知識工学と人工知能　69－1
（１９９０．３．２２）

ソフトウェア・カルチャーに基づく部品の検索

小池英樹　　広瀬通孝　　石井威望
東京大学工学部

ソフトウェアに存在するinformalな情報(ソフトウェア・カルチャー)は再利用に重要な役割を果たすと思われる。本稿では特にソフトウェア部品の名前に注目し、それらに基づいた検索について述べる。

ソフトウェア部品の名前はキーワードとしての性質とともに、格構造としての性質を持っている。この性質を検索に反映するために、階層化した用語辞書の他に名前テンプレートという新たな知識構造を導入した。

さらに検索者の視点の変化を、この２種類の知識が形成する知識空間の移動として捉え、その視点制御を(1)抽象化/具象化、(2)一般化/特殊化 という４つの操作に分類・整理を行なうとともに、これらに基づく検索システムを試作し、実際のシステムへの適用可能性を考察した。

Software Database Retrieval using Software Culture

Hideki KOIKE,　　Michitaka HIROSE　　Takemochi ISHII
Faculty of Engineering, The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan

This paper describes a method for software database retrieval using Informal Information, Software Culture, that are merged in the name of the software parts.

The title of the software parts has characteristic not only as keyword for retrieval but also as frame structure. To reflect these characteristic, we introduced two knowledge structures, Word Dictionary and Name Templates.

We regarded changing of viewpoint for retrieval as moving around inside these knowledge structures, classified these viewpoint manipulation into 4 operations, (1)abstraction/instanciation and (2)generalization/specification,and also prototyped a retrieval system.

1 はじめに

計算機ハードウェア技術の急速な進歩による生産コストの低下の一方において、ソフトウェアの開発・管理・保守費用、特に大規模なシステム (Programming in the Large) における、設計書、仕様書、ソースコード 、ドキュメント等をも含めたソフトウェア全体にしめる保守費用の割合はが大きな問題となっている。

この問題に対する回答は、いかにして既存のソフトウェアを有効に再利用するかであることが早くから指摘され、多くの研究がなされているが必ずしも効果をあげてはいない。

演者らは、遠隔地間共同作業の研究等 ([5]) を通じて、コミュニケーションには従来の formal な情報の他に informal な情報が重要な役割を果たしているとの知見を得ているが、ソフトウェア再利用が部品作成者・再利用者間のコミュニケーションであると考えると、再利用問題を解決する鍵の一つは、いかに両者間のコミュニケーションを円滑にするかという問題として捉えることができる。

本稿では、こうした informal な情報の中で、特にソフトウェア部品の名前の付け方に注目し、その名前の情報だけに基づいて検索を行なうためのデータ構造を提案し、それを基に試作した検索システムに関して述べる。

2 従来の研究

最も一般的に行なわれているソフトウェア再利用としては、サブルーチンやライブラリの使用があげられる。C における標準ライブラリ、Smalltalk-80 におけるクラス・ライブラリ等の利用がこれにあたる。こうしたソフトウェア再利用プロセスは、

1. 部品の検索
2. 部品の読解
3. 部品の修正

という 3 つのフェーズに大別できる。

この各フェーズに対して、従来行なわれてきた研究としては、プランニングを用いた自動プログラミング ([15])、プラン・ゴール理論を用いた意図理解支援 ([3]) 等があり、極めて限定された分野においては、 成功を収めているものもあるが、実際に適用され活動しているシステムはほとんどない。

仮に多くの優れた部品の存在を仮定しても、膨大な部品ライブラリから目的の部品に簡単にアクセスする手段が提供されていなければ、再利用プログラミングは決して魅力的ではない。こうした点からも再利用プログラミングの入口としての検索フェーズは特に重要である。しかるに、この検索フェーズ支援の研究はそれほど多くない。代表的なものはつぎの 2 つである。

1. 単純なキーワード管理
   最も一般的に行なわれている方法は部品のデータベースによる管理である。各々の部品はデータベース管理者あるいはプログラマによって付加された数個の適当なキーワードとともに保存され、検索者は必要に応じてキーワードを用いて部品を検索する。

2. 格文法を用いたキーワード管理
   部品の使用記述の格構造に注目し、あらかじめ用意された数種類の格をキーワードで埋めることにより各部品はデータベースに格納される。検索時には、適当な格の部分をキーワードで埋めて検索を行なう。Prieto-Díaz らの分類・検索システム ([8]) がこれにあたる。

