情報処理学会論文誌 Vol.54 No.2 870–882 (Feb. 2013)

大規模ソフトウェア保守のための影響波及量尺度インパクトスケール

小林 健一[1,a)] 松尾 昭彦[1] 井上 克郎[2] 早瀬 康裕[3] 上村 学[1] 吉野 利明[4]

受付日 2012年5月14日, 採録日 2012年11月2日

概要：ソフトウェア保守においては変更時に他に及ぼす影響の大きいモジュールほど扱いが難しい．本稿では障害予測の精度向上を目的として，変更の影響波及量を定量化するメトリクス「インパクトスケール」を提案する．ソフトウェア保守において効果的な障害予測を行うには，プロダクトメトリクスとプロセスメトリクスの両方が必要といわれている．しかし，実際の保守現場においてはプロセスメトリクスの入手性は低く利用可能な状況は多くない．インパクトスケールの目的はプロセスメトリクスの得られない状況で従来のプロダクトメトリクスより効果的な障害予測を実現することであり，その影響波及モデルは確率的な波及と関係依存的な波及という特徴を持つ．インパクトスケールの有効性検証のため，2 つの大規模企業システムを対象として，ポアソン回帰分析と工数考慮モデルを用いて障害予測を行う実験を実施したところ，インパクトスケールを既存のプロダクトメトリクスからなる予測モデルに追加することにより，10%検査工数における障害検出数が 50%以上上昇するという結果を得た．また，既存のプロダクトメトリクスに依存ネットワーク尺度をあわせた予測モデルに対しても予測性能が上昇するという結果を得た．

キーワード：ソフトウェア保守，メトリクス，障害予測，影響波及解析，グラフ解析

ImpactScale: Change Impact Metric for Maintenance of Large Software Systems

Kenichi Kobayashi[1,a)] Akihiko Matsuo[1] Katsuro Inoue[2] Yasuhiro Hayase[3]
Manabu Kamimura[1] Toshiaki Yoshino[4]

Received: May 14, 2012, Accepted: November 2, 2012

Abstract: In software maintenance, changing modules which affect many other modules are intractable. We defined a new metric, *ImpactScale*, which quantifies the scale of the change impact of a module to improve the accuracy of fault prediction. Both product metrics and process metrics are required to predict faults effectively in maintenance. However, process metrics cannot be always collected in practical situations. ImpactScale is designed to improve the accuracy of fault prediction by using only product metrics under situations without process metrics. The change propagation model for ImpactScale is characterized by probabilistic propagation and relation-sensitive propagation. To evaluate ImpactScale, we predicted faults in two large enterprise systems using Poisson regression and the effort-aware models. The results showed that adding ImpactScale to existing product metrics increased the number of detected faults at 10% effort by over 50%. ImpactScale also improved the predicting model using existing product metrics and dependency network measures.

Keywords: software maintenance, metrics, fault prediction, change impact analysis, graph analysis

[1] 株式会社富士通研究所ソフトウェアシステム研究所
Software Systems Laboratories, Fujitsu Laboratories Limited, Kawasaki, Kanagawa 211–8588, Japan
[2] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[3] 筑波大学システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[4] 富士通株式会社フィールドイノベーション本部
Field Innovation Unit, Fujitsu Limited, Kawasaki, Kanagawa 211–8588, Japan
[a)] kenichi@jp.fujitsu.com

1. はじめに

ソフトウェアの障害予測はレビュー，テスト，品質およびリスク管理を支援する手段として大きく成果をあげている [1], [2]．しかし，リリース後や保守フェーズにおける障害予測はリリース前に比べ困難となる．障害予測には障害と関係の強いメトリクスが説明変数として必要であるが，リリース前障害と相関が高いメトリクス群とリリース後障害と相関が高いメトリクス群は異なり [3], [4]，リリース前

© 2013 Information Processing Society of Japan

に有効なメトリクスはリリース後や保守では多くの場合その有効性を失うためである．たとえばFentonら[3]は，リリース前にはMcCabeの循環的複雑度[5]と障害との間に相関があったが，同じソフトウェアのリリース後には相関がほぼなかったという事例を報告している．

保守では，ソースコードから採取されるプロダクトメトリクスだけでは実用的な性能の障害予測が難しいため，ソフトウェアプロセス・障害履歴・変更履歴などから採取されるプロセスメトリクスを併用して予測性能を上げる試みがなされている[6]．たとえば，過去に変更が行われた箇所は他より障害が起こりやすいという観測に基づく研究が行われている[4], [6], [7]．

しかしながら，現実の保守では経年にともなうドキュメントの陳腐化や人材の流出，レガシーゆえの版管理システムの未整備，保守アウトソーシングビジネスのベンダ切替えなどにより，ソースコード以外の情報が残されている場合は多くない．障害案件情報は比較的入手性が高いが，修正履歴やプロジェクト履歴などはその重要性が近年まで認識されていなかったこともあって維持されている場合が少なく，プロセスメトリクスが採取できない場合が多く，プロダクトメトリクスのみが利用可能な状況で障害予測の性能を高めることは実用上の価値が高い．そのためにはソースコードから障害要因と関係の強い情報を採取する必要がある．長く保守が続くソフトウェアでも除去が難しい障害要因の1つにソフトウェア内の依存関係がある[8], [9]．ソフトウェア内のモジュールが変更されたとき，依存するモジュールにも変更が必要になる可能性がある．その変更はまた新たな変更を招き，すべての必要な変更が完了するまでソフトウェア内を波及していく．この波及効果[10]の見逃しは障害発生の典型的な一因である．Hassanら[9]は論理的結合関係（logical coupling）[8]が変更波及の認識に利用できると報告している．論理的結合関係とはソフトウェア内の同時変更関係であり，同時変更により暗黙の依存関係が観測されたと見なせる．論理的結合関係の強度が障害と関係があることがよく知られている[6]．

Gallらはソフトウェアの依存関係の分類を行い，ソースコードに記述された直接の依存関係を構文的依存関係とし，プロセスから採取される論理的結合関係などを論理的依存関係とした[8]．構文的依存関係はプロダクトメトリクスにより，論理的依存関係はプロセスメトリクスにより計測される．Cataldoら[11]は様々な依存関係と障害との相関を比較し，構文的依存関係と障害との間の相関は有意ではないまたは弱く，論理的依存関係と障害は有意であり最も強かったと報告している．この報告は，既存のプロダクトメトリクスだけでは保守の障害予測には不十分であるという一事例である．

Zimmermanら[12]はソーシャルネットワーク解析（SNA）をソフトウェアのバイナリモジュール間の依存関係グラフに対して行い，SNA分野からネットワーク尺度を導入し予測モデルに追加することで障害予測の性能が向上すると報告した．ネットワーク尺度はプロダクトメトリクスではあるが，前述のCataldoらによる構文的依存関係の分類の中には含まれない．

我々は，Zimmermanらによって用いられたネットワーク尺度と，Cataldoらの論理的依存関係の分類が障害潜在性の共通要素を示していると仮定した．そして，ソースコードの影響波及解析[13]を行えば，論理的結合関係によって曝されるような暗黙の依存関係を抽出できると仮定した．影響波及解析はソフトウェアのある部分が変更された場合にその影響を受ける範囲を同定する技術である．Cataldoらの実験では，対象モジュールに関わる論理的結合関係の数がそのモジュールの障害回数と最も相関が強かった．よって，変更の影響範囲の大きさの推定値も同様に障害回数と相関が強いと期待し，以下の仮説を立てた．

仮説1： 影響波及量を定量化したメトリクスは大規模ソフトウェアにおいて障害予測の性能を向上させることができる．

しかし実際には，影響波及範囲を正確に求めることは難しい．静的解析による影響波及解析[14]には偽陽性の範囲を過剰に導出するという短所があり，精度を上げるためには莫大な計算量を必要とする．動的解析による影響波及解析[15]では偽陽性を抑え影響範囲を絞り込むことが可能であるが，利用頻度が稀な箇所は実行履歴に残りにくく，影響範囲の見落としを招くという短所を持つ．また，実稼働システムでは，運用上の理由から動的解析を実施できない場合も多い．対象モジュールへの変更内容が事前に分かっている場合には影響範囲を求める実践的な技法も提案されているが[16]，我々の目的のためには影響波及範囲は変更を実施する前に知る必要がある．結論として，偽陽性を抑えつつ影響波及範囲を求めることは困難といえる．

ここで，仮説1の焦点は影響波及範囲を正確に求めることではなく，より緩和された問題である「影響波及の大きさの推定」であることに注目する．本稿では，影響が確率的に波及し，かつ二項間に定義された関係に依存して波及を制御する制約を持つ波及モデルを定義し，そのモデル上で影響波及範囲の広さを推定することとした．この推定値を我々は影響波及量を表すメトリクス「インパクトスケール」と名付けた[17]．このインパクトスケールの妥当性は，以下の仮説に立脚する．

