用 語 集

キーワード		意　味
ACM		the Association for Computing Machinery の頭文字を取ったもの。アメリカに本拠を置く世界最古・最大のコンピュータ関係の協会／学会。この団体の機関誌がCACM（Communication of ACM）。
CMMI-DEV		「開発のためのCMMI」のこと。「開発のためのCMM」を参照のこと。
COCOMO		バリー・ベーム（Barry W. Boehm）が作成した、ソフトウェア開発のためのコストモデル。Constructive Cost Model の省略形。
CRUD図		オンライン・システムで、データベースのレコードが入力されるどのトランザクションによって生成（C)、読み込み（R)、更新（U)、消去（D)の処理がなされるかをまとめた表。縦軸にデータベースのレコードを、横軸にトランザクションを記載することが多い。データベースの更新を含むコンピュータ処理のまとめとして使われると共に、チェックにも使用される。
DFD		「データフロー・ダイアグラム（データフロー図)」のこと。「データフロー・ダイアグラム」を参照のこと。
ERD		「実体関連図」のこと。「実体関連図」を参照のこと。
ER図		「実体関連図」のこと。「実体関連図」を参照のこと。
EVM		「アーンド・バリュー・マネジメント」のこと。「アーンド・バリュー・マネジメント」を参照のこと。
GNU		フリー・ソフトウェアを開発するために、1984年にリチャード・ストールマンが立ち上げたプロジェクト。これまで多くのソフトウェアを開発し、大きな成果を上げてきた。”GNU is Not UNIX”の頭文字を取って名付けられた。
GOTOステートメント有害論		1968年に、当時オランダの大学に勤めていたダイカストラ教授がCACMの編集長に送った手紙に記述されていた内容。プログラムの中で「GOTOステートメント」を使うとプログラムが読みにくくなって良くない、と言う趣旨。この手紙がCACMに掲載されて世界中に一大反響を巻き起こし、結果的に構造化プログラミングを作るきっかけになった。
GPL		「ソフトウェアは自由であるべき」とするリチャード・ストールマンの「コピーレフト」の考え方を表した、フリー・ソフトウェアのライセンスの代表的なもの。1989年に最初の版が公表され、現在は第3版になっている。
GQMパラダイム		ソフトウェア・メトリクスで、何を計測の対象にするかを決めるための1つの提案。米国メリーランド大学のヴィクター・バシリ教授が提案したもの。GQMは目標（Goal）／質問（Question）／測定値（Metrics）のそれぞれの英文の頭文字を取ったもの。

IEC		International Electrotechnical Commission の頭文字を取ったもの。日本語では「国際電気標準会議」と訳されている。電気関係の世界標準を定めており、コンピュータを含む情報関係ではISOと共同で多くの規格を定めている。
IEEE		The Institute of Electrical and Electronics Engineersの頭文字を取ったもの。日本語では「米国電気電子技術者協会」と訳されている。その Computer Society はソフトウェア工学にたいへん力を入れており、さらに Standards Coordination Committee はソフトウェア工学関係の多くの標準を設定している。最近はISO、IECと共同で、コンピュータを含む情報関係の規格を定めている。
ISO		International Organization for Standardization が元の正式名称。日本語では「国際標準化機構」と訳されている。幅広く世界標準を決める活動を行っており、その領域はコンピュータや情報システムに限定されない。。
ITスキル標準		経済産業省が定めた、ソフトウェア技術者のIT関連能力を職種や専門分野ごとに明確化・体系化し、IT人材に求められるスキルとキャリア（職業）を示した指標。ITサービスの分野11職種35専門分野ごとに最高7段階のスキル・レベルを設定し、それぞれのレベルについて要求される業務経験や実務能力、知識を定義し、さらに上位のレベルに達するためのキャリアパスまで一体となっているところに特徴がある。最初の版が2002年12月に発表され、現時点（2007年10月8日）での最新版は、2006年4月に発表されたバージョン2である。
Linux		リーナス・トーバルスが中心になって開発された、オープン・ソース・ソフトウェアの代表とでも言うべきソフトウェア。狭い意味のLinuxはOSのカーネルだけなので、それ以外のフリーなソフトウェアと組み合わせて、サーバなどのOSとして広く使われるようになってきている。無料（または安価）で、信頼性が高く、使っている人たちからはたいへんに評判がよい。
OMG		Object Management Group の頭文字を取ったもの。「オブジェクト・マネジメント・グループ」を参照のこと。
PDCAサイクル		PDCAとは、Plan、Do、Check、Action の頭文字を取ったもの。何かを行うときに、先ずしっかりと計画を立て（P)、それに基づいて実施し（D)、終わったら結果をチェックして必要なら修正を施し（C)、その後でうまくないことが起きた原因を究明して同じような状況が起きても再び同じ間違いを繰り返さない方策を講じる(A)ようにすることをいう。「PDCAサイクルを回す」とは、あるテーマに対してこの手順を何度も繰り返すことで、これによって製品の作り方などを継続的に改善することができる。ISO 9001の根本的な概念の1つ。
PMBOK		Project Management Body of Knowledge の頭文字を取ったもの。プロジェクト管理に関する一連の知識を体系化したもの。

