論　文

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

早瀬　康裕[†a)]　松下　誠[†]　楠本　真二[†]　井上　克郎[†]
小林　健一[††]　吉野　利明[††]

A Metric for Estimating Maintenance Effort Based on Change Impact Analysis

Yasuhiro HAYASE[†a)], Makoto MATSUSHITA[†], Shinji KUSUMOTO[†], Katsuro INOUE[†], Kenichi KOBAYASHI[††], and Toshiaki YOSHINO[††]

あらまし　ソフトウェア保守の労力を見積もる際には，客観的な基準を用いずに，熟練者が経験に基づいて見積りを行うことが多かった．本論文では，ソースコードを対象とした影響波及解析手法を用いて，保守作業の労力を見積もるメトリックスを提案する．メトリックスの値は，保守作業によって影響を受ける部分のソースコードの複雑度に重みをつけて足し合わせることで求める．実際に保守作業を行う実験によって提案したメトリックスを評価した結果，提案したメトリックスは，既存のメトリックスに比べて，労力と高い相関をもつことが確認できた．

キーワード　メトリックス，労力見積り，保守，影響波及解析

1. まえがき

ソフトウェア保守とは，文献 [1] では，“納入後，(ソフトウェア）プロダクトに対して加えられる，欠陥修正，性能またはほかの性質改善，変更された環境に対するプロダクトの適応のための改訂” と定義されている．そして，保守作業を，修正保守（発見された欠陥を取り除く），適応保守（環境が変化した際にプロダクトを利用可能に保つ），完全化保守（性能または保守性を改善する），予防保守（潜在的な欠陥が顕在化する前に対処する）のいずれかに分類している．一般に，長期間にわたって使用されるソフトウェアでは，総所有コストのうち保守コストが大きな割合を占めることが知られている [2], [3]．そのため，保守作業に必要な労力を早い段階で見積もることが求められている．

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻，豊中市
Graduate School of Information Science and Technology, Osaka University, 1–3 Machikaneyama-cho, Toyonaka-shi, 560–8531 Japan
†† (株) 富士通研究所ソフトウェア&ソリューション研究所ソフトウェアイノベーション研究部，川崎市
Fujitsu Laboratories Limited, Kawasaki-shi, 211–8588 Japan
a) E-mail: y-hayase@ist.osaka-u.ac.jp

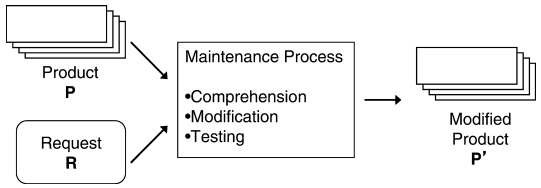


図 1　保守作業の概念
Fig. 1　Overview of maintenance.

一般に，保守作業は図 1 のような手順で行われる [4]．まず，既存のプロダクト P に対し，環境の変化や機能追加の要求，障害の発生により，変更要求 R が発生する．次に，その変更を行った場合の影響を調査し，実際に変更し，テストを行って変更によって欠陥が作り込まれていないことを確認した後に，プロダクト P' をリリースする．

現在，保守作業の見積りは，熟練者の経験に基づいて行われたり，新規開発の見積りと同様の手法が用いられたりすることが多い．しかし，熟練者による見積りは，見積りを行う人によって結果が異なるという問題がある．また，新規開発の見積りは要求仕様からファンクションポイント [5] などの機能量などを計算

することで作業量を予測するが，保守作業は要求仕様から影響を受けるだけではなく，既存のプロダクトの品質や複雑さなどからも大きな影響を受けるため，新規開発向けの見積り手法をそのまま適用するのも適切でない．

見積りに用いるべきメトリックス値と，見積りに用いる計算手法について，Jørgensen [6] は，実際の保守作業で様々なメトリックス値を収集し，見積りに有効なメトリックスと見積り手法を調べた結果，変更行数と工数との相関が高いことを明らかにした．Sneed [7] は，保守による影響範囲の大きさを様々な要因で補正して，工数を見積もる手法を提案した．Fioravanti ら [8] は，適応保守の見積りに有用なメトリックスを提案し，実際に見積りを行った．これらの手法は，保守作業の早い段階で得ることができない変更の具体的な内容を用いたり，適用対象が適応保守に限られたりする問題があった．Ramanujan ら [9] 数十行から百行程度の小規模なプログラムを対象とした保守作業にかかる時間と，様々な測定値との関係について調査した．その結果，被験者の経験，プログラムの規模と複雑度，納期の短さと作業時間には相関があることが分かったが，被験者の知識が少なくプログラムの規模が小さい場合には複雑度と作業時間には有意な相関が示されなかった．

そこで我々は，ソースコードと変更要求のみから計算できるメトリックスを提案する．ソースコードと変更要求は保守作業の早い段階で得られる情報であるため，様々な保守作業の見積りにメトリックスを使用することができる．提案するメトリックスは，プロダクトを構成する各モジュールの変更に必要な労力と，モジュール間の関係の複雑度を反映している．メトリックスを計算するために，プロダクトを有向グラフを用いてモデル化する．

