社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SS2005-37 (2005-08)

コンポーネントランクを用いたソフトウェアのクラス設計に関する分析手法の提案

市井 誠[†] 横森 励士[††] 松下 誠[†] 井上 克郎[†]

† 大阪大学 大学院情報科学研究科
†† 南山大学 数理情報学部 情報通信学科
E-mail: †{m-itii,matusita,inoue}@ist.osaka-u.ac.jp, ††yokomori@it.nanzan-u.ac.jp

あらまし コンポーネントランク法は，利用関係を基にソフトウェア部品の重要度の評価を行う手法であり，ソフトウェアリポジトリ中の重要な部品や再利用性の高い部品を知る事が出来る．本研究では，コンポーネントランク法を用いて，ソフトウェアのクラス設計に関する分析を行う手法を提案する．具体的には，ソフトウェア中で重要な役割をもつクラスを特定することにより，ソフトウェアのクラス設計の理解支援を行う．また，実際にソフトウェアに対する提案手法の適用実験を行った結果，ソフトウェア中で重要な部品を特定することができた．
キーワード コンポーネントランク，クラス設計，ソフトウェア理解

An Analysis Method of Software Class Design using Component Rank

Makoto ICHII[†], Reishi YOKOMORI[††], Makoto MATSUSHITA[†], and Katsuro INOUE[†]

† Graduate School of Information Science and Technology, Osaka University
†† Department of Information and Telecommunication Engineering, Nanzan University
E-mail: †{m-itii,matusita,inoue}@ist.osaka-u.ac.jp, ††yokomori@it.nanzan-u.ac.jp

Abstract Component Rank is a method for evaluating software component significance based on use relations, and we can know which component is significance or highly reusable in a software repository. In this paper, we propose a method analyzing software class design with component rank. Specifically, identification of important class for software comprehension. Also, we performed a case study with softwares and found that this method can identify important classes in the software.
Key words Component Rank, Class Design, Software Comprehension

1. はじめに

近年のソフトウェア開発の大規模化，および複雑化に伴い，ソフトウェアの再利用および管理を目的としたソフトウェア部品の蓄積および解析，検索を行うシステムが必要とされている．SPARS-J [10] は Java を対象としたソフトウェア部品検索システムであり，依存や類似といったソフウェア部品の特徴を生かした検索機能を提供する．また，パッケージブラウザやメトリクスの出力機能を用いてシステムの全体像が把握できる点から，企業でのソフトウェア資産の管理にも有用である [3].

SPARS-J は部品の順位付け手法としてコンポーネントランク法 [2], [3], [11] を用いている．この手法では，多くの部品から利用されている部品は重要であり，また，重要な部品から利用されている部品も重要であるという概念に基づき，それぞれの部品の順位付けが行われる．この重要度は再利用実績の高さとも言い換える事ができ，検索結果の順位付けにコンポーネントランクを利用する事で，SPARS-J は検索者に対してより再利用性が高く，重要な部品を提供する事ができる．

このように，コンポーネントランクは登録されたソフトウェア中の部品を再利用する際に用いられてきたが，登録されたソフトウェアそのものの分析にも利用できると考えられる．

あるソフトウェア中のクラスに対してコンポーネントランクを適用し，順位付けを行った結果について考える．この時上位に来るクラスは，ソフトウェア中の多くのクラスから利用されているクラスである．このようなクラスは，そのソフトウェアで中心となるクラスとみなす事ができる．また，コンポーネントランクにおいては上位クラスから利用されるクラスもまた上位に順位付けされるため，中心となるクラスと深い関連を持つクラスもまた上位に順位付けされる．このように，コンポーネントランクを用いる事でソフトウェアの中心となるクラスを知

る事ができると考えられる.

ソフトウェア, 特にオブジェクト指向設計の理解においては, まず中心となるクラスおよびその関連クラスの理解が重要であると考えられる. ソフトウェアの設計時には, ドメイン分析などにより中心となるオブジェクトを抽出し, 関連の記述や詳細化を行いクラス設計を行う. 続いて, シナリオやユースケースなどをオブジェクト間のメッセージのやりとりとし, 周辺のクラスを設計する [8].

