ソフトウェア工学　87－10
（1992. 9. 29）

ソフトウェア品質評価支援システムＥＳＱＵＴの適用と評価

小笠原秀人　　平山雅之

（株）東芝　システム・ソフトウェア技術研究所

あらまし

従来より，ソフトウェアの品質については，品質の定量評価が大きな問題となっている．我々は，ソースコードのみでなく，設計の品質も含めたソフトウェア品質定量評価技法に関して，ＥＳＱＵＴの研究／開発をすすめている．ＥＳＱＵＴは，各開発工程ごとに品質特性モデルとメトリクスを持ち，ソフトウェアの定量評価を支援するシステムである．現在，ＴＦＦで記述された設計書と，Ｃ言語で記述されたソースコードの品質を定量的に評価することができる．本報告書では，ＥＳＱＵＴシステムの概要と，実際のプロジェクトへの適用・評価結果について説明する．

和文キーワード　　品質特性モデル，品質メトリクス，品質計測，品質評価

Evaluation of software quality management system ESQUT

Hideto Ogasawara　　Masayuki Hirayama

System and Software Engineering Laboratory, Toshiba Corporation

70 Yanagi-cho Saiwai-ku Kawasaki, 210 Japan

Abstract

This paper covers a software quality evaluation technique. The software quality quantitative estimation has been a major issue these days. The ESQUT (Evaluation of Software Quality from User's viewpoinT) sytem is constructed from software quality measurements and tools. We have introduced quality metrics of design as well as source code in this system.

This paper outlines a quality quantification technique in this system, describes examples of its application to actual projects and of its evaluation.

英文 key words　　quality model, quality metrics, quality calculate, quality evaluation

1 はじめに

現代社会において人間が頼っているシステムの多くはソフトウェアを含んでおり，システムの機能が拡大するにつれてソフトウェアが大規模化，複雑化し，需要量も急速に増大している．この結果，ソフトウェアのバグが社会に与える影響も増大している．このような背景をもとに，近年，ソフトウェアの品質に関する要求が年々高いものとなってきた．ソフトウェアの品質については，常にバグを中心とした信頼性が着目されてきた．しかしながら，ソフトウェア品質に対する要求は，この範囲にとどまっていないのが現状である．大規模化，複雑化してきたソフトウェアについて，その品質をテスト工程だけで作り込むことができないことは，広く認識されはじめている．すなわち，ソフトウェアの品質を設計工程を含めた上流工程から作り込むことが必要となってきている．

我々は，ソフトウェア品質定量評価についてのより効果的な支援方法を検討するために，「ソフトウェア品質評価支援システムＥＳＱＵＴ[1]」の研究／開発をすすめてきている．ＥＳＱＵＴは，各開発工程ごとに品質特性モデルとメトリクスを持ち，ソフトウェア品質の定量評価を支援するシステムである．現在，ＴＦＦ[2]で記述された設計書と，Ｃ言語で記述されたソースコードの品質を定量的に評価することができる．ＥＳＱＵＴの研究では，これまで，品質特性モデル／メトリクスの特定を重点的にすすめてきた．この問題については，今までの研究より，品質特性モデルの絞り込み，メトリクス基準値の設定など，その有効性について結論が得られている．

本論では，“何をどのような観点から計測／評価するのか”ということから，“どんな方法で，そしてどのタイミングで計測／評価するのか”，“分野や個人差を考慮したメトリクス基準値の設定方法”ということに観点を移し，検討した結果を報告する．

2 従来の問題点

ソフトウェアの品質評価技術，計測技術については従来より研究が盛んに行なわれている．例えば，品質評価技術には，品質に関する概念，品質特性モデル，メトリクス，評価方法などがある．しかしながら，これまでのアプローチの中から，一般的に普及／定着したシステム，方法論はあまり存在しない．従来のアプローチにおける問題点を整理すると，以下のようになる．

1. 品質特性モデル／メトリクスの有効性
   いままで，多くの品質メトリクスが提案されてきた．しかし，それぞれのメトリクスを利用するタイミングや，分野に依存した部分を考慮した基準値の設定方法など，実プロジェクトに適用するには，解決すべき問題が残っている．

