プログラム相談プロジェクトConsultの全体構想

3N-7

佐藤眞木彦*1, 斉藤　哲*1, 岡本匡人*1, 小林正和*1, 垂井良平*1,
大西　淳*2,　　島崎眞昭*2
(　*1富士通株式会社, *2京都大学　)

1. はじめに

京都大学大型計算機センターでは、計算機利用についての相談サービスを行っている。この相談には、ハードウェア／ソフトウェアの進展と多様化に伴って、種々の知識を必要とし、専門家でなければ十分な対応が出来なくなってきている。また計算機利用の普及に従い、相談の量も増大の一途を辿っている。相談員の負担を少しでも軽減させると共に、利用者にとって、プログラム開発を効率良く行わせるために、相談活動の計算機支援が急務となっている。

我々は、プログラム相談の支援に関する研究プロジェクトを進めており、「FORTRAN に関するプログラム相談の自動化」を研究の対象としている。これは、大学等での計算機の利用は、FORTRAN による科学計算が大部分であるためである。さらに具体的には、コンパイル時と実行時のエラーの対処を目標として、それぞれ実現化を目指しているので報告する。

初心者が計算機を使ってエラーを起こした場合、計算機はエラーであることだけを告げるが、初心者はどうしていいか判らない。これは、初心者が計算機を習い始める時の大きな問題であると思われる。計算機の基本ソフトウェア（ＯＳ，コンパイラなど）は、正常な処理を効率良く行うことが目的であり、エラーについてはエラーコードを（及び、判り難いメッセージを）出すのみである。ある程度熟練していれば、エラーを適当に対処出来るが、初心者はそうではない。このような初心者を手助けすることが、このプロジェクトの目的である。

コンパイルエラーの相談については、現在プロトタイプを作成し、評価中である。ここでは、プロジェクトの全体的な構想、コンパイル時のエラーの具体的な対処法、実行時エラーの対処法、今後の課題などについて述べる。

2. プログラムの開発と相談の支援

プログラムの開発は、一般に図１のような段階をとる。相談活動の計算機支援を段階ごとに分類して検討した結果、最初の３段階、すなわち要求定義、設計、コーディングの相談に対しては、プログラマや開発者の意図を計算機上に取り込む必要があり、現状では機械化して一元的に扱うことは非常に困難であることが判明した。また、実行テストで、プログラムの動作がプログラマの意図したものと異なる場合のエラーの対処は、上記の問題を含むので困難であることが判明した。

従って、実行テストやデバッグ時のエラーの中でもコンパイル時のエラーと実行時の異常終了やエラーに対象

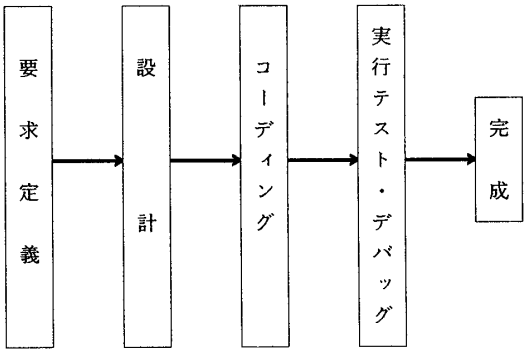

図１　プログラムの開発段階

を絞り、その相談支援手法とそれに基づくシステムの開発を行っている。システムの利用者としては、主に初心者を念頭に置いている。利用者の習熟度や特性を学習者モデルとして用意することによって、習熟度と特性に応じた対処を可能とする手法の実現も検討中である。

3. コンパイルエラーの相談

FORTRAN の初心者にとって、コンパイルエラーのデバッグは、次の理由で困難である。

- 文法知識が十分にない
- エラーコードからエラー原因を推測できない
- エラーメッセージの意味が判らない
- 正しい文でも他のエラーによって、エラーコードが出されることがある

このため、我々は次の点に特に注意してコンパイルエラーについての相談手法を確立し、それに基づいたシステムを作成中である。

i）エラー原因をソースプログラムを基に推論する。この時、コンパイラの解析とは独自に、構文解析を行う。例えば、ブランクもデリミタとして解析することにより、コンパイラでは判定できない誤りも検出し解析できる。

ii）エラーコードごとに推論のルーチンを用意する。

iii）あくまでコンパイラの外付けのソフトウェアとする。従って、コンパイラの出力情報（クロスリファレンス等）を最大限に利用する。

iv）原因が１つに絞り切れない場合は、推論結果を利用者に提示し選択させる事で、より正しい結果を得る。

これらによって、エラーコードだけからは特定不可能なエラーの原因を推定出来るようになり、従来の相談手法やシステムよりも実用的であると考えている。

OUTLINE OF PROGRAM CONSULTATION PROJECT : CONSULT

Makihiko SATO*1, Akira SAITO*1, Masato OKAMOTO*1, Masakazu KOBAYASHI*1, Ryouhei TARUI*1
Atsushi OHNISHI*2, Masaaki SHIMASAKI*2
(*1FUJITSU LTD.　*2KYOTO UNIV.)

