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

横森励士[†]　市井誠[‡]　井上克郎[‡]

[†] 南山大学数理情報学部情報通信学科
〒 489-0863 愛知県瀬戸市せいれい町 27
[‡] 大阪大学大学院情報科学研究科
〒 560-8531 大阪府豊中市待兼山町 1-3

近年のソフトウェア開発では，開発対象であるソフトウェアの大規模化，複雑化が急激に進行しており，問題が生じた際の早期発見，早期対策の重要性がきわめて高いことが認知されつつある．一方で，多くの人間が共同開発を行うことを想定して，CVS などの構成管理ツールをもとに，全体的な開発の進捗を管理するケースが増えている．本論文では，CVS に登録されている各バージョンのソースコードに対して，コンポーネントランクの時間による変動をもとに現在の開発状況を推測する手法について，その手法の有効性及び実現可能性を評価する．コンポーネントランクとは，ソフトウェア部品の利用関係から，各部品の被利用度を評価したメトリクスである．評価では，実際のオープンソースプロジェクトに対して本手法を適用し，グラフ化した結果をもとにどのような情報が読み取れるかを評価する．本手法を用いることで,『(一見安定しているように見えるが）開発の終盤でもまだ多くの機能変更が行われている』など，より正確に実態を把握することができることが期待できる．

About the Feasibility of the Evaluation Method for a Development Process by Component Rank

Reishi Yokomori[†], Makoto Ichii[‡] and Katsuro Inoue[‡]

[†] Dept. of Information and Telecommunication Engineering, Nanzan University
27 Seirei-cho, Seto, Aichi, 489-0863, Japan
[‡] Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan

In the software development in the recent years, the size and complexity of the software is getting larger and larger. Every developer and manager knows the importance of the early detection, and rapid cure to avoid the developent risk, and uses a version cotroll system, such as CVS, for the manegement. In this paper, we evaluate the feasibility of the evaluation method for a development process by using component rank. Component rank is a metrics based on the use-relation between components, and we draw a graph which represents the variation with time of component rank. By applying to an open source project, we discuss what kind of information we can accuire from the graph.

1 はじめに

近年のソフトウェア開発では，開発対象であるソフトウェアの大規模化，複雑化が急激に進行しており，それに伴う開発の遅延が問題になっている．現在の開発環境では，多くの人間が共同開発を行うことを想定して，CVS などの構成管理ツールをもとに全体的な開発の進捗を管理するケースが増えている．開発者が自分の担当する部分のソースコードを進捗にあわせて登録することで，開発者および管理者は，自分たちのプロジェクトの現状をソースコードとして把握することが出来る．

しかし，多人数が参加するような開発環境下では，進捗管理を適正に行うことが難しい．問題が生じてからそれが顕在化するまでに時間がかかる場合も多く，問題が発覚したときにはすでに手遅れである場合も多い．現状を正しく認識するために，問題が生じた際には早期に発見し，早期に対策を行うことの重要性が認知されつつある．問題の早期発見のためには，トヨタ式カイゼン方法の一つである「視える化」のように，問題点が常に「視える」ようにしておく工夫が必要である [1]．プロセス改善のプラクティスをまとめた CMM[2]/CMMI[3] においても高いレベルでは，開発中に得られるデータの分析をもとにしたソフトウェアプロセスの改善が要求される．

このため，ソフトウェア開発においても，常に現状を把握できるようにモニタリングを支援する仕組みが重要である．また，モニタリングした情報は，なるべく多くの関係者が共有できるようにしておくことで，対策もすばやく行うことができると考えられる．

Ease プロジェクトが提案している EPM[4, 5] は，CVS，メーリングリスト管理システム，バグ追跡情報管理システムの 3 種類のデータをもとに，開発者や管理者にとって有益と思われる情報をグラフ化し，開発プロセスの管理に役立てるシステムである．EPM は定量的な開発データを低いコストで収集し，開発プロジェクトのリアルタイムでの管理を支援する．また，収集されるデータは人為的操作がされにくく，一貫性のあるデータを提供できる．これにより分析結果の共有化を容易にし，議論の場で現状についての認識を統一するのに有効であると考えられる．

