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
SS2007-71(2008-3)

メトリクス値の変遷に基づくソフトウェアの特性分析手法の提案と実装

村尾 憲治† 肥後 芳樹† 井上 克郎†

† 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番3号
E-mail: †{k-murao,higo,inoue}@ist.osaka-u.ac.jp

あらまし 本稿では，ソースコードにおけるソフトウェアメトリクスの値の変遷に基づき，ソフトウェアの特性を分析する手法の提案を行う．バージョン管理システムのリポジトリに蓄積されたソフトウェアの開発履歴に関する情報を用いてメトリクス値の変遷を分析することにより，注目すべきモジュールやメトリクス，変更など，ソフトウェアの様々な特性に関する情報を得ることができる．本手法によって得られる情報は，効率的なソフトウェアの開発や保守に役立つと期待される．実際に本手法を実装したツールをオープンソースソフトウェアに対して適用し，期待した通り有益なソフトウェアの特性が分析できることを確認した．

キーワード バージョン管理システム，ソフトウェアメトリクス

Software Characteristic Analysis Based on History of Software Metrics

Kenji MURAO†, Yoshiki HIGO†, and Katsuro INOUE†

† Graduate School of Information Science and Technology, Osaka University
1-3, Machikaneyama–cho, Toyonaka–shi, Osaka, 560–8351 Japan
E-mail: †{k-murao,higo,inoue}@ist.osaka-u.ac.jp

Abstract This paper proposes a method for software characteristic analysis based on software metrics transitions. Software metrics transitions can be measured from development historical information archived by version control system, and they can be used for identifying peculiar modules, metrics, and changes of the software system. Identification of those elements is useful for efficient software development and maintenance. We have implemented a software tool based on the proposal, and conducted case studies with open source software systems. The results of the case studies show that the proposal is helpful.

Key words Version Control System, Software Metrics

1. まえがき

ソフトウェアに関する深い知識は，その開発や保守を行うにあたり有益である．例えば，問題が発生しやすいモジュールを知っていればそのモジュールに労力を注ぐことで効率的な開発や保守が行え，またソフトウェアがどういった過程を経て開発されてきたかを知っていれば今後の開発の見通しも立てやすい．しかし，開発者がこのような知識を新たに得ようと思い，自力で対象ソフトウェアについて調査を行えば，ソフトウェアの規模に応じて膨大な時間を費やすことになる．そこで，開発者にソフトウェアに関する知識を提供するため，数多くの研究がなされている．

ソフトウェアの開発履歴を基に知識を提供する研究がある．多くのものはCVS [1] や Subversion [2] などのバージョン管理システムを情報源としている．バージョン管理システムは本来ソフトウェアの開発を効率よく管理するために用いられる．管理されるソフトウェアの開発履歴に関する情報は，バージョン管理システムのリポジトリというアーカイブに蓄積されている．このリポジトリに蓄積された開発履歴を閲覧することによって，ソフトウェアの開発過程についてより深い理解が得られることが知られている [3].

一方，ソフトウェアのソースコードそのものを基に知識を提供する研究がある．これに関しては，CKメトリクス [4] に代表されるソフトウェアメトリクスが有名であろう．特に，CKメトリクスは多くの研究において問題の発生しやすいモジュールを特定するのに有効であると示されている [5].

本稿ではバージョン管理システムとソフトウェアメトリクスに基づいてソフトウェアの特性を分析する手法を提案する．具体的には，バージョン管理システムに蓄積された開発履歴情報を用いてソフトウェアメトリクスの値の変遷を求めることにより，変動度というソフトウェアの特性を分析する指標を導出する．この手法により様々な観点からソフトウェアの傾向や性質を知ることができる．

以降，2. 節では本稿における用語と変動度の計測手段につい

て述べ，3. 節では提案手法の説明を行う．4. 節では手法を実装したツールについて触れ，5. 節では手法を実装したツールを用いて行った適用事例について述べる．最後にまとめと今後の課題を 6. 節に記す．

2. 準　　備

本節では，本稿における用語の意味や手法で用いる変動度の計測手段を定義する．

2.1 用　　語

2.1.1 スナップショット

スナップショットとは，ある時点でのソフトウェアの状態におけるソースコードの集合である．ソフトウェアのソースコードに対して何らかの変更が加わると，その時点で今までのスナップショットとは異なる新しいスナップショットが生まれる．スナップショットは CVS [1] や Subversion [2] といったバージョン管理システムのリポジトリに蓄積される情報，すなわち変更の日時や変更を行った開発者名，変更の前後における差分などの情報から得ることができる．

