情 報 学 基 礎　66－10
デジタル・ドキュメント 32－10
（２００２．３．１５）

SVG-DOM によるアニメーションと XHTML 中心複合文書の可能性

大坂哲司*1　野村直之*2
*1 フジミック　*2 法政大学エクステンションカレッジ
＊共にＸＭＬコンソーシアム基盤技術部会兼務

概要： XML 文書は XSL&XSLT で変換することでスタイルを持ったドキュメントとして可視化できる。本稿では、HTML-DOM、XML-DOM、SVG-DOM を積極的に制御し、ダイナミックでかつインタラクティブな複合文書の実装方法について報告を行う。SVG-DOM を制御することで、SVG のレンダラーのみではできないテキストアニメーション効果の実装について検討した。この中で利用した DOM 操作技術を応用して、SVG ファイルのエンベッド処理による複合文書生成並びに XHTML 中の名前空間で定義された SVG データから複合文書生成の実装を検討した。現状のブラウザだけでは十分な複合文書生成ができないが、ブラウザ上で複合文書生成のためスクリプトによる実装を検討し、その中で DOM 制御の有効性を見出した。

Animation using SVG-DOM and XHTML Compound Documents

Tetsushi OSAKA *1　Naoyuki NOMURA *2
*1 Fujimic Inc.　*2 Hosei Univ.

Abstract: An XML document usually becomes visible with styles through XSL & XSLT conversions. This paper reports an alternative methodology to create visually rich XML compound documents by manipulating HTML-DOM, XML-DOM and SVG-DOM APIs. SVG-DOM, namely Scalable Vector Graphics Document Object Model, can much more attractively and efficiently control text animation effects than the simple use of SVG documents and their rendering engines. We also implemented a new type of SVG-XHTML compound document that allows multi-modal text manipulation between the XHTML area and the embedded SVG area. Current commercial versions of Web browsers require that this type of compound document embeds within itself JavaScripts with DOM APIs.

1 はじめに

SVG(Scalable Vector Graphics) は XML ベースの２次元ベクターグラフィックスに関する仕様で、2001 年 9 月 5 日 W3C は W3C 勧告として SVG1.0 を公開した[SVG1.0a]。SVG は、W3C 関連仕様と協調するようになっており、CSS や XSL スタイルシート、RDF メタデータ、XML Linking、そして SMIL Animation といった W3C の技術から数多くの恩恵を受け、最新の XML 形式となっている。

SVG 公開に際し、Tim Berners-Lee は次のようなことを述べた[SVG1.0b]。「SVG を用いることによって、Web のグラフィックスは単なる視覚的装飾から、真のグラフィックス情報へと確実に変化します。SVG は、表現力豊かでしかも再利用可能なビジュアルコンテンツを Web 上で実現するキーテクノロジです。今、Web デザイナは Web 上での単なる視覚的装飾に留まらない、検索可能で、再利用可能な Web コンテンツとしても利用可能なプロフェッショナルなグラフィックスを実現する、広く一般に公開された画像形式を手にしたのです。」と述べ、XML ファミリーとして SVG の可搬性やコンテンツ表現力を高く評価している。一方で、マイクロソフトは、”.NET ”戦略の中で、複合文書に対して次のように言っている[ZDNN2000]。記事の中で、”・・・従来ならばワープロの中に表計算を OLE で埋め込むといった方法で作っていた複合文書がすべて XML 化され、一つの XML 文書になる・・・独自の文書形式がなくなり、それを利用するデバイスが自分たちの都合の良い形で内容を表現することが可能になる・・・”と言っている。両者とも Web コンテンツやワープロ文書において、XML がドメインであることを示唆している。次世代のブラウザでは XHTML をベースにしたコンテンツが主流となると言われ、その中でリッチな、インタラクティブな SVG 表現がモジュール化でき、高度な複合文書を表示するブラウザの出現も間近であろう。