仮説2： 確率的かつ関係依存的な波及を持つ波及モデルは，正確だが計算量やデータ収集コストが大きい解析の代わりとなるのに十分な予測性能を持つ．

© 2013 Information Processing Society of Japan

本稿の構成を以下に示す．2 章で影響波及量を表す新しいメトリクス「インパクトスケール」を定義する．3 章では既存の予測モデルにインパクトスケールを追加することで予測性能が上昇するかを評価する．4 章ではインパクトスケールの効果と定義の妥当性について考察し，5 章では関連研究について述べ，最後に 6 章でまとめる．

2. インパクトスケール

本章では，新しいメトリクス「インパクトスケール」（ImpactScale，IS）を定義し，その基盤となる影響波及モデルについて述べる．

2.1 依存グラフと波及グラフ

まず本稿で用いる影響波及モデルについて述べる．ソフトウェアの構成要素であるコードエンティティ（メソッドや関数など）やデータエンティティ（データベースのテーブルやグローバル変数など）をノードとし，ノード間の依存関係を辺（本稿では，辺はすべて有向辺）とするグラフ構造で表したものを依存グラフと呼ぶ．依存関係には，コール依存やデータ依存など様々な種類があり，この種類を「関係タイプ」と呼ぶ．依存グラフはノード間の辺が多重であってもよい．依存グラフ $\mathrm{G_D}$ は $\mathrm{G_D} = \langle \mathrm{V}, \mathrm{E} \rangle$ で定義される．V はノードの集合，E は辺の集合である．辺 $e \in \mathrm{E}$ は $e = \langle s, t, rel \rangle$ で定義される．$s \in \mathrm{V}$ は始点ノード，$t \in \mathrm{V}$ は終点ノード，$rel \in \mathrm{Rel}$ は関係タイプである．Rel は関係タイプの値域集合である．

図 1 左に依存グラフの一例を示す．図の依存グラフには，CALL，READ，WRITE の 3 種の関係タイプがある．図の関数 A が関数 B をコールするとき，関数 A から関数 B への CALL 関係があると呼ぶ．関数 A と関数 B は状態を共有したり実装を分担したりする可能性があり，A から B への波及も B から A への波及も可能性を排除できない．離れたエンティティ間の論理的依存関係を見逃さないためには，波及が双方向に起こりうるという保守的な戦略をとる必要がある．

波及グラフは依存グラフに依存グラフの辺を反転させた辺を追加したものである．依存グラフ $\mathrm{G_D} = \langle \mathrm{V}, \mathrm{E} \rangle$ に対応する波及グラフは $\mathrm{G_P} = \langle \mathrm{V}, \mathrm{E_P} \rangle$ と定義される．$\mathrm{E_P}$ は $\mathrm{E_P} = \mathrm{E} \cup \{\mathrm{reverse}(e) \,|\, e \in \mathrm{E}\}$ で定義される．$\mathrm{reverse}(e)$ は辺 $e = \langle s, t, rel \rangle$ を反転した辺 $e' = \langle t, s, \mathrm{R_}rel \rangle$ を意味する．関係タイプ $\mathrm{R_}rel$ は rel を反転した関係タイプを意味する．影響波及量の計算はこの波及グラフの上で行われる．図 1 右は図 1 左の依存グラフ $\mathrm{G_D}$ に対応する波及グラフ $\mathrm{G_P}$ である．

2.2 確率的な波及

Haney [10] は，モジュールの変更が他のモジュールに確率的に伝わるという解析モデルを提案し，同様の仮定をおいた研究が行われてきた [18], [19]．本稿でも同様の仮定を

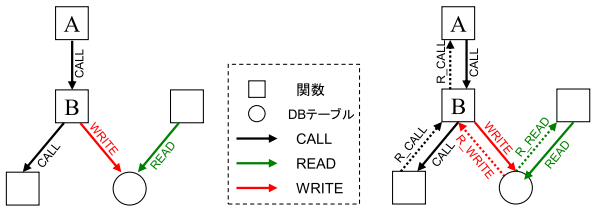


図 1 依存グラフ $\mathrm{G_D}$（左）と波及グラフ $\mathrm{G_P}$（右）の例
Fig. 1 Example of dependency graph $\mathrm{G_D}$ (left) and propagation graph $\mathrm{G_P}$ (right).

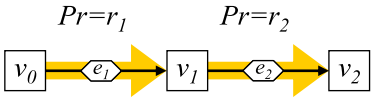


図 2 確率的な波及の例
Fig. 2 Example of probabilistic propagation.

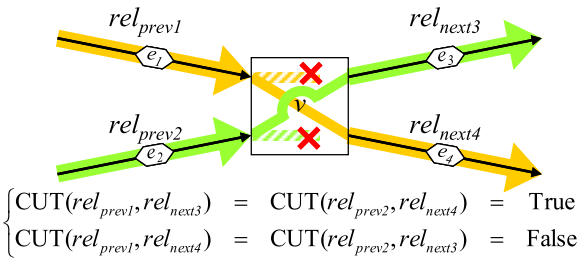


図 3 関係依存的な波及の例
Fig. 3 Example of relation-sensitive propagation.

おく．たとえば，**図 2** の波及グラフにおいて，ノード v_0 からノード v_1 への辺 e_1 があるとき，v_0 から v_1 への影響は波及率 r_1 の確率で伝わるとする．同様に，v_1 から v_2 への影響は波及率 r_2 の確率で，v_0 から v_2 への影響は $r_1 r_2$ の確率で伝わるとする．

2.3 関係依存的な（relation-sensitive）波及

前述のとおり波及の追跡戦略を保守的としたため，影響範囲が過剰に見積もられる恐れがある．これを抑制するための制約を波及追跡戦略に課する．たとえばコールグラフ解析では，解の精度を上げるために，経路探索の打ち切り（以降，カットと呼ぶ）に過去のコール経路や状態を参照するコンテクスト依存（context-sensitive）解析がしばしば利用される [14]．コンテクスト依存解析は計算量が大きくそのままでは利用できないが，我々はそこから解決のための着想を得た．

計算量の爆発を抑えるために，現実的に収集可能かつ追加の計算量が最小となるコンテクストである「前の辺の関係タイプ」を用いる．具体的には，あるノードから次の辺への波及のカットの是非を，前の辺の関係タイプと次の辺の関係タイプの組を定義域とする写像 $\mathrm{Cut}\colon \mathrm{Rel} \times \mathrm{Rel} \mapsto \{\mathrm{True}, \mathrm{False}\}$ で決定する．たとえば，**図 3** に示す波及グラフと写像 Cut において，ノード v の前の辺 e_1 の関係タイプが rel_{prev1}，次の辺 e_3 の関係タイプが rel_{next3} のとき，$\mathrm{Cut}(rel_{prev1}, rel_{next3})$

© 2013 Information Processing Society of Japan

はTrueなので波及はカットされる．この関係タイプによって制御された波及を「関係依存的な（relation-sensitive）波及」と呼ぶ．

2.4 インパクトスケールの定義

本節では，インパクトスケールの定義を示す．波及グラフ $G_P = \langle V, E_P \rangle$ 上で，ノード s からノード t へのあるパス p を，その経由する辺 e_i を用いて $p = (e_1, e_2, \ldots e_n)$ と表す．p は辺の経由順の順序列であり，e_1 の始点が s，e_n の終点が t である．このパスは厳密にはグラフ理論の用語のトレイル（trail）であり，同じノードを複数回通過してもかまわない．辺 e_i の波及率 r_i は次式で与えられる．

$$r_i = R_P \cdot \mathrm{m}(e_i)$$

R_P は基本波及率であり本稿では0.5としている．$\mathrm{m}(e_i)$ は辺 e_i に付随する波及補正値であり，$(0, 1]$ の範囲または $1/R_P$ の値をとるが，通常は1である．パス p を介する影響波及量 $\mathrm{Q}_{\mathrm{path}}(p)$ は次式で与えられる．

$$\mathrm{Q}_{\mathrm{path}}(p) = \frac{1}{R_P} \prod_{i=1}^{n} r_i = {R_P}^{n-1} \prod_{i=1}^{n} \mathrm{m}(e_i)$$

s から t への到達可能なすべてのパスの集合を $\mathrm{P}_{s,t}$ とする．$\mathrm{P}_{s,t}$ は波及グラフを探索して求められる．探索中にあるノードに到達したときの次の辺への波及の是非は，写像Cutで決定される．なお，始点から次の辺へは必ず波及するとする．s から t への影響波及量 $\mathrm{Q}(s,t)$ は次式で与えられる．

$$\mathrm{Q}(s,t) = \begin{cases} \max\limits_{p \in P_{s,t}} \mathrm{Q}_{\mathrm{path}}(p) & : \ \mathrm{P}_{s,t} \neq \phi \\ 0 & : \ \mathrm{P}_{s,t} = \phi \end{cases}$$

$\mathrm{Q}(s,t)$ の計算は本質的には最短経路問題の求解と等価である．s のインパクトスケールを $\mathrm{IS}(s)$ と表すと，$\mathrm{IS}(s)$ は次式で与えられる（$\mathrm{V} \backslash s$ はVから s を除いた部分集合の意）．

$$\mathrm{IS}(s) = \sum_{t \in \mathrm{V} \backslash s} \mathrm{Q}(s,t)$$

以上からなる4つ組 $\langle \mathrm{IS}, R_P, \mathrm{Rel}, \mathrm{Cut} \rangle$ が，インパクトスケールの定義である．

