ソフトウェア部品リポジトリへのデザインパターン導入の試み

4D－7　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

1. はじめに

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．その一環として現在，オブジェクト指向分析／設計におけるシステム抽象記述であるデザインパターンに注目し，これを部品として我々のシステムに統合化する研究を進めつつある．具体的には，デザインパターンおよびそれから生成されるソースコードの統合的管理，デザインパターンからのソースコード生成の支援，応用システムの特に構造情報・部品関係情報の可視化に焦点を当てている．本稿ではこのデザインパターンの部品化について，その方針と設計を述べる．以下，第2節でシステムの基本設計とこれまでの成果を概観し，第3節ではデザインパターンについて簡単にまとめる．そして，第4節でデザインパターンの部品としての統合化を説明し，第5節でソースコード生成支援について述べる．第6節は関連研究との比較，第7節はまとめである．

2. 分散部品リポジトリの基本構造

広域ネットワークの普及によって一般化しつつある広域分散環境でのソフトウェア協同開発を支援すべく，我々は特にオブジェクト指向ソフトウェアの多種多様な部品の分散統合管理に焦点をあてて研究を進めている．そして，様々な意味や形式の情報を任意に関係づけて保持管理できることからハイパーテキストに着目し，分散ハイパーテキストを利用したオブジェクト指向ソフトウェア部品管理機構を構築している[1]．

このシステムでは，部品はハイパーテキストのノードで，例えば継承，参照，集約などの部品間の関係はリンクで表現される．これまでに，部品管理の基本部分を実現するとともに，まずクラス部品をソースコードや関連情報も含めて管理するシステムを構築し，分散透明な管理を行うサーバ，与えられたソースコードを解析してクラス情報を抽出するツール，自動コンパイル支援ツールなどを作成した．

Integrating Design Patterns into Software Component Repository
Mika Ohtsuki, Norihiko Yoshida, Akifumi Makinouchi
Kyushu University, Graduate School of Information Science and Electrical Engineering

3. デザインパターン

デザインパターン[2, 3]とは，オブジェクト指向ソフトウェアシステムに典型的に現われる特徴的な類型を抽出して解析，記述したものであり，一般には自然言語による説明とシステム構造を示す図から構成される．そして，システム内の各構成要素が担う役割に注目して，その役割間の関連を構造化したものである．具体的には，記述項目として，目的，動機 (なぜ必要なのか)，適用可能性，構造 (図を併用)，構成要素，協調関係，結果 (得られる効果)，実装，トレードオフ，コード例・使用例，関連パターンを含む．

元々デザインパターンは建築設計の分野で提唱された手法であり，近年ソフトウェア工学におけるオブジェクト指向設計に取り入れられて以来，構造の部品としての再利用をクラスより高い抽象度で促進するものとして注目を集めている．なお，本研究でのデザインパターンの記述は文献[3]に従う．

4. デザインパターンの部品化

デザインパターンはそれ自身がソフトウェア部品として，また具体的なクラス部品などとも有機的な関係を持つ抽象的部品として扱われるべきものである．

本システムでは，デザインパターンを構造化された文書としてSGML[4]でパターン単位でカタログ化する．利用者にはこれを，前節で示した記述項目ごとにハイパーテキストのノードとして，そして記述項目間，デザインパターン間，他種の部品との間の関係を相互参照可能なリンクとして提示する．

特に構造記述項目の SGML コードはコード生成の枠組と構造可視化の入力という2つの役割を担うものであり，図1に Composite パターンの例で示すような書式で定義される（説明の簡略化のため，細部は省略）．なお，他の記述項目は自然言語のままで SGML コードに組み込まれる．

5. ソースコード生成支援

応用システムは必要なデザインパターンを組み合わせて構築していくことになる．その際にデザインパターンは，まずその応用システムについて特化 (specialize) され，そしてソースコードに具体化 (instantiate) される．本システムは，このようなデザインパターンからのソースコード生成を支援する．そして，デザインパターンと生成されたソースコードとの間の構造上の関係もリンクで保持管理する．

```
<structure name="Composite">
 <role name="Component">
  <notes> 構成要素の説明 </notes>
  <relations>
    <referred by="Client">
    <aggregated by="Composite">
    <inherited by="Leaf">
    <inherited by="Composite">
  </relations>
  <operations>
   <operation name="Operation">
    <notes> 関数の説明 </notes>
    <relations>
     <arg type="Component">
    </relations>
    <pseudocode> 疑似コード </pseudocode>
   </operation>
   <operation name="Add"> (詳細省略)
   <operation name="Remove"> (詳細省略)
   <operation name="GetChild"> (詳細省略)
  </operations>
 </role>
 (以下省略)
</structure>
```

