意思決定に基づくプログラム設計支援について

2N－1

鈴木 彦文　関本 理佳　海尻 賢二

信州大学 工学部
E-mail:csuzuki@cs.shinshu-u.ac.jp

1　はじめに

一般的にプログラムの開発方法を考える時、個々の処理における制約条件 (constraint) に重点をおくケースが多い。しかしこれはあくまでもプログラムの正当性を保証するためのものである。このため制約条件を基礎とした知識ベースを用いて実際にプログラム開発を行なおうとする場合、小さなプログラムを開発する場合においても多くの条件を作成しなくてはならなかったり、設定した条件下でのリーズニングも理解し易いとはいえない。また実際に人間の考える仕様記述と、知識ベースを適用するための記述方法とはかなりの差がある。そして *cliché*[1] として登録しておかない限り、これらの問題は回避できない。

本研究では個々の処理を、正確性を保証された *cliché* によって構成するだけでなく、ユーザーから見た方向での知識を構成し利用する必要があると考える。そして、ユーザー側の知識を含んだ知識ベースを用いた指導を考案する。ここで用いる知識ベースはプログラム開発中に生じる意思決定・詳細化に注目し、また、実際のコード化を行なうためにゴール/プラン [2] を利用する。そして具体的なプログラムの作成においては、抽象構造によるプログラミング方法を考案する。

2　抽象プログラム構造

本研究で用いる抽象的なプログラム構造は次の6つのパターンがある。いずれも処理の構成がどのようになるのかを表している。

Serial: 複数の処理の順序が固定している構成を表現する。**Parallel:** 複数の処理の順不同な構成を表現する。**Goals:** ステートメント中にステートメントを定義する構造を表すためのもので、主にループ処理に対応する。**Disperse:** 場所的には分散しているが、実際にはひとまとまりの処理であることを表す。**Procedure:** 手続きを利用する場合に用いる。**Goal:** 具体的なプログラム断片の情報を持つ処理を表す。

以上の構造において、Goal を葉とした抽象プログラム構造から構築される木を、プログラム木 (**Program tree**)[3] と呼ぶ。実際にはユーザーは抽象構造と知識ベースを基に、具体的なプログラムを構築することになる。

3　知識ベース

指導や支援を行なう際、具体的な情報を持つ知識ベースは不可欠である。本研究における知識ベースは、意思決定・詳細化というプロセスが、実際のプログラム化においてどのように反映しているのかを表すものである。このために次の情報を保持する必要がある。

意思決定:ユーザが行なう設計プロセス。さまざまな決定を考慮する必要があるが、現在はアルゴリズムとプログラム構造について考慮している。いずれは、データ構造に関する決定を盛り込む。**詳細化:**手順を細分化することで具体化して行くプロセス。**プログラム構造:**意思決定・詳細化の結果、どのようなプログラム構造となるのかを、抽象的プログラム構造を用いて表す。**識別子情報:**どのような識別子が必要となるのかの情報を表す。**説明文:**意思決定・詳細化、また、抽象プログラム構造や識別子の説明を自然言語で記述しておく。

以上の情報を表している木構造を**意思決定木** (**Decision tree**)[3] と呼ぶ。意思決定木は意思決定・詳細化の情報を木構造で表現し、木の各ノードにおいてより具体的な情報を保持している。この意思決定木が知識ベースの中核を成している。

4　指導とエディタ

ここでいう指導とは、単にプログラムのコード化の支援を行なうのではない。あくまでもプログラムの設計段階での支援を含めた指導を、意思決定・詳細化という立場から指導するのが目的である。

プログラムの開発段階には、(1) アルゴリズム設計、

Program Design Tutor based on Decision Making
Hikofumi SUZUKI , Rika SEKIMOTO , Kenji KAIJIRI
Faculty of Engineering, Shinshu University
500 wakasato,nagano 380, Japan

(2) プログラム設計、(3) プログラムコード化、の3つの局面が最終的に存在する。もちろんこの前段階として、要求解析、仕様作成などがあり、また詳しく見れば、(数学的) モデルの設定、アルゴリズムの検証があるが、ここでは前者は特筆せず、また、後者は全てではないが (1) アルゴリズム設計に含まれると解釈する。またテスト・デバッグ作業を行なう段階がある。以下の指導の過程で生じたデータをこの作業に役立てるという方法をとるが、今回は特筆しない。