PSP	Personal Software Process の頭文字を取ったもの。「パーソナル・ソフトウェア・プロセス」を参照のこと。
QCD	品質（Q：Quality）、コスト（C：Cost）、納期（D：Delivery）をまとめたもの。もちろんこの3つの中での優先順位付けはなされるが、ソフトウェア開発プロジェクトが達成しなければならない要素を表している。
Regression Test	「回帰テスト」を参照のこと。
Return On Investmen	「投資利益率」を参照のこと。
Return On Investmen	「正味現在価値」を参照のこと。
Ruby	まつもとゆきひろ（松本行弘）氏が中心になって開発された、日本発のオープン・ソース・ソフトウェアの1つ。オブジェクト指向のスクリプト言語で、既に世界中の多くのプロジェクトで使用されている。
Software Crisis	「ソフトウェア危機」を参照のこと。
Software Engineering	「ソフトウェア工学」を参照のこと。
SWEBOK	Software Engineering Body of Knowledge の頭文字を取ったもの。ソフトウェア工学に関する一連の知識を体系化したもの。IEEEのComputer SocietyとACMの合同プロジェクトが策定した。今最新のものは2014年に発行された v3 である。
TSP	Team Software Process の頭文字を取ったもの。「チーム・ソフトウェア・プロセス」を参照のこと。
UML	Unified Modeling Language の頭文字をと他もの。「ユニファイド・モデリング・ランゲージ」を参照のこと。
Unix	1960年代にAT&Tのベル研究所で、ケン・トンプソンなどが開発したOS。当初はアメリカの独占禁止法運営の考え方からほとんど無料に近い値段で配布され、大学や企業、政府機関などに広がって、フリー・ソフトウェアの考え方を築く上で大きな存在となった。今ではその著作権がしっかりと確立し、それを基にして多くの商用UNIXがコンピュータ・メーカなどから提供されている。
アーンド・バリュー・マネジメント	予算と予定をベースにし、それと実績との対比でプロジェクトの進捗状況を定量的に把握する方式の1つ。
アジャイルソフトウェア開発	バリー・ベームが提唱したスパイラル型開発手順などを採用し、迅速に、短期間でソフトウェアを開発しようとするソフトウェアの開発方式。エクストリーム・プログラミング（XP)など、いくつかの方式が提唱されている。
アジャイルソフトウェア開発宣言	アジャイルソフトウェア開発の方式を提唱していた17人が2001年2月に米国ユタ州のスキーリゾート地に集まって議論し、採択した宣言。アジャイルソフトウェア開発の精神が簡単な宣言文にうまく表されている。これとは別に、「アジャイルソフトウェア開発の原則」がある。

イタレイティブ型開発手順		ソフトウェアを開発を担当している技術者の気の向くままに、適宜テーマを決めて開発を行う方式。エクストリーム・プログラミング（XP)などのアジャイルソフトウェア開発手順は基本的にこの方式を採用している。
インクリメンタル型開発手順		大きなソフトウェアを一気に全部開発するのではなく、そのソフトウェアをいくつかの部分に分けて、それぞれを順次開発することで最終的に全部のソフトウェアを開発する方式。それぞれの開発はウォータフォール型で開発される。
インシデント（Incident）		テストの間や本番中に発生した事象の中、調査が必要なもの。バグや欠陥とは限らないが、その可能性を持つものを広く含む。
インスペクション		公式レビューの1つ。あらかじめ訓練を受けたリーダが全体の活動を取り仕切り、整然とレビューを進める方式。最も欠陥除去率が高い。
インフォメーション・エンジニアリング（IE）		1980年代後半にアメリカで議論されていた企業全体の情報システム構築の考え方。「企業にとって重要なものはそのデータである」という考え方を根本に持ち、企業全体のデータモデルをまず固めて、それに基づいてデータの重要性を定め、情報システム開発の優先順位を決めて順次個別の情報システムを開発しようとする方式。なお日本で「データ中心アプローチ」と呼ばれているソフトウェアの開発方法論は、アメリカでは「インフォメーション・エンジニアリング」と呼ばれている。
インフラ型情報システム		紙と鉛筆で仕事をしていた時代は終わって、毎日パソコンと対話をしながら情報を入手し、分析し、指示する時代に入って久しい。この場合コンピュータは特定の業務だけに使用されるのではなく、広く多くの業務で共通に使用されており、まさにビジネス遂行のためのインフラになっている。ネットワークやサーバも、このような性格を持っている。さらに、最近とみに重要性を増している「セキュリティ関係の投資」も、このインフラ型に分類される。
ウォークスルー		公式レビューの1つ。成果物を作成した人が中心になり、メンバーを募り、ミーティングを開いてレビューを実施する方式。インスペクションと比較すると簡便だが、欠陥除去率は少し低くなる。
ウォータフォール型開発手順		山から海に向かって水が流れ降りるように、後戻りをすることなく順次作業を進める方式のソフトウェア開発の手順。最も基本的なソフトウェア開発の手順である。ソフトウェア開発のための作業を「上流」、「下流」などという呼び方をするときがあるが、それはここから来ている。
運用テスト		システムテスト済みのソフトウェアをテストして、ユーザはこのソフトウェアを使って本番開始後の実作業を遂行できるか、運用担当者は円滑に運用を行ってゆくことができるかの観点から、それぞれの確認を行うテスト。一般に、ブラック・ボックス・テストの方式で実施される。このテストが終わると、移行（切り替え）の作業を経て、このソフトウェアを使用しての本番作業が開始される。

エクストリーム・プログラミング（XP）		アジャイルソフトウェア開発の1つの方式。2000年にケント・ベックが提唱した。XPと略称されている。
エンタープライズ・アーキテクチャ（EA）		企業がその全体の情報システムを構築するに当たって、単にデータだけを考えるのではなく、技術、アプリケーション、そしてビジネスまで全体を統合した形で企業の情報システムの体系を考えようとするアプローチ。日本では経済産業省が提唱している。
オオカミ人間		東ヨーロッパの言い伝えで、普段は普通の人間だが満月の夜に突然牙をむき、爪を伸ばして、恐ろしい怪物に変身して人間を襲うものがいる。この怪物を、「オオカミ人間」という。この怪物を倒すには、ピストルに「銀の弾丸」を詰めて撃つしか方法がない。銀の弾丸は普通の鉛の弾丸と比較して、堅くて、貫通力が強い。順調に進んでいると報告されていたプロジェクトの成果物が、突然バグだらけだと判明することがある。フレッド・ブルックスはこれを「オオカミ人間への変身」と捉え、この問題を一挙に解決する「銀の弾丸」をほしがる人に「銀の弾丸はない」というペーパを書いて、着実な努力の継続の必要性を説いた[BRO86]。
オープン・ソース・ソフトウェア		プログラムは一般にそれを開発した企業の所有物であり、そのソース・プログラムが社会に公表されることは、これまでは原則としてなかった。しかしオープン・ソース・ソフトウェアは、著作権の問題はともかくとして、その内容をインターネットを通して公開し、必要な修正を広く受け入れるという方式で開発されたプログラムである。Linuxがその代表選手であるが、信頼性が高く、使用料が低く抑えられ、いろんな意味で好ましい性格を持ったものと、広く受け止められている。
オブジェクト		オブジェクト指向技法を用いてソフトウェアを開発する場合の、基本の概念。オブジェクトは、「この世の中に存在する『人』や『もの』などの全ての総称」と言える。オブジェクト指向の世界では、しばしば「オブジェクト」と「クラス」が混同して使われるので、注意が必要である。
オブジェクト指向技法		この世の中に存在するもの（オブジェクト）をコンピュータの中に移し替えて、このオブジェクトを核にしてソフトウェアを開発しようとする方式。ソフトウェアの開発方法論の1つで、これからの本命と目されている。
オブジェクト・マネジメント・グループ（OMG）		アメリカにある非営利の民間団体。オブジェクト指向技法の普及と定着を図ることを目的にしており、CORBA（Common Object Request Broker Architecture）やUML（Unified Modeling language）などを開発・発表して、大きな効果を上げてきた。OMGと略称されている。
回帰テスト		システムテスト段階で欠陥が見つかってそれを取り除いたり、保守作業で既存のソフトウェアを修正したりした場合に、本来修正するべき機能以外の、それまで的確に稼働していた部分の機能が今回行って修正で損なわれて稼働しなくなることがある。それが起きていないことを確認するためのテスト。
概念データモデル		実社会のさまざまな事実をデータベースに格納するに当たって、最初に作成するデータモデル。実際は実体関連図（ER図）で表されることが多い。