2. テスト工程主導の品質評価
   バグ数はソフトウェア品質の中では，主に，信頼性に関するファクタとして考えることができる．しかしながら，このようなテスト中心のアプローチはソフトウェアの品質を考えた場合，そのごく一部を保証しているにすぎない．また，テスト中心の品質評価は，ソフトウェア開発工程のうちの一部についての評価であり，仕様，設計などの品質を評価したことにはならない．

3. 品質評価システムの使用性
   従来のソフトウェア品質評価システムでは，対象ソフトウェアの品質計測に手間を要したため，確実な計測が実施されることはまれであった．とりわけ，開発サイドからみると，品質あるいは管理データを収集されることは，実際の開発作業以外の作業が発生するため，データ収集作業は負荷の増大とみなされていた．

4. 評価／誘導ルーチンの問題
   成果物に対する品質を計測できたとしても，計測された品質データをどのように分析／評価して，品質を向上させるための対策をたてるのか明確になっていない場合が多く，品質定量データの有効活用がなされていない．

ソフトウェア品質を評価するシステムは，これらの要因を十分検討することが必要であるが，現在，提案されている品質評価システムは，これらの要因の分析が希薄である．以上の問題をふまえて，今回我々は，上記３の問題に特に注目し，開発担当者に負荷をかけず，確実に品質データが計測できるシステムを開発した．さらに，開発したシステムを実プロジェクトへ適用し，上記問題に対して考察を加えた．

3 ＥＳＱＵＴシステム

最初に，ソフトウェア品質定量化のための基本的な考え方を，「品質モデル」，「品質の等価性」，「品質特性モデル／メトリクス」について述べる．そして，これらの考え方を取り込んだＥＳＱＵＴシステムについて，その特徴を中心に説明する．

3.1 品質モデル

ソフトウェアの品質を考えた場合，以下の２点がポイントとなる．

管理サイクル　Plan，Do，Check，Action

管理対象　作業の品質と成果物の品質．ここでいう作業とは，ＷＴ(ウォークスルー)，ＤＲ(デザインレビュー)など，直接，成果物（製品）としてとらえることはできないが，成果物と密接に関係するものである．

管理サイクルの各段階で必要となる技術は，以下のとおりである．

1. Plan　【管理モデル】
   どのような開発モデル，品質メトリクスを採用して，品質管理を行なうのかを定義する技術．

2. Do　【計画技術】
   各工程ごとの品質メトリクスの目標値設定技術．

[1] Evaluation of Software Quality from User's viewpoinT
[2] Technical description Formula for Fifty steps/module design

3. Check 【モニタリング技術】
品質メトリクス計測結果のビジュアル化などの技術.

4. Action 【分析／評価技術】
品質計測結果の分析／評価，診断および高品質誘導などのフィードバック技術.

これらの管理対象と管理サイクルを合わせて，図1のようにモデル化することができる.

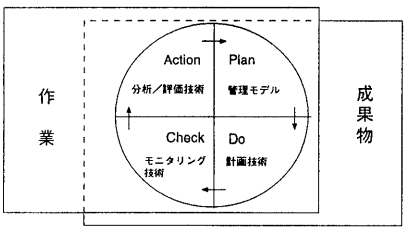

図 1: 品質管理モデル

3.2 品質の等価性

品質特性モデルは，製品としてのソフトウェアを前提としたものであるが，この特性は，ソフトウェアの開発過程の中から生みだされたものである．したがって，ソフトウェアの開発過程においても，同様の，あるいは類似した品質特性を設定することは可能であると考えられる.

ＥＳＱＵＴシステムでは，ＴＦＦで記述された設計書とソースコードの品質が等価であると考え，設計段階における品質メトリクスを設定している.

3.3 ＥＳＱＵＴ品質特性モデル／メトリクス

ＥＳＱＵＴで定義している品質特性は,「機能量」,「理解性」,「複雑さ」の３つである．各品質特性の内容を，以下に示す.

