振舞・クラス構成・目的を基準とした設計パターンの体系化の提案

奥村 和恵† 金沢 典子† 塚本 享治†

東京工科大学大学院 バイオ・情報メディア研究科 メディアサイエンス専攻†

1. はじめに

ソフトウェアパターンは、よく使われる設計の定石を集めたもので、設計のノウハウを凝集している。しかし同じ動作・役割のパターンが重複して存在したり、アーキテクチャやレイヤーなどで細分化されてパターン全体を俯瞰して把握しにくいなどの問題がある。そこで約70パターンを分析し、振舞・クラス構成・目的について分類し体系化した。そしてパターン適用ツールを作り実験したので、その結果について報告する。

2 設計パターンの問題

2.1. パターンの問題とその解決

パターンには、マイクロ設計用からアーキテクチャに特化したものまで、多くの種類・レイヤーがある。しかしパターン解説本は、GoF などの種類ごとに分断して紹介しており、全体を俯瞰して選べない。用いるパターンを決めた後も、クラスや振舞を設計図へ適用するのに手間が掛かる。

そこで本稿ではパターンを体系化して選びやすくし、設計図へのパターン適用を楽にする方法を提案する。

2.2 アプローチ

まず、GoF や POSA、J2EE パターンなどの約70パターンを分析する。分析対象は目的(効用も含む)・振舞・クラス構成である。そしてパターンを体系化し、設計者が最適なものを選ぶやすくする。次にクラス図へパターンを適用する、図2-1のパターン適用ツールを作成する。

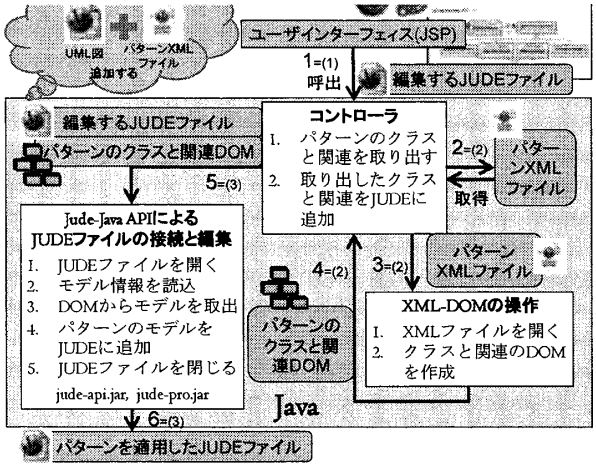

図2-1パターン適用ツールの全体図

A proposal for classifying software patterns
Kazue Okumura†, Noriko Kanazawa†, Michiharu Tsukamoto†
†Tokyo University of Technology, Graduate Course of Bio-Informational Media, Major of Media Science

2.3. 先行研究

[1]など多くの解説本が、パターンをカテゴリやレイヤーごとに分類して紹介している。しかし動的な振舞で分類した例は少ないので、これを中心に体系化する。

[2] [3]では独自の CASE ツール上でパターンを適用するツールを開発した。対して本稿は市販のモデリングツール[4]を利用して、製作するツールの汎用性を高める。開発現場で利用しやすくすることが狙いである。

3. 振舞が中心のパターン体系化

3.1. 振舞の共通構造を重視したパターンの分類

約70パターンを分析すると、レイヤーや種類などに関係なく共通する構造・目的(効用)が見つかった。これを基準に、全パターンを種類で区別せず分類する。

全体の約2/3パターンの振舞が、複数持つメンバーへのアクセスを一手に引き受ける、まとめ役のオブジェクトを持っていた。例えば Business Delegate (J2EE) や Factory Method (GoF)などである。また全体の約1/4パターンに共通して、一時保存用のオブジェクトを持つ構造が見られた。代表例は Transfer Object (J2EE)である。この2つを基準にして振舞を分類した。

3.2. 振舞・目的を統合したパターン体系化

分類結果から、振舞と目的を分類した直交表(図3-1)を作成した。振舞と目的(効用)はばらけて分布せず、ある程度のまとまりがあった。すなわち前節の振舞の基準と目的(効用)には、部分的な関連が見られた。

パターン	目的と効果の対象	複雑なサービスの簡単な入口と内部隠ぺい＊集約される側が多様な場合			ひとつの共通の入口 専門の入口＊集約者を限定する			状態やClientで振舞やデータ
振舞		データ	振舞	両方	データ	振舞	両方	両方
Clientとの間に代表者がいる	仲介と代表者が分離 / 代表は呼び出すだけ		Business Delegate, Service Locator, Service Activator, 能動オブジェクト	Forwarder-Receiver,		Service to Worker, Dispatcher View,		
	仲介と代表者が分離 / 代表者が生成する		Composite View,					
	仲介と代表者が同一あるいは仲介者なし / 代表は呼び出すだけ		Application Service,	Proxy, Facade, Broker, Client Dispatcher Server, 反応体(Reactor)	オブジェクト同期体	Front Controller, Pipes and Filters,		
	仲介と代表者が同一あるいは仲介者なし / 代表者が生成する	Composite Entity, Business Object			マネージャ, ボディガード	View Handler,	Singleton, Data Access Object, Generic Attribute,	Service Adapter,
	不明						Decorator,	
オブジェクトや振舞を代表	状態を外部から知る / 状態を外部							
	振舞を分離・独立し外に / 振舞・オブ					Command, EJB Command,		

