SEC BOOKS

ソフトウェアテスト見積りガイドブック
～品質要件に応じた見積りとは～

独立行政法人 **情報処理推進機構**　ソフトウェア・エンジニアリング・センター　編

Ohmsha

SEC BOOKS

ソフトウェアテスト見積りガイドブック
～品質要件に応じた見積りとは～

独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　編

Ohmsha

はじめに

昨今，あらゆる分野においてIT利用が広がっており，情報システム（ソフトウェア）の障害（欠陥）が社会的に大きな影響を及ぼし，深刻化するケースが増えています。そのような状況の中，ソフトウェアの信頼性をいかに高めていくかが喫緊の課題となっています。また，ソフトウェアの信頼性向上に関して，ソフトウェアテストの重要性が大きいことは言うまでもありません。

ソフトウェア・エンジニアリング・センター（以下，SECと略記）では，これまで定量的な見積りに関する以下の書籍を発刊いたしました。

- 『ITユーザとベンダのための定量的見積りの勧め』（2005年４月）
- 『ソフトウェア開発見積りガイドブック』（2006年４月）
- 『ソフトウェア改良開発見積りガイドブック』（2007年10月）

これまでの書籍では，ソフトウェアテストの工数・コストはプロジェクト当初の見積りに包含されている前提として，あまり多くは触れられていませんが，ソフトウェア開発ではソフトウェアの残存欠陥の多寡および要求性能の水準など，品質要件に応じた見積りを行い，ユーザとベンダとで合意する必要があります。

本ガイドブックでは，既発刊の見積りガイドブックを補足し，ソフトウェアの品質検証と妥当性確認に関わるテスト見積り（テスト量，テスト生産性）に焦点をあて，具体的な方法およびノウハウを『ソフトウェアテスト見積りガイドブック』として紹介するものです。

構成は大きく２つに分かれており，第１部が「総論」，第２部が「事例集」としてテスト見積りおよび品質コントロール手法の事例となっております。

表1　想 定 読 者

A）ベンダ企業のプロジェクトマネージャ
B）ベンダ企業の社内改善メンバ(企画メンバ)， 品質保証部門の担当者
C）ユーザ企業の契約担当者 ベンダ企業の営業担当者
D）ユーザ企業のトップマネジメント プロジェクトマネージャ システム部門メンバ 社内改善メンバ(企画メンバ)

第1部では，「テスト見積り」に関する基本的な考え方や心得的な内容を実際の活動につなげ，さらに向上を図るための方法について示しています。

対象読者は，**表1**に示すA)～D)のすべての方を想定しています。ソフトウェア開発プロジェクトのマネジメント・運営を行う観点から示されている事項も多く，ユーザ企業にとっては，業務要件を担保する妥当性確認方法や品質要件の明文化など，自社の仕組みを見直すための参考にしていただきたいと考えます。

第2部では見積り活動の事例集として，ソフトウェアテスト見積りおよび品質保証活動(テスト進行中における品質管理と再テスト見積りなど)に関して，先導的な取り組みを行っている各社の事例，当該方法の導入に当たっての留意点を示します。第2部は，具体的な例として第1部から必要に応じて参照されるとともに，個別の方法に興味のある方が1つひとつの事例を独立して参照できるように構成しています。

本ガイドブックで対象とする範囲は次のとおりです。

- 本ガイドブックのテスト見積りは，ソフトウェアに混入する欠陥の総量を見積り，レビューを含めて各開発工程で欠陥を除去する工数を見積もることを前提にしています。また，テスト進行中における品質の測定およびテストの再見積りに関する事例などを紹介し，ソフトウェアの品質保証としての考え方も示しています。
- ビジネスアプリケーションを中心としたソフトウェア開発を対象としてい

ます。ただし，本ガイドブックで示されたガイドラインは，組込みソフトウェアなど，他の分野でも十分に適用可能です。

- 見積りとは，規模，工数，工期，品質(性能，信頼性など)，コストなどのさまざまな要素を広く対象とするものです。本ガイドブックでは，特にテスト量から算出する工数，工期，コストの見積りに焦点をあて，その具体的な考え方および方法を示します。ここでの品質は，それぞれの関係に影響を及ぼす要因としてとらえています。

本ガイドブックがソフトウェアの品質と価格に関係する事項の明確化およびソフトウェアの品質向上への一翼を担うことを願っております。

なお，本ガイドブックの編纂，執筆にあたって並々ならぬご尽力をいただきました「見積り手法の適用推進WG」の各氏につきましては，以下に御名を記させていただき，この場を持ちまして，深くお礼申し上げさせて頂きます。

育野准治氏，井上智史氏，岩田康夫氏，大島正敬氏，小野直子氏，
高橋茂氏，中村敏夫氏，庭野幸夫氏，幕田行雄氏　（五十音順）

2008年 3 月

見積り手法部会主査　兼　見積り手法の適用推進WG長　太田忠雄

本ガイドブックの読み方

☆ **ベンダ企業のプロジェクトマネージャ**

第１部の第１章～第５章を通読してください。また，第２部の各事例を参照し，テスト見積りの根拠および品質保証の事例として示されている内容を活用してください。

☆ **ベンダ企業の社内改善メンバ(企画メンバ)，品質保証部門の担当者**

第１部の第１章～第５章を通読してください。第１部の第５章を参考にして，組織的なテスト見積り活動の成熟度向上に取り組んでください。あわせて，第２部の各事例を参照し，自組織との共通点や差異を把握して，実際のテスト見積り活動および品質保証活動の成熟度向上に役立ててください。

☆ **ユーザ企業の契約担当者，ベンダ企業の営業担当者**

最初に，第１部の第５章5.3節の箇所を読み，続いて，その背景として第１章の最初から通読してください。第２部の各事例を参照し，テスト見積りの根拠および品質保証の事例として示されている考え方を把握してください。

☆ **ユーザ企業のトップマネジメント，プロジェクトマネージャ，システム部門のメンバ，社内改善メンバ(企画メンバ)**

第１部の第１章～第４章を読み，プロジェクトにおけるテスト見積りの基本事項を理解してください。そして，第１部の第５章を参考にして，自組織でのテスト見積り向上方法を検討してください。また，第２部の各事例を参照し,自組織のテスト見積りおよび品質の向上に役立ててください。

(注) 本ガイドブックの事例紹介は，各社で使用している情報システム用語で記述しています。

目　次

第3章　ソフトウェアテスト見積りでの成功と失敗例

第4章　ソフトウェアテスト見積りの詳細

第 6 章　日本電気(NEC)の事例～ソフトウェア品質会計制度の適用～

第１部
総　　論

第１部では，「テスト見積り」に関する基本的な考え方や心得的な内容を実際の活動につなげ，さらに向上を図るための方法について示しています。

第1章　ソフトウェアテスト見積りの課題

1.1　システムの信頼性向上からみたソフトウェアテスト見積りの課題

近年，急速に進歩発展するIT(情報技術)は，企業でのIT活用のみならず，個人の生活に深く利用されてきています。さらに，業務の複雑化，商品の高機能化およびシステムの利用範囲の拡大に伴い，ソフトウェアの開発規模が増大してきています。その一方で，ソフトウェアの品質が社会的な問題になってきている反面，ソフトウェア開発コストの削減に基づくソフトウェアテストの効率化および開発期間の短縮が要求されてきています。

そのような状況下，ソフトウェア開発の見積りにおいて，裏づけに乏しいテストコストの削減や期間の短縮が見受けられ，その結果，本番稼働後においてソフトウェア欠陥による障害事例として現れています。本番稼働後に残存する欠陥によるリスク(障害などによる損失)に見合ったテスト戦略(1)を策定し，テスト戦略を裏づけとしたソフトウェアテストの見積りが望まれます。

なお，本ガイドブックでは「ソフトウェアテストにおける適正資源の確保」を目指して，ソフトウェアテスト見積りを取り上げますが，取り上げるソフトウェアテスト見積りのスコープは次のとおりです。機能要件(非機能要件含む)から見積もるソフトウェア開発規模および設計・実装工程(基本設計～コーディング)での作り込み品質予測(残存欠陥)に基づいて，品質要件から品質目標を設定して，ソフトウェアの品質を確認するためのテスト量(2)およびテストの生産性を見積ります。さらにテスト量およびテストの生産性に基づきテスト工数(コスト)を見積もることをソフトウェアテスト見積りの対象としています。

---

(1)　ここでのテスト戦略は，「ソフトウェアの重要な部分を特定したうえで，品質をどこまで確保し，そのためにテストにどれだけの資源を投入するか，また投入する資源をどう使うか」といった方針について示しています。

(2)　本ガイドブックでは，ソフトウェア開発規模とは別に，開発するソフトウェアの品質を担保するために実施するテスト作業の量をテスト量と呼びます。例えばテストケース数，テスト項目数などが該当し，テスト仕様書，テスト手順書およびテストデータの量もテスト量に含まれます。

1.1.1　レビューおよびテストを鑑みた残存欠陥の予測

ソフトウェアは，設計・実装工程から品質を作り込んでいれば，テスト工程で検出される欠陥は少なくなり，結果，テストコストも削減できることが経験的にわかっています。しかし，現実には開発期間，コスト，体制およびテスト環境などの制約と条件面から設計・実装工程で欠陥を検出するレビューなどのコストには限界があり，テストコストとのバランスを考慮したテスト戦略が必要になります。

ソフトウェアテスト見積りでは設計・実装工程において混入する欠陥の量およびレビューなどにより除去する欠陥の量，ならびにテスト工程での品質確認テストにより除去する欠陥の量に基づいて，リリース時の残存欠陥の総量を予測することが必要になってきます。

しかし，残存欠陥はテスト終了時において測ることができません。実際の開発では，残存欠陥を経験に基づいて推定しており，新たな業務などの開発領域が異なると推定精度が低下し，残存欠陥を見誤り，本番稼働後にトラブルが発生しているケースが見受けられます。

そこで本番稼働後の障害を許容範囲に抑えるために，代替尺度(テスト網羅率など)に基づく残存欠陥の定量的推定が期待されており，ソフトウェアテスト見積りでは，コストのほかに，テスト網羅率などの代替指標をテストのゴールとして見積ることが望まれます。その結果，ソフトウェアテストの見積りによって，適正な資源が導出されることになります。

1.1.2　テスト量の論理的把握と品質の完全性保証

(1)　テスト量と品質要求とのトレードオフ

ソフトウェア開発規模の増加とともに，ソフトウェア機能および実行タイミングは複雑化してきており，動作環境も考慮して，あらゆる条件を組み合わせたテストを行うと，テスト量(テストケース数など)は，天文学的な数字になります。しかし，現実的にはコストおよび開発期間などの制約により，テスト量を合理的に削減することが必要になります。つまり，テスト量を削減してテストコストを抑えることと，要求される信頼性および安全性の水準をソフトウェアが達成しなかったときのリスクに関して，許容範囲内に抑えるというトレードオフを考慮したテスト見積りが必要になってきます。

（２） テスト見積りとテスト完了基準

ソフトウェアのテスト完了基準は，テストカバレッジの度合い，信頼度成長曲線の収束度合い，仕様変更の収束率，修正未了の欠陥数，未解決・懸案件数および定性的な検出欠陥の発生傾向など，総合的な判断が必要になります。特に，テスト見積りでは，テスト完了基準との関係において，守るべきテスト網羅率などを設定し，ユーザとベンダとで相互に合意する必要があります。

1.1.3　改良開発および仕様変更におけるテスト量の把握

（１） ソフトウェア改良開発におけるテスト量の把握

ソフトウェア改良開発におけるテスト見積りに関しては，既に，SEC BOOKS『ソフトウェア改良開発見積りガイドブック』で紹介しています。特に，新規開発に比べて，既存の母体システムにおける品質面での考慮およびテスト巻き込み規模[3]の把握など，改良開発の特質を考慮したテスト量の見積りが必要になります。

（２） 仕様変更におけるテスト量の把握

仕様変更は設計・実装工程からテスト工程まで，プロジェクトの品質，コストおよび期間に影響します。また，同じ機能の仕様変更を設計・実装工程，テスト工程と後工程で対応すればするほど，コスト高(棄却量の増加などによる)となり，さらに仕様変更を加えることにより，ソフトウェア品質の劣化機会(デグレード)[4]が増加します。テストの終盤であるシステムテスト工程の段階では，ソフトウェアの機能がユーザにも把握できるようになり，仕様変更が多くなる傾向にあります。その結果，見積りを超えたコストの増加と品質問題を発生させる可能性があります。よって，仕様変更タイミングなどを考慮したテスト量を把握する必要があります。

1.1.4　非機能要件の把握と確認

情報システムの信頼性向上に関するガイドライン[5]では，企画段階における留意事項として，次の２項目が求められています。

（３） ソフトウェアを変更した際に影響を受ける部分を表す規模のことを言います。
（４） ここで，品質の劣化機会(デグレード)とは，修正ミス(修正漏れ，修正誤り)および，修正した際に，影響範囲を見誤ってそれまで正常に稼働していた部分に影響を与え，動作不正(障害)を来たすことを言います。

① 情報システムが具備すべき信頼性・安全性の水準について定め，システム利用者と供給者間で合意すること。

② 発注仕様への機能要件および非機能要件の取り込みを行うこと。

非機能要件については，特に，品質要件(例：JIS X 0129「ISO/IEC 9126」)を基準にした定量的な把握と確認方法が重要です。ソフトウェアテスト見積りでは，ソフトウェア品質を担保するうえで必要となる非機能要件の確認網羅性を，どのように取得したらよいかが課題となります。

1.2　ソフトウェア品質と価格の関係に関する認識の統一

システム(業務アプリケーションソフトウェア)には航空，鉄道，銀行および医療など，高信頼性を要求されるシステムと民間の社内システムなどのように通常の信頼性を要求されるシステムなど，必ずしも要求されるソフトウェア品質は一様ではありません。

また，1.1節に示したようなテスト見積りに関わる課題の対策など，ソフトウェア品質を高める手段を明確にして，ソフトウェア品質と価格との関係を整理し，品質を高めるための適正コストを確保する枠組みを，ユーザとベンダとで合意することが必要になります。

特に，ソフトウェアテストは，品質を保証するための最後の砦であり，品質を高めるためにソフトウェアテストに適正な資源を確保することが求められます。

---

(5)　平成18年6月，経済産業省は，情報システムの障害発生が社会的影響を及ぼし，日々深刻化している状況を受け，信頼性を高めるための指針として，「情報システムの信頼性向上に関するガイドライン」を公表しました。

第 2 章　開発プロセスモデルとテスト手法

2.1　品質保証から見た開発プロセスモデルとテスト手法

開発プロセスモデルおよびテスト手法(本ガイドブックではテスト方法とテスト技法を指し，2.3節で説明します)の選定は，テスト計画の重要な決定事項の１つです。

テスト計画は，主にソフトウェア開発過程における欠陥の作り込みリスクおよび残存欠陥の目標に基づいて策定され，さらに，実施するソフトウェアテストの量(例えば，テストケース数，テスト項目数，テストデータ量など)は，開発プロセスモデルおよびテスト手法により変化します。

残存欠陥はソフトウェア開発の過程において予測困難であり，テスト網羅率などの代替指標をテストのゴールとして見積もることが必要になります。

また，要件定義の段階からテスト計画を策定し，選定された開発プロセスモデルおよびテスト手法に基づいたソフトウェアテストの見積りを行うことが重要となります。

2.2　テストプロセスとソフトウェア開発プロセスモデル

ソフトウェア開発プロセスに関しては，ウォーターフォール型開発，インクリメンタル手法(アジャイルなど)およびモデル駆動型開発など，多くのソフトウェア開発プロセスモデルが提唱されています。

本ガイドブックでは，ウォーターフォール型開発モデルなどの開発ライフサイクルモデルを，品質保証の観点から見たモデルについて紹介します。

(1)　V字型モデル

V字型モデルは，品質保証の観点から見たモデルとして30年以上の間，広く利用されてきました。設計・実装工程で作り込まれたソフトウェアの欠陥は，当該設計・実装工程のレビューおよび相対するテスト工程で検出するモデルです。

その概念を**図 2.1**に示します。

テストプロセスはプログラムの単体テストから，プログラムを結合した統合テ

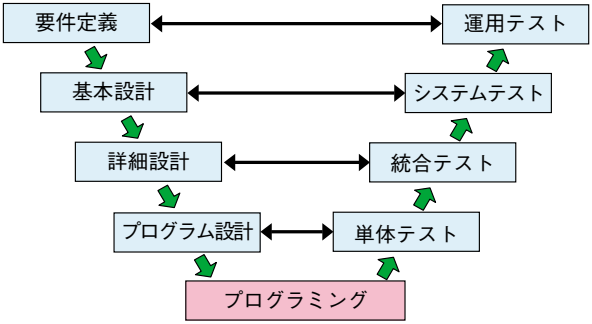


図 2.1　V字型モデル

スト，さらに業務要件を確認する運用テストなどへと順序立てて行われ，品質確認がなされます。

各テスト工程ではテスト設計，テスト実施，テスト結果検証，欠陥修正(修正作業を開発工程としてとらえる場合もあります)および再テスト実施が行われます。

（2） W字型モデル

近年，V字型モデルを進化させたダブルV字型(＝W字型)モデルが利用されるようになってきています。その概念を**図 2.2**に示します。

W字型モデルはV字型モデルと同じように，設計・実装工程で作り込まれたソ

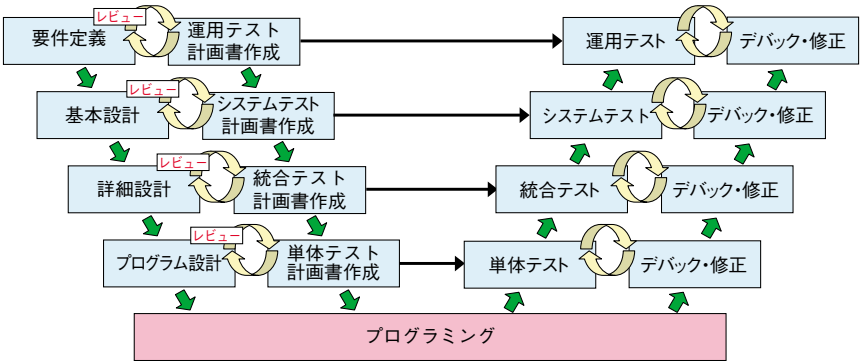


図 2.2　W字型モデル

フトウェアの欠陥に対して当該設計・実装工程のレビューで検出し，さらに相対するテスト工程のテスト設計を開発工程の水準にあわせて同時並行して行うところが特徴です。

各テスト工程ではテストの実施，テスト結果検証，欠陥修正(修正作業を開発工程としてとらえる場合もあります)および再テストの実施が行われます。

W字型モデルは，早い段階で欠陥が発見されること，および開発とテスト設計などの並行作業による効率面の向上などメリットがありますが，仕様変更が発生した場合にテスト仕様などを見直す範囲が拡大する可能性があります。また，設計・実装工程とテスト工程との作業負荷のバランスを配慮したテスト計画(テスト中におけるアクシデント対応への工数・期間の配慮など)が必要になります。

コラム：U字型モデル

U字型モデルは，日本情報システム・ユーザー協会(JUAS)から提唱されたモデルです。その概念を**図 2.3**に示します。

基本的にはV字型モデルと同じように，設計・実装工程で作り込まれたソフトウェアの欠陥は，当該設計・実装工程のレビューおよび相対するテスト工程で欠陥を検出するモデルですが，以下のような点を重要視したモデルです。当該開発工程のレビューの徹底による設計・実装工程での欠陥作り込み削減および単体テスト時点での業務要件の妥当性確認など,欠陥の早期検出,テストの重複性の削減およびシステムテスト工程など，最終工程の負荷削減を狙ったモデルです。

U字型モデルは，ユーザ側とベンダ側の作業および成果物が詳細にわたり標準化され，分担などが明確にされていることが前提になります。

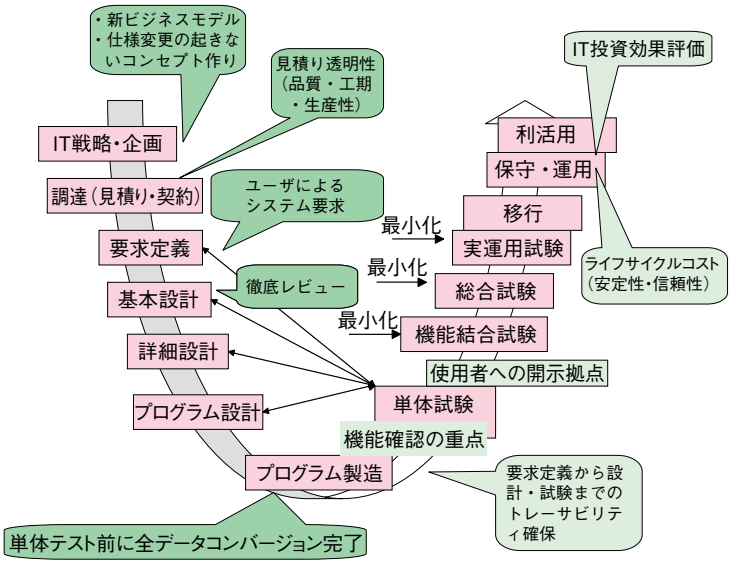


図 2.3　U字型モデル

2.3　テスト手法の一般的事項

2.3.1　ソフトウェアテスト方法

テストは，大きくブラックボックステストとホワイトボックステストの２つのテスト方法に分類されます。それぞれの特徴を**表 2.1**にまとめています。

ブラックボックステストはソフトウェアの外部要件に基づいたテストであり，ホワイトボックステストはソフトウェアの内部構造および内部仕様に基づいたテストです。いずれの方法にも一長一短があり，ユーザの立場でのブラックボックステストと開発者の立場でのホワイトボックステストとを，うまく使い分ける必要があります。

なお，最近では両者の中間的な方法としてグレーボックステスト(6)という用語が使われることもあります。

表 2.1　テスト方法の分類

テスト方法	説　　明
ブラックボックステスト	要求仕様や外部仕様を基にしてテストを設計する方法で，ユーザ視点で，外部からの入力に対するアウトプットを検証するテストである。テスト設計が比較的容易である一方で，組み合わせのテストはテスト量(テストケース数など)が爆発的に増加してしまう問題がある。
ホワイトボックステスト	テスト対象ソフトウェアの内部パス，構造，設計情報に基づいてテスト設計する方法で，テスト対象が小さい場合は効果が高い一方，テスト対象が大きい場合は，情報が多すぎて扱いきれない問題がある。

2.3.2　ソフトウェアテスト技法

テスト技法については，ほかの書籍などでも数多く紹介されています。ここでは，ブラックボックステストおよびホワイトボックステストの代表的なテスト技法について，**表 2.2**および**表 2.3**にまとめています。テスト項目の抽出に当たって

（6）　テスト対象ソフトウェアの内部パス，構造，設計情報を知ったうえで，ブラックボックステストのように，機能仕様から見た動作をテストする方法です。設計やコードがどうなっているかを知ることによって，テスト量(テストケース数など)を削減して，より少ないテスト量でバグが出そうな箇所のテストを行うというものです。

は，各種テスト技法を使用する際，いかに合理的にテストを省略するかが重要となります。詳しくは4.3.3項に合理的なテスト量の削減方法について記載します。

表 2.2　代表的なブラックボックステスト技法

テスト技法	説　　明
同値クラステスト	同値クラス(同様に取り扱われて同じ結果を生む値の集合)に対してテストケースを1つ作成する。
境界値テスト	同値クラスの各境界について，境界上の値，境界のすぐ下の値，境界のすぐ上の値を1点ずつ選んでテストケースを作成する。
異常値テスト/無効値テスト	入力範囲外のテストケースを作成する。
ペア構成テスト(オールペア法)	2因子間の組み合わせをすべて網羅するテストケースを，テスト最小化アルゴリズム(オールペア法)を用いて作成する。
ペア構成テスト(直交表)	直交表という特殊マトリクスを利用し,2因子間の組み合わせをすべて網羅する最小のテストケースを作成する。
状態遷移テスト	次々に変化していく状態遷移のすべての状態を1回経由するようテストケースを作成する。
ドメイン分析テスト	相互作用のある複数の変数を境界内，境界外，境界上の値で同時にテストする。
デシジョンテーブルテスト	可能なすべての条件(入力)と可能なすべての処理(出力)をリストアップした表に基づきテストケースを作成する。
ユースケーステスト	テストシナリオ(利用者がある目的を達成するためにシステムをどう使うか)に基づいてテストケースを作成する。

表 2.3　代表的なホワイトボックステスト技法

テスト技法	説　　明
制御フローテスト/ 制御パステスト	モジュール内の実行パスを識別して，それらのパスを網羅するようにテストケースを作成する。カバレッジ(テスト可能なコードに対して実際にテストされたコードの割合)のレベルが8段階で定義されている。
データフローテスト	モジュールのすべての変数について，すべての定義ー使用のペアを少なくとも1回は網羅するテストケースを作成する。

2.4　ソフトウェアテストの見積りの範囲

次に本ガイドブックにおけるテスト見積りの対象範囲について説明します。

ソフトウェアテストにはさまざまなレベルのテストがあり，その位置づけはユーザ企業やベンダ企業でさまざまです。本ガイドブックで用いるそれぞれのテスト工程の内容を示すことを目的として，**図 2.4**に本ガイドブックで用いるテスト工程の呼び名と共通フレーム2007に定めているテストに関するアクティビティ名とを対比しています。共通フレーム2007と合わせて見ることをお勧めします。ある業務(7)を１つひとつ取り出し，その要素を見ると，大きく分けて人による活動とシステム(8)からなっています。エンタープライズ系のシステム開発では，同一のハードウェアシステム上に，複数の業務システムを構築することが多くあり

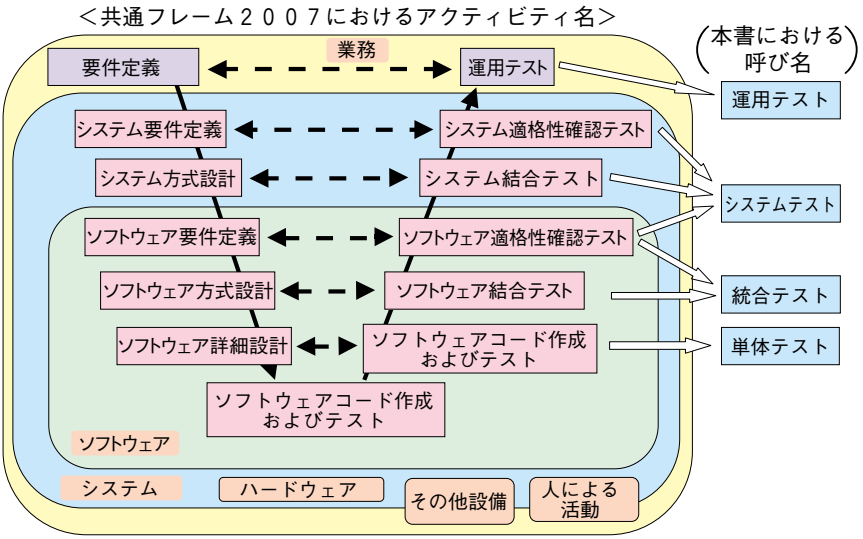


図 2.4　品質保証の観点からの設計とテストとの対応関係

（７）　組織には必ず業務と呼ばれるものがあります。企業においては販売管理の業務，物を作る業務，物流の業務，人事管理の業務などさまざまです。

（８）　１つ以上の手続き，ハードウェア，ソフトウェア，設備および人間を統合化して，規定のニーズまたは目的を満たす能力を提供するものです(JIS X 0160-1996)。

ます。したがって，システム結合テスト，ソフトウェア適格性確認テストおよびシステム適格性確認テストを一体で実施する場合が多いので，本ガイドブックでは，これら3つのテストをシステムテストとして一括りにしています。また，ソフトウェア適格性確認テストは，システムテストおよび統合テストの双方でカバーされます。

本ガイドブックでは，ソフトウェアテストの見積りを，その網羅性に焦点をあてて，ソフトウェアテストの量(テストケース数など)およびソフトウェアテストの生産性とで説明することを主眼としています。

非機能要件の確認テストには，構成テスト，セキュリティテスト，信頼性テスト，負荷テスト，回復テスト，および性能テストなどがあります。これらのテストは目的も方法もまちまちであるため，テスト量と生産性に基づく定量的な見積りまでは言及していません｛本ガイドブックでは，資料「非機能要件の把握・確認とテスト見積りへの影響表」で品質要件(JIS X 0129など)に対するテスト量および生産性への影響のみ示唆しています｝。また，システムの最終利用者も参加し，運用教育なども兼ねて行う運用テストは，テストに投下するリソースおよびテスト期間を前提に計画されることが多いため，本ガイドブックのスコープからは除外しています。

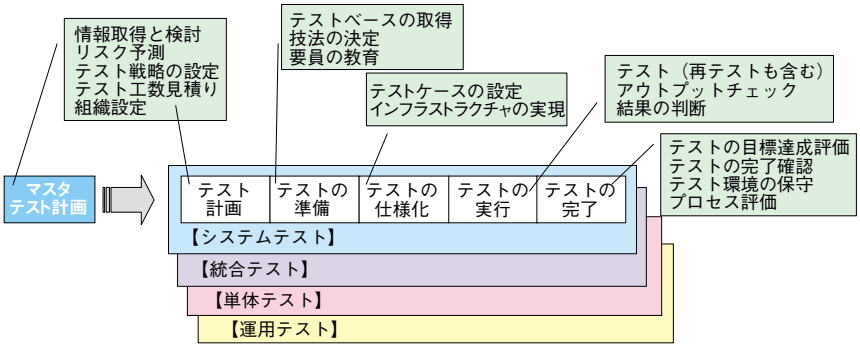
2.5　テストプロセスおよびテストドキュメント

2.5.1　テストプロセス

テストプロセスは，「テスト計画」，「テストの準備」，「テストの仕様化」，「テストの実行」および「テストの完了」のサブプロセスから構成され，ソフトウェア開発の設計・実装工程と並行して開始するライフサイクルモデルによって**図2.5**のように体系化できます。

まず，マスタテスト計画では，単体テストから運用テストのそれぞれのテストに対応する要求と，要求が満たされていない場合のリスクに基づき，それぞれのテストで最も重要な欠陥をできるだけ迅速に効果的に検出できるようにテスト戦略を設定します。テスト戦略の主要な観点は，ソフトウェアの最も重要な部分を定義することで，その目標はソフトウェアのしかるべき部分をテストによって適切にカバーすることです。また，テスト戦略ではこれに加えて，テスト組織の構造化とテストインフラストラクチャの定義の指針を定めます。

マスタテスト計画は，細部は別としてシステム要件定義およびソフトウェア要



（参考文献）　Tim　Koomen，Martin　Pol著，富野　壽監訳，テストプロセス改善－CMM流実務モデル，ISBN 4 -320-09734-3，共立出版，2002

図 2.5　テストのライフサイクルモデル

件定義の段階で作成し，だれが，どのテストを，いつ行うかを規定します。このマスタテスト計画が，単体テストから運用テストに至るまでのすべてのレベルのテストをカバーします。次に，マスタテスト計画に基づいて，**図 2.6**のように単体テスト，統合テスト，システムテストおよび運用テスト用に詳細なテスト計画を作ります。

図 2.6　テスト計画の階層

テストの準備では，テスト対象の設計書など(図 2.5ではテストベースと呼んでいます)を入手してテスト可能性を評価し，サブシステムなどのテストの単位ごとにテスト技法を決定します。テストの準備は，ソフトウェアの設計書の初期バージョンが作成され，適切な品質レベルに達していれば開始できます。

テストの仕様化では，テストケースを設定し，これに必要なインフラストラクチャを実現します。テストの実行は，テスト可能な最初のソフトウェア構成品が得られた時点で開始し，その結果を報告します。

最後に，テストの完了で，テストの目標が達成されているか否かを評価してテストの完了を判定します。また，その後のソフトウェアの改良に備えて，テストケース，テストデータおよびテストツールなどのテスト環境を整理します。

コラム：SEC BOOKS『共通フレーム2007』におけるテストに関するプロセス

テストのライフサイクルモデルの参考として，SEC BOOKS『共通フレーム2007』の中で，テストに関わるプロセスを抽出し，次に整理しています。アクティビティを構成するテストに関連するタスクに着目してみると，システム要件定義，システム方式設計，ソフトウェア要件定義およびソフトウェア方式設計のアクティビティにテストに関するタスクがあり，共通フレーム2007でも，テストプロセスを開発のライフサイクルと並行して実施することとしています。

（取得）

プロセス	アクティビティ	タスク
1.1　取得プロセス	1.1.1　開始	1.1.1.9　受入れ方針及び条件の定義
	1.1.5　受入れ及び完了	1.1.5.1　受入れの準備
		1.1.5.2　受入れ

（次頁へ続く）

（開発）

プロセス	アクティビティ	タスク
1.6　開発プロセス	1.6.2　システム要件定義	1.6.2.2　システム要件の評価
	1.6.3　システム方式設計	1.6.3.3　システム結合のためのテスト要求事項の定義
	1.6.4　ソフトウェア要件定義	1.6.4.2　ソフトウェア要件の評価
	1.6.5　ソフトウェア方式設計	1.6.5.5　ソフトウェア結合のためのテスト要求事項の定義
	1.6.6　ソフトウェア詳細設計	1.6.6.5　ソフトウェアユニットのテスト要求事項の定義
		1.6.6.6　ソフトウェア結合のためのテスト要求事項の更新
		1.6.6.7　ソフトウェア詳細設計及びテスト要求事項の評価
	1.6.7　ソフトウェアコード作成及びテスト	省略(1.6.7.1～1.6.7.5)
	1.6.8　ソフトウェア結合	省略(1.6.8.1～1.6.8.6)
	1.6.9　ソフトウェア適格性確認テスト	省略(1.6.9.1～1.6.9.5)
	1.6.10　システム結合	省略(1.6.10.1～1.6.10.6)
	1.6.11　システム適格性確認テスト	省略(1.6.11.1～1.6.11.6)
	1.6.13　ソフトウェア受入れ支援	1.6.13.1　取得者の受入れレビューと受入れテストの支援

（運用）

プロセス	アクティビティ	タスク
1.7　運用プロセス	1.7.1　プロセス開始の準備	1.7.1.10　運用テスト計画の作成
	1.7.2　運用テスト	省略(1.7.2.1～1.7.2.4)

（次頁へ続く）

（検証）

プロセス	アクティビティ	タスク
2.4　検証プロセス	2.4.1　プロセス開始の準備	省略（2.4.1.1～2.4.1.6）
	2.4.2　検証	省略（2.4.2.1～2.4.2.7）

（妥当性確認）

プロセス	アクティビティ	タスク
2.5　妥当性確認プロセス	2.5.1　プロセス開始の準備	省略（2.5.1.1～2.5.1.5）
	2.5.2　妥当性確認	省略（2.5.2.1～2.5.2.5）

2.5.2　テストドキュメントの記述項目

次にテストプロセスで作成するテストドキュメントについて説明します。IEEE Std.829－1998では，次のような8種類のテストドキュメントの標準を定めています。その概要を**表2.4**に記載します。

表2.4　テストドキュメントの概要

テストドキュメントの名称	テストドキュメントの概要
テスト計画書 (Test Plan)	テストに関するスコープ，アプローチ，リソース，スケジュール，品質目標を記述した計画書。
テスト設計書 (Test Design Specification)	テストすべき機能(詳細なテスト条件および期待される結果)を特定し，それらの機能を適切にテストする方法を記述。 ・テストする機能 ・テスト方法の具体的な記述(使用するテスト技法など) ・テストケースの列挙
テストケース仕様書 (Test Case Specification)	テスト設計書に列挙された各テストケースを定義する(確認するテスト条件を走らせるために，テストデータを指定)。 ・テストケースで確認される機能 ・入力仕様 ・出力仕様 ・テスト環境要件(ハードウェア，ソフトウェア，他) ・テスト手順に関する特記事項 ・テストケースの依存関係(先に実施しておかなければならないテストケース)
テスト手順書 (Test Procedure Specification)	テストケースを実行する手順と，ソフトウェアがテストに合格したのかそれとも不合格なのかを判定するプロセスを示す。 ・目的 ・特殊な要件 ・実施手順(テストに合格したのかそれとも不合格なのかを判定する手順を含む)

表 2.4　テストドキュメントの概要(つづき)

テストドキュメントの名称	テストドキュメントの概要
テスト項目移管レポート (Test Item Transmittal Report)	テスト対象ソフトウェア構成品のテスト環境への移管報告書。 ・移管機能 ・場所 ・状態：移管される項目の状態を記述 ・承認
テストログ (Test Log)	テスト結果エビデンスを含むテスト実施報告書(テスト実行中に観察された何かしら意義ある詳細情報を時系列に記録として残す)。 ・説明 ・アクティビティとイベントの記述： 各イベントに対して開始と終了の日付と時間，テスト実行の簡単な説明，テスト結果，独特なテスト環境情報，観察された異常などを列挙
テスト不具合レポート (Test Incident Report)	テスト不具合レポート(テストの途中で観察されたさらなる調査を必要とする事象を文書化)。 ・不具合の要約 ・不具合の記述：入力項目，期待された結果，実際の結果，環境，再現性 ・影響：他のテスト計画，テスト設計仕様，テスト手順，テストケース使用に対して与える影響
テストサマリーレポート (Test Summary Report)	テストアクティビティの結果を要約し，結果に基づいた評価を記述する。 ・要約 ・変動：計画時の予想と現実との相違を報告 ・総合評価 ・結果の要約：未解決なすべての不具合を列挙 ・個別評価：制限事項を含む各テスト項目の総合的な評価 ・活動の要約 ・承認

また，**表 2.5**はテスト計画書の記述項目をIEEE Std.829－1998に沿って記載したものです。いずれも，テスト工程や自組織の文化・標準に合わせてカスタマイズして適用されることをお勧めします。

表 2.5　テスト計画書の記述項目

セクション	説　　明
1. テスト計画書識別番号 （Test Plan Identifier）	テスト計画書のバージョンの推移を記録するための識別番号。
2. 参考資料（References）	テスト計画書作成に参考とした資料。 例）プロジェクト計画書，要求仕様書，社内標準。
3. 序文（Introduction）	テスト対象となるシステム/ソフトウェアや機能の概要。
4. テスト項目 （Test Items）	テスト対象を具体的に記載。 例）ビルドバージョン，ソースコード名。
5. ソフトウェアのリスクに関する問題 （Software Risk Issues）	テストすべき事項の優先度を決める参考となるソフトウェアのリスク。 例）大きな金額を扱う機能，顧客に影響を及ぼしうる機能，欠陥歴のあるモジュール。
6. テスト対象機能 （Features to be Tested）	テスト対象となる機能。 例）xx制御機能，yy計算機能。
7. テスト非対象機能 （Features not to be Tested）	テストの対象外とする機能とその理由。
8. アプローチ（戦略） （Approach）	テスト対象をどのように検証するのか，といったテスト戦略。マスタテスト計画では特に各テスト工程でのアプローチや各工程の開始基準，終了基準を含む。
9. テスト項目の合否基準 （Item Pass/Fail Criteria）	各テストの期待される結果，テストの合格基準。 例）テストケース中のエラーの検出有無，欠陥の数・重大性・分布，テストカバレッジ。
10. 一時停止基準と再開要件 （Suspension Criteria and Resumption Requirements）	テストを一時的に停止する条件および再開の基準。 例）大量の欠陥，重大な欠陥，クリティカルパス上のタスクの未完。
11. テスト成果物 （Test Deliverables）	テストプロセスにより生成される成果物。 例）テスト計画書，テスト報告書，テスト実施ログ，欠陥レポート。
12. テストタスク （Remaining Testing Tasks）	テストを実行するために必要な作業項目。
13. 環境的要件 （Environmental Needs）	テスト作業に関して必要な環境。 例）ハードウェア，ソフトウェア，データ，インターフェース，施設，要員。
14. 責任（Responsibilities）	テスト作業に関わる組織や担当者の責任範囲。 例）職務と主担当者のマトリクス。

表 2.5　テスト計画書の記述項目（つづき）

セクション	説　　明
15. スタッフの配置とトレーニング（Staffing and Training Needs）	必要な要員の人数と要員に必要なスキル，ならびにスキルを身に付けるためのトレーニング方針や計画。
16. スケジュール（Schedule）	各テストタスクに要する時間，各テストタスクとマイルストーン，それらのスケジュール。
17. プランニングリスクと対応策（Planning Risks and Contingencies）	テストスケジュールに悪影響を与える可能性のある予定外のイベントや作業の遅延，ならびにその対応策。 例）要件の遅れ→範囲の縮小。
18. 承認（Approvals）	テストプロセスに関係する利害関係者の署名や捺印の欄。
19. 用語集（Glossary）	テスト計画書内で使用されている用語や頭文字の定義。

第３章　ソフトウェアテスト見積りでの成功と失敗例

ソフトウェアテスト見積り方法の詳細に入る前に，テスト見積りの特徴として，どのようなことに留意しなければならないのかを事例に基づいて紹介します。テスト工程全般に関する事例およびテスト見積りの前提条件となるテスト戦略に関する事例に基づき，それぞれの教訓とメッセージを示します。また，テスト進行中における品質評価・分析に基づく再テスト見積りなどに関わる事例から，教訓とメッセージを示します。

事例からわかるテスト見積りの留意事項，考え方および方法については3.1節，3.2節で説明します。また，テスト進行中における品質管理と再テスト見積りについては3.3節で説明します。

3.1　ソフトウェアテスト全般に関する事例

ソフトウェアテスト全般コストの見積りにあたって，基準となるソフトウェアのテスト生産性(効率)および規模(テスト量)をどのように設定するかは，重要な課題です。

なお，テスト生産性およびテスト量の見積り変動要因のうち，テスト戦略に関わる変動要因に関しては，3.2節で紹介します。

（１）　ソフトウェアテストの生産性見積り

①　テスト生産性に影響を及ぼす変動要因

ソフトウェアテストの生産性に影響する変動要因を把握し，ユーザとベンダとの間で合意をすることは，ソフトウェアテストの生産性見積りを行うにあたり必要なことです。ここでは，テスト生産性に影響する既存母体システムの品質状況に関わる事例を紹介します。

事例１は，既存システムの品質調査を重視し，品質状況を見積りに反映することにより，見積り精度の向上とプロジェクト成功につなげた事例です。

一方，事例２は，システム再構築開発において，新旧差分テスト時に既存母体システムの潜在欠陥が検出され，テスト工数が増加した失敗事例です。

事例1（成功事例）：

課題となっている状況・状態
［既存システム品質（残存欠陥密度）］による影響度についてユーザと調整し，品質向上の対応要否を判定し，見積りに反映した。具体的には，既存システムの品質を向上させる目的のテストを行うことで合意し，その分のテスト量を加算して見積もった。
見積りの成功/失敗の度合い
見積り時に［規模や生産性に影響を与える変動要因（プロジェクト固有）］について顧客と調整し合意を得ることを［見積りプロセス］として実現している。 見積り時点で改良対象の既存システム（一部）に残存欠陥が高い機能があることが判明していたため，品質向上の要否および具体的な作業（テスト実施）を調整することができた。
・発生工数：軽微
教訓・メッセージ
見積り時点で既存システムの品質を確認し，品質が低い場合（例：過去の障害発生の頻度調査など）には，品質を担保する作業が必要であり，見積りに配慮することが必要である。特に，既存システムの欠陥修正は改良作業着手前に実施することが望ましい（改良案件と同時に確認を行う場合，既存欠陥との識別ができないなど，欠陥原因の分析工数が増加する）。

事例2（失敗事例）：

課題となっている状況・状態
システム再構築開発の統合テストにて，新旧差分確認実施時に，原因不明の欠陥が多発し，調査（ソース解析）しながらテストを実施することになり，何度もリスケジュールをする事態に陥った。現行システムの潜在欠陥が原因であった。
見積りの成功/失敗の度合い
「現行不備については対応不要」という方針であったが，発生障害について根本原因を特定する手段がないため，別途，調査工数が追加発生しコストが増加した。
・テスト密度（ケース数/ソフト規模）：見積比1.0倍 ・テストコスト：見積比1.8倍
教訓・メッセージ
事例にあるようなシステム再構築開発においても，現行システムの母体品質によりテスト確認作業工数が増加する場合がある。改良開発と同じように現行システムの母体品質を見積り変動要因として配慮しておく必要がある。

②　欠陥特性とテスト生産性

ソフトウェアテスト見積りは，テスト計画のインプットにもなるため，テスト生産性見積りを行う際には，欠陥特性とテスト生産性との関係を考慮する必要があります。

ソフトウェアの欠陥特性について説明しますと，一般に，テスト実施の初期は，欠陥が多く検出されるとともに，欠陥解析や修正スピードは比較的速く生産性が高くなります。しかし，テスト後半になると，１件あたりの欠陥検出および解析は難しく(難バグ)，また，テスト内容も複雑になりテスト生産性が低下する傾向にあります。

事例３(失敗事例)：

課題となっている状況・状態
① 統合テストに際して，テスト項目ごとの重みを加味せず，テスト計画を策定した。その結果，テスト密度95%消化以降に複雑なテストが集中し，テスト環境待ちなど無駄な時間(コスト)を費やすこととなった。 ② 時間的制約から，障害発生時点で原因を詳細に分析せず放置してしまった。その後，時間が経過し追及が困難になり，結局，再テストとなってしまった。
見積りの成功/失敗の度合い
① テスト終盤の進捗が数倍遅くなり,テスト期間の延長とコスト増を招いてしまった。 ② 原因は単体テストレベルにおいて見逃した欠陥であり，プロジェクト関係者の全員出動による再テスト工数が増加し，生産性が低下した。
① テストコスト：見積比1.3倍 ② テストコスト：見積比1.1倍
教訓・メッセージ
① テスト見積りはテストマネジメントのインプットとなることを認識したい。 テスト生産性の見積りはテスト対象コンポーネント(ある機能を有するプログラム群)の複雑度などによる生産性の重み付け，およびテスト終盤での難バグ(欠陥原因の解析工数の増加および欠陥検出効率「検出件数/テスト量」の低下)を考慮しておく必要がある。 ② 本事例は，単体レベルのバグを放置したため，結果的に後工程で難バグ状態となり，テスト生産性を著しく低下させた。各テスト工程での欠陥管理を個人任せでなく，組織で管理する体制が必要である。また，テスト見積りの変動要因を抑制するためにも，障害解析・管理(ツール導入など工夫)を徹底すべきである。

事例3の2つの事例[①, ②]は，テスト項目の重み付けミスと欠陥特性(難バグ)の考慮不足によるテスト生産性低下とテスト終盤の混乱を招いた失敗事例です。

(2)　ソフトウェアのテスト量見積り

①　テスト量に影響を及ぼす変動要因

既に3.1節(1)で紹介したテスト生産性見積りに影響する変動要因とともに,テスト量(テストケース数など)に影響する変動要因に関する把握および変動要因のユーザとベンダとの相互合意は，ソフトウェアテスト量見積りにとっても必要なことです。

> 既存システムの残存欠陥は，テスト生産性見積りの変動要因になるとともにテスト量見積りの変動要因にもなります。また，新たな機能の追加や既存の機能を修正する場合に顕在化することがあります。このような既存システムの状況，特に品質については，事前に十分調査することが重要です。

事例4は，外部接続テストにて接続先システムの品質問題によるインターフェース変動が，テスト量を増加させた失敗事例です。

事例4(失敗事例)：

課題となっている状況・状態
外部システム(他社ベンダでの開発)との接続試験にあたり，相手システムの品質問題で，テストケースの修正・追加および接続試験の再実施が頻発しテスト工数が増加した。
見積りの成功/失敗の度合い
外部システムの起因で手戻りが発生することは想定しており，リスクを見込んでいたが，テストの初期～中間段階で顧客を通じて他社へヒヤリングした際には問題なしとの報告であったためリスクが消滅したと判断した。 実際はテストの最終段階になり低品質が露見し，再テストが発生し工数が増加した。
・テストコスト：見積比1.1倍
教訓・メッセージ
他責(ユーザまたは他ベンダの責)による再テストは，テスト見積り前提条件としてユーザとベンダ(他ベンダ含む)との相互で合意すべきである。特に，外部接続が多い統合テストやシステムテストなどは，契約方法の工夫や定量的な外部システム品質状況の把握などの手立てが必要である。

仕様変更の特質として，同じ内容の仕様変更をテスト工程で対応すれば，設計・実装およびテストのコストが，設計・実装工程で対応した場合に比べて増加します。つまり，テスト工程で仕様変更に対応するのは，棄却量(変更作業により棄却した成果物)が増加するとともに，リグレッションテスト量を考慮する必要があるからです。最終テスト段階のシステムテストが最大のコスト増となります。

なお，仕様変更見積りに関しての詳細は4.4節で紹介します。

事例５は，保留していた仕様変更をシステムテストで対応し，テスト負荷の増加と品質低下を招いた失敗事例です。

事例６は，仕様変更の特質への理解不足および見積り合意項目として，仕様変更量(正味棄却量含む)，仕様変更タイミングならびに変更認定基準などを，ユーザと事前に合意していないために工数が増加した失敗事例です。

事例５(失敗事例)：

課題となっている状況・状態
システムテスト工程で「保留していた仕様変更と新たな仕様変更」が大量に発生したが，テスト期間や要員資源を限定されたため，結果として十分なテストが実施できず品質低下を招いた。
見積りの成功/失敗の度合い
スケジュールは厳守したが，品質は大幅に低下した。
・テスト密度(ケース数/ソフト規模)：見積比0.5倍 ・本番障害発生頻度：見積比５倍
教訓・メッセージ
同じ機能の仕様変更対応でも，タイミング(取り込み工程)が異なると，対応コストが増減する特質がある。仕様変更の取り込みは，システムテスト工程が一番のコスト高(棄却工数増など)となる。また，ソフトウェアの品質劣化(デグレード)を招きやすい。このような仕様変更の特質をユーザとベンダ相互に理解することが重要である。

事例6（失敗事例）：

課題となっている状況・状態
統合テストにて，必要十分なテストを終えユーザにレビューを依頼したところ，提供した物件とは別に「あれも欲しい」「これも欲しい」と言われ，変更作業が増加し，結局，すべてのケースを再実行し，テスト工数が大幅に増加した。
見積りの成功/失敗の度合い
工数の増加および期日超過が発生した。
・テストコスト：見積比1.3倍
教訓・メッセージ
テストも終盤になると，システムの全体像が見えてくるので，ユーザ側の仕様変更が増加する傾向にある。一般的にユーザ側は仕様変更の特質（テスト工程での仕様変更はコスト高）を理解していない場合が多い。開発全体の見積り合意項目として，仕様変更タイミング，仕様変更量（棄却含む）および変更認定基準など，事前にユーザとすり合わせておくことが必要である。

