Vol.2011-DPS-149 No.12
2011/11/25

中間表現とフレームワークを用いたWebアプリケーションのメンテナンス法の提案と評価

早川 智一[†1]　長谷川 慎哉[†1]
吉賀 祥太[†1]　疋田 輝雄[†1]

RIA（Rich Internet Application）の普及によって移植性の問題が顕在化してきている：RIA 技術間には互換性がなく，開発に使用した RIA 技術が廃れたり仕様が変更されると，他の RIA 技術による移植が必要となり，大規模なシステムでは看過できない問題となる．我々はこの問題を解決するために，XML ベースの中間表現と Java ベースのフレームワークを用いる方法を提案する．我々は，この方法を適用したプロトタイプを評価のために開発し，Web-IR と名付けた．Web-IR は，実装例として Ajax・Flex・HTML5・JavaFX・OpenLaszlo に対応する．我々は，Web-IR が既存の RIA を外観を維持しながら他の RIA 技術に変換できることを評価を通じて示し，我々の方法が前述の問題解決に有用であるという結論を得た．

Design and Evaluation of Intermediate Representation and Framework for Maintaining Web Applications

Tomokazu Hayakawa ,[†1] Shinya Hasegawa ,[†1]
Shota Yoshika [†1] and Teruo Hikita[†1]

A problem has arisen by the spread of Rich Internet Applications (RIAs): one needs to maintain RIAs by using other RIA technologies when a RIA technology used for existing RIAs become obsolete, since they have few compatibilities to each other. To solve the problem, we have proposed a method which employs an XML-based intermediate representation and a Java-based framework. We also constructed a prototype named Web-IR, which supports Ajax, Flex, HTML5, JavaFX and OpenLaszlo as examples, to evaluate our method. Our evaluations show that Web-IR can transform existing RIAs into others keeping same appearances, and we concude that our method can solve the problem.

1. はじめに

インターネットの普及とともに RIA（Rich Internet Application）が利用されるようになり，移植性の問題が顕在化してきている．これは，RIA 技術間に互換性がないことに起因する[13]．ここで，具体的な RIA 技術としては，Ajax・Flex・HTML5・JavaFX[14]・OpenLaszlo[10]・Silverlight 等が挙げられる．

この移植性の欠如は，RIA を業務で使用する場合には問題となる．何故ならば，昨今の Web アプリケーションは RIA として実装されていることが多く，同一システム内で異なる RIA 技術が使用されていることも珍しくないからである[⋆1]．そのため，システムの開発に使用した RIA 技術が廃れたり仕様が変更された場合には，他の RIA 技術を用いて既存の RIA を移植する必要が生じる．しかし，業務で使用するアプリケーション（以下，業務アプリケーション）は，ボタンやラベルなどの多くの UI（User Interface）部品（以下，ウィジェット）を含む多数の画面から構成されるため，移植のコストが看過できないものとなる．

我々の問題提起は，机上の空論と考えられがちだがそうではない．以下，具体例をふたつ挙げる．ひとつは Visual Basic 6（以下，VB6）の問題である．VB6 は，RIA が主流となるまで業務アプリケーションの開発に用いられてきたが，製品サポートが終了したために，既存のアプリケーションを移植する方法が求められている．もうひとつは JavaFX の問題である．JavaFX は，最新のリリース（2.0）で記述言語が変更され実行環境の互換性が失われた．これにより，既存のアプリケーションを移植する必要性が生じている．

我々は，前述の問題を解決するために，中間表現（以下，IR：Intermediate Representation）とフレームワークを用いる方法を提案する（3 節）．核となるアイデアは以下の 2 点である．

(1)　入力の RIA を XML ベースの IR に変換し，IR から各 RIA 技術に変換する．
(2)　RIA・IR 間の変換支援に，拡張可能な Java フレームワークを提供する．

この方法は，単なるトランスレータに過ぎないと考えられがちだがそうではない．何故ならば，RIA 技術のセマンティクスは一意に変換することが難しい場合があり，利用される環境（コンテキスト）ごとに解釈の変更が必要となるためである（3.1 節）．

†1 明治大学理工学研究科
School of Science and Technology, Meiji University.

⋆1 例えば，YouTube（http://www.youtube.com/）では HTML5 と Flash が併用されている．この場合，どちらかの RIA 技術が廃れただけでサービスレベルが低下することになりリスクが高まる．

我々は，前述のアイデアを適用したプロトタイプを評価のために実装し，*Web-IR* と名付けた．Web-IR は，入力例として Ajax と Flex に，出力例として HTML5 と JavaFX と OpenLaszlo に対応する（3.2 節）．

Web-IR の評価は，我々の提案する方法が前述の問題解決において有効であることを示した：Web-IR のウィジェット情報の変換率は Ajax で 80%・Flex で 44%であり，スタイル情報の変換率は Ajax で 100%・Flex で 33%であった（6.1.2 節・6.1.3 節）．

