デザインパターン利用支援システムのソースコード生成支援

5J－8

瀬川 淳一†　大月 美佳†　吉田 紀彦‡　牧之内顕文†
九州大学大学院システム情報科学研究科†
長崎大学工学部情報システム工学科‡

1. はじめに

近年，オブジェクト指向が提供する再利用の枠組みを効率よく利用できる手法としてデザインパターンが注目されている．しかし現在のところデザインパターンの示す意味的構造を反映した構造的な記述形式と，それに基づくデザインパターン利用のための枠組みが十分でない．

我々はソフトウェア協同開発支援を目指して，オブジェクト指向ソフトウェア部品の統合的かつ分散的なリポジトリの構築を進めている[2]．その一環としてデザインパターンを抽象度の高いソフトウェア部品としてとらえ，我々のシステムに統合しようと研究を進めている[3]．

その初段階として現在デザインパターン利用支援システムを作成している．この支援システムは，デザインパターンの説明文と意味的構造とをSGMLで記述しこれをもとにデザインパターンの検索と閲覧，ソースコード生成支援，および新規のデザインパターンの登録支援を目的としている．

本稿では，本システムにおけるソースコード生成支援部の機能と構成について説明を行う．なお，誤解のないところでは，デザインパターンを単にパターンと呼ぶこともある．

2. デザインパターン

デザインパターンとはオブジェクト指向ソフトウェアに繰り返し現れてくる特徴的な構造や機能を抽出し解析を行い，カタログ化したものである．その利点としては，過去に得ることのできた良質の設計の再利用が容易になる，開発者間の意思疎通に役立つ，等が挙げられる．通常デザインパターンは，自然言語による説明文，構造を示す図，振る舞いを示す擬似コードを用いて表現されている．説明における記述項目としては，名称，別名，目的，動機，適用可能性，構造，構成要素，強調関係，結果，実装，トレードオフ，コード例・使用例，関連パターンがある[1]．

Source Code Generation in Aiding System for Design Patterns
Jun'ichi SEGAWA, Mika OHTSUKI, Norihiko YOSHIDA, Akifumi MAKINOUCHI
Kyushu University, Graduate School of Information Science and Electrical Engineering
Nagasaki University, Department of Computer and Information Sciences

3. SGMLによるデザインパターンの記述

システムが必要とするデザインパターンの情報は，パターンの平文による説明と，パターンの意味的構造であるが，これらは統合的かつ機械的処理が可能な形で記述されなければならない．

よって，我々は意味構造を反映させて文書を構造化できる国際標準規格SGMLを採用することにした．そして，パターンの説明である平文とパターンの意味的構造とを同時に記述できるようにSGMLのインスタンスを設計し，これをPIML（Pattern Information Markup Language）と名付けた．PIML全体の説明は，文献[3]にまかせるとして，ここでは，PIMLによる意味的構造の記述について説明を行う．

意味的構造の記述はタグの階層構造を用いることによりパターンの構造を記述する．その中でもソースコード生成に必要な内容は，関係定義群と役割定義群と複数化グループである．（我々はデザインパターンでのクラスをソフトウェアの中で果たす役割としてとらえ，実際のクラスと区別するために「役割（role）」と呼んでいる）．

関係定義群では，各役割間に成立する継承，参照などの関係を，関係の名称，関係の起点と終点で明示する．

役割定義群では，そのパターンに存在する役割が記述され，属性値として役割の名前や抽象度が入る．そして，役割の内部にはその役割で定義されているメソッド群が記述され，その属性値としてメソッドの返り値，アクセス権などが入る．そして，メソッドの内部にはそのメソッドで必要となる引数群が入り，属性値として引数の型などが記述される．

複数化グループには，ソースコード生成の際に複数化可能なクラスやメソッドを記述する．デザインパターンは，構造上同じ機能をするクラスやメソッドは一つにまとめてあるので，ソースコードを生成するにはこれらを複数化しなければならない．よって，複数化グループの内部には，そういった複数化される要素のタイプ（クラスかメソッドか）と，その識別子（クラス名もしくはクラス名::メソッド名）が入る．

