

# OOXML に求められるフォント情報交換規定

鈴木 俊哉†

† 広島大学総合科学研究科

〈あらまし〉 ISO/IEC JTC1/SC34 は XML を基盤とするオフィス文書形式として ISO/IEC 26300:2006 (ODF 1.0) と ISO/IEC 29500 (OOXML) の 2 つを国際標準として採択した。ODF の既存実装にはフォント埋め込みをサポートするものはないが、ODF は W3C により策定された様々な Markup Language を多数引用しているため、フォント参照について独自の仕様を策定することはないと予想される。これに対し、OOXML は今回の標準化以前にはデータ形式が非公開であった Microsoft Office との相互運用性を非常に重視するものである。Microsoft Office のデータ形式および実装の一部は様々な形でのフォント参照およびフォント埋め込み機能を持つが、オフィス文書中のフォント資源に関しては OOXML 標準の中で十分に記述されているとは言い難い。本稿では OOXML を構成する規格のフォント関連の機能について整理し、今後のオフィス文書形式に求められる機能について提案する。

キーワード：フォント、フォント情報交換、オフィス文書形式、OOXML

〈Summary〉 ISO/IEC JTC1/SC34 has approved two XML-based data formats for the office documents; ISO/IEC 26300:2006 (ODF) and ISO/IEC 29500 (OOXML). The ODF is rather new data format which refers various existing markup languages standardized by W3C. The existing implementation of ODF does not support the embedding of the font resource in the document data. The feature related to the font resource in ODF would follow future efforts by W3C. On the other hand, OOXML is focused to the interoperability with existing data formats of Microsoft Office, whose data format had never disclosed before the standardization of OOXML. Existing implementations of Microsoft Office document generators and consumer had various methods to refer, substitute and embed the font resources in the documents. From the point of view to interchange the font resource among the existing Microsoft Office documents and OOXML, the description about the font resource in OOXML standard is insufficient. In this report, we summarize the features of font information interchange in OOXML, and propose the required features in future office document formats.

**Key words:** Font, Font information interchange, office document format, OOXML

## 1. 背景

### 1.1 工業規格における文書の種別

文書の電子化データの規格は、そのデータの用途によって様々な分類される。印刷用のデータについては、そのワークフローを、データが頻りに変更される作成・校正段階と、変更されないデータによって出力結果を保証する最終出力段階に区分し、おおまかに編集可能形式文書や最終形式文書に分類される。これに対し、オフィス文書はその用途もワークフローも様々であり、規格が目標とする要件を具体的に定めることはむずかしい。実際、JIS X 0001 ではテキスト、DTP、ワードプロセッシングなどは定義済みであるが、オフィス文書・事務文書などの用語は定義されていない。