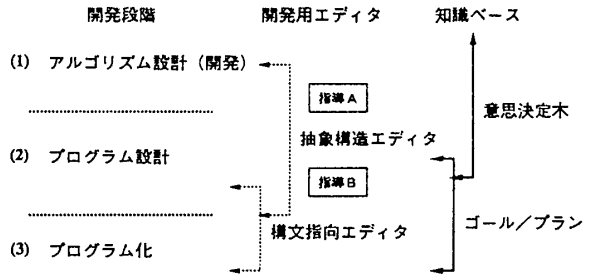

図 1: 開発段階と指導

図1は、開発段階と対応するエディタ・知識ベースを表している。広く知られている構文指向エディタは、プログラム設計から具体的なコードを入力する過程で用いられる。しかし、アルゴリズム設計・プログラム設計を補助するものではない。この設計段階におけるエディタとして、抽象構造エディタを提案する。この抽象構造エディタは、抽象構造をユーザーに提供し、これの組み合わせを促すことでプログラム設計を支援するものである。この時必要がなければ具体的なプログラム化（コード化）は行なわない。ユーザーは必要なネーミングを施したり、視覚的に構造を組み合わせることにより処理を完成させてゆく（図2)。また、抽象構造エディタは、最終的にはゴール／プランに基く知識を用いることによりプログラム化の支援も行なうことができる。

この抽象構造エディタを用いた場合における指導として、図1の指導 A と指導 B を考案している。その内容は次の通りである。

指導 A プログラムの設計段階における指導。プログラム設計段階における意思決定・詳細化プロセスをベースとした指導となる。抽象構造で構築されるプログラム木からユーザーの行なった意思決定・詳細化を汲み取り適切な指示を出す。また、ユーザーからの問い合わせにトップ・ダウン的に答えて行く。

指導 B 設計から具体的なプログラム化を行なう時の指導。実際にコード化する際の具体的な方法の提示や

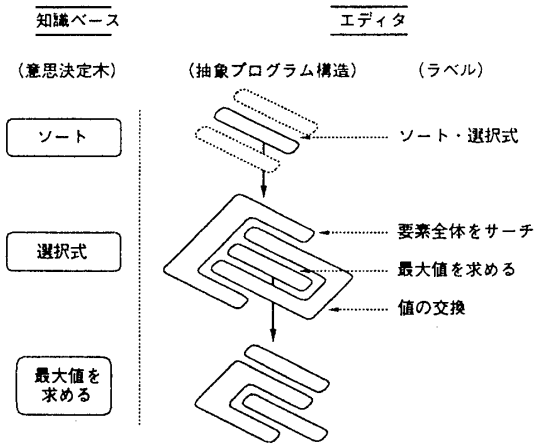

図 2: 抽象プログラムと知識ベース

指示を行なう。

これらの指導は Apprentice を基本としている。このためユーザーの状態を把握するための高度な認識が必要となる。

5 まとめ

以上述べてきた意思決定や詳細化といったプロセスに注目した知識ベースを基に、よりユーザーサイドに立った指導を施すことで、より模範的で well-formed なプログラムを構築することが可能となる。また、多数のパーツではなく基本的なパーツのみを提供することで、問題の本質であるアルゴリズムの問題と、コードに応じたテクニックを切り離して考えられる。また、ある程度パターン化されたコードを登録しておくことで、コード化の学習にも役立ち、またユーザーの手間を省くこともできる。また抽象的な構造を用いることで、要求・仕様の段階から、コード化に至るステップの中間的な存在としても活用できる。

今後は指導方法とエディタ仕様の充実化、知識ベースのルール検証を行なう。

参考文献

[1] C. Rich and R.C.Waters :*A Research Overview*, COMPUTER, Vol 21, No.1 (1988).

[2] W.Lewis Johnson,etc :*Understanding and Debugging Novice Programs*, Artifical Intelligence 42 (1990)

[3] 鈴木 彦文:プログラム設計指導システムにおけるデータベース, 電子情報通信学会技術研究報告 KBSE93-12(1993-07).