ヒューマンインタフェース 30－2
（1990．5．10）

Multidimensional Visualization Toolの開発

佐藤洋一　小池英樹　広瀬通孝　石井威望
東京大学工学部

概要

ソフトウェアを多くの人間が共同で開発していく大規模プログラミングでは、比較的少数のプログラマによるソフトウェア開発とは本質的に異なった問題が存在する。その中で非常に大きな部分を占めるのが、プログラマ間におけるコミュニケーションをいかにして行い、開発対象となるソフトウェアを理解してゆくべきかという問題である。そこで我々はこの問題に対する一つの解答を見つけ出すことを目指し、まず最初に幾つかの基礎的な実験を行い、その結果の解析を通してプログラマ間のコミュニケーションにおいて、いかなる情報伝達が必要であるかを考えた。そしてそこで得られた結果をもとに、具体的にプログラマ間のコミュニケーションを支援するツールとして、Multidimensional Visualization Toolを開発した。

Development of Multidimensional Visualization Tool

Youichi Sato　Hideki Koike　Michitaka Hirose　Takemoti Ishii
Faculty of Engineering, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113 Japan

Abstract

This paper describes the method to support communication during software development process. At fisrst, we investigated what kind of information is necessary to explain software. As a result, we found that it is important to present an abstract of whole structure of a software for comprehension. Therefore, to support communication among software development team, we developed the multidimensional visualization tool which can represent a whole structure of software using 3D computer graphics.

1 はじめに

大規模プログラミング (Programming in the Large) は、(1) 複数のプログラマによる同時並行開発、(2) 仕様書、ドキュメント、設計書、DFD、マニュアル等、ソースコード以外の多くのソフトウェア・プロダクツの存在、(3) 他のプログラマによって作成されたソフトウェアの保守等の点において、一人あるいは少人数のグループによって開発され、コメント程度の付加情報で保守される小規模プログラミング (Programming in the Small) とは本質的に異なる多くの問題を有する。

こうした大規模プログラミング支援のために、プログラミング言語としては Ada([10]) が開発され、プログラムの部品化が進められた。また最近では、ソフトウェア・プロダクツの総合的管理を計算機によって支援しようする CASE (Computer Aided Software Engineering) 環境が注目をあび、実際にいくつかのシステムが稼働している。

上述の大規模プログラミングにおける問題点には、プログラムの部品化技術、ソフトウェア・プロダクツのデータベース化といった問題の他に、ソフトウェア開発に携わる人同士のコミュニケーションの問題が存在する。

こうした視点から上述の大規模プログラミングにおける問題を眺めると、上の (1) は並行開発中のプログラマ間、(2) は設計者・プログラマ間あるいは発注側・下請け側間、(3) は過去の作成者と現在の保守者間のコミュニケーションの問題と捉えることが可能となる。

著者らは CSCW(Computer Supported Cooperative Work) の研究を通じて、人対人の円滑なコミュニケーションを実現するには、従来のような形式的に記述された情報の伝達だけでは不十分であり、冗長で曖昧な (informal な) 情報の伝達が必要であるとの知見を得ているが [?, kuzuoka1] その際、特に「絵」が伝達できる情報量の多さ、および人間のメンタル・モデル形成に対して果たす役割の大きさに注目している。

以上の視点に基づき、我々はまずソフトウェアを他人に理解させるにはいかなる情報が必要であるかを調べるために、基礎的な実験を行なった後、ソフトウェア・データベースのユーザ・インタフェースとしてソフトウェアを可視化するツールを開発した。

2 ソフトウェア説明実験

2.1 実験の目的

あるシステムを理解する最も効率的な方法は、そのシステムを理解している人に直接説明してもらうことであろう。たとえその人が保持する知識と等価な情報をデータベース (あるいは知識ベース) に格納できたとしても、対人説明以上の効率は期待できない。

これは従来のマニュアルのように文書が主体で、それに図が加わった形態の情報ではシステム全体の理解、および意図伝達に本質的に不十分であるからと考えられる。

ヒューマン・インタフェースの重要性とその困難さが認識されるようになった現在、さかんにマルチ・メディアを用いたインタフェースが研究されているが、現段階での焦点は文字情報、図形情報、動画情報、音声情報の統一的な取り扱い方や計算機上への実現手法であり、状況に応じたメディアの選択、混合といった、いわばメディアの有効な利用法については、さほど研究がなされていない。

