Vol.2018-IFAT-130 No.1
Vol.2018-DC-108 No.1
2018/3/27

コンピュータ言語環境における図形・画像情報の役割への考察

大野 邦夫†

情報メディアとしての言語の役割は図形・画像メディアの役割とは異なるが、双方のメディアは人間の知識を補完する面がある。 この概念をコンピュータ環境におけるプログラミング言語とマークアップ言語にまで拡張することを試みた。 最初のステップとして、私が半世紀の間に経験した30余のコンピュータ言語を総括して図形・画像の役割についての考察を試みた。その結果として、GUI環境の進展、MVCメタファ、位相平面による図解手法などが抽出された。併せて、スマホにおけるアイコン活用の枠組みや、図形素片を組み合わせてプログラムにするスクラッチのような言語の可能性を展望した。

A study on the Role of Graphics and Images to Computer Language Environment

Kunio OHNO†

The role of language media is different from that of the graphics / image media in view of human knowledge. Then both media complement human knowledge. A trial to extend the concept to computer environment of programming language and markup language has been conducted. For the first step, a summary of computer languages which I have experienced during half century has been made in this paper. The roles of graphics / image have been discussed, which includes GUI environment development, Model View Control metaphor, and phase plane diagram analysis, etc. Then finally, the icon application framework on smart phone displays, and possibility of Scratch language, which combines and connects graphic segments to computer programs has been described.

1. はじめに

人間における情報メディアの進展と、コンピュータにおける情報メディアの進展は表1に示すように対称的であり、以前その背景的な理由の考察を行うと共に[1]、最近は人間の言語獲得のプロセスの考察、自然言語とプログラミング言語との対比[2]、言語獲得における図形・画像情報の役割[3]、画像情報が異文化コミュニケーションに果たす役割[4]などを検討してきた。本報告ではコンピュータ言語環境における図形・画像情報の役割についての検討を行う。

表1　人間とコンピュータにおける
情報メディアの対称性

コミュニケーションメディアの歴史	コンピュータメディアの歴史
叫び声，ジェスチャ(先史以前)	映像，音声(2000s)：stream
洞窟壁画(BC.30,000~)	図形，画像(1990s):class
象形文字(BC. 5,000~)	GUI(1980s)：class
表意文字(BC. 3,000~)	漢字処理(1970s):char, string
表音文字(BC. 1,500~)	英数字(1960s):char, string
数式 代数学(AD. 1,000)	数字，数式計算(1950s):int,float
近代論理学(AD. 1,800)	2進論理(1940s):boolean

本テーマの発端は、プログラミング言語において図形素片情報を活用するスクラッチ言語の普及が挙げられる。スクラッチ言語は小学生に対するプログラミング教育の導入の観点で注目されているが、この言語の持つインパクトは、この言語で育った児童たちが社会で活動する状況になるまでは未知数である。だがそれを考察し予測することは重要である。

類似のインパクトは以前にも存在した。Xerox PARC（Palo Alto Research Center）におけるGUI（Graphical User Interface）の実用化を通じたパーソナルコンピュータ（PC）の発明である。アラン・ケイを中心とするPARCのコンピュータ文化が、PCの普及とハイパーリンクによるWebの普及を通じて学術用であったインターネットの商用化を行わせ、今日のIT文化を形成したと言える。

スクラッチのような、図形素片情報を活用するプログラミング言語が今後のIT文化にどのような影響を及ぼすかを考察するのは興味深いことであるが、そのためにはPARCによるGUIの実用化を中心とする過去の歴史を振り返り考察することが参考になるように思われる。そのように考え、コンピュータ言語環境における図形・画像情報に関して過去の歴史を振り返りつつ、今後の可能性を展望することを試みる。

2. プログラミング環境と図形・画像情報の活用経緯

2.1 装置環境としての進展

2.1.1 Fortran, BASICによる計算処理

表1のコンピュータにおける情報メディアの進展は、2進論理（1940s）、数字・数式計算（1950s）、英数字（1960s）、漢字処理（1970s）、GUI（1980s）、図形・画像（1990s）、映像・音声（2000s）という経緯を辿っているが、プログラミング言語の進展もそれと並行しているように感じられる。先ずその確認を行いたいと思うが、一般論として検証するのは難しいので、個人的な経験を紹介する。

私が最初にコンピュータを使用したのは、1960年代末の学生時代にFortranを学んだことに端を発する。その後1970年にNTTの電気通信研究所に入所し、クロスバ交換機の接点消耗の研究に携わった。この時に接点アーク特性、磁気回路負荷のような非線形回路を微分方程式で解析するFortranプログラムを作成し、接点間電圧と回路電流から放電エネルギを算出し、これに基づく接点寿命予測システムを開発した[5]。その電流と電圧の変化をXYプロッタに出力したのがプログラミング言語と図形情報に関係した最初の経験であった。その後、A/D変換装置を使用して接点放電現象をミニコンに取り込み、その波形から放電時間を算出するシステム[6]を構築したが、A/D変換装置からパラレルインタフェースでミニコンに取り込むために、アセンブラ言語を学んだ。1970年代半ばにBASIC言語によるマイコンが誕生し、その後NECのPC98上のN88BASICで図形・画像処理を行うようになった。BASIC言語は、コンパイ

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

ラ言語のFortranをインタプリータ式で扱いやすくした言語であるが、図形をピクセルベースの画像平面に描画可能であった。なお画面への手入力が必要となるのでマウスが必要となった。BASIC言語による描画機能は、ゲームなどに使用され、描画、塗りつぶし、スキャナやビデオカメラからの画像入力、さらに2次元の画像処理にまで応用が進んだ。

2.1.2 高級言語マシンとGUI環境

1980年代に入ると、Xerox PARCによるSmalltalk80が登場した。この言語はオブジェクト指向プログラミングパラダイムの言語であったが、ウインドウシステムを用いたGUIが革新的であった。GUIは、従来のプリンタやプロッタを模擬したアプリケーションではなく、図形・画像情報をコンピュータとの対話に用いるという画期的な環境を提供した。このアプローチは、MVCメタファやダイレクトマニピュレーション（直接操作）と呼ばれるユーザインタフェース設計の思想を創造した[7]。

Smalltalkの場合は、言語専用のOS環境を包含したが、その後類似のGUI言語環境のOSが続々と登場した。Pascal環境のPERQ、Modula−2環境のLilithといった高級言語マシンが登場したが、最も実用的に使用されたのはLisp環境の高級言語マシン、Lispマシン群であった。NTTもLispマシンELISを開発し、私はその開発と販売、サポートに携わった。高級言語マシンは、当初はビットスライスCPUと作業記憶のWCS（Writable Control Storage）を用い、高級言語の命令セットをマイクロコードでファームウエアとして実装するアーキテクチャのコンピュータで、Xeroxの高級言語マシンはWCSを書き換えることにより各種の高級言語のサポートを可能とした。従ってSmalltalkのみならず、Interlisp−DやMesaといった高級言語もサポートした。オフィス向けのDTP（Desk Top Publishing）アプリケーションを実装したMesaマシンは、Starワークステーションとして商品化された。

