文書構造定義のためのSGML-DTD設計支援システム

小林尋文
石塚英弘

SGML-DTD design support system for definition of a document structure

Hirofumi Kobayashi
Hidehiro Ishizuka

We are developing a SGML-DTD design support system with GUI (Graphical User Interface) using object-oriented approach. Here, SGML is ISO 8879: Standard Generalized Markup Language. In a SGML-based system, a structure of document is defined in a DTD (Document Type Definition) written in SGML. Our system has 3 features as follows; (1) a document structure is presented in a tree whose node indicates an element; (2) a user can define / update the structure using GUI; (3) the system outputs the structure in a form of DTD. We are developing the system using C++, OSF/Motif and X-Window. The system will decrease difficulties in designing DTD.

1. はじめに

本システムは、文書構造を分かりやすい形で表現するため、木構造のGUIで表すこと、そのGUIを操作してSGML(Standard Generalized Markup Language)[1)]の文書構造定義(DTD : Ducument Type Definition)を作成できるようにすることを目的とする。また、システムは、オブジェクト指向アプローチ[2)]に基づいて設計し、C++とOSF/Motifを使って構築している。

2. SGML

2.1 SGML利用の広がり

文書の電子化・電子出版・フルテキストデータベースの作成などの分野で、SGMLが注目を集めている。SGMLは、特定の処理システムに依存しない文書(SGML文書)を作り、交換や異なった利用をするためのISO規格である。SGML自体は、DTDを書くための言語である。そして、DTDが、文書の構造(フォーマット)を定義する。

2.2 SGMLとDTDと文書構造

SGML文書は、SGML宣言、DTD、DTDに従ってマーク付けされたテキストの3つで構成される。DTDは、文書の構造とテキストのマーク付けの方法を定義する部分である。DTDで文書構造を定義することによって、SGMLはどのような文書でも、作ることができる。また文書の構造が明示されるために、異機種間あるいは異なる処理系間の文書交換が可能となる。図1にDTDの簡単な例を示す。DTDは文書の構造を要素定義文を使って表す。

2.3 要素定義文の構文

要素定義文の構造は、以下のようになっている。

<!ELEMENT　要素名　開始・終了タグ省略情報　子要素の並び>

1) 要素名は、その要素を特定する一意の名前を付ける。

```
<!ELEMENT book      - - fm , body , bm>
<!ELEMENT fm        - O title , author>
<!ELEMENT title     O O #PCDATA>
<!ELEMENT author    - O person+>
<!ELEMENT person    - O name , address>
<!ELEMENT name      O O #PCDATA>
<!ELEMENT address   - O #PCDATA>
<!ELEMENT body      - O chapter+>
<!ELEMENT chapter   - O ( section | p )+>
<!ELEMENT section   - O p+>
<!ELEMENT p         - O #PCDATA --paragraph-->
<!ELEMENT bm        - O akm? , index>
<!ELEMENT akm       - O #PCDATA --aknowledgement-->
<!ELEMENT index     - O #PCDATA>
```

図1　簡単なＤＴＤの例

2) 開始・終了タグの省略情報は、1文字目が開始タグ、2文字目が終了タグの省略情報を表す。"-"の場合は省略不可、"O"の場合は省略可である。

3) 子要素の並びは、1)の要素の下に出現する要素の名前が書かれる。子要素の出現の順序と出現の頻度は記号を使って表す。

3a) 要素の出現順序は、接続子(Conector)で表現する。

","で区切られている要素は、その順序で出現しなければならない。

"|"で区切られている要素は、どちらか１つが出現する。

"&"で区切られている要素は、どちらも出現するが、順序はどちらでもよい。

3b) 要素の出現頻度は、出現指示子(Occurrence Indicator)で表現する。

1回だけ出現する要素は、その要素の名前だけを書く。

出現してもしなくてもよい要素は、その要素の後ろに"?"を書くことで示す。

1回以上出現する要素は、その要素の後ろに"+"を書くことで示す。

0回以上出現する要素は、その要素の後ろに"*"を書くことで示す。

図1のDTDの1行目は、bookという要素は、開始タグ・終了タグを省略できないこと。そして、fm、body、bmの3つの要素で構成されていて、この順序で現われなければならないことを示している。

