開発履歴に基づく分散処理フレームワークとアプリケーション間の関係の構築過程の調査

2009SE157 松波直弘　2009SE308 山田翔平
指導教員：横森励士

1　はじめに

ソフトウェアの保守工程ではシステムの現在の機能の維持，システムの変更への対応，現在の受け入れ可能な機能の改善，システム効率が容認できないレベルに落ちないよう予防などの要求に対応するような作業が行われ，それに伴いソフトウェア自身は大きく成長していく．成長に伴いソフトウェア内部の関係は複雑化し，把握が困難なものになっていく．[1] では GUI フレームワークとそれを利用するアプリケーション間の利用関係及びクローン関係について，バージョンごとに調査を行い，開発を通してどのように複雑化しているかを調査している．しかし，1 組の関係しか分析しておらず，分析として不十分であるとともに，より詳細な分析が必要である．

本研究では，[1] と異なるドメインのフレームワークとして分散処理フレームワークに注目し，分散処理フレームワークとそれを利用するアプリケーション内部の関係が開発を通してどう複雑化しているかを調査する．さらに，フレームワーク利用初期に生成されたクローン関係がどのようなものか，それらがどのように解消されているかを調査することで，分散処理フレームワークを利用した開発において注意すべき点などを考察する．

2　背景技術

2.1　部品間の関係と部品グラフ

一般的にソフトウェア部品とは，再利用できるように設計された部品とされている．それぞれの部品に注目すると，それらは互いに様々な関係を持っているといえる．例えば変数の宣言，インスタンスの作成，メソッドの呼び出し，フィールドなどで利用関係 [2] を構築している．一方で，類似したコード断片を共有している部品間にはコードクローンの関係が存在していると考えることができる．

本研究では，それらの関係を部品グラフとして表す．部品グラフにおいて，頂点は各部品を表し，辺は部品間の関係を表す．以下ではクローン関係，利用関係を対象として２種類の部品グラフを紹介する．それぞれ２つの部品間に複数の関係が存在する場合でも１つの関係として扱う．部品グラフでは，クローン関係を無向辺を使って表し，利用関係を有向辺を使って表す．

2.2　フレームワークとその利用によって生成されるクローン関係

近年のソフトウェア開発ではフレームワーク上にアプリケーションを構築する手法が広く確立されている．フレームワークにはモデル，コードパターンのような再利用可能な機構が含まれており，開発者は品質を維持しながら，独自の機能を備えたアプリケーションの開発期間を削減することができる．

フレームワークに関連したクローン関係は 2 種類考えられる．ソフトウェアをフレームワークを用いて実現しようとしたとき，フレームワークの利用例を参考に，アプリケーション側でよく似たコードを用いて実現することがある．この場合，図 1 のような形でフレームワークとアプリケーション間のクローン関係が生成する．もう一方は，フレームワークを利用しているコードを似た場面で同じように利用することがある．この場合，図 2 のような形のクローン関係がアプリケーション側の部品間で生成される．本研究では，それらの 2 種類のクローン関係に注目し，バージョンを通してこれらがどのように解消されるか分析する．

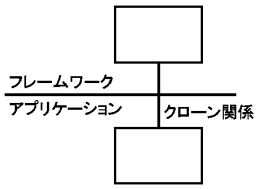

図 1　フレームワーク利用時に生じるクローン関係 1

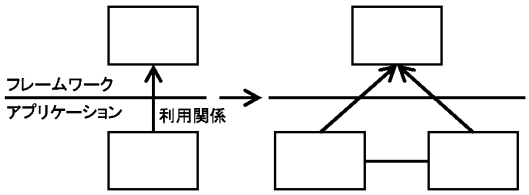

図 2　フレームワーク利用時に生じるクローン関係 2

3　部品間の関係調査に関する研究

3.1　先行研究

[1] では GUI フレームワークである JHotDraw とそれを利用するアプリケーションである JARP 間の関係について，バージョンごとに調査を行い，どのように変化しているか分析した．

アプリケーションからフレームワークへの利用関係を分析して得られた傾向として以下の２点が挙げられていた．

- フレームワーク側の各部品への入力辺はバージョンが進むごとに増加し，入力辺の最大数も増加した．
- アプリケーション側の各部品からの出力辺はバージョ

ンが進むごとに増加するが，出力辺の最大数には大きな変化が見られなかった．

クローン関係を分析して得られた傾向として以下の 3 点が挙げられていた．

- フレームワーク内のコードをコピーペーストして生成するクローン関係が存在した．
- 図 3 のように，フレームワークを利用する部分をクラスとして分離したり，継承関係でまとめた上でクラス構造が構築されているクローン関係が存在した．
- 図 4 右のように，フレームワークを利用する部分がコピーされてアプリケーション内にクローン関係が広がるとあると推測しているが，それらの存在を確認することはできなかった．

