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

信学技報
TECHNICAL REPORT OF IEICE.

ソフトウェアタグを用いた設計文書メトリクスからの低品質モジュールの予測

片山　真一† 　大蔵　君治† 　伏田　享平† 　川口　真司† 　名倉　正剛†∗
門田　暁人† 　飯田　元†

† 奈良先端科学技術大学院大学情報科学研究科
〒 630-0192 奈良県生駒市高山町 8916-5
∗ 現在，日立製作所
E-mail: †shinichi-k@is.naist.jp

あらまし　高品質なソフトウェアを作成するためには，早期段階で多くの不具合を含む可能性のある低品質モジュールを特定することが重要となる．そのために，線形判別分析，ロジスティック回帰分析，ニューラルネットワークなど様々なモデルが多数提案されている．しかし，従来手法では主にソースコードから計測されたメトリクスを用いていたため，予測のタイミングが実装工程完了後に限定されていた．本研究ではソフトウェアタグに含まれる設計文書メトリクスを用いて，低品質モジュールの予測を試みる．設計文書メトリクスを利用することで設計工程完了時に予測を行い，実装工程における品質改善活動に寄与することができる．重回帰分析・遺伝的プログラミングを適用した結果，ランダムな手法に比べて高い精度で低品質モジュールを特定することができた．

キーワード　低品質モジュール予測，ソフトウェアタグ，設計文書メトリクス，重回帰分析，遺伝的プログラミング

Low Quality Module Prediction from Design Documents Metrics using Software Tag

Shinichi KATAYAMA†, Kimiharu OHKURA†, Kyohei FUSHIDA†, Shinji KAWAGUCHI†,
Masataka NAGURA†∗, Akito MONDEN†, and Hajimu IIDA†

† Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192 Japan
∗ Presently with Hitachi, Ltd.
E-mail: †shinichi-k@is.naist.jp

Abstract　It is important to detect low quality modules which are likely to have many defects at an early development phase for creating high quality software. Hence, various prediction models are proposed such as linear discriminant analysis, logistic regression analysis, and neural networks. These methods, however, use metrics measured mainly from a source code, thus they can only be applied after implementation is completed. We attempt to predict low quality modules using design documents metrics included in software tags. Using design documents metrics, we can predict when design is completed, thus our method contribute to reliability enhancement activities at an implementation phase. We applied multiple regression analysis and genetic programming, and detected low quality modules with high accuracy compared to a random selection method.

Key words　Low Quality Module Prediction, Software Tag, Design Documents Metrics, Multiple Regression Analysis, Genetic Programming

1. はじめに

近年，ソフトウェアの社会的位置づけがますます重要性を帯びてくると共に，その品質を保証することが重要な目標となっている．高品質なソースコードの作成がプロダクトの品質向上に繋がるため，早期段階での多くの不具合を含む可能性のある低品質モジュールの特定が求められている．そのために，多数のモジュールから計測されたプロダクトメトリクス（プログラム行数やサイクロマティック数，変更行数など）を説明変数とし，モジュールの欠陥の有無を目的変数とする判別モデルが提案されており，線形判別分析，ロジスティック回帰分析，ニューラルネットワークなど様々な種類のモデルが存在する．

しかし，これらプロダクトメトリクスの多くはモジュールのソースコードを対象とするため，モジュールの実装完了段階まで適用できない．このようなプロダクトメトリクスを用いた低品質モジュールの予測は，テスト工程におけるテストの効率化やソフトウェアの信頼性向上には有効であるが，実装工程以前における早期の品質改善活動には寄与できない．また，従来の低品質モジュール予測では，モジュールの欠陥の有無でモジュールの品質を判断しているものが多く，モジュールの欠陥の多寡を考慮したものは少ない．設計工程のような開発の早期段階では，欠陥が多く含まれているのは自然であり，欠陥の有無ではなく相対的な多寡に基づいて品質改善活動のターゲットを定めることが現実的である．

そこで，本研究ではソフトウェアタグ [1] に含まれる設計文書メトリクスを用いて，相対的により多くの欠陥を含む可能性が高い低品質モジュールの予測を試みた結果について報告する．

