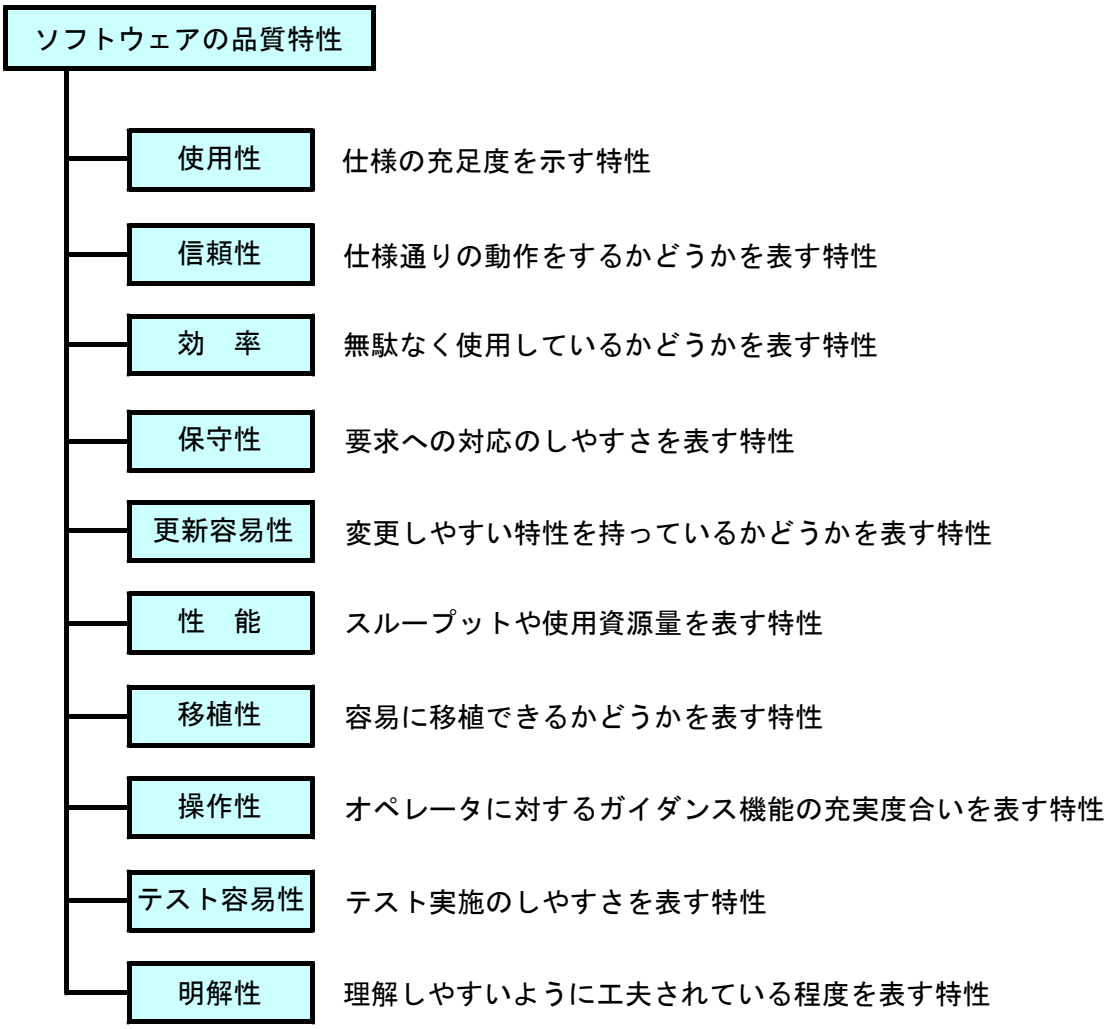
txt040411 開発品質とＣＡＳＥ

①　ソフトウェアの品質

ⓐ　ソフトウェアの品質特性

従来は、使用性、信頼性、効率が代表的な品質を示す特性要因であったが、最近は、保守性、更新容易性、性能、移植性、操作性、テスト容易性などの特性も注目されるようになった。



ⓑ　品質特性の種類と内容

㋐　使用性

仕様の充足度を示す特性で、ソフトウェアの目的とした機能が、仕様通りに表現されているかどうかを表す特性要因である。配慮する特性に、信頼性、効率、操作性がある。

㋑　信頼性

ソフトウェアが仕様通りに動作するかどうかを表す特性要因である。信頼性を高めるため

には、誤動作を起こさないようにする。即ち、バグの少ないソフトウェアにすることである。ソフトウェア開発時に、ソフトウェアの誤動作をできるだけ排除できるような開発手順や技法の適用が重要である。

㋒　効率

コンピュータ資源をどの程度に無駄なく使用しているかを表す特性要因である。効率を確認するにはコンピュータの動作や資源の計量化が必要である。主な計量化対象として、ＣＰＵ時間や主記憶装置容量、外部記憶装置容量、チャネルおよび入出力制御装置などの処理速度がある。ＣＰＵ時間はプログラムの処理内容によって異なる。従って、ハードウェアの絶対時間を計量して比較してもソフトウェアの効率比較にはならない。

㋓　保守性

ユーザからのクレームや要求への対応のしやすさを表す特性要因である。長期の運用サイクル中に、利用者からクレームや要求に対する変更や機能追加などに容易に対応する必要がある。分かりやすく、修正しやすく作ることが保守性を高めるために必要であり、構造化プログラミングやソフトウェアの処理内容の文書化が不可欠となる。

㋔　更新容易性

ソフトウェアや文書が変更しやすい特性をもっているかどうかを表すものである。修正や追加が１つのモジュールに閉じていることが望ましい。そのためには、プログラムに将来どのような修正や追加が行われるかの予測が必要である。

㋕　性能

性能は単位時間当たりの処理量であるスループットや使用する資源の量などで表される特性である。ソフトウェアの性能としては、スループット、レスポンス、プログラムの容量、主記憶装置や補助記憶装置の容量、ＣＰＵ時間などがある。

㋖　移植性

あるコンピュータで動作しているプログラムを、他のコンピュータで動作させるために、どの程度容易に移し換えることができるかを表す特性である。ＯＳやハードウェアが異なると、移植作業が必要になる。この移植作業の容易さを表す尺度である。

㋗　操作性

操作の容易さ、プログラムの実行のしやすさ、ジョブ制御文の作成のしやすさ、対話形式のガイダンス機能の充実などがある。対話形式の操作性では、画面からの問い合わせに簡単に答えられることやメニューやアイコンなどで操作性が容易に行えるなどのオペレータに対するガイダンス機能の充実がある。

