# 開発履歴に基づくGUIフレームワークとアプリケーション間の利用関係の構築過程の調査

2008MI110 小池良和　2008MI201 坂井亮太　2009SE122 清瀧拓実
指導教員:横森励士

## 1　はじめに

ソフトウェア開発工程の一つである保守工程では，システムの機能の維持，プラットフォームの変化への対応，不具合の修正などを目的として，さまざまな変更がソフトウェアに加えられる．これらの活動により，ソフトウェアの規模は徐々に大きくなり，内部の関係も複雑化する．先行研究 [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の移り変わりを分析しているが，1 組のソフトウェアしか分析しておらず，その中で行われている分析も不十分である．

本研究では，GUI フレームワークを利用する複数のアプリケーションに対して，バージョンが進むにつれて，利用関係がどのように複雑化するかを分析する．さらに，フレームワーク側の部品を多く利用しているアプリケーション側の部品において，利用数が減少している部分に注目して，どのような変化が起きているかを分析する．これらの分析をもとに，先行研究 [1] で得られた傾向が他のシステムでもあてはまるかの調査を行う．さらに，実際のソフトウェア開発に現れる利用関係が減少する修正内容がどのようなものかを提示し，実際の開発に役立つ知見の取得を目指す．

## 2　関連研究

### 2.1　ソフトウェア再利用

あたかも「部品」のように扱える形のソフトウェアの断片を用意し，それらの部品を組み合わせ，アプリケーションを組み立てることで，必要な労力の削減や作業効率の向上をはかることができる．このように既存のソフトウェア断片を利用して，ソフトウェアシステムを構築することをソフトウェアの再利用と呼ぶ [2].

実際に行われている再利用の例としてフレームワークが挙げられる．フレームワークとは，アプリケーションを開発する際に必要とされる汎用的な機能 (クラス) の集まりである．アプリケーションの骨組みとなる部分が既にフレームワークの中に用意されており，アプリケーションをフレームワークの構成をベースにアプリケーションを組み立てる．難しいアプリケーションの設計上の決断が，フレームワーク開発者によって既になされている．プログラマは単にそのひな型に肉付けをしていくだけで特定の機能を持つソフトウェアを構築することができる．

### 2.2　利用関係とコンポーネントグラフ

ソフトウェアを構成するクラスやファイルをソフトウェア部品とすると一つのソフトウェアは多くの部品で構成されていると考えることができる，このような部品の間には，変数の宣言，インスタンスの作成，メソッドの呼び出し，フィールド参照など「ある部品が他の部品を利用する」「他の部品からその部品呼び出される」という関係がある．それらの関係をコンポーネントグラフとして表現する．グラフでは頂点は部品，有向辺は利用関係を表す．2 つの部品間に複数の利用関係がある場合でも，それらをまとめて 1 つの利用関係を表す辺とする．例えば，図 1 はコンポーネントグラフの例である．このソフトウェアシステムは A〜E の 5 つの部品で構成されている．このグラフは部品 C が D と E，A と B が C を利用していることを示している．

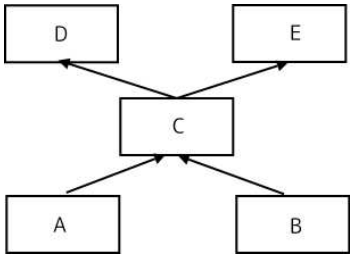


図 1　コンポーネントグラフ

### 2.3　フレームワークとアプリケーション間の利用関係

フレームワークを利用しているアプリケーションでは図 2 のような形で，アプリケーション側の部品が適宜対応する機能のフレームワーク側の部品を利用してフレームワークが備えている機能を利用している．本研究では，フレームワークとアプリケーションにまたがる利用関係に注目し，それらが開発を通じてどのように複雑化していくかを調べる．図 2 では，実線で表されている部分がフレームワークとアプリケーションにまたがる利用関係である．

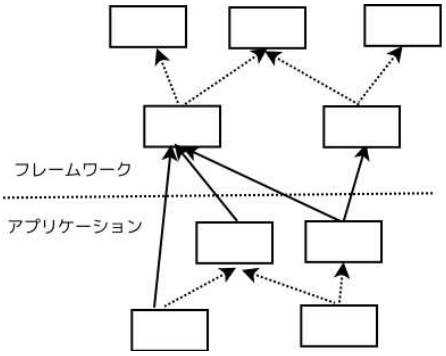


図 2　利用関係

### 2.4 過去の研究

先行研究 [1] では，2.3 節で示したフレームワークとアプリケーションにまたがる利用関係に注目して，アプリケーション側の部品から出る出力辺の数とフレームワーク側の各部品への入力辺の数を分析して，それらがどのように分布しているかを調査した．

