SGML文書データベースAstoriaのためのDTD上位互換性チェッカ

村田 真†

SGML文書データベースAstoria上に構築されたDTD上位互換性チェッカについて述べる．上位互換性チェッカは，DTDが変更されたときのSGML文書の移行を支援する．すなわち，変更後のDTDが変更前のDTDと上位互換であることがチェッカによって検証できれば，Astoriaに格納されているSGML文書を変更前のDTDから変更後のDTDに安全かつ容易に移行することができる．このような上位互換性チェッカは，SGML文書を構成する要素とDTDの両方を内部的にオブジェクトとして扱うSGML文書データベース一般に適用できる．

DTD Upper-compatibility Checker for SGML Document Database "Astoria"

MAKOTO MURATA†

A DTD upper-compatibility checker for an SGML document database "Astoria" is presented. This upper-compatibility checker supports migration of SGML documents for DTD evolution. That is, if the checker proves that any SGML document permitted by an old DTD is guaranteed to conform to a new DTD, we can safely and easily migrate SGML documents from the old DTD to the new DTD. Such an upper-compatibility checker is applicable to any SGML document database system such that both elements of SGML documents and DTD's are captured as objects internally.

1. はじめに

SGML（Standard Generalized Markup Language）[1]は，構造を持った文書情報を表現するための国際規格である．文書情報をSGML化することによって，再利用と配布を促進することができる．

これらの利点をさらに追求するため，SGML文書データベースの開発がさかんに行われている．とくに，オブジェクト指向データベース上に実装されたSGML文書データベースでは，SGML文書を構成するすべての要素を再利用や配布の対象とすることができる．たとえば，オブジェクト指向データベースObjectStore上に構築されたSGML文書データベースAstoria☆では，どの要素についてもバージョン管理や他の文書との共有を行うことができる．Astoriaの提供するSDKを利用することによって，C++プログラムから要素を操作することも可能である．

単なる汎用的な要素の集まりとして文書情報を表現するのではなく，特定の用途のための要素の集まりとして表現できることはSGMLの重要な特徴である．たとえば，部品名称や顧客名称などを表す要素を積極的に導入することにより，情報の再利用と配布を容易にすることができる．使用できる要素を定義し，それらがとりうる階層構造と属性とを規定したものがDTD（Document Type Definition）である．言い換えると，どんなSGML文書が許されるかを規定するテンプレートがDTDである．

SGML文書データベースには，SGML文書だけではなく，DTDもなんらかの形で格納される．実際の運用ではDTDはしきりに変更されることから，DTDの変更を支援することはSGML文書データベースにとってきわめて重要である．理想的には，データベースに格納されたDTDを改定しても，既存のSGML文書を容易かつ安全に新しいDTDに移行できることが望ましい．

筆者は，DTDの上位互換性を判定するプログラム

† 富士ゼロックス情報システム株式会社
Fuji Xerox Information Systems, Co. Ltd

☆ Astoriaは米国Chrystal Software社が開発したSGML文書データベースである．

を Astoria 上に作成した．このプログラムは，Astoria に格納された 2 つの DTD を比較し，一方の DTD がもう一方と上位互換であるかを判定する．DTD が改版されても，新しい DTD が元の DTD と上位互換であれば，すべての SGML 文書を容易かつ安全に移行することができる．このような DTD 上位互換性チェッカは，Astoria に限らず，DTD をオブジェクトとして表現する SGML 文書データベース一般に適用できる．

本論文の残りは次のように構成される．2 章は，SGML における DTD と SGML 文書について説明する．3 章はオブジェクト指向データベースでの DTD の表現について議論し，Astoria での表現について述べる．4 章は DTD の上位互換性の判定基準を示す．5 章は Astoria 上に作成した DTD 上位互換性チェッカの実装と性能について述べる．6 章では，考察と今後の課題について述べる．

2. DTD と SGML 文書

本章では，SGML の DTD（Document Type Definition）について説明する．DTD を構成するのは要素宣言，属性定義リスト宣言，実体宣言，記法宣言である．これらについて順番に説明する．

