コンピュータと教育 41－2
（1996．9．20）

# プログラミングアプローチに対する定量化と分析システム

中嶋 正夫*，阪井 和男**，加藤 浩***
*明治大学情報科学センター，**明治大学法学部，***ＮＥＣ情報メディア研究所

学習者がプログラムを構築する過程のアプローチを解析するための手法を提案する。学習者のプログラミング環境として独自のビジュアル言語エディタを用い、プログラミング時の操作履歴を記録した。このエディタは、モジュールの階層的構築を容易に実現でき、トップダウン、ボトムアップのアプローチのプログラミングに柔軟に対応できる。

このエディタの操作履歴をもとに、モジュールの生成とパーツの生成の時間的関係、階層化されたモジュールの生成方法、パーツレベルでのプログラムのクローズネスの観点からトップダウン、ボトムアップアプローチの傾向や抽象化、具体化の傾向を数値化した。

# A Quantification and An Analysis System of Programming Approaches

Masao NAKAJIMA* , Kazuo SAKAI* , Hiroshi KATO**
*Meiji University , **NEC Corp.

A method is proposed to analyze characteristics in the processes of algorithm construction by students The operation logs are recorded in terms of original visual editor. This can be treated both approaches of top-down and bottom-up specifying hierarchic relations of the modules.

By analyzing these logs, it is clarified that the problem-solving approaches about several programming processes such as top-down, bottom-up, abstraction and specification are quantified.

## 1．はじめに

一般にプログラミングなどのような問題解決を行う際、トップダウンアプローチやボトムアップアプローチといった解決手法がとられる。このとき一貫して単一のアプローチを用いて解決するのではなく、状況や経験などに則っていくつかのアプローチを逐次使い分けていること（アプローチの融合）が経験的に明らかになっている[1)]。

我々は、特定のプログラミング言語に依存しないアルゴリズム教育を支援するITS(Intelligent Tutoring System)の開発を行っている。このシステムは、従来の多くのシステムのように学習者が完成させたプログラムを評価するのではなく、学習者がプログラミングを行っている最中に逐次その意図を把握し、アドバイスをすることを目指している。問題解決の過程において、学習者がとったプログラミングアプローチの同定や定量分析が可能となれば、画一的なアプローチの採用や行き詰まり現象を把握でき、それに対してアドバイスしたり、学習者の思考過程の解析が可能となる。

アプローチの中で一般的に用いられるものには、トップダウンアプローチとボトムアップアプローチがある。トップダウンアプローチは、プログラムを抽象的な機能ブロックに分割したり、大まかな実行可能なプログラムを作成してから、より具体的で詳細なプログラムを作成していく方法である。一方、ボトムアップアプローチは、いきなりプログラムの詳細部から作成し、徐々に全体像を形成していく方法である。

今回は、教育支援システムの一部であり、また学習者のアプローチの抽出のためのデータを取得するためのエディタ／シミュレータを開発した。そして、このシステムを使用してプログラミング操作の履歴を記録し、それらをいくつかの観点で分類を行い、その履歴を図的に表現し、アプローチの傾向を数値化する手法を開発した。この図を解析することにより、トップダウン、ボトムアップといった、学習者が問題解決する過程でのアプローチの特徴や潜在的な傾向の解析が期待できる。

このトップダウン、ボトムアップアプローチの解析に関しては、従来はモジュールの生成とその内容の詳細化の相互関係で述べられてきたが、我々の研究ではモジュールの生成の方法に関してもトップダウン、ボトムアップアプローチがあることを示し、その定量分析について提案する。ここでいうモジュールとは、プログラムの構成要素であるパーツを機能単位にまとめたパッケージを指す。また、入力、出力、計算、代入などのプログラムの構成要素であるパーツの生成の方法に関しては、「プログラムが最低限実行可能である形にいつの時点で完成させるか（これをプログラムのクローズネスという）」に着目すると、それはトップダウン、ボトムアップアプローチと同じように、プログラムの抽象度や詳細化に関するアプローチが数値化できることも報告する。