開発のためのCMMI		米国カーネギ・メロン大学（CMU）のソフトウェア工学研究所（SEI）が開発したソフトウェアプロセス改善のための方法。兄弟の方法に、「購買のためのCMMI（CMMI-ACQ）」と「サービスのためのCMMI(CMMI-SVC）」がある。いずれも今のバージョンはv1.3である。
開発方法論		ソフトウェアの開発を行う場合、ある「考え方」に基づいて個々の作業の内容や成果物の形式／成果物に記載するべき内容などを具体的に定める、そのベースにある「考え方」と、作業を行ったり、あるいは作業成果物を作ることに関するルール集。主なものに、構造化技法、データ中心アプローチ、オブジェクト指向技法の3種類がある。
外部イベント		お客様が商品を購入してくれた／業者が商品を納入した、などの組織の外側からの働きかけでコンピュータ処理を行う場合のきっかけ。オンライン・リアルタイム方式で実行されることが多い。
改良保守		ソフトウェアの訂正、つまり問題への対応ではないソフトウェアの保守。適応保守と完全性保守を合わせたもの。（JIS X 0161：2008より）
カウボーイ開発		管理も計画も、規約もルールもない、野放図なソフトウェアの開発の仕方。ここでの「カウボーイ」は、アメリカの西部劇での無法者を表している。女性用に、「カウガール開発」という言葉もある。
下流工程		情報システムを開発する作業のうち、プログラミング以降の作業を下流工程と呼ぶ。ウォータ・フォール型の開発手順を、前提にしている。
完全化保守		引き渡し後のソフトウェア製品の潜在的な障害が、故障として表れる前に検出し、訂正するための修正。（JIS X 0161：2008より）
管理対象		構造化技法で「実体」と「関連」と呼んだものを、データ中心アプローチではまとめて「管理対象」と呼ぶ。オブジェクト指向技法での「クラス」に相当する。
関連		データベースに格納する対処物の1つ。そのデータ名が動詞の語幹で表すことができるもの。
キィ・パフォーマンス・インディケータ		投資の効果を金額換算しにくい場合、定性的な形で実現できる効果をあげて、ROIなどに代えて使用する。Balanced Scorecardも、この要素を持っている。
機能要求		ソフトウェアで実現するべきものを要求仕様書に記述する場合、アプリケーションとして実現するべき「機能」についての要求。機能以外の要求を、「非機能要求」と呼ぶ。
共通キャリア・スキルフレームワーク（CCSF）		IT技術の分野を3つのキャリア／6つの人材像に分け、さらにそれぞれに7つのレベルを設定して、どこにIT技術の領域があるかを示した図。ITスキル標準、情報システムスキル標準、組込みスキル標準の中核に位置するもの。

業務効率型情報システム		システム化によって業務効率を高め、省力化を推進するための情報システム。人手による作業を順次システム化していった第一世代のシステム化の時代には、この効果が大きかった。しかし基幹業務システムの再構築の時代に入ると、普通のアイディアの出し方では効率的な案の実現は難しくなったといわれている。
緊急保守		「是正保守」の中で、緊急に対応しなければならないもの。（JIS X 0161：2008より）
銀の弾丸		難しい問題を一挙に解決できる強力な方法。オオカミ人間を倒しには、これをピルトルにつめて撃つしかない。「オオカミ人間」を参照のこと。
クラス		オブジェクト指向技法を用いてソフトウェアを開発する場合の、核になる存在。しばしば「オブジェクト」と混同されている。今対象としているソフトウェアの世界で重要なデータで、構造化技法などで「実体」と「関連」、データ中心アプローチでは「管理対象」と呼ばれているもの。
クラス図		オブジェクト指向の世界でのそれぞれのクラスとクラス間の関係などを、図の形で表したもの。UMLで定義されているものの中で、最も重要な図の1つ。構造化技法での「実体関連図」と対応している。
クラス体系		オブジェクト指向の世界でのクラス間の関係（親子関係、全体／部分の関係）を表してもの。
継承		クラス体系で親子関係にあるクラスの間に、親のクラスの属性と振る舞いを子供のクラスが引き継ぐことができること。オブジェクト指向技法の最大の特徴の1つであるが、これを多用すると2つ以上のクラスの関係が非常に煩雑になる恐れがある。
欠陥除去率		テストやレビューで、存在している欠陥のうち発見して除去できる欠陥の割合。テストに比較するとレビューの方が、一般に欠陥除去率が高い。
結合テスト		単体テスト済みのプログラムをインタフェースを確認しながら結合して、最終的に大きなプログラムにするまでのテスト。一般に、ホワイト・ボックス・テストの方式で実施される。
限界値分析		ブラック・ボックス・テストでのテストケース設定方法の1つ。ある値が取る特定の数値で処理の内容が変わる場合、その数値を限界値という。ここのところに欠陥が埋め込まれることが多いことから、そこを重点的にテストデータを用意してテストする方法。
検証		作業成果物が明記された要件を適切に反映しているかどうか確認すること。つまり、検証は、「正しく構築した」ことを確実なものにすることである[CMM06]。英語ではverification という。
構成管理		「本来の構成管理」、または「ソフトウェアの構成管理」を参照のこと