しかし，CVS から収集したソースコードを直接理解するのは多くの労力が必要であるため，収集されたデータを何らかの観点から定量化する必要がある．現在の EPM では，LOC などのメトリクスをもとに，現状の共有理解に役立つ情報を提供している．しかし，LOC だけで実際の現状を理解するのは実際には困難であるため，実利用においては様々なメトリクスを利用して総合的に評価を行うことが必要であると考える．

そこで本論文では，CVS から収集した各バージョンのソースコードに対して，ソースコード間の利用関係の時間的な推移からどのような情報が得られるかを検証する．具体的には，CVS に登録されている各バージョンのソースコードに対して，コンポーネントランクの時間による変動をもとに開発状況を推測する手法について，実際のオープンソースプロジェクトに適用し，グラフ化した結果をもとにどのような情報が読み取れるかを評価する．

コンポーネントランク [6, 8] とは，ソフトウェア部品間の利用関係から，各部品の被利用度を評価したメトリクスである．よく利用される部品や重要な部品のコンポーネントランクは高く評価され，その部品集合において重要な役割を示していると考えられる．CVS における前後のバージョン間との比較を行うことで，それぞれのバージョン内における利用関係が安定しているかどうかを判定することができる．このような指標を用いることで，『(一見安定しているように見えるが) 開発の終盤でもまだ多くの機能変更が行われている』ことを検出することができるようになるなど，より正確に実態を把握できることが期待できる．

以下，2 章では，コンポーネントランクの説明を行い，EPM について説明する．3 章では，提案手法について説明する．さらに 4 章では，提案手法を実際のオープンソフト開発プロジェクトに適用し，結果として得られたグラフをもとに考察を行う．最後に，5 章でまとめと今後の課題について説明する．

2 背景

本節では，研究の背景としてコンポーネントランクの説明を行ったのちに，EPM について説明する．

2.1 コンポーネントランク

コンポーネントランクとは，ソフトウェア部品間の利用関係から得られる各部品の被利用度に基づいて，部品集合内でのその部品の順位を表したメトリクスである [6, 8]．以下では，ソフトウェア部品間の利用関係をモデル化した部品グラフを説明した後に，評価値の計算方法，現在のコンポーネントランクの利用方法について説明する．

2.1.1 ソフトウェア部品と部品グラフ

一般にソフトウェア部品 (Software Comopnent) とは再利用できるように設計された部品とされている [9, 10]．ここではより一般的に，ソースコードファイルやバイナリファイル，ドキュメントなどの

種類を問わず，開発者が再利用を行う単位をソフトウェア部品，あるいは単に部品と呼ぶ．これらの部品間には互いに利用する，利用されるという利用関係が存在する．本論文では各部品を頂点，部品間の利用関係を利用する側からされる側への有向辺として，グラフ上に表現する．以下，この部品間の利用関係を表現したグラフを**部品グラフ** *(Comopnent Graph)* と呼ぶ．以下では，V を部品（頂点）の集合，E を有向辺の集合として，$G = (V, E)$ と表現する．

図 1 は二つのソフトウェアシステム X と Y を部品グラフ化したものである．X は A から E の 5 つ，Y は F から I の 4 つの部品で構成されており，部品 C は 部品 A および B を利用し，部品 D および E は 部品 C を利用している．同様に部品 H と I は 部品 G を利用し, 部品 G と F はお互いに利用しあっている．

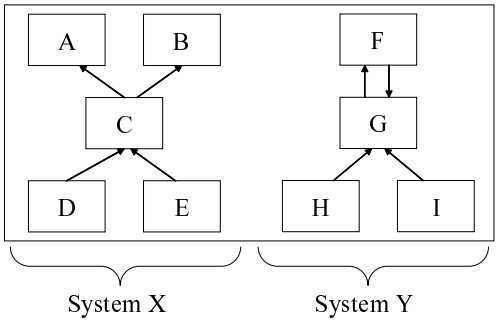

図 1: 部品グラフの例

2.1.2 評価値の計算

部品グラフ $G = (V, E)$ 上の個々の辺および頂点に対して重みを計算し，対応する頂点の重みをもとに各部品の被利用度を表す評価値を計算する．計算においては，部品グラフ G 上の各頂点 v の重みの和，辺の重みを次のように定義する．

