UML図面とJavaソースコード間の対応関係の自動生成法に関する研究

渡部 菜月 (310126)

北陸先端科学技術大学院大学 情報科学研究科

2005年2月10日

キーワード: ソフトウェア共同開発, UML, Java, デザインパターン.

1 本研究の背景と目的

ソフトウェア共同開発における変更・修正作業において、複雑な依存関係を持つ文書やソースコードを漏れなく適切に変更する作業は容易ではない。ここで、依存関係とは「BがAに依存する」とき、「Aを変更した場合、Bを変更する必要がある」という関係である。

UML図面とJavaソースコードを対象として、安全で効率的な作業変更を支援するための情報モデルの開発が進められている。具体的には、UML図面およびJavaソースコードの構成要素型を依存関係で結合したメタモデルを定義し、それを利用することで、UML図やJavaソースコードの実例に自動的に依存関係を付加する。これにより、変更支援のための情報モデルを生成し、依存関係の検出作業の自動化を図ることができる。

これを実現するためには、UML図面とJavaソースコード間の一致部分を示す、対応関係の定義が不可欠である。しかし、設計時に作成されるUML図面と、その実装であるJavaソースコードでは、一般に構造が一致していないため、両者の対応関係を定義することは容易ではない。

本研究では、このような構造の不一致の問題点を解消し、両者の対応関係を生成する手法の提案を行う。また、既存のUML図面とJavaソースコードを用いて、両者の対応関係を自動的に生成するシステムの構築を行う。UML図面は、クラス図およびコラボレーション図を対象とする。

2 問題点の定義

本研究の目的であるUML図面-Javaソースコード間の対応関係の定義を行うには、ソフトウェアの設計時構造と実装時構造の差異を考慮する必要がある。一般的に、UML図

面に存在する 1 つのモデル要素をソースコードとして実装した場合、必ずしも単一のクラスとならず、1 つ以上のクラスの集合によりモデル要素の機能を実現している。

このような差異が発生する理由としては、次のようなものが挙げられる。

- 設計者（個人、組織）のルールの適用
- デザインパターンの使用
- 拡張性・再利用性など、オブジェクト指向プログラミングにおけるメリットの確保

そのため、モデル要素の名前や属性に一致するクラスを抽出するだけでは、一般的な対応関係の生成手法としては不適である。本研究では、両者の構造の不一致をソースコード内におけるクラスの分割と定義し、ソースコード内においてモデル要素に相当する機能を持つクラス群を自動的に抽出し、呈示する手法を提案する。

3 対応関係の生成過程

本研究では、UML 図面とソースコードの対応関係の生成過程を次の 2 段階のフェーズに分割する。

1. ソースコード内の関連クラス群の抽出

   まず、UML 図面の情報を与えられていない状態で、ソースコードから全ての関連クラス群を抽出し、データベースに蓄積する。ここで、関連クラス群とは、UML 図面における 1 つのモデル要素を構成する可能性のある最大粒度のクラス集合を示す。

2. モデル要素の指定による関連クラス群問い合わせ

   UML 図面のモデル要素を指定することで、上記で提案した手法により蓄積された関連クラス群の集合から、対応関係を持つものを問い合わせる。自動生成システムのプロトタイプを設計・実装することで、これを実現する。

4 関連クラス群の抽出

ソースコードから関連クラス群を抽出するためには、抽出根拠となる基準を与えて探索する必要がある。まず、ソースコード内のクラス間に存在する特定の関係を解析する。さらに、各クラス間関係に対し、追跡する範囲を限定する条件として、追跡範囲を付加する。本研究においては、次のクラス間関係を解析する。

- 継承・実装

  ソースコード内において、継承、実装の関係で結合されたクラスまたはインタフェースは、UML 図面上においては同一のクラスを構成している可能性が高い。ユーザ定

義クラス以外のクラスを除いたこのようなクラスまたはインタフェースの集合を要素数 1 の場合も含め本研究では継承グループと定義する。継承グループに対し、追跡範囲を定義することにより、モデル要素の機能と関連がないクラスまたはインタフェースを排除し、関連クラス群のみを抽出する。追跡範囲を付加し、既存のソースコードに適用したところ、継承グループから関連クラス群のみを高い精度で抽出することができた。

- 参照関係

  ソースコード内において参照関係で結合された 2 つの継承グループは、これらの協調により、モデル要素の機能を構成していることがある。例えば、State パターンを用いて、一方の継承グループの持つ状態を他方の継承グループが表現している場合などが挙げられる。参照関係に、両端の継承グループの生存期間が一致するための条件、および機能としての独立性を保証する条件を加えソースコードの実例に適用した。その結果、継承・実装のみを解析した場合に比べ、関連クラス群に含まれるべき継承グループをさらに高精度に抽出することができた。

5 まとめと今後の課題

本研究では、ソフトウェア共同開発における作業変更を支援する情報モデルの開発には、UML 図面と Java ソースコード間の対応関係の定義と自動生成が不可欠であると認識し、これを達成するために解決されるべき困難性の定義を行った。そして、この困難性を解決し、両者の対応関係を生成する手法の提案を行った。提案手法に対し、UML 図面と Java ソースコードの実例に適用したところ、特定の条件下において、高い精度で関連クラス群を抽出できた。本研究における今後の課題を以下に挙げる。

- クラス間関係の追跡範囲の詳細化： UML 図面のモデル要素の実装方法は自由であり、関連クラス群がモデル要素に必ず合致することを保証できない。そのため、本研究で定義した参照関係の追跡範囲では、機能に無関係な継承グループを関連クラス群に含む可能性が残されている。関連クラス群として含んではならない継承グループを排除できず、精度を低下させる場合があることは重要な課題である。

- より詳細な対応部分の抽出： 本研究では、モデル要素に対応するソースコード内の箇所をクラス単位の粒度で抽出した。モデル要素によっては、さらに範囲を狭め、フィールドレベルやメソッドレベルで抽出する必要がある。

- 抽出条件の一般化： 本研究で用いたソースコードにおいて本手法は一定の成果を得たが、一般的なソースコードに適用可能であることは証明されていない。そのため、本研究で用いたソースコードに出現するデザインパターンのメタパターンに適用可能な条件に本手法を昇華させ、一般性を証明する必要がある。