構成品目		ソフトウェアの構成管理の対象にする成果物。ソース・プログラムを始め、要求仕様書、各種の設計書、各種の規約、計画書類などが構成品目になりうる。
構造化技法		ソフトウェアの開発方法論の1つ。「プロセス中心アプローチ」とも呼ばれる。一般の工学で行われている「分割と統合」の方式をソフトウェアの開発に適用しようとする方式。ソフトウェアの世界ではこれを、「トップ・ダウン・アプローチ」と呼んでいる。
構造化定理		「全てのプログラムは『順次』、『選択』、『繰り返し』の3つの制御構造だけを使って記述することができる」とする定理。数学的に証明されている。
構造化プログラミング		構造化定理に基づいて行うプログラミング。最初は単に「GOTOステートメント」を使わないでプログラミングすることを構造化プログラミングと呼んだ。
国際電気標準会議		IECのこと。「IEC」を参照のこと。
国際標準化機構		ISOのこと。「ISO」を参照のこと。
コピーレフト		「ソフトウェアは自由であるべき」とするリチャード・ストールマンのフリー・ソフトウェアに関わる考え方。GPLなどにその考え方が実現されている。「著作権」を表す英単語（Copyright）をもじって名付けられた。
ゴンペルツ曲線		残存欠陥数を推定する時に使われる曲線の1つ。1825年にイギリスの数学者ベンジャミン・ゴンペルツ（Benjamin Gompert）が提唱した。
再利用		別の問題の解決の中でのある資産の使用。および、新しいアプリケーションを構築するために既存の部品を使用して、少なくとも部分的に、ソフトウェア・システムを構築すること[ISO10a]。
サブ・クラス		クラス体系を表したときの、子供のクラス。スーパ・クラスとは逆の立場。
残存欠陥数		開発期間中にテストやレビューで取り切れなくて、本番稼働後にもまだソフトウェアの中に残っている欠陥の数。残存欠陥は本来的にはあるべきではないが、これを無くすることはたいへんに難しい。
残存リスク		リスク分析を行ってリスクを識別したが、その対策に費用がかかる／人手がかかる、などの理由で対策を実行できないリスク。顕在化しないかどうかを、注意深く監視する必要がある。
時間的イベント		コンピュータ処理のきっかけの1つ。閉店時間になった／週末が来た／月末になった／決算日になった、など時間的な要素で作業を行う場合のきっかけを指す。その処理はバッチ処理で実行されることが多い。
システム／ソフトウェアライフサイクルプロセス品質		情報システムやソフトウェアを作る時の作り方の品質。これが最も重要な「利用時の品質」のベースになる。

システムテスト		ソフトウェアの開発者として行う最終のテスト。要求仕様書で明記された機能が実現されていることの確認に加えて、非機能も実現されていることを併せて確認する必要がある。このため、規模の大きなテストになることがある。一般にブラック・ボックス・テストの方式で、実施される。
実体		データベースに格納する対処物の1つ。そのデータ名を純粋の名詞で表すことができるもの。
実体関連図		データベース化の対象物など、今注目している情報システムの領域での重要なデータを実体と関連に分けて、それらの間の関係を表した図。構造化技法とデータ中心アプローチで用いられる。
修正ウォータフォール型開発手順		基本的にはウォータフォール型の開発手順であるが、適宜修正要求に対応するために後戻りを繰り返しながら開発を進めるソフトウェアの開発方式。
修整プロセス		一般的に作られているソフトウェアの開発手順などを、これから開発しようとするソフトウェアに合わせてその内容を変更し、その開発を担当するプロジェクトなどに適用する準備をする作業。これからますます重要になる作業だが、高いレベルの技術力が必要である。
重要インフラ等システム		「他に代替することが著しく困難なサービスを提供する事業が形成する国民生活・社会経済活動の基盤であり、その機能が低下又は利用不可能な状態に陥った場合に、我が国の国民生活・社会経済活動に多大の影響を及ぼす恐れが生じるもの、人命に影響を及ぼすものおよびそれに準ずるもの。」（経済産業省、「情報システムの信頼性向上に関するガイドライン」、平成18年6月16日）
状態遷移図		実体や関連の状態の遷移を表した図。データフロー図、実体関連図と共に、構造化技法やデータ中心アプローチでのシステム分析段階で作成される。
状態マシン図		UMLで定義されたモデルの1つ。「状態遷移図」に、非常に近い。
情報隠蔽		オブジェクトでカプセル化を行って、重要ではない情報を外部から隠すこと。重要な情報は「そのオブジェクトが持っている機能」と、「それぞれの機能を稼働させるときの処理の依頼の仕方（インタフェース）」だけとされている。
上流工程		情報システムを開発する作業のうち、設計までの作業を上流工程と呼ぶ。ウォータ・フォール型の開発手順を、前提にしている。情報システムの企画や要件定義の作業を、「超上流」と呼ぶこともある。
スーパー・クラス		クラス体系上、親の位置にあるクラス。サブ・クラスの逆の立場。
スタブ		単体テストと結合テストで使用するテスト・ツールの1つ。テスト対象のプログラムから呼ばれて、必要な答えをそのプログラムに戻すのが必要な役割である。

