1L-06

API 利用に関するパターンマイニング手法について

宮里　章太[†1]　　岸　知二[†1]

[†1] 早稲田大学　創造理工学研究科　経営システム工学専攻

1. 研究背景と目的

近年、クラスライブラリが一層拡充されているが、それらの活用のためには、API(Application Programming Interface)の使い方を把握する必要がある。API とは、ソフトウェアの機能を外部から呼び出すための手続きを定めたものであり、クラスライブラリを使用する際には、その API を用いる必要がある。一般的に、API の仕様書等には基本的な機能やインタフェースは記述されているが、実際にどのような設計構造の中で使うべきかどうかは示されていない。設計構造を把握することにより、そのクラスライブラリがどのように用いられて問題の解決に使われたのかというシステムの設計意図を理解することができ、API を利用する際に有効である。

設計構造を抽象化し、設計ノウハウとしてカタログ化したものとして、デザインパターンがある。最も代表的な、Gang of Four[1]が提唱しているデザインパターンを例にとると、デザインパターンの一つであるコンポジットパターンは再帰的な構造を考える時、より拡張、修正しやすいようにする設計ノウハウである。このようなデザインパターンは、設計意図を明示的にすることができ、再利用しやすいソフトウェア開発を可能にする。そこで、既存ソフトウェアから API を利用する際の設計構造を抽出し、デザインパターンの様な抽象的な設計構造として提示することで利用者の支援ができると考えた。なお、設計構造には、静的構造と動的構造が考えられるが、本研究では、システムを構成するクラスとそのクラス間の関係を表す静的構造にのみ注目する。

上記のために、本研究では、特定の API を含む複数の既存ソフトウェアから、抽象設計構造のパターンマイニングを行う手法を提案する。

A Pattern Mining Method for API Usage
†1 Department of Industrial and Management Systems Engineering CSE Graduate School , Waseda University

2. 提案手法

2.1. クラスライブラリの抽象構文木化

本研究では、複数の既存ソフトウェアのソースコードを対象とするが、API によって利用されるクラスライブラリ自体はソースコードとして記述されていない。よって、特定 API を含む設計構造を抽出するためには、クラスライブラリも設計構造に含む必要があるため、クラスライブラリを一つのクラスとして置き換え、ソースコードを変換した抽象構文木に加える。

2.2. API の抽出

API の抽出は、それぞれの既存ソフトウェアの抽象構文木から、同じクラスライブラリを用いた API を特定することで行う。

2.3. 設計構造の抽出

対象のソフトウェアが大規模な場合、特定 API を含む設計構造が大きすぎるため、適切なパターンマイニングができない可能性がある。そこで、それぞれの既存ソフトウェアを、クラス通しのメソッドの関係数を考慮した無効グラフとして表記し、それを、スペクトルグラフ分割を用いた[2]のクラスタリング手法を用いることで、特定 API を含む必要最低限のクラスタを判別し、そのクラスタを、特定 API を含む設計構造とする。図 2 は無効グラフから特定 API を含む設計構造を抽出した図である。

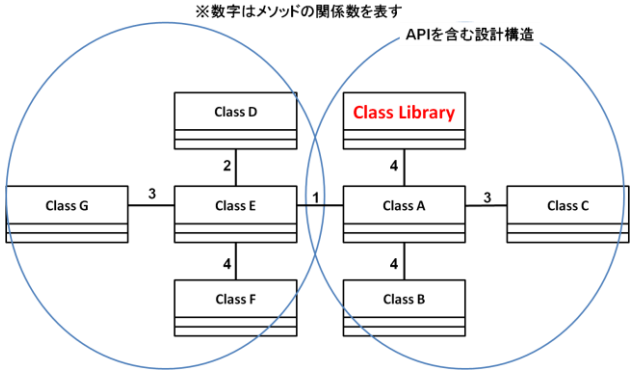

図 2　特定 API を含む設計構造の抽出

2.4. パターンマイニング

パターンマイニングを行うにあたって、それぞれの既存ソフトウェアの特定 API を含む設計構造を行列化する。そのソースコードの設計構造が図 3 のクラス図で表された場合、それを図 4 のように行列化して表記する[2]。行列化は図のように関連、継承などの関係ごとに別々にグラフ化し、それらを行列として表記することができる。