このことから, 中心となるクラスを理解する事により, ソフトウェア全体の理解が容易になると考えられる. しかし, ソフトウェアの規模が大きくなれば, その把握は困難となる. 設計に関するドキュメントが存在すれば理解の助けになるが, ドキュメントが巨大である, 古い, そもそも存在しないなどの理由で参考に出来ない場合もある.

このようなソフトウェアのクラス設計に関する情報は, ソフトウェアの開発者ならば与えられなくても把握できている事が多い. しかし, ソフトウェアの開発者と保守者が異なる事は珍しく無く, このような場合は保守の作業に先駆けてソフトウェアの理解が必要となる. しかし, 前述した理由により, クラス設計に関する理解は困難であると考えられる.

そこで, 本研究では, コンポーネントランクを用いてソフトウェア中の重要部品を特定する事による, ソフトウェアのクラス設計に関する分析手法の提案を行う. また, 実際のソフトウェアに対する適用実験を行い, その結果に関する考察を行う.

以降, まず 2. 節でコンポーネントランクについて述べた上で, 3. 節で提案する手法の詳細について述べる. 4. 節ではソフトウェアの適用実験について述べる. 5. 節では関連研究について述べる. 最後に, 6. 節でまとめと今後の課題について述べる.

2. コンポーネントランク

2.1 コンポーネントランク法による順位付け

一般にソフトウェア部品とは再利用出来るように設計された部品の事を指す [5]. ここではより一般的に, 再利用の対象となるソフトウェア資産の単位を指してソフトウェア部品, または単に部品と呼ぶ. 特に Java を対象とした場合については, Java のクラスがソフトウェア部品となる.

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合うことで一つの機能を提供する. いま, ある部品がある部品を利用する時, この部品間に利用関係が存在すると言う.

部品を頂点, 利用関係を有向辺としたグラフを部品グラフと呼ぶ. 図 1(a) は部品グラフで, この場合, 部品 c_2 は c_1 を, c_2', c_3 は c_1' を, c_1 は c_4 を, c_1' は c_5 をそれぞれ利用していることを表している.

実際の部品の集合には, 全く同じ, もしくはほとんど同じ部品が複数存在している事がある. これらは, ある部品がコピー, もしくはコピー後一部を変更して再利用されているとみなす事ができる. コンポーネントランク法ではこのような部品を類似部品群としてまとめ, 解析の単位としている.

図 1(a) において, 部品 c_1 と c_1', 部品 c_2 と c_2' はそれぞれ類似した部品とする. このとき, 図 1(b) が, 部品群グラフとなり, 部品群に属する部品はそれぞれ

$C_1 = \{c_1, c_1'\}$, $C_2 = \{c_2, c_2'\}$, $C_3 = \{c_3\}$,
$C_4 = \{c_4\}$, $C_5 = \{c_5\}$ である.

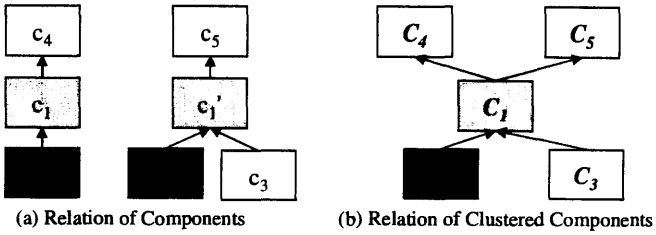

図 1 部品グラフと部品群グラフ

各部品群に対する評価値は, この部品群グラフを元に求められる. まず各頂点に対して, 総和が 1 となるような重みを与える. 続いて, 各頂点の重みをその頂点を始点とする有向辺に配分する形で, 各辺の重みを決定する. そして, 各頂点の重みを, その頂点を終点とする辺の重みの合計値として再定義する. 以上を各頂点の重みが収束するまで繰り返した時の最終的な重みが, 対応する部品群に対する評価値: コンポーネントランク値となる.