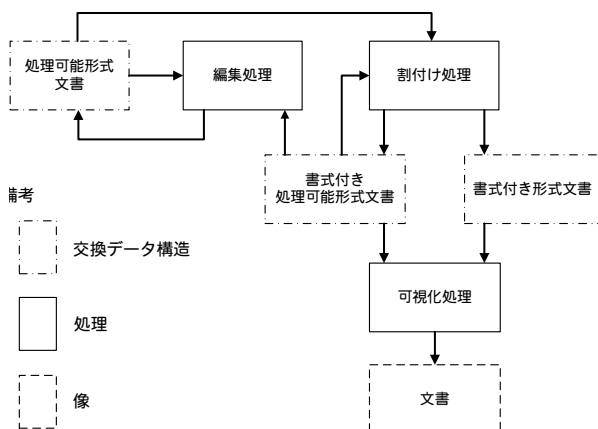


図 1 JIS X 4101(ISO/IEC 8613-1) で定義される文書処理フロー

ITU-T は 1980 年代から SGML を基盤とする文書交換体系として開放型文書体系 (ODA) と呼ばれる規格群を策定し<sup>8)~15)</sup>、異なる処理系の間で文書を交換する形式 (ODIF) や<sup>12)</sup>、必要とされる要件ごとにデータ形式の標準化に取り組んできた<sup>13)~15)</sup>。ODA においては、図 1 に示すように、文書を書式付き形式、処理可能形式、書式付き処理可能形式に分類し、書式は文書の論理構造を表現するための体裁として扱われる。現在の Web 文書が本文を記述した HTML と体裁を記述した CSS<sup>16)</sup>に分割するように、ODA においても文書本体には体裁の詳細は記述せず、DSSSL<sup>17)</sup>によって体裁を指定する構成となっている。しかし、ODA に基づくオフィス文書システムは広く普及したとは言えない。図 1 に示すように、ODA が基本とするワークフローは特定の文書の書式に依存せず、汎用的な論理構造で文書を作成した後に書式を修正するという手順である。非常に長い文書を複数人で作成し、また、文書の分割・結合などの再利用性を考えている場合には望ましい手順であるが、数ページの文書を個人で作る手順としてはやや大掛かりである。たとえば、JIS 規格における事務用ページプリンタの仕様書様式<sup>18)</sup>での和文事務文書のテストパターンは、10 ポイント明朝体を文字ピッチ 3.8mm、改行 4.2mm で並べた 1 ページの文書であり、これを 600dpi 程度の解像度で数秒~十数秒で出力できる処理系を想定している。1) このような文書を個人が手作業で作成し、少数印刷して配布する (また、文書の引用やデータベース化などは特に行なわない) 用途では、ワードプロセッサのように文書の論理構造とは無関係に最終出力のみを確認しながら文書を作成するような簡易なアプリケーションが望まれる。

1) 写真画像や、数表などを加えたさらに複雑なものとして、印刷可能枚数測定用のテストセット<sup>19)</sup>があるが、これも

- フォント: セリフ・サンセリフ 2 種類、サイズは 6~32 ポイント (1 つのテストセットあたり 2 または 3 種類)
- 画像: CMYK による 500 階調

といったレベルであり、やはり PostScript や PDF などの最終出力用データが提供する表現能力範囲に比較すると簡易なデータといえる。

“The Requirement of Font Information Interchange in OOXML” by and Toshiya SUZUKI, (Information Media Centre, Hiroshima University).

村田が既に指摘しているように、ここ数年のオフィス文書の規格化の動きには、対象とする文書自体の定義やワークフローから定義するのではなく、既存の処理系実装を起源とするものが増えてきている<sup>20),21)</sup>。従って、保証される機能の範囲や、既存標準との関連なども不明確な部分がある。本稿で扱うオフィス文書形式の OOXML や ODF も既存実装を起源とする規格であり、標準化の過程でこのような問題が指摘されてきた。本稿では特にフォント関連の機能について述べる。

## 1.2 オフィス文書形式の標準化動向

電子文書の国際標準を扱う ISO/IEC JTC1/SC34 は 2006 年に XML ベースの新しいオフィス文書形式である ODF を、2008 年に Microsoft Office 実装の互換オフィス文書形式である OOXML<sup>6)</sup>を、それぞれ国際標準 ISO/IEC 26300<sup>1)</sup>、ISO/IEC 29500<sup>2)~5)</sup>として採択した。前節で述べたように、ここ数年で提案されているオフィス文書規格は既存のアプリケーションを起源とするものである。ODF は OpenOffice.org 2.0 から導入された XML 形式のデータであり、OOXML は Microsoft Office 2003 から導入された Microsoft Word, Microsoft Excel 用の XML 形式のデータを発展させている。提案されているデータ形式は標準的な XML であり、特に ODF は W3C によって策定された様々な XML 規格を引用しているが、起源となっているアプリケーションはそれらの XML 処理系として設計されたものではなく、内部データをそれらの XML 規格に従って表現しようとしたものである。たとえば、文書レイアウト機能について、SGML 体系では DSSSL<sup>17)</sup>、XML 体系では CSS<sup>16)</sup>や XSLT-FO<sup>22)</sup>が標準化されている。ODF は CSS と XSLT-FO を引用しているが、それらの処理系があれば ODF を OpenOffice.org と同じようにレイアウトできるわけではない。また、これらのレイアウトを規定する SGML や XML 規格は、文書の論理マークアップをもとにレイアウトを行なうが、アプリケーションの内部では必ずしも論理構造とレイアウトを分離していないため、内部情報を無矛盾に表現できているかどうかにも疑問がある。

