日本ソフトウェア科学会第 27 回大会 (2010 年度) 講演論文集

AjWebシステム: AjaxアプリケーションのXML記述による自動生成

熊本 浩紀　野呂 智哉　徳田 雄洋

近年，HTML5 などが提案され，インタラクティブな UI やリアルタイムなデータの更新，オフラインでの動作などデスクトップアプリケーションと Web アプリケーションの両方の特徴を合わせ持つ Ajax アプリケーションを作成できるようになった．しかし，そのような Ajax アプリケーションを作成するためには，極めて高い専門知識が必要であり，作成を自動化，支援するツールの必要性がある．しかしながら，従来の Web アプリケーションの自動生成手法は，サーバサイドの仕事を主，クライアントサイドの仕事を副とするアプリケーションを対象にしており，そのような現代的な Ajax アプリケーションに対しては十分ではない．本研究では，現代的な Ajax アプリケーションの振る舞いを記述する XML 言語を提案し，自動生成を行うシステムを提案する．

1　はじめに

インターネットが普及するとともに，さまざまな Web アプリケーションが作成され，そのプラットフォームも発達してきた。Web アプリケーションは，配布や更新が簡単であることやメンテナンスが容易であるなどのメリットがあり，ブログや電子掲示板，ショッピングサイトなどで利用されてきた．

しかし，従来の Web アプリケーションは，操作性やユーザインターフェースが乏しいなどのデメリットがあった．従来の Web アプリケーションに画面の部分更新やサーバプッシュ機能やドラッグ&ドロップなどの操作性，表現力を高める機能を持たせたものがリッチインターネットアプリケーション（RIA）である．現在，さまざま RIA のプラットフォームが提案されている．Ajax [14]，Adobe AIR [7]，Silverlight [10]、JavaFX [13] などそれに当たる．

さらに近年，Gears [8] や HTML5 [15] の web database や [7] の SharedObject，[10] の Isolated Strage など，RIA に対してクライアント側にも永続化データを持たせたり，オフラインでも動作するような拡張がなされている。

これにより，Google Docs や Yahoo! Mail などのようにデスクトップアプリケーションと従来の Web アプリケーションの両方の特徴をもった現代的な Web アプリケーションを作成可能になった．本研究では，そのような Web アプリケーションを現代的な Ajax アプリケーション，あるいは単に，Ajax アプリケーションと呼ぶこととする．

Web アプリケーションの開発は，コードがサーバサイド，クライアントサイドの両方に存在し，それらを組み合わせて開発する必要がある．Ajax アプリケーションの開発ではさらに，イベントドリブンな UI の記述や，クライアント側のデータとサーバ側のデータの同期などに考慮する必要があり，アプリケーションの主となる振る舞いとは直接関係のない部分の処理に大きな労力をかけなければならない．したがって，アプリケーションの振る舞いをモデル化して記述し，アプリケーションまたはその一部を自動生成するツールが有効である．

現在，RIA をモデルから自動生成するための記述方式，および自動生成手法が提案されている [2] [3] [5] [6]．これらの手法では，サーバサイドの仕事を主，ク

AjWeb System: Automatic generation of Ajax applications using XML descriptions

Hiroki KUMAMOTO, Tomoya NORO, Takehiro TOKUDA, 東京工業大学大学院情報理工学研究科, Dept. of Computer Science, Tokyo Institute of Technology.

ライアントサイドの仕事を副とするアプリケーションを生成対象にしており，現代的な Ajax アプリケーションに対しては必ずしも十分でない．

本研究では，現代的な Ajax アプリケーションの振る舞いを記述する XML 言語を提案し，記述からアプリケーションを自動生成するシステムを提案する．

本論文の構成は、以下の通りである。2 節で生成の対象となる Ajax アプリケーションの特徴を述べる。3 節で記述言語 AJML について述べ、4 節では、生成システムと実装のプロトタイプについて述べる。5 節で，他の自動生成システムとの比較を行う．6 節で関連研究について述べ、7 節でまとめと今後の課題について述べる。