このような状況の中で野村らは XML 複合文書について次のようなアプローチを行った[野村 2001]。それによると、Mozilla 並びに Amaya のブラウザにおいて、XHTML がコンテナでその中に SVG を埋め込んだ XHTML を正しく表示することを見出し、今後の複合文書として XHTML 中心複合文書の形態の有効性を示唆した。また、大坂は XML で定義した電子カタログデータを PDF に変換するようなドキュメント生成のワークフローを提案した[大坂 2000a, b]。

SVG は静的なグラフィックス表現に留まらず、機能として**<animate>**要素によるアニメーション機能を有し、容易に２次元グラフィックスのアニメーションが作成できる[SVGZone]。 SVG 仕様によると、<animateMotion>要素の **path** 属性で指定する曲線上に沿って記述したオブジェクトを動かしたり、<textPath>要素が指定する曲線上にテキストの各文字を張り付けたり（各文字が曲線の接線上に描画）、高度な演出が可能となっている。しかし、曲線上に張り付いたテキストを曲線に沿って移動させるような機能はなく、このテキストアニメーション演出実装のために JavaScript による SVG-DOM 操作を行った。ここで得られた DOM 操作技術が発端となり、複合文書実装へとつながった。

本稿では現状のブラウザで利用可能な SVGViewer をプラグインし、スクリプトによるアニメーション実装、下記で述べる複合文書の実装を行った。開発・作動環境は、ブラウザとしてインターネットエクスプローラ(IE5.0 以上)、SVGViewer としてアドビシステムズの SVGViewer2.0[AdobeSVGViewer]を用いた。

以下、まず第２節では、SVG による曲線上へのテキストの表示並びにテキストアニメーションの特徴について述べると共に、SVG-DOM を操作し SVG にないテキストアニメーション効果の実装方法を述べる。またこの中で用いた汎用的に利用できる DOM 操作のソースコードについて言及する。そして、第３節では、汎用的な DOM 操作を利用した複合文書生成について考察する。複合文書の形態を考察し、SVG ファイルのエンベッドによる複合文書生成並びに XHTML 中の SVG データによる複合文書生成による２つの実装方法の特徴や問題点について述べる。第４節では、本方式、ならびに一般の XML 複合文書が次世代 B2B、B2C で利用される展望について言及する。

2 曲線上をうねうねと動くテキストアニメーションの実装方法

SVG の図形描画の各要素は、その中で定義する<animate>要素によってアニメーション効果を演出することができる。<animate>要素以外に、動きに対象とした<animateMotion>要素、色を対象した<animateColor>要素、移動・変形を対象とした<animateTranformation>要素が利用でき、多彩な演出が可能となっている。これらの要素を利用して、曲線上をうねうねと動くテキストアニメーションを検討した。

まず、図１に示すソースでテキストアニメーションを試みた。このソースは“XML コンソーシアム”と言う文字列が<animateMotion>要素で指定するパス上を 6 秒間で移動し、その表示を無限回繰り返す指定となっている。パスは、開始点(1000, 320)から負の水平方向に 2000 まで移動、のように左から右への水平移動を指定している。このソースの実行結果は、我々の期待した通りの演出が見られた。

```
<text style="font-family:'MS Gothic'; font-size:60; fill:blue">
        <tspan style="fill:green">XML</tspan>コンソーシアム
        <animateMotion dur="6" repeatCount="indefinite" rotate="auto"
                path="M 1000 320 h-2000" />
</text>
```

図1　左から右にロールするテキスト

これを応用してパスを直線から曲線に変えたところ（図２）、ソースの実行結果は図３のようになった。結果を分析すると、テキスト中の先頭文字は指定の曲線の接線上にあり、この先頭文字のみが曲線に張り付いてテキストが移動するため大変奇妙に見え、我々が期待する曲線上をうねうねと動く演出は見られなかった。

```
<text style="font-family:'MS Gothic'; font-size:60; fill:blue">
      <tspan style="fill:green">XML</tspan>コンソーシアム
      <animateMotion dur="6" repeatCount="indefinite" rotate="auto"
            path="M 100 200
                  C 200 100 300 0 400 100
                  C 500 200 600 300 700 200
                  C 800 100 900 100 900 100" />
</text>
```