ソフトウェアタグはソフトウェア開発プロセスや成果物などのソフトウェア開発に関するデータを，ユーザとベンダ間で共有するための基盤技術である．ソフトウェアタグは従来の実装工程で計測されたプロダクトメトリクスに加えて，要件定義，設計工程で計測されるメトリクスも含んでいる．このうち設計文書から計測されるメトリクス（設計書の規模，変更量など）をここでは設計文書メトリクスと呼ぶ．

この設計文書メトリクスを用いることで，設計工程完了時に低品質モジュールを生成する可能性の高い設計書を特定し，予測結果を実装工程における品質改善活動に役立てることが可能となる．実験では，複数の企業が合同で行った実開発プロジェクトから収集されたソフトウェアタグを使用し，重回帰分析，遺伝的プログラミングにより低品質モジュールの予測を行った．

以降，2. 章ではソフトウェアタグについて述べる．3. 章で予測手法について説明し，4. 章において，本研究で用いたデータの詳細について説明する．5. 章で実験方法を述べたあと，6. 章で結果に対する考察を行う．7. 章で関連研究との比較を行い，8. 章で本研究のまとめを述べる．

2. ソフトウェアタグ

StagE (Software Traceability and Accountability of Global Software Engineering) プロジェクトでは，ソフトウェア開発プロセスや成果物などのソフトウェア開発に関するデータを，ソフトウェア発注者と受注者間で共有するための基盤技術として「ソフトウェアタグ」を提案している．ソフトウェアタグを共有することで，発注者がプロジェクトの進捗状況の把握や納品時の事後分析などを行うことができ，従来受注者に開発を任せきりであった発注者のプロジェクトへの参画を促す効果が期待されている．現在ソフトウェアタグは Ver1.0 として規格化されている [2]．ソフトウェアタグに含まれるデータはタグ項目としてそれぞれ定義され，開発人数や開発期間などの開発プロジェクトの管理データと，開発プロセスの履歴や成果物の品質情報などの進捗情報から構成されている．

このようにソフトウェアタグにはソフトウェア開発プロジェクトにおける情報が幅広く網羅されており，低品質モジュール予測によく用いられる NASA IV&V Facility Metrics Data Program [3] に比べても，特に上流工程（要件定義・設計工程）のメトリクスが多く含まれている．従って，ソフトウェアタグを用いることで従来のデータセットにはない上流工程のメトリクスを用いて予測を行うことが可能となる．

3. 低品質モジュール予測

3.1 概　　要

モジュールとはソフトウェアの品質評価単位の 1 つであり，通常，ソフトウェアは複数のモジュールから構成される．低品質モジュールとは他のモジュールに比べて多くの欠陥を含む可能性の高いモジュールのことを指す．欠陥とは，モジュール中に存在する正しくない処理のことであり，ソフトウェアが所定の機能を遂行できないことの直接の原因となったプログラムの記述誤りである．

一般的にソフトウェアの欠陥の分布は全てのモジュールに均一に分布しているのではなく，特定のモジュールに偏在していることが確認されている [4]．例えば，本研究で用いたデータセットでは全体の 60 %の欠陥が 20 %のモジュールに集中している．従って，この 20 %に当たるような低品質モジュールの特定は，限られたリソースの中で品質向上を行う際に非常に重要な要素となる．

以降，3.2，3.3 節では予測手法として用いたステップワイズ重回帰分析と遺伝的プログラミングについて述べ，3.4 節では評価指標について述べる．

3.2 ステップワイズ重回帰分析

重回帰分析は，ある変数（目的変数）と，それに対して影響すると考えられる複数の変数（説明変数）の間の関係を一次式で表した見積もりモデルを作成し，作成された見積もりモデルに基づいて説明変数から目的変数を予測する手法である．本研究では欠陥数が目的変数であり，設計文書メトリクスが説明変数となる．目的変数と説明変数の関係を表す一次式は，絶対誤差の 2 乗和が最小となる係数が与えられる．

ステップワイズ重回帰分析は，ステップワイズ変数選択法により説明変数を決定し重回帰分析を行う手法である． ステップワイズ変数選択法は，目的変数に対して強く影響する変数を説明変数として選択する手法の一つである．ステップワイズ変数選択法による変数選択は以下の手順で行われる．