本論文の構成は以下の通りである．2 節では関連研究と関連技術を紹介する．3 節では Web-IR について概説する．4 節と 5 節では，それぞれ IR とフレームワークについて説明する．6 節では評価結果を報告する．7 節では結論を述べる．なお，設計や実装の詳細については文献 7)–9) を参照されたい．

2. 関連研究・関連技術

本節では，関連する研究・技術について紹介し，我々の研究との類似性・差異について述べる．また，市場にリリースされているサービスについても言及する．

2.1 フレームワーク

松塚[12] は，業務アプリケーション用の Ajax フレームワークを提案し，業務アプリケーションは規模（画面数・部品数・コード量）が大きくなる傾向にあるため，それらを効率よく記述・管理する機構が必要であると述べている．また，ウィジェットには（例えそれが単なるテキストフィールドであっても）個々にカスタマイズ性が求められるとも述べている．これらは，我々の IR とフレームワークの提案理由に合致する．すなわち我々は，移植性・汎用性を保ちながら RIA を表現するために IR が必要であり，コンテキストに応じた変換のために柔軟に動作を変更できるフレームワークが必要であると考える．

松塚のフレームワークと我々のフレームワークは，業務アプリケーションを効率的に記述する点において類似性があるが，対象とする実装技術が異なる：松塚は Ajax を対象としているが，我々は特定の RIA 技術に依存しないことを前提としている．

2.2 商用サービス

Adobe Systems は，Wallaby[1] と呼ばれる Adobe Flash Professional（FLA）ファイルを HTML5 に変換する処理系をリリースしており，これにより Flash 実行環境を備えないデバイスへの移植が容易になると述べている．また Google は，Swiffy[5] と呼ばれる Flash（SWF）ファイルを HTML5 に変換する処理系をリリースしており，これにより iPhone や iPad などへの Flash コンテンツの移植が容易になるとしている．これらの存在は，RIA 技術間の移植手段が必要であるという我々の主張と合致する．

これらの処理系と我々の Web-IR は，RIA を変換する点において類似性がある．一方で，これらが RIA 技術間で直接変換を行うのに対し，Web-IR は IR を経由して複数の RIA 技術へ間接変換する点において異なる．

2.3 そ の 他

Yu・Kontogiannis・Lau[15] は，Web アプリケーションの移植性の問題を解決するために，Web アプリケーションを MVC アプリケーションに変換するための Java ベースのフレームワークを提唱している．Linaje・Preciado・Sánchez-Figueroa[11] は，Web アプリケーションの UI 情報を適合させるための RUX-Model というモデルを提案している．Aversano・Canfora・Cimitile・Lucia[2] は，COBOL で記述されたシステムを Web アプリケーションに変換する手法について報告している．Zhang と Chen[17] は，Web アプリケーションの UI 情報を拡張可能にする XML の使用法について提唱している．Chung と Lee[3] は，Java と XML 技術を用いた Web アプリケーションの可視化モデルについて提案している．

これらの研究と我々の研究は，Web アプリケーションを汎用化し移植性を高めようとする点において類似性があるが，Web-IR は複数の RIA 技術間の変換を前提に特定のベンダや技術に依存しないように設計されている点において異なる．

3. Web-IR

3.1 概 要

Web-IR の目的は，複数の RIA の移植にかかるコストを低減することにある．

この目的を達成するためにただちに思いつくのは，RIA 技術間のトランスレータを実装することであるが，以下の理由から現実的ではない．すなわち，n 種類の RIA 技術に対して $n \times (n-1)$ 通りの処理系が必要になり手間がかかるためと，RIA 技術によっては変換時のセマンティクスが一意に定まらず，自動変換が困難であるためである[⋆1]．

そこで我々は，IR を主要な RIA 技術が共通して備える機能のみをもつように設計し，RIA・IR 間の変換を支援するフレームワークを提供する方式を採用した（**図 1**）．IR を前述のように設計した理由は次の通りである．すなわち，業務アプリケーションにとって RIA 技術は実装手段でしかなく，他の RIA 技術への移行が容易であることの方が重要だからで

⋆1 例えば，Ajax アプリケーションでは`<div>`や`<span>`が用いられるが，これらは汎用タグであり特別な意味をもたない．そのため，これらを他の RIA 技術に変換する際には，コンテキストに応じた変換が必要になる．

ある．

我々の方式では以下のメリットがある．

(1) n 種類の RIA 技術に対して，処理系の数が $2n$ 通りで済む．
(2) コンテキストに応じた変換が可能になる．
(3) 新しい RIA 技術 X が登場した際に，IR から X への処理系を追加するだけで，既に入力に対応している既存の RIA を（IR を経由して）X 形式に変換できる．

デメリットとしては，IR の導入により，直接変換を行うトランスレータに変換率で劣る点と，ウィジェットなどの表現能力が制限される点が挙げられる．これらについては，IR とフレームワークを拡張可能とすることで対処する（4.1 節・5.1 節）．