② ソフトウェア改良開発におけるテスト量の特異性

ソフトウェア改良開発の場合は影響範囲が既存システムの特定部分に局所化されているか，それともシステム全体に分散しているかによって，テスト量は大幅に変化します。

なお，ソフトウェア改良開発におけるテスト見積りは，SEC BOOKS『ソフトウェア改良開発見積りガイドブック』で説明しています。

単純にテスト量は改良ステップ数に比例することを期待してコストを見積もると失敗してしまいます。事例7は，規模見積りにおいてテスト工程の規模を機能実現の規模とは区別し，改良ステップ数に加えて，既存システムに対する改良箇所の分散度合いによる影響を加味して見積もることによって，テスト量の見積りを精緻化し，顧客の納得を得るとともに，妥当な見積りに成功した例です。

事例8は，保守開発においてデグレード確認用テストデータを常に最新化することで，データ流用率を高めている成功事例で，事例9は，修正に対する影響調査不足によりテスト量の見積りミスを犯した失敗事例です。

事例7（成功事例）：

課題となっている状況・状態
見積り値として予定工数[改良規模（ソースコード行数）に係数を掛けてテスト工程の予定工数を算出]を提示してきた顧客に対して，以下の提案を行った。 ・既存システムに対する改良規模の分散度合い，およびデグレードテスト範囲をテスト工程単位に見積り，テストに影響する変動要因（生産性，量）を評価し反映する。
見積りの成功/失敗の度合い
初回見積りでは，コストで2倍以上の開きがあったが，設計終了後に再度，テスト見積り（テスト量）を提示し（分散度やテスト対象範囲が具体化），合意を得た。
教訓・メッセージ
改良開発の場合，改良分散度，テスト対象範囲および既存母体状況など，改良開発特有の不確定要素がある。改良開発特有のアクティビティとして調査・分析作業は必須である。 本事例のように段階見積りを行い，ユーザと合意することも一考である。

事例8（成功事例）：

課題となっている状況・状態
新規開発局面で保守局面を考慮して，テストデータ（統合テスト）をテスト標準として蓄積（最新化含む）することによって，保守でのテストに流用している。
見積りの成功/失敗の度合い
・業務データ理解に掛かる「工数」の削減が達成できている。 ・リグレッション用テストデータ作成工数の削減が達成できている。 ・個人差によるテストデータを誤る確率が減少している。
・テストコスト：見積比0.6倍 （データ準備コスト単独では見積比1.2倍） ・検出欠陥密度（欠陥数/ソフト規模）：見積比1.2倍
教訓・メッセージ
テストデータの保守（最新化）は一時的にコスト高になるが，ライフサイクルコストからのテストの見積りコストは安くなる傾向がある。

事例9（失敗事例）：

課題となっている状況・状態
保守開発において，ある機能の修正のために，その機能で使用している共通ライブラリ内の，あるサブルーチンを修正した。影響調査は担当者が知っている範囲で行い，テストもその範囲で行っていた。
見積りの成功/失敗の度合い
調査不足により，サブルーチンの修正仕様がそのサブルーチンを使用している他の機能の仕様と不整合があることに気がつかず，障害が起こってしまった。
教訓・メッセージ
改良開発では，共通モジュールの影響範囲が不明確になっている場合が多々ある。自己都合を優先し影響調査を怠ると障害につながる可能性が高い。

（3） テスト量が読みきれず，顧客と合意できない場合の対応

テストの網羅性を担保し難いテスト見積りに関しては，事例10のように資源面での制約，契約（準委任契約，段階的契約）および残存欠陥に関する顧客責任などの配慮が重要になります。

事例10（成功事例）：

課題となっている状況・状態
2000年対応，コード桁数対応などシステムテストのテストケースの設定によって，テストの網羅性を担保できないので，ユーザが提供するテストデータセットを満たすことを条件に見積もった。
見積りの成功/失敗の度合い
実際には，欠陥が収束しないために，テストデータセットの追加を3回行うことでシステムテスト期間が3ヵ月延びたが，ビジネス上の問題は発生しなかった。
・テストコスト：見積比2倍
教訓・メッセージ
開発の様態によっては，テストの網羅性をベンダ側で担保できない場合は，ユーザとベンダ相互の責任範囲および手段を明確にして，見積りに反映する。 ・資源（コスト，人，テストデータなど）の制約で見積もる ・契約面の配慮（準委任契約，段階的な契約）とする ・網羅性に関わる残存欠陥は顧客責任を前提とする

3.2　テスト戦略とソフトウェアテスト見積りに関わる事例

（1）　品質指標・目標値とソフトウェアテスト見積り

テスト戦略として品質指標項目を決定し目標値を設定することは必要であり，ソフトウェアテスト量を見積もるためのインプット条件となります。また，品質目標値にはレビューとテストの両面で取り決めておく項目(検出欠陥密度，テスト網羅率など)もあり，テスト見積りを行ううえで注意が必要です。

なお，品質目標値には，一般的にテスト密度(ケース数/ソフト規模)，検出欠陥密度(レビュー，テスト)，テスト網羅率，残存欠陥密度(レビュー，テスト)などがあります。

事例11の２つの事例[①，②]は，標準的な品質目標値をそのまま使用して失敗した事例です。保有する品質目標値の精度問題およびテスト対象システムの固有な特質や環境を見きわめたうえで適用することが必要です。特に，テスト見積りと関係が深いテストの網羅性「テスト網羅率」と残存欠陥密度(代替指標含め)は，テスト戦略において調査・分析評価し，テスト見積りの入力条件としてユーザとベンダ相互に合意しておくことが重要です。

事例12は，品質目標値(テスト密度)を類似業務システムから流用しテスト見積りの入力としていた失敗事例です。類似業務でもテスト環境などの違いから，そのまま適用するのは危険です。

事例13は，テスト計画の初期段階で設計方針を配慮して品質目標値(テスト密度)を評価しテスト量を見積って，テストケースレビューにてテスト内容の精度を高めた成功事例です。

なお,テストの網羅性(テスト網羅率)と残存欠陥密度をどのようにして設定し，ソフトウェアテスト見積りを導き出すかは，第４章で説明します。

事例11(失敗事例)：

課題となっている状況・状態
① 基盤開発で，ユーザ提示のテスト密度(50項目/Ks)とテストケース抽出方法を前提に，テスト見積りを行ったが，テスト量(テストケース数)の増加となった。 ② テスト密度と検出欠陥密度による品質評価を定期的に行い，予定どおりのテストを消化してきたが，次工程にて多くの障害が発生した。原因分析したところ，前工程でのテスト漏れと判明し，前工程のやり直しとなった。
見積りの成功/失敗の度合い
① 抽出方法が細かかったためテストケース増加となったが，幸いにも1ケースあたりのテスト実施時間が短時間で済んだため，コストへの影響は少なくて済んだ。 ② 障害の原因解析や前工程テストのやり直しによりテスト工数が増加した。 テスト密度がテストプロセスの品質を表さないことにより，このメトリクスを評価する意味がなかった。
① テスト密度(ケース数/ソフト規模)：見積比2.0倍 テストコスト：見積比1.05倍 ② テストコスト：見積比1.3倍
教訓・メッセージ
標準的な品質目標値を，そのまま，テスト対象システムに適用することはきわめて危険である。品質目標値はテスト戦略において，対象システムの特質などを見きわめ(分析・評価)，その結果をテスト見積りの入力情報にすることが必要である。

事例12(失敗事例)：

課題となっている状況・状態
新規顧客の受託開発において，類似業種システムの過去実績をベースにテストの見積りを行ったが，いざテストを実施する際に，統合テストのテスト密度の認識相違が判明して，テストケースが増加した。
見積りの成功/失敗の度合い
要員を増員してスケジュールは確保できたが，テストコストが増加した。
・統合テストのテスト密度(ケース数/ソフト規模)：見積比1.2倍 ・統合テストコスト：見積比1.2倍
教訓・メッセージ
類似業種システムの過去実績(品質目標値)をテスト対象システムに適用評価しないで採用することはきわめて危険である。品質目標値は，テスト戦略において対象システムの特質などを見きわめ(分析・評価)，その結果をテスト見積りの入力情報にすることが必要である。

事例13(成功事例)：

課題となっている状況・状態
業務共通設計，処理方式設計を考慮したチェックリスト(テストケース一覧)の作成計画がなかったので，画面遷移時のデータ引継ぎと項目間入力可否チェックなどが，チェックリストに漏れていた。そのため，新たに業務共通設計，処理方式設計に合わせたチェックリストを作成することにして，さらに，テスト計画の初期段階で，チェックリストをレビューすることにより，事前にチェックリストでの品質を作り込むことができた。
見積りの成功/失敗の度合い
テスト計画の初期段階で，チェックリストをレビューすることにより，事前にチェックリストの品質を作り込むことができた。
・プロジェクトコスト：見積比0.95倍
教訓・メッセージ
テスト戦略(計画)策定時の標準的な品質目標値の適用に固守することなく，テスト対象システムの業務設計や処理方式設計などを配慮したテスト設計において，テスト量(テストケース数など)を再見積りすることも必要である。

（2） ソフトウェアコンポーネントごとの重要度とソフトウェアテスト見積り

ソフトウェアコンポーネント（業務機能を実装するプログラム群）ごとの欠陥による障害の影響度（リスク度合い）と修正に対する優先度の検討は，テスト戦略として重要な検討項目です。特に，重要度に関しては，ソフトウェアコンポーネントごとのテスト量への重み付けを，明確にすることが大切であり，テスト量の見積りをするうえで必要です。

事例14は，サブシステムごとの重要度を優先付けせずに，標準的な品質目標値を一律に設定したことによる失敗事例です。

事例14（失敗事例）：

課題となっている状況・状態
規模・特性の違うサブシステムに対し，一律，標準的な品質目標値を適用し，テスト計画を策定した。その結果，テスト過多，テスト不十分な状況が発生し，テスト期間，工数が増加した。
見積りの成功/失敗の度合い
特定のサブシステムにおいて，欠陥が収束せず，テスト期間を延長し，追加テストを実施した。当該サブシステムは，システム全体の主要機能（重要機能）を有するため，全体の本番稼働時期にも影響した。
・テスト密度（ケース数/ソフト規模）：見積比1.3倍 ・テストコスト：見積比1.2倍 ・検出欠陥密度（欠陥数/ソフト規模）：見積比0.8倍
教訓・メッセージ
標準的な品質目標値を，そのまま，テスト対象システムに適用することはきわめて危険である。品質目標値はテスト戦略において，対象システムの特質などを見きわめ（分析・評価），その結果をテスト見積りの入力情報にすることが必要である。特に，システム全体に影響する重要機能は，テスト戦略（計画）でテスト優先度の重み付けを行い，テスト見積り（テスト密度の差別化など）として考慮しておくことが重要である。

（3） ソフトウェアの非機能要件とテスト見積り

非機能要件は「業務機能」に関する要求定義より，仕様面で明文化されていない場合が多く見られます。このような非機能要件のあいまいさは，テスト漏れやテスト優先度を低く評価する方向に向き，テスト量（テストケース数など）の見積りが過少になる傾向があります。

事例15，事例16，事例17は，効率性（時間効率性）に関する事例です。事例15はデータの伸び率を見積りにおいて考慮し成功した事例で，一方，事例16はデータの伸び率を考慮しないで失敗した事例です。事例17は，大量データ準備に時間とコストがかかるため，少量データで代替し失敗した事例です。業務への影響度が大きいソフトウェアコンポーネントの場合は，テスト戦略においてテストデータ量の算定を十分に検討すべきです。事例のようにソフトウェアプログラミング工程での単純ミスが原因となる場合もあり，プログラムソースレビューの徹底も意外と効果が上がります。

事例15（成功事例）：

課題となっている状況・状態
開発期間２年という大型改良開発案件で，本番後の想定データ処理件数が提示された。しかし，開発期間中での状況変化（処理件数増加）を含め，既存データベースへのデータ件数増加を配慮したタスク「現行処理性能判定」を提案し認められた。当該タスクは機能追加・修正のない現行処理を対象に行った。
見積りの成功/失敗の度合い
機能追加や修正のない現行のDB検索処理で大量データによる性能確認を行ったところパフォーマンスの悪い（不適切な検索キーを使用している）処理が発見できた。テストデータ量は提示いただいた本番後の想定件数の２割掛けとした。
教訓・メッセージ
改良開発は既存母体システムの品質（ここでは性能面）によりテスト工数が影響される。既存母体システムの品質に問題がある場合（過去の障害状況分析など）には，事前に，問題があるコンポーネント（ある機能を有するプログラム群）の事前検証をするなど，テスト見積りの変動要因となる要素を排除しておくことが重要である。

事例16(失敗事例)：

課題となっている状況・状態
改良開発で月次バッチ処理データ件数について，「オンライン月間投入データ件数(予測最大値)と同等」として高負荷テスト実施量を見積り，負荷テストを実施したが，現行業務データで発生する派生データを考慮していなかった。本番で派生データが大量に発生し，追加月次処理のパフォーマンスが劣悪となったため，処理方式の見直しおよび再テストを実施した。
見積りの成功/失敗の度合い
当社に現行システムノウハウがなかったことも一因だが，データ量(伸び率)の想定を安易に考えていた。現行システムの調査(現行バッチ処理件数とオンライン投入データ件数との比較)タスクを準備すれば未然に防ぐことができたはずである。
・再テストを含むテスト工数：見積比1.4倍(データ作成を含む)
教訓・メッセージ
非機能要件(性能など)のテスト設計では，現行処理状況(投入データ数と処理データ数との因果関係)も調査したうえで，性能測定の基準数値(最大処理件数)を決めるべきである。 データ量の伸び率を見積もるうえで，利用部門などの参加(レビュー含め)が必須である。

事例17(失敗事例)：

課題となっている状況・状態
テスト期間が短い開発において，整合性のとれた大量データ(複数データベース：100万件)が準備できず，データ量不足(50万件)のままテストを実施した。データベースに対する非効率な抽出，検索を摘出できないまま本番を迎えてしまい，パフォーマンス低下による障害が発生してしまった。また，原因はコーディング規約ミスであった。
見積りの成功/失敗の度合い
発生した非効率処理に対し，すべてチューニング作業を実施することになった。
・追加テストを含む総コスト：見積比1.3倍
教訓・メッセージ
開発期間(短納期)やデータ(大量データ)などの制約条件がある場合は，設計・実装工程の品質作り込み(レビューなど)コストとテスト工程の確認テストコストとを配慮したテスト戦略(計画)が，特に，重要となる。また，早期のテストデータ作成および単体テスト・統合テストなどの早い段階での大量データ(データベース間整合性を求めず)での確認といったタスクの検討を考慮するなどの工夫が必要である。

事例18は，保守性(障害解析性)に関わる事例で，機能[ログ出力]のテスト優先度を低くしたことによる失敗事例です。非機能要件の中には保守性のように，開発時には軽視される傾向がある品質特性があります。事例19の障害回復性も期間制約による接続障害テストケースを省いた事例です。両事例ともシステム本番稼働後に甚大な障害となる事例として認識したいものです。

事例18(失敗事例)：

課題となっている状況・状態
保守性(障害解析性)を高めるため「ログ出力」機能を組み込んだ。単体テストにて出力機能の確認ができたので，後続のテストでは目検(プログラム机上確認)としていた。本番移行後に障害が発生したため出力ログを確認したところ，解析に有効な項目が欠落していた。
見積りの成功/失敗の度合い
後続のテスト見積り(計画)を行う際，要求レベルの高いパフォーマンス測定に当初想定以上の工数がかかることが判明したため，ログ出力確認テストは目検(プログラム机上確認)として計画してしまった。結果，ログ出力機能を再確認することになった。
・システムテスト工数：見積比1.2倍
教訓・メッセージ
非機能要件の確認テストは，‘手間がかかる’といったことから，開発の状況(予算超過状態など)に左右され，おろそかになることが多い。テスト戦略において重要度を認識すべきである。

事例19(失敗事例)：

課題となっている状況・状態
対外接続システムの障害回復確認について，事前にテストシミュレータを用いてテストを実施していた。実際に外部接続状態でのテストにあたりスケジュールがタイトであったため事前テストを理由に，接続障害ケースを絞り込んでしまった(計画変更)。本番後，想定内の接続障害(インターフェースの不備)が発生してしまった。
見積りの成功/失敗の度合い
対外接続テストの実施は外部を巻き込むため，やり直しや計画変更は簡単にできないものでありながら，目先の状況にとらわれ安易に当初計画を変更してしまったことが問題である。その結果，本番障害対応のほか外部(他社)と調整のうえ，再度，障害回復テストを実施することになった。
・接続テストコスト(本番障害対応および再テスト実施を含む)：見積比2倍
教訓・メッセージ
外部接続テストなど，外部(他社)とやり取りをする確認テストは，手間がかかりかつ計画も簡単に変更できないといった制約がある。テスト戦略において，ひっ迫した状態であっても「安易に計画変更してはならない作業」を特定しておくことが重要である。

(4)　テストプロセスとテスト見積り

ソフトウェアテスト見積りは，品質保証の観点でのモデル(2.2節参照)の選択によりテストレベルごとのテスト量(テストケース数など)が違ってきます。また，対象システムの特性や規模によってどのモデルを選択するかは，テスト戦略にて策定しておく必要があります。

ここでの事例は，主にテスト手順に関わるテスト生産性とテスト量への変動要因事例について紹介します。

事例20の3つの事例[①，②，③]は，テスト手順に関わる工夫によるテスト生産性の向上事例です。

事例21の3つの事例[①，②，③]は，逆に，期間短縮要求から安易なテスト手順書の省略などによるテスト生産性の低下と，リソース別管理(フレームワーク，SQL)によるテスト量の見積りミスを犯した失敗事例です。

事例20(成功事例)：

課題となっている状況・状態
① 事前に，以下のようなテスト手順を取り決めてからテスト実施を行った。 ・相互に依存しないテストケースを用意し，独立した処理を固めてから，連携処理のテストを実施した。 ・テスト環境とテストデータの準備が効率的に実施できるテスト順序とした。 ② 多量なデータバリエーションをとるサブルーチンのテストに関し，テストデータの重複排除を狙って，データバリエーションを呼び出し元のテストデータに設定することで，テストの効率化を図った。 ③ オフショア先の開発で,品質を確保するために,レビューについては,標準チェックリストをオフショア先に示すと同時に，現地で開発者に説明会を実施し，開発者のレビューの徹底を図った。また，プロセスQAを実施した。
見積りの成功/失敗の度合い
① 再テストの減少，テスト期間の短縮が図れ，テストコストも削減できた。 ② 呼び出し元ではサブルーチンとのインターフェース確認テストのほか，他の案件でのテストもあり，単一データで複数のテスト項目を消化できた。 ③ 国内開発と同程度の品質を確保することができた。
① テストコスト：見積比0.9倍 ② テストコスト：見積比0.9倍 ③ 検出欠陥密度(Java)：(オフショア/国内)比1.0倍 検出欠陥密度(COBOL)：(オフショア/国内)比0.9倍
教訓・メッセージ
① テスト設計において，テストケースの実施順序およびテスト実施手順を決めておくことは，データの重複性の排除，テスト実施の効率化，テスト環境の共有および結果確認の容易さなど，テスト効率の向上が期待できるので，考慮することが必要である。 ② テスト設計において，テストの重複を見きわめ，単一データでの項目消化量を増加させ，データ量を削減するなど，テストの生産性向上が期待できるので，テスト見積りとして考慮する必要がある。ただし，今回の事例のような場合は，サブルーチンの品質が一定以上確保できていることが前提であるなど，条件面の合意は必要だ。 ③ プログラムの欠陥は，ソースレビューのチェック方法を策定し徹底することで削減できる。テスト見積り(プログラミング工程)では，ソースレビューと単体テストの両面で残存欠陥密度を見積もることも重要である。

事例21(失敗事例)：

課題となっている状況・状態
① 統合テストに際して，ユーザ要求によりコスト削減(期間短縮)を行う必要が発生したために，テスト手順書作成を省略し，テスト項目一覧とテスト条件・確認ポイント一覧のみでテストを実施した。しかし，結果，手順の不明確さからテストリトライ回数が増加し，統合テストの効率が著しく低下した。 ② バッチ処理の結合テストにおいて，データの処理順序を考慮せずに結合テストを実施したため，テストデータを用意するのに想定以上の工数がかかり，テストの品質も不十分であった。 ③ フレームワークの関係にてSQLがソースとは別管理となっており，見積り時にSQLの改修規模を考慮していなかったためにテスト工数が大幅に増加した。
見積りの成功/失敗の度合い
① 開発期間短縮を意図して作業を省いたにもかかわらず，短縮した以上の期間を統合テスト工程で浪費してしまった。 ② テストデータ作成にかかる工数の増加,データ値の組み合わせが実際とは異なってしまったことによりテスト品質が低下した。 ③ もともとの改修が小規模のものであったため，開発スケジュールに大きな影響はなし。 ただし，テストケースの増加によりテスト工数が増加した。
① テストコスト：見積比1.5倍 ② テスト工数：見積比1.2倍程度(正確には測定できていない) ③ テスト密度(ケース数/ソフト規模)：見積比1.0倍 テストコスト：見積比2.0倍
教訓・メッセージ
① テスト手順書作成の範囲および記述の深さなどは，テストのプロセス標準として，見積り時点で決めておくことが必要である。また，テスト戦略ではテストのプロセス標準に対して，テスト対象システムへのテーラリング方針を策定し，見積り前提条件として明文化しユーザと合意しておくことが重要である。 ② テスト設計において，データの処理手順を考慮してテストケースの実施順序を決めておくことは，データの重複性の排除，テスト実施の効率化，テスト環境の共有および結果確認の容易さなど，テスト効率の向上が期待できるので，考慮する必要がある。 ③ 保守など改良開発では，改良開発特有の改良母体システム(改良対象量，母体品質など)の見きわめが見積りをするうえで必要条件である。

（５） テスト環境とテスト見積り

テスト環境の具備状態は，ソフトウェアテスト見積りの生産性やテスト量に影響しますので，テスト戦略において，テスト環境の構築検討は重要になります。また，テスト見積り時点では，見積りの変動要因として捉える必要があります。

特に，汎用的な自動化テストツールの適合性やテストツールの実行環境の整備などは，テスト効率を見積もるうえで留意すべきです。

事例22の６つの事例[①，②，③，④，⑤，⑥]は，対象システムのテスト効率の観点から，自前でテストツールを作成し，テスト生産性とソフトウェア品質の向上を図った成功事例です。

事例23は，テストツールの実行環境検証，ツールの性能検証および習熟度の向上とともにテスト生産性見積りを事前検証し，成功した事例です。

特に，新たなテスト環境は，不確定要素が多く，事例のように事前検証をテスト戦略として策定しておくことが大切で，テスト見積りの精度を高めるうえでも重要です。

一方，事例24の２つの事例[①，②]の１つ目は，テストツールの保守に関わる点から失敗した事例です。テストツールは時の経過とともに適応効率の低下，また，機能面や実行環境の違いなどから，必ずしも見積もったテスト生産性ほど効果が上がらない場合がありますので，注意する必要があります。

２つ目の事例は，自動ソフトウェア生成ツールで作成される生成部品の内部構造を解析しない前提でテスト生産性を見積もったが，実態は自動ソフトウェア生成ツール仕様の問題により，生成部品の内部構造解析作業が追加され，テスト見積り工数が増加した失敗事例です。

事例22(成功事例)：

課題となっている状況・状態
① オンライン入力を行うテストツールを作成し，単体テスト，統合テストで効率的なテストを実施したため，システムテスト(ST)では欠陥が少ない。生産性も高く，サービスイン後のトラブルもない。 ② 統合テストにて，従来の試験ツールを用いず，特定機能の専用試験ツールを新規に作成して適用した。結果，テスト効率とシステム品質が向上した。 ③ 短期間で品質向上させるために，本番DBから，指定した条件でテスト用ミニDB(個人情報はマスク)を抽出するツールを開発し，テスト工数減少，システムテスト代替効果による期間短縮を実現した。 ④ 改良開発において，変更ごとにリグレッションテスト(同じテストパターンである)があるゆえ，自動実行ツールを作成し，テストの効率化が図られた。 ⑤ 負荷が大きいリグレッションテストにて，ツールを活用したため，デッドロックなどに代表されるタイミング障害に対する品質と生産性が向上した。 ⑥ 本番環境データから必要項目をスクランブル化したうえでテスト環境にダウンロードするツールを作成し，統合テストで使用した。結果，テスト効率もシステム品質も向上した。特に，要件定義～外部設計時点で明らかになっていなかった，レアケースの存在が明確となった。
見積りの成功/失敗の度合い
① ST欠陥密度(欠陥数/ソフト規模)平均の0.5倍，生産性(規模/工数)平均の1.4倍 ② テストコスト：見積比0.8倍，欠陥密度(欠陥数/ソフト規模)：見積比1.2倍 ③ テスト密度(ケース数/ソフト規模)：見積比100倍，コスト：見積比0.8倍 検出欠陥密度(欠陥数/ソフト規模)：見積比1.2倍 ④ テストコスト：見積比0.8倍 ⑤ テストコスト：見積比0.8倍 ⑥ テストコスト：見積比0.7倍(データ準備コスト単独では見積比0.2倍) 検出欠陥密度(欠陥数/ソフト規模)：見積比1.2倍
教訓・メッセージ
テスト戦略の戦略策定項目の１つとして，効果的なテストツールの導入を検討(費用対効果)しておくことは，テスト実施工程でのテスト生産性に影響するため，テスト見積りのインプット条件として必要である。ただし，テスト見積りの際は，自動化テストツールは万能ではないこと，ツールの保守状況を十分に調査することも必要である。

事例23(成功事例)：

課題となっている状況・状態
テストにおけるパイロット(事前検証)を実施することで，テスト環境の問題点の早期検出と実施手順の整理を行うことができ，テスト工数を削減できた。
見積りの成功/失敗の度合い
・パイロットでテスト環境の問題点を検出できたため，テスト実施中の環境問題の発生を抑えることができた。 ・実施手順を事前に整備しておくことで，テスト開始時のロスが削減できた。
・テストコスト：見積比0.8倍
教訓・メッセージ
テスト環境の事前検証を行うことで，テスト環境の課題が早期に確認できる。テストプロセスの試行を実施した場合の費用対効果を鑑み，テスト戦略として計画しておくことが大切で，その評価を見積りに反映するなどの工夫が重要になる。

事例24(成功事例)：

課題となっている状況・状態
① 改良開発での統合テストにおいて過去に作成した予想結果生成ツールを前提とした生産性を設定したが，ツールの非変更箇所における保守不備により予想結果が正しく生成されず，テスト工数が増加した。 ② 自動生成ツールで作成されたプログラム(部品)のテストは，ブラックボックスとしてテスト量(テスト仕様&テストケース)を見積もったが，実態はホワイトボックスとしてテスト量やテストデータを作成する状況となり，テスト工数が大幅に増加した。
見積りの成功/失敗の度合い
① 原因解析作業によりテスト工数が増大した。 ② 自動生成されたプログラムは自動化ゆえの重複度，冗長度が高くテスト量(テストケース)と内容理解による生産性低下など，テストコスト増と開発期間の延長を起こした。
① テストコスト：見積比1.5倍 ② テスト密度(ケース数/ソフト規模)：見積比1.8倍 テストコスト：見積比1.5倍 検出欠陥密度(欠陥数/ソフト規模)：見積比0.85倍
教訓・メッセージ
① テストツールは効果が得られる面もあるが，反面，時間の経過や開発環境の変化などで，効果が減少する可能性がある。テスト戦略におけるテストツールの評価(費用対効果)は，見積り前提条件として考慮する。 ② 自動生成ツール(ソフトウェア部品の生成)を同時並行に開発している場合は,生成部品の品質(プログラムバグ，インターフェースの変更，利用マニュアル不備など)によるテスト生産性の低下を考慮する必要がある。テスト見積りを行う際には，提供生成部品の品質などに関し，前定条件として明文化するなどの配慮が必要である。

（６）　テスト分担とテスト見積り

テスト戦略において，テスト分担は役割内容，提供時期および提供物の精度などを５Ｗ１Ｈで取り決めておくべきです。テスト見積りは，それらの内容を見積り条件として明文化しておくことが重要です。意外とテスト進行中において，テスト分担によるトラブルが多く見られます。

事例25の２つの事例[①，②]は，外部提供システムおよび提供データの品質問題などにより，テスト期間の不本意な短縮およびテスト効率の低下を起してしまった事例です。事例26の２つの事例[①，②]は，外部提供プログラム部品やデータを得られると思っていたが，得られなかった失敗事例です。また，事例27は，開発とテスト全般にわたり，同一担当者に丸投げ依存した結果の失敗事例です。

一方，事例28は，当該ソフトウェア設計担当者以外の開発要員をテストに参加させ，内部構造にとらわれないユーザの視点による妥当性確認をベンダ側で行い，早い段階で品質を安定させ成功した事例です。

事例25（失敗事例）：

課題となっている状況・状態
① 外部システム（他社ベンダでの開発）の開発が遅延し，しかも，外部接続テストの直前に外部インターフェース仕様が変更されていたことが判明した。 ② 統合テストに際して，外部提供データは正しいデータを前提にテストしたが，提供データそのものに欠陥があり，統合テストの効率が著しく低下した。
見積りの成功/失敗の度合い
① 開発納期の変更がなかったためテスト期間が短縮されテストの生産性が低下，さらに設計・実装工程の修正およびテスト仕様（テストデータ含む）の修正工数が増加した。 ② テスト結果の原因解析作業や再テスト実施回数増によりテスト工数が増加した。
① テストコスト：見積比1.1倍 ② テストコスト：見積比1.2倍
教訓・メッセージ
他責（ユーザまたは他ベンダの責）による追加作業（再テストなど）は，テスト見積り前提条件としてユーザとベンダ（他ベンダ含む）との相互で合意すべきである。特に，外部接続が多い統合テストやシステムテストなどは，契約方法の工夫や定量的な外部システム品質状況の把握などの手立てが必要である。

事例26(失敗事例)：

課題となっている状況・状態
① 対外接続テストについて，テスト用外部受信データを外部システムの開発担当会社より入手できるものと思い込んでいたが，自社で準備する必要があった。テストデータの作成工数が増加したため，対外接続テストの効率が著しく低下した。 ② 既存の部品を組み込んだシステムの開発において，正味の開発規模をベースにテストの見積りを行った。実際のテスト局面では，部品が司る機能のテストまで要求され，テスト項目が増加した。
見積りの成功/失敗の度合い
① テストデータ作成工数増加により，テストコスト増となった。 要員を増やしてスケジュールは確保できた。 ② ただし，部品が司る機能のテストではほとんど欠陥が検出されなかった。
① テストコスト：見積比1.2倍(データ準備コストのみ増加した) ② 統合テスト密度(ケース数/ソフト規模)：見積比1.3倍 統合テストコスト：見積比1.2倍
教訓・メッセージ
① テストに関わる役割分担は，テスト見積り前提条件としてユーザとベンダとの相互で合意すべきである。特に，外部接続が多いシステムテストなどは，契約方法の工夫や定量的な外部システム品質状況の把握などの手立てが必要である。 ② 既存ソフトウェア部品の組込みは，既存部品の欠陥および既存部品の非ブラックボックス化(部品の内部構造まで調査要)によるテスト生産性の低下が，時として発生する。テスト見積りを行う際には，部品の利用状況調査，パイロット検証および見積り前提条件として明文化するなどの配慮が必要である。

事例27(失敗事例)：

課題となっている状況・状態
開発当事者が統合テストケースを設計したため，見た目のテスト密度は十分であったが，品質はその工程では十分に確認できず，システムテスト工程で欠陥が多く検出され，結果的に同工程のやり直しとなった。
見積りの成功/失敗の度合い
欠陥の原因解析や前工程テスト(テストケース設計から)のやり直しによりテスト工数が大幅に増加した。
・テストコスト：見積比1.5倍
教訓・メッセージ
テスト設計への第三者の介入方針(開発当事者以外のテスト設計担当/テスト設計レビュー参加など)は，対象システムの特質や開発量などを鑑みて，テスト戦略において設定することが重要である。テスト見積りの入力となるテストの網羅性や残存欠陥密度は，担当者能力の偏在などを考慮して，極力，変動幅を小さくする工夫を行うことが前提である。

事例28(成功事例)：

課題となっている状況・状態
ユーザインターフェース(UI)を持つ会話型のAPIのテストにおいて，そのアプリケーション設計に参加しなかった要員をテスト要員として参加させたところ，設計者が思い至らなかったような操作による不具合を早期の段階で検出でき，品質が早期に安定できた。
見積りの成功/失敗の度合い
ユーザインターフェースに関わる不具合を早期に検出できたことによる品質の安定の達成。テスト要員のモチベーションの向上。
単体テストにおける検出欠陥密度(欠陥数/ソフト規模)：見積比1.2倍　(推測)
教訓・メッセージ
ユーザインターフェースなど妥当性確認テストを対象にしたテストケースは，内部構造にとらわれない利用者側の発想を持ったテスト要員の参加も重要であり，妥当性確認テストにおいてテスト量の見積りを行う際には必要となる。

3.3　テスト進行中における品質管理と再テスト見積り

既に3.2節(1)で説明した品質目標値は,ソフトウェアテスト見積りのインプット条件となるとともに，テスト進行中における品質管理(予実管理)のインプットにもなります。

品質目標値の予実分析結果により，追加テストケースなど，テスト量の再見積りが必要になる場合があります。その再テスト見積りにはベンダに起因する欠陥とユーザに起因する欠陥とが存在しますので，ユーザとベンダ相互に見積り前提条件などで責任範囲を明確にしておく必要があります。

事例29は，テスト進行中におけるテスト密度，検出欠陥密度と本番稼働後の欠陥密度とを予実管理し相関関係を分析することで，品質指標を精錬させ，テスト見積り精度向上を目指した事例です。

事例30の2つの事例[①，②]は，テスト進行中での品質目標値の予実管理と定性的な欠陥状況分析を組み合わせた品質管理を紹介しています。テストの早い段階で品質管理を徹底した結果，品質目標値の再設定(テスト密度の再設定による

事例29(失敗事例)：

課題となっている状況・状態
統合テストおよびシステムテストのテスト密度・検出欠陥密度の実績と，本番稼働後の欠陥密度との相関関係を分析することで，テスト見積りの精度を向上している。
見積りの成功/失敗の度合い
傾向として，統合テストおよびシステムテストの検出欠陥密度が平均値より高いと，本番稼働後の欠陥密度も高くなる。
・平均的なシステムテストの検出欠陥密度および本番稼働後の欠陥密度 検出欠陥密度(2件/10KS)，本番稼働後の欠陥密度(0.2件/10KS) ・品質劣化したシステムテストの検出欠陥密度および本番稼働後の欠陥密度 検出欠陥密度(12件/10KS)，本番稼働後の欠陥密度(2件/10KS)
教訓・メッセージ
設計・実装工程(基本設計～コーディング)での検出欠陥密度(レビューなど)，テスト工程でのテスト密度および検出欠陥密度(確認テスト)，さらに本番稼働後の欠陥密度との相関分析を行うことにより，プロセス改善点を発見できるとともに品質指標の精度を高められる。

テストケースの追加)およびテストプロセスの見直しによるソフトウェア品質を確保した成功事例です。

一方，事例31の２つの事例[①,②]は，１つ目は品質目標値の不明確さと相まってテスト進行中での予実管理の不徹底による失敗事例，２つ目は信頼度成長曲線管理(事例ではグロス累計欠陥管理を適用)に依存し，特定の劣悪なソフトウェア

事例30(成功事例)：

課題となっている状況・状態
① 単体テストの完了時に，欠陥の傾向として偏りがあった。 ・欠陥の偏り：画面表示形式，項目出力不正，表示不正 ・検出不足：エラー処理の欠陥が少ない 現象別・原因別の品質マップを作成することによる欠陥の傾向に応じて品質目標値を再設定して，是正処置(追加テスト)を実施した。 単体テストの後に是正処置(追加テスト)を実施したので，組み合わせテスト開始日は遅延したが，組み合わせテスト完了日は目標を達成できた。 ② プログラム品質の中間評価により方向性を決める。 プログラム品質の中間評価で，設計不良が多くあったので，一旦，テストを中止して設計を見直した。 プログラム品質の中間評価により，テスト段階での手戻りを削減できる。
見積りの成功/失敗の度合い
① 単体テストの後に是正処置(追加テスト)を実施したので，組み合わせテスト開始日は遅延したが，組み合わせテスト完了日は目標を達成できた。 ② プログラム品質の中間評価により，テスト段階での手戻りを削減できた。
① テスト密度(ケース数/ソフト規模)：見積比1.2倍 検出欠陥密度(欠陥数/ソフト規模)：見積比0.9倍 ② プロジェクトコスト：見積比0.9倍
教訓・メッセージ
① 品質目標値はテスト見積りとテスト進行中における品質管理(予実管理)のインプットになる。この例では，品質マップを作成し欠陥偏在分析を行うことにより，品質目標値の再評価(再設定)を行い，さらに是正処置として再テスト量を見積り(追加テスト)，品質を担保している。 ② テスト戦略において，設計不良のテスト量は，設計工程での残存欠陥密度(レビューなどで検出しきれない残存欠陥数)を予測して見積もる。しかし，テスト工程において設計不良が予測値より多く検出された場合は，設計工程の見直しを早期に行うことが必要である。

コンポーネントの品質状況を見逃したことによる失敗事例です。ソフトウェアの品質は，特定のソフトウェアコンポーネントの品質が極端に劣化した場合，システム全体の品質へと影響範囲が拡大してしまう危険性があるので，ソフトウェアコンポーネント単位のきめ細かい品質指標を管理する必要があります。

事例31(失敗事例)：

課題となっている状況・状態
① 複数のサブシステムからなるシステムにおいて，欠陥の収束状況をシステム全体の信頼度成長曲線にて評価していたが，突然，欠陥が検出されるようになり，手戻りが発生した。実態はサブシステムごとの収束状況が一律でなく，ある特定の劣悪なサブシステムに全体が引きずられた結果となった。 ② 品質目標値との乖離がある場合の対処について，テスト計画時点で明確でなく，判断材料となるべく資料も十分に採取・管理されていなかった。そういう状況下，欠陥が収束せず，テストが完了できなかった。 結果，追加テストを実施し，テストコスト増となった。
見積りの成功/失敗の度合い
① 欠陥の収束状況を見誤ってしまい，手戻りによるテストコスト増が発生した。 ② 情報が不十分なため，テスト完了判断ができなくなり，さらに追加テスト発生によりテスト工数が増加した。
① テストコスト：見積比1.2倍 ② テスト密度(ケース数/ソフト規模)：見積比1.3倍 テストコスト：見積比1.2倍 検出欠陥密度(欠陥数/ソフト規模)：見積比0.8倍
教訓・メッセージ
① 信頼度成長曲線を用いて，システム全体の(コンポーネント単位などではなく)グロスの管理だけで欠陥の収束度を管理するのは危険である。また，ソフトウェアテストの特質として，特定のソフトウェアコンポーネント(ある機能を有するソフトウェア群)の品質が劣悪であると，全体のテスト効率低下となる傾向にある。テスト見積りではこのような生産性変動要因に関わるリスクをヘッジする配慮も必要である。 ② テスト戦略(計画)としてのテスト完了基準は，ユーザとベンダとの相互同意が前提である。テスト完了基準は定量的なテスト網羅性(残存欠陥密度)だけではなく，欠陥の修正状況および検出した欠陥の発生傾向などの定性的な基準策定を考慮することが必要である。また，テスト進行中や完了時点の品質目標値の評価・分析は，テスト見積り精度の向上を目指すためにも必要である。

3.4　テスト戦略の重要性

本ガイドブックの3.1～3.3節では，ソフトウェアテスト全般に関わる事例を10件，テスト戦略に関わる事例を18件およびテスト進行中における品質管理に関わる事例を3件，それぞれ紹介しています。その事例の内容および事例数から鑑みても，テスト戦略を策定し，ユーザとベンダとが相互に合意することは大切です。また，テスト戦略はソフトウェアテストの見積りのインプットになるため，見積り精度向上の観点からもきわめて重要です。

テストとは，システムの特性を立証し，実態と要求されている仕様との間の差異を明らかにすること(言い換えれば欠陥を検出すること)を目的とした計画，準備，測定のプロセスであり，品質管理におけるテストの役割の概要は次のとおりです。

テストという行為は，単にシステムの品質強化に役立つ1つの手段にすぎません。したがって，通常は，システムがある品質レベルを達成していることを確認するための測定システムにテストを組込み，ソフトウェア開発プロジェクトの品質管理活動の中にソフトウェアテストを組込んでいます。**図3.1**にその関係を示します。

品質保証の活動[9]は，予防，検知，修正に大別され，品質保証活動を構成する個々の手段(予防活動,検知活動および修正活動)相互の関係が非常に重要であり，テストは独立のアクティビティではなく，検知活動中の1つの手段と位置づけられます。したがって，ソフトウェアのテストは他の検知手段であるレビュー，シミュレーション，インスペクション，監査，サンプリング，机上チェック，ウォークスルーなどと密接に関連します。

また，ソフトウェアテストの完了は，ソフトウェアテストの費用対効果の判定に基づいて行うこととされています。言い換えれば，ソフトウェアのテストは欠陥の検出と修正コストが，ソフトウェアの欠陥による運用中の障害による損失コストより低い間は，ソフトウェアテストを継続するということです。なお，実際には費用対効果の判定結果をソフトウェアに要求する品質目標として設定していることを前提に，ソフトウェアテストの完了は，要求される品質が確保された(担

(9)　ISO 8402では，品質管理を「成果物やサービスが品質要求事項を満たすことについてのITシステムの利害関係者に十分な信頼感を供するための，すべての計画的，体系的な活動」としています。

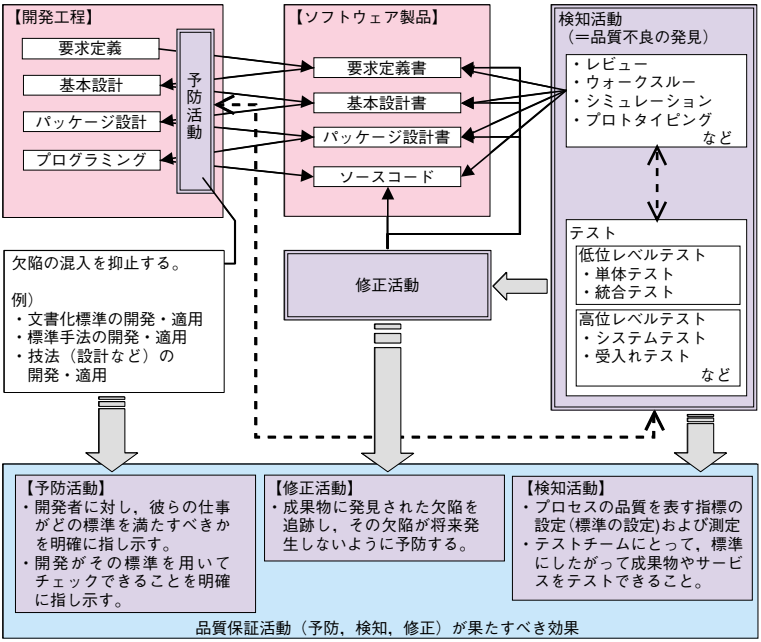


図 3.1　品質保証活動におけるソフトウェアテストの位置づけ

保された)との判断に基づいて行われます。

3.4.1　テストレベル

テストはプログラムが詳細設計どおりに機能するかどうか，アプリケーションが機能どおりに機能するかどうか，さらにシステムがユーザのニーズ・意図を十分に満足しているかどうかを確認するために必要であり，これらのテストを効果的に組織化するために，さまざまな要求，機能仕様，技術仕様に関わるいくつものテストレベル(単体テスト，統合テスト，システムテスト，受入れテストなど)が用いられます。**図 3.1**では，主にベンダが主体でユーザと協力して実施することが多い単体テストおよび統合テストを便宜的に「低位レベルテスト」と呼び，主にユーザが主体でベンダと協力して実施することが多いシステムテスト，受入れテストおよび運用テストなどを「高位レベルテスト」と呼んでいます。

3.4.2　テスト戦略

それぞれのテストレベルに対応する要求と，要求が満たされていない場合のリスクに基づいて，それぞれのテストレベルで最も重要な欠陥をできるだけ迅速に効果的に検出できるように「テスト戦略」を設定します。

特に，主にベンダが主体でユーザと協力して実施することが多い低位レベルテストの戦略と，主にユーザが主体でベンダと協力して実施することが多い高位レベルテストの戦略とを緊密に調和させ，全体のテスト戦略を最適化することが重要です。例えば，ユーザ視点のユーザインターフェースなどのテストを低位レベルテストに組み入れて，欠陥をできるだけ早期にできるだけ安く検出したり，低位レベルテストのテストケースと高位レベルテストのテストケースとが重複したりするのを避けたりといった事項が該当します。

ソフトウェアテスト見積りの精度(妥当性)を向上させるために，プロジェクトの見積り時点でテスト戦略を定め(決定できない事項はそのことを明示して)，ユーザとベンダが相互に合意したうえで，これをソフトウェアの開発サイクル前提条件として見積りにインプットすることが重要です。

以下，3.1～3.3節に記載した事例から，テスト戦略で考慮する事項をリストアップして整理します。

(1)　テスト技法の選択

テスト戦略の設定に関して重要な視点は，テスト技法の選択であり，テスト技法の選択に基づいてテストの目標を設定することになります。テスト技法は，テストベース(要件定義，基本設計，詳細設計など)からテストケースを引き出す構造化されたアプローチであり，特定の欠陥の検出を目的としています。テスト技法については，本ガイドブックの第2章および第4章の4.3節に紹介しています。テスト技法を利用することにより，アドホックにテストケースを設定するよりも遙かに効果的に欠陥を検出できるようになり，同じテスト対象システムのテストでも，選択するテスト技法によってテスト量(テストケース数など)は異なります。

成果物をいかに徹底的にテストするか否かという観点で，テスト技法を選択するとともに，品質目標値(テスト密度，検出欠陥密度，テスト網羅率など)として設定します。これは，テスト量(テストケース数など)の見積りに一番大きく関係します。

3.1～3.3節では，品質目標値に関して留意するべき事項が紹介されており，次に整理します。

・品質目標値(テスト密度，検出欠陥密度，テスト網羅率など)は，テスト対象システムの特質などを見きわめ(分析・評価)，その結果を見積りの入力情報にすることが必要です。なお，テスト対象システムの業務設計や処理方式設計などを配慮したテスト設計では，各自に設定した品質目標値に固守することなくテスト量(テストケース数など)を再見積りすることも必要です。
・システム全体に影響する重要機能は，障害によるリスクを整理して，テスト優先度およびテスト密度の差別化などを考慮しておくことが必要です。

(2) テストプロセスの設定

システムの品質を強化するために，高位レベルのテストプロセスを正しく管理すること，および報告とコミュニケーションによってプロジェクトの他の部分(設計・実装工程および低位レベルテスト)との調和をとることが重要です。

3.1～3.3節で紹介している事例でも現れていますが，経験的には，良いテストプロセスの重要性に対する認識は，低位レベルテストよりも高位レベルテストにおいて高いのが実情です。しかし，低位レベルテストのプロセスは高位レベルテストのプロセスと密接に関連しているので，高位レベルテストの視点から低位レベルテストの目標を定めて，低位レベルテストのテストプロセスを設計し，実施し，評価することが重要です。テストプロセスは，ソフトウェアテストの生産性の見積りに一番大きく関係します。

3.1～3.3節では，テストプロセス設計に関して留意するべき事項が紹介されており，次に整理します。また，事例で紹介しているもののほかに，テストプロセスを設計する視点として留意する事項2点(③および④)を挙げています。

① 非機能要件のテスト戦略

・非機能要件(性能など)のテスト設計では，現行処理状況(投入データ数と処理データ数との因果関係)も調査したうえで，性能測定の基準数値(最大処理件数)を決めます。なお，データ量の延び率を見積もるうえで，利用部門などの参加(レビュー含め)が必須です。
・期間の制約条件がある場合(テスト期間が短いなど)，データ制約条件がある場合(テストには整合性の取れた大量のデータが必要であるなど)は，設計・実装工程の品質作り込み(レビューなど)コストとテスト工程の確認テストとを配慮することが必要です。また，「早期のテストデータの作成」や「単体テストや統合テストなどの早い段階での大量データでの確認(DB間整合性を求めず)」といったタスクの検討を考慮するなどの工夫が必要です。

・非機能要件の確認テストは，「機能要件面の確認への傾注傾向」，「手間がかかる」などの理由から，開発の状況（予算超過状態など）に左右され，おろそかになることが多く見受けられます。テスト戦略においてその重要度を認識しておくことが必要です。
・対外接続テストなど外部（他社）とやり取りする確認テストは手間がかかり，かつ計画も簡単に変更できないといった制約があります。テスト戦略の中でひっ迫した状態であっても，「安易に計画変更してはならない作業」であることを特定しておくことが必要です。

②　テストプロセスの設計

・テスト設計において，テストケースの実施順序およびテスト実施手順を決めておくことは，データの重複性の排除，テスト実施の効率化，テスト環境の共有および結果確認の容易さなど，テスト効率の向上が期待できるので考慮しておく必要があります。
・テスト設計において，テストの重複を見きわめ単一データでの項目消化量を増加させることは，データ量を削減するなどテストの生産性の向上が期待できるので考慮しておく必要があります。
・プログラム欠陥はソースレビューのチェック方法を策定し，徹底することで削減できます。テスト見積り（プログラミング工程）では，ソースレビューと単体テストの両面で残存欠陥密度を見積もることも重要です。
・試験手順書作成の範囲および記述の深さなどは，テストプロセスの標準として，見積り時点で決めておくことが必要です。また，テスト戦略ではテストプロセス標準に対して，テスト対象システムへのテーラリング方針を策定し，見積り前提条件として明文化するなど，ユーザと合意しておくことが重要です。
・テスト設計において，データの処理順序を考慮してテストケースの実施順序を決めておくことは，データの重複性の排除，テスト実施の効率化，テスト環境の共有および結果確認の容易さなど，テスト効率の向上が期待できるので考慮しておく必要があります。