本メトリックスを評価するために，ある一定の条件のもとで実際に保守作業を行う実験を行い，提案するメトリックスや既存のメトリックスと労力との関係を調べた．その結果，提案するメトリックスの有効性を確認した．

以降，**2.** で，保守対象のプロダクトをモデル化し，メトリックスとその計算方法について説明する．**3.** では提案したメトリックスの評価実験について述べる．**4.** で実験やメトリックスの計算方法について議論し，最後に，**5.** でまとめと今後の課題について説明する．

2. 保守ポイント

本章では，プロダクトと変更要求をモデル化し，保守労力の見積りに用いることのできるメトリックス「保守ポイント」を定義する．

2.1 プロダクトモデル

一般に，保守対象のプロダクトは複数のモジュールから構成される．このプロダクトを，有向グラフを用いてモデル化したものをプロダクトモデル $G_P = (M, I)$ と呼ぶ．プロダクトモデルの例を 図 2 に示す．M はモジュールの集合，I は有向辺の集合で影響波及関係を表す．モジュール m1 から m2 への有向辺は，モジュール m1 に何らかの変更作業を行った場合に，モジュール m2 にも作業が発生することを表す．

各有向辺 s は，影響の強さを表す重み $w(s)$（0 以上 1 以下であり，大きな値ほど強い影響を表す）をもつ．また，各頂点 m もモジュール単体の保守作業の労力を表す値 $\mathrm{eff}(m)$（0 以上）をもつ．

保守によって直接の変更作業を行う必要のあるモジュールの集合を，変更要求 R $(R \subseteq M)$ という．

2.2 保守ポイントの定義

プロダクトモデル G_P と変更要求 R から得た影響範囲の情報を用いて，保守作業の労力の指標「保守ポイント」を定義する．

プロダクトモデル G_P 上の経路 p を構成する辺の系列を $s_1, s_2, \ldots, s_i$ とする．今，p の重み $v(p)$ を次のように定義する．

$$v(p) = \prod_{j=1}^{i} w(s_j)$$

次に，m_1 が変更されたときに，m_2 にどの程度の作業が発生するかを表す値である影響度 $\mathrm{imp}_m(m_1, m_2)$

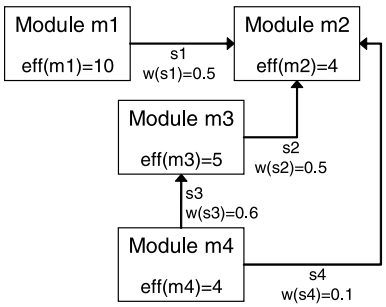


図 2　プロダクトモデルの例
Fig. 2　Example of the product model.

を以下のように定義する．

$$\mathrm{imp}_m(m_1, m_2) = \begin{cases} 1 \ \ (m_1 = m_2 \text{ のとき}) \\ 0 \ \ (Q(m_1, m_2) = \phi \text{ とき}) \\ \max(\{v(p) | p \in Q(m_1, m_2)\}) \\ \quad (\text{それ以外のとき}) \end{cases}$$

（ただし，$Q(m_1, m_2)$ は m_1 から m_2 までの全経路の集合とする）

影響度は，そのモジュールを理解・テストしなければならない割合の期待値を表す．影響度が 1 の場合は，保守作業でそのモジュールを完全に理解し，テストしなければならない．

また，変更要求 R が与えられたとき，頂点 m が受ける影響度 $\mathrm{imp}(R, m)$ を以下のように定める．

$$\mathrm{imp}(R, m) = 1 - \prod_{r \in R} \{1 - \mathrm{imp}_m(r, m)\}$$

$\{1 - \mathrm{imp}_m(r, m)\}$ は，モジュール r を変更するとき，モジュール m のうち理解・テストしなくてよい割合を表す．上記の式は，多くのモジュールから影響を受けるほど，理解・テストしなければならない割合が増えることを表している．

今，$G_P = (M, I)$, $M = \{m_1, m_2, \ldots, m_n\}$ に対し，保守ポイント $C(G_P, R)$ は次の式で与えられる．

$$C(G_P, R) = \sum_{i=1}^{n} \{\mathrm{eff}(m_i)\mathrm{imp}(R, m_i)\}$$

この保守ポイントは，R に対して，各モジュールの変更に必要な労力の総和を示している．ここで各モジュールの労力 eff を理解，変更，テストなどの部分作業の労力として与えれば，プロダクト全体に対する部分作業に要する労力の総和が得られる．

保守ポイントの値は，G_P と R が明らかになった時点，すなわち保守作業によってどのモジュールを変更するのかを決定した時点で計測可能となる．このため，R の決定が比較的容易である修正保守や適応保守，完全化保守の見積りに用いることができよう．

保守ポイントの値は保守作業による影響範囲内の労力を反映した値であるため，実際に見積りを行う際には，保守作業者の生産性や設計文書の整備状態などを考慮する必要がある．