最高の知性を要求されると思われる高級言語マシンが優れたGUIを提供した背景には、論理的・記述的・知性的な分野で創造的な仕事を行うには、対向的な感性的な支援が要求されると考えるのが妥当であろう。人間の営みは常に総合的であり、コンピュータ環境はそれをバランス良く支援することが重要と考えられる。

2.1.3 GUI環境の標準化の試み

ELISの開発に携わる前にMIT系のLispマシンSymbolics3600を使用する機会があったが、それはGUIを含むヒューマンインタフェースの研究のためであった。SymbolicsのGUIを活用して、図1のような5つのボタンを活用するFive−keyマウスの評価を行ってみた[8]。Five−keyマウスは、スイス連邦工科大学のニーバゲルト教授とNTTの共同研究の産物であったが、その当時使用され始めていたウインドウシステムの機能向上と標準化を試みるものであった。Five−keyマウスは、Modula−2マシンLilithのOSであるXS−1のウインドウシステムを想定して検討したデバイスで親指、人差指、中指、薬指、小指に対して、Move, Pick, Menu, Undo, Quitという機能を割付けたものであった。GUIによるコンピュータとの対話のためには、マウスとキーボードが基本であったが、より一般的な画像入力デバイスへの進展が予想され、そのための論理デバイスが検討されていた。選択を指定するPick、位置を示すLocator、On−Offを指示するButton、アナログ的な値を入力するValuator等である。文字入力のKeyboardも論理デバイスの一つであった。なお、Five−keyマウスのボタンは、上記論理デバイスではなくGUIとしてのコンピュータとの対話を目指す概念であった。

さらにXS−1のGUIには、Site、Mode、Trailというサブウインドウ環境が具備されており、各々、データ環境、操作環境、履歴環境を利用者に提示した。XS−1はLilith上の試作システムであり実用にはならなかったが、Site、Mode、Trailのコンセプトはその後GUIを検討・評価する上で参考になった。類似の試みをSymbolics上で行う予定であったが、ELISの商品化のために中断し、ELIS完成後にELIS上で具体化する予定であったがその機会は訪れなかった。

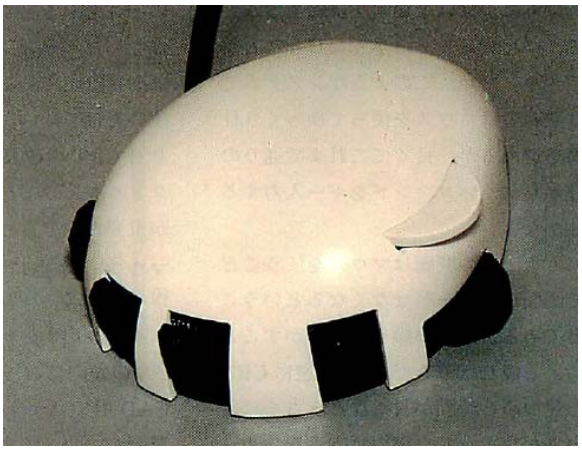

図1 Five−keyマウスの外観写真

2.2 端末上の知的アプリケーション展開

2.2.1 エキスパートシステム開発環境

Lispマシンは、エキスパートシステム構築のプラットフォームとして1980年代の前半に登場した。SymbolicsはXeroxの高級言語マシンと同様にWCS上でZetalispの関数群をファームウエアとして実装したが、データの識別のためのタグアーキテクチャ並びに高速処理のためのハードウエアスタックを強化し、Lispの高性能化を実現していた。NTTのELISは、Symbolicsが特徴としたハードウエアスタックとタグアーキテクチャをさらに大幅に強化すると共に、独自の特徴であるメモリ汎用レジスタにより、Lisp処理におけるCAR部とCDR部の一括同時処理を可能とし、Symbolicsを凌駕する性能を実現した。

1980年代後半は、Lispマシンをプラットフォームとし、フレームによる知識ベースとプロダクションシステムによる推論エンジンを用いるエキスパートシステムが多様な分野で開発された。その専用構築ツールとして、GUIをベースとするKEEやART等のエキスパートシェルが提供され話題になった。その後Lisp言語は、CommonLispによる標準化が進展し、CommomLispの仕様が汎用CPU上のコンパイラを指向したために、Lispマシンのメリットは薄れ、一世を風靡した高級言語マシンは衰退の一途を辿った。

その影響もあって、GUI環境は高級言語マシンの環境から汎用OSの環境に移行したが、当初はUnix系ワークステーションとAppleのLisa、Macintoshが採用した。さらにUnixワークステーションにおけるネットワーク透過のウインドウシステムとして、X−windowが標準化されたが、その背景として、通信プロトコルとしてのTCP/IP、遠隔手続呼出としてのRPC、遠隔ファイルシステムとしてのNFSが標準化されていた。これらの技術に基づいてインターネットが構築された。

2.2.2 ロジックプログラミング環境

Lispマシンが商品としての高級言語マシンとして販売されていた頃にロジックプログラミング言語のPrologによるアプリケーションが検討された。その端緒は、言うまでもなく第五世代コンピュータプロジェクトであったが、Prologはエキスパートシステムにおけるプロダクションシステムと類似のパターン照合に特化した言語であった。NTTのLispマシンELIS上のTAOは、SymbolicsのZetalispと類似のオブジェクト指向パラダイムを包含し、さらにS式レベルでPrologのパターン照合機能を包含するマルチパラダイム言語であった。しかしながらパターン照合による論理的な制約とオブジェクト指向によるクラ

ス継承を統合するための実用的なデモ事例を見つけるのが困難で、アルゴリズムの実装はできたものの実用性は乏しかったが、一つの要因は適合するGUI支援環境ができなかったことが背景にあると思われる。

2.2.3 DTPシステムの発展

DTPシステムは、先に紹介した通りXeroxのStarワークステーションがその発端である。これはウインドウシステム技術を文字・図形・画像という個別の情報メディアをビットマップ画面上で融合させて実現した。印刷出版業界は、既に文字・図形・画像を見やすく分かりやすくレイアウトする技術に習熟していたので、DTPはそのレイアウト規則をアルゴリズム化しプログラム言語で実装した製品と言える。DTPの特徴として、WYSIWYG（What You See Is What You Get）方式が挙げられるが、ディスプレイ上の表現がそのまま印刷された紙になるという機能は非常に分かりやすく、この製品の市場を広げるために有効であった。Xeroxという複写機メーカーがDTPシステムを実現したのは企業文化的な必然性があったと言えるであろう。