こうした背景をもふまえて、従来の文字主体の情報形態で欠落している、システムの理解および意図伝達に必要な情報を探るために以下のような実験を行なった。

2.2 実験の方法

- あるソフトウェア (C 言語、総行数 約1500行) をその作成者に説明してもらう
- 説明を受ける被験者は、対象となるソフトウェアに関してほとんど知識を持たない者とする。
- 説明時に利用する資料はソフトウェアのソースコードだけとし、その他必要な図などは説明者が適宜手で描いて被説明者に提示する形とする。
- ソフトウェアの説明を行っている間、被説明者の RRV 値を測定して被説明者の集中度を観察する。なお RRV 計測装置とは演者らの研究室において開発された装置で、被験者の心拍数とその分散値をリアルタイムで測定することが可能であり、心拍数の分散値から被験者の集中度を定量的に計ることが可能である。

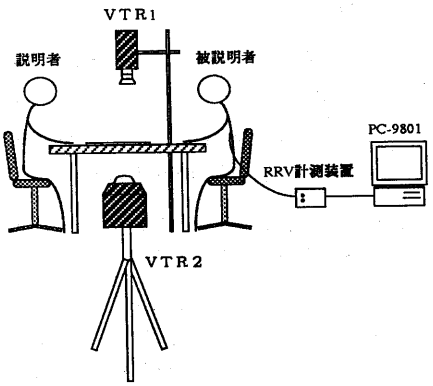

図 1: ソフトウェア説明実験概念図

- 説明の様子は 2 台のビデオカメラで記録しておく。この際、1 台は説明者と被説明者の 2 人を同時に撮影し、説明を行っている状況全体を記録し、もう 1 台で説明者の手元を真上から撮影し、説明者によって提示される情報を記録する。
- 被説明者は自由に質問を行ってもよいこととする。

図 1 は実験の概念図である。

2.3 考察

今回、対人の形式でソフトウェアの説明実験を行った結果、説明者と被説明者の間における情報伝達の中で、従来の形でのソフトウェアの説明において実現され難かったものとして以下のようなものが存在することが分かった。

ソフトウェアの全体像の把握　説明の対象となるソフトウェアの全体像が説明者によって提示されない場合に被説明者の集中度の低下がみられた。また被説明者は適宜、質問を行うことによって、ソフトウェアの全体像に対する自己のメンタルモデルを確認している。また全般にわたり、ソフトウェアの個々の部分を説明する際には、常に全体構造の中でその部分の占める位置を明確にしながら、全体と部分との相互参照をベースに説明が行われる傾向が見受けられた。

情報のマルチな提示　文字情報や図形情報による情報伝達の他に、音声による情報伝達や、身振り、手振り (一種の動画情報) による情報伝達が、幾つも並行して行われている。

フレキシブルな説明　説明者は被説明者がどの程度知識を持っているのかを意識しながら、被説明者の質問内容や視線、表情などから理解の度合を推測し、それに応じて説明の詳しさを最適に調節している 。こうすることによって必要以上に多くの情報を提示することなく、不必要な部分の情報は遮蔽しながら説明を行っている。

このような幾つかのポイントの中でも我々は、全体構造を常に把握しながら、その各部分を説明してゆくことが、ソフトウェアの全体の理解や意図伝達には特に重要であると考えた。今回この考えに基づき、ソフトウェアの全体構造を明確に提示することによって、その全体の理解や意図伝達を支援するシステムを開発した。

3 SIGHT

複雑なシステムのイメージを把握するのを支援するためには、直感的に全体構造が把握できるような形での提示が必要であり、そのためには「絵」が有効であると考えた。特にソフトウェア・システムのように実際の形を持たないものに対して、疑似的な形状を与えることは、人間にとってより理解しやすいと思われる。このための一つの実現方法として、ソフトウェアの各部分 (例えば一個の関数) をノード、そのノード間の関係 (例えば関数の呼び出し関係) をリンクとし、ソフトウェアの全体構造をそのノードとリンクによる 3 次元空間内のネットワーク構造として視覚化する方法を考えた。このようにソフトウェアを 3 次元空間内のネットワーク構造として表現する方法には、以下のような利点がある。