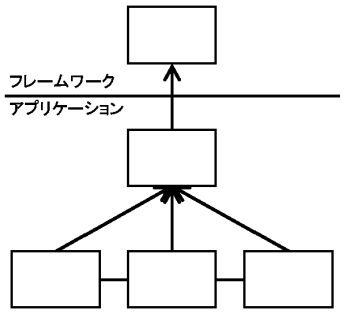

図 3 フレームワークの利用がまとめられたクラス構造

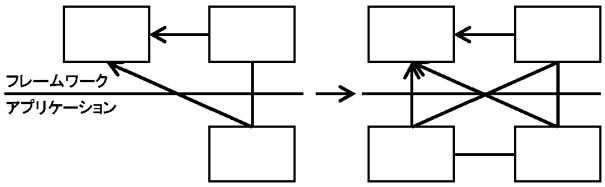

図 4 アプリケーション内に広がるクローン関係

3.2 先行研究の問題点

[1] では，JARP と JHotDraw という単一の GUI フレームワークとそれを利用するアプリケーション間の関係のみ分析しており，分析として不十分である．複数のアプリケーションとフレームワーク間の関係を調査，分析する必要がある．また，アプリケーションにおいてフレームワークを利用する初期段階でクローン関係がどう生成されているか，どう解消されているか分析されていないが，それらの情報は開発初期の指針として有益であると考えた．

4 分散処理フレームワークにおける部品間の関係の変化に関する分析

4.1 分析の目的

本研究では，分散処理フレームワークを対象に，フレームワークとアプリケーションの間に存在する利用関係の変化を分析する．全体的な傾向や減少点などに着目することで，分散処理フレームワークにおける特徴を分析し，注意するべき点などを考察する．分散処理フレームワーク特有の特徴的な処理手順を考慮すると，GUI フレームワークでのクローン関係生成の傾向と違いがあると考えられる．

また，機能が実現可能であることを示すことが主な目的となるフレームワークの利用初期では，クローン関係が生成されやすいと考える．本研究では，この時期に生成されるクローン関係に着目して，フレームワークの利用初期にどのようなクローン関係が生成されやすいか，どう解消されるかを分析し，分散処理フレームワークを用いた開発におけるクローン関係に関するノウハウを取得する．

4.2 調査方法

本研究では，フレームワークに関連したクローン関係と利用関係を調査する．利用関係の分析は Java のクラスとパッケージの依存関係のための分析ツールである Classycle[3] を使用する．クローン関係の分析はコードクローン抽出ツールである CCFinder[4] を使用する．本研究ではお互い 25 トークンより長い類似したコード断片を持っている場合，それら 2 つのファイルはクローン関係を構築しているとする．得られた結果から 2.2 節で説明したクローン関係を抽出する．

4.3 調査対象

本研究では実験に Java で書かれたオープンソースのソフトウェアを使用する．本研究では分散処理フレームワークとして Hadoop を選択した．Hadoop とそれを利用するアプリケーション間の関係を分析する．調査対象のアプリケーションとして機械学習やデータマイニングを行うアプリケーションである Mahout，Web 検索ソフトウェアである Nutch を選択した．Mahout は 7 バージョンに関して分析を行い，Nutch は ver0.1.0 から 0.7.2 の Hadoop 未対応分を除いた 12 バージョンに関して分析を行った．

5 利用関係の分析結果

Mahout，Nutch を対象にそれぞれのアプリケーション側の各部品がどれくらいフレームワーク側の部品を利用しているか，フレームワーク側の各部品がどれくらいアプリケーション側の部品から利用されているかを調査した．図 5，図 6 は Mahout の利用関係の数の推移を表しており，それぞれフレームワーク側の各部品への入力辺の数の推移，アプリケーション側の各部品からの出力辺の数の推移を表している．フレームワーク側の各部品への入力辺の数はバージョンを通して全体的に増加しており，クラス数や LOC が増加していくにつれて最大値も上昇した．一方で，アプリケーション側の各部品からの出力辺の数は最大数が頭打ちした．Mahout では ver0.3 から 0.4，ver0.6 から 0.7 にかけて利用関係が減少している点が存在した．また，Nutch でも同様の傾向が得られ，フレームワーク側の各部品への入力辺の数がアプリケーション側の各部品からの出力辺の数が ver1.4.0 から 1.5.0 にかけて利用関係が減

少している点が存在した.

