Flex コンポーネントの Web 標準技術への移行に向けた性能見積手法の確立

Consideration of Migration Web UI Component from Flex to Web Standards

複雑な工程間の関係を可視化し，プロジェクト管理業務を効率化する SynViz S2 では，マウス操作によるユーザビリティの高い編集機能を提供している。多数のオブジェクトが配置された状態でのスムーズな操作を実現するため Adobe Flex を活用しているが，実行基盤である Flash Player のサポートが 2020 年末で終了する。これに伴い，Web 標準技術へ移行する必要があるが，Flex と比較し描画性能が低いため，移行に際して事前の性能見積もりが重要となる。そこで，描画処理をコンポーネント固有のロジック処理部分とオブジェクトの描画部分に分けて性能見積もりを行うことで，移行後の性能を見積もる手法を確立した。本手法を用いて SynViz S2 のガントチャートコンポーネントを Web 標準技術へ移行した場合の性能を評価し，移行が可能であることを明らかにした。

内海　宏律　Utsumi Hironori
太齋　真吾　Dasai Shingo
石倉　直弥　Ishikura Naoya

1. はじめに

(株)日立ソリューションズ東日本(以下，HSEと記す)では，工程間の複雑な関係をガントチャートにより可視化し，現場でのプロジェクト管理業務を効率化するSynViz S2を製品化している。本製品は，マウスによる高度な編集機能をWebブラウザ上で実現しており，ユーザビリティの高さが他社との差別化要素となっている。SynViz S2はFlexを用いることで，多数のオブジェクトが配置された状態でもユーザが快適に利用できる高い操作性を実現している。

Flexを活用したUIコンポーネントはWebブラウザに組み込まれたFlash Player上で実行される。しかし，AdobeよりFlash Playerの提供を2020年末で終了するとの発表があり[1]，すでに主要なWebブラウザでもFlash Playerの無効化[2][3]が進んでいる。そのため，FlexをWebブラウザが標準でサポートする技術を用いたUIコンポーネントに移行する必要があるが，一般的にWeb標準技術はFlexと比較して描画性能が低いことが問題になる[4]。Web標準技術によるUIコンポーネントの実現可能性の判断には早期の性能見積もりが必要となるが，実装前に性能を見積もることは難しい。そこで，既存のコンポーネントの処理時間をもとに，移行後の性能見積もりを行う手法を確立し，Web標準技術への移行可能性を判断するための定量的な性能見積もりを可能とした。本手法を用い，ガントチャートコンポーネントの移行可能性を評価した。

2. Web 標準技術へ移行する際の課題

既存のFlexコンポーネントの移行では，多くのオブジェクトが画面内に配置された場合でも，十分な応答速度を確保する必要がある。しかし，Web標準技術による画面描画は，Flexと比較し実行速度が遅く[4]，性能上の懸案事項となる。移行可能性の判断では，性能見積もりが重要となるが，定量的な見積もりは困難である。また，Web標準と呼ばれる中にも表1のように複数の描画技術が存在するが，それらの性能特性が不明確であり，選択すべき描画技術の決定が困難である課題がある。これらの課題を解決することで，性能上の懸案を早期に評価可能となり，移行可能性を判断することができる。

表1 Web標準となった描画技術

	Canvas	SVG	WebGL
標準化団体	W3C	W3C	Khronos Group
描画モデル	即時 モード	保持 モード	即時 モード
描画方法	描画 API の 呼び出し	SVG DOM の操作	描画 API の 呼び出し
要件	なし	なし	GPU が必要

3. 移行時の性能見積方法の提案

描画処理をコンポーネント固有のロジック処理部分とオブジェクト描画部分に分けて性能見積もりを行うことで，移行後の性能を明確化する手法を提案する。

3.1 画面描画のフローと描画技術の移行による影響範囲

画面の描画処理時間は，ロジック処理時間とオブジェクト描画処理時間から構成される。ロジック処理では，各オブジェクトの表示位置の計算や画面内に含まれるオブジェクトの判定など，コンポーネントごとに固有なロジックを実行する。オブジェクト描画処理では，描画対象となったオブジェクトをそのプラットフォームで利用可能な描画技術を用いて画面上に描画する。ロジック処理で行われる内容は描画技術に依存しない処理である。一方，オブジェクト描画処理は描画技術に依存する処理であるため，性能見積もりには移行後の技術を用いてオブジェクトを描画した場合の処理時間を算出する必要がある。そこで，描画処理全体をロジック処理部分とオブジェクト描画部分の2つに分けることで性能見積もりが可能になる。

