R-3

ソフトウェアパターンの再利用を支援するモデリングツール

橋浦研究室　　1145101　赤木謙

1. はじめに

ソフトウェアパターン(以下パターンと呼ぶ)とは，開発過程において頻繁に表れる構造を抽出したものである．パターンを利用することで，再利用性の高い設計を行うことがある．しかし，設計の段階で既存のクラス図にパターンを適用させるためには，ほかのクラスとの相互関係や依存関係を考慮しながら，具体化作業を行わなければならない．具体化作業とは，具体的な形を表す作業のことである．パターンの場合，パターンのモデルに具体的なクラス名を付ける，モデルにメソッドを追加する等の作業が該当する．既存のクラスに変更を加えるとともに，パターンのクラス間やパターンのクラスと既存のクラス間の関係を考慮しなければならないため，開発者がこのような作業を行うことはとても困難である．

2. 研究目的

前述の問題を解決するためには，パターンを支援するモデリング環境が必要となる．このようなツールは，これまでにも様々なものが提案されてきている．しかし，そのほとんどはパターンを呼び出せるだけで，具体化作業は手動で行う必要があった．また，手動でカスタマイズする際に，変更してはいけない部分を変更してしまうなどの操作によって，パターン本来のメカニズムが得られない問題も存在する．

本研究では，前述の問題点を解決するため，既存のクラス図にパターンを適用させるよう再構成を行うモデリング環境を実現する．

3. 提案手法

本研究では，パターンそれぞれ持っている固有のスポットに着目し，パターンを呼び出すときに，既存のクラス図の中でパターンを適用させたい箇所と，パターンの変更可能なスポットを自動で合成を行う．

3. 1. パターンにおけるスポットとは

パターンにはそれぞれに固有なスポットが存在する．パターンを具体化する際に変更しなければならない箇所をパターンのホットスポットと呼ぶ．逆に，パターン本来のメカニズムを壊してしまうため，変更すべきでない箇所をパターンのフローズンスポットと呼ぶ．

3. 2. 合成のルール

パターンのホットスポットの中でパターンを利用するクラスを「クラス A」，既存のクラス図の中でパターンの部分を構成するクラスを「クラス B」とする．本手法ではクラス A をクラス B に置き換え，クラス A の被関連先をクラス B に変更することで，既存のクラス図にパターンを適用させることが可能になる．

一例として Prototype パターン[1]を適用する場合の例を図 1 に示す．Prototype パターンを利用するクラスは `Client` クラスであり，既存のクラス図の中でパターンの部分を構成するクラスは `Meet` クラスである．したがって，`Client` クラスを `Meet` クラスに置き換え，`Client` クラスの被関連先を `Meet` クラスに変更することで，既存のクラス図に Prototype パターンを適用することが可能である．

本研究では合成させるホットスポットを，パターンを利用するクラスとする．

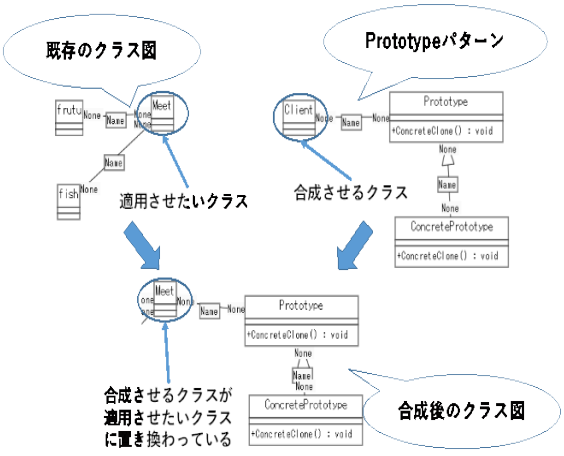

図 1　Prototype パターンの適用例

4. ツールの実装

本手法を実装したツールを実装する．実装形式は Web アプリケーションし， UML エディタは KIfU[2]を使用した．

4. 1. KIfU とは

概念モデリングの編集過程のデータを細粒度に収集し，編集過程を明らかにすることを実現するために開発された．クラスの作成，属性・メソッドの追加，関連の作成など，UML エディタに必要な機能がそろっている．本研究で提案する機能は，KIfU の拡張として実装する．

4. 2. 実装する機能

本研究で実装する機能は以下の通りである．

① エディタ内のクラス図から，クラス名のみを抽出する機能
② パターンとパターンを適用させたいクラスの選択機能
③ 選択されたパターンのホットスポットと，適用させたいクラスの合成を行う機能

5. 実験

本研究の目的である，パターンを適用させるよう再構成を行うモデリング環境が，実現できているかどうか検証する．検証方法は対象とするパターン全てにおいて，本研究で実装した手法が実現できるかどうか実験を行う．適用対象とするパターンは，GoF が考案した 23 個のデザインパターン[1]とする．

6. 実験結果と考察

実験を行った結果を表 1 に示す．

表 1 実験結果

	ホットスポットあり	ホットスポットなし	合計
合成成功	22	1	23
合成失敗	0	0	0

合成には成功したが，本手法が必ずしも有効ではないパターンが存在することが明らかになった．一例として，Façade パターンが該当する．Façade パターンの役割は，機能が多数のクラスによって相互に関連しあい複雑になってきたとき，適切な処理を行うための窓口を提供し，シンプルに利用できるようにすることである．本研究では，既存のクラス図が機能に当たるので，後から Façade パターンのモデルを合成させても，そのモデルの中に機能を記述しなければならない．そのため，既存のクラス図をモデルの中に移動する必要が出てしまう．したがって，余計な手順が増え開発者の支援につながらないため，Façade パターンは本手法において有効ではないと考える．

デザインパターンにおいて本手法が有効ではないパターンは，前述の Façade パターンのほかに，Façade パターンのようにモデルの中に既存のクラス図を組み込まなければならない Mediator パターンや，UML のクラス図では表現することが難しい Singleton パターン，部品や工場の追加が必要となる Abstract Factory パターンの 4 つが考えられる．

一方で，モデルにホットスポットが存在しないパターンは Singleton パターンである．対策として、本研究では利用するクラスである「Client」クラスをモデルに追加することで，合成を可能にした．

7. まとめと今後の課題

本研究では，既存のクラス図に後からパターンを適用させる手法を提案し，UML エディタに提案手法を実装させるよう拡張した．また，23 個のデザインパターン全てにおいて，合成が可能かどうかの検証を行った．

今後は，2 つ以上のホットスポットが存在するパターンにおいて，パターン内の合成箇所を選べるようにする機能の拡張していきたい．また，デザインパターンの中で，有効ではないと考えられるパターンは挙げたが，ほかのパターンにおいて有効なのかどうか明らかにしていきたい．

参考文献

[1] Gamma E, Helm R, Johnson R, and Vissides J, 本位田真一 (訳), “オブジェクト指向における再利用のためのデザインパターン,” ソフトバンク株式会社, 1995-10.

[2] 田中昴文,森一樹,橋浦弘明,櫨山淳雄,古宮誠一, “オブジェクトモデリング演習における学習者にとっての難所の検出手法と支援システムの提案,” 情報処理学会,ソフトウェア工学(SE), Vol.2014-SE-183, No.10, pp.1-8, 2014-3.