第48回 SEA-SPIN Meeting (Feburary 26 -Thursday-, 2004)

テーマ： ソフトウェアテストと品質保証
（テストノウハウとSQAの役割）

主催

SEA **プロセス分科会** (SEA-SPIN)

前々回のデマルコさんを招き古都鎌倉で開催したSPIN特別例会、その直後に関東と関西の中間点の都市、浜松で開催した坂本記念ワークショップと、企画ものが相次ぎましたが、今回は浜松ワークショップでの発表と意見交換の中で出てきた、テストと品質保証に関する(株) 日立システムアンドサービス　奈良隆正　様の貴重な知見を発表していただくことをきっかけとして、テストと品質保証 (SQA) の立場と役割を中心に、互いの問題点・疑問点について意見交換をしたいと考えています。

ソフトウェアテストと品質保証（要旨）

ソフトウェア開発における「テスト」は、「設計」との対称軸として、よく車の両輪にたとえられ、しかも、全開発工程の５０％以上を占める重要なプロセスであるにもかかわらず、ともすれば軽視されがちであるという現状は否定できない。

ソフトウェアの品質保証を考えるとき，テスト技術は最も重要なファクターの一つであり、ソフトウェア工学に残された最重要課題の一つでもある。しかし、ソフトウェアテストに関しては、技術とか技法と呼べるだけの系統的・革新的な方法論はきわめて少ないのが現状であり、多分に個人の経験や能力に依存していることが多い。各企業は，個々の開発プロジェクトを通して、その時々に得た経験やノウハウを体系化し、これを適用し、さらに改善するということを繰り返し行うことによって、ソフトウェアテストのやり方をを体系化、標準化することに努力してきている。

私自身は，ソフトウェア開発の初期の頃から、品質保証活動の大きな柱としてソフトウェア検査 (SQA) の実践に取り組み、それなりの成果をあげてきたと自負している。今回のミーティングでは、ソフトウェア開発、特にテストプロセスにおける SQA の位置付け、実行過程および設計プロセスとの関係(連携)ついて述べ，SEPG及びSQA のみなさんと議論したいと考えている。

**********開催要領**********

1. 日　時：　2004年2月26日(木) 13:00 ～ 17:00

2. 会　場：　労働スクエア東京　B1-A Recreation Room （中央区八丁堀）

3. プログラム：　コーディネータ：　塩谷和範 (SRA-KTL)

13:00 - 13:30 受付及びネットワーキング(情報交換と歓談)
13:30 - 15:00 「ソフトウェアテストと品質保証」（テストノウハウ）
(株) 日立システムアンドサービス　奈良隆正　様
15:00 - 15:15 休憩
15:15 - 16:45 自由討論
16:45 - 17:00 クロージング(次回予告、及び後片付け等)

SEA-SPIN例会予定: http://www.sea.jp/SPIN/meeting.html

==========

7．8　ソフトウェアの検査

当社では、ソフトウェア開発の初期の頃から品質保証活動の大きな柱の一つとして、ソフトウェア検査を実施している。検査担当部門は、開発部門とは全くの独立した組織とし、開発の開始時点から保守の段階まで一環した活動を展開している。

ソフトウェア検査は、ソフトウェア開発におけるプロセスの実行過程および生産物（中間生産物含む）の第3者機関による評価で有り、その業務や手順は、開発の全工程にわたって関与し、開発部門とは密接な連携を取りながら行われる。

本章では、当社におけるソフトウェア検査の方法論について述べる。

7.8.1　開発工程とテスト／検査の関係

ソフトウェアの生産工程とその中における検査の役割および開発プロセスとの関係を**図 7.8.1** に示す。**図7.8.1**は開発方法が主にウォータフォールモデルの場合であり、最近のオープンシステムにおける開発プロセス（主にスパイラルモデル）と検査作業の関係は、それぞれ**図7.8.2**、**図7.8.3**に示した。

ハードウェアの場合と同様、設計･製造部門とは独立した組織である検査部門が、各工程ごとに中間生産物をチェックする体制になっている。設計工程は、設計仕様からコーディングまでの段階と、デバック及びテスト段階に大別される。

検査工程は、前者に対応するドキュメント検査工程と後者に対応する品質監査及び、設計部門でテストの完了したプログラムそのものの検査、すなわち製品検査からなる。

ソフトウェアの検査の中で、ドキュメント検査が重視される理由は、次の2点である。

＜第1点＞

ハードウェアと異なり、ドキュメントがほとんど唯一の中間生産物であり、しかもその品質の良否がソフトウェア自体の品質を左右することから、品質の手先管理としての役割を果たしている点である。

＜第2点＞

とかく各工程の区分があいまいになりやすいソフトウェアの生産過程に対し、中間生産物としての各ドキュメントの検査をすることにより、一工程ずつ確実にチェックすることが可能になる点である。

さらにドキュメント検査は、製品検査のために対象物の仕様を理解する重要なプロセスでもある。

テスト･デバッグ工程は、全工程の 30～40％近くを占め、この工程の品質管理は極めて重要であり、ここでは品質評価手法の一つである「不良予測」及び「探針手法」を用いた品質監査を行っている。最終成果物であるプログラムそのものの検査、すなわち製品検査を開始するに当っては、プログラムが十分な品質に達していることを見極めることが必要である。製品検査の基本的な考え方は、顧客の立場に立ち、主として外部仕様に基づいて検査し、製品としての合否判定を行うことである。