定義：頂点の重みの和

部品グラフ G 上の各頂点 v は $0 \leq w(v) \leq 1$ の重みを持ち，G の頂点の重みの総和は，1 とする．

$$\sum_{v \in G} w(v) = 1$$

定義：辺の重み

頂点 v_i から v_j への辺 e_{ij} に関する辺の重み $w'(e_{ij})$ を次のように定義する．

$$w'(e_{ij}) = d_{ij} \times w(v_i)$$

図 2 (a) はこの定義を図示したものである．d_{ij} は配分率とよび，$0 \leq d_{ij} \leq 1$ かつ $\sum_i d_{ij} = 1$ を満たす値とする．頂点 v_i から v_j へ利用関係が存在しない場合，ここでは $d_{ij} = 0$ とする．

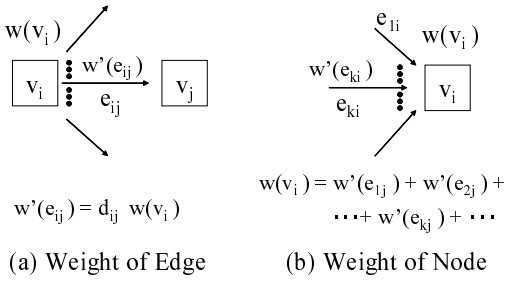

図 2: 重みの定義

このようにして定められた辺の重みをもとに，有向辺の終点となる頂点の重みを次のように決定する．図 2 (b) は次の定義を図示したものである．

定義：頂点の重み

IN(v_i) を v_i を終点とする有向辺の集合とする．この時，頂点 v_i の重みは v_i が終点となる有向辺 e_{ki} の重みの総和とする．つまり，

$$w(v_i) = \sum_{e_{ki} \in \text{IN}(v_i)} w'(e_{ki})$$

これらの重みの定義に基づいて，頂点 $w(v_i)$ に対して次の方程式が生成できる．

$$w(v_i) = \sum_{e_{ki} \in \text{IN}(v_i)} d_{ki} \times w(v_k)$$

各頂点に関してこの方程式を立てる事で，$n(= |V|)$ 個の連立方程式が生成できる．また，各頂点の重みをベクトル W として次のように表現し，

$$W = \begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix}$$

配分率を次の行列 D として表現する事で，

$$D = \begin{pmatrix} d_{11} & d_{12} & \ldots & d_{1n} \\ d_{21} & d_{22} & \ldots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \ldots & d_{nn} \end{pmatrix}$$

$n(= |V|)$ 個の連立方程式を次のように表す事ができる．ここで 行列 D^t は D の転置行列を表す．

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

このWを求めることで，各頂点に対応する部品の評価値を計算することができる．この場合，累乗法 (power method) を用いて適当な初期ベクトルに対して繰り返し D^t を掛ける事で，Wに関する近似解を容易に求める事ができる．

図3は，与えられたグラフにおいて各頂点の重み（重要度）を計算した結果である．v_1 は2つの有向辺の始点で，v_1 の重み0.4は二つの辺に0.2ずつ等分されている (つまり，$d_{12} = d_{13}$ = 0.5)．また，v_3 は2つの有向辺の終点で，それぞれの辺が0.2の重みを持つため，v_3 の重みは0.4であることがわかる．

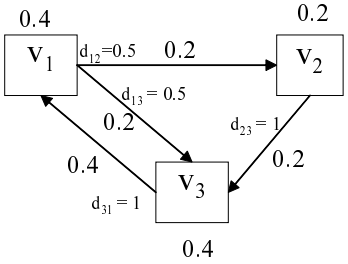

図 3: 重みの計算例

この手法は，開発者が部品をどのように参照するかをマルコフ連鎖 [11] でモデル化している．開発者はある部品を参照した後に，辺に沿って利用関係のある部品の一つを参照すると仮定している．参照を十分長い間時間繰り返した場合の，開発者によって各部品が参照されている確率が対応する各頂点の重みに相当する．つまり，重みが高い部品ほど開発者によって頻繁に参照されることになる．この値を評価値とすることで，ただ単に利用数が多い部品だけでなく，利用数が多い部品が利用している部品も重要であると評価することが出来る．実際の手法では，既約である事を保証するための補正や，コピーされた部品の存在を考慮した部品群グラフの利用などを行っている．詳細は [8] に記述されている．