③　欠陥をその混入源に一番近いところで発見し，修正コストを最小化し，かつ，できるだけ早期にシステムの品質についてソフトウェア開発プロジェクトにフィードバックする必要があります。

・全体レビューやテストを高位レベルテストから低位レベルテストにシフトする。
・最も重要な欠陥をできるだけ早期に，できるだけ安く検出するために，レ

ビューの長所および弱点，テストの長所および弱点を特定して，レビューの弱点をテストで，テストの弱点をレビューで補完するように，レビューのレベルとテストのレベルを相互に調整する。

・テストファースト(10)を実践する。

④　設計・実装工程でできるだけ欠陥を減少させることに重点をおくべきです。

・検出した欠陥を開発者に早期にフィードバックする。

テストで検出した欠陥によりソフトウェアに内在する弱みの傾向を，早く設計および実装工程にフィードバックする。

・テストファーストを実践し，要求と設計を確認するために，テストシナリオを用いる。

設計や開発に先立って，ユースケースなどに基づきテストシナリオを作る。これらのテストシナリオは，後の段階でソフトウェアのテストに適用されるものでなければならない。

・テスト担当者のテスト能力を養い，テスト管理者，テスト技法スペシャリスト，技術者などの職務を持ったテストの専門家を養成し，適切にテスト組織に組み入れる。

・「テストできるか？」という質問に答えられるように試験性(11)に留意してシステムを設計，構築する。

---

(10)　アジャイル開発プロセスで用いられている手段ですが，設計または実装に先立って，テスト設計を行うことによって，これから設計またはプログラム実装するものが満たすべき条件を網羅的に洗い出すことができます。これは，設計者または開発者の誤解を防ぎ，設計の考慮漏れを予防し，かつ，効率的な静的テストを可能にして，より安価に欠陥を検出することに寄与します。

(11)　一般的には，「改訂したソフトウェアの妥当性を確認するのに必要な労力と関係あるソフトウェアの属性」と定義されており，ソフトウェア自体の他に，テストプロセスの組織，インフラストラクチャも関係します。ここでは，これに加えて次の解釈も「試験性」という意味に含みます。

・システムの機能および性能レベルをテストすることの容易性と速度
・ソフトウェアに対する要求をテストで実証または実測できる程度

要求の内，テストで実証または実測できないものは，プロセスへの要求として「評価」などの手段で担保することになります。テスト戦略では，試験性を十分に分析し，レビュー戦略はもとより，欠陥の混入を予防する手段も含めて，検討して定義することが重要です。

（3） テスト環境

効果的なテストツールの導入を検討(費用対効果)しておくことは，テスト実施工程でのテスト生産性に影響するため重要です。また，テスト中に生ずる欠陥の修正に伴う再テスト，仕様変更に伴うリグレッションテスト，保守などの改良開発におけるリグレッションテストの生産性を高めるために，リグレッションテストリストやテストデータを整備して，維持しておくことも，テスト量(テストデータ量，テストドキュメント量など)に影響するため重要です。

3.1～3.3節では，テスト環境に関して留意するべき事項が紹介されており，次に整理します。

・自動化テストツール，現行システムとの結果照合ツールなど，効果的なテストツールの導入を検討(費用対効果)しておくことは，テスト実施工程でのテスト生産性に影響するため，テスト戦略として定めて，テスト見積りのインプット条件として必要です。ただし，テストツールの適用に際して次の事項を考慮して，ツールの利用効果を判断することが重要です。
　－自動化テストツールは万能でないこと
　－時間の経過や開発環境の変化などで，効果が減少する可能性がある(既存のツールを適用する場合は，ツールの保守状況を十分に調査する)
　－新たにテストツールを開発する場合は，信頼性の高いツールを他のテスト実施前に完成させるようスケジュールを立てる必要がある
・テスト環境を事前検証することは，対象機能と確認内容，テスト環境課題の確認およびテストプロセスの試行を実施した場合の費用対効果を鑑み，テスト戦略として計画しておくことが大切で，その評価を得てテスト見積りに反映するなどの工夫が重要です。
・テストデータの保守(最新化)は一時的にコスト高になるが，ライフサイクルコストからのテストコストを評価すると安くなる傾向があるので考慮する必要があります。

（4） テスト分担

テストに関わる役割分担は，テスト戦略で定め，ユーザとベンダとで相互に合意しておくことが重要です。特に，業務ナレッジを必要とするテストデータのバリエーションの設定やユーザ視点でみた操作・利用環境の洗い出しなどは，ユーザが担当するのが効果的です。一方，論理的にテストケースを設計するのはベンダが得意とするところです。このように，ユーザとベンダとは相互に協力してこ

そ，テストを効果的に効率的に実施し，ソフトウェアの品質を強化できます。

3.1～3.3節では，テスト環境に関して留意するべき事項が紹介されており，次に整理します。

・テスト設計への第三者の介入方針(開発当事者以外のテスト設計担当/テスト設計レビュー参加など)は，対象システムの特質や開発量などを鑑みて，テスト戦略において設定することが重要です。なお，テスト見積りの入力となるテストの網羅性や残存欠陥密度は，担当者能力の偏在など，極力，変動幅を小さくする工夫を行うことが前提です。
・ユーザインターフェースなど妥当性確認テストを対象にしたテストケースは，内部構造にとらわれない利用者側の発想を持ったテスト要員の参加も重要であり，妥当性確認テストにおいてテスト量の見積りを行う際には，考慮することが必要です。

(5) テスト進行中における品質管理と再テスト見積り

テストプロセス全体の品質と進捗を測定し，測定結果をテストプロセスの制御やテストプロセス改善のインプットとすることは，システムの品質強化のために重要であり，テストのコストとして見積りに組み入れる必要があります。

3.1～3.3節では，テスト進行中における品質管理と再テスト見積りに関して留意するべき事項が紹介されており，次に整理します。

・設計・実装工程(基本設計～コーディング)での検出欠陥密度(レビューなど)，テスト工程でのテスト密度および検出欠陥密度(確認テスト)，さらに本番稼働後の欠陥密度の相関分析を行うことにより，プロセス改善点を発見できるとともに品質指標値の精度を高めることができます。
・品質目標値はテスト見積りとテスト進行中における品質管理(予実管理)のインプットになります。事例では，品質マップを作成し欠陥偏在分析を行うことにより，品質目標値の再評価(再設定)を行い，さらに是正処置として再テスト量を見積り(追加テスト)，品質を担保しています。
・設計不良はテスト戦略において設計工程(レビューなど)での残存欠陥密度を予測して，確認テストでのテスト量を見積もります。しかし，テスト工程において設計不良が予測値より多く検出された場合は，設計工程の見直しを早期(この例では中間評価プロセス時)に行う必要があります。
・信頼度成長曲線を用いて，システム全体のグロス(コンポーネント単位などではなく)管理だけで欠陥の収束度を管理するのは危険です。また，ソフトウェ

アテストの特質として，特定のソフトウェアコンポーネント(ある機能を有するプログラム群)の品質が劣悪であると，全体のテスト効率低下となる傾向にあります。テスト見積りではこのような生産性変動要因に関わるリスクをヘッジする配慮も必要です。

・テスト戦略(計画)としてのテスト完了基準は，ユーザとベンダの相互同意が必要です。テスト完了基準は定量的なテストの網羅性(残存欠陥密度)だけではなく，欠陥の修正状況および検出した欠陥の発生傾向などの定性的な基準策定を考慮することが必要です。また，テスト進行中や完了時点の品質目標値の評価・分析は，テスト見積り精度の向上を目指すためにも必要なことです。

(6)　見積り変動要因の識別および対策の設定

以上のほかに，ソフトウェアテスト全般に関わるテスト量およびテスト生産性に関する変動要因があり，これを識別し評価して，対応策を設定しておくことが重要です。

以下，3.1～3.3節では，見積り変動要因の識別および対策の設定に関して留意するべき事項が紹介されており，次に整理します。

・システム再構築など母体のあるシステムの開発では，既存システムや現行システムの母体品質を確認し，テストの見積り変動要因として配慮しておく必要があります。また，既存システムの品質が思わしくない場合には，既存システムの品質を確保する作業が必要になるので，見積りに反映することが必要です。さらに，既存システムの品質を確保する作業は，改良作業の着手前に実施することが望ましいです(改良案件と同時に確認を行う場合，既存欠陥との識別ができず，混乱する場合があります)。

・テスト対象コンポーネント(ある機能を有するプログラム群)の複雑度などによる生産性の重み付けの調整およびテスト終盤での難バグ(テスト検出や解析時間がかかる)を考慮するとともに，各テスト工程の欠陥管理を個人任せでなく，組織で管理する体制が必要になります。また，テスト見積りの変動リスクを抑制するためにも，欠陥の解析・管理(ツール導入などの工夫)を徹底すべきです。

・外部接続が多い統合テストやシステムテストなどでは，他ベンダが開発した部分の欠陥による再テストのリスクをテスト戦略の中で整理し，再テストは他責としてテスト見積り前提条件でユーザとベンダが相互に合意すべきです。なお，契約方法の工夫や定量的な外部システムの品質状況を把握するなどの手段も必要です。

・開発全体の見積り合意項目として，仕様変更タイミング，仕様変更量および変更認定基準などをユーザとすり合わせておくことが必要です。また，仕様変更はシステムテスト工程で仕様変更を組み入れることが一番のコスト高となること(棄却工数の増加)および品質劣化(デグレード)を招きやすいことなどの特質があります。このような仕様変更の特質をユーザとベンダの相互に理解することが重要です。
・保守などの改良開発の場合，改造分散度，テスト対象範囲および既存母体状況など，改良開発特有の不確定要素があることから，影響調査を怠ると障害につながる可能性が高いため，調査・分析作業は必須であり，段階的に見積りの見直しを行うことも考慮する必要があります。
・ソフトウェア部品を自動生成するツールを同時並行に開発している場合は，生成部品の品質(プログラム欠陥，インターフェース変更，利用マニュアル不備など)によるテスト生産性の低下を考慮する必要があります。テスト見積りを行う際には，提供生成部品の品質などに関し，前提条件として明文化するなどの配慮が必要です。

第4章　ソフトウェアテスト見積りの詳細

4.1　ソフトウェアテスト見積りの手順

ソフトウェアテストの見積りにおける重要な要素の1つとして，開発するソフトウェアに対する機能要求が明確な場合であっても，ソフトウェアに対する要求を理解し整理しておくことが挙げられます。ソフトウェア開発規模の増加とともに，ソフトウェア機能および実行タイミングは複雑化してきており，動作環境も考慮して，あらゆる条件を組み合わせたテストを行うと，テスト量(テストケース数など)は，天文学的な数字になります。しかし，現実的にはコストおよび開発期間などの制約により，合理的なテスト量の削減が必要になります。つまり，テスト量によるテストコストと，その残存欠陥がソフトウェアの品質リスクを許容範囲内に抑えるというトレードオフを考慮したテスト見積りが必要になってきます。ここでは，そのようなテスト見積りについて説明します。

個々のシステム開発のテスト見積りは，**図4.1**に示すとおり「要件の洗い出し→開発規模の見積り→テスト量の見積り→テスト生産性の見積り→テスト工数/コストの見積り→テスト工期の見積り」といった手順で行うことになります。なお，テスト見積りにおいては，「品質要件→品質指標・目標値→テスト量」という流れがより重要となります。

4.2　テスト量と品質目標値

本節ではテスト見積りの基準要素であるテスト量(テストケース数など)の算出に関する一般事項および考え方に焦点をあてて説明しています。

4.2.1　一般的事項

(1)　テスト戦略に基づく品質目標値

テスト戦略にはソフトウェア機能に対応した品質保証の重要度・優先度，テスト完了基準，テスト技法の取捨選択，テストプロセスの策定，テスト環境の決定およびテスト分担の決定などがあります。品質目標値(残存欠陥密度，工程ごと

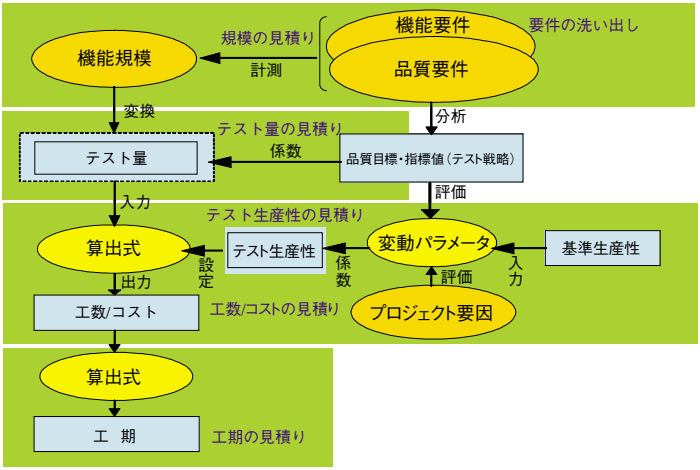


図 4.1　テスト見積りの基本手順のブレークダウン

の検出欠陥密度，テスト完了基準，テスト網羅率など品質指標に基づいて設定）は，これらのテスト戦略策定を通して設定されます。特に，テスト量に関わる品質指標の中では，残存欠陥密度（代替としてテスト網羅率）とテスト密度が，もっとも重要な品質指標になります。品質指標については，4.2.2～4.5.5項に記載します。

（2）　改良開発におけるテスト量

改良開発の見積り方法は，新規開発の見積り方法を包含していますが，改良開発のテスト量は，新規開発の場合と同じように，単純に改良ステップ数に比例することを期待してコストを見積もると失敗することがありますので注意が必要です。改良開発のテスト量は，既存システムに対する改良箇所の分散度合いによる巻き込みテスト量を加味して見積もる必要があります。このことはSEC BOOKS『ソフトウェア改良開発見積りガイドブック』で紹介しています。改良開発特有のテスト量に影響する変動要因を**表 4.1**に示します。なお，新規開発および改良開発に共通するテスト量へ影響する変動要因は，4.6節で紹介します。

4.2.2　残存欠陥密度の設定

（1）　出荷時の残存欠陥密度（または単位量あたりの残存欠陥件数）の確認

出荷時の残存欠陥数を出荷時に測ることはできないため，出荷後に発見した欠

表 4.1　改良開発に特有の機能実現に対するテスト量の変動要因

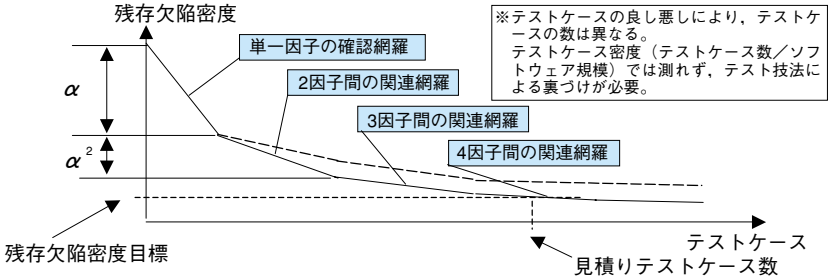
主特性	副特性	影響の理由と評価の観点
変更箇所の結合度	―	モジュール間インターフェースの方式(ファイル連動，パラメータ連動，共有領域連動，DB連動，転送連動，マクロ)によってモジュール間結合が異なり，調査作業およびテスト量に影響する。
規　模	改良開発の規模	改良開発の規模が多いとテスト対象が広がるため，テスト作業が増大する。
	既存システムの規模	既存システムの規模は，テスト対象の規模に影響を及ぼすことから，テスト作業を増大させる。
	巻き込み規模	改良要求の確認やレベルダウンしていないことの確認のために，テスト巻き込み規模が，改良開発におけるテスト量(テストケースなど)を左右する。

陥数を測定することになります。しかし，ソフトウェアテストの見積りでは，ユーザと出荷時の残存欠陥密度を合意して見積もる必要があります。現実には，ソフトウェアテストをどこまで実施するかということをユーザと合意することによって，出荷時の残存欠陥密度の合意に代えることになります。

ソフトウェアテストの手厚さはテスト網羅率で設定しますが，ここでは，組み合わせの網羅と，テストケース数との関係を考えます。

品質工学の手法であるタグチメソッドを発案された田口玄一博士は，ソフトウェアテストへの直交表の適用にあたり，設計・実装工程で作り込まれる，単一の因子(例えば，１つの入力項目)に対するソフトウェアの動作に関して欠陥が存在する確率をαとすると，２つの因子の組み合わせに関して欠陥が存在する確率はαの２乗，N個の因子の組み合わせに関して欠陥が存在する確率はαのN乗になると仮定しています。

一方，因子の組み合わせを網羅するためのテスト量は，組み合わせを網羅する因子の数が増加するにつれ幾何級数的に増加するのに対して，存在する欠陥の確率はN乗(Nは組み合わせる因子の数)で減少していきます。この様子を**図 4.2**に示します。やたらにテストケースを追加しても網羅性は必ずしも上がりません。例えば，組み合わせの網羅については，ペア構成テストのようなテスト技法を用いてテスト設計を行う必要があります。



※：ソフトウェア構成品の独立性が低いと，破線のように残存欠陥密度は高くなる。

図 4.2　残存欠陥密度とテストケース数との関連概念

ソフトウェアをテストする観点は，2.3節に記載のとおり，入力の組み合わせのほか，異常値テスト，状態遷移テスト，ユースケーステストなど，さまざまあります。

それぞれの観点でテストの網羅性を検討し，適用するテスト技法を定めて，ユーザとテスト網羅率の目標を合意します。

実際のプロジェクトで，出荷後に発見された欠陥を測定して，テスト網羅率と残存欠陥密度との相関を分析し，適用するテスト技法およびテスト網羅率にフィードバックしていきます。

（2）　欠陥の顕在化リスクとテスト量のトレードオフ

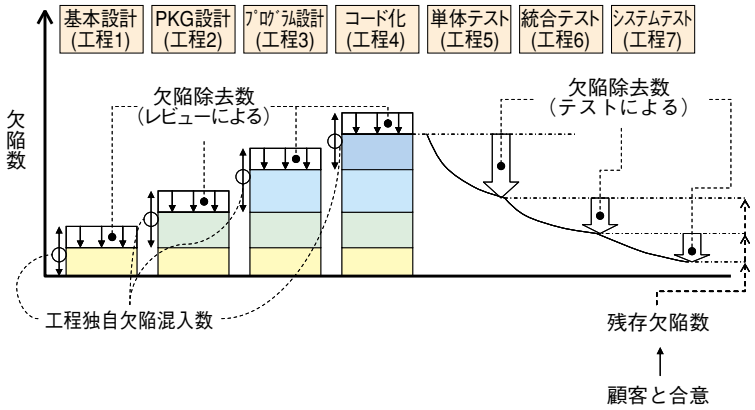
限られたテスト予算とテスト期間の制約のうえで，どこまでテストを手厚く行うかは，テスト対象機能において本番稼働後に欠陥が顕在化したときの社会的影響や復旧にかかるコストなどによって判断する必要があります。したがって，出荷時の残存欠陥は，一律に設定するのではなく，テスト対象ごとにテストの手厚さの度合いを加味することが重要です。既に，このことは第3章で紹介してきたとおりです。

4.2.3　レビューおよびテストでの欠陥検出戦略の統合

ソフトウェアに混入する欠陥の総量を見積り，レビューを含めて各開発工程で欠陥を検出する数を想定することが重要です。

ソフトウェア製品の欠陥は信頼度成長曲線などを用いて分析すると，設計・実装工程から品質を作り込んでいれば，テスト工程で検出すべき欠陥は少なく，テスト工程でのコストも少なくなる傾向があります。したがって，設計・実装工程

のレビューとテスト工程の品質確認テストの両面で欠陥数を測定して管理すべきで，テスト実施工程のみを対象にした過去の統計値(信頼度成長曲線などの品質目標値)を基にテスト見積りを行う場合は，そのことを留意する必要があります。なお，設計・実装工程とテスト工程での欠陥除去数と残存欠陥数との関係を**図4.3**に示します。



※：レビューでは当該工程で作り込んだ欠陥の除去だけでなく，前工程で作り込んだ欠陥を除去することもある。

（出典）　ジャステック社「ソフトウェアテスト見積り－欠陥の混入および除去モデル－」

図4.3　設計・実装工程とテスト工程での欠陥除去数と残存欠陥数との関係

ソフトウェアの欠陥は，それぞれの設計・実装工程で作り込まれ(レビューにより一部が除去されますが)，作り込まれた欠陥は次の工程に引き継がれます。単体テスト前のプログラムには，基本設計工程～コード化の各工程で作り込まれた欠陥が集積されています。

テストでは，設計・実装工程でプログラムに作り込まれた欠陥を除去していきますが，すべての欠陥を除去することはできないので欠陥が残ります(図4.3では，これを残存欠陥数と呼んでいます)。

それぞれの設計・実装工程で作り込む欠陥の密度およびレビューなどで除去できる欠陥の割合ならびに各テスト工程で除去できる欠陥の割合は，出荷後に検出される欠陥も含めて測定して蓄積し，基準値として精錬しておくことが重要です。

コラム：PSPでの欠陥除去の考え方

ここでは，カーネギーメロン大学ソフトウェアエンジニアリング研究所（SEI）で発表されたPSP（パーソナルソフトウェアプロセス）で用いている欠陥除去の考え方を紹介します。

欠陥除去（レビュー，コンパイル，テストなど）をフィルタとして考え，あるフィルタに欠陥を入れると，一定の割合で欠陥が除去されるという考え方をしています。例えば，単体テストでは，製品に存在する欠陥の内の45％を除去できると仮定しています。

① 用いる尺度

工程欠陥検出率：

100×（工程中の除去欠陥数）/（工程入力時の製品の欠陥数）

（例えば，テスト入力時に100個の欠陥が含まれていて，テストで45個の欠陥が検出された場合は，45％となる）

② 検出欠陥数および残存欠陥数の算出：

検出欠陥数＝工程入力時の欠陥数×工程欠陥検出率

工程完了後の欠陥数＝工程入力時の欠陥数×（１－工程欠陥検出率）

工程欠陥検出率はレビューやインスペクションでは比較的高く（50～80％），テスト工程は相対的に低い（40～50％）といわれており［Humphrey］，欠陥の少ないソフトウェアを作るには，テスト開始時にできるだけ欠陥を少なくしておくことが重要であり，設計レビューも含めて欠陥除去戦略を設定しておく必要があります。また，欠陥の修正に伴う再テストの回数も増加しますから，テストの生産性を上げるためにも，設計・実装工程でできるだけ欠陥を少なくしておくことが重要です。

テストの戦略として，設計・実装工程における欠陥除去戦略も含めることが望まれます。

このような考え方を前提にすると，レビューやテストで検出した欠陥数から，レビューやテストの後に残っている欠陥数を予測することができます。

例えば，あるテスト工程で50個の欠陥を除去したとします。そのテスト工程の工程欠陥検出率が50％だとすると，50個の欠陥は，テスト前の製品の欠陥数の半分ですから，テスト完了後にはまだ50個の欠陥が残っていると考えられます。

ソフトウェアの開発中に検出する欠陥はもとより，出荷後に検出される欠陥を含めて測定して蓄積し，工程欠陥検出率の精度を高めていくことにより，ある工程完了時に残っている欠陥数を一定の精度以上で予測することができる可能性があります。

4.2.4 ソフトウェアのテスト完了基準

顧客からの品質要求が高ければ(テスト完了基準が厳しければ)コストは高くなります。また,品質要求仕様があいまいであれば,テスト見積りの精度は悪くなります。

テスト戦略の一環としてテスト完了基準を明らかにしておくことは重要です。ここでは，テスト完了基準を目標とする残存欠陥密度を実現するために必要なテストの網羅性(テスト網羅率)に焦点をあてて，ソフトウェアテスト量の見積りについて言及します。

なお，ソフトウェアのテスト完了基準は，テスト網羅率を測定して定量的に判断する方法以外に，信頼度成長曲線の収束度合い，仕様変更の収束率，修正未了の欠陥数，未解決・懸案件数および定性的な検出欠陥の発生傾向(リスク許容評価)など，総合的な判断が必要になります。

次に，テスト完了基準で定める項目の事例を列挙しますが，テスト完了基準を満足しない場合は，追加テストなどの是正処置をとることになり，結果としてテスト量が増加します。

① 体系的なテスト設計に基づくテスト網羅性の担保
- 要件とテスト項目の対比による要件網羅
- 体系的なテスト技法の適用によるテスト網羅性の担保
- テスト網羅性とテスト量との相関関係に基づくテスト密度の担保

② テスト実施状況に基づく判定
- 設定テスト量完了，未実施テストのリスク管理，カバレッジとテスト量が過去実績から満足など

③ 欠陥の発生状況に基づく判定
- 過去実績に基づく目標と対比した実績欠陥量，欠陥密度とテスト密度との相関(基準内，少数のテスト量で多数の欠陥を検出，逆に欠陥の検出量が稀少)，検出欠陥の発生傾向分析(欠陥の偏在などを考慮して，リスクの許容範囲を評価)，信頼度成長曲線の収束度合い(収束傾向にある，残存欠陥を重要度ごとに評価・合意)

④ 欠陥の修正状況および再発防止策の実施状況
- 検出欠陥をすべて修正済，優先度を関係者で合意，デグレ発生状況と未対応類推とリスク許容評価，開発プロセスへのフィードバック状況(同種欠陥の混入を防ぐために開発プロセスにフィードバックしたものの積み残しなし)

4.2.5　テストの網羅性とソフトウェアテスト量

ソフトウェアの動作に影響するさまざまな入力などの組み合わせテストを例にとれば，すべての機能の組み合わせを完全(100%)に網羅すること，およびレビューやテスト完了後の残存欠陥密度を演繹的に設定することは，現実的には限界があります。

しかし，ソフトウェアテスト量を見積もるには，見積り前提条件として，目標とする残存欠陥密度およびそれらを実現するに足るテスト網羅率とテスト密度とが必要となります。

テスト網羅率のとらえ方は，ホワイトボックステストとブラックボックステストとで異なるので，残存欠陥密度とテスト網羅率からソフトウェアテスト量を見積もる場合は，ホワイトボックステストとブラックボックステストに分けて見積もることが肝要です。また，欠陥の影響度などを考慮して重要度と欠陥修正の優先度をシステム機能ごとに評価してテスト網羅率を設定し，ユーザとベンダ相互に合意しておくことが必要です。

4.3　テスト網羅性尺度とテスト量見積り方法

4.3.1　ホワイトボックステストでの網羅性の尺度とテスト量見積り方法

命令網羅(C0)，分岐網羅(C1)，条件網羅(C2)カバレッジのように，ソースコードに対する客観的なテストの網羅性の尺度があり，測定可能になっています。

テスト量を見積もるうえで，C0，C1，C2のカバレッジ率(テスト網羅率)は，演繹的に算出可能であるので，ユーザとベンダ相互にカバレッジ率を合意することが必要です。

次にテスト量の算出方法に関する事例を紹介します。開発規模(図4.1での機能規模)から,基準テスト密度(過去実績から統計的に得られたテスト密度の基準値)およびテスト網羅率(テスト戦略の策定を通して設定したC0，C1，C2のカバレッジ率)を使用してテスト量を求めることができます。

例えば，第2部に示すジャステックの事例から，あるテスト工程iのテスト量$_i$は，以下の式で求めることができます。

$$テスト量_i = 開発規模 \times 基準テスト密度_i \times (1 + \alpha)$$

※　α：基準テスト密度$_i$でのカバレッジ率とシステムごとの顧客要求(テスト戦略)カバレッジ率との差分によるテスト密度の変動率

コラム：ホワイトボックステストでの網羅率　C0，C1，C2

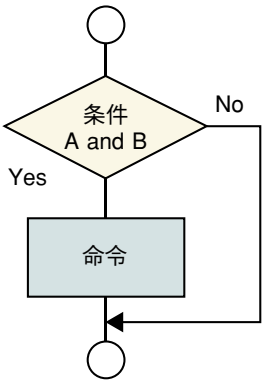
ホワイトボックステストでのテスト網羅率とは，ソースプログラムのルートのうち，どのくらいテストを達成したかを示します。ここでは，よく使用されているC0，C1，C2に関して説明します。

命令網羅率(C0)：プログラムの命令を最低1回はテストする。

分岐網羅率(C1)：すべての命令とすべての分岐をテストする。

条件網羅率(C2)：条件分岐において条件の真・偽のすべての組み合わせをテストする。

テストの網羅性からいえば，条件網羅率(C2)が一番広く網羅されています。その分テスト量は増大します。参考に，条件分岐ロジックに対し前述の網羅率別で実施すべきテストケース(網羅テスト別)を示します。



【分岐条件にてありえる組み合わせ】

ケース	A	B	A and B
No.1	真	真	真(Yes)
No.2	真	偽	偽(No)
No.3	偽	真	偽(No)
No.4	偽	偽	偽(No)

【各網羅率に対し実施すべきテストケース】

対象網羅率	実施すべきケース
命令網羅率	No.1
分岐網羅率	No.1，No.2～4のいずれか1つ
条件網羅率 ＊1	No.1，No.2，No.3，No.4

＊1：すべての条件を組み合わせた場合

図4.4　条件分岐のテストケース例

通常ホワイトボックステストでは，命令網羅率(C0)，分岐網羅率(C1)が100%になることを目指します。しかし，これだけではプログラムが異常終了しないことを示すだけなので，複雑な条件分岐に対し「条件の真・偽の組み合わせテスト」を用意し，部分的に条件網羅率(C2)を100%にすることが良いと考えられています。

4.3.2　ブラックボックステストでの網羅性の尺度とテスト量見積り方法

（1）　ブラックボックステストでの網羅性の定義

ユースケース網羅，状態遷移網羅，機能の組み合わせ網羅など，システムを受け入れる立場のユーザ視点からみたテストの最適化とテスト量の見積り方法について考察します。

あるシステムをユーザの立場でテストする場合におけるテストの手厚さに関する尺度「テスト網羅率」を以下のように定義します。

テスト網羅率＝(実際にテストするテスト項目の数
÷抽出されうるすべてのテスト項目の数)×100

ここでテスト項目をどれくらいの粒度で数えるか，ということが問題になります。単純な入力のテストにおいても，入力項目値の組み合わせのバリエーションをすべてテストしようとすると膨大なデータ量になってしまいます。例えば，3桁の整数値を2つ入力してその加算結果を表示するという単純な機能でも，すべての入力値の組み合わせは1 000×1 000＝1 000 000とおりになってしまいます。つまり，入力項目値のすべての組み合わせをテスト項目ととらえ，テストの網羅性を議論するのは現実的でないことがわかります。

本書ではテスト項目の粒度を，各種システムドキュメント(要件定義書，設計書あるいはマニュアル類など)からブラックボックス的観点により抽出可能な，「テストで確認すべき内容を文章表現した」レベルととらえ，原則的に100%のテスト網羅率を目指します。入力項目値の組み合わせのバリエーションについては，入力項目がとりうるすべての値や入力項目間のすべての組み合わせをテストするのではなく，欠陥検出の効果を損なうことなく，合理的にテストを省略するにはどうしたらよいのか，という観点で考察します。

（2）　テスト項目の抽出方法

システムを受け入れるユーザの立場に立った，テスト項目の抽出方法(ブラックボックステスト)を考察します。

(a)　要件定義からの抽出方法

要件定義書に記述された機能要件および品質要件からテスト項目を抽出する。この際，入力項目のとりうる値の範囲や出力結果のバリエーションを明確にしておく。

(b)　設計書からの抽出方法

機能仕様，メニュー構成図，状態遷移図あるいはユースケース図などからテスト項目を抽出する。この際，入力データ項目のとりうる値の範囲や出力結果のバリエーションを明確にしておく。

(c)　マニュアル類からの抽出方法

運用マニュアルや操作手引書などからテスト項目を抽出する。特に，通常は想定できないような異常な操作や，異常なデータに対する処理を確認する項目も抽出する。

(d)　テスト項目の重複の排除

上記(a)～(c)で抽出したテスト項目には重複するものがあるので，その重複を取り除く。

また，ホワイトボックステストで抽出したテスト項目と重複するものを必要最小限にとどめる。

(e)　過去の障害およびクリティカルな部分に関するテストの追加

特に，改良(保守)開発においては，過去の経験に基づき，障害事例の再発防止およびシステム上クリティカルな部分の品質を担保するために，適切にテストケースを追加する。

(3)　グレーボックス的な観点によるテスト量の削減

プログラムの構成・ロジックを意識し合理的にテストケースを削減することも有効です。例えば，共通の部品を使っている，あるいは汎用的に確立された部品を使用している部分については，テストケースの削減が効果的です。

4.3.3　合理的なテスト量の削減方法

(1)　単一項目への入力項目値設定の工夫による合理的なテスト省略方法

連続する値，あるいは複数の値を，取り得る入力項目に対するテスト量の増加を回避するための工夫について述べます。

①　同値クラステスト

入力項目が連続する領域を持つ場合，入力領域を同値クラス(同様の出力結果が得られる部分集合)に分け，それぞれの同値クラスの代表値1つをテストします。

例えば，有効な値の範囲が整数5～8である場合，同値クラスは次のように3つに分けることができます(**図 4.5**)。

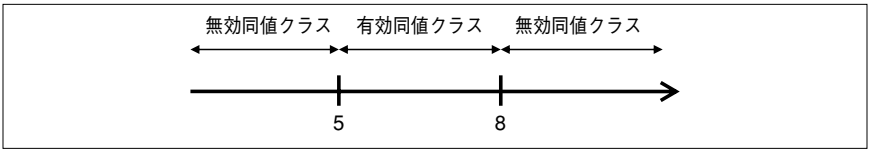


図 4.5　同値クラスの例

このとき，テストケースは次のような 3 つのケースが考えられます(**図 4.6**)。

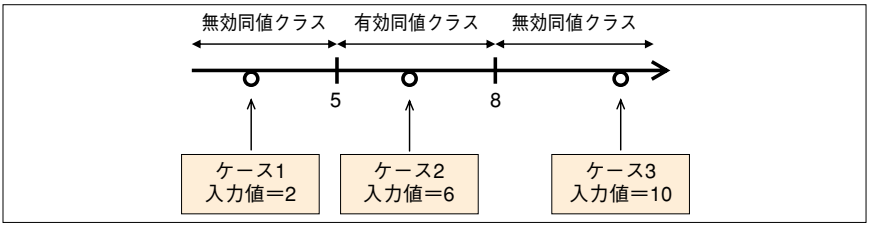


図 4.6　同値クラスのテストケース例

②　境界値テスト

入力項目が連続する領域を持つ場合，入力領域を同値クラスに分け，それぞれの同値クラスの境界値をテストします。

例えば，有効な値の範囲が整数 5 ～ 8 である場合，同値クラスは次のように 3 つに分けることができます(**図 4.7**)。

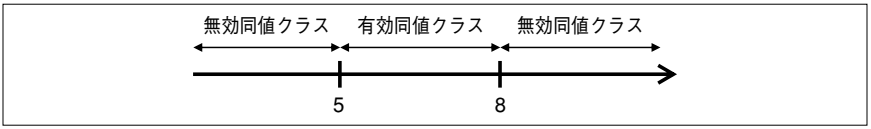


図 4.7　同値クラスの例

このとき，テストケースは次のようなケースが考えられます(**図 4.8**)。

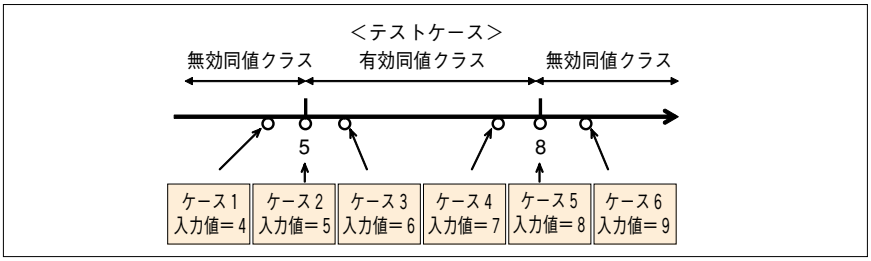


図 4.8　境界値テストのテストケース例

③　ドメイン分析テスト

相互作用のある複数の入力項目を，同値クラステスト/境界値テストの観点でテストします。同値クラステスト，境界値テストの入力項目は1つですが，ドメイン分析テストは複数の入力項目を考慮します。

例えば，入力項目A(整数値)，入力項目B(整数値)があるとき，$10 \leqq A \leqq 100$かつ$20 \leqq B \leqq 50$の領域ABが有効範囲である場合，それぞれの境界値の組み合わせにより，以下の20のケースが考えられます(**図 4.9**)。

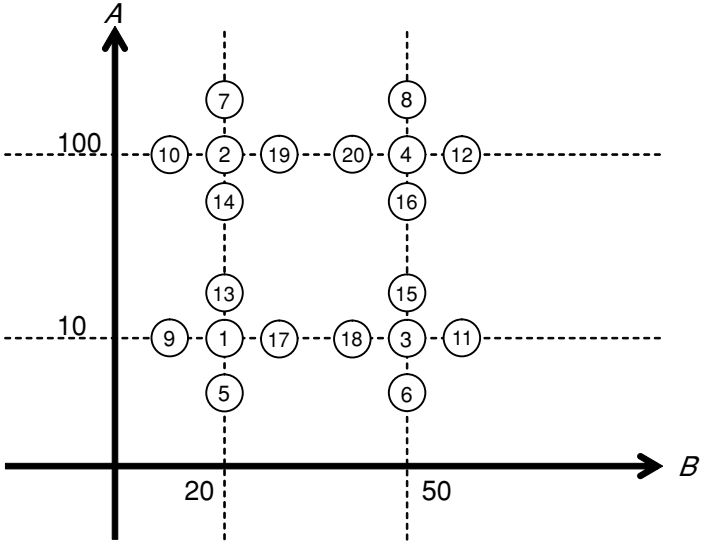


図 4.9　ドメイン分析テストのテストケース例

(2)　組み合わせ網羅に関する合理的なテスト省略方法

複数の項目や機能の組み合わせの網羅により，テスト量が爆発的に増加することを回避し，かつ，数学的に一律の深さで組み合わせを抽出する工夫について述べます。

①　ペア構成テスト(オールペア法)

複数の入力項目があるとき，任意の2項目間の入力値の全組み合わせを網羅するテストを行います。ただし，テストケースの割り付けに際しては，可能な限り同じ組み合わせが現れないようにするため，後述の直交表よりもケース数は少なくなります。

また，禁則処理を行うことが可能ですが，任意の 3 項目間の組み合わせ網羅率は直交表より低くなります。

表 4. 2は 4 つの入力項目がそれぞれ 3 つの値を取り得る場合のテストケースです。以下の 9 つのケースが考えられます。

表 4. 2　オールペア法のテストケース例

	A	B	C	D
ケース 1	1	1	1	1
ケース 2	1	2	2	2
ケース 3	1	3	3	3
ケース 4	2	1	2	3
ケース 5	2	2	3	1
ケース 6	2	3	1	2
ケース 7	3	1	3	2
ケース 8	3	2	1	3
ケース 9	3	3	2	1

②　ペア構成テスト（直交表）

オールペア法と同じく，複数の入力項目があるとき，任意の 2 項目間の入力値の全組み合わせを網羅するテストを行います。直交表の性質上，任意の 2 項目について同じ値のペアが同一回数現れるため，オールペア法に比べてケース数は多くなります。

また，禁則処理を行うことはできませんが，任意の 3 項目間の組み合わせについてもある程度，網羅されるのが特徴です。

表 4. 3は 7 つの入力項目がそれぞれ 2 つの値を取り得る場合のテストケースです。直交表では 8 つのケースが考えられます。

表 4.3　直交表のテストケース例

	A	B	C	D	E	F	G
ケース1	1	1	1	1	1	1	1
ケース2	1	1	1	2	2	2	2
ケース3	1	2	2	1	1	2	2
ケース4	1	2	2	2	2	1	1
ケース5	2	1	2	1	2	1	2
ケース6	2	1	2	2	1	2	1
ケース7	2	2	1	1	2	2	1
ケース8	2	2	1	2	1	1	2

③　ホワイトボックス分析併用テスト

ブラックボックス的観点から考察すると多数のバリエーションのテストが必要となるテスト項目(例えば，不連続なコード値の妥当性チェックなど)について，ホワイトボックス的分析(例えば，コードのチェックはテーブル化されているなど)を併用することによりケース数を削減します。

4.4　仕様変更量とテスト量

仕様変更は設計・実装工程(基本設計～コーディング)からテスト工程まで，コストおよび期間に影響します。また，同じ機能の仕様変更を設計・実装工程，テスト工程と後工程で対応すればするほど，コスト高(棄却量の増加などによる)となり，さらに仕様変更を加えることによりソフトウェア品質の劣化機会(デグレード)が増加します。

ここでは，なぜ，そのようなことが起こるのかを，仕様変更見積りに関する紹介を通して説明し，さらにソフトウェアテスト見積り(契約)時での留意点を示します。

なお，仕様変更見積りに関しては，ジャステック社の仕様変更見積り手法に基づいて説明しています。

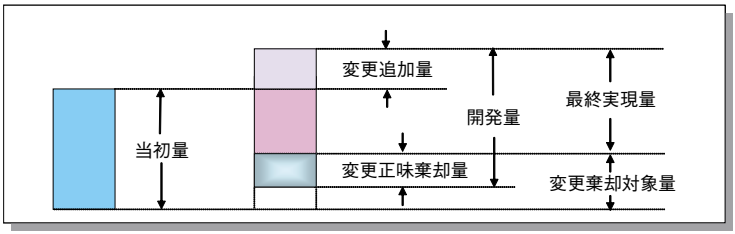
4.4.1　仕様変更量と開発量(テスト量)との関係

(1)　仕様変更量について

「仕様変更量」とは何か，について述べます。**図 4.10**に「仕様変更量の基本構造」を示します。

ここで,「当初量」とは仕様変更がまったくなかった場合の見積り量であり,「変更棄却対象量」とは，仕様変更によって開発不要となった部分の量です。その量のうち，まだ開発していない部分の量を除いた量が「変更正味棄却量」となります。また,「変更追加量」とは仕様変更により追加となるであろう量を示します。開発完了時に納品される量が「最終実現量」であり， 開発コストに比例するのが「開発量」となります。

なお，仕様変更の認定(合意)はユーザ担当者およびベンダ担当者の個人レベルではなく，ユーザ企業およびベンダ企業レベルとしてとらえることが大切です。



□ 基本ドキュメント*1　(*1:基本設計書、パッケージ設計書、プログラム設計書、ソースコード)

開発量＝当初量－変更棄却対象量＋変更追加量＋変更正味棄却量

最終実現規模

□ 変更棄却対象率＝変更棄却対象量÷当初量
□ 変更追加率＝変更追加量÷当初量
□ 変更棄却正味率＝変更正味棄却量÷当初量

□ テストドキュメント&テスト実施*2　(*2:テスト計画書、テスト仕様書、テストデータ)

開発量＝当初量－変更棄却対象量＋変更テスト追加量*3＋変更正味棄却量

(*3:変更テスト追加量＝変更追加量＋テスト巻き込み追加量)

図 4.10　仕様変更量の基本構造

（2） 仕様変更に関する課題

仕様変更の特質として注目すべきは，図4.10から理解できるように変更正味棄却量です。この量は仕様変更の発生するタイミングによって，同じ仕様変更でも大きく変動します。

表4.4は同じ仕様変更が基本設計完了時，単体テスト完了時，システムテスト完了時に発生した場合の変更追加量と変更正味棄却量を示した事例です。

仕様変更が基本設計時点で発生すれば，変更により不要となり棄却するのは基本設計書の該当部分だけですが，システムテスト時点で発生すれば，基本設計書からシステムテストまでのすべての工程の該当生産物を棄却することとなります。

一般的に，変更正味棄却量は開発量としての認識が希薄です。しかし，生産物量と生産性を基準にした見積りモデルでは，変更正味棄却量を開発量としてとらえ，ユーザとベンダ相互に合意することが前提になります。

表4.4　仕様変更タイミングと変更正味棄却量

変更タイミング / 工程	基本設計完了時			単体テスト完了時			システムテスト完了時		
	変更追加量	変更正味棄却量	合計（変更量）	変更追加量	変更正味棄却量	合計（変更量）	変更追加量	変更正味棄却量	合計（変更量）
基本設計	20KC	10KC	30KC	20KC	10KC	30KC	20KC	10KC	30KC
パッケージ設計	50KC	—	50KC	50KC	25KC	75KC	50KC	25KC	75KC
プログラム設計	100KC	—	100KC	100KC	50KC	150KC	100KC	50KC	150KC
コーディング	10KLoc	—	10KLoc	10KLoc	5 KLoc	15KC	10KLoc	5 KLoc	15KC
単体テスト	100項目	—	100項目	100項目	50項目	150項目	100項目	50項目	150項目
統合テスト	50項目	—	50項目	50項目	—	50項目	50項目	25項目	75項目
システムテスト	20項目	—	20項目	20項目	—	20項目	20項目	10項目	30項目

（注1）KC（文字数），KLoc（ソースコード数），項目（テスト項目数）
（注2）各工程生産物量V_iは同じ生産物量変換係数（$H = V_{i+1} \div V_i$ ：$_i$は工程）を適用

（3） 変更回数と変更タイミングを考慮した開発量の把握

4.4.1項（1）で述べたように，発注者（顧客）の決断項目として，ある工程iの生産物に対する仕様変更量（変更追加量，変更棄却対象量），仕様変更回数，変更タイミングがあります。さらに，相互合意の項目には仕様変更の認定基準と仕様変更に伴う量（変更正味棄却量）があります。

ここで注目すべきは，ある工程iの生産物に対する仕様変更回数，変更タイミング（完成率）です。

ある工程iの当初量V_0，仕様変更回数をn回，k回目の当初量に対する完成量P_k（完成率ε_k×当初量V_0）および各回の仕様変更率α（ここでは変更追加量）を一律として仮定した場合，1回目の仕様変更による開発量（当初量＋仕様変更量）V_1^1は

$$V_1^1=\varepsilon_1\times V_0\ (1+\alpha)$$

となり，n回目には，べき乗に増大し開発量は

$$V_1^n=\varepsilon_1\times V_0\ (1+\alpha)^n$$

となります。**表4.5**に具体的な事例を示します。

表4.5　変更回数と変更タイミングを考慮した開発量事例

【コーディング工程での事例】

完成量	1回目	2回目	3回目	4回目	5回目	6回目	7回目	8回目	9回目	10回目
(j=1~10)	10KLoc	20Kloc	30KLoc	40KLoc	50KLoc	60KLoc	70KLoc	80KLoc	90KLoc	100KLoc
開発量	11KLoc	12KLoc	13KLoc	14KLoc	15KLoc	16KLoc	17KLoc	18KLoc	19KLoc	20KLoc
		11KLoc	12KLoc	13KLoc	14KLoc	15KLoc	16KLoc	17KLoc	18KLoc	19KLoc
			11KLoc	12KLoc	13KLoc	14KLoc	15KLoc	16KLoc	17KLoc	18KLoc
				11KLoc	12KLoc	13KLoc	14KLoc	15KLoc	16KLoc	17KLoc
					11KLoc	12KLoc	13KLoc	14KLoc	15KLoc	16KLoc
						11KLoc	12KLoc	13KLoc	14KLoc	15KLoc
							11KLoc	12KLoc	13KLoc	14KLoc
								11KLoc	12KLoc	13KLoc
									11KLoc	12KLoc
										11KLoc
合　計	11KLoc	23KLoc	36KLoc	50KLoc	65KLoc	81KLoc	98KLoc	116KLoc	135KLoc	155KLoc

$V_1^1=\varepsilon_1\times V_0\ (1+\alpha)$

$V_1^{10}=\varepsilon_1\times V_0\ (1+n\times\alpha)$

近似式

$V_1^{10}=\varepsilon_1\times V_0\ (1+\alpha)^n$

n＝10回，α＝10%（変更追加量），ε_j（j＝1「10%」~10「100%」：完成率），V_0＝100KLoc（当初量）

10回目の開発量（155KLoc）＝当初量（100KLoc）＋仕様変更量（55KLoc）

表 4.6　変更回数と開発量との関係

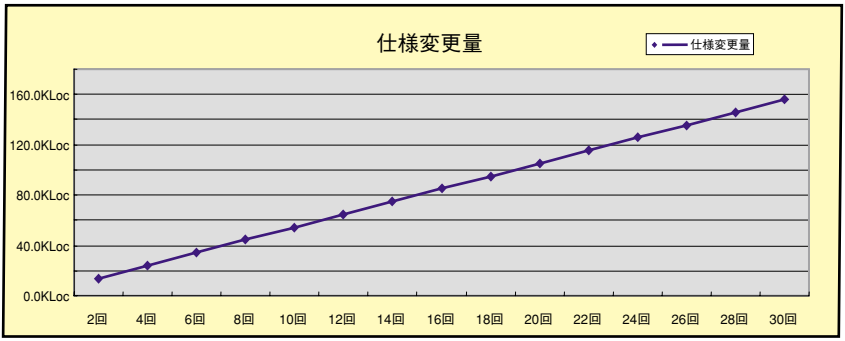


表4.5を条件に，さらに変更回数を増加させた場合の開発量を表したものを**表4.6**に示します。

4.4.2　見積りを行ううえでの仕様変更の考慮点

ソフトウェア見積り(特に影響が大きいテスト見積り)を行う際には，以下のことを留意すべきです。

契約にあたり，発注者(ユーザ)の決断を必要とする項目，および発注者との相互合意を必要とする項目について提示する必要があります。既に説明しましたが，仕様変更見積りに関わる発注者の決断項目には，仕様変更に伴う量(変更追加量，変更棄却対象量など)，変更回数および変更タイミングがあり，相互合意の項目には仕様変更の認定基準と仕様変更に伴う量(変更正味棄却量)があります。

4.5　テストの生産性に影響する変動要因

テストの生産性に影響する変動要因は，新規開発および改良開発に共通する生産性変動要因に，さらに改良開発特有の生産性変動要因が存在することを，SEC BOOKS『ソフトウェア改良開発見積りガイドブック』で紹介しています。また，改良開発特有のテスト生産性に影響する変動要因を**表 4.7**に示します。なお，新規開発および改良開発に共通するテスト生産性へ影響する変動要因は，4.6節で紹介しています。