図2　曲線上をテキストが動く

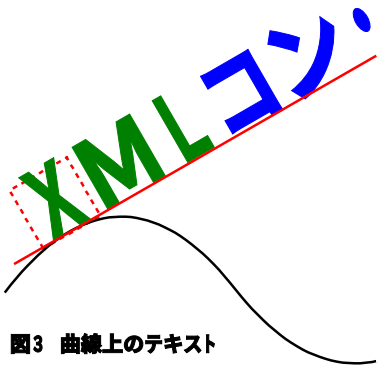


図3　曲線上のテキスト

当初目的のアニメーションは実現しなかったが、ここで重要なヒントを得ることになる。そのヒントを基づいて図４のテキストアニメーション用の<osk_animateText>要素を作成した。この要素は図３で記述されている<text>要素と<animateMotion>要素を合体したものと、文字毎の表示間隔時間を指定する internval 属性を加え、一つの要素で適すとアニメーションができるように設定した。この要素の解釈は図５の通りである。まずテキスト中の文字毎にそれぞれを非表示で描画する。次に<set>要素で文字毎に表示する時間、<animateMotion>要素でアニメーション開始時間を合わせることで、図６のように一文字毎に順次曲線の上をうねうね動かすことができた。

図4　テキストアニメーション要素

```
<osk_animateText id="animate" dur="6" interval="0.4"
        style="font-family:'MS Gothic'; font-size:60; fill:blue; visibility:hidden"
        path="M 100 200
              C 200 100 300 0 400 100
              C 500 200 600 300 700 200
              C 800 100 900 100 1000 200">XMLコンソーシアム
</osk_animateText>
```

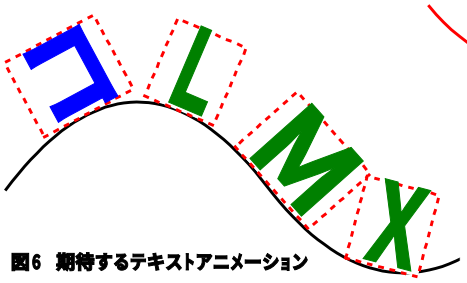


図6　期待するテキストアニメーション

```
<text style="font-family:'MS Gothic'; font-size:60; fill:blue; visibility:hidden">
        X
        <set attributeName="visibility" attributeType="CSS"
             to="visible" begin="0" dur="3" fill="freeze" />
        <animateMotion begin="0" dur="6" repeatCount="indefinite"
<text style="font-family:'MS Gothic'; font-size:60; fill:blue; visibility:hidden">
        M
        <set attributeName="visibility" attributeType="CSS"
             to="visible" begin="0.4" dur="3" fill="freeze" />
        <animateMotion begin="0.4" dur="6" repeatCount="indefinite"
<text style="font-family:'MS Gothic'; font-size:60; fill:blue; visibility:hidden">
        L
        <set attributeName="visibility" attributeType="CSS"
             to="visible" begin="0.8" dur="3" fill="freeze" />
        <animateMotion begin="0.8" dur="6" repeatCount="indefinite"
             path="M 100 200
             C 200 100 300 0 400 100
             C 500 200 600 300 700 200
             C 800 100 900 100 1000 200" rotate="auto" />
</text>
```

