O-003

デザインパターンの実行可能UMLによる記述
— 分散システム用パターンによる試行 —

Design Patterns in Executable UML
— Empirical Study on a Distributed System Pattern —

武笠 寛幸† Nurul Azma Zakaria‡ 松本 倫子‡ 吉田 紀彦‡
Hiroyuki Mukasa Nurul Azma Zakaria Noriko Matsumoto Norihiko Yoshida

1. はじめに

デザインパターンとは，ソフトウェア開発時に度々現れる典型的な問題とその解決方法をパターン化し，再利用しやすいようにまとめたカタログである．デザインパターンには，プログラムの再利用によるソフトウェア開発の効率化，コミュニケーションの促進といった利点があり，また，個々のプログラミング言語・アプリケーションから独立しているため，高い汎用性を持つ．

デザインパターンは，問題領域，問題解決のためのシステムの構築方法，利点・欠点などを文章，UML，サンプルコードを使用して記述している．この中で，サンプルコードはデザインパターンの振る舞いを記述するために使用されており，デザインパターンの理解を促進している．一方で，サンプルコードは具体的なプログラミング言語を使用して記述されるために，デザインパターンの記述をプログラミング言語依存化している．サンプルコードの代わりとして，振る舞いを文章で記述すると曖昧になり，また，UMLでは厳密に記述することができない．

そこで，本研究ではデザインパターンの記述にモデル駆動アーキテクチャ，実行可能UMLを適用し，サンプルコードを使用せずに振る舞いを記述することで，デザインパターンのプログラミング言語独立な記述を実現することを目的とする．プログラミング言語独立化するデザインパターンの例として，Webサービスなどの設計効率化も視野に研究を進めてきた分散システム用のWorker Senderパターン [1] を取り上げる．

2. モデル駆動アーキテクチャ

モデル駆動アーキテクチャ（MDA, Model Driven Architecture）は，モデルを中心としてソフトウェアを開発する手法である [2].

MDAでは，まず，特定のプラットフォーム（OS，ミドルウェア，プログラミング言語など）に依存しないモデル，PIM（Platform Independent Model）を作成する．次に，PIMを特定のプラットフォームに特化したモデル，PSM（Platform Specific Model）に変換する．最後に，PSMからコードを生成する．PIMからPSMへの変換，PSMからのコードの生成はMDAツールによって自動的に行うことができる．

本研究では，デザインパターンの記述にMDAを適用する．即ち，デザインパターンのPIMを作成し，プログラミング言語固有のPSMとコードをMDAツールを使用して自動的に生成する．これにより，プログラミング言語独立な形でデザインパターンを記述することができ，また，様々なプログラミング言語用のデザインパターンを自動的に生成することができるようになるため，デザインパターンの汎用性・利用性の向上が期待される．

3. 実行可能UML

実行可能UMLは，UMLに振る舞いを厳密に記述するためのアクション言語を追加したものである．「実行可能」とあるように，モデルコンパイラを使用することで，作成したモデルを実行することができる．

「アクション」は，UML Action Semantics [3] により，意味が厳密に定義されている．Action Semanticsに準拠したアクション言語，及び，そのアクション言語を採用している実行可能UMLを使用することで，MDAにおけるPIM，PSMを記述することができる．

実行可能UMLの特徴として，配列やリストといった実装に依存するデータ構造を取り扱わない．代わりに，関連と多重度を使用して多数のオブジェクトの所持を表現する．配列やリストは，コード生成時に実装情報として付加することで使用することができる．また，クラスはオブジェクトを属性として所持することができない．代わりとして，関連の探索によりオブジェクトへの参照を獲得することができるため，関連を使用してオブジェクトの所持を表現する．

本研究では，実行可能UMLとしてiUMLを使用する．iUMLではAction Semanticsに準拠しているAction Specification Language [4] を採用している．

4. Worker Senderパターンの言語独立化

Worker Senderパターンは，クライアントサーバシステムの構築方法を提供するデザインパターンである．ネットワーク上に存在する複数の計算機に処理を分散させる仕組みを備えており，高性能なサーバを構築することができる [1]．これは，Worker Threadパターンという，マルチスレッドを利用したクライアントサーバシステムにおいてスレッド再利用による高性能化と高応答性をもたらす並行処理デザインパターンを拡張したものである．具体的には，このWorker Threadパターンを分散ネットワーク上の複数計算機の連携にも適用できるようにした並列分散処理デザインパターンであり，Webサービスの複数サーバ上のコンポーネント連携などとも関連が