4．コンパイルエラーの推論の制御

エラーメッセージが、実際には間違っていない文に対してコンパイラにより出される事が多々ある。これは、そのエラーが検出された文の前に重大なエラーがあり、コンパイラがその文の解析を打ち切ったため、その文に含まれる情報をコンパイラが認識出来ない場合などに生じる。このようなエラーの原因を初心者が発見するのは困難である。我々はこれを波及エラーと名付けた。波及エラーの伝播のしくみを図２に示す。

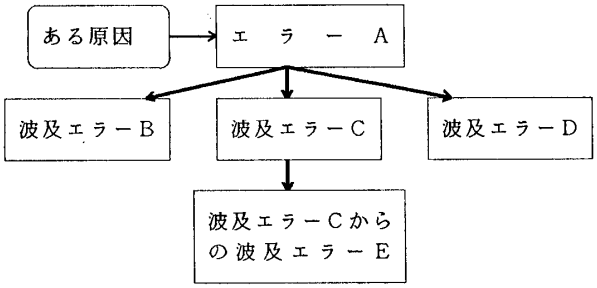

図２　波及エラーの伝播

これらのエラーを効率的に扱うために、波及エラーについての推論を工夫した。例えば、波及エラーＢについての相談を受けた場合、エラーＡからの波及エラーであることを検出すると共に、Ｃ，Ｄ，Ｅのエラーが波及エラーであることを検出する。

また、１つの原因から２つ以上のエラーコードが発生する事がある（図３）。このため、あるエラーの原因を修正する際に、別のエラーも修正される可能性があることを念頭において処理する。

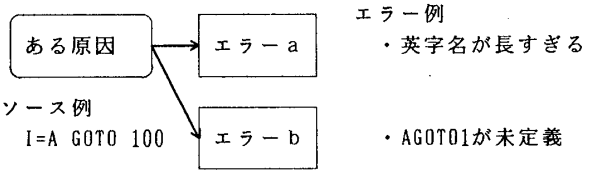

図３　単一原因による複数エラーの例

5．実行時エラーの相談

プログラムのコンパイルが正常に終わり、テスト用のデータの設定が出来れば、一般に利用者は実行テストを行う。この段階で、一回のテストで完全にプログラムが動作することは稀であり、多くの場合、次の様な間違いが生じる。

①　プログラムが全く動かない。
②　プログラムが途中で異常終了する。
③　プログラムが途中でエラーを出し、停止する。
④　プログラムは動くが、意図した通りではない。

このうち①は、ジョブ制御言語の間違い等から実行環境との不整合を生じたもので、現在のところ実行エラーに属さないと考えており、対処していない。また④は、アルゴリズムや仕様の誤りによるもので、計算機にとってはプログラムは正しい。当初述べたように現在は対処していない。従って、我々は実行時のエラーについて、上記の②、③の場合に対して、相談手法の研究を進めている。

この様なエラーを以下のように分類した。

・入出力に関するエラー
・関数の呼出しに関するエラー
・何らかの割り込みによるエラー

これらのエラーについて、人間は次のように対処していると考えられる。

(1)　エラーコードを手掛かりに、ソースプログラム上のエラー発生箇所をみる。
(2)　経験等を通じて、ソースプログラムとエラーコードからエラーの発生原因を見出す。
(3)　よく判らないときは、コンパイラのデバッグオプション等を利用して、再コンパイルしたり、再実行することによって、より詳しい情報を得る。

実行時エラーのコードから、そのエラー原因の候補が幾つか推定できれば、コンパイル時と同様にして、ソースプログラムを解析することで原因を特定できるため、ある程度実行時のエラーが対処できる。

また、初心者は用意されているデバッグオプション（例えば、SUBCHK,ARGCHK)の存在、使用方法、及び結果の見方を十分には知らないと思われるので、これらの使用に対する支援を計算機で行うことも考えている。

6．まとめ

従来からの計算機支援による相談手法と、本手法とが大きく異なる点は、エラーコードと共に、ソースプログラムを解析して、エラーの原因を推測する点である。この手法を用いることで、プログラム相談員が行っているのと同様にソースプログラムから情報を取り込み、エラーの原因を推測することが可能となり、より利用者に使い易い相談システムをつくることが出来たと考えている。この手法の有効性については、すでにコンパイルエラーの対処について、ある程度確認している。実行時エラーの対処についても現在検討中であり、手法の確立とプロトタイプの作成を進めている。

謝辞

京都大学大型計算機センターの長尾真センター長、星野聡研究開発部長、富士通㈱の小坂義裕技師長、三輪修システム本部長付をはじめとする、共同研究関係者各位に感謝致します。

参考文献

〔１〕大西・島崎，「コンパイル時におけるプログラム相談の自動化」電子情報通信学会　技術研究報告 AI88-32 (1988)