1 の単純なキーワードによる部品管理は、現在の情報検索における問題同様、それだけでは、検索者の意図をうまく反映できないという欠点がある。

一方 2 の Prieto-Díaz らのシステムは、図書館学のファセット管理をキーワードの管理に応用し、大規模なキーワードを比較的簡単にかつ効果的に管理することに成功しているという他に、格を変形した、

<機能、対象、媒体>
<システムタイプ、機能分野、環境>

という 6 つのスロットを用いた検索を取入れており、単純なキーワード検索に比べて、検索要求を出す時にある程度検索者の意図を伝える機能をもっているという点で優れている。

しかるに、1、2 のシステムが実際に運用されたときに共通しておこると予想される最大の問題点は、キーワード付加作業の繁雑さであろう。

図書館業務から明らかなように、一般のデータベース管理においてもキーワードの管理・付加は膨大な時間と労力の必要な作業である。1、2 のようなソフトウェア・データベースを機能させるには、いちいち部品登録の際に、キーワードの付加をプログラマに強制するか、さもなくば、各再利用ドメイン内に図書館のように索引課を設置してキーワードの管理・付加をする必要があるが、このようなシステムは現実的でない。

よって、実際的なソフトウェア・データベースは、上述したキーワード付加の労力をなくさなくてはな

らない。この問題を解決するのがソフトウェア・カルチャーであると我々は考えている。

3 ソフトウェア・カルチャー

3.1 ソフトウェア・カルチャーの例

ソフトウェアにはプログラミング言語仕様、クロスリファレンス等の形式的に記述された情報 (formal information) のみならず、名前の付け方、インデンテーション、モジュール間のあいまいな関係といったinformalな情報が存在すると思われるが、演者らはこれをソフトウェア・カルチャーと呼び、ソフトウェア開発・保守においてformalな情報を補完するものとして位置付け、その利用を目指している。([2])

ソフトウェア・カルチャーは、長期間の研究を経た優れたシステムにおいて特に顕著であり、名前の付け方だけに注目すると以下のようなカルチャーが存在する。

1. 文節の区切り方
   (例)
   ReadWriteStream(Smalltalk),
   with-open-file(Common Lisp),
   DESTROYCARD(INTERLISP)

2. 特定の文字列による関数の性質表示
   (例)
   numberp, adjust-array-p(Common Lisp)
   これは暗にTかNILを返す述語であることを示している

3. 特定の位置による関数の性質表示
   (例)
   fputc, fgets, printf, scanf (C)
   前2者はファイルへの入出力を意味するfであり、後2者はフォーマットを表すfである

4. 類似文字での異なる性質記述
   (例)
   delete or remove (Common Lisp)
   破壊的操作かそうでないかの違いを表す

その他にも、XOpenDisplay(X Window)等、先頭の数文字によってモジュールを明示することもある。

このようなカルチャーに従って作成された部品は、勝手に名前付けされた部品と比較して、読解時に伝達できる情報量が多くなるという点で、再利用性が非常に高くなると考えられる。

```
<cl> (apropos 'string 'user)
GET-OUTPUT-STREAM-STRING (defined)
PRINC-TO-STRING (defined)
MAKE-STRING (defined)
WRITE-TO-STRING (defined)
STRING-LEFT-TRIM (defined)
PRIN1-TO-STRING (defined)
NSTRING-DOWNCASE (defined)
NSTRING-CAPITALIZE (defined)
STRING-RIGHT-TRIM (defined)
STRING-CHAR
STRINGP (defined)
MAKE-STRING-OUTPUT-STREAM (defined)
MAKE-STRING-INPUT-STREAM (defined)
        :
        :
```

図 1: aproposによる検索

また検索時には、上述した、キーワード、格構造両方の性質を潜在的に持っている。

3.2 キーワードとしての名前

ソフトウェア部品に付加される名前は、作者の感性に基づいて決められる文献のタイトルとは異なり、なるべくその処理内容、あるいはモジュール内容を表すように意図して付けられていて、これらの名前に使用されている単語は本来キーワードとしての機能を持っている。

この点において一々キーワードを付加する必要のある普通のデータと、ソフトウェア部品とは明らかに異なる。

こうした名前に用いられる単語をキーワードの代わりに使用する考えは、決して新しいものではなく、Common Lispのaproposにも見られる。また、適当な部品名リストがあれば、UNIX上でもgrepを使って文字列検索を行なえば済む。

しかしこの方法の欠点は部品数が多くなるとある単語から検索される対象の数が非常に多くなってしまうこと、また、検索者の意図に反した無意味な検索を行なってしまうことである (図1)。

