Vol.2018-DC-109 No.4
2018/7/3

ドキュメントの定義と型に関する考察

大野邦夫†

ドキュメントは記録や証拠という概念を本質的に有するが、その構造の標準化に関する検討はテレテックス・サービスと呼ばれたワープロ文書通信の標準化が試みられた1970年代に遡る。CCITTとISOによる標準文書構造がODAとして1980年代半ばに仕様化された。それと並行してマークアップ言語のSGMLによる標準化が進展し1980年代後半に定められた。他方、XeroxPARCのALTOPCが、1980年代初頭にStarワークステーションとして商品化され、文字・図形・画像の複合文書を実現し、その結果DTP市場が生まれ、1990年代に複合文書がビジネス文書として普及した。その後複合文書標準がOMGとW3Cで検討されたがデファクト標準には敵わず標準化は見送られた。現在のWebで用いられているHTML5とXMLは、それら複合文書技術の到達点であるが、セマンティックWebに代表される意味概念と文書構造との関係付けが今後の課題である。

A Study on the Definition of Document with its Types

Kunio OHNO†

The essence of document concept is record and evidence. The study on document structure has started at the end of the 1970s when standardization of word processing document communication was attempted. Based on CCITT and ISO, document format was specified as ODA in the mid−1980s. At the same time, standardization by markup language SGML was established at the end of the 1980s. On the other hand, Xerox developed and introduced "Star Workstation" as an office application of its PARC's ALTO PC in the begining of 1980s, and then DTP market was born based on the compound document technology by Star Workstation in 1990s. The current Web constructed by HTML5 and XML is the realization of those compound document structure. Then it will be an important problem to relate the semantic concept represented by Semantic Web and the document structure.

1. はじめに

文書の定義と型については、2000年3月にDC研の前身であるDD研で報告したことがあるが[1]、その後18年を経て再考を試みた。その背景には、私どもの地磁気逆転層による地域活性化の研究報告に対する議論が存在するが[2][3]、そのような用語定義の議論よりは、それをトリガーにして18年前の不十分な考察を深める方が有意義と思うことによる。

文書（ドキュメント）という語彙の定義に関しては、18年前の報告で考察しているのでそれを参照して頂けると幸いである。書籍としての文書がページを束ねた冊子としての記録であり、その固定された証拠としてのページ記述は、層としてのリーフ（葉）相互間が奏でる文脈的な意味概念を生じ人間の知的興味を誘発するものであろう。文書は固定された静的なものであるが、その文脈から動的な知識を生じることから、ハイエンドのDTPの製品であったInterleafのActiveDocumentが発想された経緯がある。私にその趣旨を語ってくれた、Interleaf社のCEOであったデイビッド・ブシェー氏は私にとって忘れ難い人物である。

地層を文書に譬えるならば、それはleafとしての個別の層の集合であり、そのコンテキストは文書が個々のページから発する知識の香りのメタファーである。その文脈から地層をドキュメントに譬えたのであるが、その理解を提供するのはチバニアンをトリガーにする地域活性化の議論とは異質であることからこの報告をまとめ議論の対象とする次第である。

2. CCITT・ISOにおける文書形式の標準化

2.1 テレマティーク通信

私が文書に関係するようになったのは、CCITTにおけるテレテックス通信サービスの端末開発が最初であった。その発端は、1977年に遡るが、欧州を中心とするテレックスに代わる英数文字レベルの文書通信、当研究会の名称でもあるドキュメント・コミュニケーションであった[4]。その草案がベーシック・テレテックスとしてまとめられたのが1980年で、次の会期の1981年～84年でその拡張が検討された。その際に、NTTの研究所内でテレテックス・サービスの日本語化が検討され、2バイトコードの文字セット、1バイト系と2バイト系の切り替え、外字の転送などが仕様案として作成され、それを実装した端末装置の試作が私の部署で行われた。デジタル交換網を経由した接続実験で機能確認を行ったが、テキスト文字だけの原始的なワープロ文書の通信サービスは、Starワークステーションに端を発するDTPシステムと企業内LANが普及し始めた1980年代中頃には既に技術的には完全に陳腐化していた。

その後テレテックス・サービスはCCITTとISOにより進められたテレマティーク通信とODA（Open Document Architecture）の検討に包含された。1981年から84年にかけての検討の結果、テレテックス・サービスとG4ファクシミリは統合してミクストモードとし、テレマティーク・サービスとすることが決まった。その仕様体系を図1に示す。

文字文書通信のテレテックス・サービスと画像文書通信のG4ファクシミリ・サービスの統合がもたらすのは、文字・図形・画像文書通信サービスである。そこで議論になったのが文書アーキテクチャであった。

テレマティーク・サービスにおいては、ベーシック・テレテックス・サービスのような、A4の物理的なページを基準にするのは余りにも時代遅れであり、文書通信後に、受信側で文書編集を可能とする手順が検討された。その結果、下記の3種類の方式がサービスとしてて供されることになった。

(1) 固定フォーマット
(2) 処理可能フォーマット
(3) 書式付き処理可能フォーマット

以上の(1)は、従来のファクシミリやベーシック・テレテックス・サービスの延長の仕様である。(2)は所謂論理構造のデータを送信し、受信側が自由にレイアウトするものである。(3)は送信側がレイアウトフォーマットを添付し、固定

†(株) モナビITコンサルティング
Monavis IT Consulting Co. LTD.