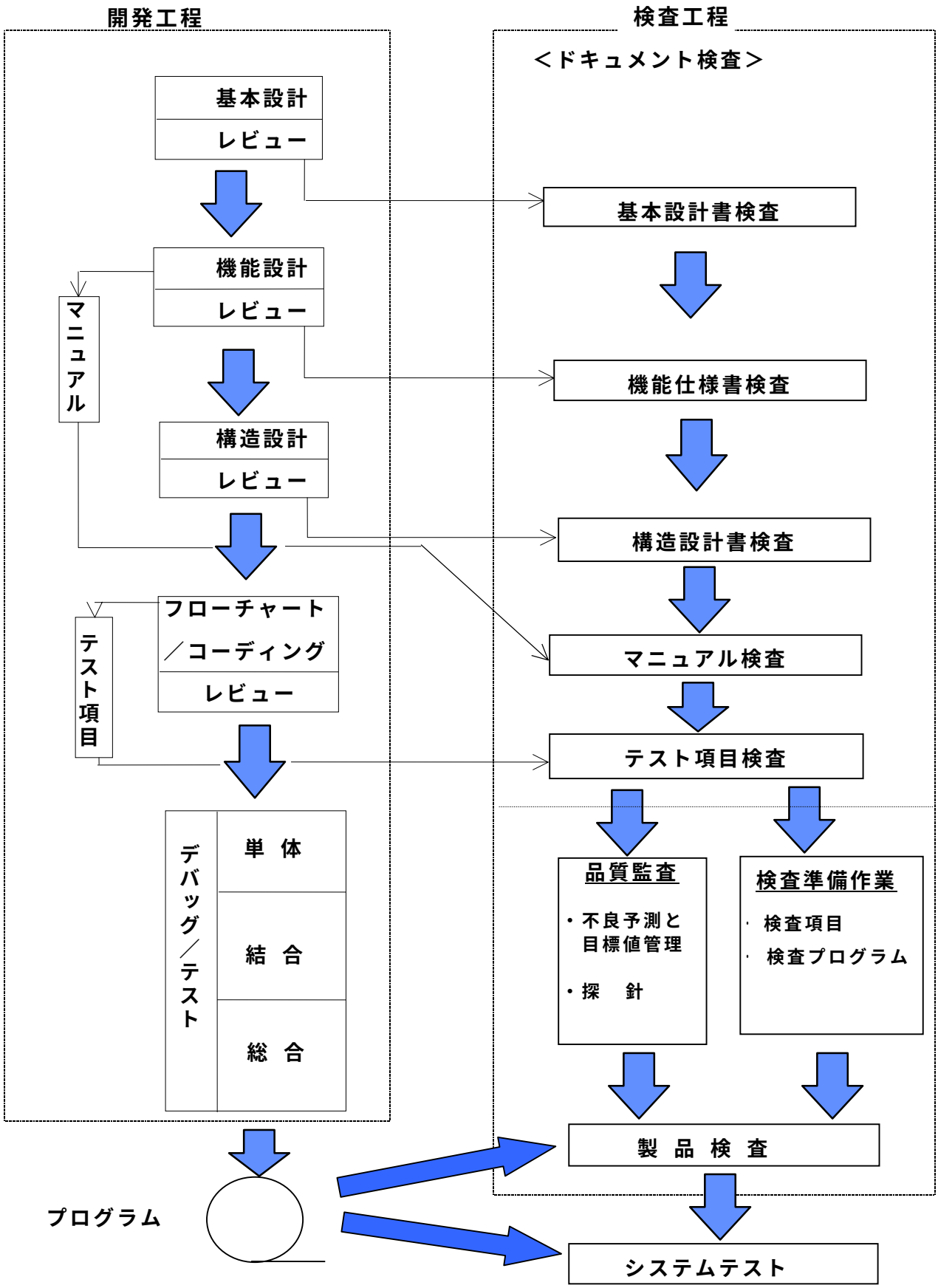


図 7.8.1　ソフトウェア開発工程と検査の役割

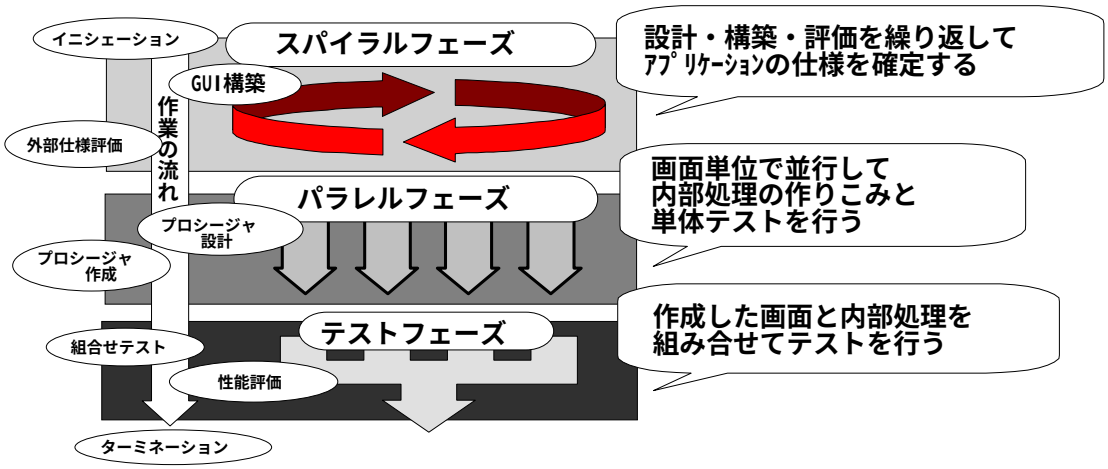


図7．8．2 オープンシステムの開発モデル

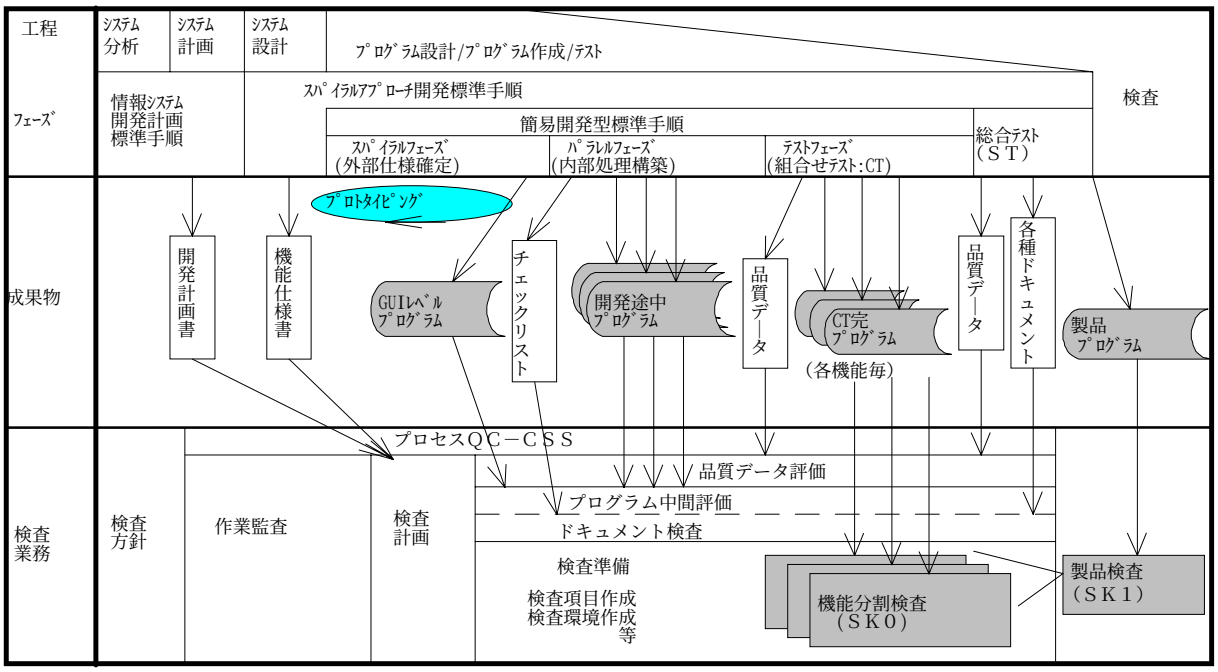


図7．8．3　オープンシステム対応標準検査作業

解説：オープンシステム開発手順は、種々の開発支援ツール個有の機能を最大限に活かすためにツール別に設定され、作業項目と成果物および作業手順は、各ツール固有のものとなる。しかし、オープンシステム型の開発標準手順の大きなフローは、一般的には**図** 7.8.2 に示したように、①スパイラルフェーズ（外部仕様決定）、②パラレルフェーズ（内部処理構築）、③テストフェーズ（組合せテスト）の3フェーズで構成されている。また、この開発方式に対応した検査作業のフローは、**図**7.8.3に示した通りとなる。

7.8.2　品質監査（テスト品質チェック）

品質監査は、テスト工程すなわち作り込まれたバグを摘出する工程に対する活動である。ここでは内在するバグの予測、すなわち摘出すべきバグの目標値を決定し、これに対するバグ摘出の実績・推移を評価するとともに、バグの分析・評価も行う。この活動は、品質目標管理の概念図にしめすように稼働品質

をも含む大きな品質目標値管理サイクルの一部に位置付けられており、評価データの蓄積稼働品質の予測も重要な作業になっている。

(1) バグ予測と目標値管理

バグ予測については、時系列的なバグの累積曲線を信頼性成長曲線（7.8.3章を参照）にあてはめる手法を用い、これに基づくバグ摘出推移を評価している。テスト開始前に上限・下限の2つの曲線から成る標準的なバグ管理曲線、及びバグ摘出目標値を設定する。これは、類似プログラムの実績を参考に、当該プログラムの特徴、プロジェクトの特徴などを加味して決定する。テストが進み、時系列のバグ累積実績が出てきた時点でバグ発生予測曲線により残存バグを推定する。予測曲線または実績が管理曲線の上限、下限を超えた場合は、適切な品質向上施策が必要である。テストの後半では、バグ実績に基づく不良発生予測曲線の収束値に対する現在の実績比、すなわちバグ収束率を重点的に監視する。