2.1.2 変　動　度

変動度とは，それぞれのスナップショットを通じたメトリクス値の変動の激しさを表す指標である．変動度の計測手段については 2.2 節および 2.3 節で述べる．

2.1.3 ソフトウェアの特性

ソフトウェアの特性とは，対象のソフトウェアに存在する何らかの傾向や性質である．様々なものがソフトウェアの特性として挙げられる．例えば，ソースコードの規模や開発日数，開発者の人数，用いるプログラミング言語，コーディング規約や開発者の士気の高さなどもその言葉の意味に含めることができる．もちろん，最新の状態のソースコードのメトリクス値もソフトウェアの特性として挙げることができる．本手法は，特に以下の項目を明らかにすることを目的としている．

- メトリクス値の安定しないモジュール
- 対象ソフトウェアにおいて値の安定しないメトリクス
- 各変更がソフトウェアに与えた影響の程度
- 各開発者がソフトウェアに与えた影響の程度

これらの特性を知ることは，ソフトウェア理解や開発管理，開発者の評価など幅広い分野で役立つと期待される [6] [7].

2.2 変動度の計測手段

対象データのばらつきを表す指標として散布度，不確かさを表す指標としてエントロピー，変化を表す指標として距離などが挙げられる．これらは主として数学や統計学で用いられる指標である．本手法では，これらのばらつきや不確かさ，変化の指標を変動度の計測手段として用いる．以降，これらの指標を用いた変動度の計測手段について説明する．説明にあたり，以下の設定を用いる．

$MD = \{md_1, md_2, \cdots, md_x\}$：　対象ソフトウェアにおけるモジュールの集合を表す．ただし，x は総モジュール数である．

$MT = \{mt_1, mt_2, \cdots, mt_y\}$：　用いるメトリクスの集合を表す．ただし，$y$ は用いるメトリクスの数である．

$CT = \{ct_1, ct_2, \cdots, ct_z\}$：　対象ソフトウェアにおける変更時刻の集合を表す．ただし，z は変更時刻の総数である．添字の値が小さいほど古い変更の時刻を表し，ct_1 を最初の変更時刻，ct_z を最新の変更時刻とする．

v_{md_i,mt_j,ct_k}：　変更時刻 ct_k におけるモジュール md_i のメトリクス mt_j の値を表す．ただし，変更時刻 ct_k においてモジュール md_i が存在しない場合，$v_{md_i,mt_j,ct_k} = null$ とする．

2.2.1 エントロピー

Shannon の提唱した情報理論 [8] に依れば，エントロピーとは不確かさを表す指標である．メトリクス値の変動の激しさをメトリクス値の不確かさと捉えることによって，エントロピーを変動度の指標として用いることができる．**エントロピー** H は対象モジュールの各メトリクスに対して導出される．メトリクス値 $v_{md_i,mt_j,ct_1}, v_{md_i,mt_j,ct_2}, \cdots, v_{md_i,mt_j,ct_z}$ において値が $null$ でないものの数を z' とし，z' 個のそれぞれの値を $v'_1, v'_2, \cdots, v'_{z'}$ と表すことにする．さらにその z' 個の値のうち，異なる値の種類数を z'' とする．z'' 種類の値をそれぞれ $v''_1, v''_2, \cdots, v''_{z''}$ とおくと，エントロピー H_{md_i,mt_j} の計算式は以下のようになる．

$$H_{md_i,mt_j} = -\sum_{l=1}^{z''} p_l \log_2 p_l$$

ここで p_l は値 v'_l の出現頻度を表し，次の式で定義される．

$$p_l = \frac{\sum_{k=1}^{z'} \mathrm{equal}(v''_l, v'_k)}{z''} \quad (1 \leqq l \leqq z'')$$

$$\mathrm{equal}(\alpha, \beta) = \begin{cases} 1 & (\alpha = \beta \text{のとき}) \\ 0 & (\alpha \neq \beta \text{のとき}) \end{cases}$$

$z'' = z'$ となるとき，エントロピーは $H_{md_i,mt_j} = -\log_2 \frac{1}{z'}$ となり最大である．$z'' = 1$ となるとき，エントロピーは $H_{md_i,mt_j} = -\log_2 1 = 0$ となり最小である．

