

# **ソフトウェアテスト標準用語集 (日本語版)**

**Version 2.0.J02 (2011 年 04 月 19 日)**

**International Software Testing Qualifications Board  
用語集作業班**

**編集者: Erik van Veenendaal(オランダ)  
(翻訳: JSTQB®技術委員会)**

## 寄稿者

Rex Black (米国)  
Sigrid Eldh (スウェーデン)  
Isabel Evans (英国)  
Dorothy Graham (英国)  
Julian Harty (英国)  
David Hayman (英国)  
Juha Itkonen (フィンランド)  
Vipul Kocher (インド)  
Fernando Lamas de Oliveira (ポルトガル)  
Tilo Linz (ドイツ)  
Peter Morgan (英国)  
Thomas Muller (スイス)  
Avi Ofer (イスラエル)  
Dale Perry (米国)  
Horst Pohlmann (ドイツ)  
Meile Posthuma (オランダ)  
Erkki Poyhonen (フィンランド)  
Maaret Pyhajarvi (フィンランド)  
Andy Redwood (英国)  
Stuart Reid (英国)  
Piet de Roo (オランダ)  
Steve Sampson (英国)  
Shane Sanders (英国)  
Hans Schaefer (ノルウェー)  
Jurrien Seubers (オランダ)  
Dave Sherratt (英国)  
Mike Smith (英国)  
Andreas Spillner (ドイツ)  
Richard Taylor (英国)  
Geoff Thompson (英国)  
Stephanie Ulrich (ドイツ)  
Matti Vuori (フィンランド)  
Gearral Welvaart (オランダ)  
Pete Williams (英国)

## 改訂履歴

Ver1.3	
<p><i>New terms added:</i></p> <ul style="list-style-type: none"> <li>- action word driven testing</li> <li>- bug tracking tool</li> <li>- coverage measurement tool</li> <li>- modelling tool</li> <li>- monkey testing</li> <li>- scripted testing</li> <li>- specification-based technique</li> <li>- stress testing tool</li> <li>- structure-based technique</li> <li>- unit test framework</li> <li>- white box technique</li> </ul>	<p><i>Terms changed:</i></p> <ul style="list-style-type: none"> <li>- basic block</li> <li>- control flow graph</li> <li>- defect management tool</li> <li>- independence of testing</li> <li>- project risk</li> <li>- risk-based testing</li> <li>- test comparator</li> <li>- test process</li> </ul>
Ver2.0	
<p><i>New terms added:</i></p> <ul style="list-style-type: none"> <li>- attack</li> <li>- buffer</li> <li>- buffer overflow</li> <li>- bug taxonomy</li> <li>- classification tree</li> <li>- control flow analysis</li> <li>- continuous representation</li> <li>- cost of quality</li> <li>- defect based technique</li> <li>- defect based test design technique</li> <li>- defect taxonomy</li> <li>- error seeding tool</li> <li>- Failure Mode, Effect and Criticality Analysis (FMECA)</li> <li>- false-fail result</li> <li>- false-pass result</li> <li>- false-negative result</li> <li>- false-positive result</li> <li>- fault attack</li> <li>- fault seeding</li> <li>- fault seeding tool</li> <li>- hazard analysis</li> <li>- hyperlink</li> <li>- hyperlink tool</li> <li>- load profile</li> <li>- operational acceptance testing</li> <li>- operational profile</li> <li>- orthogonal array</li> <li>- orthogonal array testing</li> <li>- pairwise testing</li> <li>- performance profiling</li> <li>- pointer</li> <li>- procedure testing</li> <li>- process improvement</li> <li>- production acceptance testing</li> <li>- qualification</li> <li>- reliability growth model</li> <li>- retrospective meeting</li> <li>- risk level</li> <li>- risk type</li> <li>- root cause analysis</li> </ul>	<p><i>Terms changed:</i></p> <ul style="list-style-type: none"> <li>- bebugging</li> <li>- error seeding</li> <li>- Failure Mode and Effect Analysis (FMEA)</li> <li>- Fault Tree Analysis (FTA)</li> <li>- modified multiple condition testing</li> <li>- process cycle test</li> <li>- root cause</li> <li>- specification-based technique</li> <li>- stress testing</li> <li>- test charter</li> </ul>

<ul style="list-style-type: none"><li>- safety critical system</li><li>- software attack</li><li>- Software Failure Mode and Effect Analysis (SFMEA)</li><li>- Software Failure Mode Effect and Criticality Analysis (SFMECA)</li><li>- Software Fault Tree Analysis (SFTA)</li><li>- software life cycle</li><li>- staged representation</li><li>- system of systems</li><li>- test design</li><li>- test estimation</li><li>- test implementation</li><li>- Test Maturity Model Integration (TMMi)</li><li>- test progress report</li><li>- test rig</li><li>- test schedule</li><li>- test session</li><li>- wild pointer</li></ul>	
--	--

## 改訂履歴 日本語版

バージョン	日付	摘要
Version 2.0.J01	2009-11-20	シラバスに合わせて修正 バージョン表記方法の変更
Version 2.0.J02	2011-04-19	※J02 では表記上の修正のみを行い、用語の変更、削除、新規追加等を行っていない ✓ 日本語としての揺れを一部修正 ✓ 誤字脱字を一部訂正 ✓ フォントなど一部表記を訂正 ✓ セキュリティ設定の変更

## 目次

まえがき.....	7
1. 序文.....	8
2. 対象範囲.....	8
3. 配置.....	8
4. 標準引用文献.....	9
5. 商標.....	9
6. 定義(あいうえお順).....	10
あ.....	10
い.....	10
う.....	12
え.....	12
お.....	13
か.....	13
き.....	14
く.....	16
け.....	16
こ.....	17
さ.....	19
し.....	19
す.....	22
せ.....	23
そ.....	24
た.....	25
ち.....	25
て.....	26
と.....	32
に.....	33
の.....	34
は.....	34
ひ.....	35
ふ.....	36
へ.....	38
ほ.....	39
ま.....	40
み.....	40
め.....	40
も.....	40
ゆ.....	41
よ.....	42
ら.....	42
り.....	42
れ.....	43
ろ.....	43
わ.....	44
付録 A(参考資料).....	45
付録 B (用語集へのコメント方法).....	46
索引 (英語).....	47

## まえがき

本用語集の編纂では、できるだけ広い分野で受け入れられる国際的テスト標準を作成するため、産業界、ビジネス分野、政府の諸団体・組織から可能な限り広範囲の見解やコメントを求めてきた。この類のドキュメントでは、全員が合意することはほとんどあり得ない。本ドキュメントの用語は、オーストラリア、ベルギー、フィンランド、ドイツ、インド、イスラエル、オランダ、ノルウェー、ポルトガル、スウェーデン、スイス、イギリス、アメリカのソフトウェアテスト団体から寄稿されたものである。

多数のソフトウェアテスト担当者は、BS7925-1を1998年の初版以降使用してきた。同標準は、Information Systems Examination Board (ISEB) 資格のFoundation (基礎) レベル、および、Practitioner (実務者) レベルでの重要なリファレンスとして使われてきた。同標準は、当初、コンポーネントテストを中心に編纂したものであった。しかし、同標準の出版以降、同標準がソフトウェアテストの広い範囲をカバーし、内容を改善するため、多数のコメントや提案が寄せられた。本用語集最新版では、提案を多数取り込んだ。本標準は、国際ソフトウェアテスト資格認定委員会: International Software Testing Qualifications Board (ISTQB®) ソフトウェアテスト資格認定要綱の参考資料として使用する。

Translation Copyright © 2005-2011, Japan Software Testing Qualifications Board (JSTQB®), all rights reserved.

日本語翻訳版の著作権は JSTQB® が有するものです。本書の全部、または一部を無断で複製し利用することは、著作権法の例外を除き、禁じられています。

本用語集は、Standard glossary of terms used in Software Testing Version 2.0 (dd. December, 2nd 2007) を基に翻訳をしています。

本用語集は、予告なしに変更する可能性があります。

## 1. 序文

テストの世界には、ステートメントカバレッジと判定条件カバレッジ、テストスイートとテスト仕様とテスト計画など、よく似た用語があり、これが、社会の様々な分野間で、インターフェースになっている。よく似た用語の違いを明確にできないと、産業界、ビジネス界、政府、技能や学術団体の内外部で、時間や労力の無駄となる。しかも、類似用語は、業界用語、技術用語として、別の意味を持つことも多い。

## 2. 対象範囲

本ドキュメントは、ソフトウェアテストや、関連分野でのコミュニケーションを容易にするため、概念、用語、定義を記述したものである。

## 3. 配置

本用語集は、アルファベット順に並べた単一の定義部で構成する。用語の中には、他の同義語より優先させたものもある。その場合、優先した用語に定義を記述し、他の同義語は、優先度の高い用語を参照するようにした。たとえば、structural testing は、white box testing を参照する。同義語には、「See (参照のこと)」を表示した。

「See also (～も参照)」というクロスリファレンスも使用した。これにより、目的の用語を高速で検索可能となる。「See also (～も参照)」は、広義の用語と狭義の用語や、意味の重なりが分かるような構成になっている。

## 4. 標準引用文献

出版当時、以下の版の標準は有効であった。標準は改訂されるし、標準を準拠して物事を決める団体は、以下に挙げた各標準の最新版を参照してほしい。IEC とISOのメンバーは、現在有効な国際標準の登録リストを持っている。

BS 7925-2:1998. Software Component Testing.

DO-178B:1992. Software Considerations in Airborne Systems and Equipment Certification, Requirements and Technical Concepts for Aviation (RTCA SC167).

IEEE 610.12:1990. Standard Glossary of Software Engineering Terminology.

IEEE 829:1998. Standard for Software Test Documentation.

IEEE 1008:1993. Standard for Software Unit Testing.

IEEE 1012:2004. Standard for Verification and Validation Plans

IEEE 1028:1997. Standard for Software Reviews and Audits.

IEEE 1044:1993. Standard Classification for Software Anomalies.

IEEE 1219:1998. Software Maintenance.

ISO/IEC 2382-1:1993. Data processing-Vocabulary-Part 1: fundamental terms.

ISO 9000:2005. Quality Management Systems-Fundamentals and Vocabulary.

ISO/IEC 9126-1:2001. Software Engineering-Software product Quality- Part 1: Quality characteristic and sub-characteristics.

ISO/IEC 12207:1995. Information Technology- Software Life Cycle Processes.

ISO/IEC 14598-1:1996. Information Technology- Software Product Evaluation-Part 1: General Overview.

## 5. 商標

本書では以下の商標を使用している。

CMM and CMMI are registered trademarks of Carnegie Mellon University

TMap, TPA and TPI are registered trademarks of Sogeti Nederland BV

TMM is a registered servicemark of Illinois Institute of Technology

TMMi is a registered trademark of the TMMi Foundation

## 6. 定義(あいうえお順)

### あ

**アイソレーションテスト(isolation testing)**: 周辺のコンポーネントとは独立に、個々のコンポーネントをテストすること。必要に応じて、スタブやドライバーで、周辺コンポーネントをシミュレートする。

**アイテム送付レポート(item transmittal report)**: release note を参照のこと。

**アクセシビリティテスト(accessibility testing)**: 身体的な制約を持つ人を含む全てのユーザが、どの程度容易にコンポーネントやシステムを利用できるか判定するテスト。[Genard]

**アクションワード駆動テスト (action word driven testing)**: keyword driven testing を参照のこと。

**アークテスト(arc testing)**: branch testing を参照のこと。

**アジャイルテスト(agile testing)**: エクストリームプログラミング(XP)のようなアジャイル方法論を用いたり、開発をテストの一部とみなしたり、テストファースト設計パラダイムを重視するプロジェクトで実施するテスト。Test driven development を参照のこと。

**アドホックテスト(ad hoc testing)**: 非公式に実施するテスト。公式なテストの準備をせず、実績のあるテスト設計技法を用いず、テスト結果も予測せず、毎回、テスト方法が変わる。

**アドホックレビュー(ad hoc review)**: informal review を参照のこと。

**アナライザ(analyzer)**: static analyzer を参照のこと。

**誤った合格結果(FALSE-pass result)**: テスト対象に存在する欠陥を認識できなかったテスト結果

**誤った失敗結果(FALSE-fail result)**: テスト対象には欠陥が存在しないにもかかわらず、欠陥として報告したテスト結果。

**誤り(mistake)**: error を参照のこと。

**アルゴリズムテスト(algorithm test [TMap])**: branch testing を参照のこと。

**アルファテスト(alpha testing)**: 顧客、ユーザ、開発組織とは独立したテストチームが、シミュレーションや実際のオペレーションにより実行するテスト。一般販売用ではなく、特定顧客用のソフトウェアでは、内部受入れテストとして実施することが多い。

**安全性(safety)**: 指定された使用状況の中で、人、ビジネス、ソフトウェア、資産、または、環境に対する損害リスクを受容レベルに維持できるソフトウェア製品の能力。[ISO9126]

**安全性テスト(safety testing)**: ソフトウェア製品の安全性を判定するテスト。

**安定性(stability)**: ソフトウェアの修正による予期しない影響を回避するソフトウェア製品の能力。[ISO9126] maintainability も参照のこと。

### い

**移行テスト(migration testing)**: conversion testing を参照のこと。

**移植性 (portability) :** ソフトウェア製品を、あるハードウェア環境やソフトウェア環境から他へ容易に移植できる度合。 [ISO9126]

**移植性テスト (portability testing) :** ソフトウェア製品の移植性を判定するテスト手順

**一貫性 (consistency) :** 均一性、規格統一性、および、コンポーネントやシステムのドキュメントや構成物の間の無矛盾性の度合。 [IEEE610]

**インクリメンタル開発モデル (incremental development model) :** 開発ライフサイクルの1つ。プロジェクトで開発する要件全体を部分的な機能に分割し、サブプロジェクトにおいて、部分機能を積み上げながら連続して開発する方式。各要件に優先順位を付け、優先度の高い順に適切な大きさの機能をリリースする。このライフサイクルモデルには、各サブプロジェクトが、設計、コーディング、テストというミニV字モデルで進むものもある。

**インクリメンタルテスト (incremental testing) :** すべてのコンポーネント、システムを統合・テストするまで、コンポーネントやシステムを一度こひとつ、または、いくつかを統合・テストする方式。

**インシデント (incident) :** 発生した事象の中で、調査が必要なもの。 [After IEEE1008]

**インシデント管理ツール (incident management tool) :** インシデントの記録や、状態自動を支援するツール。ワークフロー志向の機能を持っていることが多く、インシデントの場所特定、修正、再テストをトレース・制御したり、レポートを出力したりできる。 defect management tool も参照のこと。

**インシデントマネジメント (incident management) :** インシデントを認識、調査、対策、解明するプロセス。インシデントの記録、分類、影響度の分析が必要。 [After IEEE1044]

**インシデントレポート (incident report) :** 発生したあらゆるインシデント(テストの最中に調査を必要とする事象など)を報告するドキュメント。 [After IEEE829]

**インシデントログ (incident logging) :** インシデントログ:テスト中などに発生したインシデントの詳細に対する記録。

**インストルメンタ (instrumenter) :** 計装を実行するソフトウェアツール。

**インストルメンテーション (instrumentation) :** 実行中にプログラムの動作情報(たとえば、コードカバレッジの測定)を集めるため、プログラムにコードを追加すること。

**インストーラビリティ (installability) :** 設置性。指定した環境へインストールできるソフトウェア製品の能力。 [ISO9126] portability も参照のこと。

**インストーラビリティテスト (installability testing) :** ソフトウェア製品のインストーラビリティをテストする手順。 portability testing も参照のこと。

**インストールウィザード (installation wizard) :** インストールを支援するソフトウェア。いろいろな形態の媒体で提供される。インストールを実行し、インストール実施結果情報を出力し、オプション機能のガイダンスを表示するものが多い。

**インストールガイド (installation guide) :** インストール手順を最適な媒体で提供するもの。マニュアルガイド、段階的な処理手順、インストールウィザード、その他類々の手順書形式などの形式をとる。

**インスペクション (inspection) :** ピアレビューの一種。目視検査により、欠陥を摘出する方法。これにより、たとえば、開発標準の違反や、上位レベルドキュメントへの準拠違反が見つかる。最も公式なレビュー技術なので、必ず、文書化された実施基準に従って進める。 [After IEEE610, IEEE1028] peer review も参照のこと。