スパイラル型開発手順		小刻みに何度も設計・開発・テストの手順を繰り返しながら、ソフトウェアを開発する方式。一回の設計・開発・テストの手順を1つの回転と捉え、全体として渦巻き状に開発を行う。バリー・ベームが提唱した。
正規化		更新を伴うデータベースの設計を行う時、更新異常を避けるために行わなければならない処置。理論的には「第5正規形」まであるが、実務的には「第3正規形」と呼ばれる段階まで行えばよいとされている。
成熟度レベル		SW-CMMや開発のためのCMMIの段階モデルで、組織のソフトウェア・プロセスの成熟度を表す指標。レベル1からレベル5までの5段階があって、数字が大きくなるほどプロセスがよく成熟していることを表す。
製品の保証		ソフトウェアの品質保証の方法の1つ。ソフトウェアの特性を様々な観点で計測して、その計測結果に基づいて品質を保証する方法。
是正処置		何か良くないことが起きた場合、その影響を無くする／軽減するために行う処置。同じ状況があっても、その良くないことが再び起きないようにする処置を含む。
是正保守		ソフトウェア製品の引き渡し後に発見された問題を訂正するために行う受け身の修正。（JIS X 0161：2008より）
戦略型情報システム		顧客との関係を強化することにより売り上げの拡大を図る情報システム、ITの新しいツールを活用したビジネス・モデルにより市場拡大を図る情報システムなど、様々なタイプがある。ITが業務を効率化するツールから企業の戦略を実現するためのツールへと変化する時代に入り、このタイプの投資が増加した。最終的には企業全体、あるいは対象の事業部の営業利益で評価するしか方法がないことが多い。
ソフトウェア		コンピュータのプログラムを抽象的にとらえる呼称。
ソフトウェア・エンジニアリング		「ソフトウェア工学」を参照のこと。
ソフトウェア危機		ソフトウェア開発について最近起きている「開発が当初に立てたスケジュールから遅れる」、「開発費用が当初の予算を超過する」、「開発した製品の品質が悪い」、さらには「開発プロジェクトが最終の製品を作り出す前に解散させられる」といった好ましくない現象を総称したもの。
ソフトウェア工学		ソフトウェアの開発、運用、保守の系統的な方法。（IEEE）
		ソフトウェアの作成と利用に関連した概念を科学的に抽出し、正しいソフトウェアを計画的に作成・利用するための理論と実践的技術。（岩波情報科学辞典）

ソフトウェア工学		現実のコンピュータの上で稼働する信頼性の高いソフトウェアを経済的に開発するための、完全な工学の原則の確立と活用。（フリードリッヒ・バウアー）
		高品質のソフトウェアを経済的に生産することを目指したエンジニアリング。科学および数学の原理、手法、及びツールの秩序だった応用。（ワッツ・ハンフリー）
		コンピュータのソフトウェアの設計・開発・利用・保守・品質評価などを工学的見地から研究する学問領域。（広辞苑第六版）
		その本性としてよく間違いをする人間が、本来完全無欠であるべきソフトウェアを開発するために必要とする理論や技術、仕事の仕方や手順、考え方、組織や技術者個人のあるべき姿などを総合的にまとめ上げた体系。
ソフトウェア工学研究所（SEI）		1984年に米国の国防総省がカーネギー・メロン大学の中に作った組織。CMMやCMMIを開発したことで著名。
ソフトウェア構成委員会（CCB）		構成管理を実施する時に、構成品目に変更の要請があった場合、その要請を受けるか拒否するかを決定する機関。英語では Configuration Control Board と呼ばれ、その頭文字を取ってCCBとも呼ばれる。
ソフトウェアの構成管理		「本来の構成管理」が持っている変更を管理する機能に注目し、ソフトウェアの変更に対処する仕組み。構成識別、構成制御、構成状況記録、構成評価、ソフトウェアのリリース管理及び出荷、及びインタフェース制御などの作業で構成されている。英文では Software Configuration Management と呼ばれるので、その頭文字を取ってSCMと呼ばれることがある。
ソフトウェアの品質		ソフトウェアの品質について定めたISO／IEC 25010では、ソフトウェアの品質として最も重要なものは「利用時の品質」であるとしている。そしてこれを実現するためには、そのベースにあるライフサイクル・プロセス品質が良くなければならないとしている。
ソフトウェアの品質保証		「品質保証」とは、誰かがユーザなどに特定の製品の品質が充分なものであることを保証すること。ソフトウェアの品質保証とは、これをソフトウェアについて行うこと。ISO／IEC 12207：2008ではその保証の方法は、「製品の保証」、「プロセスの保証」、「品質システムの保証」の3つの方法で行うことができるとしている。
ソフトウェアの保守		ソフトウェアの完全性を維持しながら、既存のソフトウェア製品に対して修正を行うこと。（JIS X 0161：2008より）
ソフトウェア・プロセス		ソフトウェアを作るための作業、あるいはソフトウェアを作る手順。

ソフトウェア・プロセス改善		「製品を作る作り方（プロセス）が良ければその結果作られる製品の品質がよい」という今の一般の工業製品の品質保証の考え方を受けて、ソフトウェアの品質を良くするためにその作り方を要して行こうとすること。CMMIやISO／IEC 15504、ISO 9001などによる方法がある。
ソフトウェア・メトリクス		ソフトウェアそのもの、及びソフトウェアの開発過程を計測すること、及びその計測の結果。ソフトウェアの品質を向上させたり、ソフトウェア開発の生産性を向上させたり、あるいはソフトウェア開発プロジェクトの良い計画を立案するために、ソフトウェアに関わる計測をして、その結果を分析し、評価し、蓄積することが欠かせない。
ソフトウェア・ライフサイクル		あるソフトウェアについて、その必要性が認識されたときから破棄されるまでの、文字通りソフトウェアの一生を意味する。
ソフトウェア・ライフサイクル・プロセス		単に開発だけでなく、開発組織の運営や要員の教育、環境作り、監査、ソフトウェアの破棄まで含めた、ソフトウェア作りに関わる全ての作業を総称にしたもの。
多相性		オブジェクト指向技法でのオブジェクトが持っている特徴の1つ。スーパ・クラスであるメッソドが定義され、それがサブ・クラスに継承されている場合に、スーパ・クラスのメソッド名で要求を出したとき、サブ・クラスごとに異なる振る舞いを実現するとき、多相性があるという。例えば、スーパ・クラスが「図形」で、サブ・クラスに「円」、「三角」、「四角」などがあり、スーパ・クラスに「Draw（描く）」というメッソドがあって、サブ・クラスごとにそれぞれ自分の形を書くようなことが、多相性の例である。
妥当性確認		提供された (または提供されるであろう) 成果物がその意図された用途を充足しているかどうか確認すること。つまり、妥当性確認は、『正しいものを構築した』ことを確実なものにすることである[CMM06]。英語では validation という。
段階的詳細化		最初からいきなり詳細なものを記述するのではなく、最初はラフなものに留め、作業が進むにつれて記述内容を詳細にしてゆくやりかた。PMBOKでの計画の立て方など、情報システム関係では段階的詳細化で対応するものが多い。
単体テスト		ソフトウェアのテストの一種。一般に、プログラミング終了直後に、そのプログラムを作成したプログラマ自身が実施する。ホワイト・ボックス・テストの方式で行われ、全てのステートメントの通過」、あるいは全ての条件判定でTrueとFalseへの分岐について、全てを網羅することを原則とする。
チーフ・プログラマ・チーム		1970年代にIBMが提唱した軽量のソフトウェア開発の方式。「チーフ・プログラマ」と呼ばれるスーパ・SEを核にして、彼／彼女の技術を100%引き出して、迅速に品質の高いソフトウェアを開発／提供しようとする方式。

