ソフトウェア工学 40－18
（1985. 2. 8）

ソフトウェア品質評価手法

中村英夫　内平直志　佐藤誠　水谷博之
（株式会社 東芝　システム・ソフトウェア技術推進部　青梅工場　総合研究所）

1. はじめに

「質の向上」という概念は、対象となる工業製品が成熟すればするほど重要になってくる。技術進歩が急ピッチな製品（ソフトウェアもそれに含まれる）ほど、早期に品質向上の対策を施さねばならない。品質向上の基本は、その評価である。評価を定量的に行ない品質の向上策を施す必要がある。しかし、品質という抽象的な概念を定量化するには、様々困難がある。今回、その計量化手法と、それに基く評価システムを開発したので報告する。

2. 品質評価モデルの構築

ソフトウェアライフサイクルと計量

従来、ソフトウェアの計量という場合、ソフトウェア製品自体の持つ不具合数あるいは、複雑さと言った、いわば計量化し易い部分の計量化にとどまる場合が多かった。しかしながら、それらの計量値が十分再現性があり、予測に用いることができる程安定したものではないことは一般に認められている所である。個々の製品の開発環境，投入リソース、管理方式といった組織活動の違いにより、上記の計量値は変動する。本稿ではソフトウェア製品は、その開発組織の単なる側面であり、上記計量値は、開発組織の持つ生産能力が製品自体を通して具現化したものにすぎないとの立場を取る。従ってソフトウェアの品質を議論する場合、開発組織及びその活動に関する計量を無視できない。この様なあいまいな問題をモデル化する際の常とう手段は、構造的に変化の少い部分部分を取り出して、それぞれモデル化し、互いに関連付けを行うことである。ソフトウェア開発における構造的変化の少い部分は、ソフトウェアライフサイクルと呼ばれる一連の作業工程の一つ一つであろう。従ってここでは、一般に多くの組織で採用されている工程分類－要求定義，設計，製造，検査，保守－を考えモデル構築のための基本単位とする。

モデル構築の方法

ソフトウェア品質評価モデルの開発は、他の工学システムのモデル化と大きく異る。それは、何をモデル化すれば良いのか、完成するとはどのレベルまで到達したものを指すのか等、あいまいな問題をより多く含んでいる点である。我々は、モデル開発を次の四フェーズに別けて行う。

第１フェーズ　何をモデル化するべきかをシステマティックに洗い出す。
第２フェーズ　各モデルの詳細化、作成を行う。
第３フェーズ　実際のデータにより、モデルの分析を行いリファインする。
第４フェーズ　運用フェーズ

本稿は第１，２フェーズを中心に述べる。

ソフトウェア品質評価は、人間を含んだ、組織の生産活動の評価と考えられる。従ってモデル化には次の様な困難な点が有る。

(1)広い分野に問題が関係し、要素も無数に考えられる。
(2)問題の本質は、モデルの利用目的に応じてのみ定まり、客観的には存在しないと考えられる。
(3)モデルの利用目的が複数有り、また不明確になり易い。
(4)作成されたモデルに至るまで、何が抽象化されたかの記録が少い。
(5)個々の特性が十分理解されていず、要素間の関係および問題の境界が不明確である。
(6)モデル作成及び検証に用いることのできるデータが限られている。

これらの困難点を克服するため、品質評価システムの企画（第１フェーズ）にシステム計画技法ＴＵＰＰＳ[1],[2]を適用した。そのねらいは、次の点である。

(1)従来多く見られた個別のモデル作りのみに終始せず、システム全体から何が問題であるかを体系的に明らかにする。
(2)利用目的の明確化をユーザとモデル開発者が共同で進めることにより、意思統一をはかる。また、ユーザの持つソフトウェアに関する知識を明文化する。
(3)既知の知識および測定可能なデータと利用目的とを勘案し、モデル作成を行う。
(4)同一の問題に対しても、利用目的，精度，構造に関して複数のモデル案が有り得る。それらを比較，評価する。

第二フェーズは、第一フェーズの結果（各モデルの開発方針とモデル間の関係）を受けて、モデルの実際の開発を行う。企画段階では、抽象的なまま詳細化されずにいた変数を実際のソフトウェア開発に照らし、意味のあるものにしていくフェーズである。（３章，４章）

