ソフトウェアの品質／信頼性評価

山田 淳，山田 茂

1．はじめに

近年においては社会の高度情報化がますます進む状況にあり，コンピュータ技術に支えられる高機能なサービスをタイムリーにかつ経済的に実現する必要がある．このような目的でコンピュータを利用した機器やシステムが急速に増加するにつれ，ソフトウェアは非常に多種多様な製品やシステムに利用され，大規模化，複雑化の傾向を一層強めてきている．そして，情報通信ネットワークシステムなどのように社会的システムおよびソフトウェアへの依存度も高まりつつある．

このため，ソフトウェアの信頼性評価を含む品質評価，および保証が重要な課題である．ここでは，ソフトウェア開発における品質評価のプロセスおよび信頼性モデルの利用による信頼性評価を中心に述べる．情報通信システムのソフトウェア開発では，通信プロトコルに関する部分は，特徴的な検証やテスト技法を用いるが，基本的には通常のソフトウェア開発と同様の品質評価技法を用いる．

やまだ あつし 株式会社東芝 研究・開発センター
〒210 川崎市幸区柳町70
やまだ しげる 鳥取大学 社会開発システム工学科
〒680 鳥取市湖山町南4-101

2．ソフトウェア品質の評価と保証のプロセス

2.1 ソフトウェアの品質特性

ソフトウェアの品質と一口にいっても，そこには多くの事柄が含まれるであろうことは容易に想像できる．そこで70年代後半からベーム [2] やマコール [14] らが，ソフトウェアの品質を複数の品質特性から構造的に組み立てて表現するという提案を行なった．現在は，ソフトウェアの品質特性について共通した認識を国際間で持つことを目的として，機能性，信頼性，使用性，効率性，保守性，移植性という6個の特性と，さらに特性を展開した副特性からなる構造をもった品質特性がISO／IECから国際標準として提供されている [9]（**表1**）．

2.2 ソフトウェア品質メトリクス

ソフトウェア開発に伴う仕様や設計，プログラム，テスト仕様などの成果物や作業記録などを測定し，品質達成の程度を表わす指標となる方法がソフトウェア品質メトリクスである．企画から保守，廃棄までのライフサイクルプロセスを通して，その段階に対応するメトリクスを適用して測定を行ない，中間製品や完成した製品，あるいは作業や運用状況などを評価する．

表1 ソフトウェア品質特性 ISO／IEC9126

(F)機能性(Functionality)	(U)使用性(Usability)	(M)保守性(Maintainability)
(f1)合目的性(Suitability)	(u1)理解性(Understandability)	(m1)解析性(Analisability)
(f2)正確性(Accuracy)	(u2)習得性(Learnability)	(m2)変更性(Changeability)
(f3)相互運用性(Interoperability)	(u3)運用性(Operability)	(m3)安定性(Stability)
(f4)標準適合性(Compliance)		(m4)試験性(Testability)
(f5)セキュリティ(Security)	(E)効率性(Efficiency)	(P)移植性(Portability)
(R)信頼性(Reliability)	(e1)時間効率性(Time behaviour)	(p1)環境適応性(Adaptability)
(r1)成熟性(Maturity)		(p2)設置性(Installability)
(r2)障害許容性(Fault tolerance)	(e2)資源効率性(Resource behaviour)	(p3)規格適合性(Conformance)
(r3)回復性(Recoverability)		(p4)置換性(Replaceability)

2.3 ソフトウェアの品質評価プロセス

評価は次の(1)から(5)の手順で行なう．ソフトウェアのライフサイクルプロセスの中で，複数回の評価を行なうが，リリース前の最終的なテスト段階では最終的な総合評価を行なう．また，(6)メトリクス改善は，普通は運用，保守段階に入ってから，次の開発での評価の有効性を高めるための準備として行なう．