要素宣言は，SGML 文書中に出現しうる要素の型を宣言する．また，要素型に内容モデルを関連づける．内容モデルとは，要素型についての正規表現である．したがって，要素宣言は拡張文脈自由文法の生成規則に相当する．

2 つの要素宣言からなる DTD の例を以下に示す．

```
<!ELEMENT doc - - (para*)>
<!ELEMENT para - - (#PCDATA)>
```

1 行目は要素型 doc を宣言し，内容モデル (para*) を関連づける．この宣言によって，SGML 文書に要素型 doc の要素を使用することが可能になる．内容モデル (para*) は，要素型 para の要素の 0 回以上の繰返しを意味する．2 行目は要素型 para を宣言し，内容モデル (#PCDATA) を関連づける．この内容モデルは，文字データを意味する特別な内容モデルである．

SGML 文書は，文書型宣言によって DTD を参照する．また，文書型宣言は，SGML 文書のルートとなる要素の型も指定する．さきの DTD が，doc.dtd というファイルに格納されていると仮定すると，文書型宣言は次のようになる．

```
<!DOCTYPE doc SYSTEM "doc.dtd">
```

1 つの DTD に含まれるすべての要素宣言を集めると，生成規則の集合になる．文書型宣言で指定したルートの要素型を開始記号と見なすと，DTD と文書型宣言の組合せは拡張文脈自由文法に相当する．

DTD に従う SGML 文書は，要素と文字データからなる木構造である．要素は開始タグと終了タグの対によって表現される．木構造はタグの入れ子によって表現される．要素を表す開始タグと終了タグの間に現れる要素や文字データを，この要素の内容と呼ぶ．木構造のルートとなる要素は，文書型宣言で指定した要素型に属する．要素の内容は，要素宣言によって指定された内容モデルに従う．したがって，SGML 文書は，拡張文脈自由文法の構文解析木に相当する．

以下に，さきの DTD に従う SGML 文書を示す．

```
<!DOCTYPE doc SYSTEM "doc.dtd">
<doc>
  <para>1 番目の段落</para>
  <para>2 番目の段落</para>
</doc>
```

この SGML 文書のルート要素は，開始タグ <doc> と終了タグ </doc> によって表現され，要素型は doc である．ルート要素は 2 つの要素を内容として持つ．どちらの要素も，開始タグ <para> と終了タグ </para> によって表現され，要素型は para である．1 番目の要素の内容は，文字列 “1 番目の段落” であり，2 番目の要素の内容は，文字列 “2 番目の段落” である．ルート要素の内容は，要素型 doc の内容モデル (para*) にマッチする．2 つの para 要素の内容は，要素型 para の内容モデル (#PCDATA) にどちらもマッチする．

属性定義リスト宣言は，ある要素型の要素がどんな属性を持つかを宣言し，1 つ 1 つの属性について名前，型，デフォールトの 3 つを規定する．この 3 つ組を属性定義という．

以下に，属性定義リスト宣言の例を示す．

```
<!ATTLIST doc
    date CDATA #REQUIRED
    author CDATA #IMPLIED>
```

この宣言は，要素型 doc の要素が date という属性と author という属性を持つことを宣言している．CDATA は，属性値が文字データであることを意味する．#REQUIRED は属性 date が省略不可能であることを示し，#IMPLIED は属性 author が省略可能であることを示している．

さきの SGML 文書を，この属性定義リスト宣言にあわせて修正したものを以下に示す．

```
<!DOCTYPE doc SYSTEM "doc.dtd">
<doc date="1997/9/21">
  <para>1 番目の段落</para>
  <para>2 番目の段落</para>
```

```
</doc>
```

実体宣言は，文書中で参照できる実体（外部ファイルや代替文字列など）を宣言する．記法宣言はバイナリデータの形式を宣言する．これらについての詳細な説明は省略する．

DTDを実際に運用していくと，変更が何度も発生することが知られている[2]．あるDTDに従って新たな文書を作成しようとすると，内容モデルの見直しや属性の追加が必要になることが多い．たとえば，索引語を今まで想定しなかったところに指定できるようにするには内容モデルを変更しなければならない．どれほど文書分析を積み重ね，綿密にDTDを設計しても，これはほとんど避けられない．

