ミューテーションを用いたプログラミング学習支援システムの研究

橋浦研究室　　大垣　佑樹

１．はじめに

情報工学を学ぶための基礎としてプログラミングの学習は欠かすことができない要素技術である。プログラミング教育は企業でだけでまかなえるものではなく、大学などの各種教育機関で学ぶことが推奨されている。しかし、教員の不足などの諸事情により十分とは言いがたい。

一方で、ソフトウェア開発は規模の巨大化と複雑化の一歩を辿っている。巨大で複雑なソフトウェアには多様な欠陥が混入するため、欠陥除去技術の重要性も増大している。ソフトウェアテスト技術はソフトウェアの欠陥除去技術の代表的な技術のひとつである。その有用性は広く知られており、それを軽視することは、現在のソフトウェア開発だけではなく、将来的なソフトウェア産業にまで影響を与える。

２．研究目的

前述のような状況下であるにもかかわらず、ソフトウェア技術に対する理解不足により、重要性が十分認識されているとは言い難い。著者はこのような問題を解決するためのアプローチとして、ソフトウェアテスト技術を学ぶ前の段階、即ちプログラミングの基礎段階からソフトウェアの欠陥除去を十分トレーニングしておくこと重要であると考えた。

ソフトウェアテストはプログラムに関する理解を深め、プログラミング学習の助けになるものである。よって、本研究ではミューテーションと呼ばれるソフトウェアテストの手法を用いた学習支援システムを開発する。さらに、開発したシステムが前述のソフトウェアの欠陥除去をトレーニングするような問題を実際に作り出せるかどうかについて、確認を行う。

３．研究内容

３．１　ミューテーションについて

ミューテーションはソフトウェアテストの十分さを測るための手法の一種である。1971 年に R. Lipton が提案したプログラムの欠陥の解析方法から生み出された[1]。

ミューテーションの具体的な実施方法は、テスト対象となるあるプログラムの一部を意図的に書き換えた、欠陥入りのプログラムを生成する。この時になるべくプログラマが犯しやすいと同時に、即時に発見することができないような誤りを、1 箇所だけ埋め込む。このようにして作られたプログラムのことをミュータントと呼ぶ。

３．２．テストケースとは

ミュータントから誤りを除去するためには、まずプログラムに対してどのような誤りが埋め込まれているか特定しなければならない。具体的にはブラックボックステスト[2]の考え方を導入し、実際にプログラム起動し、オリジナルプログラムとミュータントの出力が異なるような入力データの組を見つけてやればよい。ソフトウェアテストで用いる入力データの組をテストケースといい、ミュータントを発見することを「ミュータントをキルする」という。

ミューテーションによって埋め込まれた

誤りはシンプルなものなので、誤りを特定さえできれば除去は比較的容易である。よって、誤りを特定することがより重要となる。プログラムをテストする前に、予めこのような出力の差異を検出できるだけのテストケースが用意されているならば、それができないテストケースと比較して、相対的にそのテストケースはミュータントを発見できるだけの強度があると言うことができる。

３．３．ミュータント作成方法

ミュータントはプログラムの一部を意図的に書き換えることによって作成されるが、前節で述べたとおり、コンパイルエラーのような即時にわかるようなものであってはならない。また、生成に高いコストを必要とすることがなく、発生頻度が低すぎるものは向いていない。本研究では間違いを犯しやすい「＋」や「－」などの多くのプログラムに挿入されている演算子に注目し、それを書き換えることにした。

３．４．システムの説明

システムは元となる C 言語のプログラムファイルと出力先の指定を行うことにより、ミューテーションが可能な位置をランダムに決定し結果を出力する。ミューテーションがうまくできない場合には場所を再決定し、書き換えに成功するまで検索を行う。

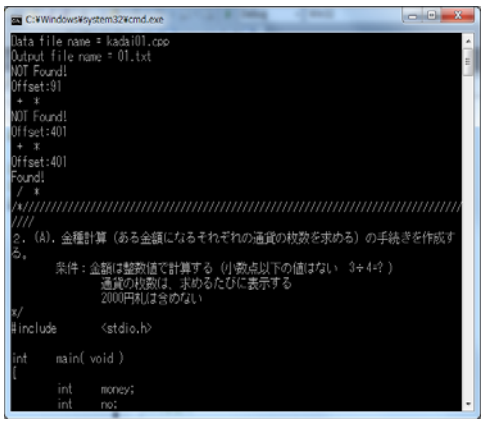

図１　システムの実行例

学習者はミュータントに対して、自力でテストケースを作成し、実際に誤りを除去することで学習を行う。

４．まとめ

４．１．考察

プログラミングを学ぶにはプログラムを理解しなければならない。ミューテーションはその足がかりになれるものであると考えられる。ミュータントをキルすることにより、どうしてここにミュータントが存在するかという理由を考え、それによりプログラマの思考をなぞることができるからだ。プログラマの思考を理解すれば、自然とプログラムについての理解も深まっていくだろう。

４．２．課題と反省点

ミュータントを作成する際のマッチングに甘い部分があり、書きかえてはならない部分や書きかえても効果の無い部分を書きかえてしまうことが多かった。さらにプログラムの中に探索対象の演算子が存在しない場合もあり、書きかえ自体ができないこともあった。今後はマッチング条件の強化と探索演算子の種類を増やすことが課題となる。

参考文献

[1] A. J. Outt, R. H. Untch, “Mutation 2000: Uniting the Orthogonal,” Mutation testing for the new century, Kluwer Academic Publishers, Norwell, MA, 2001.

[2]G. J. Myers, T. Badgett, T. M. Thomas, C. Sandler, (長尾真 監訳, 松尾正信 訳), “ソフトウェア・テストの技法 第 2 版,” 近代科学社, 2006, ISBN 4-7649-0329-6.

[3]玉井哲雄, 三嶋良武, 松田茂広, “ソフトウェアのテスト技法,” 共立出版, 1993, ISBN4-320-02388-9.