（1） 初期モデルを作成する．

（2） 作成されたモデルに対して，偏回帰係数の有意性の検定 (各説明変数の係数が 0 でないかの検定) を行い，指定した有意水準で棄却されない場合に変数を採択し，棄却される場合には変数を除去する（ただし，多重共線性を回避するために，採択する変数の分散拡大要因が 10 以上の場合，またはその変数を採択することによって，他の変数の分散拡大要因が 10 以上となる場合，その変数は採択しないという操作が行われる [6]）．

（3） (2) ができなくなるまで繰り返す．

ステップワイズ変数選択法には，変数増減法と変数減増法がある．変数増減法では，変数を全く含まない定数項のみのモデルを初期モデルとし，変数減増法では，全ての変数を含むモデルを初期モデルとして変数選択を行う．

3.3 遺伝的プログラミング

遺伝的プログラミング (Genetic Programming, GP) とは, プログラムを遺伝子とする個体の集団に対して突然変異や交叉などの遺伝的操作と選択を繰り返し適用することにより, 所望のプログラムを探索していく手法である [7]. GP は, これまで自動プログラミングやアナログ回路の合成など様々な分野に応用されており，低品質モジュールの予測にも用いられている [8].

以下に GP の流れを示す.

（1） ランダムに多数の個体を作成し，初期母集団とする

（2） 各個体の適合度を計算

（3） 終了条件に照らし合わせる．この条件を満たすときは GP を終了

- 終了条件 1：母集団の中に十分に高い適合度を持った個体が存在する
- 終了条件 2：規定回数の世代交代を終了

（4） 終了条件を満たさなかった場合

- 交叉，突然変異などの操作を行う
- 適合度の高い個体を一定数選択
- 世代数+1
- (2) に戻る

3.4 評価指標

低品質モジュール予測の評価指標としては，モジュールオーダーモデル (MOM) [5] の評価に用いられている指標を用いた．MOM では予測欠陥数の多い順にモジュールを並べたランキング（予測ランキング）を作成する．本評価指標はそれが実際の欠陥数の多い順にモジュールを並べたランキング（正解ランキング）とどれほど近いかを表す尺度である．MOM による予測ランキングの評価は以下のステップで行う．

（1） まず，評価対象の範囲を決定するパラメータである，カットオフパーセンタイル c を設定する ($0 \leqq c \leqq 1$)．MOM では c よりも上位のモジュールを品質改善活動のターゲットとする．

（2） 正解ランキング，予測ランキングそれぞれについて c よりも上位のモジュールに含まれる欠陥数 $G(c)$,$\hat{G}(c)$ を計算する．

$$G(c) = \sum_{i:R_i \geqq c} F_i \qquad (1)$$

$$\hat{G}(c) = \sum_{i:\hat{R}_{(\boldsymbol{x}_i)} \geqq c} F_i \qquad (2)$$

R_i はモジュール i の実際の欠陥数 F_i に基づく正解ランキングのパーセンタイルランクである．$\hat{R}(\boldsymbol{x}_i)$ はモジュール i の定量モデルに基づく予測欠陥数 $\hat{F}(\boldsymbol{x}_i)$ に基づく予測ランキングのパーセンタイルランクである．$\boldsymbol{x}$ は説明変数を表す．

（3） それぞれのランキングについて，$G(c)/G_{tot}$, $\hat{G}(c)/G_{tot}$ を計算する．G_{tot} は全モジュールの欠陥数の合計を表す．$G(c)/G_{tot}$，$\hat{G}(c)/G_{tot}$ は c より上位のモジュールに含まれる欠陥数が全モジュールの欠陥数の合計に占める割合を表す．

（4） $\phi(c) = \hat{G}(c)/G(c)$ を計算する．$\phi(c)$ は予測ランキングが正解ランキングとどれほど近いかを表す．この値を MOM の最終的な評価とする．

4. 対象プロジェクト

対象プロジェクトは, 経済産業省の支援を受けて, COSE(注1)参加企業によって実施されたプローブ情報システムの開発プロジェクトである [9]．このプロジェクトには以下の特徴がある．

- 2 ヶ年度に渡ったプロジェクトであり，新規開発を行った 2005 年度版と，保守や改良を行った 2006 年度版のデータがある．
- 実装内容別に分類されたコンポーネントが存在する．コンポーネントは 1 つ以上のファイルの集合であり，全てのファイルはいずれか 1 つのコンポーネントに属している．
- 欠陥情報はコンポーネント毎に記録されている．従って，これらの記録を調べることで，2005 年度，2006 年度に開発・保守されたコンポーネント毎の欠陥数が把握できる．
- コンポーネントと基本・詳細設計書との対応関係を把握できる．