2.5 カットルール

関係タイプの値域集合Relと写像Cutは抽出する依存関係と目的に応じて自由に設定でき，本稿では，$\mathrm{Rel} = \{\mathsf{CALL}, \mathsf{READ}, \mathsf{WRITE}, \mathsf{R_CALL}, \mathsf{R_READ}, \mathsf{R_WRITE}\}$ とする．CALLはコール，READはリードアクセス，WRITEはライトアクセス（リードアクセスも含意する）を意味し，R_CALL，R_READ，R_WRITEはそれぞれ反転された関係タイプである．写像Cutは次式を用いる．

$$\mathrm{Cut}(p,n) = \begin{cases} \text{True} : p = \mathsf{CALL} \wedge n = \mathsf{R_CALL} & \cdots \text{(Rule1)} \\ \text{True} : p = \mathsf{R_CALL} \wedge n = \mathsf{CALL} & \cdots \text{(Rule2)} \\ \text{True} : p = \mathsf{READ} & \cdots \text{(Rule3)} \\ \text{False} : \text{otherwise} \end{cases}$$

写像Cutの上の表記の条件部をカットルールと呼ぶ．

前記のカットルールは発見的であり，その妥当性は4章で議論するが，ここではその意図を述べる．CALLからCALLへの波及はトップダウン設計に沿って上位モジュールの変更が下位モジュールに波及することを想定し，R_CALLからR_CALLへの波及はボトムアップ設計に沿って下位モジュールの変更が上位モジュールに波及することを想定している．Rule1はCALLからR_CALLへの波及をカットし，Rule2はR_CALLからCALLへの波及をカットするものであるが，これらは，一方の設計に沿っていながら，他方の設計へ切り替えることは起きにくいという推測から設けた．Rule3は，データフロー解析からの類推から設けた．

2.6 計算量

インパクトスケールの計算の大部分は経路探索に占められる．波及グラフは多重辺を持つグラフであるため，通常のグラフアルゴリズムは適用できない．Whaleyら[20]はコンテクスト依存解析のために，ノード複製によりコンテクスト付きの複雑な経路探索問題をコンテクストなしの単純な経路探索問題に帰結する方法を示した．この方法を用いると，インパクトスケールの経路探索も多重辺のない単純なグラフ上の経路探索問題に帰結でき，通常のアルゴリズムが適用できる．ダイクストラの最短経路アルゴリズムを波及グラフ $G_P = \langle V, E_P \rangle$ に適用した場合，Rを関係タイプの種類数とすれば，計算量は $O(\mathrm{R}|\mathrm{E_P}| + \mathrm{R}|\mathrm{V}| \log(\mathrm{R}|\mathrm{V}|))$ である．これは対象システムが大規模であっても実用的な時間で計算可能であることを意味する．

2.7 計算例

図4左はノードCのインパクトスケールの計算例である．パス $(\mathrm{C} \to \mathrm{A})$，$(\mathrm{C} \to \mathrm{D})$，$(\mathrm{C} \to \mathrm{X})$，$(\mathrm{C} \to \mathrm{X}, \mathrm{X} \to \mathrm{F})$，$(\mathrm{C} \to \mathrm{X}, \mathrm{X} \to \mathrm{F}, \mathrm{F} \to \mathrm{E})$，$(\mathrm{C} \to \mathrm{X}, \mathrm{X} \to \mathrm{F}, \mathrm{F} \to \mathrm{H})$ が探索されている．辺 $\mathrm{C} \to \mathrm{A}$ の関係タイプはR_CALLで，辺 $\mathrm{A} \to \mathrm{B}$ の関係タイプはCALLであるため，パス $(\mathrm{C} \to \mathrm{A}, \mathrm{A} \to \mathrm{B})$ はRule2によって辺 $\mathrm{A} \to \mathrm{B}$ の箇所でカットされる．同様に，パス $(\mathrm{C} \to \mathrm{X}, \mathrm{X} \to \mathrm{F}, \mathrm{F} \to \mathrm{H}, \mathrm{H} \to \mathrm{G})$ はRule1によって辺 $\mathrm{H} \to \mathrm{G}$ の箇所でカットされる．Cのインパクトス

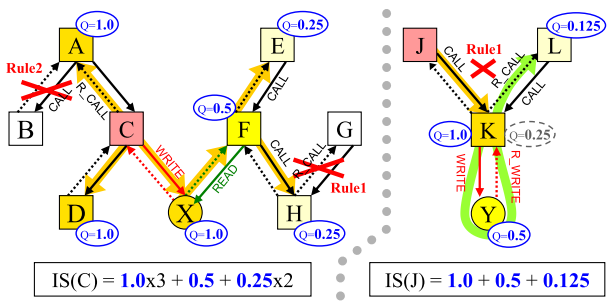


図4 インパクトスケールの計算例
Fig. 4 Example of calculating ImpactScale.

© 2013 Information Processing Society of Japan

ケールは Q(C, A)，Q(C, D)，Q(C, X)，Q(C, F)，Q(C, E)，Q(C, H) の合計であり，4.0 である．

図 4 右はノード J のインパクトスケールの計算例であり，「関係依存的な波及」の特徴的なケースが現れている．(J→K, K→L) というパスは Rule1 により辺 K→L の箇所でカットされるが，別のパス (J→K, K→Y, Y→K, K→L) というパスはカットされない．結果として，J からの影響は L まで波及する．J のインパクトスケールは Q(J, K)，Q(J, Y)，Q(J, L) の合計であり，1.625 である．データノード Y は J から L への媒介として働くため，Y がなければ K までしか波及せず J のインパクトスケールは 1.0 となる．

2.8 用途

インパクトスケールの用途として，たとえば以下のものが想定される．

- 限られた予算と期間の下での障害予測を用いた品質改善．特に本稿は次節以降でこれについて述べる．
- バグ修正作業の早期見積り．修正対象となるソースコードからの影響範囲の大きさは，コードレビューや回帰テストなど修正作業の工数に直接影響するため．
- ソフトウェアシステム全体の早期の品質監査．
- ソフトウェアのモジュラリティを維持するためのコード修正作業の監視．インパクトスケールの目立った上昇は，モジュラリティ違反の兆候である．

3. 評価

仮説 1 と仮説 2 の検証と，インパクトスケールの有効性を評価するため，以下の研究課題（research question）を設定する．

RQ1： インパクトスケールを既存のプロダクトメトリクスに加えることは障害予測性能を向上させられるか？

RQ2： インパクトスケールを既存のプロダクトメトリクスとネットワーク尺度をあわせたものに加えることは障害予測性能を向上させられるか？

続く節の構成は以下のとおりである．

- 3.1 節〜3.3 節では実験の設定について説明する．
- RQ1 については，障害予測性能の評価を，3.4 節で二項判別を用いて行い，3.5 節で工数考慮モデル [21] を用いて行う．
- RQ2 については，3.6 節で工数考慮モデルと階層モデル分析を用いて行う．

3.1 評価対象ソフトウェア

我々は評価対象として，2 つの企業から大規模な勘定系ソフトウェアシステムのデータを入手した．これらを選んだ規準は，長期間保守されており，適用領域における標準的な規模を持ち，そして，ソースコード解析の精度の影響をこの評価において最小化するために解析の容易な言語で記述されていることである．2 つのシステムから収集したデータセットを DS1，DS2 と称し，そのプロファイルを**表 1** に示す．両システムは COBOL 言語で記述されており，それぞれについて我々は近年の 40 カ月分の障害レポートを収集した．

表 1 評価対象ソフトウェアのプロファイル
Table 1 Profiles of the target software systems.

名前	モジュール数	合計 LOC	障害数	障害モジュール数
DS1	5.8k	1.6M	269	215
DS2	7.6k	3.7M	250	208

表 2 採集したメトリクス
Table 2 Collected product metrics.