ODF, OOXML とも、その起源となったアプリケーションは、文書の論理構造ではなく、レイアウト結果を提示しながら文書を作成するのに用いられる。従って、これらの文書データの実際の利用状況ではレイアウト結果の保証というものが非常に重要である。レイアウトを保証するデータ形式としては、SPDL や PDF といった最終出力用のデータ形式が既に標準化されている。これらは標準準拠の処理系で各データがどのように出力されるべきかを詳細に規定している。しかし、ODF や OOXML は、どちらもあくまでも提案された文書形式の規格であって、既存のアプリケーションとの出力結果の互換性は保証していない。たとえば、ODF は、OpenOffice.org の実装が記述されている XML のタグをどのように処理するかはまったく記述していない。OOXML は既存の処理系がどのように処理するかの情報を含むが、参考情報であって処理そのものを完全には規定しない。今年、ISO/IEC JTC1/SC34 は OOXML とその他の文書データとの相互運用・相互変換の標準化を扱うワーキンググループとして JTC1/SC34/WG5 を立ち上げたが、現時点で対象となっているのは文書データを相互変換した際の情報損失や誤変換であり、処理系での出力結果を扱っているわけではないことに注意すべきである。

本稿では、文字文書の出力結果を保証するための最も基礎的な資源であるフォントについて、OOXML およびその関連規格で定義される機能を整理し、現状不足している機能とその影響について述べる。

## 2. OOXML のフォント関連機能

本章では OOXML で定義されるフォント関連機能を整理する。ただし、本稿執筆時点では ISO/IEC 29500 は出版前であるため、本稿

は ISO/IEC 29500 の元となった ECMA-376 を参照している<sup>6)</sup>。また、Microsoft Office 2007 のような「OOXML の出力もできる処理系」について述べているのではないことに注意する。

### 2.1 OOXML 規格の構成

OOXML 規格の元となった ECMA-376 は 5 部からなる大部の規格である。その構成は以下のようになっている。

1. 基本原理
2. Open Package コンテナ構造
3. 入門 (この第 3 部は全体が参考情報で規格本体に含まれない)
4. OOXML の XML タグ解説
  - (a) WordprocessingML
  - (b) SpreadsheetML
  - (c) PresentationML
  - (d) DrawingML
  - (e) VML
  - (f) SharedML
5. 既存実装との互換性を維持するための拡張方法

Microsoft Office 2003 での Word XML, Excel XML は非圧縮の XML を 1 個だけ交換していたが<sup>2)</sup>、OOXML では文書が文書本体、スタイルシート、画像等マルチメディアデータ等に細かく分割され、それを ZIP ファイルに似た形式でアーカイブ化および圧縮して交換する。これを Open Package と呼んでいる。

また、Word XML と Excel XML のようにアプリケーション別の XML が定義されていたように、OOXML においても基本的にはアプリケーション個別に XML が定義されている。WordprocessingML は Microsoft Word, SpreadsheetML は Microsoft Excel, PresentationML は Microsoft PowerPoint などのアプリケーションからの出力に由来するが、共通度の高いものは SharedML としてくりだされている。

また、DrawingML は WordprocessingML, SpreadsheetML, PresentationML の中で用いられる描画オブジェクトで、いわゆる「Office 描画オブジェクト」の OOXML 表現と言える。VML はもともと W3C が XML によるベクタ画像表現方式を策定する際に SVG の対抗規格として Microsoft が提案したもので、その適用範囲は DrawingML と重なる。しかし、OOXML 規格の中では VML はレガシ扱いであり、DrawingML への移行が推奨されている。

### 2.2 Open Package 規格におけるフォント機能

Open Package 規格がフォントを格納できることは明示的に書かれている。

### 2.3 WordprocessingML におけるフォント機能

WordprocessingML における XML 要素がどのようにフォントを参照しているかを図 2 に示す。

OpenPackage 内にはテーマ、文書、フォントテーブル、フォントが個別のファイルで格納されている。テーマにおいては、本文書体と見出し書体を以下の 4 つの文字集合ごとに個別に指定できる。

- ASCII (ascii): ISO/IEC 2022 の C0 および G0 領域のコードポイント群を表示するのに用いられるフォント。
- East Asian (eastAsia): Unicode における東アジアの文字に対応するコードポイント群を表示するのに用いられるフォント (「東アジア」の定義がはっきりしないが、おそらく漢字、結合済みハングルのことと思われる)。
- Complex Script (cs): Unicode 描画処理において複雑な言語サポートを要求する文字群を表示するのに用いられるフォント。
- High ANSI (hAnsi): 上記 3 分類のどれにもあてはまらない