変更には，それまでに作成したSGML文書の改定を必要とするものと，必要としないものがある[2]．文書の改定を必要とする変更には，すべてのSGML文書をコンバートするか再編集するという大幅な手間がかかる．このような変更は十分な理由があるときにだけ行われる．一方，多くの変更は文書の改定を必要としない軽微なものである．先にあげた変更（索引語を今まで想定しなかったところに指定できるようにするもの）はその一例である．このような変更は何度も繰り返される．

3. オブジェクト指向データベースにおけるDTD

DTDをオブジェクト指向データベースで表現するには大きく分けて2つの方法がある[3]．1つはDTDをスキーマとする方法（tight approach）である．もう1つはDTDをオブジェクトとする方法である（loose approach）☆．

DTDをスキーマとするtight approachでは，このスキーマに従うオブジェクトの集まりとしてSGML文書を格納する．したがって，格納されたSGML文書はDTDに適合することが保証される．Tight approachの顕著な例として，オブジェクト指向データベースO_2（を拡張したもの）のスキーマによってDTDを表現したChristophidesらの研究がある[5]．

しかし，オブジェクト指向データベースのスキーマによってDTDを自然に表現することはいまのところ達成できていない．これは，内容モデルをスキーマによって自然に表現できないことによる．たとえば，Christophidesらは内容モデルをリストやマーク付きユニオンの繰返しによって表現しているため，SGML文書の階層構造にリストやマーク付きユニオンを表すオブジェクトが混在してしまう．したがって，(1) 内容モデルを等価なものに書き換えただけでも，データベース中に格納されたSGML文書の表現を大幅に変更しなければならない，(2) リストやマーク付きユニオンが混在していることを考慮して検索式を記述しなければならない，(3) 検索式によって生成されるスキーマをDTDに逆変換することが難しいなどの問題が起きる．

Astoriaが採用しているのはloose approachである（O_2上に構築されたHyO_2[6]もloose approachを用いている）．Loose approachでは，SGML文書だけではなく，DTDもオブジェクトとして表現される．要素宣言，属性定義リスト宣言，実体宣言，記法宣言もそれぞれオブジェクトとして表現される．要素宣言のためのスキーマはAstoriaで決められたC++のクラスである．SGML文書中の要素を表現するオブジェクトとDTD中の要素宣言を表すオブジェクトは関連づけられているが，要素オブジェクトの表現形式は要素宣言オブジェクトには依存しない．属性と属性定義リスト宣言，実体参照と実体宣言，記法と記法宣言についても同様である．

Loose approachでは，SGML文書がDTDに適合することはオブジェクト指向データベースが保証するのではなく，SGMLパーサが保証する．パーサによる検査は，新たな文書を格納するときや格納された文書を編集したときに行われる．

Loose approachにおけるDTDの変更について考察する．DTDは単なるオブジェクトであるからDTD自体の変更は容易である．また，オブジェクト指向データベースとして見る限り，SGML文書を表現するオブジェクトとの間で，なんら不整合は起こらない．SGML文書を表現するオブジェクトはDTDを表現するオブジェクトを単に参照するだけだからである．しかし，SGMLとして見ると不整合は起こりうる．すなわち，既存のSGML文書が変更されたDTDに適合することは保証されないという問題がある．実際，Astoriaにおいても保証はされない．

適合性を保証できるのはどのような場合だろうか．2章で論じたように，DTDの変更にはそれまでに作成したSGML文書の改定を必要とするものと，必要としないものがある．必要とする場合には，要素オブジェクトや属性オブジェクトの変更や作成・削除を行わなければならない．本論文ではこの場合は扱わない．一方，必要としない場合には，新しいDTDを参照す