## 2．プログラミングアプローチ

一般に、ある程度の規模のアルゴリズムを構築するには、処理を機能別にモジュールに分け、段階的に詳細化していくトップダウンアプローチと、具体的で詳細な知識を積み上げていくボトムアップアプローチがある。これは、いい換えるとプログラムの抽象化から具体化へのアプローチと、その逆の具体化から抽象化へのアプローチである。他にもいくつかのアプローチがあるが、本報告では上述のアプローチについてのみ言及する。

ここで、モジュールの生成だけに着目すると、トップダウン的な生成法やボトムアップ的な生成法がある。たとえば、あるプログラムを構築するのに、いくつかの階層を持ったモジュールを作成する場合には、

(1)トップモジュールを作成してから、順番に下位モジュールを作成していく方法

(2)最下位のモジュールを作成してから、それらを上位モジュールの中に入れていく方法

の２種類がある。(1)はトップダウンアプローチであり、(2)はボトムアップアプローチである。

さらにパーツの作成方法について着目すると、前述のクローズネスが抽象化から具体化へのアプローチか、具体化から抽象化へのアプローチかを示すと思われる。プログラムにおいて、構成要素であるパーツ、つまり「入力」、「計算（代入）」、「出力」の作成過程に注目すると、

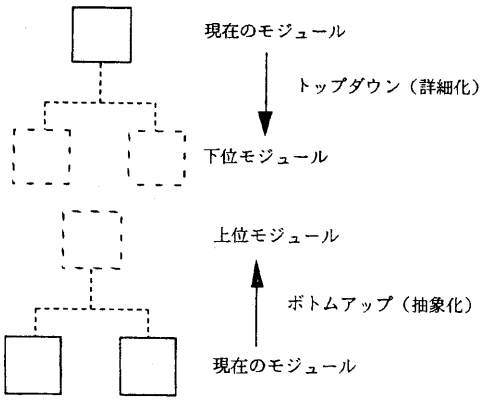


図１　モジュールの抽象化と詳細化

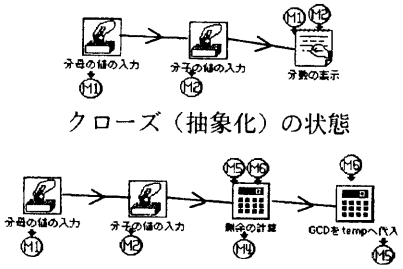


図２　パーツのクローズネスの概念

あるプログラムについて、「入力」、「出力」のパーツが作成されれば、そのプログラムが一応動作する。このときプログラムのクローズネスが完了したとみなすことができる。パーツレベルで考えると、これは詳細がまだ作成されていないが、プログラムの構成が形作られているので、抽象的なレベルといえる。一方、「入力」や「出力」のパーツを作成せずに、「計算（代入）」を作成している過程は、プログラムの詳細化を行っているといえる。つまり、プログラムの具体化をまず行っていることになる。

## 3．分析システムと定量化解析

### (3.1)　ビジュアルエディタの概要

現在、学習者がプログラムを作成する上で用いる、さまざまなアプローチの解析の枠組みを検討している。そのために、モジュールの階層関係を明示し、トップダウン、ボトムアップアプローチなどを柔軟に併用できるエディタとシミュレータの統合環境を構築した[2]。

本エディタの概観を図３に示す。システムは、研究・開発中の教育支援システムのエディタ部（ユーザ入力部）に相当する。なおこのエディタは、Macintosh の HyperCard 上に実現した。

