スプリングアルゴリズムに基づく ソフトウェア部品グラフの視覚化手法の実現

2012SE251 竹仲孝盛
指導教員：横森励士

1 はじめに

大規模化したソフトウェアを理解するためには，ソフトウェアの構成を効果的に理解出来る仕組みが必要である．本研究では，ソフトウェアを表現するソフトウェア部品から構成されるグラフをスプリングアルゴリズムを用いて視覚化する方法を提案する．提案手法を用いてソフトウェアの全体像を効果的に理解できること，および変更時のソフトウェアの構造の変化も確認しやすくなることを示す．

2 背景技術

2.1 ソフトウェア部品

一般に，ソフトウェア部品（以下，部品）とはモジュールや関数，クラスなどのソフトウェアの構成要素を指す．ソフトウェアは構成要素の部品間で相互に属性や振る舞いを利用しあうことで一つの機能を提供する．本研究では，ある部品がある部品を利用する時，部品間に利用関係が存在すると考え，部品グラフとしてモデル化する．

ソフトウェア全体を構成する機能ごとのまとまりをモジュールと呼ぶ．1 つのソフトウェアを意味のあるまとまりとして複数のモジュールに分割し，モジュール間の関係を適切に整理することで，より開発者が理解しやすいソフトウェアであるとみなされる．その際には，単一で固有の機能を実行できることを表す凝集度と，他のパッケージと利用関係が少ないことを表す結合度をもとにそれぞれのモジュールの独立性が評価される場合が多い．

2.2 スプリングアルゴリズムによる視覚化手法

多数の頂点をもつグラフを，コンピュータ上で視覚的な面から効果的に自動配置するアルゴリズムとして，様々なアルゴリズムが提案されている．本研究では，次のようなモデル上で物理シミュレーションさせることで各頂点の配置を行う，スプリングアルゴリズム [1] に着目する．

- グラフの頂点を，質量を持つ体積 0 の小物質で，クーロンの法則を伴う電荷とみなし，互いに反発させる．
- グラフの各辺を，フックの法則を伴うバネとみなし，辺が伸び縮みできるようにする．

スプリングアルゴリズムをソフトウェアの分析に利用した事例として，花川の研究 [2] では，モジュール間の結合関係の高いモジュール群の集合やモジュール間の論理関係の高いモジュール群の集合を求めるために，スプリングアルゴリズムを用いてモジュールのマッピングを行っていた．

3 スプリングアルゴリズムを用いた視覚化手法

3.1 手法の概要

本研究では，ソフトウェア部品グラフを視覚的に表現するためにスプリングアルゴリズムを用いて部品グラフを表示する機能を実現する本方法が視覚化手法として有効かを検証し，クラス間の関係を確認しながらソフトウェアの部品構造を見直す手法を提案する．

3.2 視覚化ツールの実現

スプリングアルゴリズムを用いて，部品グラフを視覚化するツール Jasptool を作成した．Jasptool は Java プログラムを対象とし，Classycle[*1] を用いて，利用関係を抽出したのちに部品グラフを作成し，作成した部品グラフをスプリングアルゴリズムに基づいて表示する．また，ソースコードを正規表現により検索することで，利用関係の詳細を表示できる．

4 Jasptool の適用例

オープンソースプロジェクト 4 個に対して Jasptool で分析を行った．一例として，jlgui[*2]に対して Jasptool で出力したグラフを図 1 に示す．また，グラフの生成時には，所属パッケージ毎に色を設定し描画を行っている．

得られたグラフには，以下のような特徴があった．

- 利用関係を多く持ったクラスは，ソフトウェアの中で核となる場合が多く，図 1 の a のように，中央に集まる．他のクラスから得られたデータを統合して画面に出力するような重要なクラスは中央に集まっている．
- パッケージで一つの機能を持つ時，図 1 の b のように，同パッケージのクラス同士は集まり，中央からは離れる．ある機能をパッケージ内だけで実現し，他のパッケージとの利用関係が入出力のみの場合が挙げられる．
- クラスが一つの機能をもち，特定の場面でのみ利用されるクラスは，図 1 の c のように，中央から離れる．ソフトウェア実行時に一度だけしか呼び出されないため直線になり，中央から離れた．

グラフの頂点同士の距離をもとに，群平均法でクラスタリングを行い樹形図を作成した．図 2 は樹形図の例で，機能ごとにパッケージを分けて設計されていて，かつ他パッケージ間との利用関係が少ない場合は，パッケージごとに

[*1] Classycle: http://classycle.sourceforge.net/.
[*2] jlgui: http://sourceforge.net/projects/jlgui/.