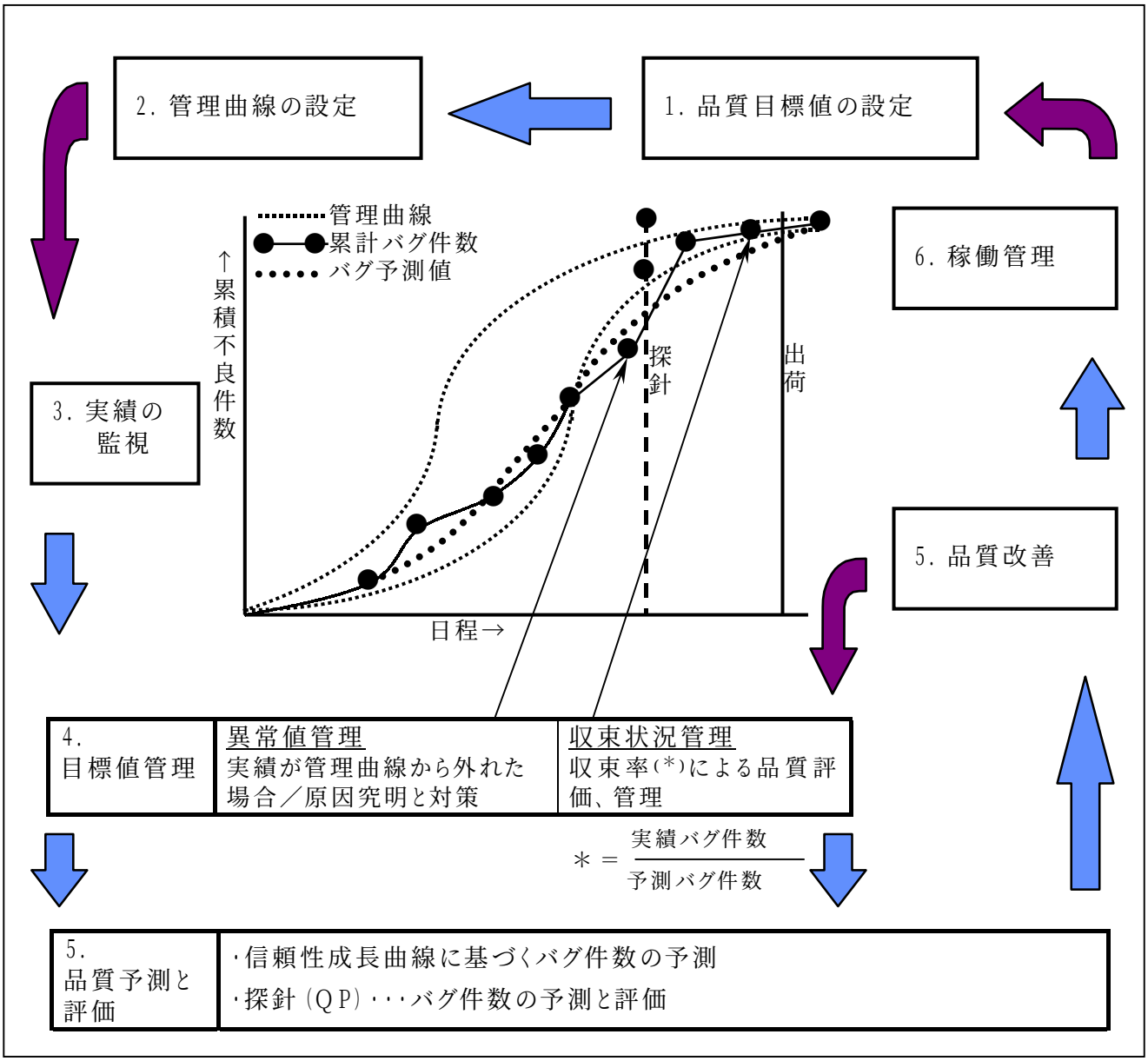


図7．8．4 品質目標値管理の概念図

(2) 探針（QP:Quality Probe）

探針とは、製品検査に先立って、デバッグ･テスト段階における品質を検査が実際に測定･評価するものである。これは製品検査と同様に実際の計算機を使用して実施し、この時の摘出バグ件数から製品検査時のバグ件数を統計的手法で推定する。それにより、バグ内容の分析･評価による弱点の具体的指摘でデバッグ･テスト段階でのバグの先取りを加速し、品質向上施策への指針を与えようというものである。探針は、基本的には検査項目をサンプリングし、その結果から全体の母不良率を求め残バグ件数を推定する方法である。

以下に探針（残バグ推定）の具体的な手順を説明する。

① 探針項目として、検査項目の 10%～20%をサンプリングする。
サンプリング方法は層別抽出方法である。

② 探針項目から出た不良に対して不良率を次の式で求める。

$$\text{標本不良率}: p = \frac{r}{n} \quad \cdots\cdots (1)$$

n：探針項目数　r：不良件数

③二項確率紙または次の近似式により母不良率の信頼限界の上限値と下限値を求める。
（近似式は不良5件以上の時に適用）

$$\text{上限値}: P_u = p + u(\alpha)\sqrt{\frac{p(1-p)}{n}} \quad \cdots\cdots (2)$$

$$\text{上限値}: P_L = p - u(\alpha)\sqrt{\frac{p(1-p)}{n}} \quad \cdots\cdots (3)$$

（注）危険率：αは通常5%としている。

7.8.3 ソフトウェア信頼度成長モデル：SRGM(Software Reliability Gross Model)

ソフトウェアは、人間の作った論理を表現した知的生産物であるため、その開発中に多くのソフトウェアエラーが潜入することは避けられず、でき上がったソフトウェアの信頼性が重要
な問題となる。ソフトウェアの信頼度は、ソフトウェアあるいは構成モジュールが、規定の環境下で意図する期間中にソフトウェア故障が発生することなく動作する確率である。

ソフトウェア開発のテスト工程では、大量のテスト資源を費やして不良の発見と修正が行われるので、ソフトウェア内に潜在する不良数は、テスト時間の経過とともに減少していき、ソフトウェアの信頼度は増加していく。従って、テスト時間とソフトウェアの信頼度との間には因果関係がある。これをある関数形にあてはめ、ソフトウェアの信頼度の挙動をモデル化し、理論的に予測、評価するのがソフトウェア信頼度成長モデルによる信頼性評価の手法である。これまで多くの信頼度成長モデルが提案されてきたが、それらを体系的に整理すると
以下の(1)～(4)のように分類することができる。

(1)時間計測モデル

ソフトウェア故障発生時間あるいはエラー発見時間に基づく確率･統計モデル。

(2)個数計測モデル

発生したソフトウェア故障数、あるいは発見されたエラー数に基づく確率･統計モデル。

代表的なものに、NHPPモデルや超幾何分布モデルがある。

(3)アベイラビリティモデル

ソフトウェアの時間的挙動を、ソフトウェア故障の発生しない動作状態と、ソフトウェア故障の発生した不動作状態により記述する確率モデル。

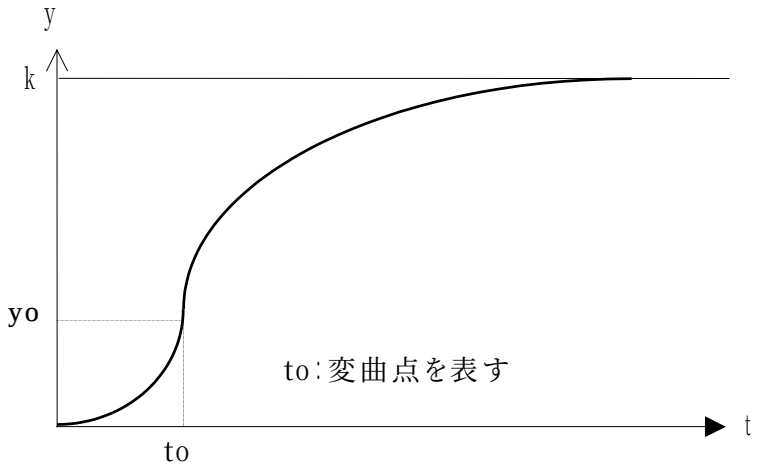
(4)傾向曲線モデル

累積バグ発生数がS字形成長曲線に類似しているという事実を利用し、バグの総出現数や、デバッグの完了時期等を推定する。ゴンベルツ曲線モデルや、ロジスティック曲線モデルがある。