メトリクス	説明
LOC	コメント・空行を除いたコード行数
WMC	モジュール全体の McCabe の循環的複雑度
MaxVG	セクション毎の McCabe の循環的複雑度の最大値
Sections	セクション数（モジュール内のブロック数に相当）
Calls	コール命令の数
Fan-in	モジュールを呼出すモジュールの数
Fan-out	モジュールによって呼出されるモジュールの数
IS	インパクトスケール

本評価では，1 モジュールは，COBOL 言語の「プログラム」であり，ソースファイル 1 本である．「プログラム」はコール命令の対象となる単位で，他言語の関数に相当する．採集したメトリクスを**表 2** に示す．「セクション」は COBOL 言語特有の概念で，関数内部のサブブロックに相当する．

3.2 インパクトスケールの測定

インパクトスケールの測定は以下の手順で行われる．まず，測定対象ソフトウェアから静的解析でコール命令（CALL）と，データベースやファイルへのアクセス操作（READ，WRITE）を抽出する．次に，依存グラフと波及グラフを構築する．そして，その依存グラフのすべてのノードについて，インパクトスケールを前述の定義に従い計算する．ここでは $m(e)$ はすべて 1.0 とした．対象システムにおける計算時間は数十秒であった（使用した CPU は Core2Duo 2.5 GHz）．**図 5** に測定されたインパクトスケールの分布と統計量を示す．図中，ほとんどのモジュールが小さいインパクトスケールの値を持ち，ごく一部が大きな値を持つ．我々の経験では，この傾向はほとんどの大規模ソフトウェアにおいて共通する．

3.3 反復検証

結果の妥当性を保証するため，各データセットの各評価につき 100 回のランダムサブサンプリングバリデーション

© 2013 Information Processing Society of Japan

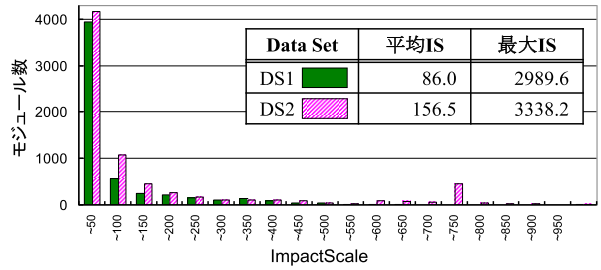


図 5 測定されたインパクトスケール
Fig. 5 Measured ImpactScale.

を行った．100 回の反復のそれぞれにおいて，対象データセットの全モジュール中の 2/3 を訓練セットとしてランダムに選び，残りの 1/3 のモジュールをテストセットとして用いて，予測性能の測定を行う．障害率は数%と低いが，両データセットともサンプルサイズが大きく障害数も十分であるため，層別サンプリング（Stratified sampling）のような特別な処置は行っていない．比較対象群の差異の有意性検定には，標本サイズ 100 に対する Wilcoxon の符号順位検定を用いた．

3.4 二項判別による評価（RQ1）

本節では，インパクトスケールを既存のプロダクトメトリクス群に追加することで，予測性能向上が得られるかを評価するために，二項判別を行う．

3.4.1 ロジスティック回帰

分類器としてロジスティック回帰モデルを用いる．ロジスティック回帰モデルは次式で表される．

$$y = \frac{\exp(b_1x_1 + b_2x_2 + \cdots + b_0)}{1 + \exp(b_1x_1 + b_2x_2 + \cdots + b_0)}$$

このとき，y は目的変数，各 x_i は説明変数，各 b_i は偏回帰係数である．式 $b_1x_1 + b_2x_2 + \cdots + b_0$ は線形予測子と呼ばれる．偏回帰係数を推定するためには最尤推定法を用いる．目的変数 y $(0 < y < 1)$ は障害潜在確率と解釈される．

3.4.2 モデル選択

各反復において，各訓練セットに対する最良の予測モデルを選択するため変数選択法を述べる．表 2 にあげたメトリクスが予測モデルの説明変数の候補であり，メトリクスの対数値（値域に 0 を含む場合は 1 を加算後の対数値）も代替候補とした．すなわち，各メトリクスにつき，値・対数値・非選択の三択である．偏った分布のデータに対数変換を施す措置は標準的に用いられる技法である．モデルの選択規準には AIC（赤池情報量規準）[22] を用いた．AIC はモデルのあてはまり度合い（尤度）に説明変数の数によるペナルティを課したもので，AIC が小さいほどモデルが良いことを示す規準である．モデルの説明変数としてインパクトスケール抜きの 7 つ，またはインパクトスケールを含めた 8 つメトリクスのすべての組合せ（3^7 通り，または 3^8 通り）から，AIC 最小となる組合せを選んだ．多重共線性を避けるために，説明変数のいずれかの VIF（Variance Inflation Factor）が 10 以上となったモデルは除外した [23]．

表 3 二項判別における障害予測性能の向上
Table 3 Performance improvement in binary classification.

データセット	性能尺度	モデル MET		モデル MET+IS		IS による向上†
		平均	(標準偏差)	平均	(標準偏差)	
DS1	適合率	0.148	(0.029)	0.168	(0.031)	+0.020
	再現率	0.315	(0.051)	0.392	(0.048)	+0.077
	F1	0.200	(0.033)	0.234	(0.034)	+0.034
DS2	適合率	0.139	(0.030)	0.162	(0.033)	+0.023
	再現率	0.253	(0.042)	0.334	(0.057)	+0.081
	F1	0.177	(0.029)	0.216	(0.034)	+0.039

† すべての向上は Wilcoxon の符号順位検定で有意 (P<0.001)

3.4.3 モデル学習とテスト

各反復において，表 2 にあげたインパクトスケール以外の既存メトリクスを用いて最良モデルを学習したものが「モデル MET」である．同様に既存メトリクスとインパクトスケールをあわせて最良モデルを学習したものが「モデル MET+IS」である．学習とテストでは，障害数が 1 以上であれば障害あり，障害数が 0 ならば障害なしとして扱った．

次に，テストセットに対して予測を行った．テストセットの各モジュールに対し，目的変数 y が所定の閾値以上ならばそのモジュールを障害モジュールと分類した．DS1 と DS2 の両方とも障害率が低い（3.7%と 2.7%）ため，閾値は 0.1 とした．

3.4.4 性能尺度

予測性能の評価のため，各バリデーションにおいて適合率（Precision）と再現率（Recall）と F1 値を用いる．TP を障害ありと観測され障害ありと予測されたモジュール数，TN を障害なしと観測され障害なしと予測されたモジュール数，FP を障害なしと観測され障害ありと予測されたモジュール数，FN を障害ありと観測され障害なしと予測されたモジュール数とすると，適合率は $TP/(TP + FP)$ と定義され，1.0 に近い適合率は障害ありと誤って判定されるモジュールがほとんどないことを意味する．再現率は $TP/(TP + FN)$ と定義され，1.0 に近い再現率はほとんどの障害が検出されることを意味する．F1 値は適合率と再現率の二値の要約尺度で，二値の調和平均（適合率の逆数と再現率の逆数の平均の逆数）で定義される．

3.4.5 結果

表 3 に 100 回の反復結果の各性能尺度の平均値と標準偏差を示す．「モデル MET」と「モデル MET+IS」の列はそれぞれのモデルでの性能尺度，「IS による向上」は 2 つのモデルの差，つまりインパクトスケールを説明変数に追加したことによる向上である．表から，インパクトスケールの追加は両データセットですべての性能尺度を有意に向上させていることが分かる．この結果は RQ1 を肯定的に

© 2013 Information Processing Society of Japan

支持する．

F1 値の向上は有意であったが低い値にとどまっている（DS1: 0.234，DS2: 0.216）．障害率が低いことが一般的な保守での障害予測ではよく見られる結果であるが，この F1 値が実用上どの程度有効であるかの解釈は難しい．そこで，次節では障害予測性能が工数に及ぼす効果を直接的に比較できる手法である工数考慮モデルを用いて評価を行う．

3.5 工数考慮モデルによる評価（RQ1）

3.5.1 工数考慮モデルと性能尺度

近年，モジュールのテストや監査にかかる工数を障害予測の性能評価でも考慮すべきと指摘されている [21], [24], [25], [26]．障害と規模には相関がある [3] ため，障害ありと分類されるモジュールは規模が大きい傾向がある．たとえば Arisholm らはモジュールのテスト工数はそのモジュールの規模に大まかに比例すると報告している [24]．実際の保守では，予算とスケジュールは多くの場合要求が厳しく，障害予測の工数効率性は実務者にとっての重要な関心となってきている．

上記の議論をふまえ我々は，予測性能評価に Mende ら [21] によって提案された工数考慮モデルを用いた．工数考慮モデルでは，相対リスク $\mathrm{R_{dd}}(x)$ がモジュールのテストや監査の優先付けに用いられる．$\mathrm{R_{dd}}(x)$ は $\#errors(x)$ をモジュール x の障害数，$\mathrm{E}(x)$ をモジュール x に必要な工数としたとき，$\#errors(x)/\mathrm{E}(x)$ と定義される．ここでは既存研究 [21], [26] と同様に $\mathrm{E}(x)$ として LOC を用いる．この場合，$\mathrm{R_{dd}}(x)$ は障害密度を意味する．工数考慮モデルでは $\mathrm{R_{dd}}(x)$（すなわち，障害密度）を予測し，モジュールを障害密度の降順でテストまたは監査を行う．予測性能を調べるためには，**図 6** に例示される工数ベース累積リフトチャートが用いられる．図の実曲線が総合的な予測性能を示している．この曲線はコスト効果曲線（cost-effectiveness curve）[24]，または工数対検出率曲線（effort-vs-PD curve）[25] と呼ばれる．X 軸は相対累積工数を表し，Y 軸は費やした工数での障害検出率を表している．曲線の立ち上がりが急ならば，予測はより工数効率が高いことを意味している．曲線が対角線を下回るならば，予測がほとんどランダムで意味のないことを表す．「完全予測曲線」と題された点線曲線は，完全な予測が行われた場合を示しており，性能の上限値を表している．

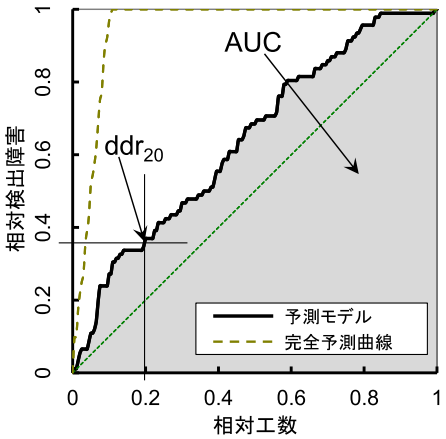


図 6 工数ベース累積リフトチャートと AUC，ddr の例
Fig. 6 Example of effort-based cumulative lift chart, AUC and ddr.

