ソフトウェア単体テストの十分性評価基準に関する考察

下村哲司[†1] 森岳志[†1] 野中誠[†2]

本稿では，ソフトウェア単体テストの十分性評価基準として利用できるメトリクスについて，商用ソフトウェア製品の実績データを用いて分析した結果を報告する．

Note on Sufficiency Criteria of Software Unit Testing

Tetsuji Shimomura,[†1] Takeshi Mori[†1] and Makoto Nonaka[†2]

This paper discusses software metrics which evaluate sufficiency of software unit testing. It also shows a result of analysis using real business software products data.

1. はじめに

ソフトウェア開発現場において，結合テストやシステムテストで摘出した欠陥のうち，単体テストで除去すべきであった欠陥が少なからず存在する．

開発プロセス全体の効率性や生産性を鑑みた場合，これらの欠陥に起因する手戻り工数は無視できないため，これらの欠陥は，可能な限り単体テストで除去しておくことが好ましい．

実務的に単体テストで見逃した欠陥数を削減するためには，容易に測定可能なメトリクスを用いて，単体テストの評価基準を設けることが望ましい．しかし，単一のメトリクスの基準を用いただけでは，単体テストの十分性を評価することは難しい．例えば，コードカバレッジが低い場合であっても，単体テスト後に欠陥が摘出されていない場合もある．

一方，メトリクスを用いてソフトウェアの十分性を判断する研究は，Rosenberg ら[1] の提案などがあるが，商用ソフトウェア製品の実績値を用いた研究は多くはない．

本稿では，単体テストの見逃し欠陥数に着目し，複数のメトリクスの分布や傾向を分析することで，単体テストの十分性評価基準について考察した結果を報告する．なお，分析対象データとして，商用ソフトウェア製品の実績データを用いる．

†1 日本電気株式会社
NEC Corporation
†2 東洋大学
Toyo University

2. 分析対象メトリクスと分析対象製品

2.1 分析対象メトリクス

筆者らの組織では，ソフトウェアの生産性と品質向上を目的としたクラウド型ソフトウェア開発環境（以下，ソフトウェアファクトリ）を利用することで，ソフトウェアの各種メトリクスが自動収集可能になっている．本稿では，実運用を考慮し，ソフトウェアファクトリを用いることで容易に測定可能なメトリクスについて分析する．

分析対象メトリクスを**表 1** に示す．表 1 の UT 見逃し欠陥数は，結合テストやシステムテストで摘出した欠陥数のうち，単体テストで除去すべきであった欠陥数[⋆1]を示す．ELOC は，コメント行と空白行を除いたソースコードの有効行数を示す．UT メソッド数は，単体テスト用のテストコードに記述されたメソッド数を KLOC[⋆2]あたりで正規化した値を示す．UT カバレッジは，ソースコード上の実行可能なステートメントのうち，単体テスト用テストコードを実行することでテストされたステートメントの割合を示す．なお，各メトリクスはファイル単位に収集した．

表 1 分析対象メトリクス（ファイル単位）

メトリクス	説明
UT 見逃し欠陥数	単体テスト見逃し欠陥数
ELOC	ソースコード有効行数
UT メソッド数	単体テストメソッド数/KLOC
UT カバレッジ	単体テストステートメントカバレッジ

⋆1 詳細設計仕様書の記載内容とソースコード記載内容の相違など
⋆2 KLOC=ELOC∗1000

2.2 分析対象製品

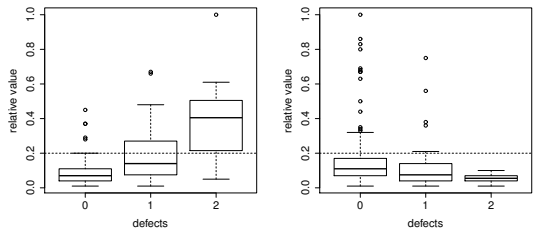
分析対象製品は，筆者らの組織で開発された汎用的システムソフトウェア製品のうち，新規性・保守性が高く，かつ，ソフトウェアファクトリを有効活用することで，UT 見逃し欠陥数，ELOC，UT メソッド数，UT カバレッジを自動収集可能な製品とした．なお，本製品は，単体テストの自動化が実現できている製品である．また，開発言語は Java である．

