プログラム書き換えのための「プログラム修正パターン」記述

松下 誠†

本稿では，プログラムの各行を単位とした再利用可能なコード辺としての「プログラムパターン」について述べる．

Yet Another “Program Modification Pattern” Description for Program Editing

Makoto Matsushita†

This paper discusses yet another “program pattern,” a reusable code fragment that comes from existing software.

1. はじめに

近年のオープンソースソフトウェアの普及の例をあげるまでもなく，インターネットを用いたソフトウェアの開発や，開発されたソフトウェアの配布等が広く行われている．この種のソフトウェア開発では，開発履歴を単一のリポジトリに集約し，それを開発者で共有することによって互いに作業を進めている．このため，ソフトウェア自身とその開発状況をあわせて容易に入手する環境が整ってきている．

こうした背景から，開発プロセスの分析，把握，管理を行うに当たって，ソフトウェア開発履歴が蓄積されたリポジトリを対象とする研究が広く行われつつある．リポジトリには版管理システム，メールアーカイブ，バグ管理システム等があるが，これらリポジトリに含まれているデータに対して，統計的な分析や，リポジトリ内データの相互関係解析，データ工学的解析を行うことによって，開発プロセスの詳細を明らかにできると期待できる．

そこで本稿では，版管理システムを分析対象として，開発者が行った作業の内容を抽象化することによって，ソースコードレベルでのプログラム書き換え内容をパターンとして記述するための方法について考察する．このパターン（以降プログラム修正パターン）を用いることにより，作業者の修正内容を容易に把握することができ，またそのパターンを他のソースコードに適用することによって，ソースコード自体ではなく，ソースコードに対する修正作業の再利用という，より抽象的な再利用を行うことが可能となると考えている．

†大阪大学大学院情報科学研究科

2. 差分情報の抽象化

ここでは，「実際に開発作業として行われた結果」が格納されている，版管理システムのリポジトリに格納されているソースコードを対象とした分析に着目する．

一般に，版管理システムでは開発の各時点において修正された内容を差分として保持している．このときの差分は多くの場合 UNIX diff 形式で表現されている．このため，「昔のソースコードから新しいソースコードを生成する」ためには十分であるが，開発者の修正意図を表現するには不十分であり，またプログラムテキストの字面に依存するために，たとえば変数名が異なる複数の修正内容は異なる内容となってしまう．

そこでまず，差分情報をその修正されたソースコードの字面になるべく依存しない形で抽象化することを考える．このとき，抽象化された内容は次のような特徴をもつことが必要であると考えられる．

- 可能な範囲でソースコード自体の構文情報が保存されていること．ソースコードのブロック構造や制御構造には，その挙動が反映されているため，これらの情報は抽象化された差分情報においても保存されていることが望ましい．
- ソースコード中の識別子が適度に抽象化されていること．そのソースコード中では字面に意味のない変数名や関数名は抽象化されて適当なシンボルに置き換えられ，その修正内容に関係の深い変数や関数名についてはそのまま保存されるなど，修正の前後におけるソースコードの内容に依存した抽象化が行えることが

望ましい.

- 修正に当たって行われる, ソースコードに対する操作が表現されていること. 単に「A と B が C に変化した」という情報だけではなく「A と B を統合して C にした」などのような操作を表現できることが望ましい.

3. 抽象化した差分のパターン化

前節で述べたような, 抽象的な差分情報の表現方法を用いることによって, 開発の各時点における差分情報を逐次抽象的な差分情報へと変換することが可能である. しかし, これらの集合だけでは単純な作業履歴を表現しているに過ぎない.

そこで次に, 抽象化した差分情報から, 「開発の際に行われた修正内容」をパターンとして表現する(プログラム修正パターンと呼ぶ)ことを考える. プログラム修正パターンを導出することにより, ソフトウェア開発を行う際に行われた典型的な作業内容をパターンとして表現することが可能となり, 作業内容の再利用を実現することができる.

抽象的な差分情報は一般的に数多く作成することができるため, パターン化を行う場合には, データ工学等で用いられている分類手法等が利用可能であると考えられる. 抽象的な差分情報の集合をその内容に応じて分類し, 多くの抽象的な差分情報を含む集合や, ある集合に含まれている抽象的な差分情報が導出された時間(作業が行われた時間)に偏りがある場合などを発見することにより, そのような集合を代表する要素をもってパターンとすることが可能であろう.

4. プログラム修正パターンの利用

2,3 節で述べた方法を用いてプログラム修正パターンを定義し, また実際に版管理システムの情報からプログラム修正パターンを導出することにより, 以下のような利用形態が考えられる.

4.1. 作業プロセスの把握

プログラム修正パターンを用いることにより, その修正作業が持つ意味を類推することが可能になると考えられる. たとえば, いくつかのフェーズに分割された修正案件が与えられた際, ある時点でその修正内容がどこまで進められてきたか, といった進捗の把握を行うことができる.

4.2. プログラム修正作業の自動化

プログラム修正パターンは適度な抽象化がなされているため, それが導出されたソースコードとは別のソースコードに対して, プログラム修正パターンの内容, すなわち修正作業を適用することが比較的容易に行えると考えられる. これにより, たとえばある開発修正案件を複数のプログラム修正パターンで表現することができるならば, その修正作業を自動的に行うことが可能となる. これにより, 開発作業を正確にかつ迅速に行うことができる.

また, 典型的なバグ修正作業や, セキュリティに係る問題の修正作業を集めてプログラム修正パターンを導出することにより, 保守作業の軽減や品質の向上に寄与することができると考えられる.

5. 関連研究

5.1. 差分情報の表現形式

ソースコードに対する修正内容を表現するための方法については, 古くから多くの研究が行われてきている. たとえば, ソースコードから構文木を生成した上で, 構文木の差分情報を木構造, あるいは近年においては XML 等の木構造を持ったデータ表現形式を用いる手法がある. また, 差分情報を視覚的に表現するために, 新旧のソースコード中で異なる部分に対し色を用いて表現しするものや, 対応する行を左右に並べることによって挿入と削除を表現するといったものがある.

5.2. 抽象化された修正内容とその利用方法

リファクタリングの研究では, リファクタリングによって行う作業がまず何らかの方法によって与えられており, その作業が適用可能な場所をさまざまな方法を用いて既存のソースコード中から検出し, あるいは実際にその作業を可能ならば自動的に適用することによってリファクタリングを行っている.

このとき, プログラム修正パターンをリファクタリングによって行う作業の表現するために用いることにより, そのリファクタリングが実際の開発作業に即したものとなり, かつ, その修正内容が具体的な形で与えられるために, より現実的なリファクタリング作業を実現することができると考えられる.

6. おわりに

本稿では, 版管理システムに蓄えられた情報を用いることによって, 開発中に行われた作業内容を抽象的に表現したプログラム修正パターンについて, その構築方法や利用形態等について考察した. 今後は抽象化した差分情報の具体的な表現形式や, プログラムパターンの導出アルゴリズムについて検討を行いたい.