ソフトウェア品質評価モデルの企画

ＴＵＰＰＳは、システムの仕様を効率良く明確化する方法論と、それを支援するツール群から成る。今回の企画はモデル設計者とユーザ（ソフトウェア製造および品質管理の責任者）の共同で行われた。図１に企画手順を示す。

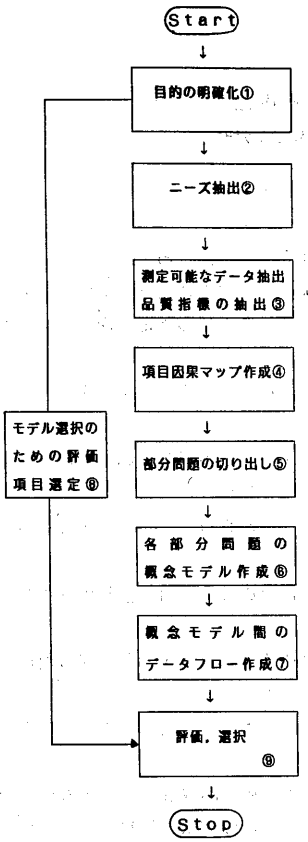

図1　ソフトウェア品質評価モデル企画手順

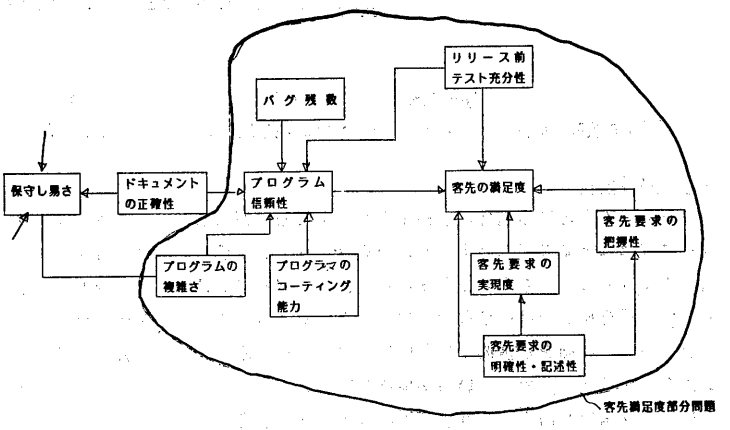

図2　項目間因果（部分）マップと部分問題の切り出し

①では、品質評価の目的をトップダウンにもれなく列挙し、樹木図としてまとめ、今回のプロジェクトでねらう範囲を明らかにする。②では、その範囲内で、ソフトウェア開発現場からどの様なニーズ，問題点があるかを拾い出す。ＴＵＰＰＳには、樹木図の作成，編集を容易にするツールおよびニーズ抽出の抜けを防ぐための方法論，専用用紙が用意されている。③では、現実に測定している、あるいは測定できる可能性のあるデータをここで明らかにしておく。更に、ニーズ項目とデータ項目とを関係づける内部変数として品質指標項目を発想し、その項目間の因果関係をマップとして作成する（④）。このマップを構造化ツール等を用いて分析し、因果関係が密であり、かつ、意味の有るまとまりを部分問題として切り出す（⑤）（図２）。ここでは、客先満足度，保守のし易さ，組織の生産性，計画の遵守性，の４つの部分問題が抽出された。⑥では、各部分問題に対して、因果マップを参考にしながら概念モデル群を作成していく。概念モデルとは、モデルの外部仕様に相当し、複数の概念モデルが集まって一つの部分問題のモデル化となる。その複合のされかたを⑦のモデルデータフロー（図２がその部分である）で表現する。⑧であらかじめ選定しておいたモデルの良否を評価する評価項目（開発することによる効果，信頼性，開発の困難度，運用の困難度）について４つの部分問題の概念モデルデータフローを評価し、最も得点の高かった“客先満足度”を選択した。