**インスペクションリーダー (inspection leader) :** moderator を参照のこと。

**インスペクタ (inspector)**: reviewer を参照のこと。

**インターフェーステスト (interface testing)**: 統合テストの一種。コンポーネントやシステムのインターフェーステストを実施する。

**インテークテスト (intake test)**: スモークテストの特別な形態。コンポーネントやシステムが、詳細テストや複雑なテストを実施できるか判定するためのもので、テスト実行フェーズの開始時に行うことが多い。smoke test も参照のこと。

**インバリッドテスト (invalid testing)**: コンポーネントやシステムが拒否する入力値を使うテスト。error tolerance も参照のこと。

## う

**ウォークスルー (walkthrough)**: 情報を集めて内容の共通理解を確立するための、ドキュメントの著者による段階的なプレゼンテーション。[Freedman and Weinberg, IEEE1028] peer review も参照のこと。

**受け入れ (acceptance)**: acceptance testing 参照のこと。

**受け入れ基準 (acceptance criteria)**: ユーザ、顧客、その他の認可団体が、コンポーネントやシステムを承認する場合、満たさねばならない 終了基準。[IEEE610]

**受け入れテスト (acceptance testing)**: 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したりできる。[After IEEE610]

**運用受け入れテスト (operational acceptance testing)**: 受け入れテストフェーズでの運用テスト。通常はオペレータやアドミニストレータ運用面に焦点を当てて現実の操作環境をシミュレーションする。たとえば、リカバリー、リソースの振る舞い、インストール性、技術的コンプライアンスなど。operational testing も参照のこと。

**運用テスト (operational testing)**: 操作環境の中で、コンポーネントやシステムを評価するテスト。[IEEE610]

**運用プロファイル (operational profile)**: 着目すべきシステムやコンポーネントを使って行われるタスクセットの描写。各タスクは、コンポーネント、システムと利用者のやりとりやそのときに起きる可能性のある事象をベースにする事が多い。タスクは物理的というより論理的であり、複数のマシン上、あるひとつのタイミングのものとして実行される。

**運用プロファイルテスト (operational profile testing)**: システム操作(短時間のタスク)モデルと、典型的操作パターンの使用確率を使う統計的なテスト。[Musa]

## え

**影響度分析 (impact analysis)**: ある要件を変更する前に、開発手引書、テスト手引書、コンポーネントの各階層が、変更によりどんな影響を受けるか評価すること。

**N スイッチカバレッジ (N-switch coverage)**: テストスイートが実施した N+1 推移のシーケンスのパーセンテージ。[Chow]

**N スイッチテスト (N-switch testing)**: 状態推移テストの一形式。N+1 推移の全ての有効なシーケンスを実行するテストケースを設計する。[Chow] state transition testing も参照のこと。

**エミュレータ (emulator)**: ある特定のシステムと同じ入力を受け入れ、同じ出力を作出す装置、コンピュータプログラム、またはシステム。[IEEE610] simulator も参照のこと。

**エラー (error)**: 間違った結果を生み出す人間の行為。[After IEEE610]

**エラーシーディング (error seeding) :** fault seeding を参照のこと。[IEEE610]

**エラーシーディングツール (error seeding tool) :** fault seeding tool を参照のこと。

**エラー推測 (error guessing) :** テスト設計技法の1つ。テスト担当者の経験を駆使し、エラーが起きた場合にどんな欠陥が被験体のコンポーネントやシステムの中に存在するかを予想して、その欠陥を検出するテストケースを設計すること。

**エラー耐性 (error tolerance) :** 誤ったデータを入力しても、通常運用を続けられるコンポーネントやシステムの能力。[After IEEE610]

**LCSAJ (LCSAJ) :** Linear Code Sequence And Jump (リニアコードシーケンスアンドジャンプ)の略。慣例として、ソースコードリスト中の行番号で識別する以下の3アイテムから成る。①実行命令文の線形連鎖の開始行、②線形連鎖の終了行、③線形連鎖の終了時に制御フローが移動する行。

**LCSAJ カバレッジ (LCSAJ coverage) :** テストスイートが実施したコンポーネントのLCSAJ (リニアコードシーケンスアンドジャンプ)のパーセンテージ。100%のLCSAJは、100%のデジジョンカバレッジを意味する。

**LCSAJ テスト (LCSAJ testing) :** LCSAJ のテストケースを実行するために設計されたホワイトボックステスト設計技法。

## お

**オラクル (oracle) :** test oracle を参照のこと。

## か

**回帰テスト (regression testing) :** 変更により、ソフトウェアの未変更部分に欠陥が新たに入り込んだり、発現したりしないことを確認するため、変更実施後、既にテスト済みのプログラムに対して実行するテスト。ソフトウェアや、実行環境が変わるたびに実施する。

**開始基準 (entry criteria) :** 定義したタスク(たとえば、テストフェーズ)を開始できるかどうかを判断する場合の基準となる一般・特定の条件。この目的は、開始基準を満足させるための労力に比べ、はるかに多くの(無駄な)労力を投入しないとタスクが収まらない状況を防ぐことである。[Gilb and Graham]

**開始点 (entry point) :** コンポーネント中の最初の実行命令文。

**解析性 (analyzability) :** ソフトウェア中の欠陥や不良の原因を診断したり、改造箇所を特定したりできるソフトウェアの能力。[ISO9126] maintainability を参照のこと。

**開発テスト (development testing) :** コンポーネントやシステムの開発中に実施する公式・非公式のテスト。開発担当者の製品開発環境で実行することが多い。[After IEEE610]

**回復性 (recoverability) :** 故障発生時に、パフォーマンスを指定レベルに回復し、影響を受けたデータを直接復旧するソフトウェア製品の能力。[ISO9126] reliability も参照のこと。

**回復性テスト (recoverability testing) :** ソフトウェア製品の回復性を判定するテスト手順。reliability testing も参照のこと。

**回復テスト (recovery testing) :** recoverability testing を参照のこと。

**拡張性 (scalability) :** 増加する負荷に応じてソフトウェア製品をアップグレードできる能力。[After Gerrard]

**拡張性テスト (scalability testing) :** ソフトウェア製品の拡張性を判定するテスト。

**確認テスト (confirmation testing) :** re-testing を参照のこと。

**カスタムソフトウェア (custom software) :** bespoke software を参照のこと。

**カバレッジ (coverage) :** 指定の網羅条件をテストスイートが実行した割合。パーセンテージで表す。

**カバレッジアイテム (coverage item) :** テストカバレッジの基礎となる実体や属性。たとえば、同値分割やコード命令。

**カバレッジ計測ツール (coverage measurement tool) :** coverage tool を参照のこと。

**カバレッジツール (coverage tool) :** テストスイートが実行した命令語や分岐等の構造要素の割合を客観的に測定するツール。

**カバレッジ分析 (coverage analysis) :** 指定した網羅条件のうち、どれだけ実行したかをテスト実行中に計測すること。追加テストが必要か、必要ならどのテストを追加するかを判定用に事前に定めた判定基準を照らして実施する。

**可用性 (availability) :** 必要な時にコンポーネントやシステムが稼働し、利用可能な割合。パーセンテージで表すことが多い。 [IEEE610]

**環境適応性 (adaptability) :** 環境順応以外の変更や方法を使うことなく、別環境でも動作できるソフトウェアの能力。 [ISO9126] portability を参照のこと。

**頑健性 (堅牢性) (robustness) :** 不正な入力や過負荷の環境の中でも、コンポーネントまたはシステムが正しく機能できる割合。 [IEEE610] error-tolerance、fault-tolerance も参照のこと。

**監査 (audit) :** 下記を規定したドキュメント、標準、ガイドライン、仕様、客観的基準に準拠した手順を遵守していることを確認するため、ソフトウェアや開発プロセスを独立に評価すること。

- (1) 開発する製品の形式や内容
- (2) 製品の開発プロセス
- (3) 標準やガイドライン遵守の測定方法 [IEEE1028]

**監査証跡 (audit trail) :** プロセスの出力をスタート地点とし、プロセスを遡って、入力(たとえば、データ)までたどるプロセスのパス。これにより、欠陥分析が容易になり、プロセス監査を実施できる。 [After TMap]

**完全テスト (complete testing) :** exhaustive testing を参照のこと。

## き

**偽陰性結果 (FALSE-negative result) :** false-pass result を参照のこと。

**机上チェック (desk checking) :** 手動の実行シミュレーションによるソフトウェアまたは仕様のテスト。 static analysis も参照のこと。

**擬似乱数 (pseudo-random) :** 見たらめに見えるが、事前に決めた一連の規則に従って生成した数値群。

**既製ソフトウェア (off-the-shelf software) :** 一般の市場用(不特定多数のユーザ用)に開発したソフトウェア製品、全く同じものを多数の顧客に提供する。

**期待結果 (expected outcome) :** expected result を参照のこと。

**期待結果 (expected result) :** 指定の条件下で、仕様や他の情報から期待できるコンポーネントやシステムの動作。

**規程テスト(regulation testing)**： compliance testing を参照のこと。

**機能性(functionality)**： ソフトウェアが指定の条件下で移動する場合、明示的・暗示的な要求を満たす機能を提供できるソフトウェア製品の能力。[ISO9126]

**機能性テスト(functionality testing)**： ソフトウェア製品の機能性を判定するためのテストの手順

**機能テスト(functional testing)**： コンポーネントやシステムの機能仕様を分析して実施するテスト。black box testing も参照のこと。

**機能テスト設計技法(functional test design technique)**： コンポーネントやシステムの内部構造を参照せず、機能の仕様を分析してテストケースを設計・選択すること。black box test design technique も参照のこと。

**機能統合(functional integration)**： システム統合法の1つ。早い時期に、基本機能を動作させるために、コンポーネントやシステムを結合すること。integration testing も参照のこと。

**機能要件(functional requirement)**： コンポーネントやシステムが実行する機能を定義した要件。[IEEE610]

**基本比較テスト(elementary comparison testing)**： ブラックボックステスト設計技法の1つ。条件半定カバレッジの概念に従った入力の組み合わせを確認するテストケースを設計する。[TMap]

**キャスト(CAST)**： Computer Aided Software Testing(コンピュータ支援ソフトウェアテスト)の頭字語。test automation も参照のこと。

**キャプチャ/プレイバックツール(capture/playback tool)**： テスト実行ツールの1つ。手動テスト中の入力を記録させ、自動テストのスクリプトを作成して、後に実行(再現)させるもの。自動回帰テストで利用することが多い。

**キャプチャ/リプレイツール(capture/replay tool)**： capture/playback tool を参照のこと。

**偽陽性結果(FALSE-positive result)**： false-fail result を参照のこと。

**境界値(boundary value)**： 同値分画した領域の端、あるいはお端のどちらか側で最小の増加的距離にある入力値または出力値。たとえばある範囲の最小値または最大値。

**境界値カバレッジ(boundary value coverage)**： テストスイートで網羅した境界値のパーセンテージ。

**境界値テスト(boundary value testing)**： boundary value analysis を参照のこと。

**境界値分析(boundary value analysis)**： 境界値を基にしてテストケースが設計される、ブラックボックステスト設計技法。boundary value も参照のこと。

**共存性(co-existence)**： 同じ環境下で同じリソースを共有する他の独立したソフトウェアと、共存できるソフトウェア製品の能力。[ISO9126] portability も参照のこと。

**記録再生ツール(record/playback tool)**： capture/playback tool を参照のこと。

**記録者(recorder)**： scribe を参照のこと。

**キーワード駆動テスト(keyword driven testing)**： テストスクリプト記述技術の1つ。入力値と予想出力値だけでなく、テスト対象アプリケーションに関するキーワードを含んだデータファイルを使う。キーワードは、テストの制御スクリプトが呼び出す特別な補助スクリプトが解釈する。data driven testing も参照のこと。

**キーパフォーマンスインジケータ(key performance indicator)**： performance indicator を参照のこと。

## く

**具体的テストケース (concrete test case)**: low level test case を参照のこと。

**クラシフィケーションツリー (classification tree)**: 階層的な順番で同値分割を示したツリーであり、クラシフィケーションツリー方式でテストケースを作成するときに使う。classification tree method も参照のこと。

**クラシフィケーションツリー法 (classification tree method)**: ブラックボックステスト設計技法の1つ。クラシフィケーションツリーを使ってテストケースを記述し、入力領域や出力領域の代表値を組み合わせて実行することを目的とする。[Grochtmann]

**ガラスボックステスト (glass box testing)**: white box testing を参照のこと。

## け

**経験ベースの技法 (experienced-based technique)**: experienced-based test design technique を参照のこと。

**経験ベースのテスト設計技法 (experienced-based test design technique)**: テスト担当の経験・知識・直感をベースにテストケースを導き出したり選択したりする方法。

**ケース (CASE)**: Computer Aided Software Engineering (コンピュータ支援ソフトウェア開発)の頭字語。

**結果 (outcome)**: result を参照のこと。

**結果 (result)**: テスト実行後の成果。画面への出力、データの変化、レポート、外部へ送信するメッセージを含む。actual result、expected result も参照のこと。

**欠陥 (defect)**: 要求された機能をコンポーネントまたはシステムに果たせなくする、コンポーネントまたはシステムの中の不備。たとえば、不正な命令またはデータ定義、実行中に欠陥に遭遇した場合、コンポーネントまたはシステムの故障を引き起こす。

**欠陥検出率 (Defect Detection Percentage (DDP))**: テストフェーズで検出した欠陥の数をテストフェーズ、および、以降の別のフェーズで見つけた欠陥の総数で除算した値。

**欠陥追跡ツール (defect tracking tool)**: defect management tool を参照のこと。

**欠陥ベースの技法 (defect based technique)**: defect based test design technique を参照のこと。

**欠陥ベースのテスト設計技法 (defect based test design technique)**: 一つ以上の欠陥カテゴリからテストケースのターゲットを導き出したり、選択したりする手続き。テストケースは特定の欠陥カテゴリから開発していく。defect taxonomy も参照のこと。

**欠陥分類法 (defect taxonomy)**: 再現性良く欠陥を分類するために有用に設計するときの(階層的)カテゴリの体系。

**欠陥マネジメント (defect management)**: 認識、調査、行動、および、欠陥の処置の手順、欠陥の記録、分類、および、影響の特定を含む。[After IEEE1044]

**欠陥マネジメントツール (defect management tool)**: 欠陥および変更の記録と状態自動を容易にするツール。多くの場合、欠陥の割り振り、訂正、再テストを追跡、制御するために、ワークフロー指向の機能を搭載している他に、レポート機能も持っている。incident management tool も参照のこと。

**欠陥密度 (defect density)**: コンポーネントまたはシステムの中で特定された欠陥の数をコンポーネントまたはシステムの

大きさを割った値。(大きさを表す標準的な言葉で表現された、たとえば、コード行数、クラス数、またはファンクションポイント数)

**欠陥レポート(defect report):** コンポーネントまたはシステムに要求された機能を果たせなくする、コンポーネントまたはシステムの中の何処かの不備を報告する文書。[After IEEE829]

**原因結果グラフ(cause-effect graph):** 入力、刺激(原因)、関連する出力(結果)を図式表現したもの。テストケースの設計で使える。

**原因結果グラフ法(cause-effect graphing):** テストケースが原因結果グラフから設計されるブラックボックステスト設計技法。[BS7925/2]

**原因結果デシジョンテーブル(cause-effect decision table):** decision table を参照のこと。

**原因結果分析(cause-effect analysis):** cause-effect graphing を参照のこと。

**検証(verification):** 検査などで特定の要件が満たされていることを客観的な証拠で確認すること。[ISO9000]

## こ

**高位レベルテストケース(high level test case):** 具体的な(実行レベルの)入力値や予測結果を使わないテストケース。論理演算子は使用するが、値のインスタンスは未定義であったり、使えなかったりといった状態がある。low level test case も参照のこと。