IR には XML を使用する．これは以下の理由による．すなわち，ファイル形式には永続性の観点から特定のベンダや技術に依存せず RIA 技術と親和性が高いものが望まれる．この候補として XML と JSON が挙げられるが，ソースの可読性や 5.2 節の変換アルゴリズムとの相性を考慮し XML を採った．

フレームワークの記述には Java を用いる．Java を選択した理由は，Web 開発において実績のあるオブジェクト指向言語であることや，フリーの実装（OpenJDK）もあり永続性が高いと考えられること，様々な OS で動作することなどが挙げられる．

まとめると，Web-IR の特徴は以下の通りとなる．

(1) 各 RIA 技術に共通する機能をもつ XML ベースの拡張可能な IR をもつ．
(2) RIA・IR 間の変換を支援する拡張可能な Java ベースのフレームワークをもつ．
(3) 複数の RIA 技術に対して，コンテキストに応じた変換処理を提供する．

3.2 実　　装

Web-IR は，入力例として Ajax と Flex 4.5 に，出力例として HTML5 と OpenLaszlo 4.9 と JavaFX 2.0（β 版）に対応する．Ajax と Flex を選択した理由は，既存アプリケーションが多いと考えられるためであり[*1]，HTML5 と OpenLaszlo を選択した理由は，仕様が公開されており特定のベンダに依存しないためである（JavaFX を選択した理由は 6.1.1 節）．

3.3 使　用　例

Web-IR の使用時には，処理系を駆動させるコードを Java で記述する．**図 5** は，Ajax から IR へ変換を行う場合の Web-IR の使用例である．ここで特筆すべきことは，各インスタンスの生成方法である．Web-IR は，クラス名への依存を避けるために，`new` 演算子を用いずに文字列を使用してインスタンスを生成する（5.1 節）．

図 5 によって自動変換された JavaFX のページを**図 4** に示す．変換元の Web ページを異なるブラウザで表示した結果を**図 2** と**図 3** に示す．これらの図より，Web-IR は僅かな UI の差異のみで RIA を異なる RIA 技術に変換できることがわかる．

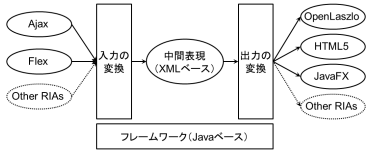

図 1 Web-IR の概要
Fig. 1 Overview of Web-IR.

図 2 Internet Explorer 9 による Web ページの表示（変換前）
Fig. 2 Web Page Rendered by Internet Explorer 9 (before Transformation).

図 3 Firefox 6 による Web ページの表示（変換前）
Fig. 3 Web Page Rendered by Firefox 6 (before Transformation).

図 4 Web-IR で変換された JavaFX ページ（変換後）
Fig. 4 Transformed JavaFX Page by Web-IR (after Transformation).

4. 中 間 表 現

4.1 概　　要

中間表現（IR : Intermediate Representation）は XML をベースとして設計されており，ルート要素に`<application>`をもつ．IR の内部は 4 種類（メタ情報・ウィジェット情報・スタイル情報・振舞い情報）に分割されており（**図 6**），それぞれ`<meta>`・`<widget>`・`<style>`・`<behavior>`をルート要素にもつ．また，必要に応じて任意の要素を追加することで拡張す

⋆1 Google Insights for Search（http://www.google.com/insights/search/）のクエリ数調査による．

```
/* Ajaxを読み込むためのApplicationReaderインタフェースのインスタンスを取得する */
ApplicationReader src = ApplicationReaders.newInstance("Ajax");
/* 引数として渡されたURIからAjaxアプリケーションを読み込む */
Application app1 = src.read("file://...(URI)");
/* 入力（Ajax）を出力（IR）に変換するTranslatorインタフェースのインスタンスを取得する */
Translator translator = Translators.newInstance("Ajax", "IR");
/* 入力（Ajax）を出力（IR）に変換する */
Application app2 = translator.translate(app1);
/* 出力（IR）を書き込むためのApplicationWriterインタフェースのインスタンスを取得する */
ApplicationWriter dst = ApplicationWriters.newInstance("IR");
/* 引数として渡されたURIにIRを書き込む */
dst.write(app2, "file://...(URI)");
```

図 5 Web-IR の使用例（Ajax から中間表現（IR）への変換）
Fig. 5 Usage of Web-IR (Transformation from Ajax to Intermediate Representation (IR)).

```
<?xml version="1.0" encoding="UTF-8"?>
<application>
 <meta>
   <!-- メタ情報 -->
 </meta>
 <widget>
   <!-- ウィジェット情報 -->
 </widget>
 <style>
   <!-- スタイル情報 -->
 </style>
 <behavior>
   <!-- 振舞い情報 -->
 </behavior>
</application>
```

図 6 中間表現の概観
Fig. 6 Skeleton of Intermediate Representation.

