2W−1

Java Beanへのロールの割り当てに基づいた相互作用のオブジェクト化による再利用 *

嶺 行伸† 西山 裕之† 溝口 文雄†

東京理科大学 理工学部‡

1 はじめに

オブジェクト指向のアプリケーション開発では, オブジェクトの機能をインターフェイスとして抽象化し, 複数のインターフェイス間の典型的な相互作用であるデザインパターンを適用する事で, 目的の機能を実装するという手法が採られる. デザインパターンは, プログラムの再利用性を高め, 柔軟にし, 全体の把握を容易にする為の設計手法のテンプレートであり, Java の様なオブジェクト指向言語でアプリケーションを開発する為には不可欠の存在である [2].

しかし, 多くの JavaBeans[1] の RAD[2] 環境では, コンポーネント間の相互作用を, 後述する様な極めて単純な相互作用の中から選ぶか, 逆に全てを新しく記述するかの選択肢しか用意されていない. その為, ごく簡単なアプリケーションならマウス操作だけで開発できるが, 少しでも複雑な処理が必要になればプログラムを書かなければならず, RAD 環境の利点の大半が失われてしまう.

本論文では, コンポーネント間の相互作用にデザインパターンを適用する仕組みを, JavaBeans の RAD 環境で実現する方法を提案する.

2 本論文の目的

Java の標準的なコンポーネントアーキテクチャである JavaBeans に基づいて提供される RAD 環境は, Bean と呼ばれるソフトウェアコンポーネントが, イベントによって駆動される相互作用 (接続) によって協調動作する仕組みになっている. イベントが発生すると, イベントアダプターのリスナーメソッドが実行され, 適切な相互作用が引き起こされる. この相互作用のレパートリーは, 開発環境によって任意に用意され得るが, 一般的には次の様なものが用意されている.

- 他のコンポーネントのメソッドの実行
- 他のコンポーネントのプロパティの変更
- スクリプトの実行

だが, 前 2 者は単一メソッドの実行であり, 複雑な処理を実行する事はできない. さらに, 開発環境によっては, 引数の無いメソッドしか実行できないものや, 定数引数しか与えられないものもあり, その場合処理内容はますます制約される. そして後者は, プログラムによって自由に処理内容を記述する事ができるが, 当然の事ながらプログラムを書かなければならず, RAD 環境の利点の大半は失われてしまう. また, この様なスクリプトの再利用性は一般に低く, 似た様なスクリプトを複数記述する無駄も発生しやすい.

本論文では, 以上の様な欠点を克服したコンポーネント間接続モデルを JavaBeans に適用する.

3 Connector とは

Tai は ICSE'98[3] において, ORB[4] ベースのシステム (CORBA 等) で用いるソフトウェアコンポーネント間の相互作用を Connector として抽象化する設計手法を示した [1]. Connector は, コンポーネント間の相互作用に適用するデザインパターンの一種であり, 次に示す要素から構成されている.

ロール ロールとは, 相互作用を実現するにあたって, コンポーネントが果たす役割と責任に名前を付けたものである. コンポーネントは, 1 つ以上のロールを持たなければならない.

ロールインターフェイス ロールインターフェイスとは, あるロールが持つべきサービス (メソッドやプロパティ) のセットをインターフェイスとして記述したものである. コンポーネントの機能は, ロールインターフェイスのサービスを介して利用される. ロールは 1 つ以上のロールインターフェイスを持ち, ロールインターフェイスは 1 つ以上のサービスを持つ.

相互作用プロトコル 相互作用プロトコルとは, Connector によって抽象化される相互作用を,

*Reuse of an interaction by objectization based on assignment of a role to Java Bean

†Yukinobu MINE, Hiroyuki NISHIYAMA, Fumio MIZOGUCHI

‡Dept.of Industrial Admin.Faculty of Sci.and Tech. Science University of Tokyo

[1] Java ベースのソフトウェアコンポーネント技術

[2] Rapid Application Development

[3] International Workshop on Component-Based Software Engineering

[4] Object Request Broker

サービスのシーケンスとして具体的に記述したものである. 記述の方法としては, シーケンスダイアグラム等を用いる事ができる.