ⓒ 2018 Information Processing Society of Japan

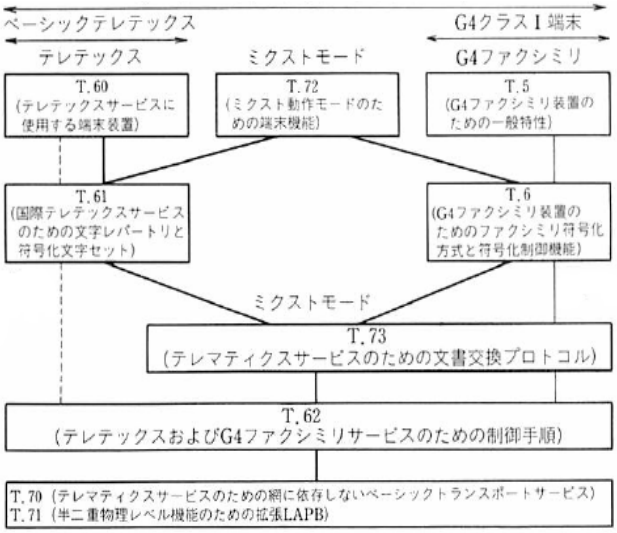

図1　テレマティーク・サービスへの移行

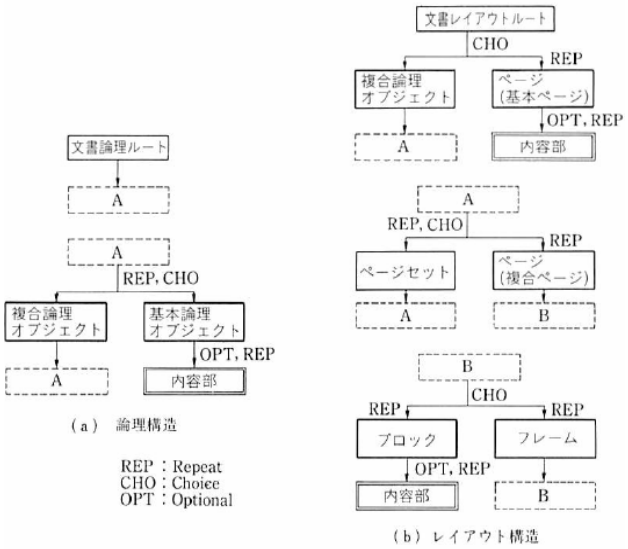

図3　処理可能文書の構造

フォーマットサービスと同様に受信可能であるが、そうでないレイアウトにも変換可能とするサービスであった。

2.2 ODAの文書アーキテクチャ

以上を実現するための検討が行われ、図2に示すような中間的な結果が取りまとめられた。図の左側は抽象的（generic）なカテゴリ区分で右側が具体的（specific）なカテゴリである。この区分はオブジェクト指向プログラミングによる実装を意識したものであり、その関係者の技術仕様であろう。

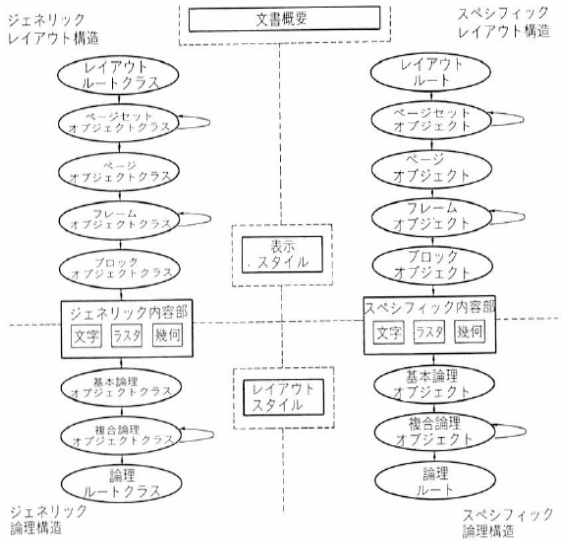

図2　処理可能文書フォーマットの検討経緯

図の上方は、レイアウト構造に関係する領域、下側が論理構造に関係する領域である。論理構造は下が木構造のルートになっていて、中間ノード（章、節、項など）としての複合論理オブジェクト、リーフとしての基本論理オブジェクトの階層になっている。基本論理オブジェクトは内容部の文字、ラスタ（画像）、幾何（図形）を包含する。

レイアウト構造は、上がルートになっており、ページセット、ページ、フレーム、ブロック、内容部という階層で管理される。図3は、その階層を具体的に示している。

論理構造は、ルート、複合論理オブジェクト、基本論理オブジェクトから構成され、基本オブジェクトが内容実体に結びついている。しかも複合論理オブジェクトは再帰構造になっている。要するに複合論理オブジェクトは、章、節、項といった見出しであろう。基本論理オブジェクトは、文字、図形、画像を包含するパラグラフ実体を含む見出しであろうと推察される。

レイアウト構造は複雑である。ページセット、ページ、ブロック、フレームという階層があり、最上階層で複合論理オブジェクトと関係している。このような屋上屋の実装が現実的かどうかは、多少文書構造の実装に関わった人であれば理解可能であろう。上記は文書の分析結果であって、システム実装の要件からは程遠い内容である。同じISOで文書構造の標準化を担当したSGML関係者からODAに関する批判をいろいろ聞かされたが、それは当然である。

2.3 OSI参照モデルの実装

高性能なDTPシステムが、構造的な階層をコンポーネントのみとし、フラットな構造にしてコンポーネントの属性だけでレイアウトをコントロールしたのとは著しく対照的である。蛇足であるが、図4は、ODAの処理系である。DTPシステムであれば簡単に済む処理である。システム実装の経験が無い技術者の処理モデルであろうが、組織的な合意手法が上手な技術者が仲間内のグループを形成して議論するとこのようなシステムが構築されるという典型である。

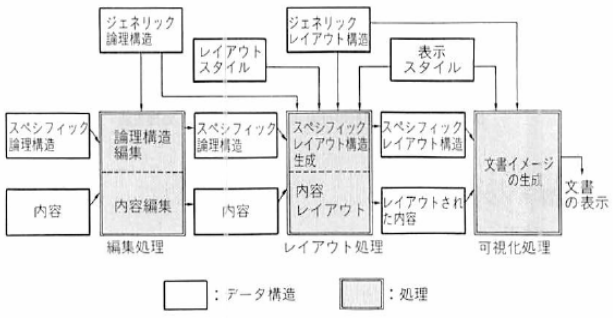

図4　ODAの処理系

図5は、テレマティークサービスにおけるプロトコルスタックである。以上の過剰な構造に基づきOSI参照モデルのプロトコルスタックが埋められていった状況が推察される。しかしながら、OSIモデルも実際に使用されることは無かったことを考えると、コンピュータやネットワーク分野における主官庁を中心とする技術標準化プロセスが機能しない状況が垣間見えるであろう。以上の状況において標準化の勝者となったの