㋖　テスト容易性

テストが実施しやすいように考慮されているかどうかを表す特性である。テストの容易性を考慮したプログラムの作り方として、テストケースが作成しやすいような細分化したプログラム構造であることやテストで確認した部分が容易に識別できるプログラム構造であること、テスト結果が正確に判断できるような処理の流れであること、テストツールが簡単に利用できる工夫がなされていることなどがあげられる。

㋗　明解性

プログラムや文書がいかに理解しやすいように工夫されているかの程度を表す特性要因である。プログラムや文書の作成方法、生産物の標準化が重要である。

②　品質評価法と信頼度成長モデル

ⓐ　ソフトウェアの品質管理

ソフトウェアの品質管理は、品質がユーザ要求に適合しているかを評価し、品質の維持向上のために管理することである。品質管理の方法には次の３つの方法がある。

㋐　レビュー方式

ソフトウェアの各開発工程で問題を発見し、品質の維持を図るためにレビューを実施する。レビューの方法には、デザインレビュー、ウォークスルー、コードインスペクションなどがある。

㋑　信頼性予測方式

ソフトウェアの信頼性を数学的理論に基づき予測する手法である。エラー植え付けモデル、信頼性成長モデルなどがある。

㋒　ＱＣ方式

生産管理で用いられている手法の考え方を利用する方法である。ＱＣ七つ道具、新ＱＣ七つ道具などの方式がある。

ⓑ　品質評価法

プログラミング工程以降では、品質評価の方法として、ウォークスルーやコードインスペクションを実施する。更に、バグ発生率によるプログラムの品質管理を行う。テスト計画で設定したエラー率を元に、目標とバグ件数を算出し、目標に対するバグの発生度合いから品質を評価する。プログラムの信頼度の予測には、プログラムのエラーデータを予測する必要がある。ソフトウェアのバグ発生数は信頼度成長モデルで近似できることが経験的に知られている。

ⓒ　信頼度成長モデル

プログラムの信頼度予測をエラーの数に着目して定量的に予測するモデルである。プログラムのテストを続けるとエラーが除去されて、結果として信頼性が向上していく過程を数式でモデル化したものである。

ⓓ　信頼度成長モデルの種類

㋐　解析的モデル

信頼性特性を示す尺度として、プログラム内に存在するエラーの数と、プログラムのテスト時間の二つを取り上げる。これらの尺度をデータによって定量化して予測する。

㋑　経験的モデル

プログラムの複雑さからエラーを予測する。

㋒　傾向曲線モデル

エラーが発生する時刻あるいは時間間隔を対象とした確率的な分布によるモデルで、テストにおける時系列的な経過とエラー発生累積数がＳ字型曲線を示すモデルである。Ｓ字型曲線は、経済成長予測や人口予測などに用いられる。ロジスティック曲線とゴンペルツ曲線の二つがある。

ⓔ　エラー植え付けモデル

プログラムの中にあらかじめ用意したエラーを意図的に埋め込むことにより、ソフトウェアの信頼性を数理的に予測するモデルである。次の手順で行う。

㋐　開発者でないものがプログラムに意図的にＳ個のエラーを埋め込む。

㋑　プログラムテストによって、エラーを検出し、意図的なエラーとプログラム固有のエラーに分ける。

ある時点におけるそれぞれのエラーをｓ、ｅとし、意図的なエラーとプログラム固有のエラーの発生確率が同じであると仮定すると、プログラム固有のエラー数Ｅは次のようになる。

Ｅ＝Ｓ・ｅ／ｓ

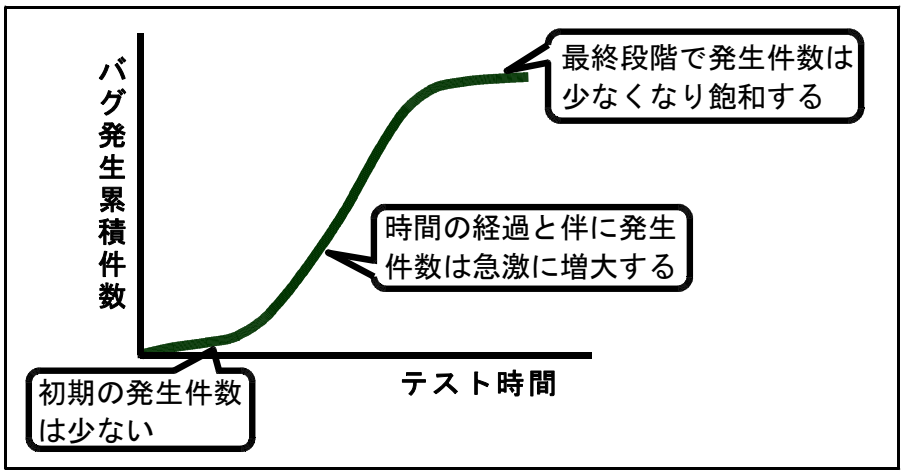
㋒　意図的なエラーＳ個全部が検出されるまでテストを継続する。

プログラム固有のエラー総数の予測値をｍとすると、テストのある時点でのｍの信頼度Ｒは

ｅ＞ｍの場合、Ｒ＝１

ｅ≦ｍの場合、Ｒ＝Ｓ／（Ｓ＋ｍ＋１）

ⓕ　バグ曲線



最初は、テストの経過時間の割には、バグの発生件数は少ないが、その後、次第にバグの発生件数は多くなり、最後に、再び少なくなり、ほとんど飽和状態になる。

テストの初期からエラーが増えない場合、デバッグの中間状態であまりにもバグの発生が少ないとテストの方法が悪いと判断し、テスト方法を変更する。テスト項目が終わりに近づいているのにエラーが減らない場合はプログラムの品質が悪いと評価しテストを中断する。

③　開発工程におけるデザインレビュー

ⓐ　デザイン・レビューとは

デザインレビューは、ソフトウェア開発工程において、成果物の品質や開発進捗状況を客観的な知識のもとに評価し、不具合が発見された場合には、改善を提案し、早期問題の発見と次工程へ問題を持ち込まないようにするための活動である。基本計画や外部設計、内部設計など、システム開発の上流工程では、徹底したレビューと標準化を実施する。

ⓑ　デザインレビューの目的

進捗の把握は、開発工程の適切な時期に実施し、出来具合を明らかにすることである。品質評価は、ソフトウェアに含まれる欠陥や障害を発見し、品質状態を評価することである。対策の策定は、評価結果に対して今後の対応策を練ることである。