代表的なソフトウエア信頼度モデル①ゴンベルツ曲線モデル②遅延S字信頼度成長モデル③超幾何分布モデルを以下に示す。また、モデルの具体例を図 7．8．6 に示す。

①ゴンベルツ曲線モデル

ゴンベルツ曲線モデルはゴンベルツという人が提案したモデルで、本来は人体の老化作用に対する抵抗力を表したモデルである。ソフトウェア不良の発生傾向が当ゴンベルツ曲線モデルと類似していること、また当初は他に適切なモデルが存在しなかったことから、日立では約15年前から適用されている。単純な成長曲線を次に示す。



モデル式は以下のような式に表せる

$$y = k \cdot a^{(b^t)} \qquad \cdots\cdots (4)$$

$$to = \frac{-\ln(-\ln(a))}{\ln(b)} \qquad \cdots\cdots (5)$$

$$yo = \frac{e}{k} \qquad \cdots\cdots (6)$$

k：予測総バグ件数　　a，b：パラメータ

②遅延 S 字形信頼度成長モデル

遅延 S 字形ソフトウェア信頼度成長モデルは鳥取大学の山田教授他が提案したモデルである。このモデルは、単位時間当りに発見されるエラー数が任意のテスト時刻においてソフトウエア内に残存するエラー数に比例すると仮定の下、テストにおけるエラーの除去過程を、ソフトウェア故障の現象を観測する過程すなわちソフトウェア故障発見過程と、その原因解析を行ってエラーの発見に至るまでのエラー認知過程を考慮した、NHPモデルである。

モデル式は以下のような式に表せる。

$$H(t) = m(t) = a\{1 - (1 + bt)e^{-bt}\} \quad \cdots\cdots\cdots (7)$$

$$(a > 0,\ b > 0)$$

H(t):時間区間(0, T)において発見される総期待ソフトウエアバグ数

a:テスト開始前にソフトウェア内に潜在する総期待バグ数

b:バグ発見率、バグ認知率

③超幾何分布モデル

超幾何分布モデルは東京電機大学の当麻教授他が提案したモデルで、あるテスト時刻にエラー反応するフォールトは、初期フォールト集合中からランダムに選ばれるとして超幾何分布理論に基づいたモデルである。初期不良件数をm、第(i－1)回目のテストまでに発見／除去された累積不良件数をC(i－1)とすると、第i回目のテストに新たに発見される不良件数N(i)の期待値は以下のような式に表せる。

$$\frac{N(i)}{\quad} = \frac{C(i-1)}{m}\}u(i)\cdot(ai+b) \cdots\cdots\cdots (8)$$

(ai＋b):テスト工程の進行に伴う検出不良件数の変化を考慮するパラメータ

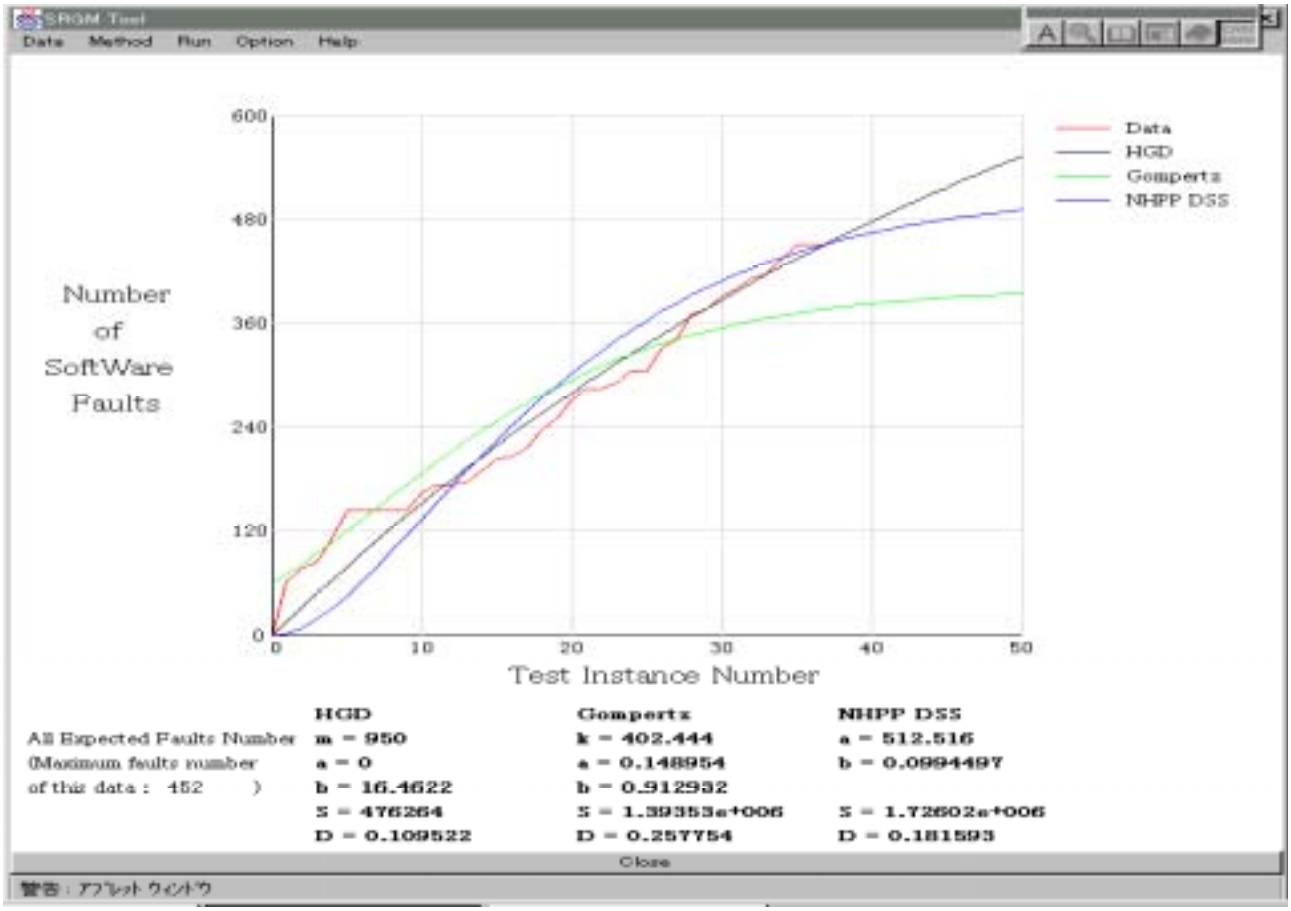
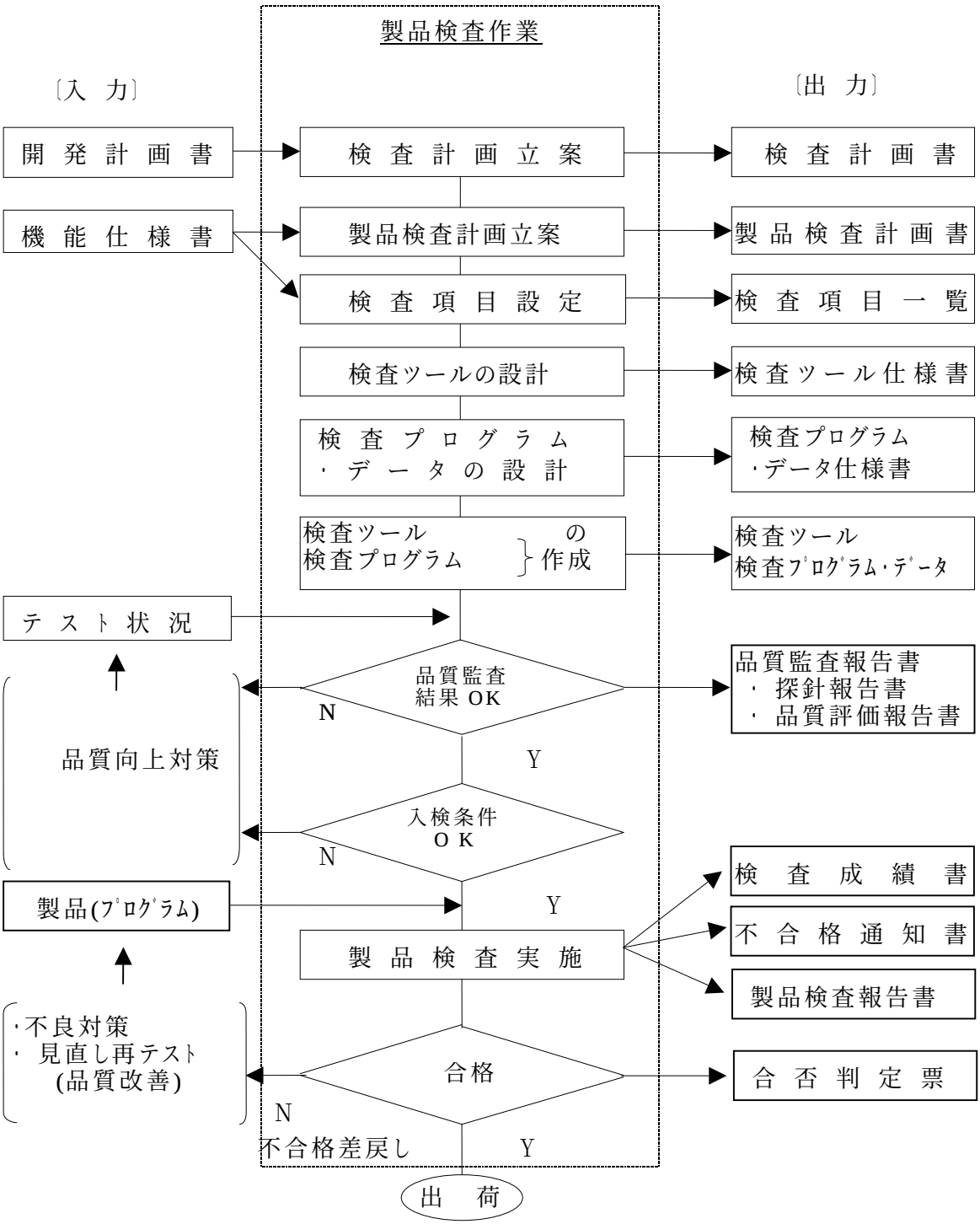