2)画像なども XML 中に巨大な要素として理め込まれていた

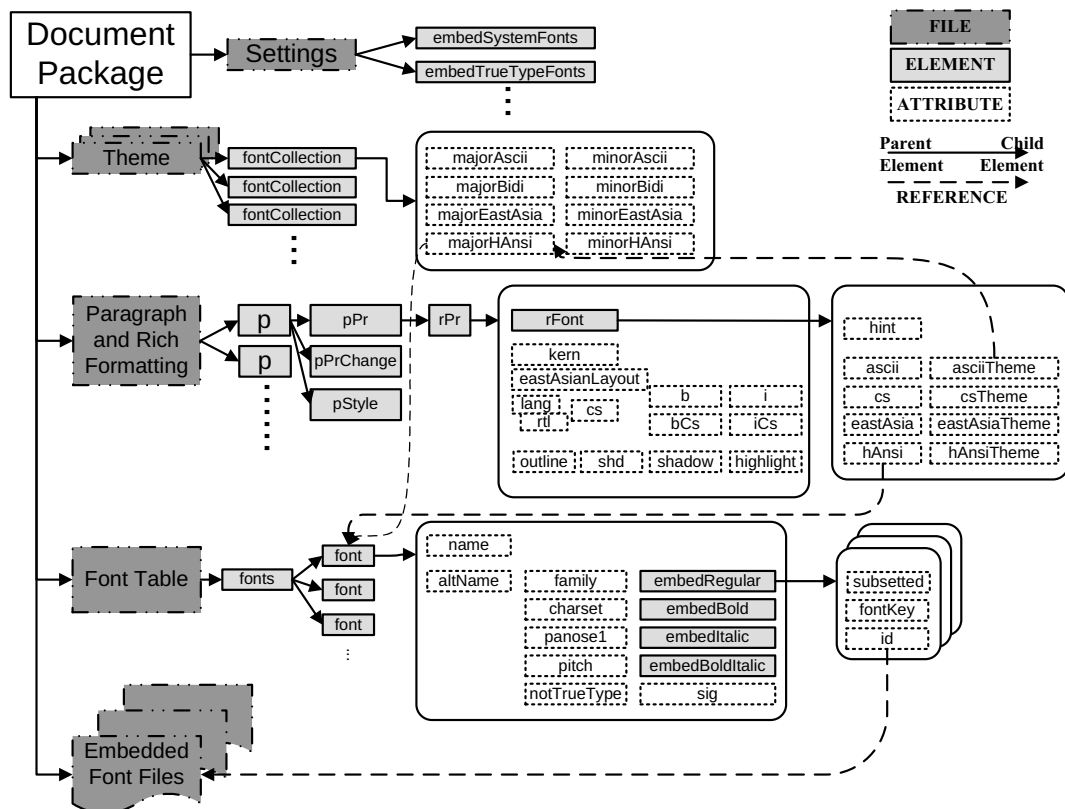


図2 Font-related XML elements in WordprocessingML

コードポイント群を表示するのに用いられるフォント。<sup>3)</sup>たとえば、ASCII 文字の見出し書体を majorAscii, ASCII 文字の本文書体を minorAscii に割り当てる。ただし、WordprocessingML の中ではテーマについては規定されておらず、どのように利用できるのか不明確である。

文書は複数のパラグラフ (p) で構成されており、各パラグラフは書式などを設定するためのパラグラフ属性 (pPr) を持つ。OOXML では一定の書式を与えられた文字列をまとまりを、「ラン (run)」という単位で呼ぶ。パラグラフ属性の中で、ランに対する詳細な書式を設定する。グリフに関するグラフィック処理 (アウトライン、シェーディング (shd)、シャドウ化、ハイライト化) などもあるが、この段階で日中韓レイアウト、双方向レイアウト (英語とアラビア語の混植など)、カーニング、言語指定など細かい機能の指定も行なわれる。各ランに対するフォントは、テーマと同様に 4 つに分類された文字種ごとに特定のファミリ名、またはテーマで定義されるフォントの参照という形でフォントを指定できる。種別が不明確な文字の表示は hint 属性によって 4 分類のうちどれに誘導するかを指示できる<sup>3)</sup> [2.9.3]。

テーマおよび文書本体の中ではフォントはファミリ名で参照された。このファミリ名は文書内で用いられるものであって、処理系の持つフォント資源のファミリ名と一致する必要はない。この一致をとるための仕組みがフォントテーブルである<sup>3)</sup>[2.9.4]。たとえば、Microsoft Office ではシステムにインストールされているフォントの他にも、プリンタドライバが見せる仮想的なフォント (デバイスフォントなどと呼ばれる) を指定することができる。Microsoft Office 文書は文書作成時に指定したプリンタドライバ情報を含んでいるが、同一のシステム上で文書を再利用する場合には適切なデバイスフォントに辿りつけるが、一般の文書交換では難しい。