ることが可能である．この場合，IR の移植性が高いという特徴は失われるが，特定の条件下においてはこれは有用である[⋆1]．

我々が IR を分割した理由は，第一に既存の RIA 技術のほとんどが各情報を分割して記述するためであり[⋆2]，第二に各情報を分割保持することで情報の部分利用が可能になるためである．後者は以下の理由から特に重要である．すなわち，業務アプリケーションにおいては画面数・部品数が多いため，UI 情報やスタイル情報の変換だけでも大幅な工数削減につながる．また，UI 情報やスタイル情報の変換は，振舞い情報の変換と比較して実装が容易であるため，少ない投資で大きな見返りを得ることが期待できる．

4.2 構　　成

4.2.1 メ タ 情 報

メタ情報部は，RIA のタイトルや文字コードなどのメタ情報を格納する．使用可能な要素を**表 1** に示す．

4.2.2 ウィジェット情報

ウィジェット情報部は，RIA のボタンやテキストボックスなどのウィジェットに関する情報を格納する．主要なウィジェットを**表 2** に示す．

4.2.3 スタイル情報

スタイル情報部は，RIA のフォントや文字サイズなどのスタイル情報を格納し，XML 化した CSS で記述する．CSS を選択した理由は，主要な RIA 技術でスタイルの指定に用いられているためである．**図 7** と**図 8** に，CSS と IR におけるスタイル情報の対応を示す．図より，`<rule>`・`<selector>`・`<property>`要素が CSS のルール・セレクタ・プロパティに対応することがわかる．

表 1 中間表現のメタ要素
Table 1 Meta Elements of Intermediate Representation.

要素名	説明
title	RIA のタイトル
charset	RIA の文字コード

```
text {
  font-size: 10pt;
}
```

図 7 CSS によるスタイルの表現
Fig. 7 Style Representation in CSS.

```
<rule>
 <selector>text</selector>
 <property>
  <name>font-size</name>
  <value>10pt</value>
 </property>
</rule>
```

図 8 中間表現による図 7 のスタイルの表現
Fig. 8 Style Representation of Fig. 7 in Intermediate Representation.

表 2 中間表現のウィジェット要素
Table 2 Widget Elements of Intermediate Representation.

要素名	説明
anchor	アンカー（HTML の`<a>`に相当）
button	ボタン
checkbox	チェックボックス
hbox	子要素を水平に並べる不可視の矩形領域
hr	水平線
image	画像の表示
list	子要素のリスト表示
menu	メニュー
radiobutton	ラジオボタン
scrollbar	スクロールバー
select	複数の子要素からの選択
slider	スライダー
space	空白領域
table	テーブル
text	読み取り専用テキスト
textarea	テキスト入力領域（複数行）
textbox	テキスト入力領域（単一行）
tooltip	カーソルが乗ると文字列を表示
vbox	子要素を垂直に並べる不可視の矩形領域

4.2.4 振舞い情報

振舞い情報部には，ボタンのクリックなどのイベント情報を ECMAScript で記述する．ECMAScript を選択した理由は，主要な RIA 技術において振舞いの記述に使用されているためと，XML 化すると記述量が増え可読性が低下するためである．

4.3 変　換　例

Web-IR が変換したページを**図 9**・**図 10**・**図 11** に示す．これらは，同じ IR（**図 13**）から変換されており，変換後の RIA 技術はそれぞれ OpenLaszlo・HTML5・JavaFX である．各 RIA 技術は異なる UI ポリシーをもつため，画面のレイアウトに差異が生じる．これを補正するためには，フレームワークの機能を用いて変換動作を調整する必要がある．

⋆1 例えば，ある組織内で使用されている RIA 技術が限定的である場合には，IR をそれらの共通仕様を逸脱しない範囲まで拡張することで，組織内での移植性を担保したまま表現能力を向上させることが可能になる．
⋆2 Ajax アプリケーションを例に挙げると，ウィジェット情報・スタイル情報・振舞い情報に，HTML・CSS・JavaScript がそれぞれ対応する．

Vol.2011-DPS-149 No.12
2011/11/25

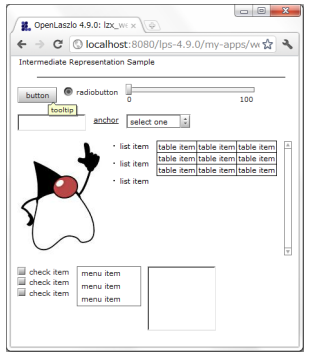

図 9 図 13 の中間表現から変換されたページ（OpenLaszlo）
Fig. 9 Transformed Page from Intermediate Representation Shown in Fig. 13 (OpenLaszlo).

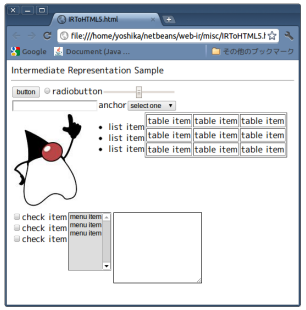

図 10 図 13 の中間表現から変換されたページ（HTML5）
Fig. 10 Transformed Page from Intermediate Representation Shown in Fig. 13 (HTML5).