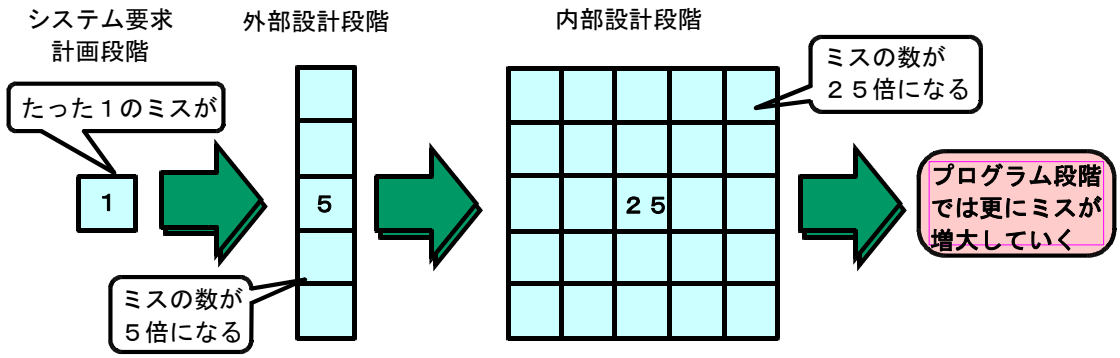
ⓒ　デザインレビューの作業内容

㋐　誤り、不良を早期発見し、後戻り工数の削減を図る。
㋑　当該フェーズの完了度合やプロジェクト進捗状況を把握し、当該工程の完了を確認する。
㋒　後続工程の計画を明確にする。

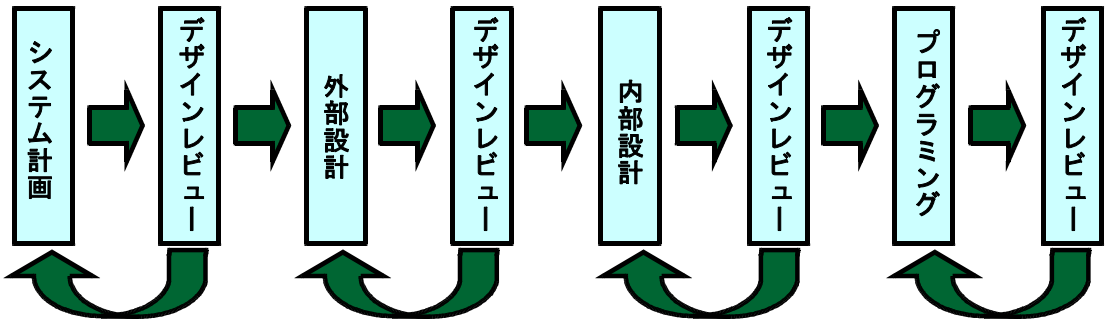
㊓　品質状況の評価結果を分析し、今後の改善施策へフィードバックする。

ⓓ　ミスの後工程への影響度

システム要求仕様における１ミスは、外部仕様段階には５となり、内部仕様段階では２５となり、プログラム段階では更に増大する。



ⓔ　開発工程とレビューの内容



㋐　システム計画

システムの目的、作業範囲、開発コスト、予算計画、リスク管理体制、システムの将来、実行可能性などについて、その妥当性と必要性についてレビューする。

㋑　外部設計

要求仕様に対する機能充足度についてレビューする。

㋒　内部設計

外部仕様に対して、プログラムの構造、データベース、ヒューマンインタフェースなどが適切に定義されているか、インタフェースが正しく設定されているかレビューする。

㋓　プログラミング

内部仕様に対する処理アルゴリズムの適切さ、モジュールの独立性、可視性、保守性などについてレビューする。

㋔　テスト

テスト結果の妥当性、システム稼働時の運用の妥当性、容易性、トラブル時の運用方法などについてレビューする。

ⓕ　デザインレビューの方式

㋐　ラウンドロビン方式

ラウンドロビン方式は、レビューに参画したメンバが持ち回りでレビュー責任者を務めながら、全体としてレビューを遂行していく方法である。

㋑　バスアラウンド方式

バスアラウンド方式は、レビュー対象となる成果物を電子メールなどで複数のレビューアに配布・回覧し、フィードバックを求める方法である。回覧式、配布式、集中式（掲示板式）などがある。

㋒　インスペクション方式

インスペクション方式は、レビュー対象の正しさをチェックする手法である。目的を明確に決めて資料を事前に準備し、レビュー責任者をおき、一堂に会してレビューを行う手法である。

㋓　ウォークスルー方式

ウォークスルー方式は、レビュー対象の手続きに対していくつかのテストケースを用意し、各テスト毎、その手続きを机上に追いかけて、シミュレーションし、妥当性を確認する方法である。手続きの使用条件、利用環境、手続き自身の機能や果たす役割についてもレビューする。

④　ウォークスルーとインスペクション

ⓐ　ウォークスルーおよびインスペクションとは

ウォークスルーやインスペクションは、各工程のドキュメントや成果物について担当者を含めた開発メンバーで検査することである。その目的はエラーの早期発見である。エラーを早期に発見し、システムの品質向上や開発の生産性向上を期待する。エラーの検討・改善の対策はこの場では実施しないで開発担当者が行う。

ⓑ　ウォークスルー・インスペクションの期待効果

㋐　再作業の範囲と量が少なくなる。

㋑　早期検出による潜在的なエラーの減少
㋒　ソフトウェアの品質向上
㋓　ソフトウェア開発の生産性の向上
㋔　技術者の育成

ⓒ　ウォークスルー・インスペクションの特徴

項　　目	特　　徴
㋐　構成人数	４～６人程度の少人数で行う。
㋑　資料の配付方法	事前に参加者に配布し、個人レベルで十分事前検討する。
㋒　開催時間の目安	１～２時間程度とする。２時間を超える場合は日を改める １人の人が１日に参加する回数は１日２回程度とする。
㋓　開催の目的	エラーの検出が目的である。解決策は別途検討する。
㋔　会議記録の方法	検出されたエラーは必ず文書化する。
㋕　責任の所在	参加者全員の同意を前提とする。 会議で作成された文書は、参加者全員が連帯責任を負う。
㋖　参加者の決定権	ウォークスルーの場合は開発担当者が決定する。 インスペクションの場合はモデレータが決定する。
㋗　参加資格者	検討対象について知っている人が参加する。 管理者かどうかは関係ない
㋘　管理者の出欠	管理者は極力参加しない。エラーが多いとき、その作成者あるいは開発者の人物評価につながるという危惧がある。

ⓓ　インスペクションの５つの役割

㋐　モデレータ
司会としてインスペクション全体を運営する。

㋑　オーナー
レビュー対象となる成果物の作成者で、発見された問題に応じて成果物の修正を行う。

㋒　インスペクタ
評価者としてレビュー対象となる成果物の問題発見を行う。

㋓　プレゼンタ
ミーティングにて参加者に資料の説明を行う。

㋔　スクライブ
書記としてレビューで発見された問題などを記録する。

ⓔ　ピアコードレビュー

ソースコードに対して、作成者の同僚が実施するレビューである。コードインスペクション、ピュアレビューとも呼ぶ。コーディング作業が終了した段階のコードに対して、次の内容を確認する。