2.1.3 コンポーネントランクの利用

我々の研究チームでは，このコンポーネントランクをソフトウェア部品の検索システムであるSPARS-Jにおける順位付けの指標として用いている [6, 7]．大きな部品集合の中での利用実績をコンポーネントランクとして計算し，それに基づいて順位付けを行うことで，よく利用されていて利用例の多い部品を効率的に検索できる．[6, 7] における評価では，検索結果全体の順位はユーザの想定する順位付けに近いという結果が得られており，重み付きのキーワードの出現回数と併用して順位付けを行うことで，さらによい結果が得られるということがわかっている．

さらに我々は，CR法を実際に多くのソフトウェア中に再利用されているという実績を定量的に評価した指標として，ソフトウェア部品の再利用性評価に利用できるとも考えている．再利用は，一般に生産性と品質を改善し，結果としてコストを削減するといわれており，コストを削減する事ができた事例の報告も行われている [12][13]．これまでに個々のソフトウェア部品の再利用性を評価する方法がいくつか提案されている．例えば，Etzkornらは，レガシーソフトウェア中の部品に対して，様々なメトリクス値を計算し，それらの値を正規化して足し合わせることで，再利用性とすることを提案した [14]．また，山本らは，ソースコードが非開示なソフトウェア部品に対して，インタフェース部分の情報のみを用いて再利用性を評価する方法を提案している [15]．

これらの手法は全て，部品そのものから読み取れる特性のみを計算して再利用性を評価するもので，実際のプログラマによる主観的な再利用性の評価の結果と似ているという結果が得られている．しかし現実には，従来手法では再利用性が低いと評価されても，多くのシステムで再利用されている部品は数多く存在すると考えられ，コンポーネントランクを用いることで利用実績に関する評価を補完できると考えている．

2.2 EPM

ソフトウェア開発においては，通常，複数のメンバーがそれぞれの役割に応じて作業を分担し組織的に開発を行う．その中では，バグ報告書などを用いてテストの運用を行うなどの手法が一般化されており，メールをコミュニケーション手段として使うことが一般的になっている．その際，CVSなどの構成管理システムを利用してソフトウェアを開発することや，BugzillaやGNATSのようなフォールト情報の管理ツールなどが利用されることは珍しくない．

EPMは，これらの開発支援システムにおける履歴データを自動的に収集し，分析・グラフ化を行うシステムである．EPMは，実際の開発現場では困難がつきまとう一貫性のある定量的なデータ収集を支援し，プロジェクト管理を目的としたリアルタイムでのデータの分析・利用を容易にする．

図4は，EPMの概要図である．EPMは，Linux上で動作するシステムで，データ収集，データ変換，データ蓄積，データ分析・グラフ化の4つの基本機能から構成されている．データ収集部では，構成管理，メーリングリスト管理，障害管理など，ソフトウェア開発においてよく利用される開発支援

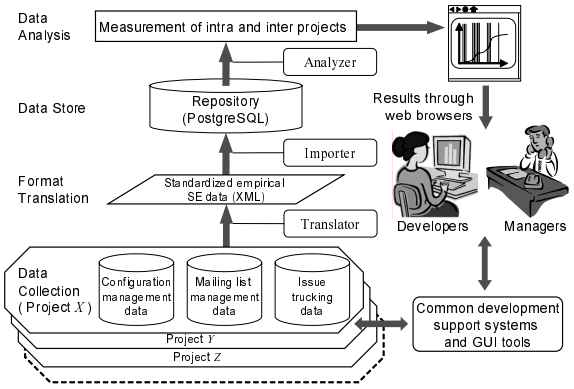

図 4: Empirical Project Monitor の概要図

システムからデータを収集する．収集されるデータはツールを利用した際のログ情報で構成されるため，管理者・開発者がデータ収集のために特別な作業を行う必要はない．データ変換部では，Ruby スクリプトによって，XML 形式のデータに変換し，データ蓄積部では，PostgreSQL データベースに格納する．分析を行う際には，このデータベースが逐次必要な情報を提供する．

データ分析・可視化部では，格納されたプロセスデータをユーザ (管理者・開発者) の要求に応じて分析し，結果を提示する．これらのデータ分析は Java サーブレットにより行われ，結果をグラフや表としてウェブブラウザから閲覧することができると共に，CVS 内のイベントやメール情報の集計表なども参照できる．

