文法フォールト除去情報の共有による初級プログラマへのデバッグ支援

3P08

天野直樹†　石澤崇裕†　野中誠†　東基衛†

† 早稲田大学大学院　理工学研究科　経営システム工学専門分野

1　はじめに

今日 GUI やネットワークの発展及びオブジェクト指向言語の浸透等によって開発言語が複雑化し開発者が覚えなければならないプログラム文法・ライブラリに関する情報量が増大している．初級プログラマはそのスキルを向上させていく過程において必要とする範囲の言語知識に関しては当然学習しなくてはならないし，一度学習したとしても言語の仕様変更に対する再学習は必要不可欠であり，この行動は言語仕様の進化が続く限り永遠に続くものである．

2　研究目的

本研究ではコンパイルプロセスを監視し，文法フォールト除去情報を共有する手法を提案する．この情報はどの文法フォールトをどのように修正したかというものであり，これを初級プログラマに提示することで，文法フォールトの除去，及び学習支援を行うことを目的とする．また本研究では情報獲得の負荷を軽減させるため自動獲得可能な情報についてのみ共有する．そのため獲得できる情報がコードベースの修正情報に限定され，新規開発者にとってはその内容が理解できないという可能性を持つ．もし理解できなければ文法フォールト除去及び学習支援も成立しない．そこで本研究で提案する手法が有効であるかを確かめるために，コードベースレベルでの修正情報の参照が初級プログラマのデバッグ支援につながるかについても検証する．

3　コンパイル支援ツール

コンパイル中のプログラマを監視することにより，文法フォールト除去のプロセス情報を獲得する．しかし第 3 者がプログラマの行動を観察することは現実的ではない．多くの労力を要するし，プログラマに心理的な負担をかけるためである．そこで本研究ではプログラマの文法フォールト除去プロセスを監視・分析するツールを開発し，自動的に獲得可能な文法フォールト除去情報を共有する方式を提案する．

文法フォールト除去開始時点に着目することで

① 現在のエラーメッセージ

② 修正前のコード

が自動獲得できる．

また文法フォールトの修正終了時点に着目することで

③ 修正後のエラーメッセージ

④ 修正後のコード

が自動獲得できる．ここで①と③からエラーメッセージの差分を取ることで，どの文法フォールトを修正したかが自動獲得できる．また②と④からその文法フォールトをどのように修正したかがコードベースレベルで自動獲得できる．①～④の情報を文法フォールト除去情報とする．

図 1 にシステム全体図を示す．

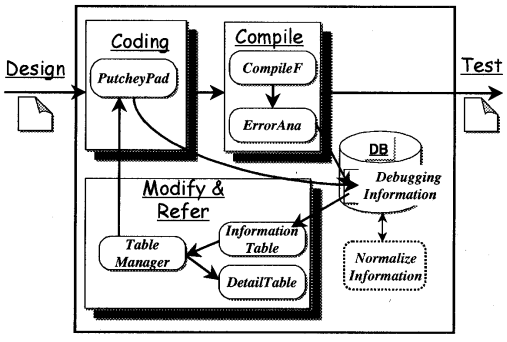

図 1　システム全体図

システムは《CompileF》及び《ErrorAna》によってコンパイルする毎にエラーメッセージを獲得し，《InformationTable》，《TableManager》によって修正前コード，修正後コードを獲得する．図 1 のシステ

Debugging Support to Novice Programmer by Sharing Syntactic Fault Removal Information : Naoki Amano, Takahiro Ishizawa, Makoto Nonaka, Motoei Azuma, Graduate School of Sci.&Eng., Waseda Univ.

ム構成に基づいてプロトタイプを作成した．その画面の一部(文法フォールト除去情報)を図2に示す．

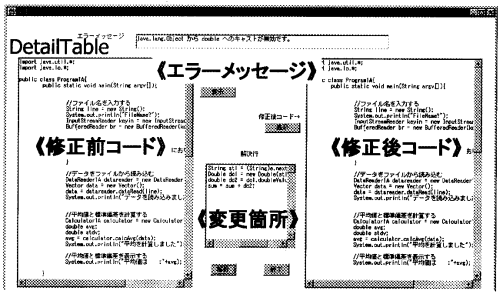

図 2　プロトタイプの画面例

文法フォールトの除去方法が分かっている開発者がコーディング，コンパイル，及び修正を繰り返すことで，DB に文法フォールト除去情報が蓄積されていく．また欠陥を解決できないで停滞中の開発者は文法フォールト除去情報（図 2）を参照することで過去の解決事例を参考にし類推的学習を行うことができる．

4　検証実験

文法フォールト除去情報はコードベースの情報しか含んでいない．これを提示することが，初級プログラマのコンパイル支援及び学習支援に有効であるかどうかを検証する．

4.1　実験方法

初級プログラマを対象に文法フォールト除去実験を行った．被験者は初級プログラマ 10 名であり，本ツールを利用した時とそうでない時とで．同一の文法フォールトに対してどれだけ除去時間が短くなるかについて検証した．デバッグプロセスの特性上，過去に同じ欠陥を除去している場合とそうでない場合とで除去時間が相当異なるはずである．一度除去したことがあるケースでは 2 回目は難なく除去できるはずである．本研究は過去に同様の欠陥を除去したことが無く，デバッグ中に停滞してしまうケースを対象としているため，課題の文法フォールトに対してそれを過去に除去したことがある被験者を除いて実験を行っている．なお課題は PSP[1]からのもので初級プログラマにとって多く停滞が見られたキャストに関する表現を含んでいる．

4.2　実験結果

表 1 に実験結果を示す．表内の数字は文法フォールト除去時間の平均である．

表 1　実験結果

文法フォールト除去時間 (min) ツール有り	文法フォールト除去時間 (min) ツール無し
12(7)	59 注1

表 1に示す通り，ツール有りの方が除去時間が短い．これは本ツールで提供する文法フォールト除去情報が文法フォールト除去支援に有効であったことを示している．しかし本ツールを用いた場合，被験者は解決内容を理解せずに欠陥除去をしてしまう可能性がある．そこで被験者には欠陥除去の後，その除去情報から後戻り的に学習をさせた．その時間が括弧内の数字である．よって学習を含めた時間で考えても本ツールの情報は有効であった考えられる．また実験後のアンケートからでも「本ツール利用によって文法フォールトが取り除きやすかった」，「本ツール利用によって学習しやすかった」という回答が全員に見られた．

5　まとめ

上級プログラマはエラーメッセージを見たときに自分の過去の経験を基に修正案を立てることが可能である．しかし初級プログラマはその経験が浅いため修正案を立てることが困難な場合がある．そこで本ツールの提供する文法フォールト除去情報によってあたかも上級者のように過去の経験を参照し修正することが可能になった．経験を参照して修正案を立てることは上述の通り上級者でも行っていることであり（頭の中でではあるが），初級プログラマにとってメンタルモデル的にも自然であり，上級者に近づけると考えられる．また学習においても文法フォールト除去情報から情報中に内在するルールや構造を抽出する帰納的学習を行うことができ，手掛かり無しに闇雲に学習するのに比べて学習効率も向上すると考えられる．

参考文献

[1]　Watts S. Humphrey，“A Discipline for Software Engineering”, Addison Wesley , 1995

注 1　一部の被験者には時間の制約上解答のヒントを与えているがその被験者にはヒントを与えた時刻でコンパイル終了にしている．よって実際はもっと長い時間を示すはずである．