Font Table は、適切なフォント資源の選択を補助するために、各ファミリ名に対し以下の情報を格納する<sup>3)</sup>[2.9.5]。

- name: 文書内で参照されるファミリ名
- altName: 処理系がフォントを検索する際に用いて良いファミリ名
- panose1: 書体デザインの Panose 1.0 分類値
- charset: フォントの文字集合 ANSI, Symbol, MacRoman, JIS (JIS X 0208 または CodePage 932 と思われる), Hangul (Wansung と思われる), Johab, GB-2312, Big5, Greek, Turkish, Vietnamese, Hebrew, Arabic, Baltic, Russian, Thai, Eastern European, か、あるいはシステム既定、OEM、未定義といった種別から選択する。
- family: Windows GDI におけるファミリ分類 (Serif, Sans Serif, Symbol 等)
- pitch: Windows GDI におけるピッチ分類
- codepage: フォントが対象とする文字集合の Microsoft Codepage 分類
- Unicode range: フォントが含む Unicode の範囲

Windows GDI はあるフォントから太字、斜体、太字斜体を合成できるため、ファミリ名は基本的にこれらをまとめて定義されている。本質的な基本の書体だけを埋め込み、他の 3 書体は処理系が (Windows GDI 等を用いて) 合成させる場合もあるが、フォントによっては太字や斜体を独立な資源として持つ場合もある。そこで、フォントテーブルでは埋め込みに関して特定のファミリの中で合成の可能性がある embedRegular, embedBold, embedItalic, embedBoldItalic と 4 つの資源を独立に指定できるようにしている。

フォントを埋め込んでいる場合は、フォントがサブセット化されているかどうか (subsetted) と、難読化鍵 (fontKey)、OpenPackage 内の資源 ID (id) が定義されている。OOXML の規定では、フォント

<sup>3)</sup>この名称はもともと ISO/IEC 2022 の C1, G1 領域のコードポイントを指していたと推測されるが、現在の OOXML 規格ではこのように定義されている。

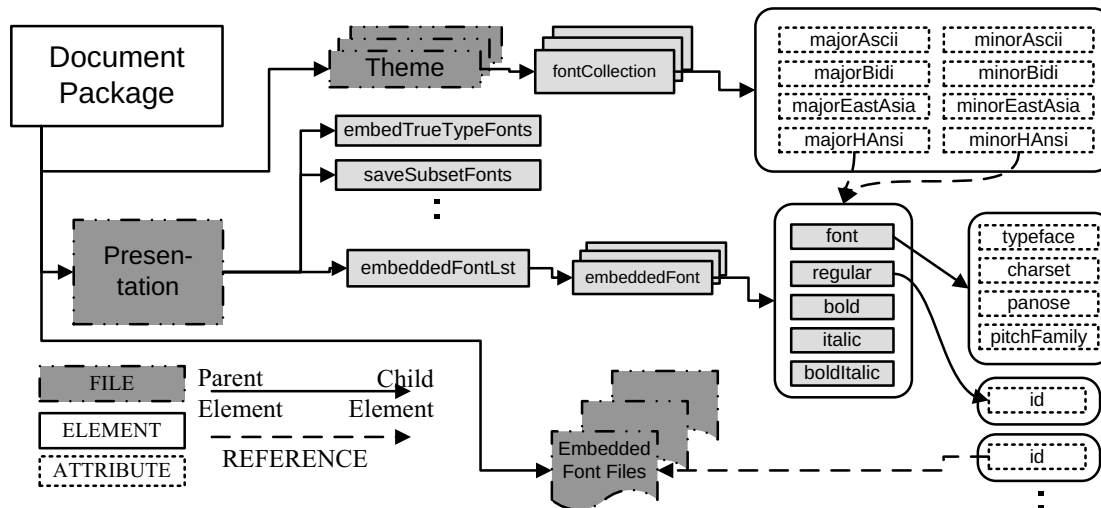


図3 Font-related XML elements in PresentationML

埋め込みは文書の再編集ではなく表示が目的であり、商用フォントを文書から抽出して他の文書への転用を防ぐため、難読化することが望ましいとされる。

#### 2.4 SpreadsheetML におけるフォント機能

SpreadsheetML でも、セル内の文字列をランと見たててフォントを指定することができるが、WordprocessingML のように文字種別ごとに複数のフォントを指定する仕組みはない。また、フォントテーブルに相当するフォント資源選択を抽象化する仕組みもない。

