

大規模 RIA アプリケーション開発を支援するシステム

氏 名[†] 福田 浩章 氏 名[†] 山本 喜一

所 属[†] 慶應義塾大学 所 属[†] 慶應義塾大学

1. はじめに

RIA アプリケーションは従来型ウェブアプリケーションの欠点である貧弱な操作性、ネットワークのオーバーヘッドによる遅延、オフライン処理などを解決でき、次世代のウェブアプリケーションとして期待されている。RIA アプリケーションでは、洗練されたインタフェースや、インタラクティブな処理を実現し、ユーザの操作性を向上させる傾向にある。そのため、従来型ウェブアプリケーションと同等、またはそれ以上にビジネスロジックを記述する開発者と、インタフェースやアニメーションを設計するデザイナーとの協業が必要不可欠である。しかし、アプリケーション開発では完成後はもちろん、開発中におけるデザイン変更は一般的であり、その影響はデザイナーだけでなく開発者にも及ぶ。そのため、規模が大きくなればなるほど変更による開発コストは増大する。

そこで本論文では、デザイナーと開発者の作業を分離し、デザイン変更による開発者の影響を削減するシステムの提案と具体的な手法について述べる。

2. 開発プロセス

図 1 に従来型ウェブアプリケーションの開発プロセスを示し、番号で示す手順について述べる。

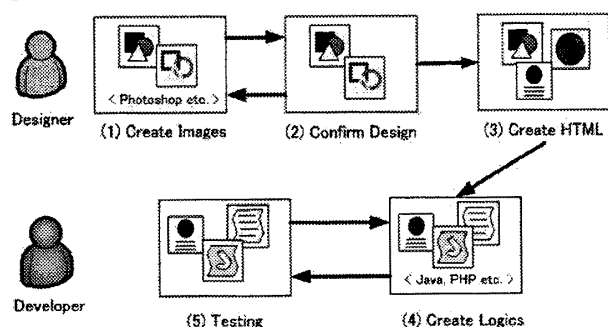


図 1. 開発プロセス

1. デザイナはクライアントの要求をもとに、Photoshop や Illustrator などのツールを使用してウェブサイトと全体をデザインする。
2. クライアントがデザインを確認し、変更があればデザイナーが修正を行う。
3. 一度デザインが決定されると、デザイナー(または HTML コーダ)はデザインから画像を切り出し、HTML と CSS を使用して静的な HTML でウェブサイトを完成させる。
4. 開発者が Java や PHP, Javascript などを使用してビジネスロジックを実装する。この時、ユーザの行動に応じて動的に表示データを作成するため、開発者はデザイナーが作成した HTML にロジックを埋め込む。
5. 開発者は単体/結合テストを行い、不具合があれば手順 4 に戻り、修正する。

現在のウェブアプリケーション開発では、多くの場合 MVC アーキテクチャが採用されるため、ビジネスロジックをインタフェース部分に埋め込むことは稀である。しかし、ロジックの実行結果をインタフェースに反映させるため、動的に HTML タグを生成する必要がある。そのため、手順 3 で作成した HTML に手を加えざるを得ない。特にデザインを重視する場合、デザイナーが作成する HTML は非常に複雑であり、そこにロジックを埋め込む作業は開発者にとって非常に困難な作業になる。また、ウェブサイトのデザインも初期のまま変更されないことは稀であるため、変更されるたびにデザイナーだけでなく開発者の手を煩わすことになる。このため、大規模開発になればなるほど変更に伴うコストは増大する。これは、本来文書の論理構造を記述するために作成された HTML がインターネットとともに普及し、ブラウザだけで利用できるウェブアプリケーションに発展したことが一つの原因である。

3. アプローチ

本論文では、RIA 開発フレームワークである Flex を利用し、デザイナーと開発者の作業をソースレベルで完全に分離することにより、大規模開発における変更コストの削減を行う。