利用関係の分布を分析して得られた傾向として，以下の 2 点が挙げられる．

- 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．

### 2.5 問題点

先行研究 [1] では，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調査対象が 1 つしかなく分析として不十分である．複数のアプリケーションを分析し，原因を調査する必要がある．

さらに先行研究 [1] では，ある一定以上大きくなると機能を実現する部分が別のクラスへ移動することで機能分割されるようだと述べているが，具体的にどのような行為が行われたかの分析が詳しく行われておらず，より詳しい分析が必要である．

## 3 フレームワークとアプリケーションにまたがる利用関係の分析結果

### 3.1 目的

#### 3.1.1 一般性の確認

フレームワークとそれを利用する複数のアプリケーション間の利用関係をそれぞれのバージョン毎に調査して，フレームワークとアプリケーション間の利用関係がどのように複雑化するかを調査する．これにより，先行研究 [1] で得られた知見が他のアプリケーションに対しても成り立つかどうかを分析する．本研究では，GUI フレームワークを研究対象としているがその理由として，先行研究 [1] で利用しているフレームワークと同一であるという事以外に，GUI フレームワークは規模が大きく，多くのアプリケーションに利用されており，利用関係の変化が多く見られるのではないかと考えたからである．

#### 3.1.2 利用関係の減少原因の分類

出力辺の数が大きく減少している部品について原因を追求し分類する．開発者がどのようなクラス構造の整理を行うことで利用関係が減少するのかをつきとめ，利用関係に基づく分析を行う際の指針とする．

### 3.2 調査対象

JHotDraw は，Java で記述されている図面エディタ用の 3D グラフィックスアプリケーションである．JHotDraw は，GUI フレームワークとしても利用されており，実験ではオープンソースソフトウェアの中から JHotDraw をフレームワークとして利用している Java アプリケーションを収集し，それらを分析対象とした．

これらのアプリケーションと GUI フレームワークである JHotDraw の間の利用関係の変化の傾向を比較する．ソフトウェアがバージョンアップする際に利用関係が減少した点を特徴的な点とみなし，その原因について調査する．実際に調査したアプリケーションは Renew，ChemSense，JStock，xmlBlaster の 4 種類で，以下それらに対する分析結果を紹介する．

### 3.3 実験内容

利用関係を分析するために Classycle を利用した．Classycle はクラスファイルを読み込んで，それぞれのクラスファイルに記述されている他のクラスの利用情報を入手し，それぞれのクラスの利用関係，被利用関係を出力する．各バージョン毎に Classycle を適用し，利用関係を入手したあとでフレームワークとアプリケーションにまたがる利用関係を抽出した．

抽出した利用関係から，利用関係が減少した特徴的な点が見られた場合，その前後のバージョンのクラスファイルのソースコードを比較し，利用しているクラスやパッケージを Classycle を用いてたどっていくなどして原因を探し出す．

### 3.4 実験結果

本節では，4 つのアプリケーションの調査結果をそれぞれ説明する．それぞれのアプリケーションの出力辺の変化の推移を図 3 から図 6 に示す．

図 3 は，ChemSense のグラフである．バージョンは 1.0 から 1.2 の 3 バージョン公開されており，その 3 つバージョン間で本研究の調査対象としている利用関係の変化が見られた．図 3 のグラフから一部利用関係が増加している点が見られるが基本的に一定で大きな機能の変化はない．そして，A で示されている場所のように利用関係が減少している点が見られた．図 4 は，JStock のグラフである．バージョンは 70 バージョンが公開されているが利用関係の変化が見られたのは図 4 中 B の 1.0.4j と 1.0.5 の間だけである．B において利用関係は大きく増加しておらず，一部利用関係が減少している点が見られた．図 5 は，xmlBlaster のグラフである．バージョンは 40 公開されているが利用関係の変化が見られたのは図 5 中 C の 1.0.7 と 1.1 の間だけである．利用関係は C で示されているように全体的に減少しているものが多い．図 6 は，Renew のグラフである．バージョンは 1.0 から 2.3 の 11 バージョン公開されており，そのすべてのバージョン間で変化が見られた．Renew は，バージョンアップされていくごとに利用関係の総数は増加しているが，図 10 のグラフと比較しても出力辺の数は大きく増加していないことがわかる．そして，D や E のような出力辺が減少している点が見られた．

次に，入力辺の変化の推移を図 7 から図 10 に示す．図 7 は，ChemSense の入力辺のグラフである．図 7 において 1.0 から 1.1 で入力辺の数が大きく増加していた．1.1 から 1.2 は特に変化が見られなかった．これは，1.1 から 1.2 では大きな機能の変化がなかったからである．図 8 は，