文字・図形・画像編集のアルゴリズムは興味深い。一般の文書は、開始点から終端点までの一次元文字列を基本に構成される。文字列の区切りとして、章、節、項、文節、文、語、文字といった階層がある。文字列以外の要素としては、図、表、数式、注などの情報があり、これらは一次元の文字列にぶら下がった情報として位置付けられる。この体系を論理構造と呼ぶ。この構造は、文書の目次に対応する木構造と考えることも可能であり、最初に述べた一次元の文字列は、木構造を深さ方向にトラバースして得られる文字列である。他方、文書のレイアウトはページ、コラム、行、語、文字という階層があり、その構造に従う形式で文書はレイアウトされる。この構造をレイアウト構造と呼ぶ。

文書作成編集処理は、論理構造をレイアウト構造に変換する処理を含み、nroffやTeXのような当初のマークアップ言語は、論理構造の中にレイアウトコマンドを明示的に付加する言語であった。nroffの出力は文字だけであるが、TeXは図形や数式を含むのでデバイス独立なdviファイルを生成した後にpdfやPostScriptに変換したり個別の印刷デバイスに渡される。dviファイルを実時間的に表示するのがWYSIWYGであるが、きめ細かいレイアウトの文書、複雑で大規模な文書であるほどWYSIWYG機能は要求される。これも高級言語マシンと同様に知的な作業が優れたGUIを要求することに通じるであろう。

その後、汎用マークアップ言語としてのSGML（Standard Generalized Markup Language）が導入された後は、マークアップ言語はレイアウトコマンドではなく、文書の論理構造の要素にレイアウト属性を付与するようになり、マークアップ言語の世界は進展したが、マークアップ言語とWYSIWYGとの共存は難しくなった。

2.3 MVCメタファの導入

2.3.1 MVCメタファ

DTPシステムは、論理構造をレイアウト構造に変換する処理を行うが、これは先に述べたMVCメタファを拡張して考えることが可能であろう。

文書の作成、修正、追加、削除といった編集作業は文書の構造モデルとしての論理構造をベースに行われる。ところが編集作業は、レイアウト構造として画面表示された位置で操作（Control）が行なわれ、それが論理構造（Model）に反映され、反映された結果を再度レイアウト構造に変換して表示する（View）という手間のかかる処理ステップを必要とする。GUI処理を洗練させるには、内包するMVCメタファに配慮する必要がある。

DTPは、AppleのLisa、Macintoshの主要なアプリケーションであったが、CPUとして用いたモトローラの68000はLisaのためには性能が不十分であった。そのためにMacintoshでは苦肉の策として画面サイズを縮小し画面応答を高速化して実用性を確保した。その結果は功を奏しMacintoshによるDTP事業は、印刷出版業界のビジネスモデル、雇用形態を変革し、シリコンバレーの存在感を著しく高めた。Starは高級言語マシンのアプリケーションであったが、その後性能が向上したUnixワークステーション上のDTPシステムが登場し、技術文書を中心とする大規模な文書システムの開発や維持管理に使用された。そのようなDTPシステムの代表的な製品であるInterleaf社のInterleaf5は、カスタマイズ用の開発言語であるInterleaf−lispを提供していた[9]。この言語は、プログラミング環境がDTPシステムそのもので、文書上のメニュー、アイコン、イベントなどをトリガーにしてアプリケーションの実行が可能であった。そのようなアプリケーションとしてSGMLツールキットがあり、ISO標準のSGMLでタグ付けされた文書のDTD（Document Type Definition）に基づく編集や管理を実現していた。

2.3.2 文書における型と操作

先に文書の論理構造をMVCメタファにおけるモデルとして位置付けたが、DTDはモデルとしての文書の属性を階層的に定義する枠組み（型）である。型はバートランド・ラッセルが、集合定義の際に生じ得るパラドックスの回避の必要性に基づき考案した論理的な帰結で、命題の集合的概念であるが、計算機科学では命題に代わって値と演算（操作）の集合として位置付けている。

MVCメタファにおけるDTDの位置づけは、文書の論理構造モデルをメタな段階で規定する仕様であるが、それをレイアウトされた表現に変換する操作、すなわちDTP処理の要件としての枠組がが考えられる。その機能もISOでDSSSL（Document Style Semantics and Specification Language）として標準化されている。SGMLのDTDのインスタンス文書にDSSSLに基づくレイアウト属性の値を引数として与えると、ページ整形された清書の文書が得られることになる。清書文書はISOではSPDL（Standard Page Description Language）として位置付けられたが、PostScriptがデファクト標準となったので具体化しなかった。以上から、文書構造的にMVCメタファをマクロ的視点で考察すると、ModelはSGML、ViewはSPDL、ControlはDSSSLに相当することになるであろう。しかしながら、DSSSLは、DTPのように実時間的に操作するものではないので、ミクロ的にはControlとは言えない。

2.3.3 オブジェクト指向パラダイムの進展

Interleaf −lispは、MVCメタファに基づくオブジェクト指向パラダイムをサポートし、Interleaf5は徹底したオブジェクト指向プログラミング手法で構築されていた[9]。ウインドウシステムはもちろん、文書の論理構造もレイアウト構造も木構造のクラスにより定義され、木構造をトラバースするメソッド（XMLのDOMに相当）により編集操作が実現されていた。要するにオブジェクト指向は、具体的な事物をモデルとして模擬するようなプログラミングに適合する。文書環境もその思想で構築され、デスクトップには、文書を格納するフォルダ、ドロワ、キャビネットといった階層的な道具立てまで整備されていた。

物理的なシステム、例えば運動方程式の解析や電気回路の解析などは、オブジェクト指向でモデル化してエレガントに解くことが可能である[10]。FortranやBASICで解析した接点放電現象もSymbolicsのZetalispのフレーバを用いてモデル化し、放電現象の分類をプロダクションシステムで行わせて、その後に微分方程式の数値計算を行わせ、電圧電流波形を表示して接点寿命を行うエキスパートシステム的な構成にして接点国際会議で紹介した[11]。その後このシステムをELISのTAOに移植したところ、計算性能は劇的に向上した。これはTAOのeval関数が通常のLisp処理系ではエラーになる部分にオブジェクト指向のメッセージ処理を組み込んでいるためにLispの性能を全く損なうことなくオブジェクト指向を実現したためであった。メッセージ演算のsend関数をfuncall関数に展開して処理する

Zetalispに比べオーバーヘッドが殆ど生じないためである。アーク特性と摩擦特性の非線形性の類似、力学系の2階微分方程式の位相平面によるトラジェクトリ描画を通じた解析シミュレーションは、非線形微分方程式の数値解法としては視覚的に把握できるので分かり易かった[12]。図2は、摩擦振動における振動物体の位置（x軸）と速度（y軸）に関する位相平面のトラジェクトリを示している。