プログラムの構成要素である基本パーツ、制御構造のパーツ、モジュールはその性格ごとに分類し、エディタの左上に配置した。また、変

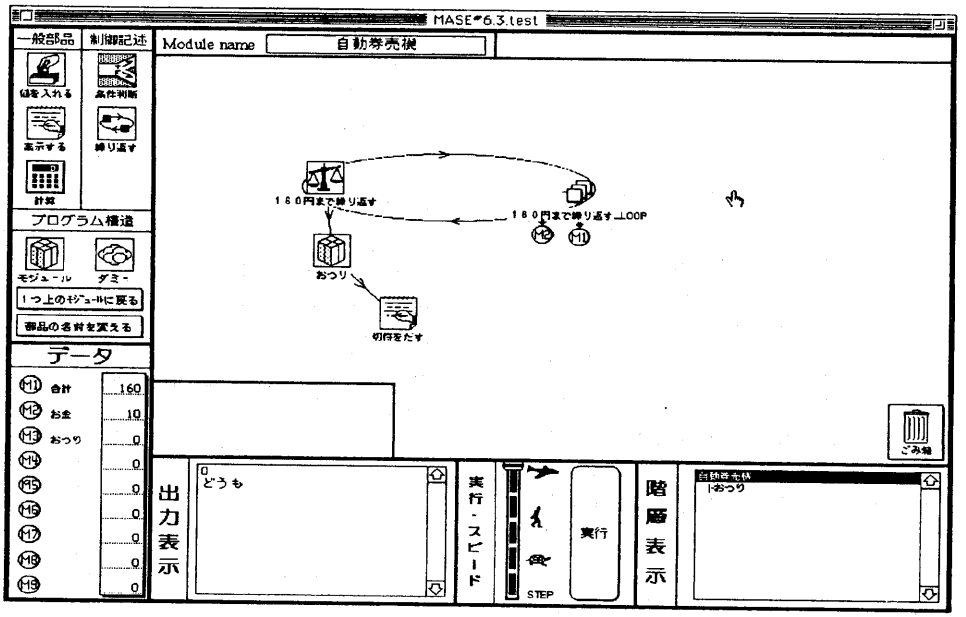


図３　エディタの概要

数データや数式、条件式の入力欄、そして実行結果の出力欄のようなユーザに必要な値の表示を行うものは、エディタの左下に集中表示させた。そして、実行ボタンやモジュールの階層表現などのエディタの操作系を右下に配置し、初心者がなるべくエディタの操作に困惑しないように考慮した。

エディタの操作に関しては、対象となるパーツが異なってもマウスのダブルクリック、ドラッグなどの操作は同じ働きを持つことによって操作を統一化したり、あるパーツに対する操作は、なるべくそのパーツを直接操作するなど、視点があまり変わらないようにした。

以下にパーツに対する操作方法の具体例を示す。

・パーツに対してのダブルクリック
→パーツの数式、条件式、出力データ、モジュールの詳細表示など、パーツの内容に関わる操作
・パーツに対する[option]+ドラッグ
→操作をしたパーツを始点として、マウスボタンを離したところにあるパーツまでの実行順序を定義
・パーツをごみ箱へドラッグ
→パーツを削除する

このほかに本エディタは、次のような特徴を持つ。

(1)初心者にも抵抗なくまた直感的に操作ができ、プログラムの構造が把握しやすいように、独自のビジュアル言語を用いた。
(2)モジュール内のパーツの詳細化や、パーツ群のモジュール化を明示的に表現した。
(3)モジュール同士の関係、つまり階層構造をわかりやすく視覚的に表現した。また、この構造図のモジュール名をクリックすると、直接そのモジュールに詳細画面に移動できる。
(4)プログラムに記述する計算式や条件式を電卓に似たインタフェースで入力するようにした。
(5)パーツにおいて使われる入力データや出力データをオブジェクトとして定義するようにし、その変数データを学習者に対して明示的に表現し、その値の実行時の変化をリアルタイムに表示する。
(6)学習者のプログラミングアプローチの解析のためのデータ収集のために、学習者がエディタの操作をすると、その時刻や操作内容が自動的に記録できるようになっている。

(3.2) エディタによるプログラミング

学習者は、画面左にある「一般部品」、「制御記述」、「プログラム構造」、「データ」にあるアイコンをマウスでワークエリア上に移動し、リンク線でパーツの実行順序を決定したり、パーツにデータアイコンを接続することによって、プログラムを構築する。なお、計算式や条件式は、電卓にアナロジーをとった入力インタフェースによって入力する。また、モジュールを示すパーツをダブルクリックすると、下位構造へ移動し、モジュールの内容が開示されて編集が可能になる。

