NRI IT Infra 基盤は進化する
技術創発 －NRI情報技術レポート－
2012年4月発行

クライアント開発技術(RIA・HTML5)の選定時・システム構築時の考慮点

野村総合研究所
共通基盤推進部 上級テクニカルエンジニア 佐藤 洋平 (さとう ようへい)
共通基盤推進部 主任テクニカルエンジニア 山城 俊介 (やましろ しゅんすけ)

フロント技術に関する新技術の調査、アーキテクチャの検討、および、新技術をスムーズに適用するためのフレームワークの設計・開発を行っている。

要 旨

システムのユーザビリティの向上がフロントシステムにおいて重要なポイントとなっている。

従来から、ユーザに使いやすいシステムを構築する手法として、RIA(Rich Internet Application)技術が用いられてきているが、スマートフォンやタブレット PC のようなタッチデバイスの登場・HTML5 の登場によって技術の選定やアーキテクチャの設計が複雑になってきている。

本稿では、RIA・HTML5 技術に関する動向を紹介し、さらに、これらを大規模システムに適用する際の技術選定のポイントや、設計上の考慮ポイントについて説明する。

1．はじめに

近年、端末（デバイス）の多様化と、HTML5の登場により、クライアント開発技術の選定やアーキテクチャ・処理方式の検討が複雑になってきている。本稿ではNRI（野村総合研究所）におけるR&D活動や筆者の実プロジェクトでの経験をもとに、クライアント開発技術の選定時とアーキテクチャ・処理方式の検討時に必要となるポイントについて、概要を説明する。

2．クライアント開発技術の変遷

Webシステムの広まりと共に課題としてあがった「システムの使いやすさ」に対応するため、Adobe Flex(以降、Flex)やMicrosoft Silverlight(以降、Silverlight)等のRIA[1]技術を業務システムのインターフェースとして用いる動きが広まってきた。

(出所)
W3C HTML Working Group Charter 2.3 Milestones より作成
http://www.w3.org/2007/03/HTML-WG-charter.html

図 1 HTML5ロードマップ

一方で、iPhone/iPadやAndroid等のスマートフォンやタブレットPCのように、コンシューマ向け端末の発達に追随して、企業システムでもこれらの端末の特性を生かした新しい業務システムや、新しいビジネスの検討が開始されている。

また、上記背景と合わせて、HTML5技術の仕様が策定されようとしており(図 1)、クライアント開発技術が大きく変わろうとしている。

さらに、2011年11月にAdobeより、以下のような発表がなされた。

・携帯向けFlash Player開発の廃止
・HTML5開発環境への追従
・端末向けにAIR技術採用を推進

Flashのターゲット範囲の変更とHTML5へのシフトに関する方針が告げられた瞬間である。

このように、クライアント開発技術を取り巻く環境は、日々、目まぐるしく変化をしている。我々は常に状況を確認し、最適なクライアント開発技術の選定方法・アーキテクチャについて検討する必要がある。

3．クライアント開発技術選定時のポイント

本章では、クライアント開発技術を選定する際に検討すべき5つのポイントについて説明する。

（1） ユーザシステムの特性と機能
（2） サポートする端末
（3） 開発コスト

[1] RIAとは「豊かな表現力を持ち、より機能的で、操作性の良いWebの仕組み」と定義しており 、単に見た目だけではない、より「ユーザのため」のアプリケーションを指したものである(RIAコンソーシアム ： RIAシステム構築ガイド Essential 2より)。

（4） ベンダーサポート

（5） 技術の成熟度と今後の動向

それぞれについて以下に述べる。

(1) ユーザシステムの特性と機能

クライアント開発技術は、保持する機能に差異がある。ユーザシステムの要件を満たすために必須の機能が存在しているか、あるいは運用で代替可能であり、該当機能が存在していなくても問題ないかなど、影響度を鑑みて検討することが肝要である。

機能特性で検討すべき点の一例を以下に示す(図 2)。例えば、利用する外部機器との接続・連携が可能および容易であるか、オフライン時に業務を停止することなく業務を遂行するための機能が盛り込まれているかなどを評価する必要がある。