㋐　詳細設計書の内容が正確に反映されているか。
㋑　コーディング規約を守っているか。

開発計画時にピアレビューの計画を行い、開発途中で成果物一式を同僚に提示しレビューを実施する。成果物の生産性と品質向上のために実施する。ピアレビューはＣＭＭＩの成熟度レベル３での重要なプロセスエリアになっている。

ⓕ　共同レビュープロセスの技術レビュー

共同レビューはユーザを含む関係者が共同で実施するレビューであり、レビューの結果と対応方法は文書化する。技術レビューは、検討中のソフトウェア製品またはサービスを評価し、次の事項を明らかにする。

㋐　ソフトウェア製品またはサービスが完全であり、標準および仕様に従っている。
㋑　ソフトウェア製品またはサービスに対する変更が、適切に実施され、構成管理プロセスで識別される部分にしか影響しない。
㋒　ソフトウェア製品またはサービスが、適切な予定に沿っている。
㋓　次の計画された活動の準備ができている。
㋔　企画、要件定義、開発運用、または保守は、プロジェクトの計画、予定、標準、および指針に従って実行されている。

⑤　ＣＡＳＥの機能と種類

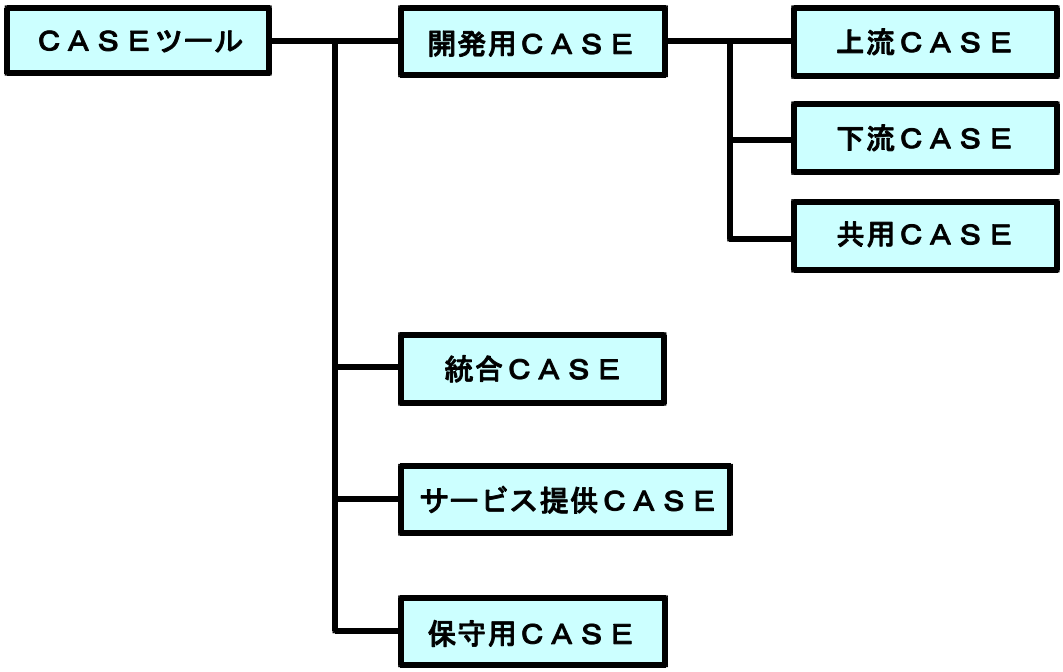
ⓐ　ＣＡＳＥ

ＣＡＳＥはソフトウェア開発や保守を自動化する助けとなるソフトウェアツールである。要求仕様や設計情報など開発に必要な情報をＤＦＤなどのダイヤグラムで表現する。各種情報は共通のリポジトリという開発設計情報データベースに蓄積し、一元管理し、設計情報の一貫性、完全性のチェックを図る。保守面でも、システムの理解、変更に対する影響の分析、修正に反映させるために利用することが可能になる。

ⓑ　ＣＡＳＥツールの種類

㋐　開発段階の前半を支援する上流ＣＡＳＥツール、後半を支援する下流ＣＡＳＥツール

㋑　保守工程を支援する保守ＣＡＳＥツール
㋒　開発の全工程を支援する共通ＣＡＳＥツール、統合ＣＡＳＥツール
㋓　開発プラットフォームサービス提供のＣＡＳＥツール



ⓒ　ＣＡＳＥが提供する機能

㋐　組み合わされたツールでライフサイクルすべてをカバー
㋑　各工程を支援するツール間の円滑なインタフェースを提供
㋒　全工程で参照可能なリポジトリの提供
㋓　構成管理や変更管理を含むライブラリの提供
㋔　すべてのツールが、一貫したユーザインタフェースを提供
㋕　工程管理や品質管理を含むプロジェクト管理技法の提供
㋖　新しい手法やツールを取り込む拡張性の提供

⑥　開発用ＣＡＳＥ

ⓐ　上流ＣＡＳＥ

上流ＣＡＳＥは適用業務の要求定義や分析、設計など、開発の上流工程における作業を支援するものである。

ⓑ　上流ＣＡＳＥが提供する機能

㋐　システムの機能の分析と定義

㋑　システムの機能を構成する要素の洗い出し
㋒　システムの構成要素間の関連性の分析
㋓　構成要素の処理設計および構成要素間の処理の流れの設計
㋔　ファイルとデータベースの設計
㋕　ネットワークの設計
㋖　プロトタイピング機能
㋗　上記の機能を実現するためのダイアグラム作成機能やデータ項目の完全性や一貫性のチェック機能

ⓒ　下流ＣＡＳＥ

下流ＣＡＳＥはプログラミング工程およびテスト工程における作業を支援する機能をもつ。

ⓓ　下流ＣＡＳＥが提供する機能

㋐　プログラミング支援機能
㋑　コード自動生成機能
㋒　テスト支援機能
㋓　画面設計支援および帳票作成支援機能
㋔　ファイルやデータベースの設計支援およびＤＤＬの自動生成機能