2.2.2 正規化エントロピー

エントロピー H の最大値は変更が行われた回数に依存するため，異なるソフトウェア間では正しく値を比較することが難しい．この問題を解決するため，新たに正規化エントロピーを定義する．**正規化エントロピー** H' はエントロピー H と同様に対象モジュールの各メトリクスに対して値が得られる．上記のエントロピー H で用いた設定に準じると，正規化エントロピー H'_{md_i,mt_j} の計算式は以下のようになる．

$$H'_{md_i,mt_j} = \frac{H_{md_i,mt_j}}{-\log_2 \frac{1}{z'}}$$

エントロピー H_{md_i,mt_j} をその最大値で除算するため，正規化エントロピーは必ず $0 \leqq H'_{md_i,mt_j} \leqq 1$ となる．

2.2.3 四分位偏差

統計分野で用いられる散布度は，データのばらつきの程度を表現するものである [9]．メトリクス値の変動の激しさをメトリクス値のばらつきの程度と捉えることで，変動度の指標として散布度を用いることができる．一般にメトリクス値が正規分布に従うとは言えないので，散布度としては四分位偏差を用いる．**四分位偏差** Q は対象モジュールの各メトリクスに対して値が得ら

れる．メトリクス値 $v_{md_i,mt_j,ct_1}, v_{md_i,mt_j,ct_2}, \cdots, v_{md_i,mt_j,ct_z}$ における第 1 四分位数を q_1，第 3 四分位数を q_3 とすると，四分位偏差 Q_{md_i,mt_j} の計算式は以下のようになる．

$$Q_{md_i,mt_j} = \frac{q_3 - q_1}{2}$$

2.2.4 四分位分散係数

四分位偏差 Q の値は絶対値であり，用いるメトリクスのスケール差に影響されてしまう．このため，異なるメトリクス間では正しく値を比較することが難しい．そこでこの問題への対処として四分位分散係数を用いる．**四分位分散係数** Q' は四分位偏差 Q と同様に対象モジュールの各メトリクスに対して値が得られる．メトリクス値 $v_{md_i,mt_j,ct_1}, v_{md_i,mt_j,ct_2}, \cdots, v_{md_i,mt_j,ct_z}$ における中位数を m とすると，四分位分散係数 Q'_{md_i,mt_j} の計算式は以下のようになる．

$$Q'_{md_i,mt_j} = \frac{Q_{md_i,mt_j}}{m}$$

ただし $m = 0$ の場合は，$Q_{md_i,mt_j} = 0$ ならば $Q'_{md_i,mt_j} = 0$，$Q_{md_i,mt_j} \neq 0$ ならば $Q'_{md_i,mt_j} = \infty$ とする．

2.2.5 ハミング距離

情報理論 [8] においてハミング距離は，等しい文字数を持つ 2 つの文字列の中で対応する位置にある異なった文字の個数を意味する．文字列をメトリクス値の列に置き換えることで，ハミング距離を 2 つの変更間における変動の激しさを示す指標として用いることができる．**ハミング距離** DH は対象モジュールの連続する 2 つの変更間について導出される．ハミング距離 DH_{md_i,ct_k} は以下の計算式で表される．

$$DH_{md_i,ct_k} = \sum_{j=1}^{y} \mathrm{different}(v_{md_i,mt_j,ct_{k-1}}, v_{md_i,mt_j,ct_k})$$

$$\mathrm{different}(\alpha,\beta) = \begin{cases} 1 & (\alpha \neq \beta \text{のとき}) \\ 0 & (\alpha = \beta \text{のとき}) \end{cases}$$

この計算式が示すように，ハミング距離 DH_{md_i,ct_k} は変更時刻 ct_{k-1} と ct_k 間において値が変化したメトリクスの数を表している．したがって計算式が成り立つ条件は $\forall k((1 < k \leqq z) \wedge (v_{md_i,mt_j,ct_{k-1}} \neq null) \wedge (v_{md_i,mt_j,ct_k} \neq null))$ となる．この条件が成り立たない場合は，$DH_{md_i,ct_k} = null$ とする．