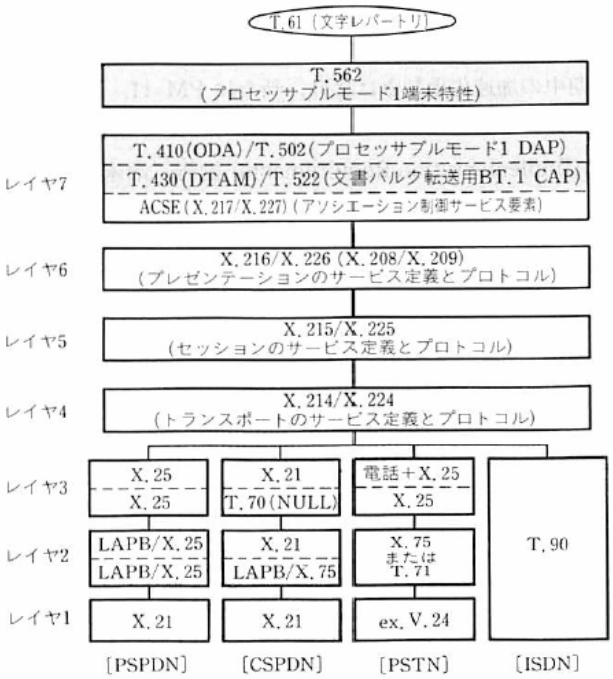

図5　テレマティークサービスのプロトコルスタック

は、現実のコンピュータネットワークを構築し使用していたUnixを活用したARPANETの関係者であった。Unixは、1970年代の米国の大学でPDP−11上のTSSシステムとしてアカデミックな利用者で便利に使用され、その上でC言語に基づくプログラム開発ツールや文書編集ツールが開発され使用された。当初のテキストエディタはedと呼ばれた行エディタでありプログラム開発と文書作成に使用された。文書作成のための文書整形ツールとして、nroffといったマークアップ言語によるアプリケーションが開発され、レイアウトされた英文作成用に使用された。1980年代になると、VAX−11が米国の大学や企業の研究所に導入され、UCバークレーやシリコンバレーの挑戦的な技術者たちによりUnixの改良が図られた。その結果UnxはミニコンのTSSシステムとしてだけでなく、個人の業務用のSunワークステーションのOSとして普及すると共に、キャンパス内や先進の事業所内で従来のモデムによる通信に代わるローカルネットワークが普及した。Sunワークステーションでは、ウインドウシステムによるGUIが使用されるようになり、エディタも行エディタからスクリーンエディタに代わり、viやEmacsのような製品が使用されるようになった。他方、文書フォーマットに関しては、nroff、troffのような原始的なものから発展したScribeやTeXといった高機能な製品が出現し、出版業界などに普及した。他方マークアップ言語をより汎用的に活用するための工夫が行われ、GML（Generalized Markup Language）という用途に応じたマークアップ言語を定義するメタ言語が考案された。さらにそれがISOで標準化されてSGML（StandardGeneralizedMarkup Language）となった。SGMLは、文書構造を定義する言語であり、レイアウト構造を定義するDSSSL、清書構造を定義するSPDLとセットで系統的に定義される方針であった。その後DSSSLは仕様化されたがSPDLは、PostScriptが事実上の清書文書の標準となったことから標準化は見送られた。

3. 文書型

3.1 DTD

次に文書型について考察する。DTD（Document Type Definition）は文書型定義と訳され、SGMLにおける文書構造の型の定義として使用されてきた。文書構造は書籍が最も一般的な対象であるがその論理構造は、表紙、書名、目次、章、節、パラグラフ、索引、裏表紙といった構造になっている。これをDTDで記述すると図6のようになる。但し、論理構造なので表紙の代わりに書名とし、裏表紙の代わりに奥付とした。DTDの記述だと感覚的な把握は難しいので図で示すと図7のようになる。DTDについての知識がない人は、図6を見ても何のことやらまったく分からないであろう。それに対して図7は書籍に対する知識がある人であれば、理解可能であろう。

```
<!DOCYPE 書籍[
<!ELEMENT 書籍（書名,目次,章構造+,索引,奥付）>
<!ELEMENT 章構造（章タイトル 節構造+）>
<!ELEMENT 節構造（節タイトル パラグラフ+）>
<!ELEMENT 書名（#PCDATA）>
<!ELEMENT 目次（#PCDATA）>
<!ELEMENT 章タイトル（#PCDATA）>
<!ELEMENT 節タイトル（#PCDATA）>
<!ELEMENT パラグラフ（#PCDATA）>
<!ELEMENT 索引（#PCDATA）>
<!ELEMENT 奥付（#PCDATA）>
]>
```

図6　書籍モデルのDTD

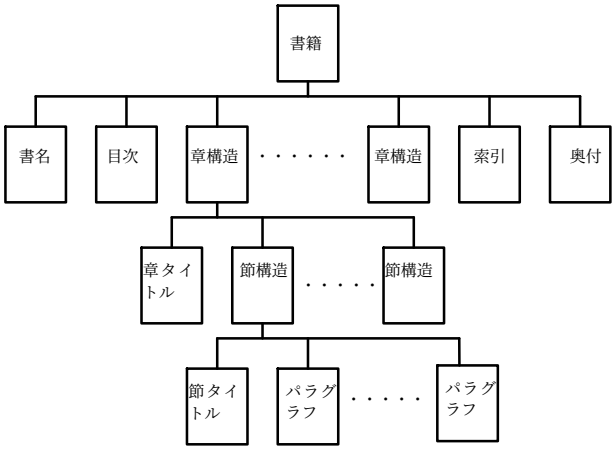

図7　書籍の論理構造モデル

言語的記述と図形的記述の差異はこのようなものであり、全体を感覚的に把握する空間的な認識と細部を分析する言語的な把握・認識の差が両者の対比により理解できるであろう。

ところで、文書型とは何か。汎用マークアップ言語は、nroffのような文書レイアウトのための個別マークアップ言語のメタ言語であり図6の記述から察せられる通り、文書型は文書構造を提示する。この機能が型であることを知るには、型の一般論を把握しておく必要がある。