ⓔ　共用ＣＡＳＥ

共用ＣＡＳＥは開発工程全体を通して行われる作業を支援する機能を持つ。統合ＣＡＳＥツールとは区別されるものである。

ⓕ　共用ＣＡＳＥが提供する機能

㋐　文書化支援機能
文書作成、作表、作図
㋑　プロジェクト管理機能
見積もり、日程作成、進行管理、品質管理
㋒　システム構成管理支援機能

ⓖ　統合ＣＡＳＥ

統合ＣＡＳＥはシステムのライフサイクルの全工程をカバーするものである。システムの保守工数の増大に対処し、開発工数低減のため、システムのライフサイクル全体の生産性向上のために、設計段階からソフトウェアの品質を向上させる必要が認識されるようになった。システム開発の工程間のインタフェースを整備し、システム開発の全工程を支援するツールである。既存の開発支援ツールを有効利用する支援ツールの役割や各ツール間の設計情報を、自由にやり取りするためのインタフェースを提供する。

ⓗ　開発プラットフォームサービス提供ＣＡＳＥ

開発プラットフォームサービス提供ＣＡＳＥはツールの統合化を進めるときに使用される。既存のＣＡＳＥツールを統合化し、システムのライフサイクルのすべてを支援する場合、既存のＣＡＳＥツール間のインタフェースを定義するものが必要になる。

ⓘ　開発プラットフォームサービス提供ＣＡＳＥが提供する機能

㋐　開発資源情報管理機能

データディクショナリ、リポジトリ管理

㋑　ライブラリ管理

設計書、仕様書、原始プログラム、目的プログラム、テストケースなどの保管

㋒　ツール間共通インタフェースの提供

㋓　ツールの基本となるサービスや統合のための各種サービスの提供

⑦　保守用ＣＡＳＥ

ⓐ　保守用ＣＡＳＥとは

保守用ＣＡＳＥは保守工程における作業を支援する機能を持つ。既存のソフトウェアには、設計情報の一元管理のもとで開発されたものが少なく、システム機能の変更に対して、影響範囲が明確でないものが多い。また、多くのプログラムが構造化されていないため、解析に時間がかかる問題がある。

これらの既存のプログラムに対して、リバースエンジニアリング機能やリエンジニアリング機能を活用し、システム開発の生産性の向上と費用の削減を図ったり、既存のソフトウェアの再利用の促進のために活用する。

ⓑ　提供する機能

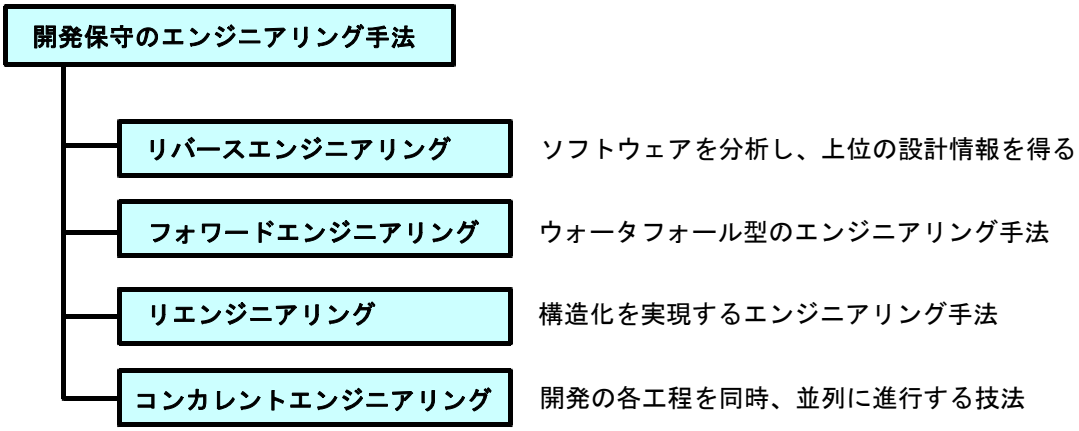
㋐　原始プログラムの解析、修正および再構造化機能

モジュール構造図、プログラムフローの作成、非構造化プログラムの構造化

㋑　データベースの解析と再設計機能

⑧　開発・保守のためのエンジニアリング手法

ⓐ　エンジニアリング手法の分類



ⓑ　リバースエンジニアリング

リバースエンジニアリングは設計方針→開発作業→製品の通常の工程の逆の手順をたどる開発技法であり、次のような機能や特徴がある。

㋐　原始プログラムから、上位の設計情報を自動生成する。
㋑　既存ソフトウェアを分析し、基本的な設計方針、システムの仕様を導き出す。
㋒　他社の製品を解体して構造や性能、技術内容を調べ、対抗製品や互換製品を作成する
㋓　オブジェクトプログラムを逆コンパイルし、プログラムの内容を把握し、開発の参考にしたり、実装済みのソフトウェアから設計仕様を抽出して、ソフトウェアの再開発に利用したりする。
㋔　リバースエンジニアリングは違法行為ではないが、その結果に基づいて許可なく、開発・販売した製品は元の製品の著作権侵害になる可能性がある。

ⓒ　互換性

ハードウェアやソフトウェアを交換しても、もとの環境と同様に動作する性質である。最近のパソコン関連のアプリケーションやＯＳ、ハードウェアなどの製品は、過去の資産が活用できるように互換性をもたせ、機能の向上を図っている。

次のような場合に互換性があると言える。

㋐　アプリケーションソフトウェアが異なるコンピュータで動作する。
㋑　あるフォーマットのファイルが異なるソフトウェアで扱える。
㋒　あるインタフェースに異なる周辺機器を接続しても使用できる。

ⓓ　コールグラフ（マルチグラフ）

コールグラフはコンピュータプログラムのサブルーチン同士の呼び出し関係を表現した有向グラフである。各ノードが手続きを表現し、各エッジ(f, g)は手続きfが手続きgを呼び出すことを示す。従って、循環したグラフは再帰的な関数呼び出しを示す。

コールグラフはプログラムを人間が可読なものにするため、手続き間の変数の追跡を行う解析といった発展的な解析のための基礎として用いることができる。コールグラフはプログラムの実行結果を記録するため、リバースエンジニアリングのツールとして使用する。

ⓔ　リエンジニアリング

ソフトウェアを保守する際に、作業の効率や品質を高めたり、厳密に管理するために使用する技法であり、モジュール構造図の作成や構造化されていないプログラムの機能を変えないで、構造化を図ったりすることである。リエンジニアリングはソフトウェアの再構造化の概念であり、再構造化の考え方はデータや設計、要求仕様などにも応用される。

ⓕ　再利用と部品化

再利用は、ソフトウェアの開発で開発の生産性や品質を高めることを目的に、過去に開発した設計デザインやプログラムを使用することである。再利用するためには、システム設計の段階で、機能の構造化や部品化を考えることが重要である。

部品は、ソフトウェアの再利用を目的として共通化したデータやアルゴリズムである。部品化したプログラムの再利用で、開発の生産性や品質を高めることができる。良質の部品を作るためには、ソフトウェア機能を構造化分析し部品として独立度の高い機能を取り出すことである。そのためには、部品の仕様が明確であること、動作性能が高いことなどが求められる。