☆ D-STREAT[4]ではメタクラスを採用しているので，DTDはオブジェクトでもクラスでもある．

るように変更するだけで適合性を保証できる．

SGML 文書の改定が必要かどうかを判定するためには，すべての SGML 文書に対して SGML パーサを実行すればよい．しかし，数多くの文書を持つような DTD については，この作業は多くの時間と手間を要するという問題がある．また，最新版の SGML 文書に対して SGML パーサを実行するだけでは不十分で，Astoria が保持しているすべての版の SGML 文書に対して実行しなければならない．

4. DTD の上位互換性

ある DTD D_1 がもう 1 つの DTD D_2 と上位互換であるとは，D_2 に適合する SGML 文書すべてが D_1 にも適合することである．DTD が変更されても，変更後の DTD が変更前の DTD と上位互換であることが判定できれば，SGML 文書を改定する必要はない．したがって，すべての SGML 文書について SGML パーサを実行するという作業を省くことができる．

以下，上位互換性を判定する基準を要素宣言，属性定義リスト宣言，実体宣言，記法宣言に分けて考察する．

4.1 要素宣言

2 章で論じたように，DTD に含まれる要素宣言の集まりに文書型宣言を付け加えると拡張文脈自由文法に相当し，SGML 文書はその構文解析木に相当する．したがって，D_1 が D_2 と上位互換であるためには，D_1 の構文解析木の集合が，D_2 の構文解析木の集合を含む必要がある．これは，(1) D_2 の構文解析木のルート要素になりうるものは，D_1 の構文解析木のルート要素になりうること，(2) D_2 の構文解析木に現れることができる要素に対し，D_2 の構文解析木において内容となりうるものは，D_1 の構文解析木においても内容となりうることの 2 つが満たされることと同値である．(1) は，D_2 のすべての要素型が，D_1 にも含まれることを意味する．(2) は，D_2 の現れるどの要素型についても，D_2 における要素型宣言の内容モデルが，D_1 における要素型宣言の内容モデルに（正規言語として）含まれることを意味する．

以上の考察から，D_2 を構成するどの要素宣言 E_2 に対しても，それと上位互換の要素宣言 E_1 が D_1 に含まれればよいことが分かる．要素宣言 E_1 が要素宣言 E_2 と上位互換であるとは，E_1 の内容モデル（r_1 とする）と E_2 の内容モデル（r_2 とする）を正規表現と見なしたとき，r_2 が表す正規言語が r_1 が表す正規言語の部分集合であることをいう．

なお，この定義はタグの最小化を考慮の対象から外している．これは，オブジェクト指向データベースに格納された SGML 文書は木構造であって，開始タグや終了タグを含んだ文字列ではないからである．他にこの定義で考慮していないものとしては，内容モデルの例外（inclusion と exclusion）がある．これらについては 6 章で述べる．

4.2 属性定義リスト宣言

前節では，要素の木構造だけについて考察した．しかし，SGML 文書は要素の単なる木構造ではなく，各要素は属性を持つ．したがって，D_1 が D_2 と上位互換であるためには，D_2 で許容されている属性が D_1 でも許されている必要がある．

以上の考察から，D_2 のどの属性定義リスト宣言 L_2 についても，それと上位互換な属性定義リスト L_1 が D_1 に存在すればよいことが分かる．ここで，L_1 が L_2 と上位互換であるとは，(1) L_1 と L_2 は同じ要素型に関わり，(2) L_2 のどの属性定義に対しても，それと上位互換の属性定義が L_1 にあり，(3) L_1 の持つそれ以外の属性定義はすべて省略可能であることである．属性定義 A_1 が属性定義 A_2 と上位互換であるとは，(1) A_1 の属性名と A_2 の属性名が同じで，(2) A_2 の許容する属性値すべてを A_1 が許容し，(3) A_2 で指定したデフォールトが`#REQUIRED`であるか，A_1 で指定したデフォールトと等しいことである．

4.3 実体宣言

SGML 文書には属性値または文字データの一部として実体が現れる．D_1 が D_2 と上位互換であるためには，D_2 で許容されている実体が D_1 でもそのまま許されている必要がある．したがって，D_2 のどの実体宣言についてもそれと等しい実体宣言が D_1 に存在すればよい．