(1)品質要求定義：品質面からユーザの要求を定義し仕様化する．品質特性（機能）展開などを利用．

(2)メトリクス計画：測定方法の選択，開発，トレーニングなどの適用準備，適用範囲と時期を設定．

(3)メトリクス適用：測定を行ない，記録，文書化．

(4)評定：測定時点での評価（測定時点評価）や，測定時点以後の傾向を推定して評価（予測的評価）．

(5)総合評価：評定結果と費用や納期なども考慮して，工程進行や納品受入の可否判断（管理的評価）．

(6)メトリクス改善：テスト時，納品時，運用保守時の評価と開発中の評価結果とを比較し，メトリクス自体の利用方法も改善．

3．ソフトウェア品質の評価・保証技法

3.1 ソフトウェア品質要求定義

ソフトウェアについて機能の要求定義だけでなく，品質の要求も定義することにより，要求に対応する品質保証の計画と仕様の品質評価を行なうことができる．

品質要求定義では，(1)まず品質特性展開(品質機能展開）技法により［7, 11, 18］，機能に関する要求仕様をもとに品質面から要求を洗い出して，品質要求特性，品質要件項目のリストアップを行なう（展開図にまとめる．図1）．(2)次に，リストアップしたそれぞれの品質要件項目を，実現方法や機能，設計項目や開発手法，制約や運用方法として表現した品質機能項目にまで展開した品質仕様を作り，品質作り込みのための品質設計の方針を決める．(3)これらの項目の目標提示及び実施チェック方法と時期を計画して，品質保証計画を立案する．

実施チェック方法には，実施の程度と達成された品質の程度を，調査し保証する方法として，メトリクスを用いることができる．メトリクスの利用により得られる測定・評価結果をもとに，設計改良や重点的レビューとテスト，進捗審査などを指示し，開発作業を品質面からコントロールすることで，品質保証を行なう．特定の品質要件項目に重点化したとき，これらは要求水準の高い品質特性をゴールとして，到達方法を質問形式で展開し，その実施をメトリクスにより測定，評価するバシリのGQM(Goal／Question／Metric)パラダイム［1］に相当する．また，品質要求の重要度に応じて評価の細かさを決め，コストの最適化を図る．評価の細かさは，(a)評価を実施する工程や回数，時期を選択する，(b)評価に利用する測定法と品質データを選択する，などにより調節する．通信ソフトウェアシステムでは，通信制御と通信システム運用のサブソフトウェアシステムに分けて，それぞれ品質を展開するとよい．

3.2 設計品質評価

通信プロトコル設計は，通常非常に多くの状態数を含む．そこで，まず(1)通信のフェーズやレイヤーをもとに，取り扱いやすい状態数の部分に分けて，(2)イベントツリー（事象木）という状態遷移図または状態マトリクスを作り，(3)効率的とされている深さ優先探索などにより要求する状態への到達可能性を検証する［12］．最近ではプロトコル設計の検証に適した形式的記述言語や支援ツールも利用され始めている．

3.3 プログラム品質評価

詳細設計およびプログラムのモジュール構造や制御構造について，設計・プログラムの複雑度という考え方を導入する．そして，複雑な部分ほど欠陥が混入し潜在する傾向があり，また影響も大きいことから，そのような複雑なモジュールやプログラムの多いソフトウェアで信頼性や保守性が低

ソフトウェア品質特性 ＼ 品質機能項目 ／ 品質要件項目	データ暗号化	アクセス記録機能	利用者認証機能	データフォーマット共通化	送受信プロトコル共通化	データ変換機能	異常検出点の増設	ロールバック機能	データバックアップ機能	通信ルートの冗長化切替	縮退運転機能	テスト密度向上	推定残存欠陥密度の低減	MTBF向上までテスト
機能性	通信内容の機密保護	○		○										
	遠隔複数端末から運用可能				○	○								
	他システムとの接続が容易				○	○	○							
信頼性	故障を起こしにくい											○	○	○
	障害への耐性を強化したい							○	○		○	○		
	接続の失敗回復を容易								○					
	データの復旧操作を容易									○				