2 生成の対象とその特徴

2.1 対象とする Ajax アプリケーションの特徴

ここでは，生成の対象とする Ajax アプリケーションの特徴を述べる．Ajax アプリケーションの特徴としては，インタラクティブな UI があげられる．従来の Web アプリケーションは，ハイパーリンクによるページ遷移やフォーム送信によるデータの送信などによってアプリケーションが動作していた．したがって，UI は単純な入力フォームや結果データの表示などに制限されていた．Ajax アプリケーションでは，クライアントサイドの JavaScript により，マウスのクリックやキーボードの入力，サーバからのメッセージなどさまざまイベントに対して処理を呼び出すことができ，よりインタラクティブな UI を記述することが可能である．画面の部分更新，リアルタイムなデータの更新，ページング，入力補完，ドラッグ&ドロップなどが挙げられる．

また，インタラクティブな UI に加えて，クライアントに永続化データをもつことが特徴として挙げられる．図 1 のようにクライアント側でサーバ側のデータの一部を永続化データとして保持し，主な処理をクライアント側で行うことができる．これによりオフライン時にも動作するようなアプリケーションが作成可能である． オフライン時にはクライアントのデータのみに変更を加え，オンラインになったときにサーバのデータと同期を取り, サーバにクライアントのデータの変更を反映させる．このようなアプリケーションでは，サーバプログラムはデータの作成，保存，更新，取得，同期処理などが主である．

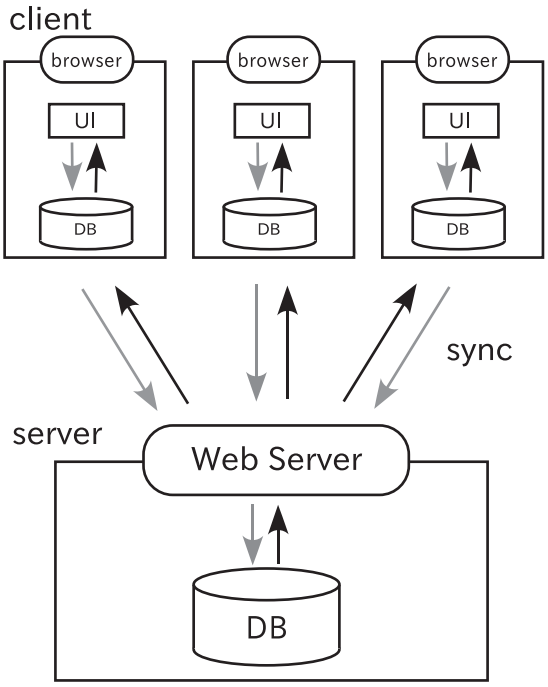

図 1 対象とするアプリケーションのアーキテクチャ

従来の Web アプリケーションのようにサーバサイドで複雑な処理を行うこともできるが，本研究では，生成の対象をこのようなクライアント主体のアプリケーションに絞る．

2.2 ケーススタディ

例として，複数のクライアントが一つのデータベースを共有して，お互いに予定をを書き込んでいくカレンダーアプリケーションを考える．一人のクライアントが予定を書き込むと，サーバにデータが送られ，サーバから他のクライアントへ予定の書き込みがリアルタイムに反映される．また，回線が不安定で一時的に回線が途切れたなどの理由でクライアントがオフライン状態の場合，クライアントの予定の書き込みは，すぐにはサーバに送信されず，クライアントのローカルなデータベースに保存される．再度オンラインになったときに送信され，サーバや他のクライアン

トにも書き込みが反映される．このときオフラインの状態の間に別のクライアントが書き込みを行っていた場合，その変更が受信され同期が行われる．

3 XML による Ajax アプリケーションの記述

この節では提案する XML 言語 AJML(Ajax Application Markup Language) について述べる．まず，最初に AJML の全体像について述べ，次に 2 節で述べたカレンダーアプリケーションを例にして，記述例を示しながら説明を行う．