JStock の入力辺のグラフである．図 8 において利用関係は減少しているものが多い．一部増加している利用関係もあったが基本的には変化してない利用関係が多い．図 9 は，xmlBlaster の入力辺のグラフである．図 9 において利用関係に大きな変化が見られなかった．一部利用関係が減少している点が見られた．図 10 は，Renew の入力辺のグラフである．図 10 において入力辺の数はバージョンアップするごとに増加する傾向が見られた．1.6 から 2.0.1 で大きく入力辺が減少している．これは，クラスの構造が大きく変わったため入力辺の数が減少した．

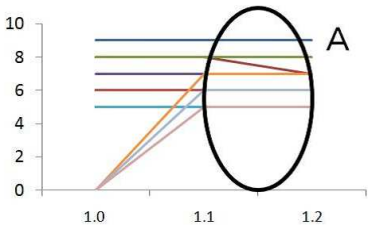


図 3 ChemSense の出力辺の変化

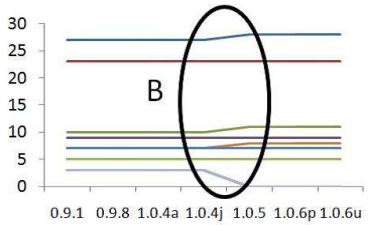


図 4 JStock の出力辺の変化

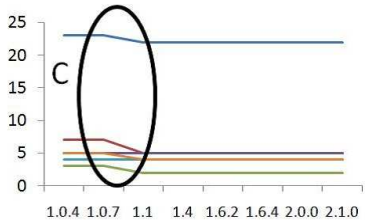


図 5 xmlBlaster の出力辺の変化

### 3.5 利用関係が減少した時の原因の調査

今回調査したそれぞれのアプリケーションの中に利用関係が減少している特徴的な点が見られた．図 3 から図 6 の A から E がそれにあたる．それぞれのアプリケーションの利用関係が減少した前後のバージョンの変化をまとめ，減少したクラスについてどのような変化が起きたか調査した．減少した点について分析し，原因を以下の 5 つに分類した．詳細は考察で述べる．

- 機能が別のパッケージ・クラスへ移動
- import の整理
- フレームワークとの中継クラスの整備
- 機能の削減
- 機能の実現の仕方が変わった

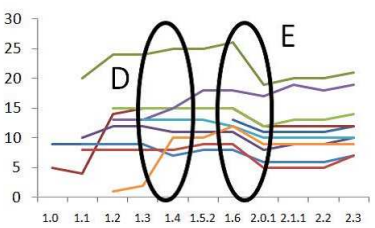


図 6 Renew の出力辺の変化

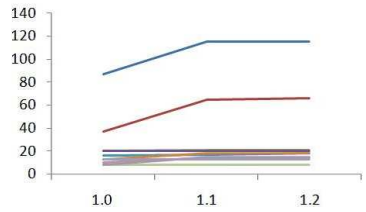


図 7 ChemSense の入力辺の変化

## 4 考察

### 4.1 一般性についての考察

今回 4 つのアプリケーションの利用関係を分析した．Renew については先行研究 [1] と同様の結果が得られたがその他の 3 つは大きな変化がなかった．ChemSense は 1.1 から 1.2 で大きな機能の変化が見られなかったため出力辺や入力辺で大きな変化が見られなかった．xmlBlaster と JStock の本研究で対象としている利用関係は何度もバージョンアップされている中で，1 度しか変化していなかった．これらの 3 つのアプリケーションはバージョン開発においてフレームワークを利用する部分が安定しており，利用関係の変化がほとんど見られなかった．一方で，Renew に関しては JHotDraw に関連した場所で機能追加が頻繁に行われており，大きな変化が見られ，先行研究と比較し同様の傾向が得られた．

### 4.2 利用関係の減少の原因についての考察

利用関係が減少した原因について 3.5 節のように大きく 5 つにまとめた以下の分類について考察する．

#### 4.2.1 機能が別のパッケージ・クラスへ移動

大きくなったクラスの機能が別のパッケージのクラスに移動しているのを確認した．例として Renew というアプリケーションの CPNApplication というクラスを挙げる．1.6 から 2.0.1 にバージョンアップした際，ソースコードが約 2000 行から約 500 行に大きく減少していた．なくなった部分の機能はメニューに関する機能に関する機能である．分割された機能は，CPNApplication と同じパッケージ内の Menu というパッケージ内で機能を実現されており，機能分割が行われたことがわかる．