ただし実際には以下のように求められる. W を各頂点の重みのベクトル, 行列 D を配分率, すなわち各有向辺の重みのその始点の重みに対する割合の行列とし, D^t は D の転置行列とする.

$$W = D^t W \tag{1}$$

この W を求めることで, 各頂点に対応する部品群の評価値を計算することができる. 式 (1) の性質より, D^t の固有値 1 の固有ベクトルを求める事で W を求める事ができる.

以降, 以上により求められた評価値をコンポーネントランク値, コンポーネントランク値の高い順に部品を整列した際の順位をコンポーネントランクと呼ぶ.

コンポーネントランク値は, 開発者が, 開発者が利用関係に沿って参照を行うと仮定した場合の各部品の参照されやすさをあらわしており, よく利用される部品や, 重要な部品から利用される部品の順位が高くなる.

2.2 SPARS-J

SPARS-J は Java を対象としたソフトウェア部品検索システムであり, 登録された部品に対してコンポーネントランク法を適用し, 順位付けを行っている. SPARS-J は図 2 に示すような構成となっている.

登録部 (Register) は与えられた Java ソースファイルをデータベース (Database) へ部品として登録する. この時, Java ソースファイルからの単語の切り出しや利用関係の解析などが行われる. コンポーネントランク計算部 (Ranker) はデータベースに登録された部品全てに対し, 利用関係に基づきコンポーネントランク値の計算を行い, 結果をデータベースに登録する. 検索部 (Searcher) には, 部品の検索や, 表示を行う CGI, および部品情報の一覧出力を行うコマンドが含まれ, データベースに登録された部品情報をユーザに提供する.

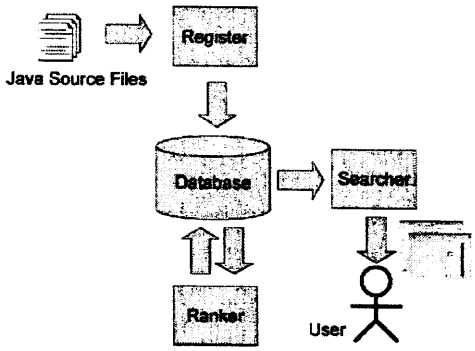

図 2　SPARS-J の構成

3. 提案手法

本研究では，2. 節で説明したコンポーネントランク法を利用し，クラス設計に関する分析を行う手法を提案する．コンポーネントランク法では，単純な被参照数，すなわち対象のクラスを利用しているクラスの総数を用いた評価と異なり，重要なクラスから利用されるクラスに対してより大きな値が与えられる．これより，単に多くのクラスから利用されるクラスだけではなく，ソフトウェア中で重要な意味を持つクラスや，重要な部位から利用されるクラスを知る事が出来る．ソフトウェアのクラス設計の理解において，コンポーネントランクの上位クラスの理解から開始する事により，全体の把握が容易になると考えられる．

3.1 手　　順

提案手法の手順を以下に示す．

（1） コンポーネントランク値の計算

解析対象のソースコードを用意し，SPARS-J への登録を行いコンポーネントランク値を計算する．この際，利用しているライブラリの Jar アーカイブや，ビルドのためのクラスパス情報などは必要無く，解析対象のソースコードのみが有れば良い．

（2） 重要クラスの特定

コンポーネントランク順に全てのクラスをソートし，閾値以上の順位のクラスを重要クラスとする．閾値は以下の 3 段階で設定する．

閾値 (1)　上位 3〜10 クラス
閾値 (2)　上位 10%
閾値 (3)　上位 20%

なお，この時のクラス数や割合は，総部品数およびコンポーネントランク値の分布に応じて増減させるべきだと考えられる．また，この際に被参照数による順位も求めておく．

（3） 重要クラスの分析

重要クラスに関して，役割や他クラスとの関連をソースコードの内容から調査する．まず，閾値 (1) 以上のクラスに関して，役割や，それらの間での関連を SPARS-J 検索部の部品表示機能・利用関係表示機能を用いて調査する．またメソッドやフィールドの意味について，可能ならば他のドキュメントも参照しながら調査する．続いて閾値 (2) のクラスに関して，閾値 (1) 以上のクラスとの関連に注意しながら，役割や関連を調査する．さらに，閾値 (3) 以上のクラスに関しても同様に，コンポーネントランク上位のクラスとの関連に注意しながら，役割や関連を調査する．