ここで、パーツやモジュールで使用する変数データは、入力データであるのか、出力データであるのかをパーツごとに定義する。これは学習者に、パーツ間でやり取りするデータを明確に意識させるためである。また、このデータは(M1)や(M2)といった抽象的な名前があらかじめ付けられて用意されている。これは、変数を電卓のメモリ機能のメタファーとして対応させているためである。しかし、学習者は変数データを定義するときには、各変数に自由に名前を付けることによって具体的に意味付けることがで

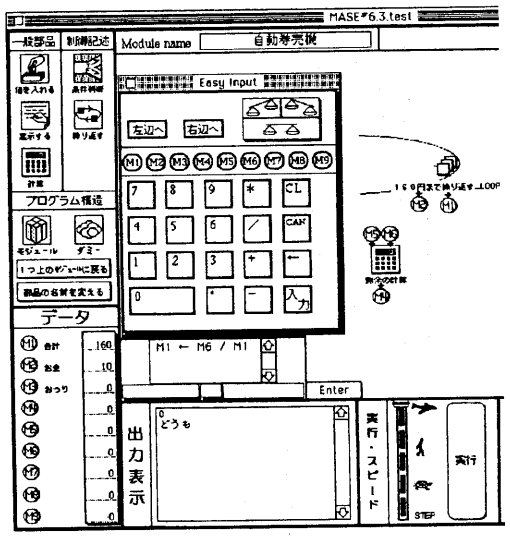


図4 プログラミング例

きる。現在、本エディタでは変数データは9つまで使用可能である。これは、このエディタで無理なく作成できるプログラムの規模をもとに決めた。

このシステムは、学習者の一連の操作の履歴を時刻とともに記録できる。そして、このデータを分析することによって、例えば今どのパーツに着目しているか、どこで迷いが生じているのか、どのようなアプローチで解こうとしているのか、などといった学習者の意図を以下に述べる手法によって、ある程度定量的にくみ取ることができる。

(3.3) プログラミングアプローチの抽出

今回着目したプログラミングアプローチは、抽象化から具体化へのアプローチとその逆の具体化から抽象化へのアプローチである。それは、一般的にいうトップダウン、ボトムアップアプローチに加え、モジュールの生成に関するトップダウン、ボトムアップアプローチ、そしてプログラムのクローズネスに関するアプローチである。

プログラミングアプローチの抽出と解析のために、実際にエディタを使い、ある問題についてプログラミングをした。本エディタでは、前述のように操作履歴をとっている。具体的には、

(a)入出力や計算といった基本パーツの操作に関するもの
（例：基本パーツの生成、削除、移動など）

(b)プログラムのモジュール化に関するもの
（例：モジュールの生成、削除、複数のパーツのモジュール化など）

(c)プログラムの制御構造に関するもの
（例：実行順序の指定、条件分岐／ループなどの定義など）

(d)データに関するもの
（例：データの生成、削除、移動など）

(e)その他
（例：テストラン、エディタのスコープの移動など）

これらの項目のうち、トップダウン、ボトムアップアプローチに関連する(a)〜(c)について履歴を抽出し、解析を行った。

(3.4) 操作履歴によるプログラミングアプローチの解析

モジュールの生成とその内容の詳細化の作業の時間的な関係で述べられるトップダウンやボトムアップアプローチの解析は、モジュールの生成・削除を縦軸とし、パーツの生成・削除を横軸とした2次元平面上に、それぞれに関係する操作が発生するごとにある一定量点を移動させ、その軌跡をプロットすることにした。

エディタの操作履歴を図に示すために、座標軸に対応する操作の種類を以下のようなそれぞれ相対する2種類の操作の組に分類した。

a. モジュールを生成する操作 (a+)
　　←→ モジュールを削除する操作 (a-)
b. 基本パーツを生成する操作 (b+)
　　←→ 基本パーツを削除する操作 (b-)