3.1 全体像

全体像について述べる．AJML は大きく 3 つの部分からなる．UI モデルを記述する interfaces 要素，データモデルを記述する databases 要素，イベントモデルを記述する events 要素の三つである．同期の記述は databases 要素内の observe 要素で行う．AJML の全体の構造を図 2 に示す．次にそれぞれの要素について順に記述例を挙げながら説明していく．

3.2 データベースモデル

提案する AJML では RDBMS のテーブル型のデータを想定したデータモデル定義する．テーブルの構造は，子要素の schema 要素および property 要素で定義する．また，クライアント，サーバのどちらにデータベースが存在するかは type 属性で指定を行う. カレンダーアプリケーションの場合，予定のタイトルと日付を保持するサーバサイドの schedule データテーブルはソースコード 1 のように記述する．

ソースコード 1 データモデルの記述例

```
<database type="server" name="schedule">
 <schema>
    <property name="title" type="string"/>
    <property name="date" type="date"/>
 </schema>
</database>
```

また，クライアントとサーバのデータの同期はクライアント側のデータテーブル以下の observe 要素で指定を行う．observe 要素で関連付けられるサーバ側のデータテーブル名を指定する. また，サーバに保存されているデータのすべてをクライアント側に同期することは，パフォーマンス，セキュリティ面で問題がある．そこで，同期を行うデータを制限する．link 要素でカラム，condition 要素でレコードに対して制限を行う.link 要素は from 属性で指定した同期元のデータベースのカラムと to 属性で指定した同期先のデータベースのカラムを関連付ける．リンク元とリンク先のカラムのデータ型は同一である必要がある．condition 要素では，指定した条件を満たすレコードのみを同期させる．例えば，カレンダーアプリケーションの場合，サーバにはすべての年月の予定が保存されているが，クライアントは，ある月についての予定を保持することとする．したがって，記述例はソースコード 2 のようになる．2010 年 8 月の title と date をサーバ側のデータベースから取得，同期するようにしている．condition 要素については，イベントの際に呼び出されるアクション要素から動的に変更することが可能である．詳しくは 3.4.2 で述べる．また，アプリケーションによっては，パスワードなどの個人情報は，クライアントには保存しないようにカラムに制限をかけたり，ユーザタイプやカテゴリなどでレコードに制限をかける場合も考えられる．

ソースコード 2 クライアントとサーバの同期の記述例

```
<database type="client" name="schedule_c">
 <schema>
    <property name="title" type="string"/>
    <property name="date" type="date"/>
 </schema>
 <observe target="schdule">
  <link from="title" to="title"/>
  <link from="date" to="date"/>
  <condition id="schedule_con">
    <and>
     <gt property="date">
        <date>2010/8/1</date>
    </gt>
     <lt property="date">
        <date>2010/9/1</date>
     </lt>
    </and>
  </condition>
 </observe>
</database>
```