#### 2.5 PresentationML におけるフォント機能

WordprocessingML では埋め込み・外部参照を問わず、文書中のフォントはフォントテーブルに集約された。PresentationML では別ファイルとしてのフォントテーブルがなく、埋め込みフォントだけを管理する `embeddedFontList` がこれに近い役割を担う。この仕組みを図3に示す。WordprocessingML でのパラグラフに相当するものが図3には無いが、これは描画オブジェクトを DrawingML に分割して定義しているためである。

埋め込みフォントリストも WordprocessingML と同様にファミリー単位で属性を管理しており、各フォントファミリーの一要素には以下の属性を記述する。

- `font`: ファミリー全体の属性
  - `charset`: 文書中で使用された文字に最も近い文字集合
  - `panose`: Panose 1.0 分類値
  - `pitchFamily`: ピッチ分類値
  - `typeface`: フェイス名
- `regular`: ファミリー中の通常の書体の属性
  - `id`: 文書パッケージに格納されるフォントデータの参照情報
- `bold`: ファミリー中の通常の太字書体の属性
  - `id`: 文書パッケージに格納されるフォントデータの参照情報
- `italic`: ファミリー中の通常の斜体の属性
  - `id`: 文書パッケージに格納されるフォントデータの参照情報
- `bolditalic`: ファミリー中の通常の太字斜体の属性
  - `id`: 文書パッケージに格納されるフォントデータの参照情報

文字集合・Panose 値・ピッチ分類値などを書き込むことにより、埋め込みフォントが利用できない場合に適切なフォントへの置換を補助することができる。しかし、WordprocessingML にあった `altName`, `family` に相当するものが無いことは注意しなければならない。また、もともと文書パッケージに埋め込まず、単に参照されるフォントに関しては、WordprocessingML におけるフォントテーブルのよう

な抽象化機構がない。

#### 2.6 DrawingML におけるフォント機能

DrawingML は WordprocessingML, SpreadsheetML, PresentationML の中で使用されることを前提に設計されているため、DrawingML 自体ではフォント埋め込みやフォント参照の抽象化の仕組みを持たない。フェイス名による外部参照のみ可能である。DrawingML 中でのフォント資源参照を図4に示す。

DrawingML は文字列描画について短い文字列の描画と、ある程度の長さのテキストの描画のための2つを想定し、前者に対する書式を Style、後者に対する書式を Paragraphs and Rich Formatting と個別に定義している。また、前者の派生として表のセル内の文字列描画の Style として `tcTextStyle` がある。

Style においてはフォントは `fontScheme` という単位で参照される。`fontScheme` は WordprocessingML におけるテーマと同様に本文書体 (`minorFont`) と見出し書体 (`majorFont`) の対を定義する。ただし、それを構成する要素として用字系ごとのフォントを (フェイス名によって) 列挙し、それによって明示的に指定されていないものをラテン用字系 (`latin`)、CJK 用字系 (`ea`)、高度言語サポート (`cs`) の3分類にわりあてる。`tcTextStyle` は (セル内では見出し・本文の区別の必要がないため) これらの要素を直接指定する。Paragraphs and Rich Formatting の場合、この他にシンボル用フォント (`sym`)、簡条書き記号フォント (`buFont`) などを追加することができる。

従って、DrawingML におけるフォント参照は最終的には以下の属性によって参照される。

- 明示的に指定されたもの: `typeface`, `script`
  - フォールバックによるもの: `typeface`, `charset`, `panose`, `pitchFamily`
- によって参照

#### 2.7 VML におけるフォント機能

W3C への提案という性格上、フォント資源の扱いも OOXML の他のパートとは異なり、CSS との連携を強く意識したものになっている。DrawingML は Microsoft Visio 2003 の XML 出力がベースと考えられるが、OOXML における DrawingML は単体のオーサリングソフトも表示系も存在しない。これに対し VML は IE5 以降で表示が可能であり、現時点では DrawingML より適用範囲が広いと言える。フォント参照は CSS2(CSS2.1 ではない) に準ずるとされているが、CSS で具体的なフォント資源を参照するための `@font-face` 属