#### 4.2.2 import の整理

バージョンアップに伴い必要最低限のクラスをインポートするように変更がなされることで実際に利用されるクラスがより明確になった．

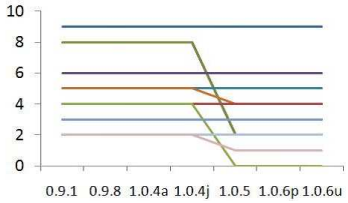


図 8 JStock の入力辺の変化

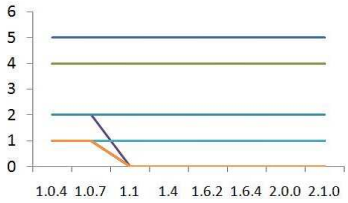


図 9 xmlBlaster の入力辺の変化

#### 4.2.3 フレームワークとの中継クラスの整備

図 11 のように，バージョンアップ前は CompositeFigure を直接利用していたがバージョンアップ後は xmlBlaster-Drawing が利用し，その結果を利用するようになった．

#### 4.2.4 機能の削減

例えば，JStock の AbstractOperator というクラスは，1.0.4j では input や output に関する機能があったがバージョンアップ後には削減されていた．始めの段階では必要な機能として実装されていたが，運用していきその機能が必要なくなり機能が削減されたのではないかと考えられる．

#### 4.2.5 機能の実現の仕方が変わった

図 12 のように，Vector メソッドが NullHandles を利用することで機能を実現していたのが Vector，layout，FigureSetChanged のメソッド内だけで機能を実現するように変更した．

#### 4.2.6 利用関係の減少原因のまとめ

利用関係が減少した原因としてこのようにアプリケーション側でフレームワークとの接点となるクラスを整備す利用関係が大きく減少した点では，4.2.1 のような機能分割が行われていた．その他の減少した点では 4.2.2 から 4.2.5 のようなクラス内の機能の整理が行われ利用関係が減少することを確認した．実際の分析において，利用関係が大きく減少した点を中心に調べることで機能分割の様子を把握できると考えた。

## 5 おわりに

本研究では，GUI フレームワークとアプリケーションの利用関係について調査した．アプリケーションと GUI フレームワークである JHotDraw の利用関係の変化を調査し，アプリケーション側とフレームワーク側の各部品について考察した．そして，利用関係の変化で特徴的な部分について詳しく調査し，その原因について考察した．

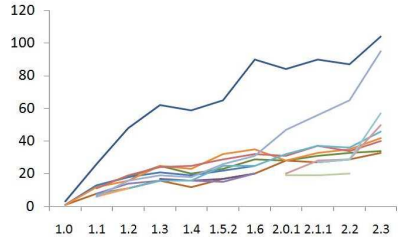


図 10 Renew の入力辺の変化

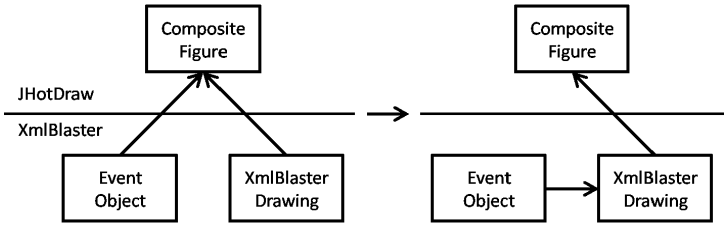


図 11 フレームワークとの中継クラスの整備

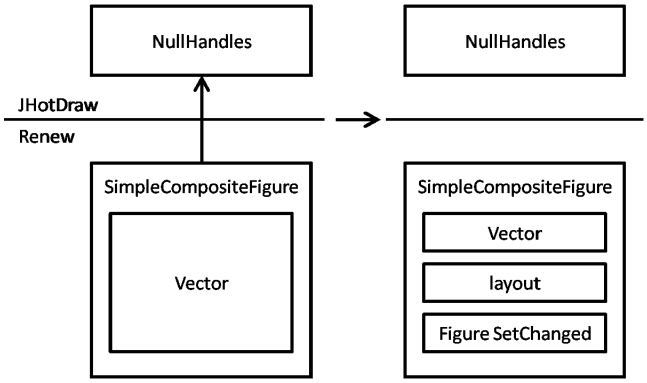


図 12 機能の実現の変化例

フレームワークに対する変更点が多いアプリケーションに関して，先行研究 [1] と同様の傾向が得られた．利用関係が減少した理由として，利用関係が大きく減少した点では機能分割が行われ，その他の点ではアプリケーション側でフレームワークとの接点となるクラスを整備することでフレームワーク間の利用関係が減少することを確認した．実際の分析において利用関係が大きく減少した点を中心に調べることで機能分化つの様子を把握できると考えた．

## 参考文献

[1] Reishi Yokomori, Harvey Siy, Norihiro Yoshida, Masami Noro, and Katsuro 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] C. Krueger, ”Software Reuse,” *ACM Computing surveys*, Vol. 24, No. 2, pp.131-183, 1992.