機能量 モジュールが実現する大きさを示す属性

理解性 モジュールの読みやすさ，理解しやすさを示す属性

複雑さ モジュール内処理の複雑さを示す属性

これらの品質特性にもとづいて，ＥＳＱＵＴで計測するメトリクスが設定されている．ＥＳＱＵＴ品質特性モデルとメトリクスの関係を，図2に示す.

3.4 ＥＳＱＵＴシステムの特徴

今回開発したシステムは，開発担当者の多くがパーソナルコンピュータを利用しているという現状をふまえて，J-3100/DynaBook シリーズの MS-DOS 上で稼働する.

ＥＳＱＵＴシステムは，品質評価プロセスの自動化支援を目指すものであり，以下の特徴をもつ.

1. ソフトウェア品質の定量評価
図2に示した品質特性モデル／メトリクスを採用した．これらのメトリクスについては，平均値をはじめとする

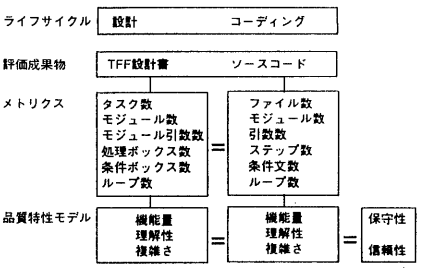

図 2: 品質特性モデルとメトリクス

統計的特性が十分に把握されたものである．したがって，これらの品質特性モデル／メトリクスを使用することにより，対象とするソフトウェアの品質の一側面を定量的に評価することが可能となる.

2. ライフサイクル／開発プロセスの重視
ソフトウェアの開発プロセスを重視し，設計工程，作成工程等で等価な品質特性モデルを採用している．これにより工程間での品質の相対的な比較評価を可能とするとともに，工程間品質の劣化などを早期に評価することができる.

3. 品質計測の自動化
計測対象となる開発担当者の負荷を軽減するために，開発スケジュールと連動した成果物品質の自動計測を実現した．これにより，開発プロセス中での品質評価に最適な時点での品質計測／評価が可能となる.

4. 品質計測対象成果物の自動識別
品質計測では，計測する成果物（TFF シート，ソースコード）を識別して計測する必要がある．しかしながら，これらを指定する作業は，対象システムの規模が大きくなるとかなり大がかりなものとなってしまう．したがって，本システムでは，指定されたディレクトリ以下の計測対象成果物を自動的に識別できるようにした.

5. 品質管理帳票の提供
品質計測結果の開発担当者および開発管理者へのフィードバックを行なうために，それぞれの担当者，管理者向けの品質管理帳票を提供する．これにより，開発担当者は自らの開発したソフトウェアの品質傾向の確認が容易となる．また，開発管理者は，ＤＲなどの参考資料として計測結果を利用することが可能となる.

上記の各項目がどのように実現されているのかを，図3に示す．まず，利用者は，計測スケジュール，ターゲットディレクトリ，および品質基準値を設定する．ここまでの設定が終ると，あとは，ＥＳＱＵＴシステムが，設定されたスケジュールにしたがって自動的に計測を行なう．これによって，利用者にほとんど負荷をかけずに，成果物の品質を計測することが可能となる.