図 5 は表示されるデータの例で，このグラフでは『ソースコードの規模推移』を表し，ソースコード規模の推移 (図中，黒の折線) を LOC で示している．図中，灰色の垂線は開発者が CVS リポジトリを更新した (チェックインした) 時期を表しており，リポジトリ中のファイルに対する追加/修正/削除等の操作が行われたことを示している．グラフの規模推移と更新時期との関係から，開発が停滞しているのか，開発やテストによる更新が活発に行われているのかを明確に確認することができる．

このように，EPM は定量的な開発データを低いコストで収集し，開発プロジェクトのリアルタイムでの管理を支援する．また，収集されるデータは人為的操作がされにくく，一貫性のあるデータを提供できる．例は単一プロジェクトに対する分析結果であるが，過去のプロジェクトにおける情報を同時に表示することも可能である．このようなグラフを用いて説明を行うことで，プロジェクト管理者にとってリアルタイムで開発状況の把握をやりやすいということだけではなく，開発者にとっても定量的データを用いて具体的に開発状況

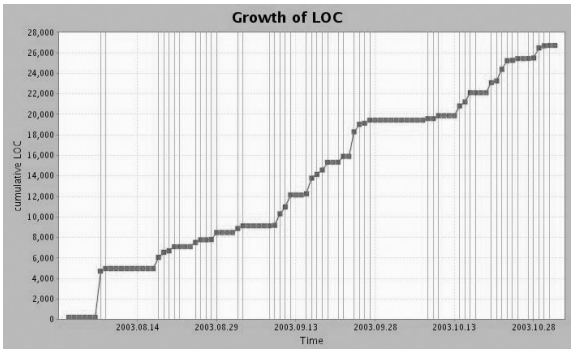

図 5: ソースコードの規模推移

が説明できるという点で有益である．

3 コンポーネントランクの時間変化に基づく評価手法

前節で紹介した EPM では，CVS から収集したソースコードを LOC を用いて定量化を行い，現状の共有理解に役立つ情報を提供している．CVS から収集したソースコードを直接理解するのは多くの労力が必要であるため，収集されたデータを何らかの観点から定量化する必要がある．現在の EPM では，LOC などのメトリクスをもとに，現状の共有理解に役立つ情報を提供している．

しかし，LOC だけで実際の現状を理解するのは実際には困難である．ただ単にソースコードの LOC が変化しない場合であっても，LOC だけでは何も手が加えられていないのか，大きな内部変更が行われた末での現状維持なのかを判別することはできない．また，削除追加の情報だけでは，実際に内部でどのような変更が行われたかを直感的に理解するのは難しい．そのため，実利用においては様々なメトリクスを利用して総合的に評価を行うことが必要であると考える．

本論文では，CVS 中に存在する各バージョンに対してコンポーネントランクを計算し，その時間による変動をもとに，現在の開発状況を推測する手法を提案する．コンポーネントランクは，各部品の被利用度を表したもので，よく利用される部品だけでなく，重要な部品から利用される部品も評価される．

開発対象のソフトウェアに機能が追加される場合，機能が削除される場合，ライブラリの変更などがあった場合などでは，部品間の利用関係が大きく変動するため，それぞれの部品におけるコンポーネントランクが大きく変動すると推測できる．また一方で，細かいデバッグや修正などの場合は，利用関係はほとんど変化しない．そのため，実際に内部で何が起こっているかを推測するための情

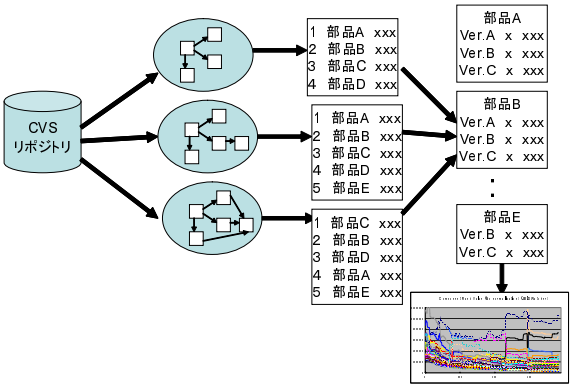

図 6: 提案手法の流れ

報として有益であると我々は考える．コンポーネントランクの変化の度合いが，開発の安定度を表す指標となり，『（一見安定しているように見えるが）開発の終盤でもまだ多くの機能変更が行われている』ことを検出することができるようになるなど，より正確に実態を把握することができることが期待できる．

