場合はユーザに対して警告を行ったり解決策の提案を行うことができる．従来の Wiki システムでは，不特定多数の編集者が Wiki 上の情報の更新を行っているために，Wiki ページの情報に矛盾が生じることが多い．矛盾が起きた情報は他のユーザがそれに気づき，修正が行われるまではそのままの状態で置かれる．

本システムでは Wiki ページ同士，そして Web 上の Semantic Web 情報との意味的な整合性検証を行うことができる．矛盾を検出するためのエージェントが存在し，Wiki ページ上の整合性を常に監視している．エージェントは，矛盾が生じていたり修正をすべき事態が発生した場合にはユーザに対して修正を求めるために警告表示を行うことができる．また，矛盾を解消するための代替案をエージェントが推論し，それをユーザに対して提案することもできる．

具体的な整合性検証の例としては，Wiki 上に存在する各個人のプライベートな予定と，ゼミや打ち合わせなどの研究室の公式的な予定が衝突しているか検証を行うことが挙げられる．もし衝突していれば，プライベートな予定を変更するようにユーザに警告を行うことができる．また，各個人の予定を自動的に取得し，次のゼミを何時に設定すればよいかをユーザに提案することも可能である．

4. KawaWiki

人間とエージェントの協調を行うために，Wiki 上に存在する情報をエージェントが理解する必要がある．従来のテキストベース Wiki は，人間が理解するための自然言語によって記述されているため，エージェントがそれを理解することは困難である．そこで，Wiki 上の情報に Semantic Web 情報を付加することで，エージェントが Wiki を理解する仕組みを実現する．Semantic Web を仲介することでエージェントと人間の協調が可能になる．

一方で人間が Semantic Web 情報を生成することは容易ではない．そこで，Semantic Web 情報を生成するためのテンプレートシステム [6], [7] を提案する．またエージェントはユーザが生成した Semantic Web 情報や Web 上にある Semantic Web 情報を検索して取得するために Semantic Web クエリ言語である SPARQL を利用し，エージェントの動作を記述するためにスクリプト言語である PHP を利用する．

4.1 テンプレートシステム

Semantic Web を利用することで，コンピュータによる自動的な情報の収集や検索を正確でかつ効率的に処理することが可能になる．しかしながら，従来の Semantic Web システムは一般ユーザが使用することが困難であるために，学術機関や企業などによる Web サービスを目的としたものが主流であった．本システムは，テンプレートシステムを用いることで一般ユーザでも簡単な操作で Semantic Web の表現形式である RDF の生成をすることができる仕組みを持つ．

テンプレートシステムでは専門知識を持ったユーザが記述したテンプレートに対して，一般ユーザが値やリソースなどの具体的なデータを穴埋めするだけで RDF データと HTML ページ（Wiki ページ）が生成される．一般ユーザによるデータの入力は，複雑な RDF 構文を覚える必要がないように，Web 上のアンケートや個人情報の入力画面などで見られるような，属性名に対応する属性値をフォームから選択したり，直接入力する方式で行うことが可能である．これにより，専門知識を持ったユーザと持っていない一般ユーザの役割分担を行うことができ，多くのユーザが積極的に Semantic Wiki による情報発信が可能になる．

図 1 は，個人プロフィールの Wiki ページ（RDF データ）を作成するためのテンプレートを示している．RDF 形式の記述の中には，氏名を示す`<%Name\\/^.+$/%>`（8 行目），趣味を示す`<%Hobby\\/^.+$/%>`（9 行目），自分の所属するプロジェクトを示す`<%Project\\project%>`（11 行目）という変数の定義がされている．

これを基にしてユーザが値を入力するための編集フォーム（図 2）が生成される．変数 `Name` と `Hobby` の値の取り得る範囲は，一文字以上の任意の文字列を示す正規表現である`/^.+$/`と記述されており，編集フォームにはテキストボックスが表示される．変数 `Project` は取り得る範囲が project クラス（project テンプレートを利用して作成された Wiki ページの集合）となっており，編集フォームではプルダウンメニューからプロジェクトを選択することができるようになっている．もし定義された値の取り得る範囲と実際にユーザが入力した値が異なっている場合は，画面上に警告表示を行うことでユーザに修正を促す．テンプレートで，変数 `Project` は`&list();` というタグに括られており（10〜12 行目），これは変数 `Project` が複数の値をもつことができるということを示している．編集フォームでは値を入力する度に新たなボックスが追加されるようになっている．ユーザが編集フォームから値を入力して Wiki ページの更新を行うと，図 3 のように Wiki ページと同時に RDF が生成される．