実際のプロダクトとプロダクトモデルを対応づけるには，頂点（モジュール）と辺（影響波及関係）に何を用いるかを決定する必要がある．本研究の評価実験（**3.**）では比較的簡単に求められるものを頂点と辺にしたが，**4. 2** で述べるように，他の選択も多数ある．

2. 3 計 算 例

図 2 のプロダクトモデル上での，保守ポイントの計算例を示す．

変更要求 $R = \{\mathtt{m1}, \mathtt{m4}\}$ の場合を考える．モジュール m4 の変更により影響を受けるのは，モジュール m3 とモジュール m2 である．モジュール m4 から m2 への経路は，経路 1：s4 と，経路 2：s3，s2 の二つが存在する．経路 1 の影響度は 0.1，経路 2 は $0.6 \times 0.5 = 0.3$ で，経路 2 の方が大きい．同様に，モジュール m1 から影響を受けるのは m2 であり，経路 s1 の影響度は 0.5 である．

よって，この場合の保守ポイントは，以下の計算で与えられる．

$$\begin{aligned} & C(G_P, \{\mathtt{m1}, \mathtt{m4}\}) \\ & \quad = \mathrm{eff}(\mathtt{m1})\mathrm{imp}(\{\mathtt{m1}, \mathtt{m4}\}, \mathtt{m1}) \\ & \qquad + \mathrm{eff}(\mathtt{m2})\mathrm{imp}(\{\mathtt{m1}, \mathtt{m4}\}, \mathtt{m2}) \\ & \qquad + \mathrm{eff}(\mathtt{m3})\mathrm{imp}(\{\mathtt{m1}, \mathtt{m4}\}, \mathtt{m3}) \\ & \qquad + \mathrm{eff}(\mathtt{m4})\mathrm{imp}(\{\mathtt{m1}, \mathtt{m4}\}, \mathtt{m4}) \\ & \quad = 10 \times \{1 - (1 - 1)(1 - 0)\} \\ & \qquad + 4 \times \{1 - (1 - 0.5)(1 - 0.3)\} \\ & \qquad + 5 \times \{1 - (1 - 0)(1 - 0.6)\} \\ & \qquad + 4 \times \{1 - (1 - 0)(1 - 1)\} \\ & \quad = 19.6 \end{aligned}$$

3. 評 価 実 験

保守ポイントと保守作業の労力の相関を調べるために実験を行った．実験では被験者があるソフトウェアに対して実際に保守作業を行い，その作業に要した労力と保守ポイントとの関係を分析した．本実験で実施した保守作業は，欠陥に対する修正保守と機能追加にかかわる完全化保守である．

被験者は，大阪大学大学院情報科学研究科の博士前期課程に所属する 6 人である．保守の対象となるソフトウェアや，作業内容は以下のようになっている．

［保守対象のソフトウェア］

保守作業の対象として，酒屋問題 [10] の仕様に基づいたソフトウェアを用いた．これは，酒屋の倉庫への

商品の入庫や出庫を管理することを目的としたソフトウェアであり，コマンドラインインタフェースをもつ．実装には Java を用い，8 クラス，25 メソッド，309 行の規模である．

本実験の保守作業は修正保守を含んでいるため，保守作業で取り除くための潜在的な欠陥を 2 個埋め込んでおいた．

［課題］

被験者に与えた課題は，ソフトウェアに埋め込まれた欠陥修正の作業 D1，D2 の二つと，機能追加を行う作業 E1〜E4 の四つの，計六つである．

このうち，課題 E1 は保守対象のソフトウェアに慣れさせるための練習として用いる．そのため，評価に用いる課題は D1，D2，E2，E3，E4 の五つである．

課題の概要は以下のようなものである．

- D1：ライブラリ使用法の誤りの修正
- D2：関連するモジュールが仮定している条件が矛盾している誤りの修正
- E1：酒屋の倉庫の中を表示するコマンドの追加
- E2：倉庫を表すデータ構造の変更
- E3：処理順序の変更
- E4：既存のコマンドの拡張

被験者に対する課題の作業内容の指示は以下のように行った．欠陥修正の課題では，保守対象のソフトウェアへの入力と異常な出力を被験者に示し，正しい出力をするようにソフトウェアを変更するよう指示した．機能追加の課題では，新しい機能の要求仕様として，自然文による機能の説明と，その機能を実装したときの入出力の例とを与えた．

欠陥修正と機能追加のどちらの作業でも，どのモジュールを変更するかは指示せず，被験者の判断に委ねた．保守ポイントの計算モデルでは変更要求は外から与えられることとなっているが，本実験は変更するモジュールと労力との関連を調べることが目的であるため，変更するモジュールを指示しないことは問題とはならない．