表 4.7　改良開発に特有のテストに対する変動要因(生産性に影響)の例

主特性	副特性	影響の理由と評価の観点
既存システムの理解容易性	ドキュメントの整備環境	既存システムのテストの参考情報として状況を正しく反映したドキュメントがある場合は，テスト作業(テストケース作成・実施)の効率化につながる。
	システムの作り	既存システムの作りが理解容易な構成(例：モジュール化が進んだもの)であれば，テスト作業の効率化につながる。
	データ構造	データ構造の作りが理解容易な構成(例：正規化)であれば，テスト作業の効率化につながる。
変更箇所の分散度	―	変更箇所が分散していると，テスト作業が多岐にわたり，また，テストの分散など，テスト作業の効率化を悪化させる。
既存システムの正確性	―	既存システムの正確性が不足していると，テスト時の不具合の発生など，手戻りの発生を招き，テスト作業を増加させる。
改良開発の波及度合い	―	波及範囲が多いと波及先のテストを実施する必要があり，テスト作業を増加させる。
既存のテスト環境における再利用性	テスト環境	テスト環境を再利用できるとテスト作業を効率化することができる。
	テストケースおよびテストデータ	テストケースおよびテストデータを再利用できるとテスト作業を効率化することができる。
運用上の制約	―	改良開発におけるテスト環境は，多くの場合，運用環境となんらかの連携(ハードウェアやネットワークの共有など)を持つ。また，テストの一部を実施するために，運用環境そのものや実データを使用する場合がある。そうした場合，テストの実施時間帯やリソースの使用量，アクセス権などに制限を受け，テスト効率の悪化につながる場合がある。

4.6　非機能要件の把握とテスト見積りへの反映

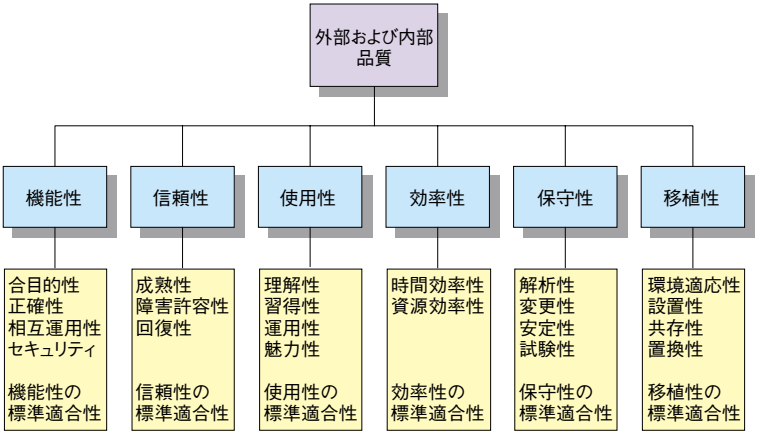
4.6.1　非機能要件の把握と確認方法

非機能要件[12]については，SEC BOOKS『経営者が参画する要求品質の確保』に示していますが，その概要に関して**表 4.8**に示します。

ここで扱う非機能要件はソフトウェアの品質要件を対象としています。ソフトウェア品質の定義は**図 4.11**に示すJIS X 0129[ISO/IEC 9126]に基づいています。なお，品質特性に関する概要説明を**表 4.9**に示します。

表 4.8　非機能要件の概要

分　類			概　要
非機能要件	品質要件		システムに要求される品質
	技術要件	ハードウェア	ソフトウェア開発に必要な機器，納品後の稼働環境に必要な機器および通信機器など
		ソフトウェア	ソフトウェア開発に必要なパッケージソフト，オペレーティングシステム(OS)およびツールなど
	その他の要件		運用・操作要件，移行要件および付帯作業など



(出典)JIS X 0129：2003(ISO/IEC 9126：2001)

図 4.11　外部および内部品質のための品質モデル

表 4.9　品質特性の概要説明

品質特性	概　　要
機能性	ソフトウェアが，指定された条件の下で利用されるときに，明示的および暗示的必要性に合致する機能を提供するソフトウェア製品の能力
信頼性	指定された条件下で利用するとき，指定された達成水準を維持するソフトウェア製品の能力
使用性	指定された条件下で利用するとき，理解，習得，利用可能となり，利用者にとって魅力的であるソフトウェア製品の能力
効率性	明示的な条件の下で，使用する資源の量に対比して適切な性能を提供するソフトウェア製品の能力
保守性	修正のしやすさに関するソフトウェア製品の能力。修正は，是正若しくは向上，または環境の変化，さらには業務仕様の変更などにソフトウェアを適応させることを含めても良い
移植性	ある環境から他の環境に移すためのソフトウェア製品の能力

（出典）JIS X 0129：2003（ISO/IEC 9126：2001）

非機能要件をテスト見積りに反映するには，非機能要件が要求仕様として明文化されていることが前提となります。しかし，一般的に，非機能要件は明文化されていない場合が多く，要求の度合いがあいまいとなる傾向にあります。

つまり，テスト見積りを行うに当たり，ソフトウェア品質を担保するうえで必要となる非機能要件の確認網羅性をどのように取得したらよいか，その点が課題です。

ここでは，課題に対する糸口として，非機能要件（品質要件）の度合いに対応するテスト見積り（テスト量およびテスト生産性）への影響を資料「非機能要件の把握・確認とテスト見積りへの影響表」に示しています。

なお，非機能要件（品質要件）の度合いに関しては，日本情報システム・ユーザー協会（JUAS）で発表している検収フェーズのモデル取引・整備報告書「非機能要求仕様定義ガイドライン」を参照して作成しています。

---

(12)　IEEEのコンピュータ・ソサイエティとACMの合同プロジェクトは，2004年‘Software Engineering Body of Knowledge 2004（SWEBOK2004）’を策定し，その中で非機能要件について定義しています。

4.6.2　非機能要件のテスト見積りへの反映

テスト見積りには，テスト量の見積りとテスト生産性の見積りが基本要素として存在します。当該システムに対して非機能要件(品質要件)の要求度合いが特定できれば，テスト量とテスト生産性に影響する度合いを評価し，ベースラインからの変動値としてユーザとベンダとで相互合意が可能になります。

例えば，資料「非機能要件の把握・確認とテスト見積りへの影響表」に示した非機能要件(品質要件)のテスト見積り(テスト量およびテスト生産性)への影響度合いに関して，ベースラインからの変動値として設定することで，定量化が可能になることを示しています。

4.7　欠陥修正に関わる工数の把握

4.7.1　欠陥修正量(工数)の把握

ソフトウェアテストでは検出した1つの欠陥を除去するために，その欠陥を作り込んだ工程にさかのぼって修正を行います。その修正作業は欠陥を作り込んだ工程以降，改良開発と同じ作業プロセスをたどります。通常，ソフトウェアテストでは多くのソフトウェアの欠陥が検出され，欠陥を作り込んだ工程はさまざまです。よって，各工程の欠陥修正量は，さかのぼった当該工程での欠陥修正量と先行する工程の欠陥修正によって修正される欠陥修正量とを総和したものとなります。**図4.12**にその関係を示しています。

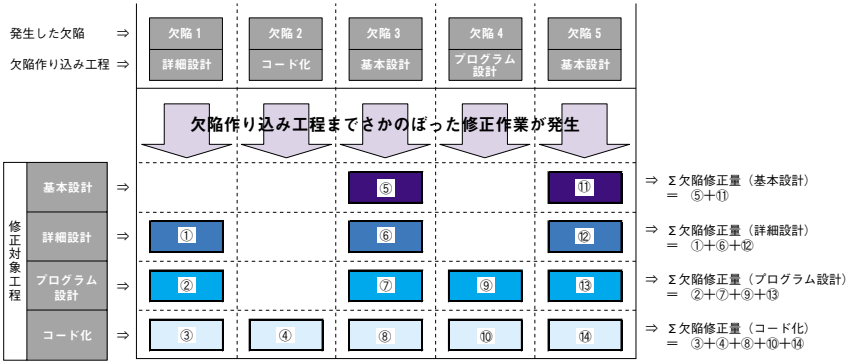


図4.12　開発工程に対応したソフトウェア欠陥修正量の概念図

作り込んだ欠陥の修正工数は，あらかじめ蓄積した実績欠陥修正データ(欠陥修正量，欠陥修正工数)から単位欠陥修正量あたりの欠陥修正生産性を設定しておくことで，予測が可能になります。例えば，ある工程 i の欠陥修正工数 $_i$ は，以下の式で求めることができます。

$$\text{欠陥修正工数}_i = \Sigma\text{欠陥修正量}_i \times \text{欠陥修正生産性}_i$$

なお，設計・実装工程(基本設計～コーディング)で作り込んだ欠陥の修正工数の扱いは，テスト見積りの範囲とするか，設計・実装工程のソフトウェア開発見積りの範囲とするかを，あらかじめ決めておく必要があります。

4.7.2　欠陥修正による再テスト工数の把握

ソフトウェアテストで検出された欠陥は修正され，再度，確認テストが行われますので，再テスト工数は欠陥を作り込んだ欠陥修正量を入力として算出することができます。

再テスト工数は，あらかじめ蓄積した実績再テストデータ(欠陥修正量，再テスト工数)から単位欠陥修正量あたりの再テスト生産性を設定しておくことで，予測が可能になります。例えば，ある工程 i の再テスト工数 $_i$ は，以下の式で求めることができます。

$$\text{再テスト工数}_i = \Sigma\text{欠陥修正量}_i \times \text{再テスト生産性}_i$$

見積り時点で，ソフトウェア欠陥の内容を予測することは困難です。しかし，修正は改良開発と同じ作業プロセスとなるため，欠陥修正による影響範囲が修正母体の特定部分に局所化されているか，それともシステム全体に分散されているかによって，工数は違ってきます。よってテスト進行中での修正作業計画には，修正によるテスト巻き込み量などを考慮しておく必要があります。

第５章　ソフトウェアテスト見積り精度の向上

5.1　見積りの前提条件のモニタリングとコントロール

ソフトウェアテストの見積りに限られるものではありませんが，見積り値と実績値とに差異が出てしまう原因の１つに見積りを行ったときの前提と実際の状況が変わってしまうことがあります。例えば，当初想定した母体システムへの影響範囲が設計，実装が進むにつれて明らかになり膨張し，最終的な影響範囲の規模が大きくなってしまったり(**図 5.1**)，プロジェクトの構想段階で何を作成するのかということすらあいまいな状況で見積もった結果が最後まで固定されてしまったり(**図 5.2**)することなどが端的な例です。また，さまざまな変動要因が当初想定していたもの(ユーザとベンダとで合意していたもの)が，状況の変更により，変わってしまうということも少なくありません。これらについては，SEC BOOKS『ソフトウェア開発見積りガイドブック』および『ソフトウェア改良開発見積りガイドブック』に記載しています。

ソフトウェアテストでは，見積りを行ったときの前提があいまいで実際のテストで問題が顕在化して混乱するというような事例が多く見受けられます。

- 後続テスト工程における前工程テスト漏れの判明
- 設計および実装フェーズの遅れによりテスト期間が短縮
- 仕様変更を組み入れた際の十分なリグレッションテストの未実施
- テストは実施したが欠陥が見つからない
- ソフトウェアテストの段階でなかなか欠陥が収束せず，ソフトウェアテストの期間もコストも増加

そのような状況が生じた場合，見積りの前提条件が変わってしまうことから，見積りの精度を確保することを見積り方法で解決することは無理です。見積りの前提条件を明らかにしたうえで，関係者でその情報を共有し，モニタリングコントロールを行っていくしかありません。

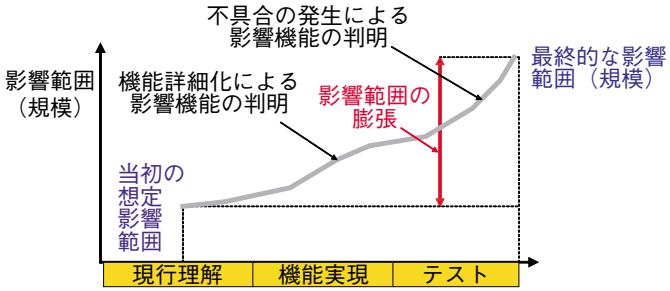


図 5.1　母体システムへの影響範囲の膨張

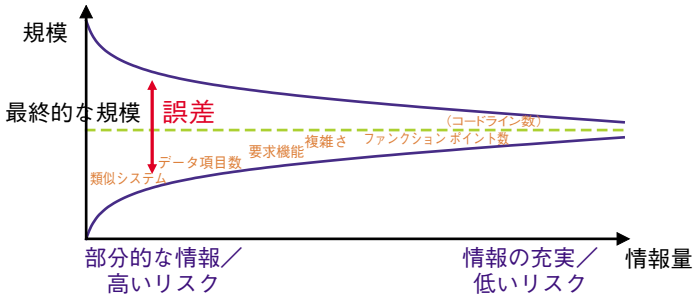


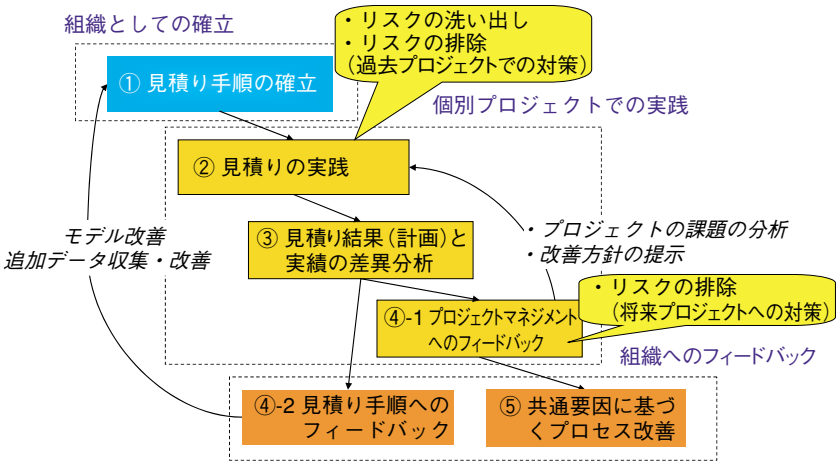
図 5.2　前提があいまいな状況

5.2　見積り手法と継続的な改善

こちらもソフトウェアテストに限ったものではありませんが，見積り精度の向上のためには，見積り手法・モデルを確立した後で，見積り予測値と実績値の違いの差異分析を通して，見積り手法・モデルの改善を行う必要があります。また，このとき重要なのは，手法またはモデルの改善だけでなく，プロジェクトマネジメントについても十分なチェックを行うことです。見積り値が妥当であっても，プロジェクトマネジメントに失敗した場合は，改善すべき対象はプロジェクトマネジメントの方になります。

また，リスクの排除も重要な見積りと実績を一致させるために重要なポイントとなります。ソフトウェアテストでは，採用するテスト技法，目標値，利用するツール，設計・実装工程段階における作り込み品質などが大きな変動要因となります。これらの変動要因が，「悪い」方向に変化した場合，見積りでは予測していない事態が起こることは経験的に明らかです。変動要因の影響を評価できない（状況がわからない）場合は言うまでもありませんが，リスクがわかった場合でも，そのぶれ幅を正確に予測することは難しく，逆にそのリスクを排除してぶれを除く努力をする方が効率的と考えられます。

以下に見積り精度向上の手順について記載します。また，**図 5. 3**に見積り手法の構築からフィードバックまでのサイクルを示します。



① 見積り手順の確立
　組織で一貫した手順として見積り手順を確立する。

② 見積りの実践
　確立した見積り手順を実践する。また，見積り活動そのものではないが，見積りにあたって可能な限りリスクを明らかにするとともに，排除しておく。

③ 見積り結果（計画）と実績の差異分析
　計画時の見積り結果とプロジェクト完了時の実績値との間の差異分析を通して，差異の根本原因を特定する。

④ 差異分析結果のフィードバック
　差異分析結果に基づいて改善対策を検討し，対策を展開する。差異分析の結果反映される対象として，次の2つがある。
・プロジェクトマネジメントへのフィードバック
・見積り手順（見積り手法を含む）へのフィードバック

⑤ 共通要因に基づくプロセス改善
　複数のプロジェクトで共通な課題を見積り，実績評価の繰り返しに基づき分析し，対策を検討して，プロセス改善を展開する。

図 5. 3　見積り精度向上のPDCA

5.3　契約によるリスク解消の糸口

5.3.1　見積りにおけるユーザ企業とベンダ企業の役割

見積りにおいて，ユーザおよびベンダのそれぞれの役割は以下のとおりです。

まず，ユーザはシステム開発におけるすべての意思決定の主体であり，機能要件や非機能要件の内容は，ユーザが決定します。機能要件の内容や品質要件・技術要件などは，システムの規模や開発の生産性を決定します。したがって，システムの規模や開発の生産性は，ユーザ企業がコントロールできます。

逆に，ベンダ企業側は，そのような要件をユーザ企業が判断・確認することをシステム開発のプロフェッショナルとしてサポートする必要があります。これは，ユーザ企業すべてがシステム開発に慣れているわけでないことが背景にあります。例えば，非機能要件のうちシステムの技術的難易度など，ユーザから見てシステム構築の知識がないと判断できないものは，ベンダのサポートが不可欠です。

ソフトウェアテストでは，システムテストや受入れテストなど，高位レベルのテストは，主にユーザ企業が主体でベンダ企業と協力して行い，単体テストや統合テストなどの低位レベルのテストをベンダ企業が主体でユーザ企業と協力して行うことが多いのが一般的です。3.4節で記載しましたが，低位レベルテストのプロセスは，高位レベルテストのプロセスと密接に関連していますので，高位レベルテストの視点から低位レベルテストの目標を定めて，低位レベルテストのテストプロセスを設計し，実施し，評価することが重要です。また，設計・実装工程段階で品質を作り込む主たる責任はベンダ企業にあり，これは低位レベルのテストのみならず，高位レベルのテストの量および生産性に大きく影響します。したがって，ソフトウェアテストの見積りでは，ユーザ企業とベンダ企業とが密接に協力して，テスト戦略を検討して，ユーザ企業およびベンダ企業相互に合意することから始めます。テスト戦略で，ソフトウェアテストにおけるユーザ企業とベンダ企業との役割および目標を明確にし，ソフトウェアテストの量や生産性に関してユーザがコントロールする事項と，ベンダがコントロールする事項とを区分して，ユーザ企業とベンダ企業とが相互に協力して，最終的なソフトウェアテストの工数またはコスト(ひいては価格)の低減や工期の短縮を図ることができます。

5.3.2　変動要因に関するユーザ企業とベンダ企業の調整プロセス

要求される品質のレベルと同様に，ユーザ企業とベンダ企業との間で，個別のソフトウェア開発プロジェクトでの変動要因のレベルをチェックして，今回のソフトウェアのテストは，個々の変動要因の基準から，どの程度高いのか，低いのかを確認しあい，そのレベルに応じて生産性の高低を評価し，見積りに反映することで，見積りの妥当性を確保する必要があります。これは新規開発，改良開発にかかわらず，同じような取り組みが必要です(13)。

5.3.3　ソフトウェアテストの段階的な見積りと多段階契約の採用

多段階契約では，契約作業にかかわる手間は増大しますが，開発途中で発生しがちなソフトウェアテストの前提条件の変化を確認しつつ，その時点で明確になった部分の反映が可能であるため，特に改造の影響範囲が不明確なシステムや高い信頼性を要求されるシステムを開発するプロジェクトに適しています。

図 5.4に多段階契約と契約・再見積りのタイミング例を示します。

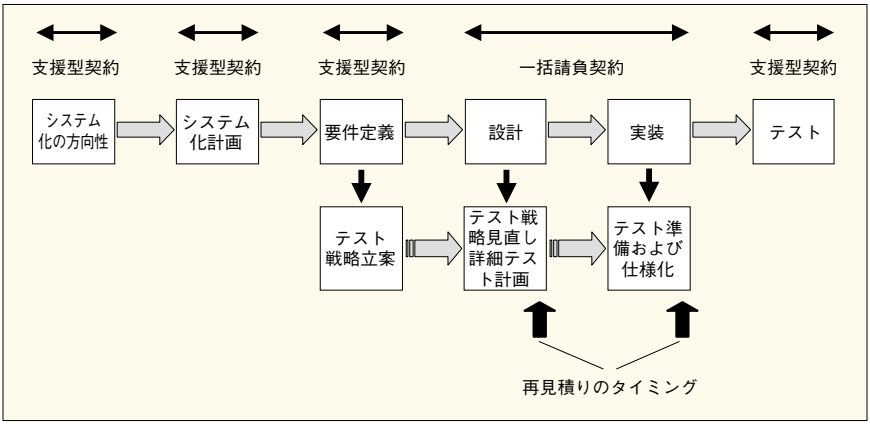


図 5.4　ソフトウェアテストに関する契約・再見積りのタイミング例

(13)　SEC BOOKS『ソフトウェア開発見積りガイドブック』の「1.3.4項(4) 変動要因に関するユーザとベンダとの調整プロセス」を参照ください。

第2部
事　例　編

第2部では見積り活動の事例集として，ソフトウェアテスト見積りおよび品質保証活動(テスト進行中における品質管理と再テスト見積りなど)に関して，先導的な取り組みを行っている各社の事例，当該方法の導入に当たっての留意点を示します。

第１章　事例編の見方

ここでは，ソフトウェアテスト見積りおよびテスト進行中における品質管理と再テスト見積りに関して，先導的な取り組みを行っている各社の事例を紹介します。

各事例は，原則として，以下のような構成となっています。なお，事例編において，欠陥や障害などの用語は企業の固有用語により記載しており，第１部とは意味合いが異なる場合があります。

（１）　取り組みの背景

ソフトウェアテスト見積り and/or 品質管理活動に関する各社のこれまでの取り組み，当該テスト見積り方法 and/or 品質管理を利用するに至った経緯，さらに，取り組みにおける問題意識などについて記述しています。

（２）　ソフトウェアテストの取り組み

ソフトウェアテスト見積り and/or 品質管理活動に関する概要を記述しています。

（３）　ソフトウェアテスト見積り

当該テスト見積り方法におけるモデル(入力パラメータ，出力，基本的なアルゴリズム・方法)を記述します。

ただし，本事例ではソフトウェアテスト見積り方法として，既発刊の『ソフトウェア開発見積りガイドブック』,『ソフトウェア改良開発見積りガイドブック』で紹介済などの場合は割愛しています。

（４）　テスト進行中における品質管理と再テスト見積り

テスト進行中における品質指標値の定量的な品質予測と再テスト見積り(品質制御)方法について記述しています。

（５）　プロジェクト実績の蓄積と活用

当該「ソフトウェアテスト見積り方法」and/or「テスト進行中における品質管理と再テスト見積り方法」でのプロジェクト実績の蓄積と活用事例を記述しています。

（6） 当該取り組みの課題

今後の課題，利用に当たって留意すべき点などについて記述しています。

なお，ここで紹介している各事例は，各社より次の内容となっています。

● 日本ユニシス

弊社では，「VモデルとV&V（検証および妥当性確認）」を基本の考えとし，大きく設計のフェーズとテストのフェーズに分けて，遵守すべき手順と管理指標を定めてきた。

また，昨今のソフトウェアの品質問題による社会的影響を鑑み，さらなる品質強化へ向け，改めてテスト技術にフォーカスするとともに，Wモデル開発の推進に取り組んでいる。

本稿では，開発工程の大きな割合を占め，コストに大きく影響するテストに焦点をあて，見積り，およびテスト進行中の品質制御と再テスト見積りに関し，留意している事項について紹介する。

● 日立製作所

弊社における品質指標を活用したテスト進行中における，品質管理と再テスト見積りの事例を紹介する。特に，プロジェクト自身の実績データを活用した品質マップによるプログラムの早期品質確保の事例を紹介する。品質マップとは，縦列に現象を，横列に原因などでマトリクスで表示して，不良摘出の実績データを記入したものである。

本事例では，品質マップによる不良の収束性を分析することにより，追加テストの方針を導き出すことができた。

● 東京海上日動システムズ

ソフトウェアテストの品質とテスト工程の見積りの関係について，弊社プロジェクトの実績をもとに，ソフトウェアテストの品質と生産性や本番での障害発生状況に関する分析結果を紹介する。

テスト工程の見積りのブレを早期に是正する仕組みとして，テスト工程のなかで中間評価を行い，品質を低下させる要因の芽を早期に発見し，対応要員・期間の確保などの調整を行うこと，要件変更・ペンディング状況などの残作業も明確にし，開発チーム内外と調整を行う取り組みを紹介する。

● **ジャステック**

弊社のモデルの基本アルゴリズムは，生産物量見積り方式および生産性見積り方式から成り立ち，さらにおのおのの方式において開発環境の違いや品質要求の多寡による変動を吸収する「環境変数」と呼ぶパラメータを導入している。

本稿ではソフトウェアのテストに焦点をあてて，テスト見積り，ソフトウェアテストの管理およびソフトウェアテスト効率の改善などに対する弊社の取り組みを紹介する。

● **日本電気（NEC）**

弊社における品質指標をもとにした，上流工程からの品質管理とソフトウェアテストの取り組みについて紹介する。弊社では上流工程からの品質管理としてソフトウェア品質会計制度と呼ぶ管理手法を採用し，上流工程からの欠陥管理，ソフトウェアテストの計画ならびに評価を行っている。

本稿では，ソフトウェア品質会計制度を中心にその概要と上流工程，下流工程それぞれの工程における適用方法，適用効果について紹介する。

第２章　日本ユニシスの事例
～品質強化への取り組みと留意点～

2.1　取り組みの背景

検証(レビューとテスト)工数を投入するほど品質は向上するが，ビジネスでは必ずコストと納期の制約が存在する。経済的なコストと妥当な期間で，要求された品質を実現するため，以下を実践する。

・顧客の要件を満たし実現可能な品質目標を定量的に設定する。
・開発途中の品質状況を計測し，品質目標と対比して分析する。
・品質目標と実績に乖離が存在した場合，品質目標達成に向けてプロセスを制御する。
・結果を評価し改善点を明確にして，継続的にプロセスを改善する。

本稿では，開発工程の大きな割合を占めコストに大きく影響するテストに焦点をあて，見積り，およびテスト進行中の品質制御と再テスト見積りに関し，留意している事項について紹介する。

2.2　ソフトウェアテストへの取り組み

顧客の満足を得る品質を実現するためには，開発の各フェーズごとに品質を定義しそれを確認することで，品質が満たされていることを検証しなければならない。

弊社では，「VモデルとV&V(検証および妥当性確認)」を基本の考えとし，大きく設計のフェーズとテストのフェーズに分けて，遵守すべき手順と管理指標を定めてきた。また，当該管理指標に基づいた実績データを蓄積し，次回の見積りに活用してきた。

図 2.1に示すように，Vモデルは，ライフサイクルを品質の確保という観点から定義したもので，早い段階でテスト仕様を作成できるので，品質の作り込みがより確実に行える。V&Vは欠陥の検出と除去の技術で，検証はあるフェーズでの作業結果が前フェーズで確定した仕様を満たしていることを明らかにする過程であり，妥当性確認では開発フェーズの最後に開発されたソフトウェアが要求事

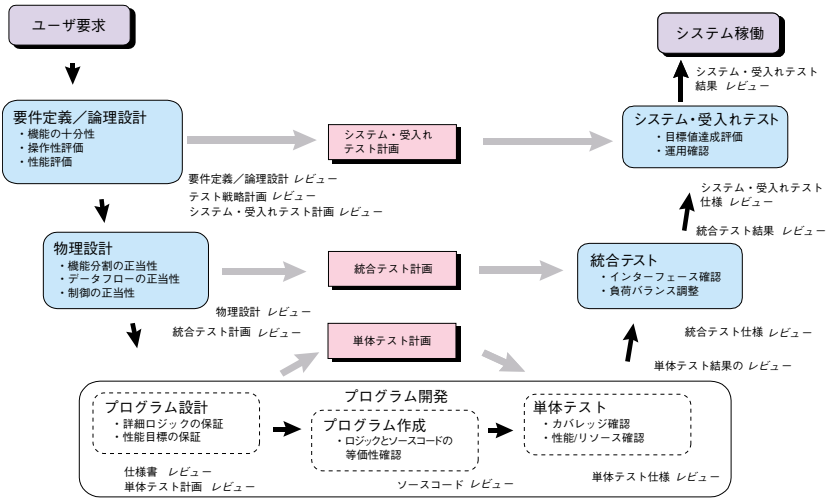


図 2.1　標準的な開発モデル：VモデルとV&V(検証および妥当性確認)

項をすべて満たしていることを確認する。

そういう状況下，昨今のソフトウェアの品質問題による社会的影響を鑑み，さらなる品質強化へ向け，改めてテスト技術にフォーカスするとともに，Wモデル開発の推進に取り組んでいる。

Wモデルでは，開発チームとテスト/品質保証チームが連携(並行)してVモデルに沿った開発を行うことにより，ソフトウェアの品質は確実に向上する。上流フェーズでテスト設計を実施し，テストケースを抽出することによって，テストを実施しなくても，要件や設計の抜け，あいまいな箇所，矛盾点などを見つけ出せる。端的にいえば，テストケースを使って，レビューを実施していることになる。また，上流フェーズでテスト設計を行うと，テストが難しかったりテストケースが増えすぎる仕様や構造がわかるので，要件定義や設計の質が向上する。

2.3　ソフトウェアテストの見積り

2.3.1　テスト関連作業の明確化

VモデルからWモデルへの変遷なども踏まえ，改めてテスト関連作業について，実施する時期と作業の内容を明確にする。従来は，テスト実施工程に行う作業をテスト(のスコープ)ととらえ，テストの作業量・工数を議論することが多かっ

た。Wモデルでは，実施する作業内容には大きな変化はないが，時期の前倒し，作業者の変化を始め，テスト戦略，計画などの重要性がより強調されるようになった。そこで，実施工程ごとの時間軸における実績を活用した見積りだけではなく，テストという切り口で実績を把握し，テスト作業の見積りの精度向上につなげている。

表 2.1は，テスト関連作業の実施時期を開発作業との対応で表している。

表 2.1　テスト関連作業

<table>
<tr><th></th><th>テスト戦略</th><th>システムテスト</th><th>統合テスト</th><th>単体テスト</th></tr>
<tr><td>要件定義</td><td>＜テスト戦略策定＞</td><td>＜テスト計画策定＞
（前半）</td><td></td><td></td></tr>
<tr><td>論理設計
物理設計</td><td></td><td>＜テスト設計＞
・要求事項の洗い出し
・確認事項への展開
・確認事項の整理</td><td>＜テスト計画策定＞
（前半）</td><td></td></tr>
<tr><td rowspan="3">プログラム開発</td><td rowspan="3"></td><td>・テスト項目への変換
・テスト項目レビュー</td><td>＜統合テストの結合順序＞
＜テスト設計＞
・設計事項の洗い出し
・確認事項への展開
・確認事項の整理
・テスト項目への変換
・テスト項目レビュー</td><td rowspan="3">＜テスト計画策定＞
＜テスト設計＞
・テスト項目作成
・テスト項目レビュー
＜単体テスト準備＞
・テスト環境整備
＜単体テスト実施＞
・テスト実施
・実行結果のチェック
＜テスト結果レビュー＞
・テスト報告策定
・レビュー</td></tr>
<tr><td colspan="2">・テスト項目の移動（両テストで同期して実施）</td></tr>
<tr><td>＜テスト計画策定＞
（後半）</td><td>＜テスト計画策定＞
（後半）</td></tr>
<tr><td>統合・システムテスト</td><td></td><td>＜システムテスト準備＞
・テスト環境整備
＜システムテスト実施＞
・テスト実施
・実行結果のチェック
＜テスト結果レビュー＞
・テスト報告策定
・レビュー</td><td>＜統合テスト準備＞
・テスト環境整備
＜統合テスト実施＞
・テスト実施
・実行結果のチェック
＜テスト結果レビュー＞
・テスト報告策定
・レビュー</td><td></td></tr>
</table>

（注）　縦方向は時間軸を示し，＜　＞はテスト関連作業，・はテスト関連作業の中の詳細作業項目を示している。

ここでは，特に重要視されてきた“テスト戦略”と“テスト計画”について解説する。

まず，各テスト(システムテスト，統合テスト，単体テスト)の考え方，達成すべきゴール(テスト終了条件)，障害管理の方法，テスト支援ツールの適用有無など基本方針をテスト戦略として定める。

そのテスト戦略に沿って，以下のような項目を検討し，テスト計画書としてまとめる。

① テストの対象範囲と対象外範囲，② テスト環境と準備事項，③ テスト結果記録方法とレビュー方法，④ テスト開始に必要な事項と条件，⑤ テスト終了基準，⑥ テストの準備，設計，実施，結果レビューなど作業のスケジュール，⑦ 体制と役割

なお，ここで特筆すべきことは，テスト計画を前半(テスト設計の前)と後半(テスト設計の後)の2段階に分けて策定していることである。前半では，テストの概要およびテスト設計作業を含めたテストに必要な作業と概略スケジュールを明確にし，後半はテスト項目が確定した段階で行い，テスト項目と詳細な実施スケジュールを策定してテスト計画を完成させる。

2.3.2　テストの量

「どこでテストを終われるか？」は，どのプロジェクトでも必ず直面する課題である。

テストの量を左右するテストの網羅性とテスト終了基準について言及する。

(1)　テストの網羅性

ソフトウェアは複雑で多様な機能の側面を持ち，それぞれの側面からソフトウェアを見た場合異なった姿に見えることがある。そこで，テスト項目の策定にあたっては，担当者個人のスキルに依存する部分を極力減らし，テストの網羅性を確保するため，以下のような各側面より，テストすべき事項を考えることが有効である。

① 時間効率，② 資源効率，③ 耐久性，④ 大容量，⑤ 互換性，⑥ 整合性，⑦ 環境，⑧ セキュリテイ，⑨ 障害，⑩ 入力データ，⑪ 構文，⑫ 操作，⑬ 状態遷移，⑭ 言語，⑮ 導入，⑯ 保守，⑰ 文書，⑱ 機能，⑲ インターフェース，ほか

（2） 代表的なテスト終了基準

① 収束を使ったテスト終了判定

・テスト工程で障害が収束すれば，本番環境で障害が出にくいということが経験的にわかっている。

・通常は障害の収束の判定に「信頼度成長曲線」を使用する。

② 「カバレッジ」による十分性チェック

一般的な基準では，「分岐網羅（C1カバレッジ）」を100％達成する。

③ 「レビュー」による十分性チェック

考え得る仕様パターンを網羅したか（ブラックボックステスト）をレビューでチェックする。

④ 「品質メトリックス」による十分性チェック

・品質メトリックスとして，テスト密度と障害検出率を使用する。

　－ テスト密度＝テスト項目数（件）/開発規模（KLoc）

　－ 障害検出率＝障害件数（件）/開発規模（KLoc）

・以下の事項を考慮して，「プロジェクト品質目標（品質メトリックス値）」を決定する。

（a）組織の品質目標：「品質メトリックス値目標ガイド」

（b）顧客要件とプロジェクトの特性を反映した品質特性

高信頼性要求（障害が人命にかかわる，障害が公表されると企業の評価に重大な影響を及ぼす，など）

短期開発要求（品質を犠牲にしても短期開発を実現したい，など）

コスト削減要求（開発コストを低下させたい，など）

（c）開発担当組織の過去の実績

これまで開発した類似システムでの品質メトリックス実績値

外部委託する場合は協力会社の品質メトリックス実績値

2.3.3 テストの生産性改善

テストフェーズの生産性は，テスト関連作業の各局面での工夫により，改善することが可能である。

（1） テスト手順の最適化

テスト環境とテストデータの準備を効率的に実施できる順序を検討する。

・前工程の処理結果を次工程の入力として利用することで，データ準備工数を

削減する。
・サブシステム内で完結する処理を固めてから，連携処理のテストに入る。
・テスト実施者の習熟効果を高めるため，同じ種類のテストパターンを集中的に実施する。

（２） テスト準備の効率化

テスト効率の高いテストケースを設計する。
・ロジックの網羅性が高いテストケースを設計して，総テスト件数を減少させる。
・テストケースを使い回せるように，正常系と異常系を分けて設計する。
・モジュールの制御構造をサンプリングで確認して，Ｃ１網羅率をカバーできるテストケース数の目安を把握しておき，テストケース数の妥当性を評価する。
・条件の組み合わせは，デシジョンテーブルを使用してテストケースを設計する。

（３） テスト実施の効率化

①　ある部分がエラーになると残りのテストケースが実施不可能となる状況を回避する。
・相互に依存しないテストケースをあらかじめ用意しておく。
・必ず実行されるクリティカルなロジックは，事前にレビューやミニテストを実施する。
・途中の処理で誤った結果が出力されても処理が続行できるようにするため，チェックポイントごとの正しい入力をあらかじめ用意しておく。
・目的とする結果が得られたか直ぐに判断できるようにする。
・期待する出力を事前に文書化しておくことで，検証者の判断を待たなくてもテスト実施者がその場で結果を確認できるようにしておく。
・期待する結果が得られた場合と得られなかった場合の取得すべき証跡を明示しておく。

②　テスト用のコードをあらかじめ組み込んでおき，実行時のオプションなどでテスト用コードのオン/オフを制御し，本稼働時を含めて必要なタイミングで重要情報を取得できるようにしておく。

③　リグレッションテストの効率化

各サブシステムごとに必ずテストすべきパターンをあらかじめ設定してお

き，修正結果受入れの都度実施する。可能ならばツールを使用して当該作業を効率化する。

④　ツールの有効活用

・カバレッジツールによる網羅性チェック，メモリリーク/バッファオーバーフローなどチェックツールを適用すると，テスト効率の向上だけでなく，テストプロセスの信頼性が向上する。

・負荷テスト，リグレッションテストなど繰り返し実施し，同一のタイミングを発生させる必要があるテストでは，効率面だけでなく，テスト自体の品質向上という面でも有効である。

（4）　修正作業の効率化

・期待する結果が得られない場合,正しくないと想定される部分とその根拠(設計書の当該部分)，および期待する結果を示すことで，修正者の作業効率を高める。

・修正後に確認すべき項目を標準化しチェックリスト化して，修正結果に証跡を添付する。

2.3.4　ソフトウェアテストの見積り

2.3.1項で洗い出した作業を，ドキュメント作成工数，テスト工数と管理工数として見積もる。

（1）　ドキュメント作成工数

＜テスト戦略策定＞，＜テスト計画策定＞，＜統合テストの結合順序＞，＜テスト設計＞と＜テスト結果レビュー＞のテスト報告策定作業について見積もる。

■成果物作成工数：テスト戦略書，テスト計画書，テスト仕様書，テスト報告書など作成工数規模(要件数，機能数，取引数，論理DB数，画面数，帳票数など)×生産性

■レビュー工数：開発規模×レビュー実施率×平均レビュー参加人数

■手直し工数：開発規模×不具合検出率×平均不具合手直し工数

（2）　テスト工数

■所要工数：開発規模×テスト密度×平均テスト工数

平均テスト工数＝(① テストケース準備＋② テスト実施＋③ 結果検証＋④ レビュー)の各平均工数

① テストケース準備：＜テスト準備＞のテスト環境整備

②テ　ス　ト　実　施：＜テスト実施＞のテスト実施
③結　　果　　検　　証：＜テスト実施＞の実行結果のチェック
④レ　　ビ　　ュ　　ー：＜テスト結果レビュー＞のレビュー
■手　直　し　工　数：開発規模×不具合検出率
　　×(平均不具合手直し工数＋平均テスト工数)

（3） 管理工数

上記以外に，“テスト終了基準の検証”，“テストの終了や収束判断”など，各フェーズごとの品質レビューと引き渡し直前の「リリース判定会議」関連の工数が必要である。

（4） 期間見積り

統合テスト以降になると，コンポーネントの結合，テスト環境の使用調整など，時期を意識した計画が必要である。よって，単なる工数からの期間見積りにならないように留意することが重要である。

2.4　テスト進行中における品質制御と再テスト見積り

2.4.1　テストと品質制御

精緻な計画を立て，工程の途中で成果物の品質を把握し，開発作業を制御しながら，着実に品質を高めていく。品質制御には，①あらかじめ，環境変化の工程への影響などを考慮して，必要な品質を達成するための方策を計画する“前向き制御”と，②工程の途中の状態を検出して，それに基づき最終的な品質を予測し，所期の品質が達成されるように対策を講じていく“後ろ向き制御”がある。

また，コントロールを適切に実施するためには，「品質特性ごとに品質の要求レベルが定量的に表現可能」であり，「品質が各工程で計測可能」でなければならない。

2.4.2　開発途上の品質の監視と管理

開発成果物を評価し，欠陥数を予測したり，開発が完了した時点で本稼働時の欠陥発生数を見積もったりすることは，重要な活動である。しかし，成果物品質を十分確保するためには，開発途上においてソフトウェアの品質を監視し管理することが，さらに重要である。

2.4.3　品質評価

低品質であることが評価結果から明確である場合，その原因を究明し対策を明確にする。成果物全体がまんべんなく低品質という場合もあるが，この場合は根本に戻って，プロジェクト編成から見直す必要がある。一般的には，特定の部分で障害が多発しており，その部分に対応することによって品質を大幅に改善できるといわれている。この特定の不良部分を見つけるために，「層別」と呼ばれる手法を用いる。

（1）「層別」による分析

通常は以下の観点で「層別」し，品質を評価して不良部分を浮き出させる。

・サブシステム別・機能単位別・モジュール別，協力会社別・チーム別・担当者別，ユーザインターフェース部分・制御部分・業務処理部分・DBアクセス部分，モジュールの開発規模別，インフラ別・開発言語別，開発時期(先行開発・追加開発など)別，など

（2）　品質メトリックス値の目標達成度

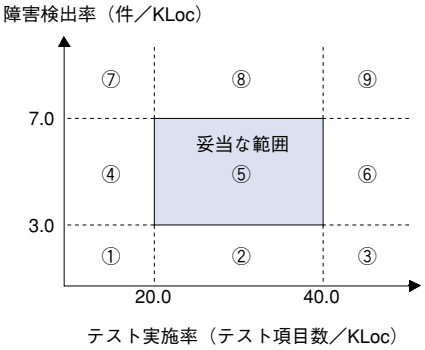
・**図 2.2**を参考にして，テストの有効性と成果物の品質を評価する。

「妥当な範囲」に入っていても，レビュアやテストケースの作成者の実力によっては十分な品質が確保されていない場合があるので，懸念がある場合は，現物をサンプリング調査して確認する。

・各フェーズの途中で評価し，問題がある場合は直ぐに対策を実施する。
・開発フェーズが進むと，当該フェーズの品質メトリックス値だけでなく，それ以前のフェーズの品質メトリックス値も参照して品質状況の推移にも注意する。
・フェーズごとに偏りが上へいったり下へいったり一定しない場合は，レビュー/テストのプロセスが安定していない可能性があるので，「障害原因フェーズ」を確認して，次フェーズへ障害が持ち越されていないかチェックする。当該フェーズ以前に作り込まれた不具合/障害が多い場合は，さかのぼって強化テスト/レビューを実施する必要がある場合がある。

図2.2に統合テストの品質メトリックスをグラフ化した。

適切なテスト項目数で，適切な量の不具合を検出できた場合に，テストの品質が妥当であると評価している(図2.2の⑤)。

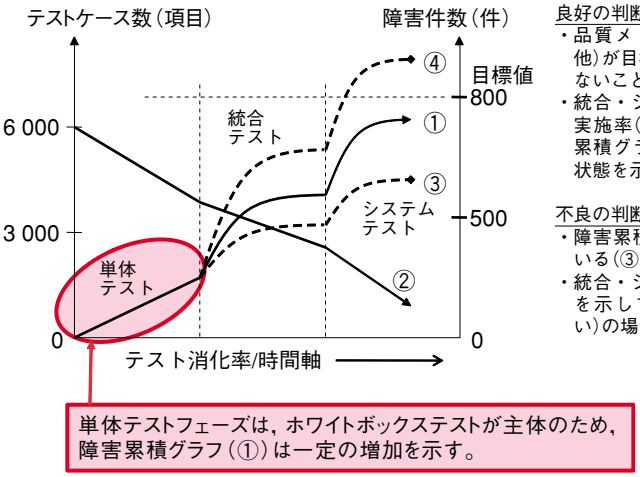


① テスト項目数，検出数ともに過少
→現在のテストを継続する必要がある。
②,③ 検出数が過少でテスト項目数は範囲以上
→テストの観点が間違ってないか確認が必要。
（テストの観点が合っていれば非常に品質の良いプログラムといえる）
④,⑦ テスト項目数が過少で検出数が範囲以上
→プログラムの品質が明らかに良くない。
⑥ テスト項目数が過大だが検出数は範囲内である
→テストの仕方，テスト内容に無駄がないか。
⑧ 検出数が過大だがテスト項目数は範囲内である
→プログラムの品質が良くない。
⑨ テスト項目数，検出数ともに過大
→プログラムの品質が悪く，テストにも手間を掛けすぎてないか。

図 2.2　テスト時の品質評価（品質メトリックスの見方）：ゾーン分析（統合テスト）

（3）　信頼度成長曲線の分析

テストフェーズが進捗すると残存障害が少なくなり，次第に障害を検出することが困難となる。成果物の信頼度が向上（成長）していく様子をグラフ化したものを「信頼度成長曲線」と呼び，その傾きによって障害の収束状況を判定する。**図 2.3**を参考にして品質状況を判断するが，判断に当たっては以下の事項に留意する必要がある。



良好の判断
・品質メトリックス値（テスト密度他）が目標値に近く，大幅な乖離がないこと。
・統合・システムテストはテストの実施率（②）が一定で，徐々に障害累積グラフの傾きが寝てきて収束状態を示す。

不良の判断
・障害累積グラフが目標と乖離している（③，④）。
・統合・システムテストで収束状態を示していない（傾きが寝てこない）の場合。

図 2.3　信頼度成長曲線グラフの評価方法

① テストの消化状況が一定でない場合は，時間軸を横軸としたグラフでは，傾きが障害の収束状況を適切に表していない可能性があるので，テストの消化率を横軸にとったグラフで障害の収束状況を確認する。
② 信頼度成長曲線が収束を示していても，品質メトリックス値が適切でない場合は，現物の品質状況をサンプリングで確認する。
③ 信頼度成長曲線のグラフでは，「障害未解決残」の曲線もチェックして，障害が着実に解決され未解決残が増加傾向を見せていないことを確認する。
④ 工数の消化状況と障害の収束状況を対比して，両者のバランスが取れていることを確認する。
⑤ 障害が一見収束しているように見えても，未実施のテスト項目が多く残っている場合があるので注意が必要である。信頼度成長曲線は変曲点を持つモデルであるため，モデルが適切に機能するためには，変曲点(約60%)後のデータが必要である。
⑥ 複数のサブシステムのテストが異なるタイミングで実施され，それらを合算してシステム全体を表示したグラフでは，一見収束しているように見える場合があるので，サブシステムごとのグラフも確認する。

(4) 過去の実績と比較する品質評価

・対象とする業務領域，適用するインフラストラクチャ，担当するプロジェクトマネジャ，プロジェクトの構成メンバなどが，同一であったり似かよっていると，一般的には，前回と今回のプロジェクトで実績が大幅に異なることはない。ただし，開発期間が極端に短いとか，非常に厳格な管理が要求されるなどの特別な条件が加わる場合は，その限りではない。
・過去の実績は，品質目標の妥当性評価，レビューやテストの進捗(生産性)評価，品質の予測に有用な情報を提供してくれる。基本的には，過去のやり口に対して何らかの改善が加えられていない限りは，習熟効果だけでは大幅な品質や生産性の向上は実現できない。
・過去の実績と大幅に乖離した目標を設定している場合は，工数や開発期間の見積りで無理をしている可能性がある。したがって，品質を評価する場合には，単に品質だけを見るのではなく，進捗や投入工数にも目を配る必要がある。納期やコストを守ろうとすると，品質にしわ寄せが行き，検証工数を削減してしまう可能性が高い。
・品質保証部門による第三者品質レビューでは，プロジェクトのプロセス品質

とプロダクト品質を評価するため，品質データの評価だけでなく収集・分析・整備の各活動の内容も検証される。これによって，信頼性の高い品質情報が蓄積されていくので，次回以降のプロジェクトでは，過去の品質実績との比較が可能となり，品質評価の信頼性が格段に向上する。

2.4.4　再テスト見積り

ゾーン分析，あるいは，信頼度成長曲線にてテスト途上の品質を評価するが，当初の品質目標値を達成しないことがある。その場合は，速やかにその事象を分析し，無駄な作業，あるいは，対応が手遅れにならないよう，以降のテスト方法を見直す必要がある。

（1）　いくらテストしても障害が出ない。障害検出率が目標値に到達しない。

層別分析を繰り返し，層別ごとの品質目標値を設定し直すなどの対応が必要である。

（2）　十分なテスト(テスト密度)を実施したが，障害がおさまらない。

層別分析を繰り返し，不良部分を特定する。障害の根本原因分析(なぜ，なぜ，なぜ・・・)をし，品質の作り込みが不十分だったのか，品質確認が不十分なのかを究明する。障害分類(仕様/コーディング不良，漏れ/誤り/違反など)と原因分類(技術/検討/確認/注意不足など)，および，障害が発生した工程と障害が混入した工程を明確にし，その障害分類と原因分類に応じた対応を実施する。品質の作り込みが不十分な場合は，レビュー状況，担当者の偏りの有無，同様の問題がほかにもないかなどを確認し，チェックリストの改善，熟練者による指導，追加レビューの実施，あるいは，担当者の交代などによる対応の後に，追加テストを実施する。また，障害を混入させた工程内で発見できなかった原因を追究し，今回も含め以降のプロセス改善につなげている。

一方，品質確認が不十分な場合は，テスト戦略，テスト設計を見直し，テスト計画を再策定する。

2.5　プロジェクト実績の蓄積と活用

経験則だけに頼らず，事実に基づくデータにより意思決定を行うことは，ソフトウェア開発では必須となっている。プロダクトの成果物とプロダクトメトリックスおよびプロジェクト内での各種プロジェクトメトリックスを設定し，実績データを収集，そのデータにより，プロジェクトをコントロールする。