4.4 記法宣言

SGML 文書には属性値として記法が現れる．また，実体宣言や属性リスト宣言からも記法が参照される．D_1 が D_2 と上位互換であるためには，D_2 で宣言されている記法が D_1 でもそのまま許されている必要がある．したがって，D_2 のどの記法宣言についてもそれと等しい記法宣言が D_1 に存在すればよい．

5. DTD 上位互換性チェッカの構築

Astoria に格納された 2 つの DTD を比較し，一方がもう一方と上位互換であるかどうかを，前章で述べた基準に従って判定するプログラムを構築した．本章では，このプログラムの実装について説明する．また，実際の DTD に適用した結果を示す．

DTD 上位互換性チェッカは，Microsoft VC++ 4.0

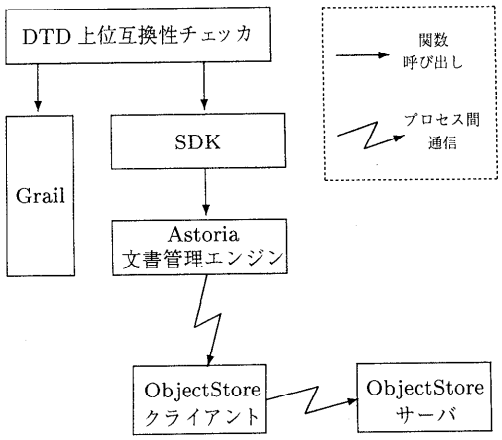

図1 DTD 上位互換性チェッカと Astoria と Grail
Fig. 1 DTD upper-compatibility checker, Atoria, and Grail.

で書かれており，簡単なユーザインタフェースを持つ．Astoria に格納された DTD を参照するには，Astoria の SDK を用いている．内容モデルの上位互換性判定には，オートマトン処理用の C++ライブラリ Grail[7] を用いている（図1）．

5.1 Astoria の SDK

Astoria に格納された DTD や SGML 文書は，C++のオブジェクトとして ObjectStore によって管理される．これらの C++のオブジェクトに対する操作を提供するのが Astoria の SDK である．SDK は C++のライブラリとして提供される．アプリケーションプログラムは，SDK を利用して SGML 文書や DTD を参照する．

SDK を用いることにより，DTD，要素宣言，属性定義リスト宣言，属性定義，実体宣言，記法宣言を C++のオブジェクトとして参照することができる．たとえば，DTD は `XD_DTD` という C++クラスのインスタンスである．DTD 中で宣言されているすべての要素宣言を得るには，`EnumerateElements` というメンバ関数を用いる．同様に，要素宣言は `XD_Element` という C++クラスのインスタンスであり，そのすべての属性定義を得るには，`EnumerateAttributes` というメンバ関数を用いる．内容モデルを得るには，`GetContentModel` というメンバ関数を用いる．内容モデルは，DTD に記述された文字列である．

5.2 Grail

Grail は，オートマトン処理用の C++ライブラリである．C++プログラムから Grail の関数を呼び出すことによって，オートマトンについてのさまざまの処理を行うことができる．

Grail は，正規表現を表す C++のクラス，オートマトンを表す C++のクラスを提供している．また，これらのクラスを操作するための関数を提供している．

Grail の最大の特徴は，正規表現とオートマトンに関するさまざまの演算が提供されていることである．正規表現からの非決定的オートマトンの生成，非決定的オートマトンからの正規表現の生成，非決定的オートマトンの決定的オートマトンへの変換，決定的オートマトンの最小化，オートマトンに関する論理演算などのための関数がすべて提供されている．また，提供されている関数を利用して，他の演算を実装することも可能である．

5.3 処理概要

DTD 上位互換性チェッカは，Astoria に格納された2つの DTD をユーザが選択するための GUI を持つ．2つの DTD のうちの1つは変更前のものであり，もう1つは変更後のものである．後者が前者と上位互換であるかどうかを判定する処理について以下に説明する．