図1　品質特性展開（品質機能展開）の例

下する危険度が高いと評価し，危険性を調査する．また，モジュールやプログラムごとの複雑さの度数分布から，特異なものや計画値を越えるものを抽出し，欠陥が潜在している可能性が高いとして，重点的レビューやテスト対象の選定に利用する［5, 19, 20］．以下に代表的なメトリクスを示す．

(1)サイクロマティック数（Cyclomatic Number）［13］

プログラム実行経路が取り得る独立経路の数を表現する．プログラムの制御の流れをグラフ構造で表現したときに，$e-n+2$（e：辺の数，n：節点の数）として求められるが，通常はグラフ構造に変換する手間を省略して，プログラムの条件判定数＋1で計算する．

(2)情報フロー量（Infornation Flow）［4］

着目したプログラムモジュールとそれ以外のモジュールとの関係構造の複雑度を表現する．モジュールが外部との間で輸入(Fan-in)または輸出(Fan-out)する変数データが相互作用する際の組合せ可能な数による複雑さを（Fan-in×Fan-out)^{2}により表わす．

3.4 テスト品質評価

テストでは欠陥の摘出除去状況とテスト自体の状況について，複数の定量的な指標を設定し，計画した基準と照合して，テストの十分さと，テストにより確保できたソフトウェアの信頼性を評価する．同時にテストを終了してリリースするかどうかの判断を行なう．このような指標の代表例として，後述する信頼度成長モデルの利用による，総欠陥数（総フォールト数または総ソフトウェア故障数），未摘出の残存欠陥数，MTBFそれぞれの推定値を用いる．また，これらに加えて欠陥検出密度，テストカバレッジ（テスト項目実施密度，テスト実行した機能，プログラムのモジュールや分岐，ブロック，パスの網羅度），未解決欠陥密度なども指標として併用する．

3.5 問題点と欠陥の分類管理による評価

テストをはじめ開発および運用保守のそれぞれの段階で検出する問題点および欠陥は，通常，故障，フォールト，エラーとして分類区別する．さらに，発生箇所と，致命的(システムダウン)，回避策なし／あり，軽微などの影響の重大度別などに分類する．そして，分類別に発生頻度やその時間的推移から信頼度の評価を行なう．また，フォールト混入プロセスとそのヒューマンエラーメカニズムを調査して再発防止策を立てる．人的要因は品質の非常に重要な要因の1つである［8, 3, 11］．

4．ソフトウェア信頼性モデル

4.1 ソフトウェア信頼度成長モデル

ソフトウェアの信頼性は品質の良いソフトウェアを効率良く開発するために必要な生産技術のうち，ソフトウェア管理技術に位置づけられる品質管理の問題として取り扱われることが多い［11, 21, 22］．ソフトウェア内に潜在するフォールト数や，フォールトに起因してソフトウェアが期待通りに動作しないというソフトウェア故障の発生時間間隔により，ソフトウェア信頼性が評価できるので，ソフトウェアの信頼性技術には，2つ技術分野が存在する．すなわち，人為的誤りや欠陥を開発プロセスで作り込まないようにするフォールトアボイダンス（fault avoidance）と，ある程度の誤りや欠陥は避けられないものとして，その影響を最小限にしようとするフォールトトレランス（fault tolerance）がある［23］．

近年，特にフォールトアボイダンスの立場から，製品としてのソフトウェアの生産性と信頼性の向上を目指すソフトウェア工学の視点より，ソフトウェアの定量的信頼性評価のために，さまざまなソフトウェア信頼性モデルが提案されてきた．開発プロセスのテスト工程までの上流工程と，テスト工程で適用されるソフトウェア信頼性モデルを，それぞれソフトウェアの複雑性モデル（software complexity model）と信頼度成長モデル（software reliability growth model）として比較したのが**表2**である．特に後者は，ソフトウェア開発のテスト工程あるいは運用段階におけるソフトウェア故障発生現象やフォールト発見事象を，ソフトウェアの信頼度成長過程とみなして確率則を導入してモデル化したものである［21, 22, 23］．