3. システムの機能

1) DTDの構造表示

DTDを読み込み、DTDの定義する文書の構造を木構造の形で表示する。ユーザーが定義した要素だけでなく、接続子、出現指示子もノードとして取り入れ、DTDが持つ詳しい情報を表現している。

木構造の見方は、左側が親要素である。そして、DTDでは、子要素の並びに順序に意味があるので、その順序を、上から下で表示する。

2) GUIを使用したDTDの作成・変更

表示される木構造のノードやダイアログを操作してDTDを作成・変更をすることができ

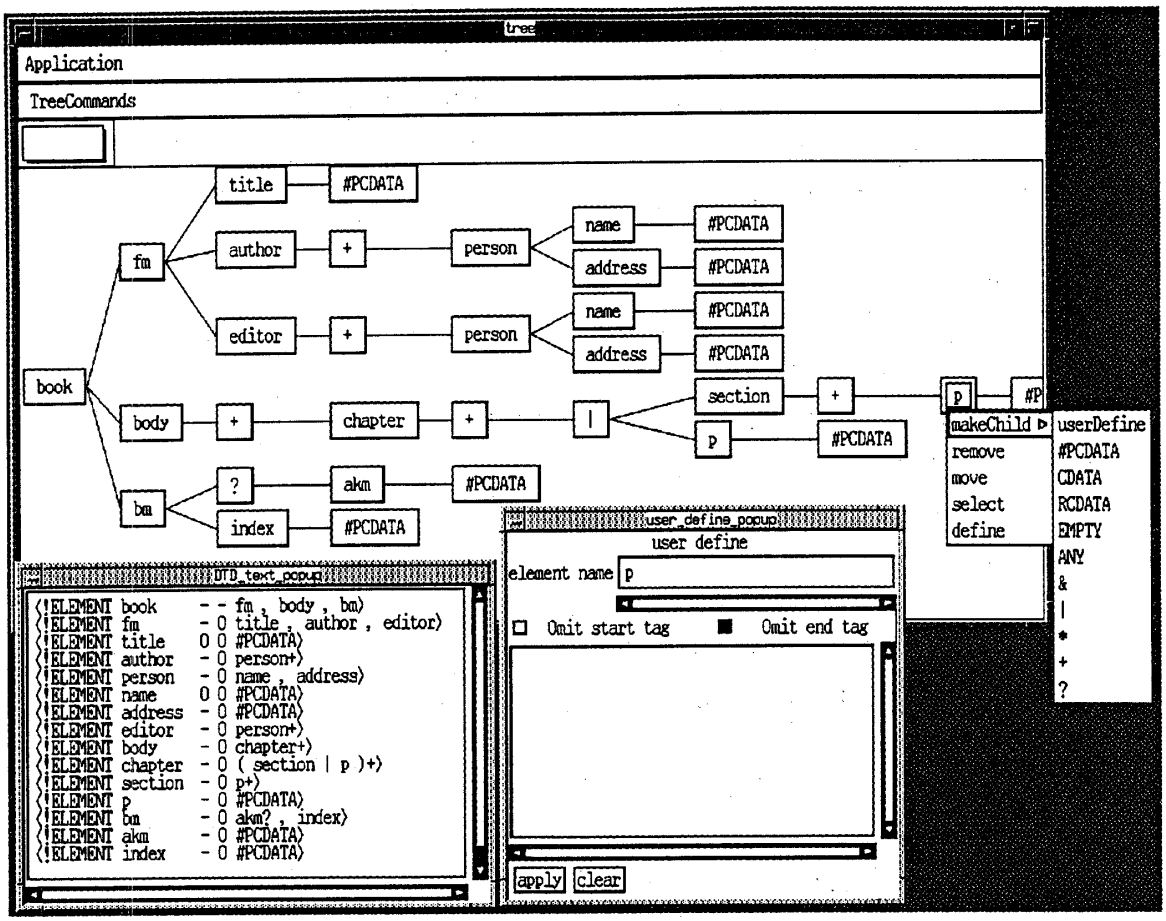

図２　システム実行例

る。現在、本システムでは、DTDの基本機能：要素定義文、属性定義文、実体定義文のうち、要素定義文については機能を実現し、残り2つについては、設計中である。これらの機能が完成後、短縮参照などその他の機能も付け加えたい。