4. 型に関する議論

4.1 ラッセルの型の理論

文書型の型の思想は、型の一般論に基づくものであり、プログラミング言語の型と同様に集合論における型の理論に関係する。集合論における型は、バートランド・ラッセルの数理哲学におけるパラドックスの回避に端を発するので簡単に

紹介する。ラッセルの型の理論は、自己参照を含む集合の定義におけるパラドックスを回避するアルゴリズムとして考案された体系である。この哲学的分析は、彼の著書である“The Principles of Mathematics − Appencix B: The Doctrine of Types”[5]の記述がオリジナルである。ラッセルはこの内容に関して彼の晩年の著書である “My Philosophical Development”[6]に分かりやすく記述しているのでその内容を簡単に紹介する

(1) 命題は全面的（totality）な範囲を扱うものとそうでないものに大別される。

(2) 前者の全面的な範囲はそれ自体の要素ではあり得ないであろう。

(3) 命題の全体を指示しない命題を第一階の命題（first−order preposition）と呼ぶ。

(4) 全面的な範囲を扱う命題を第二階の命題と呼ぶ。

第二階の命題に対して、(2)〜(4)を無限に繰り返すことによりパラドックスの回避が可能となる。ラッセルによる型とは、上記の手順につづく系列的な命題の集合である。

型というとコンピュータ関係者はプログラミング言語のデータ型を思い浮かべるが、ラッセルはトータルな世界を集合論で定義することを試みてこの問題にぶつかり、型を考案した。それは、“The Principles of Mathematics”の内容から明らかである。ラッセルはこの著書で代数学的な演算と引数となる集合群で世界を記述することを試みている。

自然数から加減乗除の演算を通じて整数、小数、有理数、複素数、無限・・・という数の世界を拡大し、無限、連続、時間、空間、物質・・・という観点で世界を拡大する一方で、演算、論理、メタ概念、といった関係を通じて意味的な世界を追求しようとした。その第一歩のトータルな世界の記述で、パラドックスにぶつかったのであるが、世界を数理的な枠組みで思弁的に捉える発想は今日の自然科学に引き継がれ、その数理思弁的な仮説の実験的検証が素粒子から宇宙論に至る最新の物理学であろう。

4.2 データ型

プログラミング言語のデータ型は、整数、浮動小数、文字、ブーリアンなどの型であり、Fortranでは、整数と浮動小数の区別程度であったが、Pascalで文字、文字列やブーリアン、ベクトル、配列など、さらにC言語では、整数に対しては、short、long、unsignedshort、unsignedlong、浮動小数に対しては、倍精度のdouble、構造体のstructなど実用に応じた拡張が行われた。

このようなデータ型に関しては、J.C.Cleavelandの“データ型序説”[7]によると、当初は「型とは値の集合である」と考えられていたとのことであった。しかし、Jim Morrisという人が「型は値の集合ではない」という論文を書き多様な議論が生じた（データ型論争）。1970年代に入り、抽象データ型の概念が普及してからは、「型とは値の集合およびその上の演算の集合である」という定義が定説になっている。なお、値と演算を命題に置き換えるとラッセルの型の定義に近いものとなる。型を値だけでなく演算も含める集合とすることにより、演算の引数としての値の型と、演算の結果としての値の型を集合として扱う必要が生じる。要するに、引数:x、演算:f、結果:yとすると、

$$y=f(x)$$

となるが、型を考慮すると、

$$y:\rho=f(x:\sigma)$$

となる。要するに型σの値のxが、演算fにより型ρの値yの結果を得るということである。この概念を文書構造のレイアウト構造の変換に対応付けることが可能である。すなわち、yをレイアウト変換された文書、xを論理構造の文書とするのである。σが論理構造の型（章、節、項など）、ρがレイアウト構造の型（ページ、コラム、行など）と考えると、文書の編集処理は、この演算と見なすことができる。この概念に類似の処理を関数型プログラミングが実行する。MLやHaskelのような言語は、引数と結果の型を対応付けるだけでなく、その関係に基づく型推論を行う。DTDとDSSSLは、この関係で把握することにより、型として位置付けることが可能であり、その後のXMLとXSLとの関係も同様である。

5. クラス

5.1 Simula

クラスは、Simulaで最初に使用され、その後オブジェクト指向プログラミング言語の元祖であるSmalltalkに適用されて広く知られるようになった。その後大半のオブジェクト指向プログラミング言語において使用され、最近のプログラミング言語としては必須とも言える重要な概念である。

Simulaは、オスロのノルウェー計算センターのオルヨハン・ダールとクリステン・ニガードによりAlgol60を拡張して開発が試みられたシミュレーション用の言語であった。Algolの、begin〜endのブロック要素を事例とし、それをクラスとオブジェクトの概念に分離し、個々のオブジェクトの振る舞いを集合的に扱えるようにした言語で、オブジェクト指向プログラミングの基本的特徴を備えた言語であった。

5.2 Smalltalk

Smalltalkは、全てをオブジェクトとして扱うので型は無いことになるが、実際には、図8に示すように、digit, digits, number, letter, special character, characterといった型的なオブジェクトの系列が示されており、言語文法的にはオブジェクトであるが意味的実態は型である。この図は、さらに拡張されていて、identifier, symbol, symbol−constant, character−constant, string, comment, array, array−constant, literal, variablename, unary selector, binary selector, keyword, primary, unary object description, binary object description, unary expression, binary expression, message expression, cascaded message expression, expression,statements, block, temporaries, message pattern, method というかなり膨大なオブジェクト群が示されているが、これは概念的な分類ではなく、視覚的に明瞭な分類に基づくデータ型的な分類である。

5.3 ZetalispとTAO

Lispのオブジェクト指向プログラミングは、Zetalisp (Lisp Machine Lisp) から始まった。Zetalispでは、クラス概念をflavorという名称で仕様化し、クラスという名称を使用しないオブジェクト指向言語である。これはMITのハッカー集団が、クラスの代わりにアイスクリームのフレーバをメタファにして多重継承をアピールしたためである。私自身のオブジェクト指向プログラミング経験はこのフレーバに端を発する。LispマシンELIS上のマルチパラダイム言語TAOのオブジェクトシステムは、Zetalispのflavorに近いものであるが、名称はクラスとし、メソッドはSmalltalk的なinfixによる表記で実現した。