Astoria では，DTD を必ず文書型宣言と同時に登録する．したがって，文書のルート要素の型は1つに固定されている．このため，要素宣言の上位互換性を判定するとき，すべての要素型について要素宣言の上位互換性を判定する必要はない．比較するのは，変更前の DTD のルート要素型と，ルート要素型から内容モデルによって直接的または間接的に参照される要素型だけで十分である．

処理はルートの要素型から再帰的に行われる．まず，2つの DTD のルート要素型が等しいことを確認する．次に，それぞれの内容モデルを表現する文字列を得る．これらの処理は SDK を利用して行っている．次に，内容モデルを表現する2つの文字列を構文解析し，Grail の提供する C++クラスのインスタンスとして2つの正規表現を構築する．

2つの正規表現の包含関係を判定するのは Grail の関数によって行う．まず，非決定的オートマトンを構築する関数によって，2つの正規表現からそれぞれ非決定的オートマトンを構築する．この2つの非決定的オートマトンも，Grail の提供する C++クラスのインスタンスとして表現される．次に，決定的オートマトンへの変換を行う関数と決定的オートマトンの最小化を行う関数を実行する．その次に，変更後の DTD の内容モデルに対応するオートマトンの終了状態集合をその補集合で置き換える．以上のようにして構築された2つのオートマトンの積オートマトンを構築し，さいごに最小化を行う．最小化のあとで終了状態集合

が空ならば，要素宣言の上位互換性が成り立ち，そうでなければ成り立たない．

内容モデルから参照される要素型について，以上の処理を再帰的に実行することにより，要素宣言についての上位互換性を判定している．上位互換でない要素宣言は画面上に表示される．

作成した DTD 上位互換性チェッカは，属性定義リスト宣言，実体宣言，記法宣言の上位互換性判定は行っていない．しかし，これらの実装は問題なく実装することができる．SDK は，これらを表現する C++オブジェクトを参照するための豊富な関数群を提供しているからである．

5.4 実験結果

DTD 上位互換性チェッカの機能と性能を検証するため，実際の DTD を用いて実験を行った．実験に用いたのは，CALS のために開発された MIL-M-38784C という DTD である．この DTD は 149 の要素型を持ち，テキストファイルに格納すると 967 行になる．この DTD の内容モデルを 36 カ所変更して新しい DTD を作成し，元の MIL-M-38784C と上位互換であるかどうかを判定した．使用した計算機は，CPU が Pentiurm 100 MHz で，メモリは 64 MB である．Astoira のサーバとクライアント，ObjectStore のサーバとクライアントを，すべてこの計算機の上で動作させている．

DTD 上位互換性チェッカは，内容モデルの変更のうち 9 つを非互換であると判定し，それ以外は互換であると判定した．実行に要した時間は 37 秒であった．DTD の変更はたかだか数日に 1 度程度であるから，これは十分に実用に耐える処理速度であると考えられる．

なお，DTD 上位互換性チェッカのプログラムから，正規表現の包含関係を判定する部分を取り除いて実行しても，実行時間は数秒しか短くならない．したがって，処理時間のほとんどは，ObjectStore のサーバからクライアントに DTD を転送する時間と，SDK によって DTD を参照するのにかかる時間である．属性定義リスト宣言，実体宣言，記法宣言の上位互換性判定は未実装であるが，DTD に最も多く含まれるのは要素宣言であることから，これらを実装しても性能はそれほど低下しないと考えられる．

6. 考察と今後の課題

DTD 上位互換性チェッカによって得られたもう 1 つの利点は，部品の共有についての安全性である．Astoria では異なる DTD に属する文書の間で文書部品を共有することが可能であるが，共有された部品をオリジナルの文書の側から編集すると，その部品を参照している文書が DTD に適合しなくなることがある．DTD 上位互換性チェッカを利用すれば，この可能性を排除することができる．すなわち，参照する側の文書の DTD の中でその部品に関する部分が，オリジナルの文書の DTD の中でその部品に関する部分と上位互換であることを検証してやればよい．