実際に2次元平面上にプロットする方法は、モジュールに関する操作のプロット同様に、原点を始点として操作の時間的順序で、a+ の操作はy軸の＋方向に、a- の操作は－方向、b+ の操作はx軸の＋方向に、b- の操作は－方向に一定量、点を移動させることとした。

図5は、「任意の分数を約分するプログラムを作成せよ」という問題に対して、(1)意図的にトップダウンのみで作成する、(2)意図的にボトムアップのみで作成する、(3)トップダウン、ボトムアップのアプローチを適宜組み合わせて作成するの3種類のアプローチで解いた結果である。

(1)のトップダウン・アプローチによる解法の履歴では、はじめにモジュールを作成し、後半に個々の部品アイコンや変数データを作成する傾向があるため、まず上に移動してから右方向に軌跡が描かれた。また、(2)のボトムアップ・アプローチでは、前半にパーツの作成といったプログラムの具体的な部分の作成を行い、後半にそれらをモジュールにまとめる傾向があるため、まず右移動してから上方向に軌跡が描かれた。そして、(3)の特に意図を持たないアプローチでは、階段状に右上がりの軌跡が描かれた。この図によって、トップダウンやボトムアップのアプローチの使い分けについて、明らかな定

性的な違いが確認された。つまり、この軌跡を分析することで、大まかにどのようなアプローチをとったかを推測することができる。

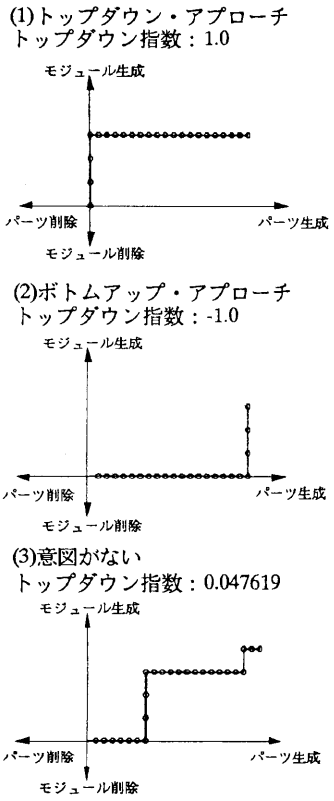


図５　モジュールの生成とパーツの生成の関係によるアプローチの解析

次に、定量的に操作履歴を分析するために、図６のようにチャートの始点から終点までを結んだ直線とプロット結果との面積差を求め、それを始点から終点までの直線を斜辺とする直角三角形の面積で除算し正規化することにより、アプローチの特徴を数値化した。その結果についても図５に示した。この値をトップダウン指数と呼ぶ。極端なトップダウンアプローチでは、トップダウン指数は＋１となる。また極端なボトムアップアプローチでは、トップダウン指数は－１となる。つまり、トップダウン指数は－１から＋１の値のとる。

このトップダウン指数の正負やその値の絶対値の大小から、学習者のとったトップダウン、ボトムアップアプローチの種類とその程度がわかる。

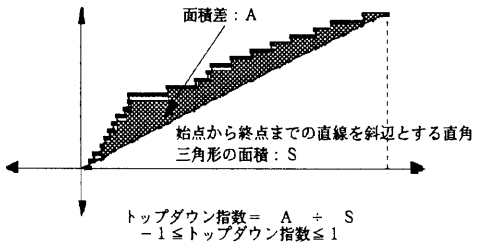


図６　トップダウン指数

しかし、トップダウンアプローチをプログラムの抽象化から具体化への作業、ボトムアップアプローチをその逆の作業とすると、モジュールの生成だけに関してもトップダウン的な生成法やボトムアップ的な生成法が存在する。