上記の特徴から，本研究ではコンポーネントをモジュールとみなす．変更量のメトリクスを収集している 2006 年度のデータのみを用いてソフトウェアタグを作成し，その中の設計文書メトリクスを説明変数として実験に用いた．実験に用いた設計文書メトリクスの一覧を表 1 に示す．分類，タグ項目は Ver1.0 のソフトウェアタグ規格で定義されているものである．重要変更箇所数，重要追加箇所数，重要削除箇所数については，変更・追加・削除箇所のうち，表紙・変更履歴・目次等，仕様・設計内容に関連のない箇所を除いて計測を行っている．重大原因指摘数については，指摘された設計書の欠陥が混入した原因が，次の 3 つのいずれかに当てはまる指摘の数である．

- 要求の確認不足
- 設計条件の確認不足
- 実現方式の検討不足

基本設計書と詳細設計書のそれぞれについて表 1 に示す 19 のメトリクスを計測したため，合計 38 の設計文書メトリクスが得られた．

次に目的変数となる欠陥数の計測について述べる．テスト工

(注1)：ソフトウェアエンジニアリング技術研究組合 (COSE): COnsortium for Software Engineering.

表 1　設計文書メトリクス一覧
Table 1　A list of design documents metrics

分類	タグ項目	変数名	説明
設計	規模	新規ページ数	新規作成ページ数
		流用ページ数	再利用ページ数
		全ページ数	新規＋流用ページ数
	変更	更新回数	設計書の更新回数 (CVS 上のバージョン数)
		重要変更箇所数	更新毎の重要な変更箇所数の合計
		変更文字数（/100）	更新毎の設計書の変更文字数
		重要追加箇所数	更新毎の重要な追加箇所数の合計
		追加文字数（/100）	更新毎の設計書の追加文字数/100 の合計
		重要削除箇所数	更新毎の重要な削除箇所数の合計
		削除文字数（/100）	更新毎の設計書の削除文字数/100 の合計
品質	レビュー状況	レビュー時間（/新規ページ数）	レビュー時間/新規ページ数
		レビュー時間（/全ページ数）	レビュー時間/全ページ数
	レビュー指摘率	指摘数	レビューで見つかった指摘数
		重要指摘数	上記指摘のうち，重要度が高・中のものの総数
		重大原因指摘数	設計内容に関して指摘された指摘数
		指摘密度（/新規ページ数）	指摘数/新規ページ数
		指摘密度（/全ページ数）	指摘数/全ページ数
		重要指摘密度（/新規ページ数）	重要指摘数/新規ページ数
		重要指摘密度（/全ページ数）	重要指摘数/全ページ数

程で発見された欠陥データには, コンポーネント名が入力されており，基本・詳細設計書との関連付けが可能である．基本・詳細設計書を基準にして設計書に対応するコンポーネントの欠陥数を目的変数とする．欠陥数は単体・社内結合・社間結合・総合テストの 4 工程の合計値を用いた．

尚，各設計文書から生成されたモジュールの欠陥数を使って評価するため，下流工程までのトレースができない文書は分析対象外とした．また，他システムや前年度開発分からの流用コードで，基本設計や詳細設計が行われていないコンポーネントも分析対象外とした．上記のデータの前処理の結果，分析対象となる文書は 24 となった．

5. 実　　験

5.1 実 験 手 順

実験の手順は次の通りである．

（1） 24 の文書からランダムに 16 の文書を選び，それらをモデル構築用のフィットデータ，残りを予測値算出用のテストデータとした．

（2） フィットデータを用いて重回帰および GP モデルを構築した．重回帰モデル構築にはステップワイズ変数選択を行った．

（3） 構築した重回帰，GP モデルにテストデータを適用して予測欠陥数を算出し，予測欠陥数の多い順に並べて文書の予測ランキングを作成した．また，比較のためにランダムで文書を並べたランキング，基本・詳細設計書の全ページ数の合計値の多い順に並べたランキングも作成した．