また，従来のテキストベースの Wiki と同様のインタフェースから，タグを利用した入力を行うことでデータの穴埋めを行うこともできる．図 4 では，`[[変数名:値]]` という形式で変数への値の入力が行われている．タグにより記述された部分はエージェントのための RDF として生成され，その他のテキスト部分は人間のための自然言語として Wiki ページ上にそのまま表示される．タグを利用した入力の場合でも，テンプレート上の定義と入力された値の間に誤りがあった場合は警告表示を行う．

```
01: <?xml version="1.0"?>
02: <rdf:RDF
03:  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04: xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
05:  xmlns:wiki="http://kawamoto/kawawiki_agent/kawawiki.rdf#"
06: >
07:  <wiki:Person <%RDFID%>>
08:   <wiki:Name rdf:parseType="Literal"><%Name\\/^.+$/%></wiki:Name>
09:   <wiki:Hobby rdf:parseType="Literal"><%Hobby\\/^.+$/%></wiki:Hobby>
10: &list(
11:   <wiki:Project rdf:resource="<%Project\\project%>"/>
12: );
13:  </wiki:Person>
14: </rdf:RDF>
```

図 1　テンプレートの例

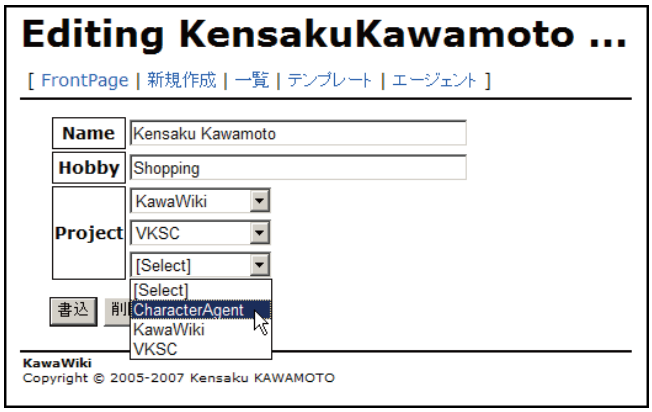

図 2　テンプレートから生成された編集フォーム

生成された Wiki ページ

生成された RDF

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:wiki="http://kawamoto/kawawiki_agent/kawawiki.rdf#"
>
 <wiki:Person >
  <wiki:Name rdf:parseType="Literal">Kensaku Kawamoto</wiki:Name>
  <wiki:Hobby rdf:parseType="Literal">Shopping</wiki:Hobby>
  <wiki:Project rdf:resource="http://kawamoto/.../rdf/KawaWiki"/>