3.1 提案手法の手順

提案手法の手順は次のとおりである．図 6 に提案手法の流れを示す．

(1) CVS リポジトリの各バージョンのソースコードを取得し，それぞれの部品集合に対してコンポーネントランク（およびその評価値）を計算する．

(2) その後，部品毎にその部品が存在するバージョンとその際のコンポーネントランクを抽出する．

(3) 抽出されたコンポーネントランクの推移をもとに，各部品の利用関係の変動の度合いなどを評価する．

4 評価実験

本節では，提案手法の有用性を評価するために，実際のオープンソースプロジェクトに対して適用を行い，グラフ化した結果をもとにどのような情報が読み取れるかを評価する．

4.1 適用対象

本実験での評価対象として，JBIDWatcher の CVS 開発履歴データを利用した．JBIDWatcher は eBay などを対象とした，ネットオークションの監視ツールで，Java で実装されている．実験においては 2000 年 6 月 1 日から 2003 年 11 月 27 日までの 383 リビジョンに対して，各リビジョンにおけるソースコードを取得した．このソースコード群をそれぞれ SPARS-J に登録し，それぞれコンポーネントランクを計算している．各開発リビジョンにおける

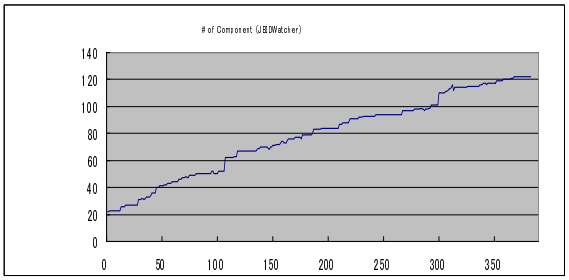

図 7: 部品数の推移

部品数を図 7 に示す．図 7 からは開発の進行によって，部品数が増加していることがわかる．

4.2 結果とその考察

評価値の推移

前述の JBIDWatcher に対して，そのコンポーネントランクと評価値をそれぞれ計算した．適用結果における評価値の推移を図 8 に示す．評価値の総和が一定であるため，部品数の増加により，一つあたりの評価値は下がっていることがわかる．上位に存在する部品は，期間の全般を通じて，エラー処理用のクラスであったり，値などを設定するためのクラス，JConfig 利用のためのクラスや，Base64 フォーマットからの入力をサポートするためのクラスなど，ソフトウェア内部で機能的に中心的な役割を果たす部品であった．

特徴的な値の動き方をする部品としては，変動が他の部品と一致している部品が確認できた．これらの部品間には密接な利用関係があり，どちらかに大きく依存している．また，変動の一致が受け継がれた部品も存在し，機能が別部品に吸収されたことが推測できる．また，評価値の変化パターンとしては，ほとんどの部品は単調減少で全体の順位としてはあまり変わらないのに対し，最初は評価値が低いが，直後に急上昇する部品も存在した．これは，部品が最初は限定された部品を使ってテストされた後に，それがソフトウェア全体で使われたことを示している．

コンポーネントランクの推移

次に，各部品のコンポーネントランク X を次の評価関数 $Normal(X)$ で 0 から 1 の値に正規化した．N はそのリビジョンでの部品数を表し，評価関数の値が 1 に近いほど順位が高いことを表す．