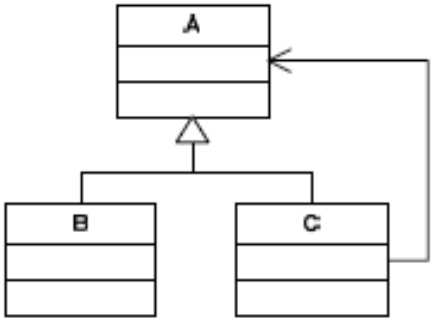

図 3 サンプルクラス図[2]

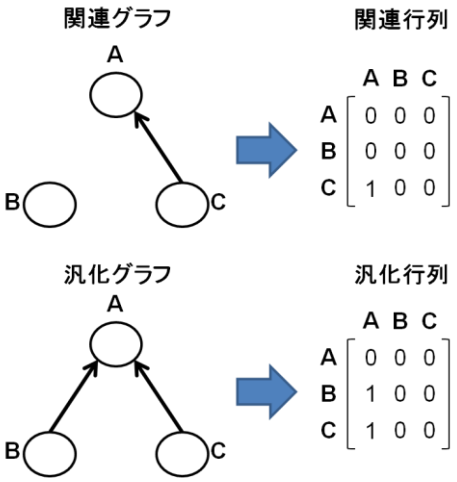

図 4 サンプルの行列表記[2]

マイニングはそれぞれの既存ソフトウェアの特定 API を含む設計構造との行列の類似度を導出することにより行う。

①それぞれの既存ソフトウェアにおける特定 API を含む設計構造の集合を S、ソフトウェア数を n とする。

②S をルールに従い行列化し、それぞれのソフトウェアごとの行列を S・matrix とする。k が 1 から n の間③、④を繰り返す。

③S_k・matrix を用いて、[3]で提案するアルゴリズムにより、それぞれの S_1 から S_n までの S・matrix に対して、類似度を計測する。これを S_k・similarity とする。

④S_k・similarity が、汎化、関連などの関係の数を x とした場合、すべてのクラスにおいて、x-1/x より高ければ、その設計構造をパターンとして抽出する。

⑤手順を終了する。

図 5 に本研究の全体像を示す。まず、既存ソフトウェアのソースコードを抽象構文木に変換し、クラスライブラリを一つのクラスとして加える。次に、抽象構文木から、同じクラスライブラリを用いた API を抽出する。次に、それぞれのソースコードのクラス通しのメソッド関係数を導出し、クラスタリングによって、特定 API を含む設計構造をソフトウェアの数分抽出する。最後にそれぞれの設計構造に対して、提案しているパターンマイニングを行うことで、類似度から抽象設計構造を検出する。

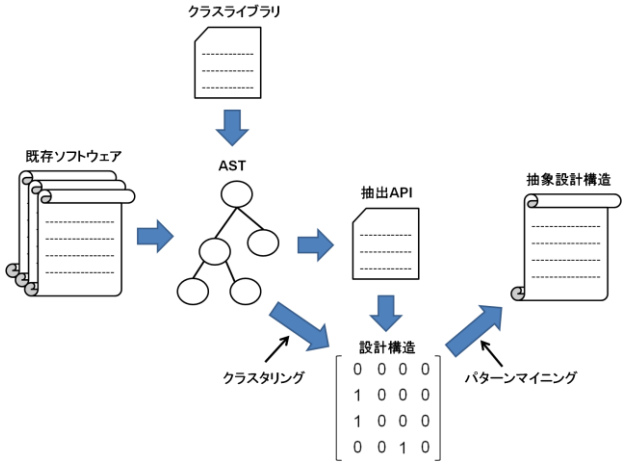

図 5 本研究の全体像

3. おわりに

本研究では、特定の API を用いている既存ソフトウェアにおけるパターンマイニング手法を提案した。

今後はマイニング手法の妥当性を評価するとともに、実際の API 利用支援の有効性について検討したい。

参考文献

[1] E.Gamma , R.Helm , R.Johnson , J.Vlissides , オブジェクト指向における再利用のためのデザインパターン , 改定版 , 本位田真一 , 吉田和樹 , 監訳 , ソフトバンクパブリッシング , 1999

[2] Alexander Chatzigeorgiou , Nikolaos Tsantails , George Stephanides , Application of Graph Theory to OO Software Eingineering , pp.29-36 , 2nd International Workshop on Interdisciplinary Software Engineering Research , 2006