図 1: デザインパターンの記述例

ソースコード生成は次の手順に従って，システムが利用者と対話しながら進める．

デザインパターン
↓
(特化に要する情報の補填)
↓
応用システムに特化したデザインパターン
↓
(集約の実現も含めたトレードオフの選択)
(既存部品の収集と組込み)
(疑似コードからのコード断片の生成)
↓
ソースコード

特化に要する情報とは，具体的なクラス名や関数名などである．集約については配列・リストなど幾つかの実装がありえるため，デザインパターンに記述されたトレードオフともども，必要に応じて利用者が選択する．また，利用可能な既存の部品があるようであれば，それを組み込む．

システムのソースコード生成部は，擬似コードやトレードオフなどの選択結果に基づいて，コード断片の型枠をシステムまたは利用者が展開，具体化していくことによって応用システムのソースコードを構築していく．

6. 関連研究

デザインパターンのカタログの機械化とソースコード生成の支援を目指したシステムは，IBM T. J. Watson 研究所でも試作されている[5]．しかしこのシステムは，パターンの表わすシステム構造を内部に反映していない (例えば，表示される構造図は画像として埋め込まれている)，生成されたソースコードをパターンと関連付けて管理することができない (プログラム全体は生成されるコード断片から利用者が切り貼りで作成する)，コード生成は単純なマクロ展開の応用であって既存部品の組込みも考慮されていないなど，幾つかの難点を持つ．我々のシステムは基本的な目的は同様であるが，パターンの表わす構造も内部記述に反映させており，生成されるソースコードをパターンと有機的に関連付けて，ともに部品として統合的に管理しようとしている．

ソフトウェアの構造記述はソフトウェア・アーキテクチャ[6, 7]や適応型プログラミング (Adaptive Programming)[8]といった方面でも研究が進められており，そこでも構造情報の管理・ソースコード生成・可視化が重要なテーマとなっている．その成果には参考にすべきところも多いが，それらが個別の応用システムの構造記述であるのに対して，デザインパターンは個別のシステムに特化されない一般的でより抽象度の高い構造記述である点が重要な相違である．

7. おわりに

本稿では可視化にまで触れる紙面的余裕がなかったが，これについては本質的な技術的困難はない．

現在は設計の詳細な検討を行っており，引き続いてシステム試作を進めていく．最大の課題はコード生成のさらなる洗練である．また，将来的にはリバースエンジニアリングの一環として，ソースコードからのデザインパターンの抽出[9]にも取り組んでいきたいと考えている．

参考文献

[1] 大月 美佳, 吉田 紀彦, 牧之内 顕文, “オブジェクト指向ソフトウェア部品のための分散ハイパーテキストに基づくリポジトリ”, オブジェクト指向シンポジウム 95 論文集 (1995)

[2] P. Coad, “Object-Oriented Patterns”, *Comm. of ACM* 35:9 (1992)

[3] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: 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)

[4] ISO 8879 Standard Generalized Markup Language (SGML) (1986)

[5] F. J. Budinsky, M. A. Finnie, J. M. Vlissides and P. S. Yu, “Automatic Code Generation from Design Patterns”, *IBM Systems J.* 35:2 (1996)

[6] Special Issue on Software Architecture, *IEEE Trans. on Soft. Eng.* SE-21:4 (1995)

[7] Special Issue on Software Architecture, *IEEE Software* 12:6 (1995)

[8] Karl J. Lieberherr, *Adaptive Object-Oriented Software*, PWS Publishing (1995)

[9] F. Shull, W. L. Melo and V. R. Basili, *“An Inductive Method for Discovering Design Patterns from Object-Oriented Software Systems”*, *Tech. Rep. CS-TR-3597*, Univ. of Meryland (1996)