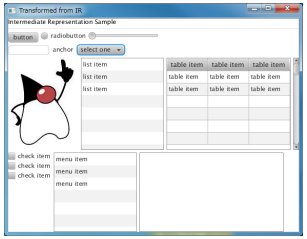

図 11 図 13 の中間表現から変換されたページ（JavaFX）
Fig. 11 Transformed Page from Intermediate Representation Shown in Fig. 13 (JavaFX).

5. フレームワーク

5.1 概　　要

Web-IR のフレームワークは，入力の RIA から IR および IR から出力の RIA への変換を支援する．変換過程ではすべての情報を DOM に似た木構造で表現する．これは，IR や主要な RIA 技術がタグによる宣言的記法を採用していることに依る．

我々は，フレームワークを SPI（Service Provider Interface）パターンを用いて設計した．SPI により，クラス名ではなく文字列（Web-IR では RIA 技術名）を用いてインスタンスを生成することが可能になる．従って，処理系の実装の変更や拡張が可能になり，変換時の振る舞いの変更や新しい RIA 技術への対応が容易になる．

フレームワークは，135 個のクラスと 19,144 行の Java コードからなる．**図 12** は，フレームワークの概要クラス図である．重要なクラスとインタフェースを**表 3** に示す．これらは，SPI の骨格となるインタフェース群であり，フレームワークを構成するための抽象レイヤを提供する．これにより，利用者はクラスではなくインタフェースに対して実装を行うことが可能になり，フレームワークの理解が容易になる．

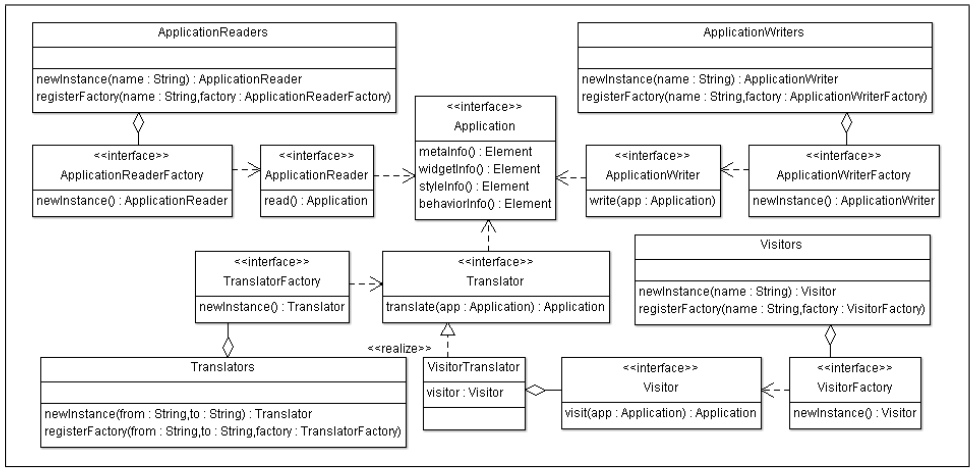

図 12 Web-IR のフレームワークの概要クラス図
Fig. 12 Class Diagram of Framework of Web-IR.

表 3 Web-IR のフレームワークにおける重要なクラス・インタフェース
Table 3 Important Classes and Interfaces of Framework in Web-IR.

クラス・インタフェース名	説明
Application	任意のアプリケーションを表す
ApplicationReader	入力を Application インスタンスとして読み込むリーダを表す
ApplicationWriter	Application インスタンスを出力として書きこむリーダを表す
Translator	トランスレータを表す
Visitor	Visitor パターンで使用される Visitor を表す
VisitorTranslator	Visitor パターンを用いて Translator インタフェースを実装したクラス

5.2 変 換 方 法

フレームワークは，変換に Visitor パターン[4] を用いる．これは，IR や主要な RIA 技術の XML による記法と相性が良いためである．フレームワークの変換アルゴリズムの概観を**図 14** に示す．図中の `visit` メソッドが示すように，引数として与えられる要素を，対応する要素に再帰的に変換する．ただし，振舞い部の変換時には，XML ではなく抽象構文木（AST：Abstract Syntax Tree）が変換対象となる．

なお，振舞い情報を変換する際には以下の要件を備えたエミュレーションライブラリを併用することが望ましい：(1) 出力の RIA 技術で実装されており，(2) 入力の RIA 技術の振

```
<?xml version="1.0" encoding="UTF-8"?>
<application>
 ...
 <widget>
  <vbox>
   <text>
    Intermediate Representation Sample
   </text>
   <hr />
   <hbox>
    <button>
     <text>button</text>
    </button>
    <radiobutton>
     <text>radiobutton</text>
    </radiobutton>
    <slider />
   </hbox>
   <hbox>
    <textbox />
    <anchor><text>anchor</text></anchor>
    <select>
     ...
    </select>
   </hbox>
   <hbox>
    <image src="duke.jpg" />
    <list>
     ...
    </list>
    <table>
     ...
    </table>
    <scrollbar />
   </hbox>
   <hbox>
    <checkbox>
     ...
    </checkbox>
    <menu>
     ...
    </menu>
    <textarea />
   </hbox>
  </vbox>
 </widget>
 ...
</application>
```