これまでにも，多くの実際のソフトウェアプロジェクトから採取された観測データにより，提案されてきたソフトウェア信頼度成長モデルの適用性・妥当性も検証されてきた．たとえば，最近特に有望視されているのが非同次ポアソン過程（nonhomogeneous Poisson process）（以下NHPPと略す）モデルである［22, 23, 16, 15］．NHPPモデルは，任意の時刻tまでに発見される総フォールト数（または発生するソフトウェア故障数）を表わす確率変数$N(t)$にNHPPを仮定するものであり，$N(t)$の確率分布は次式のポアソン分布により与えられる．

表2　ソフトウェア信頼性モデルの分類と比較

	ソフトウェア信頼度成長モデル	ソフトウェア複雑性モデル
目　的	テストの履歴から現時点でのソフトウェア故障率や潜在フォールト数を推定し，信頼性の達成状況やリリース後の運用段階における信頼性を予測する。 （動的信頼性の推定・予測）	ソフトウェア自体の構造的特性から複雑性を評価して信頼性を計測する。 （静的信頼性の推定・予測）
主な利点	テストの進捗状況やソフトウェア開発の結果を定量的尺度により把握できる。	ソフトウェア開発の初期の段階で構造的尺度によりソフトウェア特性が評価できる。
問題点	● モデルが多岐にわたり，モデルユーザがそれらの選択を誤ると評価結果に信頼がおけない。 ● モデルを適用するにあたり，その仮定がソフトウェア開発の実状に即していない場合がある。	● モデルによる結果と信頼性データとの客観的な関連がみられない。 ● 複雑性と検出されたソフトウェアフォールトとの観測された相関が当該プロジェクトあるいはアプリケーションに特定的である。
直接的に関連する固有技術	テスト技法	構造化プログラミング 構造化技法
分　類	（1） 時間計測モデル （2） 個数計測モデル （3） アベイラビリティモデル	（1） プログラム複雑性モデル （2） プロセス複雑性モデル

(1)　$(1a)\ Pr\{N(t)=n\}=\{H(t)^n/n!\}\exp^{[-H(t)]}$　$(n=0,1,2,\cdots)$

$(1b)\ H(t)=\int_0^t h(x)\,dx$

ここで，$Pr\{A\}$は事象Aの生起する確率を表わす．式$(1a)$の$H(t)$は，$N(t)$の平均値を表わす平均値関数であり，時間区間［0，t］において発見される総期待フォールト数を意味する．また式$(1b)$の$h(t)$は，時刻tにおけるフォールト発見率を表わし，強度関数と呼ばれる．式(1)で定式化されるNHPPモデルを実際のソフトウェアプロジェクトに適用して信頼性評価を行なうには，当該テスト環境あるいは運用環境に適合する平均値関数$H(t)$を特定化する必要がある．代表的な平均値関数をもつNHPPモデルをまとめたのが**表3**である．このようなNHPPモデルにもとづいて，ソフトウェア内の潜在フォールト数をはじめとして，ソフトウェア信頼度や平均ソフトウェア故障発生時間間隔（または平均フォールト発見時間間隔）などの信頼性尺度を導出でき，ソフトウェアの信頼性評価を行なうことができる．