(2) サポートする端末

クライアント開発技術は、動作する環境が技術によって異なる。サポートする端末や、今後サポートする可能性がある端末を検討の上、選択する必要がある(図 3)。

(3) 開発コスト

① 開発ツールの生産性

Flex, Silverlight, HTML5 の開発ツールに関して、概要を説明する。

No.	機能特性
1	直感的で分かりやすい画面が提供可能(インタラクション等)
2	PDF連携が可能
3	低速回線でも使用可能
4	オフラインでも使用可能
5	動的な画面生成が可能
6	外部機器との接続・連携が可能
7	Office連携が可能
8	ネイティブAPI連携が可能
9	メディア(動画・音声)連携が可能
10	IME(文字変換ソフト)連携が可能
11	PC環境非依存
12	モバイル端末で利用可能

図 2 RIAシステムに求められる特性

端末		HTML5	RIA	
			Flex	Silverlight
スレートPC/タブレットPC	Android	○	○(AIR)	×
	iPad	○	○(AIR)	×
	Windows	○(※)	○(Flex/AIR)	○
スマートフォン	Android	○	○(AIR)	×
	iPhone	○	○(AIR)	×
	Windows Phone	○	×	○

(※) IE9/10を想定

図 3 端末と利用可能技術

Flexに関しては、Adobeがコーディング環境としてFlash Builderを、デザイン環境としてPhotoshop, Illustrator, Catalyst等を用意し、相互で連携できる機能を備えている。この他、フリーの開発環境として、OSSのFlashDevelopを用いる方法や、OSS（オープンソースソフトウェア）のFlex SDKとApache Antを組み合わせる方法も挙げられる。ビジュアルデザインが作業の主である場合は前者(有償ツール)が必須であるが、純粋なロジック開発やツールサポートがいらない範囲のコンポーネント開発などは後者(無償ツール)でよいだろう。

Silverlightに関しては、Microsoftがコーディング環境としてVisual Studioを、デザイン環境としてExpression Blendを用意しており、こちらも相互連携の機能を備えている。この他、フリーの開発環境として、MicrosoftのVisual Web Developer Expressを用いることもできる。ただし、一部デバッグ機能やリファクタリングサポート等の支援機能がないため開発効率はVisual Studioに比べると見劣りするだろう。

HTML5に関しては、コーディング環境が持つべき機能の一部が、「開発者ツール」としてブラウザに備えられ始めている。特に、JavaScriptに関するデバッグ機能(インスペクタ・ブレークポイント等)や、DOM動的書き換え等、開発生産性が向上する機能が充実しつつある。また、デザイン環境としてAdobe Dreamweaver(Webオーサリング), Adobe Edge(インタラクションオーサリング), Adobe Shadow(デスクトップ・モバイル同時開発補助)など、徐々にHTML5に関する製品が出揃いつつある。

上記の通り、各クライアント開発技術には、複数の開発ツールが存在する。さらに、開発ツールそのものに特性がある。どのツールをどのように利用するかを検討した上で、開発ツールの生産性を評価すべきである。

② 開発者の確保

大規模プロジェクトにおいては、スキルを有する開発者の確保が重要なポイントとなる。オープンな技術については開発者が集まりやすいとされるが、今までの構築経験・ノウハウを活かすため、実際の開発メンバーのスキルを調査の上選定する必要がある。

また、システムの特性に応じて、デザイン的要素が強いか、システム的要素が強いかというところも重要なポイントとなる。前者の場合は、デザイナーとの連携が容易な技術を選択すべきである。

(4) ベンダーサポート

各技術をサポートするベンダーのサービスレベルについても、選定のポイントとなる。検討ポイントとしては下記のようになる。

- サポート料金
- 適用技術バージョンの保守期限(一般的に追加料金を支払えば延長サポート

が得られる場合がある)

- 問い合わせ形態(電話・メール・オンサイト)
- サービスレベル(パッチ提供の有無・サービス時間帯)

なお、Flexに関してはApache財団へ寄贈されたこともあり、今後のサポート動向に注視する必要がある。

(5) 技術の成熟度と今後の動向

① 技術の成熟度

実際に開発が開始された後、ランタイムに関する不具合や、開発ツールの不具合に影響を受ける場合がある。このため、技術の成熟度についてはあらかじめ見極めておく必要があるだろう。