図3-1目的と振舞による分類の直交表（一部抜粋）

4. パターン適用ツールの実現

4.1. パターン適用ツールの製作

クラス図にパターンを追加する時、クラスと関連だけでなく、属性や振舞・多重度なども追加する必要があり煩雑である。そこで指定したパターンを図に適用するツール(図 2-1)を作成した。

ツールは次の手順を踏む。まず利用者は画面へ適用したいパターンを指定する。該当するパターンのクラス図が画面へ現れる。次に利用者は、表示されたクラス図に合わせて、適用する Jude ファイルのクラスに識別番号を追加する。そして適用する Jude ファイルを画面上で指定し、実行ボタンを押す。するとツールは該当するパターンの XML を取得し、DOM に読み込む。そして DOM からパターンの情報を読み取り、クラスなどを Jude ファイルへ追加する。

4.2. パターンのクラスを定義する XML 形式

パターンの構成は、UML 図やソースコードなどで表される。ツールがクラス図の情報を扱うために、図 4-1 の様に XML でクラス図を定義した。各パターンを図 4-2 の形式で記述し、ツールが利用できるパターンのライブラリを用意する。

XML はクラスノード<class>と、関連ノード<association>から成る。クラスノードは属性に id とクラス名を、子要素に属性ノード<attribute>と振舞ノード<method>を持つ。関連ノードはその種類を属性に、関連の端クラス<sourceEndClass>を子要素に 2 ヶ持つ。関連端ノードは同じパターン内のクラスノードを参照する。

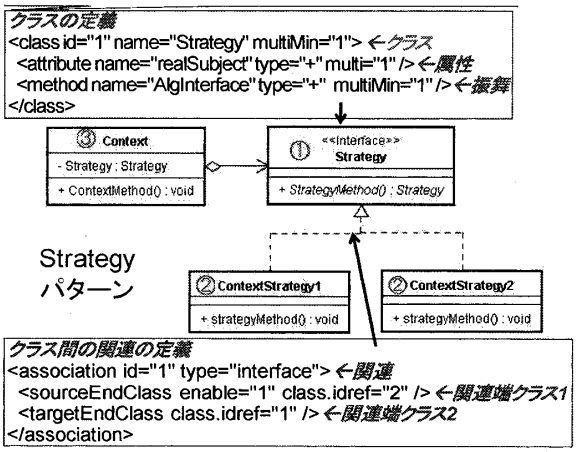

図 4-1 Strategy パターンのクラス図を XML で表す

```
<?xml version="1.0" encoding="Shift_JIS"?> ↓パターン
<pattern id="2002" name="Strategy" pattern.idref="2000">
<classes>←クラスを定義
  <class id="1" name="Strategy" multiMin="1">←クラス
    <attribute name="Strategy" type="+" multi="1" />←属性
    <method name="AlgInterface" type="+" multiMin="1" />←振舞
  </class>・・・・・・
 </classes>
 <associations>←関連を定義
  <association id="1" type="interface">←関連
   <sourceEndClass enable="1" class.idref="2" /> ←関連端クラス 1
   <targetEndClass class.idref="1" />←関連端クラス 2
  </association>・・・・・・
 </associations>
</pattern>
```

図 4-2 Strategy パターンのクラス図を定義した XML

5. パターン適用実験と考察

5.1. 実験

パターン適用ツールを使い、指定されたパターンをクラス図に適用する実験をした。Strategy パターンと適用対象の Jude ファイルを指定し、ツールを実行した。その結果ツールは Jude ファイルへパターンを適用し出力した。適用後のクラス図が図 5-1 である。新しいクラスや操作、関連などが追加されている。

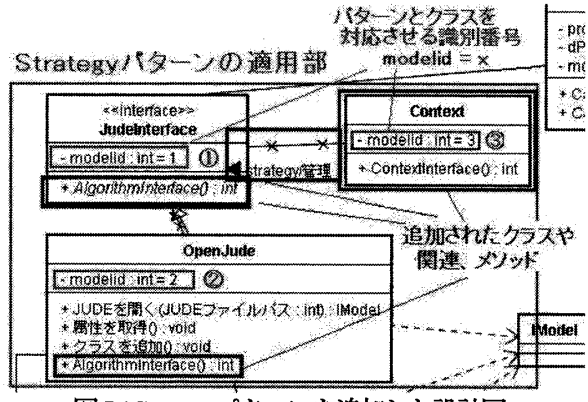

図 5-1 Stragety パターンを追加した設計図

5.2. 考察

ツールはパターンのクラスや関連を、既存の Jude ファイルに追加した。しかし操作において人が担う役割が大きいこと、パターン追加後に人による細かな修正が必要なことが課題として残った。

6. おわりに

約 70 パターンを分析し、種類に関係なく共通する項目から、振舞・目的・構成を分類した。分類結果を総合し、振舞と目的の関連を利用して体系立てる手法を提案した。

次にパターンをクラス図に適用するツールを作成した。実験の結果、Jude ファイルにパターンを追加できたが、人が担う役割も多かった。今後は最適パターンの自動検索や自動体系化などを試みたい。

参考文献

1. Erick Gamma, ほか. デザインパターン改訂版. SoftBank Creative, 1999.
2. 山本純一，松本一教. CASE ツールによるデザインパターン適用支援. 情報処理学会研究報告 SE-111-6 Vol. 1996 No. 84 pp. 41-48, 1996.
3. 永山英嗣，原田実. Design Patetrn Applying Support COPAS by Design Diagram Merging. IEICE TRANS. INF. & SYST. Vol. E83-D No. 6, 2000.
4. (株)チェンジビジョン . Jude-Professional. http://jude.change-vision.com/jude-web/index.html.