B-012

モデル駆動型アーキテクチャによるUMLエディタの開発

Applying Model Driven Architecture to Developing UML Editors

工藤 智広 Tomohiro Kudo　佐野 建樹 Tateki Sano　柴田 晃宏 Akihiro Shibata　福本 純一 Jun'ichi Fukumoto　徳岡 宏樹 Hiroki Tokuoka　杉山 玄一 Gen-ichi Sugiyama　和田 美江子 Mieko Wada

1．まえがき

モデル駆動型アーキテクチャ(MDA)[1][2]は、ソフト開発の中心をコードからモデルに移し、必要な成果物をモデルから自動生成することで、設計資産の長寿命化・設計と実装の乖離の防止を図るアプローチである。

MDA 実施においては、開発支援ツール(モデルを編集するエディタ、成果物を自動生成するジェネレータ等)の利用が前提となるが、開発支援ツールには以下の二つの性質が望まれる。

- ✓ UML 等、OMG が策定する MDA 関連の標準への準拠。これによりモデルの再利用性・流通性を高めることができる。
- ✓ カスタマイズの容易性。対象領域やプロジェクトの特性を反映することにより、効率的な開発支援が可能となる。

著者のグループでは、上記の性質を満たす MDA 支援方式・環境を研究しており、その一環としてモデル編集用エディタをカスタマイズ容易にするための方式の設計・開発を行っている。具体的には、以下に取り組んでいる。

1. 実際のエディタ開発事例を分析し、共通機能のフレームワーク化・部品化を進めることで、個別エディタ(カスタムエディタ)向けのカスタマイズ作業を効率化する。
2. エディタのカスタマイズに MDA アプローチ自体を適用して、さらに効率化を図る。

このうち 2 については、実際に UML エディタを利用して既存のエディタ(ユースケース図エディタ)に対する仕様の一部をモデル化するとともに、そのモデルからエディタのコードを自動生成するジェネレータのプロトタイプを開発した。

本論文では、この試行についての結果とそれを踏まえて今後の研究課題について述べる。

2．システム構成

2.1 エディタのアーキテクチャ

まず、開発のターゲットとなるエディタのアーキテクチャについて述べる(図 1)。図に示したように、エディタは、オープンソースのツールプラットフォームである Eclipse[3]および Eclipse Tools プロジェクト[4]が提供するプラグイン(EMF・GEF)を利用して構築されている。

MDA 向けエディタフレームワークは、この Eclipse ベースの基盤上に、MDA 支援に必要と思われる以下の機能

- ✓ モデル間の連携機能
- ✓ ジェネレータやチェッカ等の外部コンポーネントをエディタから起動する機能

等を備えたものであり、また、次に述べるエディタ部品を利用するためのランタイムともなっている。

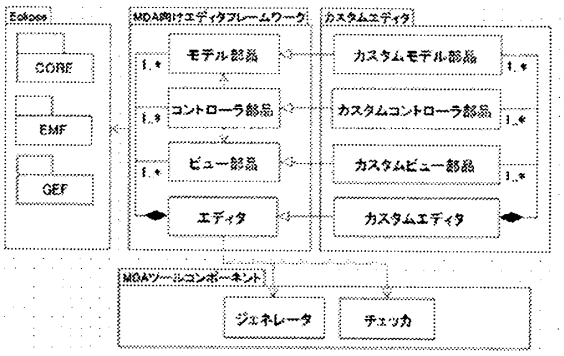

図1 エディタのアーキテクチャ

エディタ部品は、エディタを構成する MVC アーキテクチャにしたがった 3 つの部品：**モデル部品・ビュー部品・コントローラ部品**の総称であり、これらを組み合わせることにより、エディタ上で編集対象となる視覚要素 (図形・線・ラベル等)を実装することができる。

MDA 向けエディタフレームワークでは、UML やその他モデルエディタ間で再利用可能なエディタ部品をあらかじめ用意している。

例えば、UML のコメントに対しては、

- ✓ コメントの表示位置・内容を保持するモデル部品
- ✓ ノート型シンボルを表示するビュー部品
- ✓ 基本的な編集操作を実装したコントローラ部品

が用意されており、個々のエディタ(カスタムエディタ)でこれらを利用可能である。

カスタムエディタを開発する場合には、編集対象とする視覚要素に対応したエディタ部品の３つ組を用意する必要があるが、この場合、MDA 向けエディタフレームワークに用意された共通部品をそのまま利用することも、あるいは共通部品を継承によりカスタマイズして利用することも可能である。

2.2 エディタ開発支援ツールの構成

続いて、MDAアプローチに基づいてエディタのカスタマイズを行うための開発支援ツールの構成について述べる(図 2)。このツール自体も開発ターゲットのエディタ同様、Eclipse上に構築されており、モデルの編集からコード生成・ビルドまでを同一環境で行うことができる。

エディタモデルエディタは、実際にはクラス図エディタ等の UML エディタであり、UML の記法を利用してエディタのモデル(エディタモデル)を記述する。

コードジェネレータは、エディタモデルエディタにより起動され、エディタモデルを入力として、個別のエディタ

日本電気株式会社 システム基盤ソフトウェア開発本部、System Platform Software Development Division, NEC Corporation