これをJavaで実現すると, ロールインターフェイスはJavaのインターフェイス, ロールは複数のロールインターフェイスの多重継承, サービスはメソッド,Connectorは複数のロールへの参照を持つパターンクラス, 相互作用プロトコルはパターンクラスのメソッドとなる.ConnectorをJavaのクラスとして実現する場合,Bean同士を結び付ける役割を果たすので, 他のBeanと共にシリアライズする事ができなければならない.

4 Connectorの実例

例えば, あるBeanから他のBeanへ, 何らかのデータを送るDataTransferというConnectorを実現するとする. この時, データを送る側はDataProducer, 受け取る側はDataConsumerというロールインターフェイスを持っていると考える事ができる. この構造を図1に示す.

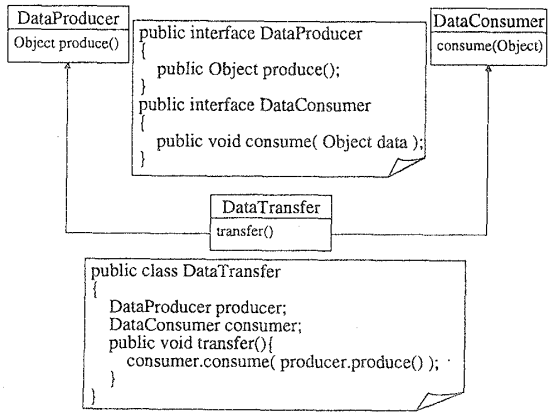

図 1: DataTransferのコード

このDataTransfer Connectorのtransfer()メソッドを実行する事により,DataProducerからDataConsumerにデータが送られる.

5 Beanへのロールの適用

例えば,JavaのTextField[5]をDataProducerとして用いる為には,DataProducerのproduce()メソッドを,TextFieldのgetText()メソッド[6]に変換する為のアダプタークラスを作成すれば良い. 同様に, ネットワーク経由で文字列を出力するSocketOutputコンポーネントも考えられる. これは,DataConsumerの役割を果たす事ができる.

[5]文字列入力コンポーネント
[6]入力文字列を取得するメソッド

従来の方法で, これを全てイベント処理で記述するとすると,TextFieldのテキスト入力イベントを起点として,TextFieldのgetText()の返り値を引数としてSocketOutputの文字列出力メソッドを実行する事になる. 両者の違いを図2に示す.

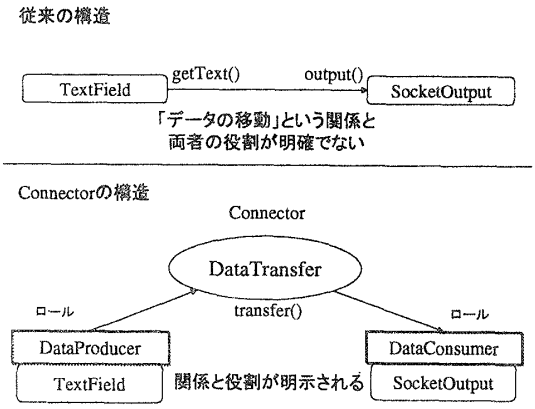

図 2: Connectorと従来の構造の比較

従来の方法では, なぜ両メソッドを実行しているのかが判り難いが,Connectorを導入する事により, データを移動する為である事が明示される.

6 まとめ

DataTransferの様な典型的な相互作用は, 他にも様々な種類が存在する. しかし, 従来のRAD環境は, イベントによる単一のメソッド実行しか考慮していないので, 高度なアプリケーションではその数が膨大となり, コンポーネント間の関係を把握するのが困難になる.

一方,Connectorは次の様な利点を持つ.

- コンポーネント間の関係を明確化してアプリケーション全体の構造の理解を助け, 保守を容易にする
- デザインパターンの様に再利用が可能である
- コンポーネント間の相互作用の実装を隠蔽するので, アプリケーションの設計者はConnectorとコンポーネントの接続だけを行えば良く, 従来より少ない作業量でアプリケーションが構築できる.

参考文献

[1] Stefan Tai, A Connector Model for Object-Oriented Component Integration: *International Workshop on Component-Based Software Engineering*,1998.

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.