本エディタでは、あるモジュールの中にスコープを移動し、その中身に下位モジュールを作成していくトップダウン的なモジュール構築法と、モジュールのアイコンに別のモジュールを重ねることによって、そのモジュールの中に別のモジュールを入れることによるボトムアップ的な構築法が容易にできる。よって、以下のようにモジュールの生成に関するトップダウンアプローチの度合いを示す数値を定義した。

プログラミングをはじめる時点で数値は０であり、あるモジュールに対して、

・モジュールの中に下位モジュールを作成したときには、数値は変えない。
・モジュールを別のモジュールの中に入れたときには、数値を１マイナスする。

とした。これは、数値のマイナス値が大きくなればなるほど、学習者はより下位のモジュール、つまりより詳細なモジュールから作成する傾向があることを示す。これはボトムアップアプローチである。一方、この数値が０からあまり変化しなければ、トップダウンアプローチでモジュールを作成していることを示す。

以下に分析例を示す。ここでのデータは、トップダウン指数のときと同様に「任意の分数を約分するプログラムを作成せよ」という問題について、(1)モジュールを作成してから各モジュ

ール内の詳細なパーツを作成するという極端なトップダウンアプローチと、(2)実行可能なパーツ群を生成してから、機能ごとにモジュールにまとめていくという極端なボトムアップアプローチでいくつかプログラムを作成したものである。

| | トップダウン指数 | モジュールの作成に関するトップダウンアプローチの度合い |
|---|---|---|
| 実行例1 | +1．0 | 0 |
| 実行例2 | +1．0 | −2 |
| 実行例3 | −1．0 | −2 |

表1　トップダウン指数とモジュール生成に関するアプローチの度合い

実行例1は上位モジュールから下位モジュールの順に生成し詳細化したもので、実行例2は下位モジュールを生成してからそれをまとめる形で上位モジュールを生成した結果である。この2つの実行例は、モジュールの生成とパーツの生成の時間的関係から判断すると、2つともトップダウン指数は1．0となり、トップダウンアプローチを表しているが、モジュールの作成のアプローチの違いが、この分析法からわかる。

モジュールの生成や階層構造化をすることは、プログラムの構造を構築することである。この数値は、プログラムの階層構造をその時点でどのようなアプローチで構築しているかの基準になる。

次にパーツの作成方法について着目すると、前述のとおりプログラムのクローズネスが抽象化から具体化へのアプローチか、具体化から抽象化へのアプローチかを示す。このクローズネスの度合いを数値化するために、プログラムにおいてモジュールの枠を外して、プログラム全体の「入力」、「計算（代入）」、「出力」の作成過程に注目した。あるプログラムについて、「入力」、「出力」のパーツが作成されれば、プログラムのクローズネスが完了したとみなす。

一方、「入力」や「出力」のパーツを作成せずに、「計算（代入）」を作成している過程は、プログラムの詳細化を行っているといえる。ここで、本エディタでの「入力」と「出力」に対応するパーツは、それぞれ「値を入れる」と「表示する」である。

以上のことから、プログラムのクローズネスの状態を次のように数値化した。

・プログラム全体で、「入力」と「出力」の両方のパーツが作成されるまで、「計算（代入）」のパーツが作成されるごとにクローズネスの数値を1ずつ増やす。

このクローズネス数値が大きければ大きいほど、学習者はパーツレベルで詳細化を行っていることが示される。また「入力」、「出力」のパーツが生成された時間がわかるので、プログラムを作成してからどのくらいの時間でプログラムのクローズネスが完了したかがわかる。

以下に、課題「１６０円専用の自動券売機の基本動作モデルをプログラムせよ」という問題についてのプログラムのクローズネスに関する結果を示す。被験者は2人で、被験者Aは初心者、被験者Bはある程度のプログラミング経験者である。

| | クローズネスの値 | クローズネスが完了した時間 | プログラムが完成するまでの時間 |
|---|---|---|---|
| 被験者A | 2 | １０分５８秒 | ５０分５６秒 |
| 被験者B | 2 | ３分２２秒 | １５分４０秒 |