特にHTML5の技術は、ブラウザの成熟度が大きく関係する(図 4)。準拠度だけでなく、ブラウザ毎に細かい動作が異なることが見受けられるため、その点注意が必要である。現状では、マルチプラットフォームではなく単一プラットフォーム向け、特にタッチデバイス向けにターゲットを絞ると、リスクを軽減できる。

② 今後の動向

冒頭で述べたが、Adobeのように今後の方針を明確に示している場合は、技術の方向性を確認する上で参考になる。

一方で、HTML5はタッチデバイスを中心に活用が進みつつあるが、後々は基幹システムへ導入されていくことになるだろう。そのためには、HTML5の仕様が確定し大きく変更されないフェーズになること、ブラウザが仕様に準拠しHTML5の機能が充分に活用できるようになることが必要条件である。

Summary

Calculation of support for currently selected criteria

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
		3.6: 53%						10.0: 41%	2.1: 39%
	6.0: 8%	8.0: 81%				3.2: 43%		11.0: 59%	2.2: 44%
	7.0: 12%	9.0: 81%				4.0-4.1: 50%		11.1: 65%	2.3: 46%
	8.0: 22%	10.0: 82%	16.0: 87%	5.0: 65%		4.2-4.3: 57%		11.5: 66%	3.0: 62%
Current	9.0: 48%	11.0: 82%	17.0: 87%	5.1: 74%	11.6: 70%	5.0: 73%	5.0-6.0: 27%	12.0: 74%	4.0: 67%
Near future	10.0: 77%	12.0: 82%	18.0: 87%	6.0: 75%	12.0: 75%				
Farther future		13.0: 82%	19.0: 87%						

図 4 ブラウザのHTML5への準拠度

(出所) 「When can I use...」より引用 (2012/03/14 時点情報)
http://caniuse.com/

4．アーキテクチャ検討時のポイント

RIA，HTML5のアーキテクチャの設計時に重要となるポイントについて説明する(図 5)。

（1） 配布・運用

（2） 他システムとの混在

（3） データ保持方式

（4） 通信方式

（5） セキュリティ

（6） 障害運用

それぞれについて以下に述べる。なお、図5中の番号は対応する上記の各項目の番号である。

(1) 配布・運用

① ランタイムの配布方式

ランタイム（ランタイムライブラリ）やブラウザの配布・運用についての検討ポイントは、以下の点である。

・バージョンアップに関する考え方

・ランタイム配布サーバの考え方

一般に利用するランタイムは事前にバージョンを固定する。しかし利用される端末がインターネットに接続されている場合、自動的に更新される場合があるためこれを停止するなどの方針を検討する必要がある。

また、そのランタイムにセキュリティ脆弱性が発見された場合に確実にアップデートするためにWSUS（Windows Server Update Services）等を用いた「ランタイム配布サーバ」を用意する必要性がある。なお、ランタイムの