図7.8.5　信頼性成長モデルの具体例

7.8.4　製品検査

製品検査は、プログラムの出荷に先立ち製品としての合否を判定するもので、基本的には顧客の立場に立ち、主として外部仕様に基づいて実施している。図 7.8.6 に製品検査の手順を示す。製品検査を開始するに当たっては、プログラムが十分な品質に達していることを見極めることが重要であり、先に述べた品質監査とは、密接な関係がある。製品検査は、検査が独自に計算機を使用し、実際にプログラムを動作させて、製品としての合否を判定する。不良が発見されたときは、不合格差戻しとなる。設計で品質向上対策（品質改善）が取られたあと、再度製品検査を行う。製品検査準備作業としては、検査項目の設定、検査ツール・プログラム・データの作成などがある。このなかでもっとも重要な作業は検査項目の設定（7.8.5章参照）であり、製品検査の成否は検査項目の出来の良し悪しに左右される。設計部門が内部仕様に基づいた品質チェックを行うのに対して、検査部門ではあくまでも外部仕様からみたチェックを行う。



資料:保田勝通「ソフトウェア品質保証の考え方と実際」

図7.8.6　製品検査の手順

7.8.5 検査項目の設定

検査項目は製品の合否の判定をするための基準となるのもである。検査項目の質が検査の成否を決定する。検査項目は主に外部仕様に基づいてブラックボックス法で設定することが基本であるが、特異な処理や複雑な処理については、内部仕様も参考にしてホワイトボックス法で設定している。検査項目のベースとなるのは、機能仕様書、マニュアルさらに構造設計書等であり、設定技法としては、これ以外に、FME(C)A、FTAあるいは内部仕様に基づくパス抽出法等が用いられている。しかし、検査項目の設定は、各種の技法を用いるだけでは不十分であり、検査ノウハウの蓄積・活用が不可欠である。

主なノウハウは以下の通りである(表 7.8.2 参照)。

(1) 検査項目設定ポイント集

プログラム種別ごとに、検査項目として設定すべき重要な機能や処理条件およびその理由を過去の実績をもとに蓄積したもの。

(2) 不良事例IR(Information Retrieve)

過去の不良事例をデータベース化しておき、プログラム種別、機能、処理方式等のキーワードで検索できるようにしたもの。

(3) 重要障害防止チェックリスト

過去に発生した重要障害について、その障害が如何なる理由で発生し、何が誤りで、何をなすべきであったかを蓄積したもの。

表 7.8.2 検査項目の作成方法

検査項目設定の方針
1. 外部機能仕様に基づくブラックボックス法
2. 特異／複雑な処理は内部仕様に基づくホワイトボックス
検査項目設定技法
1. 系統的テスト項目設定技法
2. FME(C)A／FTA
3. パス抽出法
検査項目設定ノウハウ
1. 検査項目設定ポイント集
2. 不良事例IR
3. 重要障害防止チェックリスト

また、検査項目設定にあたっては量的、質的の両面から基準を設けている。

(4)　量的基準

検査項目の総数を規定している。一般には1000ライン(KLOC)当りの数で表わし、
通常、制御系は30～50件／KLOC、業務系は20／～25件　／KLOCを目安としている。なお、FP ベースの目標値については、**表7.4.2**に示した。

(5)　質的基準

検査項目の内訳を規定している。検査項目を、①基本／正常②異常／障害③限界／境界④周囲条件／インタフェースに分類しそれぞれの総数に占める比率を規定している。通常は、①が50～60%、②が10～20%、③が5～10%、④が20%前後を目安としている。

7.9　システムテスト

システムテストは、当該ソフトウェアが製品検査で合格になった後、顧客に納入する
以前に実施される。システムテストは、実際の顧客システムに近いシステムをSST(システムテスト専用設備)に構築し、ハードウェアおよびソフトウェアの各種製品を組合せた状態で、システムが要求仕様どおりの機能を実現し、所定の性能を発揮するかを確認するとともにシステム全体の信頼性を確認するものである。大規模オンラインシステムを例にとった、システムテストの概念を図7.8.7に示す。
システムテストを真に効果的に実施できるかどうかは、実稼働の環境作成いかんにかかっている。すなわち、

①システムの目的とする実業務または摸擬業務の実現

②実稼働で予想される障害、あるいは異常操作などの実現

である。このため、ハードウェアやソフトウェアの各種支援ツールをSSTに設置するのはもちろんのこと、必要に応じて、顧客設備機器、実業務プログラム、さらには摸擬ファイルやデータをも借用して顧客の実稼働に近い環境をシミュレートしている。