図5　テキストアニメーション要素の展開

文字列に対していつも手作業で図５のような解釈をおこなっているのでは非効率である。そこで、<osk_animateText>要素によって、指定のテキストがパス上を動くようなスクリプトの実装を行った。図７の関数 drawText_onCurve は<osk_animateText>要素を解釈し、図５に示した SVG 要素の生成を行う。この関数で要素の追加を行う関数 addTextNode が指定したノードへ子ノードの追加を行うもので、汎用的に利用できる関数となっている[KevLinDev]、[Microsoft]。この関数で対象とする DOM を引数 docs で指定するが、この docs に document.documentElement を指定すると HTML や XHTML の DOM に対して、getSVGDocument とすれば SVG の DOM にノード追加ができる。

```
function drawText_onCurve( txt) {
    for (var i=0; i < txt.length; i++) {
        var iv=interval * i;
        var oNode=addTextNode("text", "","", txt.substring(i, i+1), root,svgDoc);
        oNode.setAttribute("style", style);
        var sNode=addTextNode("set", "", "","", oNode,svgDoc);
            sNode.setAttribute("attributeName", "visibility");
            sNode.setAttribute("attributeType", "CSS");
            sNode.setAttribute("to", "visible");
            sNode.setAttribute("fill", "freeze");
            sNode.setAttribute("dur", dur);
            sNode.setAttribute("begin", iv);
        var aNode=addTextNode("animateMotion", "","", "", oNode,svgDoc);
            aNode.setAttribute("repeatCount", "indefinite");
            aNode.setAttribute("rotate", "auto");
            aNode.setAttribute("path", path);
            aNode.setAttribute("dur", dur);
            aNode.setAttribute("begin", iv);
    }
}
```

図7　osk_animateText要素から一文字毎にSVG要素を生成する関数

```
//タグtagNameで、attrNameでattrValueの属性を持つtextの要素ノードを生成し、
//oNodeに追加する。docsは対象とするDOMを指定する。
function addTextNode(tagName,attrName,attrValue,text,oNode,docs){
//tagNameの指定があれば、その要素を生成しoNodeに追加する。
    if(tagName){
        var oNewElement=docs.createElement(tagName);
//attrNameの指定があれば、attrValueの属性を設定する。
        if(attrName) {
            if(attrName.indexOf("xlink:")==0)//名前空間属性の場合
                oNewElement.setAttributeNS(uri,attrName,attrValue);
            else
                oNewElement.setAttribute(attrName,attrValue);
        }
//テキストの指定があれば、oNewElementにテキストオブジェクトを生成する。
        if(text){
            var oNewText=docs.createTextNode(text);
                oNewElement.appendChild(oNewText);
        }
        oNode.appendChild(oNewElement);
    }
//tagNameの指定がなければ、oNodeにテキストオブジェクトを生成する。
    else {
        if(text){
            var oNewText=docs.createTextNode(text);
                oNode.appendChild(oNewText);
        }
    }
    return oNewElement;
}
```

図8　要素、属性、テキストを生成する汎用関数

HTML でのテキスト表示は、OS が管理する TrueTypeFont から文字生成を行い、また HTML レンダラーは曲線を描く機能がないがため、単純なものしか表示できない。SVG 以前に HTML レンダラーの貧弱さを補うプラグインとして Macromedia Flash がある[Macromedia]。Flash でも文字の回転など種々のグラフィックスを搭載している。SVG と Flash との大きな相違は、SVG が XML そのもので、例えばサーバーサイドでのスクリプトやアプリケーションによってダイナミックの生成できることにある。

３ SVG を含む XHTML を中心とした複合文書の生成について

本稿における ”複合文書 ” の定義は、ブラウザ上で表示されている XHTML(HTML)文書に対して、XLink で言うところのローカルリソースやリモートリソースの一部または全部を、表示中の文書に埋め込んで一つの文書とした表示形態とする。対象となるリソースとして、画像、映像、音声などのマルチメディアデータ、種々の形式（XML 文書、HTML 文書、プレーンテキストなど）のテキストデータなどがある。本稿では XML で記述された文書の中でも、XML 文書でありながら XHTML(HTML)とは表示機構が異なる SVG 文書を対象とする。

本稿の ” 複合文書 ” の生成手段として、onRequest（マウスイベントなど）処理によってインタラクティブに文書を生成できる文書生成手段を対象とする。これ以外の複合文書生成手段として、onLoad 処理による文書生成がある。この形態の複合文書は静的に定められているものであり、サーバー上での文書生成処理できるため、本稿の対象とない。また DynamicHTML のように表示モードを操作して、文書を出現させたり、隠したりしてできあがる文書も対象とない。