図 6 は本稿で使用する 2 つの性能尺度についても説明している．「AUC」（Area Under the effort-vs-PD Curve）は工数対検出率曲線の下部領域の面積を意味する [25]．AUC は全領域での平均性能を表しており，AUC が 1.0 以下の上限値に近ければ，予測性能が高いことを意味する．一方，AUC が 0.5 前後かそれ以下ならば，その予測に意義はない．「ddr」（defect detection rate）は障害検出率である [21]．本稿では「ddr x」は全工数の x%において検出した障害率を意味する．保守では典型的に工数が逼迫しており，この尺度は実践者のたとえば「10%の行数をレビューするとどれだけの障害を検出できるか？」などの問いに直接答えることができる．高い ddr x は高い予測性能を意味する．既存研究には x として 20%を用いるものがあるが，大規模システムにおける実践的見地からは 20%は大きすぎるため，本稿では $\mathrm{ddr_{10}}$ の方を $\mathrm{ddr_{20}}$ よりも重視する．もし $\mathrm{ddr_{10}}$ が 0.1 より小さければ，予測による利得がないことを意味する．

3.5.2 ポアソン回帰分析を用いた障害密度予測

障害密度予測のためには連続値か計数値が出力となる予測モデルが必要となる．本稿では，ポアソン回帰モデル [27] を使用する．ポアソン回帰モデルは確率事象のカウント予測に一般的に用いられるモデルであり，障害発生をポアソン過程に従う確率事象と見なすことにより従来から障害予測の分野で用いられている [6], [28]．ポアソン回帰モデルの式は一般的には次式で表される．

$$y = \exp(b_1 x_1 + b_2 x_2 + \cdots + b_0)$$

偏回帰係数 b_i を推定するためには最尤推定法を用いる．目的変数 y（$0 < y$）は障害回数の期待値と解釈される．目的変数が発生回数なので発生密度はただちに導出可能である．密度を予測するためには係数を 1.0 に固定したオフセット項 $\log(\mathrm{LOC})$ を線形予測子に以下のとおり加えて学習すればよい．

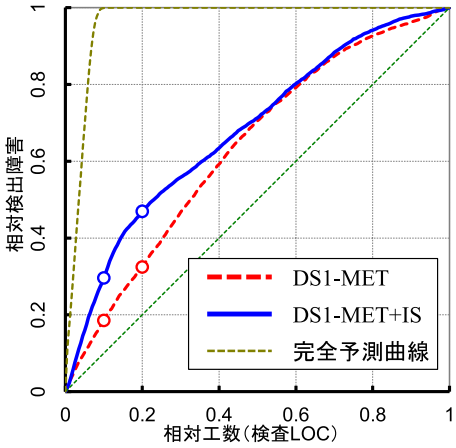
$$y = \exp(b_1 x_1 + b_2 x_2 + \cdots + b_0 + \log(\mathrm{LOC}))$$

オフセット項の存在にかかわらず，$\log(\mathrm{LOC})$ を説明変数に加えることが可能である．

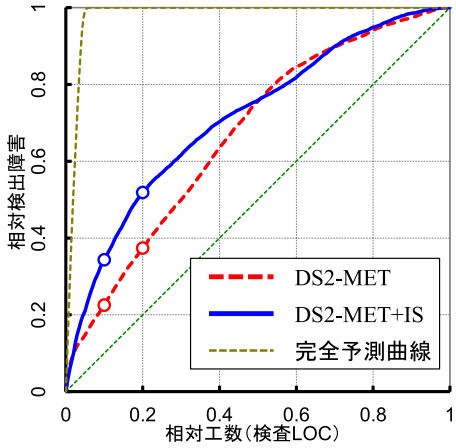
3.5.3 モデル選択，モデル学習とテスト

最良モデルの選択および，モデルの学習とテストは 3.4 節と同様に行われる．差異は，障害数を学習に直接用いることのみである．

© 2013 Information Processing Society of Japan



(a) DS1の工数ベース累積リフトチャート



(b) DS2の工数ベース累積リフトチャート

図 7 インパクトスケールあり/なしモデルの工数考慮モデル比較

Fig. 7 Effort-aware comparison between models with/without ImpactScale.

表 4 障害密度と工数考慮モデルにおける障害予測性能の向上

Table 4 Performance improvement in fault density prediction and effort-aware-model.

性能尺度	モデル DS1-MET		モデル DS1-MET+IS		IS による向上†
	平均	(標準偏差)	平均	(標準偏差)	
AUC	0.635	(0.027)	0.680	(0.027)	+0.045
ddr_{10}	0.186	(0.042)	0.296	(0.051)	1.60倍
ddr_{20}	0.325	(0.043)	0.470	(0.055)	1.45倍

AUC の上限値は0.977.
†すべての向上は Wilcoxon の符号順位検定で有意(P<0.001)

性能尺度	モデル DS2-MET		モデル DS2-MET+IS		IS による向上‡
	平均	(標準偏差)	平均	(標準偏差)	
AUC	0.669	(0.025)	0.714	(0.025)	+0.045
ddr_{10}	0.225	(0.047)	0.343	(0.046)	1.53倍
ddr_{20}	0.374	(0.053)	0.518	(0.051)	1.39倍

AUC の上限値は0.984.
‡すべての向上は Wilcoxon の符号順位検定で有意(P<0.001)

3.5.4 結果

図 7 と表 4 に 100 回の反復結果を示す．図中，赤い破線の各曲線がインパクトスケールなしのモデルの性能を表す（DS1-MET/DS2-MET）．青い実線の各曲線がインパクトスケールありのモデルの性能を表す（DS-MET+IS/DS2-MET+IS）．曲線上の白い中抜き点は表 4 にある ddr_{10} と ddr_{20} の測定点である．DS1 と DS2 の両方で，図ではモデル MET+IS の曲線はモデル MET の曲線をほぼすべての範囲で上回り，表ではすべての性能尺度で MET+IS が上回る．特に，ddr_{10} の向上は顕著である．この結果は，全検査工数の 10%の工数において，検出できる障害数に 1.5 倍の向上が得られることを意味する．この結果もまた，RQ1 を肯定的に支持する．

3.6 ネットワーク尺度との比較（RQ2）

Zimmermann と Nagappan らは，ソーシャルネットワーク解析（SNA）を依存グラフに行い，障害予測に利用した [12]．彼らの研究と追試 [29], [30], [31] は，ネットワーク尺度の有効性を示した．インパクトスケールは依存グラフ上で測定されるため，その有効性を確認するために RQ2 を調べる必要がある．

RQ2（再掲）： インパクトスケールを既存のプロダクトメトリクスとネットワーク尺度をあわせたものに加えることは障害予測性能を向上させられるか？

3.6.1 SNA のネットワーク尺度

ネットワーク尺度はネットワークの様々なトポロジ的特徴量である．Zimmermann らは 58 のネットワーク尺度を UCINet ツール [32] を用いて収集し，障害予測を行った．使用されたネットワーク尺度の完全なリストと説明は文献 [12] と [33] を参照のこと．本稿では既存研究 [12], [29], [30], [31] と同じく UCINet を用い，インパクトスケールの測定に用いた DS1 の依存グラフに対してネットワーク尺度を収集した．DS2 の依存グラフは UCINet が扱える規模を超えているため，本節の評価では DS1 のみを対象とした．

3.6.2 主成分回帰分析

本節でもポアソン回帰分析を用いて障害密度予測を行い，工数考慮モデルを用いて評価を行う．この場合，説明変数が多く（60 以上）多重共線性は避けられないため，既存研究と同様に主成分分析（PCA）[34] を用いる．PCA は主成分を発見するための教師なし学習である．すべての主成分は直交するため，主成分を説明変数として用いる回帰分析では多重共線性の問題は発生しない．この組合せは主成分回帰分析と呼ばれる．

3.6.3 モデル選択，モデル学習とテスト

以下の 4 つのモデルを用意した．

- MET 既存メトリクスからなる最良モデル
- MET+IS 既存メトリクスとインパクトスケールからなる最良モデル
- MET+SNA 既存メトリクスとネットワーク尺度か

© 2013 Information Processing Society of Japan

らなる最良モデル

- MET+SNA+IS 既存メトリクス，ネットワーク尺度，インパクトスケールからなる最良モデル

各反復において，4 つのモデルごとに，全メトリクスと全尺度を対数変換し PCA を用いて主成分に変換する．そのうち，累積寄与率 99%までの主成分をポアソン回帰分析の説明変数として使用する．使用された主成分数の平均値は MET で 6.0，MET+IS で 7.0，MET+SNA で 27.8，MET+SNA+IS で 28.8 であった．テスト手順は 3.5 節と同じである．