さらに，ソフトウェア信頼性評価の精度を高めるために，フォールト検出時における保守の不完全性や新規フォールトの混入を考慮した不完全デバッグモデル［23, 16］やテスト工程におけるテストケースとフォールト検出の応答関係に着目した超幾何分布モデル［17, 10］などのように，従来モデルを修正・拡張する研究も多くみられるようになった．一方，日本では従来より，テスト工程における品質評価モデルとして，テストにより発見された総フォールト数がS字形成長曲線を示すことから，人口や需要の予測などに用いられてきたロジスティック曲線やゴンペルツ曲線をフォールト発見数データに直接あてはめる決定論的方法もとられている．この方法は各成長曲線の収束値kをソフトウェア内に潜在する総フォールト数として回帰分析により推定するものであり，それぞれテスト時刻tまでに発見される総フォールト数は次式で与えられる［11, 22］．

ここで，m, α, a, bは定数パラメタである．

(2)　$L(t)=k/\{1+m\ exp^{(-\alpha t)}\}$　$(m>0, \alpha>0, k>0)$

(3)　$G(t)=ka^{b^t}\{b^{**}t\}$　$(0<a<1, 0<b<1, k>0)$

前述したようなモデルを組み込んだソフトウェア品質／信頼性評価ツールが，コンピュータメーカやソフトウェアハウスなどで開発され，実用に供されるようになっている．たとえば，SRETIIと呼ばれるソフトウェア信頼性評価ツールによる信頼性評価の例を示したのが**図2**である［23］．

4.2　信頼度成長モデルの利用

信頼度成長モデルは，テスト工程で次のように利用される．テストの日数や工数または実際にテスト稼働させた時間，実行されたテスト項目数などを時間軸とし，ある時点までに検出された欠陥数の累積値を観測して，モデル式にあてはめ，推定値を計算する．通常，検出除去すべき欠陥検出数を初めに計画する［8, 22］．

計画はまずプログラム作成直後の時点で，欠陥総数として1000行当り40から80個程度が潜在していると見積る．または一般事例や過去の事例をもとに，規模と複雑度を説明変数にして両対数尺度で回帰分析で見積る．テスト期間の途中では数回にわたり，総ソフトウェア欠陥数と残存欠陥推定値，MTBF（瞬間MTBF）などを推定する．さらに，推定前の欠陥摘出計画の数値を，推定値で置き換えて修正し，新しい計画値かつ欠陥摘出の目標値としてテストを続行する．また，テスト実施項目数や工数が並行して成長する様子も同時に観測しておく．そして欠陥摘出の時間推移の推定値の上下に限界値を設定し，実際の観測値が限界値を連続して越えた時点で，テストの実施状況や内容とソフト

表3　代表的なNHPPモデル

NHPP モデル	平均値関数 $H(t)$	強度関数 $h(t)$	環　境
指数形ソフトウェア信頼度成長モデル (exponential SRGM)	$m(t)=a(1-e^{-bt})$ $(a>0,\ b>0)$	$h_m(t)=abe^{-bt}$	テスト期間を通じてフォールト発見率が一定のソフトウェア故障発生現象を記述する.
修正指数形ソフトウェア信頼度成長モデル (modified exponential SRGM)	$m_p(t)=a\sum_{i=1}^{2}p_i(1-e^{-b_it})$ $(a>0, 0<b_2<b_1<1,$ $\sum_{i=1}^{2}p_i=1, 0<p_i<1)$	$h_p(t)=a\sum_{i=1}^{2}p_ib_ie^{-b_it}$	テストの進行にともなうフォールトの発見難易性を記述する（発見の容易なフォールトの発見率b_1，発見の困難なフォールトの発見率b_2）.
遅延S字形ソフトウェア信頼度成長モデル (delayed S-shaped SRGM)	$M(t)=a[1-(1+bt)e^{-bt}]$ $(a>0, b>0)$	$h_M(t)=ab^2te^{-bt}$	フォールトの発見に至る過程を，ソフトウェア故障発見過程とフォールト認知過程という，引き続く二つの事象により記述する.
習熟S字形ソフトウェア信頼度成長モデル (inflection S-shaped SRGM)	$I(t)=\dfrac{a(1-e^{-bt})}{(1+ce^{-bt})}$ $(a>0, b>0, c>0)$	$h_I(t)=\dfrac{ab(1+c)e^{-bt}}{(1+ce^{-bt})^2}$	テスト労力の変動やテストチームの習熟度を考慮して，ソフトウェア故障発生現象を記述する.
テスト労力依存型ソフトウェア信頼度成長モデル (test-effort dependent SRGM)	$T(t)=a[1-e^{-rW(t)}]$ $W(t)=\alpha[1-e^{-\beta t^m}]$ $(a>0, 0<r<1, \alpha>0,$ $\beta>0, m>0)$	$h_T(t)$ $=ar\alpha\beta mt^{m-1}e^{-\beta t^m}e^{-rW(t)}$	テスト工程における工数などの投入労力量と発見される総フォールト数を関係づけ，その時間的変化を記述する.
対数型ポアソン実行時間モデル (logarithmic Poisson execution time model)	$\mu(t)=\dfrac{1}{\theta}\ln(\lambda_0\theta t+1)$ $(\lambda_0>0, \theta>0)$	$\lambda(t)=\dfrac{\lambda_0}{(\lambda_0\theta t+1)}$	テスト時間をCPU時間で計測する時に，ソフトウェア故障率が発生するソフトウェア故障数に関して指数関数的に減少する.