ＴＵＰＰＳ適用の効果

前節で見た様に、ＴＵＰＰＳは、広い視野から問題をシステマティックにとらえていく。従って重要な事項の見落しを防止し、作業のあともどりを無くすことができる。従来多く見られた個別のモデルでは不明朗なまま残されていた、全体系の中での各モデルの位置付けが明らかにできることも適用の効果である。更に重要なことは、ＴＵＰＰＳは、ユーザと共同作業で企画を進めることを基本としており、その結果、現場への移行を容易に行うことができる。

３．　客先不満足度システムの開発

客先不満足度モデルデータフローの概要

客先の満足度とは、機能・性能の良さ，保守・拡張性の良さ，客先の要求を満たしている度合，安定性，コストといった要素からなる。このうちの客先の要求を満たしている度合とは、他の要素と異なりこれが欠けることによって客先が不満になるといった性質の尺度である。そこで、この度合を客先不満足度と名付けた。ここでは今回開発したシステムの概要について述べる。

客先不満足度システムは、客先不満足度モデル，プログラム信頼性予測モデル，プログラマ実績予測モデル，要員継続性モデル，信頼性成長モデルといったモデル群からなる。客先不満足度モデルデータフローを図３に示す。

要員継続性モデルは、メンバー間の共働年数等の情報を入力することにより、要員継続性（工程間の情報の伝わり具合の指標）を出力する。プログラマ実績予測モデルはキャリア，ドキュメントの記述量等の情報を入力することによりプログラマの実績予測値を出力する。これらのモデルの出力値は、ソフトウェア製造過程における投入資源のデ

ータと共にプログラム信頼性予測モデルへの入力となる。このモデルの出力は、ソフトウェアリリース後の発生エラー数予測値である。リリース後発生エラー数の予測値としては、予測時期によって信頼性成長モデル予測値も用いる。リリース後エラー数の予測値および要求定義時の情報は、客先不満足度モデルへの入力となり、このモデルによって客先不満足度が出力される。本システムの最終的な出力は、客先不満足度であるが、各サブモデルの出力も捕え難いソフトウェア品質を定量化，視覚化するうえで重要な役割りを果す。

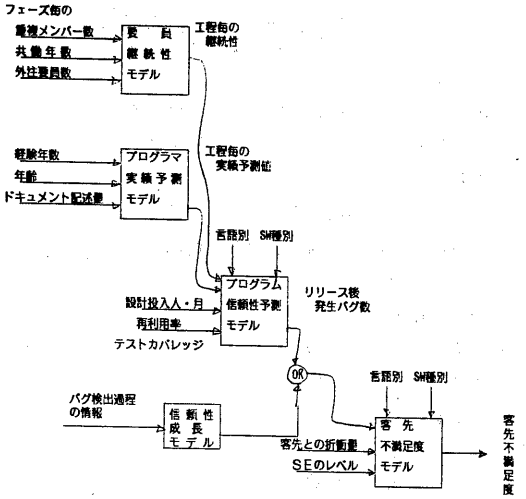

図3　客先不満足度モデルデータフロー

システム構成

本システムは、ＵＸ－３００上で稼動し、主に３つの部分からなる。

(1)　入力・編集プログラム

会話形式の入力，メニューによるガイドによりマニュアルを必要とせずに操作が可能である。

(2)　分析プログラム

本システムは重回帰分析を行なう機能を持つデータが蓄積される毎に重回帰分析を行い、モデル式のパラメータを更新できる。分析を行うデータは、ソフトウェア種別，開発言語別に層別できる。

(3)　予測・出力プログラム

多次元データを視覚的に把握するためにフェイス分析、レーダーチャート等の図形出力機能がある。

4．　各モデルの説明

客先不満足度モデル

このモデルの特徴は、ソフトウェアの品質は顧客が決定するといった観点から、客先の不満を客先の要求とリリース時製品の間のギャップとして定式化しているという点である。

客先不満足度モデルの基本式

本モデルの基本式は次のようになる。

［客先不満足度］＝［客先要求－要求仕様書］＋
［要求仕様書－リリース時製品］
…(1)

式(1) の右辺第１項は、要求定義における欠落を表わしており、これに強く影響を与えると考えられる要素を組込んだ次式で表わされる。

［客先要求－要求仕様書］＝ｆ（客先との折衝量，ＳＥの能力，客先のレベル）　…(2)