図 13 中間表現のサンプル
Fig. 13 Sample Intermediate Representation.

```
/**
 * Web-IRの変換を説明するための擬似コード.
 * この例では，入力と出力をそれぞれAjaxと
 * 中間表現と仮定している.
 * @param e0 変換されるHTMLの要素
 * @return e0と等価な中間表現の要素
 */
Element visit(Element e0){
 switch(e0){
  case "<input type='checkbox'>":
   Element e1 = new Element("<checkbox>");
   while(e0.hasMoreChild()){
    e1.appendChild(visit(e0.nextChild()));
   }
   return e1;
  case "<input type='text'>":
   Element e1 = new Element("<textbox>");
   return e1;
  ...
 }
}
```

図 14 Web-IR の変換アルゴリズム（擬似コード）
Fig. 14 Transformation Algorithm of Web-IR (Pseudo Code).

表 4 Web 上のトラフィック量上位 10 サイト
Table 4 Top 10 Web Sites Ordered by Traffic.

順位	サイト名	URL
1	Google	http://www.google.com/
2	Facebook	http://www.facebook.com/
3	YouTube	http://www.youtube.com/
4	Yahoo!	http://www.yahoo.com/
5	Blogger.com	http://www.blogspot.com/
6	Baidu.com	http://www.baidu.com/
7	Wikipedia	http://www.wikipedia.org/
8	Windows Live	http://www.live.com/
9	Twitter	http://www.twitter.com/
10	QQ.COM	http://www.qq.com/

©2011, Alexa Internet (www.alexa.com)

舞いに関する機能を模倣するもの．これは，各 RIA 技術がほぼ同等の機能（例：非同期通信など）をもつものの，セマンティクスにおいて共通点が乏しく，直接変換が困難であるためである．長谷川の処理系もこの方式を採用している（5.3 節）．

5.3 Web-IR を用いた開発事例

長谷川・早川・疋田[6] は，Ajax を Flash に変換する処理系を開発した．これは，Web-IR のフレームワークを使い，IR の代わりに OpenLaszlo を用いることで，入力の Ajax アプリケーションをほぼ等価な Flash に変換するものである．特徴として，OpenLaszlo の拡張機能を用いて Ajax エミュレーションライブラリ（JavaScript や DOM などの機能を含む）を作成し，変換を容易化した点が挙げられる．この処理系は，9 個のクラスと 1,304 行のコードと 115 個のクラスで構成される再現ライブラリからなる．

吉賀・早川・疋田[16] は，Flex の UI を HTML5 に変換する処理系を開発した．これは，Web-IR のフレームワークを用いており，Flex のソースファイル（MXML）を独自の DOM 木に変換してから HTML5 の DOM 木に変換する点に特徴がある．この処理系は，56 個のクラスと 3,181 行のコードからなる．

我々は，これらの結果から，Web-IR のフレームワークが RIA 技術の変換システムの開発において，必要な能力を有しているものと考える．

6. 評　　価

6.1 変　換　率

6.1.1 準　　備

我々は Web-IR の変換率を算出した．ここでの変換率とは，各 RIA 技術と IR の UI 情報がどの程度相互に変換可能であるかによって定義される．変換率の計測においては 3.2 節で述べた理由から入力には Ajax と Flex を選択し，出力には OpenLaszlo と HTML5 を選択した．さらに，追加評価として JavaFX も加えた．これは，JavaFX が他の RIA 技術と異なる記述形式をもつ（タグによる UI 記述を行わない）ために IR の記述能力およびフレームワークの変換能力の評価に有用と考えたためである[⋆1]．また，出現頻度による重み付けを考慮した変換率を算出するために，Web におけるトラフィック量の上位 10 サイト（**表 4** ）から Ajax の記述言語である HTML のタグと CSS のプロパティの出現頻度を計測した．ここで，重み付きの変換率は以下の理由から重要である．すなわち，タグやプロパティには多くの種類が存在するが，実際にはほとんど使用されないものも含まれるため[⋆2]，単純に数値のみで比較すると見かけの変換率が下がってしまうが，実際には頻出するタグやプロパティへの対応の方が重要であるためである．

計測の結果を**表 5** と**表 6** に示す．注目すべき点として，`<div>`と`<span>`が多く使用されている点が挙げられる．これらは，Ajax アプリケーションにおいて，JavaScript から操作される領域を示すために使用されているためと考える．

⋆1 JavaFX は本論文の評価時点・執筆時点では β 版であったが，その後の正式リリース（2.0）で，タグ（FXML）による GUI の記述がサポートされた．
⋆2 例えば，CSS には音声や印刷に関する多くのプロパティがある．

表 5 表 4 のサイトにおけるタグの出現頻度
Table 5 HTML Tag Occurrences in Top 10 Traffic Sites Shown in Table 4.