［作業順序］

まず最初に，保守対象のソフトウェアに慣れるために，被験者全員が機能追加課題 E1 を行った．この作業に要した時間は実験の評価に用いない．

次に 2 件の欠陥修正の課題を行った．慣れの影響を除くため，半分の被験者は D1 を先に行い，残りの半分の被験者は D2 を先に行った．

最後に，E2〜E4 の 3 件の機能追加の課題を行った．欠陥修正と同様に，慣れの影響を除くため，被験者ごとにすべて異なる順序で作業を行った．

課題の作業対象は，直前の課題の作業結果のソースコードである．

［メトリックス値の計測］

プロダクトモデルとの対応付けは以下のように行った．

- 頂点（モジュール）：メソッド
- 頂点の労力：LOC と McCabe のサイクロマチック数の 2 通り
- 辺（影響波及関係）：メソッド間の呼出しと，フィールド変数を介したデータ依存関係
- 辺の重み：各辺同じ 0 から 1 まで 0.05 刻みの 21 通りの値

図 3 に，保守対象ソフトウェアの課題 E1 を行う前の状態から作成したプロダクトモデルを示す．頂点の労力としては，LOC と McCabe のサイクロマチック数の 2 通りを計測するので，二つを併記した．有向辺の重みは 21 通りの場合を用いるので，表記していない．

変更要求 R は，被験者が課題の内容を理解した後に直接変更する必要があると判断したメソッドの集合を用いた．

以下，サイクロマチック数を用いた保守ポイントを，保守ポイント [McCabe] と呼び，LOC を用いた保守ポイントを，保守ポイント [LOC] と呼ぶことにする．

［作業時間の計測］

被験者が作業に要した時間として，被験者に課題を渡してから，プログラムが正しい入出力を行うことを確認するまでの時間を用いた．

3.1 実験結果

各被験者が五つの作業を終えるのに要した時間は，最短で 1 時間 43 分，最長で 5 時間 2 分となり，被験者ごとに大きな差があった．このため，作業時間を正規化した値を労力を表す値として用いた．この値を正規化労力と呼ぶ．正規化労力は，各作業に要した時間を五つの作業全体に要した時間で割った値である．

また，30 件の作業のうち 9 件において，平均 24.2 行（標準偏差 23.8）の新しいモジュールの導入がなされていた．

3.1.1 労力と保守ポイントの相関

保守ポイントの値と正規化労力との相関と，辺の重みとの関係を図 4 に示す．すべての辺の重さの場合において，保守ポイント [McCabe] は保守ポイント [LOC]

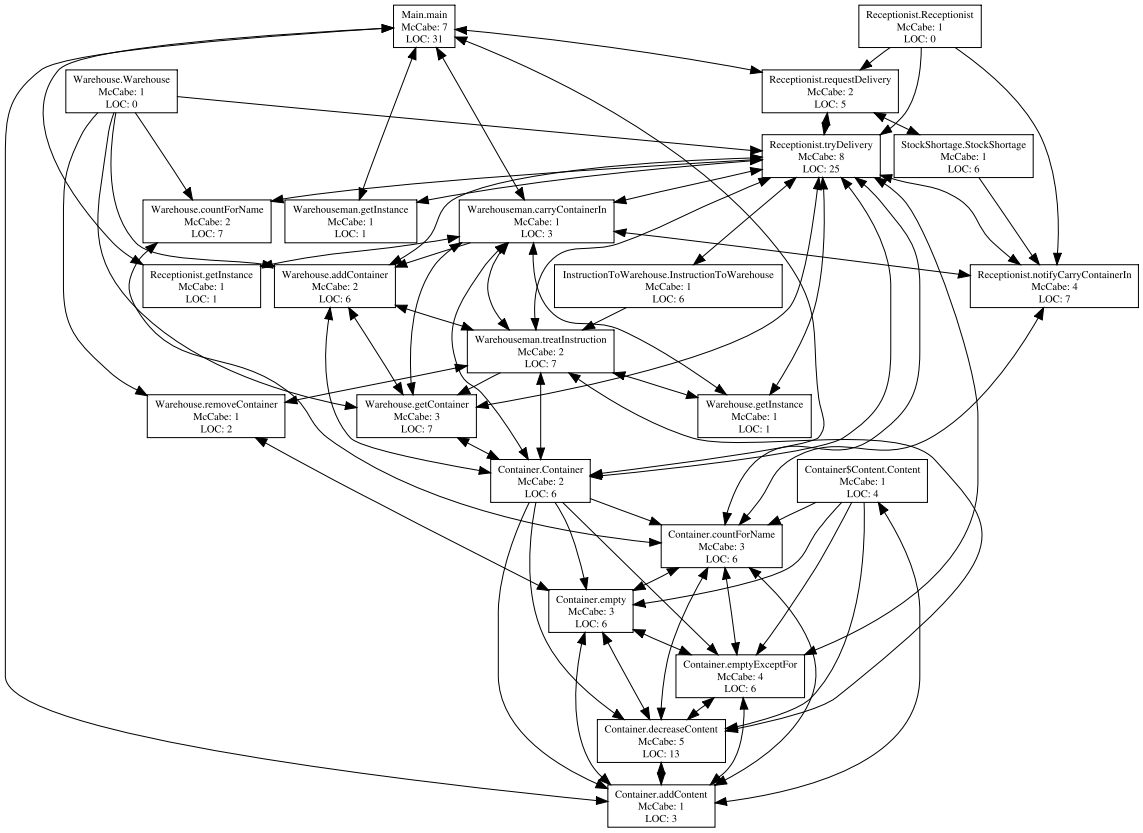


図 3　作業対象のプロダクトモデル
Fig. 3　The product model of the maintenance target.