3.2 ロジック処理部分の性能見積

ロジック処理部分の性能は，Flexコンポーネントの ActionScriptで書かれたソースコードをJavaScriptに移植することで実測できる。しかし，実測のためにはソースコード一式を移植する必要があり，早期に性能面での実現可能性を評価する目的は達成できない。ロジック処理部分で実行される内容自体は，FlexとWeb標準では大きく異なることはない。そこでこの点に着目し，Flexコンポーネントで実測した処理時間をもとに，移行後の性能を見積もる。

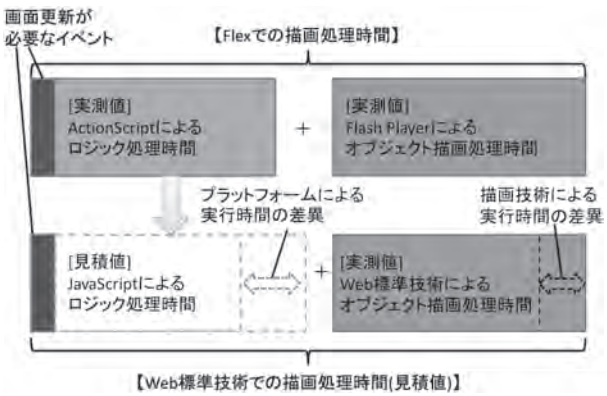

図1 Flexコンポーネントの実行時間に基づく性能見積

Flexを用いて作成されたコンポーネントの描画は，保持モード[5]によって行われる。描画では，図1のようにscrollイベントなどの画面更新が必要なイベントを起点とし，ActionScriptによるロジック処理が実行される。その後，Flash Playerによるレンダリング処理で画面が更新される。Flexコンポーネントのソースコードが存在する場合，ソースコードの中に計測コードを埋め込むことでActionScriptによるロジック処理に必要な時間とFlash Playerによるレンダリングに必要な時間を個別に取得することができる。Flexの描画は保持モードであるため，ActionScriptの実行時間内には描画時間は含まれない。そのため，この時間をコンポーネント固有のロジックの処理に必要な時間とみなすことができる。

ActionScriptとJavaScriptの実行時間に差異があることが想定できるが，近年のJavaScript実行エンジンの高速化も考慮し，見積値の算出ではJavaScriptの実行時間がActionScriptの実行時間と比較し2倍程度遅い[6]とし，係数として2.0をかけることで求める。

3.3 オブジェクト描画部分の性能見積

Web標準技術では，描画APIの呼び出しや描画モデルの操作を繰り返すことで画面を描画する。一方，図2に示すガントチャートなどの業務向けWebアプリケーションでは，画面が矩形や直線，文字列などの組み合わせによって構成される。つまり，画面上に表示されると見込まれる図形の種類と個数およびサイズが算出できれば，それと同様の内容を実際にWeb標準技術で描画する性能を測定することで，移行後の性能見積もりができる。