2.2.6 ユークリッド距離

ハミング距離 DH は変更前と変更後でメトリクス値が変化したかどうかのみを考慮しており，値がどの程度変化したかということは反映されない．ハミング距離 DH に対し，**ユークリッド距離** DE はメトリクス値の変化の大きさを反映した変動度を表す．変動度としてのユークリッド距離 DE は，用いるメトリクスの数 y 次元ユークリッド空間上の距離を表す [10]．ハミング距離 DH と同様に，ユークリッド距離 DE は対象モジュールの連続する 2 つの変更間について導出される．ユークリッド距離 DE_{md_i,ct_k} は以下の計算式で表される．

$$DE_{md_i,ct_k} = \sqrt{\vec{V}^T_{md_i,ct_k} \cdot \vec{V}_{md_i,ct_k}}$$

$$\vec{V}_{md_i,ct_k} = \vec{v}_{md_i,ct_k} - \vec{v}_{md_i,ct_{k-1}}$$

$$\vec{v}_{md_i,ct_k} = [v_{md_i,mt_1,ct_k}, v_{md_i,mt_2,ct_k}, \cdots, v_{md_i,mt_y,ct_k}]^T$$

ハミング距離 DH と同様に，計算式が成り立つの条件は $\forall k((1 < k \leqq z) \wedge (v_{md_i,mt_j,ct_{k-1}} \neq null) \wedge (v_{md_i,mt_j,ct_k} \neq null))$ となる．この条件が成り立たない場合は，$DE_{md_i,ct_k} = null$ とする．

2.2.7 マハラノビス距離

ユークリッド距離を用いた変動度は，メトリクス間に相関がなく，かつスケール差もないと見なした状態での値となっている．しかし，実際にはメトリクス間に相関やスケール差の存在することが多い．例えば，異なる定義によりモジュールの結合度を求める 2 つのメトリクスがあった場合，目的を同じくするこの 2 つのメトリクスには強い正の相関関係があると考えられる．また，ソースコードの行数を表すメトリクスなどは変更によって絶対値が大きく変化することも珍しくないが，サブクラスの数を表すメトリクスなどは変更によって変化する絶対値があまり大きくなることは少ないと考えられる．そこで，相関やスケール差を考慮した値を求めることのできるマハラノビス距離も利用する．マハラノビス距離は多変量解析など統計学でよく用いられる距離の一種であり，情報分野においてもクラスタリング手法などにしばしば利用される [11][12][13]．ハミング距離 DH やユークリッド距離 DE と同様に，**マハラノビス距離** DM は対象モジュールの連続する 2 つの変更間について導出される．また，マハラノビス距離 DM はその計測に対象のソフトウェア全体でのメトリクスの共分散行列 Σ を必要とする．この共分散行列は $y \times y$ の正方行列である．ユークリッド距離 DE における設定を用いて，要素に $null$ を含まないベクトル $\vec{v}_{md_i,ct_k}$ の個数を n $(1 \leqq n \leqq (x \times z))$ とする．この n 個のベクトルをそれぞれ $\vec{v}'_1, \vec{v}'_2, \cdots, \vec{v}'_n$ とおくと，メトリクスの共分散行列 Σ は以下の式で表すことができる．

$$\Sigma = \frac{1}{n} \sum_{l=1}^{n} (\vec{v}'_l - \vec{\mu}) \cdot (\vec{v}'_l - \vec{\mu})^T$$

$$\vec{\mu} = \frac{1}{n} \sum_{l=1}^{n} \vec{v}'_l$$

この共分散行列 Σ を用いて，マハラノビス距離 DM_{md_i,ct_k} は以下の計算式で表される．

$$DM_{md_i,ct_k} = \sqrt{\vec{V}^T_{md_i,ct_k} \Sigma^{-1} \vec{V}_{md_i,ct_k}}$$

計算式が示すように，マハラノビス距離はメトリクスの共分散行列 Σ の逆行列を計算に用いる．したがって，共分散行列 Σ が特異行列（非正則行列）となる場合はマハラノビス距離の計測自体を行わないものとする．また，ハミング距離 DH やユークリッド距離 DE と同様に，上記の計算式が成り立つの条件は $\forall k((1 < k \leqq z) \wedge (v_{md_i,mt_j,ct_{k-1}} \neq null) \wedge (v_{md_i,mt_j,ct_k} \neq null))$ となる．この条件が成り立たない場合は，$DM_{md_i,ct_k} = null$ とする．