図2. 4は開発工程に対応した成果物とメトリックスを体系化した例であるが,特に今回は，その中で，テスト関連作業の成果物とその作成時期(の変遷)を強調している。

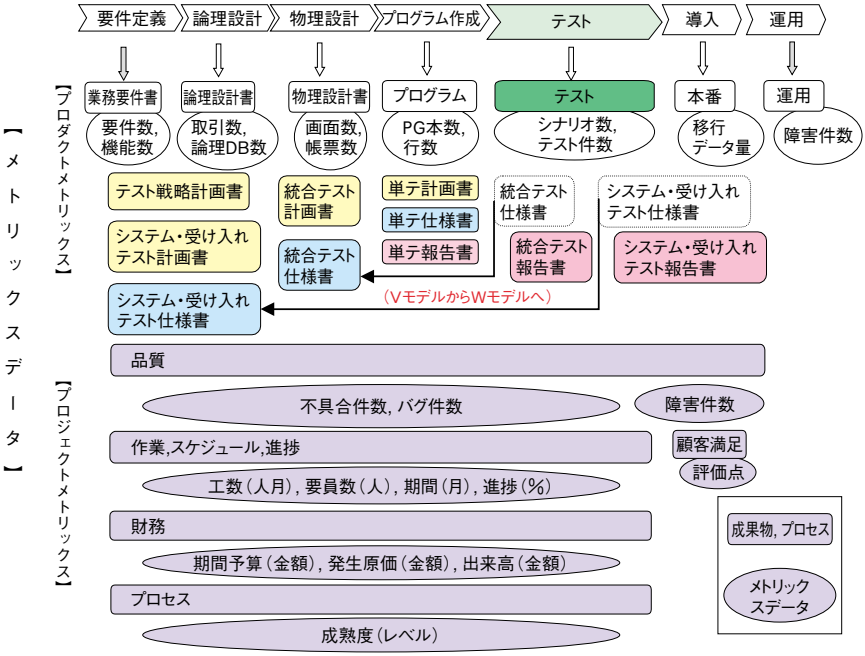


図 2. 4　成果物と指標/実績データ体系図

弊社では，プロジェクト完了報告書に基づきプロジェクト終結レビューを実施してプロジェクトを総括し，各種実績データの蓄積を行っている。生産性，品質，工程，要員，規模，期間，再利用ほかCOCOMOⅡパラメータなど，定量データは，業務領域，開発規模，開発期間，開発・改良など，種別，開発フレームワーク/言語など，識別のうえ，次回以降の見積りに活用される。また，計画対実績差異理由，ツール活用，再利用内容，その他工夫点など，定性的な面でも有効な情報を提供している。

2.6　当該取り組みの課題

現在，プロジェクト実績の蓄積，活用は，プロジェクト完了時の情報としているため，その情報の鮮度が問題となることがある。そこで，現在のプロジェクト完了時の実績データに加え，品質保証部門による検証済みの開発途上の実績データの蓄積を進めている。

手戻りなどのことを考えると，未完了プロジェクトの実績データの精度には多少の不安が残るが，各工程ごとのきめ細かな生産性・品質など“旬”のデータが，プロジェクトの詳細な特性および各種組織資産とともに蓄積が進むと，同プロジェクトの中での最大限の活用だけでなく，他のプロジェクトにも有益な情報が提供できることになる。

第３章　日立製作所の事例
～品質マップによるプログラムの早期品質確保～

3.1　取り組みの背景

近年，社会的に混乱を引き起こす障害がITシステムに発生したことを契機に，社会インフラとして，ITシステムの信頼性に対する重要性がますます増加している。企業は，ビジネス環境の変化にタイムリーに対応することが，競争力を維持するためには必要である。そのために，タイムリーに業務プロセスを見直し，その業務プロセスに対応したITシステムをタイムリーに構築することが求められている。このように，信頼性，短納期化の要請が大きくなるだけでなく，ITシステムの大規模化，複雑化，および利用範囲の拡大化の傾向にある。このような状況において，品質を確保するための具体的な課題を以下に列挙する。

・ユーザニーズ，使用条件などを正確に把握した十分なテスト計画の立案
・設計工程の主な成果物であるドキュメントの設計品質の定量的把握
・ユーザニーズを満足する信頼性を確保したかの把握
・信頼性が個人の能力とチーム能力に大きく依存

本章では，弊社におけるソフトウェアテストへの取り組みを3.2節で，ソフトウェアテスト見積りを3.3節で，品質指標を活用したテスト進行中における品質管理と再テスト見積りの事例を3.4節で紹介し，さらに，プロジェクト自身の実績データを蓄積し，テスト工程の早期段階における品質マップを活用した品質管理と再テスト見積りの事例を3.5節で紹介する。品質マップとは，縦列に現象(プログラムの機能種別)を，横列に原因(技術要素)などをマトリクスで表示して，不良摘出の実績データを記入したものである。この品質マップを適用し，テストの十分性や不良の収束性を確認することにより，追加テストの方針を導き出すことができた。

3.2　ソフトウェアテストへの取り組み

品質保証の主要な方針としては，① 開発プロセスの標準化によるインプロセス品質コントロールによる品質の作り込み，② 開発プロセスの標準化による定

量的な品質管理による品質の可視化，③ 開発部門と品質保証部門の分離による権限の分離などである。

これらの方針にしたがった施策の 1 つとして，品質確保のための管理基準を規定している。ユーザの要求仕様に応じた品質を確保するために，公共性，社会的な影響の度合いおよび信頼性要件に応じて 5 段階の管理ランクを設定し，管理ランク別に管理項目と管理内容を規定している。管理項目の具体的な例としては，工程完了判定，デザインレビュー，稼働前品質・稼働準備状況のレビューなどがある。

また，設計品質を向上するために，事前に登録した有識者を参加させて設計ドキュメントのデザインレビューを実施することを規定している。なお，デザインレビューとしては，処理方式設計，信頼性設計，性能設計，運用設計，テスト計画などの設計ドキュメントが対象である。さらに，各プロセス完了時に品質保証部門によるドキュメント検査も実施している。この節では，ソフトウェア品質の考え方およびソフトウェア開発プロセスと不良の関係についてそれぞれ解説する。

3.2.1　ソフトウェア品質の考え方

広義には，ソフトウェアの品質とは，ユーザの満足度と考えることができる。ユーザが要求仕様を定義し，設計者がユーザの要求仕様を入力にして設計仕様を設計し，開発者が設計仕様を入力にしてプログラムを製造しテストする。ここで，品質を，ユーザ，設計者，および，開発者の三者間の関係で考えてみる。

品質を確保するためには，ユーザと設計者，設計者と開発者のインターフェース部分で十分なコミュニケーションをとることが重要である。ユーザと設計者間のインターフェース部分に関係する品質は設計品質であり，設計者と開発者のインターフェース部分に関係する品質はプログラム品質である。設計品質は，「ユーザの要求仕様を入力にして設計した設計仕様が，ユーザの要求に適合しているレベル」で定義できる。一方，プログラム品質は，「プログラムが設計仕様を満足して正しく動作するレベル」で定義できる。したがって，ソフトウェアの品質は，設計品質とプログラム品質の両方が優れていて実現できる。

3.2.2　ソフトウェア開発プロセスと不良

ソフトウェア開発プロジェクトの開発プロセスと不良分布の関係を**図 3.1**に示す。

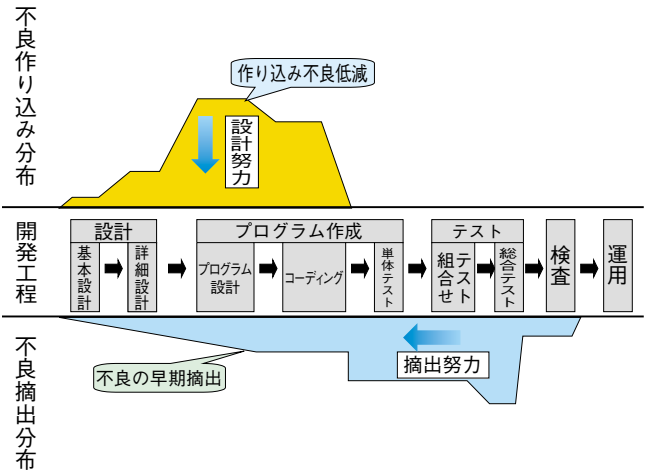


図 3.1　ソフトウェア開発プロセスと不良分布の関係

一般的に，ソフトウェア開発プロジェクトでは，コーティング量の数%の不良を作り込み，それを摘出するのに約半分の工数がテスト工程にかかっていると考えられている。高品質なソフトウェアの開発プロセスを実現することができれば効率も大幅に向上することが可能である。そのために，高品質を実現するためには，まず，設計者がユーザの要求仕様を入力にして設計仕様を設計するときに，不良の作り込みを未然に防止することが最重要である。また，開発者が設計仕様を入力にしてプログラムを製造・テストするときに，不良をできるだけ早期に摘出することが重要である。

3.3　ソフトウェアテスト見積り

3.3.1　プロジェクトの品質指標の目標値の設定

プロジェクトでは，テスト計画時に，**表 3.1**に示すようなテスト工程のプロセスごとに品質指標の目標値を設定している。プロジェクトでは，過去の類似プロジェクトの品質指標の実績値を考慮して，対象プロジェクトのユーザや社会への影響，システム特性，体制，スキル，新技術，および，作業条件などを考慮して，適切な品質指標の目標値を設定している。なお，テスト工程のプロセスの品質指標ごとに，品質指標の実績値を弊社内で公開している。

表 3.1　テスト工程のプロセスごとの品質指標の事例

テスト工程のプロセス	主要な品質指標
机上デバッグ	ウォークスルー，テスト項目密度（件数/KS），不良摘出密度（不良摘出件数/KS）
単体テスト	テスト項目密度（件数/KS），C0 ガバレージ率，C1 ガバレージ率，不良摘出密度（不良摘出件数/KS）
組合せテスト	テスト項目密度（件数/KS），不良摘出密度（不良摘出件数/KS）
総合テスト	テスト項目密度（件数/KS），不良摘出密度（不良摘出件数/KS）

さらに，テスト項目の目標値を設定する場合には量的な件数だけを満足していてもプログラム品質を効率的に確保できないので，質的な基準も規定している。プロジェクトで定めた質的な基準の一例として，正常系のテスト項目件数の割合（60%以下），異常系のテスト項目の割合（15%以上），境界/限界系のテスト項目件数の割合（15%以上），インターフェース系のテスト項目件数の割合（10%以上）などがある。

3.3.2　不良摘出曲線での管理曲線の設定

ソフトウェア開発のテスト工程では，テスト項目を消化し，不良を発見し，不良を除去する。したがって，テスト実施時間が長くなると，潜在する不良は減少して，その結果として信頼度は向上する。このテスト実施時間と信頼度の関係を，モデル化することにより理論的に予測・評価するのがソフトウェア信頼度成長モデルによる不良予測の手法である。

弊社では傾向曲線モデルの 1 つであるゴンペルツ曲線を適用して，不良を予測している。標準的な過去プロジェクトの実績データから，上限と下限の管理曲線を設定している。標準的なプロジェクトであれば，この上限と下限の管理曲線内になることが実証されている。

プロジェクトでは，3.3.1項で設定した不良摘出の目標値にしたがって，テスト開始前に，**図 3.2**に示すようにゴンペルツ曲線を適用したテスト工程管理図（不良摘出曲線）の上限と下限の 2 つの管理曲線を設定する。

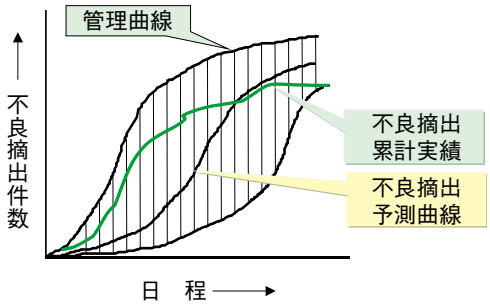


図 3.2　テスト工程管理図(不良摘出曲線)

3.4　テスト進行中における品質管理と再テスト見積り

3.4.1　テスト工程管理図によるテスト管理

テスト工程の代表的な品質指標は，テスト項目件数と不良摘出件数である。**図 3.3**に示すテスト工程管理図により，２つの品質指標の推移を把握することができる。テスト工程管理図を活用して，テスト項目の消化・進捗状況，不良の収束状況を分析・評価し，できるだけ早く品質状況を把握し，適切な対策を講じるこ

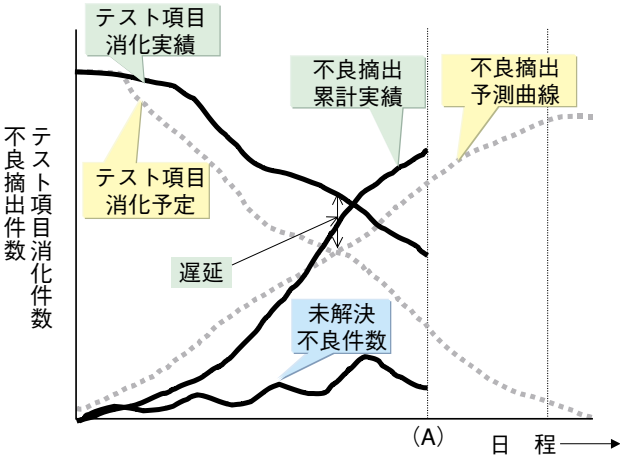


図 3.3　テスト工程管理図

とが重要である。さらに，テスト工程管理図には，未解決不良の増加を把握するために，未解決不良件数をプロットしている。

プログラムの品質管理では，仕様変更管理票(C票)，障害管理票(B票)，および，プログラム変更票(P票)で管理をすることもポイントである。これらの管理により，プロジェクト途中での再テスト計画を導き出す場合にも活用できる。

3.4.2　再テスト計画

図 3.3の事例では，日程(A)時点で，テスト項目の消化は遅れていて，不良は多く発生している状況である。考えられる問題要因としては，テスト項目の量と質が不十分，前工程での不良摘出不足，仕様変更・追加による不良の作り込みが考えられる。不良を作り込んだ該当工程を分析し，適切な対策を講じる必要がある。例えばもし，テスト項目の質が不十分である場合には，テスト項目の追加および見直しなどの品質向上施策を早く講じる必要がある。

3.4.3　不良摘出予想と異常値管理

テスト工程が進行し，時系列の不良摘出累積件数の実績が出てきたら，定期的に不良摘出予測曲線を設定して，不良摘出予想を行う。不良摘出累積件数の実績が不良摘出予想の上限と下限の 2 つの管理曲線を超えた場合は，その原因を分析して，適切な品質向上施策を講じることがきわめて重要である。テスト工程の後半では，不良摘出累積件数の実績に基づいて不良摘出予測曲線の収束状態を監視すると同時に，重要不良の発生状況も十分に監視する必要がある。

3.5　プロジェクト自身の実績の蓄積と活用

弊社で実施しているプロジェクト自身の実績データを活用した品質管理の事例として，品質マップによるプログラムの早期品質確保の事例を紹介する。品質マップとは，縦列に現象(プログラムの機能種別)を，横列に原因(技術要素)などでマトリクスで表示して，不良摘出の実績データを記入したものである。この品質マップの目的としては，単体テストの十分性や不良の収束性を確認することにより，追加テストの方針を導き出すことである。

3.5.1　品質指標による品質評価方法の課題

3.4節で紹介した品質管理の手法では，品質指標の目標値は，過去の類似プロジェクトの品質指標の実績値，対象プロジェクトの特性を考慮して設定する。そして，品質指標(テスト項目密度と不良摘出密度)の目標値と実績値を比較して，テストの十分性，不良の収束性を見きわめる手法である。しかし，プログラムの難易度，開発者のスキルなどの要因により，品質指標の目標値と実績値の乖離が発生してしまう。したがって，品質指標の目標値と実績値を比較するだけではテストの十分性や不良の収束性の評価は難しいのが現実である。テスト工程の早期段階で品質を可視化し追加テストの方針を導き出すことが，ソフトウェアエンジニアリングの課題の１つであった。

3.5.2　品質マップによる分析とその評価結果

プログラムの品質は，プログラムの難易度，開発者のスキルなどの要因に依存することに注目した。縦列の現象はプログラムの難易度を考慮した機能種別ごとに，横列の原因は設計時に必要とする技術要素ごとに分類し，現象別・原因別の品質マップを作成した。**図 3.4**に示すように，単体テスト完了後で摘出した不良を現象別・原因別で品質マップにて集計し，どんな現象がどんな原因で不良が発生しているのか，発生していないのかという傾向から，テストの十分性，不良が

原因 / 現象	規約		定義・設定			編集			出力処理 SQL					判定 処理抜け カウンタ操作				仕様誤り		
	C1	C2	T1	T2	T3	T4	T5	T6	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	S1
画面表示形式不正	6	2	16	2					3											
画面項目出力不正	2		5						2				3			1				
表示不正	4		3				2	2	3	3						3				
結果不正	1		1																	
エラーチェック不正		1	1						1											
画面遷移不正				1					1											
異常終了													3							
操作不能																				

図 3.4　現象別・原因別の品質マップ(事例)

摘出しきれているかを分析した。品質マップを作成することにより，不良の偏りと摘出不足の可能性がある範囲が明確となった。

不良摘出の状況とプログラムの機能種別，技術要素を考慮した現象・原因の組み合わせでまんべんなく不良が摘出されているかの観点で評価した結果を**表 3.2** に示す。この評価結果より，単体テストの後に，組合せテスト前に品質向上施策として追加テストを実施する必要があると判断した。この不良傾向の評価結果を考慮して，当該プログラムの品質指標の目標値の再設定，テスト項目の作成を行い追加テストを実施した。

表 3.2　品質マップの評価結果（事例）

不良の傾向	評価結果
不良の偏り **・画面表示形式・項目出力不正** **・表示不正**	画面形式や編集処理などの不良に偏っており，判定処理などのロジックに起因した不良が摘出されていない（テスト項目の十分性を確認）。
摘出不足 **・エラー処理不良が少ない**	エラーチェック処理に関する摘出不足 結果不正の不良の摘出不足

3.5.3　品質マップを活用した結果

この結果として，単体テストの後に，品質向上施策として追加テストを実施したので，組合せテスト開始は遅延したが，組合せテスト以降はスムーズに工程が進んで納期を守ることができ，納入後は安定稼働することができた。結果として，品質マップ評価による追加テストを実施したことで，単体テストトータルで最終不良摘出累計件数の73％の摘出ができた。これは，弊社の統計上の成功プロジェクトの実績値に相当する効果となった。品質マップを適用していなければ，最初の単体テストでは，最終不良摘出累計件数の38％しか摘出できていなかった。

この事例で示すように，現象別・原因別で不良摘出の実績データを品質マップで分析することにより，単体テストの十分性や不良の収束性を可視化した分析が可能となる。また，品質向上施策としてテスト工程の早期段階で必要な追加テストの方針を明確化することが可能となる。

3.6　当該取り組みの課題

プロジェクト自身の現象別・原因別などで不良摘出の実績データを蓄積して，品質マップで分析することにより，テストの十分性や不良の収束性の評価のレベルを向上させたい。さらに，品質マップの評価ノウハウを蓄積していくことにより，残存不良の予測精度を向上させたい。

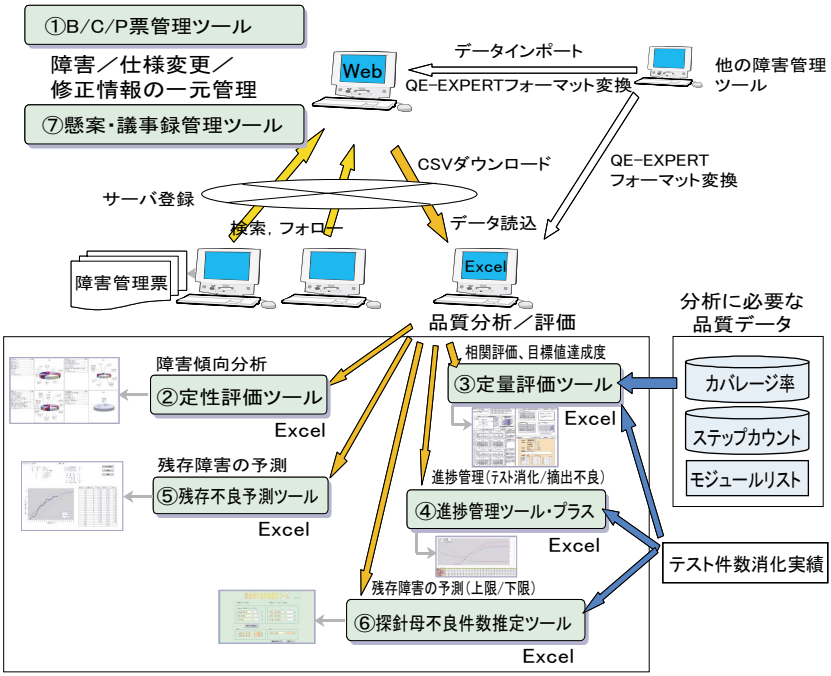
また，弊社では，品質指標による品質管理，テスト工程における品質データを収集し分析するツールとして，「ソフトウェア品質管理アプリケーションQE-EXPERT(QEエキスパート)」を開発し，適用を推進している(**図 3.5**)。今後も品質分析・評価の精度向上のためにツールの改善を引き続き図っていく予定である。

設計工程においては，インプロセス品質コントロールによる品質の作り込みを図ると同時に，定量的な品質管理による品質の可視化のレベルを向上することによる高品質・高生産な開発プロセスを実現していくことが課題である。品質を確保するためには，ユーザと設計者，設計者と開発者のインターフェース部分で十分なコミュニケーションをすることが重要であり，インターフェース部分でのコミュニケーション品質を向上するような施策を考えていきたい。

【1】総合的品質管理支援アプリケーションの概要

総合的品質管理支援アプリケーションQE-EXPERT（Quality Evaluation-EXPERT）は品質管理を中心としてプロジェクト管理を支援するツール群の総称です。管理ツール2種類，分析ツール5種類，計7種類のツールを用意しています。

＜QE-EXPERTを使った品質管理図＞



＜QE-EXPERTアプリケーション構成＞

No	総合的品質管理アプリケーション QE-EXPERT	内　　容
①	B/C/P 票管理ツール	障害管理票（B 票），仕様変更管理票（C 票），プログラム変更票（P 票）を登録，参照，検索する Web ツールです。
②	定性評価ツール	B/C/P 票データから不良傾向分析，クロス集計など定性的な面から品質評価を行うツールです。
③	定量評価ツール	不良件数，テスト項目数，カバレージデータの品質目標値に対する達成度評価や相関評価など定量的な面から品質評価を行なうツールです。

図 3.5　QE-EXPERT/ソフトウェア品質管理アプリケーション（1）

④	進捗管理ツール	不良摘出解決状況管理図（バグ死滅曲線）により工程の進捗管理を行うツールです。
⑤	残存不良予測ツール	摘出した不良件数を時系列で見ることでプログラムに残存する不良件数を予測するツールです。
⑥	探針母不良件数推定ツール	サンプリングテストの結果からプログラムに残存する不良件数を算出するツールです。
⑦	懸案・議事録管理ツール	懸案と議事録を登録，参照，検索する Web ツールです。

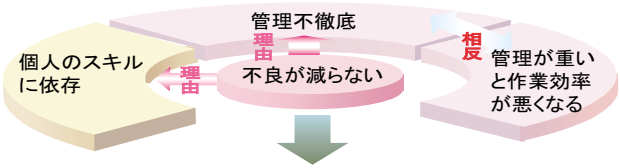
QE－EXPERTは日立製作所／品質保証部の品質管理／分析のノウハウを製品化したものです！

日立製作所／品質保証部の代わりにQE－EXPERTが品質分析を行ってくれます！

だれもが品質分析エキスパートになれます！

【２】導入の利点

QE-EXPERTを導入することで，様々な利点があります。



利点1 管理項目をユーザで自由に変更可能な QE-EXPERTで**最適な管理項目に設定**します

利点2 QE-EXPERTによるデータの一元管理、分析帳票の自動出力により、**作業効率向上**を図ります

利点3 調査中、対策済などステータス管理、承認などの再鑑管理、仕様変更管理が容易にできる QE-EXPERTで**管理精度の向上**を図ります

利点4 品質管理／評価方法を標準化、品質評価帳票を定型化しているQE-EXPERTを導入することで、誰もが**精度の高い品質評価**を可能とします

利点5 QE-EXPERTは品質管理・評価のノウハウそのもの。品質管理／品質評価**スキルの向上**が期待できます

【３】アプリケーションの特長

QE-EXPERTには以下のような特長があります。

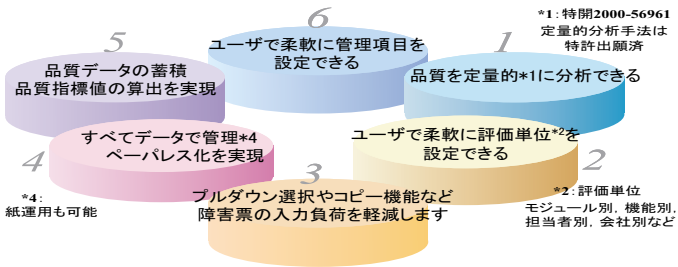


図 3.5　QE-EXPERT/ソフトウェア品質管理アプリケーション（２）

第 4 章　東京海上日動システムズの事例
～テスト品質と障害発生状況の分析～

4.1　取り組みの背景

弊社で開発・運用しているシステムは，保険会社の業務全般を担うだけでなく，全国の数万店の代理店さんにもさまざまな機能をオンラインでご利用いただいている。そのため，システム障害が発生した場合は，社内だけでなく，広く社外・お客様まで影響を及ぼす可能性が高い。また，システムが停止するとビジネス自体が停止しかねず，経営リスクとなる危険性もはらんでおり，品質確保と適正な開発力の確保は経営サイドまで重要課題と認識されている。

従来，開発しているシステムのテストの実施状況，品質評価，テスト完了見きわめなどのため，サービスイン可否の判定には一般的な信頼度成長曲線を使用してきたが，以下の課題が見えてきた。

- 発生する障害件数の予測が難しく妥当性を検証できない。
- 仕上り品質をテスト完了報告時に評価していたため，テスト不足が判明した場合，時間的に対応が困難となるリスクがある。
- 残存する仕様変更，ペンディング事項の収束によりテストがノイズを受ける。
- 上記の要因により，当初の見積りからの変動が最後の局面で顕在化し，プロジェクトのコスト・期限などの超過が発生するリスクがある。

上記の課題を解決するため，テスト工程のなかで中間評価を実施することで，品質を低下させる要因の芽を早期に発見し，対策の立案，対応要員・期間の確保などの調整を行うことで，より高い品質を確保することが必要との認識を持った。さらに，中間評価のタイミングで，要件変更・ペンディング状況などの残作業も明確にし，開発チーム内外と調整を行うことで，当初見積りからのブレを早期に確認し，是正できる仕組みを目指すこととした。

4.2節では，弊社プロジェクトでのソフトウェアテストの品質と生産性や本番での障害発生状況に関する分析結果を紹介する。

4.3節では，ソフトウェアテストの品質評価とテスト工程の見積り事例を紹介

する。

4.4節では，テスト工程での品質管理と再見積りのポイントを紹介する。

4.2　ソフトウェアテストと障害発生状況についての分析

弊社で実際に開発した案件について，テスト工程の品質（障害発生状況）と作業量や期間を分析した結果，プロジェクトごとに状況は異なるものの，テスト工程での品質と作業量・期間にはある程度の相関関係があることが明らかになってきている。

4.2.1　システムテスト工程と品質

結合テスト，システムテストの障害密度と生産性，本番での障害発生度合いとの関係を，平均以上・以下で4分類（**表4.1**）してみると，以下の結果となった。

- 結合テスト，システムテストともに平均以下のグループは，全体の生産性が高く，サービスイン後の障害は少ない。
- 一方，結合テストは平均以下だが，システムテストで平均以上の障害が発生するグループが，全体の生産性が最も低く，サービスイン後の障害も多い。

表4.1　テスト工程の障害密度分類と生産性・本番障害密度の関係

テスト工程の障害密度分類	合計生産性	本番障害密度
結合テスト：少，システムテスト：少	100%	100%
結合テスト：多，システムテスト：少	81%	201%
結合テスト：多，システムテスト：多	76%	319%
結合テスト：少，システムテスト：多	56%	409%

結合テスト・システムテストともに平均以下のプロジェクトを100%とした場合の生産性，本番障害密度の割合を示したもの。

この結果，システムテストで多くの障害が発生するプロジェクト，すなわち結合テストの完了時点で品質が悪いプロジェクトは，サービスイン後も含めてその後の工数や費用が多くかかる傾向が見える。システムテストの開始前に，結合テストの品質を確認・評価することは非常に重要なことである。

しかし，結合テストの終了後，システムテスト開始前に，結合テストまでの品質を詳細に検証する時間を取ることは，スケジュール的にも工数的にも難しいことが多い。システムテストの前半（遅くとも中間時点）には障害発生状況などを踏

まえ，品質をしっかり評価するとともにテストケースの追加や要員の追加投入などの対策を立てることがきわめて重要である。

4.2.2　システムテスト工程と生産性

さらに，テスト工程の品質と生産性の相関を分析してみると以下の結果となった。

- システムテスト工程の費用(工数)は開発全体の30～50％を占める(**図 4.1**)。
- テスト工程での品質が低いものほど，全体の生産性が低い傾向がある(**図 4.2**)。

図 4.1は，表 4.1の 4 分類のうち，最も本番障害が少ないケースと，最も本番障害が多いケースについて工程別の工数割合を分析したものである。

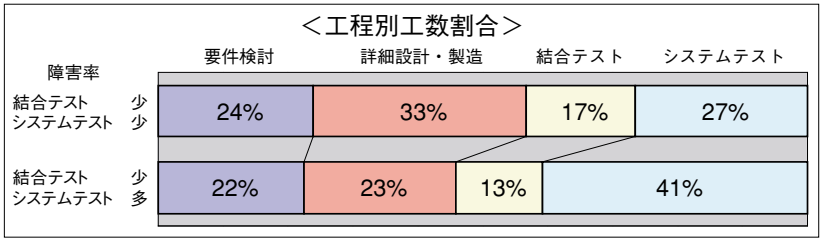


図 4.1　プロジェクトの工程割合と障害率

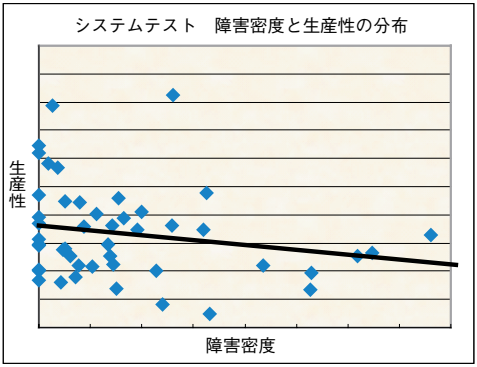


図 4.2　システムテストの生産性と障害密度

図 4.2は，システムテスト工程での障害密度と，合計生産性の相関を表したものである。システムテスト工程での品質が高い(検出障害が少ない)ものは，テスト工程の工数割合が少なく，生産性も高いという結果となった。

テスト工程の効率化と品質対策はあわせて考えていく必要がある。また，テスト期間の短縮と品質は必ずしもトレードオフという関係でなく，テストの内容・実施方法・管理方法により改善すべき問題であることがわかる。

4.2.3　開発期間の予実績と品質

開発期間全体の予実績で，予定より期間が長くなったもの・ほぼ同じ・少ないものに 3 分類して工程別期間の割合の予実績とシステムテストの障害密度の相関を分析した(**表 4.2**)。工程別期間の予定はほぼ同じであるが，期間延長となったものは，要件検討とシステムテスト期間の実績が予定より長くなり，工程の重なりも多いことがわかる。

さらに,期間延長となったものはシステムテスト工程での障害摘出密度も高い。

表 4.2　合計期間に対する工程別期間の割合予実績と，システムテストの障害密度

	期間短縮 (予定)	期間短縮 (実績)	ほぼ予定どおり (予定)	ほぼ予定どおり (実績)	期間延長 (予定)	期間延長 (実績)	平均 (予定)	平均 (実績)
要件検討	33%	31%	29%	34%	30%	43%	30%	34%
設計・製造	38%	34%	37%	38%	35%	30%	37%	37%
システムテスト	40%	36%	39%	38%	40%	45%	40%	38%
のべ期間※ 1	111%	102%	106%	111%	106%	117%	107%	109%
障害密度※ 2		85%		102%		170%		100%

※ 1 ：のべ期間：各工程ごとの割合の合計。100%より大きいほど工程の重なりが多い。
※ 2 ：障害密度(システムテスト)：全体を100%とした場合の開発規模あたり障害検出割合。

4.2.4　全体として

当初の見積り時は平均的な工数で計画することが多いため，テスト工程の品質が悪いものほど，プロジェクトの最終局面で工数超過・費用超過が発生しやすく，プロジェクト管理上のリスクが高まる。システムテスト工程は全開発工程の中で大きなウェイトを占める部分であり，全体の見積・実績に影響が大きい。

テスト工程での品質の低下の回避の取り組みはもちろん重要であるが，ばらつきのあるプロジェクト 1 つずつについて，品質低下の兆候を早期に把握し，その

後の計画見直し(見積りの見直し)につなげ，プロジェクトを成功に導くことの重要性があらためて明らかになった。

4.3　ソフトウェアテストの品質評価とテスト見積りの取り組み

4.3.1　品質評価報告書の導入

テスト工程の評価を多面的に行うため，日立製作所のご協力をいただき，いわゆる信頼度成長曲線と 8 つの指標の組み合わせで評価を行う「品質評価報告書」(**表 4.3**，**図 4.3**)を導入し，テスト計画と評価に活用している。

テストの評価を担当者任せにした場合，極端なケースでは思いついたテストケースをすべて消化すれば終了としてしまう可能性もある。それぞれの指標自体は一般的なものであるが，定型的なシート(「品質評価報告書」)でグラフ・数値を用いて計画・評価を行うことで，テスト計画と評価のレベルアップを図ることとした。

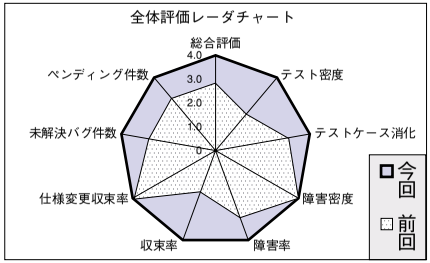
「品質評価報告書」をベースとして，さらに，障害検出工程や障害発生工程の分析を行い，実施済みのテスト工程の評価を行うようにしている。

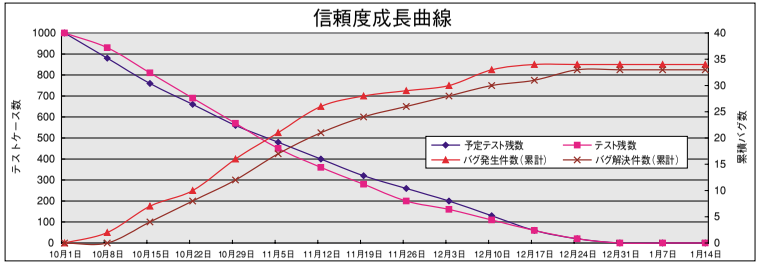
表 4.3　8 つの評価指標

評価指標	指標の意味
テスト密度	テストケース数/開発規模(Kstep)。 テストケース設定の十分性を評価。
テストケース消化	テストケース残消化数/総テストケース数。 テストケースの消化見通しを評価。
障害密度	摘出障害件数/開発規模(Kstep)。 問題摘出の過不足を評価。
障害率	摘出障害件数/テストケース数×100。 テストケース設定内容の妥当性を評価。
収束率	摘出障害件数/推定障害総数×100。 問題摘出十分性を評価。
対応必須の要件変更残存数	要件追加・変更未解決総数/総発生件数。 仕様FIX度を評価。
未解決バグ数	未解決障害件数/摘出障害件数。 サービスインへの影響評価。
ペンディング件数	未解決懸案事項件数/総懸案事項件数。 サービスインへの影響評価。

タイトル

会社名		案件名	ああああああああ	規模（Kstep）	25.0	作成日	
所属名		本番予定日	2008年2月1日	工数（人月）	25.0	開発主担当	フルネーム
案件Ｎｏ	XXXXXXXX	子案件Ｎｏ	XXXXXXXYY	発注Ｎｏ	ZZZZZZ	開発主担当 TEL	（Tel：1234）
全体評価		信頼度成長曲線					





項番	評価指標	評価点 前回 中間品質評価 X月X日実施	評価点 今回 最終品質評価 X月X日実施	評価指標 指標	評価指標 実績値	見　解
1	テスト密度 テストケース数／Kstep。テストケース設定の十分性評価	2	4	xx～xx件／Kstep （システム毎に設定）	ST 34.0 （件／Kstep） OT 6.0 （件／Kstep） 合計 40.0 （件／Kstep）	・当初のケースの挙げ方では機能による偏りがあったため、中間評価時に見直しを行った。その結果、テスト密度は標準実績値より高くなったが、十分なテストが出来たと評価する。
2	テストケース消化 テストケース数消化件数。消化見通しを評価	3	4	中間：残数／総数（残40%） 最終：残数（0）／総数	残数 ／ 総数 残割合 0 ／ 1000 0%	・中間評価時は想定より早いペースでテストケース消化していたが、障害摘出率が低かった。テストケースの挙げ方に問題があると判断し、ケース見直しを実施し、ケースを追加した。残ケースは大量テスト、性能テスト、プレ本番環境でのリモメン対象画面の確認のXケースであり、現在最終確認中。X月X日に完了予定。
3	障害密度 障害件数／Kstep。問題摘出の過不足を評価	4	4	ｘｘ～ｘｘ件／Kstep （システム毎に設定）	ST 1.2 （件／Kstep） OT 0.2 （件／Kstep） 合計 1.4 （件／Kstep）	・平均値より少ないが、単純改訂が多いためであり問題ないと判断する。 ・平均値より多いがテスト初期で環境不備によるものが多発したため。バグ数としては平均的。 ・平均値より多いのはXX機能の品質不足と判断し、品質向上施策を追加実施し改善。
4	障害率 障害件数／テストケースX100。テストケース設定内容の妥当性評価	3	4	ｘｘ～ｘｘ％ （システム毎に設定）	ST 3.5% OT 2.7% 合計 3.4%	・中間評価時は障害率が1%と低かったが、テストの観点を見直しケースを追加してテストを実施した結果、標準的な数値となった。 ・障害発生率が低いためテストケースを見直したが、内容・範囲とも問題なし。単純改訂によるものと判断。
5	収束率 累積摘出件数／推定障害総数（摘出目標値）X100。問題摘出十分性評価	2	4	中間：６０～７０％ 最終：１００％	ST 100.0% 目標値→ 30 OT 80.0% 目標値→ 5 合計 97.1% 最終週バグ数 0	・テストケースの消化に応じて障害検出も収束しており、問題はない。 ・X月X日の週前後に山があるが、テストケース内容の見直しを行ったためであり、問題ない状況と判断する。 ・バグ摘出目標値はⅡ期実績値2.0件/Ｋｓ×開発規模とした。
6	仕様変更収束率 （対応必須の要件変更残存数） 要件追加・変更未解決件数／総発生件数。仕様FIX度評価	4	4	中間：残数／総数 最終：残数（０）／総数	残数 ／ 総数 残割合 0 ／ 12 0%	・オーナテスト時に帳票のレイアウトについて、仕様変更が発生。全体への影響は少ないと判断し、対応を行い完了しており、問題はない。
7	未解決バグ件数 残対策問題点件数／総発生件数。Ｃ／Ｏへの影響評価	3	4		残数 ／ 総数 残割合 1 ／ 34 3%	・未対応の障害はない。 ・未対応の障害は１件あるが、本番稼働はＣ／ＯのＳＷ後であり、対応は完了する見込。 ・画面項目の誤字があるが、社内利用であり次回対応でオーナ了解済み。
8	ペンディング件数 未解決懸案事項／総発生件数。Ｃ／Ｏへの影響評価	3	4		残数 ／ 総数 残割合 1 ／ 3 33%	・ペンディング事項は別紙記載のX件。いずれもＣ／Ｏ直後には影響なし。
9	総合評価 上記評価項目の平均	3.0	4.0			・テスト内容、障害検出状況ともに問題ないと判断する。 ・テスト段階でXX機能に要件変更がいくつか発生し、その分障害の収束も遅れた。追加テストを実施し、問題ないレベルに達したと判断する。

（中間品質評価・スポット品質評価）　4点：このままテストケースを消化し品質確保可能　3点：品質確保可能だが若干の軌道修正が必要　2点：品質確保不安有り。追加テスト要　1点：品質確保不十分。工程見直し要

（最終品質評価）　4点：品質確保十分。懸案無し　3点：品質確保したが調整懸案有り　2点：品質確保不安有り。追加テスト要　1点：品質確保不十分。やり直し

図4.3　品質評価報告書

4.3.2　評価のタイミング(図 4.4)

（1）　テスト計画時

前工程までの品質状況と今後の見込を勘案し，予定数を見積もる。予定ケース数や障害予測をふまえて，テストの期間・工数の見積りを確認し，必要であれば見直しを行いテスト計画レビューを実施して承認を得る。

前工程までで特に品質が低いところや，要件変更・追加が多いところは，今後のテストでの障害件数も増加することが見込まれるため,特に配慮が必要である。

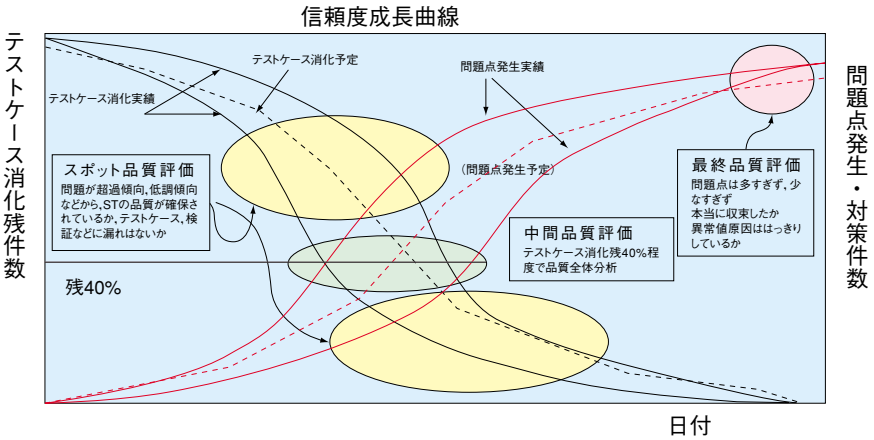


図 4.4　評価のタイミング

（2）　中間評価

テストケースを50～60%程度消化した時点で,テストの進捗や品質を評価する。想定以上に障害が検出されている，逆にほとんど障害が検出されないなど，プロジェクトによって，この時点でかなりばらつきが発生することが多い。

後戻りのリスクを軽減するとともに，過剰な分析ロードがかからないよう，ある程度テストの状況が見えたところで評価を行うこととしている。

（3）　スポット評価

テストの進捗状況が悪い場合，想定外に障害が検出される場合にはさらにスポット評価を行う。長期にわたるプロジェクトなどでは，進捗確認時に障害検出状況も確認し，プロジェクトの見積りへの影響が懸念される場合には，早急に評価・対応を行うことが必要である。

（4）　最終評価

中間評価と同様に実施するが，テストケースは100%消化，障害も収束し，対応も完了していることが必要。

4.4　テスト進行中における品質管理と再テスト見積りのポイント

4.4.1　中間評価

テスト工程の見積りにおいては，特に中間評価が特に重要なポイントとなる。中間評価におけるポイントは以下の 3 点である。これらを総合的に判断し，必要であれば追加テストの実施や，前工程へ戻ってのデバッグなどを実施するため，テスト計画の見直し(見積り見直し)を行う。

（1）　信頼度成長曲線による評価

信頼度成長曲線および諸指標をもとに，各担当(チーム)が状況分析し，今後の見通しを明らかにして対策を立てることが重要である。

- テストケースは十分であるか(計画時の指標値の見直しも行う)
- このまま進めて品質は確保できるか
- テストツール・要員・スケジュールの見直しは不要か
- ペンディング・要件変更はどの程度残存するのか

図 4.5に信頼度成長曲線の評価例を示す。

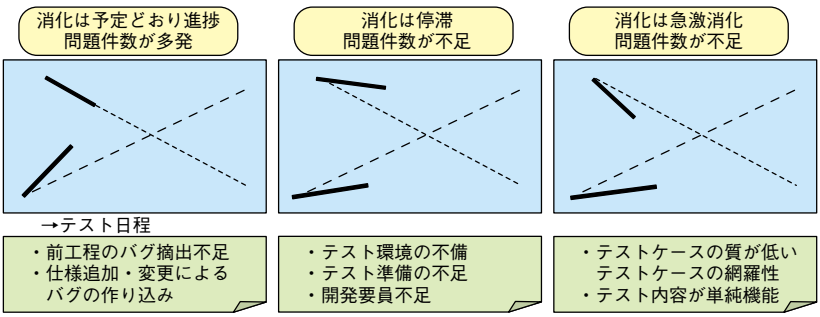


図 4.5　信頼度成長曲線の評価例

（2）　8 つの指標に対する概略評価(4 段階評価)

4 段階評価を定量的＋定性的に行う。

状況をチーム・組織にわかりやすく伝えるために，評価指標ごとに，**表 4.4**の

表 4.4　概略評価

評価点	内　　容
1	品質確保不十分，工程見直し要
2	品質確保に不安あり，追加テスト要
3	品質確保可能だが，若干の軌道修正が必要
4	このままテストケースを消化し，品質確保可能

ような概略評価を行う。

（3）　発生した障害の原因分析

単なる数字だけでなく，発生した障害の内容(本来発見すべき工程，障害を作り込んだ工程，発生場所の偏り，特定の担当者に集中していないかなど)も分析する必要があり，障害管理シートの起票・蓄積が必要である。

また，場合によってはテストを中断し，前工程のテスト内容の再確認，仕様書・プログラムの机上デバッグを行うことも有効な対策である。

表 4.5に障害の原因分析のポイントを示す。

表 4.5　障害原因分析のポイント

発生工程	障害発生のテスト工程(単体テスト，結合テスト，システムテストなど)を分類し，工程にあった障害が出ているかを分析する。
問題分類	障害を摘出した場所(帳票，画面，データチェックなど)を分類することにより，品質向上ポイントを推測する。
影響度	対策の優先度を決めるための情報。影響度の高い傾向の障害が多い場合は，基本的な要件定義が不十分である場合が多く，要件定義の再確認など，品質分析にも役立てる。
原因分類	障害を作り込んだ工程を明らかにする。 例えば，コーディング誤りが多すぎないか，弱い工程はどこか，作り込み工程に偏りがないかなどを分析する。
不備未発見理由	前工程までに発見できなくてもやむを得ない障害なのか，なぜ，いままで不備が発見されずにきたのか，不備未発見の原因を明らかにし，見直しポイントを検討する。
原因主体	不備を作り込んだ主体はどこにあるのかを明らかにする。 要件定義に問題があるのか，設計に反映されてないのか，特定の担当者に偏っていないかなどを見て，見直しポイントを検討する。

4.4.2　最終評価

評価内容は中間評価と基本的に同じであるが，この時点ではテストケースは100%消化され，障害も収束し，対応も完了している。ペンディング事項はない，もしくは解消の時期が明確になっており，関係者で合意が取れていることが必要である。

最終評価では，**表 4.6**のように各指標に対して，見積りや標準値との差異について，合理的な差異であるか，サービスインするに足る品質であるかを定性的に分析を行うことがポイントである。

差異があることは必ずしも問題ではなく，なぜ差異が発生したか，もしくは想定範囲に収まった理由は何かを考察し，説明できることが重要である。

表 4.6　最終評価

評価点	内　　容
1	品質確保不十分。やり直し
2	品質確保不安有り。追加テスト要
3	品質確保したが調整懸案あり
4	品質確保十分。懸案なし

4.5　今後の取り組み課題

上述したように，弊社では特にウェイトの高いテスト工程に焦点をあて，テスト計画・テスト品質管理のレベルアップを行い，見積りと実績の差異や品質との関連についてデータの蓄積・分析を行ってきたが，解決すべき課題はまだまだ多い。

① 評価指標に対する基準値があいまい。特に改良保守のプロジェクトでは，母体となるシステムやプロジェクトの特性によって大きく異なるため，一律の基準値をベースに評価を行っても適切でないケースが多い。同種のプロジェクト，システムごとの基準値の策定が必要である。

② システムテストの中間評価時点で再見積りが必要となっても，プロジェクトの予算や時間の制約をかなり受ける。分析したデータでは品質に問題があるプロジェクトは，システムテストの工数がより多くかかり，期間も延びる傾向にある。さらにサービスイン後の障害率も高いため，QCDともに影響を受けるという結果がでている。もっと早期に問題を検出できる仕

組みが必要である。

③ テスト計画の策定ではテストケース数や予想摘出障害件数を考慮してテストロード・テスト期間を見積もるが，必要十分なケース数や障害件数予想は母体となるシステムの特性や，当該プロジェクトの前工程までの品質によって大きく左右される。この影響要因を考慮した見積り方法については，今後のデータ蓄積と分析が必要である。

④ 「品質評価報告書」は主にシステムテスト工程の評価に使用しているが，単体テストや結合テストでの評価に活用できるよう，指標および基準値の策定が必要である。

第5章　ジャステックの事例
～ソフトウェアの生産管理に基づくテスト見積り～

5.1　取り組みの背景

弊社では，創業以来，役務提供を前提とした人月単価から脱却し，一括請負契約を拡大すること，および生産性原理に立脚した能力主義を実践するために独自の生産管理システム「ACTUM」を構築し運用している。生産管理システムの中心がソフトウェア開発の見積りモデルである。その見積りモデルは，2006年4月発刊のSEC BOOKS『ソフトウェア開発見積りガイドブック』の「第9章ジャステック手法」および2007年10月発刊の『ソフトウェア改良開発見積りガイドブック』の「第3章ジャステック」で紹介した。受託ソフトウェア開発の価格は開発量と正の相関関係にあることを前提としており，開発量はソフトウェア開発工程ごとに作成する生産物の量として可視化している。

見積りモデルの基本アルゴリズムは，生産物量見積り方式および生産性見積り方式から成り立ち，さらにおのおのの方式において開発環境の違いや品質要求の多寡による変動を吸収する「環境変数」と呼ぶパラメータを導入している。本章ではソフトウェアのテストに焦点をあてて，テスト見積り，ソフトウェアテストの管理およびソフトウェアテスト効率の改善などに対する弊社の取り組みを紹介する。