3.3 格構造としての名前

キーワードのみの場合の検索の非力さを補い、またできるかぎり検索者の意図を反映するために、Prieto-

Díaz らは格文法を変形した6スロットの検索方式を用いた。

しかし、名前に用いられる単語の並べ方に注目すると、これらはおおよそ

“何をどうする”

あるいは

“何をどこにどうする”

といった形式で記述されており、これらが格文法的要素を持っていることは明らかである。

これまでにも、格文法を基本に理解しやすい名前付けを支援するシステムの提案は幾つかあったが、それらのシステムは一般的な格文法構造に基づいた名前付けにこだわりすぎて、ソフトウェア部品にありがちな省略語の使用、品詞の特殊な並び方による情報圧縮を見落としていたと考えられる。

これに対し我々の提案するシステムは、再利用ドメイン内でのコンセンサスさえ得られているならば、格の出現する順番は問題とせず、また省略形や特殊な文字の使用を許す。例えば、文字列連結のための関数名は、

string-concatenate
concatenate-string
concatenate strcat
append-s

等が許される。

4 検索システムのインプリメンテーション

前章までの考察から、ソフトウェア・データベース、及びソフトウェア部品は次のような特徴を持っている。

- キーワードの付加を強制するシステムは実現が困難である
- 部品の名前にはキーワードとしての機能がある
- 部品の名前には格構造としての機能がある

こうしたソフトウェア部品の特徴、つまり名前付けのカルチャーに基づいた検索システムを試作するにあたり、システムの特徴は以下のとおりである。

1. 部品名に用いられる用語間の関係を取り扱う必要があるが、ここでは簡単のため概念的な上位/下位関係のみを考慮する
2. 名前の格構造の格数は不定であるので、それを取り扱えなければならない

これらの仕様を実現するため以下のようなデータ構造、オペレーションを持った検索システムを Allegro Common Lisp + PCL (Portable Common Loops) 上に試作した。

4.1 データ構造

ソフトウェア部品の名前のキーワードとしての性質、及び格構造としての性質を検索に反映するためにシステムには次の2つのクラスを用意する。

1. 用語ファセット・クラス
   用語ファセットは名前に用いられる単語で、それらのうち上位概念/下位概念にあたるものが、いわゆる isa リンクで結ばれた木構造を形成している。(図2)

2. 名前テンプレート・クラス
   名前テンプレートは名前において用語ファセットの出現する順番、つまり名前の鋳型を保持する。

   例えば Action ファセット 及び Data ファセットからなる名前テンプレートは、Action ファセットのみをもつテンプレートと Data ファセットのみをもつテンプレートのサブ・テンプレートとして定義される(図3)。用語ファセットの順番が異なるものは違うテンプレートとして定義されなければならない。

つまり、システムがもつ二つのカルチャーに関する知識は全体として、名前テンプレートがネットワーク平面を形成し、また個々の名前テンプレート中の用語ファセットがその平面に対して垂直な方向に木構造を形成している。

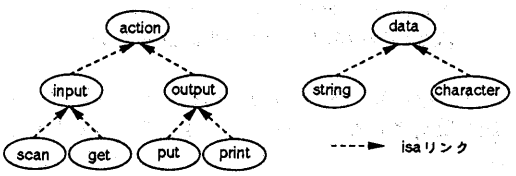

図 2: 用語ファセットの木

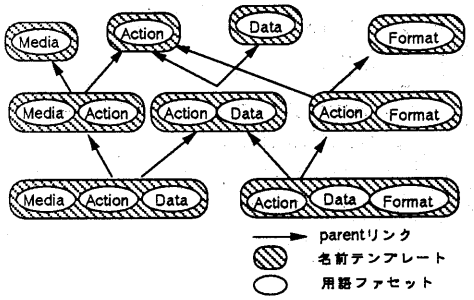

図 3: 名前テンプレートのネットワーク

4.2 検索に関する考察

4.2.1 キーワードの展開

キーワード検索をより強力にする手法は幾つか提案されているが、主なものは次の 2 つであろう。

1. KWIC(KeyWord In Context) における展開
   例えば、"substitute" というキーワードに対して、

   "substitute string"
   "substitute backspace to tab"

   というように、そのキーワードがいかなる文脈 (Context) で用いられているか、に注目して展開していくのが KWIC である。これを便宜上、横方向への展開と呼ぶ。