以下にCompositeパターンのPIMLの例を示す．

```
<pattern name="Composite">
 ...
 <structure>
  <relations>  <!-- 関係定義群 -->
   <inheritance origin="Composite" target="Component">
    ...
```

```
  </relations>
  <roles>  <!-- 役割定義群 -->
   <role syslabel="Component" abstract="abstract">
    <operations>
     <operation override="done" access="public"
                return="void" syslabel="Add">
      <args>
        <arg syslabel="target" type="Component">
      </args>
      ...
    </operations>
   </role>
   ...
  </roles>
  <cloneables>  <!-- 複数化グループ -->
   <cloneable>
    <celem type="role" id="Leaf">
   </cloneable>
  </cloneables>
 </structure>
</pattern>
```

4. ソースコード生成支援

ソースコード生成は完全な自動化は行わず，生成過程の各段階で利用者が必要な情報をシステムに与えることによりソースコードを生成する．

4.1 生成の手順

利用者はまず，アプリケーション作成に用いたいデザインパターンを選択する．そして，名前の入力，複製化を行うことにより，アプリケーションに対するデザインパターンの具体化を行う．次に，言語依存情報を入力し，最後にメソッドの内部などの詳細な実装を利用者が行うことによりソースコードが出来上がる．

生成の手順をまとめると以下のようになる．

アプリケーション作成を開始
↓
使用するデザインパターンの選択
↓
具体化に要する情報（名前の入力，構成要素の複数化）の入力
↓
[アプリケーションに対して具体化されたパターン]
↓
言語依存情報（specifier）を入力
↓
[特化されたデザインパターンを組み込んだ応用システム]
↓
メソッドの内部などの詳細な部分の補填
↓
[ソースコード]

4.2 生成系の構成

生成系があつかうデータ構造は，(1)PIML に対応するデータ構造，(2) 具体化されたパターンのデータ構造，(3) ソースコードに対応するデータ構造の３つがある．

(1) は，PIML の構造記述のタグ階層に対応した木構造をしている．具体的には，「構造」の下に「役割」と「関係」が，「役割」の下には「メソッド」がある．

(2) は，ソースコードを生成するために，作成するアプリケーションに具体化されたパターンが入っている．したがって，(1) と同様の木構造をしている．(1) と異なる点は，各要素が対応する (1) の要素への参照を持っている，利用者がつける名前を保持する変数がある，という点である．

(3) も，同様に木構造をとっており，異なる点は，言語依存な情報 (specifier など) が入っている点である．

システムは，まず (1) よりデフォルトの名前が入った (2) を生成する．利用者は，(2) に対し生成系のモジュールを介して，名前付け，複数化を行う．こうしてできた (2) より (3) を生成する際に言語依存な情報を入力する．

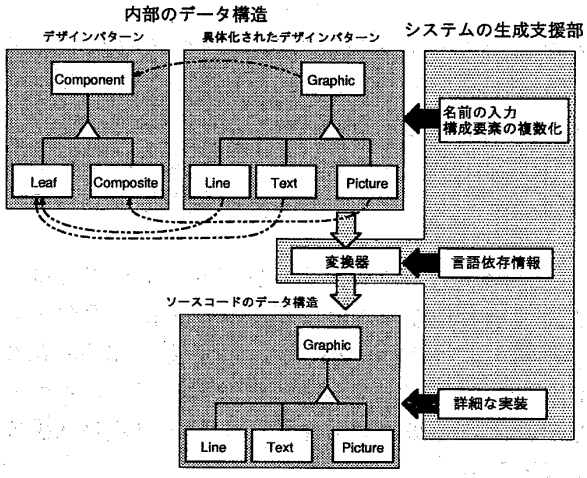

図 1: 生成系の構成

5. おわりに

今回は，デザインパターンからのソースコード生成の手順と生成系の構成についての説明を行った．これにより利用者は，使用するデザインパターンを決定すれば，後はシステムの問い合わせに応じて適切な入力を行うことにより，選択したパターンを反映させたソースコードを手にいれることが可能となる．

今後は，生成過程の中間生成物および結果のソースコードを元のデザインパターとの関係情報も含めて分散カタログ管理する予定である．

参考文献

[1] E.Gamma, R.Helm, R.Johnson, and J.Vlissides, *Design Pattern: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1995); 本位田他訳，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 (1995)

[2] 大月 他，“オブジェクト指向ソフトウェア開発支援のための分散部品リポジトリ”，ソフトウェア工学の基礎 II（ソフトウェア工学の基礎ワークショップ FOSE'95 論文集，大蒔編）近代科学社, pp.207-212(1996)

[3] 大月 他“デザインパターンの SGML に基づく構造化文書化とその閲覧”，情報処理学会論文誌 vol.39 No.3 pp636-645(1998)