5.4 Common Lisp

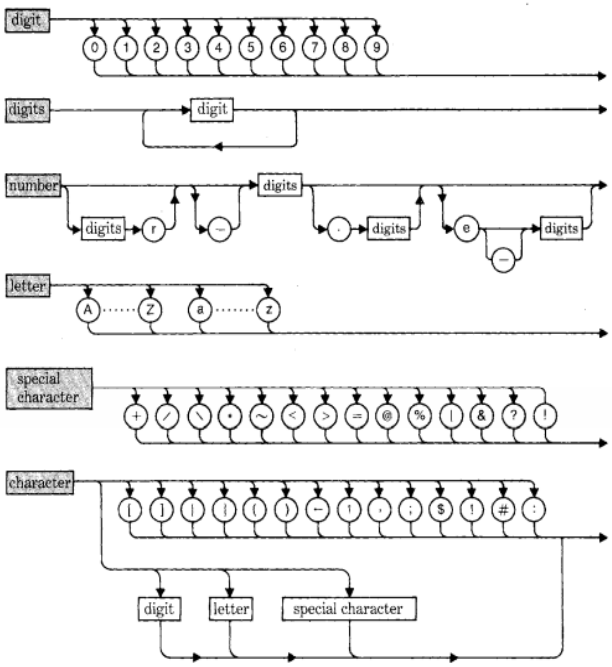

図8　Smalltalk−80における基本オブジェクトの図示

Common Lispのオブジェクトシステム（CLOS）は、ZetalispやTAOのクラス的な概念を引き継ぎつつ、オブジェクトシステムを従来の関数型から分離しない汎用オブジェクトシステムとした。総称関数（genericfunction）の思想に基づくメソッド実装としたので、メソッドは関数とシームレスな実装となり、ポリモルフィズムが実現され新たなパラダイムが開かれた観がある。TAOのマルチパラダイムは、異なる文法を同一言語に持ち込んでオブジェクト指向を取り込んだが、CLOSは同一の文法で、引数のクラスに対応する振舞い（メソッド・関数）を呼び出すのでオッカムの剃刀的なエレガントな仕様であり、ポリモルフィズムに基づく高度な実装を実現している。

5.5 C++、Javaなど

C++、Javaは、継承メカニズムを有するクラスを用いるオブジェクト指向プログラミングを行うが、引数や結果に対しては、型を宣言するのでポリモルフィズムではない。しかしながら、関数の実行という観点では、先に記述した形式である$y{:}\ \rho = f(x{:}\ \sigma)$に対応しており、クラスを型の延長として扱っていると考えることが可能である。

5.6 CORBAのインタフェース定義とIDL

なお、Javaは、クラス継承の外にネットワーク環境におけるクラスに対応するインタフェース継承機能を有している。このインタフェースの概念は、OMGのCORBAが提起したもので、ニュートラルな型（IDL型）についても仕様化しているので、型やクラスの議論に際しては興味深い対象である。

CORBAのインタフェースは、クライアント・サーバシステムにおけるクライアントからサーバへの要求と応答の関係を、代数学における関数の引数と結果に対応させるモデルである。すなわち、先の $y{:}\ \rho = f(x{:}\ \sigma)$ の多次元版でありCORBA仕様書において下記のように記述されている[8]。

$$\Omega{:}\ (x_1{:}\ \sigma_1, x_2{:}\ \sigma_2, \ldots. x_n{:}\ \sigma_n) \rightarrow (y_1{:}\ \rho_1, y_2{:}\ \rho_2, \ldots.. y_n{:}\ \rho_n)$$

この記述は操作シグニチャ（Operation Signiture）と呼ばれ、CORBAの基本概念である。操作名称と入出力の変数と型の定義をインタフェース定義と呼び、定義されたインタフェースに基づき、各種プログラミング言語に変換される。例えば、IDLで記述された下記のインタフェースを考える。

```
interface rectangle {
    float ares (in float horizontal−length,
                in float vertica−length);
    };
```

このインタフェースは、CORBA 1.1の仕様では下記のC言語によるコードに変換することが可能である。

```
typedef Object Rectangle;
    float Rectangle_area (,
        float horizontal−length,
        float vertica−length
        );
```

Cはクラスが定義されないので、インタフェース名は型と手続き（操作）名の接頭辞として付与され識別される。IDL型は、表1のようにC言語のデータ型にマッピングされる[8]。

表1　C言語へのIDL型のマッピング（CORBA 1.1）

IDL	C
short	short
long	long
unsigned short	unsigned short
unsigned long	unsigned long
float	float
double	double
char	char
boolean	unsigned char
octed	unsigned char
enum	unsigned long
any	typed of struct any{TypeCord_type...}any

その後CORBA2.0が仕様化され、C++とのマッピングが定義された。先の例のC++コードを示す。

```
class rectangle {
    float area (
        float horizontal−length
        float vertica−length
        );
    };
```

この場合はインタフェースは自然な形でクラスに対応付けられる。IDL型のC++言語の型へのマッピングを表2に示す[8]。

表2　C++言語へのIDL型のマッピング（CORBA 2.0）

IDL	C++
short	CORBA::Short
long	CORBA::Long
unsigned short	CORBA::UShort
unsigned long	CORBA::ULong
float	Float
double	Double
char	Char
boolean	Boolean
octed	Octed
enum	C++ enum
any	typed of struct any{TypeCord_ptr...}any

ⓒ 2018 Information Processing Society of Japan

IDL型は、C、C++以外の多くの言語へのマッピングが定義されたので、言語中立の型として注目され、期待されたがCORBA自体が普及しなかったので、あまり活用されなかった。ただしXMLのDOMは言語中立にAPIを記述することを必要としたために、IDLを用いている。

5.7 構造体の拡張としてのクラス

以上、SimulaからSmalltalk、Lisp系オブジェクト、さらにC、C++、CORBA Interfaceに関して、クラス的な仕様に関して述べた。以上において、C以外は構造体の拡張であるクラスとしてのカプセル化された枠組みを持ち、しかもサブクラス定義により継承機能を有する。