2. 用語の階層化における展開
   同様に、"substitute" というキーワードに対して、

   "substitute" -> 置換の一般概念
   "string" -> "data" -> 物の一般概念

   というように、キーワード (概念) 間の上位・下位関係に注目して展開していくのが階層化における展開である。これを縦方向への展開と呼ぶ。

以上を概念的に図で表すと 図5 のようになる。

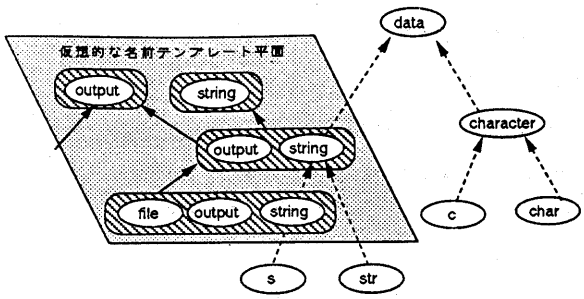

図 4: 2 種類のカルチャー知識

4.2.2 キーワード展開の定式化

いま、検索者は検索の初期段階において

"string を output したい" – (A)

という漠然とした要求をもっているとする。これは図4.1における名前テンプレート・ネットワークの中の

(output string) – (B)

に注目していることに相当する。(図6)

この時点で検索者は式 (B) のテンプレートに照合するような名前を検索する。そして、検索結果の中に目的の部品が存在するが、数が多すぎる場合には、より詳細な記述に変えて再び検索を行なわせる。

反対に検索結果が少ない場合には、よりおおまかな記述に変更して再び検索を実行する。 その際、前述した縦方向への展開と横方向への展開に相当する、以下のような 2 種類の展開が存在する。

1. 一般化 / 特殊化 (generalization/specification)
2. 抽象化 / 具象化 (abstraction/instanciation)

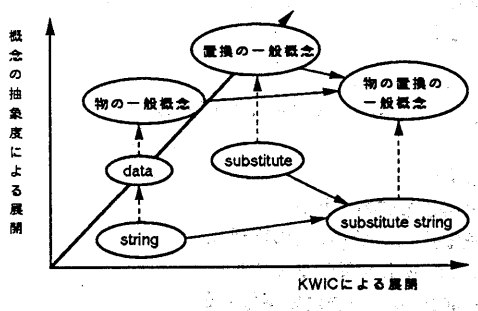

図 5: キーワードの展開

4.2.3 一般化 / 特殊化

いま、式(A)を

"string を file に output したい"

と、より詳細な要求に展開することは、図6 で注目している名前テンプレートの子テンプレートのうち

(file output string)

に注視点を移動することに相当する。(図7)

このように、子テンプレートのどれかに注視点を移動することを特殊化と定義し、その反対に親テンプレートのどれかに移動することを一般化と定義することにする。

4.2.4 抽象化 / 具象化

次に、"string" の上位概念として "data" が辞書に登録されている場合、(A)式を

"data を output したい"

と展開することは、図6 におけるテンプレート平面で string ファセットを data ファセットで置換し、

(output data)

とすることに相当する。(図8)

このように、注目する名前テンプレートは変更せず、その名前テンプレート中のある用語ファセットをその上位概念で置換することを抽象化とよび、その反対に用語ファセットを下位概念で置換することを具象化とよぶことにする。

この一般化 / 特殊化は前節における横方向の展開、抽象化 / 具象化は縦方向の展開に対応している。

4.3 検索アルゴリズム

上で定義した一般化 / 特殊化、抽象化 / 具象化の各オペレーションを用いると、検索アルゴリズムは以下のようになる。

```
begin
   適当な名前テンプレートに視点を移動する
   repeat
      検索実行
      if 検索結果が多すぎる
         具象化 or 特殊化を実行する
      else if 検索結果が少なすぎる
         抽象化 or 一般化を実行する
   until 目的の部品がみつかる
end
```