  <wiki:Project rdf:resource="http://kawamoto/.../rdf/VKSC"/>
 </wiki:Person>
</rdf:RDF>
```

図 3　生成された Wiki ページと RDF

4.2 SPARQL クエリによる情報更新

本システムでは Wiki 上の情報検索を行い，得られた結果からページの更新内容を決定する仕組みを SPARQL クエリによっ

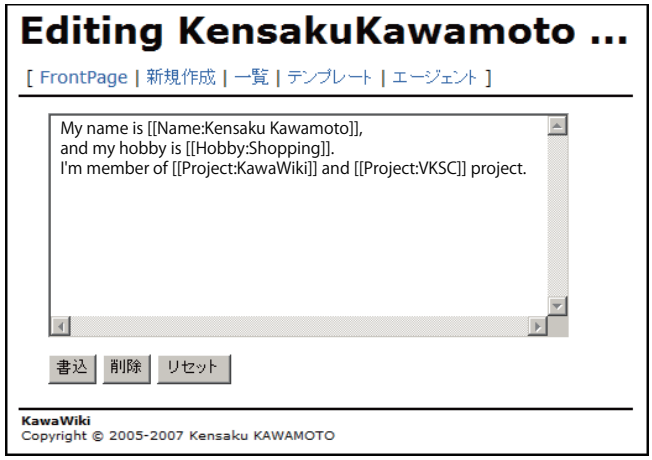

図 4　タグを利用した入力

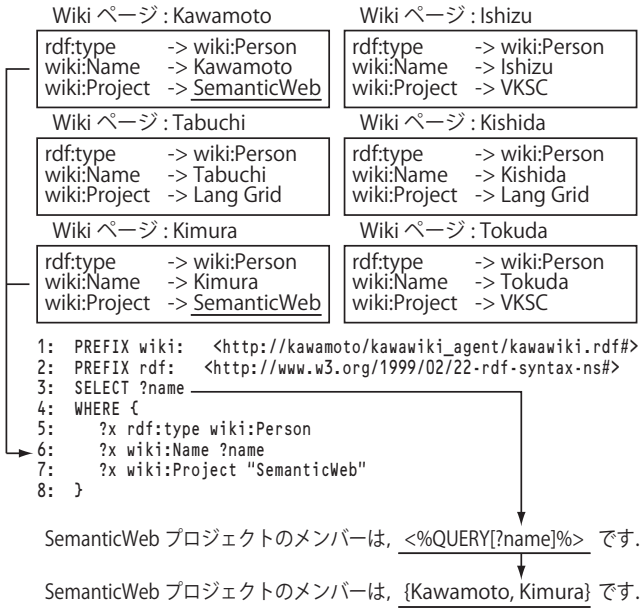

図 5　メンバー抽出クエリと表示例

て実現している．図 5 は具体的なクエリ例とその結果から Wiki ページが更新される様子を示している．この例では，あるプロジェクトに所属しているメンバーの一覧を取得している．各個人ページを参照し，自分が所属するプロジェクトに SemanticWeb と記述している人の名前 (wiki:Name) を?name という変数にマッピングして抽出する．抽出した結果は SemanticWeb プロジェクトに所属しているメンバーの一覧であるので，エージェントは Wiki ページの更新を行うことができる．

4.3　SPARQL クエリによる整合性検証

意味的な整合性検証のために，ユーザはあらかじめ矛盾が生じると想定されるクエリを記述しておく．そのクエリにヒットするような RDF グラフは矛盾が起きていると判断することができる．そのクエリをエージェントスクリプトに記述しておき，矛盾が起きた場合はユーザに警告を行ったり，解決策の提案を行ったりすることができる．

図 6 に，個人的な予定 (private) と公式的な予定 (seminar, meeting) が同じ日に衝突しているような個人的な予定を取得するクエリを示す．6 行目〜10 行目で予定のタイプ (wiki:Type) が seminar か meeting である予定をすべて取得し，その公式的な予定の日時 (wiki:Time) と内容 (wiki:Summary) を変数 (?time_official) と (?summary_official) として取得している．同様にして，11 行目〜15 行目では個人的な予定の日時と内容を変数 (?time_private) と (?summary_private) として取得している．16 行目では衝突している予定を絞り込むために，個人的な予定の日時と公式的な予定が同じである予定のみを抽出している．最終的に取得できる結果が衝突している予定の日時と内容であると判断できる．その結果を利用することでエージェントはユーザに対して警告を表示することができる．