（4） MOM の最終的な評価指標である $\hat{G}(c)/G_{tot}$ と $\phi(c)$ を算出した．c には 0.875, 0.75, 0.625, 0.5 の 4 つの値を用いた．

（5） 上記 (1) から (4) までを 100 回繰り返し，$\hat{G}(c)/G_{tot}$ と $\phi(c)$ の平均を算出した．

表 2　GP のパラメータ
Table 2　parameters of GP

個体数	2000
世代数	50
個体の木の最大深さ	17
初期個体の木の最小深さ	3
初期個体の木の最大深さ	6
交叉率	0.75
突然変異率	0.1
淘汰率	0.5
個体選択手法	エリート戦略
交叉手法	一点交叉

重回帰分析のパラメータ

ステップワイズ重回帰分析のパラメータとして，偏回帰係数の有意性の検定で用いる有意水準は 5 %，変数選択に変数増減法を用いた．

GP のパラメータ

GP の適合度 $f(k)$ の計算には以下の式を用いた．

$$f(k) = \frac{1}{1 + \log\left(\sum_{j \in O} \exp|\hat{F}_k(\boldsymbol{x}_j) - F(j)|\right)} \tag{3}$$

ただし，k は個体，O はフィットデータである文書群, j は文書，$\hat{F}_k(\boldsymbol{x}_j)$ は個体 k の予測欠陥数，$F(j)$ は文書 j に関連付けられたモジュールの欠陥数を表す．

非終端記号は $\{+, -, \times, /, \sin, \cos, \exp', \log\}$ である．ただし，$\exp' = \exp(\sqrt{x})$．終端記号は $x_1, x_2, \cdots, x_n, R$ ($x_1, x_2, \cdots, x_n$ は設計文書メトリクス，R は $0 \leqq R \leqq 1$ のランダムな実数) である．表 2 にその他の GP のパラメータを示す．

表 3　$\hat{G}(c)/G_{tot}$
Table 3　$\hat{G}(c)/G_{tot}$

c	ランキング				
	正解	重回帰	GP	ランダム	全ページ数
0.875	0.4	0.3	0.29	0.13	0.12
0.75	0.63	0.49	0.48	0.27	0.31
0.625	0.79	0.63	0.62	0.39	0.52
0.5	0.89	0.73	0.73	0.5	0.72

表 4　$\phi(c)$
Table 4　$\phi(c)$

c	ランキング			
	重回帰	GP	ランダム	全ページ数
0.875	0.75	0.73	0.33	0.30
0.75	0.78	0.76	0.43	0.49
0.625	0.80	0.78	0.49	0.66
0.5	0.82	0.82	0.56	0.81

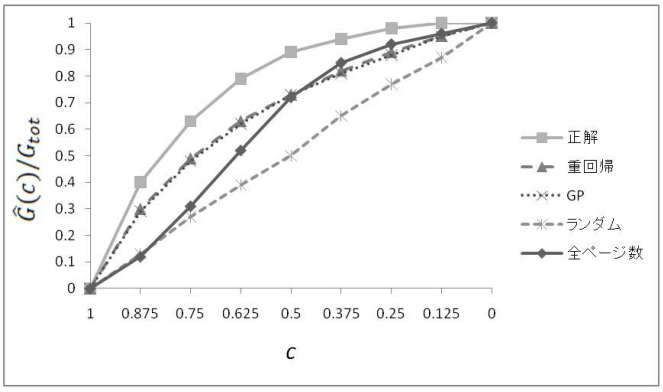

図 1　リフトチャート
Fig. 1　A lift chart

6. 結果・考察

6.1 ランキングの比較

それぞれのランキング毎の $\hat{G}(c)/G_{tot}$ の値を表 3 に，$\phi(c)$ の値を表 4 に示す．また，$\hat{G}(c)/G_{tot}$ をすべての c について表したリフトチャートを図 1 に示す．

表 3 より，正解ランキングの上位 12.5 % ($c \geqq 0.875$) のモジュールは全欠陥の約 40 %，上位 50 % ($c \geqq 0.5$) のモジュールは全欠陥の約 89 %を含んでいる．従って，この正解ランキング上位の低品質モジュールを生成する可能性のある文書を特定することで，限られたリソースの中で最大限に品質改善活動を行うことが可能となる．