(SRGM: Software Reliability Growth Model)

a = テスト開始前にソフトウェア内に潜在する総期待フォールト数
b, b_i, r = フォールト発見率を表すパラメータ
c = フォールト発見の習熟度を表すパラメータ
α, β, m = テスト労力関数 $W(t)$ を決めるパラメータ
λ_0 = 初期ソフトウェア故障率
θ = ソフトウェア故障率の減少率

ウェアを詳細に調査し，新しく推定を行なって計画を修正する．テスト方法や内容などに変化があると，欠陥摘出の時間推移は影響を受けやすいので，テスト時間軸上の全体区間と部分区間で推定を行なう場合もある．

4．おわりに

今後は，ソフトウェア信頼度成長モデルによる信頼性評価結果を，テスト進捗管理に反映するテスト工程管理法や，テストを終了してユーザの運用段階へ移行するのに

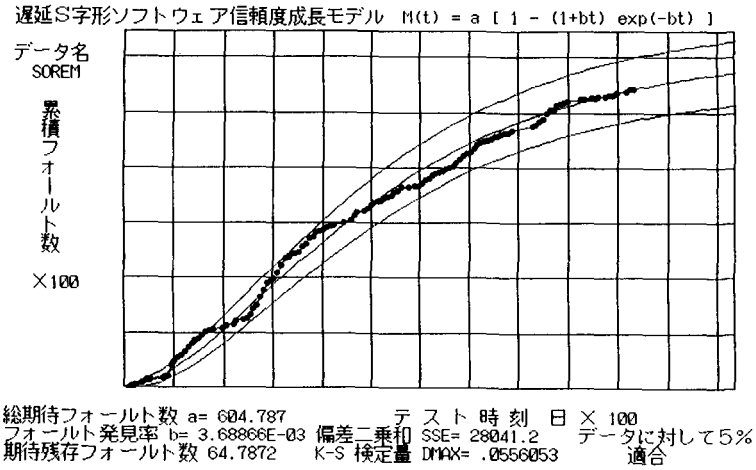

図2　SRETIIによるソフトウェア信頼性評価例

最適な時期を見積もる最適リリース時刻決定法などのソフトウェアマネジメントにおける問題の解決に反映する方法について，より実際的な観点から議論する必要がある．最近ではソフトウェア工学について国際規格がISO／IEC JTC1／SC7の委員会などで作成されつつあるが，それにはソフトウェアの品質保証とプロセスとの相互の強い関係が示されるなど，ソフトウェア開発組織のもつプロセス自体の継続的な評価と改善[6]に焦点を当てた品質改善も課題である．

参考文献

[1] Basil, V. and Weiss, D. M. : A methodology for collecting valid software engineering data, IEEE Trans. Softw. Eng., vol. 10, no. 6, pp. 728-738, 1984.