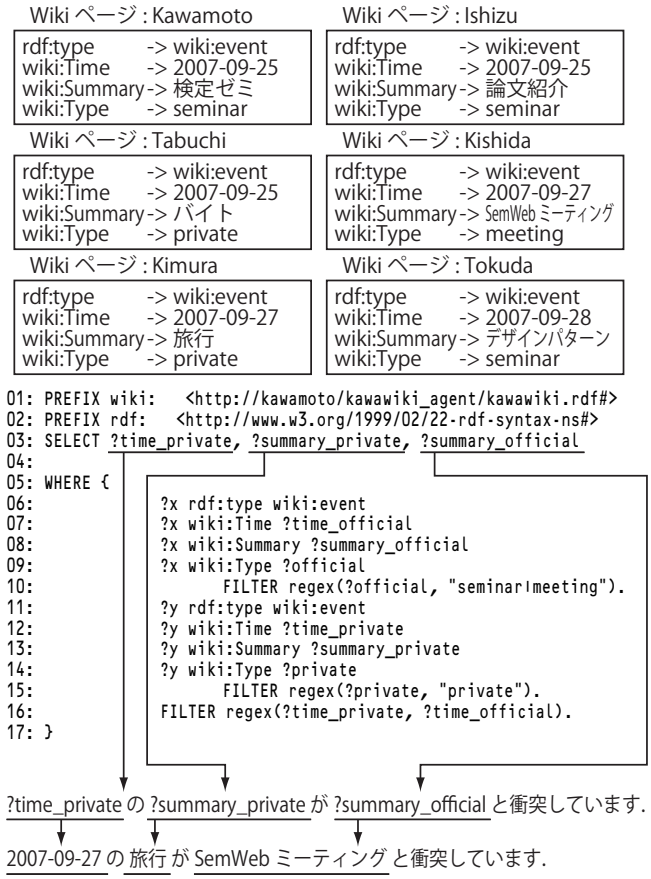

図 6　クエリによる整合性検証

Wiki 上からクエリを利用した際に，クエリ呼び出しがループする問題が生じる可能性がある．つまり，クエリ結果 A を生成する際には必ず他の Wiki ページ (RDF) を参照する必要があるが，参照した他の Wiki ページもクエリによって生成されており，そのクエリ結果 B を生成する際に再びクエリ結果 A を参照するような記述になっている場合である．このループ問題を回避するために，Wiki 上でのクエリ呼び出しの様子が記述されたクエリツリーを生成し，それに基づいてループが生じないことを保証する．本システムではクエリを含む RDF グラフとそのクエリを基にしたクエリツリーを保持しており，クエ

リを含む Wiki ページの更新時にループのチェックを行う．ループが生じる場合には，エラー表示を行うことでループを回避する．また，クエリツリーを利用することで，クエリの実行時に子クエリ，孫クエリにクエリの実行が伝播するようになる．

4.4 エージェント記述言語

Wiki の思想に従い，エージェントは人間のユーザによって Wiki 上から自由に動作内容を書き換えることで拡張を行うことができる．エージェントの書き換えは専門知識を持ったユーザが行い，RDF 構文や SPARQL クエリ言語などの Semantic Web の知識や，PHP のプログラミング言語の知識を必要とする．エージェントの呼び出しはあらかじめユーザに指定された次の 4 つのタイプの「トリガ」によって実行される．

- 常に実行
- 一定時間毎
- エージェントエリアからのユーザ呼び出し
- 新規作成や更新などの特定の処理時

図 7 は整合性検証を行うエージェント（eventrule エージェント）の記述例を示している．この整合性検証エージェントは，ユーザが Wiki を呼び出す度に常に実行されるように設定されている．エージェントはひとつのクラスとして定義され，KawaWiki 側から `execute` メソッドが呼ばれることにより実行される．

4 行目〜24 行目には図 6 で示したものと同様に個人の予定と公式的な予定が同じ日に衝突しているような予定を取得するクエリが記述されている．25 行目で実際にクエリによる問い合わせを行い，その結果が`$result` 配列として返される．28 行目〜35 行目で`$result` 配列の内容を順番に走査し，衝突するような予定があればその結果に基づいて，30 行目〜33 行目の記述で警告メッセージが生成される．最後に 36 行目で警告メッセージの配列が Wiki 側に返され，それがユーザに警告として表示される．