チーム・ソフトウェア・プロセス（TSP）		ソフトウェア技術者がチームを作ってソフトウェアを開発する時に、そのチームが所持するべきソフトウェア・プロセス。当初はパーソナル・ソフトウェア・プロセス（PSP）を習得したソフトウェア技術者がその成果を生かせる環境を作るという目的で策定されたが、PSPとTSP（Team Software Process）の両方を習得した技術者で構成されるチームの成果物の品質と生産性が素晴らしいことが明らかになり、今やその目的を達成するために目標としてTSPを習得するというように変わってきている。
超高速開発		超高速開発とは、業務デザインから運用・保守工程を含めたシステム・ライフサイクル全般にわたる生産性向上と継続的品質改善を行うやり方。「超高速」には、「期間短縮」、「工数削減」と「品質向上による手戻り削減」の意味を含む[ICT14]。
データ中心アプローチ（DOA）		ビジネスの世界では、処理のプロセスよりもデータ構造の方がはるかに安定しているという経験則から、そのデータを全体の中心において情報システムを構築しようとする開発方法論。開発の最初の段階でデータ分析を行ってデータベースの設計を行うところに特徴がある。
データの標準化		1つの組織の中で、1つのデータ項目の名称はただ1つに限定すること。
データフロー・ダイアグラム		コンピュータの中のデータの流れを記述した図。実体関連図、状態遷移図と共に、構造化技法やデータ中心アプローチのシステム分析段階で作成される。
データ分析		情報システムが作成するべき出力を基に、データベースに格納するデータを確定する作業。トップ・ダウンとボトム・アップの方法がある。
適応保守		ソフトウェアの引き渡し後に、変化した、または変化している環境において、ソフトウェア製品を使用できるように保ち続けるために実施するソフトウェア製品の修正。（JIS X 0161：2008より）。この環境には、運用環境と組織／社会環境がある。
テスト		作成したプログラムをコンピュータにかけて実際に動かして、欠陥などを見つける方法。単体テスト、結合テスト、システムテスト、運用テストなどのフェーズで構成される。
テストドライバ		単体テストと結合テストで使用するテスト・ツールの1つ。上位のプログラムに変わってテスト対象のプログラムを呼び出すことに加え、テストの状況をつぶさに記録を取る機能も持つことが多い。カバレージを計測する機能があれば、もっと好ましい。
デスマーチ・プロジェクト		「デスマーチ」とは「死の行軍」を意味し、太平洋戦争中にフィリピンで起きた、旧日本軍が米軍捕虜に課したたいへん厳しい徒歩による移動を指す。ここからこの言葉は「たいへん苦しい思いをしなければならないのに、その結果が報われることがない」ことを意味するようになった。「デスマーチ・プロジェクト」とは、「通常必要とする時間の半分以下の時間しか与えられていないようなソフトウェア開発プロジェクト」を指す（「デスマーチ（第2版）」より）。

テラーリング		「修整プロセス」を参照のこと。
同値分割法		ブラック・ボックス・テストでのテストケース設定方法の1つ。ある値が取る範囲を処理の内容でいくつかの領域に分けて、それぞれの領域毎にテストデータを1件（以上）用意してテストを行う方法。
トレーサビリティ		要求仕様書上に書かれた特定の機能が、設計書のどの部分に記述され、それがどのモジュール／クラスで実現されているか、及びあるモジュール／クラスに記述されている機能が設計書のどこで定義されたもので、さらに要求仕様書のどこで要求されたものであるかを、双方向に追跡できる機能。
ノイマン型コンピュータ		アメリカの数学者フォン・ノイマン（John von Neumann）が提案したとされる、ハードウェアとしてのコンピュータの方式。今のほとんどのコンピュータはこの方式のものである。
能力成熟度モデル統合（開発のための）		カーネギー・メロン大学ソフトウェア工学研究所（SEI）が開発したソフトウェア・プロセス改善のためのモデル。CMM-DEVIとも呼ばれる。当初SEIが開発した能力成熟度モデル（SW-CMM）を次ぐもの。ソフトウェア工学だけでなく、システム工学の領域も対象にしている。段階表現と連続表現の2種類の表現形式を持つ。
能力レベル		能力成熟度モデル統合（CMMI）の連続表現で、プロセス・エリア別にプロセス改善を進めた結果として到達したレベルを示す指標。れベル0からレベル5までの6段階がある。
パーソナル・ソフトウェア・プロセス（PSP）		ソフトウェア技術者が個人として所持するべきソフトウェア・プロセス。自分自身の生産性を把握する部分と弱点を認識する部分から構成される。これを習得することで、個人として作成する成果物の品質と生産性が飛躍的に向上する。Personal Software Process（PSP）の習得が、Team Software Process（TSP）を習得するための前提条件になっている。
パレートの法則		イタリアの経済学者ヴィルフレド・パレート（Vilfredo Federico Damaso Pareto）が発見したとされる法則で、「80：20の法則」とも呼ばれ、数では全体の20%のものが、実質80%のものを占めている、ということを表す場合に使われる。ITの世界では、「20%のモジュールが全体の80%のバグを保持している」とか、やはり「20%のモジュールでCPU使用率の80%のCPUを使っている」とかということがある。このような場合に、「パレートの法則が成立する」という。
ハロウィーン文書		Linuxがその中主力製品であるWindowsを凌駕するのではないかという問題を提起し、分析し、対応を検討したマイクロソフトの社内文書。本来これは機密文書のはずだが、なぜか1998年のハロウィーン（10月31日）の頃に公表されたのでこの名前が付いている。
ピープルウェア		ハードウェア、ソフトウェアに次ぐ、情報システムを構成する3つ目の要素としての「人」の問題を考えようといういうことで、トム・デマルコとティモシー・リスターが作った「造語」。この言葉を書名にした名著がある。