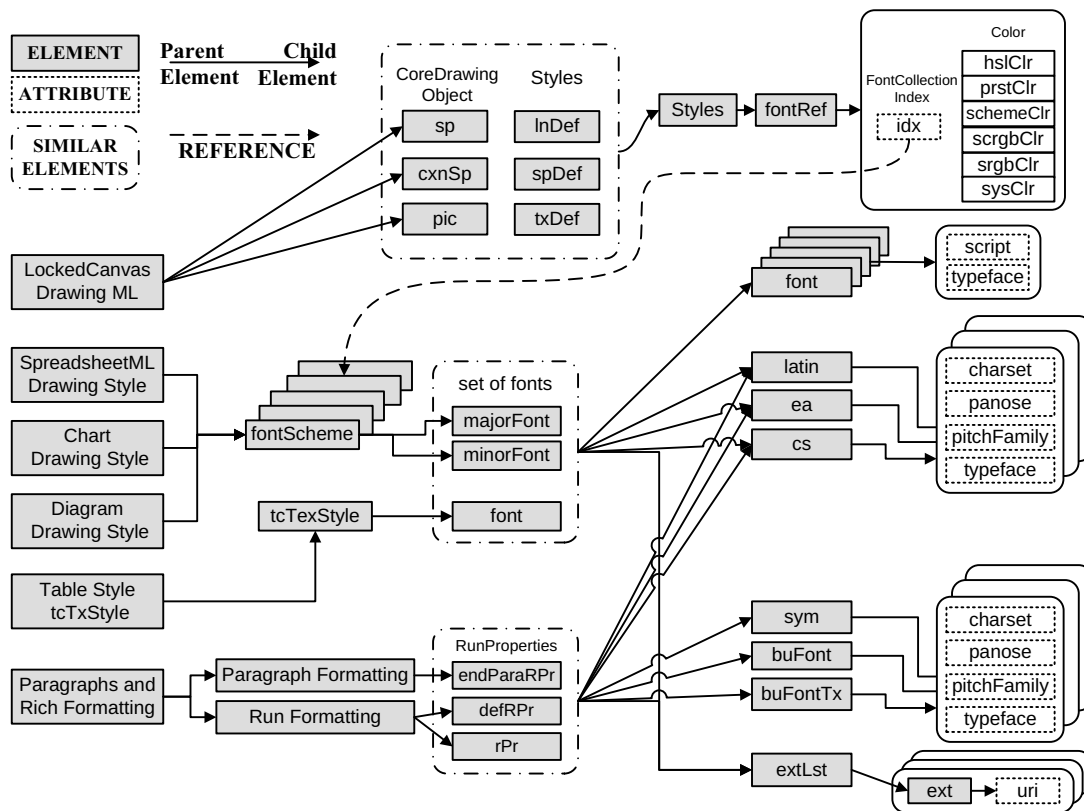


図 4 Font-related XML elements in DrawingML

性に相当するものは VML では定義されない。

### 3. まとめ

#### 3.1 OOXML のフォント資源管理の問題点

前章で示したように、OOXML は WordprocessingML、SpreadsheetML、PresentationML の間でフォント参照の仕組みがかなり異なる。WordprocessingML がもっとも抽象化した後に処理系に渡すのに対し、SpreadsheetML にはフォント埋め込みの機能が定義されておらず、単にフェイス名を処理系に渡すため、代替処理のための十分な情報を渡すことが難しい。また、PresentationML には抽象化の仕組みがあるが、WordprocessingML に比較すると以下の点で劣る。

- 埋め込みフォントに関する情報を抽出するのに PresentationML 全体をパースする必要がある。WordprocessingML では独立したファイルとして格納されているフォントテーブルだけをパースすれば良い。
- 埋め込みフォントと単なる外部参照でデータ構造の対称性が低い。埋め込まれているフォントを削除したり、事後的に埋め込むことが難しい。
- embeddedFont はフォントテーブルのエントリに比べて属性情報が不足している。
  - altName 属性のように別名宣言ができない。
  - notTrueType 属性のように出力デバイス依存のフォントを宣言することができない。
- 埋め込みフォントのそれぞれについてサブセット化されているかどうか記述する部分がない。
- 埋め込みフォントを WordprocessingML と同じ仕組みで難読化できない (fontKey 属性がないため)。

各 Markup Language 間でこれらのレベルを合わせる必要がある。特に、DrawingML は単体ではフォント参照を制御できないため、同一のコンテンツを配置しても WordprocessingML と SpreadsheetML の中では処理結果が異なることがありうる<sup>4)</sup>。