この時，コンポーネントランクによる順位と，被参照数による順位を比較し，それらの差が大きいクラスについては何らかの設計的な要素が影響していると考えられるので注意して分析を行う．例えば，被参照数での順位と比較してコンポーネントランクで上位にあるクラスは，ソフトウェア中で重要な部分で利用されるクラスである事が考えられる．逆にコンポーネントランクによる順位の方が低いクラスについては，あまり重要な役割の無い，様々な所から呼び出されるユーティリティ的なクラスである事が考えられる．

また，ソースコードからのリバースエンジニアリングにより，閾値以上のクラスのみから構成されるクラス図を作成する．これにより，重要クラス間の関連を視覚的に確認でき，分析作業がより容易になると考えられる．一般に，ソースコードから生成したクラス図は，全てのクラスおよび関連を取得し表示してしまうために巨大で理解が困難になりがちである．しかし，ここで生成されるクラス図は，コンポーネントランクにより重要なクラスのみに絞られているために，小規模で理解しやすく，かつ要点を押さえている事が期待できる．

4. 適用実験

本節では，提案手法の有用性を評価するために，実際のソフトウェアに対して適用を行う．そして，クラス設計に関する重要な情報が分析できるかどうかを評価する．

適用対象は，研究室内で開発されたソフトウェアおよび，ある企業で開発されたソフトウェアである．以下，それぞれについて述べる．

4.1 研究室内で開発されたソフトウェア

4.1.1 適用対象

適用実験はコードクローン分析ツール Aries [4] のクローン情報表示部のソースコードに対して行った．総クラス数は 283 クラス，総行数は 31,000 行である．

Aries 全体ではなくデータ表示部のみを対象とした理由は以下の通りである．Aries は大きく分けてクローン解析部とクローン情報表示部，およびそれらを統合的に扱う部分に分かれるが，それぞれの間で依存関係は無く，独立している．これらを同時に扱った場合，元々関連の薄いクラスを同時に分析する事になり，分析の煩雑さが増大するからである．また，これらの中でデータ表示部が最も大きい規模であったため，データ表示部に対する分析を行った．

4.1.2 適用結果および考察

a） コンポーネントランク値の分布

図 3 より，ごく一部のクラス，具体的には上位 5 クラス程度が特に高いコンポーネントランク値を持ち，以降はなだらかになっている．これより，閾値 (1) は 5 クラスに設定する．また，閾値 (2),(3) に関しては，前節で示した値，すなわち上位 10%，

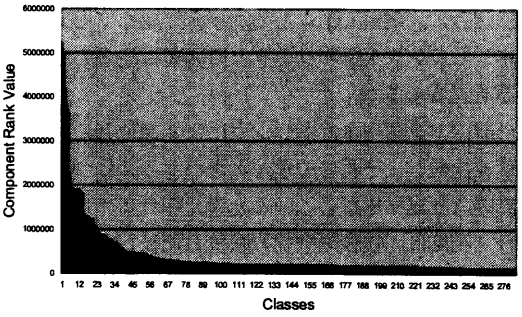

図 3　CR 値の分布

上位 20%に設定する.

b)　重要クラス

閾値以上の重要クラスを表 1 に示す. ここでは閾値 (2) 以上のもののみ示しており, 閾値 (1) 以上のクラスについては太字で示している. クラスの概要とは, ソースコード中のクラスのドキュメンテーションコメントを要約したもので, 利用関係や内部構造については考慮していない. また, パッケージ名はソフトウェア内で共通な部分については省略して記述している. 例えば, パッケージ data.clone は, 正確には jp.ac.osaka_u.es.ics.sel.gesmcoc.data.clone であり, jp.ac.osaka_u.es.ics.sel.gesmcoc. については省略している.