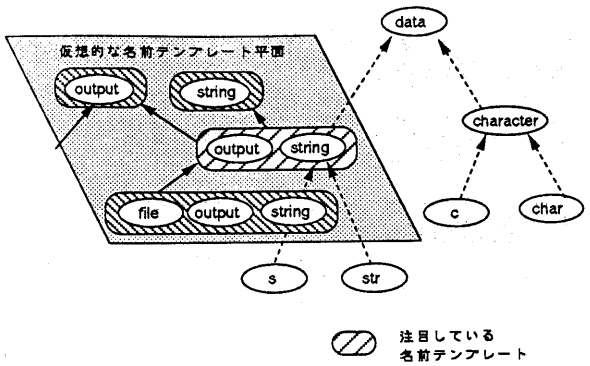

図 6: 初期状態

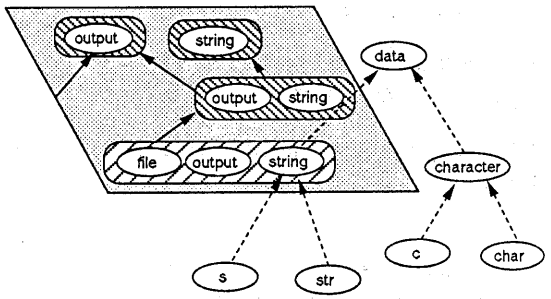

図 7: 特殊化操作を行なった後

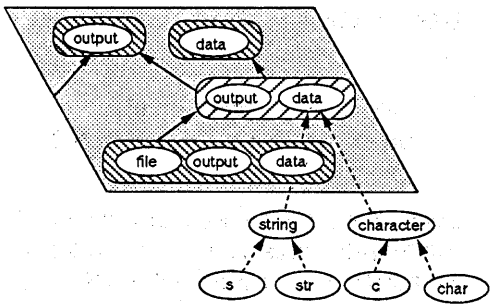

図 8: 具象化操作を行なった後

4.4 ソフトウェア再利用支援システム

現在開発中のソフトウェア再利用支援システムの全体構成は図4.4のようになっている。このうち検索モジュール以外のシステムの他の部分について簡単に説明する。

4.4.1 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 [6] を参照されたい。

4.4.2 Multidimensional Information Browser

データベース全体及び検索結果の表示には、Multidimensional Information Browser ([4]) を用いている。この Browser の特徴は、

1. 3次元グラフィックスを基に、色、形等を用いたデータの多次元表示
   本来は形状のないデータの集合に疑似形状を与え提示することにより、ユーザのメンタル・モデル形成の支援が可能
2. グラフィックス表示されたデータベースの全体像、細部を見れる
   ユーザはデータベースの大きさをイメージとして把握でき、システムに対する不安感を減少することができる
3. ある評価関数に基づいたデータの自動配置機能
   formal な関係以外の informal な関係の把握に役立つ
4. データグローブ等の入力デバイスを用いた直接操作が可能である

詳細は文献 [4] を参照されたい。

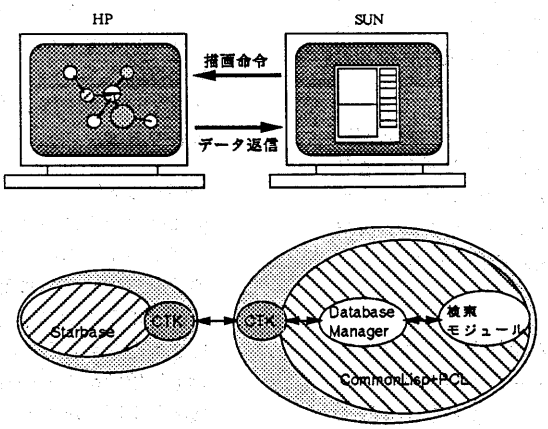

図 9: システム全体図

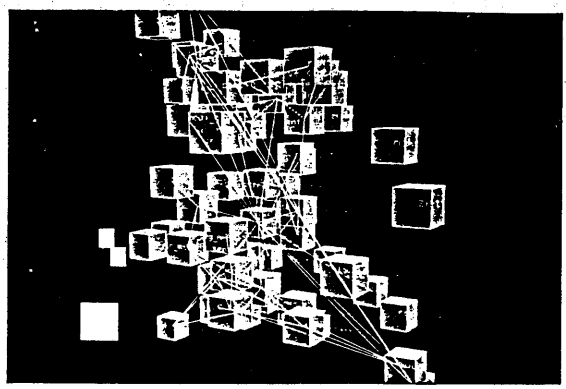

図 10: Multidimensional Information Browser の表示例

5 考察

今回試作した検索システムに関しての考察を以下に述べる。

- 用語ファセット管理
  実際のシステム、例えばEmacsライブラリで用いられている単語の出現頻度を調べて見ると、図5のようにほぼZipfの法則に従っていることがわかる。つまりこれは出現頻度が1の単語が非常に多いことを意味する。こうした単語をisaリンクだけで管理するのは困難かもしれない。しかるにリンクの不用意な増加は、管理の手間が増加するのみならず、検索時の操作が複雑になるという短所がある。