2.3 変動度の計測手段の拡張

2.2 節で述べた各計測手段によって導かれる変動度は，2 つの次元を持つ．例えばエントロピー H はモジュールとメトリ

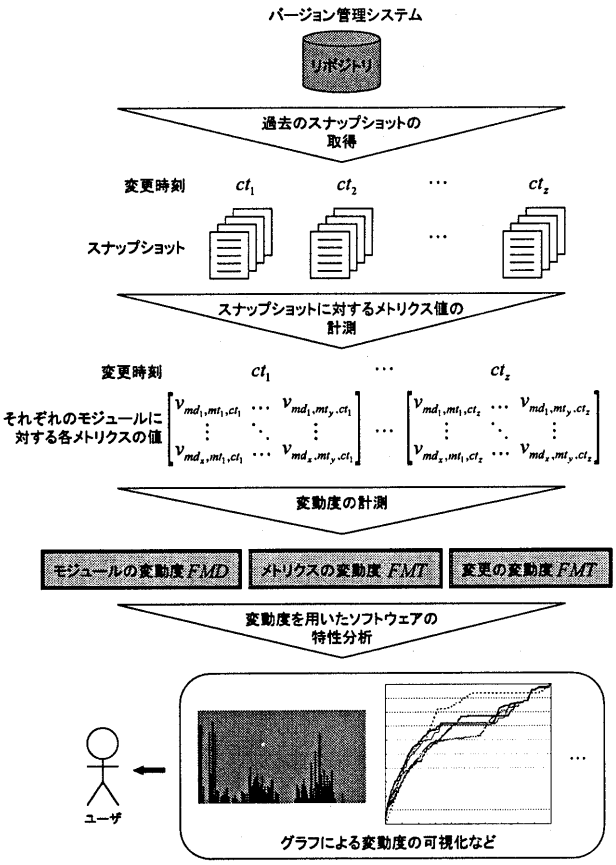

図 1　提案手法の概要

クスの 2 つの次元を持ち，ハミング距離 DH はモジュールと変更時刻の 2 つの次元を持つ．これらの変動度を用いて，以下に述べるモジュールの変動度，メトリクスの変動度，変更の変動度を導出できる．

2.3.1　モジュールの変動度

モジュールの変動度（fluctuation of module, FMD）は，それぞれのモジュールについて 1 つの値を持つ．この変動度を用いれば，対象のソフトウェアにおいて注目すべきモジュールを知ることができる．変動度の大きいモジュールはメトリクス値が変化しやすいので，他のモジュールに影響を与えやすい，あるいは逆に影響を受けやすい，または頻繁に変更が行われていると考えられる．この変動度の大きさに応じて力を割くモジュールを決定すれば，効率的な開発や保守が期待できる．モジュールの変動度 FMD の計算式は以下のようになる．

$$FMD(\omega)_{md_i} = \begin{cases} \sum_{j=1}^{y} \omega_{md_i,mt_j} & (\omega = H, H', Q') \\ \sum_{k=1}^{z} \omega_{md_i,ct_k} & (\omega = DH, DE, DM) \end{cases}$$

ただし，ω は H, H', Q', DH, DE, DM のいずれかである．四分位偏差 Q についても同様の計算は可能だが，用いるメトリクス間にスケール差が存在する場合，四分位偏差 Q を上記の計算式に当てはめるのは適切でないため，四分位偏差 Q は用いない．

2.3.2　メトリクスの変動度

メトリクスの変動度（fluctuation of metric, FMT）は，それぞれのメトリクスについて 1 つの値を持つ．この変動度を用いれば，対象のソフトウェアにおいてどのメトリクスの値が変化しやすく，どのメトリクスの値が変化しにくいか，というソフトウェアの特性を知ることができる．メトリクスの変動度 FMT の計算式は以下のようになる．

$$FMT(\omega)_{mt_j} = \sum_{i=1}^{x} \omega_{md_i,mt_j}$$

ただし，ω は H, H', Q, Q' のいずれかである．

2.3.3　変更の変動度

変更の変動度（fluctuation of change time, FCT）は，それぞれの変更時刻について 1 つの値を持つものである．この変動度を用いれば，対象のソフトウェアにおいていつどのような変更が行われたのか，というソフトウェアの特性を知ることができる．この特性はソフトウェアの開発履歴を理解するのに有効である．変更の変動度 FCT の計算式は以下のようになる．

$$FCT(\omega)_{ct_k} = \sum_{i=1}^{x} \omega_{md_i,ct_k}$$

ただし，ω は DH, DE, DM のいずれかである．

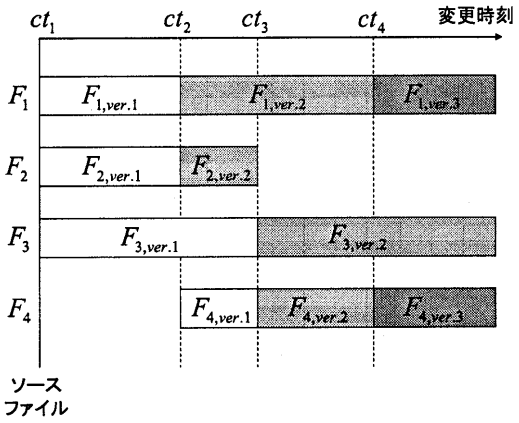

図 2　ソフトウェアの変更履歴の例

3. 提案手法

提案手法の概要を図 1 に示す．前提となるのは，対象のソフトウェアの開発にバージョン管理システムが利用されていることである．本手法は次の手順から構成される．

手順 1：過去のスナップショットの取得
手順 2：スナップショットに対するメトリクス値の計測
手順 3：変動度の計測
手順 4：変動度を用いたソフトウェアの特性分析