クラスは演算を管理する外殻として存在すると言える。それはクラスが存在しないC言語において、潜在的なクラスに相当するインタフェース名を演算名称の前に付けていることからも理解できる。

5.8 RELAXNG

RELAXNGはDTDと等価な記述をXML文法で実現した。図6のDTDであれば12行で済む記述が、RELAXNGだと実に30行余りになる。XML制定時にさんざん嫌われたDTDであるが、RELAXNGやXML Schemaに比べると極めてコンパクトな記述が可能である。そのために、オッカムの剃刀の譬え通り、文書型を記述する際には、今でもDTDが使用されていると思われる。

5.9 XML Shema

XML Schemaは、W3CとしてのDTDに代わるべきXML文書の正式な構造仕様である。行数だけだとRELAXNGと同等であるが、ComplexTypeを余分に定義するために記述量は倍近くになる。XML Schemaで問題なのはデータ型である。

string、boolean、decimal、float、double、duration、dateTime、time、date、gYearMonth、gYear、gMonthDay、gDay、gMonth、hexBinary、base64Binary、anyURI、QName、NOTATIONといった基本型が存在する。さらにnormalizedString、token、language、NMTOKEN、NMTOKENS、Name、NCName、ID、IDREF、IDREFS、ENTITYといった文字列による派生型、integer、nonPositiveInteger、negativeInteger、long、int、short、byte、nonNegativeInteger、unsignedLong、unsignedInt、unsignedShort、unsignedByte、positiveIntegerといった数情報の派生型、yearMonthDuration、dayTimeDurationといった期間情報（duration）、dateTimeStampの日時情報といった膨大なの派生型を含むデータ型が挙げられている。

先に4.2節で述べた通り、型は一般に「値とその演算の集合」というのが現在の常識であるが、その観点からするとXML Schemaのデータ型は膨大である。それでも当初に比べると基本型が整備され多少は改善された。

最初の標準化仕様が固まった際に、XML Schemaの標準化関係者に、integerだけでも数種類決められている状態なので、「CORBAのIDL型に決めておけば幅広い言語へのマッピングが可能なので良いと思うが検討しなかったのか」と質問したが検討されなかった模様であった。その後、RELAXNGを村田真氏と共に提案したジェームス・クラーク氏と議論する機会がありこの問題について話し合った。その際クラーク氏は「XMISchemaのデータ型は禍であるが、CORBAのIDLにすれば発展の可能性があり」と語ってくれた。DOMの仕様はCORBAのIDLで記述されている。IDLで仕様を記述しておけば主なプログラミング言語とのAPIがマッピング可能だからである。とは言え、XML関係者でCORBAのIDLを知る人は少ない。

それにしてもXML Schemaのデータ型の用途や目的は不合理であった。データベース関係者からの要望で多数決で決まったが、村田真氏やジェームスクラーク氏の言うとおり型は文字と文字列だけの方がすっきりする。しかしながら、RDF SchemaはXML Schemaのデータ型を使う。基礎が不安定な状況でしっかりしたシステム構築は困難であろう。

セマンティックWebは、かなりあいまいな技術である。そのためにはデータ型は明確にしておく方が良い。というより、セマンティックWebの実装にプログラム言語の支援は絶対に必要なので、プログラム言語の扱うデータ型と、その引数データとしてのXMLの値は整合される必要がある。CORBAの言語マッピングはそのためには極めて有効な技術であるが、W3Cの関係者はそのあたりの見通しを持っていなかった。

6. テキスト文のデータ型

6.1 DTD、RELAXNG、XML Shemaの比較

図7に示した比較的単純な文書構成すら、DTD、RELAXNG、XML Schemaで記述すると煩雑になってしまうのは困ったものである。そもそも文書がそのような構成を取るようになったのは、人間の記憶や情報流通に便利なように木構造に基づく抽象性から具体性への関係付けで考えられたものであろう。その発想は、企業組織における社長、事業部長、部長、課長、係長、社員といった系列、軍隊における大将、中将、少将、大佐・・・兵卒といった階層序列と同様である。大量の情報を管理するには階層が必要である。

だが、大規模でない伝達情報の場合は単なるテキストであり、大規模な情報も要素的な情報はテキスト文で伝達される。そこで区切り付きのテキスト文（パラグラフ）を、DTD、RELAXNG、XML Schemaで記述し比較を試みる。

6.2 DTD

例えばテキストエディタで入力したテキスト文のデータ型は文字列（string）である。

```
<!DOCTYPE 文字列[
<!ELEMENT 文字列(#PCDATA)>
] >
```

以上で、単一のテキスト文書ファイルを意味する。しかしこれではタグを付ける意味はないので、パラグラフという要素が複数存在するパラグラフ群という要素を考える。

```
<!DOCTYPE パラグラフ群[
<!ELEMENT パラグラフ群(パラグラフ+)>
<!ELEMENT パラグラフ(#PCDATA)>
] >
```

パラグラフの繰り返しを+にしたのは、パラグラフが何もない文書はあり得ないと考えたからである。

6.3 RELAXNG:

以上のパラグラフと等価なRELAXNGの定義は下記のとおりである。

```
<xml version=”1.0” encoding=”UTF−8”>
<element name=”パラグラフ群”
        xmlns=”http://relaxng.org/ns/structure/0.9”>
  <oneOrMore>
    <element name=”パラグラフ”>
      <text/>
    </element>
  </oneOrMore>
</element>
```

DTDに比べると煩雑であり、木構造の定義を木構造記述言語で行わせるのは決して効率的ではない。木構造は、ルート、ノード及びリーフの3要素から構成されるので、その仕様を明確化すれば良い。DTDは、ルートをDOCTYPEで、ノードをELEMENTで、リーフを#PCDATAで記述しており、木構造の仕様としては妥当である。

6.4 XML Schema

XML Schemaの場合は、複合型（complexType）の概念を導入したために記法がさらに複雑になった。

```
<xml version=”1.0” encoding=”UTF−8”>
<xsd:schema xmlns:xsd=
            ”http://www.w3.org/2001/XMLSchema”
<xsd:element name=
            ”パラグラフ群” type=”パラグラフ群Type”/>
  <xsd:complexType name=
            ”パラグラフ群Type”
            minOccurs=”1”
            maxOccurs=”unbounded”>
      <xsd:element name=
            ”パラグラフ” type=”xsd:string”/>
    </xsd:complexType>
  </xs:element>
</xsd:schema>
```