再利用や部品化に関係するものとして次のものがある。

㋐　入出力機能を標準化し、部品化しておく。
㋑　ラジオボタンやポップアップメニューの活用
㋒　オブジェクト指向のカプセル化や継承の考え方

ⓖ　フォワードエンジニアリング

システムの仕様からソフトウェアを作り出すことであり、上流工程での設計文書からプログラムを生成できる。ウォータフォール型の開発技法である。

ⓗ　コンカレントエンジニアリング

製品の開発過程において、企画、設計、生産、販売、サービスなどの各工程を同時に、並列に進行する技法で、開発期間の短縮、開発コストの削減が期待できる。前工程の作業が完了する前に、後工程で問題点が表面化し、それを反映して設計変更が可能になったり、前工程の進行状況や決定事項を後工程の作業者が把握できるため、前もって適切な対応が可能になるなどの利点がある。ＣＡＤ／ＣＡＭ／ＣＡＥなどのコンピュータツールによるデータの一元化、ネットワークによる情報の迅速・正確な伝達が必要である。

例題演習

ソフトウェアの品質特性に関する記述のうち、適切なものはどれか。

ア　移植性とは、ソフトウェアの不良原因を容易に解析でき、また修正できることをいう。
イ　効率性とは、ユーザが要求する特定の目的にソフトウェアの仕様が合致していることをいう。
ウ　信頼性とは、与えられた条件で規定の期間中、要求された機能を果たすことをいう。
エ　保守性とは、使用環境、使用条件の変更なしにソフトウェアを置き換えても同じ機能が引き続き使えることをいう。

解答解説

ソフトウェアの品質特性に関する問題である。

アの移植性は、あるコンピュータで動作しているプログラムを他のコンピュータで動作させるためにどの程度容易に移し換えることができるかを表す特性であり、アの記述内容は更新容易性を表している。

イの効率性は、ソフトウェア製品がコンピュータ資源をどの程度無駄なく使用しているかを表す特性要因であり、イの記述の内容は使用性を表している。

ウの信頼性は、ソフトウェアが仕様通りに動作するかどうかの特性で、規定期間中、要求された機能を果たすことになる。正しい。求める答えはウとなる。

エの保守性は、ユーザからのクレームや要求への対応のしやすさを表す特性であり、エの記述内容は移植性を表している。

例題演習

ソフトウェアの品質特性の定義において，あるコンピュータ用に作成したプログラムを別のアーキテクチャのコンピュータで動作できるようにすることの容易さを表す特性はどれか。

ア　移植性(Portability)　　イ　使用性(Usability)
ウ　相互運用性(Interoperability)　　エ　変更性(Changeability)

解答解説

ソフトウェアの品質特性に関する問題である。

アの移植性は、あるコンピュータで動作しているプログラムやあるコンピュータ用に作成したプログラムを、他のコンピュータで動作させるために、どの程度容易に移し換えることができるかを表す特性である。求める答えはアとなる。

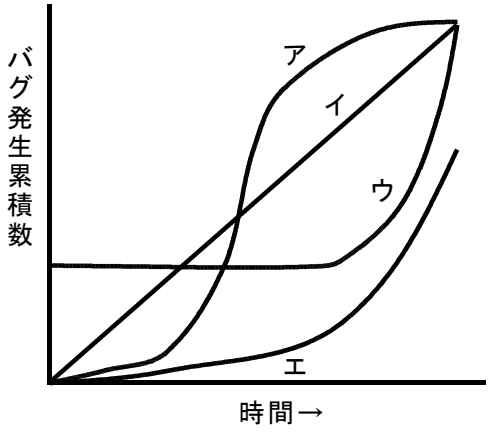
イの使用性は、仕様の充足度を示す特性で、ソフトウェアの目的とした機能が、仕様通りに表現されているかどうかを表す特性要因である。

ウの相互運用性は、複数の機器を接続してシステムを構築したときに，トラブルなくシステム運用できるかどうかを表すことである。

エの変更性は更新容易性で、ソフトウェアや文書が変更しやすい特性をもっているかどうかを表すものである。

例題演習

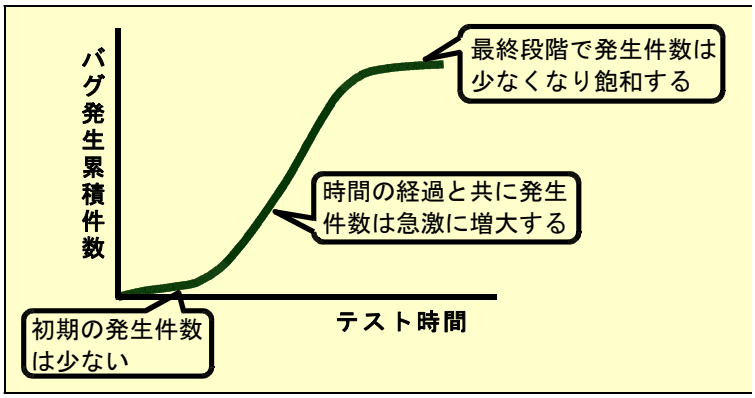
ソフトウェアの開発過程における、テストの進捗とバグ発生累積数の関係を示すグラフとして、最も一般的なものはどれか。ただし、横軸はテストの時間の経過、縦軸はバグ発生累積数である。テスト項目は適切かつ十分に設計されているものとする。



解答解説

テストの進捗とバグ発生累積数のグラフに関する問題である。

ソフトウェアのバグ発生数は、ロジスティック曲線、ゴンベルツ曲線などの成長曲線で近似できる。バグの発生状況は、最初はコンピュータの使用時間の経過する割には発生件数は増加しないが、その後、時間の経過と共に次第に発生件数が多くなり、最終段階では再び少なくなり飽和状態になる。この状態を表している曲線はアである。求める答えはアとなる。



例題演習

バグ理込み法によってソフトウェア内に残存するバグを推定する。テストによって現在までに発見されたバグは４８個であり，総埋込みバグ２２個のうち，テストによって発見されたものは１６個であった。あと幾つのバグが潜在していると推定されるか。ここで，埋込みバグの発見数とソフトウェアのバグの発見数は比例するものとする。