以下では，これらの手順についてそれぞれ説明する．

3.1 過去のスナップショットの取得

バージョン管理システムに保存された分析対象のソフトウェアの変更履歴情報を利用し，過去全てのソースコードのスナップショットを取得する．

図 2 はソフトウェアの変更履歴の例を示したものである．図 2 では，このソフトウェアに存在する 4 つのソースファイル F_1, F_2, F_3, F_4 が縦軸に配置されている．横軸は時刻を表し，時

刻 ct_1, ct_2, ct_3, ct_4 において変更が行われている．また，$F_{1,ver.1}$ や $F_{4,ver.3}$ などはソースファイルとそのバージョンを示している．図 2 に示すようなソフトウェアの変更履歴があったとき，それぞれの変更時刻におけるスナップショットは表 1 の通りになる．

3.2 スナップショットに対するメトリクス値の計測

全ての変更時に対応するスナップショットを対象にメトリクス値の計測を行う．これにより，任意のモジュールに対するメトリクス値の変遷が得られる．

用いるメトリクスはモジュールの単位や目的に合わせて選ぶ必要がある．例えば，モジュールの単位をクラスとしたいのであればクラスを対象としたメトリクスを，ファイルとしたいのであればファイルを対象としたメトリクスを用いる．あるいは，結合度に注目したいのであれば結合度を表すメトリクスを，凝集度に注目したいのであれば凝集度を表すメトリクスを用いる．後述の適用事例では，モジュールの単位をクラスとし，主に CK メトリクス [4] を用いる．

3.3 変動度の計測

メトリクス値の変遷を用い，2.3 節で述べたモジュールの変動度，メトリクスの変動度，変更の変動度をそれぞれ計測する．

3.4 変動度を用いたソフトウェアの特性分析

3.3 節で求めた変動度を用いてソフトウェアの特性分析を行う．ソフトウェアの特性分析は，変動度の値やそれをグラフなどの形式で可視化したものを用いてユーザが実行する．例えば，モジュールの変動度からどのモジュールのメトリクス値が変化しやすいのか判断したり，メトリクスの変動度から対象のソフトウェアにおいてどのメトリクスの値が変化しやすいのか把握したりする．

4. 実　　装

3. 節で述べた提案手法の手順 1〜3 を自動で行うツールを実装した．このツールは現在のところ，Java で開発されたソフトウェアを対象とし，バージョン管理システム CVS のリポジトリを用いてスナップショットを取得し，CK メトリクス [4] を計測し，それぞれの変動度を求めるものとなっている．CK メトリクスは対象とするモジュールの粒度がクラスであるので，本ツールにおいてもクラスをモジュールの単位として扱う．

本ツールは CVS のリポジトリを入力とし，2.3 節で述べたモジュールの変動度，メトリクスの変動度，変更の変動度をそれぞれ CSV 形式のファイルとして出力する．ユーザは，この出力された変動度をグラフなどの形で可視化し，ソフトウェアの特性を分析する．

表 1　図 2 に対応するスナップショット

変更時刻	スナップショット
ct_1	$(F_{1,ver.1}, F_{2,ver.1}, F_{3,ver.1})$
ct_2	$(F_{1,ver.2}, F_{2,ver.2}, F_{3,ver.1}, F_{4,ver.1})$
ct_3	$(F_{1,ver.2}, F_{3,ver.2}, F_{4,ver.2})$
ct_4	$(F_{1,ver.3}, F_{3,ver.2}, F_{4,ver.3})$

5. 適 用 事 例

本節では，4. 節で述べたツールを用い，実存するオープンソースソフトウェアに手法を適用した事例について述べる．複数のソフトウェアに対し手法を適用したが，紙面の都合上，ここでは FreeMind [14] というオープンソースソフトウェアに対して手法を適用した結果の内，モジュールの変動度についてのみ述べることにする．対象ソフトウェア FreeMind についての概要を表 2 に示す．提案手法を実装したツールの実行時間は CPU 1.86GHz，メモリ 2GB の Windows マシン上でおよそ 28 分であった．