5.1.1　ソフトウェアテスト見積りに関する問題意識

ソフトウェア開発規模の増加とともに，複雑化するソフトウェア機能，実行タイミングおよび動作環境に呼応して，あらゆる条件を組み合わせたテストを行うと，テスト量は天文学的数字になるので，実施するテストケースを人為的に限定せざるを得ない。一方，ソフトウェアの出荷時に残存する欠陥を見積り時にユーザと合意したとしても，残存欠陥はテスト終了時には測れないため，代替尺度（テスト網羅率など）に基づく残存欠陥の把握に基づいて，その実現策を顧客と合意して，ソフトウェアテストを見積もることが必要と考えている。

弊社ではホワイトボックステストとブラックボックステストに分けて取り組ん

でいるが，本章ではホワイトボックステストに関するテスト見積りを中心に紹介する。

5.2　ソフトウェアテストへの取り組み

5.2.1　ソフトウェアテストのプロセス

弊社では，ソフトウェアテストをソフトウェアの品質管理における欠陥の検知活動の一環としてとらえて，**図 5.1**に示すとおり基本設計工程からテストプロセスを開始してソフトウェアの開発工程に合わせて各工程で実施するテストタスクを定めている。

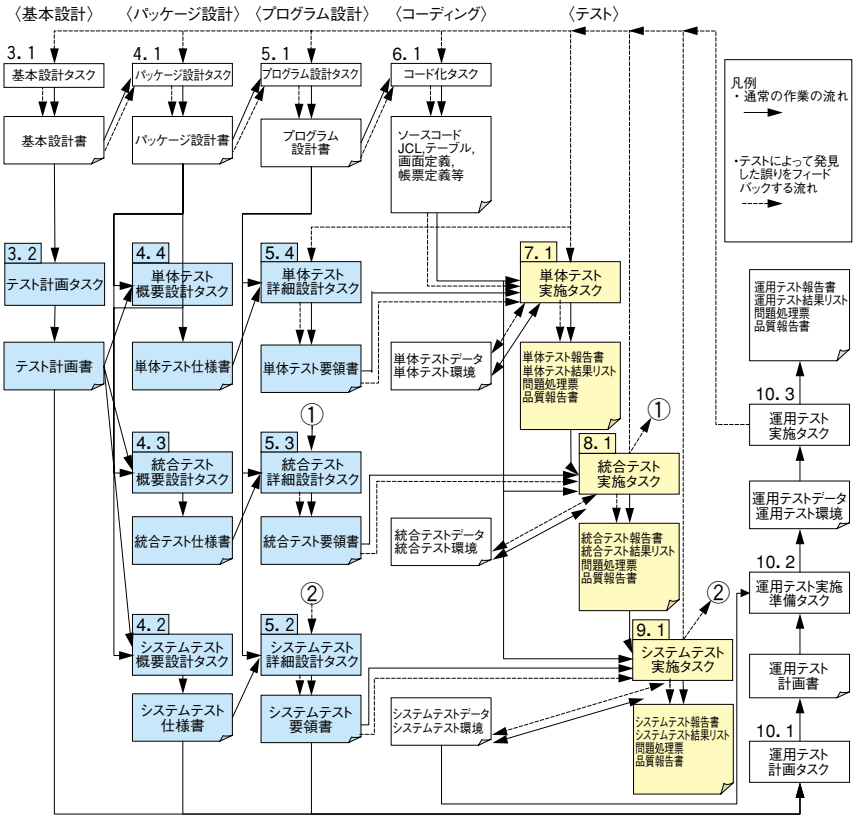


図 5.1　ソフトウェアテストのプロセス

基本設計工程では，出荷時の残存欠陥密度の目標およびテスト完了基準を設定して，設計レビュー，単体テスト，統合テストおよびシステムテストを含む欠陥検出戦略を作成するとともに，ソフトウェアテストの効率化およびテストの検証精度向上のためにテスト対象システムに要求する試験性などの要件を具体化し，顧客と調整合意して，ソフトウェアテストの見積りにインプットする。パッケージ設計工程では，テスト戦略を実現するテスト環境およびテスト作業標準を定め，プログラム設計工程でテストケースを定めるとともにテストの手順書を作成し，コーディング工程でテストデータおよびテスト環境を実装して，テストを開始する。

ソフトウェアのテスト完了基準は，テスト網羅率，信頼度成長曲線の収束度合，修正未了の欠陥数，テスト項目の消化率，未解決・懸案件数および検出した欠陥の定性的な傾向などを総合的に判断することとして，判断に関与する関係者，判断におけるそれぞれの優先順位などを定める。また，判定はプロジェクト全体はもとより，構成品別や機能別など層別にも評価することを定めている。

5.2.2　ソフトウェアテスト見積りに適用する生産物

弊社では，5.2.1項に述べたソフトウェアテストのプロセスに基づいて，ソフトウェアテストの見積りに適用する生産物量を**表5.1**に示すとおり工程ごとに定めている。プログラム設計工程以降のテスト量は，テスト対象のソフトウェアの規模ではなく，テスト対象のソフトウェアから欠陥を除去するために設定し，実施するテスト項目数を採用している。

なお，‘＊’についての生産物量はテスト項目数を媒介変数として求める。テスト項目数の求め方については，5.3.2項(2)で述べる。

表5.1　ソフトウェアテスト見積りに適用する生産物

工　程	生産物	見積りに適用する生産物量	
基本設計	テスト計画書	左記ドキュメントの量	Kc
パッケージ設計	単体テスト仕様書	左記ドキュメントの量	Kc
	統合テスト仕様書	左記ドキュメントの量	Kc
	システムテスト仕様書	左記ドキュメントの量	Kc
プログラム設計	単体テスト要領書＊	左記ドキュメントの量	Kc
	統合テスト要領書＊	左記ドキュメントの量	Kc
	システムテスト要領書＊	左記ドキュメントの量	Kc
コーディング	単体テストデータ＊，環境設定データ＊	左記データの量	Kds
	統合テストデータ＊，環境設定データ＊	左記データの量	Kds
	システムテストデータ＊,環境設定データ＊	左記データの量	Kds
単体テスト	単体テスト報告書＊	左記ドキュメントの量	Kc
統合テスト	統合テスト報告書＊	左記ドキュメントの量	Kc
システムテスト	システムテスト報告書＊	左記ドキュメントの量	Kc

（注）　Kc（文字数），Kds（データ行数）

5.3　ソフトウェアテストの見積り

5.3.1　弊社の見積り基本アルゴリズム

弊社が考案したコスト見積りモデルの基本アルゴリズムを示す。本アルゴリズムは，SEC BOOKS『ソフトウェア開発見積りガイドブック』および『ソフトウェア改良開発見積りガイドブック』で紹介した。その中で，ある開発工程 i の標準生産物量をV_i^B，標準生産性をP_i^Bで表現すると，生産物量をV_i，生産性P_iおよび開発コストC_iは次式で求まることを述べた。

$$V_i = V_i^B \times (1 + \delta_i) \times (1 + a_i + a_i') \qquad (a_i = \Sigma a_{ij},\ a_i' = \Sigma a_{ij}')$$
$$P_i = P_i^B \times (1 + \gamma_i) \times (1 + b_i + b_i') \qquad (b_i = \Sigma \beta_{ij},\ b_i' = \Sigma \beta_{ij}')$$
$$C_i = V_i \times P_i$$

a_iおよびa_i'は生産物量環境変数と呼ぶパラメータであり，V_i^Bに対して品質要求の多寡などによる変動を吸収する変数であり，a_i'は改造開発特有の変数であ

る。また，b_i およびb_i' は生産性環境変数と呼ぶパラメータであり，P_i^Bに対して開発環境の違いや品質要求の多寡などによる変動を吸収する変数であり，b_i' は改造開発特有の変数である。a_i，a_i'，b_i，b_i' は品質特性と環境特性から影響される独立した変動要素(α_{ij}，α_{ij}'，β_{ij}，β_{ij}')から構成されている。生産性環境変数の具体例を**表 5. 2〜表 5. 4**に，生産物量環境変数の具体例を**表 5. 5**および**表 5. 6**に，それぞれ記載している。δ_iはテスト工程の改造開発生産物影響度で，新規開発の場合はもとより,改造開発の場合でも基本設計からコーディング工程の値はゼロである。γ_iは基本設計からコーディング工程の改造開発生産性影響度(「改造密度・改造分散・改造母体練度変数」とも呼ぶ)で，新規開発の場合はもとより，改造開発の場合でもテスト工程の値はゼロである。

なお，テスト工程については，テスト対象のソフトウェア規模として，改造部分とインターフェースを介して巻き込むテスト巻き込み規模(「インターフェース規模」と呼ぶ)とレベルダウンを防ぐために巻き込むテスト巻き込み規模(「リグレッション規模」と呼ぶ)を考慮する。すなわち，インターフェース規模(改造正味規模を含む)をV_i'，リグレッション規模をV_i''とすると，それぞれ次式で求まる。

$$V_i' = V_i^B \times (1 + \delta_i) \times (1 + a_i + a_i') \qquad (a_i = \Sigma\alpha_{ij},\ a_i' = \Sigma\alpha_{ij}')$$

$$V_i'' = V_i^B \times (1 + \delta_i') \times (1 + a_i + a_i')$$

なお，δ_i'はリグレッション規模に対応する改造開発生産物影響度であり，改造正味規模に加えてリグレッションテストでテストすべきテスト巻き込み規模の改造正味規模に対する倍率である。

新規開発の場合は，巻き込み規模は存在せず(δ_iおよびδ_i'はゼロになる)になり，さらに改造開発特有の変数a_i'も存在しない。よってテスト対象のソフトウェア規模V_iは次式で求まる。

$$V_i = V_i^B \times (1 + a_i) \qquad (a_i = \Sigma\alpha_{ij})$$

表 5.2　環境特性変動要素評価表（生産性に影響する外部環境変数）

主特性	副特性	変動要素	ベースラインからの変動率（%）			
			要件定義	設計	製作	テスト
業務特性	業務ナレッジ	顧客の開発対象業務に対する業務ナレッジが生産性に及ぼす影響	−10 ～50	−10 ～10	—	−10 ～10
ハードウェア特性	安定度/信頼度/使用度	システムもしくは製品となるハードウェアの安定度・信頼度	—	−5 ～5	—	−5 ～5
ソフトウェア特性	安定度/信頼度/使用度	システム/製品に組み込む他社作成ソフトウェアもしくはCOTSの安定度・信頼度	—	−5 ～5	—	−5 ～5
コミュニケーション特性	顧客窓口特性	意思決定能力（期限遵守，決定事項の覆る度合）	−10 ～20	−10 ～10	—	−7 ～7
	工期の厳しさ	基準工期（月）＝2.7×（人月）$^{1/3}$に対し▲30%限度とした短期化度合	0～10	0～10	0～5	0～7
	コミュニケーション基盤	開発拠点分散，資料など情報共有，電子媒体・システム具備など物理的基盤充実度	−10 ～10	−5 ～5	−3 ～3	−3 ～3
	レビュー体制	無駄なレビュー（重複多段階など）の排除およびレビュー効率向上への工夫度合	−5 ～5	−5 ～5	−3 ～5	−5 ～5
開発環境特性	開発手法/開発環境	開発手法・環境（ソフト/ハード/ツール）の信頼性，占有率などを考慮した使用実績	−3 ～3	−3 ～3	−5 ～5	−5 ～5
	テスト手順書水準	テスト手順の具体化度（操作手順＆入出力の具体化の要求水準）	—	—	−3 ～3	−3 ～3
工程入力情報特性	業務関連資料	必要資料の具備状況（正確性，信頼性を含む）および使いやすさ（検索性，理解性）	−10 ～10	−7 ～7	−3 ～3	−3 ～3
	他システム関連資料	必要資料の具備状況（正確性，信頼性を含む）および使いやすさ（検索性，理解性）	−10 ～10			
	規約・標準化関連資料	必要資料の具備状況（正確性，信頼性を含む）および使いやすさ（検索性，理解性）	−7 ～7			
顧客の協力特性	役割分担特性	顧客がベンダ企業に協力する度合および顧客とベンダ企業との役割分担の明確性	−10 ～20	0 ～10	—	0 ～10

表 5.3　品質特性変動要素評価表(生産性に影響する外部環境変数)

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
機能性	合目的性(要求仕様の網羅性)	要求の記述水準および網羅性。要件定義については新規性，方針明確性，ステークホルダーの多様性などを考慮	0〜100	0〜30	—	0〜10
	正確性	正確性(検証)に関わる標準レビュー工数(各工程 8 %)を基準にした要求水準	0〜5	0〜5	0〜3	0〜5
	接続性	基準単位(100ks)に対する社内/社外システムとのインターフェース先の数	−10〜10	−5〜10	—	−5〜5
	整合性	整合をとる社内/社外の規格・基準の数，全体適合性やグローバル化対応も含む	−10〜10	−5〜10	−3〜5	−3〜5
効率性	実行効率性	実行効率に対する一般的要求水準(既知)の標準事例を基準にした要求水準	0〜5	0〜10	0〜5	0〜10
	資源効率性	資源効率に対する一般的要求水準(既知)の標準事例を基準にした要求水準	0〜5	0〜10	0〜5	0〜10
保守性	解析性	ソースコードの解析性をコード化規約に定めるコメントに対する要求水準により評価	—	—	0〜5	—
	安定性	ソフトウェア変更に対しシステム品質維持可能とする水準をライフサイクル目標年数の長さにより評価	0〜5	−3〜7	−3〜5	—
移植性	環境適用性 移植作業性 規格準拠性 置換性	ソフトウェアをどの程度，多様な環境に移すことができるかに対する要求の水準	0〜28	0〜28	0〜12	0〜25

表 5.4　現行資産特性変動要素評価表(生産性に影響する外部環境変数)

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
改造・再構築特性	母体調査ツールの機能水準	調査ツールの機能の数 ・絞込み ・モニタリング(ルート解析) ・ドキュメント生成(リバース) ・実機での稼働確認 ・データディクショナリ ・その他調査ツール	—	−20〜15	−10〜5	−20〜10
	既存設計書具備状態	改造/流用母体の設計書有無および設計書が存在した場合のメンテナンス状態	0〜10	0〜30	0〜30	0〜25
	既存のテスト環境流用水準	既存のドキュメントおよびテストデータ(テスト環境を含む)の流用可能度合	—	—	−20〜0	−30〜0
	リソース管理水準	ソースとロードモジュールのバージョン管理水準	—	—	0〜15	0〜15
	既存母体品質(正確性)	既存システムが正しく動作しない場合の生産性に及ぼす影響度	0〜8	0〜10	0〜10	0〜10
	既存母体品質(解析性)	既存システムの改造箇所特定(改造設計)における標準化項目の遵守度合によるソースコードの解析容易性 ・ソースコメント ・修正履歴 ・モジュールサイズ ・規約 ・その他顧客規約事項	0〜10	0〜25	0〜25	0〜15
	既存母体品質(環境適用性)	現行システム資産を別環境へ移植し改造する開発における，別環境への適用容易性(適用可能な環境の個数) ・ハードウェア ・OS ・ネットワーク ・フレームワーク(ミドル) ・DB/DC ・バッチ運用システム	0〜10	0〜25	0〜25	0〜15

※改造開発特有の環境変数

表 5.5　品質特性変動要素評価表(規模に影響する外部環境変数)

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
機能性	合目的性	利用者/利害関係者の広がりコンテンジェンシー対応，不正移行データ対応などの該当事象数	0〜50	0〜50	0〜50	0〜50
	正確性	正確性(検証)に関わる標準テスト密度を基準にしたテスト項目量への要求水準	—	—	0〜20	0〜50
	接続性	他システムとの接続によるコード変換，フォーマット変換数	0〜5	0〜20	0〜20	0〜20
	セキュリティ	対応が必要なセキュリティ実現機能数，ただし機能要件に定義されている部分は除く	0〜20	0〜20	0〜20	0〜20
信頼性	成熟性	故障低減に必要な実現機能数	0〜5	0〜10	0〜10	0〜10
	障害許容性	異常検知に必要な機能数	0〜5	0〜10	0〜10	0〜10
	回復性	再開処理に必要な実現機能数	0〜5	0〜10	0〜10	0〜10
使用性	理解性	理解性向上(機能など)のためのプレゼンツールなどの作成対象数	—	0〜10	—	—
	習得性	習得性向上(使い方など)のためのマニュアルなど作成対象数	—	0〜10	—	—
	操作性	操作性向上(心理的/肉体的配慮，運用やインストール容易性など)のための実現機能数	0〜10	0〜20	0〜20	0〜20
保守性	解析性	解析に必要な実現機能数	—	0〜10	0〜10	0〜10
	変更作業性	作成する保守用ドキュメントの数	—	0〜10	—	—
	試験性	試験に必要な機能数	—	0〜5	0〜15	0〜20

表 5.6　現行資産特性変動要素評価表(規模に影響する外部環境変数)

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
機能性	正確性(既存母体)	改造/流用母体が正しく動作しない場合のテスト量(現行保証)に及ぼす影響	—	0〜5	0〜15	0〜20

※改造開発特有の環境変数

5.3.2　ソフトウェアテスト見積り基本アルゴリズム

（１）　ソフトウェアテスト見積りに関連する指標

弊社で採用している，ソフトウェア見積りに関連する指標を**表 5.7**に示す。

表 5.7　ソフトウェア見積りに関連する指標

No.	指　　標	定	義
1	レビュー指摘密度	レビュー指摘件数÷生産物量	件/Kc または KLOC
2	工程独自欠陥混入密度	欠陥数÷生産物量	件/Kc または KLOC
3	テスト密度	テスト項目数÷テスト対象ソースコード行数	項目/KLOC
4	検出欠陥密度	検出欠陥数÷テスト対象ソースコード行数	件/KLOC
5	欠陥検出効率	検出欠陥数÷テスト項目数	件/項目

（２）　生産物量見積り方式

①　欠陥の混入および除去との関係

弊社ではソフトウェアに混入する欠陥の総量を見積り，レビューを含めて各開発工程で欠陥を除去する数を想定している。**図 5.2**に，設計工程および実装工程で欠陥が作り込まれ，これをテスト工程で除去するモデルを示す。

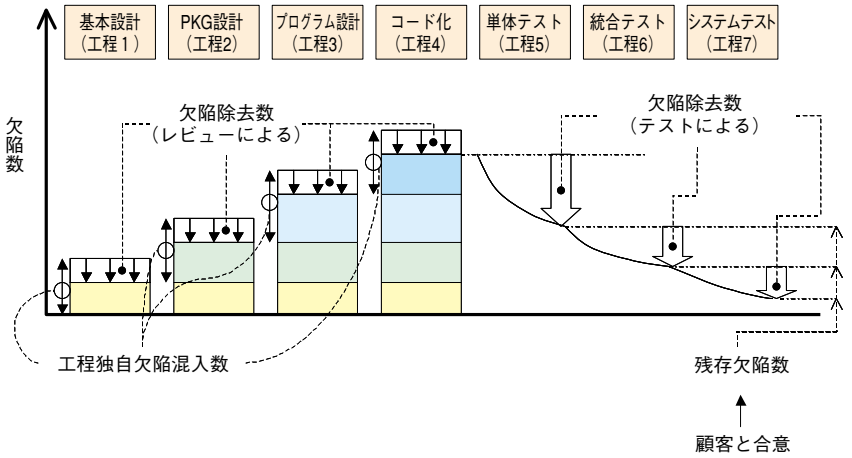


図 5.2　欠陥の混入および除去モデル

②　ソフトウェアテスト量の見積り方法

ソフトウェアテストでは，ソフトウェアの欠陥を検出し，これを修正して欠陥を除去する。弊社ではテストの目的をソフトウェアの欠陥を検出することと考え，欠陥を除去するために行う修正作業とは区分して，ソフトウェアの欠陥の検出を行うテスト作業と，ソフトウェアの欠陥を除去する修正作業とを区分して見積もっている。

ソフトウェアの欠陥の検出を行うテスト量(テスト項目数)は，欠陥の混入および除去モデルに基づいて，テスト開始時の残存欠陥密度およびテスト終了時に目標とする残存欠陥密度に基づいて見積もる。残存欠陥密度は，ソフトウェアを運用に供してから一定期間の欠陥数を測定して実測するが，ソフトウェアの開発段階では測定できない。また，テスト工程でソフトウェア信頼度成長曲線などを用いて統計的に推定する方法はあるが，テストの設計段階では推定できない。このため，テスト設計段階では，残存欠陥密度の目標をテストの網羅性を測る尺度に置き換えて目標を設定し予実を管理する。ホワイトボックステストにおける網羅性を測る尺度としては，C0，C1，C2カバレッジなどがよく知られており，テスト項目数の見積りではテスト網羅率の目標値に基づいてテスト設計した結果，期待されるテスト密度(項目/KLOC)を見積りの基準値として使用する。

また，統合テストおよびシステムテストなどでは，ユースケーステスト，状態遷移テスト，デシジョンテーブルテストおよび異常値/無効値テスト，など，さまざまなテスト観点があり，それぞれのテスト観点についてテスト項目の抽出基準を定めてテストを実施し，テスト密度を代替尺度としている。

3.1節に述べたとおり，改造型開発の場合は，テスト対象のソフトウェア規模として，テスト巻き込み規模(改造正味規模を含む)を用いる。

テスト巻き込み規模にはインターフェース規模(改造正味規模を含む)とリグレッション規模とがある。あるテスト工程iのインターフェース規模をV_i'，リグレッション規模をV_i''，インターフェース規模に対する基準テスト密度を「基準テスト密度$_i'$」，リグレッション規模に対する基準テスト密度を「基準テスト密度$_i''$」とすると，改造型開発のテスト量(テスト項目数)V_i^tは次式で求まる。

$$V_i^t = ((V_i' \times \text{基準テスト密度}_i') \times (1 + \varepsilon_i')) + ((V_i'' \times \text{基準テスト密度}_i'') \times (1 + \varepsilon_i''))$$

次に新規開発の場合のテスト量を示す。テスト工程 i のテスト量(テスト項目数)V_i^t は，新規開発のソフトウェア規模を V_i および基準テスト密度を「基準テスト密度$_i$」とすると，次式で求まる。

$$V_i^t = (V_i \times \text{基準テスト密度}_i) \times (1 + \varepsilon_i)$$

ここで，ε_i，ε_i' および ε_i'' は基準テスト密度 i，基準テスト密度$_i'$ および基準テスト密度$_i''$ に対応する変動率である。

基準テスト密度 i は基準とするテスト網羅率[14]を定め設定している。ε_i，ε_i' および ε_i'' は，当該システムのユーザからの要望[15]に基づいて，基準となるテスト網羅率からの変動率を求めて設定している。

例えば，単体テストでのテスト網羅率基準(C0，C1，C2)を例にとると，基準とするテスト網羅率(カバレッジ率)をC1(100%)としていた場合，当該システムでのテスト網羅率要求水準がC2(100%)なら，ε_i はC2の水準のテスト網羅率に調整すべく，基準とする網羅率からの差分を変動率として設定する。

③　検出欠陥数の見積り方法

弊社ではテスト工程 i で検出する欠陥数を次式にて見積もっている。

$$\begin{aligned}\text{検出欠陥数}_i &= \text{テスト対象のソフトウェア規模}_i \\ &\quad \times (\text{テスト開始時残存欠陥密度}_i \\ &\quad - \text{テスト終了時残存欠陥密度}_i) \\ &= \text{テスト対象のソフトウェア規模}_i \\ &\quad \times \{(\Sigma\text{工程独自欠陥混入密度}_i - \Sigma\text{レビュー指摘密度}_i) \\ &\quad - \text{テスト終了時残存欠陥密度}_i\}\end{aligned}$$

なお，**図 5.3**に示すように，C0→C1→C2とテストの網羅性を高めるほどテ

(14)　弊社の基準とするテスト網羅率は，単体テストをC1(100%)，結合テストおよびシステムテストはオールペアを標準としています。ただし，5.6節に述べているようにシステムテストなどテスト量と品質リスクとのトレードオフの観点から，課題があります。

(15)　顧客のテスト戦略(業務機能の重要度，テスト技法の選択など)に基づいたソフトウェアコンポーネント(ある業務機能を有するプログラム群)毎のテスト網羅率。

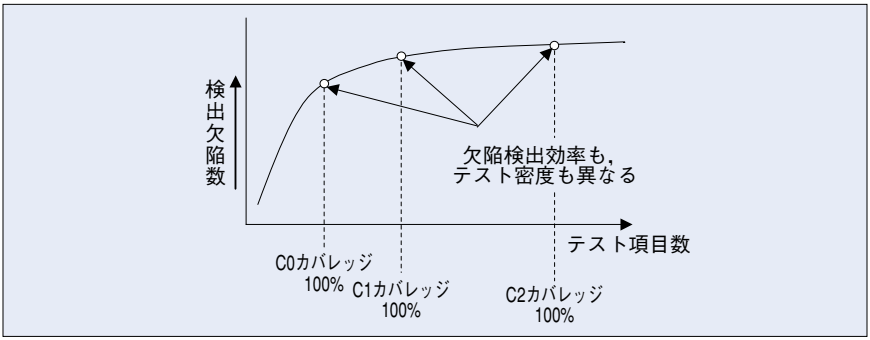


図 5.3　テスト項目数と検出欠陥数の関係

スト項目数は急激に増加する(検出する欠陥数と実施するテスト項目数との関係は線形ではない)が，欠陥が混入している確率は減少するため，欠陥検出効率(検出欠陥数÷テスト項目数)は減少する。よって，特にC２でのテスト網羅率(カバレッジ100%)を目標とする場合，単体テスト対象プログラムの重要度およびテスト予算(テスト期間含む)などを鑑みたテスト網羅率を配慮することが重要である。

④　欠陥修正作業の見積り方法

一方，ソフトウェアの欠陥を除去する修正作業は，テストで検出する欠陥を除去するために修正すべき生産物の規模に基づいて見積もる。

ソフトウェアテストで検出する欠陥の除去は，その欠陥を作り込んだ工程にさかのぼり修正するので，例えば基本設計工程で作り込んだ欠陥を修正する場合は，基本設計書，パッケージ設計書，プログラム設計書およびソースコードを修正することになる。これは，**図 5.4**のようになるので，テストで検出する欠陥を作り込んだ工程の修正の程度を，当該生産物の総量に対する修正量の比率(図 5.4では工程独自欠陥修正率と表現している)として見積もる。ある工程iの生産物の総量に対する修正量の比率は，当該工程の工程独自欠陥修正率iに先行する工程の工程独自欠陥修正率の総和を加えた値になるので，ある工程iの生産物の修正量を次式のとおり求め，これに欠陥の修正作業の生産性を乗じて修正工数を見積もる。

$$\text{工程}i\text{の修正量} = \text{工程}i\text{の生産物量} \times \Sigma\text{工程独自欠陥修正率}_j \quad (j = 1 \sim i)$$

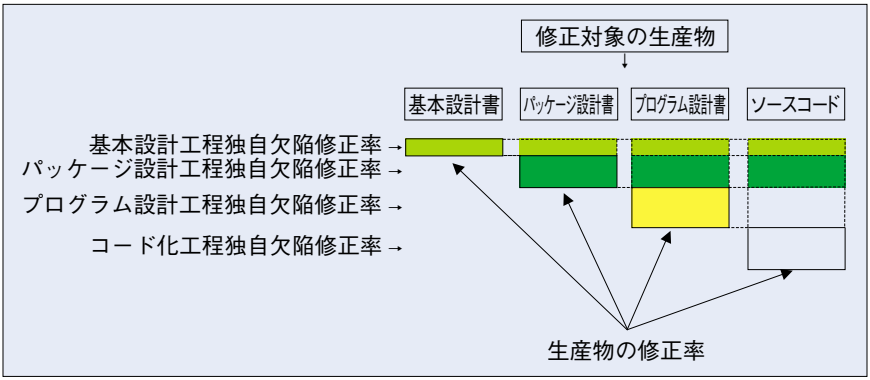


図 5.4　テストで検出した欠陥の修正

なお，例えば統合テストで検出した欠陥を修正した場合は，修正後に再度，単体テストを行うことになるので，修正量をインプットして再テストの量を見積もるが，改造開発の場合と同様にテスト巻き込み規模としてインターフェース規模とリグレッション規模の２つのテスト巻き込み規模を考慮する。再テストの生産性は初期のテストの生産性とは異なるので，初期のテストの生産性とは別に基準値を持ち，これを適用して次式で求める。

再テスト工数i＝工程iの修正量×再テスト生産性$_i$

⑤　生産物量環境変数と生産性環境変数

生産性見積り方式は5.3.1項に記載のとおりである。なお，生産物量環境変数の具体例は**表 5.5**および**表 5.6**に記載し，生産性環境変数の具体例は表 5.2～表 5.4に記載している。

生産物量環境変数の１つに「正確性：正確性(検証)に関わる標準テスト密度を基準にしたテスト項目量への要求水準」があり，これは「(２)生産物量見積り方式」に基づいて見積もったテスト項目数とテスト巻き込み規模とから導出して，テスト工程を管理するための１つの指標として利用している。また，「試験性：試験に必要な機能数」は，ソフトウェアテストの効率化およびテストの検証精度向上のためにテスト対象システムに要求する試験性などの要件を具体化して，見積りにインプットしている。

なお，テスト工程では生産性環境変数として，特に次の事項に着目して，見積り時にユーザと調整し合意に努めている。

(a)　業務ナレッジおよび役割分担

業務ユースケースの網羅，テストデータ作成に関する業務ナレッジの活用などに関するユーザとベンダとの協力体制の良否は，テストの生産性への影響が大きい。

(b)　工程入力情報特性

テストデータの誤りおよびテスト実施環境の誤動作は，テストの生産性を著しく低下させる。一方，テスト対象のソフトウェアに欠陥が少なければテストの生産性は向上する。したがって，設計工程および実装工程のレビュー(レビュー指摘密度を指標として目標を設定し管理)において，より多くの欠陥を除去することは，テストの生産性を向上させる。

(c)　開発環境特性

既存のテストウェア(テストデータ，テスト手順書，テスト環境など)の流用可能性，テストの自動化などは，テストの生産性向上効果が大きい。

5.3.3　見積り方法の前提条件

(1)　見積り時期

当該見積りモデルは企画・要件定義工程以降の基本設計工程から適用する。なお，基本設計工程のそれぞれの後続工程において再見積りを可能としている。

(2)　見積り対象

システム分類上では，エンタープライズ系および組込み系ともにソフトウェア開発を行うすべてのプロジェクトを対象とする。

(3)　体制・役割・企業文化

弊社では開発部門を牽制し支援する品質保証関連の組織を設けている。

①の組織は検査課で成果物の検査を行う専門組織である。テストと関連の深いものとしてはテストプロセス検査および出荷リスク検査がある。②の組織は品質保証委員会であり，品質マニュアルなどの組織標準およびプロセス資産を管理推進する専門組織であり，出荷後 1 年間の検出欠陥数を測定し，出荷時に設定した残存欠陥密度目標値の監視を行い，目標値を超える場合は是正処置を実施させている。

①　プロセス管理・実行検査

テストプロセス(統合テストおよびシステムテスト)については，統合テストお

よびシステムテスト期間に，テストに参画しているメンバを対象にアンケートして評価し，失敗プロジェクトの兆候を見つけて現場に勧告し，失敗プロジェクトの発生を抑える１つの手段としている。

②　最終ドキュメント検査

顧客へ引き渡す製品(生産物)自体を検査し，同製品の出荷の可否を審査している。

5.4　テスト進行中における品質管理と再テスト見積り

5.4.1　目標値の設定

（１）　目標値の設定時期

ソフトウェア開発プロジェクトの初期計画段階でソフトウェアテストの目標値を設定し，以降，各工程の区切りを契機に，見直している。なお，ソフトウェアテストの目標値として使用している品質メトリクスは「(２)品質メトリクス」に記載する。

（２）　品質メトリクス

ソフトウェアテストに関連して，弊社で使用している品質メトリクスを**表 5.8** に記載する。

表 5.8　ソフトウェアテストに関連する品質メトリクス

No.	品質メトリクス	目　　的
1	検出欠陥密度	ソフトウェア構成品の残存欠陥の評価および設計・実装工程における欠陥の混入特性を評価する。
2	欠陥検出効率	設定したテスト項目の良否(少ないテストケースでより多くの欠陥を検出できているか)を評価する。
3	レビュー指摘密度	設計・実装工程のレビューによる欠陥の除去精度を評価する。
4	テスト密度	テスト設計で設定したテスト項目の網羅性を評価する。

5.4.2　工程進行中の評価

設計・実装工程およびテスト工程において，その工程完了時の残存欠陥を評価するために，工程途中の監視および工程完了時の評価を実施している(**図 5.5**)。

レビューおよびテストで検出した欠陥の内容，重要度などの分布を分析し，設計・実装工程にフィードバックして，欠陥の作り込みを予防する。また，テスト設計も設計・実装工程と並行して行い，テスト密度に着目して，テスト網羅性が目標を満たしているか否かを評価して，テスト設計にフィードバックする。

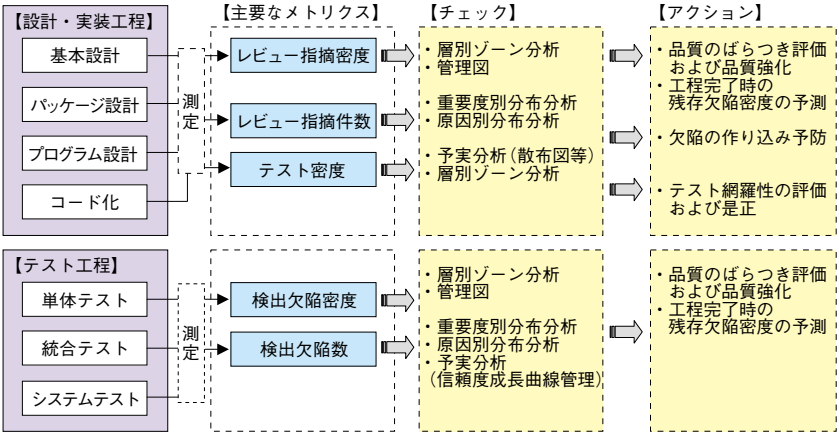


図 5.5　工程進行中の評価

また，欠陥の機能別やソフトウェア構成品別の分布特性，異常値を分析し，品質のばらつきを評価するとともに，再設計，再レビュー，再テスト（追加テスト含む）などの品質強化の要否を判断し実施する。レビューおよびテストによって，設計・実装工程で作り込んだ欠陥を目標どおり除去できているか否か（残存欠陥目標を達成できているか否か）を評価する。基準範囲よりも高くても，低くても問題がある可能性がある（**表 5.9**および**表 5.10**）。

なお，統合テストおよびシステムテストについては，信頼度成長曲線によって欠陥が収束していることも確認する。

表 5.9　設計・実装工程完了時のレビュー指摘密度の評価

レビュー指摘密度	レビューの網羅性および精度を実査	評価およびアクション
基準範囲よりも低い	レビューの網羅性および精度には，問題なし	① 設計・実装時に作り込まれた欠陥は少なく，工程完了時の残存欠陥密度も基準よりも低いことが期待できる。
	レビューの網羅性または精度に問題あり	② 再レビューを実施し，その結果を含めて再評価する。
基準範囲よりも高い	レビューの網羅性および精度には，問題なし	③ 設計・実装時に作り込まれた欠陥は多く，工程完了時の残存欠陥密度も基準よりも高いことが予測されるので，追加レビューの要否を判断し，追加レビューを実施した場合は，その結果を再評価する。追加レビューを行わない場合は，次工程に品質強化策を申し送る。
	レビューの網羅性または精度に問題あり	④ 設計・実装時に作り込まれた欠陥は多く，工程完了時の残存欠陥密度も基準よりも著しく高いことが予測される。再設計も含め品質強化策を実施する。

表 5.10　テスト工程途中の検出欠陥密度の評価

検出欠陥密度	テストの網羅性および精度を実査	評価およびアクション
基準範囲よりも低い	テストの網羅性および精度には，問題なし	① 設計・実装時に作り込まれた欠陥は少なく，テスト完了時の残存欠陥密度も基準よりも低いことが期待できる。
	テストの網羅性または精度に問題あり	② 追加テストを実施し，その結果を含めて再評価する。
基準範囲よりも高い	テストの網羅性および精度には，問題なし	③ 設計・実装時に作り込まれた欠陥は多く，テスト完了時の残存欠陥密度も基準よりも高いことが予測されるので，追加テストの要否を判断し，追加テストを実施した場合は，その結果を再評価する。追加テストを行わない場合は，次工程に品質強化策を申し送る。
	テストの網羅性または精度に問題あり	④ 設計・実装時に作り込まれた欠陥は多く，テスト完了時の残存欠陥密度も基準よりも著しく高いことが予測される。再設計および再テストも含め品質強化策を実施する。

5.4.3　再テスト見積り

（1）　設計・実装工程完了時の再テスト見積り

5.4.2項に記載した「工程進行中の評価」によって，設計・実装工程後の残存欠陥を推定し，テストの観点およびテスト技法ならびにテスト網羅率などの目標値を再設定して，テスト工程の再見積りを実施する。

- 設計後の残存欠陥が減少していれば(表5.9の①に該当する場合)，低下した残存欠陥リスクを特定してテスト量を削減する。
- 設計後の残存欠陥が増加していれば(表5.9の③および④に該当する場合)，設計の再レビューなどの品質強化を行う一方，増加した残存欠陥リスクを特定してテスト量を増加する。

（2）　テスト進行中の再テスト見積り

テスト進行中の再テスト見積りの契機として，次の3点を考慮している。

①　前工程の品質目標未達

表5.10の③に該当し，前工程から品質強化策の申し送りがある場合

②　品質レベルの偏りの是正

表5.10の②～④に該当する場合

③　欠陥除去目標未達

実施しているテストの網羅性が疑わしい場合は，機能ごとのテスト完了付近でランダムテストなどを追加して，信頼度成長曲線の収束状況をチェックして，追加テストの要否を判断する。

5.5　プロジェクト実績の蓄積と活用
（～テストプロセスの計測およびフィードバック～）

弊社で実施しているプロジェクト実績の活用方法の事例として，プロジェクトのテスト実績を用いて，プロセス改善につなげている活動事例を紹介する。

まず，プロジェクトの完了時に，レビューおよびテストで検出した欠陥を，欠陥の作り込み工程，欠陥の検出工程別に分類して，**図5.6**に示すマトリクスに整理したうえで，改善点を分析する。

図5.6に示すマトリクスは，次のように作成する。まず，それぞれの欠陥を，欠陥を検出した工程別に分類し，その件数を左右両方の合計欄に書き入れる。最初に，右側の＜検出した欠陥を作り込んだ工程＞のマトリクスを作る。これは，それぞれの欠陥を作り込んだ工程別に分類して，その件数を各工程のセルに書き

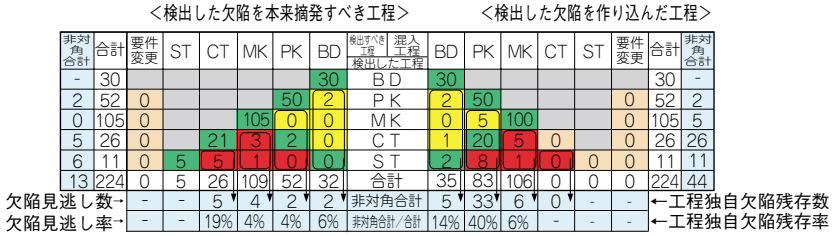


図 5.6　欠陥作り込み/検出マトリクス

込むとできあがる。次に，＜検出した欠陥を本来検出すべき工程＞のマトリクスを作る。そのために，プロジェクトの各工程のレビュー方針および各テストフェーズのテスト方針に照らして，それぞれの欠陥について，どの工程で検出されるべきだったかを判定して分類して，各工程のセルに書き込むとできあがる。

ここで，欠陥見逃し率とは，当該工程で検出すべき欠陥の総数に対する，当該工程で検出できなかった欠陥件数の占める比率であり，当該工程の設計レビューあるいはテスト工程の欠陥検出能力として利用している(値が小さいほど欠陥の検出能力は高い)。また，工程独自欠陥残存率は，当該工程完了時の残存欠陥の密度(＝工程独自欠陥混入密度－レビュー指摘密度)であり，各工程においてソフトウェアに欠陥を作り込む特性として利用している(値が小さいほど欠陥を作り込む確率は低い)。この２つ特性に焦点をあてて，**図 5.7**に示すとおりプロセスの改善可能性を分析してプロセスの改善案を作成し，改善を施した結果を推定しマトリクスに整理する。

改善後のマトリクスからは，各工程で作り込む欠陥の密度(工程独自欠陥混入密度)，レビューで摘発できる欠陥の密度(レビュー指摘密度)およびテストで検出できる欠陥の密度(検出欠陥密度)が導出されるので，これを見積りの基準値にフィードバックして，次のプロジェクトの見積りに適用する。

5.6　当該取り組みの課題

5.6.1　システムテストにおける妥当なテスト網羅率の設定

5.3.2項(２)で新規開発および改造型開発のテスト量(テスト項目数)の見積り方法を述べた。現在は，単体テストについてはホワイトボックステストでＣ０，Ｃ１，Ｃ２カバレッジを尺度とし，統合テストについてはオールペア法に基づく

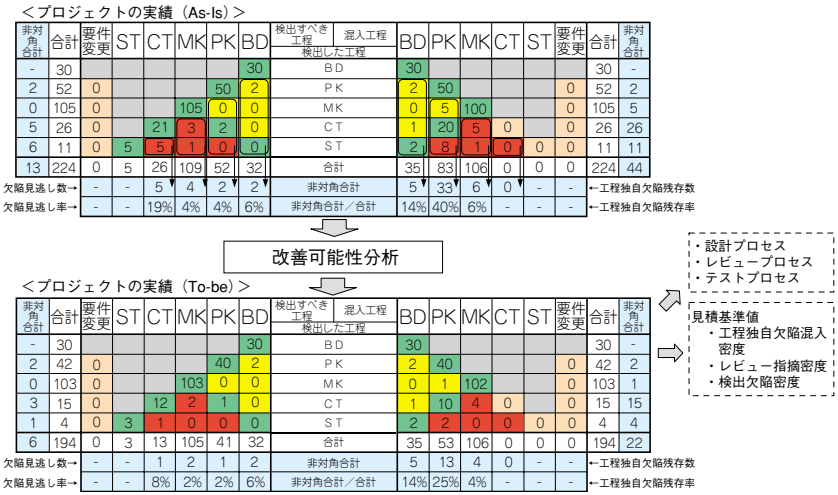


図 5.7　プロセス改善へのプロジェクト実績の利用例

2 因子間の網羅率100%を尺度としてユーザに提案している。ただし，システムテストに関してもオールペア法を基準としているが，さらに，各種仕様記述(ユースケース図，状態遷移図，デシジョンテーブル，運用マニュアルなど)からテスト項目を追加する必要があり，テスト量と品質リスクとの[16]の観点から見た妥当なテスト網羅率を設定するテスト方法論が課題となっている。

5.6.2　非機能要件の確認テストの見積り

弊社では非機能要求(品質特性‘JIS X 0129’)の多寡によるテスト量の変動を吸収するパラメータとして「環境変数」(5.3.1項)を導入し，品質特性‘JIS X 0129’とテスト量(テスト項目数)との定量的な関係の把握を行っている。しかし，非機能要求の測定項目については，さらなる充実が課題となっている。現在，ユーザ(日本情報システム・ユーザー協会(JUAS)など)の協力を得て，課題への取り組みを開始している。

(16)　合理的に削減されたテスト量によるテストコストと，その残存欠陥がソフトウェア製品の品質リスクを許容範囲内に抑えるというトレードオフを考慮したテスト見積りが必要になります。

第 6 章　日本電気（NEC）の事例
～ソフトウェア品質会計制度の適用～

6.1　紹介見積り事例プロフィール

弊社における品質指標をもとにした上流工程からの品質管理とソフトウェアテストの取り組みについて紹介する。ここでは，上流工程からの品質管理手法としてソフトウェア品質会計制度について概要，適用手順について紹介するとともに，テスト工程におけるソフトウェアテスト見積り上の考慮点について紹介する。

6.2　取り組みの背景

弊社では，1980年代初頭から基本ソフトウェア，応用ソフトウェア製品開発を中心としてソフトウェア品質の向上に取り組み，ソフトウェア開発領域での小集団活動，開発プロセス/生産技術，管理技術の標準化，開発や管理ツールの工夫などに取り組んできた。

特にソフトウェア管理技術，特にソフトウェアの品質管理に関して，上流工程から品質を作り込むことを目的とした管理活動に取り組み，その中でソフトウェアの品質の計量化の応用としてソフトウェア品質会計制度を考案し展開している。当初品質会計制度は基本ソフトウェア開発分野，応用ソフトウェア製品開発分野といった開発者自らが仕様を決定し開発するというソフトウェア開発領域に適用されていたが，この適用範囲を順次拡大し，発注者，開発者で仕様を確認し決定したうえで開発を行うSIを伴う業務システム開発領域に適用するまでに至っている。ここでは，ソフトウェア品質会計制度を中心とした弊社のソフトウェア品質管理手法とテストの見積りについて紹介する。

品質会計制度は単なる数値目標の管理だけではなく，

・なぜバグが予測以下/以上なのか？
・その理由は正しいか？
・レビューやテストの項目や観点は十分か？

などの点をチェックし，不足であればその対策を実施する，またその結果も製造側の品質見解としてまとめるなどの組織的な品質管理活動となっている。

6.3　ソフトウェア品質管理の考え方と ソフトウェアテスト見積りの取り組み

6.3.1　ソフトウェア品質会計制度の基本概念

ソフトウェア品質会計制度とは，財務会計としての簿記制度，管理会計の目標原価と実績原価の差異分析手法を参考にした管理手法である。具体的には，

① ソフトウェアライフサイクルの品質状況(欠陥の検出と欠陥の除去)を簿記における取引となぞらえ，開発工程という期間を通じて計測・記録し，開発工程の管理期間単位に集計し，決算する。

② 設定した各工程における品質目標(作り込み品質予測と欠陥除去目標)を管理会計の目標原価に，実施面における実績値(検出した欠陥数)を管理会計の実績原価となぞらえ，目標と実績との間の差異分析を行う

という管理手法をとる。ここで欠陥とは，

① ソフトウェアの要件定義，ソフトウェアの外部設計工程で作り込まれた仕様の誤り，仕様の漏れ

② プログラムのロジック設計工程，プログラム製造工程で作り込まれた仕様の誤り，仕様の漏れ，ソースプログラム上のロジックの誤り，規約に反したプログラムの記述

を指す。

6.3.2　ソフトウェア品質会計制度による品質管理の考え方

ソフトウェア品質会計制度では，要件定義工程，設計工程，製造工程までの欠陥が作り込まれる工程を上流工程として，単体テスト，結合テスト，総合テストなどといったテスト工程を下流工程ととらえる(**図 6.1**)。

要件定義工程，設計工程で作り込まれる欠陥は仕様の誤り，本来実現すべき仕様の検討漏れであり，これらは設計のレビューを通じて検出する。また，製造工程の欠陥は仕様で定義された要件をプログラムとして作成する際のデータ宣言の誤り，ロジックの記載漏れ，ロジック誤りなどであり，これらはプログラム開発者によるプログラムコードレビュー，第三者によるコードインスペクションにより検出される。

ソフトウェア品質会計制度では，上流工程で作り込んだ欠陥の摘出は望ましくは同一工程で，悪くても上流工程完了までに除去するという目標のもとに除去目

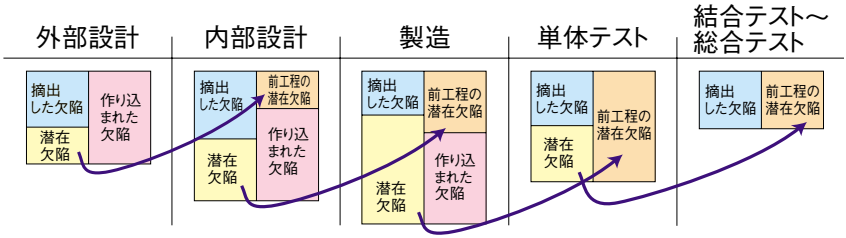


図 6.1　開発工程における欠陥の作り込みと摘出

標値を立て，工程ごとに欠陥の摘出数を計測し目標値との差異分析を行い，目標値と実績値に有意な差がある場合にはさらにレビューを行う。

下流工程，すなわちテスト工程は作成されたプログラムが仕様どおりに動作するかどうかを確認する検証工程である。テスト工程ではあらかじめ計画されたテスト項目にしたがってテストを実施し，正しく仕様どおりに動作することを検証ならびに欠陥の除去が主な作業となる。

下流工程で摘出すべき欠陥の目標値は上流工程では除去できなかった欠陥数(目標値と実績値との差異)であり，下流工程で摘出すべき欠陥の目標値とする。下流工程では各工程のテストを実施し，上流工程と同様に工程ごとの摘出目標値と摘出実績との差異分析を実施し，その摘出の十分度を把握する。また，摘出した欠陥数の予実差異分析だけではなく，欠陥摘出の累積状況から品質の安定度を評価し，最終的に作られたソフトウェアの品質安定度を評価しテストを完了させるか否かを判断する。

6.3.3　ソフトウェアテスト見積りの考え方

6.3.2項で述べたように，ソフトウェア品質会計制度では要件が正しく設計に反映されていることを前提として，「テスト工程は作成されたプログラムが仕様どおりに動作するかどうかを確認する検証工程である。」ととらえている。したがって，ソフトウェアテストの検証工数は上流の設計工程において合意されたソフトウェア機能数に依存する。また，同じソフトウェア機能であってもユーザが求める品質要求(信頼性，柔軟性，効率性など)が異なれば，その品質要求を満たしていることを検証する必要があり，要求の程度が高いほど評価検証は難しくなり，工数も増加する。

また，テスト対象が追加あるいは改良された機能であった場合には，その機能の追加，改良が既存の機能に影響を与えていないことの検証も必要となる。機能の追加，改良により影響を受ける既存機能が多ければ多いほど検証工数は増加する。

以上述べたことは，開発される機能の検証に関する工数の見積りの考え方であり，実際にテスト工数で発生する工数は，検証の工数以外にテスト工程で発見された欠陥の除去にかかる工数がある。欠陥の除去にかかる工数は一般にその欠陥を作り込んだ工程と，検出した工程が離れれば離れるほど除去に必要な工数は増大するといわれている。