表2　初心者と経験者とのクローズネスの違いの一例

クローズネスの数値は両者とも2であり、違いがないが、被験者Aは、クローズネスを完了するまでに比較的時間が経過している。一方、被験者Bはクローズネスの完了までの時間が短い。これは、この経験者はまずプログラムの構造を把握しながらメインの部分を作成していくのに対し、この初心者はパーツの生成と削除を繰り返すなど試行錯誤が多いためである。

次に、被験者Bに対し課題A「１６０円専用の自動券売機の基本動作モデルをプログラムせよ」と、課題B「任意の金額の切符が発行できる自動券売機の基本動作モデルをプログラムせよ」を解いたときの結果を示す。

| | クローズネスの値 | クローズネスが完了した時間 | プログラムが完成するまでの時間 |
|---|---|---|---|
| 課題A | 2 | ３分２２秒 | １５分４０秒 |
| 課題B | 5 | ４分３８秒 | ３０分０７秒 |

表3　プログラムの規模によるクローズネスの違いの一例

結果より、課題Bのほうがクローズネスの数値が大きくなっている。これは、プログラムの規模が大きくなったので、いくつかのモジュール構造を構築し、ある程度の詳細化を行った後でメインのプログラムを構築した結果である。

出題した課題に対してのパーツの数は、およその予想がつくため、学習者によって作成されたパーツの数やクローズネスの状態から、プログラムの抽象化をしようとしているのか、それとも詳細化をしようとしているのかといった学習者の思考過程を逐次読みとることができる。

トップダウン、ボトムアップアプローチや抽象化、詳細化のアプローチの分析については、以上の3つの観点を用いることによって、一般的なモジュールの生成とパーツの生成の時間的関係のみから判断するよりも詳細な分析が可能である。

特に、ある問題に対して標準的なモジュール構造、パーツの数などの条件をあらかじめ決めておけば、モジュールの階層構造に関するアプローチやプログラムのクローズネスに関するアプローチについて逐次その傾向を評価することができる。これらのアプローチの詳細な解析や相互関係の分析ができれば、現在開発中の教育支援システムに、学習者のアプローチの傾向を逐次同定する機能を付加し、学習者に対してのアプローチに関するアドバイスが可能となる。

## 4．おわりに

今回は、プログラミングにおけるトップダウン、ボトムアップアプローチや抽象化、詳細化のアプローチの抽出と定量解析について述べた。そのために我々は、さまざまなアプローチに適応でき、操作履歴を記録するビジュアルエディタを開発した。

プログラミングアプローチの解析に関しては、一般的なパーツの生成／削除、モジュールの生成／削除の2つのプログラミングに関する操作履歴を図的に表現することにより、一般に言うトップダウン、ボトムアップアプローチについて解析できる可能性を示した。また、モジュールの生成とその詳細化の相互関係で述べられてきたトップダウン、ボトムアップアプローチに加え、我々の研究ではモジュールの階層構造の構築に関するアプローチの分析や、プログラムのクローズネスに関連した抽象化、詳細化のアプローチがあるということを提案した。そして、これらを数値化することにより、定量的な解析手法を示した。

今後は、プログラミングの被験者を対象とした実験を行い、学習者が陥りやすい画一的なアプローチの採用や行き詰まり現象などの学習者の思考過程を解析し、それに対してアドバイスをおこなう教育支援システムを開発する予定である。

【謝辞】

本研究にあたり、ご協力頂いたＮＥＣ文教システム事業部鈴木栄氏に感謝する。

【参考文献】

1)宮本　勲：”ソフトウェアエンジニアリングの現場展望”、ソフトウェアリサーチアソシエイツ.

2)中嶋　正夫他：”アプローチ指向の教育支援のためのビジュアル言語エディタとシミュレータの統合”、情報処理学会　第47回全国大会論文集　(1)、p. 35-36 1993.

3)中嶋　正夫他：”操作履歴のチャートによるプログラミング・アプローチの分析”、情報処理学会　第48回全国大会論文集　(1)、p. 45-46 1994.