また, クラス図の生成に関しては UML モデリングツール JUDE [6] を用い, ソースコードからクラス図へのリバースエンジニアリングを行った. 閾値 (3) 以上のクラスでのクラス図の一部を図 4 に示す.

図中で示される関連は, ラベルとして示される名前でフィールドが宣言されている事をあらわしている. ローカル変数の宣言や, メソッドの引数および返り値で利用されている場合については関連は引かれておらず, この図からは読み取る事はできない. また, 閾値以上でも図中に無いクラスへの関連は非表示としてある.

c)　重要クラスの分析

i)　閾値 (1) 以上のクラスの分析

閾値 (1) 以上のクラスに関して, 表 1 およびソースコードから詳しく分析を行った結果は以下の通りである.

CloneSet　クローンセットを表現する抽象クラスであり, 各種のメトリクスをフィールドとして持つ. また, Fragment を引数とするメソッド add()・Fragment の配列を返値とする toArray() メソッドをもつ事から, 意味的に Fragment の集合を内包する事がわかる.

Variable　クローンの外部で定義される変数をあらわすクラスである. String 型, およびプリミティブ型のフィールドのみを持つデータクラスである. 被参照数では 17 位であるが, コンポーネントランクでは上位である事から, 重要な部分で利用されている事が分かる.

Fragment　クローンのコード片をあらわすクラスである. フィールド名より, このクラスにはファイルに関する情報やクローングループに関する情報が含まれる事がわかる. また, Variable の各フィールドを引数とするメソッド addVariable(), Variable の配列を返値とするメソッド toArray() をもつことから Variable の集合を内包することがわかる. また, CloneSet が Fragment を持つ事より, CloneSet, Variable, Fragment の 3 クラスは関連を持つことがわかる.

DeclarationUnit　Java クラスの単位をあらわし, その位置情報やメトリクスを保持するデータクラスである.

PackageName　パッケージ名, およびクラス名をあらわすデータクラス. 被参照数では 44 位であり, 重要な箇所で利用されることがわかる.

以上より, 上位クラスはクローン分析時の重要な概念であるクローンセットや, その関連クラスをあらわすエンティティクラスである事がわかった. これは, Aries がコードクローン分析ツールであり, 特に, 対象としている表示部は解析済みのデータを扱う, という事実と一致する. また, Variable については直接コードクローンと関係ないが, Aries はコードクローンがリファクタリング可能であるかどうかの判断要素としてクローンの外部定義変数を用いている事から, 重要クラスとして妥当であると考えられる.

ii)　閾値 (2) 以上のクラスの分析

図 4 より, CloneSet はサブクラス StatementCloneSet, DeclarationCloneSet, ProcessCloneSet を持つ. それらの被利用クラスのうち, 閾値 (2) 以上のものをみると, 例えば StatementCloneSet ならば, パッケージ gui.cloneset.statement.metrics のクラス MetricsGraph から利用されている. DeclarationCloneSet についても同様に, パッケージ名のみが異なる同名クラスから利用されている. これは, Aries ではクローンセットは文単位・宣言単位・メソッド単位という単位別に扱われ, それぞれ別のメトリクスグラフをもち, さらにそれぞれは独立したクラスで実装されているという事実を反映している.

CloneGroup クラスに注目する. このクラスは, 互いに依存関係を持つクローンセットをあらわすエンティティクラスであり, 概念として重要であるが, 他のエンティティクラスと比較して順位が低い. これは, 全体の中でクローングループを扱う機能の比重が低いという事をあらわしている. 実際, クローングループは最近実装され始めた機能である.

また, その他の GUI 部品クラスとして, ClassListViewPopupMenu, PackageTreePopupMenu, SourceCodePainView, MetricsGraph などが挙げられる. これらより, Aries の画面を構成する要素を読み取る事ができる. ただし, 直感的には ClassListViewPopupMenu よりも, 閾値 (2) 以下である ClassListView の方がより重要でありコンポーネントランクでの順位に反している. この点に関しては理解を妨げていると考えられる. しかし, Aries においてはポップアップメニューが多用されているという点を考えると大きな問題では無いと言える.