部品やエディタ部品の組み合わせを実装するコードを生成する。コードジェネレータにより生成されたコードをビルド・実行することで、最終的にターゲットとなるエディタを得ることができる。

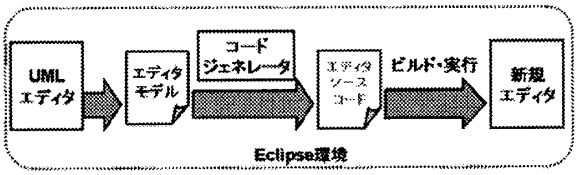

図2 エディタ開発支援ツール

3. エディタモデルとコード生成

次に、本方式で記述するエディタの設計情報について、ユースケース図エディタの例を使って説明する。

エディタモデルで記述する内容は以下の通りである。

- ✓ 個別エディタ部品の定義 (モデル部品・ビュー部品・コントローラ部品)
- ✓ エディタ部品の組み合わせの定義

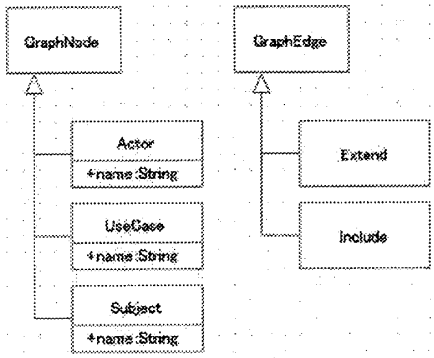

図3 モデル部品の定義

まず、モデル部品の定義の例を図 3 に示す。この例では、追加する 5 つの視覚要素について、共通部品 (GraphNode、GraphEdge) を継承する記述を行っている。

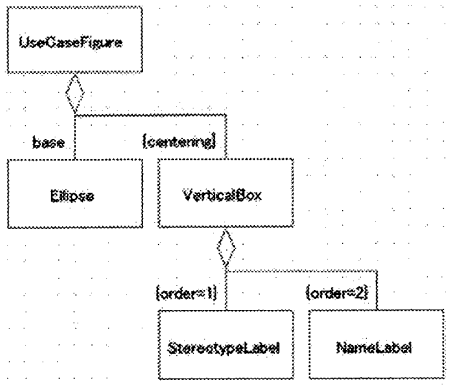

図4 ビュー部品の定義

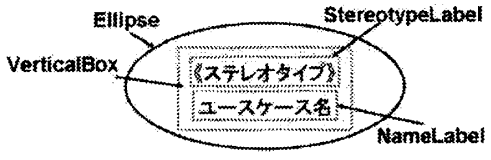

図5 ユースケース図形

続いて、ビュー部品の定義の例を図 4 に示す。この例ではユースケースを表現する図形 UseCaseFigure の構成要素および配置制約を記述しており、楕円形内部の上にステレオタイプのラベルが、下に名前のラベルが配置されるように指定している。この定義に対する実際の図形表示は図 5 のようになる(なお点線は実際には表示されない)。

最後に、コントローラ部品の定義では、コントローラが利用するモデルとビューの役割を持つクラスを定義する。実際にエディタの動作では、モデルとビュー間の対応づけを定義する(例えば、ある属性があるラベルに表示される等)必要があるが、現状のエディタモデルでは対応していない。

コードジェネレータは、以上のエディタモデルからエディタのソースコードを生成する。ただし、コントローラ部分のロジックに関しては、現状では処理記述は手でコーディングする必要がある。

4. 今後の研究課題

今回の MDA 適用では、モデルから出発するのではなく、実装アーキテクチャ(フレームワーク)から出発して、逆算によりモデルを定義するアプローチを取った。このアプローチでは、ターゲットを最初から限定するために、最終的な実行可能性を保証しやすく、ジェネレータ開発も容易である。一方で、ターゲットプラットフォームに依存した前提条件が入り込む可能性があるため、MDA で目指すプラットフォーム独立性については、今後、検証を行っていく必要がある。合わせて、本方式を実際のエディタ開発に適用し実用性を検証していく必要がある。

さらに、現状のエディタモデルでは構造的側面しか記述しないが、モデルとビューとの対応づけやマウス操作等のユーザインタラクションについては、UML の振る舞い図(アクティビティ図やステートマシン図)で記述が可能と思われるので、動的側面への対応が今後の課題である。

5. 関連研究

本論文で述べたグラフィカルなモデル編集用エディタの自動生成を目指したものとして、Eclipse の GMF(Graphical Modeling Framework)プロジェクト[5]がある。

GMF でも、EMF と GEF の利用を前提とし、モデルからエディタコードを生成するなど、本論文の方式と基盤とする技術やアプローチが非常に近い。

ただし、著者の最終目的は、MDA 支援ツール全体のカスタマイズ容易性にあり、エディタの編集機能の自動生成はその一環に過ぎない。したがって、今後は、本方式を拡張して、エディタの編集機能のみならずその他のエディタ機能(モデル間連携等)やジェネレータ・チェッカ等の関連ツールについても、MDA に基づいたカスタマイズ効率化を図る予定である。

参考文献・URL

[1] http://www.omg.org/mda/
[2] Frankel : Model Driven Architecture: Applying Mda to Enterprise Computing, John Wiley & Sons, 2003
[3] http://www.eclipse.org/
[4] http://www.eclipse.org/tools/index.html
[5] http://www.eclipse.org/gmf/