1. 従来の 2 次元平面上での表示に比べて、3 次元空間へ次元数が増加することによって伝達情報量が増大する。
2. 色、形状などを自由に用いること (multidimensional 化) によって、伝達情報量を更に増加させることが可能である。
3. 2 次元での情報提示では視点の変更が不可能であったのに対して、3 次元では視点を自由に変えていくことが可能である。

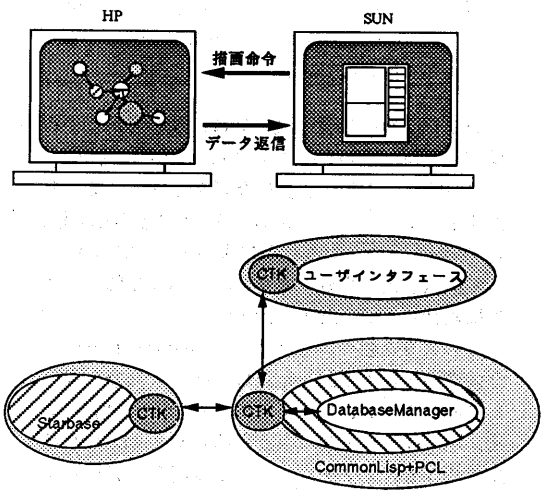

図 2: SIGHT のシステム構成図

4. 3 次元空間内のネットワーク構造として表現されたソフトウェアの全体構造を、連続的に視点を変化させながら眺めることによって、ソフトウェアの全体に対するメンタルモデルの形成が容易になる。

そこで我々は、ソフトウェアの全体構造を 3 次元空間内のノード、リンクのネットワーク構造として表現することにより、ユーザのソフトウェア・システムに対するメンタル・モデルの形成を積極的に支援し、更にその環境内で実際のソフトウェアの作成を可能にするためのツール SIGHT を構築した (図 2)。

3.1 SIGHT の機能

ソフトウェアの全体構造をノード、リンクのネットワーク構造で表現し、その環境内でソフトウェアを実際に作成していくために必要となる機能を考えると、それは大きく分けて次のようになる。

1. ノード、リンクの生成、管理、表示などのための基本的機能
2. ネットワーク構造を 3 次元空間内で直接操作するための機能
3. SIGHT 環境内で実際にソフトウェアの編集を行うための機能

現在、2 番目の”ネットワーク構造を 3 次元空間内で直接操作するための機能”としては、

- データグローブを利用して 3 次元空間内においてデータを直接操作する機能
- 2 平面上でマウスを操作することにより 3 次元空間内のデータを操作する機能

の 2 通りの方法が実現されている。

また、3 番目の”SIGHT 環境内で実際にソフトウェアの編集を行うための機能”としては、ネットワーク構造の中で編集したい部分のノードを選択すると、それに対応したウィンドウが Sun3 の画面上に開き、emacs 環境内で編集作業できるようになっている。

3.2 ノードの配置について

SIGHT は複雑なソフトウェアの全体構造のイメージを表現し伝達することを目標としているために、ネットワークのノードを 3 次元空間内においてどの様に配置するかという問題は非常に重要である。そこで SIGHT におけるノードの配置についての説明を行う。

SIGHT ではネットワーク構造のノードの配置は大きく分けて

1. ユーザがマニュアルで配置を行う
2. あるアルゴリズムに基づいて、システムが自動的に配置を行う

の 2 通りの方法で行っている。

それぞれの方法について具体的に述べていくことにする。

3.2.1 ユーザによるマニュアル配置

現在 SIGHT ではマウスを用いる方法とデータグローブを用いる方法との 2 つでユーザによるマニュアル配置を行っている。

マウスによる配置 マウスによって、ユーザ・インタフェース部の操作メニュー内にある xy 平面と xz 平面を表す 2 平面上でポイントを指定することにより空間内の x 座標、y 座標、z 座標を得ている。

データグローブによる配置　データグローブによる方法では、ネットワーク構造が表示されている3次元グラフックの中で、自分の手の動きに連動して動く仮想的な手を用いて位置を指定する。

ユーザがマニュアルで配置を行うという方法は、計算機に自動的に配置させる方法と比較して、ノードをユーザの考えた通りに自由に配置できるという点では優れているが、ノードの数が非常に多くなってしまうと人間の側でネットワーク構造の全体が把握できなくなり、ノードの配置場所を特定することが困難になってしまう。