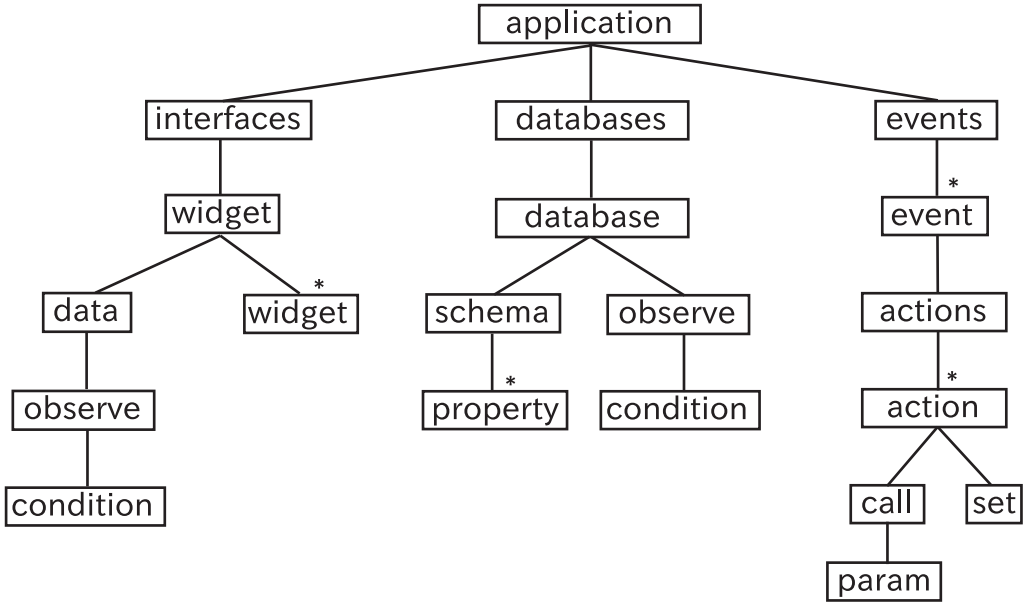

図 2　AJML の木構造

3.3　ユーザインターフェースモデル

GUI 部品である widget 要素を用いて UI を記述する．widget 要素の例としては，label, button, textbox, table 要素などの基本的な要素や，widget 要素を配置するためのページを表す panel 要素などがある．

アプリケーション設計者は，widget 要素を組み合わせて UI を記述する．それぞれの要素は，width, height などの属性をもつ．また，widget 要素によっては，保持するデータを表す data 要素や，さらに widget 要素などを子要素としてもつ．AJML からそれらを指定してユーザインターフェースを設計する．

また，widget 要素は，属性以外にイベント要素によって指定されるイベントをもつ．例えば，button 要素の場合は onClick イベント，panel 要素の場合は，onLoad イベントをもつ．また，widget 要素によっては，独自の振る舞いやメソッドを持ち，イベントの際呼び出される．具体的なイベントの記述は，3.4 で示す．

また，本研究で対象としているアプリケーションは，リアルタイムにデータが更新される．したがって、更新されたデータベースのデータをリアルタイムに UI に反映させる必要がある．AJML では，データベースと UI 要素をリンクさせ，データベースへの変更を UI にリアルタイムに反映する．クライアントとサーバのデータベースの同期と同様に data 要素以下の observe 要素を用いて指定を行う．カレンダーアプリケーションにおける UI の記述の一部をソースコード 3 に示す．ある月の 1 日の予定を表示する table(id：shedule_day_1) は，クライアントのデータベースの 2010 年 8 月 1 日にリンクしている．

ソースコード 3　ユーザインタフェースの記述例

```
<panel id="schedule_list_panel" width="500px
    " height="500px">
  <label id="title_label" content="title" />
  <textbox id="title_textbox" />
  <label id="date_label" content="date" />
  <textbox id="date_textbox" />
  <button id="submit" content="submit" top
      ="50px" left="100px"/>
  <label id="day_1_label" content="day_1"
      top="200px" left="100px"/>
  <table id="schdule_day_1"/>
   <data>
    <observe id="day_1_store" target="
        schedule_c">
```

```
      <link from="title" to="title"/>
      <condition>
        <eq property="date">
          <date>2010/8/1</date>
        </eq>
      </condition>
    </observe>
  </data>
 </table>
   ......
</panel>
```

3.4 イベントモデル

3.4.1 トリガ

イベント要素は，ユーザからのイベント（マウスのクリックやキーボードの入力）やサーバからのイベント（データの変更の通知など）と，それに対応して行われる処理を紐付ける. 例えば，ボタンのクリックによって発生するイベントは，ソースコード 4 ように記述する.

ソースコード 4　イベントの記述例

```
<event id="submit_message" element="submit"
       type="onClick">
    <actions>
       .....
    </actions>
</event>
```

どのようなイベントが定義可能は，それぞれの UI 要素ごとに異なる．ボタンであれば，onClick イベントが定義可能で，テキストボックスでは，onFocus や，onBlur, onKeypress などが定義可能である.

サーバやクライアントのデータに対しては，変更が起こったタイミング発生するイベント onChange, onDelete,onCreate, onUpdate が定義可能である.