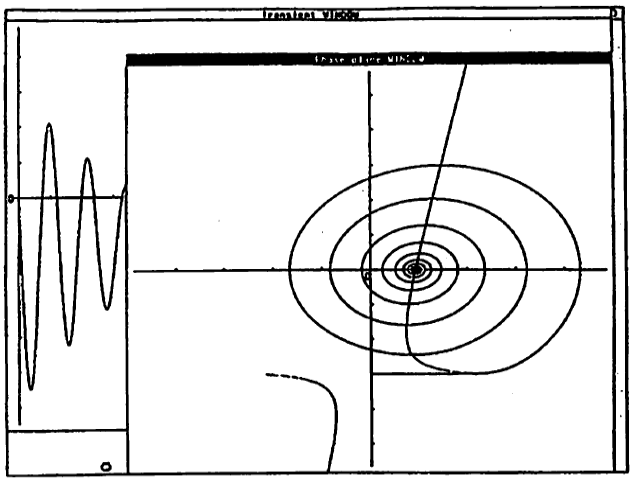

図2　ELISウインドウシステムによる位相平面上の摩擦振動消滅時のトラジェクトリと摩擦特性の表示

上記の延長として多次元位相空間（ヒルベルト空間）における1階ベクトル微分方程式の解、線形微分方程式のサブクラスにおける非線形項の追加による非線形微分方程式の解法、インスタンス変数の無次元化など、従来の非線形微分法的式の解法にオブジェクト指向を適用してその有効性を確認した。ヒルベルト空間の発想は関数空間の概念で、論理的・数学的なものであるが、その第一歩は位相平面における点が複合的な要素の値を有するという発想である。それを多次元に拡大することにより、多様な属性を有する状態点の概念を物理的意味を伴って思い浮かべることが可能になり、それを変換する発想が生まれる。論理構造からレイアウト構造への文書の変換もその発想の一環と捉えることが可能である[13]。

さらに種々の複合的なモデルをオブジェクト分析設計を通じてクラス図を作成し、それをTAO又はCLOSによるクラス定義に移行し実際のシミュレーションを行う手法などを考案すると共にその可能性を検討した[14]。

Lispがオブジェクト拡張されたのと同様にUnixで用いられていたC言語もオブジェクト指向拡張されてC++となった。他方AppleはLisaの開発の頃からOSなどのシステム記述にC言語を別の手法でオブジェクト指向拡張したObjective−Cを用いていた。さらにマイクロソフトは、C++ライクなC#を開発しシステム記述に用い始めていた。だがC++、Objective−C、C#は、クラス継承のメカニズムに相違がある。C++が多重継承なのに対して、Objective−C、C#は単純継承である。単純継承でありかつC言語の拡張なのでObjective−CとC#は似てはいると思われるかもしれないが、表記は全く異なることから3者3様の言語となり現在も使用され続けている。

2.4 マークアップ言語による進展

2.4.1　HTML

PCやワークステーションのプログラミング環境がウインドウシステムとなり、そのウインドウがDTPに拡張され、文字・図形・画像を複合的に包含する文書、すなわち複合文書の作成環境へと発展したのであるが、その傍流の技術として、HTML言語によるWebブラウザが1989年に誕生した。HTMLはSGMLのDTDの一例であり、その観点では、技術文書用のDTDであるDocBookやLeafDoc等と同様な位置づけであるが構造的には数種類の文字サイズの文字列と画像のみから構成される単純な文書の枠組みであった。但しハイパーリンクを通じて他のHTML文書を逐次参照することを可能としていたので、これが情報探索や収集という知的活動を強力に支援する枠組みとなりかつその仕様が無料で公開され無料のツールが提供されたことにより一気に普及した。

HTMLが進展した背景には、ネットワーク経由のMVCメタファの拡張が挙げられる。TCP/IPによるインターネットは、プログラミング環境におけるネットワーク透過性を要求し、先に述べたX−Window以外にRPC（Remote Procedure Call）やNFS（Network File System）を実現させた。このような技術を背景によりアプリケーションに近いレベルのプログラミング言語から独立したMVCにおけるViewとしての表示ツールが要求され、それがWebブラウザとなったと言えるであろう。HTMLはその属性と値を提供するための基本的な記述言語である。

2.4.2　クライアント・サーバ方式

HTMLによるWebブラウザが表示ツールとしての事実上の標準となったことにより、ネットワーク環境やサービスは大きく変化し始めた。従来、サーバはファイルサーバ、プリントサーバ、DBサーバのようなアプリケーションの補助的な位置付けであったのだが、Webブラウザに対するWebサーバやEメールのためのSMTPサーバが重要な役割を担うようになった。さらに種々のアプリケーションサービスは、ミドルウエアとしてサーバーのフロントエンドで処理されるようになり、その背後にDBを擁する三層クライアント・サーバ方式がサービス構築の基本的な枠組みとなった。その当時、ILLUSTRAというマルチメディアデータベースが販売されてので、マルチメディアコンテンツをSGMLの枠組みで作成し、Webブラウザで公開するシステムを試作したことがある（図3）。

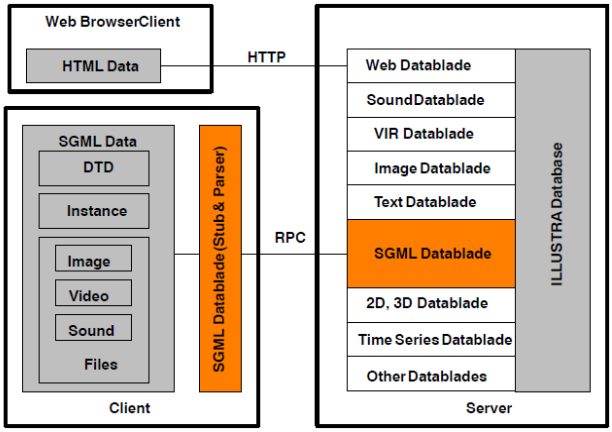

図3　ILLUSTRAによるSGMLマルチメディアコンテンツ作成管理システム

ILLUSTRAは、ミドルウエアとして位置付けられる各種の処理モジュールをデータブレードとして位置付けて管理する枠組みを提供し、それらのミドルウエアを相互に活用して新規のデータブレードを開発することが可能であった。

クライアントには、パノラマというSGMLエディタのGU環境を用い、それに対して実体参照でマルチメディアをILLUSTRAのデータブレードから取得する機能を実現したが、ILLUSTRAがInformixに買収されたために試作のみに終わった[15]。MVCメタファのマクロな視点で考えると、ILLUSTRAによるORDBとミドルウエアがModelを構成し、ViewはWebブラウザ、ControlがSGMLエディタに対応する。その後ILLUSTRAの代わりに通常のRDBのOracleとし、文字情報以外はファイルでマルチメディアを管理する現実的なシステムになったが[16]、新規性と商品性の両立は難しいと痛感した。