3.6.4 結果

図 8 は，結果を階層モデル比較で示したものである．4 つの楕円は 4 つの予測モデルを表し，それぞれの性能尺度 AUC と ddr_{10} が記されている．実線矢印は説明変数の追加によるモデルの拡張を表しており，付された数字は上が AUC の向上，下が ddr_{10} の向上を表している．たとえば，矢印 (a) はインパクトスケールをモデル MET に追加し，予測性能が AUC で 0.054 向上，ddr_{10} で 0.074 向上したことを表している．これは 3.4 節，3.5 節の結果に整合する．矢印 (b) はネットワーク尺度の効果を表し，既存の SNA を用いた研究の結果に整合する．

RQ2 を調べるため，矢印 (d) と矢印 (e) に注目する．矢印 (d) はモデル MET+SNA にインパクトスケールを追加することが有意であることを表している．これはインパクトスケールがネットワーク尺度とは異なる障害要因説明要素を含んでいることを意味している．矢印 (e) はインパクトスケールとネットワーク尺度との比較であり，ddr_{10} が増加していることからインパクトスケールが工数に限りがあるときにネットワーク尺度よりも高い検出率を持つといえる．一方，AUC が減少していることから工数に際限がない想定では予測性能が若干低くなるといえる．いずれにせよ，インパクトスケール単独がもたらす予測性能の向上はネットワーク尺度の集合全体による向上に匹敵し，これは予測モデルの解釈容易性という点で大きな利点となる．以上から，評価対象ソフトウェアが 1 システムではあるが，RQ2 を支持する結果が得られた．ネットワーク尺度については関連研究の章でさらに述べる．

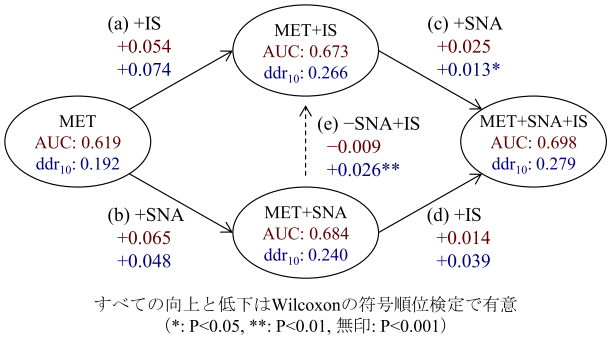


図 8 インパクトスケールとネットワーク尺度の階層モデル比較

Fig. 8 Hierarchical model comparison between ImpactScale and network measures.

4. 考察

本章では，インパクトスケールが確かに障害予測に寄与しているかどうか，2 章で与えたインパクトスケールの定義が妥当かについて考察する．

4.1 障害予測への寄与

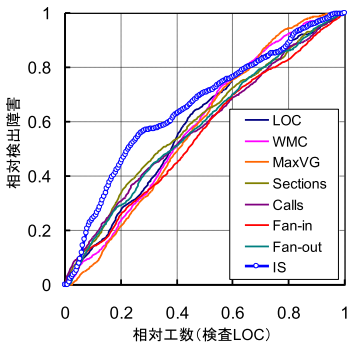
まず，インパクトスケールと他のメトリクスとの関係を調べる．DS1 についての，表 2 のメトリクス間のスピアマンの順位相関係数を**表 5** に示す．絶対値が 0.5 以上のものを太字で示している．インパクトスケールは他のメトリクスとの相関係数の絶対値がたかだか 0.38 であり明らかに低い．この結果は DS2 でも同様であった．ネットワーク尺度に関しては，インパクトスケールとの相関係数はたかだか 0.60 であり，図 7 に示した向上に寄与するほど十分に低い．これはインパクトスケールが他のメトリクスとは独立，つまり他のメトリクスの示さない障害要素を示しているといえる．

次に，DS1 で，各メトリクスによる 1 変数ポアソン回帰で障害密度予測を行った結果（100 回平均）が**図 9** の工数ベース累積リフトチャートと図中の表である．インパクトスケール以外のメトリクスは AUC が 0.5 をやや上回る程度，ddr_{10} は 0.1 前後であり予測が無判別に近いのに対し，インパクトスケールは単独で高い予測性能を発揮して

表 5 メトリクス間のスピアマンの順位相関係数

Table 5 Spearman rank correlation coefficients between metrics.

	WMC	MaxVG	Section	Calls	Fan-in	Fan-out	IS
LOC	**0.90**	**0.71**	**0.80**	**0.69**	-0.26	**0.66**	0.18
WMC	-	**0.88**	**0.59**	**0.51**	-0.13	0.48	0.11
MaxVG	-	-	0.35	0.34	-0.05	0.32	0.04
Section	-	-	-	**0.78**	-0.33	**0.79**	0.31
Calls	-	-	-	-	-0.32	**0.94**	0.33
Fan-in	-	-	-	-	-	-0.32	0.17
Fan-out	-	-	-	-	-	-	0.38



Metrics	AUC	ddr_{10}
LOC	0.591	0.137
WMC	0.586	0.107
MaxVG	0.578	0.073
Sections	0.596	0.156
Calls	0.579	0.173
Fan-in	0.549	0.139
Fan-out	0.580	0.162
IS	**0.654**	**0.242**

図 9 1 変数障害密度予測の結果

Fig. 9 Results of fault density prediction with single metric.

© 2013 Information Processing Society of Japan

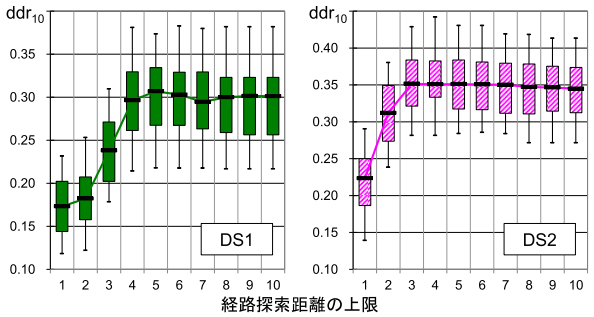


図 10 遠隔ノードを考慮することの効果
Fig. 10 Effect of considering distant nodes.

いる．よって，インパクトスケールは DS1 において障害予測に強く寄与しているといえる．DS2 も同様であった．

RQ1 と RQ2 に関する結果と本節から，仮説 1 は本稿において妥当といえる．

仮説 1（再掲）：影響波及量を定量化したメトリクスは大規模ソフトウェアにおいて障害予測の性能を向上させることができる．

4.2 インパクトスケールの定義の妥当性

ここでは，インパクトスケールの定義が妥当かどうか検証する．インパクトスケールの定義を変化させ，それに対し予測性能がどう変化するかを ddr_{10} を基準に調べる．

まず，依存グラフ上でグラフ距離が離れた遠隔ノードを考慮することの意義を調べる．インパクトスケールは経路探索により計算される．そのため，経路探索の距離に上限を設けてその上限を変化させ，それによる予測性能の変化を調べる．**図 10** は DS1，DS2 について，3.5 節に述べたインパクトスケールありの最良モデルを用いて探索上限距離（横軸）に制約を課す変更を加えたインパクトスケールで評価した予測性能 ddr_{10}（縦軸）である．探索上限距離が 1 ならば，インパクトスケールは定義上 Fan-in と Fan-out の和にほぼ等しいため，上限距離が 1 で ddr_{10} が十分大きければ，遠隔ノードを考慮することに意義がないことになる．しかし，図では，DS1 では距離 5 まで，DS2 では距離 3 まで，ddr_{10} が増加し続けており，遠隔ノードの考慮に意義があることを示している．また，考慮すべき上限距離もソフトウェアごとに異なることを示している．ただ，DS1，DS2 ともに上限距離を十分にとっても ddr_{10} に大きな影響を与えないことから，上限距離は単に十分に大きくとればよいといえる．

次に，「関係依存的な波及」を組み込んだことに意義があるかを調べる．**図 11** の箱ひげ図中のモデル MET+IS は，3.5 節に述べたインパクトスケールを含んだ最良モデル，モデル MET はインパクトスケールを含まない最良モデルである．モデル NoCut はカットルールを除いて関係依存的な波及を抑制するよう定義を変えたインパクトスケールを含む最良モデルである．三者の ddr_{10} を比較すると，モデル NoCut はモデル MET に対しきわめてわずかな向上しかもたらさない．よって，カットルールのない定義には意義がないといえる．逆にいえば，カットルールの存在がインパクトスケールと障害との相関を高めることができたといえる．ddr_{20} と AUC についても図 10 と図 11 と同様の結果が見られた．

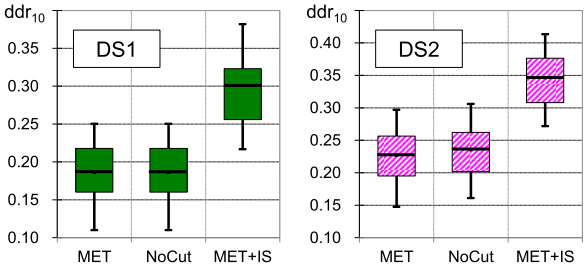


図 11 関係依存的な波及の効果
Fig. 11 Effect of relation-sensitive propagation.