2.3.1 節で述べたように，モジュールの変動度が高いクラスは一般に問題が発生しやすい状態にあると考えられる．したがって，モジュールの変動度が高いクラスに注意して開発を進めれば，効率的な開発が行えると期待できる．このことを確かめるため，変動度の高いモジュールとバグとの関係を調べる実験を行った．実験は次に示す手順で実行した．以降では，バグを修正するための変更を**バグ修正変更**，バグ修正変更の対象となったファイルの延べ数を**バグ修正数**と呼ぶ．

（1） FreeMind のスナップショットを前半と後半に分割する．

（2） 前半のスナップショットを用いてモジュールの変動度を計測する．このとき，内部クラスは除く．

（3） CVS リポジトリに含まれる開発者による変更時のメッセージログを用いて，後半のスナップショットにおけるバグ修正変更を特定し，各スナップショットにおけるバグ修正数を計測する．

（4） 変動度の高いクラスのランキングによる，バグ修正数に対する被覆率を求める．

FreeMind の開発においては，バグ修正変更の際のメッセージログには，その先頭に「Bug fix: 」と記載する習慣がある．したがってバグ修正変更を特定する際には，開発者による変更時のメッセージログに「bug」と「fix」の文字列（大文字小文字は区別しない）が含まれているかどうかを判定する．このようにして特定されたバグ修正数は 36 となった．

モジュールの変動度が高いクラスのランキングによる，バグ修正数に対する被覆率を図 3 に示す．図 3 を見れば，どのモジュールの変動度の指標を用いても，ランキングの上位 20%

表 2　FreeMind の概要

ソフトウェア名	FreeMind
種別	ダイアグラム生成ツール
開発言語	Java
開発者数	12
総スナップショット数 z	225
最初の変更時刻 ct_1	2000/08/01 19:56:09
最後の変更時刻 ct_z	2008/01/13 20:55:35
ct_1 におけるソースファイル数	67
ct_z におけるソースファイル数	221
ct_1 におけるソースコードの総行数	3,882
ct_z におけるソースコードの総行数	39,350

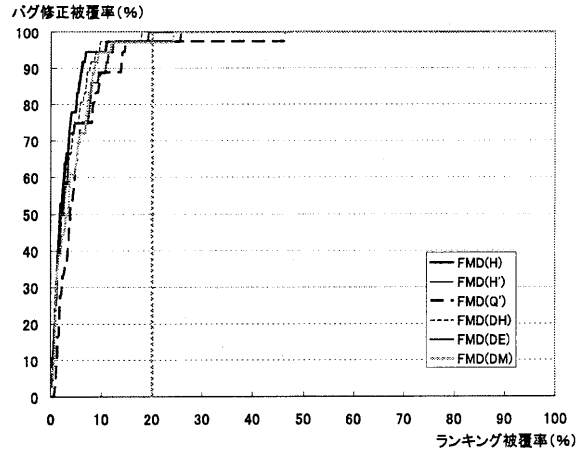

図3　モジュールの変動度のランキングによるバグ修正数の被覆率

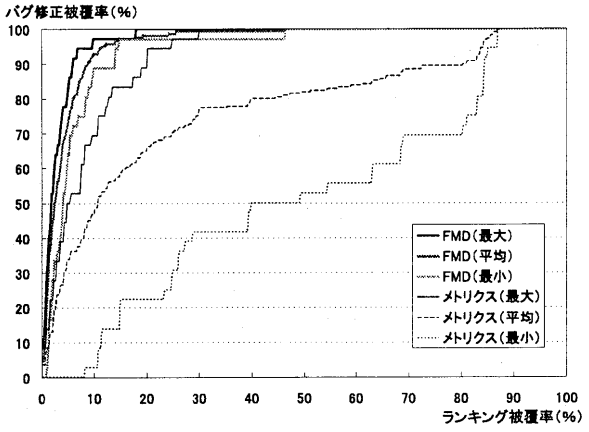

図4　モジュールの変動度と CK メトリクスの比較

(図3の点線で示された箇所) におよそ 97% ～100% もの後のバグ修正変更が含まれていることが分かる.