したがって，上流工程での品質確保が十分でない機能の開発は，上流で十分に確保した機能と比べると評価検証工数は同じであっても，テスト工程で発生する欠陥除去の工数分大きくふくらむことになってしまう。

ソフトウェア品質会計制度を適用することは，こうしたテスト工程の欠陥除去の工数増加を抑えるだけではなく，上流工程での機能仕様を明確化するとともに，求められる品質要求の程度を上流工程できちんと把握するうえで効果を発揮する。

6.4　ソフトウェア品質会計制度の適用方法

6.4.1　ソフトウェア品質会計制度の適用手順

システム開発でソフトウェア管理会計制度を適用する場合には次の手順で行う。

- 手順1　品質計画の立案
  - ①　見積規模をもとに全工程の潜在欠陥数を予測する。
  - ②　工程別の欠陥摘出比率を設定し，工程別の摘出欠陥目標値を設定する。
- 手順2　上流工程(設計・製造工程)の品質管理
  - ①　作成した設計書のレビューあるいは作成したプログラムのレビュー，コードインスペクションによる欠陥摘出を行うとともに，欠陥の予定と実績の管理を行い，摘出した欠陥に関する分析を行う。
  - ②　各工程の完了時に評価を行い，工程の移行可否を判断する。
  - ③　上流工程(設計・製造工程)の各工程終了時に摘出する欠陥数の目標値を見直す。

    仕様変更などの開発状況の変化が発生した場合には必要に応じて中間評価のタイミングであっても摘出する欠陥数の変更が必要かどうかをチェック

する。

- 手順 3　下流工程(テスト工程)の品質管理業務
  - ①　テストによる欠陥摘出と予定と実績の管理を行い，摘出した欠陥に関する分析を行う。
  - ②　各テスト工程の完了時に摘出した欠陥に関する分析と評価を行い，工程の移行可否を判断する。目標値と実績の間に有意な差異がある場合には，必要に応じてテスト項目の追加や再テストなどを行う。
  - ③　開発工程全体の品質データのまとめと評価を行う。
- 手順 4　最終判定(開発作業終了判定)と本番稼働後のフォローアップ
  - ①　全工程で摘出した欠陥に関する分析を行い終了時点の品質評価を行う。
  - ②　本番稼働後の品質状況をチェックし，当初の目標どおりの品質に達しているかをチェックする。

以上の適用手順を図示すると**図 6. 2**のような流れとなる。

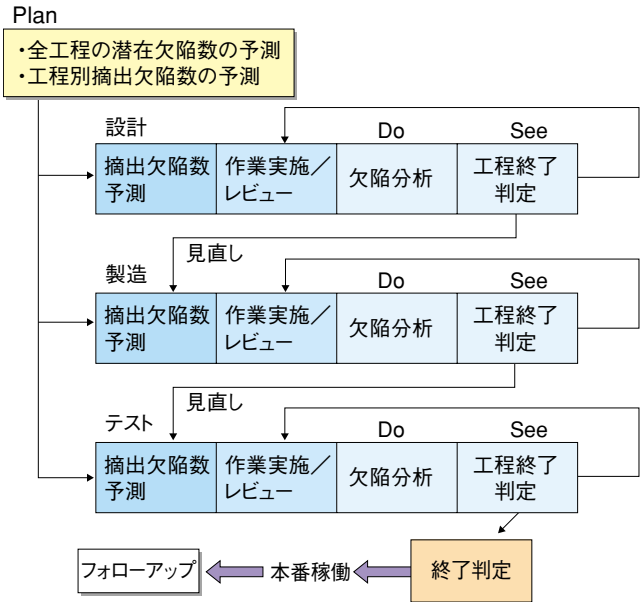


図 6. 2　品質会計制度の適用手順

6.4.2　品質計画の立案

（1）　潜在欠陥数の予測

計画段階における全工程の欠陥数の予測の基本的な考え方は，開発規模を元に標準的な潜在欠陥率，その他重要要因による補正係数をかけて求める。すなわち，予測潜在欠陥数は次式となる。

予測潜在欠陥数(件)＝Σ(開発規模×標準潜在欠陥率×プログラム係数
×言語係数×改造係数×スキル係数)

※　開発規模：当該のシステム開発で新規に開発する，もしくは改造するプログラムの見積規模(KS)
後に示すプログラムの種別ごとに見積り
標準潜在欠陥率：同種の業務，業種システム開発で過去に発生したバグ数(件数/KS)(プロジェクト規模，システム特性(複雑度，難易度)により異なる)
プログラム係数：開発するプログラム難易度を示す係数(開発するプログラムの種別と難易度(複雑度，規模)により異なる)
言語係数：開発に用いるプログラム言語の違いによる指数
スキル係数：開発を担当する要員のスキルやPJ経験

例えば，新規に業務アプリケーション50KSをC言語でプログラム経験の少ない要員で開発するPJがあったとする。その組織の標準欠陥率が平均10件/KS，この種の業務アプリケーションのプログラム係数が1.0，C言語のプログラム係数が1.0，スキルの少ない要員の場合のスキル係数が1.4であった場合には，このPJの潜在欠陥数は，

潜在欠陥数＝50×10×1.0×1.0×1.4＝700件

となる。

（2）　潜在バグ数の工程別展開

次に(1)で得られた潜在欠陥数を工程別に配分する。欠陥の配分は次のように行う。

①　作り込み工程ごとの欠陥目標値の設定

過去の実績を元に欠陥を作り込んだ工程の比率を設定し，これに潜在欠陥数をかけて工程ごとに作り込む欠陥数の目標値を設定する。

②　工程ごとの検出欠陥目標値の設定

欠陥は作り込んだ工程ですべてが検出されるわけではないので，当該工程で検出される欠陥と，その工程では見逃がされ後工程で検出される欠陥に比率を分けて工程ごとの摘出目標値を設定する。また，設定においては管理目標として上流工程で検出する目標摘出率(α[%]以上)を設定し，それを達成するように工程ごとの目標値の見直しを行う。

実際の品質会計制度の運用にあたっては，**図 6. 4**のような管理シートを用いて目標値と実績値を管理している。

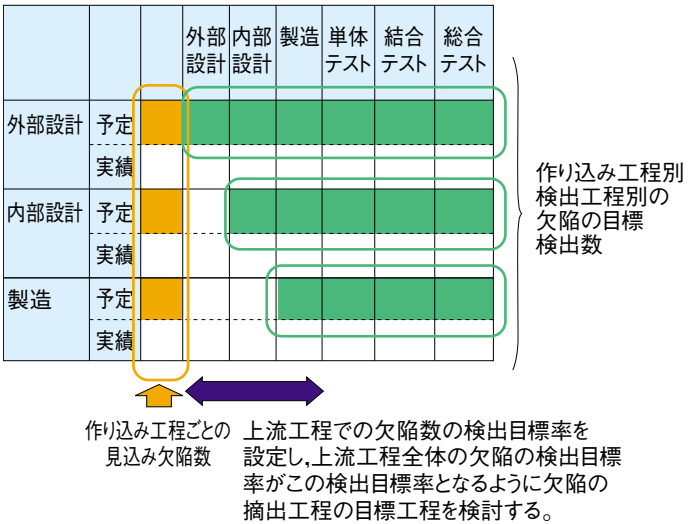
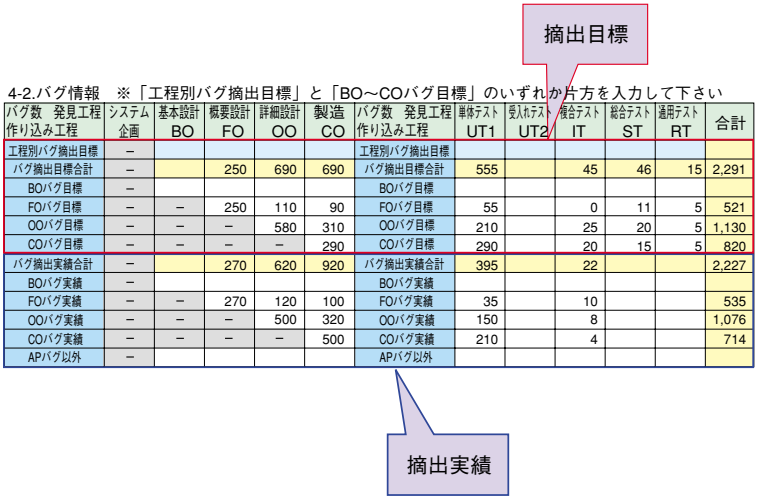


図 6. 3　工程別検出目標値の設定



バグ数　発見工程 作り込み工程	システム 企画	基本設計 BO	概要設計 FO	詳細設計 OO	製造 CO	バグ数　発見工程 作り込み工程	単体テスト UT1	受入れテスト UT2	結合テスト IT	総合テスト ST	適用テスト RT	合計
工程別バグ摘出目標	–					工程別バグ摘出目標						
バグ摘出目標合計	–		250	690	690	バグ摘出目標合計	555		45	46	15	2,291
BOバグ目標	–					BOバグ目標						
FOバグ目標	–	–	250	110	90	FOバグ目標	55		0	11	5	521
OOバグ目標	–	–	–	580	310	OOバグ目標	210		25	20	5	1,130
COバグ目標	–	–	–	–	290	COバグ目標	290		20	15	5	820
バグ摘出実績合計	–		270	620	920	バグ摘出実績合計	395		22			2,227
BOバグ実績	–					BOバグ実績						
FOバグ実績	–	–	270	120	100	FOバグ実績	35		10			535
OOバグ実績	–	–	–	500	320	OOバグ実績	150		8			1,076
COバグ実績	–	–	–	–	500	COバグ実績	210		4			714
APバグ以外	–					APバグ以外						

図 6.4　品質会計の管理シート例

6.4.3　品質管理の実施

（1）　上流工程（設計・製造工程）における品質管理

設計・製造工程では，その工程の立ち上がり時にその工程の成果物である仕様書の想定ページ数や，プログラムの想定規模を元に欠陥摘出目標を確認する。

設計工程では仕様書のドラフトが完成した時点から，製造工程ではプログラムのコンパイルが完了した時点から，レビューやインスペクションを通じて欠陥数をカウントし，その修正状況をチェックする。各設計工程の仕様書完成時（レビュー完了時），製造工程ですべてのソースプログラムの完成時（レビュー完了時）にその工程の欠陥の摘出目標と実際の予実評価を行い，工程完了判定を行う。もし，欠陥摘出数が目標値と有意な差がある場合には，その工程の成果物の品質に問題あると判断し，追加のレビューを行う。

工程完了判定時にはこれまでの欠陥摘出目標達成上の評価を行い，潜在している欠陥数の見直しを行う。また，設計・製造工程の途中で仕様変更，仕様追加などの理由により開発規模が変更になった場合には，これらの変更を考慮して潜在している欠陥数の見直しを行う。特に製造工程の完了時（上流工程）の完了時はテスト工程（下流工程）の欠陥摘出目標の見直し（変更）を検討する。この見直しによりテストすべき項目の見直しが必要であれば，テスト仕様書の見直しを図り，実

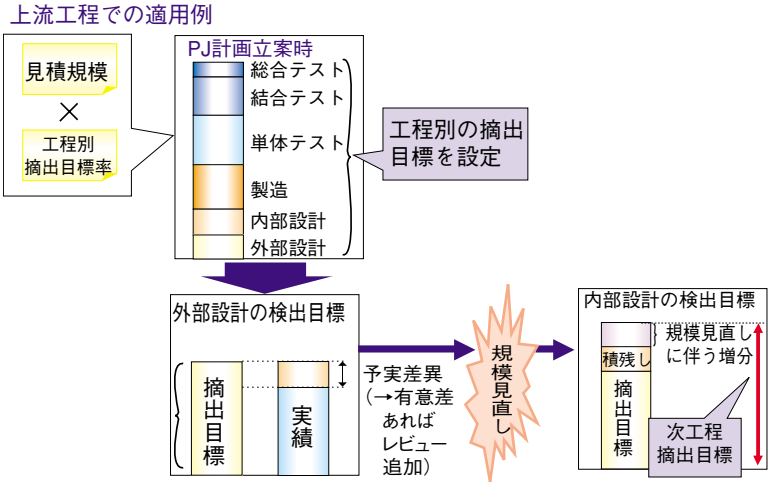


図 6.5　上流工程での適用例

施予定テスト項目数および実施計画を見直す。

図 6.5に上流工程での適用例を示す。

(2)　下流工程(テスト工程)における品質管理

下流工程(テスト工程)ではテスト計画にしたがってテストを実施し，欠陥を摘出する。

実施するテストは次の 3 種類に分け，それぞれテスト工程で実施する項目数，設定根拠を明確にする。

① 新規機能テスト項目：新規作り込み機能の確認項目

② 流用機能テスト項目：新規流用・移植機能の確認項目

③ 既存機能テスト項目：既存機能の確認項目(デグレードテスト)

新規開発では，①あるいは②を中心にテストを行い，テストを実施する際には，過去の開発における欠陥摘出の実績値(欠陥数/KS)，欠陥 1 件あたりのテスト項目数，および上流工程での欠陥摘出状況を考慮して開発プロジェクトとしての標準値を設定し，これにしたがってテスト種別ごとの実施予定テストの実施スケジュールを策定する。特に，上流工程で欠陥の検出が少なかった業務機能は本来上流工程で検出されるべき欠陥がテストを通じて検出される可能性が高い。テスト設計ではこの点を考慮する必要がある。

単体テストは，要求機能の評価(ブラックボックステスト)に偏りがちであるが，ロジックの網羅度(命令網羅，条件網羅，複数条件網羅)を考慮したホワイトボックステストも併せて行うこと，命令網羅を効率的に行うためにはテスト項目を増やすのではなくバグ摘出率の高いテスト手法を採用することを提唱している。

結合テストやシステムテストでは，機能特性に偏った評価を行うのではなく，ユーザの要求事項として求めているシステム特性や業務特性，ソフトウェア特性を考慮して，信頼性，使用性，効率性，保守性といった品質特性の観点からも評価することを求めている。

また，網羅性の高いテスト設計書が用意できても，テスト環境やテストデータの準備，検査者の能力によりテスト工数や検出される欠陥数が変わるため，テスト設計にあたってはこれらの要素も考慮して見積りや準備を行う必要がある。

テストの途中であっても中間評価でテスト項目の消化数，テスト消化率および検出した欠陥数からテストの品質状況，プログラムの品質を評価し，評価が悪い場合にはその原因を調べ，必要ならばテスト設計の見直し，上流工程のレビューの再実施を行い，テストの再見積りを行う。

テストを実施する際には，過去の開発における欠陥摘出の実績値(欠陥数/KS)，欠陥1件あたりのテスト項目数，および上流工程での欠陥摘出状況を考慮して開発プロジェクトとしての標準値を設定し，これにしたがってテスト種別ごとの実施予定テストの実施スケジュールを策定する。テストの実施に伴って摘出された欠陥の実績は，テスト項目の消化状況と欠陥の摘出状況を表したグラフなどを利用して，品質目標の予定と実績の確認・評価を行う。

評価は各テスト工程(単体テスト工程，結合テスト工程，総合テスト工程)の完了時のタイミングで品質会計の中間評価を行う。

テスト評価は，テスト項目の網羅性，適切性，十分性およびテスト記録の正当性の観点から行う。下流工程では，目標摘出欠陥数は目標値以下(1－α)(%)となる(αは上流工程で検出する目標摘出率)ように管理しており，もしこの目標値を超えるような場合には，上流工程での品質管理に問題があったとみなし，問題点を洗い出してその対策立案と再テストを実施する。評価では検出した欠陥数と目標値の差異分析だけではなく，テストの消化状況と欠陥の摘出状況から見た品質の安定度も評価の対象として品質状況を判断し，次の工程に移行するかどうかを判断する。**図6.6**に下流工程での適用例を示す。

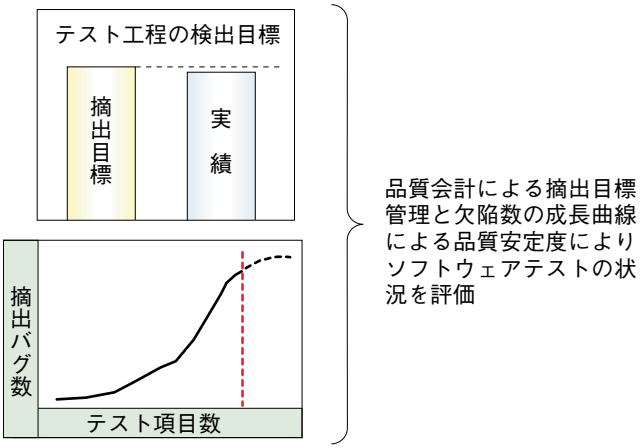


図 6.6　下流工程での適用例

総合テスト工程終了時の評価後，全工程での欠陥の摘出目標と実勢を合計した最終判断を行い，本番稼働後に発見される可能性のある残潜在欠陥数の予測とそのリスク評価を行う。

（3） 本番稼働後のフォローアップ

開発期間中には検出されずに本番稼働後に発覚する欠陥もあるため，本番稼働後も定期的な評価を行う。ここでは，本番稼働後に発覚した欠陥は総合テスト終了時に予測値以下であるかどうか，また発覚した場合のリスク対策に問題はなかったかを評価する。

また，開発プロジェクト全体を振り返って，

・実績値から見て工程別の欠陥摘出目標値の設定は適切であったか？

・是正処置などは組織的に展開されているか？

について評価し，今後の管理の参考とする。

6.4.4　品質状況の評価

品質会計制度では，下流工程の終了時の判定において次のような指標により品質状況を評価する。

（1）　テスト消化率とテスト合格率による評価

計画されたテスト項目予定数(テスト項目数)，実施されたテスト項目数(テスト実施項目数)，および実施され合格したテスト項目数(テスト完了項目数)をもとに，

テスト合格率＝テスト完了項目数/テスト実施項目数
テスト消化率＝テスト実施項目数/テスト項目数

の 2 つ指標を算出し，これらの関係によりテストの十分性を評価する(**図 6.7**)。

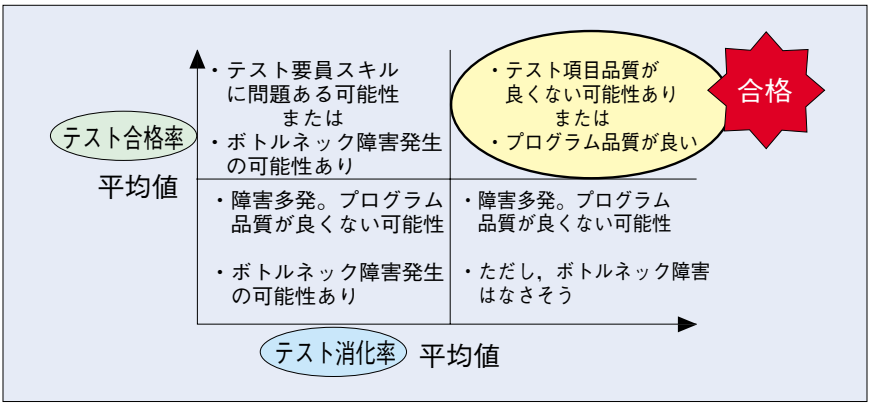


図 6.7　テスト消化率とテスト合格率による評価

（2）　テスト項目数と検出した欠陥数による相対的品質評価

実施したテスト項目数とその間に検出した欠陥数の関係により品質を評価する方法である。テスト項目数をX座標に，検出した欠陥数をY座標にとり，テスト項目数と検出した欠陥数の関係を平均値を中心に 4 象限に分け，どの象限にいるかにより品質評価を行う方法である(**図 6.8**)。

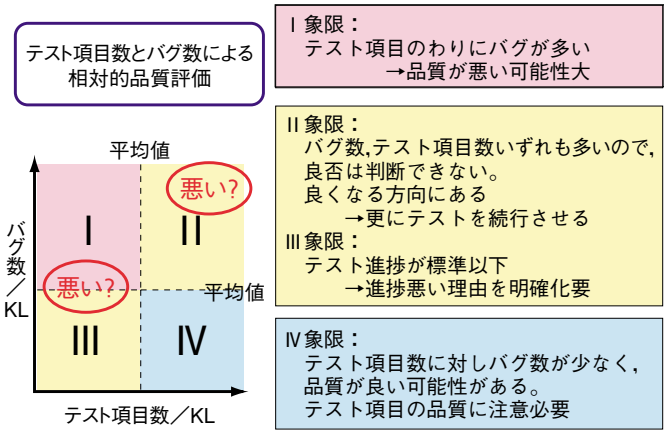


図 6.8　テスト項目数とバグ数による相対的品質評価

6.5　今後の課題

以上，品質会計制度ならびにその適用方法について紹介を行った。実際の運用では，その潜在欠陥数の算定の精度そのものは問題ではなく，工程終了時の評価において開発プロジェクトの欠陥の摘出状況から品質に関するチェックの十分性，適格性を評価し，品質管理方法で是正すべき点がないかをセルフチェックしセルフイノベーションできる体制の確立のほうが重要である。これはCMMIのような組織的な取り組みであり，品質会計制度の運用においても組織的な品質の計測，改善の仕組みを充実させていくことが課題である。

また，実際のSIの場面では開発途中における仕様変更や仕様追加が多く発生する。品質会計制度でもこの点は考慮している，仕様変更により廃棄された部分で検出された欠陥は，以降の管理においては排除する，また追加された仕様に関しての潜在欠陥数を追加して管理するなどの運用上の改善も必要である。

資　料

「非機能要件の把握・確認とテスト見積りへの影響表」(機能性 1/5)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
機能性	合目的性	実行時に提供される機能の,利用者からの要求に対する妥当性の評価	機能の妥当性	実行時の機能として正しく実装されている要求の割合。正しく対応されているかどうかはテストによって判断する。正しく対応されていない要求とは,対応する実装機能が欠けているか,対応する実装機能が要求に適合していないものを指す。	実行時の機能として正しく対応されている要求の数/全要求の数	0～1に近いほど良い		①	②	③	④	④	⑤		○	○	
		要求外の機能が含まれている割合についての評価	機能の超過度	要件として挙がっている機能,もしくはその要件を満たすうえでの暗黙の機能以外の,要求外の機能がどの程度含まれているかの割合。	要求から超過した機能数/全要件機能数	0～10に近いほど良い						④	⑤			○	
		機能要件の変更・追加の割合によって推定される,元々の機能要件,設計,実装の妥当性の評価	機能要件の安定性	運用・保守時においての機能要件の変更・追加の割合を測定し,全機能数との割合を求める。環境変化による機能要件の変更・追加は除くか,標準的な数値との比較によって妥当性を判断する。	利用者から変更・追加のあった機能数/全機能数	0～10に近いほど良い	①							⑤	○		
		利用者からのクレームの多寡によって推定される,元々の機能要件,設計,実装の妥当性の評価	利用者からのクレーム率	利用者からの機能に対するクレーム数を経過時間および規模で割った数値。経過時間の単位は月数など。規模はソースコードの行数,画面数など。	クレーム数/(経過時間×規模)	0～10に近いほど良い	①							⑤			
	正確性	計算精度の要求に対しての,実現の割合の評価	計算精度の妥当性	計算途中結果の保持の際のデータ精度,計算の際の桁落ちの防止などの有効桁数に対する要求が満たされているかどうかを,レビューおよびテストによって判断する。	特定の精度要求事項が実現されたと評価で確認した機能の数/特定の精度要求事項を実現する必要のある機能数	0～10に近いほど良い		①	②	③	④	④	⑤		○	○	
		データ精度の要求に対しての,実現の割合の評価	データ精度の妥当性	数値であれば桁数,文字列であれば文字列長,時刻であれば時分秒,あるいはミリ秒などの,データとして保持される際の精度についての要求が満たされているかどうかを,レビューおよびテストによって判断する。	特定の精度要求事項が実現されたと評価で確認した機能の数/特定の精度要求事項を実現する必要のある機能数	0～10に近いほど良い		①	②	③	④	④	⑤		○	○	
		実動作に対するマニュアルの一致の程度の評価	マニュアル記述の正確性	マニュアルの記述が正確かどうか,すなわち実際の動作と一致しているかどうかの割合。	利用者がマニュアルの記述と実際の動作との差異が許容範囲を超えていると判断した項目の数/対象となる全項目の数	0～10に近いほど良い						④	⑤			○	

①:要求仕様の決定(ユーザの役割) ②:要求を実現するための仕様提示(ベンダSE,またはユーザSE) ③:設計仕様への盛り込み(および確認) ④:実装結果の確認 ⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(機能性 2／5)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
機能性	正確性	実装機能についての利用者の合理的な期待に対する，一致の程度の評価	実装機能の正確性	定められた入力と期待される出力からなるテストケースを実施し，実際の出力と比較し，利用者が受け入れ難い相違があると判断した項目の数によって実装機能の正確性を判断する。	利用者の合理的な期待と実際の結果との差異が許容範囲を超えている項目の数/対象となる全項目の数	0～1 0に近いほど良い						④	⑤		○	○	
		工程に対するインプットがアウトプットにおいて正確に反映されているかどうかの評価	検査の密度	ある工程において前提となる要求や設計などの情報が，その工程でのアウトプット(設計・実装についてのドキュメント，プログラム)において正確に反映されているかどうかを，検査の密度によって間接的に判断する。 検査とは，上流工程においてはレビュー，下流工程においてはレビューに加えてテストなどの手段を指す。	組織として定められた基準としてのレビューやテストなどの項目数，時間，回数などに対する倍率	0～ 1を超えることが望ましい	①		②	③	④	④	⑤			○	
	相互運用性	他システムとの接続実現の評価	他システムとの接続性ー1	他システムとの接続性を，接続可能なシステム数で判断する。要件定義および設計工程においては，システム数，データ形式に関する事項についてレビューによって判断し，設計以降は接続に関する試験によって判断する。運用開始以降は，利用者が遭遇したデータ交換の失敗ケースによって判断する。	接続可能な他システムの数/接続の必要がある他システムの数	0～ 1を超えることが望ましい		①	②	③	④	④	⑤		○	○	
			他システムとの接続性ー2	他システムとの接続性を，交換可能なデータ形式数で判断する。	他システムとのデータ交換の試験中に良好に交換できたと認められるデータ形式もしくはデータの数/交換されたデータ形式もしくはデータの数	0～ 1を超えることが望ましい		①	②	③	④	④	⑤		○	○	
			他システムとの接続性ー3	他システムとの接続性を，試験時および運用時のデータ交換の成功の割合で判断する。	利用者が接続に成功したデータ交換ケース数/利用者が接続を試みたデータ交換ケース数	0～ 1を超えることが望ましい		①	②	③	④	④	⑤		○	○	
		接続の要求がある他システムに対して，実際に接続する場合の容易性の評価	他システムとの接続容易性	あらかじめ想定される範囲での他システムとの接続の実現に必要な予想工数によって，他システムとの接続をどの程度まで考慮した作りになっているかを判断する。	他システムとの接続に必要な工数	0～ 小さいほど良い		①	②	③				④ ⑤			

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(機能性 3/5)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用	テスト量	テスト生産性
機能性	セキュリティ	データの機密性の評価	データの暗号化ー1	社内規約で暗号化機能を要求するデータ項目数に対する,暗号化するデータ項目数の割合。	暗号化するデータ項目数/社内規約で定めた暗号化機能を要求するデータ項目数	0～1 1に近いほど良い	①	①	①	②③		④	⑤	⑤	○	○	○
			データの暗号化ー2	全データ項目数に対する,暗号化するデータ項目数の割合。	暗号化するデータ項目数/全データ項目数	0～1 小さいほど良い	①	①	①	②③		④	⑤	⑤	○	○	○
		セキュリティ事故発生時の追跡可能性,監査能力の評価	アクセス履歴管理率	全ユーザに対する,アクセス履歴を保存するユーザの割合。	履歴を取るユーザ数/全ユーザ数	0～1 1に近いほど良い	①	①	①	②③		④	⑤	⑤	○	○	
			データ・アクセス履歴管理率ー1	社内規約でアクセス履歴を要求するデータ項目数に対する,アクセスログ(更新ログ)を保存するデータ項目数の割合。	更新ログを保管するデータ項目数/社内規約で定めたアクセスログを要求するデータ項目数	0～1 1に近いほど良い	①	①	①	②③		④	⑤	⑤	○	○	
			データ・アクセス履歴管理率ー2	社内規約でアクセス履歴を要求するデータ項目数に対する,アクセスログ(更新ログと参照ログ)を保存するデータ項目数の割合。	更新ログおよび参照ログを保管するデータ項目数/社内規約で定めたアクセスログを要求するデータ項目数	0～1 1に近いほど良い	①	①	①	②③		④	⑤	⑤	○	○	
			データ・アクセス履歴管理率ー3	全データ項目数に対する,アクセス履歴(更新ログ)を保存するデータ項目数の割合。	更新ログを保管するデータ項目数/全データ項目数	0～1 1に近いほど良い	①	①	①	②③		④	⑤	⑤	○	○	
			データ・アクセス履歴管理率ー4	全データ項目数に対する,アクセス履歴(更新ログと参照ログ)を保存するデータ項目数の割合。	更新ログおよび参照ログを保管するデータ項目数/全データ項目数	0～1 1に近いほど良い	①	①	①	②③		④	⑤	⑤	○	○	
			システム操作履歴管理率	全操作数に対する,操作ログを保存する操作数の割合。	ロギング(操作ログを保存)する操作数/全操作数	0～1 1に近いほど良い	①	①		①②③		④	⑤	⑤		○	
			履歴保存期間	前述の履歴を保存する期間。 ・アクセス履歴 ・データ・アクセス履歴 ・システム操作履歴	保存期間	0～ 大きいほど良い	①	①	②	①②③		④	⑤	⑤		○	
		脆弱性の予防の評価	脆弱性指摘数	セキュリティ診断の実施による脆弱性の指摘項目数。	脆弱性指摘数	0～ 小さいほど良い	①	①			②③	④	⑤	⑤			
			セキュリティパッチの適用数の割合	すべてのパッチ提供数に対する,セキュリティパッチの適用数の割合。	パッチ適用数/パッチ提供数	0～ あらかじめ定めた適正値に近いほうが良い		①						⑤			

①:要求仕様の決定(ユーザの役割) ②:要求を実現するための仕様提示(ベンダSE,またはユーザSE) ③:設計仕様への盛り込み(および確認) ④:実装結果の確認 ⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(機能性 4/5)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
機能性	セキュリティ	脆弱性の予防の評価	セキュリティパッチの平均適用時間	セキュリティパッチの平均適用時間。	パッチ適用時間/パッチ提供数	0～あらかじめ定めた適正値に近いほうが良い		①						⑤			
		アクセス制御性の評価	不正アクセス数	全アクセスに対する，不正アクセスの割合。 ・ネットワーク ・ハードウェア ・ソフトウェア ・データ	物理的な不正アクセス数/全アクセス数	0～10に近いほど良い	①	①			②③	④	⑤	⑤			
			不正ログイン数	全ログインに対する，不正ログインの割合。	不正ログイン数/全ログイン数	0～10に近いほど良い	①	①			②③	④	⑤	⑤			
			不正アクセスの原因究明時間	不正アクセスの原因を究明するまでの時間。	原因究明時間	0～小さいほど良い		①						⑤			
			ユーザID登録時間	あらかじめ定めた時間に対する，ユーザIDを登録する時間の割合。	ユーザIDを登録する時間/あらかじめ定めた時間	0～1に近いほど良い		①		②③		④	⑤	⑤		○	
			ユーザID抹消時間	あらかじめ定めた時間に対する，ユーザIDを抹消する時間の割合。	ユーザIDを抹消する時間/あらかじめ定めた時間	0～1に近いほど良い		①		②③		④	⑤	⑤		○	
			なりすましのモニタリング・警告機能数	なりすましと思しき操作パターンを定義し，ソフトウェアが自発的に警告を発する機能の数。 「なりすまし」そのものを防ぐことは困難だが，アクセス管理で予防は可能。	用意した機能数	0～大きいほど良い		①		②③		④	⑤			○	
			なりすましのモニタリング・警告回数	上記の機能によって，自動的に警告を発した単位時間あたりの回数。	警告回数/運転時間	0～あらかじめ定めた適正値に近いほうが良い		①						⑤			
		データ保全性の評価	データ損傷の予防性(テスト時)	テストにおける，テストケースあたりのデータ損傷の発生回数。	デスト時のデータ損傷事故の発生回数/試験ケース数	0～10に近いほうが良い	①	①					⑤		○	○	
			データ損傷の予防性(運用時)	運用時における，単位時間あたりのデータ損傷の発生回数。	運用時におけるデータ損傷事故の発生回数/運転時間	0～0に近いほうが良い	①	①						⑤	○		

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(機能性 5／5)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
機能性	標準適合性	機能性標準適合性の評価	機能性標準適合率	整合を取る必要のある法規や標準などに対して，機能およびインターフェース，要件，設計，実装が適合している割合。 レビューや試験によって判断する。	規約・規則に適合する形で設計・実装されている機能・インターフェースの数/規約・規則に適合する必要のある機能・インターフェースの数	0～1 1に近いほど良い	①	①	②	③	④	④	⑤		○	○	○

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(信頼性 1/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
信頼性	成熟性	障害摘出の十分性の評価	推定潜在障害密度	ソフトウェアの規模に対する，潜在的障害件数(将来出現可能性がある障害の残存件数)の割合。 信頼度成長評価モデルなどで潜在的障害件数を予測する。	潜在的障害残存件数/ソフトウェアの規模	0～ 小さいほど良い	①	①	②	③	④	④	⑤		○	○	
		テストケース設定の十分性の評価	テスト密度	ソフトウェアの規模に対する，設定したテストケースの割合。	設定したテストケース数/ソフトウェアの規模	0～ 標準値に近いほうが良い	①	①	②	③		④	⑤			○	
		障害発見の十分性の評価	障害発見率	設定したテストケース数に対する，テストで発見された障害数との割合。	テストで発見された障害の件数/設定したテストケース数	0～ 標準値に近いほうが良い	①	①	②			④	⑤			○	
			障害発生密度	ソフトウェアの規模に対する，テストで発見された障害数との割合。	テストで発見された障害の件数/ソフトウェアの規模	0～ 標準値に近いほうが良い	①					④	⑤			○	
		障害解決状況の評価	障害解決率	テストで発見された障害の累計に対する，解決済みの件数の割合。	解決済みの件数/テストで発見された障害の累計件数	0～1 1に近いほうが良い	①					④	⑤		○	○	
			障害除去率	障害の発生件数に対する，これまでに修正を完了した件数の割合。	これまでの障害の修正完了件数/障害の発見件数	0～1 1に近いほうが良い	①					④	⑤		○	○	
		テストの網羅性/十分性についての評価	テストの網羅性	テストの網羅性を確保するために必要なテストの件数に対する，実際に実施したテストの件数の割合。	実施したテストの件数/実施すべきテストの件数	0～1 1に近いほうが良い	①				③	④	⑤		○	○	
			テストの成熟性	実施すべきテストの総件数に対する，テストに合格した件数の割合。	テストに合格した件数/実施すべきテストの総件数	0～1 1に近いほうが良い	①				③	④	⑤		○	○	
		仕様変更の評価	仕様変更率	全体の仕様の中，仕様変更されたものの割合。	対応することにした仕様変更の全件数/仕様の総件数	0～1 0に近いほうが良い	①					④	⑤				
			仕様変更収束率	対応しなければならない仕様変更に対する，未対応の仕様変更の割合。	未対応の仕様変更の件数/対応しなければならない仕様変更の総数	0～1 0に近いほうが良い	①					④	⑤				
		未解決懸案事項残存状況の評価	ペンディング件数率	発見された問題の総件数に対する，未解決で懸案事項として残っている件数の割合。	未解決懸案事項件数/総発生件数	0～1 0に近いほうが良い	①					④	⑤				
		運用中の障害発生頻度の評価	平均障害発生間隔	障害発生の平均時間間隔。	運転時間/障害発生件数	0～ 長いほうが良い		①	②				④	⑤	○		

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(信頼性 2/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
信頼性	障害許容性	全体の稼働停止頻度	機能停止回避性	全部の障害の中，その障害のために機能を停止した障害の割合。	1−(障害によって機能を停止した回数)/全障害件数	0〜1 1に近いほうが良い		①	②	③			④	⑤	○		
		障害パターンの解決	障害回避	テストで実行した障害パターン数の中，致命的，あるいは重大な障害を回避したパターンの件数の割合。	回避された致命的障害と重大な障害発生の件数/障害パターンの全テストケース数	0〜1 1に近いほうが良い		①	②	③		④	⑤		○	○	
		誤操作回避能力を備えた機能の実装数	誤操作回避	テストで実行した誤操作のパターンについてのテストの総件数に対する致命的，あるいは重大な障害を回避した誤操作のテストケース数の割合	回避された致命的障害と重大な障害発生件数/誤操作パターンのテストケース数	0〜1 1に近いほうが良い		①	②	③		④	⑤		○	○	
	回復性	可用性の評価	運転時間の割合	全体の時間(運転できた時間+修復に要した時間)の中，実際に稼働できた時間(運転できた時間)の割合。	操作できた時間/操作できた時間と修復に要した時間の合計	0〜1 1に近いほうが良い		①	②	③			④	⑤	○	○	
		障害による利用不可時間の評価	平均ダウン時間	障害のために利用できなかった平均の時間。	ダウンしていた時間の合計/機能を停止した総件数	0〜1 小さいほうが良い		①	②	③			④	⑤	○	○	
		障害からの回復時間の評価	平均回復時間	障害回復の平均の時間。	回復のために必要だった時間の合計/機能を停止した総件数	0〜1 小さいほうが良い		①	②	③			④	⑤	○	○	
		サービス提供再開頻度の評価	再開能力	再起動した回数の中，要求された時間内で再開できた回数の割合。	要求された時間内で再開した回数/再起動した回数	0〜1 1に近いほうが良い		①	②	③			④	⑤	○	○	
		自分自身を復元できるか	復元能力	要求されテストされた復元回数の中，成功した復元の回数との割合。	うまく復元できたケース数/要求事項ごとにテストした復元ケース数	0〜1 1に近いほうが良い		①	②	③			④	⑤	○	○	
		復元能力の程度の評価	復元の有効性	指定目標時間を要求された復元件数の中，目標復元時間を満足した復元件数の割合。	目標復元時間内に復元できたケース数/実施したケース数	0〜1 1に近いほうが良い		①	②	③			④	⑤	○	○	
	信頼性標準適合性	標準適合性の評価	信頼性標準適合性	信頼性に関して規定や規約に適合していることが要求されている項目の中，実際に適合している項目の割合。	適合項目数/適合要求項目数	0〜1 1に近いほうが良い		①	②	③		④	⑤		○	○	○

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(使用性 1/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
使用性	理解性	製品能力の理解性の評価	製品能力の理解度	説明書などを読んだ後で，ユーザが理解できた機能の全機能に対する割合。	理解できた機能数/全機能数	0～1 高いほど良い		①	②	③		④	⑤		○		○
		説明入手の容易性の評価	説明資料入手の容易性	情報システムを利用中にユーザが困難に遭遇し，そのときに実物説明をうまく見ることができた割合。	うまく見ることができた回数/トライした総回数	0～1 高いほど良い		①					④	⑤		○	
		実物説明の有効性の評価	実物説明の有効性	ユーザが実物説明を受けた後で，その情報を元に実際にうまく操作できた割合。	操作できた回数/トライした総回数	0～1 高いほど良い							④	⑤	○	○	
		機能の明確性の評価	機能の明確性	ユーザが利用を開始しようとする時点で，うまく情報システムを操作できた割合。	操作を試みた回数/総操作回数	0～1 高いほど良い					④		④	⑤	○	○	
		機能の理解度の評価	機能の理解度	ユーザが情報システムを使用するにあたって，理解できたメニューの全体に対する割合。	理解できたメニュー数/総メニュー数	0～1 高いほど良い					④		④	⑤	○	○	
		入出力の理解可能性の評価	入出力項目についての理解度	ユーザが理解できた入出力項目の，全体に占める割合。	理解できた入出力項目数/全項目数	0～1 高いほど良い		①	②	③	④		④	⑤	○	○	
	習得性	ユーザのイメージの特定	ユーザイメージを超える機能数	ユーザ(初心者/中級者/熟練者別)が想定する機能を超える情報システムの機能の数。	想定ユーザの理解を超えた機能数	0～小さいほど良い		①	②	③	④		④	⑤		○	
		機能の習得容易性の評価	機能理解に要する平均時間	ユーザが機能を理解できるまでの平均時間。	ユーザの機能習得に要した時間の平均	0～少ないほど良い		①	②	③	④		④	⑤	○		
			作業習得の平均時間	ユーザが効果的に作業できるようになるまでの平均時間。	ユーザが効果的に作業できるまでの平均時間	0～小さいほど良い					④		④	⑤	○		
			業務を習得できた人の割合	平均的能力の未経験者の中，一定時間内に業務遂行にさしつかえない習熟度に達することのできる人の割合。	パーセント/月，など	多いほど良い	①	①			④		④	⑤			
		説明文書とヘルプシステムの有効性の評価	説明文書/ヘルプシステムの有効性	説明文書，またはヘルプ機能を使用した後，正確に作業できた件数の割合。	ヘルプを利用した後うまく作業できた数/トライした総回数	0～1 高いほど良い	①	①	②	③	④		④	⑤	○	○	
		ヘルプアクセスの容易性の評価	ヘルプアクセスの容易性	問題が起きたときに，ヘルプ機能をうまく見つけられた割合。	正しいヘルプを見つけた回数/総トライ回数	0～1 高いほど良い					④		④	⑤	○	○	
		ヘルプシステムの利用頻度の評価	ヘルプシステムの利用頻度	どの程度頻繁にヘルプ文，またはマニュアルを見たかの回数。	ユーザが作業完了までにヘルプ，説明書などを参照した回数	0～少ないほど良い					④		④	⑤	○	○	

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(使用性 2/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
使用性	操作性	操作性評価の指標/目標の設定状況についての評価	操作性評価の指標/目標の設定度	操作性評価の項目とその目標値を決めた件数。	満足度調査の対象と目標評価項目(目標値の値を含む)	0～目標値に近い,または目標値以上が良い	①	①	②		④	④	⑤	⑤		○	
		誤り修正の容易性の評価	誤り修正の容易性	ユーザが誤りの修正をできた平均時間。	誤りの処理時間の平均	0～短いほうが良い	①	①	②	③	④		⑤	⑤		○	
		誤操作からの回復性の評価	誤操作から回復できた割合	誤操作から回復できた割合。	回復できた回数/誤操作件数	0～1高いほど良い	①	①	②	③	④		⑤	⑤		○	
		取り消し作業の可能性についての評価	取り消し作業の可能性	作業中に犯した誤りを容易に取り消しできた回数の割合。	犯した誤りを容易に取り消しできた回数/誤りを犯した総回数	0～1高いほど良い	①	①	②	③	④		⑤	⑤		○	
		利用時のデフォルト値の可用性の評価	デフォルト値の可用性	利用時にデフォルト値の変更を試みて,それに成功した割合。	容易に変更できた回数/変更を試みた回数	0～1高いほど良い	①	①	②	③	④		⑤	⑤		○	
		利用時のメッセージ理解性の評価	利用時のメッセージ理解性	ユーザが,システムを利用中に,そのシステムから出力されたメッセージを容易に理解できた割合。	容易に理解できた回数/メッセージ解読を必要とした回数	0～1高いほど良い				③	④		⑤	⑤	○	○	
		カスタマイズの可能性の評価	カスタマイズの可能性	ユーザが,自ら行う操作手順を容易にカスタマイズできた割合。	容易にできた回数/カスタマイズをトライした総回数	0～1高いほど良い	①	①	②	③	④		⑤	⑤	○	○	
	使用性標準適合性	標準適合性の評価	使用性標準適合率	使用性に関して規程や規約に適合していることが要求されている項目の中,実際に適合している項目の比率。	適合項目数/適合要求項目数	0～1高いほど良い	①	①	②	③	④		⑤	⑤	○	○	

①:要求仕様の決定(ユーザの役割) ②:要求を実現するための仕様提示(ベンダSE,またはユーザSE) ③:設計仕様への盛り込み(および確認) ④:実装結果の確認 ⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(効率性 1/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用	テスト量	テスト生産性
効率性	時間効率性(コンピュータシステムの効率)	コンピュータシステムの効率(リアルタイム処理)の評価	レスポンスタイム	ユーザの入力操作から応答までにかかる時間。	秒	0～少ないほど良い	①	①	②	③		④	④	⑤	○	○	
			スループット	一定の時間内に何件の処理を終えることができるか。	件数/時間。あるいは件数/秒	0～多いほど良い	①	①	②				④	⑤	○	○	
		コンピュータシステムの効率(バッチ処理)の評価	ターンアラウンドタイム	ユーザによる作業開始時(またはジョブの起動)から，求められた情報の出力を終了するまでの時間。	秒，分，あるいは時間	0～少ないほど良い	①	①	②				④	⑤	○	○	
			スループット	一定の時間内に何件の処理を終えることができるか。	件数/時間。あるいは件数/秒	0～多いほど良い	①	①	②				④	⑤	○	○	
			実行時間	コンピュータが作業を始めてから，終了するまでにかかる時間。	秒，分，あるいは時間	0～少ないほど良い	①	①	②	③			④	⑤	○	○	
	時間効率性(業務効率)	業務効率の評価	スループット	一定時間内に何件の業務を処理することができるか。	件数/時間の要素(日，時間，分，あるいは秒)	0～多いほど良い	①	①				④	④	⑤		○	
		デリバリータイムの評価	提供業務	情報の提供/伝達を目的とする業務の開始から，利用者の手元に情報/実体物が届くまでにどれだけの時間がかかるか。	時間(日，時間，分，あるいは秒)	0～少ないほど良い	①	①				④	④	⑤		○	○
		業務処理時間の評価	処理方式把握・準備時間(平均)	利用者が，ターミナルへの移動，端末のセットアップ，マニュアルの取り出し・参照，コード区分の準備など，作業の準備に要する時間。	秒，分	0～少ないほど良い	①	①				④	④	⑤			
			処理時間	マニュアル・ガイドなどを参照しながら，業務処理を行うのに要する時間。処理中のエラー訂正のための時間を含む。	秒，分	0～少ないほど良い	①	①				④	④	⑤		○	○
			結果確認・訂正時間	入力後の更新内容チェックなど，一定処理終了時のチェックおよび訂正作業に要する時間。想定エラー頻度を明示する。	秒，分	0～少ないほど良い	①	①				④	④	⑤		○	○
			リカバリー時間	担当者の業務が完結した後の訂正時間。想定エラー頻度を明示する。	秒，分	0～少ないほど良い	①	①				④	④	⑤		○	○

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(効率性 2/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
効率性	資源効率性	コンピュータシステム効率の評価	CPUタイム	特定の処理全体を行ううえで必要なCPU時間。あるいはオンラインシステム1件のトランザクションを処理するために必要とするCPU時間。	時間の要素(上は時間から,下はミリ秒,あるいはマイクロ秒)	0～少ないほど良い	①	①	②			④	④	⑤		○	
			メモリ容量	特定の処理を行う際に必要とするコンピュータの主記憶の容量。	記憶容量の大きさの要素(GB,MB,KB)	0～少ないほど良い	①	①	②			④	④	⑤		○	
			伝送	特定の処理を行う際に必要とするネットワークの伝送能力。	伝送能力の要素(上はGbps,下はKbps)	0～少ないほど良い	①	①	②			④	④	⑤		○	
			サーバなどのハードディスク容量	特定の処理を行う際に必要とするサーバなどのハードディスクの容量。	記憶容量の大きさの要素(TB,GB,MB)	0～少ないほど良い	①	①	②			④	④	⑤		○	
			入出力装置	特定の処理を行う際に必要とする入出力装置の台数。	台数	0～少ないほど良い	①	①	②			④	④	⑤		○	
			スペース	特定の処理を行うためのコンピュータなどを設置するうえで必要なスペース。	平方メートル	0～少ないほど良い	①	①	②			④	④	⑤			
		コンピュータシステム設置条件の評価	環境	特定の処理を行うためのコンピュータなどを稼働させるために必要な環境要素(電源容量,空調能力の要素,など)。	電源容量,空調能力の要素,など	0～少ないほど良い	①	①	②			④	④	⑤			
	効率性標準適合性	効率性標準適合率の評価	効率性標準適合率	標準化,社内規程類,法令などの全部の件数と,それらに適合している件数の割合。	標準化,社内規程類,法令などに適合している件数/それらの総件数	0～1 1に近いほど良い	①	①	②	③ ④	③ ④	④	④	⑤	○	○	

①:要求仕様の決定(ユーザの役割) ②:要求を実現するための仕様提示(ベンダSE,またはユーザSE) ③:設計仕様への盛り込み(および確認) ④:実装結果の確認 ⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(保守性 1／4)

品質特性	副品質特性	詳細分析ニーズ（測定目的）	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
保守性	解析性	活動記録保有能力の評価（故障原因の解析のため）	データログ実装率	故障の原因になった特定の操作を見つけるために，データログに記録することになっているデータ項目の中，実際に記録されているデータ項目数の割合。	実際に記録されたデータ項目数/記録することになっているデータ項目数	0～1 高いほど良い		①	②	③	④	④	⑤	⑤	○	○	
			状況監視データ取得成功率	ソフトウェアの状態を記録したモニタデータについて，保守担当者が取得しようとしたケースの中，取得できた割合。	モニタデータについて(1－取得できなかったケース数)/全ケース数	0～1 高いほど良い		①	②	③	④	④	⑤	⑤	○	○	○
		故障原因解析能力の評価	診断機能実装率	仕様で要求される診断機能の中，実装された診断機能の比率。	診断機能実装数/診断機能要求数	0～1 高いほど良い		①	②	③	④	④	⑤	⑤	○	○	
			故障原因判明率	登録されている故障の中，原因が分かっている故障の数の割合。	(1－原因不明の故障数)/全故障数	0～1 高いほど良い		①	②	③	④	④	⑤	⑤	○	○	○
			故障解析時間	故障解析にかかった平均時間。	故障解析に要した時間数/故障数(原因が判明した故障のみの統計量)	0～ 短いほど良い					④	④	⑤	⑤	○	○	○
		解析容易性の評価	保守ドキュメント充足	解析容易性向上につながる保守ドキュメントについて，実際に用意できているドキュメント数。 以下に具備候補の保守ドキュメント例を挙げる。 ・機能仕様書 ・DBクロスリファレンス ・データ項目クロスリファレンス ・トランザクションリファレンス ・変更手順書(組織変更，制度変更，限度額変更など)	実際に用意できている保守ドキュメント数	0～ 重複しない内容で，多いほど良い	①	①	②	③	④	⑤	⑤	⑤			
			トレースツール利用率	実装機能をトレースする際に，トレースツールを利用できた機能数の割合。	ツールを用いてトレースできた実装機能数/トレースした実装機能数	0～1 高いほど良い	①		②	③	④	⑤	⑤	⑤		○	
			プログラムソースコメント率	プログラムソースに組み込んだコメント行の割合。	実装したコメント率/組織で定めている標準コメント率	0～1 高いほど良い	①		②		⑤						