さらに、WordprocessingML においても以下の問題がある。

- フォント選択の補助情報として、コードポイント群 (charset で記述されるもの)、Microsoft Codepage 値、Unicode 部分集合、script 種別、Panose 1.0 分類値、ピッチ分類値などがあるが、相互に重複した情報であり、優先順位がはっきりしない。
- フォントテーブル外のレイアウト指定情報の中に OpenType レイアウト処理と関連するものがあり、フォントテーブルを走査しただけでは適切なフォントを特定するのが難しい。たとえば、フォントが縦書き可能かどうかといった情報はフォントテーブルには含まれない。

また、WordprocessingML はフォント埋め込みの必要性を文書表示、特にメトリックの保存のためと記述しており、さらに商用フォントを埋め込んだ場合の目的外利用を防止するためにサブセット化や難読化を推奨している。しかし、PDF などではメトリックの保存などの目的ではフォントからグリフ描画データを取り除いてメトリックのみを埋め込むことを可能としている。このような手法も検討されるべきであろう。

#### 参考文献

- 1) ISO/IEC JTC1/SC34: "Information technology – Open Document Format for Office Applications (OpenDocument) v1.0", ISO/IEC 26300:2006
- 2) ISO/IEC JTC1/SC34: "Information technology – Document description and processing languages – Office Open XML File Formats – Part 1: Fundamentals and Markup Language

<sup>4)</sup>Microsoft Office の中で、異なるアプリケーション間で Office 描画オブジェクトのコピー・ペーストの際に結果が異なるのは同様の背景があると予想される。

- Reference”, ISO/IEC 29500-1:2008
- 3) ISO/IEC JTC1/SC34: “Information technology – Document description and processing languages – Office Open XML File Formats – Part 2: Open Packaging Conventions”, ISO/IEC 29500-2:2008
  - 4) ISO/IEC JTC1/SC34: “Information technology – Document description and processing languages – Office Open XML File Formats – Part 3: Markup Compatibility and Extensibility”, ISO/IEC 29500-3:2008
  - 5) ISO/IEC JTC1/SC34: “Information technology – Document description and processing languages – Office Open XML File Formats – Part 4: Transitional Migration Features”, ISO/IEC 29500-4:2008
  - 6) ECMA: “Standard ECMA-376 Office Open XML File Formats”, <http://www.ecma-international.org/publications/standards/Ecma-376.htm>.
  - 7) 情報処理用語—基本用語, JIS X 0001-1994
  - 8) “Information processing – Text and office systems – Office Document Architecture (ODA) and interchange format – Part 1: Introduction and general principles”, ISO 8613-1:1989
  - 9) 開放型文書体系 (ODA) 及び交換様式—第 1 部 総則, JIS X 4101-1993
  - 10) 開放型文書体系 (ODA) 及び交換様式—第 2 部 文書構造, JIS X 4102-1993
  - 11) 開放型文書体系 (ODA) 及び交換様式—第 4 部 文書概要, JIS X 4104-1993
  - 12) 開放型文書体系 (ODA) 及び交換様式—第 5 部 開放型文書交換様式 (ODIF), JIS X 4105-1993
  - 13) 開放型文書体系 (ODA) 及び交換様式—第 6 部 文字内容体系, JIS X 4106-1993
  - 14) 開放型文書体系 (ODA) 及び交換様式—第 7 部 ラスタ図形内容体系, JIS X 4107-1993
  - 15) 開放型文書体系 (ODA) 及び交換様式—第 8 部 幾何学図形内容体系, JIS X 4108-1993
  - 16) W3C “Cascading Style Sheets” <http://www.w3.org/Style/CSS>
  - 17) ISO/IEC JTC1/SC34 “Information technology – Processing languages – Document Style Semantics and Specification Language (DSSSL)” ISO/IEC 10179:2006
  - 18) ビジネス機械・情報システム産業協会: 事務機器—ページプリンタの仕様書様式, JIS B 9527:2004
  - 19) ビジネス機械・情報システム産業協会: 事務機械消耗品の印刷可能枚数測定用カラーテストページセット, JIS X 6938:2008
  - 20) 村田真: Open Document Format の標準化について, 画像電子学会第 17 回研究会 2006 年 7 月 10 日
  - 21) Chinese Engineering Standard Institute: “Specification for the Chinese Office File Format”, GB/T 20916-2007
  - 22) W3C “Extensible Stylesheet Language (XSL)”, <http://www.w3.org/TR/xsl/>