**合格/失敗基準(pass/fail criteria):** テストアイテムや機能が、テストに合格したか失敗したかを判定するための判定規則。[IEEE829]

**攻撃(attack):** 直接的に焦点を定めて、品質を評価する試み。特定の故障が発生するような強制を試みることによって、テスト対象の品質、特に信頼性を評価する。

**公式レビュー(formal review):** 文書化した手順や要求事項をベースにするレビュー。たとえば、インスペクション。

**構成(configuration):** コンポーネントやシステムの構成。数量、特性、相互連結により定義する。

**構成アイテム(configuration item):** ハードウェア、ソフトウェア、または、両方の集合体。構成管理の対象であり、構成管理プロセスでは、1つの実体として扱われる。[IEEE610]

**構成監査(configuration auditing):** 構成アイテムのライブラリの内容をチェックする機能。たとえば、標準法令遵守をチェックするなど。[IEEE610]

**構成管理(configuration management):** 技術的、管理的な指揮・監視により、構成を制御すること。構成アイテムの特性を機能的、物理的に定義・文書化すること、特性に対する変更を制御すること、変更のプロセスや変更実施状況を記録・報告すること、定義した要求に遵守していることの実証を目的とする。[IEEE610]

**構成管理ツール(Configuration management tool):** 構成アイテムの識別やコントロール(変更やバージョンの状態や構成アイテムをまとめたベースラインの公開)を支援するツール。

**構成コントロール(configuration control):** 構成マネジメントの1つの要素。構成アイテムを公式に定義後、構成アイテムに対する変更を、評価、調整、承認・否認、実施すること。[IEEE610]

**構成管理委員会(configuration control board (CCB)):** 構成アイテムに対する変更提案を評価、承認・否認することに責任を持つグループ。承認した変更の実施にも責任を負う。[IEEE610]

**構成識別(configuration identification):** 構成管理の1つの要素。システムの構成アイテムを選択し、構成アイテムの機能

的、物理的特性を技術文書に記録する。[IEEE610]

**構成テスト(configuration testing)**： portability testing を参照のこと。

**構造化ウォークスルー(structured walkthrough)**： walkthrough 参照のこと。

**構造カバレッジ(structural coverage)**： コンポーネントまたはシステムの内部構造を基にした網羅の尺度。

**構造テスト(structural testing)**： white box testing 参照のこと。

**構造テスト設計技法(structural test design technique)**： white box test design technique 参照のこと。

**構造ベース技法(structure-based technique)**： white box test design technique 参照のこと。

**構造ベーステスト(structure based testing)**： white-box testing を参照のこと。

**合目的性(suitability)**： 指定されたタスクやユーザの目的を実行するため、機能の適正な集合を提供するソフトウェア製品の能力。[ISO9126] functionality も参照のこと。

**効率性(efficiency)**： 決められた条件下で使われるリソースの量に関連する、適正なパフォーマンスを提供するためのソフトウェア製品の能力。[ISO9126]

**効率性テスト(efficiency testing)**： ソフトウェア製品の効率を測定するテスト。

**互換性テスト(compatibility testing)**： interoperability testing を参照のこと。

**故障(failure)**： コンポーネントやシステムが、期待した機能、サービス、結果を提供できないこと。[After Fenton]

**故障モード(failure mode)**： 物理的または機能的な故障の兆候・明示。たとえば、故障モードのシステムは、遅い運用、間違った出力、または実行の完全な打ち切りなどで特徴づけられる。[IEEE610]

**故障モード影響解析(Failure Mode and Effect Analysis (FMEA))**： リスクを特定し、起こり得る故障モードを特定して、発生の防止策を分析する体系的なアプローチ。Failure Mode, Effect and Criticality Analysis (FMECA)も参照のこと。

**故障モード影響・致命度解析(Failure Mode, EFFECT and Criticality Analysis (FMECA))**： 基本的なFMEAに、故障モードの結果の重要度が発生するかもしれないことを加味したチャートを使って致命度分析を付加し拡張したもの。Failure Mode and Effect Analysis (FMEA)も参照のこと。

**故障率(failure rate)**： 測定単位に発生したあるカテゴリの故障数の率。たとえば、単位時間あたりの故障数、トランザクション数あたりの故障数、コンピュータの運用回数あたりの故障数。[IEEE610]

**COTS(COTS)**： Commercial Off-The-Shelf software (市販ソフトウェア)の頭字語。off-the-shelf software を参照のこと。

**コード(code)**： プログラミング言語、またはコンパイラやその他の変換装置によってアウトプットされたコンピュータ命令とデータ定義を表現。[IEEE 610]

**コードアナライザ(code analyzer)**： static code analyzer を参照のこと。

**コードカバレッジ(code coverage)**： テストスイートが、ソフトウェアのどの部分を実行(カバー)し、どの部分が未実行かを判定する分析方法。たとえば、ステートメントカバレッジ、ブランチカバレッジ、条件カバレッジ。

**コードベーステスト(code-based testing)**： white box testing を参照のこと。

**コンパイラ(compiler)**： 高度な命令語で表現されたプログラムを、等価の機械語に翻訳するソフトウェアツール。[IEEE610]

**コンポーネント(component)**: 独立してテストできるソフトウェアの最小単位。

**コンポーネント仕様(component specification)**: 指定の条件下で、指定の入力値に対する出力値として記述したコンポーネントの機能、および、コンポーネントが満足すべき非機能的な動作(たとえば、資源利用)。

**コンポーネントテスト(component testing)**: 個々のソフトウェアコンポーネントのテスト。[After IEEE610]

**コンポーネント統合テスト(component integration testing)**: 統合したコンポーネント間のインターフェースや相互作用の欠陥を検出するためのテスト。

**根本原因(root cause)**: もしその原因が取り除かれることで、同種の欠陥が減少する、もしくは発生しなくなるような欠陥の根本原因。[CMMI]

**根本原因分析(root cause analysis)**: 欠陥の根本原因の特定を目的とした分析技法。根本原因に是正を行うことで、欠陥再発を最小化する事が期待できる。

## さ

**再開基準(resumption criteria)**: テストを中断した後、再開したときに繰り返さなければならないテストの活動。[After IEEE829]

**サイクロマティック数(cyclomatic number)**: cyclomatic complexity を参照のこと。

**サイクロマティック複雑度(cyclomatic complexity)**: プログラムの中を通る独立したパスの数。  $L-N+2P$  で定義する。

-L= グラフ中のエッジ/リンクの数

-N= グラフ中のノードの数

-P= グラフの繋がっていない部分の数(たとえば、呼ばれるグラフとサブルーチンの数)[After McCabe]

**再テスト(re-testing)**: 修正を行った結果が正しいことを証明するために、前回不合格に終わったテストケースを再実行するテスト。

**サイト受け入れテスト(site acceptance testing)**: ユーザや顧客が自分のサイトで実施する受け入れテスト。コンポーネントやシステムが、ユーザや顧客のニーズを満たしているか、ユーザや顧客のビジネスプロセスに適合するかを判定するために実施する。通常、ソフトウェアだけでなく、ハードウェアも含める。

**サニティテスト(sanity test)**: smoke test を参照のこと。

**サブパス(subpath)**: コンポーネント内の一連の実行命令文。

## し

**資格認定(qualification)**: 定義された要件を満たした能力を明らかにするプロセス。資格有(qualified)と言う用語は相当する能力を有する場合に使われる [ISO 9000]

**資源利用(resource utilization)**: 規定の条件下でソフトウェアが機能を実施する場合、適正なサイズの適正なリソース・タイプ(たとえば、プログラムが使うメインメモリとセカンダリメモリ量、必要な一時ファイルやオーバーフローファイルの大きさ)を利用するといふソフトウェア製品の能力。[After ISO9126] efficiency も参照のこと。

**資源利用テスト(resource utilization testing)**: ソフトウェア製品の資源利用を判定するためのテストの手順。efficiency testing も参照のこと。

**システム(system)**: 特定の機能や、機能の組み合わせを実現するために組織化したコンポーネントの集合。[IEEE610]

**システムオブシステムズ(system of systems)**: 特定分野に共通する目的や課題のために、広範囲にわたる分散した複数のレベル・ドメインをネットワークでつないだ、異機種環境システム

**システムテスト(system testing)**: 統合されたシステムが、指定された要件を満たすことを実証するためのテストの手順。[Hetzel]

**システム統合テスト(system integration testing)**: システムとパッケージを統合するテスト。また、外部システム(たとえば、電子データ交換やインターネット)とのインターフェースのテスト。

**事前条件(precondition)**: 特定のテストやテスト手順でコンポーネントやシステムを実行する前に、満足しなければならぬ環境と状態の条件。

**事後条件(postcondition)**: テストやテスト手順の実行後に満足すべき、環境と状態の条件。

**実行可能パス(feasible path)**: パスを実行するための、入力値と事前条件の組み合わせが存在するパス。

**実行後比較(post-execution comparison)**: ソフトウェアの実行後に、実際と期待結果を比較すること。

**実行ステートメント(executable statement)**: コンパイル時にオブジェクトコードに翻訳され、プログラムが走る時に手順に沿って実行されて、データに対して行為を行うであろう命令。

**実行不可能パス(infeasible path)**: 入力値をどのように組み合わせても実行できないパス。

**実際の結果(actual outcome)**: actual result を参照のこと。

**実際の結果(actual result)**: テストで、コンポーネントやシステムが出力、表示した動作。

**失敗(fail)**: 実行結果が予想した結果と一致しない場合、テストは「失敗」と見なす。

**自動化テストウェア(automated testware)**: ツールスクリプトのような自動化されたテストの中で使われるテストウェア。

**シナリオテスト(scenario testing)**: use case testing 参照のこと。

**市販ソフトウェア(COTS)(commercial off-the-shelf software)**: off-the-shelf software を参照のこと。

**シミュレーション(simulation)**: 物理的なシステム、あるいは、抽象的なシステムの代表的な動作特性を他のシステムで模倣すること。[ISO2382/1]

**シミュレータ(simulator)**: テストで使われる装置、コンピュータプログラム、システムで、ある入力組み合わせに対し、特定のシステムのような振る舞いや動作をするもの。[After IEEE610, DO178b] emulator も参照のこと。

**習得性(learnability)**: ユーザが当該ソフトウェアの使用法を学ぶことができるソフトウェア製品の能力。[ISO9126] usability も参照のこと。

**終了基準(completion criteria)**: exit criteria を参照のこと。

**終了基準(exit criteria)**: あるプロセスを公式に完了させるため、ステークホルダが承認した一般・特定条件。終了基準の目的は、未完了部分のあるタスクが、完了と見なされるのを防ぐことにある。終了基準は、テスト完了の報告や、計画で利用する。[After Gilb and Graham]

**終了点(exit point)**: コンポーネント中の最後の実行命令文。

**重要度(severity)**： コンポーネントやシステムの開発、また運用に対し欠陥が与える影響の度合。[After IEEE610]

**出力(output)**： コンポーネントが書く変数(コンポーネントの内部、外部に格納される)。

**出力値(output value)**： 出力のインスタンス。outputも参照のこと。

**出力ドメイン(output domain)**： 有効な出力値を選択できる集合。domainも参照のこと。

**仕様(specification)**： コンポーネントやシステムの要件、設計、動作、その他の特性を記述したドキュメント。記述内容を満足していることをどのように判定すべきかの手順を示したものも多い。完全で、正確で、検証可能な方式で記述することが望ましい。  
[After IEEE610]

**小規模統合テスト(integration testing in the small)**： component integration testingを参照のこと。

**条件(condition)**： 真または偽として評価することのできる論理的表現。たとえば、 $A \wedge B$ 。test conditionも参照のこと。

**条件カバレッジ(condition coverage)**： テストスイートが実行した条件のパーセンテージ。条件カバレッジを100%にするには、各判定文の全ての単一条件に対し、真と偽をテストする必要がある。

**条件組合せカバレッジ(condition combination coverage)**： multiple condition coverageを参照のこと。

**条件組合せテスト(condition combination testing)**： multiple condition testingを参照のこと。

**条件結果(condition outcome)**： 真または偽になる条件の評価。

**条件テスト(condition testing)**： ホワイトボックステスト設計技法の1つ。条件を実行するテストケースを設計する。

**条件判定カバレッジ(condition determination coverage)**： 判定結果に対し、独立に影響する全単一条件をテストスイートが実行したパーセンテージ。100%の条件判定カバレッジは、100%の判定条件カバレッジを意味する。

**条件判定テスト(condition determination testing)**： ホワイトボックステスト設計技法の1つ。判定結果に対し、独立に影響する単一条件を実行するテストケースを設計する。

**状態遷移(state transition)**： コンポーネントやシステムにおいて、ふたつの状態の間を遷移すること。

**状態遷移図(state diagram)**： コンポーネントまたはシステムが取りうる状態を示し、ある状態から他への状態の変化の原因となる、(または)その結果として生ずる、イベントや状況を表わすダイアグラム。[IEEE610]

**状態遷移テスト(state transition testing)**： ブラックボックステストの設計技法で、無効と有効の状態遷移を実行するテストケースを設計する。N-switch testingも参照のこと。

**状態遷移表(state table)**： 発生する可能性のあるイベントと状態の組み合わせから、生じる結果を示す遷移のテーブル。無効な遷移と、有効な遷移の両方を示す。

**仕様ベーステスト(specification-based testing)**： black box testing参照のこと。

**仕様ベーステスト設計技法(specification-based test design technique)**： black box test design technique参照のこと。

**仕様ベースの技法(specification-based technique)**： black box test design techniqueを参照のこと。

**証明(certification)**： コンポーネント、システム、人が、定義された要求事項に従っていることを確認する手順。たとえば、試験に合格するなど。

**書記 (scribe) :** レビューミーティングで指摘された欠陥や、改善の提案を、所定の用紙に記録する人。筆記者は、記録が読みやすく、理解しやすいことに留意する必要がある。

**シンタックステスト[構文テスト](syntax testing) :** 入力定義域と出力定義域の定義を基にテストケースを設計するブラックボックステストの設計技法。

**信頼性 (reliability) :** あらかじめ決めた運用回数、または、指定した期間、決められた条件下で、必要な機能を果たせるソフトウェア製品の能力。[ISO9126]

**信頼性テスト (reliability testing) :** ソフトウェア製品の信頼性を判定するテスト手順。

**信頼テスト (confidence test) :** smoke test を参照のこと。

**信頼度成長モデル (reliability growth model) :** コンポーネントやシステムのテストにおいて、信頼性に関する故障を発見し、その欠陥を取り除いていくことで時間と共に信頼性が成長することを示すモデル。

## す

**遂行済み(の物) (exercised) :** 入力値が、命令、判定、または他の構造的要素のような要素の実行を引き起こす時、テストケースによってプログラム要素は遂行されると言われる。

**垂直トレーサビリティ[垂直追跡性](vertical traceability) :** 開発ドキュメントからコンポーネントまでの各階層を通して要件を追跡すること。

**水平トレーサビリティ(horizontal traceability) :** テスト関連ドキュメント(テスト計画書、テスト設計仕様書、テストケース仕様書、テスト手順仕様書、テストスクリプトなど)の階層を通して、あるテストレベルで対象となる要件を追跡すること。

**スクリプト言語 (scripting language) :** 実行可能テストスクリプトを書くプログラミング言語。テスト実行ツール(たとえば、キャプチャ・プレイバックツール)で使う。

**スクリプトテスト (scripted testing) :** すでに記述されているテスト順序をたどることで実行するテスト方法。

**スタブ (stub) :** 特定のコンポーネント(仮にAと呼ぶ)をテストするため、Aに呼び出される(特定目的のための最小限度の)コンポーネント。スタブがないと、実物ができるまで、開発やテストを待たねばならない。スタブは、最終的には、呼び出されるコンポーネントで置き換える。[After IEEE610]

**ステータスアカウンティング (status accounting) :** 構成管理の要素で、構成を効果的に管理する場合に必要な情報を記録、報告すること。必要情報には、承認済み構成識別表、承認済み構成に対する変更提案の状態表、承認済み変更の実施状態の一覧表などがある。[IEEE610]

**ステートメント (statement) :** プログラミング言語の実体。実行の最小単位。

**ステートメントカバレッジ (statement coverage) :** テストスイートによって実行されたステートメントのパーセンテージ。命令(文)網羅とも呼ぶ。

**ステートメントテスト (statement testing) :** ホワイトボックステストの設計技法で、命令文を実行するテストケースを設計する。

**ストレージ (storage) :** resource utilization 参照のこと。

**ストレージテスト (storage testing) :** resource utilization testing 参照のこと。

**ストレステスト (stress testing) :** 予測または仕様化した負荷、もしくはメモリやサーバなどのリソースの可用性を低減したとき

の限界、または、それを越えた条件でシステムやコンポーネントを評価するテスト。[After IEEE610] performance testing, load testing も参照のこと。