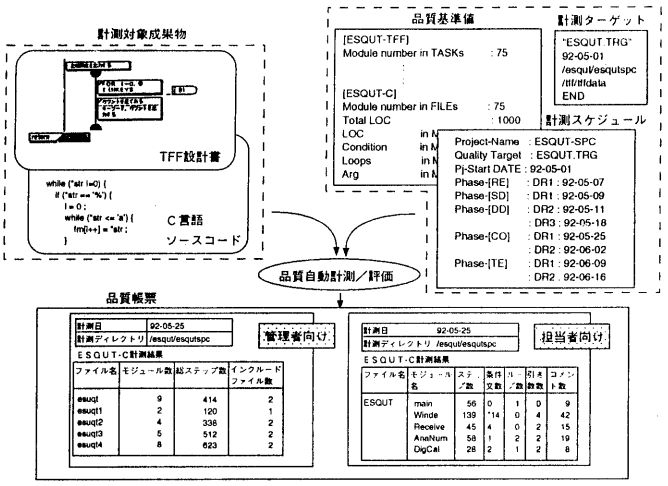

図 3: ＥＳＱＵＴシステムの全体象

4　ＥＳＱＵＴシステムの適用

今回開発したＥＳＱＵＴシステムを実プロジェクトへ適用し，その効果について検討した．ここでは，適用目的，対象システム，および適用方法について述べる．

4.1　適用目的

ＥＳＱＵＴシステムの使用性を中心とした以下の項目に関する検討を，ＥＳＱＵＴシステム適用の目的とした．

1. ＥＳＱＵＴシステムの使用性
   各開発担当者のマシンにＥＳＱＵＴシステムをインストールし，スケジュールにしたがって成果物の品質データを計測した．その経過をとおして，開発担当者および開発管理者の負荷を調べる．
2. 品質特性モデル／メトリクスの有効性
   ＥＳＱＵＴ品質特性モデル／メトリクスにもとづいて計測された結果と，バグとの関係を調べることによって，メトリクスの有効性を確認する．また，対象部門に応じた品質データの特性分析／評価を行ない，ＥＳＱＵＴシステムの有効な活用方法を確立する．
3. 品質評価／誘導ルーチンの確立
   スケジュールにしたがって品質データが計測できるので，複数回にわたる計測結果から，品質評価／誘導に関する効果的な支援方法を検討する．また，ＥＳＱＵＴシステムを開発過程に組み込むための運用ルーチンを確立する．

4.2　対象システム

適用対象としたシステムは，当社のある部門で開発した，オンラインデータ処理システムである．開発期間／体制を以下に示す．

【開発体制】'91 年 10 月～'92 年 6 月

【開発体制】リーダー　　1 名
　　品質管理者　1 名
　　開発担当者　6 名

開発体制の特徴としては，リーダーの下に品質管理者をおき，成果物に対する標準化や仕様の確認等を徹底させたことである．また，設計段階ではＴＦＦを利用していなかったので，ＥＳＱＵＴシステムの適用は，Ｃ言語で作成されたソースコードに対してのみ実施した．

4.3　ＥＳＱＵＴ適用方法

ＥＳＱＵＴの運用は，対象部門の品質管理者が，ＥＳＱＵＴシステムで設定したスケジュールにあわせて定期的にデータを収集し，それを我々が受け取り，データの分析をして品質管理者に返す，というルーチンで行なった（図 4 参照）．ＥＳＱＵＴの効果的な活用のためには，分析結果にもとづいてＷＴを実施し，検討を加えることが大切である．しかし，今回の適用は開発途中（７～８割出来た段階）から行なったこともあり，ＥＳＱＵＴ計測結果にもとづく開発へのフィードバックを前提としたＷＴは実施しなかった．すなわち，担当者には，計測結果にもとづいた品質均一化のための指針を示さなかった．

5　ＥＳＱＵＴシステム適用結果

今回開発したＥＳＱＵＴシステムを実プロジェクトへ適用した結果を，４章で述べた目的にしたがい説明する．

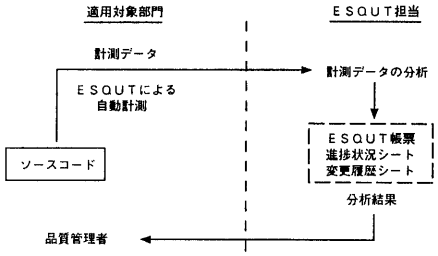

図 4: ＥＳＱＵＴ適用ルーチン

5.1 ＥＳＱＵＴシステムの使用性

ソフトウェアの品質を計測することで重要なポイントは,「定期的な品質データの収集」と「計測の高速化」である.