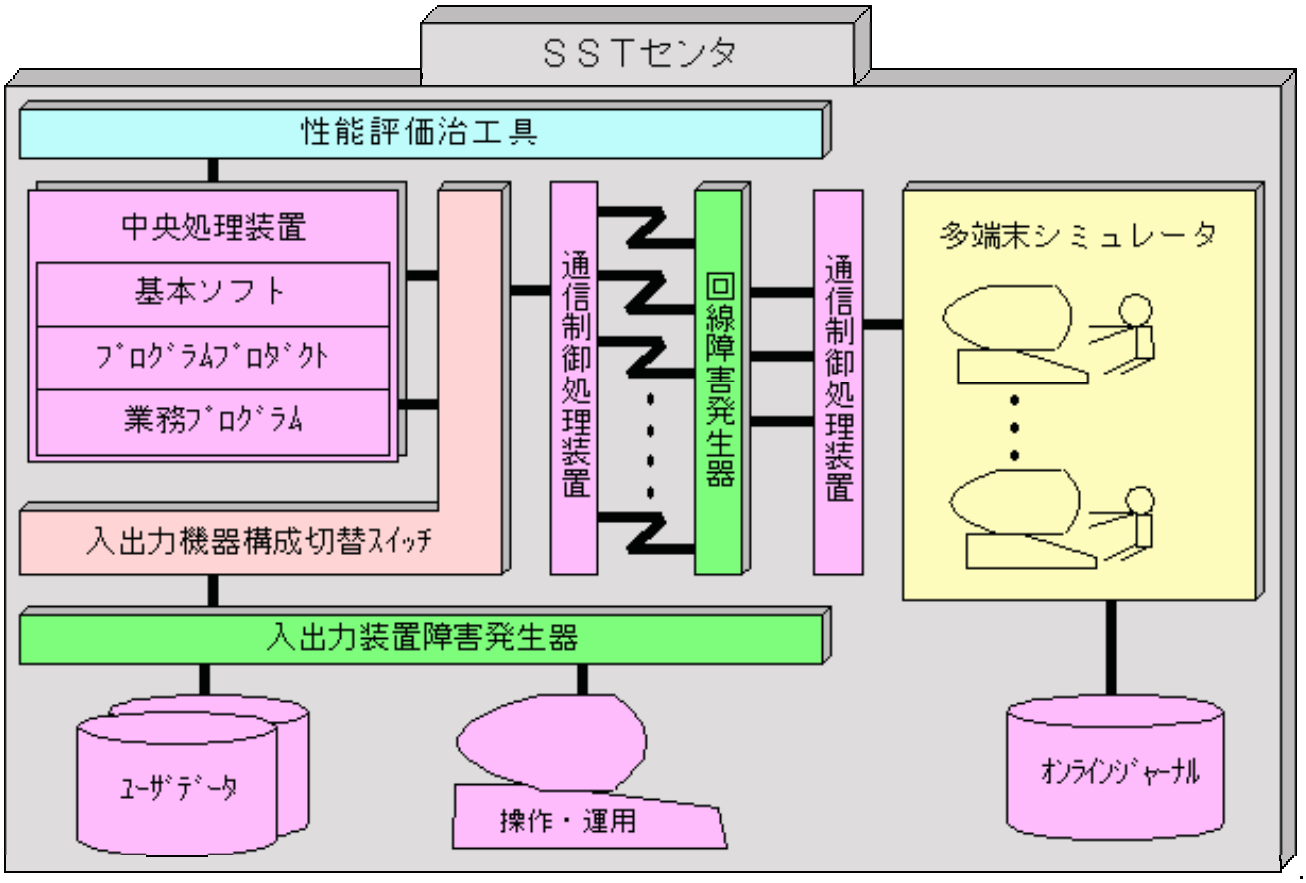


図 7.8.7　システムテストの概念

[参考文献]

保田勝通著:ソフトウェア品質保証の考え方と実際

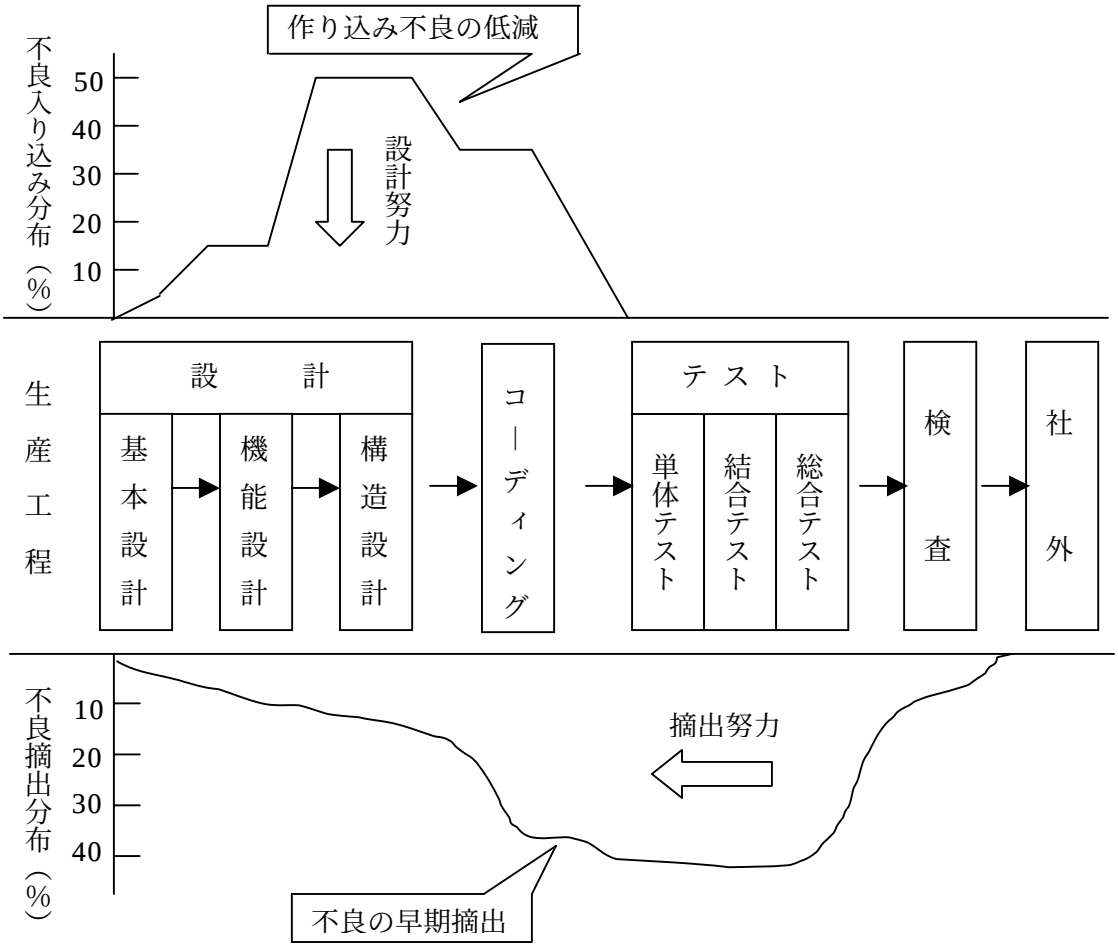


図７．１　ソフトウエア開発工程と不良分析

テストは一般的に、「システムで実現すべき機能が完全に実現されている事を明示的に証明する作業」と定義されるが、ソフトウエアを中心とするシステム開発においては狭義と広義の両面から考える必要が有る。狭義には、ソフトウエア仕様書に記述（定義）された機能が実現されている事の確認であり、広義にはユーザ要求への適合度、すなわちシステム完成度の確認である。これの関係を**図 7．2．2** に示す。