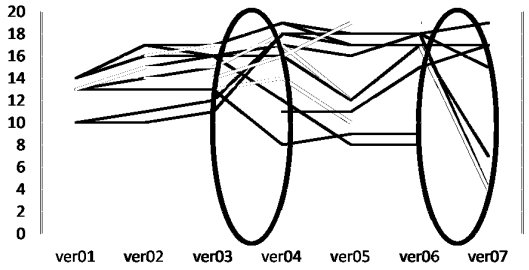

図 5 Mahout アプリケーション側の各部品からの出力辺の数の推移

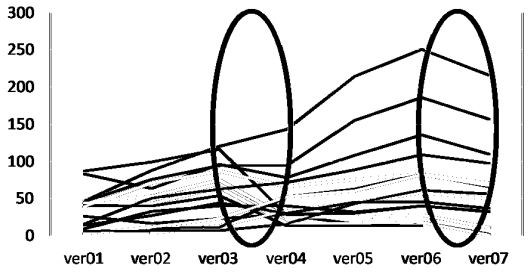

図 6 Mahout フレームワーク側の各部品への入力辺の数の推移

以下に利用関係が減少した原因とその例を示す.

5.1 (原因 1) Mapper，Reducer の変更

Mahout の ver0.3 から 0.4 にかけて DirichletMapper が org.apache.hadoop.mapred.Mapper から org.apache.hadoop.mapreduce.Mapper を利用するようになり DirichletMapper の利用関係の数が減少した.

5.2 (原因 2) フレームワークの利用がまとめられたクラス構造の構築

図 7 のように Mahout の ver0.3 から 0.4 にかけて BayesDriver，CBayesDriver がフレームワークを利用する部分を継承関係でまとめられて BayesDriver，CBayesDriver の利用関係の数が減少した.

5.3 (原因 3) 継承元の利用

図 8 のように Mahout の ver0.3 から 0.4 にかけて OutputUtil が JobConf を利用していたのをその継承元である Configuration を利用するように変更したことで，OutputUtil の利用関係の数が減少した.

5.4 (原因 4) 消滅

Mahout の ver0.3 から 0.4 にかけて MeanShiftCanopyClusterer の中のメソッド emitCanopy が消滅したことにより，フレームワークの利用辺が消滅した.

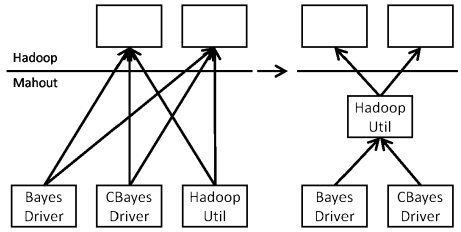

図 7 フレームワークの利用がまとめられたクラス構造の構築

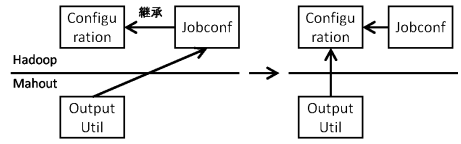

図 8 継承元の利用

6 クローン関係の分析結果

フレームワーク利用初期に生成されたクローン関係を分類した結果, 次のようなクローン関係が得られた．Mahout のフレームワークの利用初期に生成されたクローン関係を分析した.

6.1 フレームワークを利用している部品間におけるクローン関係 (Mahout)

Mahout のフレームワークを利用している部品間におけるクローン関係を分析した結果，以下の 3 種類のクローン関係が存在した.

6.1.1 分散処理の実行内容の記述部分におけるクローン

分散処理の実行内容が規定されている場所が複数あり，その中で同一の設定が与えられた場合にクローン関係が生成した．図 9 のように Mahout の ver0.1 から 0.2 にかけて Hadoop の JobConf を利用していた部品がその継承元である Configuration を直接利用するようになった．これによりアプリケーションの部品間に構築されていたクローン関係の一部が検出されなくなった．しかし，これらのクローンは本質的な解消が難しく，多くのクローン関係は解消されなかった.

6.1.2 アルゴリズムに関するクローン

同一の目的を持つ複数のアルゴリズム内で類似した処理が現れることでクローン関係が生成した．Mahout の ver0.6 から 0.7 にかけて Bayes，CBayes が NaiveBayes に移行したことにより Bayes，CBayes で構築されていたクローン関係は実質的に解消された.