A system for supporting development of large scale RIA applications

[†] Keio University

3.1 Flex フレームワーク

Flex では、ボタンやラベルなどのコンポーネントがあらかじめ用意されており、XML ベースの言語である MXML を利用してインタフェースを作成することができる。また、Flash や Illustrator で作成したコンポーネントを取り込み、独自コンポーネントとして利用することも可能である。したがって、デザイナーは普段使い慣れたツールでインタフェースを作成することが可能である。一方、ロジックの記述には ActionScript 3.0 を利用することができるため、開発者はオブジェクト指向を利用し、インタフェース部分である MXML とロジック部分のクラスを分離して記述することができる。しかし、Flex アプリケーションはイベントドリブンで動作するため、コンポーネントのイベントを処理するイベントハンドラの記述、およびハンドラを所有するオブジェクトの生成は MXML に記述する必要がある、デザイナーと開発者の作業をソースレベルで完全に分離することはできない。

3.2 Dependency Injection

3.1 節の問題を受け、本研究ではイベントソースとなるコンポーネントと、イベントを処理するハンドラの関連付けを開発時ではなく実行時に行うことにより、インタフェースとロジックをソースレベルで分離可能にする。これを実現するため、デザイナーと開発者は次の 1 から 3 に示す規約に従って開発を行う必要がある。そして、この規約に従う限り、次の 4 から 6 の手順で実行時にイベントハンドラとコンポーネントの関連付けが行われる。

1. デザイナーは MXML を用いてインタフェースを設計する。このとき、id 属性を使用し、重複のないよう各コンポーネントに名前をつける。
2. 開発者はデザイナーが作成したインタフェース 1 つに対して、イベント処理のクラス（リスナークラス）を 1 つ作成する。リスナークラスにはインタフェースの各コンポーネントで発生するイベントを処理するメソッドを、「コンポーネントの ID + イベント名 + “Handler”」という命名規則に従って記述する。たとえば、“calcButton”と id が付けられたコンポーネントのクリックイベントを処理するメソッド名は“calcButtonClickHandler”となる。
3. デザイナーが作成したインタフェースと、開

発者が作成したリスナークラスを、同一の MXML ファイルにタグとして記述する。このとき、インタフェースには“View”、リスナークラスには“Listener”を接尾語にし、同一の名前をつける。

4. アプリケーションが開始されると、各コンポーネントの生成完了イベントを契機に、“View”を接尾語としたコンポーネントの参照を取得し、保持する。
5. View と同じ階層に付加されたリスナークラスの参照を取得する。View とリスナークラスの対応は、それぞれにつけられた名前と、接尾語をもとに判定する。
6. View に付加されたコンポーネントの名前と、リスナークラスに定義されたメソッド名をもとに、手順 2 で示した命名規則に合致した場合、addEventListener メソッドを利用してイベントとイベントハンドラを関連付ける。

このように、本システムを利用することによって、インタフェースである MXML と、イベントハンドラを記述するクラスをソースレベルで完全に分離することができる。そのため、インタフェースに変更がある場合でも、コンポーネントの id が変更されない限り動作が保障されるため、変更に伴うロジックの改修、テストといった開発者の作業を削減することが可能になる。

4. まとめと今後の課題

本論文では、RIA 開発ツールである Flex を利用し、ウェブアプリケーションのインタフェースを作成するデザイナーと、ビジネスロジックを実装する開発者の役割を、ソースレベルで完全に分離するシステムの提案と、具体的な手法について述べた。本システムにより、従来デザイナーと開発者で共有せざるを得なかったインタフェースのソースを完全に分離することができるため、デザイン、またはロジックの変更が生じても、独立して作業をすることができ、開発効率の向上が期待できる。

今後の課題として、本論文で述べたように、イベントとイベントハンドラの対応は id をベースにしているため、タイプミスなどによって実行時エラーが増加する可能性がある。そのため、開発を支援する仕組みとして Eclipse プラグインの開発を検討している。