その他に読み取れる点としては, ファイル関連の FileData などのクラスがあり, 単なるデータ入出力では無くクローン分析に利用されるファイルのことである点や, このソフトウェ

表 1　閾値 (2) 以上のクラス群

CR 順位	パッケージ名	クラス名	被参照数での順位	クラスの概要
1	**data.clone**	**CloneSet**	**4**	**クローンセット**
2	**data.clone**	**Variable**	**17**	**クローンの外で宣言された変数**
3	**data.clone**	**Fragment**	**1**	**クローンセットに含まれるコード片**
4	**data.classes**	**DeclarationUnit**	**6**	**単体のクラス**
5	**data.classes**	**PackageName**	**44**	**パッケージ名もしくはクラス名**
6	data.clone	CloneType	3	クローンのタイプに関する定数定義
7	gui.classes.list	ClassListViewPopupMenu	17	クラス一覧のポップアップメニュー
7	gui.classes.tree	PackageTreePopupMenu	17	パッケージツリーのポップアップメニュー
9	data.classes	DeclarationUnitData	23	単体のクラスの集合
10	gui.classes.source	SourceCodeUI$SourceCodePlainView	96	ソースコード表示領域
10	gui.clonegroup.source	SourceCodeUI$SourceCodePlainView	96	同上
10	gui.cloneset.common.source	SourceCodeUI$SourceCodePlainView	96	同上
13	data.clone	VariableData	96	Variable の集合
14	data.file	FileData	2	ファイルの集合
15	data.clone	StatementCloneSet	5	文単位でのクローンセット
16	gui.cloneset	MetricsGraph	7	メトリクスグラフ
17	gui.cloneset	MetricsGraphPopupMenu	9	メトリクスグラフのポップアップメニュー
18	gui.classes.tree	PackageTree	9	パッケージツリー
19	data.clone	ProcessCloneSet	7	メソッド単位でのクローンセット
20	data.file	Element	23	ファイル
21	data.clone	DeclarationCloneSet	9	宣言単位でのクローンセット
22	gui.classes.list	ClassListView	12	クラス一覧の表示領域
23	data.clone	CloneSetData	17	クローンセットのデータ
24	gui.cloneset.declaration.metrics.graph	MetricsGraph	16	メトリクスグラフ
25	gui.cloneset.declaration.metrics.graph	MetricsGraphPopupMenu	22	メトリクスグラフのポップアップメニュー
26	gui.cloneset.statement.metrics.graph	MetricsGraph	17	メトリクスグラフ
27	data.file	Key	96	Element を比較するキー
28	data.clone	CloneGroup	23	クローングループ