```
01: <?php
02: class eventrule{
03:   function execute(){
04:     $query = <<<EOD
05:     PREFIX wiki:<http://kawamoto/.../kawawiki.rdf#>
06:     PREFIX rdf:<http://www.w3.org/.../22-rdf-syntax-ns#>
07:     SELECT
08:            ?summary_official,
09:            ?summary_private,
10:            ?time_private
11:     WHERE {
12:         ?x rdf:type wiki:event
13:         ?x wiki:Time ?time_official
14:         ?x wiki:Summary ?summary_official
15:         ?x wiki:Type ?official
16:           FILTER regex(?official, "seminar|meeting").
17:         ?y rdf:type wiki:event
18:         ?y wiki:Time ?time_private
19:         ?y wiki:Summary ?summary_private
20:         ?y wiki:Type ?private
21:           FILTER regex(?private, "private").
22:         FILTER regex(?time_private, ?time_official).
23:     }
24: EOD;
25:     $result = query_sparql($query);
26:
27:     $output = Array();
28:     if(!empty($result)){
29:       foreach($result as $value){
30:         $output[] = $value["?time_private"]->label . " の"
31:           . $value["?summary_private"]->label . " が"
32:           . $value["?summary_official"]->label
33:           . " と衝突しています．";
34:       }
35:     }
36:     return $output;
37:   }
38: }
39: ?>
```

図 7　エージェントの記述例

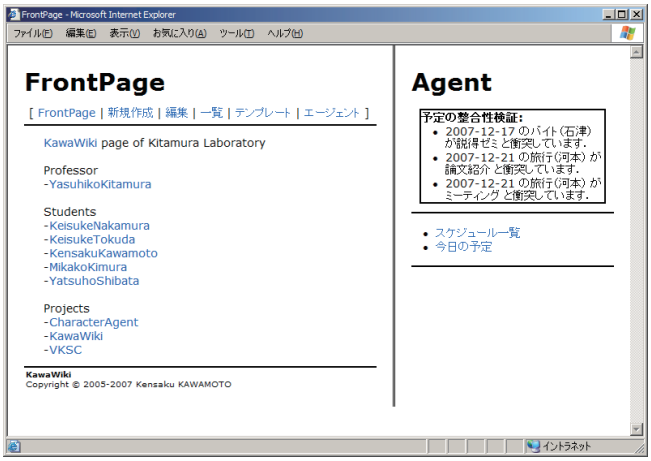

図 8　Wiki エリアとエージェントエリア

5. KawaWiki プロトタイプ

本章では，複数のエージェントが活動し，ユーザとの協調を行っている様子をスケジュール Wiki を例にして紹介する．

本システムではエージェントとの協調のために，図 8 のように従来の Semantic Wiki ページの表示を行う Wiki エリア（左）に対して，エージェントが様々な活動を行うためのエージェントエリア（右）が用意されている．エージェントはエージェントエリア上でユーザに対して何らかの表示を行ったり，ユーザからの呼び出しを待機する．

ユーザは Wiki ページの追加や編集を行うことで，予定の追加や編集を行うことができる．ユーザが作成した予定が記述されている Wiki ページは，テンプレートシステムを利用することで，エージェントにも理解可能な RDF 情報が同時に自動的に生成される．

ユーザはエージェント管理画面からエージェントの管理を行うことができる（図 9）．有効になって活動しているエージェント以外にも複数のエージェントが存在しており，ユーザは必要に応じて様々なエージェントを切り替えて Wiki を利用することができる．新たなエージェントの追加や，エージェントの処理内容の変更などもこの画面から行うことができる．

Wiki 上には次のようなエージェントが活動している．

（1） Web 上から天気予報を収集し，Wiki ページに書き込むエージェント

（2） Web 上から日本の休日情報を取得し，Wiki ページに書き込むエージェント

（3） Wiki 上から予定をスケジュール一覧や天気予報，日本の休日を取得し，統合して表示するエージェント

（4） Wiki 上の予定の整合性を監視するエージェント

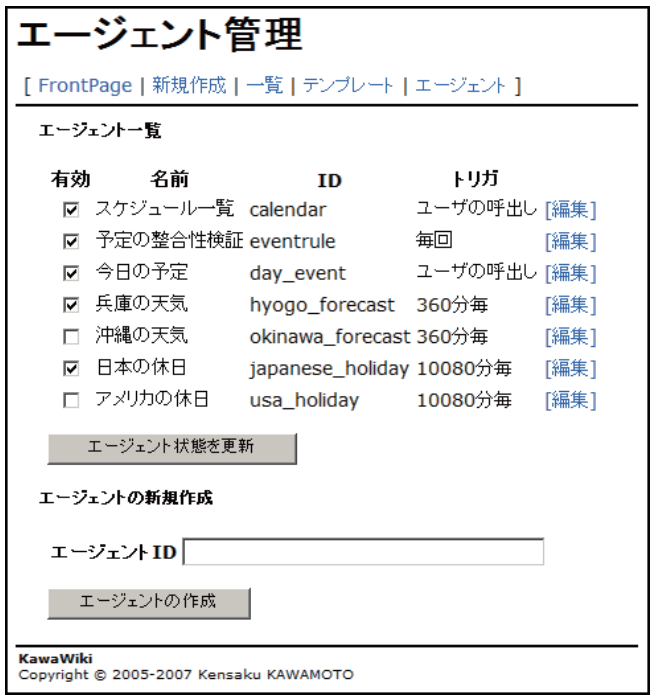

図 9　エージェント管理画面

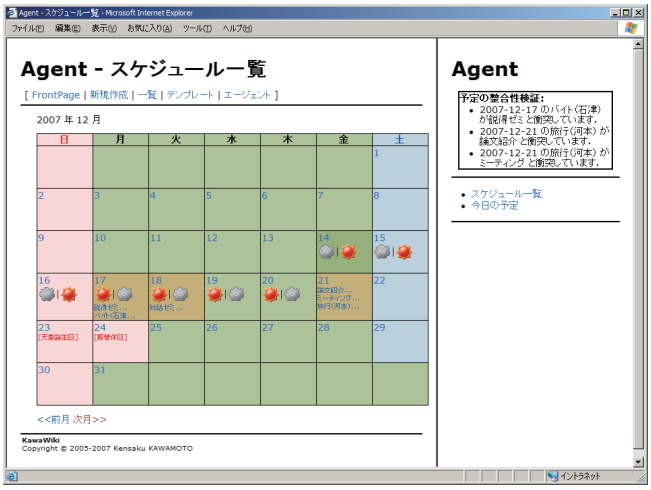