本適用では，各開発担当者とも，１週間間隔で計測スケジュールを設定した．したがって，１週間ごとに，マシン起動時にＥＳＱＵＴシステムが稼動し，設定したターゲットディレクトリ上の成果物の自動計測が行なわれた．任意の開発担当者の成果物の計測時間は，表１のようになった．

表 1: 計測負荷の分析

計測対象	データ	計測時間
計測ディレクトリ	28	70 sec
総モジュール数	74	
総ステップ数	4574	

計測対象としたシステムは，１タスク１ディレクトリという形式で開発が行なわれていたため，ディレクトリ数が多く，計測負荷がかかるシステムであった．

これらの結果から,「定期的な品質データの収集」と「計測の高速化」については，一人あたりで開発できる量を考慮すると，実用に耐えうるシステムであることがわかる．

また，開発担当者および開発管理者の両方の立場で，ＥＳＱＵＴシステムを利用する場合の負荷について検討を加えた．

1. 開発担当者の負荷
   ＥＳＱＵＴシステムのインストール時に,「開発スケジュール」,「品質基準値」,「ターゲットディレクトリ」を設定しておくだけで，開発成果物を自動計測するので，開発担当者にはほとんど負荷はかからない．

2. 開発管理者の負荷
   計測した成果物を容易に帳票形式で出力できるので，開発管理者にかかる負荷も非常に小さいものといえる．また，品質基準値と計測結果を比較して，基準値を超えているデータについては，“＊”マークが付加されるので，開発管理者にかかるデータ分析の負荷も軽減される．

さらに，今回開発したＥＳＱＵＴシステムは，スケジュールによらない任意の時点でも計測ができるので，開発状況に対してより柔軟な対応が可能となっている．

5.2 品質特性モデル／メトリクスの有効性

今回対象としたシステムの総成果物量を，表２に示す．

表 2: 対象システムの総成果物量

開発担当者	経験年数	モジュール数	総ステップ数	装置
A	3	100	11,780	A
B	4	253	10,348	B
C	1	18	845	B
D	1	74	4,574	C
E	3	226	16,318	C
F	3	54	2,074	C

ＥＳＱＵＴシステムの有効性を評価するために，システムテスト時のバグと計測結果との関係について，分析を行なった．また，対象部門の品質データの特性については,「個人／分野による特性」と「ＥＳＱＵＴメリクスの基準値」に着目し，分析を行なった．これらの分析結果にもとづいて，ＥＳＱＵＴシステムの有効な活用方法を検討した結果を，以下に示す．

5.2.1 システムテスト時のバグと計測結果の関係

開発担当者Ｅが開発した部分に関して，システムテスト時に発生したバグの件数は６件であった．システムテストとは，モジュールの結合試験から出荷前の総合試験までである．Ｅの開発した部分は，Ｄ～Ｆで共同開発した一部分である．

今回の適用において，メトリクスの基準値を以下のように設定した．これらの値は，開発部門の経験にもとづいて導出した値である．

ステップ数 ≧ ２００
条件文数 ≧ ２０

テストの結果，発生したバグ６件のうち，３件が基準値オーバーのモジュールであった．そこで，開発者Ｅの基準値内のモジュールと基準値オーバーのモジュールのバグ発生頻度について分析した．開発者Ｅのバグ発生状況を，表３に示す．

表 3: バグ発生状況

モジュール	バグあり	バグなし	計
基準値オーバー	3	12	15
基準値内	3	208	211
計	6	220	226

この結果から，基準値オーバーのモジュールのバグ発生率が，基準値内でおさまったモジュールより，明らかに大きいことがわかる．

基準値オーバーのモジュールから検出された３つのバグの原因は，以下のとおりであった．これらのバグは，処理の抜け，タイミングの問題に起因するものと考えられる．ステップ数，条件文数の増加によってモジュール内処理の複雑さが増したことが，バグの一つの原因であると考えられる．

- 表示画面の自動更新要求が発行されない．
- ブザー出力要求を出したのち，セマフォをリセットする処理が抜けている．
- 画面モードを"再生停止中"にするタイミングがおかしい．

一方，基準値内のモジュールから検出されたバグの原因は，「仕様解釈の不整合」，「ロジックの変換ミス」に起因するものであった．これらのバグは，「上流工程で発生したバグ」，「論理的なバグ」であるので，ＥＳＱＵＴシステムの計測結果をトリガとして原因を追求することはできない．このようなバグに対しては，設計とＷＴを確実に行なうことが最も重要である．