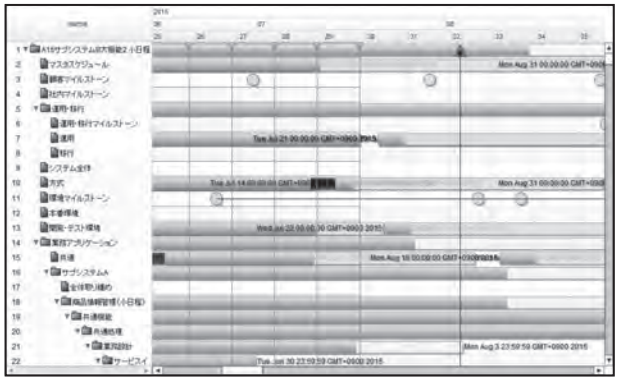

図2 SynViz S2のガントチャート画面

そこで，画面内に描画する必要がある図形の種類と個数を入力とし，表1に示した3つの描画技術を利用して画面を描画した際の性能を測定するツールを開発した[7]。

本ツールはガントチャート以外にも汎用的に活用可能であり，移行対象とするコンポーネントによらず広く適

用できる。これにより，画面に表示されるオブジェクトの種類と数の情報だけから描画に必要な時間を定量的に見積もり（もしくは推定）可能となる。また，表1に示す技術ごとに描画性能の比較が可能であるため，選択すべき描画技術が決定困難である課題が解決できる。

以上より，ActionScriptの処理にかかる時間に係数をかけた値と描画性能測定ツールにより得られた実測値を図1のように加算することで，Web標準技術で構築したコンポーネントの処理時間を見積もることができる。その結果，開発着手前に描画性能見積もりができない課題が解決できる。

4. 移行時の性能見積と技術選定

SynViz S2のガントチャート画面に対し提案手法を適用し，移行後の性能を見積もり，Web標準技術に移行した場合の実現可能性の判断と採用技術を決定した。

4.1 性能見積条件

SynViz S2で表示される実際のプロジェクトの画面を想定し，性能見積もりの対象とする条件を設定した。性能見積もりにあたっては，図2に示すようなプロジェクト管理で利用される標準的な規模のプロジェクトを想定し，その工程状況を可視化した画面を想定した。

画面の大きさは1280×1080ピクセル(px)のディスプレイ上で，SynViz S2で標準的な上部と左部にメニューがあるレイアウトでの表示を想定し，1000×600pxと設定した。そのうえで画面内に描画されると想定されるオブジェクト数を算出し表2の結果を得た。オブジェクトのサイズはガントチャート画面上に存在するオブジェクトのサイズを考慮し，200×20pxと設定した。

表2 ガントチャート画面を構成するオブジェクト数

図形	オブジェクト数[個]
矩形	127
三角	21
円	30
直線	578
テキスト	161
画像	49

4.2 目標性能と測定環境

目標性能はユーザがシステムの反応が瞬時に行われていると感じる限界値[8]とされる100msと設定する。性能測定の環境は表3のとおりとし，クライアントはInternet Explorer (IE) 11を用いた。測定は各100回行った。

表3 性能測定環境

項目	スペック
OS	Windows 8.1 Pro
CPU	Intel Core i7 4770 3.4GHz
メモリ	8GB
GPU	Intel HD Graphics 4600

4.3 提案手法に基づく描画性能の見積結果

Flex版ガントチャートコンポーネントでの処理時間を表4に示す。各値は平均値とし表の括弧内には分散値を示す(以下，同様)。また，画面の表示技術ごとの性能測定結果を表5に示す。

表4 FlexによるUIコンポーネントの描画処理時間

計測値[ms]		
ロジック処理	オブジェクト描画	合計
41.55 (120.65)	45.50 (69.85)	87.05 (161.25)

表5 実アプリケーションを想定した描画性能見積

描画技術	計測値[ms]
Canvas	18.65 (9.13)
SVG	15.28 (4.32)
WebGL	14.49 (1.12)

ロジック処理時間とWeb標準技術による描画時間を合算した結果を表6に示す。表4のFlexのロジック処理に対し係数として2.0をかけ，表5で取得した各描画技術の描画時間と合算することで移行後の性能を算出した。性能目標である100msを超過したものについては背景をハイライトして示す。

表6 Web標準技術への移行時の描画時間見積値
(表4のロジック処理時間×係数(2.0)＋表5の計測値)

描画技術	見積値[ms]
Canvas	101.75
SVG	98.38
WebGL	97.59

SynViz S2のユースケースで用いられる規模のガントチャートの描画性能は，WebブラウザによるJavaScriptの実行時間がActionScriptの2倍遅いとした場合でも102ms程度に収まると見積もれる。また，描画技術による性能差は最大でも4ms程度であり，体感可能な差はない。そのため，すべての描画技術でガントチャートは実現可能であるといえる。したがって，Web標準技術によるコンポーネントへ移行しても大きな問題は発生しないと考えられ，移行可能と判断できる。