  例えば電子辞書編集委員会で作成中の辞書が完成すれば、一般的な単語に関してはそれで管理し、特殊な単語、省略形などについてのみ付加的に管理するようなシステムが望ましい。汎用の効果的な電子辞書の開発が期待される。

- 名前テンプレート管理
  部品に対するキーワードの付加という作業をなくした一方で、名前テンプレートの管理という新たな作業が必要となるが、この名前の管理は前述のように単に検索時のみならず、読解時の意図伝達に役立つものである。よってそのメリットの方が大きいと思われる。

- 実際のシステムへの適用
  実際のシステムの名前は必ずしも、すべてがきれいな名前付けをされているとは限らないため、本システムを適用しようとすると、名前テンプレートの数が非常に多くなってしまう。

  現在のシステムはこうした名前付けのカルチャーからはずれた名前を検索する手段を一切持たない。今後、これらをサポートする機能が必要だと思われる。

- ユーザ・インタフェース
  現在のシステムは一般化 / 特殊化、抽象化 / 具象化の各オペレーションに対するユーザ・インタフェースが整備されていない。これは今後の課題である。

- 分類への応用
  仮に全ての名前がこの用語ファセットと名前テンプレートから作られるとすると、その部品は名前テンプレートに従って一意に分類される。これは部品の数が増えてきた時の検索時間の短縮をもたらすと同時に、前述のBrowserによる物理的位置による分類表示が可能である。一々検索を実行しなくとも、Browserを眺めることにより部品を見つけることができるであろう。

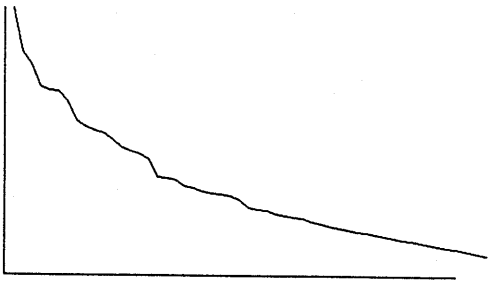

図 11: Emacs ライブラリに使用されている単語の出現頻度

6 結論

本稿ではソフトウェアの名前付けのカルチャーに基づく検索システムについて述べたが、現在、このシステムを実際のソフトウェア開発に適用するため、一般的な単語を集めた辞書と名前テンプレートを整備中である。

しかし、ソフトウェア再利用を本当に実際的なものにするためには、初めに述べた検索・読解・修正という再利用の 3 フェーズ全体に渡って支援する必要があるのはもちろんであるが、その際、今まで見過ごされがちであった informal な情報を積極的に活用していくことが今後の課題だと考えている。

参考文献

[1] 石井, 広瀬, 小池, “プラン・ゴール理論を用いたプログラム再利用支援システム”, 情報処理学会 第 36 回全国大会論文集, 1988.

[2] 石井, 広瀬, 小池, “ソフトウェア・カルチャーとプログラム再利用”, 情報処理学会 第 37 回全国大会論文集, 1988

[3] 石井, 広瀬, 小池, “プログラム読解支援に対する自然言語理解的アプローチ”, 情報処理学会ソフトウェア工学研究会,1988.

[4] 佐藤, 小池, 広瀬, 石井, “プログラム読解支援に対する自然言語理解的アプローチ”, 情報処理学会ヒューマンインタフェース研究会 (投稿中),199C

[5] 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.

[6] 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.

[7] R.Prieto-Díaz and P.Freeman, “Classifying Software for Reusability”, IEEE Software, vol.4, no.1,1988.

[8] R.Prieto-Díaz, “Classification of Reusable Modules”, Software Reusability Volume 1 Concepts and Models, (T.J.Biggerstaff, A.J.Perlis eds.),Addison Wesley, 1989.

[9] T.J.Biggerstaff, “Reusability Framework, Assessment, and Directions”, IEEE Software, vol.4, no.2, 1988.

[10] T.J.Biggerstaff, “Design Recovery for Maintenance and Reuse”, Computer,vol.5, no.7, 1989.

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

[12] 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.),Lawrenc Erlbaum Associates,1988.

[13] 伊藤, “情報検索”, 昭晃堂, 1986.

[14] 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.

[15] R.C.Waters, "The Programmer's Apprentice: A Session with KBEmacs", IEEE trans. on Software Engineering, vol.SE-11,no.11,1985.

[16] G.Booch, "Software Engineering with Ada", The Benjamin/Cummings Publishing,1983.

[17] A.Goldberg and D.Robson, "Smalltalk-80: The Language and its Implementation", Xerox,1983.