[2] Boehm, B. W., Brown, J. R., Kaspar, J. R., Lipow, M., and MacCleod, G. J., Merrit, M. J. : Characteristics of Software Quality, North-Holland, New York, 1978.

[3] DeMarco, T. and Lister, T. : Peopleware-Productive Project and Teams, Dorset House, 1987.（松原友夫他日立ソフト生産研究会訳：ピープルウェア：日経BP，1988）

[4] Henry, S. M. and Kaufra, D. : Software structure metrics based on information flow, IEEE Trans. Softw. Eng., vol. 7, no. 5, pp. 510-518, 1981.

[5] Hirayama, M., Sato, H., Yamada, A. and Tsuda, J. : Practice of quality modeling and measurement on software life-cycle, Proc. on Int. Conf. Softw. Eng., pp. 98-108, 1990.

[6] Humphrey, W. S. : Managing the Software Proc-ess, Addison-Wesley, 1989.（藤野喜一監訳：ソフトウェアプロセス成熟度の改善，日科技連出版社，1991）

[7] 情報処理振興事業協会（IPA）：品質機能展開による高品質ソフトウェアの開発手法，コンピュータエージ社，1989.

[8] 石井康雄編：ソフトウェアの検査と品質保証，日科技連出版社，1986.

[9] ISO／IEC 9126 : Software Product Evaluation -Quality Characteristics and Guide-lines for Their Use, 1991.

[10] Jacoby, R. and Tohma, Y. : Precise formulation and applicability of a software reliability growth model based on hyper-geometric distribution, J. Inform. Process., vol. 14, no. 2, pp. 192-203, 1991.

[11] 菅野文友編：ソフトウェアの品質管理，日科技連出版社，1986.

[12] Lin, F. J. and Liu, M. T. : Protocol validation for large-scale applications, IEEE Software, pp. 23-26, vol. 9, no. 1, 1992.

[13] McCabe, T. J. : A complexity measure, IEEE Tran. on Softw. Eng. vol. 2, no. 4, pp. 308-320, 1976.

[14] McCall, J. A., Richards, P. K., and Walters, G. F. : Factors in Software Quality, US Rome Air Development Center Reports NTIS AD／A-049014, 015, 055, 1977.

[15] Musa, J. D., Iannino, A. and Okumoto, K. : Software Reliability : Measurement, Prediction, Application, McGraw-Hill, New York, 1987.

[16] 大場充：ソフトウェアプロジェクトの実績データ収集・分析技法，ソフト・リサーチ・センター，1993.

[17] Tohma, Y., Jacoby, R., Murata, Y. and Yamamoto, M. : Hyper-geometric distribution model to estimate the number of residual software faults, Proc. COMPSAC'89, pp. 610-617, 1989.

[18] 吉沢正，東基衛，片山禎昭：ソフトウェアの品質管理と生産技術，日本規格協会，1988.

[19] 山田淳，平山雅之，佐藤弘行，津田淳一郎：ソフトウェア設計品質の定量化手法，電子情報通信学会，信頼性R89-5，pp. 25-30, 1989.

[20] Yamada, A., Hirayama, M., Sato, H. and Tsuda, J. : Quantitative analysis method of software design characteristics for quality improvement, Proc. 2nd European Conf. Softw. Quality Assurance, 1990.

[21] Yamada, S.: Softwar. quality／reliability measurement and assessment : software reliability growth models and data analysis", J. Inform. Process., vol. 14, no. 3, pp. 254-266, 1991.

[22] 山田茂，高橋宗雄：ソフトウェアマネジメントモデル入門，共立出版，1993.

[23] 山田茂：ソフトウェア信頼性モデル－基礎と応用－日科技連出版社，1994.