式(2) の右辺における各説明変数について具体的な例を示すと次の様になる。

1. 客先との折衝量…要求仕様書ページ数，打合せ回数，インタビュー項目数
2. ＳＥの能力…担当業務・業種の経験，過去作成したドキュメントページ数，１日あたり記述できるドキュメントページ数
3. 客先側レベル…客先からのインタビュー項目数，客先側で作成する要求文書のページ数

ＳＥの能力についてはプログラマ実績予測モデル（後述）の出力結果を用いる。客先のレベルに関しては、上記指標および、ＳＥに対するアンケートによる数量化を併用する。

式(1) の右辺第２項は、要求仕様書の実現度を表わしており次式で表わされる。

［要求仕様書－リリース時製品］＝α×［出荷後バグ数］
…(3)

（αはソフトウェア種別および使用頻度を考慮した１件あたりバグ量の期待値）

ただし、

$$\text{［出荷後バグ数］} = \begin{cases} g(x) & (t \leqq t_{test}) \\ h(t) & (t_{test} < t) \end{cases}$$

g(x)　…プログラム信頼性予測モデル

x　　…再利用率および投入資源

h(t)　…信頼性成長モデル

t_{test}…検査工程開始後、信頼性成長曲線が十分に推定される時刻

以上の関係を図４に示す。

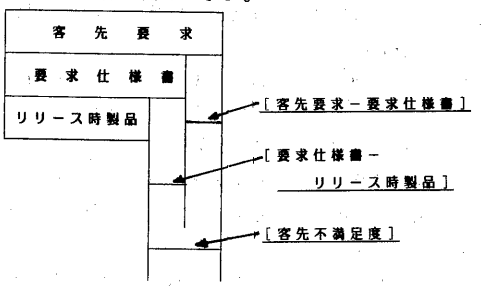

図4

本モデルは、リリース前に客先の不満足度を予測するモデルである。前述したようにその入力は、ほとんどがソフトウェア開発工程中に投入される資源の量である。このため工程途中であっても、未定のデータを見積って入力することにより予測が可能となる。ただし［出荷後バグ数］については当初は、プログラム信頼性予測モデルの予測値を用いるが、検査工程に入りある程度の数のバグが検出された後は、信頼性成長曲線モデルによる予測値を用いる。その理由は、前者に比べて後者は製品自体から実際にバグを検出する過程から予測を行なうモデルであるため、リリース直前にならないと予測できない半面、精度の点では前者に勝るからである。

プログラム信頼性予測モデル

プログラム信頼性予測モデルは客先不満足度モデルのサブモデルである。前述の客先不満足度モデルにおいて説明した客先不満の要因の一部、すなわち、要求仕様書と作製されたプログラム（リリース時）とのギャップをリリース後発生バグ数の形で予測する。本モデルは、ソフトウェア製造プロセスから収集されるデータを用いるため検査工程に入る以前であってもソフトウェア信頼性について予測することができる。すなわち、未定のデータを見積って入力することより、ソフトウェアライフサイクルのより早期において、将来投入すべき資源（人，物，時間，方法論）に関する意志決定についての支援を行なうことができる。

プログラム信頼性予測モデルの基本式

本モデルの基本的発想は、十分に資源を投入して作成したプログラムの信頼性は高いということである。ここでは信頼度をバグ数（［要求仕様書－リリース時製品］のバグ）で表わす。また十分資源を投入した率は、設計，製造工程の種々のデータを引数とする関数で表わす。

$$\frac{\text{（バグ総数）}}{\text{（プログラムの規模）}} = \alpha \times \{1 - \text{（プログラムの信頼度）}\} \quad \cdots(4)$$

α：比例定数

$$\text{（十分に資源を投入した率）} = f(X1, X2 \cdots, Xn) \quad \cdots(5)$$

X1, …, Xn：設計・製造工程のデータ

$0 < f(X1, \cdots, Xn) < 1$

さて、ここで再利用した部分をどのように取り扱うかが問題になる。（この再利用は、信頼性を論じる時に必ず問題となる）ここでは単純に、既製のプログラムを再利用した成分にはバグは存在しないという仮定をおく。

$$\text{（プログラムの信頼度）} = \text{（再利用率）} + (1 - \text{再利用率}) \times \text{（十分資源を投入した率）} \quad \cdots(6)$$