2.4.3 Java

インターネットの普及に伴い、ネットワークにおけるサービス記述を指向したJava言語がC言語に代わってネットワークサービスの中核的言語として位置付けられ、それを包含したWebブラウザとしてHotJavaが出現した。C言語は、OS環境であるUnixのウインドウシステムとしてX−Windowをライブラリ（xlib）として持っていたのに対して、Javaはその表示環境としてWebブラウザにフォーカスし、カスタマイズを通じた表現力の向上を目指したと言える。その結果、Webホームページは動的機能を包含する新たな展開を迎えた。さらにJavaよりも扱いやすいスクリプト系の言語でカスタマイズするDHTMLやVRMLによる新規ブラウザが誕生し、それらの相互運用性が問題として浮上した。その解決案として誕生したのがXMLであった。

2.4.4 XML

HTMLがネットワーク透過のMVCメタファにおけるViewのための言語であったのに対し、XMLはプログラミング言語から独立したModelのための言語である。1997年から翌年にかけてXMLの規格が制定されると共に関連規格としてのDOM（Document Object Model）、RDF（Resource Description Framework）、XLink、XML Schema等が整備された。さらにWebサービスやセマンティックWebのようなXMLをプラットフォームとするアプリケーション構築の枠組みも進展した。その後大規模なシステムはデータセンターのサーバ上に構築されるようになり、システムのクラウド環境への移行が一般化した。

XMLがSGMLをリファインした記述言語として標準化され、さらにDSSSLに相当するレイアウト記述言語としてXSL（Extensible Stylesheet Language）が議論された。当初の議論の結果、構造変換のXSLT（XSL Transformation）とレイアウト実行のXSL−FO（XSL Formatting Object）の2段階に分けることになり、先ずXSLTが具体的に標準化され、XML記述をテキストや表でWebに公開するために活用された。他方、XSL−FOの方はかなり難産であった。その理由としては文書レイアウト規格としてのCSS（Cascading Style Sheet）の存在が挙げられる。XSL−FOとCSSとが同様の機能に対して若干異なる仕様を定めているからである。

XMLとXSLの関係は、SGMLとDSSSLの関係に対応するので、マクロ的なMVCメタファに対応付ける場合は、XMLがModelでXSLの処理がControlに相当する。なおViewは、ControlがXSLTであればHTMLに対応し、XSL−FOであればPDFに対応することになる。

2.4.5 オントロジ言語

オントロジという概念は、エキスパートシステムの知識ベースに端を発している。プロダクションシステムの推論機構におけるルール規則が浅い知識として位置付けられ、フレームによる語彙の意味属性的な概念構造が深い知識として位置付けられたが、オントロジは後者の深い知識の構造に端を発する。この分野の応用が試みられたのは、エージェントシステムであった。この分野の言語としては、KQML（Knowledge Query and Manipulation Language）とKIF（Knowledge Interchange Format）が挙げられる。これらの言語は、CLOS（Common Lisp Object System）をベースとするクラスライブラリのシステムであるが、S式レベルの論理変数の活用を可能にしていた。従ってその発想としてはNTTのLispマシンELIS上のTAOの言語仕様に近い。KQMLとKIFは、孤立した言語ではなくエージェントとしての言語処理系が対話して問題解決するための言語である。この対話環境は、言語行為論（Speech Act Theory）に基づくものであるが、言語行為の代わりに通信行為（Communicative Act）という概念を構築していた[17]。KQMLはその後FIPA（Foundation for Intelligent Physical Agents）のACL（Agent Communication Language）に発展し、オントロジ記述はS式からXMLやRDFに拡張された[18]。FIPAのACLのオントロジ記述は、DAML（DARPA Agent Markup Language）、OIL（Ontology InferenceLayer）から、両者を統合したDAML+OILへと移行したが、W3Cは、DAML+OILをOWL（Web Ontology Language）として標準化した。

2.4.6 xfy

ジャストシステムでは一太郎Arkという一太郎の文書編集機能をPureJavaで記述した製品を保有していたが、その実装をXML対応とすることによりW3Cの複合文書とすることを検討した。W3Cの複合文書は私もメンバーであったCDF−WGで検討されたが、SVG, MathML, XForms, SMILなどのXMLコンテンツをXHTMLに包含させることにより、HTMLによるWeb画面をDTPによるレイアウトされた文書と同様に、文字・図形・画像を自由に包含するXHTML文書とし、さらに数式や帳票まで包含させようとしていた。その機能実現のために開発されたジャストシステムのxfyは優れたプラットフォームであった[19]。

システムとしてのxfyの機能を一言で表現するならば双方向化されたXSLTである。XSLTがソースのXMLからデスティネーションのXML（主にGUI画面としてのXHTML）への一方向の変換しか行わないのに対してxfyはデスティネーションの画面を通じてソースのXMLを操作する。もちろんXSLTと同様にソースからデスティネーションへの変化もサポートしている。その双方向変換を可能とする言語がXVCD（XML Vocabulary Connection Descriptor）でxfyの技術の中核であった。要するにXMLデータとGUIの関係を実時間で編集操作可能なWYSIWYG方式で実装したのであった。2.2.3項でTeXのdviファイルとWYSIWYGについて述べたが、xfyはSGML以降のマークアップ言語の処理で、WYSIWYGを実現させた希少な注目すべき例であった。

2.3.1項で、DSSSLはModelとしてのSGMLに対してマクロ的にはControlと言えるがミクロ的には言えないと述べたが、これはModelとしてのXMLに対してXSLTのControlについても同様であった。しかし、XVCDはミクロ的にもControlとして機能させたのであった。しかし残念ながらW3Cの複合文書規格は、HTML5の登場により実現しなかった。XVCDを活用するxfyの特徴を有効に生かすべくXBRL応用分野への開発が進められた。この開発は,日本IBM, 東京三菱銀行と提携して進められ, プロトタイプの開発が行われたが, XBRL自体の普及が進まなかったために残念ながらxfyはプロトタイプ止まりで終わった[20]。

2.5 最新の状況

2.5.1 HTML5

HTML5は, 従来のHTML4.2にXHTML2.0を統合した仕様で、CDF−WGが検討してきたSVG, MathML, XForms, SMILなどのXMLコンテンツを包含させる複合文書の技術を継承し,さらにオーディオやビデオにまで構造を拡張したリッチな複合コンテンツである。CDF−WGで検討された携帯電話画面は, スマホ画面に移行し, PCのWeb画面と共通のHTML5となり、さらにIoTやAIコンテンツのインタフェースとなりつつある。

