# オブジェクト指向フレームワークの標準的使用法に基づく対話的理解支援

B-17

# A Support Method to Understand the Designers' Intention of Object-Oriented Framework with its Regular Usage

宮城 真弓† 小野 康一‡ 深澤 良彰§
Mayumi Miyagi Kouichi Ono Yoshiaki Fukazawa

## 1. はじめに

ソフトウェア開発におけるプログラム再利用技術のひとつとして、オブジェクト指向フレームワーク (フレームワーク)[1] が注目されている。フレームワークにはアプリケーションの骨組み (フローズンスポット) が提供され、中に含まれる抽象クラスや仮想関数等の抽象的要素 (ホットスポット) を具体化する。これによって、与えられたアプリケーションの要求にあうようにカスタマイズしたソフトウェアを開発することができる。我々は、フレームワーク設計者の意図する具体化に従うことで、このカスタマイズを効果的に行なえると考えている。しかし、アプリケーション開発者にとって、フレームワーク設計者の意図は暗黙的な情報であり、理解することが困難である。本稿では、フレームワーク設計者の意図する標準的な具体化を、アプリケーション開発の進行状況に応じて提示することで、対話的に理解支援する手法を提案する。

## 2. 本研究の目的

本研究では、フレームワークを使用したソフトウェア開発において、フレームワーク設計者の意図をアプリケーション開発者に適確かつ対話的に伝えることを目的とする。フレームワーク設計者の意図を表す具体的な情報として、「クラス/メソッドの組み合わせ」と「クラスまたはメソッドの呼び出し順序」を用いる。これらの情報は、フレームワーク設計者が意図したある特定のフレームワーク使用における標準的な使用方法である。本稿ではこの情報を、「使用標準情報」と呼ぶ。我々は、使用標準情報を与えることによって、アプリケーション開発者がフレームワークの利用にかかる手間を省けることができると考える。

## 3. 本研究の概要

### 3.1 使用標準情報記述言語 FUML

使用標準情報の形式的記述言語として FUML (Framework Usage Markup Language) を定義した。FUML は、フレームワークの構成要素の関係を構造化文書として表現する。FUML 文書はフレームワーク設計者によって作成される構造化文書であり、フレームワークの具体化作業を単位とする並行プロセスモデルの記述である。OCL(Object Constraint Language)[2] などのオブジェクト指向モデリング言語とは異なり、FUML ではフレームワークの構成要素であるクラスやメンバ間の関係を表現できる。図 1 に示す記述例では、DataModel クラスを直接継承したクラス (仮に newModel としている) が存在する時 (記述 A)、そのクラス (newModel) のある具体化作業の際に getContentName と getContentTips の 2 つのメソッドを同時にオーバーライドすること (記述 B) が標準的であることを表現している。

```
<fuml name="MemoPad Framework">
 <usage name="Creation of a new type of memo capability">
  <action>
   <condition>
   <extended id="newModel" ancestor="DataModel" />
   </condition>                                  記述 A
   <override>
    <class id="newModel">
     <method name="getContentName" />
     <method name="getContentTips" />
    </class>
   </override>                                   記述 B
  </action>
 </usage>
</fuml>
```

図 1: FUML の記述例

### 3.2 本手法の前提

フレームワーク設計者
: フレームワークの使用標準情報を整理し、FUML で表現できる。

アプリケーション開発者
: フレームワークを使ったアプリケーション開発における一般的な知識、対象フレームワークのアプリケーションドメインの知識を持つ。

対象フレームワーク
: 特定のアプリケーションドメインに特化されている。

### 3.3 本手法の入出力

フレームワーク設計者から提供される入力情報

- フレームワークのクラス階層とメンバのシグニチャ
- 使用標準情報

アプリケーション開発者から提供される入力情報

- その時点までに書き終えたソースコード
- その時点で手がけているクラス/メソッドの情報
- システムに対する応答

システムからの出力情報
: アプリケーション開発者の進行状況に対応して、推奨される作業情報 (推奨作業)

†早稲田大学大学院 理工学研究科
‡日本アイ・ビー・エム株式会社 東京基礎研究所
§早稲田大学 理工学部

## 4. 本手法の実行例

本節では、本システムによって対話的に支援が行なわれる例を示す。適用するフレームワークは、簡易な情報を記録するためのメモを作成するアプリケーションのためのホワイトボックスフレームワーク (メモパッドフレームワーク) である。メモパッドフレームワークにおいて、ホットスポットとなる部分は機能拡張部分である。本システムでは、具体化の作業中に作業を支援する推奨作業を対話的に提示する。この提示された推奨作業を実際に具体化するかどうかを選択するのは、アプリケーション開発者である。以下に、アプリケーション開発者が推奨作業を受諾した場合と拒否した場合のそれぞれについての例示を行なう。