このような状態になると、もはや人間の手作業でネットワークのノードを配置することはほとんど不可能になってしまう。そこで計算機による自動配置を行うことにする。

3.2.2　SIGHTによる自動配置

SIGHTはソフトウェアの複雑な全体像のイメージを得ることの支援を目的としているということからも、ネットワーク構造が複雑で大きくなった場合に困難になってしまうようなマニュアル配置しか行えないのでは不十分である。そこでSIGHTではSemNet([8])で用いられているものを含めたいくつかのアルゴリズムを用いて自動配置を行うようにした。

力学的シュミレーション手法　ノード間にリンクがある場合には2つのノード間に引き合う力が働き、リンクがない場合には反発しあう力が働くとして、ノード全体が安定する点を求める方法。このときネットワーク全体が振動し続けるのを避けるためにノードの移動速度に応じた減衰力も考慮する。

このアルゴリズムでは引力、斥力、減衰力の大きさの決定が困難であり、ネットワーク全体の振動を有効に抑えて、安定な点を求めるのが難しい。

中心化手法　ノードの新しい位置を、そのノードとつながっている全ノードの重心位置として与える方法。このときにリンクの種類によって各ノードの座標に重み付けをして重心の位置を計算することも行う。　このとき計算した重心位置が、他のノードからあらかじめ定めておいたノード間最短距離以内の位置であった場合、ノードが重なり合うのを避けるために元の位置と計算位置との間の線分上に配置される。もしもこの位置でも他のノードと近すぎる場合には一番最初の位置に戻す。このような計算をネットワークの全ノードに関して行うというサイクルを必要なだけ繰り返す。

このアルゴリズムでは比較的少ない計算で安定点に到達することが可能であるが、最初にランダムにノードを配置したときの分布によって安定したときの結果が影響を受けてしまう傾向がある。

焼きなまし手法　焼きなまし手法は、焼きなましプロセスのシミュレーションの研究から考えられた。

まず中心化手法と同様に、あるノードとつながっている全ノードの重心位置を計算する。ここで元の位置からあらかじめ決めたノードの最大移動距離以内のランダムな位置を計算し、その位置に移動すると元の位置よりも重心位置に近くなる場合にのみ実際に移動していく。このときも移動先の位置の近くに他のノードが存在している場合には移動を行わない。　このような計算をネットワークの全ノードに関して計算するというサイクルを必要なだけ繰り返す。

このアルゴリズムでは移動先の位置がランダムに決定されるために収束するまでに必要なサイクル数が多くなってしまう。しかしランダムに移動するということから中心化手法のように安定になった時の配置が、ノードの初期分布に影響を受けてしまうといったことが少なくなる。

多軸中心化手法　通常の中心化手法ではノードの移動に関してx軸、y軸、z軸の3軸の方向で差がなかったが、多軸中心化手法はノード間に張られているリンクの種類によって引き合う方向を変えてしまう方法である。

例えば、あるノードの移動先のx座標は、そのノードとAという種類のリンクでつながっている全部のノードのx座標の平均値、y座標はBというリンクでつながっている全部のノードのy座標の平均値、z座標はCというリンクでつながっている全部のノードのz座標の平均値として求めていくといったぐあいである。

この多軸中心化手法によりノードを自動配置した結果では、各軸方向によって反映されているリンクの種類が異なっている。このため、このアルゴリズムを利用するとネットワーク構造をどの軸の方向から見るかによって違った観点に注目してネットワーク構造を見ることが可能になる。

多軸焼きなまし手法　多軸中心化手法と同じ拡張を焼きなまし手法に適用した方法である。この方法でも多軸中心化手法と同じように視点の切り替えが可能になる。この方法では焼きなまし手法と同様に、最初の分布に影響を受けにくいという点が優れている。

3.3 ハードウェア概要

SIGHT はグラフィック表示部を行なう Hewlett Packard の HP340 ワークステーションと、ネットワーク構造のデータベース部とユーザ・インタフェース部の乗った Sun3 ワークステーションの 2 台の計算機で構成されている。

3.4 システム構成

HP 上のグラフィック・インタフェース部は、C 言語で記述され、ディスプレイ上にネットワーク構造を表示し、グラフィックオブジェクトに対する直接操作を行う機能を持たせてある。

Sun3 ワークステーション上のデータベース部は、オブジェクト指向の CLOS で記述され、ネットワーク構造のデータを管理し、HP ワークステーションのグラフィック・インタフェース部と Sun3 ワークステーションのユーザ・インタフェース部をデータベース部のデータと接続している。