$$Normal(X) = \frac{N - (X - 1)}{N}$$

この正規化後のコンポーネントランクの推移を図 9 に示す．前半は毎回の順位の変動が大きいが，後半になるにつれて順位の変動が起こらない場合

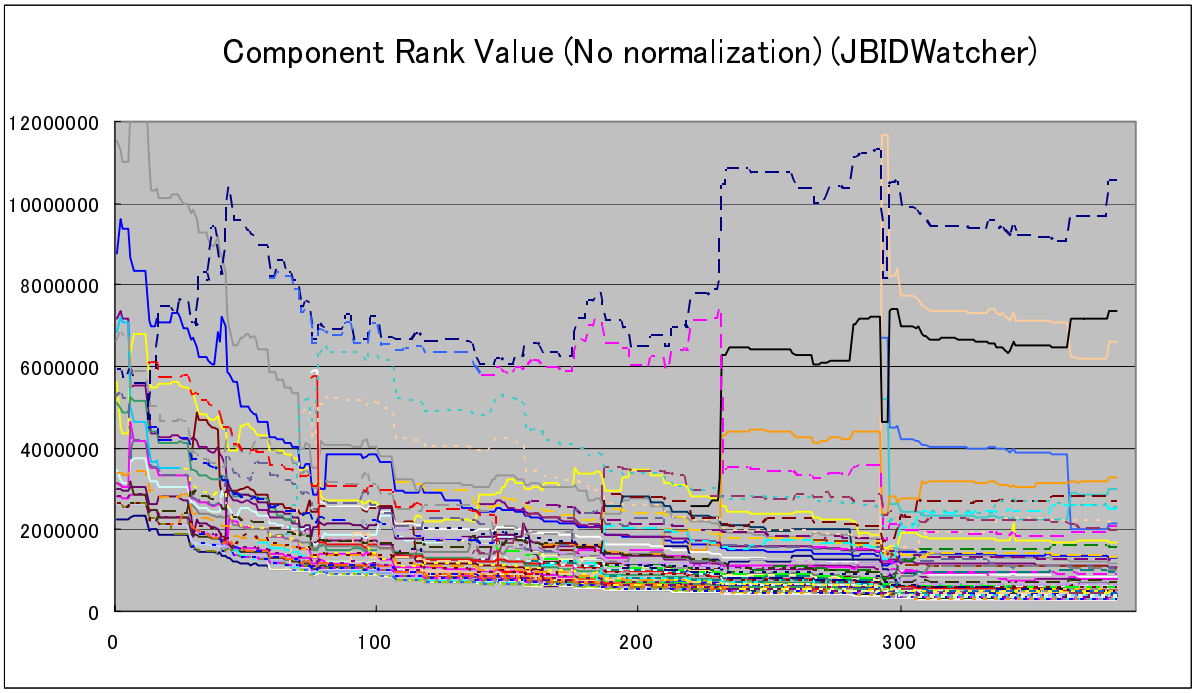

図 8: 評価値の推移

も多くなり，利用関係が次第に安定していくことがわかる．ただ，後半でも順位が変動する場合も時々発生している場合があり，その際に機能追加が行われていることが推測できる．

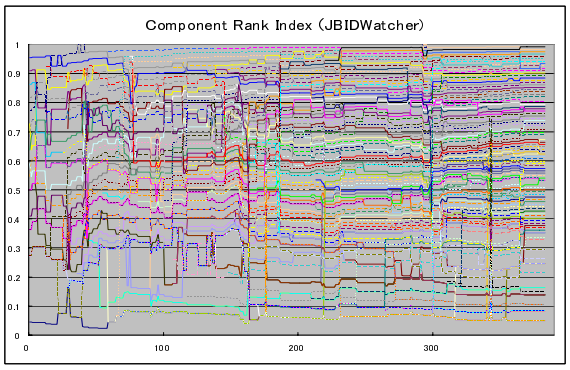

図 9: コンポーネントランク (正規化) の推移

平均変動率の推移

最後に，各部品について，前バージョンからの変動率をそれぞれ求めた上で，その平均を各バージョンにおける利用関係の変動度として計算した．この変動度をグラフ化し，図 10 に示す．グラフ中の移動平均は右肩下がりになっており，部品間の利用関係が安定していく様子を示している．一方で，開発の後期でも大きな変動が時々発生しており，機能追加などが行われたことが推測できる．