型の概念は、ラッセルが考えたのは集合論的な一貫性を必要とする。それに対して、complexTypeは、個々のXML Schema定義の中でしか意味を持たない冗長なタグなので効率的ではない。このような背景で、RDFやOWLの枠組みも議論されるのでは不合理きわまりない。

6.5 InterleafのDTPの型

1980年代～90年代に大規模な文書システムの構築で使用されたInterleafのDTPシステム[9]の文書構造は、先のパラグラフ群による構造定義に近い。その基本論理構造を図9に示す。

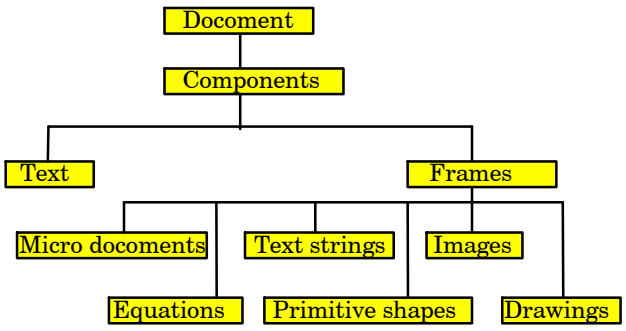

図9　Interleaf5文書の論理構造

先のDTD、RELAXNG、XML Schemaのパラグラフ群が図9の論理構造のComponentsに相当し、それらがTextとFramesで構成される。Framesは、SGMLでは外部実体として定義されたが、XMLの構造定義ではあまり議論されていない。3.1～5.9節において扱われた章、節、パラグラフといった階層区分は、論理構造ではComponentsの名称事例として、同じ階層で扱われる。章、節、パラグラフというような概念は、人間が区分する概念であり、コンピュータは関知する必要はないと考えたのであろう。これは卓越したアイデアである。そのように設計すると、論理構造とレイアウト構造の変換は容易になる。

さらにComponentsの管理に、属性情報の管理とポインター参照の概念を用いて、論理構造とレイアウト構造の間の参照や関係付けを効率化している。

要するに、現実のDTP文書の作成には、従来基本的な構成として考えられてきた図7のような論理構造は、人間の観念の産物であり、コンピュータの視点から見た文書の本質ではないという発想である。その妥当性も含め、文書の本質を考察する。

6.6 複合文書

1990年代に入り、多様な文書フォーマットを相互に変換するニーズが登場し、大規模文書向けのDTPシステムは主流の文書フォーマットを自システムのフォーマットに変換するツールの開発が必要とされた。文字コードのレベルは標準化されていたので大きな問題は無かったが、図形・画像に関しては各種CADツールのデータ形式が存在し、そのフィルター開発が課題であった。Interleaf社のDTPは、その変換をシステマティックにLisp言語で簡潔に行い、他社を差別化していた[10]。

OMGは、共通ファシリティの一環としてGUIを包含する複合文書を標準化項目とた。その内容は、ローカル又は遠隔のサーバ上の文字図形画像を含む編集可能な文書への操作画面をEditor、清書文書の操作画面をViewerとして、そのインタフェースを定めようとした[11]。その際にかな漢字変換についてもInputMethodとしてJapanSIGが提案した。しかしOMG複合文書に対抗して開発された類似機能のOLE（Object Linking & Embedding）がMS Wordに実装され、OMG複合文書は実現しなかった。

その後、2003年にW3CのCDFワーキンググループで複合文書の標準化が検討された。OMGが分散オブジェクトへのインタフェースとして複合文書の標準化を検討したのに対し、各種XMLフォーマットを統合する複合文書フォーマットの標準化であった。基本的にはXHTMLによる基本文書、図形・画像のSVG、数式のMathML、帳票のXFormsなどを統合するWebコンテンツであった[12]。ジャストシステムのxfyは、実装的には最も完成度が高かったが、Webコンテンツの標準がXHTMLからHTML5へと移行したためにCDFワーキンググループの複合文書は挫折した。しかしながら、HTML5の図形・画像フォーマットなど複合文書的機能に関しては、CDFワーキンググループでの検討結果が反映されている。

7. 考察

7.1 文書における型と演算

型の定義は、当初は値の集合であったのが、値の集合およびその上の演算の集合に改まったと記したが、それは文書に付いても適用可能であろうか。その検討を試みたい。

OMGのオブジェクトモデルにおける汎用モデルはオペレーションが演算インタフェースであるという代数学的な概念である。このインタフェースは、下記の式で表現され、オペレーションシグニチャと呼ばれていた。

$$\omega：（\chi_1:\sigma_1, \chi_2:\sigma_2, ・・・ \chi_n:\sigma_n）\\ \rightarrow（y_1:\rho_1, y_2:\rho_2, ・・・ y_m:\rho_m）$$

ただし、ωは操作の名称、χは型σのリクエストパラメータ、yは型ρの結果のパラメータである。パラメータ数nは1以上、mは0以上、すなわち結果が出力されない場合も含むのである。このオペレーションシグニチャは、4.2節の

$$y:\rho = f(x:\sigma)$$

の関係に包含される。論理構造からレイアウト構造への変換が文書にとって重要な操作であることは疑う余地が無い。それはSGMLの論理構造仕様とDSSSLのレイアウト構造仕様の関係、同様にしてXMLとXSLの仕様の重要性にも象徴される。

7.2 InterleafのDTP

だが、6.5節で説明したInterleaf5やInterleaf6によるコンポーネントによるレイアウト制御は、論理構造とレイアウト構造という発想ではない。論理構造はコンポーネントのリストであり、コンポーネントの属性コントロールによってレイアウトを変換する。マージンやコラム数のような文書全体の属性も要因としては存在するが、きめ細かいレイアウト管理はコンポーネントによってなされるのである。DTDで記述すると下記のようになる。

```
<!DOCTYPE Interleaf−Document[
<!ELEMENT Interleaf−Document (Component+)>
<!ELEMENT Component (#PCDATA)>
] >
```