とする。再利用率の定義は、文献[5] に準じる。

(4),(5),(6) の３式は、いわばモデルのモデル公理的“仮説”であり、以後はこれらの式が正しいと仮定した上で議論を進めていく。

式(4),(5),(6) において、（バグ総数），（プログラム規模），（再利用率）は、それぞれ厳密な定義が与えられる。ここでは特に、最も抽象的な指標である（十分資源を投入した率）について詳細化する。“十分資源を投入した率”とはテスト工程前のプログラム作成に投入した資源（人，物，時間，方法論）を、プログラムの規模（物理的かつ論理的規模）で正規化したものと考える。すなわち、

$$\text{（十分に資源を投入した率）} = \frac{\text{（プログラム作成に投入した資源）}}{\text{（プログラムの大きさ・難しさ）}}$$

$$= \gamma \frac{X1^{\alpha 1} \cdot X2^{\alpha 2} \cdot X3^{\alpha 3} \cdots Xn^{\alpha n}}{Y1^{\beta 1} \cdot Y2^{\beta 2} \cdot Y3^{\beta 3} \cdots Ym^{\beta m}} \quad \cdots(7)$$

分子の“プログラム作成に投入した資源”の説明変数は、

X1：外部仕様設計工数＋内部仕様設計工数＋コーディング工数

X2：外部仕様書と内部仕様書のページ数

X3：設計時のレビュー回数および延時間

X4：プログラマ実績

X5：要員継続度

等を考える。また分母の“プログラムの大きさ・難しさ”を説明する変数としては、

Y1：Step数

Y2：外部仕様設計工数

Y3：要求仕様書ページ数

等を考える。

プログラマ実績予測モデル

ソフトウェア品質に占める、人的資源のウェイトはかなり高いことが経験的に知られているが、総合的な管理を行なう立場では、これを定量化して把握することが必要である。プログラマ実績予測モデルはプロジェクト各工程におけるＳＥ，あるいはプログラマの実績のランクを予測するものである。

各工程での実績をその工程に関連の深い因子の得点により表現し、得点を５段階に分けて評価する。

要員継続性モデル

プログラマ実績予測モデルは、個々の要員の実績をポイント化するモデルであった。これに対し、プロジェクトメンバー間のコミュニケーションの良さの度合および参加プロジェクトに関する習熟度合を数量化するモデルが、要員継続性モデルである。モデルの出力は２つの工程間の継続性といった形で表わされる。

要員継続性モデルの基本式

$$\text{（要員継続性）} = \text{（メンバ重複度）} + \text{（メンバ間親和度）} \quad \cdots(8)$$

ここで、式(8) の右辺第１項は対象とするプロジェクトに関する習熟度を表わす指標である。工程が変る際の情報

の伝わり具合はメンバの変更がない場合のみ完全で、変更がある場合は、新しく参加したメンバが対象とする工程に関して経験が豊富なほど良いと考える。そこで、メンバ重複度を以下のように定義する。

（メンバ重複度）＝（前工程からのメンバ残存率）＋
α×（対象工程からの参加者率）×
（対象工程での経験年数の平均値）
α：比例定数

次に式(8) の右辺第２項であるが、これはプロジェクトメンバー間のコミュニケーションの良さを表わす指標である。工程間での情報の伝わり具合は、前工程作業者と後工程作業者の共働年数が長い程良いと考える。そこで、メンバ間親和度を次の様にする。

（メンバ間親和度）＝（メンバ間の共働年数平均値）

５．　客先不満足度システム運用例

客先不満足度システム全体の流れを例を用いて説明する。ここでとりあげたのは、当社で社内用に開発した支援ツールの例である。Ｃ言語約３万ステップで記述されており、７名のメンバーで開発した。この場合の客先としては、ツールを使用する部所の人員をみたてて要求定義時の折衝量を測定した。