複合文書の実施例として、次の形態について実装を検討する。

ケース１．安易に、window.document.open("image/svg-xml"); による文書生成

ケース２．SVG ファイルの embed 処理による複合文書の生成

ケース３．XHTML 中の名前空間で定義された SVG データから複合文書の生成

ケース１では、window.document.open("image/svg-xml"); による文書生成を試みる。期待する動きとして、新しいウィンドウ内に SVG で記述した図形描画が表示されることである。しかし結果は、ウィンドウ内に SVG ソースがテキスト表示されるのみで、SVG 描画には至らなかった。この原因として、ドキュメントオープン時に指定する MIME タイプの認識が無視され ” text/html ” と判断されていること、SVG レンダラーの起動ができなかったことによるものと考えられる。

ケース２において、前述の関数 addTextNode を用いた関数 embedSVG によって、SVG ファイルを表示中の HTML 文中に埋め込んだ[大坂 2001]。この関数では HTML-DOM に対して、その中にあるノード node の下に embed 要素を追加する。埋め込まれるファイルは src で指定され、エンベッド空間の大きさは w×h とする。埋め込みは HTML 文だけでもできるが、インタラクティブに SVG の埋め込みを行った。ここで一つの問題が挙がった。それは SVG ファイルの中で定義されている SVG 空間とこの関数で指定するエンベッド空間の大きさの整合性が取れなくなることにある。整合性を取るためには SVG ファイル中で定義されている大きさに基づいてエンベッド空間を設定するしかない。

```
function  embedSVG(src,id,w,h,node,document){
    var oNode=addTextNode("embed","src",src,"",node,document);
        oNode.setAttribute("id",id);
        oNode.setAttribute("width",w);
        oNode.setAttribute("height",h);
        oNode.setAttribute("type","image/svg-xml");
}
```

図9　addTextNode関数を利用して、embed要素を生成する

ケース３はケース２の問題点を解決すると共に、XHTML[XHTML1.0]の中に名前空間で浮かぶ SVG の島を、本来の形で行う複合文書表示処理を、スクリプトによって実現することを試みる。複合文書生成のシナリオは、エンベッド空間を定義し、その中に SVG 文書の構造に従って svg 要素配下の要素を追加すればできる筈である。しかしながらこのシナリオを実施すると、単に真っ白なエンベッド空間が表示されるだけで、期待する SVG は表示されなかった。その大きな原因は SVG レンダラーの起動ができていないことによるものと