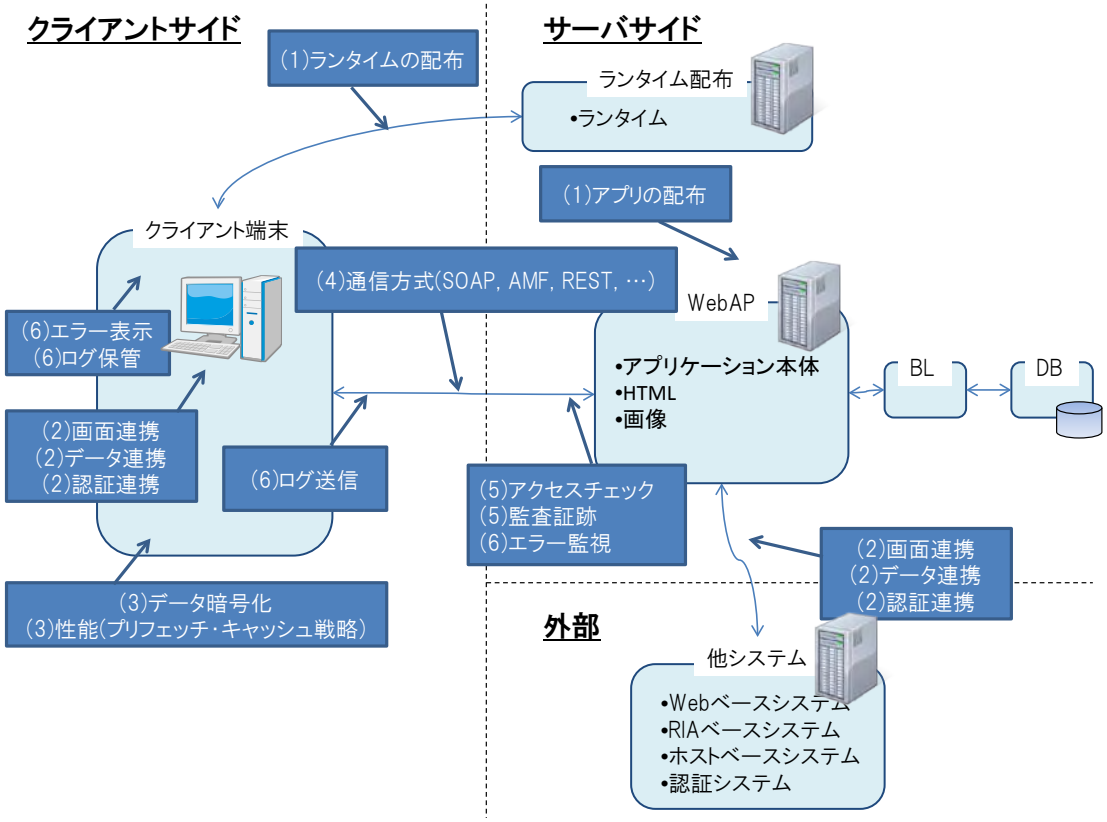

図 5 アーキテクチャ概要図

配布にはライセンス契約締結が必要な場合もあるため、事前の確認が必要である。

② アプリケーションの配布方式

アプリケーションの配布は、性能と合わせて考慮が必要である。

RIA, HTML5は、画像などのリソースに加え、これまでのWebと比べてインタラクションロジック・業務ロジック等に関する開発が多い傾向にあるため、それに合わせてモジュールサイズが大きくなる傾向にある。そのため、モジュールをいつ、どのようにしてクライアント端末に配布するかを考えなければならない。たとえば、必要となるまではダウンロードしない遅延ダウンロードや、処理の空きがあるうちにダウンロードするプリフェッチなどが考えられる。その際、モジュールの旧戻しや、ブラウザのキャッシュ利用状況等も合わせて検討が必要となる。

その他、AIR (Adobe Integrated Runtime)等、モバイル向けアプリケーションとして配布する場合はMDM(Mobile Device Management)ツール活用の検討が必要である。また、HTML5アプリケーションの配布に関しては、通常はサーバアクセス時に1ファイルずつダウンロードされるが、Application Cacheという仕組みを利用することで、アプリケーションが指定したリソースをクライアント環境にキャッシュすることができる。これを用いれば、頻繁に読み込まれる部分をクライアント側にキャッシュさせ、ロード時間・帯域の節減をすることが可能となる。

(2) 他システムとの混在

企業システムの開発においては、既存のシステム(レガシーなシステムやERPパッケージなど)が存在していることが多いため、他システムとの連携方法について検討する必要がある。

連携する場合の考慮点としては、以下が考えられる。

- 画面連携
- データ連携
- 認証連携

ここでは、RIAからHTMLで画面が構成されたアプリケーションに画面連携する場合の例について説明する。

アプリケーション間を画面同士で連携する場合は、次の二つの方式が考えられる。

一つ目は、連携先アプリケーションのHTML画面を自アプリケーションと併存させ、表示する方式である。この場合、RIAと並列に存在させる方法(OBJECTタグとIFRAMEタグを併存させ、IFRAME内部に連携先アプリケーションを表示)が一般的に取られる。連携方法としては、Flash PlayerやSilverlightランタイムの機能を用いてRIAからJavaScript(JScript)を利用することになる。

