

ClassFactory におけるパターン支援機能の提案

5 J-7

市原 潤 杉山 安洋

日本大学大学院工学研究科

1. はじめに

オブジェクト指向方式でソフトウェアを開発する際の難しさの一つとして、ソフトウェアの構造の複雑化があげられる。デザインパターン[1]を使用すると、優れたプログラム構造を再利用することが可能となり、構造の複雑さによるプログラムの負担を軽減することができる。

我々の研究室で開発しているソフトウェア開発支援ツールに ClassFactory[3][4]がある。ClassFactory は 1 つのシステムを構成する複数のソースファイルに宣言されているすべてのクラスを一括して管理、編集することを可能にし、ソースファイルとクラスの間を認識せずソフトウェア開発を行う機能を提供する。この ClassFactory に新たな機能として、デザインパターンを認識させ、その構造を維持しながらソースコードの編集を可能にする機能の追加を考えた。本稿では、ClassFactory へ拡張する機能の概要について述べる。

2. ClassFactory でのパターン支援機能の概要

パターンを使用し、ソフトウェア開発を ClassFactory 上で行う際、プログラマを支援する 3 つの機能について以下に述べる。3 つの機能とは ClassFactory にパターンを認識させる機能、パターンからソースコードを生成する機能、ソースコード編集にもパターンを維持する機能である。

2. 1 パターンを認識する機能

パターンは文章や図などを用いた抽象的な表現によって記述されることが多い。ClassFactory において、パターンを支援するために、そのような特長を持つパターンを認識する機能が必要である。

パターンは通常、いくつかのクラスの集合として表すことができ、Subclass of、Instance of、Member of などのクラス同士の関連や、多重度やアクセス制限などのクラスが持つ固有のパラメータの違いによって、パターンを分類することができる。このことを利用し、ClassFactory がパターンを構成しているクラス同士の関連と、クラスの持つ固有のパラメータを把握することで、パターンを認識することにした。

また、パターンを使用したソフトウェア開発では、1 つのソフトウェア開発で使われるパターンが 1 つであることは少なく、多くの場合は複数のパターンによって構成されている。このようなことも考え、複数のパターンを読み込み、その読み込んだ複数のパターンからソフトウェア開発に必要なパターンを選択できるようにする。

2. 2 パターンを基にしてソースコードを生成する機能

パターンからソースコードを生成するためには、プログラマがパターンの構造を理解し、パターンを構成しているクラスの関係に気を配りながら、ソースコードを記述することが必要である。このようなプログラマの負担を軽減するために ClassFactory へパターンを基にソースコードを生成する機能を追加する。

パターンは2.1節で述べたように、クラス同士の関連やクラスの持つパラメータによって種類を分類できるので、それらをソースコード内で表現することで、それぞれのパターンが持つ特徴を反映したソースコードを生成できると考えた。また、パターンを形作る要素には他にもいくつかあげられるが、パターンを区別し、そのパターン固有の優位性を表現するにはクラスの関連、クラスの持つパラメータの2点で十分であると現時点では考えている。今後、パターンを構成する上で、ソースコードの生成に必要と思われる重要な要素が出現したら、新たにそのパターン内の表現をソースコードへ反映させる。

2.3 ソースコードを編集してもパターンの整合性が保たれることを保証する機能

パターンが適応されたソースコードを編集する場合、パターンの構造を崩す場合が生じる。そこで、ClassFactoryにソースコードを編集時にもパターンの構造が崩されることのないようにする機能を追加する。機能の1つは、ソースコードを編集している最中にパターンの構造を崩す可能性が生じた時には、そのような編集をさせないように、意図的に編集作業を制限する機能である。これによってプログラマが不用意にパターンの構造を崩すことがなくなる。機能の2つ目は、単純な挿入や削除といった編集機能の他に、デザインパターンの構造を保存しながら編集できるコマンドを用意しておき、ユーザがこれらを選んで実行できるようにする。例えば、パターン中のクラス名やインスタンス名を一斉に変更するコマンドがそれにあたる。以上の機能をClassFactoryに追加することで、適応したパターンと編集時のソースコードに表現されているパターンとの整合性を常に保つことが可能になる。

3. 現状と今後の課題

現状は、今回述べた機能を実装している段階である。2.1節で述べたパターンの認識にはプログラマが記述した、デザインパターンが反映されているソースコードを利用することもできるが、オブジェクト指向解析、設計を支援するツールのRose[2]を使用し、作成することを考えている。作成する手順は、Rose上でデザインパターンを記述した後、そのデザインパターンをRoseの持つソースコード自動生成機能を使用することでパターンが反映されたソースコードを作成する。

今後の課題はパターンの変更がされた時の処理が挙げられる。パターンを用いてClassFactory内で編集している時に、パターンを修正する必要が生じるかもしれない。その時、パターンの変更を、どのようにClassFactory内に反映されるかを検討する予定である。

参考文献

- [1]Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 監訳 本位田真一, 吉田和樹, デザインパターン 第2版, SOFTBANK, 1997
- [2]Rational Rose / C++ 3.0 ユーザガイド, 1995
- [3]高倉賢一, 杉山安洋 C++Browserを用いたC++プログラミングの開発, 第38回日本大学工学部学術報告会講演要旨集, pp.567~570, 1995
- [4]市原潤, 杉山安洋 C++Browserにおけるデザインパターンビューアの構想, 第40回日本大学工学部学術報告会講演要旨集, pp.544~547, 1997