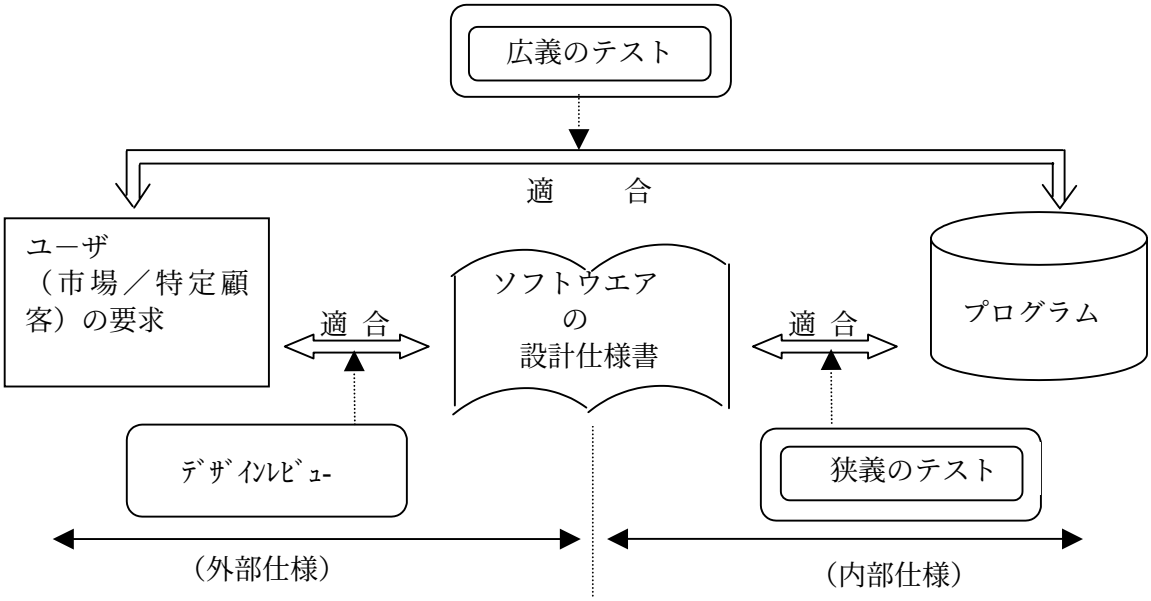
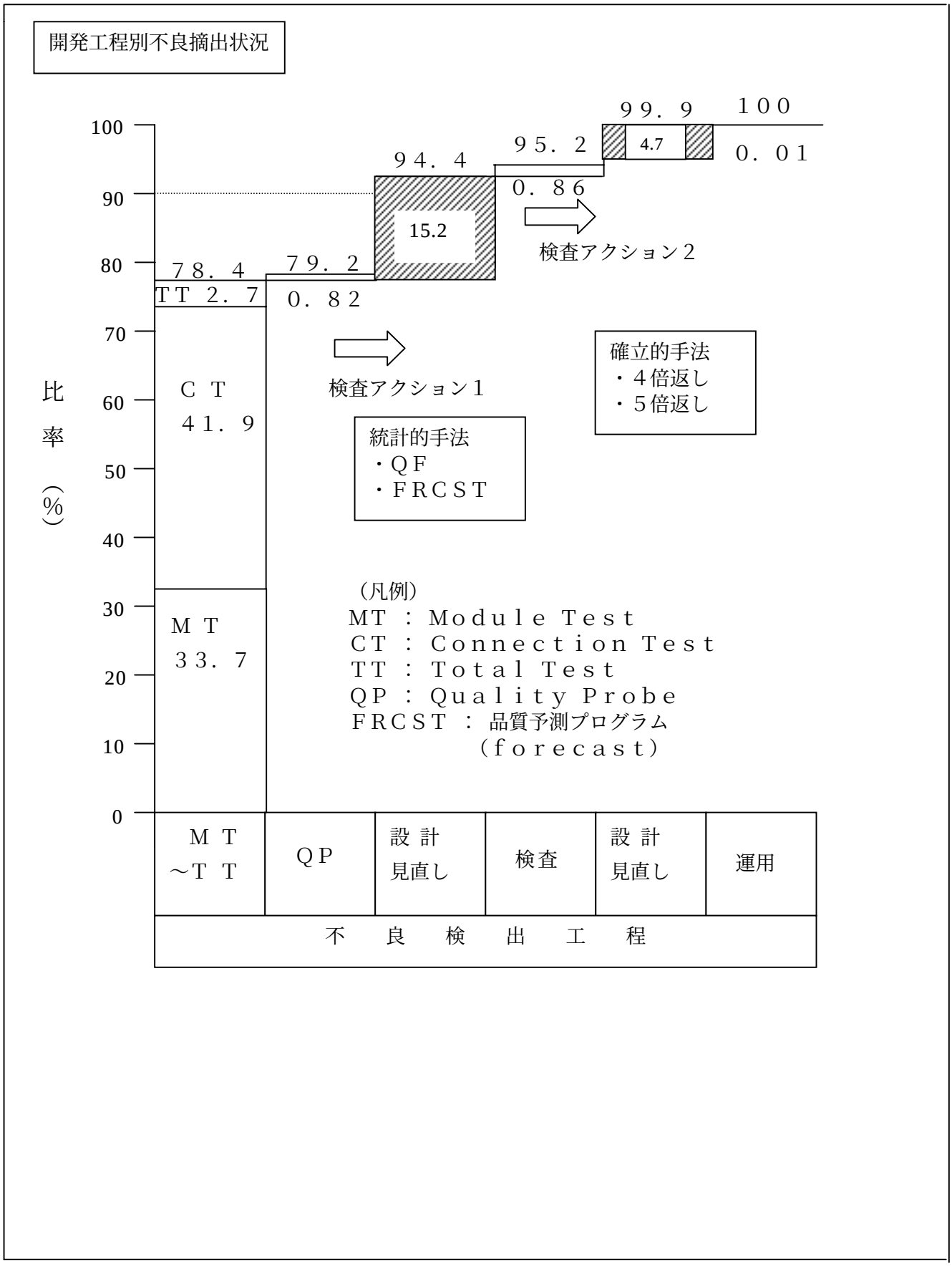
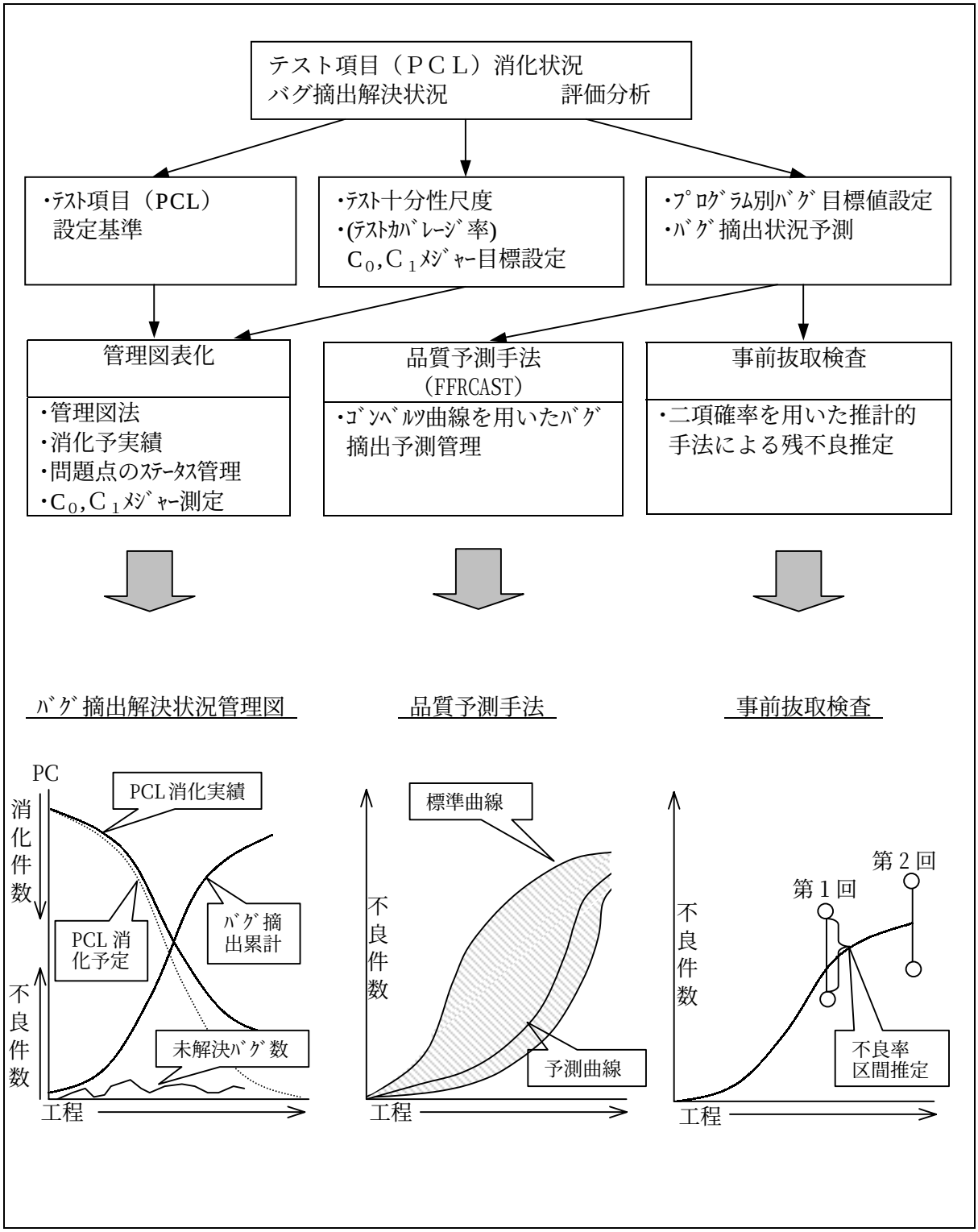