3.4.2 アクション

イベントが発生したときにどのような手続きを行うかを action 要素で定義を行う．行われる手続きは call 要素を用いて関数呼び出しとして記述する．引数を param 要素で指定する．システムでネイティブにサポートされている関数は，UI 要素の各属性に変更を加える関数と，データベースに対する CRUD(create, read, update, delete) の関数，同期の情報を変更する observe 関数などである．現在は分岐や繰り返し，計算などを含む複雑な手続きについてはモデル化は行っていない．より複雑な処理を行うためには，アプリケーション設計者が外部 JavaScript ファイルを用意し，関数を定義する必要がある．カレンダーアプリケーションにおける予定の書き込み処理は，ソースコード 5 のように記述する.

ソースコード 5　アクションの記述例

```
<action>
  <call name="insert">
     <param name="_tablename">
      <string>schedule</string>
     <param name="title">
         <value element="title_textbox"
              property="content"/>
     </param>
     <param name="date">
         <value element="date_textbox"
              property="content"/>
     </param>
  </call>
</action>
```

value 要素は他の要素の属性の参照する．title_textbox, date_textbox は，それぞれタイトルと日付の入力用のテキストボックスである．content 属性は，テキストボックスに入力された値を表す．したがって，この例では，insert というデータベースに書き込み操作を行う関数に，table 名とテキストボックスに入力された予定のデータを引数として書き込み操作を行っている.

4 生成システム

ここでは，前節で述べた XML 記述から Web アプリケーションを自動生成するシステムとその実装のプロトタイプについて述べる.

提案する自動生成システムを JAVA 言語を用いてコマンドラインのツールとして実装した．生成後のアプリケーションのプラットフォームとしては，クライアントは HTML5 と JavaScript，サーバサイドは Java サーブレットコンテナを含む Web サーバと JDBC をサポートする RDBMS を想定している.

XML 記述を入力として，HTML, JavaScript, Java

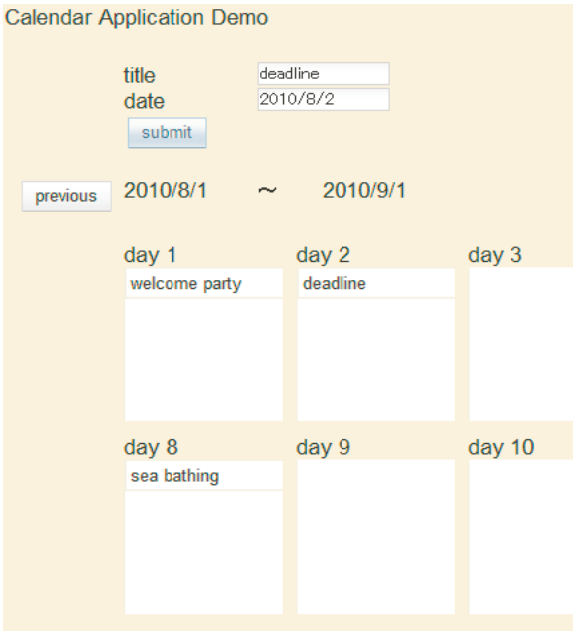

図 3　生成されたカレンダーアプリケーションの一部

のソースコード，各種設定ファイルをもしくはそれらをまとめた war ファイルを生成する．作成された war ファイルは直接サーブレットコンテナをもつ Web サーバにデプロイ可能である．また，本システムに組み込みのサーバーを用い，XML 記述から直接 Web アプリケーションとして実行することが可能である．

また生成例として，カレンダーアプリケーションを XML 記述から自動生成を行い，提案の手法の実現可能性と有効性を確認した．生成したカレンダーアプリケーションの一部を図 3 に示す． 以下，実装上のポイントとなった点を述べる．

クライアントのデータベース

クライアント側のデータベースとしては，HTML5 で提案されている Web Database API を用いた．

サーバからクライアントへのデータの更新

サーバからクライアントの通信方式は Comet 方式を用いた．サーバでは，セッションを用いて各クライアントの同期のためのデータ (observe 要素の情報など) を保持し，サーバに変更があった段階で，その情報を元にクライアントへ変更を伝搬させるよう実装を行った．しかしながら，サーバーの負荷に応じてポーリング方式，ピギーバック方式などから選択可能にする必要がある．