このように、SIGHT はデータベース部、グラフィック・インタフェース部、ユーザ・インタフェース部の 3 つの部分から構成されている。この 3 つの部分は、CTK というプロセス間通信ツールキットを利用してのメッセージ通信で接続されている。

3.5 各モジュールの説明

3.5.1 LISP データベース部

HP ワークステーションのグラフィック・ディスプレイに表示するネットワーク構造のオブジェクト・データを統一的に管理している。実際に HP ワークステーション上に画面表示を行うのはメッセージを HP ワークステーション側のグラフィックインタフェース部に送ることにより実現している。

このデータベースはオブジェクト指向で実現されていて、データは階層構造を持ったクラスのインスタンスとなっており、各インスタンスはメソッドを持ち、データベースへの操作はこのメソッドが主体となっている。

3.5.2 ユーザ・インタフェース部

LISP データベース部への操作をメニュー形式で実現している。ここでのメニュー操作は CTK を通してメッセージとして LISP データベース部に送られ実際に、データベースのオブジェクトデータが処理される。　またマウスによるノードのマニュアル配置もこのメニュー上で行われる。　このユーザインタフェース部は X Window システムのアプリケーション作成支援ツールである xtoolkit を利用してつくられている。

3.5.3 グラフィックインタフェース部

CTK によるメッセージ交換のための HP ワークステーション側のサーバとなっている。また実際に Sun ワークステーション側と CTK を利用してメッセージ交換を行い、HP ワークステーションのグラフィックディスプレイに starbase グラフィックライブラリを利用してネットワークの表示を行なう。また HP ワークステーションの画面上におけるマウス入力を受けとり、CTK により Sun ワークステーション側にメッセージを返す。

3.5.4 CTK

CTK とは共同作業用ソフトウェア (グループウェア) 開発を支援することを目的として開発された、プロセス間通信のためのツールキットである。

このツールの特徴は、

1. プロセス間通信に関するプログラミングが非常に短時間に行なえること
2. 既に通信を行なっているプロセス群に簡単に接続、切り離しができること
3. 複数の言語、インタフェースに対応していること

などである。我々はこの特徴を「プラガブル」という言葉で表現している。詳細は文献 [5] を参照されたい。

4 応用例

4.1 CLOS Browser

CLOS(Common Lisp Object System) は Common Lisp へのオブジェクト指向導入案として開発された。CLOS では Smalltalk([11]) に代表さ

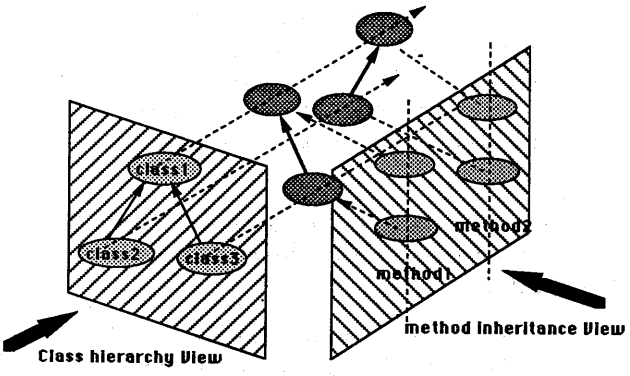

図 3: CLOS Browser

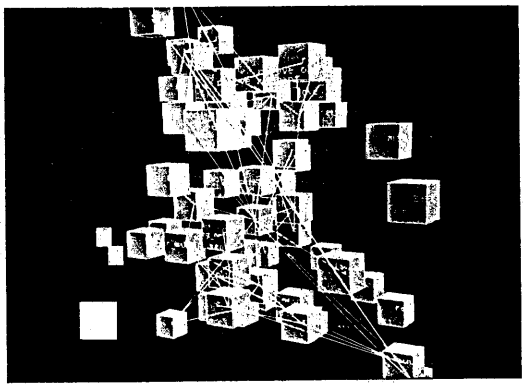

図 4: SIGHT の画面表示例

れる他のオブジェクト指向言語と異なり、メソッドはクラスから独立して存在し、同じ名前のメソッドは総称関数として 1 つにまとめられている ([9])。

Smalltalk の System Browser 利用時に不便なのは、あるクラスのインスタンスにメッセージが送られた場合、実際にクラス階層におけるどのメソッドが起動されるのかを追いかけるのが困難な点である。CLOS では :before、 :after、 :around の各デーモンを用いたメソッド結合が存在するため、デバッグ作業は一層複雑になる。