次に，内容モデルの例外への対処について述べる．Kilpeläinen らは，例外（inclusion や exclusion）を含む DTD を，それらを含まない DTD に変換する方法を示した[8]．変換後の DTD には，元の DTD にあった要素型のほかに，A+I-X の形の要素型が導入される．ここで，A は本来の要素名，I は inclusion の対象となる要素名の並び，X は exculsion の対象となる要素名の並びである．新しい DTD が許容する SGML 文書の集合に対し，A+I-X を A に変換する射影を施せば，元の DTD が許容する SGML 文書の集合が得られる．これを木オートマトンの理論[9),10)]から見ると，inclusion や exclusion を持つ DTD は木オートマトンによって表現できることになる．したがって，inclusion と exclusion を考慮して DTD 上位互換性を判定するには，木オートマトンに対する演算を実装すればよい．

今後の課題としては，属性定義リスト宣言，実体宣言，記法宣言の上位互換性の判定を実装すること，inclusion と exclusion を考慮した上位互換性の判定を実装することがある．より困難な課題は，上位互換でないような DTD に対して，SGML 文書を移行させることである．属性の追加や削除などで対処できる場合は，オブジェクト指向データベースにおけるスキーマ進化の手法[11)]を応用することが考えられる．

謝辞 Grail を提供していただいた Darrell Raymond 博士と Derick Wood 教授に深く感謝いたします．

参考文献

1) International Organization for Standardization: *Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)* (1986).
2) Maler, E. and Andaloussi, J.E.: *Developing SGML DTDs: From text to model to markup*, PTR Prentice-Hall, Englewood Cliffs, NJ 07632, USA (1996).
3) Brown, A.: Infomation object: Managing their evolution, use and reuse, *CALS Pacific*, Harumi, Japan (1995).

4) Aberer, K., Böhm, K. and Hüser, C.: The prospects of publishing using advanced database concepts, *Electronic Publishing—Origination, Dissemination, and Design*, Vol.6, No.4, pp.469–480 (1993).
5) Christophides, V., Abiteboul, S., Cluet, S. and Scholl, M.: From Structured Documents to Novel Query Facilities., *SIGMOD Record.*, Vol.23, No.2, pp.313–324 (1994).
6) Futtersack, P.F.P. and Espert, C.: SGML/HyTime repositories and object paradigms, *Electronic Publishing—Origination, Dissemination, and Design*, Vol.8, No.2 and 3, pp.63–79 (1995).
7) Raymond, D. and Wood, D.: Grail: A C++ Library for Automata and Expressions, *Journal of Symbolic Computation*, Vol.17, No.4, pp.341–350 (1994).
8) Kilpeläinen, P. and Wood, D.: SGML and Exceptions, *Principles of Document Processing '96*, Lecture Notes in Computer Science, Vol.1293, Springer-Verlag (1997).
9) Takahashi, M.: Generalizations of Regular Sets and Their Application to a Study of Context-Free Languages, *Information and Control*, Vol.27, pp.1–36 (1975).
10) Thatcher, J.W.: Characterizing Derivation Trees of Context-Free Grammars through a Generalization of Finite Automata Theory, *Journal of Computer and System Sciences*, Vol.1, pp.317–322 (1967).
11) Roddick, J.F.: Schema Evolution in Database Systems – An Annotated Bibliography, *SIGMOD Record*, Vol.21, No.4, pp.35–40 (1992).

(平成9年 1 月30日受付)
(平成9年12月 1 日採録)

村田 真（正会員）

昭和35年生．昭和57年京都大学理学部卒業．昭和61年富士ゼロックス（株）入社．平成5年から7年まで米国ゼロックス社のWebster Research Centerに滞在．平成7年から富士ゼロックス情報システム（株）に出向．構造化文書の研究に従事．W3C XMLワーキンググループのメンバ．国際ジャーナルElectronic Publishing（Wiley）の編集ボードメンバ．国際会議Electronic Publishingと国際ワークショップPrinciples of Digital Document Processingとのプログラム委員．ソフトウェア科学会会員．