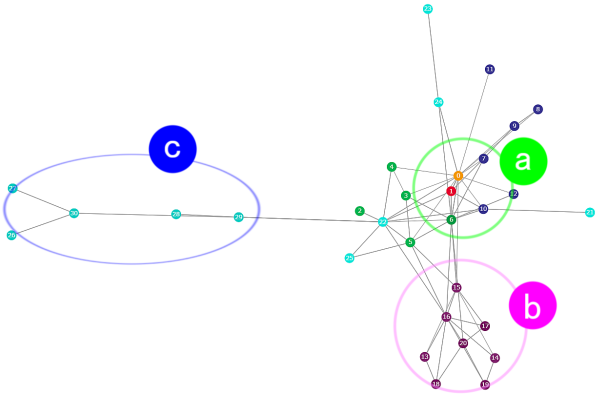

図 1　jlgui の部品グラフに対する出力結果

クラスタが生まれた．図 1 の b のように，他のパッケージとの利用関係が少ないパッケージは，図 2 のように樹形図でもまとまっていることがわかる．

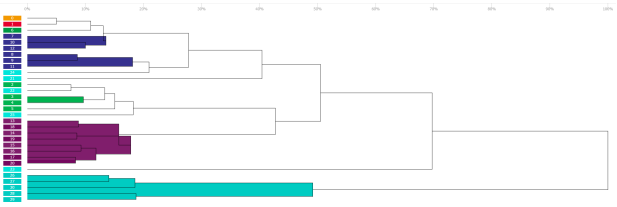

図 2　jlgui の樹形図

4 章の結果からは，一つの機能を実現するようなパッケージは，まとまりのある形で配置される例が多かった．一方で，実現する機能が幅広く他パッケージ間にも利用関係が多く存在する場合では，まとまりが確認できなかった．つまり，単一で固有の機能を実行できるように凝集度が高く，他のパッケージと利用関係が少ない結合度の低い構成のソフトウェアは，パッケージとしてまとまりがある形で表示される．提案手法を用いることで，部品間の独立性がわかるような形で部品グラフを表示できることが分かる．

5　Jasptool を用いた部品構造の改善

提案手法を用いることで，部品間の独立性がわかるような形で部品グラフを表示できることが分かる．ただし，開発者がグラフを見ただけでパッケージ内のクラス同士の凝集度が高く，パッケージ間の結合度が低くなるように設計することは難しいと考えられるので，Bridge パターン [3] を参考にして Jasptool を使った実現可能なリファクタリング手法について考察する．

はじめに，本ツールでソースコードを解析した結果から，得られた他のクラス間との関係がある要素のみを取り出し，プロトタイプを作成する．そのプロトタイプに存在するパッケージをモジュールとみなして，スプリングアルゴリズムを適用したグラフを確認しながら以下の手順を踏むことで，より効率的にモジュール分割と同等の効果が得られると考える．

1. 機能ごとにモジュールを分ける．
2. 各モジュールに対して，モジュール毎にパッケージ間の関係を持たせるインターフェースを用意する．
3. モジュール間のデータの行き来はインターフェースを通過するようにソースコードを書き換える．
4. インターフェースごとにテストケースを用意する．

図 3 では，実際に Jasptool のソースコードを手順に従って書き換えた例である．例のようにクラス間の関係が複雑な場合も，プログラムの機能を考慮してパッケージを分けて，機能と実装の橋渡しを追加することで部品の構成が分かりやすいものとなった。さらに，インターフェース部分にテストケースを用意することで，開発者が各モジュールの機能を把握することが容易になると考えられ，モジュールの独立性を十分に考慮した部品構成を構築することができる．

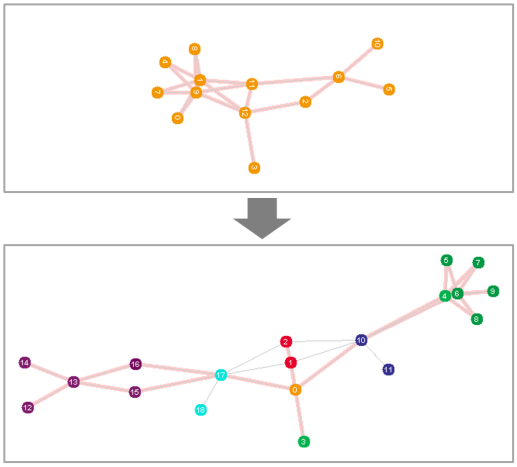

図 3　提案手法の例

6　まとめ

本研究では，スプリングアルゴリズムを用いて部品グラフを視覚化する手法を提案し，実際に適用を行い有効性を確認した．また，提案手法を用いて部品を機能的なまとまりとして分割するための方法論について考察し，ソフトウェアの部品構造の改善支援を行うツールとして有効であることを確認した．今後は，様々なソフトウェアに対して適用を行い，一般性の確認と手法の精錬を行いたい．

参考文献

[1] T.Kamada, S.Kawai, ”An algorithm for drawing general undirected graphs”, Information Processing Letters, Vol.31, No.1, pp.7-15, 1989.

[2] 花川 典子, ”論理結合マップとモジュール結合マップの重なりを用いたソフトウェア進化尺度の提案”, コンピュータソフトウェア, Vol.26, No.4, pp.157-172, 2009.

[3] Eric Gamma, Richard Helm, Ralph Johnson, John 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”, ソフトバンク クリエイティブ , pp.163-173, 1999.