**ストレステストツール(stress testing tool)**: ストレステストをサポートするツール。

**スペシファイドインプット(specified input)**: 仕様から結果を予測した入力。

**スモークテスト(smoke test)**: 定義・計画した全テストケースのサブセット。プログラムの必須機能が正常に動作することを確認するのが目的で、コンポーネントやシステムの主要機能を網羅し、細かい点は無視する。日次のビルドやスモークテストは、産業界での最も効率的・効果的な実践方法のひとつ。intake test も参照のこと。

**スレッドテスト(thread testing)**: コンポーネント統合テストの1つ。要件のサブセットを開発した後、コンポーネントを段階的に統合する。階層レベルごとにコンポーネントを統合する方式とは対極に位置する。

## せ

**正確性(accuracy)**: ソフトウェア製品が、必要な正確さを備え、正しい結果や効果、あるいは、承認範囲内の結果や効果を出力する能力。[ISO9126] functionality testing も参照のこと。

**制御フロー(control flow)**: コンポーネントやシステムの実行における一連のイベント(パス)

**制御フロー分析(control flow analysis)**: コンポーネントやシステムを実行するときのイベント(パス)のシーケンス表現をベースにした静的解析の一種

**制御フローグラフ(control flow graph)**: コンポーネントやシステムの実行における全てのイベント(パス)のシーケンスを抽象表現したもの。

**制御フローパス(control flow path)**: path を参照のこと。

**成熟度(maturity)**:

- (1)プロセスや作業の有効性と効率に関する組織の能力。Capability Maturity Model、Test Maturity Model も参照のこと。
- (2)ソフトウェアの欠陥の結果として発生する故障を避けることのできるソフトウェア製品の能力。[ISO9126] reliability も参照のこと。

**静的アナライザ(static analyzer)**: 静的解析を行うツール。

**静的解析(static analysis)**: (たとえば、要件またはコードなどの)ソフトウェア成果物を実行せずに解析すること。

**静的解析ツール(static analysis tool)**: static analyzer 参照のこと。

**静的コードアナライザ(static code analyzer)**: 静的コード解析を実施するツール。ソースコードをチェックし、コーディング基準や品質メトリクスへの準拠、データフローの不整合などを検出する。

**静的コード解析(static code analysis)**: ソフトウェアを実行せずにソースコードを解析すること。

**静的テスト(static testing)**: そのソフトウェアの実行をせずに実施する仕様レベルまたは実装レベルのコンポーネントまたはシステムのテスト。たとえば、レビュー、または、静的コード解析。

**性能(performance)**: システムやコンポーネントが、処理時間やスループット率の制約内で、定義した機能を果たす度合。[After IEEE610] efficiency も参照のこと。

**性能テスト(performance testing)**: ソフトウェア製品のパフォーマンスを判定するためのテスト手順。efficiency testing も参照のこと。

**性能テストツール(performance testing tool)：** 性能テストを支援するツール。負荷生成機能とテストランザクション測定機能の2つを備えることが多い。負荷生成機能は複数ユーザや大量の入力データをシミュレートする。テストの実行中、選択したランザクションの応答時間を計測・記録する。性能テストツールは、テストの結果記録をもとにしたレポートや、応答時間に対する負荷のグラフを提供することが多い。

**性能プロファイル(performance profiling)：** 性能テスト、ロードテスト、ストレステストでの利用者のプロファイルの定義。プロファイルには、コンポーネントやシステムの運用プロファイルをベースにした想定、もしくは実際の使い方や負荷の急変箇所を反映しているべきである。

**製品受け入れテスト(production acceptance testing)：** operational acceptance testing を参照のこと。

**セーフティクリティカルシステム(safety critical system)：** 故障や誤動作が人命や人々への深刻な損害、もしくは機器へのダメージや環境被害となる可能性のあるシステム。

**セキュリティ(security)：** プログラムとデータに対して、偶然または意図的な不正アクセスを防ぐソフトウェア製品の能力。[ISO9126] functionality も参照のこと。

**セキュリティツール(security tool)：** セキュリティツール:セキュリティの操作を支援するツール。

**セキュリティテスト(security testing)：** ソフトウェア製品のセキュリティを判定するテスト。functionality testing も参照のこと。

**セキュリティテストツール(security testing tool)：** セキュリティの脆弱性をテストするための支援をするツール。

**全数テスト(exhaustive testing)：** テストのアプローチの1つ。テストスイートにより、入力値と事前条件の全組み合わせをテストすること。

## そ

**相互運用性(interoperability)：** 1つ以上の指定されたコンポーネントやシステムと情報を交換できるソフトウェア製品の能力。[After ISO9126] functionality も参照のこと。

**相互運用性テスト(interoperability testing)：** ソフトウェア製品の相互運用性を判定するテスト手順。functionality testing も参照のこと。

**操作環境(operational environment)：** ユーザや顧客のサイトにインストールしたハードウェアやソフトウェア。この環境で、テスト中のコンポーネントやシステムを動作させる。操作環境のソフトウェアには、オペレーティングシステム、データベース管理システム、その他のアプリケーションを含むこともある。

**操作性(operability)：** ユーザがソフトウェア製品を制御できるようにするソフトウェア製品の能力。[ISO9126] usability も参照のこと。

**測定(measurement)：** ある実体の特性を表すため、数字や種別を付加する手順。[ISO14598]

**測定尺度(measurement scale)：** 実行するデータ分析の様式を限定する尺度。[ISO14598]

**測定値(measure)：** 測定することによって、ある実体の特性に付加した数字や種別。[ISO14598]

**ソースステートメント(source statement)：** statement 参照のこと。

**ソフトウェア(software)：** コンピュータプログラム、プロシージャ。場合によればコンピュータシステムの運用に関連したドキュメントとデータも含む。

**ソフトウェア・故障モード影響解析(software Failure Mode and EFFECT Analysis (SFMEA))：** Failure Mode and

Effect Analysis (FMEA)を参照のこと。

**ソフトウェア・故障モード影響・致命度解析 (software Failure Mode EFFECT, and Criticality Analysis (SFMECA))**: Failure Mode and Effect, and Criticality Analysis (FMECA)を参照のこと。

**ソフトウェア攻撃 (software attack)**: attackを参照のこと。

**ソフトウェア製品特性 (software product characteristic)**: quality attribute 参照のこと。

**ソフトウェアテストインシデント (software test incident)**: incident 参照のこと。

**ソフトウェアテストインシデントレポート (software test incident report)**: incident report 参照のこと。

**ソフトウェアフィーチャ (software feature)**: featureを参照のこと。

**ソフトウェア・フォールトツリー解析 (software Fault Tree analysis (SFTA))**: Fault Tree Analysis (FTA)を参照のこと。

**ソフトウェア品質 (software quality)**: ソフトウェア製品の全体として、機能が、明示的、暗示的なニーズを満たしている能力。  
[After ISO9126]

**ソフトウェア品質特性 (software quality characteristic)**: quality attribute 参照のこと。

**ソフトウェア有用性測定一覧表 (Software Usability Measurement Inventory (SUMI))**: コンポーネントまたはシステムのユーザ満足などの有用性を評価するための、有用性テスト技法に基づいた質問表。[Veenendaal]

**ソフトウェアライフサイクル (software life cycle)**: ソフトウェアプロダクトの最初から最後、つまり企画段階から利用終了までの期間。ソフトウェアライフサイクルは通常、コンセプトフェーズ、要件フェーズ、設計フェーズ、実装フェーズ、テストフェーズ、インストールとチェックアウトフェーズ、運用と保守フェーズを含み、時に除去フェーズを含むこともある。これらのフェーズは重複することもあるし、反復することもある。

## た

**大規模統合テスト (integration testing in the large)**: system integration testingを参照のこと。

**タイムビヘイビア (time behavior)**: performance 参照のこと。

**ダーティテスト (dirty testing)**: negative testingを参照のこと。

**妥当性確認 (validation)**: 検査、および、特定の使用方法や応用に対する要件が満たされていることを客観的な証拠で確認すること。[ISO9000]

**段階表現 (staged representation)**: 成熟度レベルとして確立したプロセスエリアののゴールに達したかを見るモデル構造であり、各レベルは次のレベルの土台となるよう組み立てられている。[CMMI]

**探索的テスト (exploratory testing)**: 非公式なテスト設計技法の1つ。テストを実施する過程で、テスト担当者がテスト実施情報を活用しながらテスト設計を制御し、積極的かつ質の高い新しいテストケースを設計する。[After Bach]

## ち

**チェック担当 (checker)**: reviewerを参照のこと。

**置換性(replaceability)**: 同じ環境で同じ目的のために使用している別ソフトウェア製品の代替として使用可能となるソフトウェア製品の能力。[ISO9126] portability も参照のこと。

**中止基準(suspension criteria)**: テストケースのすべて、または、一部のテスト作業を(一時的に)止める場合に参照する基準。[After IEEE829]

**抽象的テストケース(abstract test case)**: high level test case を参照のこと。

**注文ソフトウェア(bespoke software)**: 特定ユーザや特定顧客専用開発したソフトウェア。反対語は off-the-shelf software (市販ソフトウェア)。

**Chow のカバレッジメトリクス(Chow's coverage metrics)**: N-switch coverage を参照のこと。[Chow]

**直交表(orthogonal array)**: 特殊な数学的性質を使って構築した2次元の配列であり、配列の中から選択した2つの列から、その配列の中の各値に対して全てのペアの組み合わせを提供する。

**直交表テスト(orthogonal array testing)**: 直交表を使った変数のオールペア組み合わせテストの体系的な方法。変数を全て組み合わせたときの数をオールペア組み合わせでテストできるまでに減らす。pairwise testing も参照のこと。

て

**低位レベルテストケース(low level test case)**: 入力データ用の具体的な(実行レベルの)値と予期する結果を用いるテストケース。ハイレベルテストケースからの論理演算子は、実効値によって置き換えられる。high level test case も参照のこと。

**定義使用ペア(definition-use pair)**: 変数の定義と、その変数の使用とを結合したもの。変数の使用の例として、計算(たとえば、乗算)、パス実行の制御(制御的使用)がある。

**提供品(deliverable)**: (業務用)製品の開発担当者以外に提供するあらゆる(業務用)製品。

**デイリービルド(daily build)**: システムやアプリケーション全体を毎日(通常、夜間)、コンパイル、リンクして作成する開発の活動。これにより、最新の変更を反映した一貫性のあるシステム、アプリケーションにいつでもアクセスできる。

**デヴィエーション(deviation)**: 逸脱。incident を参照のこと。

**デヴィエーションレポート(deviation report)**: 逸脱報告。incident report を参照のこと。

**適合テスト(conformance testing)**: compliance testing を参照のこと。

**テクニカルレビュー(technical review)**: どのような技術的アプローチを取るかで意見を一致させることを目的とした、ピアグループによるディスカッション。[Gilb and Graham, IEEE1028] peer review も参照のこと。

**デザインベーステスト(design-based testing)**: テストに対するアプローチ法の1つ。コンポーネントやシステムの構造や詳細設計を基にテストケースを設計するもの(たとえば、コンポーネントやシステム間のインターフェースのテスト)。

**デシジョンカバレッジ(decision coverage)**: テストスイートによって用いられた、判定結果のパーセンテージ。100%のデシジョンカバレッジは、100%の branch coverage (ブランチカバレッジ)と100%の statement coverage (ステートメントカバレッジ)の両方を意味する。

**デシジョンテーブル(decision table)**: 入力と刺激(原因)、および、対応する出力と処理(結果)の組み合わせを示す表。テストケースの設計に利用できる。日本語で決定表とも呼ぶ。

**デシジョンテーブルテスト(decision table testing)**: ブラックボックステスト設計技法の1つ。決定表にある入力と刺激(原因)の組み合わせを実行するテストケースを設計する。decision table も参照のこと。

**テスト(test)**: ひとつ以上のテストケースの組み合わせ。[IEEE829]

**テスト(testing)**: 成果物が定義した要件を満足するかを判定し、目的に合致することを実証し、欠陥を見つけるため、ソフトウェア製品や関連成果物に対し、計画、準備、評価をすること。全てのライフサイクルを通じて実施する静的、動的なプロセス。

**テストアイテム(test item)**: テストを実施する個々の要素。通常、ひとつのテスト対象に対し、多数のテストアイテムがある。test object も参照のこと。

**テストアイテム送付レポート(test item transmittal report)**: release note 参照のこと。

**テストアプローチ(test approach)**: あるプロジェクトのためのテスト戦略を実現化したもの。この中には、(テスト実施)プロジェクトのゴールと評価済みリスクを基に決めた決定事項、テストプロセスの開始ポイント、適用するテスト設計技法、テスト終了基準、実施するテストタイプを含む。

**テストインシデント(test incident)**: incident 参照のこと。

**テストインシデントレポート(test incident report)**: incident report 参照のこと。

**テストインフラ(test infrastructure)**: テスト環境、テストツール、オフィス環境、処理手続きから成るテストの実施に必要な構造的な物。

**テストウェア(testware)**: 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。たとえば、ドキュメント、スクリプト、入力、期待結果、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。[After Fewster and Graham]

**テストオラクル(test oracle)**: テスト対象のソフトウェアが実際に出力した結果とを比較する期待結果のソース。オラクルは、実在する(ベンチマーク用の)システム、ユーザマニュアル、個人の専門知識の場合があるが、コードであってはならない。[After Adrion]

**テストカバレッジ(test coverage)**: coverage を参照のこと。

**テスト環境(test environment)**: テストの実行に必要なハードウェア、計装、シミュレータ、ソフトウェアツール、その他の支援要素を含む環境。[After IEEE610]

**テスト完了基準(test completion criteria)**: exit criteria 参照のこと。

**テスト技法(test technique)**: test design technique を参照のこと。

**テスト駆動型開発(test driven development)**: 本体のソフトウェアを開発する前に、テストケースの開発(場合によっては自動化)するソフトウェア開発手法。

**テスト計画(test plan)**: 計画されたテスト活動の狙い、アプローチ、リソース、スケジュールを記述するドキュメント。テストアイテム、テストすべきフィーチャ、タスク、各タスク担当者、テスト担当者の独立の度合、テスト環境、使われるテスト設計技法と開始/終了基準、それらの選択の理論的根拠、それに代替計画を必要とするあらゆるリスクを特定する。これはテスト計画プロセスの記録である。[After IEEE829]

**テスト計画作業(test planning)**: テスト計画を作成し、更新すること。

**テストケース(test case)**: 入力値、実行事前条件、期待結果、そして、実行事後条件の組み合わせで、特定のプログラムパスを用いることや指定された要件の遵守を検証することのような、特定の目的またはテスト条件のために開発されたもの。[After IEEE610]

**テストケース仕様(test case specification)**: テストアイテム用のテストケースの組み合わせ(目的、入力、テスト実行、期待

される結果、実行事前条件)を規定するドキュメント。[After IEEE829]

**テストケーススイート(test case suite)**: test suite 参照のこと。

**テストケース設計技法(test case design technique)**: test design technique 参照のこと。

**テスト結果(test outcome)**: result 参照のこと。

**テスト結果(test result)**: result 参照のこと。

**テスト結果記録(test log)**: テスト実行の詳細を時系列的に記録したもの。[IEEE829]

**テスト結果記録作業(test logging)**: 実行されたテストの情報を記録するプロセス。

**テスト結果比較(test comparison)**: テスト対象のコンポーネントやシステムの実行結果と期待結果の違いを明らかにする手順。テスト比較は、テスト実行中(動的比較)、または、テスト実行後に実施する。

**テストプロセス(test process)**: 基本的なテストプロセスは、テストの計画とコントロール、テストの分析と設計、テストの実装と実行、終了基準の評価と報告、テスト終了作業によって構成される。

**テストコントロール(test control)**: 監視中に計画から外れていることを検出した場合、テストプロジェクトを軌道修正するための対策を考えたり実施したりするテストマネジメントタスクのひとつ。test management も参照のこと。

