# 品質考慮型生産性評価モデルの提案

— 全ソフトウェアライフサイクルを通じた生産性評価法 —

5S−1

若木　守*　宮村　修一**

(　NTT情報システム本部　NTTソフトウェア株式会社　)

## 1.はじめに

開発完了時の生産性評価メトリックスには，開発稼働（人年）当たりの開発規模(NCSL:Non-Comment-source-line)がある．しかし，このメトリックスには，開発の後続工程である保守工程の稼働を決定づける品質が考慮されておらず，品質の悪いプログラムと良いプログラムとが，開発の規模や稼働が同一の場合，両者の生産性は全く同じになってしまうという問題がある．

本論文では，保守工程の作業内容を分析し，品質に係わる作業を抽出し，当該作業を含めた生産性評価モデルを提案する．また，2,3 の適用例から，従来の生産性評価方法の無効性の実例を示し，改めて，開発完了時に残留欠陥数を予測して，生産性を評価することが重要であることを示す．

## 2.品質考慮型生産性評価基本モデル

欠陥修理費用（稼働）は，下流工程になる程著しく大きくなる．TRW 社のデータによると，運用段階の保守工程での欠陥修理費用は，開発段階の試験工程でのそれに比較して，約10となっている．[1] このことは，開発段階で品質を作り込まないとそのツケが増幅されて保守工程に廻ることを意味する．本モデルは，当該保守工程での稼働を考慮したモデルである．

●従来の開発完了時生産性 [-1]

$= \text{開発稼働} / \text{開発規模} = a / \alpha$

●品質考慮型生産性 [-1]

$= (\text{開発稼働} + \text{保守稼働}) / \text{開発規模}$ — (1)

$= (a + b) / \alpha$

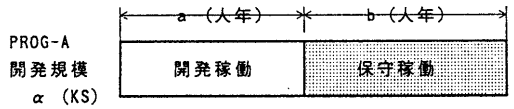



図1　生産性評価　基本モデル説明図

## 3.　プログラム品質に絡んだ保守作業の抽出

石井氏の分類に従うと，保守作業には，大きく事後保守と予防保守がある．[2] これら保守作業を構成する作業から，開発完了時の品質によって，その作業量が左右されるものを抽出する．

( 抽出法 )

①品質影響度：大

開発完了時に無欠陥であれば，当該作業は存在しない．

②品質影響度：中

無欠陥でも当該作業が存在するが，その量が半減する．

③品質影響度：小

欠陥の有無によらず，当該作業量が決定する．

この結果，影響力大の作業だけを抽出すると，(1)式の保守稼働は，(2)式になる．

保守稼働 =トラブル処理稼働＋予防修理処理稼働 — (2)

表1　保守作業と品質影響度

| 保守作業 | | | 品質影響度 |
|---|---|---|---|
| 事後保守 | トラブル（46%） | 品質管理<br>トラブル解析 | 大 |
| | | 修理方法検討<br>修理資材の作成& 試験<br>ﾄﾞｷｭﾒﾝﾄ修理& 資材提供 | 大 |
| 予防保守 | 予防修理（14%） | 修理資材統合<br>配付・提供 | 大 |
| | 製品改良（13%） | 改良方法検討<br>改良資材の作成& 試験<br>ドキュメント修正<br>改良資材提供 | 小（利用者ニーズに対応して改良するもので，品質とは異次元） |
| | 技術支援（21%） | 問合せ対応<br>SGチェック<br>システムバックアップ | 小（マニュアル品質による影響の方大） |
| 共通 | 環境保全（6%） | 環境保全 | 中（修理資材や改良資材の試験に共用） |

### 3.1 トラブル処理作業（(2)式・第 1項）での迷惑稼働の扱い

トラブルの現象(failure) から，欠陥(fault) を含むプログラムが特定化出来ない場合，被疑プログラム作成者間を渡り歩いて解析作業( 多段解析 )が行われる．表2 に多段解析の様子を示す．

上記表2 の例では，PROG-Bの作成者は，PROG-Aの欠陥の為に迷惑な稼働を生じたことなる．この稼働の扱いが課題となる．ソフトウェアが階層構造になっている場合，一般に上位プログラムから解析がなされ，下位（核）プログラムへディスパッチされる．従って，下位は上位の迷惑解析稼働を背負うことが多い．このことは，欠陥 1件の解析稼働の差別（不公平）と考えることも出来る．しかし，開発完了時に欠陥が無ければ，当該迷惑稼働が生じないことを考えると，当該稼働は，PROG-Aのトラブル解析稼働に含むべきものである．

従って，(2)式・第 1項は，

トラブル処理稼働 =( 自プログラム＋他プログラム )* の単位トラブル処理稼働×発見欠陥数 — (3)

( * :　以降の式では，自他を省略する．)

---

A model for Software Productivity Evaluaion Including Software Quality.