4.4 採用技術の選定

性能面では，どの描画技術を用いた場合でも，既存のFlexを用いたコンポーネントと同等の性能を確保できる見込みを得た。SynViz S2は，クラウド環境での実行もサポートしており，クライアントの環境は顧客により様々であると想定できる。そのため，WebGLのようなハードウェアに強く依存する技術は採用しにくい。また，オブジェクトの選択や移動の際には，JavaScriptによるオブジェクトの状態制御が必要となる。SVGではオブジェクトの選択などの制御がブラウザ側で実行されるため，プログラムによる制御が困難である問題がある。そこで，今回のSynViz S2のWeb標準技術への移行では，Canvasを採用することとした。

5. おわりに

Flexと比較し描画性能が低いとされるWeb標準技術への移行可能性を早期に明確化するために，移行後の性能を見積もる方法を確立した。これにより，課題であった定量的な性能見積もりと採用すべき技術の選定を可能とした。

SynViz S2のガントチャートはWeb画面でもデスクトップ製品と同等の操作性と表現力を兼ね備えている。これを下支えしていたFlash Playerが2020年でサポートを終了することを受け，Flash Playerに代わるWeb標準に準拠した新たな共通基盤および可視化コンポーネントの試作評価を進めている。試作の中では，Flash Playerで実現できていた水準のUIや機能，性能要件をWeb標準技術に移行しても満たせるかどうかを検証し，概ね課題が解決できる見通しが立った。

今後は，試作を通して得られた知見を活かし，2020年末までに可視化コンポーネントとしてガントチャート，リソースグラフの製品化を図る。製品化した可視化コンポーネントは，SynViz S2の新基盤として適用し，SynViz S2の事業拡大に貢献する。SynViz S2に限らず，ガントチャートやリソースグラフをSynシリーズ製品など他のHSE製品にも適用していく。さらに，散布図やグラフなどの描画が可能な新たな可視化コンポーネントを開発し，生産計画，需要予測，在庫管理など幅広く製品に提供していくことを検討している。

参考文献

1) Flash & The Future of Interactive Content, https://blogs.adobe.com/conversations/2017/07/adobe-flash-update.html, Accessed 2017/9.
2) Extending User Control of Flash with Click-to-Run, https://blogs.windows.com/msedgedev/2016/12/14/edge-flash-click-run/#TAWwIWpwTXPYOCih.97, Accessed 2017/9.
3) The Chromium Projects Flash Roadmap, https://sites.google.com/a/chromium.org/dev/flash-roadmap, Accessed 2017/9.
4) 山本，他, "HTML5 Canvas および SVG における自動選択アルゴリズムを用いた描画パフォーマンスの最適化",オーディオビジュアル複合情報処理（AVM）2014-AVM-84(6), pp.1-2, 2014.
5) SVG と Canvas: どちらを選ぶか, https://msdn.microsoft.com/ja-jp/library/gg193983(v=vs.85).aspx, Accessed 2017/9.
6) Flash・Unity・openFrameworks の計算速度比較, http://b.i-tach.com/?p=835, Accessed 2017/9.
7) 内海，他, Web 標準技術による Web UI コンポーネントの実現に向けた描画性能の見積による実現可能性の検討，第 16 回情報科学技術フォーラム講演論文集.
8) ヤコブ・ニールセン，"ユーザビリティエンジニアリング原論"，東京電機大学出版局，p.107, 2002.

内海　宏律　　2006 年入社
ビジネスインキュベーション部
研究開発グループ
Web システムおよびソフトウェアコンポーネントに関する研究開発
hironori.utsumi.zc@hitachi-solutions.com

太齋　真吾　　2002 年度入社
第一パッケージ開発部
第一グループ
自社パッケージ製品開発
shingo.dasai.dc@hitachi-solutions.com

石倉　直弥　　2009 年度入社
ビジネスインキュベーション部
研究開発グループ
Web システムおよびソフトウェアコンポーネントに関する研究開発
naoya.ishikura.kk@hitachi-solutions.com