**テストサイクル(test cycle)**: 特定可能な単一のテスト対象のリリースに対し、テストプロセスを実行すること。

**テスト再現性(test reproducibility)**: テストを実行する度に、同じ結果を出力できるかどうかを示す特性。

**テストサマリレポート(test summary report)**: テスト作業と結果を要約した報告用ドキュメント。実行したテストケースがテスト終了基準を満足しているかの評価も含む。[After IEEE829]

**テストシチュエーション(test situation)**: test condition 参照のこと。

**テスト実行(test execution)**: テスト対象のコンポーネントやシステムでテストを実行し、実際の結果を出力するプロセス。

**テスト実行技法(test execution technique)**: 実際のテストを、手動または自動で実行する方法。

**テスト実行自動化(test execution automation)**: たとえば記録・再生ツールのようなソフトウェアを使用して、テストの実行、実行結果と期待結果の比較、テスト条件の設定、その他のテスト制御やレポート機能を実施すること。

**テスト実行スケジュール(test execution schedule)**: テスト手順を実行していくための計画。テスト実行スケジュールには、テスト手順書の実際に実行する内容とその実行順を記述する。

**テスト実行ツール(test execution tool)**: テストツールの一種。キャプチャ・プレイバックのような自動化テストスクリプトを使い、他のソフトウェアを実行できる。[Fewster and Graham]

**テスト実行フェーズ(test execution phase)**: ソフトウェア開発ライフサイクル中で、ソフトウェア製品のコンポーネントを実行し、要件を満たしているか判定する期間。[IEEE610]

**テスト実装(test implementation)**: テストデータを考え出し、テスト手順の開発および優先度付けを行うプロセス。テストハynesの準備や自動テストスクリプト記述を含むこともある。

**テスト失敗(test fail)**: fail 参照のこと。

**テスト自動化(test automation)**: ソフトウェアを使って、テストマネジメント、テスト設計、テスト実行、結果チェックなどのテスト

ト作業の実行を支援すること。

**テストシナリオ (test scenario) :** test procedure specification 参照のこと。

**テスト終了作業 (test closure) :** テストプロセスのテスト終了作業フェーズの間、経験、テストウェア、事実、数字を纏めるために、データは完了した活動から集められる。テスト終了作業フェーズはテストウェアの仕上げ、保管とテスト評価レポートの準備を含むテストプロセスの評価から成る。test process も参照のこと。

**テスト仕様化技法 (test specification technique) :** test design technique 参照のこと。

**テスト条件 (test condition) :** コンポーネントやシステムのアイテムやイベントで、テストケースにより検証できるもの。たとえば、機能、トランザクション、品質特性、構造要素など。

**テスト仕様書 (test specification) :** テスト設計仕様、テストケース仕様、テスト手順仕様から成るドキュメント。

**テスト進捗レポート (test progress report) :** 定期的に作成するテストの活動と結果をまとめたドキュメント。テスト活動の進捗がベースライン(当初のテスト計画など)に沿っているか報告するため、かつリスクや代替案の決定が必要なマネジメント層へ伝えるために作成する。

**テストスイート (test suite) :** テスト対象のコンポーネントまたはシステムのためにテストケースをまとめたもの。ひとつのテストの事後条件は、次のテストの事前条件として利用される。

**テストスケジュール (test schedule) :** 活動、タスク、またはテストプロセスのイベントに関して、開始日終了日、時間、依存関係が識別できるリスト。

**テストスクリプト (test script) :** 一般的にテスト手順を指して用いられる。特に自動化時のスクリプトを指す。

**テストステージ (test stage) :** test level 参照のこと。

**テストセッション (test session) :** テスト実行を続ける一定時間。探索的テストでは、各テストセッションはひとつのチャータに焦点を当ててテストを行う。しかし、セッション中にテストは新しい気づきや問題に対してもまた探索する。テストはまとめてテストケースを考えて実行し、進捗を記録する。exploratory testing も参照のこと。

**テスト成熟度モデル (TMM) (Test Maturity Model (TMM)) :** 能力成熟度モデル (CMM) に関連した、テストプロセス改善のための5レベルのフレームワークであり、効果的なテストプロセスの重要要素を記述している。

**テスト成熟度モデル統合 (test Maturity model Integrated (TMMi)) :** 能力成熟度モデル統合 (CMMI) と関連させた5段階からなるテストプロセス改善のためのフレームワークであり、効果的なテストプロセスのためのキーとなる要素を表現している。

**テスト設計 (test design) :**

(1) test design specification を参照のこと。

(2) 全般的なテストの目的を具体的なテスト条件とテストケースに変換するプロセス。

**テスト設計技法 (test design technique) :** テストケースを作成したり選択したりするための手順

**テスト設計仕様 (test design specification) :** テストアイテムのテスト条件(網羅アイテム)、詳細なテストアプローチ、および、関連するハイレベルテストケースを記述したドキュメント。[After IEEE829]

**テスト設計ツール (test design tool) :** テスト設計を支援するツール。たとえば要件マネジメントツールなどの CASE ツールのリポジトリに保存している仕様やテスト設計ツールの中に保存してある仕様化したテスト条件、またはコードからテスト用の入力を生成する。

**テストセット (test set) :** test suite 参照のこと。

**テスト戦略(test strategy):** テスト戦略:組織や(ある特定の or 複数の)プロジェクトで実施するテストレベルと各テストレベルでのテスト内容を高位レベルで説明したもの。

**テスト対象(test object):** テストされるコンポーネントまたはシステム。test item も参照のこと。

**テストタイプ(test type):** コンポーネントまたはシステムに相関する品質特性に向けたテスト活動の分類。各テストタイプは、たとえば信頼性テスト、有用性テスト、回帰テストなどのように特定のテストの目的にフォーカスしている。テストタイプはひとつまたは複数のテストレベルまたはテストフェーズで行われる。[After TMap]

**テストターゲット(test target):** テスト終了基準の集合。

**テスト担当者(tester):** コンポーネントやシステムのテストを実施する熟練した専門家。

**テストチャータ(test charter):** テストの目的を明記したもの。テスト実施法のアイディアを含む場合もある。探索的テストにて使用することが多い。exploratory testing も参照のこと。

**テストツール(test tool):** 計画、制御、仕様、初期ファイルやデータの作成、テスト実行、テスト分析等のテスト作業を支援するソフトウェア。[TMap] CAST も参照のこと。

**テスト手順(test procedure):** test procedure specification 参照のこと。

**テスト手順仕様(test procedure specification):** テストの実行のために、一連の手順を指定するドキュメント。テストスクリプト、または、手動テストスクリプトとしても知られる。[After IEEE829]

**テストデータ(test data):** テスト実行前に実在する(たとえば、データベースの中)データであり、テスト対象のコンポーネントやシステムに影響を与えたり、影響を受けたりするもの。

**テストデータジェネレータ(test generator):** test data preparation tool 参照のこと。

**テストデータ準備ツール(test data preparation tool):** テストに使うデータを選択(データベース内の実データなど)、または、作成、生成、操作、編集をするためのツール。

**テストドライバ(test driver):** driver 参照のこと。

**テスト入力データ(test input):** テスト実行中にテストオブジェクトが外部ソースから受け取ったデータ。外部ソースは、ハードウェア、ソフトウェア、人の場合がある。

**テストの目的(test objective):** テストを設計、実行する理由や目的。

**テストパス(test pass):** pass 参照のこと。

**テストハーネス(test harness):** テスト実行に必要なスタブやドライバから成るテスト環境。

**テストパフォーマンスインジケータ(test performance indicator):** テストをよい方向へ導き制御するために利用する有効性と効率のハイレベルなメトリクス。たとえば、欠陥検出率(DDP)。

**テスト比較ツール(test comparator):** テスト実行時の実際の結果と期待結果との比較を自動的に実施するテストツール。

**テスト評価レポート(test evaluation report):** 全てのテスト作業と結果を要約した、テストプロセスの最後に作るドキュメント。テスト手順の評価と、学んだ教訓を含むこともある。

**テストフェーズ(test phase):** テスト活動をプロジェクト中で管理(マネジメント)しやすいフェーズにまとめた組み合わせ。たとえば、あるテストレベルの実行活動。[After Gerard]

**テストプロセス改善(TPI) (Test Process Improvement (TPI))**: 特定システムテストと受け入れテストを対象とした、テストプロセスの効率を上げるための重要な要素が記述してある、テストプロセス改善のための連続したフレームワーク。

**テストベース(test basis)**: コンポーネント要件やシステム要件を推測できる全てのドキュメント。これらのドキュメントがテストケースのベースとなる。公式な改訂手順を経ないとドキュメントの改訂ができない場合、そのテストベースを「凍結テストベース」と呼ぶ。[After TMap]

**テストベッド(test bed)**: test environment 参照のこと。

**テストポイント分析(TPA) (Test Point Analysis (TPA))**: ファンクションポイント分析を基に、数式でテストの必要リソースを見積もる方法。[TMap]

**テストポリシー(test policy)**: 組織ごとでのテストにかかわる原理原則、アプローチ、主要な目的を記述するハイレベルのドキュメント。

**テストマネジメント(test management)**: テスト活動の計画、見積り、監視、制御。主としてテストマネージャによって実施される。

**テストマネジメントツール(test management tool)**: テストプロセスの管理と制御を支援するツール。テストウェア管理、テストスケジューリング、結果の記録、進捗管理、インシデント管理、テスト報告等の能力を持つことが多い。

**テストマネージャ(test manager)**: テストの活動とリソースのマネジメント、テスト対象の評価に責任を持つ個人。テストプロジェクトを指揮、コントロール、運営し、テスト対象の評価を計画し実施する。

**テスト見積り(test estimation)**: 各種概算結果(e.g.工数(effort spent), 完了日(completion date), コスト(costs involved), テストケース数(number of test cases), など)。ただし、入力情報が完全ではなく不確か、もしくは余計な情報を含む可能性もある。

**テストモニタリング(test monitoring)**: テストプロジェクトの状態の周期的なチェックに関連した活動を扱うテスト管理タスク。実際の活動と計画した活動を比較した報告が準備される。test management も参照のこと。

**テスト容易化要件(testable requirements)**: 要件が満たされているかどうかを判定するために、テストの設計(と、テストケースの作成)や実行が可能となる要件の度合。[After IEEE610]

**テスト容易性(testability)**: 変更されたソフトウェアがテストしやすいかどうかを示すソフトウェア製品の能力。[ISO9126 maintainability も参照のこと。

**テスト容易性レビュー(testability review)**: テストプロセスの入力ドキュメントとして十分な品質レベルにあるかどうかを判定するため、テストベースを詳細にチェックすること。[After TMap]

**テスト要件(test requirement)**: test condition 参照のこと。

**テストラン(test run)**: テスト対象を特定のバージョンでテストを実行すること。

**テストランログ(test run log)**: test log 参照のこと。

**テストリグ(test rig)**: test environment を参照のこと。

**テストリーダー(test leader)**: test manager を参照のこと。

**テストレコーディング[テスト結果記録作業](test recording)**: test logging を参照のこと。

**テストレコード[テスト結果記録](test record)**: test log を参照のこと。

**テストレベル(test level)**: 組織的に管理されるテスト作業のグループ。テストレベルはプロジェクトの責任に対応する。テスト

レベルの例には、コンポーネントテスト、統合テスト、システムテスト、受け入れテストがある。[After TMap]

**テストレポート(test report)**： test summary report を参照のこと。

**データ完全性テスト(data integrity testing)**： database integrity testing を参照のこと。

**データ駆動テスト(data driven testing)**： スクリプト作成技術の1つ。テスト入力と期待結果をテーブルやスプレッドシートに格納し、1つの制御スクリプトでテーブル中の全テストを実行するもの。キャプチャ/プレイバックツールのような、テスト実行ツールでのアプリケーションで使うことが多い。[Fewster and Graham] keyword driven testing も参照のこと。

**データ定義(data definition)**： 変数値を書き当てる実行命令文。

**データフロー(data flow)**： データオブジェクトの順序と、起こり得る状態の変化を抽象的に表現したもの。オブジェクトの状態は、生成、使用、破壊のいずれかになる。[Beizer]

**データフロー解析(data flow analysis)**： 変数の定義と用法を基にした静的解析のひとつ。

**データフローカバレッジ(data flow coverage)**： 「変数の定義・使用」を1組とし、テストスイートがどれだけ実行したかをパーセンテージで表したものの。

**データフローテスト(data flow testing)**： ホワイトボックステスト設計技法の1つ。変数の「定義と使用」の組を実行するようにテストケースを設計する。

**データベース完全性テスト(database integrity testing)**： データ(ベース)をアクセス・管理する方法や手順をテストすること。アクセス方法、手順、データ規約が期待通りに動作すること、データベースへのアクセス中にデータが破壊されたり、期待と反して削除、更新、作成されたりしないことを確認する。

**ディフェクトマスキング(defect masking)**： ひとつの欠陥が他の欠陥の発見を妨げる現象。[After IEEE610]

**デシジョンテスト(decision testing)**： ホワイトボックステスト設計技法の1つ。判定を実行するテストケースを設計する。

**手続きテスト(procedure testing)**： コンポーネントやシステムが、新規、もしくは現行利用者のビジネス手順、または運用手順と共に運用できることの確保を目的としたテスト。

**デッドコード(dead code)**： unreachable code を参照のこと。

**デバグガ(debugger)**： debugging tool を参照のこと。

**デバグ(debugging)**： ソフトウェアの故障の原因を見つけて、分析して、取り除くプロセス。

**デバグツール(debugging tool)**： 故障を再現して、プログラムの状態を調査し、対応した欠陥を見つけるために、プログラマが使用するツール。デバグガは、プログラムの変数をセツトし検査するために、どのプログラム命令の場所でもプログラムを停止したり、プログラムを1ステップずつ実行したりできる。

## と

**統計的テスト(statistical testing)**： 入力の統計的な分散モデルを使い、代表的なテストケースを作成するテスト設計技法。operational profile testing も参照のこと。

**凍結テストベース(frozen test basis)**： 公式な変更制御手順によってのみ、改訂が可能なテストベースドキュメント。baseline も参照のこと。

**統合(integration)**： コンポーネントやシステムを組み合わせ、さらに大きな集合体を作る手順。

**統合テスト(integration testing)**: 統合したコンポーネントやシステムのインターフェースや相互作用の欠陥を抽出するためのテスト。component integration testing、および、system integration testing も参照のこと。

**同時処理テスト(concurrency testing)**: 同じ時間枠内に発生した2つ以上の処理をコンポーネントやシステムがどう扱うか(処理を分割して交互に実行したり、並列に実行したりするなど)を判定するためのテスト。[After IEEE610]

**到達不能コード(unreachable code)**: 到達できないため、実行不能なコード。

**同値クラス(equivalence class)**: equivalence partition を参照のこと。

**同値分割(equivalence partition)**: 仕様上、コンポーネントやシステムの動作が同じと見なせる入力定義域や出力定義域の部分。

**同値分割カバレッジ(equivalence partition coverage)**: テストスイートが実行した同値領域のパーセンテージ。

**同値分割法(equivalence partitioning)**: ブラックボックステスト設計技法の1つ。同値領域から代表値を実行するテストケースを設計する。最低1回各同値領域を実行するように設計するのが原則。

**動的解析(dynamic analysis)**: 実行中のシステムやコンポーネントの動作(たとえば、メモリの使用効率、CPU の使用状況)を評価するプロセス。[After IEEE610]

**動的解析ツール(dynamic analysis tool)**: ソフトウェアコードの状態に関する実行時(ランタイム)情報を提供するツール。未害当てのポインタの検出、チェックポイントの計算、メモリの害当て・使用・解放のチェック、メモリリークの検出で使われることが多い。

**動的テスト(dynamic testing)**: コンポーネントやシステムのソフトウェアを実行させて確認するテスト。

**動的比較(dynamic comparison)**: (たとえば、テスト実行ツールを使用し)ソフトウェアの実行中に、実際の結果と期待結果を比較すること。

**ドキュメンテーションテスト(documentation testing)**: ユーザガイドやインストールガイドのような手引書の品質をテストすること。

**独立性(independence of testing)**: 責任を分離すること。これにより、客観的なテストを促進できる。[After DO-178b]

**トップダウンテスト(top-down testing)**: 統合テストを段階的に実施する方法。コンポーネント階層の最上位にあるコンポーネントからテストを開始する。下位レベルのコンポーネントは、スタブでシミュレーションする。上位コンポーネントのテストが終了すると、それを使って、下位コンポーネントをテストする。このプロセスは、最下位レベルのコンポーネントをテストするまで繰り返す。integration testing も参照のこと。