CK メトリクスはバグの予測に有効であるとの報告がなされている [5]. そこで, このモジュールの変動度の比較対象として, FreeMind の前半のスナップショットにおける最後のスナップショットに対する CK メトリクスの値をそれぞれ求め, 各 CK メトリクスの値のランキングによるバグ修正数の被覆率を求める. 図3で示した被覆率の平均値, 最大値, および最小値と, 各 CK メトリクスの値のランキングによるバグ修正数の被覆率の平均値, 最大値, および最小値を示したグラフを図4に示す. 図4から, 次のことが言える.

- モジュールの変動度のどの指標を用いても, 後のバグ修正の予測に極めて有効であり, また指標毎の差異もそれほど大きくはない.
- メトリクス値を用いた後のバグ修正の予測は有効であるが, 用いるメトリクスによる差異が極めて大きい.
- 後のバグ修正を最も有効に予測できるメトリクスを用いた場合でも, モジュールの変動度を用いた後のバグ修正の予測の方が一般に有効である.

以上のことより, ソフトウェアの開発においてモジュールの変動度が高いモジュールに力を割くことは, 効率的な開発の実現に繋がると大きく期待できる.

6. ま と め

本稿では, バージョン管理システムのリポジトリに蓄積された開発履歴情報を用いてメトリクス値の変遷を求め, モジュールの変動度, メトリクスの変動度, 変更の変動度を導出することによりソフトウェアの特性を分析する手法を提案した. さらに, 提案手法を実装し, 実存するソフトウェアに対して手法を適用したところ, 変動度から得られるソフトウェアの特性の有効性が確認された.

今後の課題として重要なのは, 手法を実装したツールの拡張である. 現状では制限が多く, 以下にあげるような機能が望まれる.

- C++や C# など, 他のプログラミング言語への対応
- Subversion など, 他のバージョン管理システムへの対応
- 他の様々なソフトウェアメトリクスへの対応
- 変動度を可視化する作業の自動化

謝辞　本研究は一部, 文部科学省「次世代 IT 基盤構築のための研究開発」(研究開発領域名：ソフトウェア構築状況の可視化技術の開発普及) の委託に基づいて行われた.

文　献

[1] CVS. http://ximbiot.com/cvs/wiki/.
[2] Subversion. http://subversion.tigris.org/.
[3] Peter H. Feiler. Configuration Management Models in Commercial Environments. CMU/SEI-91-TR-7 ESD-9-TR-7, 1991.
[4] Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object Oriented Design. In *IEEE Transactions on Software Engineering*, Vol.20, No.6, pp.476-493, 1994.
[5] Victor R. Basili, Lionel C. Briand, and Walcelio L. Melo. A Validation of Object-Oriented Design Metrics as Quality Indicators. In *IEEE Transactions on Software Engineering*, Vol.22, No.10, pp.751-761, 1996.
[6] Ahmed E. Hassan and Richard C. Holt. Studying the Chaos of Code Development. In *Proceedings of 10th Working Conference on Reverse Engineering (WCRE '03)*, pp.123-133, 2003.
[7] Nachiappan Nagappan, Thomas Ball, and Andreas Zeller. Mining Metrics to Predict Component Failures. In *Proceeding of the 28th International Conference on Software Engineering (ICSE '06)*, pp.452-461, 2006.
[8] Claud E. Shannon. The Mathematical Theory of Communication. In *Bell System Technical Journal*, Vol.27, pp.379-423 & 623-656, Jul & Oct 1948.
[9] 宮川 公男. 基本統計学. 有斐閣, 1999.
[10] 中村 幸四郎, 寺阪 英孝, 伊東 俊太郎, 池田 美恵 (翻訳). ユークリッド原論. 共立出版, 1996.
[11] Francoise Fessant, Patrice Aknin, Latifa Oukhellou, and Sophie Midenet. Comparison of Supervised Self-Organizing Maps Using Euclidian or Mahalanobis Distance in Classification Context. In *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks (IWANN '01): Connectionist Models of Neurons, Learning Processes and Artificial Intelligence-Part I*, pp.637-644, 2001.
[12] Camilo P. Tenorio, Francisco de A. T. de Carvalho, and Julio T. Pimentel. A Partitioning Fuzzy Clustering Algorithm for Symbolic Interval Data based on Adaptive Mahalanobis Distances. In *Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS '07)*, pp.174-179, 2007.
[13] 村上 正憲, 松島 正直, 山田 博章. Mahalanobis' 特徴平面による識別. 電子情報通信学会総合大会講演論文集, Vol.1995 情報・システム, No.2. p.213, 1995.
[14] FreeMind. http://freemind.sourceforge.net/wiki/.