ア　６　　イ　１０　　ウ　１８　　エ　２２

解答解説

バグ埋め込み法による潜在バグ推定の問題である。

総埋め込みバグ２２個の内、発見されたバグの個数は１６個である。

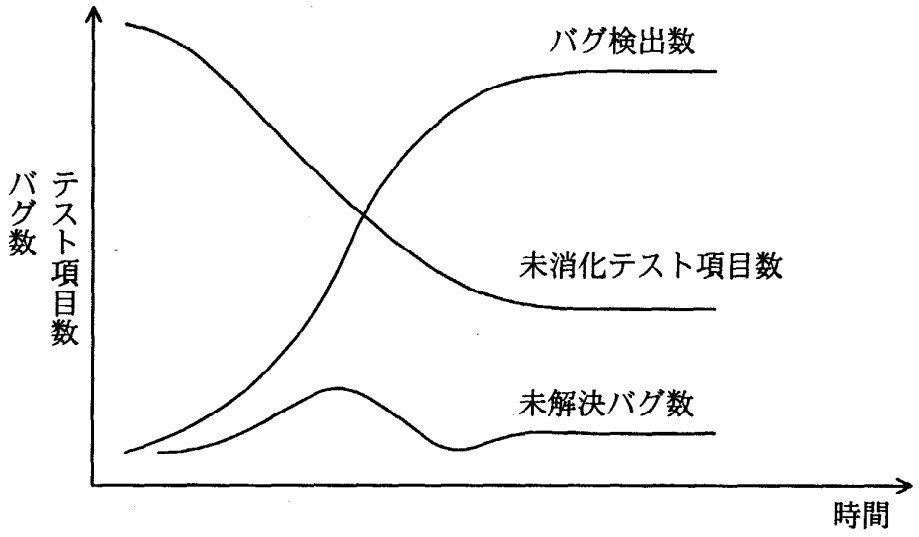
バグの総数は２２×（４８÷１８）＝６６

現在発見されたバグ数は４８であるから、残りのバグ数は６６－４８＝１８

求める答えはウとなる。

例題演習

バグ管理図において，図のようにすべての線が横ばい状態になった。この状況から推測できることとして，適切なものはどれか。



ア　解決困難なバグに直面しており，その後のテストが進んでいない。

イ　テスト項目の消化実績が上がっており，バグの発生がな．くなった。

ウ　バグが多発し，テスト項目の消化実績が上がらなくなった。

エ　バグ発生とテスト項目消化の比率が一致し，未解決バグがなくなった。

解答解説

バグ管理図に関する問題である。

問題のバグ管理図は時間の経過と共に次のような特徴的な現象を示している。

①　バグ摘出累計が変化していない。

②　未消化テスト項目数が減少しなくなった。

③　未解決バグ数も変化しない。

アの解決困難なバグに直面すると、①、②、③の現象が現れる。求める答えはアとなる。

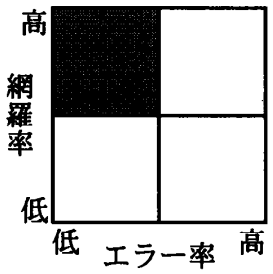
イのテスト項目の消化実績が上がっているは、②の現象が一致しない。

ウのバグが多発は、①のバグ摘出累計が変化しないが一致しない。

エの未解決バグがなくなったは③が一致しない。

例題演習

網羅率とエラー率の組合せによって，プログラム品質を評価した。図の網掛け部に位置付けられるプログラムの評価として，最も適切な記述はどれか。



$$網羅率 = \frac{プログラムのテストで実行したステップ数}{プログラムのステップ数}$$

$$エラー率 = \frac{エラー件数}{プログラムのステップ数}$$

ア　一般には品質が良いと判断されるが，例外処理がテスト項目に含まれているかどうかの確認が必要である。

イ　エラーの収束状況が分からないので，この評価方法ではプログラム品質について判断できない。

ウ　エラーの発見率が少なすぎるので，テスト方法に問題があると判断すべきである。

エ　網羅率が高いので，テストは十分である。これ以上，テストを続ける必要はない。

解答解説

網羅率とエラー率との関係から品質評価の適切さを判定する問題である。

アの内容は、網羅率が高くエラーの発生が少ないのであるから品質良好であるが、例外処理について不明確であるということになる。求める答えはアとなる。

イの内容は、網羅率が高くエラーが少ないのであるから、品質について判断できないというのは正しくない。

ウの内容は、網羅率が高くエラーが少ないからテスト方法に問題があると断定することはできない。これだけの内容からは不明確である。

エの網羅率が高いということとテストが十分ということは必ずしも一致しない。

例題演習

ウォークスルーの進め方の説明として，適切なものはどれか。

ア　主に解決策の検討を行う。

イ　開発管理者主導で開発者は会議に参加しない。

ウ　対象となる資料はウォークスルー用の要約版を使用する。

エ　問題点の検出に専念する。

解答解説

ウォークスルーに関する問題である。

アのウォークスルーの目的はエラー検出であって、解決策の検討は別途行う。

イの会議への参加者は開発担当者が参加し、開発管理者は参加しない。

ウの会議で使用する資料は詳細のものを事前に配布し、慎重に検討し事前準備する。要約版では目的を達成することができない。

エの問題点の検出に専念するは、ウォークスルーの進め方として適切な内容である。求める答えはエとなる。

例題演習

設計資料の品質を確保するために、開発の各段階においてレビューを行う。このときに行うレビュー技法の一つであるインスペクションの説明として、適切なものはどれか。

ア　参加者が持ち回りで責任を務めながら、全体のレビューを遂行する。
イ　対象ソフトウェアの一部を試作し、実際に動作させてレビューする。
ウ　レビュー実施の焦点を絞っておき、一度に１項目を確認することによって、迅速に資料を評価する。
エ　レビュー対象の設計資料の作成者がレビューを主催する。

解答解説

デザインレビューのインスベクション方式に関する問題である。

インスベクション方式は、レビュー対象の正しさをチェックする手法である。目的を明確に決めて資料を事前に準備し、レビュー責任者をおき、一堂に会してレビューを行う手法である。

アはラウンドロビン方式、イはプロトタイピング方式、ウがインスペクション方式、エがウォークスルー方式である。求める答えはウとなる。

例題演習

デザインレビューの一つの技法で、モデレータの進行のもとで、関係者が集まって、会議形式で検証を行うのはどれか。

ア　インスペクション　　イ　レビュー
ウ　ウォークスルー　　エ　システム監査

解答解説

インスベクションに関する問題である。

アのインスペクションは、システム開発の各フェーズで、ドキュメントなどを第三者が検査し、欠陥や問題点を洗い出すことで、モデレータが主催する。求める答えはアである。

イのレビューは、システム開発の過程で各フェーズごとにドキュメントやプログラムなどの成果物を確認検証することである。インスペクションとウォークスルーの方法がある。

ウのウォークスルーは、システム開発の各フェーズで、ドキュメントなどを開発メンバが討議して、欠陥や問題点を洗い出すレビューで、開発担当者同士で設定し実施する。モデレータは関係しない。