二つ目は、自アプリケーションの内部に表示する方式である。この場合、RIA内にWebVie

w(ブラウザコンポーネント)を配置し、その中に表示する方式である。なお、ブラウザ上で動作する技術(Flash Player, Silverlightランタイム)ではブラウザコンポーネントがサポートされておらず、利用できないため注意が必要である。一方、AIRやSilverlight OOB(Out-of-Browser)ではこのコンポーネントがサポートされているため利用可能である。

(3) データ保持方式

RIA, HTML5では、クライアント側にデータを保持することが可能であるため、各種データをどこに保持しながら動作するのか、検討が必要となる。

主な検討点は、以下のとおりである。

・データ暗号化

・性能

① データ暗号化

ノートPCや、タブレット端末など外部に持ち出す場合は、情報漏えい等を防止するため、端末にキャッシュされたデータの暗号化が重要になる。そのため、データを保存する際に、常に暗号化する仕組みを導入する必要がある。

② 性能

データをいかに保持するかで、性能改善が図れる。

図 6のように、データを保持する箇所は、大きく分けてサーバサイドの「永続化データ」、クライアントサイドの「オンメモリデータ」・「ローカルDB」がある。目的に応じてこれらの保持箇所を活用することが肝要となる。ここでは、「サーバへの問い合わせ時間短縮」と「非同期更新」について説明する。

まず「サーバへの問い合わせ時間短縮」には、クライアントサイド「オンメモリデータ」・「ローカルDB」にデータをロードする方式が有効である。問い合わせ処理時間は「オンメモリデータ」のほうが速いが、大量のデータを保持できないため、データ量に応じて使い分けする必要がある。そして、いつロードするかについても検討が必要である。初回にすべてダウンロードする方式の場合、運用初日に大量のアクセスが来る危険性がある。

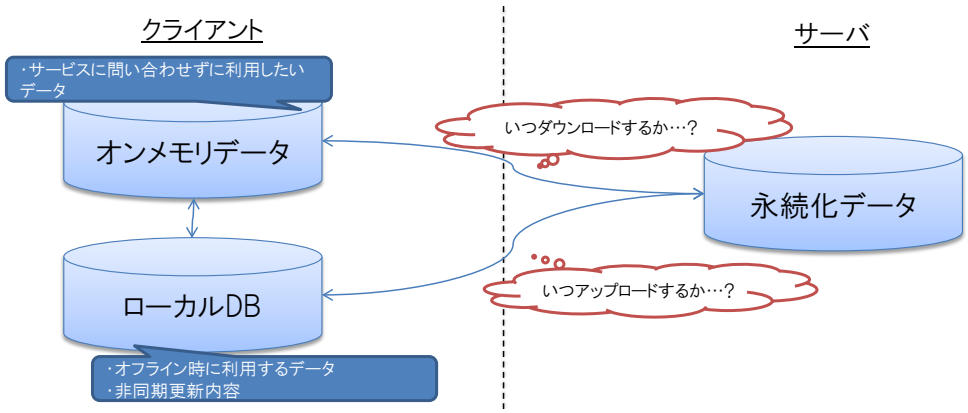

図 6 データ保持方式

これに対応するためには、優先度に応じて必要なときに必要な分だけ読みだすようにし、残りは処理の空き、もしくは帯域が空いている時間帯にダウンロードすると効率化が図れる。

次に「非同期更新」には、「ローカルDB」に更新内容をためておくことが有効である。ログやエラー内容など、即アップロードする必要がないものを一時的にためておき、業務ピーク時や帯域が狭い時を避けてアップロードすると効率化が図れる。

(4) 通信方式

通信方式による性能の差異は、データ構造やデータ量など、業務特性によって異なり、一意にどの方式が良いか決めることができない。

例えば、複雑な構造を持つ表を表示したい場合は、単純にデータをXML化すると、電文内に占めるタグ量が増加し、シリアライズ・転送・デシリアライズの時間がそれぞれ増大するという問題が発生する。この問題に対応するために、AMF3等のバイナリデータフォーマットを用いるなどの方式選定が必要となる。