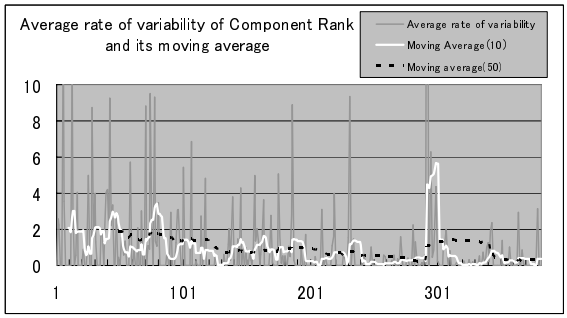

図 10: CR の平均変動度の推移

この変動度を開発プロセスの性質と組み合わせることで，開発状況を直感的に把握できる手助けになると我々は考えている．例えば，一般的なウォーターフォール型の開発において開発の終盤において変動度が高い値を保持し続けることは，作り直しが発生していたり，機能などの実装がまだ完了していないため，利用関係が変化していると推測できる．このままでは十分なテストがなされないまま出荷されるという状況を引き起こしかねない．この指標をグラフ化することで，危険な兆候をグラフから直感的に理解できることが期待できる．また，オープンソース開発の場合は，逆に定期的に変動度が高い変更が行われていることが開発が活発に行われていることを示すと考えられる．

これらの傾向は，LOC だけで判別できる場合も

あれば，そうでない場合も多い．LOC などのメトリクスと組み合わせてグラフ化することで，より正確に開発状況を直感的に把握することができると考えられる．

5 まとめ

本研究では，CVS に登録されている各バージョンのソースコードに対して，コンポーネントランクの時間による変動をもとに開発状況を推測する手法を提案した．提案手法を実際のオープンソースプロジェクトに適用し，その結果を表すグラフからは，部品の利用関係の推移を読み取ることができ，時間とともに利用関係が安定していることがわかった．これらの情報をグラフ化することで，現状の直感的な把握がよりやりやすくなることが期待できる．

今後の課題としては，他の開発プロジェクトへの適用による更なる評価や，分析システムの実現および，EPM 上での実利用を視野に入れた実現などがあげられる．

謝辞

この研究テーマは,「2005 年度南山大学パッヘ研究奨励金」(Pache Research Subsidy) I-A-2 の支援を受けており，本奨励金による研究成果である．

参考文献

[1] 若松義人, 近藤哲夫, “ トヨタ式改善力, ” ダイヤモンド社, 東京, 2003.

[2] M. C. Paulk, B. Curtis, M. B. Chrissis and C. V. Weber, “ Capability maturity model (version 1,1), ” IEEE Software, Vol.10, No.4, pp.18–27, 1993.

[3] CMMI product team:“ capability maturity model integration (CMMI) version 1.1, ” CMMI for systems engineering and software engineering, continuous representation, (CMMI-SE/SW, v1.1, continuous), CMU/SEI-2002-TR001, 2002.

[4] 大平 雅雄, 横森 励士, 阪井 誠, 岩村 聡, 小野 英治, 新海 平, 横川 智教: ”ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム”, 電子情報通信学会論文誌 D-I, Vol.J88-D-I, No.2, pp.228–239, 2005.

[5] 大平 雅雄, 横森 励士, 阪井 誠, 松本 健一, 井上 克郎, 鳥居 宏次: ”Empirical Project Monitor: プロセス改善支援を目的とした定量的開発データの自動収集・分析システムの試作”, 電子情報通信学会技術報告, Vol.103, No.708, SS2003-48, pp.13–18, 2004.

[6] Katsuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, Shinji Kusumoto: ”Ranking Significance of Software Components Based on Use Relations”, IEEE Trans. Software Engineering, Vol. 31, No. 3, pp213-225, 2005.

[7] 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, VolJ87-D-I, No.12, pp1060–1068, 2004.

[8] 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.

[9] 青山幹雄, 中所武司, 向山博: コンポーネントウェア, 共立出版, 1998.

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

[11] G. Blom, L. Holst and D. Sandell, 森真 [訳]: “ 確率問題ゼミ”, シュプリンガー・フェアラーク東京, 1995.

[12] V. R. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page and S. Waligora: “The software engineering laboratory - an operational software experience”, *Proc. of ICSE14*, pp. 370-381,1992.

[13] S. Isoda: “Experience report on a software reuse project: Its structure, activities, and statistical results”, *Proc. of ICSE14*, pp.320-326,1992.

[14] L. H. Etzkorn, W. E. Huges Jr., C. G. Davis: “Automated reusability quality analysis of OO legacy software”, *Information and Software Technology*, Vol. 43, Issue 5, pp. 295-308, 2001.

[15] 山本浩数, 鷲崎弘宜, 深澤良彰: “再利用特性に基づくコンポーネントメトリクスの提案と検証”, ソフトウェア工学の基礎ワークショップ (FOSE2001), 2001.