図５はプログラマ実績予測モデルの出力例である。各プロジェクトメンバーの参加した工程にはポイントが表示されるためこれにより、どのようなメンバーがどの工程に配されていたかが一見して把握できる。また、メンバー個々のポイントから、プロジェクトとしてのポイントを予測し許容値を満足しているか、あるいは標準値に比べてどうであるかのチェックを行なうことができる。このプロジェクトの場合、全工程を通じて実績予測値は標準値を上回っていることが見てとれる。図６は要員継続性モデルの出力例である。工程間および全工程を通じての継続性計算値および標準値が数値とグラフで表示される。これにより従来行なえなかった人的要因からなる工程間の情報伝達度の定量化が可能となる。このプロジェクトの場合、同一組織内のメンバのみで開発を行なったため継続性はやや良くなっている。

これらの人的投入資源に関する情報がプログラム信頼性予測モデルの入力データとして式(7) のX4, X5 として用いられる。この場合新規開発のソフトウェアであったため再利用率は社内標準を下回った。プログラム信頼性予測モデルを用いて出荷後バグ数を予測した場合の出力例を図７に示す。このプログラムの場合顔の表情は中庸でありバク数の予測値は新規開発ということもあり、多少多い。予測した時点はコーディングを行なう工程であったため、この時点で未定のテストカバレッジ値には標準の値を入力した。そのためリリース後発生バク数予測値は多くなった。そこでテストおよびデバッグに力を入れてテストカバレッジを高めるように信頼性の向上を計った。このことは信頼性成長モデルでも確かめられた。図８は当社のソフトウェア品質管理システムＳＰＡＲＣにおける信頼性成長モデルの出力例である。ここではＳ字モデル[4] を用い、これによって得た予測値は前出の回帰分析による推定値を約15%上回り、本プロジェクトではバグ総数が標準値以上であったことが確認された。

一方客先との折衝量では、相手が社内部門であり、かつ相手が専門家であったこともあったためコミュニケーションが良く厳密な要求定義が行われた。図９は客先不満足度モデルの出力例である。要求定義が厳密に行われたため口が笑っている。また検査工程に力を入れたためテストカバレッジは向上し、目は丸い。またこれによりリリース後のバグ数は減少し、眉のつり上がりも少くなっている。

本システムを用いることによって、検査工程に入る前に検査の目標水準が設定できたため、出荷基準にそったソフトウェアを納期どおりにリリースすることが可能となった。

＊プログラマ実績予測モデル

ソフトウェア（システム）名 マイコンクロス V1.0

	要求定義	外部仕様設計	内部仕様設計	プログラム作成	テスト
標準値	2.2	2.1	2.8	2.5	2.2
期待値	4.1	3.1	3.1	3.4	2.6
山城	4.1	4.8	4.7		
大田		2.4	2.4	3.6	
加藤		3.1	3.9	3.9	3.5
田沢		2.3	2.0	2.6	1.8
渡辺			1.6	2.0	1.6
吉田			4.6	4.7	3.4
小林			2.6	3.5	2.4