### 4.1 推奨作業の受諾例

例えばアプリケーション開発者が、メモの本体部分である抽象クラス DataModel のサブクラス TextModel を継承して新しいクラス RichTextModel の作成に取り掛かったとする (図 2)。このとき、図 1 に示した使用標準情報の前提条件が満たされるので、システムは「getContentName,getContentTips メソッドをオーバーライドする」ことを推奨作業として提示する。アプリケーション開発者は、提示された推奨作業に従って具体化を行なう。

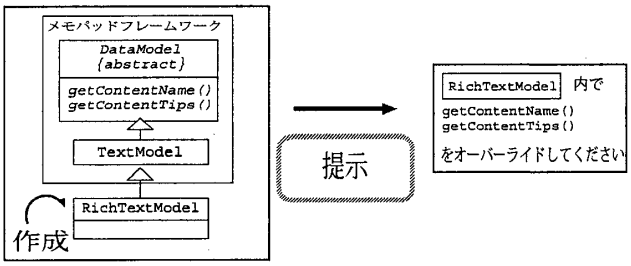


図 2: RichTextModel の具体化

### 4.2 推奨作業の拒否例

アプリケーション開発者は、システムに前述の作業の終了を伝える。フレームワーク設計者は、「DataModel を継承したクラスを作成した場合 (記述 C) に、新しいダイアログを作成すること (記述 D)」を意図している (図 3)。そのため、システムから RichTextModel と対応するダイアログクラスとして TextDialog を継承したクラスの作成が、アプリケーション開発者に推奨される。

```
<action>
 <select>
  <when>
   <condition>
    <extended id="newModel" ancestor="TextModel" />
   </condition>                                  記述 C
   <newclass id="newDialog" ancestor="TextDialog" />
  </when>                                        記述 D
 </select>
</action>
```

図 3: ダイアログの具体化に関する FUML の記述部分

ここで、アプリケーション開発者は既存のダイアログを使用したいと判断し、システムに「拒否」と応答したとする。推奨作業が行なわれなかったので、システムは続く作業を見直す。新しいダイアログが作成されないので、フレームワーク設計者の意図である「新しいダイアログの登録」(図 4) の前提条件 (記述 F) が満たされず、この推奨作業 (記述 G) が提示されない。このように、アプリケーション開発者と対話することで作業の進捗やアプリケーション開発者の意向に応じた支援を行なうことができる。

```
<action>
 <conditon>
  <extended id="newDialog" ancestor="AbstractDialog" />
 </condtion>                                    記述 F
 <choose>
  <case>
   <describe>
    <class id="newModel">
     <method name="createDialog">
      <params>
       <param class="javax.swing.JFrame" id="owner" />
      </params>
     </method>
    </class>
    <![CDATA[return new $newDialog($owner);]]>
   </describe>
  </case>
 </choose>                                      記述 G
</action>
```

図 4: ダイアログの登録に関する FUML の記述部分

## 5. 関連研究

既存のフレームワークの理解を支援する手法として、ブラックボックスフレームワークを対象とするアクティブクックブック [3] をあげる。アクティブクックブックは、クックブックにアプリケーション開発者の行なう実装クラス名を入力する事で、実装で用いる「一連のクラス間の関係」を具体例を用いて視覚的に提示する。アクティブクックブックの手法を用いる事で、本手法では補えなかった「一連のクラス間の関係」を取り入れることができる。反対に、本手法を用いる事で一連のクラス間の関係だけでなく、「クラス/メソッドの関係」や「クラスまたはメソッドの呼び出し順序」を取り入れることができる。

## 6. おわりに

フレームワークを用いたアプリケーションの開発では、フレームワーク設計者の意図の理解が必要である。そのために、フレームワークの使用標準情報を導入する事で、開発作業を支援する方法を示した。また、フレームワークの標準的な使用法から逸脱する場合、アプリケーション開発者から対話的に情報獲得することで、適切なガイドを再作成できることも示した。

## 参考文献

[1] R. E. Johnson, 中村宏明, 中山裕子, 吉田和樹, パターンとフレームワーク：オブジェクト指向ソフトウェア開発技術, 共立出版, 1999.

[2] IBM Corporation, *Object Constraint Language*, http://www.ibm.com/software/ad/library/standards/ocl.html, 2002.

[3] A. Schappert, P. Sommerlad and W. Pree, "Automated Support for Software Development with Frameworks," *Proc. of SSR'95*, ACM Press, pp. 123-127, 1995.