非機能要求		ソフトウェアへの要求の中、「機能要求」に属さないもの全ての総称。パフォーマンスや品質、保守性などについての要求を含む。開発されるソフトウェアのアーキテクチャの設計で重要な位置を占める。
標準化団体		ISOやIEC、IEEEや日本規格協会のように、国際規格や国内規格を定めている団体。
品質		現時点での正式の定義は、「本来備わっている特性の集まりが、要求事項を満たす程度（ISO 9000：2015）」というもの。品質についてのオーソリティであるクロスビーは、「ユーザの要求を満たす程度」と定義している。
品質システムの保証		ソフトウェアの品質保証の方法の1つ。ソフトウェアを開発する組織がその品質マネジメントシステムにISO9001を使用していることを通して、開発されるソフトウェアの品質を保証しようとする方法。
品質マネジメントシステム		ISO 9001 によって構築されるマネジメントシステム。高品質の製品を製造する上で必要なもので、その活動を通して製品の品質を継続的に改善することができる。
ファンクション・ポイント		プログラムの大きさを表す方式の1つ。よく使われていたステートメント（ステップ）数の方式では同じ機能のプログラムでも、使用するプログラム言語で数字が大きく異なるという弊害がある。その弊害を避けるために、プログラムの機能に注目して量を決める所に特徴がある。IFPUG方式など、いくつかの方式が提案されている。
フォワード・エンジニアリング		設計書から製品を作る方式。普通のエンジニアリングの方式だが、「リバース・エンジニアリング」との対比でこの言葉が使われる。
ブラック・ボックス・テスト		プログラムの中を見ずに、要求仕様書などからテストケースを設定（仕様ベースのテストケース設定）して行うソフトウェアのテストの方法。一般にシステムテストなど、テストの後半のテストはこの方式でテストされる。
フリー・ソフトウェア		「ソフトウェアは自由であるべき」とするリチャード・ストールマンの考え方に基づいて開発され、配布されているソフトウェア。
ブルックスの法則		フレッド・ブルックス（Frederick P. Brooks, Jr）がソフトウェア開発プロジェクトの運営について述べたもの。「スケジュールから遅れ始めたプロジェクトに新たに人を追加投入すると、スケジュール遅れが一層ひどくなる」というもの。
プロジェクト		独自のプロダクト、サービス、所産を創造するために実施する有期性のある業務[PMI13]。

プロセス		入力を出力に変換する作業。次の属性を持つ。 ☐目的（Purpose） ☐入力（Inputs） ☐開始基準（Entry criteria） ☐活動（Activities） ☐役割（Roles） ☐測定（Measures） ☐検証のステップ（Verification steps） ☐出力（Outputs） ☐終了基準（Exit criteria）
プロセス・エリア		能力成熟度モデル統合（CMMI)で、プロセス改善の対象になっている領域。22の領域がプロセス・エリアに指定されている。
プロセス中心アプローチ（POA）		構造化技法を使用してソフトウェアを開発する場合の、開発の方式の1つの呼び方。「データ中心アプローチ」と対比されている。
プロセスの保証		ソフトウェアの品質保証の方法の1つ。「製品を作るプロセスが良ければ、その結果作られる製品の品質が良い」という現在の一般の品質保証の考え方をソフトウェアに適用して、ソフトウェアの品質を保証しようとする方法。
プロトタイプ		システムの分析や設計作業の一部で、ユーザの要求を具体的に把握するために作成する一種の模型。例えば極力実物に近い出力を作ってユーザに提示し、そのコメントを基に修正を繰り返して、最終的にユーザの要求を明確にするために使用する。さらにそのプロトタイプは、最終の製品に取り込まれる。（「モックアップ」を参照のこと。）
分割と統合		一般の工業製品の設計で行われている方法。複雑な最終製品を設計するために、その製品を各構成要素に分割し、そこで最良の解を求めてその結果を持ち寄り、統合して最終製品を設計しようとする方式。
ペア・プログラミング		エクストリーム・プログラミング（XP）で採用されている方式の1つ。何の作業を行う時も、プログラマは必ず二人一組で仕事をしなければならないとする方式。
ベスト・プラクティス		ある作業を行う時に幾通りもの方法があるのが普通だが、その中で何らかの基準で評価して、最も優れた作業方法。
ホット・モジュール		多くのバグ（エラー）を持ったモジュール。仕様が明確ではなかったとか、設計が良くなかったとかの理由で、このようなプログラムが作られることがある。レビュー段階や単体テスト段階で多くの誤りやバグを検出してこのようなプログラムを識別することができる。この場合、すでに多くのバグを検出したからもうバグが残っていないと考えるのではなく、無数にあったバグの一部を検出しただけなのでまだ多くのバグが残っていると考える方がよい。つまりこのようなプログラムは放棄して、もう一度慎重に作成し直す方が、良い結果が得られることになることが多い。