以上の結果から，ＥＳＱＵＴメトリクスを計測し，基準値オーバーモジュールを早めにチェックすることで，プログラムが複雑になることをおさえるとともに，プログラム作成中のミスを早期に発見することができるはずである．

5.2.2 個人／分野による特性

ＥＳＱＵＴシステムによる計測結果にもとづいて，各開発担当者の個人差について分析した．これは，個人によるバラツキがどれほどあるのかを認識することによって，品質の均一化をめざした，標準化の指針（品質基準値の導出を含む）を設定するためである．

各開発担当者の開発した成果物に対して，各計測項目（ステップ数，条件文数，ループ数，コメント文数，引き数数）ごとに分析した結果を，以下に示す．

1. **ステップ数**
   各開発担当者ごとのステップ数のバラツキを，図５に示す．設計に十分時間をかけたシステムであっても，ソースコードの段階では，各開発担当者ごとに差があることがわかる．モジュールあたりのステップ数が多いということは，そのモジュールで実現している処理数が多いと考えられる．このことは，モジュール間インタフェースと，モジュール内の複雑さを増大させる要因となる．このような問題を解決するために，モジュール化の指針，目標とする１モジュールあたりのステップ数などを示すことによって，個人差を縮めていく努力が必要となる．

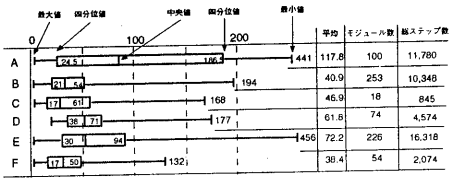

図 5: ステップ数の箱ひげ図

2. **条件文数**
   各開発担当者ごとの条件文数のバラツキを，図６に示す．１モジュールあたりの条件文数が多いということは，そのモジュールではかなり複雑な処理が行なわれているはずである．テストの容易さ，モジュールの読みやすさなどを考慮すると，１モジュールの条件文数が多くならないように，階層化してコーディングすることが必要である．

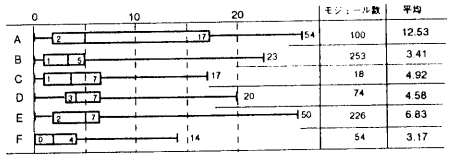

図 6: 条件文数の箱ひげ図

3. **ループ数**
   ループ数は，処理の内容に依存するので，開発対象が同じであれば，個人差はあまりでないようである．

4. **コメント文数**
   コメント文は，プログラムの理解性や保守性を考えた場合，重要な要素となる．しかし，１モジュールあたりのコメント文数だけでは，ステップ数にもバラツキがある以上，品質を評価することはできない．したがって，コメント文に関する評価は，ステップ数あたりのコメント文数であるコメント率で評価すべきである．

5. **引き数数**
   設計段階で十分検討されていたこともあり，引き数に関するバラツキは小さい．

以上，ＥＳＱＵＴ計測結果による「個人差」と「分野による差異」の分析結果をまとめると，以下のようになる．

- 【個人差について】
  コーディング規約が定まっていて，開発対象が同じシステムであっても，個人による成果物量には，大きな差が生じている．成果物量のバラツキが大きいということは，ＥＳＱＵＴ品質特性モデル／メトリクスにもとづく計測結果，すなわち，ソフトウェアの品質のバラツキも大きいといえる．個人差が存在することを認めたうえで，バラツキを小さくし，品質の均一化を目指す努力が必要である．

- 【分野による差異について】
  今回の適用において，品質基準値として利用したデータは，過去の経験から導出した値である．経験値と適用の結果導きだされた値には，相違がある（次節参照）．個人差によるバラツキが同程度と考えると，この差異は，分野による差と考えることができる．