表 4, 図 1 より重回帰モデルと GP を用いて算出された予測ランキングは，ランダムなランキングと全ページ数を用いたランキングよりも正解ランキングに近いことが分かった．ランダムなランキングでは上位 12.5 %のモジュールは全欠陥の 13 %，上位 50 %のモジュールは全欠陥の 50 %を含むのに対し，重回帰モデルを使用した予測ランキングでは上位 12.5 %のモジュールは全欠陥の 30 %，上位 50 %のモジュールは全欠陥の 73 %を含んでいる．また，GP を用いた予測ランキングでは上位 12.5 %のモジュールは全欠陥の 29 %，上位 50 %のモジュールは全欠陥の 73 %を含んでいる．全ページ数を用いたランキングと比較した場合，上位 12.5 %では重回帰，GP のランキングの方がより多くの欠陥を含んでいるのに対し，上位 50 %以上になるとほぼ差がなくなった．しかし，限られたリソースでの品質改善活動という目的を考えると，$0.875 \leqq c \leqq 0.5$ の範囲で全ページ数を用いたランキングよりも $\phi(c)$ の値が大きい重回帰や GP のランキングの方がより有効であると考えられる．重回帰モデルを用いたランキングと GP を用いたランキングとの $\phi(c)$ の平均値の差を有意水準 5 %で t 検定を行った結果，$\phi(c)$ の平均値には有意な差がみられた．

6.2 予測手法の検討

本研究の実験結果を用いることで，設計工程が完了した時点で低品質モジュールを生成する可能性のある設計文書を特定することができる．ただし，モジュールの品質は設計書の品質だけでなく開発者の技量，開発プロセスなどの開発環境にも影響を受けると考えられる．そのため，本研究の実験結果を活用するためには，フィットデータでモデルを構築した時の開発環境と，テストデータで実際に予測を行う時の開発環境とをできるだけ近づける必要がある．

6.3 予測結果の応用プロセス

本研究の予測結果の応用例として，設計工程完了時に低品質モジュールを生成する可能性のある文書を特定し，実装工程における品質改善活動を行うことが考えられる．6.2 節の注意点を踏まえた実装工程で品質改善活動を行うプロセスを以下に示す．

（1） 現在進行中のプロジェクトと開発環境（開発体制，開発プロセスなど）が似た過去のプロジェクトで生成されたソフトウェアタグを取得する．ソフトウェアタグにはプロジェクト情報としてこのような開発環境の情報も含まれているため，この情報を利用して類似プロジェクトを検索可能である．

（2） (1) で取得したソフトウェアタグから設計文書メトリクスとモジュールの欠陥数を抽出し，設計文書と各設計文書から生成されたモジュールの欠陥数を関連付ける．

（3） (2) の設計文書メトリクスとモジュールの欠陥数をフィットデータとして重回帰モデルと GP を構築する．

（4） 現在進行中のプロジェクトで生成されているソフトウェアタグに含まれる設計文書メトリクスを (3) のモデルに適用し，低品質モジュールを生成する可能性のある設計文書を特定する．

（5） リソースの制約に応じて (4) の設計文書を対象に実装工程での品質改善活動（経験のある開発者を割り当てる，コードレビューを重点的に行うなど）を行う．

7. 関 連 研 究

これまで低品質モジュールを予測する研究が盛んに行われてきた [10] [11]．これらの多くはソースコードから計測したメトリクスを用いて予測を行っているが，本研究と同じく設計工程で計測可能なメトリクスを用いて低品質モジュールの予測を行っている研究もいくつかみられる．本研究ではページ数のよ

うに設計書の内容に関わらず計測可能なメトリクスを使用しているのに対し，これらは Formal Description Language(FDL) や Unified Modeling Language(UML) 等で記述された詳細設計書の内容に着目している点が異なる．

Ohlsson ら [12] は，FDL オブジェクトの数，送信，受信シグナルの数などの設計工程で計測されるメトリクスと，サイクロマチック数，サブルーチン数などのソースコードから計測されるメトリクスを用いて低品質モジュールの予測を行っている．

予測モデルとしては Spearman の順位相関係数を用いて欠陥数と相関の高いメトリクスを複数選択し，メトリクスの合計値が高い順を予測ランキングとしている．大規模な通信システムのプロジェクトに適用した結果，予測ランキングの上位 20 %のモジュールには全欠陥の 49 %が含まれていた．(正解ランキングでは 57 %)