Mamoru WAKAKI *　Syuichi MIYAMURA **

* NTT Information Systems Headquater

** NTT Software corporation

表2　トラブル解析作業

| 単一解析 | 多段解析 |
|---|---|
| ○現象より，PROG-A が被疑者として上がり，解析した結果，やっぱり星 | ○現象より，PROG-B が被疑者として上がったが，最終的には，PROG-A が星 ） |
| PROG-A 作成者 ● 解析& 欠陥判明 | PROG-B 作成者 ○ 解析&ディスパッチ<br>PROG-A 作成者 ● 解析& 欠陥判明 |

3.2 予防修理処理（(2)式・第 2項）での共通稼働の扱い

予防修理処理作業には，発見された欠陥に対する修理資材をMT媒体に纏める上げる修理資材統合作業と，当該統合資材を利用システムへ配付・提供する作業がある．前者は，発見欠陥数に比例すると言えるが，後者は，利用システム数に依存し，欠陥数には依存しない．即ち，欠陥を発見したプログラムに取っては，共通稼働として投影される．ここでは，一回の統合作業で纏め上げる欠陥数は毎回同じと仮定して，共通稼働を欠陥数で正規化する．従って，(2)式・第 2項は，

予防修理処理稼働 =単位予防修理処理稼働×発見欠陥数 — (4)

(2)，(3)，(4)式より，　(1)式は，

品質考慮型生産性$^{-1}$
=（開発稼働＋（単位ﾄﾗﾌﾞﾙ処理稼働＋単位予防修理処理稼働）×発見欠陥数））/ 開発規模 — (5)

3.3 本モデルの適用時期と発見欠陥数の予測法

(1)予測に基づく評価（開発完了時）

この時期は，発見欠陥数の実績はなく当然予測となる．開発工程に於ける残留欠陥数（利用時に必ず発見されるとは限らないが）を予測する技法として数多くの信頼度成長モデルが提案されている．[3] ゴンペルツ曲線モデル，指数型モデル，及び，遅延 S字型モデルが代表例として挙げられる．これらのモデルを利用して，残留欠陥数を予測する．

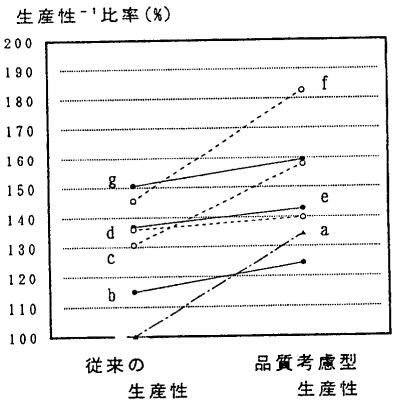
(2)実績に基づく評価（出荷後数年以内）

この時期は，保守工程での発見欠陥数の実施値が，ある程度蓄積されており，これら蓄積データを先の信頼度成長モデルの適用させれば，(1)より更に精度の高い残留バグ数の予測が出来る．三角氏は，ゴンペルツ曲線モデルの適用の結果から，精度高い最終発見欠陥数を求めるには，経験的発見欠陥数の 6〜 8割の欠陥数が発見されていることと述べている．[4] 我々の経験では，出荷後2 年で最終発見欠陥数の 7〜9 割の欠陥を発見してることから，この時期を実績評価時期とした．

4.適用結果

(1)適用対象プログラムのプロフィール

大規模オンラインシステム用汎用OSで，その中でも核の制御プログラム，通信プログラム，及びこれらのプログラム機能を業務プログラムへ簡易に提供するミドルプログラムである．実績に基づく評価結果を以下に示す．




(凡例)
制御ﾌﾟﾛ ： c, d, f
通信ﾌﾟﾛ ： b, e, g
ﾐﾄﾞﾙﾌﾟﾛ ： a
単位保守稼働 ： 同一分類では ほぼ同じ
最終欠陥数 ： a, d, gは 1年 他は 2年以上 のﾃﾞｰﾀより予測

図2　品質考慮型生産性評価モデル適用結果

(2)分析

① a，c，fは品質の作り込み状況が悪く，傾きが大きい中でも aは，上位のプログラムであり，迷惑稼働が c，f に比較して小さい．即ち，単位保守稼働が他より小さい．にも係わらず，傾きが c，fと同じであることは，品質の作り込みが一番悪いと言える．従来の生産性評価では，この aを最優良プログラムと誤認している．

②通信プログラムの b，e，gは傾きが一定である．このことは，開発工程での品質の作り込み状況が同じことを意味する．品質の作り込み技術・環境・体制が出来ている．

③制御プログラムの c，d，fでは，d の品質が良く，品質考慮型評価では，c と順位が逆転している．

以上の結果から分かる様に，従来の生産性評価では誤認が発生し，全ソフトウェアライフサイクルと通した生産性評価では，順位が逆転することが分かる．このことを考えれば，開発完了時期に残留バグ数を予測して，品質を考慮した生産性を評価することが大切であることが分かる．

5.おわりに

品質考慮型生産性評価モデルを提案し，適用結果も示してその重要性を述べた．評価モデルをそのまま適用出来ないにしても，その考え方は採用すべきと考える．採用に当たって大切なのは，まず，出荷時品質目標値による品質影響度の概算を掴むことである．(5)式を以下に変形する．

品質考慮型生産性$^{-1}$
= 従来生産性$^{-1}$ ×（1 + 品質影響係数）

但し，品質影響係数=
（単位トラブル &予防修正稼働 /従来生産性$^{-1}$）
×（目標残留欠陥数/ 開発規模）

例えば，開発完了時の目標残留バグ数が 0.3 件/KS で，単位トラブル &予防修正稼働 /従来生産性$^{-1}$ が 0.6倍とき，開発完了時の生産性に与える影響度は 20% である．

（参考文献）

(1)宮本勲著："ｿﾌﾄｳｪｱｴﾝｼﾞﾆｱﾘﾝｸﾞ現状と展望", TBS 出版会, P.14
(2)石井康夫編："ｿﾌﾄｳｪｱの検査と品質保証", 日科技連, P.185
(3)山田　茂著："ｿﾌﾄｳｪｱ信頼度評価技術", HBJ出版(1989)
(4)三角　武著："ｿﾌﾄｳｪｱの品質評価法", 日科技連, P.188(1981)