2.5.2 スクラッチ

スクラッチは、図形素片を組み合わせてプログラムを構成する子供向けのプログラム言語である。手続き型プログラミング言語の命令セットを図形素片として定義し、それを組み合わせることにより通常の手続き型言語と遜色のない機能を実現するだけでなく、アニメーションや音声出力機能を具備することにより、子供向けの有力なデモンストレーション効果を有する。ゲームやロボット制御など、子供が楽しむためには効果的なプログラミング言語である。

3. まとめ

3.1 表の説明

以上、私が経験した種々のプログラミング言語、マークアップ言語やその環境、特徴などに関して述べてきたが、全体的にまとめると、表2のようになる。表中の言語のうち、Lilith上のModula－2だけは手をふれておらず、Pascal/PERQ、KQML+KIF、FIPA－ACLは環境に接したが使いこなしてはいない。なお仕様や解説書は読んで内容は一通り把握した。その他の言語に関しては、実際にプログラミングの作成や、マークアップによる文書やコンテンツの作成を行った経験がある。とは言え実用的に使いこなしたと言えるのは、Fortran、BASIC系、Pascal、Lisp系、C言語系、Java、マークアップ言語であり、現在何とか使えるのはプログラミング言語としてはCommon Lisp (CLOS)、JavaScript 程度である。HTML、XMLも基本的な部分は使えるが、HTML5で拡張された機能は使用していない。

表2　私が経験したコンピュータ言語とその特徴一覧

時期	言語	ブーリアン	数値	文字	GUI環境	図形	画像	映像	アニメ	音声	手続	関数	Obj	Log	MarkUp	参照	通信行為
1968	Fortran	○	◎	△							○						
1973	アセンブラ	◎	○	△												○	
1974	BASIC	○	◎	△							△						
1981	N88BASIC	○	○	○		○					△						
1981	Pascal	○	○	○							○						
1981	C	○	○	○							○					○	
1981	nroff			○											◎		
1981	Smalltalk80	○	○	○	○	○							◎				
1981	Maclisp	○	○	○								◎					
1983	Pascal/PERQ	○	○	○	○	○					○						
1983	Modula-2/Lilith	○	○	○	○	○					○						
1983	Zetalisp	○	○	○	○	○						◎	○				
1984	Prolog	○	○	○										◎			
1985	TAO/ELIS	○	○	○	○	○						○	○	○			
1989	CLOS	○	○	○								○	○				
1991	Interleaf lisp	○	○	○	○	○	○					○	○				
1991	SGML			○											◎		
1993	TeX			○											◎		
1993	C++	○	○	○							○		○			○	
1995	Java	○	○	○							○		○				
1995	HTML			○	○		○								○	○	
1998	XML	△	△	○		○			○						○		
1998	JavaScript	○	○	○							○		○				
2000	Ruby	○	○	○							○		○				
2000	RDF	△	△	○										○	○		
2000	KQML+KIF	○	○	○								○	○	○			○
2001	FIPA-ACL	○	○	○											○		◎
2005	OWL	△	△	○									○	○	○		
2007	xfy	△	△	○	◎	○	○		○		△				○		
2010	HTML5	△	△	○	○	○	○	○	○	○					○	○	
2017	Scratch	○	○	○	○	○			○	○	○						

表の項目としては、左側のブーリアン～音声までは基本的なデータ型としての属性で表1の右の欄の項目に対応する。手続、関数、Obj、Log、Markupは、言語としての文法的な特性で、いわゆる手続型、関数型、オブジェクト指向プログラミング、ロジックプログラミング、マークアップ言語の区分を示している。その次の参照は、アドレスやURLへのポインタの機能である。さらにその次の通信行為は、エージェント通信機能を意味する。内容としての○は、機能を有することを意味し、△は基本的な機能ではないが、不可能ではないことを意味する。◎は、特に注力されている機能を意味する。この評価は個人的な経験に基づくもので、必ずしも一般的ではないことをお断りしておく。

3.2 全体の流れ

Fortran、アセンブラ、BASICでブーリアン、数値の処理に始まり、Pascal、Cで文字が加わり、基本的な型を処理していた。Smalltalk80を契機にOS環境を含めた専用の高級言語環境となりウインドウシステムが描画機能をサポートするようになり、GUIが標準的にサポートされるようになった。

GUI環境を備えた高級言語マシンの主流はリスプマシンであり、エキスパートシステムの開発がブームとなりハードウエアは大量に販売されたが、エキスパートシステム自体はビジネスにならなかった。それに代わって登場したのがDTPシステムである。これは高級言語マシンではなく汎用のCPUとOS上のシステムとして構築された。それと共にTCP/IPによるインターネットが進展し、マークアップ言語のHTMLによるWebブラウザが普及した。Webのアプリケーションのためのデータ構造として汎用マークアップ言語のXMLの標準化により、Webをインフラとするグローバルなサーバネットワークが将に蜘蛛の巣のような巨大なインフラとなり、クラウド環境を構築した。エージェント通信やオントロジといった先端的なプロトタイプも試作されたがそれらは具体的なサービスには至らなかった。Webはその後XMLのコンポーネントとしての図形、画像、数式、帳票などのデータや、オーディオ、ビデオの動的なコンテンツまで包含するHTML5で標準化され、従来の携帯電話ネットワークを継承したスマホをも取り込んだネットワークとなり今日に至っている。さらに子供向けのプログラミング言語とし

てスクラッチが登場し、従来のオフィスを中心としたネットワークインフィラから教育、家庭環境も含めた地域社会のインフラとしても注目されている。

4. 考察

4.1 言語環境への図形・画像情報の寄与

以上の総括的な流れに基づいて、言語環境への図形・画像情報の寄与に関して考察を加える。以上の流れを年代的に概観すると下記のようになる。

・1970～：文字・数字を扱う古典言語
・1980～：高級言語マシンによるGUI環境
・1990～：GUIアプリとしてのDTPによるオフィス革新
・2000～：Webによるビジネス革新
・2010～：スマホによる個人・家庭・地域への浸透
・2020～：スクラッチ等による子供への浸透

最後は大胆な予測であるが、以上の推移が概観できるように思われる。この観点から図形・画像情報の言語環境への寄与を考える。1970年代から80年代の変化は、プロッタや専用端末で描画していた図形を、GUIとすることにより、ポインティングデバイスを活用する対話が実現され、キーボードと紙の時代に比べると格段に使いやすくなった。対話のためのウインドウの構成としては、SmalltalkのブラウザやLilithのSite，Mode，Trailのようなデータ構造の可視化、対話のための論理デバイスなどが考えられた。1980年代は、LispマシンのGUIによるエキスパートシステムが注目されたが、GUIの地に着いた応用はDTPであり、書類作成の効率化によりオフィスか革新された。1990年代から2000年代にかけては、HTMLとXMLによるWebブラウザが普及し、オフィスはペーパーレスになると共に、プログラムやコンテンツの作成環境と、実行・参照環境の分離が起こり、Webブラウザはコンポーネントウエア的に高機能化した。他方作成環境は、高級言語マシン当時の文化を引き継いでおり、Smalltalkやエキスパートシェルのようなテキストエディタを支援するデータ環境である。