(5) セキュリティ

クライアント端末上のデータに関するセキュリティについては、(3)で述べた。

その他にシステムへのアクセスへのセキュリティについて検討が必要である。アクセス

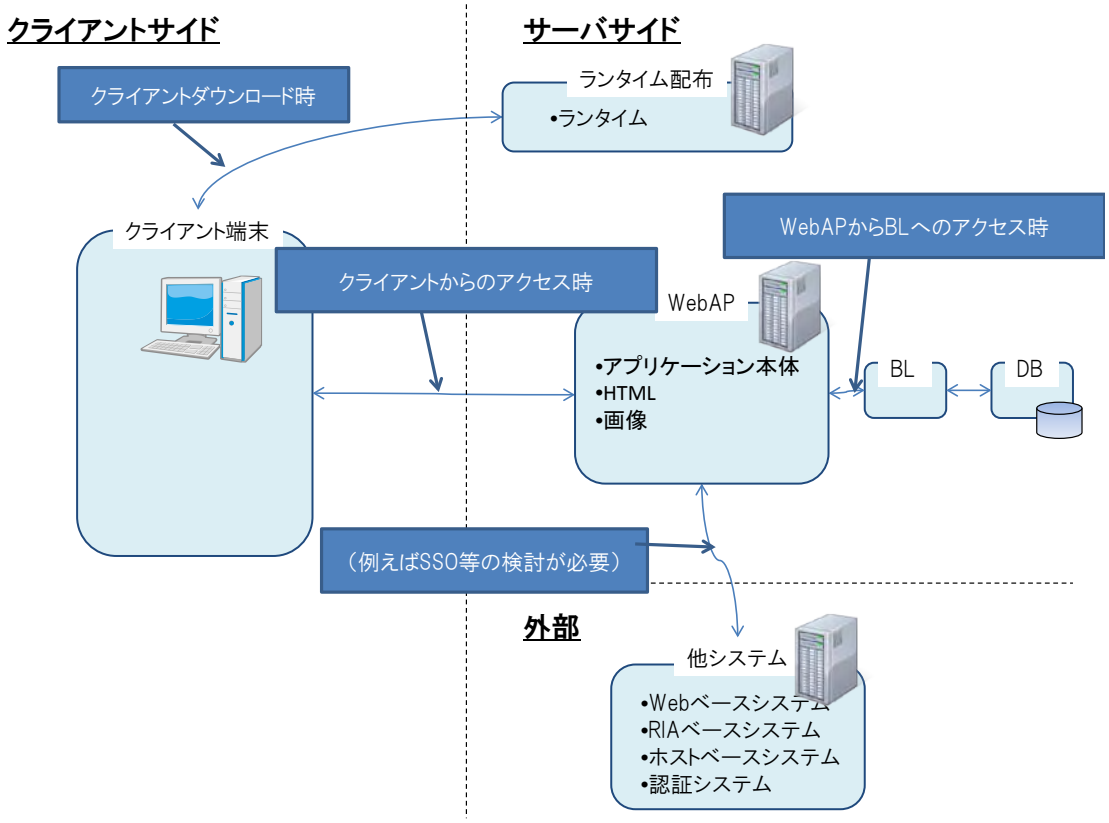

図 7 アクセス制御ポイント

制御を行える箇所は、以下の点になる(図7)。

・クライアントダウンロード時

・クライアントからのアクセス時

・WebAPからBL（Business Logic）へのアクセス時

システムの要件に応じて、アクセス制限個所・監査証跡の取得個所を決定する必要がある。

(6) 障害運用

RIA, HTML5では、クライアントサイドで実行される機能が豊富であるため、クライアントサイドで障害が発生した場合の通知方式や障害解析の方法を検討する必要がある。

たとえば、ローカルストレージにエラー内容等のログ情報を格納しておき、復旧時にサーバサイドにログを送信し、分析・監視を実現する等、障害運用をスムーズにする仕組みを検討すべきである。

5．まとめ

本稿では、クライアント開発技術の選定時と、アーキテクチャ検討時のポイントについて概要を説明した。

今後はHTML5技術の登場により、端末上での実装方法や、オンライン・オフラインの検討などさらに検討ポイントが増えると考えられる。常に、ユーザビリティの向上とシステム構築コスト・運用コストを意識しながら技術選定・方式の策定を行っていく必要がある。

NRIではクライアント開発技術について調査と評価を実施しており、今回はその一端を紹介した。今後も引き続きよりよいユーザーインターフェース実現に向けた技術蓄積を行っていく予定である。