†NEC 情報システムズ NEC Informatec Systems
‡埼玉大学 Saitama University

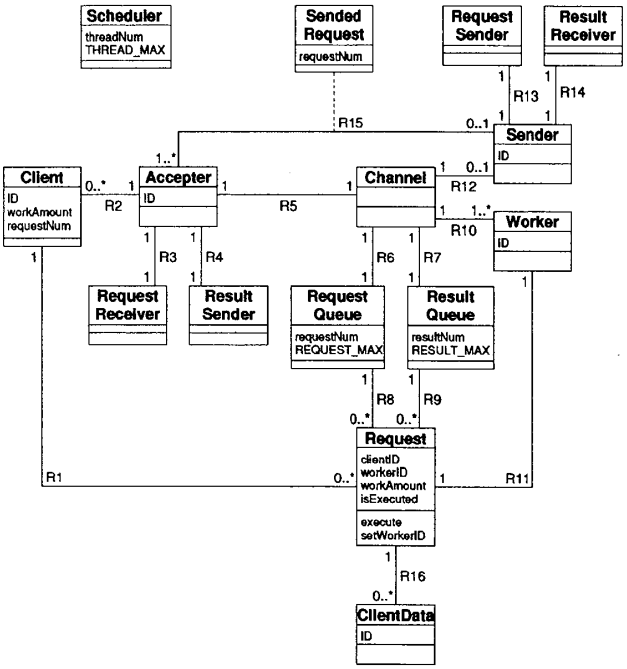

図 1: Worker Sender パターンのクラス図

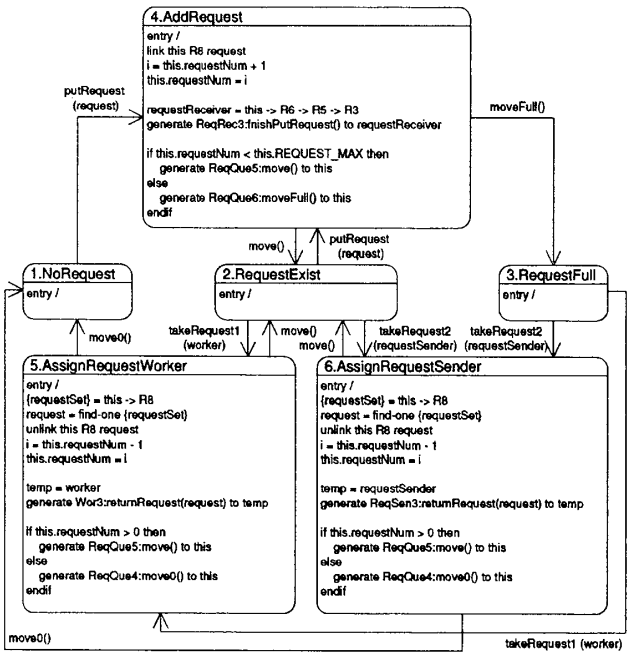

図 2: RequestQueue クラスのステートチャート図

深い．

本研究で特にこの並列分散処理用パターンを適用事例として取り上げた理由は，実行可能 UML について，通常の逐次処理のみならず並行処理，並列分散処理への適用可能性をも探るためであり，特にスケジューリングの記述などが課題となる．なお，Worker Sender パターンでは，元の Worker Thread パターンと同様，これまでサンプルコードの記述に Java を使用しており，並行処理も Java の枠組で記述されていた．

4.1 クラス図

実行可能 UML の制約により，クラスは属性としてオブジェクトを所持できないため，関連と多重度を使用してオブジェクトの所持を表現している．また，iUML のシミュレーションでは Java における Thread の振る舞い，Request の処理時間を上手くシミュレーションできないため，スケジューリングを行う Scheduler クラスを追加している．

4.2 ステートチャート図

作成したステートチャート図の例として，図 3 に RequestQueue クラスのステートチャート図を示す．

RequestQueue は Request オブジェクトを取り扱うデータ構造を表すクラスである．Java 言語では putRequest(Request)，(Request)takeRequest() を synchronized メソッドとして定義することで排他制御を実現していたが，ここでは言語独立な形で排他制御を実現するために，putRequest(Request)，takeRequest() をシグナルとして用意した．

シグナルの規則により，状態におけるアクションの実行が終了するまで他のシグナルは実行されない．また，多数のシグナルを受信している場合であっても，実行されるシグナルは 1 つだけである．そのため，処理の途中で割り込まれることが無く，Request オブジェクトを安全に取り扱うことができる．

5. まとめ

本研究では，デザインパターンの記述に MDA を適用し，デザインパターンの PIM を実行可能 UML で記述することで，プログラミング言語独立なデザインパターンを提案した．また，シミュレーションを行い，モデルの振る舞いを検証した．これにより，Web サービスシステムなどの設計も，より高い抽象度でできるようになり，工程効率化が期待できる．

今後の課題としては，コードジェネレータから得られるコードとサンプルコードを比較し，コードの違いに関する情報をモデルにフィードバックすることで，より良いモデルに洗練していくことが挙げられる．

参考文献

[1] 武笠, 吉田, “Worker Thread デザインパターンの分散化”, 情報処理学会/電子情報通信学会 情報科学技術フォーラム 2005 論文集, Vol.1, pp.119–120, 2005.

[2] S. J. Mellor, M. J. Balcer (二上, 長瀬 監訳), “Executable UML: MDA モデル駆動型アーキテクチャの基礎”, 翔泳社, 2003.

[3] UML Action Semantics, http://www.omg.org/cgi-bin/doc?ptc/02-01-09

[4] I. Wilkie, A. King, M. Clarke, et al., “UML ASL Reference Guide”, http://www.kc.com/download/index.php

[5] 武笠, 実行可能 UML による分散システム用デザインパターンの記述, 埼玉大学修士論文, 2007.