結果として，仮説 2 は本稿において妥当といえる．

仮説 2（再掲）：確率的かつ関係依存的な波及を持つ波及モデルは，正確だが計算量やデータ収集コストが大きい解析の代わりとなるのに十分な予測性能を持つ．

4.3 一般化のための留意点

本節では，本稿の提案や評価結果を広く一般化する際に課題となること，留意すべきことをあげる．インパクトスケールの定義は言語非依存である．とはいえ，評価対象システムの記述言語である COBOL は他言語とは流儀もパラダイムも異なるため，本稿の結論をそのまま他の言語に拡大するには配慮を要する．たとえば，本稿では障害予測の単位を「プログラム」（関数相当）としたが，他の言語では障害予測の単位は複数の関数からなるクラスやソースファイルであることが多いだろう．また，対象システムは勘定系システムの領域に属するが，一般化のためには他の領域のシステムの評価も必要となる．

本稿の評価では，インパクトスケールのほかに採取したメトリクスが 7 つと少ない．対象言語がオブジェクト指向言語であれば既存の多くのオブジェクト指向メトリクスがあり [35]，それらの集合にインパクトスケールを加えたそのうえで意義があるかは検証を要する．また，プロセスメトリクスが採取できる状況でインパクトスケールの意義があるか，どのプロセスメトリクスに対しどの程度の代替となりうるかは未評価であるが興味深い課題である．

また，IS の測定にはコールグラフなど依存関係の抽出が必要である．もし抽出した情報の精度が低ければ，インパクトスケールの測定値は影響範囲量を正確には反映しない．インパクトスケールは要約統計量なので抽出精度に対し過敏ではないと期待されるが，本稿では動的束縛の少ない COBOL 言語で書かれたシステムを評価対象とすること

© 2013 Information Processing Society of Japan

でこの問題を極力避けたため，この影響については未評価である．

5. 関連研究

ソフトウェアのコードとその関係からなるソフトウェア構造メトリクス [2] の既存のものには Fan-in，Fan-out，LCOM，CBO [35] がある．これらは依存グラフの隣接ノード間の直接的な関係のみを考慮するものだったが，本稿の評価ではグラフ上の間接的な遠隔ノードも障害に強く影響することが分かった．よって，遠隔ノードの情報を障害予測に組み入れるメトリクスは有効といえる．

変更量に関するプロセスメトリクスは障害予測の説明変数として有効である [4], [6], [7]．インパクトスケールがそれらのプロセスメトリクスとは独立の障害要素であれば価値が高い．Cataldo ら [11] は変更量と論理的結合関係が相互に独立でかつ両方が障害数と相関があることを報告した．論理的結合関係とインパクトスケールは依存関係と影響波及といった共通要素を持つため，インパクトスケールも変更量メトリクスと独立であると推測される．

Geiger らはコードクローンと論理的結合関係の間にある程度の関係があると報告した [36]．コードクローンはコールグラフやデータ依存のみではすべてを捕捉することはできないため，論理的結合関係のすべてをインパクトスケールで捕捉することはできない．コードクローンの関係をインパクトスケールの波及グラフに加えることは容易であり（たとえば，新しい関係タイプ CLONE を追加），そのような情報の追加により予測性能の向上に寄与する余地がある．

近年は依存グラフを探索するメトリクスが現れてきている．Inoue らは，ソフトウェアコンポーネントの利用依存グラフをマルコフ連鎖モデルと見なすことで，コンポーネントの利用性の重要度を求めるコンポーネントランク [37] を提案している．早瀬らは，影響波及解析により構築した依存グラフを探索して保守工数の推定量を得る保守ポイントを提案している [38]．保守ポイントは評価値がグラフ経路上で漸減する点はインパクトスケールと共通するが，インパクトスケールは関係依存的な波及を導入していることが差異である．影響波及が確率的に起こるという仮定の下に，変更の予測を行うモデルの研究もある [18], [19]．Tsantalis らのモデル [19] は変更履歴を用いてモジュールの変更確率を予測する．このモデルは，各変更が波及によるものか起源的なものかの教師情報を人間が与える必要があるため，スケーラビリティに課題がある．

ネットワーク尺度の導入によって障害予測性能が向上するかについては Zimmermann らの提案 [12] 以後，主成分回帰を用いて追試が行われている [29], [30], [31]．Tosun ら [29] は小規模ソフトウェアと大規模ソフトウェアを対象に追試し，大規模では有効だが小規模では効果がないと報告した．主成分分析により多数の説明変数から合成される「主成分」は人間にとって解釈が難しいという欠点を持つ．一方，インパクトスケールは単一のメトリクスでありながら，障害と相関があり，解釈が容易という利点を持つ．また，ネットワーク尺度ではデータ依存などを数値に反映できないが，インパクトスケールは関係依存の波及を備えることによりネットワーク尺度では扱えない障害要素を扱うことができ，障害予測性能のさらなる向上を可能にした．

6. まとめ

本稿では，影響波及の大きさが大規模ソフトウェアにおける障害発生の要因の 1 つであると仮定して，影響波及量を定量化したメトリクス「インパクトスケール」（ImpactScale，IS）を定義した．インパクトスケールは波及の有無を確率的に扱い，関係依存的な波及探索を行うモデルを用いることを特徴とし，大規模ソフトウェアでも現実的な時間で測定でき，直観的な解釈が可能という性質を備える．

2 つの大規模企業システムを対象とする実験により，インパクトスケールを障害予測モデルに追加することで予測性能が向上するかを，適合率・再現率・F1 値と工数考慮モデルの 2 つの基準で評価した．工数考慮モデルは，障害と予測されたモジュールに必要な工数を考慮するという実践的な観点で予測性能を評価する手法である．すべての評価において，インパクトスケールを含む予測モデルは，含まないモデルに比べて障害予測性能が高いという結果が得られた．たとえば，インパクトスケールを既存のプロダクトメトリクスに追加することで，10%検査工数において 50%以上の障害検出数の向上が確認された．また，既存のプロダクトメトリクスとネットワーク尺度をあわせて用いた予測モデルにさらにインパクトスケールを追加した場合にも予測性能の向上が確認された．以上の結果は，既存の多くのメトリクスがありながら，あえて新しいメトリクス「インパクトスケール」を定義することの有用性を示している．

我々はすでに，インパクトスケールを用いた障害予測による品質管理を実施しており，たとえばある顧客の事例では，予測障害密度が上位のモジュールに集中して年間予算の 1/20 にあたる工数でレビューを施したところ，年あたりの障害発生が数件程度のシステムにおいて，8 件の欠陥を検出することができ，少ない工数で有効な予防措置を講じることができた．

今後の課題としては，Java など他言語での効果の実証を行うとともに，インパクトスケールの用途として意図されている工数予測，品質劣化の監視などにも応用を広げ実証していく予定である．

謝辞 本研究を進めるうえで多大な助言と議論をいただいた大阪大学の楠本真二教授，松下誠准教授，そしてソフトウェア工学講座のメンバ諸氏に篤く感謝する．有益な議論をいただいた NICTA（National ICT Australia）の ESE

© 2013 Information Processing Society of Japan

グループのメンバ諸氏に感謝する．評価のためのデータの提供および継続的な支援をいただいた富士通モダナイゼーション事業部の鎌倉潤一氏ならびにメンバ諸氏に深く感謝する．

参考文献

[1] Basili, V.R., Briand, L.C. and Melo, W.L.: A validation of object-oriented design metrics as quality indicators, *IEEE Trans. Softw. Eng.*, Vol.22, No.10, pp.751–761 (1996).

[2] Briand, L.C., Wüst, J., Daly, J.W. and Porter, D.V.: Exploring the relationships between design measures and software quality in object-oriented systems, *J. Systems and Software*, Vol.51, No.3, pp.245–273 (2000).

[3] Fenton, N.E. and Ohlsson, N.: Quantitative analysis of faults and failures in a complex software system, *IEEE Trans. Softw. Eng.*, Vol.26, No.8, pp.797–814 (2000).

[4] Ostrand, T.J. and Weyuker, E.J.: The distribution of faults in a large industrial software system, *Proc. ACM SIGSOFT Int'l Symp. on Software Testing and Analysis* (*ISSTA*), pp.55–64 (2002).

[5] McCabe, T.J.: A complexity measure, *IEEE Trans. Softw. Eng.*, Vol.SE-2, No.4, pp.308–320 (1976).

[6] Graves, T.L., Karr, A.F., Marron, J.S. and Siy, H.: Predicting fault incidence using software change history, *IEEE Trans. Softw. Eng.*, Vol.26, No.7, pp.653–661 (2000).

[7] Nagappan, N. and Ball, T.: Use of relative code churn measures to predict system defect density, *Proc. Int'l Conf. on Software Engineering* (*ICSE*), pp.284–292 (2005).