その後のプログラム開発環境は、Eclipsが典型だが、基本的には高級言語マシン当時の開発環境を継承している。これらは創造的なプログラム記述のためには、視覚に訴える感性的な環境が必要なことを物語り、知性と感性による弁証法的な進歩を示唆するように感じる。なお、言語処理の発展のお陰で、構文誤りなどをチェックし、誤りを指摘する機能等には著しい進展が見られる。

4.2 木構造の表示とその活用

SGMLエディタやXMLエディタは、木構造の中でトラバースしながらテキストエディタを操作する環境であるが、これも1990年代のSGMLエディタ以来それほど変化していない。その環境や機能は2.1.3項で紹介したLilith上のSiteやModeの発想の拡張であり、木構造の中での操作対象の認識の重要性を物語る。XMLエディタにおいては、XSLビュワが付随しているものも存在し、これはある種のWYSIWYGと言えるであろう。

オントロジ言語のOWLには、オントロジエディタのProtegeのような興味深いツールが存在する。GUIでクラス階層を構築し、属性値を代入していくとOWLによるオントロジが構築できる。その原型はLispマシン時代のエキスパートシェルによるフレームの構築と同様に感じられるが、テキストエディタやXMLエディタでOWLのコードを入力するよりは構造を直感的に考えながら入力編集できるのではるかに効率的である。

システム理解において、全体と部分の関係把握は本質的に重要で、それは多くの場合木構造で管理される。従ってこの木を視覚的に意識されつつGUI操作を行うことを可能な画面設計を行う必要がある。

4.3 MVCメタファと直接操作

MVCメタファについては2.3節で紹介したが、GUIのインタラクションデザインを考慮する際に基本的に考慮すべき要件である。従来議論されたのは、実装設計のプログラムコードレベルのミクロな手法であるが、マクロにはDTPシステムにおけるコンテンツの作成編集プロセスなどにも拡張可能である。よりマクロには、クライアント・サーバ方式におけるリクエストとレスポンスも類似の概念で扱うことが可能である。要するにリクエストガControlで、処理がModelで、レスポンスがViewに相当する。この発想は、2.4.6項で述べたxfyのXVCDの概念の説明で思いついたアイデアであるが、さらに見方を変えると、Controlを引数として。Viewを返り値とする代数学的な関数に相当すると見ることも可能である。この概念は、分散オブジェクトにおけるCORBAの操作シグニチャ[21]であり、関数の変換内容がモデルに相当する。

このような操作は、具体的なGUIのレベルでは、人間の直感に自然に受け容れられることが望まれ、メリーランド大学のシュナイダーマンが提唱したダイレクト・マニュピュレーションの思想でもある。必要なControlに対しては、分かりやすく対応するViewがが要求され、副作用を伴うような応答は望ましくない。

4.4 位相平面とヒルベルト空間

MVCの延長として、Modelを可視化してViewに出力するという発想が重要である。オントロジやIoT関連で物理現象をシミュレートするアプリケーションは、今後増大すると考えられる。物理現象の基本は、力学、電磁気学、電気電子回路等が存在し、その多くは微分方程式で記述される。微分方程式の解法は、ルンゲクッタ法などが知られているが、実際には微分方程式を解くよりは、系の状態を把握し、そのコントロールが重要な課題である。そのためには、状態空間における系の状態を明確化し（状態点を定める）コントロールを与えて状態点を最適に移動させることが要求される。そのような最適制御に関しては、ベルマンのダイナミックプログラミングやポントリヤーギンの最大原理などで定式化されている。そのような手法の最も簡単な例は、質点の運動に関する2階微分方程式の解法でこれは位相平面と呼ばれる。これは最も簡単な例であるが、質点の運動を把握するためには便利である。多次元の位相空間（ヒルベルト空間）についても、状態空間の考え方を応用することにより考えが整理され、物理的直感（フィジカルミーニング）が得られる場合が多い。多次元の状態空間の点（状態ベクトル）に、制御ベクトルを操作することにより、状態ベクトルの速度ベクトルをコンロトールし、状態点を移行させ手法が、現代制御論として確立されている。このモデル化手法は、定性的推論でも採り上げられており[22]、今後IoT分野で活用されることが期待される。

4.5 スマホ画面とアイコン

前節のような抽象度の高い概念への支援と共に、スマホ画面のような、日常生活レベルにおけるGUIの支援も重要な課題である。スマホでプログラム開発やコンテンツ開発をすることは先ずあり得ないであろうが、HTML5の出力としてのスマホ画面への対応は重要であろう。

スマホ画面に対するGUIに関しては、一般のPC画面との対比で考察したことがあるが、アプリケーション開発における一つの提言は物語性の付与である[23]。そのために意味的文脈が重要で、映像、アニメ、音声などとの連携が重要である。小さい画面であるが故に、高い解像度を背景とする情報の活用には無理があり、画面アイコンの選択やメニュー指示で操作を完結される必要がある。従ってスマホのGUIは、アノテーション程度と割り切り、アイコン活用の標準的な枠組み[24]のようなアプローチが期待される。

4.6 スクラッチ言語

この言語は、いわきでこの言語の普及活動を行っている市川弘幸氏に紹介され、いわき市と東京におけるスクラッチデイ2017の活動等を通じて知った言語である。通常の手続型言語の命令セットを図形素片とし、グラフィック画面でアニメキャラクターを移動させるのが基本的なプログラム機能であるが、文

字列の処理、描画、音声出力等も可能で、音声付きのアニメコンテンツやゲームソフトが制作できる。

図形素片の組み立て操作なので、GUIによるプログラミングである。ウインドウは3つのブロックに分かれ、左側は図形素片命令群を格納したブロックパレット、中央はスプライトと呼ばれるアニメキャラクタ情報とプログラムエリア、右側が実行画面としてのステージとスプライトのリストである。この画面配置は、LilithのXS−1におけるSite、Mode、Trailの発想を思い起こさせる。ブロックパレットはModeであり、中央画面はデータとしてのスプライトのSiteである。従来のGUIシステムの多くは、データ環境（Site）はグラフィカルに提示していたが、命令環境（Mode）をサポートしてはいなかった。スクラッチのような子供向けの言語がユニバーサルコマンドのMode概念を有しているのは興味深い。言語を使えるようになった子供にとって言語の習熟は、名詞よりは動詞の使い方が重要なのでそれを反映していると言える。