エのシステム監査は、コンピュータシステムの安全性や信頼性、経済性を総合的に評価し、助言、勧告、改善活動のフォローアップを行うことである。

例題演習

ソフトウェアのレビュー方法の説明のうち，インスペクションはどれか。

ア　作成者を含めた複数人の関係者が参加して会議形式で行う。レビュー対象となる成果物を作成者が説明し，参加者が質問やコメントをする。
イ　参加者が順番に司会者とレビュアになる。司会者の進行によって，レビュア全員が順番にコメントをし，全員が発言したら，司会者を交代して次のテーマに移る。
ウ　モデレータが全体のコーディネートを行い，参加者が明確な役割をもってチェツクリストなどに基づいたコメントをし，正式な記録を残す。
エ　レビュー対象となる成果物を複数のレビュアに配布又は回覧して，レビュアがコメントをする。

解答解説

デザインレビューのインスベクション方式に関する問題である。

インスベクション方式はレビュー対象の正しさをチェックする手法である。目的を明確に決めて資料を事前に準備し、レビュー責任者をおき、一堂に会してレビューを行う手法である。

アはウォークスルー方式、イはラウンドロビン方式、ウがインスペクション方式、エがバスアラウンド方式である。求める答えはウとなる。

例題演習

上流ＣＡＳＥツールに分類されるものはどれか。

ア　システム設計支援ツール　　イ　テストデータ生成ツール
ウ　プログラム自動生成ツール　　エ　プロジェクト管理ツール

解答解説

上流ＣＡＳＥツールに関する問題である。

アのシステム設計支援ツールは上流ＣＡＳＥ、イのテストデータ生成ツールは下流ＣＡＳＥ、ウのプログラム自動生成ツールは下流ＣＡＳＥ、エのプロジェクト管理ツールは共用ＣＡＳＥである。求める答えはアとなる。

例題演習

ＣＡＳＥツールは適用する開発工程や範囲によって分類できる。プログラム自動生成機能はどの分類に含まれるか。

ア　開発プラットフォーム　　イ　下流　　ウ　上流　　エ　保守

解答解説

ＣＡＳＥツールの分類に関する問題である。

アの開発プラットフォームＣＡＳＥは、開発資源情報管理、ライブラリ管理、ツール間共通

インタフェースの提供がある。開発の全工程に関係する。

イの下流ＣＡＳＥは、コード自動生成、プログラミング支援機能、テスト支援機能、画面設計・帳票作成の支援機能である。求める答えはイである。

ウの上流ＣＡＳＥは、システムの機能分析や定義、システムの構成要素間の関連性の分析、ネットワークの設計、データベースの設計等が含まれる。

エの保守ＣＡＳＥは、リバースエンジニアリング機能、リエンジニアリング機能、プログラムの解析・構造化、データベースの解析・再設計機能が含まれる。

例題演習

ソフトウェアに関するリバースエンジニアリングの説明として、最も適切なものはどれか。

ア　実装されたソフトウェアから設計仕様を抽出して、ソフトウェア開発に利用する。
イ　出力、処理、入力という順にソフトウェアの設計を行う。
ウ　ソフトウェアとして実現されていた機能をハードウェアで実現する。
エ　ソフトウェアの処理の内容に応じて、開発言語や開発ツールを選択する。

解答解説

リバースエンジニアリングに関する問題である。

アはリバースエンジニアリング、イはフォワードエンジニアリング、ウはソフトウェアのファームウェア化の考え方であり、エはソフトウェアの開発環境の選択の考え方である。求める答えはアとなる。

例題演習

リバースエンジニアリングに関する記述として、適切なものはどれか。

ア　既存の業務を分析し、業務の再構築によって業務改革を行うことである。
イ　実装済みのソフトウェアから設計仕様を抽出して、ソフトウェアの修正及び再開発に利用することができる。
ウ　ソースプログラムを構造化プログラムの形態に変換することができる。
エ　データ名とテータ定義をシステム全体で標準化することができる。

解答解説

リバースエンジニアリングに関する問題である。

リバースエンジニアリングは既存のソフトウェアからシステムの仕様を導き出すことである。オブジェクトプログラムを逆コンパイルし、プログラムの内容を把握し、開発の参考にしたり、実装済みのソフトウェアから設計仕様を抽出して、ソフトウェアの再開発に利用したりする。

アはリエンジニアリング、イはリバースエンジニアリング、ウはリストラクチャリング、エは標準化である。求める答えはイとなる。

例題演習

プログラムからＵＭＬのクラス図を生成することは何と呼ばれるか。

ア　バックトラッキング　　イ　フォワードエンジニアリング
ウ　リエンジニアリング　　エ　リバースエンジニアリング

解答解説

リバースエンジニアリングに関する問題である。

アのバックトラッキングは、相手の発した言葉をおうむがえしのように返す事を指します。

イのフォワードエンジニアリングは、システム仕様からソフトウェアを作り出すことである。

ウのリエンジニアリングは、既存のシステム資源を利用して、新しいシステムを再構築することで、定義の見直し、コードの変換などを行い、再利用や保守を容易にする。

エのリバースエンジニアリングは、既存のソフトウェアやハードウェアなどを分解又は解析し、その設計目的や仕様、構成部品、要素技術などを明らかにする技術で、ソフトウェアの保守や対抗製品・互換製品の開発に利用する。求める答えはエとなる。

例題演習

システム開発における品質管理に関する記述のうち，適切なものはどれか。

ア　幾つかのサブシステムに分割して開発するとき，サブシステム単位での品質が保証できれば，同時にシステム全体としての品質も保証できる。
イ　応答時間やバッチ処理時間などの性能は品質管理の対象外であるが，業務に与える影響が大きいので限界性能を計測しておく。
ウ　システムへの要求機能の充足度だけでなく，ドキュメントなどすべての成果物を含めて品質管理の対象とする。
エ　市販製品と自社開発プログラムを組み合わせてシステムを開発する場合，品質管理の対象は自社開発のプログラムだけとなる。

解答解説

システム開発における品質管理に関する問題である。

開発工程における品質保証は、作業成果物やプロセスが定義された条件、計画に従った開発であることを保証することである。製品の保証、プロセスの保証、品質システムの保証が含まれる。

アのサブシステム単位に品質が保たれても、サブシステム間のインタフェースを含めた全体システムの品質が保証されたことにはならない。

イの応答時間やバッチ処理性能も、非機能要件としてソフトウェアの品質評価の対象になる。

ウの品質管理の対象となる保証すべきものは、製品(成果物)、プロセス、品質システムであり、成果物としてドキュメントが含まれる。求める答えはウとなる。

エの市販製品、自社開発品を含めたトータルシステムが対象になる。