想像する。この現象を解決するため、図 10 に示す方法により、SVG レンダラーの起動並びに SVG 表示を行った。その詳細なプロセスは次の通りである。

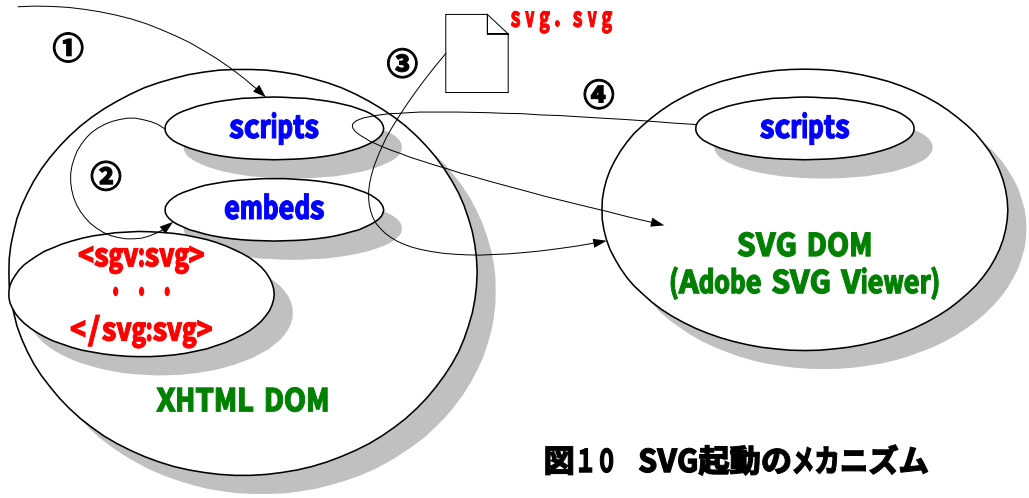


図１０　SVG起動のメカニズム

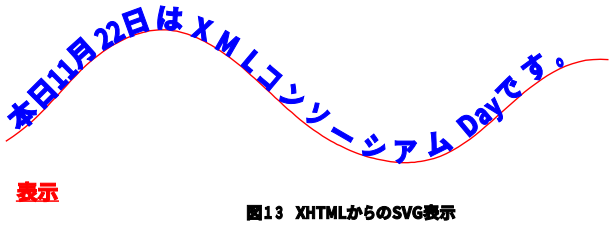
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN" "http://www.w3.org/TR/
2000/CR-SVG-20001102/DTD/svg-20001102.dtd">

<svg id="svg" onload="dispSVG(evt);" xmlns:xlink="http://www.w3.org/1999/xlink">
</svg>
```

図１１　svg.svgファイルの中身

①画面上のアクションポイントをクリックすると、SVG を埋め込む処理が実行される。②イベントが生じたターゲットにある id 情報を取得し、これを ID とする。この ID を持つ要素をターゲット要素し、この要素の中に SVG を埋め込む。ターゲット要素内に要素があれば、それらの要素を削除する。埋め込みスクリプトは、この ID に基づいて、XHTML-DOM から対応する SVG データを読み込む。読み込んだ SVG データは文字列 txt に格納する。文字列 txt 中から ID を持つ svg:svg 要素全体を文字列 doc として抽出する。この中にある svg:svg 要素中から width, height 属性の値（それぞれ w, h）を求める。この大きさを有する embed 要素をターゲット要素内に埋め込み、エンベッド空間を生成する。ここで真っ白な空間がターゲット要素の下に表示される。③embed 要素に指定した SVG ファイルが自動的に読み込まれる。このファイルは SVG レンダラーを起動するためのもので、svg 要素以外に何もない SVG ファイルである。但し、svg 要素の属性として、onLoad 関数があり、SVG ファイル読込時に実行する関数を記述する。④SVG ファイル読込終了後、直ちに onLoad 関数が実行される。onLoad 関数は、先に求めた w, h と、svg:svg 要素の中から width, height 属性以外の属性を求め、それらを svg 要素の属性として追加し、SVG 空間を定義する。次に文字列 doc を元に SVG 文書の階層構造を分析し、その構造を svg:svg 要素配下の要素を svg 要素に追加して SVG-DOM 構造を構築する。この結果、XHTML 内に SVG がインライン表示される。svg:svg 配下の要素名は名前空間を使用しているため ”svg: ”から始まり、この文字列を削除した要素名が追加する要素になる。例えば要素名が ” svg:text ” である場合、” svg: ” を削除し、” text ” を追加する SVG の要素名とした。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml11/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja" xmlns:svg="http://
www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<head>
    <script type="text/javascript" src="html_svg.js"></script>
</head>
<body id="body" background="backgrnd.gif">
<p style="font-size:24; color:green; font-weight:bold">ＸＨＴＭＬの中にＳＶＧを表示します。</p>
<svg:svg id="svg2" width="500" height="150" viewBox="0 0 1000 300">
    <svg:defs>
        <svg:path id="MyPath" d="M 100 200 C 200 100 300 0 400 100 C 500 200 600 300 700
200 C 800 100 900 100 900 100" />
    </svg:defs>
    <svg:use xlink:href="#MyPath" style="fill:none; stroke:red" />
    <svg:text style="font-family:'MS Gothic'; font-size:42; fill:blue">
        <svg:textPath xlink:href="#MyPath">
            本日11月22日はＸＭＬコンソーシアムDayです。
        </svg:textPath>
    </svg:text>
</svg:svg>
<a id="svg2_" onmousedown="dupXML()">表示</a>
</body>
</html>
```