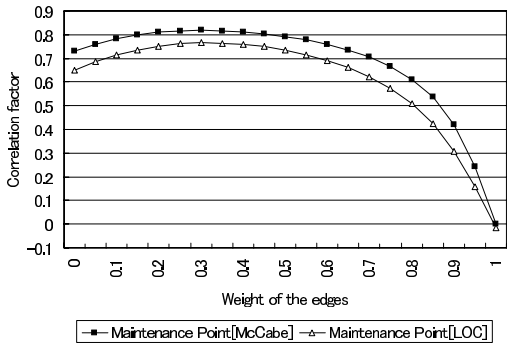


図 4　辺の重みを変化させたときの保守ポイントと正規化労力との相関
Fig. 4　Correlation between maintenance point and normalized effort.

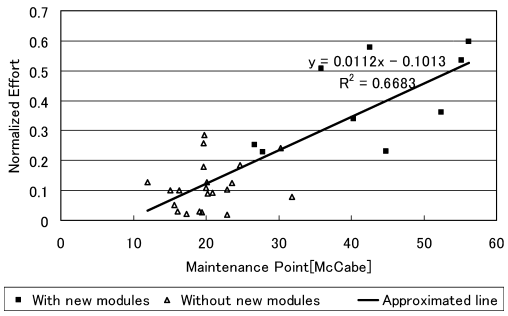


図 5　辺の重みを 0.3 とした場合の保守ポイント [McCabe] と正規化労力の散布図
Fig. 5　Scatter plot between Maintenance Point [McCabe] and normalized effort when the weight of the edges is 0.3.

よりも高い相関を示した．保守ポイント [McCabe] と保守ポイント [LOC] の両方で，辺の重みが 0.3 付近で相関が最大となり，辺の重みが 0.7 以上である場合の相関は，辺の重み 0 のときの相関よりも小さくなった．辺の重みを 1 としたときの相関はほぼ 0 であった．

相関が最大となる辺の重みが 0.3 の場合の，保守ポイント [McCabe] と正規化労力の散布図を図 5 に示す．■ は新規モジュールが追加されていた場合，△ は

表 1　正規化労力と各メトリックスの相関
Table 1　Correlation between effort and metrics.

	保守ポイント [McCabe]	単純和 [McCabe]	保守ポイント [LOC]	単純和 [LOC]	変更行数
正規化労力との相関 (p 値)	0.82 (<0.01)	0.73 (<0.01)	0.77 (<0.01)	0.65 (<0.01)	0.87 (<0.01)
符号付順位和検定					
二乗誤差 (p 値)	101(<0.01)		95 (<0.01)		
MRE (p 値)	105(<0.01)		86 (<0.01)		

新規モジュールの追加がなかった場合のデータである．新規モジュールが追加されるのは，正規化労力と保守ポイントの両方が大きい場合であることが分かる．直線は最小二乗法によって求めた一次関数による近似である．**4.1** で述べるように，新規モジュールの追加にかかる労力は大きくないため，新規モジュールが追加された場合とそうでない場合とで近似直線を分ける必要はないと考えられる．

3.1.2　既存のメトリックスとの比較

次に，保守ポイントを既存のメトリックスと比較する．比較に用いる保守ポイントの値は，すべて辺の重みが 0.3 のときのものである．

一つ目の比較対象として，保守により変更されるメソッドの変更前の複雑度を，単純に足し合わせたものを用いた．複雑度には，保守ポイントと同じく，McCabe のサイクロマチック数と LOC の 2 種類を用いた．以下，サイクロマチック数を足し合わせたメトリックスを，単純和 [McCabe] と呼び，LOC を足し合わせたメトリックスを，単純和 [LOC] と呼ぶことにする．単純和は，すべての辺の重みを 0 としたときの保守ポイントと同じである．

二つ目の比較対象として，Jørgensen の提案した見積り手法 [6] で用いられている，作業によって書き換えられた行数（変更行数）を用いた．変更行数は労力との相関が高いことが知られているが，保守作業の早い段階で計測することができないため，見積りに用いるのは困難である．

表 1 に取得したメトリックスと，正規化労力との相関を示す．正規化労力との相関が最も高かったのは変更行数であり，次に正規化労力との相関が高かったのは保守ポイント [McCabe] であった．また，モジュールの複雑度に用いたメトリックスが同じ場合，単純和よりも保守ポイントの方が，高い相関を示した．

次に，保守ポイントの正規化労力見積りの誤差が，単純和に比べて有意に小さいかを調査した．調査は，保守ポイント [McCabe] と単純和 [McCabe] の組 T_M と，保守ポイント [LOC] と単純和 [LOC] の組 T_L について行った．