**ドメイン(domain)**: データの集合。ここから、有効な入力値や出力値を選ぶ。

**ドライバ(driver)**: コンポーネントやシステムを制御したり呼出したりする上位コンポーネントの代わりとなるソフトウェアコンポーネントやテストツール。[After TMap]

**トレーサビリティ(traceability)**: ドキュメントとソフトウェアの関連事項(たとえば、ある要件と、それを検証するテストケース)を特定する能力。horizontal traceability、vertical traceability も参照のこと。

に

**入力(input)**: コンポーネントが読む変数。コンポーネント内部または外部にある。

**入力値(input value)**: 入力のインスタンス。input も参照のこと。

**入力ドメイン(input domain)**： 入力値の集合。この集合から、有効なテストの入力値を選択する。domainも参照のこと。

## の

**能力成熟度モデル(CMM)(Capability Maturity Model (CMM))**： 効率的なソフトウェアプロセスの重要要素を5段階に分類して定義した枠組み。ソフトウェアの開発・保守における計画、製造、管理(マネジメント)でのベストプラクティスをカバーする。[CMM]

**能力成熟度モデル統合(CMMI)(Capability Maturity Model Integration (CMMI))**： 効率的に製品を開発・保守するためのプロセスの重要要素を記述した枠組み。製品の開発・保守における計画、製造、マネジメントでのベストプラクティスをカバーする。CMMIはCMMの正式後継版。[CMMI]

## は

**ハイパーリンク(hyperlink)**： ウェブページで、他のウェブページへ導くためのポインタ。

**ハイパーリンクツール(hyperlink tool)**： ウェブサイトに壊れたハイパーリンクがないことをチェックするためのツール。

**バグ(bug)**： defectを参照のこと。

**バグ追跡ツール(bug tracking tool)**： defect management toolを参照のこと。

**バグ分類法(bug taxonomy)**： defect taxonomyを参照のこと。

**バグレポート(bug report)**： defect reportを参照のこと。

**ハザード分析(hazard analysis)**： リスクの要素を特徴づけるために使う技法。ハザード分析の結果はシステム開発やテストのために使う方法論を牽引する。risk analysisも参照のこと。

**バージョンコントロール(version control)**： configuration controlを参照のこと。

**パス[合格](pass)**： 実際の結果が期待結果と一致した場合、テストは「合格」と見なす。

**パス(path)**： コンポーネントやシステムで、開始点から終了点へ至る一連のイベント(たとえば、実行命令文)。

**パスカバレッジ(path coverage)**： テストスイートが実行したパスのパーセンテージ。100%のパスカバレッジは100%のLCSAJカバレッジを意味する。

**パスセンシタイジング(path sensitizing)**： あるパスを実行させるため、入力値の組み合わせを選択すること。

**パステスト(path testing)**： ホワイトボックステスト設計技法の1つ。パスを実行するようテストケースを設計する。

**バックツーバックテスト(back-to-back testing)**： ふたつまたはそれ以上の異なるコンポーネントまたはシステムにおいて、同じ入力を実行し、出力を比較し、相違を調べるテスト。[IEEE610]

**バッファ(buffer)**： 一時的にデータを保管するために使うデバイスもしくはストレージの領域。データのやり取りや利用にかかわるデバイスもしくはプロセスによるデータフローの割合の違いやイベントの発生率やタイミングの違い、データ量の違いに対処することができる。[IEEE 610]

**バッファオーバーフロー(buffer overflow)**： 固定長のバッファの境界を越えてデータを保管するプロセスによる試みに起

因して、結果的に隣接するメモリ領域を上書きしたり、オーバーフロー例外が発生してしまうなどのメモリアクセスの欠陥、bufferも参照のこと。

**パーティションテスト(partition testing)**: equivalence partitioning を参照のこと。[Beizer]

**パフォーマンスインジケータ(performance indicator)**: 効果や効率を示す高レベルのメトリクス。開発を制御し、方向性を決めるために利用する。たとえば、ソフトウェア開発のためのリードタイムの遅れ。[CMMI]

**判定(decision)**: 制御フローとして、2つ以上のルートがあるプログラムポイント。分岐を切り分けるため、2つ以上のリンクを持ったノード。

**判定結果(decision outcome)**: 判定の結果(これにより、どの分岐を選択するかが決まる)。

**判定条件カバレッジ(decision condition coverage)**: テストスイートが実行した条件と判定のパーセンテージ。100%の判定条件カバレッジは、100%の条件カバレッジと100%のデジジョンカバレッジを意味する。

**判定条件テスト(decision condition testing)**: ホワイトボックステスト設計技法の1つ。条件とデジジョンカバレッジを実行するようテストケースを設計する。

**反復開発モデル(iterative development model)**: プロジェクトをいくつかの(通常は、多数の)反復部分に分けて開発するライフサイクルの一種。ひとつの反復部分は、完全な開発ループを形成しており、実行可能製品を(内部、あるいは、外部へ)リリースする。この実行可能製品は、開発している最終製品のサブセットであり、開発を反復することで最終製品に至る。

## ひ

**ピアレビュー(peer review)**: 欠陥を検出して修正するため、製品開発を担当している同僚がソフトウェア成果物をレビューすること。たとえばインスペクション、テクニカルレビュー、ウォークスルー。

**比較ツール(comparator)**: test comparator を参照のこと。

**非機能テスト(non-functional testing)**: コンポーネントやシステムで、機能に関係しない特性のテスト。たとえば、信頼性、効率性、有用性、保守性、移植性のテスト。

**非機能テスト設計技法(non-functional test design techniques)**: 非機能テストケースを設計し、選択するための手順。コンポーネントやシステムの内部構造を参照することなく、仕様の分析を基に設計する。black box test design technique も参照のこと。

**非機能要件(non-functional requirement)**: 機能以外の要求。信頼性、効率性、有用性、保守性、移植性に関する要件。

**非公式レビュー(informal review)**: 公式な(文書化した)処理手続きに従わないレビュー。

**ビジネスプロセステスト(business process-based testing)**: ビジネスプロセスの記述や知識を基にテストケースを設計するテスト手法。

**ビッグバンテスト(big-bang testing)**: 統合テストの一形式。ソフトウェアの構成要素、ハードウェアの構成要素、または両方を、段階的ではなく、一挙にコンポーネントやシステムに結合して実施する。[After IEEE610] integration testing も参照のこと。

**否定テスト(negative testing)**: コンポーネントやシステムが、正しく機能しないことを証明するためのテスト。否定テストは、特別のテストアプローチやテスト設計技法ではなく、テスト担当者の意向を反映したもの。たとえば、無効入力値や例外値を用いたテスト。[After Beizer]

**ビーバギング(bugging)**: fault seeding を参照のこと。[Abbott]

**ヒューリスティック評価 (heuristic evaluation) :** ユーザインターフェースが、一般に認められた有用性規則を遵守しているかどうかを判定するための静的な有用性テスト技法。(いわゆる、ヒューリスティクス[発見的方法])

**評価 (evaluation) :** testing を参照のこと。

**標準ソフトウェア (standard software) :** off-the-shelf software 参照のこと。

**標準テスト (standards testing) :** compliance testing 参照のこと。

**品質 (quality) :** コンポーネント、システム、プロセスが、指定された要求、ユーザ、顧客のニーズ、期待を満たす度合。[After IEEE610]

**品質コスト (cost of quality) :** 品質にかかる活動や問題のトータルコスト。予防コストと評価コスト、内部失敗コスト、外部失敗コストというように分ける場合が多い。

**品質特性 (quality attribute) :** アイテムの品質に影響を与える機能や特性。[IEEE610]

**品質特性 (quality characteristic) :** quality attribute を参照のこと。

**品質保証 (quality assurance) :** 品質管理(マネジメント)の一部。品質要件を満たしていることの自信度に焦点を当てている。[ISO9000]

**品質マネジメント (quality management) :** 品質に関して、組織を管理、制御するためのコーディネートされた活動。一般に、品質関連の管理や制御は、品質方針と品質目標、品質計画、品質制御、品質保証、品質改善の制定を伴う。[ISO9000]

## ふ

**ファンクションポイント法 (Function Point Analysis (FPA)) :** 情報処理システムの機能の大きさを測定する方法。FPAでの測定は、対象技術に依存しない。生産性測定、必要リソースの見積り、プロジェクト制御のベースとして利用できる。

**V 字モデル (V-model) :** 要求仕様からメンテナンスまでのソフトウェア開発ライフサイクル活動を記述するための枠組み。

**フィーチャ (feature) :** 要求仕様ドキュメントで、明示的、暗示的に規定したコンポーネントやシステムの特性となる機能群(たとえば、信頼性、有用性、設計上の制約など)。[After IEEE1008]

**フィールドテスト (field testing) :** beta testing を参照のこと。

**フェーズテスト計画 (phase test plan) :** 主に、ひとつのテストフェーズを扱うテスト計画。test plan も参照のこと。

**フォールト (fault) :** defect を参照のこと。

**フォールト検出率 (Fault Detection Percentage (FDP)) :** Defect Detection Percentage(DDP)を参照のこと。

**フォールト攻撃 (fault attack) :** attack を参照のこと。

**フォールトシーディング (fault seeding) :** 残存欠陥数の見積りや検出、除去の割合を監視する目的でシステムやコンポーネントに既知の欠陥を意図的に加えるプロセス。[IEEE 610]

**フォールトシーディングツール (fault seeding tool) :** システムやコンポーネントにフォールトを埋め込む(つまり、意図的に挿入する)ためのツール。

**フォールトツリー解析 (Fault Tree Analysis (FTA)) :** FTA。フォールト(欠陥)の原因分析で使用する方法。故障やヒューマンエラーと原因となる特定の顕在化したフォールトと結合できる外部イベントの論理的関係を視覚的なモデルにする技法。

**フォールトトレランス(fault tolerance)**: ソフトウェアフォールト(欠陥)や、インターフェース違反が発生した場合でも、あるレベルの性能を維持できるソフトウェア製品の能力。[ISO9126] reliability, robustness も参照のこと。

**フォールトマスキング(fault masking)**: defect masking を参照のこと。

**フォールト密度(fault density)**: defect density を参照のこと。

**複合条件(compound condition)**: 論理演算子(AND、OR、またはXOR)によってふたつまたはそれ以上の単一条件が結合されたもの。たとえば、「A>B AND C>1000」。

**複合条件(multiple condition)**: compound condition を参照のこと。

**複合条件カバレッジ(multiple condition coverage)**: テストスイートが、ひとつの命令文中で、全ての単一条件の組み合わせを実施したパーセンテージ。100%の複合条件カバレッジは、100%の条件判定カバレッジを意味する。

**複合条件テスト(multiple condition testing)**: ホワイトボックステスト設計技法の1つ。(ひとつの命令文中の)単一条件の組み合わせを実行するテストケースを設計する技法。

**複雑度(complexity)**: コンポーネントやシステム的设计・内部構造において、理解、保守、検証することが難しい度合。cyclomatic complexity も参照のこと。

**不正(anomaly)**: 要求仕様、設計ドキュメント、ユーザドキュメント、標準などから期待する事象、直感、経験から逸脱するあらゆる状態。レビュー、テスト、分析、コンパイルをする中で検出できるが、それだけのことと見えず、ソフトウェアや該当するドキュメントを利用するときに検出できることもある。[IEEE1044] bug, defect, deviation, error, fault, incident, problem も参照のこと。

**不適合(non-conformity)**: 指定の要求を満足しないこと。[ISO9000]

**ブラックボックス技法(black-box technique)**: black box test design technique を参照のこと。

**ブラックボックステスト(black-box testing)**: コンポーネントまたはシステムの内部構造の照会なしで行う、機能的でないし非機能的なテスト。

**ブラックボックステスト設計技法(black-box test design technique)**: コンポーネントやシステムの内部構造を参照することなく、機能的仕様や非機能的仕様を分析することでテストケースを設計、選択する手順。

**ブランチ(branch)**: 分岐。基本ブロックの1つ。2つ以上のプログラム・パスから1つを選べるプログラム構造をもとに、実行を選択する。たとえば、case、jump、go to、if-then-else。

**ブランチカバレッジ(branch coverage)**: テストスイートによって用いられた分岐のパーセンテージ。100%のブランチカバレッジは100%のデジジョンカバレッジと100%のステートメントカバレッジの両方を意味する。分岐網羅とも呼ぶ。

**ブランチコンディション(branch condition)**: 分岐条件。condition を参照のこと。

**ブランチコンディションカバレッジ(branch condition coverage)**: condition coverage を参照のこと。分岐条件網羅率とも呼ぶ。

**ブランチコンディション組合せカバレッジ(branch condition combination coverage)**: multiple condition coverage を参照のこと。

**ブランチコンディション組合せテスト(branch condition combination testing)**: multiple condition testing を参照のこと。

**ブランチテスト(branch testing)**: ホワイトボックステスト設計技法の1つ。分岐を実行するためのテストケースを設計する。

**振り返りミーティング (retrospective meeting) :** プロジェクトチームの間でプロジェクトを評価し、次のプロジェクトに適用することができる教訓を学ぶために行うプロジェクト終了時のミーティング。

**振る舞い (behavior) :** 入力値や事前条件の組み合わせに対する、コンポーネントやシステムの反応。

**プログラムインスツルメンタ (program instrumenter) :** instrumenter を参照のこと。

**プログラムテスト (program testing) :** component testing を参照のこと。

**プロジェクト (project) :** 制御の対象となる作業のユニークな組み合わせ。時間、コスト、リソースの制約を含む要求事項に準拠して、目標を完成させるための作業であり、開始日付と終了日付がある。[ISO9000]

**プロジェクトテスト計画 (project test plan) :** master test plan を参照のこと。

**プロジェクトリスク (project risk) :** (テスト)プロジェクトの管理(マネジメント)と制御(コントロール)に関するリスク。たとえば、スタッフの不足、厳しい締め切り、変更管理などがこれに当たる。risk も参照のこと。

**プロセス (process) :** 相互関係のある動作の組み合わせ。入力を出力に変換する。[ISO12207]

**プロセス改善 (process improvement) :** 組織のプロセスの成熟度とパフォーマンスを改善するために設計した活動プログラムとその結果。[CMMI]

**プロセスサイクルテスト (process cycle test) :** ブラックボックステスト設計技法の1つ。ビジネスの処理手続きや手順を実行するためのテストケースを設計する。[TMap] procedure testing も参照のこと。

**プロダクトリスク (product risk) :** テスト対象に直接関係するリスク。risk も参照のこと。

**ブロックドテストケース (blocked test case) :** 事前条件を満足できないため、実行不能のテストケース。

**プローブ効果 (probe effect) :** 性能テストツールやモニタなどでコンポーネントやシステムを測定する場合、測定ツールによって埋め込まれる計測のためのコード(インスツルメント)がコンポーネントやシステムに及ぼす影響。たとえば、性能テストツールを使うことによって、パフォーマンスは若干悪化する。

へ

**ペアテスト (pair testing) :** ふたり(たとえば、テスト担当者2名、開発担当者とテスト担当者が1名ずつ、エンドユーザとテスト担当者が1名ずつ)が、共同で欠陥を見つけること。テスト期間中、ふたりで1台のコンピュータを共有し、交互に使用することが多い。

**ペアプログラミング (pair programming) :** ソフトウェア開発のアプローチの1つ。1台のコンピュータを共有するふたりのプログラマが、(製品用、テスト用の)コードを書く。これにより、コードレビューをリアルタイムに実施できることを期待している。

**ペアワイズテスト (pairwise testing) :** 入力パラメータの各ペアを、設定可能な個々の組み合わせの全てで実行するためのテストケースを設計するブラックボックスのテスト設計技法。orthogonal array testing も参照のこと。

**ベーシックブロック (basic block) :** 分岐を含まない1つ以上の連続した実行命令文。制御フローグラフにおける接点がベーシックブロックを表す。

**ベーステストセット (basis test set) :** コンポーネントの内部構造や仕様から設計した一連のテストケース。目標とする網羅基準の100%カバーを目的とする。

**ベストプラクティス (best practice) :** 与えられた状況下で、組織の作業効率や品質改善に寄与する優れた方法や革新的な実践法。同業他社から、「最善」と認められることが多い。

**ベースライン(baseline)**： 公式のレビュー、承認を受けた仕様やソフトウェア製品。将来の開発の基礎となることを意図しており、変更は公式の管理プロセスによってのみ可能。[After IEEE610]

**ベータテスト(beta testing)**： 製品開発環境以外の外部環境で、現在、あるいは、将来のユーザや顧客が実施するテスト。コンポーネントやシステムが、ユーザや顧客のニーズを満たし、ビジネスプロセスに適合するかを判定する。マーケットからのフィードバックを得るため、既成ソフトウェアの外部受け入れテストの一形式として用いられることが多い。