[8] Gall, H., Hajek, K. and Jazayeri, M.: Detection of logical coupling based on product release history, *Proc. IEEE Int'l Conf. on Software Maintenance* (*ICSM*), pp.190–198 (1998).

[9] Hassan, A.E. and Holt, R.C.: Predicting change propagation in software systems, *Proc. IEEE Int'l Conf. on Software Maintenance* (*ICSM*), pp.284–293 (2004).

[10] Haney, F.M.: Module connection analysis — a tool for scheduling software debugging activities, *Proc. Fall Joint Computer Conference*, pp.173–180 (1972).

[11] Cataldo, M., Mockus, A., Roberts, J.A. and Herbsleb, J.D.: Software dependencies, work dependencies, and their impact on failures, *IEEE Trans. Softw. Eng.*, Vol.36, No.2, pp.864–878 (2009).

[12] Zimmermann, T. and Nagappan, N.: Predicting defects using network analysis on dependency graphs, *Proc. Int'l Conf. on Software Engineering* (*ICSE*), pp.531–540 (2008).

[13] Bohner, S.A. and Arnold, R.S. (Eds.): Software change impact analysis, Bohner, S.A. and Arnold, R.S.: *An introduction to software change impact analysis*, pp.1–26, IEEE Computer Society Press (1996).

[14] Grove, D. and Chambers, C.: A framework for call graph construction algorithms, *ACM Trans. Progr. Lang. Syst.*, Vol.23, No.6, pp.685–746 (2001).

[15] Law, J. and Rothermel, G.: Whole program path-based dynamic impact analysis, *Proc. Int'l Conf. on Software Engineering* (*ICSE*), pp.308–318 (2003).

[16] Ren, X., Shah, F., Tip, F., Ryder, B.G. and Chesley, O.: Chianti: a tool for change impact analysis of Java programs, *Proc. Conf. on Object-Oriented Prog., Syst., Lang., and App.* (*OOPSLA*), pp.432–448 (2004).

[17] Kobayashi, K., Matsuo, A., Inoue, K., Hayase, Y., Kamimura, M. and Yoshino, T.: ImpactScale: Quantifying Change Impact to Predict Faults in Large Software Systems, *Proc. Int'l Conf. on Software Maintenance* (*ICSM*), pp.43–52 (2011).

[18] Sharafat, A.R. and Tahvildari, L.: A probabilistic approach to predict changes in object-oriented software systems, *Proc. European Conf. on Software Maintenance and Reengineering* (*CMSR*), pp.27–38 (2007).

[19] Tsantalis, N., Chatzigeorgiou, A. and Stephanides, G.: Predicting the probability of change in object-oriented systems, *IEEE Trans. Softw. Eng.*, Vol.31, No.7, pp.601–614 (2005).

[20] Whaley, J. and Lam, M.S.: Cloning-based context-sensitive pointer alias analysis using binary decision diagrams, *Proc. ACM SIGPLAN Conf. on Prog. Lang. Design and Impl.* (*PLDI*), pp.131–144 (2004).

[21] Mende, T. and Koschke, R.: Effort-aware defect prediction models, *Proc. European Conf. on Software Maintenance and Reengineering* (*CSMR*), pp.107–116 (2010).

[22] Akaike, H.: A new look at the statistical model identification, *IEEE Trans. Automatic Control*, Vol.AC-19, No.6, pp.716–723 (1974).

[23] Chatterjee, S. and Hadi, A.S.: *Regression analysis by example, 4th Edition*, John Wiley and Sons (2006).

[24] Arisholm, E. and Briand, L.C.: Predicting fault-prone components in a Java legacy system, *Proc. ACM/IEEE Int'l Symp. on Empirical Software Engineering* (*ISESE*), pp.8–17 (2006).

[25] Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y. and Bener, A.: Defect prediction from static code features: current results, limitations, new approaches, *J. Automated Software Engineering*, Vol.17, pp.375–407 (2010).

[26] Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K., Adams, B. and Hassan, A.E.: Revisiting common bug prediction findings using effort-aware models, *Proc. IEEE Int'l Conf. on Software Maintenance* (*ICSM*), pp.1–10 (2010).

[27] Cameron, A.C. and Trivedi, P.K.: *Regression analysis of count data*, Cambridge University Press (1998).

[28] Khoshgoftaar, T.M., Geleyn, E. and Gao, K.: An empirical study of the impact of count models predictions on module-order models, *Proc. IEEE Int'l Software Metrics Symp.* (*METRICS*), pp.161–172 (2002).

[29] Tosun, A., Turhan, B. and Bener, A.: Validation of network measures as indicators of defective modules in software systems, *Proc. Int'l Conf. on Predictor Models in Software Engineering* (*PROMISE*), pp.5:1–5:9 (2009).

[30] Nguyen, T., Adams, B. and Hassan, A.: Studying the impact of dependency network measures on software quality, *Proc. IEEE Int'l Conf. on Software Maintenance* (*ICSM*), pp.1–10 (2010).

[31] Premraj, R. and Herzig, K.: Network Versus Code Metrics to Predict Defects: A Replication Study, *Proc. Int'l Sympo. on Empirical Software Engineering and Measurement* (*ESEM*), pp.215–224 (2011).

[32] Borgatti, S.P., Everett, M.G. and Freeman, L.C.: *UCINet for Windows: Software for social network analysis*, Analytic Technologies (2002).

[33] Hanneman, R.A. and Riddle, M.: *Introduction to social network methods*, University of California, Riverside (2005).

[34] Jolliffe, I.T.: *Principal component analysis, 2nd edition*, Springer (2002).

© 2013 Information Processing Society of Japan

[35] Chidamber, S.R. and Kemerer, C.K.: A metrics suite for object oriented design, *IEEE Trans. Softw. Eng.*, Vol.20, No.6, pp.476–493 (1994).
[36] Geiger, R., Fluri, B., Gall, H. and Pinzger, M.: Relation of code clones and change couplings, *Int'l Conf. of Fundamental Approaches to Software Engineering* (*FASE*), LNCS 3922, pp.411–425, Springer (2006).
[37] Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M. and Kusumoto, S.: Ranking significance of software components based on use relations, *IEEE Trans. Softw. Eng.*, Vol.31, No.3, pp.213–225 (2005).
[38] 早瀬康裕，松下　誠，楠本真二，井上克郎，小林健一，吉野利明：影響波及解析を利用した保守作業の労力見積りに用いるメトリックスの提案，電子情報通信学会論文誌 D，Vol.J90-D, No.10, pp.2736–2745 (2007).



小林 健一 （正会員）

平成 4 年東京大学工学部計数工学科卒業．平成 6 年同大学大学院工学系研究科情報工学専攻修士課程修了．同年富士通研究所入社．現在まで富士通株式会社および同研究所に勤務．ソフトウェア工学（特にソフトウェア保守，エンピリカルソフトウェア工学），データマイニング，HPC の研究に従事．



松尾 昭彦 （正会員）

昭和 62 年東京理科大学物理学科卒業．同年富士通研究所入社．ソフトウェアシステム研究所所属．ソフトウェア保守効率化技術（特にソフトウェアテスト技術，仕様抽出技術，影響検索技術），クラウド間連携技術，コンピュータグラフィックスの研究に従事．



井上 克郎 （フェロー）

昭和 59 年大阪大学大学院基礎工学研究科博士後期課程修了（工学博士）．同年大阪大学基礎工学部情報工学科助手．昭和 59〜61 年ハワイ大学マノア校コンピュータサイエンス学科助教授．平成 3 年大阪大学基礎工学部助教授．平成 7 年同学部教授．平成 14 年大阪大学大学院情報科学研究科教授．平成 23 年 8 月より大阪大学大学院情報科学研究科研究科長．ソフトウェア工学，特にコードクローンやコード検索等のプログラム分析や再利用技術の研究に従事．



早瀬 康裕 （正会員）

平成 14 年大阪大学基礎工学部情報科学科卒業．平成 19 年同大学大学院博士後期課程修了．同年同大学特任助教．平成 22 年東洋大学総合情報学部助教．平成 23 年筑波大学システム情報系助教．博士（情報科学）．オープンソースソフトウェア開発，ソフトウェア保守の研究に従事．IEEE 会員．



上村 学

平成 12 年上智大学理工学部機械工学科卒業，平成 14 年同大学大学院博士前期課程修了．同年富士通研究所入社．平成 23 年上智大学大学院理工学研究科理工学専攻情報学領域博士後期課程修了．博士（工学）．平成 24 年ブリティッシュコロンビア大学客員研究員．現在，富士通研究所でソフトウェア工学，モデリング，プログラム分析の研究に従事．



吉野 利明 （正会員）

昭和 55 年九州工業大学工学部情報工学科卒業，昭和 57 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．同年富士通研究所入社．平成 21 年富士通株式会社に転籍，フィールド・イノベータ．ソフトウェア工学，エージェント技術，自然言語処理，データベース開発，ナレッジマネージメントの研究に従事．平成 4〜8 年情報処理学会誌編集委員．

© 2013 Information Processing Society of Japan