6.1.3 コンバイナとリデューサに関するクローン

複数あるコンバイナとリデューサにおいて，似た機能を提供しているのでクローン関係が生成した．Mahout の ver0.6 から 0.7 にかけて FuzzyKMeans，KMeans におい

て Mapper，Combiner，Reducer が ClusterIterator に置換されたことにより FuzzyKMeans，KMeans で構築されていたクローン関係も消滅した．

6.2 フレームワークとアプリケーション間のクローン関係 (Mahout)

Mahout のフレームワークとアプリケーション間のクローン関係を分析した結果，以下の 2 種類のクローン関係が存在した．

6.2.1 分散処理の実行内容の記述部分におけるクローン

6.1.1 節で示したようなクローン関係が生成され，同様な結果が得られた．

6.2.2 リデュース操作に関するクローン

リデュースを実現するコードにおいて，ほぼ操作内容が同一であったのでクローン関係が生成した．Mahout の ver0.1 から 0.2 にかけて BayesThetaNormalizerReducer，BayesFeatureReducer，BayesWeightSummerReducer，CBayesThetaNormalizerReducer においてそれぞれの機能ごとに成長してクローン関係が解消した．

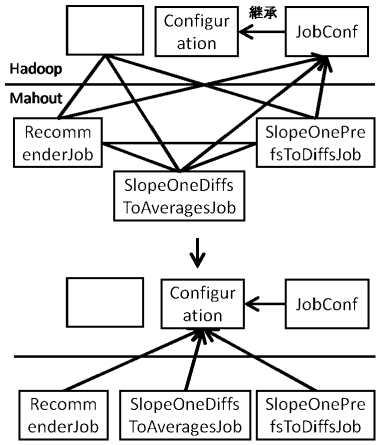

図 9 分散処理の実行内容の記述部分におけるクローン

6.3 フレームワークに関連するクローン関係 (Nutch)

フレームワークを利用している部品間におけるクローン関係，フレームワークとアプリケーション間のクローン関係ともに 6.1 節と同様の分散処理の実行内容の記述部分におけるクローン関係が生成され，これらのクローン関係の解消は難しく，解消されなかった．

7 考察

7.1 利用関係

Mahout，Nutch の利用関係を調査した結果，分散処理フレームワークとそれを利用するアプリケーションの間でも [1] と同様にソフトウェアの成長に伴ってフレームワーク側の各部品への入力辺の数は単調に増加した．また，アプリケーション側の各部品からの出力辺の数は最大数が頭打ちした．フレームワークとそれを利用するアプリケーション間の利用関係の減少した地点を調査することで，クラス構造などがどのように整理されたかを追跡できる．

7.2 クローン関係

Mahout，Nutch のクローン関係を調査した結果，Mahout の分析結果のように同じような機能を実現するために複数のアルゴリズムを使用していることでクローン関係が生成された例を確認した．これらのクローン関係は機能の実現を主な目的とした場合には起こりうるものであるが，最終的には整理されるべきものである．実際に開発の進行につれて，これらのクローンは解消された．再設計するにあたりこれらのクローン関係をまとめておき，整理することによってクローンを解消することができると考える．しかし，分散処理の実行内容の記述部分におけるクローン関係は実行内容によって個々に異なるので解消は難しく，解消されない傾向があるものも存在するので注意が必要である．

8 おわりに

本研究では，[1] と異なるドメインのフレームワークとして分散処理フレームワークに注目し，分散処理フレームワークとそれを利用するアプリケーション内部の関係が開発を通じてどう複雑化しているか調査した．利用関係に関して [1] と同じ傾向が得られた．クローン関係に関して，フレームワーク利用初期には実装優先で開発するので多く生成される傾向があるが，それらが開発が進むにつれてまとめられて解消されていく．その際，無理にクローン関係を解消すべきでないものも存在するので注意が必要である．フレームワーク利用初期には実装優先で開発してどのように動作するかを確認した上で再設計するというアプローチが実際のソフトウェア開発においてよく用いられ，有力な方法であると確認した．

参考文献

[1] R. Yokomori, H. Siy, N. Yoshida, M. Noro, and K. Inoue, "Evolution of Component Relationships between Framework and Application," *Journal of Computers, Computer Society of The Republic of China*, vol. 23, no.2, pp.61-79, 2012.

[2] I. Jacobson, M. Friss, and P. Jonsson, *Software Reuse*, Addison-Wesley Professional, 1997.

[3] Franz-Josef Elmer, *"Classycle: Analysing Tools for Java Class and Package Dependencies,"* http://classycle.sourceforge.net/, 2012.

[4] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: A multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp.654-670, 2002.