**変異解析(mutation analysis)**： テストスイートの完全性を判定する方法の1つ。どの程度のプログラムの変形(変異)をテストスイートが識別できるかを測定する。

**ミューテーションテスト(mutation testing)**： back-to-back testing を参照のこと。

**変換テスト(conversion testing)**： 現行システムから別システム用にデータ変換するソフトウェアに対するテスト。

**変更コントロール(change control)**： configuration control を参照のこと。

**変更コントロール委員会(change control board)**： configuration control board を参照のこと。

**変更条件判定カバレッジ(MC/DC)(modified condition decision coverage)**： condition determination coverage を参照のこと。

**変更条件判定テスト(modified condition decision testing)**： condition determination coverage を参照のこと。

**変更性(changeability)**： 条件指定された改修が履行されることを可能にするためのソフトウェア製品の能力。[ISO9126] maintainability も参照のこと。

**変更複合条件テスト(modified multiple condition testing)**： condition determination testing を参照のこと。

**変更複合条件カバレッジ(modified multiple condition coverage)**： condition determination coverage を参照のこと。

**変数(variable)**： ソフトウェアプログラムから名前を参照することでアクセスできる。コンピュータ中のストレージの要素。

**ベンチマークテスト(benchmark test)**：

- (1) 測定や比較のベースとなる標準
- (2) コンポーネントやシステム同士の比較、または、(1)の標準と比較するためのテスト。[After IEEE610]

## ほ

**ポインタ(pointer)**： 他のデータアイテムのロケーションを指定するデータアイテム。たとえば、処理される次の従業員レコードアドレスを指定したデータアイテム [IEEE 610]

**法令遵守(compliance)**： 法律や類似規則の標準、協定、規定を遵守するソフトウェア製品の能力。[ISO9126]

**法令遵守テスト(compliance testing)**： コンポーネントやシステムの法令遵守を判定するテストのプロセス。

**保守性(maintainability)**： ソフトウェア製品の変更の容易さ。変更には、欠陥の修正、新しい要求を支援するための変更、将来の保守性を上げるための変更、移動環境の変更に対応するソフトウェア製品の変更などがある。[ISO9126]

**保守性テスト(maintainability testing)**： ソフトウェア製品の保守性を判定するテスト手順

**保守性テスト(serviceability testing)**： maintainability 参照のこと。

**保守テスト(maintenance testing)**: 運用システム自体の変更や、稼働環境の変更が運用システムに与える影響をテストすること。

**ボトムアップテスト(bottom-up testing)**: 段階的に統合テストを進める手法の1つ。最下位レベルのコンポーネントを最初にテストし、そのコンポーネントを利用して上位コンポーネントをテストする方式。最上位コンポーネントをテストするまで、この手順を繰り返す。integration testing も参照のこと。

**ポリュームテスト(volume testing)**: システムに大量のデータをかけるテスト。resource-utilization testing 参照のこと。

**ホワイトボックス技法(white-box techniques)**: white-box test design techniques を参照のこと。

**ホワイトボックステスト(white-box testing)**: コンポーネントまたはシステムの内部構造の分析を基にしたテスト。

**ホワイトボックステスト設計技法(white-box test design technique)**: コンポーネントやシステムの内部構造の分析を基にテストケースを設計、選択する手順

## ま

**マイルストーン(milestone)**: 指定した(中間)提供品や成果物を提供するプロジェクト日程上の1つの時点。

**マスターテスト計画(master test plan)**: 通常、複数のテストレベルを扱うテスト計画。test plan も参照のこと。

**マネジメントレビュー(management review)**: ソフトウェアの購入、提供、開発、運用、保守を体系的に評価すること。管理者や管理者の代理者が、進捗を監視し、計画やスケジュールの状態を判定し、要求やシステム割り当てを確認し、目的遂行用に最適化されるよう管理アプローチの効率を評価する。[After IEEE610, IEEE1028]

## み

**魅力性(attractiveness)**: ユーザに対して魅力的であるためのソフトウェア製品の能力。[ISO9126] usability を参照のこと。

## め

**メトリクス(metric)**: 測定尺度、および、測定方法。[ISO14598]

**メモリーリーク(memory leak)**: プログラムの動的記憶装置割り当てロジックの中の欠陥。この欠陥により、メモリ使用後の再要求失敗が発生し、将来、メモリ不足によりプログラムが動かなくなる。

**メンテナンス(maintenance)**: リリース後のソフトウェア製品を変更すること。欠陥の修正、性能や他の特性の改善、別環境へ製品適合などを目的とする。[IEEE1219]

## も

**モジュール(module)**: component を参照のこと。

**モジュールテスト(module testing)**: component testing を参照のこと。

**モデリングツール(modelling tool)**: ソフトウェアまたはシステムの妥当性確認をサポートするツール。[Graham]。

**モデレータ(moderator)**: インспекションや、他のレビュー手順に責任を持つ、リーダーや中心人物。

**モニタ (monitor) :** テスト時に、コンポーネントやシステムと同時に走り、コンポーネントやシステムの動作を監視、記録、分析するソフトウェアツールやハードウェア装置。[After IEEE610]

**モニタリングツール (monitoring tool) :** monitor を参照のこと。

**モンキーテスト (monkey testing) :** 製品の用法についてはまったく考慮せず、広範囲の入力からランダムに選択し、ボタンをランダムに押すことでテストを行う方法。

**問題 (problem) :** defect を参照のこと。

**問題マネジメント (problem management) :** defect management を参照のこと。

**問題レポート (problem report) :** defect report を参照のこと。

## ゆ

**有限状態機械 (finite state machine) :** 計算モデルの1つ。有限個数の状態と、有限個数の状態間遷移から構成される。状態遷移に対応する処理も記述できる。[IEEE610]

**有限状態テスト (finite state testing) :** state transition testing を参照のこと。

**優先度 (priority) :** あるアイテム(たとえば、欠陥)に割り振った(ビジネス上の)重要さのレベル。

**ユーザ受け入れテスト (user acceptance testing) :** acceptance testing 参照のこと。

**ユーザシナリオテスト (user scenario testing) :** use case testing 参照のこと。

**ユーザテスト (user test) :** コンポーネントまたはシステムの有用性を評価するために現実のユーザが参加するテスト。

**ユーザビリティ (usability) :** ソフトウェア製品が、指定された条件下で理解され、学びやすく、使用しやすく、ユーザにとって魅力的である能力。[ISO9126]

**ユーザビリティテスト (usability testing) :** ソフトウェア製品が、指定された条件下で理解され、学びやすく、使用しやすく、ユーザにとって魅力的である能力を判定するためのテスト。[After ISO9126]

**ユースケース (use case) :** ユーザとシステム間の対話における一連のトランザクション。視覚できる結果を伴う。

**ユースケーステスト (use case testing) :** ユーザシナリオを実行するテストケースを設計する。ブラックボックステストの設計技法。

**ユニット (unit) :** component 参照のこと。

**ユニットテスト (unit testing) :** component testing 参照のこと。

**ユニットテストフレームワーク (unit test framework) :** 適切なスタブやドライバを併用した状態または独立した状態でコンポーネントをテストできるユニットテストまたはコンポーネントテスト用の環境を提供するツール。デバッグ機能など、開発者をサポートするその他の機能もある。[Graham]

**ユニットテストフレームワークツール (unit test framework tool) :** unit test framework 参照のこと。

## よ

**要件 (requirement) :** ユーザが問題解決や目的を達成するために必要な条件や能力。契約、標準、仕様、その他の公式ドキュメントを満足するために、システムやシステムコンポーネントが満たし、保持すべき条件や能力。[After IEEE610]

**要件フェーズ (requirements phase) :** ソフトウェアライフサイクルの一時期。ソフトウェア製品の要件を定義、記録する。[IEEE610]

**要件ベーステスト (requirements-based testing) :** テストのアプローチで、要件から導き出したテスト目的とテスト条件を基にしてテストケースを設計する。たとえば、指定された機能を実行するテスト、または、信頼性や有用性のような非機能特性を調べるテスト。

**要件マネジメントツール (requirements management tool) :** 要件、要件特性(たとえば、優先順位、ベースとなる情報)、注釈を記録し、要件の階層をたどる追跡や、要件変更管理を容易にするツール。要件管理ツールの中には、あらかじめ定義した要件規約をもとに、整合性や違反チェックのような、静的解析をするものもある。

**予測結果 (predicted outcome) :** expected result を参照のこと。

**予備テスト (pretest) :** intake test を参照のこと。

## ら

**ランダムテスト (random testing) :** ブラックボックステスト設計技法の1つ。擬似乱数生成アルゴリズム等を使い、運用プロフィールに合致するテストケースを設計する。信頼性、性能等、機能外特性のテストに利用できる。

## り

**理解性 (understandability) :** ソフトウェアが適したものか、どうすれば特定のタスクや条件で使えるかをユーザが理解できるソフトウェア製品の能力。[ISO9126] usability も参照のこと。

**リスク (risk) :** 将来、否定的な結果を生む要素。通常、影響度と発生可能性として表現する。

**リスク軽減 (risk mitigation) :** risk control 参照のこと。

**リスクコントロール (risk control) :** 指定レベルまでリスクを減らす(あるいは、リスクレベルを維持する)ために、判定を下したり、対策したりする手順

**リスク識別 (risk identification) :**ブレインストーミング、チェックリスト、故障履歴などを使ったリスクを識別する手順

**リスク分析 (risk analysis) :** 影響度と発生確率(可能性)を見積るために、定義したリスクを評価する手順

**リスクタイプ (risk type) :** リスクを軽減(制御)できるテストタイプと関連したリスクのカテゴリ。たとえば、誤断から来る利用者との相互作用に関するリスクは、ユーザビリティテストングにて軽減できる。

**リスクベーステスト (risk-based testing) :** プロジェクトの初期段階からプロダクトリスクのレベルを低下させ、当事者にその状態を通知するテストの方法。プロダクトリスクの識別の他、テストプロセスをガイドする際のリスク活用もこれに含まれる。

**リスクマネジメント (risk management) :** リスクの識別、分析、優先順位付け、制御のタスクや実施方法を体系的に適用すること。

**リスクレベル(risk level)**: 影響度合いと発生頻度からみた特徴で定義したリスクの重要性。リスクのレベルはテストを行う時の強弱を決定するときに使われる。リスクレベルはまた、定性的(高/中/低など)または定量的に表現できる。

**リリースノート(release note)**: テストケース、その構成、現在の状態、その他のリリース情報を記述したドキュメント。テスト実行フェーズの開始前に、開発側からテスト関係者へ、場合によっては他の利害関係者に提供する。[After IEEE829]

**リンクテスト(link testing)**: component integration testing を参照のこと。

## れ

**例外処理(exception handling)**: 人間、他コンポーネント、システムからの誤入力、内部の故障に対するコンポーネントやシステムの動作。

**レビュー(review)**: 製品やプロジェクトの状態を評価する手法。計画した結果との違いを分析し、改善を提案する。例として、マネジメントレビュー、非公式レビュー、テクニカルレビュー、インスペクション、ウォークスルーがある。[After IEEE1028]

**レビューア(reviewer)**: レビューの中でレビュー対象のプロダクトやプロジェクトの不整合を特定し、指摘する個人。レビューアは、様々な視点やレビュープロセスでの役割を担った人達を選ばれる。

**レビューツール(review tool)**: レビュー手順を支援するツール。レビューの計画、追跡の支援、コミュニケーションのサポート、協働的なレビューの推進、収集や報告用の外リクスの保存する機能を備えていることが多い。

**レベルテスト計画(level test plan)**: 通常、ひとつのテストレベルを扱うテスト計画。test plan も参照のこと。

**連続表現(continuous representation)**: 特定のプロセスエリア群内でのプロセス改善アプローチのための推奨順番として提供された、能力レベルを見る成熟度モデル構造。[CMMI]

## ろ

**ロードテスト(load testing)**: コンポーネントやシステムの動作を測定するテストの一種。負荷(たとえば、並列実行ユーザ数やトランザクションの数)を増加させ、コンポーネントやシステムがどの程度の負荷に耐えられるか判定する。stress testing も参照のこと。

**ロードテストツール(load testing tool)**: 「テスト技術者資格制度 Foundation シラバス(日本語版)」を参照のこと。

**ロードプロファイル(load profile)**: テストされるコンポーネントやシステムが稼働中にユーザが行うであろう活動内容を記した仕様書。ロードプロファイルは、決められた時間内に所定の運用プロファイルに準じ決められたトランザクションを処理する仮想ユーザを複数指定して構成する。operational profile も参照のこと。

**ロバストネステスト(robustness testing)**: ソフトウェア製品の頑健性を判定するテスト。

**論理駆動テスト(logic-driven testing)**: white box testing を参照のこと。

**論理テストケース(logical test case)**: high level test case を参照のこと。

**論理網羅テスト(logic-coverage testing)**: white box testing を参照のこと。[Myers]

## わ

**ワイドバンドデルファイ(Wide Band Delphi):** 専門家によるテスト見積り技法。チームメンバーから集めた知識を用い、正確な見積りをするもの。

**ワイルドポインタ(wild pointer):** 範囲外のロケーションを参照しているポインタ、もしくは存在しないロケーションを参照しているポインタ。pointerも参照のこと

## 付録 A(参考資料)

出典索引: 本用語集では、以下の非標準ソースを使用した。

[Abbott] J. Abbot (1986), Software Testing Techniques, NCC Publications.

[Adrion] W. Adrion, M. Branstad and J. Chemiabsky (1982), Validation, Verification and Testing of Computer Software, in: Computing Surveys, Vol. 14, No 2, June 1982.

[Bach] J. Bach (2004), Exploratory Testing, in: E. van Veenendaal, The Testing Practitioner – 2nd edition, UTN Publishing, ISBN 90-72194-65-9.

[Beizer] B. Beizer (1990), Software Testing Techniques, van Nostrand Reinhold, ISBN 0-44220672-0

[Chow] T. Chow (1978), Testing Software Design Modelled by Finite-State Machines, in: IEEE Transactions on Software Engineering, Vol. 4, No 3, May 1978.

[CMM] M. Paulk, C. Weber, B. Curtis and M.B. Chrissis (1995), The Capability Maturity Model, Guidelines for Improving the Software Process, Addison-Wesley, ISBN 0-201-54664-7

[CMMI] M.B. Chrissis, M. Konrad and S. Shrum (2004), CMMI, Guidelines for Process Integration and Product Improvement, Addison Wesley, ISBN 0-321-15496-7

[Fenton] N. Fenton (1991), Software Metrics: a Rigorous Approach, Chapman & Hall, ISBN 0-53249-425-1

[Fewster and Graham] M. Fewster and D. Graham (1999), Software Test Automation, Effective use of test execution tools, Addison-Wesley, ISBN 0-201-33140-3.

[Freedman and Weinberg] D. Freedman and G. Weinberg (1990), Walkthroughs, Inspections, and Technical Reviews, Dorset House Publishing, ISBN 0-932633-19-6.

[Gerrard] P. Gerrard and N. Thompson (2002), Risk-Based E-Business Testing, Artech House Publishers, ISBN 1-58053-314-0.

[Gilb and Graham] T. Gilb and D. Graham (1993), Software Inspection, Addison-Wesley, ISBN 0-201-63181-4.

[Grochtmann] M. Grochtmann (1994), Test Case Design Using Classification Trees, in: Conference Proceedings STAR 1994.

[Hetzel] W. Hetzel (1988), The complete guide to software testing – 2nd edition, QED Information Sciences, ISBN 0-89435-242-3.

[McCabe] T. McCabe (1976), A complexity measure, in: IEEE Transactions on Software Engineering, Vol. 2, pp. 308-320.

[Musa] J. Musa (1998), Software Reliability Engineering Testing, McGraw-Hill Education, ISBN 0-07913-271-5.

[Myers] G. Myers (1979), The Art of Software Testing, Wiley, ISBN 0-471-04328-1.

[TMap] M. Pol, R. Teunissen, E. van Veenendaal (2002), Software Testing, A guide to the TMap Approach, Addison Wesley, ISBN 0-201-745712.