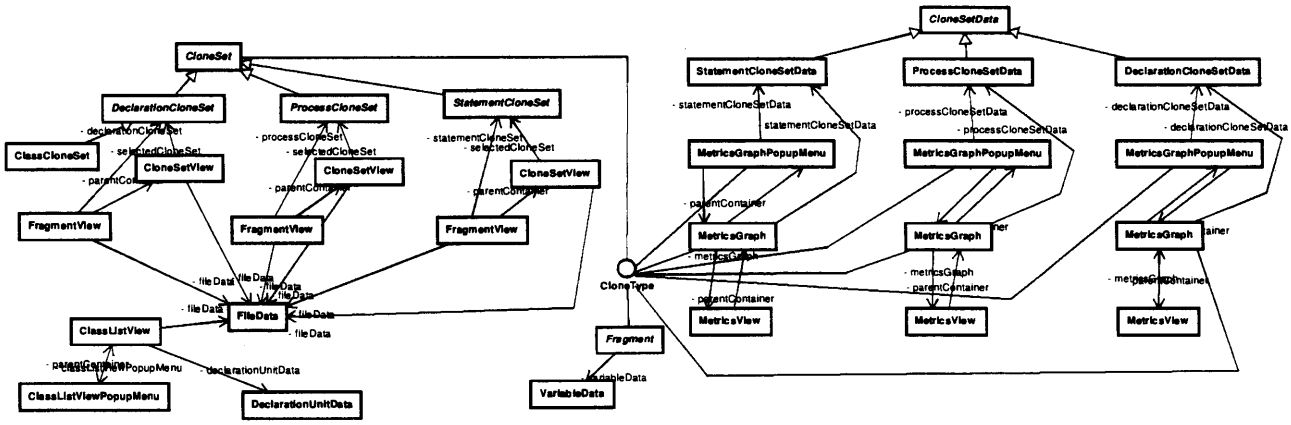

図 4　閾値 (3) 以上のクラス図の一部

アではエンティティクラスの集合を Collection などではなく DeclarationUnitData, VariableData, CloneSetData などのクラスを通して扱っている，という点が挙げられる．

iii)　閾値 (3) 以上のクラスの分析

ソースコードおよびクラス図をもとに分析を行う．

CloneSetData は，サブクラス StatementCloneSetData, DeclarationCloneSetData, ProcessCloneSetData をもち，それぞれ，StatementCloneSet, DeclarationCloneSet, ProcessCloneSet の集合をあらわすクラスで，MetricsGraph との関連をもつことがわかる．

また CloneSetView, SourceCodeUI, MetricsView, FlagmentView など，主要な GUI 部品が含まれる．このような GUI 部品の関連は，クラス図により容易に確認できる．例えば，CloneSetView は FragmentView の親コンテナであり，どちらも他の CloneSet の子クラスのいずれか (StatementCloneSet など) を扱う事がわかる．

4.1.3　分析に関する考察

以上の分析より，Aries の主要なエンティティクラス，および GUI 部品のクラスを特定し，その機能や関連を確認する事ができた．また，この結果は，実際の Aries の設計と一致し，本手法が有効である事を示している．

4.2　企業における適用実験

4.2.1　適 用 対 象

対象としたソフトウェアはデータベース・ロジック・プレゼンテーションの 3 層構造をもつ Web アプリケーションであり，規模は約 850 クラス,110,000 行である．それぞれのレイヤーを担当するクラスおよび，エンティティクラスを含む共通クラス群や，レイヤー間のデータ受け渡しを行うパラメタクラスが存在する．また，自動生成クラスも含まれている．なお，分析は閾値 (1) 以上のクラスに関する分析のみを行った．

4.2.2　適用結果および考察

閾値 (1) としては，上位 10 クラスとした．その内訳としては，エンティティクラスが 5 クラス, ユーティリティクラスが 2 クラス，および自動生成のパラメタクラスが 3 クラスであった．開発者への聞き取り調査により，これらのクラスがソフトウェア中での重要クラスとして妥当である事が確認できた．

しかし，レイヤー間の受け渡しは局所的な役割であり，閾値(1)以上に来る事は不自然であるとの見方もできる．このようなクラスが上位に順位付けされたのは，以下の2点が原因だと考えられる．まず1つめの理由としては，実際にその部位がシステム中で大きな比重を占めている事である．実際に，関連するレイヤーであるデータベース層・ロジック層は，あわせて全体の半分以上である480クラスをもっている．もう1つの理由として，JSPファイルの登録を行わなかった為に，利用関係が不完全であった事が考えられる．JSPファイルからJavaファイルへの変換は対象ソフトウェアの実行環境に依存し，今回の実験では行わなかったが，出来る限り変換を行い登録する事が望ましいと考えられる．

5. 関連研究

クラス設計の分析手法として，ソフトウェア中のクラスをライブラリとしてのクラスとアプリケーションとしてのクラスに分類する手法[1]が提案されている．この手法は，汎化-特化構造の指標(HF)，全体-一部構造の指標(RF)，ポリモルフィズムの指標(PF)の3つの指標を定義し，これらを組み合わせる事でクラスの分類を行っている．