図 10　カレンダー表示エージェント

（1）と（2）のエージェントは一定時間毎に実行され，Web 上の情報が更新されていれば最新の情報を基にして Wiki ページの更新を行う．今回は，気象庁の Web サイトをラッピングすることで天気予報を収集し，Google カレンダーからは XML 形式で配信されている日本の休日情報を取得している．

（3）のエージェントは（1）と（2）のエージェントが作成した Wiki ページとユーザが作成した Wiki ページを参照し，それらをカレンダー上に統合して表示を行う（図 10）．

（4）のエージェントはユーザが作成した Wiki ページを参照し，予定間に衝突が発生するなどの矛盾が生じている場合にはエージェントエリアを利用してユーザに対して警告表示を行う．図 10 の画面では右側のエージェントエリア内で整合性検証エージェントがユーザに対して警告表示を行っている様子がわかる．ユーザはエージェントからのこれらの警告を見て予定の調整を行う．

以上のように，ユーザが Wiki ページを記述すれば，テンプレートシステムを利用して Semantic Web 情報が生成され，SPARQL クエリを利用してそれをエージェントが理解し，Wiki ページの内容に対して提案や警告をしたり更新を行うといった，人間とエージェントの協調作業がこの Wiki 上で行われている様子がわかる．

6.　まとめと今後の課題

本論文では，エージェントと人間の協調を実現する新たな Wiki システムの開発を目指した．エージェントが Wiki 上の情報を理解し，Wiki 上の編集活動に参加することで得られる利点として，情報の自動更新，エージェントによる情報統合，意味的な整合性維持を提案した．それを実現するためにエージェントと人間の共通情報表現として Semantic Web の枠組みを利用し，エージェントが情報を収集するための Semantic Web クエリ言語として SPARQL を，人間が Semantic Web 情報を生成するために，テンプレートシステムを利用した．また，人間がエージェントの動作を記述するためにスクリプト言語である PHP を利用した．

今後の課題としては，スケジュール管理エージェントに留まらず他の具体的応用例を提案し，利用することで本システムの評価を行うことが挙げられる．応用例としては，Semantic Wiki 上でのユーザとエージェントの協調作業によって，海外のレシピを自分の国で調理可能な内容に変換する Cooking Wiki，エージェントとの協調作業によって，ユーザの好みに合わせた旅行プランを作り上げる旅行支援 Wiki などが挙げられる．

文　献

[1] Bo Leuf and Ward Cunningham. *Wiki Way*. ソフトバンクパブリッシング, 2002.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May 17, 2001.

[3] 神崎正英. セマンティック・ウェブのための RDF/OWL 入門. 森北出版, 2005.

[4] Jochen Fischer, Zeno Gantner, Steffen Rendle, Manuel Stritt, and Lars Schmidt-Thieme. Ideas and Improvements for Semantic Wikis, *Proceedings of ESWC 2006*, pp.650-663, 2006.

[5] Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer. Semantic Wikipedia, *Proceedings of the 15th international conference on World Wide Web, WWW 2006*, Edinburgh, Scotland, May 23-26, 2006.

[6] Kensaku Kawamoto, Yasuhiko Kitamura and Yuri Tijerino. KawaWiki: A Template-Based SemanticWiki Where End and Expert Users Collaborate, *5th International Semantic Web Conference*, Poster paper, 2006.

[7] Kensaku Kawamoto, Yasuhiko Kitamura, and Yuri Tijerino. KawaWiki: A Semantic Wiki Based on RDF Templates, *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology - Workshops*, 425-432, 2006.