図12　XHTMLのソース



表示

図13　XHTMLからのSVG表示

このようにしてXHTML中のSVG（図12、図13）が表示できた（即ち、XHTML上にスクリプトを駆使して複合文書の実装ができた）。またケース２で生じた不具合に対して、エンベッド空間とSVG空間の大きさを一致させることができた。しかし今回の環境ではいくつかの制約がある。①SVGはHTML文中の順番に関係なく、最前面に表示される。②ブラウザ上で複数のSVG空間を設けた場合、最初に定義したSVG空間に最後のSVGデータが表示され、他のSVG空間は真っ白のままである。この現象は印刷時にも見られる。これらの現象はIEの制限によるものなのか、SVGViewerとの相性なのか、現在調査中である。

４　考察と今後の展望

本稿では、DOMを制御することによりSVGのみでは得られないテキストアニメーションの実装、このDOM操作技術を用いてXHTMLをコンテナとして、その中に埋め込まれたSVGがインタラクティブに複合文書を生成するメカニズムの実装について検討を行い、それぞれの実装が実現できた。現状で多くの人に利用されているブラウザで、DOM制御技術の有効性を見出した。MozillaやAmayaがネイティブでSVGをサポートしているように、IEやネットスケープも同様のサポートを始めることを願うものである。

B2Bにおいて定番商品の売買は、今後Webサービスが導入されるであろう。しかし新製品においては、商品説明のための電子カタログが必要になるし、Webサービス上でそのカタログの流通も必要となる。種々の形式／様式を持つカタログを一つの言語規約に統一することは困難であろうし、製品が持つ豊かな情報を阻害するやも知れない。このような電子カタログにおいて、作成者に自由な表現を任せるには名前空間の利用は必須である。その名前空間の中でSVGが使われることにより、豊かなカタログ作成が実現できる。

参考文献：

[AdobeSVGViewer] ”Adobe SVG Viewer Download Area”、http://www.adobe.com/svg/viewer/install/old.html、4/2001

[KevLinDev] ”advanced SVG Tutorial”、http://www.kevlindev.com/tutorials/

[Macromedia] http://www.macromedia.com/jp/software/flash/

[Microsoft] ”About the W3C Document Object Model”、http://msdn.microsoft.com/workshop/author/dom/domoverview.asp

[SVG1.0a] ”Scalable Vector Graphics (SVG) 1.0 Specification”、http://www.w3.org/TR/2001/REC-SVG-20010904/、9/2001

[SVG1.0b] ”Scalable Vector Graphics (SVG) 1.0 の公開について (W3C 勧告)”、http://www.w3.org/2001/09/svg1-pressrelease.html.ja、9/2001
[SVGZone] ”SVG ゾーン”、http://www.adobe.co.jp/svg/
[XHTML1.0] ”XHTML 1.0: The Extensible HyperText Markup Language”、http://www.w3.org/TR/xhtml1/、6/2000
[ZDNN2000] 「.NET」のポイントは“妥協のないユーザー体験”、http://www.zdnet.co.jp/news/0006/23/ms.html、6/2000
[大坂 2000a] 大坂哲司：”XML を用いたカタログ作成”、ON DEMAND Printing Japan 2000、4/2000
[大坂 2000b] 大坂哲司：”XML/PDF を活用したソリューション”、PDFConference2000、7/2000
[大坂 2001] 大坂哲司、野村直之：”SVG-DOM による豊かなクライアント XML 表現力"、第 1 回 XML コンソーシアム Day、11/2001
[野村 2001] 野村、川口ら：”XML 複合文書の実現方式に関する一考察　XHTML，SVG，MathML のボキャブラリ活用によるアプローチの有効性について”、 情報処理学会デジタルドキュメント研究会予稿集、3/2001

開発並びに作動環境：

OS：Windows2000、WindowsNT

Browser：IE5.0、IE5.5

SVGViewer：AdobeSVGViewer2.0

AdobeSVGViewer3.0 は機能的に優れているが、SVG 上のスクリプトから HTML 要素の操作ができなかったため、AdobeSVGViewer2.0 を使用する。

作成ツール：テキストエディタ(秀丸)