[Veenendaal] E. van Veenendaal (2004), The Testing Practitioner – 2nd edition, UTN Publishing, ISBN 90-72194-65-9.

## 付録 B (用語集へのコメント方法)

テストのコミュニティのニーズに答えられるよう用語集を更に良くしていくため、このドキュメントに対するコメントを受け付ける。

コメントをする場合、以下の情報を含めるようにしてもらいたい。

名前とコンタクトの詳細

用語集のバージョン(現在は2.0)

用語集の該当箇所

補足情報(変更依頼理由、用語集の使用例)

コメントの受付方法は以下の方法がある。

1. Eメールの場合の連絡先 [eve@improveqs.nl](mailto:eve@improveqs.nl);
2. 郵便の場合の送付先 Improve Quality Services BV, attn. Mr. E. van Veenendaal, Waalreseweg 39, 5554 HA, Valkenswaard, The Netherlands;
3. FAX の場合の連絡先 +31 40 20 21450, marked for the attention of Mr. E. van Veenendaal.

## 索引 (英語)

---

### A

abstract test case .....	26
acceptance .....	12
acceptance criteria .....	12
acceptance testing .....	12
accessibility testing .....	10
accuracy .....	23
action word driven testing .....	10
actual outcome .....	20
actual result .....	20
ad hoc review .....	10
ad hoc testing .....	10
adaptability .....	14
agile testing .....	10
algorithm test [TMap] .....	10
alpha testing .....	10
analyzability .....	13
analyzer .....	10
anomaly .....	37
arc testing .....	10
attack .....	17
attractiveness .....	40
audit .....	14
audit trail .....	14
automated testware .....	20
availability .....	14

---

### B

back-to-back testing .....	34
baseline .....	39
basic block .....	38
basis test set .....	38
debugging .....	35
behavior .....	38
benchmark test .....	39

bespoke software .....	26
best practice .....	38
beta testing .....	39
big-bang testing .....	35
black-box technique .....	37
black-box test design technique .....	37
black-box testing .....	37
blocked test case .....	38
bottom-up testing .....	40
boundary value .....	15
boundary value analysis .....	15
boundary value coverage .....	15
boundary value testing .....	15
branch .....	37
branch condition .....	37
branch condition combination coverage .....	37
branch condition combination testing .....	37
branch condition coverage .....	37
branch coverage .....	37
branch testing .....	37
buffer .....	34
buffer overflow .....	34
bug .....	34
bug report .....	34
bug taxonomy .....	34
bug tracking tool .....	34
business process-based testing .....	35

---

### C

Capability Maturity Model (CMM) .....	34
Capability Maturity Model Integration (CMMI) .....	34
capture/playback tool .....	15
capture/replay tool .....	15
CASE .....	16
CAST .....	15
cause-effect analysis .....	17
cause-effect decision table .....	17

cause-effect graph	17
cause-effect graphing	17
certification	21
change control	39
change control board	39
changeability	39
checker	25
Chow's coverage metrics	26
classification tree	16
classification tree method	16
code	18
code analyzer	18
code coverage	18
code-based testing	18
co-existence	15
commercial off-the-shelf software	20
comparator	35
compatibility testing	18
compiler	18
complete testing	14
completion criteria	20
complexity	37
compliance	39
compliance testing	39
component	19
component integration testing	19
component specification	19
component testing	19
compound condition	37
concrete test case	16
concurrency testing	33
condition	21
condition combination coverage	21
condition combination testing	21
condition coverage	21
condition determination coverage	21
condition determination testing	21
condition outcome	21
condition testing	21
confidence test	22

configuration	17
configuration auditing	17
configuration control	17
configuration control board (CCB)	17
configuration identification	17
configuration item	17
configuration management	17
Configuration management tool	17
configuration testing	18
confirmation testing	14
conformance testing	26
consistency	11
continuous representation	43
control flow	23
control flow analysis	23
control flow graph	23
control flow path	23
conversion testing	39
cost of quality	36
COTS	18
coverage	14
coverage analysis	14
coverage item	14
coverage measurement tool	14
coverage tool	14
custom software	14
cyclomatic complexity	19
cyclomatic number	19

---

## *D*

daily build	26
data definition	32
data driven testing	32
data flow	32
data flow analysis	32
data flow coverage	32
data flow testing	32
data integrity testing	32
database integrity testing	32
dead code	32

debugger .....	32
debugging .....	32
debugging tool .....	32
decision .....	35
decision condition coverage .....	35
decision condition testing .....	35
decision coverage .....	26
decision outcome .....	35
decision table .....	26
decision table testing .....	26
decision testing .....	32
defect .....	16
defect based technique .....	16
defect based test design technique .....	16
defect density .....	16
Defect Detection Percentage (DDP) .....	16
defect management .....	16
defect management tool .....	16
defect masking .....	32
defect report .....	17
defect taxonomy .....	16
defect tracking tool .....	16
definition–use pair .....	26
deliverable .....	26
design–based testing .....	26
desk checking .....	14
development testing .....	13
deviation .....	26
deviation report .....	26
dirty testing .....	25
documentation testing .....	33
domain .....	33
driver .....	33
dynamic analysis .....	33
dynamic analysis tool .....	33
dynamic comparison .....	33
dynamic testing .....	33

---

*E*

efficiency .....	18
------------------	----

efficiency testing .....	18
elementary comparison testing .....	15
emulator .....	12
entry criteria .....	13
entry point .....	13
equivalence class .....	33
equivalence partition .....	33
equivalence partition coverage .....	33
equivalence partitioning .....	33
error .....	12
error guessing .....	13
error seeding .....	13
error seeding tool .....	13
error tolerance .....	13
evaluation .....	36
exception handling .....	43
executable statement .....	20
exercised .....	22
exhaustive testing .....	24
exit criteria .....	20
exit point .....	20
expected outcome .....	14
expected result .....	14
experienced–based technique .....	16
experienced–based test design technique .....	16
exploratory testing .....	25

---

*F*

fail .....	20
failure .....	18
failure mode .....	18
Failure Mode and Effect Analysis (FMEA) .....	18
Failure Mode, EFFECT and Criticality Analysis (FMECA) .....	18
failure rate .....	18
FALSE–fail result .....	10
FALSE–negative result .....	14
FALSE–pass result .....	10
FALSE–positive result .....	15
fault .....	36
fault attack .....	36

fault density .....	37
Fault Detection Percentage (FDP).....	36
fault masking.....	37
fault seeding.....	36
fault seeding tool.....	36
fault tolerance.....	37
Fault Tree Analysis (FTA).....	36
feasible path.....	20
feature.....	36
field testing.....	36
finite state machine.....	41
finite state testing.....	41
formal review.....	17
frozen test basis.....	32
Function Point Analysis (FPA).....	36
functional integration.....	15
functional requirement.....	15
functional test design technique.....	15
functional testing.....	15
functionality.....	15
functionality testing.....	15

---

## *G*

glass box testing.....	16
------------------------	----

---

## *H*

hazard analysis.....	34
heuristic evaluation.....	35
high level test case.....	17
horizontal traceability.....	22
hyperlink.....	34
hyperlink tool.....	34

---

## *I*

impact analysis.....	12
incident.....	11
Incident logging.....	11
incident management.....	11

incident management tool.....	11
incident report.....	11
incremental development model.....	11
incremental testing.....	11
independence of testing.....	33
infeasible path.....	20
informal review.....	35
input.....	33
input domain.....	34
input value.....	33
inspection.....	11
inspection leader.....	11
inspector.....	12
installability.....	11
installability testing.....	11
installation guide.....	11
installation wizard.....	11
instrumentation.....	11
instrumenter.....	11
intake test.....	12
integration.....	32
integration testing.....	33
integration testing in the large.....	25
integration testing in the small.....	21
interface testing.....	12
interoperability.....	24
interoperability testing.....	24
invalid testing.....	12
isolation testing.....	10
item transmittal report.....	10
iterative development model.....	35

---

## *K*

key performance indicator.....	15
keyword driven testing.....	15

---

## *L*

LCSAJ.....	13
LCSAJ coverage.....	13

LCSAJ testing.....	13
learnability.....	20
level test plan.....	43
link testing.....	43
load profile.....	43
load testing.....	43
load testing tool.....	43
logical test case.....	43
logic-coverage testing.....	43
logic-driven testing.....	43
low level test case.....	26

---

## *M*

maintainability.....	39
maintainability testing.....	39
maintenance.....	40
maintenance testing.....	39
management review.....	40
master test plan.....	40
maturity.....	23
measure.....	24
measurement.....	24
measurement scale.....	24
memory leak.....	40
metric.....	40
migration testing.....	10
milestone.....	40
mistake.....	10
modelling tool.....	40
moderator.....	40
modified condition decision coverage.....	39
modified condition decision testing.....	39
modified multiple condition coverage.....	39
modified multiple condition testing.....	39
module.....	40
module testing.....	40
monitor.....	41
monitoring tool.....	41
monkey testing.....	41
multiple condition.....	37

multiple condition coverage.....	37
multiple condition testing.....	37
mutation analysis.....	39
mutation testing.....	39

---

## *N*

negative testing.....	35
non-conformity.....	37
non-functional requirement.....	35
non-functional test design techniques.....	35
non-functional testing.....	35
N-switch coverage.....	12
N-switch testing.....	12

---

## *O*

off-the-shelf software.....	14
operability.....	24
operational acceptance testing.....	12
operational environment.....	24
operational profile.....	12
operational profile testing.....	12
operational testing.....	12
oracle.....	13
orthogonal array.....	26
orthogonal array testing.....	26
outcome.....	16
output.....	21
output domain.....	21
output value.....	21

---

## *P*

pair programming.....	38
pair testing.....	38
pairwise testing.....	38
partition testing.....	35
pass.....	34
pass/fail criteria.....	17
path.....	34

path coverage .....	34
path sensitizing .....	34
path testing.....	34
peer review .....	35
performance.....	23
performance indicator.....	35
performance profiling.....	24
performance testing.....	23
performance testing tool.....	23
phase test plan.....	36
pointer .....	39
portability.....	11
portability testing .....	11
postcondition .....	20
post-execution comparison.....	20
precondition .....	20
predicted outcome.....	42
pretest.....	42
priority.....	41
probe effect .....	38
problem.....	41
problem management.....	41
problem report.....	41
procedure testing.....	32
process.....	38
process cycle test.....	38
process improvement.....	38
product risk.....	38
production acceptance testing.....	24
program instrumenter .....	38
program testing.....	38
project.....	38
project risk.....	38
project test plan.....	38
pseudo-random .....	14

---

## Q

qualification.....	19
quality.....	36
quality assurance.....	36

quality attribute.....	36
quality characteristic .....	36
quality management .....	36

---

## R

random testing.....	42
record/playback tool.....	15
recorder .....	15
recoverability.....	13
recoverability testing .....	13
recovery testing .....	13
regression testing.....	13
regulation testing.....	15
release note .....	43
reliability .....	22
reliability growth model .....	22
reliability testing .....	22
replaceability .....	25
requirement.....	42
requirements management tool.....	42
requirements phase .....	42
requirements-based testing .....	42
resource utilization .....	19
resource utilization testing.....	19
result .....	16
resumption criteria.....	19
re-testing.....	19
retrospective meeting .....	37
review .....	43
review tool .....	43
reviewer .....	43
risk .....	42
risk analysis.....	42
risk control.....	42
risk identification .....	42
risk level.....	43
risk management.....	42
risk mitigation.....	42
risk type.....	42
risk-based testing.....	42

robustness .....	14
robustness testing.....	43
root cause.....	19
root cause analysis.....	19

---

## *S*

safety .....	10
safety critical system .....	24
safety testing.....	10
sanity test.....	19
scalability.....	13
scalability testing.....	14
scenario testing.....	20
scribe.....	21
scripted testing .....	22
scripting language.....	22
security.....	24
security testing .....	24
security testing tool.....	24
security tool.....	24
serviceability testing.....	39
severity.....	21
simulation.....	20
simulator.....	20
site acceptance testing.....	19
smoke test.....	23
software.....	24
software attack .....	25
software Failure Mode and EFFECT Analysis (SFMEA).....	24
software Failure Mode EFFECT, and Criticality Analysis (SFMECA).....	25
software Fault Tree analysis (SFTA) .....	25
software feature.....	25
software life cycle .....	25
software product characteristic .....	25
software quality.....	25
software quality characteristic .....	25
software test incident.....	25
software test incident report.....	25
Software Usability Measurement Inventory (SUMI).....	25

source statement .....	24
specification.....	21
specification-based technique.....	21
specification-based test design technique .....	21
specification-based testing.....	21
specified input.....	23
stability .....	10
staged representation .....	25
standard software.....	36
standards testing.....	36
state diagram .....	21
state table.....	21
state transition.....	21
state transition testing.....	21
statement.....	22
statement coverage.....	22
statement testing .....	22
static analysis .....	23
static analysis tool.....	23
static analyzer.....	23
static code analysis .....	23
static code analyzer.....	23
static testing.....	23
statistical testing.....	32
status accounting .....	22
storage.....	22
storage testing .....	22
stress testing .....	22
stress testing tool .....	23
structural coverage.....	18
structural test design technique.....	18
structural testing.....	18
structure based testing.....	18
structure-based technique.....	18
structured walkthrough .....	18
stub .....	22
subpath.....	19
suitability.....	18
suspension criteria.....	26
syntax testing.....	22

system.....	20
system integration testing.....	20
system of systems .....	20
system testing.....	20

---

## *T*

technical review .....	26
test.....	27
test approach.....	27
test automation.....	28
test basis.....	31
test bed.....	31
test case.....	27
test case design technique .....	28
test case specification.....	27
test case suite .....	28
test charter.....	30
test closure.....	29
test comparator .....	30
test comparison .....	28
test completion criteria.....	27
test condition.....	29
test control.....	28
test coverage.....	27
test cycle.....	28
test data.....	30
test data preparation tool .....	30
test design.....	29
test design specification .....	29
test design technique .....	29
test design tool .....	29
test driven development .....	27
test driver.....	30
test environment.....	27
test estimation.....	31
test evaluation report.....	30
test execution.....	28
test execution automation .....	28
test execution phase .....	28
test execution schedule.....	28
test execution technique.....	28
test execution tool .....	28
test fail .....	28
test generator .....	30
test harness .....	30
test implementation .....	28
test incident.....	27
test incident report.....	27
test infrastructure .....	27
test input .....	30
test item .....	27
test item transmittal report.....	27
test leader.....	31
test level.....	31
test log .....	28
test logging.....	28
test management.....	31
test management tool .....	31
test manager .....	31
Test Maturity Model (TMM).....	29
test Maturity model Integrated (TMMi).....	29
test monitoring.....	31
test object .....	30
test objective .....	30
test oracle.....	27
test outcome.....	28
test pass.....	30
test performance indicator.....	30
test phase.....	30
test plan.....	27
test planning.....	27
Test Point Analysis (TPA).....	31
test policy.....	31
test procedure.....	30
test procedure specification.....	30
test process.....	28
Test Process Improvement (TPI).....	30
test progress report.....	29
test record.....	31
test recording.....	31

test report .....	32
test reproducibility .....	28
test requirement.....	31
test result.....	28
test rig.....	31
test run.....	31
test run log .....	31
test scenario.....	29
test schedule .....	29
test script.....	29
test session .....	29
test set.....	29
test situation.....	28
test specification.....	29
test specification technique .....	29
test stage .....	29
test strategy.....	29
test suite .....	29
test summary report.....	28
test target.....	30
test technique.....	27
test tool.....	30
test type.....	30
testability.....	31
testability review .....	31
testable requirements.....	31
tester.....	30
testing .....	27
testware.....	27
thread testing.....	23
time behavior .....	25
top-down testing.....	33
traceability.....	33

---

## U

understandability .....	42
unit.....	41
unit test framework .....	41
unit test framework tool.....	41
unit testing.....	41
unreachable code.....	33
usability.....	41
usability testing .....	41
use case .....	41
use case testing.....	41
user acceptance testing.....	41
user scenario testing .....	41
user test .....	41

---

## V

validation.....	25
variable .....	39
verification .....	17
version control .....	34
vertical traceability.....	22
V-model.....	36
volume testing.....	40

---

## W

walkthrough.....	12
white-box techniques .....	40
white-box test design technique.....	40
white-box testing.....	40
Wide Band Delphi.....	44
wild pointer.....	44