今後，個人差，分野を考慮した品質基準値設定のための方法（指標）を構築することが，ＥＳＱＵＴシステムの普及には必要となってくる．

5.2.3 ＥＳＱＵＴメトリクス基準値の設定

前節で説明のあったとおり，ＥＳＱＵＴメトリクスを設定するには，個人差，分野による特徴を考慮する必要がある．ここでは，個人によるバラツキが，開発担当者Ｄ～Ｆのレベルであると仮定して，品質基準値を導出した．品質基準値の導出は，以下の方法で行なった．

1. ステップ数，コメント率
   ステップ数，コメント率の小さすぎるモジュールも問題があると考えた．したがって，下限と上限の基準値を設定した．ステップ数に関しては，計測結果を対数変換し，そこから下限と上限を設定した．基準値は，「平均値±標準偏差×２」とした．このように設定した理由は，今回の開発が，品質管理者によって管理された状態で行なわれていたので，「まれにしか発生しない値」を“しきい値”，すなわち基準値として設定することが可能と考えたからである．

2. 条件文数，ループ数，引き数数
   下限を設定したとしても，下限以下の値がでた場合，それは，品質を劣化させるものではないと考えられる．したがって，基準値としては上限だけを設定する．基準値は，「平均値＋標準偏差×２」とする．

上記の方法にしたがって設定した品質基準値を，表４に示す．今後，今回適用対象とした部門では，この品質基準値を使ってＥＳＱＵＴシステムを利用する．ただし，品質基準値は不変のものではないので，ＥＳＱＵＴシステムの適用をすすめ，実績データを蓄積することによって，タイムリーに変更していくことが大事である．

表４: 基準値に関するまとめ

計測項目	基準値
ステップ数	10 ≦ ステップ数 ≦ 230
条件文数	17 以下
ループ数	4 以下
コメント文数	0.88 ≦ コメント率 ≦ 0.38
引き数数	4 以下

今回のＥＳＱＵＴシステムの適用にあたって設定した基準値は，今までのＥＳＱＵＴ適用結果に較べて，かなり「ゆるめ」に設定した値であった．開発部門の経験上の値とほぼ同等の結果が得られたことから，表４が，この分野（部門）における特性と考えることができる．

5.3 品質評価／誘導ルーチンの確立

ＥＳＱＵＴシステムで計測された結果を一番効果的に利用できる場面は，ＷＴである．ＷＴは，技術的に内容を理解しているリーダーと担当者間で行なわれるものである．リーダーは，定期的に計測される結果から，「進捗」と「品質」に関する情報をチェックすることができる．

ソースコードに関しては，以下のような観点から進捗のチェックが可能となる．

- 全体的な進捗
- 各ディレクトリごとの進捗
- 各モジュールごとの進捗

「全体的な進捗」の帳票形式を図７に，「各モジュールごとの進捗」の帳票形式を，図８に示す．

計測日	CO-1(92/5/8)	CO-2(92/5/12)	CO-3(92/5/22)	・・・
ディレクトリ数	4	5	8	
ファイル数	11	16	18	
モジュール数	20	28	38	
総ステップ数	774	994	1,473	

図 7: 全体的な進捗

計測日 / 計測モジュール	CO-1(92/5/8)		CO-2(92/5/12)		CO-3(92/5/22)		・・・
	ステップ数	条件文数	ステップ数	条件文数	ステップ数	条件文数	
main	23	3	39	5	59	8	
select_val	18	4	22	4	22	4	
esqut_spc	26	2	35	3	40	3	
get_file	15	1	18	1	42	4	
get_std_val	7	0	16	0	16	0	

図 8: 各モジュールごとの進捗

このような進捗に関する帳票を追加し，開発工程をモニタすることによって，進捗だけでなく，品質の観点からも成果物をチェックすることができる．例えば，これらの進捗の帳票に，以下のような考え方を導入することによって，品質のチェックが可能となる．

1. 設計品質との比較
   設計段階における品質（例えば，ＴＦＦ記述によるボックス数）とソースコードの進捗状況を比較することにより，収束具合やコーディングの十分性をみることができる．また，設計品質とコーディングの品質が乖離していれば，どちらかの品質が十分でないと考えることができる．