ホワイト・ボックス・テスト		プログラムを読んで、それに基づいてテストケースを設定（構造ベースのテストケース設定）し、テストを行う方法。この方式でテストを行う場合、全てのステートメントか全ての判定条件を網羅することが原則である。一般に単体テストや結合テストなど、テストの前半に実施されるテストはこの方式で行われる。
本来の構成管理		ある製品が長い期間にわたって作られ続けるとき、技術革新などでその最終製品を構成している部品などに変更が入る。あるいは部品の組み合わせ方を記述した図面にも変更が入るかも知れない。本来の構成管理とはこのような状況の下で、それぞれに版番号（バージョンナンバー）を付けて、その版番号を用いて最終製品を構成している部品群、使用されている図面類の関係を管理する仕組み。1960年代から米国の国防総省で使用されてきた。
メソッド		オブジェクト指向技法でソフトウェアを開発する場合の、プログラムのこと。
メッセージ		オブジェクト指向技法でソフトウェアを開発する場合、他のプログラムを駆動させる手段。「メッセージをパスする」という表現を使う。
モジュール		「1つの入口、1つの出口、1つの機能」を持った小さなプログラム。複数の機能を持った大きなプログラムをモジュールに分割し、それを組み合わせて元のプログラムの機能を実現する設計方法を構造化設計と呼び、モジュールの強度が強く、モジュール間の結合度が弱いモジュール群を作る設計が良い設計とされている。
モジュール間の結合度		モジュール分割の結果としての、2つのモジュールの間の関連性を表す尺度。モジュール間の結合度は、弱い方が望ましい。
モジュールの強度		モジュール分割の結果としての、1つのモジュール内部のステートメントの結びつきを表す尺度。モジュールの強度は、強い方が望ましい。
モックアップ		システムの分析や設計作業の一部で、ユーザの要求を具体的に把握するために作成する一種の模型。例えば極力実物に近い出力を作ってユーザに提示し、そのコメントを基に修正を繰り返して、最終的にユーザの要求を明確にするために使用する。そのモックアップは、使い終わると破棄する。（「プロトタイプ」を参照のこと。）
ユースケース図		「ユニファイド・モデリング・ランゲージ（UML)」の中の、主要な図の1つ。情報システムが誰に、どのように使われるかを簡便に表したもの。構造化時代には、このような立場で記述する図はなかった。
ユニファイド・モデリング・ランゲージ（UML）		オブジェクト指向技法を用いてソフトウェアを開発する場合、そのソフトウェアを表記するモデル集。オブジェクト・マネージング・グループ（OMG）の活動の1つとして、開発された。最初のバージョンは1995年に発表され、2005年7月にはバージョン2.0が発表された。バージョン2．0にはクラス図、ユースケース図など13種類の図が定義されている。UMLと略称されている。

ユビキタス・コンピュータ		コンピュータが我々の身の回りのどこにでも存在する状態。またはそのような状態になったコンピュータ。
要求工学		開発しようとするソフトウェアについてのユーザなどの要求を明確にし、それを定義するための方法。ソフトウェア工学の一分野だが、その重要性からこの分野は「工学」を付けて呼ばれている。英語では Requirement Engineering と呼ばれている。
要求仕様書		これから開発しようとするソフトウェアについての、ユーザなどの要求を記述した文書。品質を「ユーザの要求を満たす程度」と考えると、高品質のソフトウェアを得るために、この文書の位置づけはたいへんに重い。要求には機能についての要求と、品質やパフォーマンスなどについての要求（非機能要求）がある。
予防処置		何か良くないことが起きそうな場合、それが起きないようにする／起きた場合の影響を軽減するために行う処置。
予防保守		引き渡し後のソフトウェア製品の潜在的な障害が運用障害になる前に発見し、是正を行うための修正。（JIS X 0161：2008より）
ラショナル統一プロセス（RUP）		オブジェクト指向向けの開発手順。全体としてスパイラル型で開発を繰り返すが、スパイラルの開発のポイントを「方向付け」、「推敲」、「作成」、「移行」と順次移して開発を行う方式。米国のソフトウェア会社であった「ラショナル社」が提案した。RUPと略称されることがある。この簡易版として、UP（統一プロセス）がある。
リ・エンジニアリング		「フォワード・エンジニアリング」と「リバース・エンジニアリング」を合体したもの。例えばメインフレーム上で稼働している情報システムを、クライアント／サーバ方式で稼働できるように変換する技術。情報システムでは、まだリ・エンジニアリングは実現されていない。
リスク管理		何かを実施するに当たって、その実施や完成を阻害する可能性のあるもの（リスク）を洗い出し、優先順位付けし、優先度の高いものについて対応方法を考え、そのリスクが起きていないか／起きそうにないかを監視し、必要なら考えて用意した対策を実施して、リスクの影響を軽減／解消すること。
リバース･エンジニアリング		一般に製品から設計書を作成する技術をいう。ソフトウェアの場合には、ソース・プログラムなどから設計書を復元することをいう。フォワード・エンジニアリングの逆方向の作業に当たる。
リファクタリング		エクストリーム・プログラミング（XP）で採用されている方式の1つ。設計書やプログラムなどの成果物は全てチーム内で共同所有され、その成果物が適切ではないと考えた人は誰に断ることなくその成果物の修正や再作成を行って良いとする方式。
量子コンピュータ		量子力学的な方式を用いて並列処理を実現し、これまでのコンピュータより高速性を実現する新しい方式のコンピュータ。
利用時のシステム品質		ソフトウェアを利用する時の、利用者にとっての品質。ソフトウェアの品質の中で最も重要なもの。

倫理		行動の規範としての道徳観や善悪の基準。コンピュータが普及し、強力になり、我々の生活で大きな位置を占めるようになると、このコンピュータのソフトウェアを作成するソフトウェア技術者は高い倫理観を持つことが必要になる。
レビュー		作成された成果物を、作成直後に作成者の同僚などのグループでチェックする方式。インスペクションやウォークスルーなどのレベルがある。テストより欠陥除去率が高い上、テストはプログラミングが終了しないと実施できないことと比較すると、もっと早い時期に欠陥を発見することができ、結果として手戻りを少なくする効果がある。高品質のソフトウェアを開発する上で欠かせない作業。
ロジスティック曲線		残存欠陥数を推定する時に使われる曲線の1つ。1838年にベルギーの数学者ピエール・ベルハルスト（Pierre Verhulst）が考案した。