データの競合について

オフライン時には，クライアントはローカルに変更を行う．その後，オンラインになった場合には自動的にローカルの変更がサーバに送信され反映される．オフライン時変更があった場合，クライアントは変更の履歴を保持しておく．そしてオンラインになったときにその履歴情報をサーバへ送信する．しかしながら，複数のクライアント間で同じデータに対して変更が行った場合，変更の競合が発生する。どちらの変更を反映するのかが問題となるが，アプローチとしてはバージョン管理システムのようにユーザにどちらかを選択させるアプローチと，システムがある法則によって自動的にどちらかを決定するかのアプローチが考えられる．現在の実装ではシステムが自動的に最新の変更を優先するというアプローチをとった．どちらのアプローチをとるのかをアプリケーション設計者が XML 記述から指定できるようにすることが必要であると考えられる．

5　比較

他の自動生成ツール [2] [3] [5] [6] と提案手法との比較を行う．主にモデルの表現力について比較を行う．まず，データベース，およびデータの同期に関して比較を行う．[3] [6] では，クライアント側の永続化データは想定されていない．また，リアルタイムにデータを更新することも想定されていない．[2] [5] では，クライアントの永続化データの定義，サーバの変化を検知を記述することはできる．しかしながら，クライアントのデータとサーバ側のデータの同期を直接記述することはできない．したがって，クライアントからサーバへのデータベース変更の反映，サーバからクライアントへのデータベース変更の反映，データベースから UI への反映は，すべて明示的に行う必要がある．また，オフラインでの処理は考慮されておらず記述することができない．

次に，UI，イベント定義に関して比較を行う．従来の手法は，サーバサイド主体のモデルに拡張を加えて RIA の自動生成を可能にしている．したがって [2] [6] では，UI については，ページに表示されるハ

表 1　モデルの表現力についての比較

	提案手法	[2]	[3]	[5]	[6]
データベースへのアクセス	○	○	○	○	○
クライアントの DB	○	○	×	△	×
サーバプッシュ	○	○	○	△	×
DB の同期，UI への反映	○	△	△	△	×
オフラインでの動作	○	×	×	×	×
UI のデザインの自由度	○	×	○	○	×
画面の部分更新，ページング	○	△	○	○	△
高レベルなイベント定義	△	×	△	△	×
ビジネスロジック	×	○	○	△	△

イパーリンクやデータベースしか記述しない．さらに処理を呼び出すタイミングも HTML のフォームの送信時に制限されている．したがって，画面の部分更新などは記述することはできるが，画面のデザイン，イベント定義は制限されている．

[3] [5] や提案手法では，GUI 部品を用いた画面デザインを可能にしていて，イベント定義も画面部品に応じて行うことができ記述力は高いと言える．しかしながら，アプリケーションの見た目，イベントの定義は，GUI 部品，widget 要素に依存するので，よりインタラクティブな UI を自動生成するには，高機能な widget を多数提供すること，widget を追加，拡張できるようにすることが重要である．

ビジネスロジックについて，2 節で述べた通り本提案ではサーバとクライアントの同期に重点を置いたためサーバサイドの複雑な処理は現在記述できない．[2] [3] などで提案されているようなサーバサイドの処理の自動生成手法と組みあわせることが今後の課題といえる．以上のモデル記述力の特徴についてまとめたものが表 1 である．

6　関連研究

ここでは，自動生成ツール以外の Web アプリケーションの開発支援手法について，提案手法と比較を行う。Web アプリケーションの開発支援として，現在さまざま技術が提案されている．jQuery や Dojo toolkit, Prototype JS に代表される JavaScript フレームワークは，クライアントサイドのプログラミングを支援する．Ruby on Rails, Struts などの Web アプリケーションフレームワークは，サーバーサイドやデータベースまわりのプログラミングを支援する．しかしながら，これらはクライアントサイド，サーバサイドどちらか一方のプログラミングを支援するが両方が複雑に連携するアプリケーションを支援するのには十分でない．本提案では，クライアントサイド，サーバサイド両方のコード生成を行い開発を支援する．