3) DTD注釈ファイルの作成

各要素の役割などを書いた注釈ファイルを作成する。DTDの中に注釈を書き込むことも、できるが、後で利用する時の便利を考え、注釈を別ファイルにすることにした。

ここで要素定義文のGUIを使った作成機能について図2を用いて説明する。各ノードは、プルダウンメニューになっている。そのメニューの中の操作を選ぶことによってDTDの作成、変更を行う。

すべての種類のノードに共通の操作は、"remove"、"move"、"select"である。

"remove"：その要素とその子孫の要素を取り除く。

"move"：その要素を他の要素の子要素とする。"move"を選ぶとカーソルが変わるので、移動したい要素を選ぶと移動する。その要素の子孫要素も移動する。

"select"：その要素を選ぶことができる。複数の要素を選んだ後で、その選んだ複数の要素を1度に"move"、"remove"することができる。

オプションノード、ユーザ定義ノードに共通の操作は、"makeChild"がある。

"makeChild"：その要素の子要素を作る。このボタンを選ぶとプルダウンメニューができるので、その中から作りたい子要素を選ぶ。

ユーザ定義ノードの操作

"define" : このメニューを選ぶとこの要素についての詳細な情報が表してあるダイアログが出て来る。このダイアログに表示される情報は、要素の名前、開始・終了タグの省略情報、子要素の一覧である。なお、属性の一覧、注釈ファイルの情報として使用する注釈の入出力も実装するつもりである。

4. システム設計の方針

このシステムは、オブジェクト指向に基づいて設計を行った。オブジェクト指向を取り入れることによって、各クラスは、それ自身で独立して機能するために、各クラス(要素、属性、実体、注釈、ヘルプ)を個々に設計し、テストを行うことができ、システムを効率的に開発することができる。

本システムでは、DTD設計に必要な5つのクラス(要素、属性、実体、注釈、ヘルプ)とそれらを制御するクラスを設計した。

ここでは、設計・機能の実現ができている要素クラスについて説明する。

本システムは、DTDの木構造を表すために、要素オブジェクトとノードオブジェクトを用いている。前者はDTDのデータを記憶するためのもので、後者はGUIを与えるためのものである。

要素オブジェクトは11種類あるが、DTDでユーザが定義する要素(以下、ユーザ定義要素)、SGMLで定義済の要素(以下、プリミティブ要素)、接続子、出現指示子の4つに分けることができる。ユーザ定義要素とプリミティブ要素は、1つの要素ごとに1つだけ作られる。接続子、出現指示子は、DTDで現れているところすべてに１つずつ作られる。

ノードオブジェクトはGUIの機能の面から3つに分けられる。ユーザ定義要素を表すユーザ定義ノード、プリミティブ要素を表すプリミティブノード、接続子・出現指示子を表すオプションノードである。

このように、データ構造とGUIの機能を分けたのは、1つの要素が、複数の要素の子要素となることがあり1つの要素に対する木構造のノードが複数になることがあるためである。またこれらの要素オブジェクト・ノードオブジェクトでは、継承の機能を用いることによって、プログラムコードの量を減らすこと、共通の機能を1カ所で管理することができた。

5. 終わりに

現在、文書構造を木構造で表現すること、DTDの要素定義文を出力すること、の2つは、実装した。本研究の目的のうち、文書構造をわかりやすい形で表現すること、要素定義文から構成されるDTDをGUIを使って作成することは、細かな変更を残すだけとなった。しかし、DTDの読み込み、要素定義文以外のDTDの文については、現在、設計中で、近く実装する予定である。

文献

1) Herwijnen,E.V. SGML懇談会WG監訳. 実践SGML . 東京, 日本規格協会, (1992)

2) D.A.Young.著 磯谷正孝, 林秀幸訳. オブジェクト指向プログラミングとC++:OSF/Motif版. 東京,株式会社トッパン, (1993)

(図書館情報大学, Univ. of Library and Information Science)