図6のピラミッド的なDTDと比べると文鎮のような単純極まのりない構造である。この構造の単純化により処理の高速化とオートナンバー、オートリファレンスといった章、節、項や図、表の番号付けの自動化、目次・索引の自動作成などの機能を実現している。肩すかしの様な印象を持つが、レイアウト操作を文書の本質と考えるなら、章・節・項といった論理階層は不必要である。論理構造は文書の意味を反映するから価値があり、レイアウト化された表示媒体を得るだけならば、それは不要である。

7.3 セマンティックWebの課題

セマンティックWebはその階層モデルを考えれば分かるとおり意味的論理構造の上位階層への延長でありオントロジ層以上の検討を期待したいところである。OWLが標準化されて10年余りになるが、進展は止まっている。その理由は標準的な処理系が誕生しないことにあるが、ゲーデルの不完全性定理に基づく、UpperOntologyの本質的な問題も存在する。実用的な処理系とドメイン・オントロジの開発普及が妥当なアプローチと思われるが、ビッグデータの扱いとの整合性はデータベース・アプリケーションに比べると課題が多い。UMLに基づく実装において、クラス図をOWLで扱うことが出来れば興味深いと思われるが、適用範囲としてのドメインの選択が課題であろう。openEHRが、電子カルテの世界でオントロジ活用を展開しているが[13]、それは医療・介護に特化しているからである。とは言え必ずしも普及は進展していない。

8. 結言

以上の総括としての結言を述べる。文書構造のカテゴリには、意味を象徴する論理構造、視覚的な対象としてのレイアウト構造、さらにレイアウト構造を固定させた清書構造が存在し、ODA、SGMLの仕様化に反映されたが、XMLでは清書構造は省かれた。文書は、複合的な型を包含する構造と考えることが可能で、意味概念的な認識の反映である論理構造の型集合から視覚的認識のためのレイアウト構造の型集合への演算が文書の作成・編集に相当する。他方、WYSIWYGのGUIによるDTPツールは、コンポーネントとしての簡単な論理構造を持つだけで、高度の論理構造を不要としていた。セマンティックWebは高度の論理構造の将来展望モデルであるが、その構造に関連する型の集合と処理系のプログラミング言語のデータ型の整合が課題である。他方、ゲーデルの不完全性定理との整合も重要で、先ずはその解決が必要である。

9. おわりに

最近個人的には専門技術自体よりも技術スキルや技能を背景とする人材育成や地域社会の活性化に関心を持っている。今後の専門家は、特定分野のスペシャリストであるよりはリベラルアーツを背景に持つオールラウンドなプロフェッショナルとしての素養を必要とするとの指摘があり[14]、職業能力開発総合大学校ではその観点から技能科学（PolytechnicScience）を指向する取り組みが行われている[15]。私もささやかながらそのお手伝いをしているが、そのような人材にとって文書関係の素養、特に執筆力、情報発信力は必須であろう[16]。これは中世のリベラルアーツの自由七科（文法・論理・修辞・算術・幾何・天文・音楽）における文法・論理・修辞に相当するものである。今後DC研には、高等教育を通じてそのようなスキルを若手の技術者、研究者に伝承すると共に、企業や官庁のスペシャリスト人材にオールラウンドな素養を培わせ、若手人材の技能スキルの底上げの取り組みを期待したい。先端技術に関しては、GoogleやAmazonのビッグデータを統計的・効率的に扱うハイレベルな取り組みに太刀打ちできる状況とは思えないからである。そのためには、文書のページをめくるだけでなく、そのページ間の間隙から拡張された文脈を把握して発想する能力を期待したいと考える。それを地層に潜む化石や地磁気の記録などの情報にまでメタファー的に結びつけ得るような発想力を期待したい。信学会のLOIS研ならびに情処学会のDC研に参加する若手研究者の奮起を期待したい。

文献

[1] 大野邦夫, 吉田正人；”文書を構成する型についての一考察”,情報処理学会研究報告, DD22−1（2000.3）

[2] 大野邦夫, 梶原俊男;”地磁気逆転地層をコミュニケーション媒体とする地域活性化の検討”, 情報処理学会研究報告, DC104−5（2017.3）

[3] 梶原俊男,大野邦夫;”地磁気逆転地層理解に関するバーチャルミュージアムの構成に関する検討”,情報処理学会研究報告, DC108−2（2018.3）

[4] 松本充司, 鈴木良太, 菱山和利； “テレマティクス通信”, 電子通信学会（1990）

[5] B. Russell; “The Principles of Mathematics”, Routledge, London (1903)

[6] B. Russell; “My Philosophical Development”, Allen & Unwin（1959）

[7] J.C.Cleaveland（小林光夫訳）; “データ型序説”, 共立出版（1990）

[8] OMG；”The Common Object Request Broker: Architecture and Specification” OMG Document, No. 92.12.1, (1992.12)

[9] 大石進; “オーバービューオブInterleaf5−Part1”, SuperASCII, Vol.3, No.10（1992.10）

[10] 大石進;“オーバービューオブInterleaf5−Part2”, SuperASCII, Vol.3, No.11（1992.11）

[11] 大野邦夫; “OMGのコンパウンドドキュメント標準”, Object World Expo / Tokyo 1995シンポジウム講演資料, C14（1995.11）

[12] 大野邦夫；”複合ドキュメントの進展とxfy”, 画像電子学会第15回VMA研究会資料(2005.7.8)

[13] 大野邦夫；”個人の情報環境へのオントロジ適用の検討”,情報処理学会研究報告, DD88−1（2013.1）

[14] 高橋俊介; ”21世紀のキャリア論”,東洋経済新報社 (2012.4)

[15] PTU技能科学研究会； “技能科学入門: ものづくりの技能を科学する”, 日科技連（2018.2）

[16] 大野邦夫, 西口美津子, 芥川一則；”グローバル企業の文書管理と企業文化に関する検討〜異文化コミュニケーションと人材育成へのドキュメント文化の役割”, 情報処理学会研究報告, IFAT122−4/DC101−4（2016.3）

ⓒ 2018 Information Processing Society of Japan