オブジェクト指向言語においては、大別して「クラス階層に注目した視点」と「メソッド継承に注目した視点」が存在し、既存の Browser ではこれを同時に支援することができない。

図 3 は本システムによる CLOS Browser の実現方式の図である。この 2 つの視点をともに満足し、かつ両視点間を連続的に移動することが可能である。

またクラス階層を可視化することによる、ユーザのメンタル・モデル形成をも支援できると考えられる。

4.2 クロス・リファレンスの可視化

図 4 は C 言語で記述された、あるソフトウェア・システムを可視化した例である。この場合各関数をそれぞれ一つのノードとし、またクロス・リファレンスから得られる関数の呼び出し関係をリンクとして表現し、焼きなまし手法を用いてネットワークを再構成している。

ネットワーク再構成によって、必ずしも何らかの意味を持った形になるわけではないが、少なくとも可視化したことにより使用されていない関数、呼び出しの頻繁な関数といった幾つかの情報が一目でわかる。

5 考察

今回試作したシステムに対する定量的な評価はまだであるが、被験者からはおおむね良好な感想を得ている。その理由は以下のようなものと考えられる。

- システムに対するメンタル・モデルの形成が容易になること
- 全体像と個々の情報とのフレキシブルな行き来が可能であること

ただし、現システムの問題点は次のようなものである。

- 自動配置
  今回用いた自動配置のアルゴリズムでは、ネットワークは必ずしもなんらかの意味を持った状態に落ち着くとは限らない。ノード間のいかなる関係を用いてネットワーク再構成を行なうかを含めて改善の余地がある。

- 表示スピード
  Starbase グラフィック・ライブラリの制限上、多数のノード・リンクを表示すると、画面の更新が遅くなり、実用上耐えられなくなる。

6 結論

本稿では、基礎的なソフトウェア説明実験の結果から、ソフトウェア・システムの理解には全体像の把握が特に重要であることを述べ、これらの考察に基づき実際に試作した、全体像を可視化するツールについて報告した。

7 謝辞

本システムを作成するにあたり、東京電力制御研究室の名井健氏, 甘利治雄氏の御助言・御協力を頂きました。

参考文献

[1] 佐藤、“アイディアプロセッサに関する研究”、東京大学機械工学科 卒業論文,1990.

[2] 小池, 広瀬, 石井, “ソフトウェア・カルチャーに基づく部品検索”, 情報処理学会知識工学と人工知能研究会,1990.

[3] H.Hirose,T.Myoi,H.Amari,K.Inamura,L.Stark, “Development of visual 3D virtual environment for control software”, Human Machine Interfaces for teleoperator and virtual environment, 1990.

[4] T.Ishii,M.Hirose,H.Kuzuoka,T.Takahara and T.Myoi, “Collaboration System for Manufacturing System In the 21st Century”, Proceedings on MSEC21 (投稿中),1990.

[5] 葛岡, 三井, 広瀬, 石井, ”プラガブルなネットワーク・アプリケーションの開発”, 情報処理学会ソフトウェア工学研究会 (投稿中), 1990.

[6] 岸本、西田、坂口、“ソフトウェア開発プロセスにおけるコミュニケーションサポートに向けての一考察”、計測自動制御学会 第3回ヒューマンインタフェースシンポジウム,1987.

[7] K.Fairchild,G.Meredith and A.Wexelblat, “The Tourist Artificial Reality,”, CHI'89 Proceedings,1989.

[8] K.M.Fairchild, S.E.Poltrock and G.W.Furnas, “SemNet: Three-Dimensional Graphic Representation of Large Knowledge Bases”, Cognitive Science And Its Application For Human-Computer Interaction, R. Guindon (eds.), Lawrence Erlbaum Associates, 1988.

[9] 井田, 元吉, 大久保, “Common Lisp オブジェクトシステム -CLOS とその周辺-”, 共立出版,1989.

[10] G.Booch, “Software Engineering with Ada”, The Benjamin/Cummings Publishing,1983.

[11] A.Goldberg and D.Robson, “Smalltalk-80: The Language and its Implementation”, Xerox,1983.

[12] J.Ramanathan and R.L.Hartung, “A Generic Iconic Tool for Viewing Databases”, IEEE Software,1989.