順位	タグ名	出現数	出現率（%）
1	a	1,482	27.15
2	div	1,290	23.63
3	span	828	15.17
4	img	376	6.89
5	li	313	5.73
6	br	298	5.46
7	button	146	2.67
8	script	106	1.94
9	p	70	1.28
10	option	67	1.22
...	...	...	...
合計		5,457	100.00

表 6 表 4 のサイトにおけるプロパティの出現頻度
Table 6 CSS Property Occurrences in Top 10 Traffic Sites Shown in Table 4.

順位	プロパティ名	出現数	出現率（%）
1	width	1,235	7.44
2	padding	919	5.53
3	height	902	5.43
4	display	902	5.43
5	color	842	5.07
6	background	692	4.17
7	margin	605	3.64
8	font-size	575	3.46
9	background -position	575	3.46
10	position	559	3.36
...	...	...	...
合計		16,592	100.00

6.1.2 ウィジェット情報

RIA から IR および IR から RIA へのウィジェットの変換率を**表 7** に示す．結果として，Web-IR は Flex を除いて 90%以上のウィジェット情報を変換できた．Flex の変換率が低い理由は，Flex が IR の 2 倍以上のウィジェットをもつためである．ここで，4.1 節で述べた IR の拡張機能を用いると，変換率は 100%となった[⋆1] ．

6.1.3 スタイル情報

表 8 に，RIA から IR および IR から RIA へのスタイルの変換率を示す．結果として，Web-IR が CSS の 50%以上を処理できることがわかる．Flex から IR への変換率が低い理由は，Flex のスタイルにはコンポーネントに依存した特殊なプロパティが多いためである．IR から HTML5 を除く各 RIA 技術への変換率が低い理由は，これらの RIA 技術が CSS のサブセットをスタイルの記述に利用するためである．

6.1.4 振舞い情報

振舞い情報の変換率は 5.2 節で述べた理由から，IR から出力形式の RIA への変換を行う処理系の実装とエミュレーションライブラリの精度に依存する．

⋆1 IR は拡張可能な XML であるため，Flex の MXML 情報を無損失で保持できるため．

表 7 ウィジェット情報の変換率
Table 7 Transformation Ratio for Widget Information.

入力	出力	変換率（%）	変換率[⋆2]（%）
Ajax	IR	80.6	93.8
Flex	IR	44.0[⋆3]	N/A[⋆4]
IR	OpenLaszlo	100.0	100.0
IR	HTML5	90.0	N/A[⋆5]
IR	JavaFX	100.0	100.0

表 8 スタイル情報の変換率
Table 8 Transformation Ratio for Style Information.

入力	出力	変換率（%）	変換率[⋆2]（%）
Ajax	IR	100.0	100.0
Flex	IR	33.3[⋆3]	N/A[⋆4]
IR	OpenLaszlo	42.6	54.8[⋆6]
IR	HTML5	100.0	100.0[⋆6]
IR	JavaFX	62.3	95.8[⋆6]

6.2 開 発 効 率

6.2.1 UI の開発における比較

我々は，Web-IR の使用の有無による RIA の UI 開発の工数比較を試みた．これは，新しい RIA 技術が登場して UI の移植が必要になった際に，人手で移植を行う際の工数を測るためである．被験者には RIA 技術の初学者を用いて，JavaFX と OpenLaszlo を用いた開発を行い発生するコストをそれぞれ学習コストと開発コストに分けて測定した．作業内容は，図 13 のソースから自動生成されたページ（図 11）と同じ UI を作成した．結果として，JavaFX は学習コストに 1.0 時間，開発コストに 3.5 時間を要し，OpenLaszlo は学習コストに 1.0 時間，開発コストに 9.5 時間を要した．Web-IR を用いた場合には，学習コストが 0.5 時間，開発コストが 0.5 時間であった．

これらの結果から，（図 11 のような単純な UI であっても）初学の RIA 技術による UI 開発は工数がかさむことが示唆された．これは，RIA 技術ごとに UI ポリシーが異なる上に，提供される UI コンポーネント数が多いため概要の把握だけでも時間がかかるということが大きい．一方で，Web-IR は既存の RIA を IR を経由して出力の RIA 技術へ自動変換することで，少ない工数で UI の移植を実現できることを示した．

6.2.2 UI 変換システムの開発における比較

我々は，Web-IR のフレームワークの有無による RIA の UI 変換システムの開発工数を

⋆2 表 4 のサイトにおける出現率を重みとして考慮した場合の変換率．
⋆3 Flex の Spark の UI コンポーネントを対象とした．
⋆4 Flex はコンパイルされてバイナリ形式になるため出現率が算出できない．
⋆5 中間表現のウィジェットの出現頻度が得られないため．
⋆6 中間表現のスタイル情報は CSS であるため，表 6 の値を用いた．

表 9　Web-IR の使用の有無による開発工数の比較
Table 9　Comparison of Production Costs with/without Using Web-IR.