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(保守性 2/4)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用	テスト量	テスト生産性
保守性	変更性	変更を制御する能力の評価	変更内容文書化率	プログラムコードを変更した機能中,変更内容を仕様書やコメントなどに文書化して記録している割合。変更文書がレビューされるという考えに基づく。	文書化されてレビューの対象になった機能数/プログラム変更のあった機能数	0～1 高いほど良い	①	①	②	③	④				○	○	
			変更履歴記録率	ソフトウェアの変更をトレースするための変更履歴について,記録対象件数のうち,実際に記録した件数の割合。	ソフトウェアの変更履歴の記録について,実際の記録データ数/計画で記録すべきデータ数	0～1 高いほど良い	①		②			⑤	⑤	⑤			
			構成管理効率	保守作業における各種資源(プログラムやドキュメント,使用データ(DBを含む))のバージョン管理について,機械管理の実現割合。	プログラム,ドキュメント,使用データの3種のうち,バージョン管理が機械にて実現できている資源数/3	0～1 高いほど良い	①		②	③		⑤	⑤	⑤		○	
			変更範囲検証ツール具備	変更範囲や変更対象プログラムを特定するためのツールの具備状況。コードクローン検出ツールなど。	具備しているツール数	0～多いほど良い	①		②	③	④	⑤	⑤	⑤		○	
		変更容易性の評価	変更生産性	ソフトウェア変更量に対する変更作業時間の割合。	変更に要した作業時間/変更規模の平均	0～短いほど良い			②			⑤	⑤	⑤			
			パラメータ修正成功率	パラメータ化されたソフトウェア機能について,保守者がパラメータ修正でソフトウェアの変更をしようとしたケースの中,成功する割合。	パラメータ修正による変更について(1－失敗したケース数)/やろうとしたケース数	0～1 高いほど良い	①		②	③	③	⑤	⑤				
			母体システムの構造化度合い	「対象システムのステップ数」と,「対象システムを再構築した場合の見積りステップ数」との比較。	対象システムのステップ数/再構築した場合の見積りステップ数	1～1に近いほど要求度合いが高い(1より小さい場合は,1とみなす)	①	①	②	③	④	⑤	⑤	⑤			
		変更に対応できる早さの能力評価	問題解決までの平均時間	使用者が変更要求を問題報告書とともに保守者に提示してから,変更後の改訂版を報告書とともに受け取るまでの経過時間。	変更要求から変更終了報告書までの所要経過時間の平均	0～短いほど良い			②			⑤	⑤		○	○	
			変更実施の平均時間	保守者が故障の原因を見つけてから,変更後の改訂版を報告書とともに使用者に渡すまでの経過時間。	故障の原因が判明してから変更終了報告までの所要経過時間の平均	0～短いほど良い			②			⑤	⑤		○	○	

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(保守性 3／4)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
保守性	安定性	保守作業でのバグの少なさの評価	修正によるバグ混入率	変更の影響の測定。 ソフトウェアの修正後に見つかった修正の影響によるバグの，修正回数あたりの発生頻度。	修正後に検出されたバグ数/修正回数	0～小さいほど良い	①			③		⑤	⑤	⑤	○	○	
			故障発生率	変更成功率の測定。 単位期間の間に故障が発生する頻度を時系列に監視して傾向を評価する。	故障発生件数/時間	0～小さいほど良い，低減しているのが良い	①					⑤	⑤	⑤		○	
			母体品質	保守母体システムに残存するバグ量。母体システムの過去障害統計より残存量を類推する。	バグ量(規模)/母体システム全体規模	0～小さいほど良い	①		②	③	④	⑤	⑤	⑤		○	○
			故障発生低減率	変更成功率の測定。 故障が発生する頻度の保守実施前後の変化。	保守後の故障発生率/保守前の故障発生率	0～小さいほど良い	①					⑤	⑤	⑤	○	○	
			修正失敗率	修正(本番リリース)後，一定期間内に故障・追加要求などの発生により再修正が必要になった頻度。	再修正を行った回数/全修正回数	0～小さいほど良い	①					⑤	⑤	⑤	○		
		故障耐久能力の評価	自動リカバリー機能充足率	実装できている機械による自動リカバリー機能の割合。	実装できている自動リカバリー機能数/故障時にリカバリー処置が必要な機能数	0～1高いほど良い	①	①	②	③	④	⑤	⑤	⑤		○	
		システム保有データ整合性の評価	整合性判定時間	保守対象システムが保有するデータについて，データ間の整合性確認を行うために必要な時間。機械化により正確に効率よく確認することができる。	データ間整合性判定に費やした時間/整合性判定を行った件数	0～小さいほど良い			②	③		⑤	⑤	⑤		○	
	試験性	試験の再開能力評価	試験実行の中断・再開機能の提供度	試験の再開能力の測定。保守後のチェックポイントでの確認のため，試験実行を任意のタイミングで中断した後再開できる機能の提供度合い。	試験実行を途中で中断して再開させることの，可能な機能数/中断させる機能数	0～1高いほど良い		①	②	③	④	⑤	⑤	⑤	○	○	
		他システム連携の自律試験能力の評価	他システム連携テストのスタブ化	試験容易性の自律性の測定。他システムとの連携テストで，全テスト件数の中，スタブでシミュレートしてテストした割合。	他システムとの連携テストに関して，スタブ利用のテスト件数/全テスト件数	0～1高いほど良い		①	②	③	④	⑤	⑤	⑤	○	○	
		再試験能力の評価	再試験時生産性	再試験効率の測定。故障が解決しているか否かをテストするために費やす時間。	故障の解決にあたったテスト時間/障害解決件数	0～小さいほど良い		①	②	③		⑤	⑤	⑤	○	○	

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(保守性 4/4)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
保守性	試験性	内蔵試験機能の能力評価	内蔵試験機能の実装率	内蔵試験機能の完全性の測定。保守者のために用意した内蔵試験機能について,要求のある機能のうち,要求どおり実装した機能の割合。	内蔵試験機能の実装機能数/要求機能数	0～1 高いほど良い		①	②	③	④	⑤	⑤	⑤	○	○	
			内蔵試験機能の有効度	内蔵試験機能の可用性の測定。保守者のために用意した内蔵試験機能について,使用する機会の中,うまく使えた機能の割合。	内蔵試験機能のうまく使用できたケース数/使用機会数	0～1 高いほど良い		①	②	③	④	⑤	⑤	⑤	○		○
		試験容易性の評価	保守環境整備度合い(ツール)	保守作業での試験において,試験ツール利用により「実施」「検証」作業を効率よく正確に実施できた割合。	ツール利用できた試験ケース数/実施した全ケース数	0～1 高いほど良い	①		②	③	④	⑤	⑤	⑤	○	○	
			保守環境整備度合い(流用)	保守作業での試験において,本番と同等環境(データ,CPU性能,実施時間帯)下で実施することができた割合。	本番と同等の環境で実施した試験ケース数/実施した全ケース数	0～1 高いほど良い	①		②	③	④	⑤	⑤	⑤	○		○
	保守性標準適合性	標準適合性の評価	保守性標準適合率	保守性に関して規格や規約に適合していることが要求されている項目の中,実際に適合している項目の割合。 以下に組織レベルで保有すべき規約(標準化)類を列挙する。	適合項目数/適合要求項目数	0～1 高いほど要求度合いが高い									○	○	
				①ソフトウェア設計指針			①			③							
				②ドキュメント作成基準			①			③	④						
				③コーディング規約			①				④						
				④(工程別)テスト実施基準,同実施手順			①					⑤	⑤				

①:要求仕様の決定(ユーザの役割)　②:要求を実現するための仕様提示(ベンダSE,またはユーザSE)　③:設計仕様への盛り込み(および確認)　④:実装結果の確認　⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(移植性 1/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
移植性	環境適応性	ソフトウェアを異なる環境に適用できる能力の評価	データ構造への適応性	データ構造の変更後,制限なく操作できるデータ構造の数をカウントし,これを,データ構造の変化に適応する能力を要求されているデータ構造の総数と比較する。	データ構造の変更後に制限なく操作できるデータ構造の数/データ構造の総数	0～1 高いほど良い	①	①	②	③		④	④	⑤	○	○	
			ハードウェア環境への適応性	複数のハードウェア環境で所要の結果を達成できる機能の数をカウントし,これを,ハードウェア環境の変化に適応するべく要求されている機能の数と比較する。	複数のハードウェア環境で所要の結果を達成できる機能の数/ハードウェア環境の変化に適応するべく要求されている機能の総数	0～1 高いほど良い	①	①	②	③		④	④	⑤	○	○	
			ソフトウェア環境への適応性	複数のシステムソフトウェア環境で所要の結果を達成できる機能の数をカウントし,これを,システムソフトウェア環境の変化に適応するべく要求されている機能の数と比較する。	複数のシステムソフトウェア環境で所要の結果を達成できる機能の数/システムソフトウェア環境の変化に適応するべく要求されている機能の総数	0～1 高いほど良い	①	①	②	③		④	④	⑤	○	○	
			組織環境への適応性	複数の組織および事業環境の下で所要の結果を達成できる機能の数をカウントし,これを,組織環境への変化に適応するべく要求されている機能の数と比較する。	複数の組織および事業環境の下で所要の結果を達成できる機能の数/組織環境への変化に適応するべく要求されている機能の総数	0～1 高いほど良い	①	①	②	③		④	④	⑤	○	○	
			ユーザ利便性のポーティング	ユーザがソフトウェアを特定の操作環境に適用させようとしているときの,適用させるために必要な時間を測定する。	ユーザがソフトウェアを特定の操作環境に適用させようとしているときの,適用させるために必要な時間	0～ 短いほど良い	①						④	⑤	○		
	設置性	ソフトウェアをユーザの特定の環境に導入できる能力の評価	導入の努力	自動化によって導入できるステップの数をカウントし,これを,規定の導入ステップの数と比較する。	自動化導入ステップの数/必要な導入ステップ数	0～1 高いほど良い	①						④	⑤	○	○	
		ソフトウェアをユーザの特定の環境に導入できる能力の評価	導入の柔軟性	実行したカスタマイズ可能な導入操作の数をカウントし,これを,カスタマイズできるように要求されている導入操作の数と比較する。	規定のとおり実行したカスタマイズ可能な導入操作の数/必要な,カスタマイズ可能な導入操作の数	0～1 高いほど良い	①						④	⑤	○	○	
	共存性	同じハードウェアを共有する他のソフトウェアと共存できる能力の評価	利用可能な共存	製品が共存できる他のソフトウェアの数をカウントし,これを,共存を必要とする稼働環境における他のソフトウェアの数と比較する。	製品が共存できるエンティティの数/共存を必要とする環境におけるエンティティの数	0～1 高いほど良い	①						④	⑤	○	○	

①:要求仕様の決定(ユーザの役割) ②:要求を実現するための仕様提示(ベンダSE, またはユーザSE) ③:設計仕様への盛り込み(および確認) ④:実装結果の確認 ⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(移植性 2/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
移植性	置換性	他の特定のソフトウェアの代わりにこのソフトウェアが使用できる能力の評価	データの継続的使用	ソフトウェアを交換後も使用を継続できるデータ項目の数をカウントし,これを,ソフトウェア交換後,古いソフトウェアが使用していたデータから,改めて使用する必要のあるデータ項目総数と比較する。	交換後に継続して使用できるデータ項目の数/古いソフトウェアから使用する必要のある古いデータ項目の総数	0～1 高いほど良い	①	①	②	③		④	⑤		○		
			機能の継続性	同じ結果をもたらす新しいソフトウェアの機能の数をカウントし,これを,古いソフトウェアの機能の数と比較する。	同じ結果をもたらす新しいソフトウェアの機能の数/古いソフトウェアの機能の数	0～1 高いほど良い	①	①	②	③	④	④	⑤	⑤	○		
	再利用性	再利用対象の資産を活用できる能力の評価	再利用性の実施	再利用ライブラリで管理されている,再利用できる対象資産の数をカウントする。	再利用ライブラリで管理されている再利用対象の資産の数	0～ 多いほど良い	①	①	②	③	④ ⑤						
	移植性標準適合性	移植性標準適合性の評価	移植性標準適合率	該当する規則,規制および規約に適合している項目数をカウントし,仕様書で適合性を要求している項目数と比較する。	移植性適合性に関して正しく実行した機能の数/適合性項目の総数	0～1 高いほど良い	①	①	②	③		④	⑤		○		

①:要求仕様の決定(ユーザの役割)　②:要求を実現するための仕様提示(ベンダSE,またはユーザSE)　③:設計仕様への盛り込み(および確認)　④:実装結果の確認　⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(運用性 1/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
運用性	運用サービスの品質目標(SLA)	サービスの提供時間についての評価	サービス提供(実施)時間についての割合	要求定義で明確に定義されたサービス提供時間に対する,実際のシステムのサービス時間の割合。	サービス提供実績時間/事前定義されたサービス提供時間	0～1 高いほど良い		①	②	③			④	⑤			
		システムの稼働率についての評価	システムの稼働率についての割合	業務要件で許容される(または目標とする)システム稼働率に対する,一定期間内のシステム稼働率の割合。 構成要素別とシステム全体の稼働率が存在する。	一定期間の実績稼働率/事前定義された実績稼働率	0～1以上が望ましい	①	②	②	③	④		⑤	⑤	○	○	
		システムのMTBFについての評価	現実のMTBFについての割合	業務要件で許容される(または目標とする)MTBFに対する,一定期間内のMTBFの割合。 構成要素別とシステム全体のMTBFが存在する。	一定期間のMTBF実績/事前定義されたMTBF	0～1以上が望ましい		①	②					⑤	○		
		オペレーションミスについての評価	ミスオペレーションの割合	目標回数に対する,運転要員のミスオペレーションおよび運転管理の指示ミスによるミスオペレーションの回数の割合。	ミスオペレーションの実績回数/ミスオペレーションの目標回数	0～小さいほど良い	①	①						⑤			
		重大ミスオペレーションについての評価	重大ミスオペレーションの割合	目標回数に対する,業務に大きな影響を与える重大事故(経営レベルに報告するもの)の発生回数の割合。	重大ミスオペレーションの実績回数/重大ミスオペレーションの目標回数	0～小さいほど良い	①	①						⑤			
	運用容易性	運転開始条件などの明確化の評価	運転開始条件などの明確化の割合	運転の開始,中断,終了などで条件を明確にされたケースと,本来明確にしなければならないケースの割合。	明確化されたケース/明確化しなければならないケース	0～1 高いほど良い	①	①	②	③	④		④	⑤		○	
		オペレータの介入オペレーションについての評価	介入オペレーションの最小化	運転中のオペレータの介入の回数。 介入がないことが望ましい。	オペレータの介入操作の回数	0～小さいほど良い	①	①	②	③			④	⑤		○	
		介入操作の容易性についての評価	介入オペレーションの割合	操作期待の総件数に対する,介入操作に問題がないと認めた条件数の割合。	操作に問題がないと認めた条件数/操作期待件数	0～1 高いほど良い	①	①	②			④	④	⑤		○	
		運用体制構築に関わる評価	運用体制構築のための要件確定の割合	事前に定義や明確化された項目数に対する,文書化項目の明確化,運用スキル定義,引継ぎ要件の明確化など,運用引継ぎ時に明確にされた項目の割合。	運用引継時に定義や明確化された項目/事前に定義や明確化された項目	0～1 高いほど良い		①	②				④	⑤			
	障害対策	異常検知能力の評価	異常を検知できる条件の割合	総必要条件に対する,異常であることをソフトウェアで見きわめられる機能数の割合。	組み込み数/必要条件数	0～1 高いほど良い	①	①	②	③	④	④	④	⑤		○	

①:要求仕様の決定(ユーザの役割) ②:要求を実現するための仕様提示(ベンダSE,またはユーザSE) ③:設計仕様への盛り込み(および確認) ④:実装結果の確認 ⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(運用性 2/2)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
運用性	障害対策	異常時に障害を回避できる能力の評価	中断を回避できた回数の割合。	全異常発生回数に対する，一定期間内に処理を回避できた回数の割合。 全システムを通して，異常現象とアクションの関係の明確化が必要。	中断を回避できた回数/異常発生回数・期間	0～1 小さいほど良い	①	①	②	③			④	⑤		○	
		障害対策の適正化，容易化の評価	障害対策でのミスオペレーション発生数の割合	全障害発生件数に対する，障害対策でのミスオペレーション発生数の割合。 障害対策のアクションが容易かつミスが起こりにくいことが必要。	ミスオペレーション発生数/全障害数	0～1 小さいほど良い	①	①	②	③			④	⑤			
		予防訓練の実施についての評価	予防訓練の実施の割合	あらかじめ定義された予防訓練数に対する，実施した予防訓練数の割合。 運用フェーズで実施する予防訓練の内容を定義することが必要。	実施した予防訓練数/定義された予防訓練数	0～ 高いほど良い	①		②				③ ④	⑤			
		障害発生に伴う広義の中断時間の評価	中断時間の目標に対する割合	中断の目標時間に対する，障害発生に伴う広義の中断時間の割合。 バックアップ操作，リカバリ/リスタート，MDT(Mean Down Time)などの障害時間指標の整備が必要。	障害発生時の実際時間/目標時間	0～ 小さいほど良い	①	①	②			④	④	⑤	○	○	
	災害対策(DR)	広域災害対策の評価	災害の復旧計画日数に対する，実際の復旧に要した日数の割合	あらかじめ定義した日数に対する，災害の発生に伴うシステムの不稼働状態から，正常またはフェールソフト状態で稼働するまでに要した日数の割合(広域災害の割合)。	実際に稼働できるまでの日数/定義された日数	0～ 小さいほど良い	①	①	②				④	⑤		○	
		局所災害対策の評価	災害の復旧計画日数に対する，実際の復旧に要した日数の割合	あらかじめ定義した日数に対する，災害の発生に伴うシステムの不稼働状態から，正常またはフェールソフト状態で稼働するまでに要した日数の割合(局所災害の割合)。	実際に稼働できるまでの日数/定義された日数	0～ 小さいほど良い	①	①	②				④	⑤		○	

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(障害抑制性 1/3)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
障害抑制性	発生防止	品質と価格の関係の評価	品質評価値	高品質ソフトウェアを開発するに当たり,品質について通常のシステムと区別した目標を設定し,その分開発予算を増額してプロジェクトを管理する必要がある。 所定の品質目標を達成するための基準の単価をあらかじめ定めておき,その単価と今回の開発での予算単価の割合。	今回の開発での予算単価/品質目標評価基準値に対する単価	0～1に近いほど良い	①	①			④	④	④	⑤			
		工期と価格の関係の評価	工期評価値	高品質ソフトウェアを開発するに当たり,工期について通常のシステムと区別した目標を設定し,その分開発予算を増額してプロジェクトを管理する必要がある。 所定の工期目標を達成するための基準の単価をあらかじめ定めておき,その単価と今回の開発での予算単価の割合。	今回の開発での予算単価/工期目標評価基準値に対する単価	0～1に近いほど良い	①	①			④	④	④	⑤			
		要求仕様書の作成についての評価	要求仕様書書式活用度	要求仕様書の書式を定め,それに基づいてこの文書を作成する必要がある。 「要求には理由を付ける」「仕様の総数を管理し,仕様変更率を抑える」などが,この書式に伴って実施されるべき事項。	1.要求仕様書書式を活用している/0.いない	0/1 1が望ましい	①	①	②	③		④	④	⑤			
		イレギュラー処理の実装についての評価	イレギュラー処理実装率	イレギュラー処理(エラー処理)を一次対策,二次対策に分けて明確にし,二次スクリーニングについての実装を明確に記述する必要がある。 二次スクリーニングについて記述する必要がある箇所に対する,実際に二次スクリーニングが定義された箇所の割合。	二次スクリーニングの実装箇所数/二次スクリーニングが必要な処理箇所の数	0～1 1に近いほど良い	①	①		①	④		⑤			○	
		サブシステム間のトレース機能	サブシステム間のトレース機能実装率	障害の原因を早期に特定するために,サブシステム間の正当性をトレースできる仕組みが必要である。 このトレース機能が必要とされる箇所に対する,実際に実装された箇所の割合。	実装された箇所の数/トレース機能を必要とする箇所の数	0～1 1に近いほど良い		①	②	③	④		⑤			○	
		テスト環境準備状況の評価	テスト環境の準備状況	高品質ソフトウェアを稼働させるためには,365日24時間,本番環境と同じ状況で常時テストできる環境(テストツールなどを含む)を保守SEのために提供することが必要である。 その準備状況を問う。	2.データベース含めての24時間使用環境あり/1.限られた資源を随時使用可能/0.限られた資源を限られた時間だけ使用可能	2/1/0 2が望ましい		①	②				④	⑤		○	○

①:要求仕様の決定(ユーザの役割)　②:要求を実現するための仕様提示(ベンダSE,またはユーザSE)　③:設計仕様への盛り込み(および確認)　④:実装結果の確認　⑤:最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(障害抑制性 2／3)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解　釈	プロセス								ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
障害抑制性	発生防止	テストカバレージについての評価	テストカバレージ率	ソフトウェアの品質を確保するために，十分なテストを実施しなければならない。 プログラムの全ステートメント数に対する，使用したテストデータで1回以上通過したステートメントの割合。	テストで1回以上通過したステートメント数/総ステートメント数	0～1 1に近いほど良い	①	①			④	⑤	⑤			○	
		バックアップ機への切替についての評価	バックアップ機への切替熟練度	バックアップ装置への切替，戻しが容易にできるように，常日頃訓練が実施されていなければならない。 その習熟度を問う。	1.常日頃実施しているので問題ない/0.切替に不安を感じる	0／1 1が望ましい	①						④	⑤			
		リスク確認についての評価	リスクの解決率	開発前，および本番稼働前に，システムのどこに障害が発生する可能性があるのかを関係者間で十分に議論し，リスクを洗い出して対応を明確にし，対策を立てておくことが必要である。 指摘されたリスクの総数に対する，対応済みのリスクの数の割合。	対策済みのリスクの数/指摘されたリスクの総数	0～1 1に近いほど良い				② ③		① ②	④	⑤			
		テスト確認体制の評価	第三者によるテスト確認体制－1	高品質ソフトウェアでは，第三者のテスト専門機関に委託して，システム障害の可能性についての分析を実施する必要がある。 まず，その実施状況を問う。	1.第三者の確認を実施済み/0.第三者の確認を実施していない	0／1 1が望ましい				③		③ ④ ⑤	④ ⑤	⑤		○	○
			第三者によるテスト確認体制－2(指摘事項確認率)	前設問で実施済み(1)の場合に，対策が必要と指摘された箇所の総数に対する，実際に対策を実施した箇所の割合。	対策を実施した数/指摘された総数	0～1 1に近いほど良い				③		③ ④ ⑤	④ ⑤	⑤		○	○
		妨害に対する対策の評価	防御処置の対策率	高品質ソフトウェアでは，外部から侵入してくる妨害入力に対する処置を明確にし，対策を立てておかなければならない。 防御処置が必要な機能の総数に対する，実際に対策を実施した機能の数の割合。	対策を実施した数/対策が必要な機能の総数	0～1 1に近いほど良い	①	①	②	③	④	④	⑤	⑤		○	
	障害拡大防止策	稼働状況の評価	情報システムの稼働率	稼働すべき総時間に対する，実際に稼働した時間の割合。	稼働した実際の時間数/稼働すべき総時間数	0～1 1に近いほど良い	①			②		④	④	⑤	○	○	
		停止防止対策の実施状況の評価	停止時間についての目標達成率	停止時間を極力短くする停止防止の対策実施による効果の把握。 総停止回数に占める，停止防止の対策による効果で停止時間の目標内に回復できた回数。	停止時間の目標内に回復できた回数/総停止回数	0～1 1に近いほど良い	①			②		④	④	⑤	○	○	

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

「非機能要件の把握・確認とテスト見積りへの影響表」(障害抑制性 3/3)

品質特性	副品質特性	詳細分析ニーズ(測定目的)	測定項目	項目定義と測定方法	測定尺度と導出式	解釈	プロセス							ISO9126(○該当)	見積り変動対象(○該当)	
							組織統制	要求定義	要件定義	設計	プログラム作成	システムテスト	運用テスト	保守・運用		テスト量	テスト生産性
障害抑制性	障害拡大防止策	利用者に迷惑をかけていないことの評価	稼働品質率	利用者に，障害発生による迷惑をかけていないことの確認。 総資産規模に対する，障害で利用者に迷惑をかけた回数の割合。	利用者に迷惑をかけた回数/資産規模	0～0に近いほど良い	①	①	②	③	④	④	④	⑤		○	
		稼働初期に起きる故障対策の評価	稼働初期故障対策率	システムが立ち上がらない場合の原因を特定する仕組みを準備していることの確認。 仕組みとしての必要な全機能数に対する，初期状況の確認を実装した機能の数の割合。	初期状況の確認を実装した機能の数/仕組みとしての必要な全機能数	0～1 1に近いほど良い	①	①	②	③		④	④	⑤		○	
		取り扱い可能なデータ件数把握についての評価	取り扱い可能なデータ件数把握の有無	システムの稼働前に，取り扱い可能な入力データの件数の確認を行う必要がある。 その実施状況を問う。	1. 確認を行っている/0. 確認を行っていない	0/1 1が望ましい	①	①	②	③	④	④	④	⑤		○	
		既存システムとの差の確認状況の評価	変更管理確認率	JCL，プログラム，データベースなどについて，既存システムと新システムとの差を確認する必要がある。 変更した箇所のすべての数に対する，確認済みの数の割合。	確認済みの数/変更した全箇所数	0～1 1に近いほど良い	①							⑤			
		ハードウェアのアラームへの対応状況の評価	ハードウェアのアラームに対する対応実行率	ハードウェアの故障警告回数に対して，アクション開始基準と手順を定めて実行する必要がある。 ハードウェアの故障発生総数に対する，基準を守ってアクションを実施した回数の割合。	基準を守ってアクションを実施した回数/ハードウェアの故障発生総数	0～1 1に近いほど良い	①							⑤			
		他社製ソフトウェアの監視状況の評価	他社製ソフトウェア監視状況率	他社製ソフトウェアを使う場合は，現時点でどのようなバグ報告があり，対策がどこまでとられているのかを把握しておく必要がある。 他社製のソフトウェアを利用している総数に対する，確認を実施しているソフトウェアの数の割合。	確認を実施しているソフトウェアの数/他社製のソフトウェアを利用している総数	0～1 1に近いほど良い	①							⑤			
		ミスオペレーションの防止策の有効性の評価	ミスオペレーション率	オペレータのミスオペレーションに対して，歯止めを利かせる方法を十分に用意しておく必要がある。ここではまず，ミスオペレーションの発生率を把握する。 総資産規模に対する，ミスオペレーションの発生回数の割合。	ミスオペレーションの発生回数/総資産規模	0～1 0に近いほど良い	①							⑤			

①：要求仕様の決定(ユーザの役割)　②：要求を実現するための仕様提示(ベンダSE，またはユーザSE)　③：設計仕様への盛り込み(および確認)　④：実装結果の確認　⑤：最終確認

用語解説

アジャイル開発プロセス

ソフトウェアをすばやく，臨機応変にむだのないように開発することを目的とした開発プロセスの総称。

受入れテスト

システムが，ユーザのニーズ，要件，ビジネスプロセスを満足するかをチェックするための公式のテスト。

このテストにより，システムが承認基準を満たしているかを判定したり，ユーザ，顧客，その他の認可主体がシステムを承認するかしないかを判定する。

受入れテストは，システムを使う顧客やユーザが実施することが多い。また，ステークホルダーが参加することもある。

受入れテストの目的は，システム全体，システムの一部，非機能的な特性が正しいことを確認することにあり，欠陥を検出することは主目的ではない。受入れテストは，システムが稼働できるかをテストするものであり，最終テストである必要はない。例えば，受入れテストの後で，大規模システム統合テストを実施することがある。

受入れテストは，通常，以下の形がある。

- ユーザ受入れテスト

  システムがビジネスで使えるかをユーザがテストするもの。

- 運用受入れテスト

  システム管理者による受入れテスト。これには以下がある。

  - バックアップ/リストアのテスト
  - 災害復旧テスト
  - ユーザマネジメント
  - 保守
  - セキュリティの脆弱性の定期的チェック

- 契約，および規程による受入れテスト

  契約による受入れテストは，カスタムメイドのプロダクトを開発する

場合，契約書に記述した判定条件にしたがって検証する。

受入れ条件は，契約締結時に決めてある。

規程による受入れテストは，法律や安全基準などに合致しているかを検証する。（「ソフトウェアテスト標準用語集」より）

運用テスト

システム開発において，開発者によるテストが終了した段階でシステムの運用者によって行われるテストで，実際の稼働状況において問題なく動作するかどうかを検証する。

開発工程

本ガイドブックでは，次のとおり設定している。

- 新規開発：新規開発の場合は，一般的なウォーターフォールモデルに準拠した，システム化の方向性，システム化計画，要件定義，設計，製作，システムテストを設定。
- 保守：要件定義(システム変更計画，現行システムの理解を含む)，設計，製作，テストを設定。

機能規模

機能要件を定量化して得られるソフトウェアの規模。

（「JIS X 0135-1：1999」より）

既存システム

あるシステムに対して機能変更(削除を含む)や機能追加を行う際に，手を加える対象となる元のシステムのこと。

禁則処理

組み合わせの網羅において，組み合わせることができないパラメータがある場合，それらの組み合わせを除外すること。禁則を除外せずに割り付けると禁則だらけのテストデータとなり，テストが実施できなくなる。

組合せ網羅

複数因子(項目，機能など)の全水準(とり得る値，条件など)について組み合わせることができる場合を網羅すること。

組込みソフトウェア

電子機器などに組み込まれて，製品の動作を制御するシステム。携帯電話や家電製品，カメラ，自動車など，コンピュータ制御を行う製品に搭載されている。

グレーボックステスト(グレーボックス)

「ブラックボックステスト」の弱点を「ホワイトボックステスト」の観点から補完するテスト技法。

ブラックボックス的観点から考察すると多数のバリエーションのテストが必要となるテスト項目(例：不連続なコード値の妥当性チェックなど)について，テスト対象ソフトウェアの内部パス，構造，設計情報を知ったうえで，テスト量(テストケース数など)を削減して，より少ないテスト量で欠陥が出そうな箇所のテストを行うというもの。

欠　陥

本ガイドブックでは，次のとおり設定している。

障害の原因(「バグ」の概念に最も近い)。技術者，プログラマの誤認や誤解など，作業者の認識に関わる問題が原因となって，仕様，設計，プログラムなどの成果物上の記述内容に現れた誤りや矛盾，表現上の問題を指す。

欠陥密度

ソフト規模(サイズ)あたりの欠陥数を表すメトリクス。ソフトウェアの信頼性を評価する尺度として使用される。

検出欠陥密度

システムドキュメントのレビューや各フェーズでのテストで検出された欠陥数の度合いを表すメトリクス。

コントロール

計画と実績の比較，差異分析，プロセスが改善する傾向の評価，代替案の評価，および必要に応じた適切な是正処置の提言などを行うこと。

（「PMBOK」の定義より）

残存欠陥密度

レビューやテストなどで検出されないままとなっている欠陥数の度合いを表すメトリクス。

システムテスト

統合テストの後で，システム全体とその特性を評価するために実施する総合的なテスト。通常は，システムの要件および設計に基づいて実施する。

障　害

本ガイドブックでは，次のとおり設定している。

システムが期待されるサービスを提供できなくなること。欠陥の混入が原因となって，実行結果が仕様や要求などで定義された「期待される結果」と異なる現象として表出し，検知されたものを指す。

信頼度成長曲線

ソフトウェアの信頼性の評価に用いる曲線。例えば，縦軸に欠陥の累積個数，横軸にテスト項目消化数をとると，一般に消化数が増えるごとに水平な直線（欠陥が収束していく状態）になる。

単体テスト

個々のモジュールを検証するために実施するレベルのテスト。モジュールの作成者が，ホワイトボックステストを利用して行われる場合が多い。

直交表

任意の２因子について，その水準のすべての組合せが同数回ずつ現れるという性質をもつ実験のための割り付け表。

一般に多元配置の実験では，少なくとも因子の水準数の積の回数だけ実

験数が必要になり，因子数が多くなると実験回数は膨大な数になってしまう。ところが，求める交互作用が少なければ，直交表を用いることによって，多くの因子に関する実験を比較的少ない回数で行うことができる。

この特性を利用して，ソフトウェアテストにおいて2因子間の組み合わせをすべて網羅する最小のテストケースを作成するために利用する方法がペア構成テスト(直交表)である。

テストカバレッジ

テストセット(テストケースのグループ)によってシステムのどの部分が実際に実行されたか(または，実行されるか)を表すメトリクス。

テスト完了基準

あるテストプロセスを公式に完了させるため，ステークホルダーが承認した一般・特定条件。

テスト完了基準の目的は，未完了部分のあるタスクが，完了とみなされるのを防ぐことにある。

テスト完了基準は，テスト完了の報告や，計画で利用する。

(「ソフトウェアテスト標準用語集」より)

テストシナリオ

「ある入力に対してこういった操作を行うと，このような結果が得られる」といった個々のテストケースをつなぎ合わせ，一連の手順の流れに沿って示したもの。

テスト手法

テスト技法やテスト方法など類似の用語があるが，本ガイドブックでは便宜的に，ホワイトボックスやブラックボックスを「テスト方法」，境界値テストや直交表などを「テスト技法」，両者を合わせて「テスト手法」と呼ぶ。

テスト生産性

テストに関わる工数を算出するために，テスト項目やテストケースなどテスト量に対する効率を表す指標。

テスト戦略

「ソフトウェアの重要な部分を特定したうえで，品質をどこまで確保し，そのためにテストにどれだけの資源を投入するか，また投入する資源をどう使うか」といった方針であり，ソフトウェア機能に対応した品質保証の重要度・優先度，テスト完了基準，テスト技法の取捨選択，テストプロセスの策定，テスト環境の決定およびテスト分担の決定などがある。

テスト戦略に基づいて，テスト対象をどのように検証するのか，誰が検証するのか，何を持って完了と判断するのかといったことをテスト計画として具体化する。

テストドキュメント

テストプロセスを通じて作成される，テストの計画，設計，実行に不可欠な物。

例えば，ドキュメント，スクリプト，入力，予想結果，セットアップとクリーンアップの処理手順，ファイル，データベース，環境，その他，テストで使用する付加的なソフトウェアやユーティリティなど。

テストウェア(testware)とも呼ばれる。

(「ソフトウェアテスト標準用語集」より)

テストファースト

設計または実装に先立って，テスト設計を行うことによって，これから設計またはプログラム実装するものが満たすべき条件を網羅的に洗い出すという考え方。設計者または開発者の誤解を防ぎ，設計の考慮漏れを予防し，かつ，効率的なテストを可能にしてより安価に欠陥を検出することが期待できる。

テストプロセス

基本的なテストプロセスは計画，仕様，実行(テスト実施，テスト結果検証，欠陥修正および再テスト実施)，記録，完了チェック，テスト終了作業から成る。 (「ソフトウェアテスト標準用語集」より)

テスト見積り

ソフトウェアをテストする総量を見積り，テスト生産性に基づき工数，工期，コストを見積もること。

テスト密度

ソフト規模(サイズ)あたりのテスト量を表すメトリクス。テスト量の目安を決める尺度として使用される。

テスト網羅率

テストしうるテスト量に対する実際にテストするテスト量の度合いを表すメトリクス。テストの手厚さを決める尺度として使用される。

テスト量

開発するソフトウェアの品質を担保するために実施するテスト作業の量。例えば，テストケース数，テスト項目数などが該当し，テスト仕様書，テスト手順書およびテストデータの量もテスト量に含まれる。

テストレベル

テスト作業の分類のこと。テストレベルには，プログラムの単体テストから，プログラムを結合した統合テスト，さらに業務要件を確認するシステムテストなどがある。

統合テスト

システムの内部コンポーネント(プログラムなど)同士のインターフェースを検証するために実施するテスト。

難バグ

テスト検出や解析に時間がかかる欠陥。

PSP

米国カーネギーメロン大学ソフトウェア工学研究所(SEI)が提唱する考え方。担当するプロジェクトの品質を管理し，決定した目標に向けて，計

画と予測の差を解消するように，プロジェクトの不具合を低減させる能力を向上させるプロセス。

非機能要件

「要件」を参照。

ビジネスアプリケーション

財務，会計，人事，給与などの事務処理に特化したソフトウェア。

品質指標

品質目標を設定するために使用するメトリクス。残存欠陥密度，工程ごとの検出欠陥密度，テスト完了基準，テスト網羅率などがある。

品質特性

ソフトウェアの品質を定義し，評価するために用いられるソフトウェアの属性の集合をいう。品質特性は，さらにいくつかのレベルの品質副特性(sub characteristics)に詳細化される(「ソフトウェア品質評価ガイドブック」より)。JIS X 0129-1：2003に示されている品質特性と品質副特性を**図1**に示す。

品質目標値

テスト戦略策定を通して設定される品質指標に基づく基準，到達目標。

品質要件

「要件」を参照。

ブラックボックステスト(ブラックボックス)

プログラムの内部構造と動作には一切関知せず，仕様書/設計書からテストデータを作成し，ユーザ視点で外的な動きを確認するテスト。代表的なテストデータを設計する技法として，同値分割，境界値分析，直交表などがある。

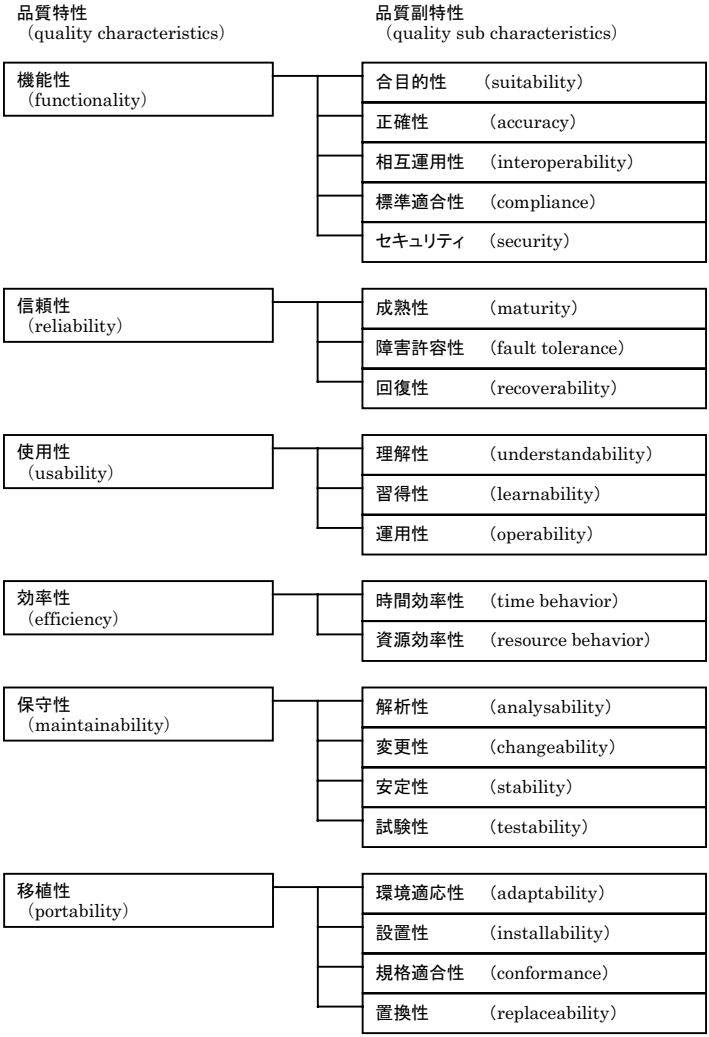

品質特性
（quality characteristics）
品質副特性
（quality sub characteristics）
機能性
（functionality）
合目的性 （suitability）
正確性 （accuracy）
相互運用性 （interoperability）
標準適合性 （compliance）
セキュリティ （security）
信頼性
（reliability）
成熟性 （maturity）
障害許容性 （fault tolerance）
回復性 （recoverability）
使用性
（usability）
理解性 （understandability）
習得性 （learnability）
運用性 （operability）
効率性
（efficiency）
時間効率性 （time behavior）
資源効率性 （resource behavior）
保守性
（maintainability）
解析性 （analysability）
変更性 （changeability）
安定性 （stability）
試験性 （testability）
移植性
（portability）
環境適応性 （adaptability）
設置性 （installability）
規格適合性 （conformance）
置換性 （replaceability）

図１　品質特性の構成

ベースライン

- 見積りの基準値。プロジェクトや組織における過去のプロジェクトデータなどに基づいて設定される。実際の見積りでは，この基準値にさまざまな変動要因を考慮して現実的な値を使う。
- 本ガイドブックでは，基準となる値などの上記の意味で使用している。構成管理などで使われる「正式にレビューされ合意された，一連の仕様または一連の作業成果物」という意味合いではない。

ホワイトボックステスト（ホワイトボックス）

プログラムの内部構造に着目して妥当性・整合性を調べていくテスト方法。プログラム内のすべての命令，すべてのルーチンを最低1回以上実行して検証することにより，プログラム記述者の意図どおりに動作していることを確認する。

マイルストーン

プロジェクトにおいて重要な意味をもつ時点やイベント。

（「PMBOK」の定義より）

巻き込み規模

ソフトウェアを変更した際に影響を受ける部分を表す規模のこと。

マスタテスト計画

単体テストから運用テストに至るまでのすべてのレベルのテストをカバーする計画書。単体テストから運用テストのそれぞれのテストに対応する要求と，要求が満たされていない場合のリスクに基づき，それぞれのテストで最も重要な欠陥をできるだけ迅速に効果的に検出できるようにテスト戦略を設定するというもの。

見積りモデル

データ収集，ベースラインの設定，変動要因の設定，見積りのためのアルゴリズムを数式化したもの。

モジュール化

ソフトウェアを，機能的に独立したいくつかの構成要素(モジュール)に分割して構成すること。モジュールの機能と呼び出し形式がわかっていれば，プログラムの他の部分と独立にその機能を実現することが可能であり，またモジュール単位での再利用も容易となる。

モニタリング

計画にかかわるプロジェクトのパフォーマンスデータを収集し，パフォーマンス測定を行い，パフォーマンス情報を報告し配布すること。

(「PMBOK」の定義より)

要　件

- システムに対する要件
  本ガイドブックでは，機能要件とそれ以外の要件(非機能要件)に分類する。
- 機能要件
  利用者の要求を満足するためにはソフトウェアが実現しなければならない利用者の業務および手順を表す。 (「JIS X 0135-1：1999(1)」)
- 非機能要件
  本ガイドブックで，機能要件以外の要件をまとめて呼ぶ用語。
- 品質要件
  JIS X 0129-1：2003で定義されたソフトウェア品質に関連した要件のすべて。 (「JIS X 0135-1：1999」より)
- 技術要件
  ソフトウェアの開発，維持管理，支援および実行のための技術・環境に関連した要件。技術要件の例としては，プログラム言語，試験ツール，オペレーティングシステム，データベース技術，利用者インターフェース技術などがある。 (「JIS X 0135-1：1999」より)

---

(1) JIS X 0135-1：1999(ISO/IEC 14143-1：1998)ソフトウェア測定－機能規模測定－第1部：概念の定義。

ライフサイクル

一般に順序どおり実施される各プロジェクトフェーズを集めたもの。プロジェクトフェーズの名称や数は，プロジェクトにかかわる組織におけるコントロールの必要性により決まる。ライフサイクルは方法論によって文書化する。（「PMBOK」の定義より）

リグレッションテスト

ソフトウェアを変更した際に，その影響を確認するテストのこと。例えば，大規模なソフトウェアでは，各部分のソフトウェアが複雑に関係しあいながら構成されていることが多い。したがって，ある箇所を改善しようとして加えた修正が，想定していない部分に影響してバグを呼び起こしていないことを確認するために行われる。

リスク

見積りにおいて最終的な規模やコストに対して起こりうる誤差。

「仮に発生すると，見積り（プロジェクト目標）にプラスまたはマイナスの影響を及ぼす不確実な事象または状態」（「PMBOK」の定義より）

参考文献

［1］ SEC見積り手法部会：『ITユーザとベンダのための定量的見積りの勧め』，オーム社(2005)

［2］ SEC見積り手法部会：『ソフトウェア開発見積りガイドブック』，オーム社(2006)

［3］ SEC見積り手法部会：『ソフトウェア改良開発見積りガイドブック』，オーム社(2007)

［4］ SEC開発プロセス共有化部会：『共通フレーム2007』，オーム社(2007)

［5］ Lee Copeland：『はじめて学ぶソフトウェアのテスト技法』，日経BP社(2005)

［6］ Rick D. Craig, Stefan P. Jaskiel：『体系的ソフトウェアテスト入門』，日経BP社(2004)

［7］ 吉澤正孝，秋山浩一，仙石太郎：『ソフトウェアテストHAYST法入門』，日科技連出版社(2007)

［8］ Tim Koomen, Martin Pol，富野　壽訳：『テストプロセス改善－CMM流実務モデル』，共立出版(2002)

索　引

た　行

な　行

は　行

ま　行

や　行

ら，わ　行

アルファベット

執 筆 者(敬称略，五十音順)

主　　査：	太田　忠雄	株式会社ジャステック
副 主 査：	合田　治彦	富士通株式会社
	粟野　憲一	富士通株式会社
	飯泉　純子	株式会社一(いち)
	育野　准治	日本ユニシス株式会社
	石谷　　靖	ソフトウェア・エンジニアリング・センター(株式会社三菱総合研究所)
	稲葉由貴子	株式会社NTTデータ
	井上　智史	TIS株式会社
	岩田　康夫	東京海上日動システムズ株式会社
	大島　正敬	日本電気株式会社
	大槻　　繁	株式会社一(いち)
	岡田　　章	一橋大学
	小野　直子	東京海上日動システムズ株式会社
	小浜　耕己	住生コンピューターサービス株式会社
	尾股　達也	社団法人情報サービス産業協会
	菊地奈穂美	ソフトウェア・エンジニアリング・センター(沖電気工業株式会社)
	楠本　真二	大阪大学
	芝元　俊久	株式会社日立システムアンドサービス
	須斉　智孝	KDDI株式会社
	高橋　　茂	ソフトウェア・エンジニアリング・センター(株式会社三菱総合研究所)
	高橋　宗雄	桐蔭横浜大学
	角田　千晴	社団法人日本情報システム・ユーザー協会
	中村　敏夫	ソフトウェア・エンジニアリング・センター(T&D情報システム株式会社)
	庭野　幸夫	株式会社ジャステック
	引地　信寛	KDDI株式会社
	細川　宣啓	日本アイ・ビー・エム株式会社
	堀　　明広	ソフトウェア技術者ネットワーク
	幕田　行雄	株式会社日立製作所
	向井　　清	住商情報システム株式会社
	森本　　聡	キャッツ株式会社
	山本修一郎	株式会社NTTデータ
	渡辺　政彦	キャッツ株式会社
経済産業省：		
	安田　　篤	商務情報政策局　情報処理振興課
	廣田　和也	商務情報政策局　情報処理振興課

編 者 紹 介

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター
2004年10月に独立行政法人 情報処理推進機構（IPA）内に設立されたソフトウェア・エンジニアリング・センター（SEC）は，エンタプライズ系ソフトウェアと組込みソフトウェアの開発力強化に取り組むとともに，その成果を実践・検証するための実践ソフトウェア開発プロジェクトを産学官の枠組みを越えて展開している。

［所在地］ 〒113-6591 東京都文京区本駒込2-28-8
文京グリーンコート センターオフィス
電話 03-5978-7543，FAX 03-5978-7517
http://sec.ipa.go.jp/index.php

SEC BOOKS
ソフトウェアテスト見積りガイドブック
～品質要件に応じた見積りとは～

平成 20 年 9 月 19 日 第 1 版第 1 刷発行

編 者 独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
発 行 者 竹 生 修 己
発 行 所 株式会社 オーム 社
郵便番号 101-8460
東京都千代田区神田錦町 3-1
電 話 03 (3233) 0641(代表)
URL http : //www.ohmsha.co.jp/

印刷・製本 報 光 社
ISBN 978-4-274-50198-2 Printed in Japan

独立行政法人　情報処理推進機構
ソフトウェア・エンジニアリング・センター　編　書籍

ソフトウェア開発見積りガイドブック
～IT ユーザとベンダにおける定量的見積りの実現～
A 5 判・240頁

ソフトウェア改良開発見積りガイドブック
～既存システムがある場合の開発～
A 5 判・172頁

共通フレーム2007
～経営者，業務部門が参画するシステム開発および取引のために～
B 5 変形版・320頁

経営者が参画する要求品質の確保
～超上流から攻める IT 化の勘どころ～
第 2 版
A 5 判・128頁・CD-ROM 付き
※【事例検索システム】URL : https : //sec.ipa.go.jp/enterprise/index.php

プロセス改善ナビゲーションガイド
～なぜなに編～
A 5 判・124頁

プロセス改善ナビゲーションガイド
～プロセス診断活用編～
A 5 判・160頁

プロセス改善ナビゲーションガイド
～ベストプラクティス編～
A 5 判・208頁

組込みシステムの安全性向上の勧め（機能安全編）
A 5 判・72頁

もっと詳しい情報をお届けできます.
◎書店に商品がない場合または御注文の場合も右記宛にご連絡ください.

ホームページ　http://www.ohmsha.co.jp/
TEL/FAX　TEL.03-3233-0643　FAX.03-3293-6224

オーム社/雑誌局

ISBN978-4-274-50198-2

C3055 ¥1429E

9784274501982

定価(本体1429円【税別】)

1923055014299

IPA® 独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター

SEC−TN08−003