図 7．2．2　　狭義のテストと広義のテストの関連

６．３テストと品質管理（２／２）

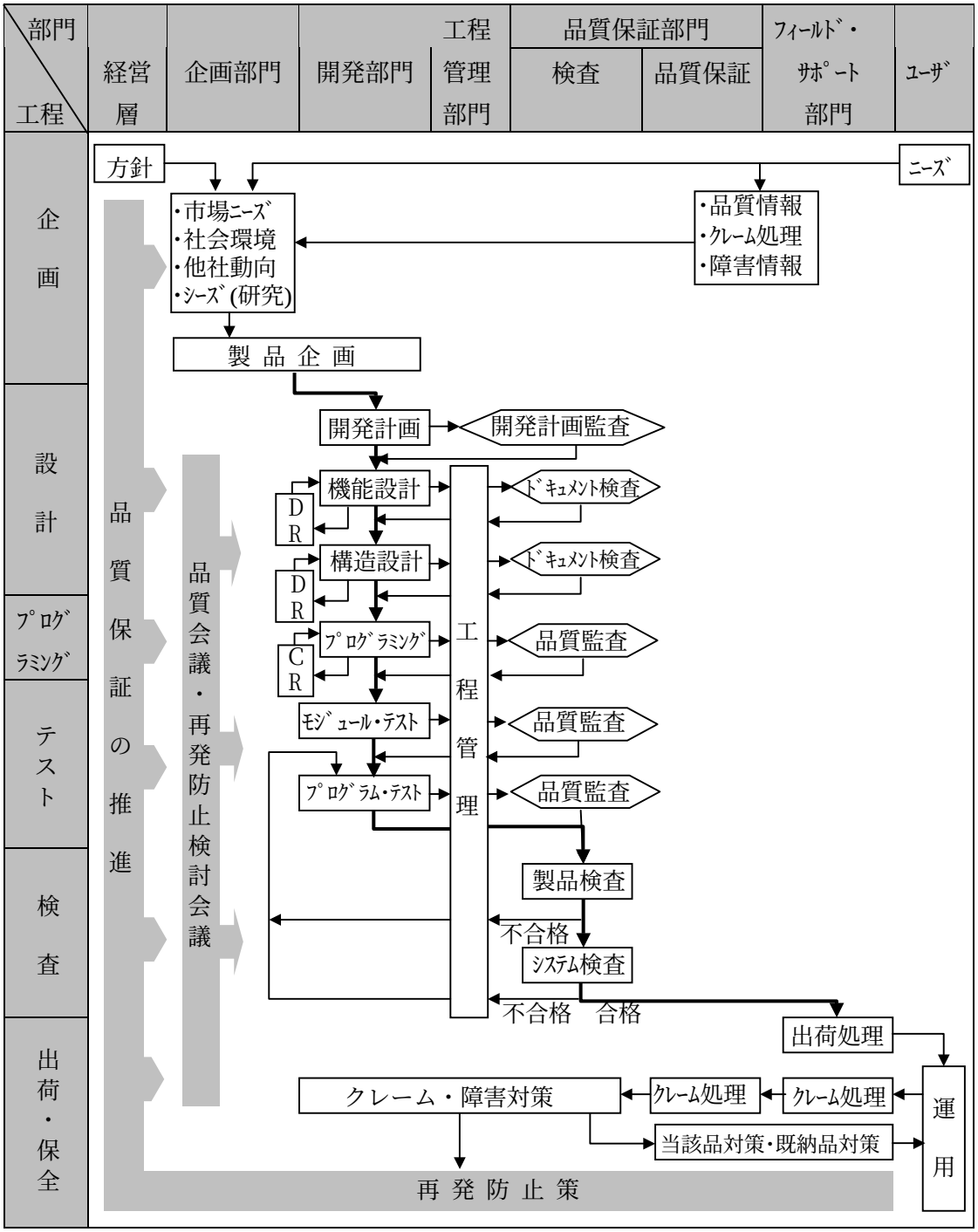


7．4 品質管理技法（1／3）

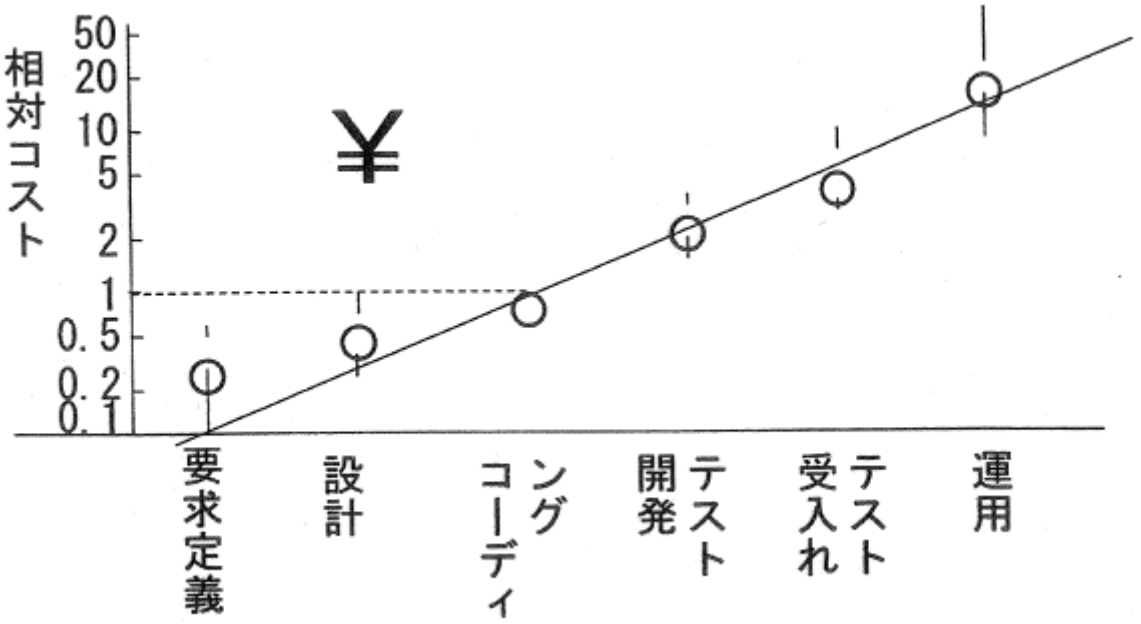

テスト項目（ＰＣＬ）消化状況
バグ摘出解決状況　　評価分析
・ﾃｽﾄ項目（PCL）
設定基準
・ﾃｽﾄ十分性尺度
・(ﾃｽﾄｶﾊﾞﾚｰｼﾞ率)
C0,C1ﾒｼﾞｬｰ目標設定
・ﾌﾟﾛｸﾞﾗﾑ別ﾊﾞｸﾞ目標値設定
・ﾊﾞｸﾞ摘出状況予測
管理図表化
・管理図法
・消化予実績
・問題点のｽﾃｰﾀｽ管理
・C0,C1ﾒｼﾞｬｰ測定
品質予測手法
(FFRCAST)
・ｺﾞﾝﾍﾞﾙﾂ曲線を用いたﾊﾞｸﾞ
摘出予測管理
事前抜取検査
・二項確率を用いた推計的
手法による残不良推定
ﾊﾞｸﾞ摘出解決状況管理図
品質予測手法
事前抜取検査
PC
消化件数
PCL消化実績
PCL消化予定
ﾊﾞｸﾞ摘出累計
不良件数
未解決ﾊﾞｸﾞ数
工程
標準曲線
不良件数
予測曲線
工程
不良件数
第1回
第2回
不良率
区間推定
工程

４．１ ソフトウェア品質保証システム（２／２）

（株）日立製作所における事例



不良修正コスト



出典：日本規格協会「ソフトウェア品質管理ガイドブック」

●コーディング工程で作りこんだ不良
　⇒ テストでの発見　　：相対コストは、約2倍
　⇒ 運用テストで発見　：相対コストは、約10～50倍
●一番コストがかかるのは、下流のテスト工程での不良発覚
　（社外事故の場合は、コストはMaxとなる）