まず，それぞれのメトリックスについて，一次式による労力予測式を最小二乗法を用いて作成した．そして，それぞれの組について，予測式による二乗誤差と，メトリックスの評価で用いられる相対誤差 $\left(= \left|\frac{\text{見積り値} - \text{実測値}}{\text{実測値}}\right|\right.$．以下 MRE と表記する $\left.\right)$ の値に有意な差があるかを調べた．二乗誤差や相対誤差の値は正規分布とならないため，平均値の検定ではなく，ウィルコクソンの符号付順位和検定を用いて片側検定を行った．

その結果，表 1 に示したように，T_M と T_L 両方の組に対して，二乗誤差と MRE のどちらでも有意水準 1% で，保守ポイントの誤差が小さいといえることが分かった．

更に，計測した 30 件のデータをランダムに 10 件ずつの 3 群に分割し，クロスバリデーションを行った．見積り式には一次式を用い，最小二乗法でパラメータを決定した．評価項目としては，メトリックスの評価として頻繁に用いられる以下の四つを用いた．

- MAE：絶対誤差の平均
- MMRE：MRE（相対誤差）の平均
- PRED(25)：相対誤差が 25% 以下である割合
- PRED(50)：相対誤差が 50% 以下である割合

MAE と MMRE の値は小さいほど，PRED(25) と PRED(50) は大きいほど見積りが正確であることを示す．

クロスバリデーションの結果を 表 2 に示す．表の各行のうち，太字で書かれた列が最良の結果であったことを表す．MMRE では変更行数が最良であったが，次に良いのは保守ポイント [McCabe] であった．また，PRED(25) では保守ポイント [McCabe] と変更行数が，その他の評価項目では保守ポイント [McCabe] が最良となり，保守ポイント [McCabe] の見積り精度が高いことが分かった．

表 2　クロスバリデーションの結果
Table 2　Result of the cross validation.

	保守ポイント [McCabe]	単純和 [McCabe]	保守ポイント [LOC]	単純和 [LOC]	変更行数
MAE	**0.07840**	0.09270	0.09405	0.1177	0.07868
MMRE	0.9616	1.427	1.263	1.766	**0.8266**
PRED(25)	**0.3667**	0.3000	0.3333	0.1667	**0.3667**
PRED(50)	**0.6667**	0.6000	0.6000	0.5000	0.6000

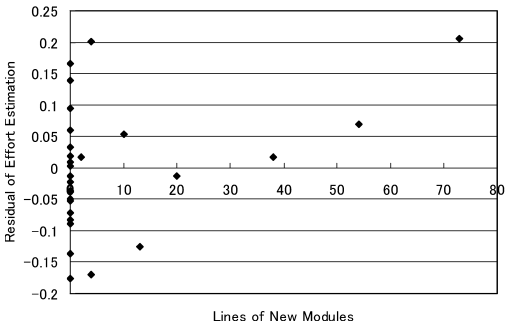


図 6　保守ポイント [McCabe] による見積り残差と新モジュールの行数の散布図
Fig. 6　Scatter plot between residual of effort estimation using Maintenance Point [McCabe] and lines of new modules.

3.1.3　新規モジュールが正規化労力に与える影響

保守ポイントは新規に追加されるモジュールの量を考慮していない．そこで，新規モジュールを考慮することで，正規化労力の見積り精度がどの程度改善するかを調査した．

まず，辺の重みを 0.3 とした保守ポイント [McCabe] の影響を除いた，新規モジュールの行数と正規化労力との偏相関係数は 0.4903 であった．

3.1.1 で行った保守ポイント [McCabe] による見積りの残差と，新規モジュールの行数との散布図を，図 6 に示す．新規モジュールが追加された場合と追加されない場合とで，残差が同程度に分布していることが分かる．

以上のことから，今回の実験では新規モジュールの量が正規化労力に与える影響は大きくなかったといえる．

3.2　考　　察

実験結果が示すように，保守ポイントは既存のメトリックスである単純和よりも高い労力との相関をもっており，より高い精度で労力を予測することができると分かった．

Sneed の提案した保守コスト見積り手法 [7] は，保守労力見積りの主な基準として影響範囲の大きさを用いる．影響範囲の大きさは辺の重みを 1 としたときの保守ポイント [LOC] の値と等しいが，**3.1.1** で示したとおり，辺の重みが 1 のときの保守ポイントと労力との相関はほぼ 0 である．

4.　議　　論

4.1　モデルの妥当性

保守ポイントは，プロダクトモデルと変更要求をもとに計算する．一般に，ソフトウェア保守では理解とテストの労力が大きいといわれており [11]〜[14]，その労力が影響範囲に強い影響を受けることが分かっている [15], [16]．そのため，保守作業の労力は影響範囲内の複雑さから大きな影響を受けると考えられる．

保守ポイントは，保守作業によって変更されるモジュールから計算されるので，保守作業によって新規に追加されるモジュールを考慮していない．そのため，新規モジュールの開発に大きな労力が割かれる保守案件の労力を保守ポイントだけで見積もるのは適当でない．このような場合は，新規モジュールの開発に要する労力を機能量などで見積もり，保守ポイントを用いて見積もった既存モジュールの変更に要する労力を足し合わせることで，全体の労力を求めるのが適当であると考えられる．