スクラッチは自然言語の語彙体系を確立していない子供が習熟するためには適切な言語で、この分野の子供向きの言語が今後発展することは十分に考えられる。さらにこのような言語から一般のプログラミング言語への習熟プロセスも重要な課題であろう。

5. 結言

コンピュータ言語環境における図形・画像情報は、GUIを通じて提供される。プログラミング言語においては、Smalltalkに端を発するデータ環境・コマンド環境等の可視化による操作性の改善がGUIの効果として挙げられる。文書整形のマークアップ言語は、DTPシステムやそれを継承するWebコンテンツ作成管理環境を支援する。

知性的な言語と感性的な図形・画像は、対向的な概念と思われるが、人間が創造的な作業を行うためには相補的な関係がある。コンピュータプログラムや技術文書のような、論理的・記述的な知的なコンテンツの創造のためには、視覚に基づく感性的な環境が必要で、コンピュータのGUIはそれを支援してきたと言える。

プログラミング言語環境においてGUIは、モデルやデータ、コマンド環境を常時把握可能にすることがメリットでMVCメタファはその基本概念モデルである。プログラミングにおけるGUI環境の具体的な応用例は、描画・画像表示・画像処理・画像認識等に求められるが、数理分析や数学モデル的な用途としての位相平面・位相空間的な処理への活用も興味深いと思われる。

マークアップ言語は、文書やコンテンツの論理構造、レイアウト構造における符号（タグ）を定義するが、DTPやブラウザ環境における表示のための属性情報を定義する。従って、言語を図形・画像情報で支援するというよりは、GUIで提示されるコンテンツをマークアップ言語が支援すると言える。OWLのProtegeは、頭の中の概念をGUI経由でOWLに出力するので、文書やコンテンツのためのマークアップ言語とは異なり、コード作成のプログラミング環境に近い。MVCメタファはこの世界でも有効であるがプログラミング環境よりは巨視的な概念を包含する。

狭い画面でのスマホコンテンツは、大きな画面のGUIとは異質な世界で、アイコン活用の枠組みの検討、時間的推移を伴う映像、アニメ、音声のような情報の活用、価値観に訴える物語的なコンテンツ等を考慮する必要がある。さらに子供へのプログラミングのための図形・画像メディア活用、AIとの対話へ向けた通信行為モデルなどが今後の課題と考えられる。

6. おわりに

本報告は、これまで検討してきた自然言語の習得・習熟における図形・画像情報の役割[3]の延長としてコンピュータ言語環境における図形・画像情報の役割について検討したものである。知性的な言語の記述を通じた意味的創造が、感性的なGUI環境で支援され、それがプログラミング言語とマークアップ言語で弁証法的に繰り返されてきたのが、コンピュータ言語環境の歴史と言えるであろう。個人的な経験の歴史を振り返った考察なので、一個人に関するサンプルでしかないが、これまでこのような総括を行った事例は不在と思われるので、一つのデータとして参考にして頂けると幸いである。最後に、筆者にスクラッチ言語を紹介下さった市川技術士事務所の市川弘幸氏、ていねいなコメントを頂いた横浜商科大学の木村登志子先生に感謝します。

文献

[1] 大野邦夫，吉田正人；”情報メディアを構成する型概念に関する考察”,情報処理学会研究報告，DD30−2（2001.9）

[2] 大野邦夫, 木村登志子;”言語習得プロセスのモデル化に関する一検討〜自然言語の習得とプログラミング言語Lispによる開発の類似性”,情報処理学会研究報告, DC105−10（2017.7）

[3] 大野邦夫, 木村登志子；”言語習得プロセスにおける画像情報の役割”, 2017年度画像電子学会年次大会講演論文（2017.6）

[4] 大野邦夫, 木村登志子;”言語理解と異文化コミュニケーションにおける画像情報の役割”, 情報処理学会研究報告, DC107−1（2017.11）

[5] Kunio Ohno; ”Modeling Contact Erosion Using Object−Oriented Technology”, IEICE Trans. Electron, Vol.E77−C, No.10,(1994.10)

[6] K.Ohno, G. Suzuki, S. Tada; “A Study of Contact Transfer Using On−line Computer”, Proc. Holm Conference on Electric Contacts, pp.183−189（1977）

[7] https://en.wikipedia.org/wiki/Direct_manipulation_interface

[8] Kunio Ohno, Ken−ichi Fukaya, Jurg Nievergelt; ”A Five−key Mouse with Built−in Dialog Control”, IEEE SIGCHI Bulletin, Vol.17, No.1, (1985.7)

[9] 大石進；”オーバービュー オブ Interleaf5”, SuperASCII, Vol.3 #10, #11 (1992.10〜11)

[10] 大野邦夫；”オブジェクト指向による非線形振動のモデル化”, 電子情報通信学会論文誌 D−II, Vol.J77−D−II, No.9 (1994.9)

[11] K. Ohno; “An Expert System for Predicting the Relay Contact Arc Erosion”, Proc. of the International Conference on Electrical Contacts and Their Applications, Nagoya, Japan, pp.481−490, (1986.7)

[12] 大野邦夫；”摩擦振動とシャワー放電のアナロジ”, 日本機械学会論文集（C編）Vol.59, No.568,論文 No.93−0211（1993.12）

[13] 大野邦夫, 新麗；”アプリケーション仮想化環境における複合文書”,情報処理学会研究報告, DD95−1（2014.10）

[14] 大野邦夫；”オブジェクト指向プログラミング技法による通信機用接点消耗予測システム”, 名古屋工業大学 学位論文（1997.3）

[15] 大野邦夫，佐藤和也；”ORDBによるマルチメディア・ドキュメントの管理”,情報処理学会研究報告, DD7−5,（1997.5）

[16] Kunio Ohno, Morten Bayer;”Development of SGML/XML Middleware Component”, Proc. SGML/XML Europe’98 (1998.5)

[17] 大野邦夫, 芥川一則；”エージェント通信と異文化コミュニケーションの類似性に関する検討”, 情報処理学会研究報告, DC99−1（2015.10）

[18] 大野邦夫；”FIPAエージェントにおけるXMLの適用動向”,情報処理学会研究報告,DD23−3（2000.5）

[19] 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)

[20] Kunio Ohno;”xfy with XBRL Extends Financial Application”, 13th XBRL International Conference Madrid, Spain (2006.5)

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

[22] Howard E. Shrobe et. al., ”Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence”, Morgan Kaufmann Publisher, Inc.（1988）

[23] 大野邦夫；”画面推移プロセスに基づくスマホ時代のテクニカルライティング”, 2014年度画像電子学会年次大会講演論文（2014.6）

[24] 大野邦夫；”図形・画像によるシンボル情報の意味概念に関する検討”, 第4回画像関連学会連合会大会講演論文（2017.12）