入力	出力	Web-IR の使用の有無	工数（時間）	行数
Ajax	OpenLaszlo	×	10.0	1,450
Ajax	OpenLaszlo	○	5.0	960
Flex	HTML5	×	24.0	1,818
Flex	HTML5	○	18.0	1,231
Ajax	JavaFX	×	20.0	3,156
Ajax	JavaFX	○	12.0	1,627

計測した（**表 9**）．これは，RIA の UI の移植が必要となった際に，Web-IR がどれだけ工数を削減できるかを測るためである．開発するシステムは，各 RIA 技術の UI 要素のみを変換する仕様とした．結果として，Web-IR を使用することで，工数と行数がそれぞれ 2/3 以下に削減されることが示された．これは，Web-IR のフレームワークが開発方法論を強制するために，設計と実装・クラスとインタフェースが分離され，効率的な開発を促進するためと考える．

7. おわりに

本論文では，RIA の移植性の問題を解決するために，XML ベースの中間表現と Java ベースのフレームワークを用いる方法を提案した．プロトタイプである Web-IR を用いた評価は，我々の提案する方法が前述の問題解決（特に RIA の UI の移植）において有用であることを示した．一方で，振舞い情報の変換については更なる改良が必要であると考える．これについては別の機会に報告したい．

参考文献

1) Adobe Labs: Wallaby, Adobe Systems Inc. (online), available from ⟨http://labs.adobe.com/technologies/wallaby/⟩ (accessed 2011-09-05).
2) Aversano, L., Canfora, G., Cimitile, A. and Lucia, A.D.: Migrating Legacy Systems to the Web: an Experience Report, *Proc. of CSMR 2001*, CSMR 2001, pp. 148–157 (2001).
3) Chung, S. and Lee, Y.: Modeling Web Applications Using Java And XML Related Technologies, *Proc. of HICSS 2003*, HICSS 2003, pp.322–331 (2003).
4) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional (1995).
5) Google: Swiffy, Google (online), available from ⟨http://www.google.com/doubleclick/studio/swiffy/⟩ (accessed 2011-09-05).
6) 長谷川慎哉，早川智一，疋田輝雄：Ajax アプリケーションの Flash 変換システムの提案と実装，第 73 回全国大会講演論文集（分冊 1），情報処理学会，pp.699–700 (2011).
7) 早川智一，長谷川慎哉，疋田輝雄：中間表現とフレームワークを用いた Web アプリケーション変換，DPSWS2010 論文集，宮崎，情報処理学会，pp.187–192 (2010).
8) Hayakawa, T., Hasegawa, S. and Hikita, T.: Design and Implementation of Intermediate Representation and Framework for Web Applications, *Proc. of CSIE 2011*, Changchun, China, CSIE 2011, pp.137–141 (2011). (to appear in Springer LNEE).
9) Hayakawa, T., Hasegawa, S., Yoshika, S. and Hikita, T.: Maintaining Web Applications by Translating Among Different RIA Technologies, *Proc. of SEA 2011*, Singapore, SEA 2011 (2011). (to appear).
10) Laszlo Systems, Inc.: OpenLaszlo, Laszlo Systems, Inc. (online), available from ⟨http://www.openlaszlo.org/⟩ (accessed 2011-09-05).
11) Linaje, M., Preciado, J.C. and Sánchez-Figueroa, F.: Engineering Rich Internet Application User Interfaces over Legacy Web Models, *IEEE Internet Computing*, Vol.11, No.6, pp.53–59 (2007).
12) 松塚貴英：業務アプリケーションに適用する Ajax フレームワーク，情報処理学会論文誌， Vol.49, No.7, pp.2360–2371 (2008).
13) 三菱総合研究所：OSS デスクトップの普及に資する Web コンテンツ互換性向上に関する調査，情報処理推進機構オープンソフトウェア・センター（オンライン），入手先⟨http://www.ipa.go.jp/software/open/ossc/seika.html⟩（参照 2011-09-05）.
14) Oracle Corporation: JavaFX, Oracle Corporation (online), available from ⟨http://javafx.com/⟩ (accessed 2011-09-05).
15) Ping, Y., Kontogiannis, K. and Lau, T.C.: Transforming Legacy Web Applications to the MVC Architecture, *Proc. of STEP 2003*, STEP 2003, pp.133–142 (2003).
16) 吉賀祥太，早川智一，疋田輝雄：Flex アプリケーションの iOS 環境での HTML5 変換による実行，FIT2011 講演論文集（第 4 分冊），FIT2011, pp.385–388 (2011).
17) Zhang, P. and Chen, J.: The Scheme to Extending the Web Application Frame with XML, *Proc. of CICC-ITOE 2010*, CICC-ITOE 2010, pp.149–152 (2010).