また，[1] [4] では，より柔軟な UI を記述するための UI のモデルが提案されている. また， [11] [12] など，アプリケーションの UI を記述する XML 言語も存在する．それらでは，GUI 部品を配置して柔軟な UI 記述する．また，GUI 部品に対するイベントにコールバック関数を登録することができ，イベントドリブンな操作性の高い UI を記述することができる．しかしながらこれらのモデルは，UI を定義するものであり，サーバサイドとのデータのやり取りを含めたアプリケーションの自動生成の手法については提案されていない．提案手法では，サーバサイドも含めたモデルを定義することで，ユーザからの入力イベントだけでなく，サーバからのデータの更新のタイミングでイベントを定義することが可能となった．これによってよりインタラクティブな UI をもったアプリケーションを記述可能である．

また，Google Web Toolkit [9] は，Java 言語で書

かれたプログラムを HTML,JavaScript にクロスコンパイルを行い，Ajax アプリケーションを単一の言語で記述可能にする．Java 言語を用いて開発が行うため記述力は高いが，作成のコストは高い．提案手法では，XML から宣言的に記述できるため，記述力は劣るが作成のコストは低い．

7 まとめと今後の課題

本論文では，従来の手法では扱えなかった特徴をもつ Ajax アプリケーションを自動生成するため XML 記述を提案し，記述から自動生成するシステムを提案した．今後の課題として，XML 記述の作成を支援するツールの作成や widget 要素の充実，ユーザが widget 要素を追加できるような機構の提案， より複雑な手続きの記述方法の提案などが今後の課題である．

参 考 文 献

[1] Abrams, M. and Phanouriou, C. and Batongbacal A.L. and Williams S.M. and Shuster , J.E.: UIML: an applicance-independent XML user interface language, Conputer Networks. The International Journal of Computer and Telecommunicaitons Networking 31, 1695-1708 (1999)

[2] Bozzon, A. and Comai, S. and Fraternali, P. and Toffetti Carughi, G.: Conceptual Modeling and Code Generation for Rich Internet Applications. In: International Conference on Web Engineering, pp. 353.360. Springer, Heidelberg (2006)

[3] Gharavi, V. and Mesbah, A. and Deursen, V. A.:Modelling and Generating Ajax Applications: A Model-Driven Approach, Proceedings of the 7th International Workshop on Web-Oriented Software Technologies, 2008

[4] Linaje, M. and Preciado, J.C., and Sanchez-Figueroa, F.: A Method for Model Based Design of Rich Internet Application Interactive User Interfaces, International Conference on Web Engineering, Springer, Como (Italy), 2007, vol. 4607

[5] Preciado, J.C. and Linaje, M. and Comai, S. and Sanchez-Rigueroa, F.: Designing Rich Internet Applications with Web Enginnering Methodologies, Proceedings of the International Symposium on Web Service Evolution (WSE). IEEE Computer Society, 23.30.

[6] 大嶽智裕, 野呂智哉, 徳田雄洋: ダイアグラムに基づくリッチインターネットアプリケーションの生成法, 日本ソフトウェア科学会 第 24 回大会, pp.2B-3. 2007 Sep

[7] Adobe Systems Incorporated: Adobe Integrated Runtime (AIR), http://labs.adobe.com/technologies/air

[8] Google Inc: Gears, http://gears.google.com/

[9] Google Inc: Google Web Toolkit, http://code.google.com/webtoolkit/

[10] Microsoft Corporation: Silverlight, http://silverlight.net/

[11] Microsoft Corporation: XAML(Extensible Application Markup Language) http://msdn.microsoft.com/en-us/library/ms752059.aspx

[12] Mozilla Foundation: XUL(XML User Interface Language), https://developer.mozilla.org/En/XUL

[13] Oracle Corporation: JavaFX. http://javafx.com/

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

[15] W3C: HTML5 Working Draft, http://www.w3.org/TR/html5/