神谷ら [13] は，分析・設計・実装フェーズに 4 つのチェックポイントを設け，それぞれのチェックポイントで計測可能なメトリクスを用いて低品質モジュールの予測を行い，チェックポイント毎の結果を比較している．メトリクスにはオブジェクト指向設計を対象とする C&K メトリクス [14] を用いている．メール配送システムのプロジェクトに多変量ロジスティック回帰分析を適用した結果，どのチェックポイントでも適合率は高く，後工程になるにつれて再現率が上昇することを示している．

Adrian ら [15] は，コンポーネントの依存関係を用いて低品質モジュールの予測を行っている．52 の Eclipse プラグインを対象に線形回帰分析，リッジ回帰，回帰木，サポートベクターマシーンを適用した結果，サポートベクターマシーンが最も精度が高く，適合率 67 %，再現率 69 %という結果が得られたと報告している．コンポーネントの依存関係は設計完了時に決まっているため，この予測結果を利用することで本研究と同じく実装工程における品質改善活動に役立つと述べている．

8. ま と め

本研究ではソフトウェアタグに含まれる設計文書メトリクスを用いて，多くの欠陥を含む可能性が高い低品質モジュールの予測を試みた．設計文書メトリクスに重回帰分析，GP を適用した結果，ランダムにモジュールを選択する場合や設計書のページ数の多い順に選択する場合よりも高い精度で低品質モジュールの予測を行うことができた．

今後，GP のパラメータ調整，ニューラルネットワークなどの他の予測手法を行った場合との結果の比較を行うことが挙げられる．また，他のデータセットも用いて本研究の有効性をさらに検証していく予定である．

謝　　辞

本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．

文　　献

[1] StagE プロジェクト：http://www.stage-project.jp/.
[2] StagE プロジェクト：“ソフトウェアタグ規格 ver.1.0”, http://www.stage-project.jp/kanri/data/dl/20081029151602.pdf (2008).
[3] N. I. Facility: “Metrics data program”, http://mdp.ivv.nasa.gov/.
[4] F. Shull, V. Basili, B. Boehm, A. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R. Tesoriero and M. Zelkowitz: “What we have learned about fighting defects”, Proceedings of the 8th International Symposium on Software MetricsIEEE Computer Society Washington, DC, USA, p. 249 (2002).
[5] T. Khoshgoftaar, B. Cukic and N. Seliya: “An empirical assessment on program module-order models”, Qual Technol Quant Manag, **4**, 2, pp. 171–190 (2007).
[6] 田中豊, 垂水共之：“統計解析ハンドブック 多変量解析”, 共立出版 (1998).
[7] J. R. Koza1: “Genetic Programming”, The MIT Press (1992).
[8] Y. Liu and T. Khoshgoftaar: “Genetic programming model for software quality classification”, Proceedings of the 6th IEEE International Symposium on High Assurance Systems Engineering, volume, pp. 127–136.
[9] 情報処理推進機構ソフトウェア・エンジニアリング・センター1：“ソフトウェアエンジニアリングの実践ー先進ソフトウェア開発プロジェクトの記録”, 翔泳社, 東京 (2007).
[10] Y. Jiang, B. Cuki, T. Menzies and N. Bartlow: “Comparing design and code metrics for software quality prediction”, Proceedings of the 4th international workshop on Predictor models in software engineeringACM New York, NY, USA, pp. 11–18 (2008).
[11] C.Catal and B.Diri: “A systematic review of software fault prediction studies”, Expert Systems With Applications (2008).
[12] N. Ohlsson, M. Helander and C. Wohlin: “Quality improvement by identification of fault-prone modules using software design metrics”, Proceedings: International Conference on Software Quality, pp. 1–13 (1996).
[13] T. Kamiya, S. Kusumoto and K. Inoue: “Prediction of fault-proneness at early phase in object-oriented development”, Proceedings of the Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 253–258 (1999).
[14] S. Chidamber and C. Kemerer: “A metrics suite for object oriented design”, IEEE Transactions on software engineering, **20**, 6, pp. 476–493 (1994).
[15] A. Schröter, T. Zimmermann and A. Zeller: “Predicting component failures at design time”, ISESE ’06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, New York, NY, USA, ACM, pp. 18–27 (2006).