図５　プログラマ実績予測モデル出力例

＊要員継続性モデル　OUT−PUT＊

ソフトウェア（システム）名 マイコンクロス V1.0

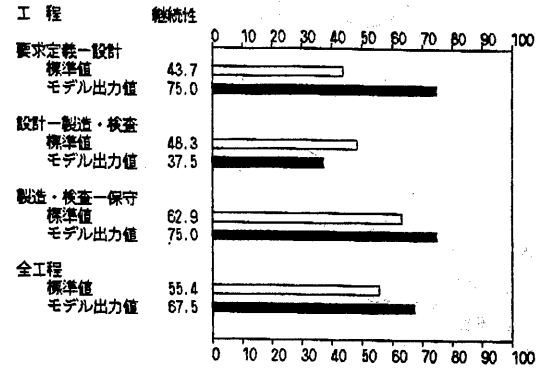

図６　要員継続性モデル出力例

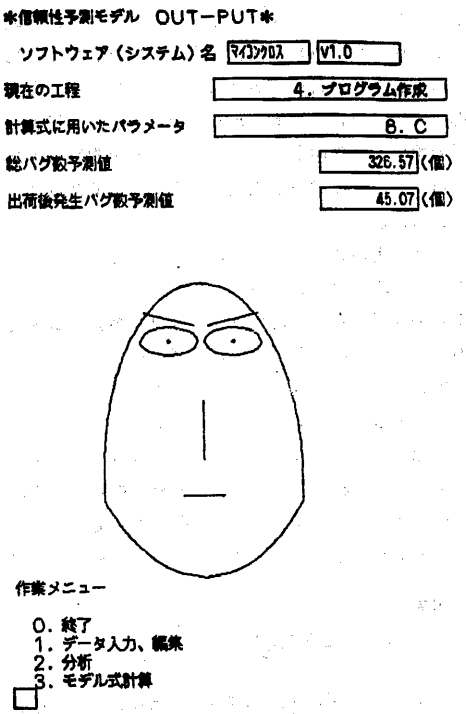

図7　プログラム信頼性予測モデル出力例

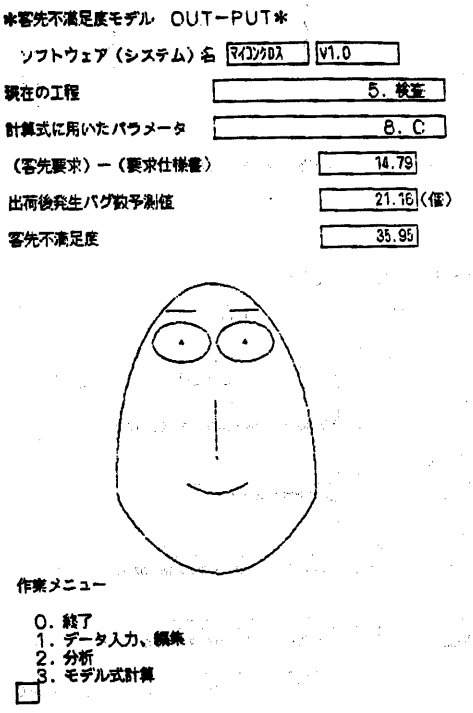

図9　客先不満足度モデル出力例

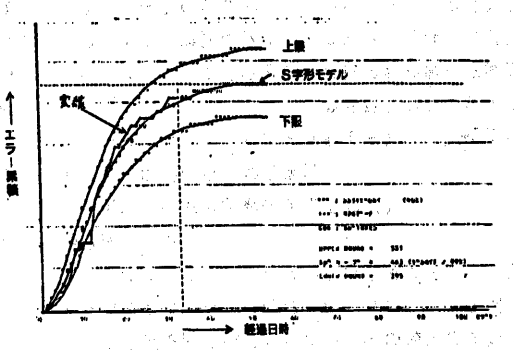

図8　信頼性成長モデル出力例

6．　おわりに

ソフトウェア生産現場でソフトウェアの品質を予測し不具合が表面化する以前に対策をとる手法及びシステムについて説明してきた。本稿では、全モデルデータフロー[3]の一部しか述べていないが、ソフトウェア品質は様々な側面を持つ対象である。そのため、現場のソフトウェア管理システム[6]と連動させ、更に多くの側面からの評価に取組んで行きたい。

また、本稿では触れなかったが、超高信頼性や使い勝手の向上等に対する要求が高まっている。今後、こういった方面の品質を適確に評価する研究も必要となってくると考えられる。

参考文献

[1]　新井，田村，岡村，「システム計画技術の動向」，東芝レビュー，1980年12月，vol.35，no.13，pp.1128-1131

[2]　松村，小島，鈴木，「要求分析技法」，東芝レビュー，1983年10月vol.38，no.11，pp.962 − 965

[3]　中村，内平，佐藤，水谷，「ソフトウェア品質評価システムESQUT(1)〜(3)」，「情報処理学会第29回（昭和59年後期）全国大会講演論文集」，pp.673〜678 1982年 9月

[4]　山田，尾崎，「ソフトウェアの信頼性成長モデルとその比較」，「電子通信学会論文誌」vol.J65-D，no.7，pp.906 −912，1982年 7月

[5]　佐藤，「ソフトウェア開発データ項目と計測法」「情報処理学会第２８回全国大会講演論文集」4B-13，pp.651〜652，1984年 3月

[6]　佐々木，朝日，飯塚，「ソフトウェアの生産管理」，東芝レビュー，1983年10月，vol.38，no.11，pp.957-961