コンポーネントランクの応用として，版管理されている部品の各リビジョンにおけるコンポーネントランク値を求め，その変遷よりソフトウェア開発の状況を推測する手法[9]が提案されている．また，再利用を目的とした，コンポーネントランクを用いた理解支援手法[7]も提案されている．この手法では，まずメソッドをひとつ決定し，そのメソッドに依存するメソッドを分析する．それらのメソッド内でコンポーネントランクの計算・およびメソッドごとのサイクロマチック数の計算を行い，最初のメソッドの理解に重要なメソッドを示す事で，再利用時の理解支援に役立てている．

6. まとめ

本研究では，コンポーネントランク値を用いてソフトウェアのクラス設計を分析する手法を提案した．また，実際のソフトウェアに対する適用実験を行い，提案手法はソフトウェアをクラス設計の面から理解する事に有効である事が分かった．

今後の課題としては，適切な閾値の決定が挙げられる．今回はクラス数とコンポーネントランク値の分布から閾値を決定し，良好な結果が得られたが，全ての場合に適用できるかどうかは分からないため，適切な閾値の決定方法の確立が今後の課題となっている．

また，本研究ではクラス図の生成に一般のUMLツールを利用したが，閾値以上のクラスを人間が選択し，図に加えていく，という作業が必要であった．また，全ての関連が表示できていないという問題があった．これらより，分析に必要な情報を備えたクラス図を自動生成する事が望ましいと考えられる．

最後に，より多くの適用実験が挙げられる．本研究では表示に重点が置かれたGUIソフトウェアに対して適用を行ったが，処理に重点が置かれたソフトウェアや，大量のエンティティクラスを扱うソフトウェアなど，異なった設計のソフトウェアに対する適用が必要であると考えられる．また，より規模の大きいソフトウェアに対しての適用も必要である．

謝辞 本研究は一部，文部科学省リーディングプロジェクト「e-Society基盤ソフトウェアの総合開発」の支援を受けている．

文献

[1] 青木 淳: "オブジェクト指向システム分析設計入門", ソフトリサーチセンター, 1992.

[2] Katsuro Inoue, Reishi Yokomori, Hikaru Fujiwara, Tetsuo Yamamoto, Makoto Matsushita, Shinji Kusumoto: "Component Rank: Relative Significance Rank for Software Component Search", Proceedings of the 25th International Conference on Software Engineering (ICSE2003), pp14-24, Portland, Oregon, U.S.A., 2003.

[3] Katsuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, Shinji Kusumoto: "Ranking Significance of Software Components Based on Use Relations", IEEE Transactions on Software Engineering, Vol.31, No.3, pp.213-225, 2005.

[4] 肥後 芳樹, 神谷 年洋, 楠本 真二, 井上 克郎: " コードクローンを対象としたリファクタリング支援環境", 電子情報通信学会論文誌 D-1, Vol.J88-D-I, No.2, pp.186-195, 2005.

[5] I. Jacobson, M. Griss and P. Jonsson: "Software Reuse", Addison Wesley, 1997.

[6] "JUDE", http://jude.esm.jp/

[7] 小堀 一雄, 山本 哲男, 松下 誠, 井上 克郎: " メソッド間の依存関係を利用した再利用支援システムの実装", 電子情報通信学会技術研究報告, SS2004-58, pp.13-18, Vol.104, No.722, 2005.

[8] J. Schmuller: "Teach Yourself UML in 24 Hours", Sams publishing, 1999.

[9] 横森 励士, 市井 誠, 井上 克郎: " コンポーネントランクを用いた開発プロセス評価手法の有効性について", 情報処理学会研究報告, 2005-SE-149, 2005. (掲載予定)

[10] 横森 励士, 梅森 文彰, 西 秀雄, 山本 哲男, 松下 誠, 楠本 真二, 井上 克郎: "Java ソフトウェア部品検索システム SPARS-J", 電子情報通信学会論文誌 D-I, Vol.J87-D-I, No.12, pp.1060-1068, 2004.

[11] 横森 励士, 藤原 晃, 山本 哲男, 松下 誠, 楠本 真二, 井上 克郎: " 利用実績に基づくソフトウェア部品重要度評価システム", 電子情報通信学会論文誌 D-I, Vol.J86-D-I, No.9, pp.671-681, 2003.