3. 分析手順と考察

以下に，UT 見逃し欠陥数，ELOC，UT メソッド数，UT カバレッジの分析手順と考察を示す．なお，UT カバレッジについては，詳細を省略する．

3.1 分 析 手 順

UT 見逃し欠陥数に対する ELOC，UT メソッド数の分布を，それぞれ**図 1**(a)，(b) に示す．図 1 の縦軸は，ELOC と UT メソッド数それぞれの最大値を 1 とした場合の相対値である．図 1 の横軸が 0 の場合は UT 見逃し欠陥数が 0 であったファイルの分布，1 の場合は UT 見逃し欠陥数が 1 件以上 n 件未満であったファイルの分布，2 の場合は UT 見逃し欠陥数が n 件以上であったファイルの分布を示す．なお，n は特定の数値であるが，都合により変数で表現する．



(a) ELOC　　(b) UT メソッド数

図 1 欠陥数で層別したメトリクスの分布（相対値）

3.2 考　　察

図 1(a) を見ると，UT 見逃し欠陥数が 0 件のファイルは，そのほとんどが ELOC の相対値 0.2 以下に分布していることが分かる．一方，UT 見逃し欠陥数が n 件以上のファイルは，その約 75%が ELOC の相対値 0.2 以上に分布していることが分かる．次に，図 1(b) を見ると, UT 見逃し欠陥数が 1 件以上のファイルは，そのほとんどが UT メソッド数の相対値 0.2 以下に分布していることが分かる．

そこで，ELOC の相対値が 0.2 超，UT メソッド数の相対値が 0.2 未満，UT カバレッジの相対値が x 未満を基準値として仮設定する．そしてその基準値をそれぞれ独立して適用した場合の UT 見逃し欠陥を含むファイル，および UT 見逃し欠陥の検出率を**表 2** に示す．なお，x も特定の数値であるが，都合により変数で表現する．

表 2 メトリクスによる検出率（%）

メトリクス	基準値	ファイル		欠陥
		欠陥数 >0	欠陥数 $\geq n$	
ELOC	0.2	42%	75%	60%
UT メソッド数	0.2	97%	100%	98%
UT カバレッジ	x	68%	92%	78%

表 2 より，ELOC の場合，本基準値を用いると，欠陥が発生したファイルの半分弱を検出でき，n 件以上の欠陥が発生したファイルの 75%を検出できることが分かる．また，欠陥の検出率を見ると，60%を検出できることが分かる．同様にして，UT メソッド数，UT カバレッジについても，本基準値を用いることで高い割合で欠陥と欠陥が発生したファイルを検出できることが分かる．

また，今回の対象製品とは部門や言語が異なる他の商用ソフトウェア製品においても，本分析方法と同じ手法を適用できることが分かった．これらの製品は共通して，新規性・保守性が高いという特徴があった．

4. おわりに

本稿で報告した結果は，実績データに基づいているため，基準値として今後のソフトウェア開発へ適用可能と判断している．また，本稿で報告した手法は，複数のメトリクスや複数の製品へも適用可能な方法であることから，汎用性が高いことも分かった．

今後は，ソフトウェアファクトリを活用することで自動収集可能な他のメトリクス，例えば，サイクロマチック数[2] や CK メトリクス[3] についても本手法を適用し，これらメトリクスを組み合わせた考察を行う．そして，分析対象製品を拡大し，組織全体をカバーできる単体テスト基準値の導出を目標とする．

参 考 文 献

1) Rosenberg, L., Stapko, R., and Gallo, A.: Applying Object-Oriented Metrics, *6th Int'l Symposium on Software Metrics*, Nov.(1999)
2) McCabe, T.: A Complexity Mesure, *IEEE-Trans. Softw. Eng.*, Vol.2, pp.308-320(1976)
3) Chidamber, S. and Kemerer, C.: A Metrics Suite for Object Oriented Design, *IEEETrans. Softw. Eng.*, Vol.20, pp.476-493(1994)