しかし，本論文で行った実験では，新規モジュールのサイズを考慮しても見積り精度はあまり改善しないという結果になった．これは，新規モジュールを開発するためには，新規モジュールを利用する既存のモジュールを理解し，テストする必要があり，その理解及びテストの労力が新規モジュールの開発にかかる労力の大部分を占めるためであると考えられる．

4.2　プロダクトモデルの対応付け

プロダクトとプロダクトモデルの対応付けには，**3.** で示したもの以外にも，多くの方法が考えられる．

モジュールの候補としては，プログラミング言語により様々なものが考えられる．例えば Java の場合，メソッド，クラス，パッケージといった階層があり，モ

ジュールの候補となり得る．モジュールの保守作業の労力には，複雑度メトリックスを用いる．

影響波及関係の計算方法として，いろいろなものが考えられる．例えば，モジュール m_1 から m_2 への呼出しの数を k，外にデータフローを起こす変数の数を l としたとき，$1-\alpha^k\beta^l$（α と β は 0 より大きく 1 より小さい定数）を重みとする．この式は，変数 k 及び l の定義域を 0 以上の整数とした場合，値域は 0 以上 1 未満となり，k と l それぞれについて単調増加関数となる．k と l が無限に増加していけば，この式の値は限りなく 1 に近づく．α，β の値は，過去に計測したメトリックス値と労力との相関が最大となるように決定する．

また，CVS リポジトリなどから細粒度の変更履歴が入手可能な場合には，同時更新傾向に基づいた影響波及関係の推測手法である Logical Coupling [17] を影響波及関係として用いることもできる．

モジュールとその労力，影響波及関係の選択は，対象とするプロダクトの規模やドメインを主な基準として，メトリックスの計測者が行う．このとき，過去の類似プロダクトと同じ条件で計測することで，回帰分析などを用いた労力見積りが可能となる．

4.3 実験の設定

我々の行った実験は，実際のソフトウェア開発現場に比べて以下のような違いがある．

（1） 被験者が少ない

（2） ソフトウェアの規模が小さい

（3） 被験者の能力が，実務者に比べて低い可能性がある

（4） 被験者の能力にばらつきがある

（1）と（2）については，今後，より大規模な実験を行う必要があるだろう．

（3）については，実験問題は特殊な実務経験を必要としない一般的な問題であり，被験者は情報科学科の課程を修めているため，問題に対して十分な能力を有していると思われる．

（4）の能力のばらつきは，実際のソフトウェア開発現場でも存在するものである．また，労力を計算する際に個人の作業能力で正規化しているため，実験結果からは排除されている．

4.4 具体的な変更要求が得られないときのメトリックス

保守ポイントを求めるためには，プロダクトモデル G_P と変更要求 R が必要である．しかし，ソフトウェア保守を包括的に請負うときなどのように，どのような作業が発生するかが分からないときには，具体的な R を求めるのは非常に困難である．

そこで，R の代わりに保守要求 R の確率分布 R_d を用いた，以下のような労力指標モデル $C_{\mathrm{gen}}(G_P, R_d)$ を考える．

$$C_{\mathrm{gen}}(G_P, R_d) = \sum_{i=1}^{n} \{q_i C(G_P, \{m_i\})\}$$

（ただし，$R_d = (q_1, q_2, \ldots, q_n)$（n は G_P の頂点数），$0 \le q_i \le 1$，$q_1 + q_2 + q_3 + \cdots + q_n = 1$ である．q_i は一つの保守作業においてモジュール m_i が変更される確率を表す）

R_d の要素を均等に $1/n$ とすれば，簡単に試算することができる．このモデルを一般化保守ポイント $C_{\mathrm{ave}}(G_P)$ と呼ぶ．

$$\begin{aligned} C_{\mathrm{ave}}(G_P) &= C_{\mathrm{gen}}(G_P, (1/n, \ldots, 1/n)) \\ &= \frac{\sum_{i=1}^{n} C(G_P, \{m_i\})}{n} \end{aligned}$$

より正確に R_d を計算する方法として，q_i の値を $C(G_P, \{m_i\})$ から求める方法も提案されている [18]．

これらの提案するメトリックスの妥当性は，今後，長期にわたる実験を行い検証する予定である．

5. むすび

この論文では，保守作業の労力見積りに用いるメトリックスを定義するために，プロダクトを有向辺と頂点からなるグラフとしてモデル化した．このモデル上で，保守作業の労力見積りに用いるメトリックスである保守ポイントを定義した．更に，保守ポイントと労力との関係を調べる実験を行った結果，保守ポイントは，既存のメトリックスに比べて，より正確な労力を見積もる上で有用であることを確認した．

今後の課題としては，より大規模な実験を行ってメトリックスを評価すること，長期的な実験を行って一般化保守ポイントを評価すること，実際のソフトウェア保守現場での使用が挙げられる．

文　献

[1] IEEE Standard for Software Maintenance (IEEE Standard 1219-1998), 1998.

[2] S.R. Schach, Software Engineering, Asken Associates, 1990.

[3] S. Yip and T. Lam, “A software maintenance survey,” First Asia-Pacific Conference on Software Engineering, pp.70–79, 1994.

