4A-03

スプリングアルゴリズムを用いた ソフトウェア部品グラフの視覚化手法について

横森 励士　竹仲 孝盛

南山大学†

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，変更を行った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．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，ソフトウェアはより理解しやすくなる．その際，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ことが多い．

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，次のモデル上で物理シミュレーションさせることで各頂点を描画する，スプリングアルゴリズム[1]に着目する．

- 頂点を，質量を持つ小物質で，クーロンの法則に従う電荷とし，互いに反発させる．
- 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をマッピングし，ソフトウェア進化を計測する尺度を提案した．

3. スプリングアルゴリズムによる視覚化手法

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視覚化する機能を有するツールを Jasptool として作成した．Jasptool は Java プログラムを対象とし，Classycle [3] を用いて利用関係を抽出したうえで，部品グラフを作成し，スプリングアルゴリズムに基づいて表示する．このツールに対し実際に適用を行い，提案手法が視覚化手法として有効かを確認したうえで，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．

4. Jasptool の適用例

オープンソースプロジェクトに対して Jasptool で分析を行った例として，jlGui[4] に対して Jasptool で出力したグラフを図１に示す．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う．jlGui では 4 つの機能ごとにパッケージが分かれており，パッケージごとにまとまって描画されていることがわかる．

得られたグラフには，以下のような特徴がある．

- 利用関係を多く持ったクラスは，中核となるクラスであることが多く，中央に集まる．
- 図１の a のように，機能を統合して全体を扱うような重要なクラスは中央に集まる．
- パッケージで一つの機能を実現している場合，それらは中央からは離れて集まる．図１の b では，曲の情報を読み込む機能をもつ tag パッケージのクラスが集まり，tag.ui パッケージの 2 つのクラスのみが他のクラスと利用関係を持つ．

Visualization of Software Component Graph by using Spring Algorithm

†Nanzan University

- 特定の場面でのみ利用されるクラスは中央から離れる．図１の ｃ のように，設定ファイルの呼び出しを行う util.ini パッケージのクラスは，実行時に一度だけ呼び出されるので，中央から離れ直線状になる．

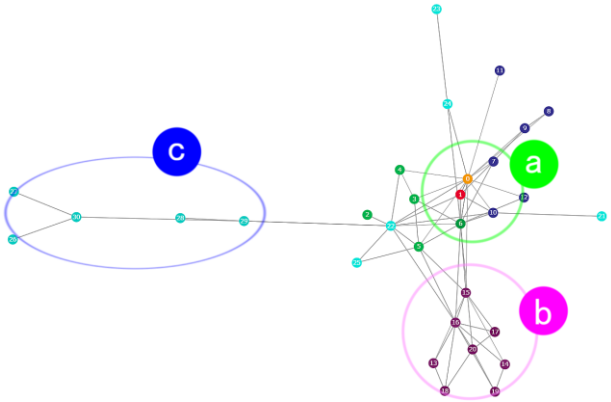

図１：jlgui の部品グラフに対する出力結果

次に，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，パッケージごとにクラスタが生じた．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の樹形図上でもまとまって表示される．

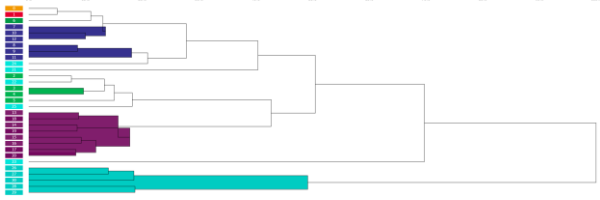

図２：jlgui の樹形図

このように，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，まとまりとならない．単一で固有の機能を実現し凝集度が高く，他との利用関係が少ない結合度の低い構成のクラス群が多い場合，パッケージとしてまとまって表示され，表示結果から，ソフトウェアを構成する部品群の独立性が推測できると考えられる．

5. Jasptool を用いた部品構造改善方法の考察

提案手法を用いることで，対象ソフトウェアの構成部品の独立性が推測できると考えられるが，開発者が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，グラフを見ただけで設計することは難しい．そこで，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，より効率的にパッケージ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. モジュールのインターフェース部分ごとにテストケースを用意する．

このように，プログラムの機能を考慮してパッケージ分割を行うことで，モジュール内の凝集度を最大にし，モジュール間の関係が最小になるような分割が可能となる．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の構成を実現できる．

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，有効性を確認した．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適用し，一般性の確認と手法の精錬を行いたい．

参考文献：

[1]Tomihisa Kamada, Satoru Kawai, ”An algorithm for drawing general undirected graphs”, Information Processing Letters, 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] Classycle: http://classycle.sourceforge.net/
[4] jlGui: http://www.javazoom.net/jlgui/jlgui.html