2. 変動モジュールの検出
   進捗の帳票から，複数回にわたって，連続的に小さな変更のあるモジュールを検出する．このような状況にあるモジュールは，直接的あるいは間接的に何らかの問題が存在していると考えられる．

さらに，従来の帳票にもとづく品質チェックについては，前項で示したように，基準値オーバーのモジュールを中心にチェックすることによって，早めの対策をたてることができ，品質の作り込みが期待できる．

ＥＳＱＵＴシステムを効果的に活用するためのモデルを，図９に示す．テスト段階では，ＥＳＱＵＴ計測結果から，目安となるテスト項目数を算出し，支援する．

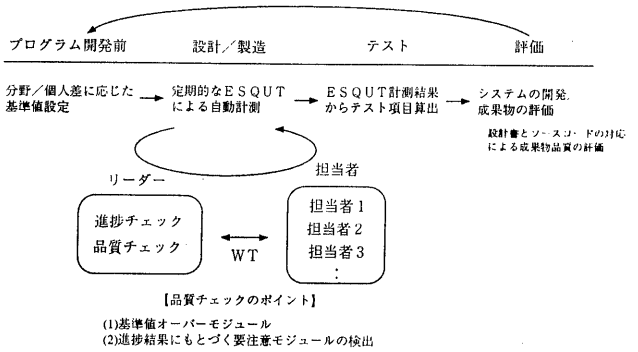

図 9: ＥＳＱＵＴ運用ルーチン

6 まとめ

今回，我々は，開発担当者／管理担当者の「品質向上のための作業負荷」を軽減させるという観点からＥＳＱＵＴシステムを開発し，それを実プロジェクトへ適用した．４章で示した具体的な目的と対応付けて，適用結果をまとめる．

1. ＥＳＱＵＴシステムの使用性
   計測スピードは，許容範囲内で成果物の計測ができることを確認できた．また，スケジュール機能を設定したことで，抜けのない品質データの計測が可能となった．さらに，品質帳票の出力機能によって，開発管理者の負荷も軽減することが確認できた．このような結果から，容易にＥＳＱＵＴシステムを，開発環境に組み込めることが明らかになった．

2. 品質特性モデル／メトリクスの有効性
   基準値によるソフトウェア品質評価の有効性を，バグとの関係により明らかにした．これによって，ＥＳＱＵＴシステムを開発のルーチンにのせ，活用することの有効性が明らかになった．また，各開発担当者の成果物に個人差があることを示し，さらに，分野向けの基準値を設定したことで，プログラム作成のための標準化および評価の指標を示すことができた．

3. 品質評価／誘導ルーチンの確立
   進捗状況の帳票を提示することによって（図７，図８参照），ＥＳＱＵＴシステムを利用した進捗状況把握の方法と，帳票を利用した品質チェックの方法を示した．さらに，ＥＳＱＵＴシステムを開発過程に組み込むための運用ルーチンを提案した．

今回のＥＳＱＵＴシステムの適用例でもわかるように，ソフトウェアの品質評価については，自動化支援技術が今後ますます重要になってくるはずである．本報告で述べた適用例は，ソースコードレベルのものであるが，今後，設計段階からのプロジェクトにＥＳＱＵＴシステムを適用し，有効性を検証していく予定である．

さらに，分散開発環境下におけるソフトウェア開発に対応するために，ＥＳＱＵＴシステムをネットワーク対応とすることなどを検討している．最終的には，ネットワークに対応した，開発ライフサイクル全体にわたる，ソフトウェア品質評価の自動化支援技術を追求していく．

参考文献

[1] 平山他：“ソフトウェア設計品質メトリクスの検証 ― ＩＭＡＰシステムにおける品質保証―”，情処学ソフトウェア工学研報，64-5，1989.

[2] 平山他：“ソフトウェア品質モデルの体系化とその適用評価”，情処学ソフトウェア工学研報，71-11，1990.

[3] 平山他：“ソフトウェア品質保証プロセスの考察”，情処学ソフトウェア工学研報，77-7，1991.

[4] 山田茂：ソフトウェア信頼性評価技術，ＨＢＪ出版局，1989.