[4] I. Sommerville, Software Engineering, 5th ed., Addison-Wesley, 1996.
[5] International Function Point Users Group, Function Point Counting Practices Manual: Release 4.2, 2004.
[6] M. Jørgensen, “Experience with the accuracy of software maintenance task effort prediction models,” IEEE Trans. Softw. Eng., vol.21, no.8, pp.674–681, 1995.
[7] H.M. Sneed, “Estimating the costs of software maintenance tasks,” ICSM, pp.168–181, IEEE Computer Society, 1995.
[8] F. Fioravanti and P. Nesi, “Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems,” IEEE Trans. Softw. Eng., vol.27, no.12, pp.1062–1084, 2001.
[9] S. Ramanujan, R.W. Scamell, and J.R. Shah, “An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort,” J. Syst. Softw., vol.54, no.2, pp.137–157, 2000.
[10] 山崎利治, “共通問題によるプログラム設計技法解説,” 情報処理, vol.25, no.9, pp.934–935, 1984.
[11] R.K. Fjeldstad and W.T. Hamlen, “Application program maintenance study: Report to our respondents,” in GUIDE 48, eds. G. Parikh and N. Zvegintzov, Philadelphia, 1983.
[12] T.A. Corbi, “Program understanding: challenge for the 1990’s,” IBM Syst. J., vol.28, no.2, pp.294–306, 1989.
[13] C. McClure, The Three Rs of Software Automation: Re-engineering, Repository, Reusability, Prentice-Hall, Upper Saddle River, NJ, USA, 1992.
[14] M.J. Harrold, J.A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S.A. Spoon, and A. Gujarathi, “Regression test selection for java software,” OOPSLA ’01: Proc. 16th ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications, pp.312–326, New York, NY, USA, 2001.
[15] A.J. Ko, H. Aung, and B.A. Myers, “Eliciting design requirements for maintenance-oriented ides: A detailed study of corrective and perfective maintenance tasks,” ICSE ’05: Proc. 27th International Conference on Software Engineering, pp.126–135, New York, NY, USA, 2005.
[16] L.C. Briand, Y. Labiche, and G. Soccar, “Automating impact analysis and regression test selection based on uml designs,” 18th International Conference on Software Maintenance (ICSM 2002), pp.252–261, 2002.
[17] H. Gall, K. Hajek, and M. Jazayeri, “Detection of logical coupling based on product release history,” ICSM ’98: Proc. International Conference on Software Maintenance, p.190, Washington, DC, USA, 1998.
[18] 小林健一, 吉野利明, 井上克郎, 早瀬康裕, 松尾昭彦, 上村学, “保守の影響波及範囲に基づいたレガシーシステムの障害予測,” 信学技報, SS2005-68, Dec. 2005.

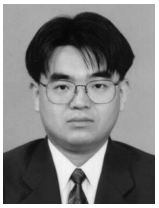
（平成 18 年 10 月 31 日受付，19 年 2 月 14 日再受付）

早瀬　康裕

平 14 阪大・基礎工・情報卒．現在同大大学院情報科学研究科博士後期課程在学中．ソフトウェアプロセスの研究に従事．情報処理学会会員．

松下　誠

平 5 阪大・基礎工・情報卒．平 10 同大大学院博士課程了．同年同大・基礎工・情報・助手．平 14 阪大・情報科学・コンピュータサイエンス・助手．平 17 同専攻助教授．博士（工学）．ソフトウェアプロセス，オープンソースソフトウェア開発の研究に従事．



楠本　真二　（正員）

昭 63 阪大・基礎工・情報卒．平 3 同大大学院博士課程中退．同年同大・基礎工・情報・助手．平 8 同学科講師．平 11 同学科助教授．平 14 阪大・情報科学・コンピュータサイエンス・助教授．博士（工学）．平 17 同専攻教授．ソフトウェアの生産性や品質の定量的評価，プロジェクト管理に関する研究に従事．情報処理学会，IEEE，IFPUG，PM 各会員．

井上　克郎　（正員）

昭 54 阪大・基礎工・情報卒．昭 59 同大大学院博士課程了．同年同大・基礎工・情報・助手．昭 59〜61 ハワイ大マノア校・情報工学科・助教授．平 1 阪大・基礎工・情報・講師．平 3 同学科・助教授．平 7 同学科・教授．平 14 阪大・情報・コンピュータサイエンス・教授．工学博士．ソフトウェア工学の研究に従事．情報処理学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

小林　健一

平 4 東大・工・計数卒，平 6 東大・工・情報了．同年（株）富士通研究所入社．ソフトウェア工学，並列処理，計算機アーキテクチャ，データマイニングの研究に従事．情報処理学会会員．

吉野　利明

昭 55 九工大・工・情報卒，昭 57 九大大学院・総理工・情報システム学修士了．同年（株）富士通研究所入社．平 4〜8 情処会誌編集委員．ソフトウェア工学，エージェント，自然言語処理，データベース，ナレッジマネジメントの研究に従事．情報処理学会，IEEE-CS 各会員．